

J. A. Hoffer • M. B. Prescott
F. R. McFadden • V. Morabito

MANAGEMENT DELLE INFORMAZIONI AZIENDALI

PEARSON

Prentice
Hall

Jeffrey A. Hoffer
Mary B. Prescott
Fred R. McFadden

Management delle informazioni aziendali

Prima edizione italiana

Copyright © 2005 Pearson Education Italia S.r.l.
Via Fara, 28 - 20124 Milano
Tel. 02/6739761 Fax 02/673976503
E-mail: hpeitalia@pearson.com
Web: <http://hpe.pearsoned.it>

*Authorized translation from the English language edition, entitled: **MODERN DATABASE MANAGEMENT** published by Pearson Education, Inc, publishing as Prentice Hall, Copyright © 2002*

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc,

Italian language edition published by Pearson Education Italia Srl, Copyright © 2005

Le informazioni contenute in questo libro sono state verificate e documentate con la massima cura possibile. Nessuna responsabilità derivante dal loro utilizzo potrà venire imputata agli Autori, a Pearson Education Italia o a ogni persona e società coinvolta nella creazione, produzione e distribuzione di questo libro.

I diritti di riproduzione e di memorizzazione elettronica totale e parziale con qualsiasi mezzo, compresi i microfilm e le copie fotostatiche, sono riservati per tutti i paesi.

LA FOTOCOPIATURA DEI LIBRI È UN REATO. L'editore potrà concedere a pagamento l'autorizzazione a riprodurre una porzione non superiore a un decimo del presente volume. Le richieste di riproduzione vanno inoltrate ad AIDRO (Associazione Italiana per i Diritti di Riproduzione delle Opere dell'Ingegno), Via delle Erbe, 2 - 20121 Milano - Tel. e Fax 02/80.95.06.

Curatore per l'edizione italiana: Vincenzo Morabito
Realizzazione editoriale: Edimatica srl
Grafica di copertina: Vittoria Pasca Raymondi
Stampa: Bianca & Volta – Truccazzano (MI)

Tutti i marchi citati nel testo sono di proprietà dei loro detentori.

ISBN 88-7192-223-9

Printed in Italy

1ª edizione italiana: gennaio 2005

Parte prima

Evoluzione e rappresentazione dei sistemi informativi aziendali



CAPITOLO

1

L'evoluzione del ruolo dei sistemi informativi e dell'IT

Al termine di questo capitolo sarete in grado di fare quanto segue.

1. Definire e distinguere correttamente i concetti di sistema informativo e information technology.
2. Identificare le tre generazioni più importanti dell'evoluzione del ruolo dei sistemi informativi in azienda: dal data processing ai sistemi informativi strategici.
3. Descrivere la differenza e le relazioni tra le strategie di business, dei sistemi informativi e dell'information technology.
4. Definire il contesto e i fattori strategici di un sistema informativo.
5. Identificare e spiegare gli impatti organizzativi di un sistema informativo.
6. Spiegare l'importanza dell'allineamento della strategia dei sistemi informativi con la strategia di business.

Introduzione

Scopo di questo capitolo è quello di aiutare a comprendere il ruolo del sistema informativo e dell'information technology nelle aziende. Dopo aver chiarito la differenza tra il concetto di *sistema informativo* (SI) e quello di *information technology* (IT), vengono presentati i principali modelli che hanno caratterizzato le tre generazioni del processo evolutivo in azienda sia dei SI sia dell'IT: dal data processing (DP) ai sistemi informativi strategici (SIS).

Il percorso di tale evoluzione dimostra il crescente ruolo organizzativo, strategico e di business dei sistemi informativi. In particolare, si rileva ormai una stretta relazione tra la strategia dei sistemi informativi e dell'IT in generale e quella di business.

1.1 Perché studiare il ruolo organizzativo è strategico dei sistemi informativi in azienda

La maggior parte delle aziende di ogni settore (industriale, commerciale, governativo e così via), è oggi fortemente legata ai propri sistemi informativi. Come ha affermato Rockart (1998) "l'Information Technology (IT) si è inestricabilmente connessa con il business". Nei settori quali le telecomunicazioni, i media, l'entertainment e i servizi finanziari, dove il prodotto viene veicolato sempre più in maniera

digitale, la sopravvivenza dell'azienda dipende in modo cruciale dall'effettiva applicazione dell'Information Technology.

Le aziende stanno guardando con crescente interesse all'applicazione della tecnologia non solo come supporto alle attività aziendali esistenti, ma anche per creare nuove opportunità che possono essere fonte di vantaggio competitivo. Con l'avvento dell'e-commerce, per esempio, l'uso della tecnologia nelle transazioni operative si afferma e diffonde sempre di più.

Conseguentemente un approccio organizzativo e strategico ai sistemi informativi e all'information technology (SI/IT) si rivela significativo per comprendere l'evoluzione del ruolo dei sistemi informativi nelle aziende.

Molte aziende, infatti, vorrebbero adottare un approccio più organizzativo e strategico nella gestione dei SI/IT, il che in alcuni casi potrebbe richiedere la profonda revisione o il rifacimento ex novo dei loro investimenti in SI/IT i quali, nella maggior parte dei casi, derivano da scelte passate che difficilmente è possibile cambiare radicalmente. Per esempio, molte società del mondo bancario o assicurativo utilizzano ancora sistemi sviluppati più di trent'anni fa e, a breve termine, non si prevedono sostanziali "aggiornamenti" se non quelli legati a vincoli normativi (per esempio, Basilea II o IAS, per citarne alcuni).

Imparare dall'esperienza, cioè dai successi e dai fallimenti del passato, è uno dei più importanti aspetti di gestione strategica. Earl (1992) sostiene che le aziende tendono a imparare a gestire i loro SI/IT in modo esperienziale, cioè facendo, e che non si rendono conto realmente delle sfide finché non si trovano a doverle affrontare.

Tuttavia nessuna azienda può contare su una gamma completa di esperienze nei SI/IT, né è probabile che ciò che è già stato sperimentato sia sempre valutabile obiettivamente.

L'evoluzione dei SI/IT nelle aziende è stata analizzata sotto da numerosi punti di vista, utilizzando diversi modelli, alcuni dei quali sono stati ulteriormente sviluppati prendendo in considerazione gli aspetti della pianificazione strategica degli investimenti in SI/IT.

Numerosi fattori influenzano la frequenza e l'efficacia dei progressi nell'utilizzo dei SI/IT per incrementare il "valore" di business delle aziende. Tali fattori includono:

1. le capacità "abilitanti" dell'IT/SI;
2. le economie derivanti dall'implementazione dell'IT/SI;
3. il portafoglio applicativo disponibile sul mercato;
4. le competenze, interne o da fonti esterne, disponibili per sviluppare/personalizzare le applicazioni;
5. le competenze e la propensione, interne all'azienda, nell'uso delle applicazioni;
6. le pressioni aziendali o di settore dirette al miglioramento delle performance.

La maggior parte degli studi sull'evoluzione dei SI/IT nelle aziende tende a focalizzarsi su uno o due aspetti del loro sviluppo, relativo alla struttura organizzativa, alle applicazioni, alla gestione della tecnologia, alla progettazione e così via.

1.2 Aspetti terminologici: sistemi informativi e/o information technology

Prima di formulare un qualunque approccio organizzativo o strategico, è importante comprendere bene la distinzione tra i sistemi informativi (SI) e l'information technology (IT), spesso utilizzati indifferentemente ma la cui differenza di significato è importante almeno per due motivi: anzitutto per "facilitare"

e rendere coerente il dialogo tra la funzione sistemi informativi e il resto dell'azienda; in secondo luogo, per sviluppare strategie di successo sia in termini di SI sia di IT. Vale la pena, a questo proposito, ricordare che i sistemi informativi sono presenti in azienda da molto tempo, ancor prima dell'avvento dell'information technology e, addirittura oggi, esistono molti sistemi informativi non intaccati dalla tecnologia.

Con il termine *information technology (IT)* si fa riferimento essenzialmente all'hardware, al software e alle reti di telecomunicazioni. È costituita sia da elementi tangibili (per esempio server, PC, reti e cavi) sia intangibili (software di ogni genere). L'IT facilita l'acquisizione, l'elaborazione, l'archiviazione, la distribuzione e la condivisione delle informazioni e di altri contenuti digitali. Nell'Unione Europea, il termine *Information e Communication Technology* o *ICT* è utilizzato al posto di *IT* per riconoscere la convergenza tra l'information technology e le telecomunicazioni, che erano viste una volta come aree distinte.

L'Accademia del Regno Unito dei Sistemi Informativi (UKAIS) definisce i sistemi informativi come *"the means by which people and organizations, utilizing technology, gather, process, store, use and disseminate information"*. Secondo questa definizione i SI sono quindi basati sull'utilizzo dell'information technology (il tema sarà approfondito nel Capitolo 2).

In effetti in alcune aziende, come per esempio la Dell Computer, esiste un sistema informativo in cui nessun intervento umano è richiesto: dalla raccolta degli ordini del cliente, alla consegna dei componenti alla fabbrica Dell per l'assemblaggio, alla spedizione ai clienti finali, tutto è realizzato tramite l'uso di strumenti tecnologici.

L'ambito di studio dei SI, come definito dall'UKAIS, coinvolge dunque lo studio delle ragioni e circostanze che determinano lo sviluppo, l'uso e gli effetti dei sistemi informativi nelle aziende.

Mingers (1995), diversamente, sostiene che, nonostante l'IT sia un fattore un abilitante per i SI, *"IS actually is part of the much wider domain of human language and communication, that IS will remain in a state of continual development and change in response both to technological innovation and to its mutual interaction with human society as a whole"*. In questa prospettiva, oltre a quanto detto, diventa rilevante il tema dei motivi di adozione e selezione dell'IT.

Chiarita la differenza tra IT e SI, è necessario chiarire il significato del termine *applicazione*. Un'applicazione, o software applicativo, utilizza l'IT per automatizzare un'attività aziendale o un processo. Le applicazioni possono essere classificate in due tipologie:

1. di office automation, quelle cioè che fanno un uso generico dell'IT per eseguire attività come l'elaborazione di testi, la posta elettronica o le presentazioni;
2. gestionali, quelle cioè che utilizzano la tecnologia per svolgere specifiche attività o processi, come la contabilità generale, la produzione o la gestione degli ordini.

Tali applicazioni possono essere costituite da pacchetti software oppure essere sviluppate ad hoc per offrire particolari funzionalità. Alcuni pacchetti software possono essere "confezionati" su misura o possono essere personalizzati per specifiche esigenze dell'azienda.

Checkland e Holwell (1998) sostengono che molte persone hanno difficoltà nel distinguere fra SI e IT, poiché la tecnologia sembra "sommersgere" il sistema informativo mentre, in realtà, è di supporto al SI. Checkland (1981) sostiene inoltre che per creare un SI che supporti efficacemente gli utenti è necessario prima concettualizzare ciò che dovrà essere supportato (il SI) e successivamente il modo in cui ciò verrà supportato (l'IT).

Questo fornisce delle indicazioni sul perché molte aziende non si rendono conto di alcuni vantaggi dei loro investimenti in IT; spesso gli investimenti in tecnologia sono fatti senza comprendere o analizzare la natura delle attività che la tecnologia deve strategicamente o operativamente supportare in azienda.

Per esempio, nel corso degli ultimi anni, molte aziende hanno costruito siti Web senza una razionale e sufficiente analisi a supporto della decisione, ma solo perché tutti utilizzano la “Rete”.

È importante sottolineare che l'acquisto di tecnologia, fine a se stesso, non garantisce necessariamente all'azienda immediati vantaggi, i quali invece devono essere individuati attraverso un'analisi organizzativa e strategica.

1.3 L'information technology e i sistemi informativi in azienda: anni '50-anni '80

L'uso dei computer in azienda risale ai primi anni '50, ma questi si sono utilizzati in modo più intensivo solo nella seconda metà degli anni '60.

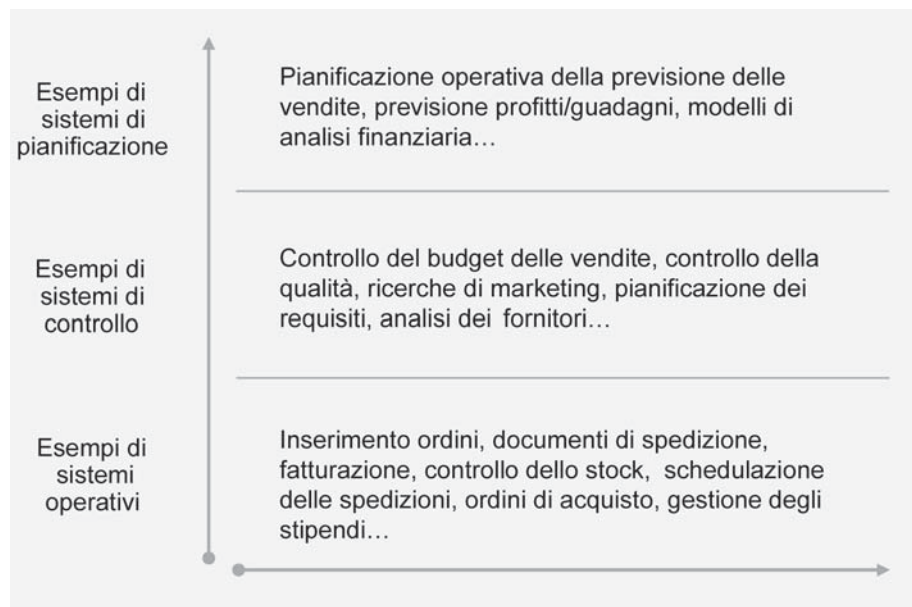
I modelli sviluppati sul ruolo dei sistemi informativi e sulla loro evoluzione attesa erano fortemente fondati sui concetti di centralizzazione e di integrazione (derivanti dai sistemi mainframe). Il più noto di questi modelli è quello sviluppato da Gibson e Nolan (1974) durante gli anni '70 ed è basato sul modello di portafoglio applicativo gerarchico descritto da Anthony (1965), il quale ha definito una struttura per i sistemi informativi in azienda stratificando l'attività di gestione in:

- pianificazione strategica;
- controllo di gestione;
- controllo operativo.

In funzione delle differenti applicazioni a supporto dei vari livelli di management, il modello di Anthony rappresenta un primo metodo per la loro classificazione, come mostrato nella Figura 1.1.

Figura 1.1

Tipici sistemi operativi, di pianificazione e di controllo.



Il modello evolutivo di Nolan e Gibson conteneva inizialmente quattro fasi, alle quali successivamente Nolan (1979) ne aggiunse altre due. Le sei fasi del modello sono Inizializzazione, Contagio, Controllo, Integrazione, Amministrazione dei dati e Maturità. Ciascuna fase risulta caratterizzata da alcuni aspetti riferibili all’orientamento del sistema informativo, all’orientamento dell’architettura applicativa e del portafoglio applicativo e all’approccio dell’utente (Tabella 1.1).

Tabella 1.1 Le sei fasi del modello evolutivo dei sistemi informativi.

	Fase 1 Inizializzazione	Fase 2 Contagio	Fase 3 Controllo	Fase 4 Integrazione	Fase 5 Amm. dati	Fase 6 Maturità
Orientamento del sistema informativo	Inserimento selettivo delle applicazioni per automatizzare le operazioni	Proliferazione delle applicazioni	Aggiornamento documentazione e ristrutturazione delle applicazioni esistenti	Aggiornare le applicazioni esistenti utilizzando la tecnologia dei database	Organizzazione e integrazione delle applicazioni	L'integrazione degli applicativi rispecchia il flusso informativo
Orientamento architettura informativa e portafoglio applicativo	Organizzazione tecnologica	Orientamento agli utenti	Supporto alle attività del middle management	Definizione di team di lavoro coinvolgendo anche gli utenti	Amministrazione e condivisione dei dati	Pianificazione strategica e gestione delle fonti dati
Approccio dell'utente	Distaccato	Entusiasmo superficiale	Arbitrariamente responsabile	Cultura della responsabiliz- zazione	Efficacemente responsabile	Aggregazione degli utenti e responsabiliz- zazione nell'elabo- razione dei dati

Il portafoglio applicativo inizialmente è molto ridotto, cresce lentamente e la tecnologia informatica utilizzata è semplice e meno evoluta, gli schemi organizzativi supportati sono semplici. Successivamente, man mano che l’uso dei sistemi informativi si diffonde, in azienda si manifestano richieste via via crescenti di nuove applicazioni, la tecnologia adottata diviene più avanzata e complessa e gli schemi organizzativi supportati sono sempre più complessi. Da un certo punto in poi la domanda di applicazioni si stabilizza e ci si orienta all’ottimizzazione/rinnovo delle applicazioni esistenti. Alla fine di questo ciclo di vita diventano possibili due alternative: il declino, ossia l’obsolescenza tecnologica del portafoglio applicativo e del sistema informativo globale, o uno spostamento verso l’alto della curva rappresentante il ciclo di vita, con nuovi traguardi da raggiungere. Tale spostamento della curva verso l’alto venne verificato dagli studiosi con la considerazione che l’innovazione nel settore IT introduceva nuove tecnologie che obbligavano il sistema informativo a un nuovo ciclo di vita a partire dal livello raggiunto.

La validità e l’utilità del modello a sei fasi sono state analizzate da molti ricercatori, fin da quando è stato pubblicato. In un’analisi delle precedenti ricerche sulle ipotesi del modello di Nolan, alcuni autori, quali, Benbasat et al. (1984) e King e Kraemer (1984), hanno trovato il supporto empirico in generale debole e non soddisfacente. Drury (1983) sostenne che, in pratica, la valutazione non considerava adeguatamente le fasi come proposte dal modello originale; in particolare, le ultime fasi sembravano essere molto semplici e non in grado di riflettere la complessità del mondo reale.

King e Kraemer credevano che il modello avesse parecchi punti di debolezza. In particolare, l’evidenza empirica per ciascuna fase era inconsistente e molte delle assunzioni erano troppo semplicistiche

per essere utili. Gli stessi autori tuttavia hanno ugualmente indicato che molti aspetti del modello hanno un riscontro positivo da parte di professionisti e ricercatori, il che ha avuto un'influenza considerevole sulla gestione dei SI fin dagli anni '70. La sua debolezza, la sua semplicità, probabilmente è allo stesso tempo la chiave della sua notorietà.

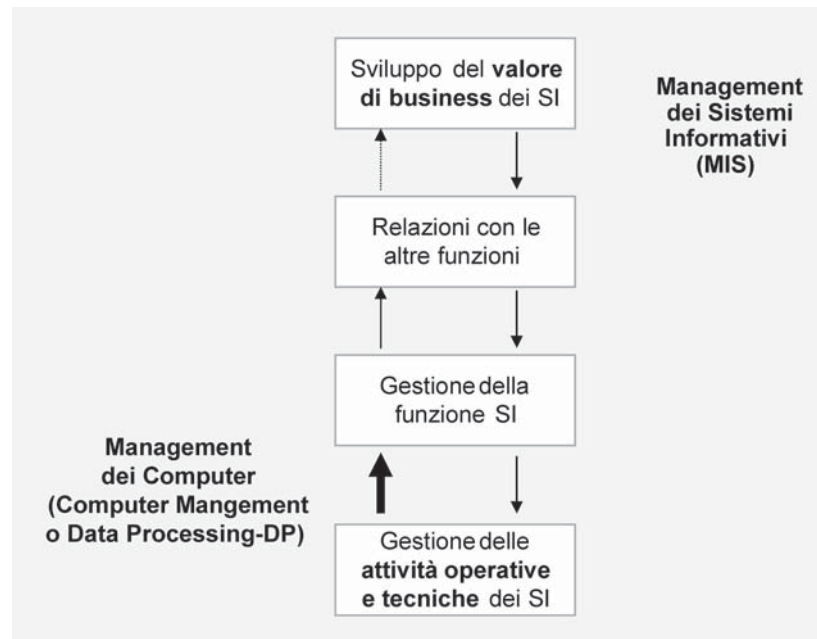
Il modello suggerisce un approccio evolutivo nel quale diverse forze controllano il successo dei SI/IT in azienda. Sin dall'inizio degli anni '90 la ricerca empirica supportava l'ipotesi che il modello forniva uno scarso aiuto al manager IT nel tentativo di creare una funzione sistemi informativi di successo all'interno dell'azienda. Nonostante le sue limitazioni, il modello continua ancora oggi a essere valido e utilizzato dai professionisti ed esperti.

Da un punto di vista più generale, il modello di Nolan (1979) si può suddividere in due macro-fasi, separate da un punto di transizione, posizionato tra la fase 3 e la fase 4, e cioè la fase di Management dei Computer (Computer Management o Data Processing, DP) e quella di Management dei Sistemi Informativi Aziendali.

In sostanza, si tratta di un cambiamento importante sia nel modo in cui vengono gestite le risorse dei SI/IT sia nel modo in cui valutare il ruolo dei SI/IT in azienda. Il cambiamento di tali relazioni è rappresentato nella Figura 1.2.

Figura 1.2

Transizione dalla fase di management dei computer a quella dei sistemi informativi (adattata da *EDP Analyser*, "How the management job is changing", giugno 1984, vol. 22, n. 6).



Nelle fasi iniziali la preoccupazione dei manager era rivolta alla gestione delle attività operative, di programmazione, di raccolta di dati e così via e quindi gli sforzi erano focalizzati prevalentemente sulle difficoltà tecniche e applicative piuttosto che sul valore di business delle applicazioni.

L'orientamento a un sempre maggiore valore di business dei sistemi informativi è invece il cardine della fase di Management dei Sistemi Informativi. Si tratta di una fase in cui è prioritaria la comprensione del ruolo dei sistemi informativi nell'azienda che, a sua volta, dipende dal ruolo dei sistemi informativi nell'ambiente esterno (particolarmente nell'ambiente transazionale o circostante).

Alcune ricerche, di Hirschheim e altri colleghi (1988), hanno studiato i problemi connessi al management dei sistemi informativi in azienda. I problemi di tale gestione sono stati descritti facendo riferimento a un modello a tre fasi.

1. *Erogazione dei servizi applicativi*: i problemi dei SI sono principalmente interni alla funzione sistemi informativi. Si tratta di problemi prevalentemente connessi alla capacità di erogazione dei servizi e al supporto dei sistemi informatici. Si tratta dei requisiti minimi che assicurano la credibilità nei confronti del top management, ma che non sempre assicurano la soddisfazione delle esigenze funzionali degli utenti.
2. *Ridefinizione dell'orientamento*: è lo stadio in cui è necessario stabilire buone relazioni con le principali funzioni aziendali, con gli uomini di business. Il focus sui problemi si estende all'esterno della funzione sistemi informativi e gli obiettivi diventano quelli di fornire servizi ad applicazioni ad alto valore aggiunto per le diverse funzioni aziendali.
3. *Riorganizzazione*: è lo stadio in cui l'elevato livello di consapevolezza creato, sia a livello di top management sia a livello di singola funzione o di middle management, determina la necessità di una riorganizzazione/revisione delle responsabilità con l'obiettivo di integrare/sincronizzare gli investimenti in SI sia con la strategia aziendale sia con quella delle varie funzioni aziendali. Un obiettivo chiave è soddisfare nel miglior modo possibile ogni richiesta di ciascuna funzione aziendale attraverso una condivisione delle responsabilità per la gestione di sistemi informativi e delle informazioni.

Dagli anni '60 ai primi anni '80, i SI e l'IT e la loro diffusione nelle aziende hanno visto il più importante processo di transizione, riconducibile a due importanti stadi.

1. Dal 1960 al 1970 circa, il *management dei computer* (CM) o *data processing* (DP).
2. Dal 1970 in avanti fino agli anni '80, il *management dei sistemi informativi* o MIS.

Naturalmente i due stadi non sono perfettamente distaccati ma esiste una loro parziale sovrapposizione: mentre lo stadio CM/DP continua a svilupparsi, emerge e si sviluppa quello MIS.

Gli anni '80 videro inoltre l'inizio di una terza importante generazione, descritta più avanti, che può essere definita come la fase dei sistemi informativi strategici (SIS) o di governance dei sistemi informativi (GIS).

1.4 Il contesto e gli insegnamenti dei primi due modelli

Tre sono stati sostanzialmente gli elementi di contesto, la cui evoluzione ha consentito lo sviluppo della concezione dei sistemi informativi e, dunque, il passaggio da un concetto di CM/DP a uno di MIS.

1. *L'hardware*: i sistemi informativi risultano "più a portata di mano" grazie, da una parte, alla riduzione del costo e della dimensione dell'hardware e, dall'altra, al miglioramento dell'affidabilità e della connettività.
2. *Il software*: è possibile sviluppare applicazioni aziendali più rapidamente, con maggiore precisione e anche da personale non necessariamente esperto. Questo grazie all'evoluzione dei software di base, divenuti più flessibili e con maggiori funzionalità, al miglioramento dei linguaggi di sviluppo e di programmazione e allo sviluppo dei software per la gestione delle basi dati. Inoltre, si verifica un aumento della disponibilità di pacchetti applicativi già "pronti da installare e utilizzare".

3. *La metodologia*: cioè modi di organizzare ed eseguire diverse attività, in modo più coordinato, sincronizzato ed efficiente facilitando/abilitando sia l'implementazione di sistemi più complessi sia la gestione di progetti di dimensioni maggiori.

L'evoluzione progressiva dei tre elementi di contesto determina numerose conseguenze con riferimento ai due approcci.

L'approccio *computer management* (CM) o *dell'elaborazione dei dati* (DP), prevalentemente orientato a migliorare l'efficienza aziendale attraverso l'automazione delle singole attività di business, rileva i suoi limiti:

- nello sviluppo di una visione informativa globale a livello aziendale;
- nell'assenza di una serie pianificata di legami tra le diverse procedure applicative;
- nella valutazione dell'economicità degli investimenti a livello aziendale;
- nell'uso di un approccio di programmazione, test e documentazione più strutturato;
- nella gestione di progetti complessi riguardanti diverse funzioni aziendali;
- nella necessità di pianificazione degli interventi applicativi aziendali così da avere miglioramenti nella produttività aziendale e non solo della singola procedura.

L'approccio di management dei sistemi informativi (MIS), prevalentemente focalizzato sul valore di business delle applicazioni, presenta dei limiti in merito:

- alla difficoltà di giustificare il ritorno sull'investimento con parametri alternativi a quelli economico-finanziari;
- alla necessità di intraprendere grandi progetti di revisione delle architetture informative orientate alla creazione di un'unica base dati;
- alla necessità di coinvolgere un congruo numero di utenti chiave nei progetti applicativi;
- all'orientamento del passaggio dall'erogazione di un servizio applicativo a quello di "on-demand information" in cui ogni utente ha a disposizione la possibilità di accedere/configurare le informazioni secondo le proprie esigenze;
- alla necessità di basare lo sviluppo del SI su linee guida organizzative e non solo tecnologiche.

In conclusione, nonostante il nuovo potenziale organizzativo e strategico dei SI e dell'IT, abilitato dallo sviluppo dei tre elementi di contesto prima individuati, ancora a metà degli anni '80/'90 si trovavano in mezzo al guado. Per superare tali problemi evolutivi e per controllare e coordinare le diverse esigenze complesse, e in crescita, furono introdotte procedure, pianificazione, strutture organizzative e processi di sviluppo sistemi.

1.5 Il modello di terza generazione: i sistemi informativi strategici

Oltre ai modelli delle prime generazioni, appena discussi, nei primi anni '80 si afferma il modello di una nuova generazione, cioè dei "sistemi informativi strategici". Secondo questa nuova concezione dei sistemi informativi, scopo degli investimenti in SI/IT è quello di migliorare la competitività cambiando la natura o il modo di condurre il business creando una fonte di vantaggio competitivo. Tale scopo non esclude, ma invece include l'obiettivo delle precedenti due fasi:

- migliorare l'efficienza operativa automatizzando l'elaborazione dei dati connessi ai processi basati sulle informazioni;
- aumentare l'efficacia di gestione aumentando la disponibilità di informazioni necessarie a supporto dei processi decisionali.

Gli obiettivi dei primi due modelli sono sostanzialmente un sottoinsieme dell'obiettivo dei SIS, cioè il miglioramento della competitività. Questo tende spesso a essere realizzato indirettamente utilizzando SI/IT per migliorare il modus operandi correnti. Per esempio, la ridefinizione dei processi aziendali business (BPR) è spesso vista come un modo per migliorare la competitività, ma questa è realizzata attraverso la revisione dei processi tenendo conto delle capacità dell'IT di fornire nuove possibilità innovative di design dei processi. Mentre l'obiettivo dei SIS è immediatamente riflesso nei risultati di business, non è così immediato nel caso degli obiettivi del DP e del sistema informativo nonostante possano contribuire notevolmente al successo del business.

Tra i vari autori, Wiseman (1985) ha descritto, sebbene in modo forse più sintetico di altri, la relazione sia tra le tre fasi sia tra l'evoluzione del portafoglio applicativo e gli obiettivi degli applicativi stessi. Le sue principali osservazioni sono le seguenti.

1. Così come un buon MIS fa affidamento su un buon sistema DP per avere informazioni accurate e al momento opportuno, un SIS (come quelli che collegano direttamente l'azienda con i suoi clienti via Internet) fa affidamento su buoni DP o MIS per ottenere le informazioni o le elaborazioni necessarie. Molte aziende hanno creato dei siti Web per consentire ai propri clienti di fare ordini online, ma che allo stesso tempo non sono integrati con la gestione dell'ordine e gli altri sistemi di back-office dell'azienda.
2. I sistemi SIS sono applicazioni sostanzialmente non diverse da quelle dei sistemi DP o MIS; molte funzioni infatti sono le stesse. È diverso invece il loro impatto sul business in relazione ai cambiamenti che causano o abilitano.
3. Le applicazioni strategiche potrebbero richiedere considerevoli cambiamenti sulle applicazioni dei sistemi DP o MIS, fino alla necessità di un nuovo sviluppo, non a causa di qualche difetto intrinseco ma perché potrebbero inibire i vantaggi ottenibili dal SIS.

Verso la fine del 1970, diverse aziende hanno iniziato a utilizzare i SI e l'IT in modo da cambiare fondamentalmente il modo di condurre il loro business, cambiando di conseguenza la loro posizione nel settore di appartenenza rispetto ai loro concorrenti, fornitori e/o clienti.

L'utilizzo dei SI e dell'IT influenzò direttamente la loro posizione competitiva, divenendo così una nuova arma per il miglioramento della loro competitività; questo ha implicato la definizione di una nuova relazione fra gli investimenti in SI/IT e lo sviluppo strategico.

1.6 I SI e l'IT come fattori strategici: classificazione

Da una base di ricerca su più di cento esempi e case study, condotta su un periodo di vent'anni in cui si è parlato di sistemi strategici, è emersa la seguente classificazione dei sistemi strategici, considerando anche le loro implicazioni. In generale, tali sistemi possono essere classificati in quattro tipologie, anche se alcuni presentano chiaramente delle caratteristiche comuni a più di una tipologia.

1. Quelli che *condividono le informazioni*, attraverso sistemi basati sulla tecnologia, con clienti/consumatori e/o fornitori e che modificano/cambiano la natura della relazione.
2. Quelli che producono una più efficace *integrazione delle informazioni* nei processi aziendali a valore aggiunto.
3. Quelli che agevolano l'azienda a *sviluppare, produrre, vendere e consegnare* nuovi o migliori prodotti e servizi basati sulle informazioni.
4. Quelli che *forniscono ai manager le informazioni a supporto dello sviluppo e della implementazione della strategia* (in particolare, per quelle analisi che integrano informazioni interne ed esterne).

Tra le altre classificazioni proposte, Notowidigdo (1984) suddivide i sistemi informativi strategici in:

1. sistemi interni che hanno vantaggi diretti per l'azienda;
2. sistemi esterni che hanno vantaggi diretti per i clienti dell'azienda.

Un approccio simile fu adottato da Venkatraman (1991) nel valutare come i vantaggi strategici dell'uso dell'IT risultino dal maggiore grado di cambiamento (e quindi di rischio) del business. Egli descrisse tre tipologie di utilizzo "rivoluzionario" dell'IT, le quali richiedono trasformazioni significative in termini di quello che l'azienda fa o di come lo fa.

- *Ridefinizione dei processi di business*: i SI/IT consentono di riallineare le attività di business e le loro relazioni per migliorare le performance.
- *Ridefinizione delle relazioni aziendali*, cambiando cioè sia il modo in cui l'azienda e i suoi partner commerciali utilizzano le informazioni sia il modo con cui il settore esegue/gestisce i processi che forniscono valore aggiunto.
- *Ridefinizione degli obiettivi di business*: estendendo il mercato o l'insieme dei prodotti offerti, basati sulle informazioni o cambiando il ruolo dell'azienda nel settore di appartenenza.

Tali approcci, sebbene non identici, sono simili alla classificazione iniziale delle quattro tipologie dei sistemi informativi strategici. È chiara l'enfasi che viene data al "cambiamento" come fattore critico per ottenere vantaggi strategici dall'utilizzo dei SI/IT.

Tornando alla classificazione iniziale, ogni tipologia d'applicazione strategica dei SI/IT ha implicazioni diverse in termini di identificazione, di pianificazione e di implementazione.

1.6.1 Collegamento con i clienti e i fornitori

Le funzioni coinvolte nei sistemi di collegamento con l'ambiente esterno all'azienda sono quelle delle vendite/marketing, di distribuzione ai clienti e di acquisto/ricezione/controllo di qualità dai fornitori.

1.6.2 Integrazione più efficace dei processi interni

Produrre un'efficace integrazione interna delle informazioni richiede all'azienda di superare alcune barriere tradizionali per adottare con successo i SI/IT proprio nelle fasi del DP e dei MIS: condividere informazioni, riorganizzare i ruoli e così via. Per esempio, l'attività di telemarketing può ridurre in maniera considerevole il costo per la generazione di nuovi ordini. Tutte le informazioni rilevanti relative al cliente e alla capacità dell'azienda di rispettare le consegne sono essenziali per rendere la vendita efficace.

In questo contesto, gli *Enterprise Resource Planning* (ERP) sono sistemi che integrano le informazioni e i processi basati sulle informazioni intra e inter-aree funzionali aziendali.

1.6.3 Prodotti e servizi basati sulle informazioni

Realizzare dei vantaggi da questa tipologia di sistemi richiede una piena conoscenza dei prodotti/servizi offerti dal settore, i loro vantaggi e, in particolare, con che scopo vengono utilizzati dal cliente e qual è il valore che questi ne ottiene. Evidentemente, è necessario conoscere molto bene anche i propri prodotti/servizi e gli strumenti per offrirli.

Nell'utilizzo di Internet, molte aziende si sono preoccupate di come aggiungere più valore ai prodotti tangibili fornendo servizi aggiuntivi basati sulle informazioni. Esempi di tali servizi possono essere il supporto online, la possibilità di monitorare lo stato di evasione dell'ordine, lo stato della spedizione e così via. Molti di questi servizi sono basati su una stretta relazione sia con clienti sia con i fornitori.

Un esempio è rappresentato dai servizi di online banking che le banche offrono accanto ai tradizionali prodotti/servizi.

1.6.4 Sistemi informativi direzionali

Il top management, a supporto delle decisioni strategiche, ha bisogno di informazioni strutturate riguardanti i mercati, i clienti attuali e potenziali, il grado di tecnologia del proprio settore e degli altri e l'andamento della finanza e dell'economia a livello locale e globale.

È altresì importante l'esperienza dei "decisioni" cioè di coloro che sono chiamati a prendere le decisioni. Spesso, l'intuizione o le "sensazioni" giocano un ruolo determinante in molti processi decisionali.

I sistemi informativi manageriali, storicamente, hanno dimostrato di non soddisfare questa necessità di informazioni e, per tale motivo, hanno una bassa considerazione da parte del top management dell'azienda.

Questo accade almeno per due principali ragioni:

1. la mancanza, nei sistemi, di informazioni esterne;
2. la semplicità dei sistemi, la presenza di dati grezzi a volte non contestualizzati (per esempio, i manager richiedono "conoscenza", non solo informazioni).

La Figura 1.3 schematizza i sistemi informativi strategici in base al loro contesto e ai loro obiettivi. Per esempio, l'automazione delle transazioni di business esistenti non è considerato strategico fintantoché migliora solamente l'efficienza della gestione delle transazioni.

Inoltre, i sistemi informativi a supporto delle attività strategiche (EIS) sono ricompresi tra i MIS, così come accade nella maggior parte delle versioni "più estese" dei MIS.

1.7 Fattori critici di successo nei sistemi informativi strategici

Di seguito sono evidenziati alcuni fattori chiave che sembrano critici per il successo dei SIS (sistemi informativi strategici). Tali fattori sono spesso in disaccordo con gli approcci tradizionali ai SI/IT, mentre hanno più elementi in comune con l'innovazione di business.

Figura 1.3
Sistemi informativi
aziendali: contesto
e obiettivi.

Obiettivi Contesto	Efficienza operativa	Efficacia gestionale	Vantaggi di business dal cambiamento
	1. Elaborazione dei dati; automatizzazione delle attività e dei processi di business	2. Sistemi informativi aziendali (e "SI direzionali")	3. Integrazione dei processi interni di business e ridefinizione dell'organizzazione
Interno			
Esterno	4. Collegamenti elettronici tra aziende per automatizzare lo scambio di dati	5. Condivisione delle informazioni tra aziende accedendo reciprocamente alle risorse informative	6. Integrazione processi esterni di business, cambiando i ruoli delle aziende nel settore

1. *Focus verso l'ambiente esterno:* prestano cioè attenzione ai clienti, ai concorrenti, ai fornitori, agli altri settori, e in generale alle relazioni con l'ambiente esterno. Tradizionalmente i SI/IT erano concentrati sui processi e sulle problematiche interne.
2. *Creare valore aggiunto, senza focalizzarsi solo sulla riduzione dei costi:* sebbene la riduzione dei costi possa aumentare in seguito all'espansione del business, riducendo i costi marginali, l'ottimo sembra essere "fare meglio e non solo in modo più economico". Questo obiettivo è inoltre compatibile con la necessità aziendale di differenziarsi dai concorrenti: migliori prodotti, migliori servizi.
3. *Condividere i vantaggi:* all'interno dell'azienda, con i fornitori, i clienti, i consumatori e, se necessario, anche con i concorrenti. In molti casi in passato, i vantaggi dei sistemi informativi non erano condivisi all'interno dell'azienda ma erano utilizzati come vantaggio competitivo della singola funzione o divisione aziendale. Questo riduce complessivamente i vantaggi e ne limita il conseguimento. Condividere dei vantaggi implica un "buy in", un forte impegno a operare con successo e un costo di scambio. Molti esempi utilizzano la condivisione dei vantaggi per creare delle barriere all'entrata nel settore.
4. *Capire i clienti e ciò che fanno con il prodotto o il servizio:* come ottengono valore dal prodotto/servizio e i problemi che possono incontrare nel realizzare tale valore.
5. *L'innovazione guidata dal business e non dalla tecnologia:* le pressioni del mercato hanno guidato gli sviluppi in molti casi. "I più grandi fallimenti nell'uso dell'IT sono spesso basati sull'utilizzo di una tecnologia vincente e su una poco attenta gestione prospettica del business. I casi di successo provengono dall'adozione di una tecnologia "abbastanza" buona e una chiara comprensione del cliente".
6. *Sviluppo incrementale,* cioè step dopo step e non un'applicazione totale alla realtà. Molti esempi mostrano un approccio per step, realizzando un oggetto/prodotto di successo ed estendendolo con ulteriori sviluppi. L'estensione non si ferma nel momento in cui si raggiunge un successo, ma considerando come si può ancora sviluppare e migliorare.

7. *Utilizzare le informazioni acquisite* dai sistemi informativi per sviluppare il business. Molte aziende di vendita al dettaglio hanno segmentato i loro clienti secondo alcuni modelli di acquisto, ricavati dalle transazioni, potendo così offrire per esempio cataloghi differenziati e offerte speciali.

Tali fattori, in generale, implicano una diversa attitudine all'utilizzo dei SI/IT, prevalsa in passato, il che significa che sono necessari nuovi approcci ai SI/IT per scoprire nuove opportunità e per gestirle con successo.

La tecnologia di per sé è un fattore “abilitante”, che fornisce dei vantaggi a breve termine e l'opportunità di sviluppare nuovi sistemi in grado di catturare e utilizzare informazioni potenzialmente preziose. Allo stesso tempo i concorrenti potrebbero però acquistare la stessa tecnologia, con la conseguenza che alcuni vantaggi potrebbero ridimensionarsi (anche di molto) già nel breve periodo. Tuttavia, il nuovo sistema informativo sviluppato, grazie all'IT, potrebbe offrire dei vantaggi difficilmente replicabili dalla concorrenza. Il guadagno potenziale dipenderà, in sostanza, da come e quanto saranno modificati i processi di business e le relazioni.

L'azienda che intende sostenere il proprio vantaggio competitivo dovrà utilizzare le informazioni ottenute dai sistemi informativi per migliorare i suoi prodotti e/o servizi in modo sia da recepire l'influenza del mercato sia da soddisfarne i fabbisogni.

1.8 Le implicazioni manageriali

Il valore di business dei sistemi informativi è divenuto sempre più rilevante da quando il livello di coinvolgimento richiesto al management è stato elevato al livello strategico.

Naturalmente è necessaria un po' di cautela per evitare di esagerare nell'importanza da dare ai SI/IT. King (1987) esprime la sua preoccupazione nel fatto che “l'argomento del vantaggio competitivo iniziava a essere utilizzato in maniera eccessiva, soprattutto per razionalizzare progetti che non potevano essere altrimenti giustificati”. Questo aveva come effetto la perdita di credibilità del management. Egli affermò che occorreva gestire i SI/IT e le loro varie applicazioni in funzione del tipo di vantaggio che erano in grado di fornire, miglioramento dell'efficienza, dell'efficacia e/o della competitività attraverso i cambiamenti di business.

Pur con qualche cautela, comunque, un'azienda non può permettersi di ignorare le opportunità strategiche che derivano dai SI/IT, per cui deve considerare “le informazioni potenziali come una risorsa strategica da incorporare nei processi quotidiani di gestione del proprio business, in modo che tutti i manager inizino a pensare in questi nuovi termini”.

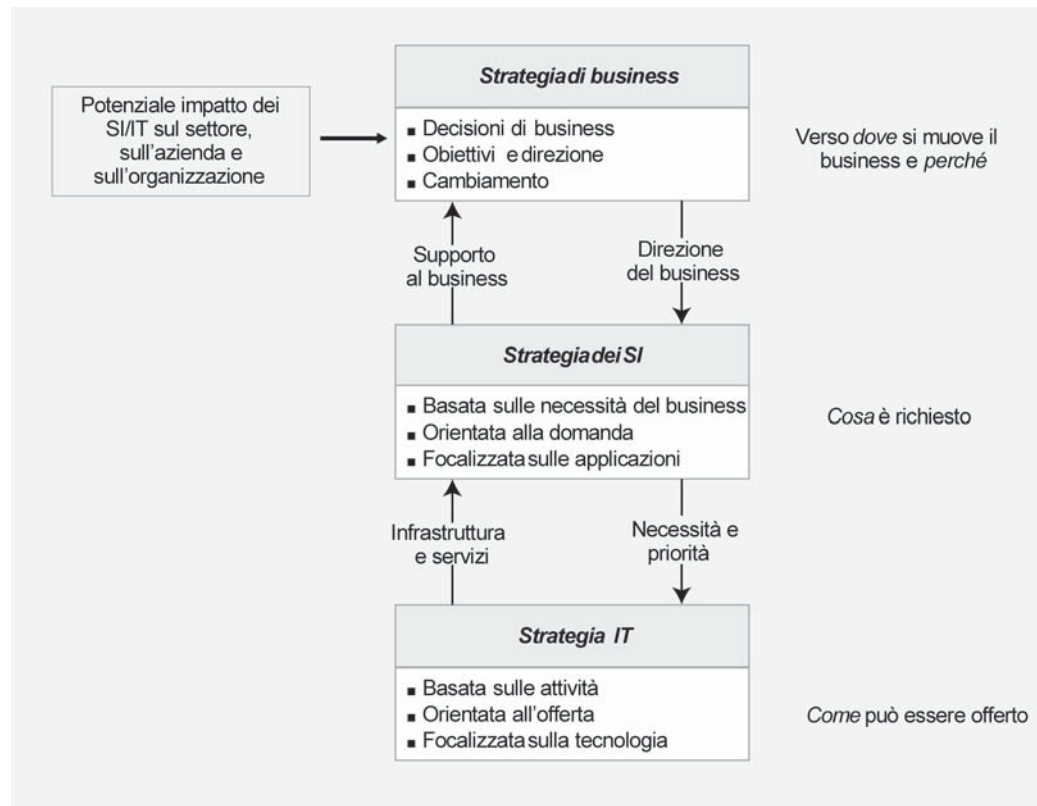
Earl (1992) sostiene che concentrarsi esclusivamente sulla tecnologia non conduce ad applicazioni strategiche dell'IT di successo. Egli suggerisce che il percorso migliore per ottenere dei vantaggi strategici dai SI/IT è “di focalizzare l'attenzione sulla ridefinizione del business partendo dall'analisi dei problemi di business attuali e dai cambiamenti ambientali, e considerando l'IT solo come un ingrediente della soluzione”. Earl ricorda la distinzione che deve essere fatta tra strategia dei SI e strategia dell'IT. Trova, infatti, che la maggior parte delle strategie IT, nel periodo osservato, sono fortemente orientate alle questioni tecnologiche e alla terminologia tecnica, mentre sono deboli nell'identificazione dei fabbisogni applicativi e del business. Egli suggerisce che la strategia dei SI debba riguardare i sistemi informativi, o il set di applicazioni, richiesti dall'azienda. In sostanza, la strategia dei SI deve porsi la domanda del “cosa” mentre la strategia IT dovrebbe riferirsi alle questioni tecnologiche, dell'infrastruttura e delle competenze specialistiche necessarie, deve cioè porsi la domanda del “come”. Questa relazione è descritta nella Figura 1.4.

Conseguentemente, i SI/IT dovrebbero essere considerati come un elemento, tra gli altri, del business che – come, per esempio, il marketing, la produzione o gli acquisti – per produrre dei vantaggi dovrebbero essere coordinati efficientemente ed efficacemente con la strategia aziendale, potendo rappresentare – se opportunamente gestiti – un'importante leva strategica e competitiva. Questo implica adottare un approccio strategico nello sviluppo dei sistemi informativi e tecnologici che sono quindi interrelati con gli altri elementi.

La Figura 1.4 mostra le relazioni tra i tre livelli strategici: le strategie di business, la strategia SI e quella IT. Il flusso di tali relazioni vede a) l'identificazione dell'impatto potenziale dei SI/IT sulla strategia di business, b) la valutazione di quali sistemi ed informazioni sono necessari per il perseguimento della strategia, c) della definizione di come realizzare tali sistemi informativi sfruttando la tecnologia.

Figura 1.4

Relazione tra le strategie di business, SI e IT.



1.9 Cos'è una strategia SI/IT?

È stato già accennato al concetto di strategia SI/IT, senza però definire esattamente cosa si intende. La Figura 1.4 fornisce uno schema generale dei suoi elementi fondamentali. Sostanzialmente, una strategia SI/IT è composta da due parti: una componente SI e una componente IT. La strategia SI definisce i requisiti aziendali in termini di “domanda” di informazioni e di sistemi di supporto alla strategia complessiva aziendale.

Di conseguenza la strategia SI definisce e assegna le priorità agli investimenti necessari per costruire il portafoglio applicativo “ideale” e per realizzare i vantaggi attesi e i cambiamenti richiesti per conseguirli, nell’ambito delle risorse disponibili e delle interdipendenze tra i sistemi.

La strategia IT riguarda la definizione di come la tecnologia può supportare “la domanda aziendale in termini di informazioni e sistemi”. Tale strategia è sostanzialmente relativa all’offerta tecnologica; infatti indirizza l’approvvigionamento delle risorse e competenze in ambito IT (hardware, software e telecomunicazioni) e dei servizi, come per esempio le attività di sviluppo e di supporto all’utente.

1.9.1 Allineamento strategico

C’è differenza tra avere una strategia SI/IT e avere una strategia SI/IT che contribuisca alla creazione di valore dell’azienda. Verso la fine degli anni ’80, furono sviluppati alcuni modelli per valutare l’estensione e il grado di allineamento delle strategie di business con le strategie SI/IT.

Il concetto di allineamento strategico è oramai noto da molti anni, mentre diverso è il caso dell’allineamento strategico con i SI/IT. In questo contesto il Massachusetts Institute for Technology (MIT) ha svolto un progetto di ricerca, *Management in the 1990s*, dal quale è emerso un significato particolare del concetto di allineamento strategico nel contesto della gestione dei SI/IT.

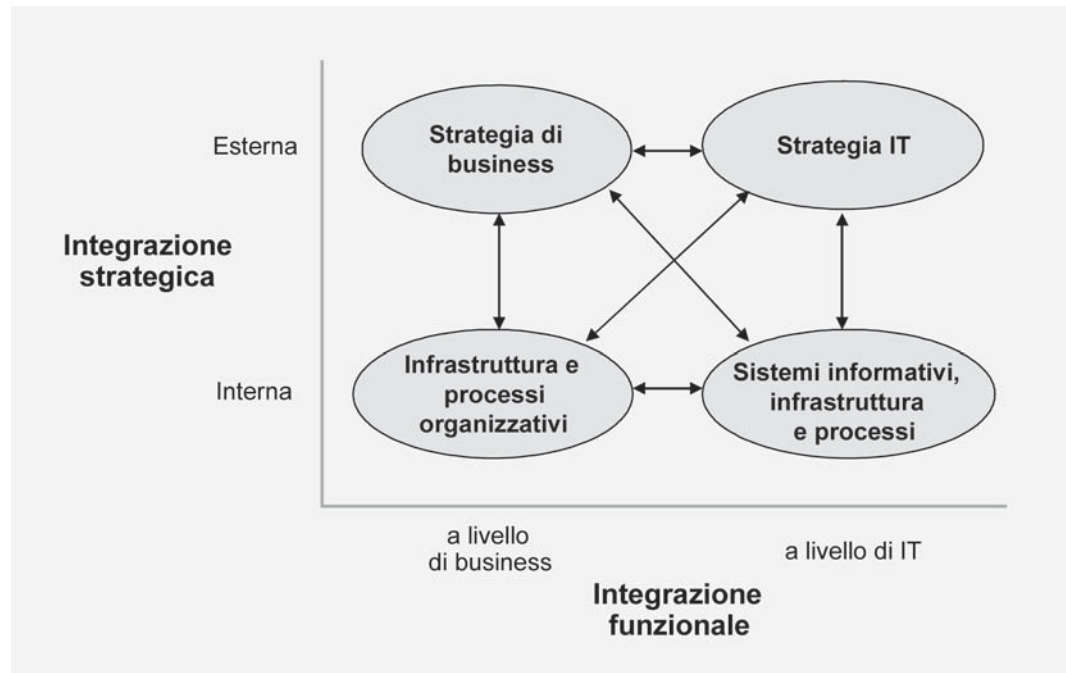
L’interpretazione del MIT è basata sulla premessa che l’incapacità dell’azienda di ottenere valore dagli investimenti in SI/IT è, in parte, dovuta alla mancanza di allineamento tra strategie di business e strategie SI/IT. Il progetto ha sviluppato un modello che ha rappresentato l’allineamento dinamico tra il contesto strategico di business e il contesto strategico IT. Tale modello è fondamentalmente basato sulla relazione tra l’integrazione strategica e l’integrazione funzionale. Henderson e Venkatraman affermano che la prospettiva di allineamento dovrebbe considerare almeno quattro dimensioni strategiche di scelta: la strategia di business, l’infrastruttura e i processi organizzativi, la strategia IT e le infrastrutture e i processi IT (Figura 1.5). Ogni dimensione ha le proprie sotto-dimensioni fondamentali. Il modello di allineamento strategico (MAS) valuta le varie scelte strategiche che i manager affrontano e come le correlano.

1.9.2 Perché avere una strategia SI/IT?

È stato discusso quanto importante sia per un’azienda avere una strategia SI/IT. Ci sono infatti numerose ricerche ed evidenze empiriche che illustrano le conseguenze alquanto severe di *non* avere una strategia SI/IT. Tra queste:

- gli investimenti in sistemi informativi fatti senza supportare debitamente gli obiettivi di business;
- la perdita di controllo dei SI/IT porta a volte a perseguire degli obiettivi che sono incompatibili con gli attuali sistemi e infrastrutture tecnologici;
- i sistemi informativi risultano non integrati. Questo può portare anche a una duplicazione di sforzi e di dati dai quali si ricavano informazioni imprecise e non coerenti;
- non avere alcuno strumento per gestire le priorità dei progetti e delle risorse SI e il continuo cambiamento dei piani conducono a ridurre la produttività;
- non avere meccanismi e strumenti per decidere il livello ottimale di risorse e per il loro approvvigionamento;
- il sistema informativo può risultare non disponibile, incoerente, impreciso o troppo lento;

Figura 1.5
Il modello
di allineamento
strategico (MAS).



- le incomprensioni tra gli utenti e gli specialisti IT possono creare dei conflitti e delle insoddisfazioni;
- alcuni investimenti in infrastrutture possono risultare inadeguati;
- la valutazione dei progetti è solamente di tipo finanziario;
- gli eventuali problemi creati da investimenti in SI/IT possono essere una fonte di conflitti tra le diverse parti dell'azienda;
- giustificare degli investimenti in modo localizzato o parziale può produrre dei vantaggi che possono essere allo stesso tempo controproducenti nel contesto complessivo aziendale;
- i sistemi, mediamente, hanno un'aspettativa di vita inferiore rispetto a quella attesa e richiedono una spesa maggiore per il loro ri-sviluppo che è maggiore di quella che dovrebbe essere necessaria.

1.10 Il contesto per la strategia SI/IT

Prima di iniziare lo sviluppo di una strategia SI, è importante capire il contesto nel quale deve essere sviluppata, che probabilmente è diverso in aziende diverse.

Di seguito vengono presentate due prospettive:

- quella interna, che si focalizza sul ruolo dei SI/IT in azienda;
- quella esterna, che esplora le dinamiche complessive dei SI/IT.

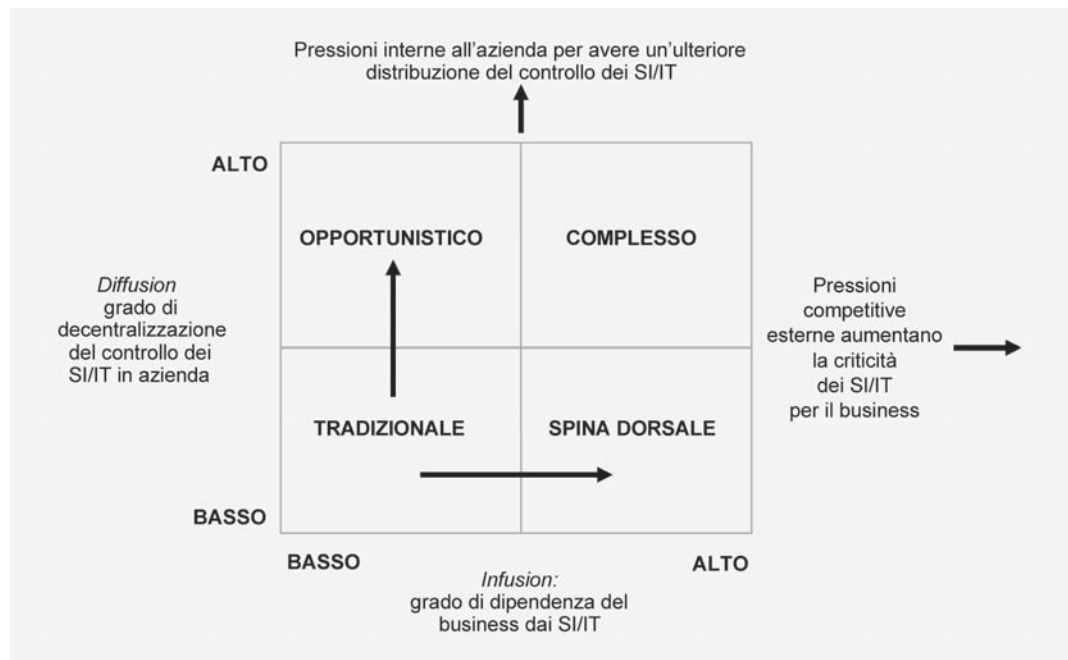
1.10.1 Il contesto interno

Sullivan (1985) ha proposto una semplice matrice per spiegare come l'ambiente strategico dei SI/IT sia influenzato da forze esterne al controllo di qualunque azienda. Egli descrive due dimensioni secondo cui un'azienda può considerare le implicazioni di tali forze:

- *infusion*: esprime il grado di dipendenza dell'azienda dal SI/IT per lo svolgimento della sua attività tipica e fondamentale;
- *diffusion*: esprime il grado di diffusione dell'IT in azienda e delle decisioni riguardanti il suo impiego.

Queste due dimensioni riflettono non solo la natura sempre più strategica dei SI/IT ma anche i vantaggi derivanti dal cambiamento della tecnologia e dalla possibilità di utilizzarla senza il bisogno di personale tecnico specializzato. Il framework di Sullivan è mostrato nella Figura 1.6. Considerando i livelli alto e basso sia di infusion sia di diffusion, si definiscono quattro scenari essenzialmente diversi.

Figura 1.6
Scenari della strategia
SI/IT.



1. *Bassa diffusion/bassa infusion*: in questa situazione si ha un controllo fortemente centralizzato delle risorse IT dove il SI non è critico per il business. Tale scenario è descritto da Sullivan come l'ambiente "tradizionale" e tipico di un'azienda che utilizza l'IT solamente per migliorare l'efficienza su una base di sistemi diversi.
2. *Bassa diffusion/alta infusion*: anche in questo caso il controllo è fortemente centralizzato e il SI è critico per lo svolgimento dell'attività tipica dell'azienda. In caso di fallimento del sistema informativo, l'attività principale potrebbe esserne seriamente svantaggiata. Per tale motivo, sono necessari

sistemi ad alta qualità con un elevato grado d'integrazione. I sistemi informativi diventano quindi, come dice Sullivan, parte della "spina dorsale" dell'azienda.

3. *Alta diffusion/bassa infusion*: in questa situazione il controllo è molto decentrato e i manager hanno la possibilità di soddisfare le loro priorità locali. Ogni integrazione tra i sistemi informativi richiede la cooperazione tra utenti (una "federazione" di interessi) e non dell'azienda nel suo insieme. L'approccio del management è essenzialmente "opportunistic", guidato da priorità a breve termine che possono creare vantaggi anche solo in alcune aree.
4. *Alta diffusion/alta infusion*: in questo scenario il controllo è molto decentrato e il successo dell'attività aziendale dipende dai sistemi informativi, entrambi gli elementi permettono di realizzare gli obiettivi di business complessivi. Sullivan descrive questo come un ambiente "complesso" che è difficile da gestire. Un controllo troppo centralizzato potrà limitare l'innovazione e quindi si potranno perdere le nuove opportunità strategiche; un controllo troppo frammentato, invece, può incidere negativamente sull'efficace e efficiente funzionamento dell'azienda nel suo insieme.

Le aziende, nell'evoluzione dalla fase dei DP a quella dei MIS, tendevano a spostarsi dal quadrante basso-basso verso uno dei due alto-basso. Lo spostamento dipendeva dal momento della loro evoluzione e dalla disponibilità di soluzioni tecnologiche centralizzate (il mainframe) o decentralizzate (distribuite o PC) in funzione dei fabbisogni dei DP e dei MIS. L'arrivo della fase dei sistemi informativi strategici (SIS) costrinse le aziende a orientarsi verso il quadrante alto-alto e, secondo lo scenario in cui si erano collocate nelle fasi precedenti, i cambiamenti necessari erano diversi e, a volte, critici.

Complessivamente la conseguenza di tutto ciò è che le aziende divennero sempre più dipendenti dai SI/IT trovandosi ad acquisire i vantaggi ed evitare gli svantaggi sia di una maggiore diffusion sia di una maggiore infusion. La gestione di maggiore dipendenza (maggiore "infusion" e maggiore "diffusion") richiede un approccio bilanciato alla gestione dei sistemi informativi, che ha come conseguenza il trovarsi a dover gestire diversi approcci manageriali e crescenti pressioni sia interne sia esterne, come indicato nella Figura 1.6. Da un lato è necessaria una maggiore pianificazione e controllo dei sistemi informativi (maggiore centralizzazione), dall'altro lato, però, per facilitare l'uso innovativo dei SI/IT, al fine di creare vantaggi futuri, chi controlla e gestisce la tecnologia ha bisogno di essere vicino all'utente in modo da "facilitare" gli opportuni collegamenti tra i fabbisogni di business e la soluzione tecnologica da adottare.

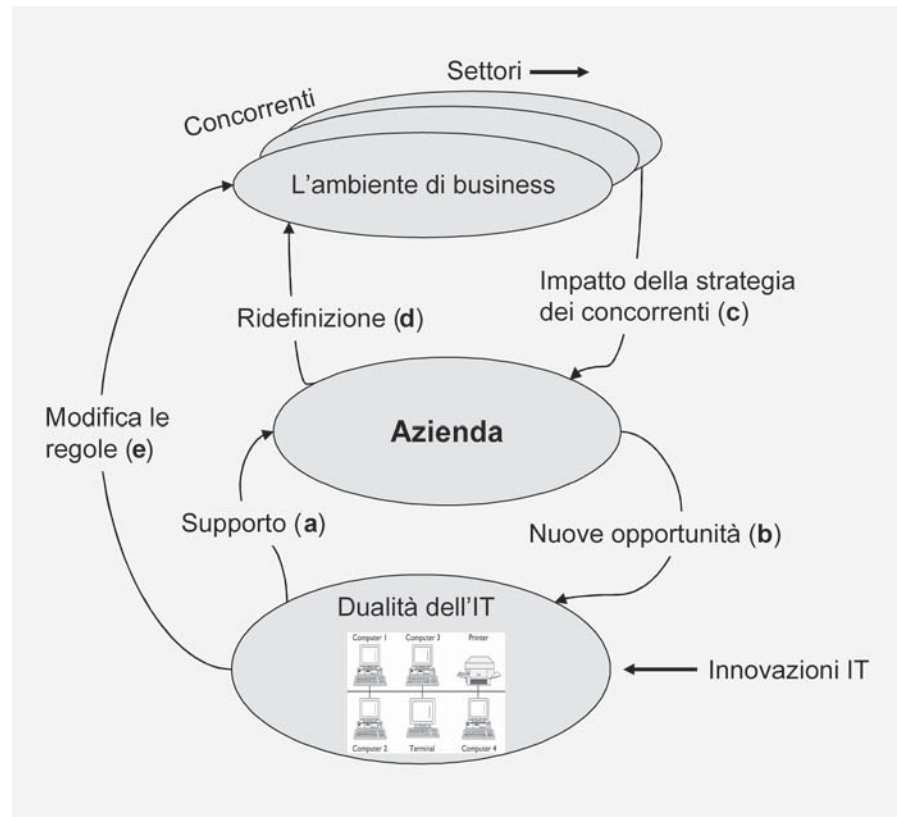
1.10.2 Il contesto esterno

Le dinamiche dell'IT e le relative conseguenze sullo sviluppo delle strategie sia di business sia dei SI/IT sono complesse. La Figura 1.7 cerca di rappresentare tale complessità e catturarne le dinamiche. È rappresentata la "dualità" della tecnologia in quanto non solo *supporta* la strategia aziendale (freccia "a", allineamento strategico) ma può anche *definire* il business, cioè la tecnologia è fattore critico per perseguire la strategia aziendale (freccia "b", impatto competitivo). La tecnologia facilita anche nuovi modi di organizzare, nuove innovazioni di processo e può abilitare la creazione di aziende innovative "basate sulla rete".

Un'azienda non opera singolarmente (a meno che non occupi una posizione di monopolio) ma ha concorrenti ed è parte di un settore e di un ambiente più ampi. Le mosse dei concorrenti, inclusi i nuovi entranti, incidono sulle dinamiche di un settore e, di conseguenza, sull'azienda stessa e sulle sue strategie (freccia "c"); allo stesso momento, le mosse strategiche fatte dall'azienda incidono su quelle dei concorrenti (freccia "d").

Le innovazioni tecnologiche possono avere effetti dirompenti su un settore (freccia "e"), riscrivendo le regole competitive e impattando sulla sua struttura tradizionale del settore.

Figura 1.7
Dinamiche SI/IT: influenza
e impatto.



È all'interno di questo contesto che il management deve definire come l'azienda può utilizzare al meglio la tecnologia per colmare la discontinuità delle informazioni, le asimmetrie e le imperfezioni per conseguire dei vantaggi di business.

1.11 Verso la quarta generazione: la potenzialità organizzativa dei SI

Sia la letteratura sui SI, prima del 1990, sia vari report e indagini riportano un ottimismo generale nella capacità dei SI/IT di creare vantaggi competitivi sostenibili.

Sebbene alcune aziende possano ottenere qualche vantaggio per essere "first mover" nell'adozione di un'applicazione innovativa, questa può essere tuttavia rapidamente copiata, eliminando così la sostenibilità di qualunque vantaggio. In particolare, la protezione dell'innovazione è quasi inesistente e, anzi, mantenerla segreta è difficile, specialmente per i sistemi utilizzati dai clienti o fornitori. D'altro canto, l'utilizzo di pacchetti applicativi standard come quelli sviluppati da alcune aziende produttrici (per esempio, SAP, Baan o J.D. Edwards), strategia oggi molto comune, può limitare la capacità dell'azienda a innovare. Allo stesso tempo, gli investimenti in infrastrutture tecnologiche stanno divenendo sempre più significativi, e decisioni inadeguate possono influenzare severamente la capacità dell'azienda di

rispondere rapidamente e flessibilmente ai cambiamenti delle condizioni di mercato, tanto da risultare in una significativa perdita di competitività.

La sostenibilità, dal punto di vista dei SI, può essere definita come la capacità di un'azienda di acquisire continuamente valore di business attraverso i SI/IT, ottenendo così un vantaggio. La sfida che oggi affrontano sia i professionisti sia i ricercatori è capire cosa contribuisce allo sviluppo di questa sostenibilità.

In un'analisi concettuale dei SI/IT e del vantaggio competitivo, alcuni ricercatori (Mata e altri (1995)) conclusero che solamente le capacità di gestione dei SI possono essere fonte di vantaggio sostenibile, cioè la capacità dei manager dei SI di capire e apprezzare i bisogni del business, la loro capacità di collaborare con i manager funzionali, la loro capacità di coordinare le attività dei SI in modo che siano di supporto ai manager di funzione e la loro capacità di anticipare i fabbisogni aziendali futuri. Tali ricercatori inoltre propongono che, nella ricerca dei vantaggi competitivi basati sui SI/IT, le aziende debbano concentrarsi meno sull'IT in quanto tale e maggiormente sul processo di organizzazione e gestione dell'IT. Un ulteriore supporto a questa considerazione è stato dato da altri studi (Dvorak e altri (1997)) i quali affermarono che quello che distingue aziende con un alto ritorno degli investimenti in IT non è l'aspetto prettamente tecnico ma il modo in cui i manager gestiscono i SI/IT. Sostiene la stessa tesi anche Keen (1980), il quale ha affermato che "la maggiore differenza nei vantaggi competitivi che le aziende acquisiscono dall'IT è contenuta proprio nel modo in cui l'IT è gestito e non è una questione prettamente tecnica".

Ripasso del capitolo

Parole chiave

Applicazione
Classificazione dei sistemi strategici
Data processing
Due generazioni evolutive
Elaborazione dei dati

Fattori critici di successo nei SIS
Gestione SI esperienziale
Information Technology
Modello evolutivo di Nolan e Gibson
Ruolo dei sistemi informativi

Sistema informativo
Sistema informativo aziendale
Sistema informativo strategico (SIS)
Software di gestione delle basi di dati

Strategie dei sistemi informativi
Strategie di business
Strategie IT
Tecnologia
Utilizzo rivoluzionario dell'IT

Domande di ripasso

1. Molte aziende hanno già fatto delle scelte in termini di SI e IT. Nel caso in cui volessero rivedere tali scelte in un'ottica strategica, quali sarebbero le implicazioni?

2. Elencare e descrivere i vantaggi aziendali derivanti dall'utilizzo di un SI.

3. Sistema informativo e information technology spesso vengono utilizzati indifferentemente. Spiegate quali sono le differenze significative.

4. Il sistema informativo si riferisce alla concettualizzazione di "cosa" deve essere supportato, mentre l'information technology descrive il modo con cui fornire tale supporto. Siete d'accordo con questa affermazione? Motivare la risposta.

5. Cosa rappresenta il modello di Gibson e Nolan?

6. Quali sono le principali critiche mosse al modello di Gibson e Nolan?
7. Descrivere le due generazioni principali di evoluzione dei sistemi informativi.

8. Nella terza generazione del processo evolutivo dei sistemi informativi, quali sono gli obiettivi nell'utilizzo dei SI e dell'IT?

9. Come possono essere classificati i sistemi informativi strategici?

10. Comprendere il comportamento dei consumatori, in termini di prodotti o servizi acquistati e di valore che questi ne trae, costituisce un fattore critico di successo per un sistema informativo strategico? Motivare la risposta.

11. Descrivere le relazioni tra la strategia di business, la strategia SI e la strategia IT.

12. Definire formalmente cosa sono le strategie SI e IT.

13. Quali sono gli scenari della strategia SI/IT (proposti dal framework di Sullivan per l'analisi del contesto interno)?

14. Quali sono le dinamiche del contesto esterno della strategia SI/IT?

Bibliografia

- Anthony, R. N., 1965, *Planning and Control: A Framework for Analysis*, Harvard University Press, Cambridge, MA.
- Barney, J. B., 1991, "Firm resources and sustained competitive advantage", *Journal of Management*, vol. 17, pp. 99–120.
- Benbasat, I., Dexter, A. S., Drury, D. H., Goldstein, R. C., 1984, "A critique of stage hypothesis: theory and empirical evidence", *Communications of the ACM*, vol. 27, n. 5, pp. 476–485.
- Benjamin, R. I., Rockart, J. F., Scott Morton, M. S., Wyman, J., 1984, "Information technology: A strategic opportunity", *Sloan Management Review*, primavera, pp. 3–10.
- Checkland, P., 1981, *Systems Thinking Systems Practice*, John Wiley & Sons, Chichester, UK.
- Checkland P., Holwell S. 1998, *Information, Systems and Information Systems: Making Sense of the Field*, John Wiley & Sons, Chichester, UK.
- Davenport, T. H., 1998, "Putting the enterprise back into enterprise systems", *Harvard Business Review*, luglio/agosto, pp. 121–131.
- Dvorak, R. E., Holen, E., Mark, D., Meehan, W. F., 1997, "Six principles of high-performance IT", *The McKinsey Quarterly*, n. 3, pp. 164–177.
- Drury, D. H., 1983, "An empirical assessment of the stages of DP growth", *MIS Quarterly*, vol. 7, n. 2, pp. 59–70.
- Earl, M. J., 1992, "Putting IT in its place: A polemic for the nineties", *Journal of Information Technology*, vol. 7, pp. 100–108.
- Earl, M. J., 1996, "The risks of outsourcing IT", *Sloan Management Review*, primavera, pp. 26–32.
- Friedman, A., 1994, "The stages model and the phases of the IS field", *Journal of Information Technology*, vol. 9, pp. 137–148.
- Galliers, R. D., Somogyi, E. K., 1987, "From data processing to strategic information systems: A historical perspective", in Galliers, R. D. e Somogyi, E. K., editor, *Towards Strategic Information Systems*, Abacus Press, pp. 5–25.
- Galliers, R., Sutherland, A., 1991, "Information systems management and strategy formulation: The "stages of growth" model revisited", *Information Systems Journal*, vol. 1, pp. 89–114.
- Gibson, C. F., Nolan, R. L., 1974, "Managing the four stages of EDP growth", *Harvard Business Review*, gennaio/febbraio, pp. 76–88.
- Hirschheim, R., Earl, M. J., Feeny, D., Lockett, M., 1988, "An exploration into the management of the IS function: Key issues and an evolving model", in *Proceedings of the Joint International Symposium on IS: Information Technology Management for Productivity and Strategic Advantage*, IFIP TC-8, Open Conference, Singapore, marzo.
- King, W. R., 1987, "It's time to get out of the dark", *Datamation*, luglio.
- King, J. L., Kramer, K. L., 1984, "Evolution and organisational information systems: An assessment of Nolan's stage model", *Communications of the ACM*, vol. 27, n. 5, pp. 466–485.
- Mata, F. J., Fuerst, W. L., Barney, J., 1995, "Information technology and sustained competitive advantage: A resource-based analysis", *MIS Quarterly*, vol. 19, n. 4, pp. 487–505.
- Mingers, J. C., 1995, "What is the distinctive nature and value of IS as a discipline?", *Systemist*, vol. 17, n. 1, pp. 18–22.
- Nolan, R. L., 1973, "Managing the computing resource: a stag hypothesis", *Communications of the ACM*, vol. 16, n. 7, pp. 399–405.
- Nolan, R. L., 1979, "Managing the crises in data processing", *Harvard Business Review*, marzo/aprile, pp. 115–126.
- Notowidigdo, M. H., 1984, "Information systems: Weapons to gain the competitive edge", *Financial Executive*, vol. 52, n. 3, pp. 20–25.
- Porter, M. E., 1979, "How Competitive Forces Shape Strategy", *Harvard Business Review*, 52, 2, pp. 137–146.
- Porter, M., 2001, "Strategy and the Internet", *Harvard Business Review*, marzo, pp. 63–78.
- Prahalad, C. K., Krishnan, M. S., 1999, "The new meaning of quality in the information age", *Harvard Business Review*, settembre/ottobre, pp. 109–118.
- Rangan, S., Adner, R., 2001, "Profitable Growth in Internet-Related Business: Strategy Tales and Truths", *Working Paper*, febbraio, INSEAD, Fontainebleau, France.
- Rivard, S., Huff, S., 1988, "Factors of success for end-user computing", *Communications of the ACM*, vol. 31, n. 5, pp. 552–561.
- Rockart, J., 1988, "The line takes leadership – IS management in a wired society", *Sloan Management Review*, estate, pp. 57–64.
- Strassman, P. A., 1985, *The Information Payoff*, Free Press, New York.
- Sullivan, C. H., 1985, "Systems planning in the information age", *Sloan Management Review*, inverno, pp. 3–11.
- Trauth, E. M., Cole, E., 1992, "The organizational interface: A method for supporting end users of packaged software", *MIS Quarterly*, vol. 16, n. 1, pp. 35–53.
- Venkatraman N., 1991, "IT induced business re-configuration", in Scott Morton M. S., ed., *The Corporation of the 1990s: Information Technology and Organizational Transformation*, Oxford University Press, New York, pp. 122–158.
- Wiseman, C., 1985, *Strategy and Computers*, Down Jones-Irwin, Homewood, Illinois.



Le basi di dati e i modelli di rappresentazione dei sistemi informativi aziendali

Al termine di questo capitolo sarete in grado di fare quanto segue.

1. Definire i seguenti termini chiave: sistema delle informazioni, sistema informativo, sistema d'integrazione aziendale, portafoglio IT, infrastruttura IT, IT capabilities, risorse IT, IT dynamic capabilities, portafoglio tecnico, facility management, portafoglio applicativo, architettura informativa, portafoglio tecnologico, application management, service delivery, service management.
2. Spiegare la differenza tra sistema delle informazioni e sistema informativo.
3. Identificare e spiegare, da almeno due punti di vista, gli elementi che compongono il sistema informativo aziendale.
4. Descrivere e spiegare l'importanza delle basi dati e dell'architettura informativa nel sistema informativo aziendale.
5. Individuare le principali voci di spesa connesse al sistema informativo.
6. Identificare le funzioni di supporto e operative nella gestione del sistema informativo aziendale.

Introduzione

Scopo di questo capitolo è quello di aiutare a comprendere il ruolo delle informazioni, dell'architettura informativa e delle basi dati, nell'ambito di possibili modelli di rappresentazione dei sistemi informativi aziendali.

Dopo una breve introduzione, relativa agli aspetti organizzativi e di business connessi alle tecnologie informatiche, il capitolo si soffermerà sull'analisi di due possibili modelli di rappresentazione dei sistemi informativi aziendali: il sistema informativo secondo il punto di vista del ricercatore, come sistema dinamico di dieci elementi, e il sistema informativo secondo il punto di vista di chi lavora nella funzione sistemi informativi, cioè come sistema modulare a strati.

2.1 Organizzazione, business e tecnologia

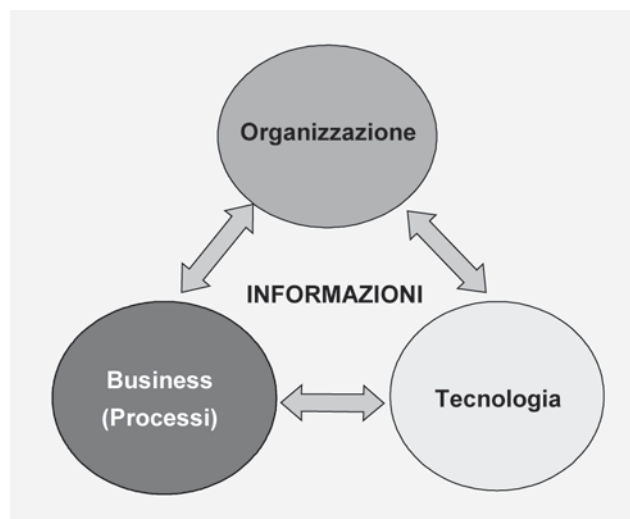
Nel recente passato era prevalsa l'idea che i business model della new economy differissero in modo sostanziale da quelli tradizionali. Il risultato di ciò è stata l'eccessiva valutazione di tutte le iniziative di business. Dopo lo scoppio della bolla speculativa delle dot.com è stato evidente che le regole per una

buona condotta sul mercato non sono cambiate. La tecnologia informatica, di per sé, non costituisce un vantaggio competitivo. Il vantaggio competitivo sta nell'uso che si è capaci di farne, disegnando processi e modi di operare in funzione delle potenzialità tecnologiche e delle informazioni che diventano via via disponibili. Le nuove sfide di mercato e il potenziale tecnologico ci spingono verso un obiettivo molto più ambizioso: l'impresa dove il flusso delle informazioni e delle transazioni sia immediato, senza impedimenti. Si tratta di un'impresa integrata, flessibile, capace di reagire rapidamente ai cambiamenti e di configurarsi nel modo migliore per cogliere le occasioni che si presentano e reagire ai rischi inaspettati.

Concretamente si tratta di un'azienda in cui tre elementi, la tecnologia (T) gli aspetti di business/processi (B) e l'organizzazione (O), sono mantenuti in continuo equilibrio grazie alla presenza di un adeguato "collante" informativo (Figura 2.1).

Figura 2.1

L'interazione tra i tre aspetti.



In primo elemento, quello di base, è la tecnologia. La tecnologia è qui intesa in senso ampio (sia tecnologia non informatica sia tecnologia informatica o sinteticamente IT). La tecnologia è lo strato che consente il concreto funzionamento dei processi produttivi. Qualora ci si riferisca anche alla tecnologia informatica, questa potrà avere un duplice ruolo: potrà essere sia tecnologia di produzione sia tecnologia di "supporto" ai processi produttivi. In questo secondo ruolo l'IT servirà per un miglior coordinamento all'interno dell'azienda, andando a svolgere quindi una funzione di supporto rispetto alle attività primarie.

I processi sono le routine operative, o le attività sequenziali, che permettono di trasformare input in output. Per rendere operativi i processi c'è bisogno, oltre che dei materiali e dei macchinari, anche delle altre infrastrutture tecniche necessarie.

L'organizzazione riguarda tutte persone che lavorano in azienda sia trasformando operativamente input in output, sia svolgendo attività amministrative e manageriali. Considerato che il lavoro è svolto in modo più efficiente quando è diviso, è necessaria una struttura organizzativa per coordinare le persone fra loro. Il coordinamento non sarà solo il risultato di una buona pianificazione organizzativa ma anche il risultato dell'agire della cultura prevalente in quella determinata azienda.

Questo lavoro si concentra sul collante informativo e sull'elemento tecnologico, con particolare riferimento alle tecnologie informatiche o IT.

Una delle principali caratteristiche dell'Information Technology (IT) è la sua pervasività. L'IT si caratterizza, infatti, non solo per essere una competenza autonoma, ma risulta essere una quota rilevante nell'ambito delle altre competenze su cui si basa il portafoglio di business dell'impresa. Ipotizzando una generica impresa con un portafoglio di business diversificato, l'IT, oltre a rappresentare una competenza autonoma a supporto dell'intera attività d'azienda (cioè oltre a occupare una buona parte dell'area della tecnologia), rappresenta una quota sostanziale delle competenze nell'ambito delle diverse competenze di business e organizzative (occupa una parte dell'area dell'organizzazione e del business/processi). Per esempio, risulterebbe quasi impossibile pensare a una diversificazione geografica del generico business "Uno" senza pensare alla realizzazione di un adeguato sistema informatico a supporto della nuova struttura organizzativa e dei nuovi processi gestionali.

L'ultimo argomento, con riferimento ai tre elementi del modello, è quello della coerenza. In particolare i tre elementi si caratterizzano per una forte interdipendenza sia reciproca (tra loro) sia nei confronti dell'ambiente generale e specifico (o transazionale di riferimento).

L'interdipendenza reciproca implica che i tre elementi debbano essere caratterizzati da una coerenza interna, o coerenza di livello 1. Ciò significa che tecnologia, business/processi e organizzazione devono caratterizzarsi per un comune livello di evoluzione e devono essere reciprocamente compatibili e funzionali. Avere dei computer o delle attrezzature molto evolute non servirà a nulla se non ci sono le persone che hanno le competenze per utilizzarle. Analogamente, è inutile sviluppare delle procedure complicatissime se non si dispone della tecnologia informatica sufficientemente evoluta per renderle operative.

In ultimo, i diversi elementi si devono caratterizzare per una coerenza esterna, ossia devono essere coerenti sia con l'ambiente generale sia con l'ambiente specifico di riferimento. L'assenza di coerenza esterna implica per l'azienda il rischio di non rispettare il vincolo dell'economicità. L'azienda che si dotasse di tecnologie informatiche ancora prodotte a livello sperimentale o di prototipo, avrebbe molte difficoltà o addirittura si troverebbe nell'impossibilità di acquistare sul mercato servizi di manutenzione, con evidenti aggravii sul conto economico e problemi operativi.

2.2 La definizione di sistema informativo

I tre elementi di base del sistema aziendale non sono isolati, ma interagiscono tra loro grazie alle informazioni (Figura 2.1) che transitano da un elemento all'altro.

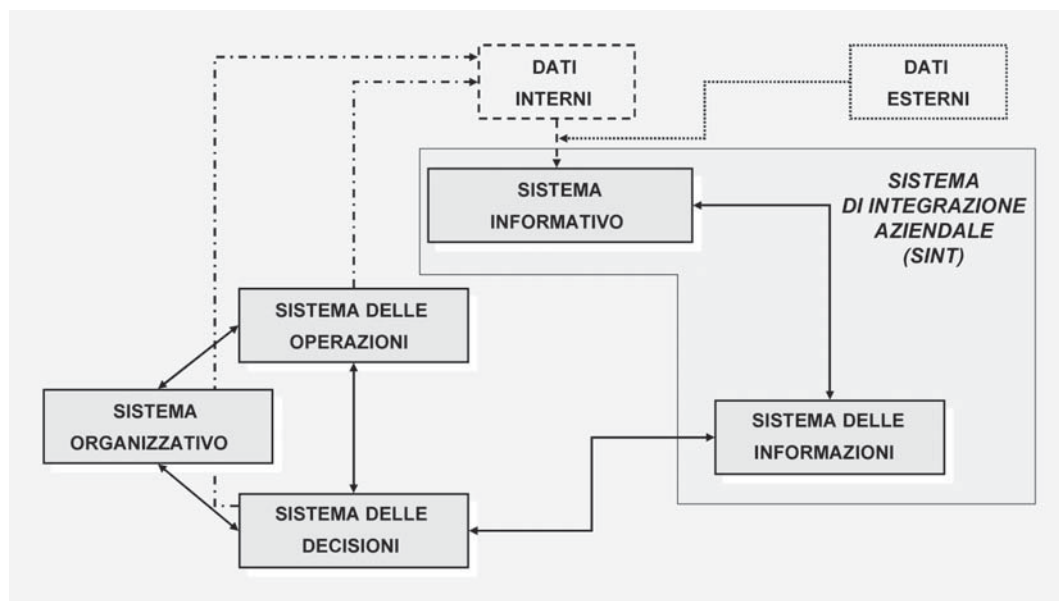
L'insieme delle informazioni che costituiscono il collante che unisce organizzazione, processi e tecnologie è definito *sistema delle informazioni*. Tale sistema è, in altre parole, un *complesso organico di informazioni che rappresenta lo stato di impresa* (Rugiadini, 1970). Il sistema delle informazioni ha l'obiettivo di *soddisfare il fabbisogno di conoscenza manifestato dal sistema delle decisioni e dal sistema delle operazioni*. Il sistema delle informazioni, si presenta, quindi, come un insieme di flussi di informazioni aventi lo scopo di rendere le decisioni più rispondenti ai vincoli di economicità.

Il sistema delle operazioni rappresenta l'attività svolta in azienda considerata da un punto di vista oggettivo. All'interno di questo sistema operano numerosi sottosistemi, quali per esempio il sottosistema delle operazioni interne e delle operazioni di scambio.

Il sistema delle decisioni ha come obiettivo il raggiungimento dei fini aziendali, definito in modo particolare per i diversi organi, nell'ambito del sistema organizzativo, cioè dell'altro sistema generale, che si occupa di analizzare l'azienda sotto l'aspetto delle persone che collaborano al suo interno. Lo scopo di tale sistema è quello di realizzare, direttamente o indirettamente, il sistema delle operazioni. In

un'ottica sistemica e ricorrendo a una rappresentazione semplificata del sistema aziendale secondo una *visione informativa* o *prospettiva informativa* che considera l'informazione come una risorsa organizzativa, e le tecnologie come il mezzo per supportare l'elaborazione di tale risorsa, questi quattro sistemi generali (in quanto si riferiscono all'azienda nel suo complesso) possono essere rappresentati come tra loro interconnessi (Figura 2.2). In dettaglio il sistema delle informazioni alimenta, ed è alimentato, dal sistema delle decisioni. Il sistema delle decisioni alimenta, ed è alimentato, dal sistema delle operazioni. Ogni operazione, infatti, per essere eseguita necessita di una decisione, non è vero il viceversa. Il sistema organizzativo mette a disposizione le persone per lo svolgimento delle operazioni. Il sistema delle decisioni e quello delle operazioni contribuiscono ad alimentare la base dati aziendale e quest'ultima, arricchita dai dati esterni, è elaborata dal sistema informativo (o sistema IT nel caso di uso prevalente della tecnologia informatica) che, quindi, provvede ad alimentare il sistema delle informazioni.

Figura 2.2
Il sistema aziendale secondo una visione "informativa".



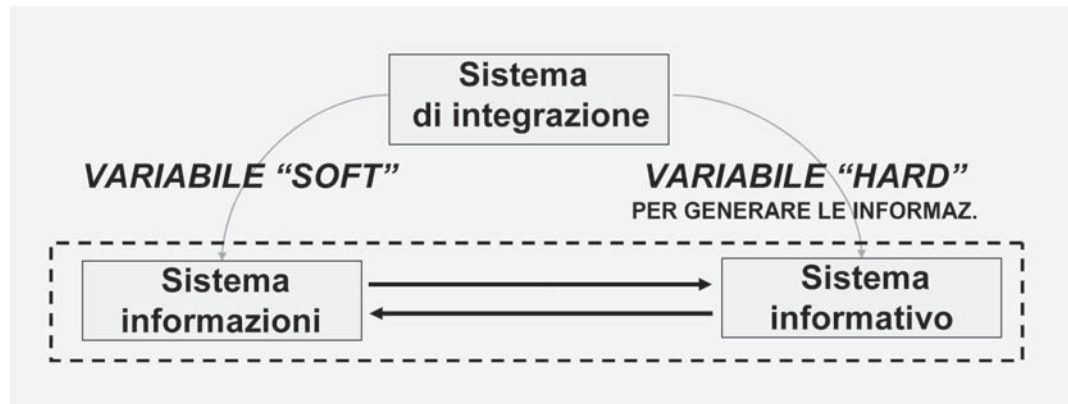
Il concetto di *sistema delle informazioni* non è da confondere con quello di *sistema informativo*. Quest'ultimo concetto costituisce il "mezzo" di produzione delle informazioni e non s'identifica con esse. Sistema informativo e sistema delle informazioni costituiscono invece quello che è definibile come "sistema di integrazione aziendale"

Il *sistema di integrazione aziendale* (Figura 2.3) è infatti definibile come un (super)sistema dinamico organizzato composto da due sottosistemi: il sistema delle informazioni e il sistema informativo (o sistema tecnologico/portafoglio tecnologico).

In base a quanto detto, quando si parla di sistema d'integrazione aziendale ci si riferisce sia al flusso delle informazioni all'interno dell'azienda, ossia al sistema delle informazioni, sia al complesso di elementi che ne hanno permesso la produzione, ossia al sistema tecnologico sottostante.

Le informazioni sono il prodotto del sistema informativo (o tecnologico) e servono a dare una rappresentazione della realtà aziendale. Tale rappresentazione dipenderà dalle logiche e dai metodi di rilevazione che si ritengono più adeguati riguardo le necessità conoscitive della realtà in questione. Le logiche di produzione e rappresentazione sono studiate e sviluppate dalle discipline economico-aziendali.

Figura 2.3
Il sistema
di integrazione
aziendale.



Quando il sistema informativo corrisponde a quello informatico si parla di *sistema informativo informatizzato* o più in generale di *portafoglio IT*, ossia dell'*insieme degli asset e delle capabilities IT posseduti dall'azienda*. Il *portafoglio IT* (o sistema informativo informatizzato) include per esempio hardware, software, apparati di telecomunicazione, apparati di archiviazione dati, device (dispositivi) di raccolta e rappresentazione dati e informazioni, persone che forniscono servizi operativi e di supporto. Il *portafoglio IT* include sia le *capabilities IT realizzate in modalità in-sourced (o modalità make)*, sia quelle *acquisite all'esterno (modalità buy)* da terze parti che svolgono attività di *outsourcing*.

2.3 Il sistema informativo informatizzato o portafoglio IT: il punto di vista del ricercatore

Il sistema informativo informatizzato o portafoglio IT, inteso come l'insieme degli asset e delle capabilities IT posseduti dall'azienda, ha lo scopo di raccogliere, elaborare e archiviare dati al fine di produrre e distribuire le informazioni necessarie, al momento e al posto giusto, ai soggetti che ne hanno bisogno per assumere decisioni e compiere azioni (adempiere a compiti), durante lo svolgimento delle loro attività aziendali.

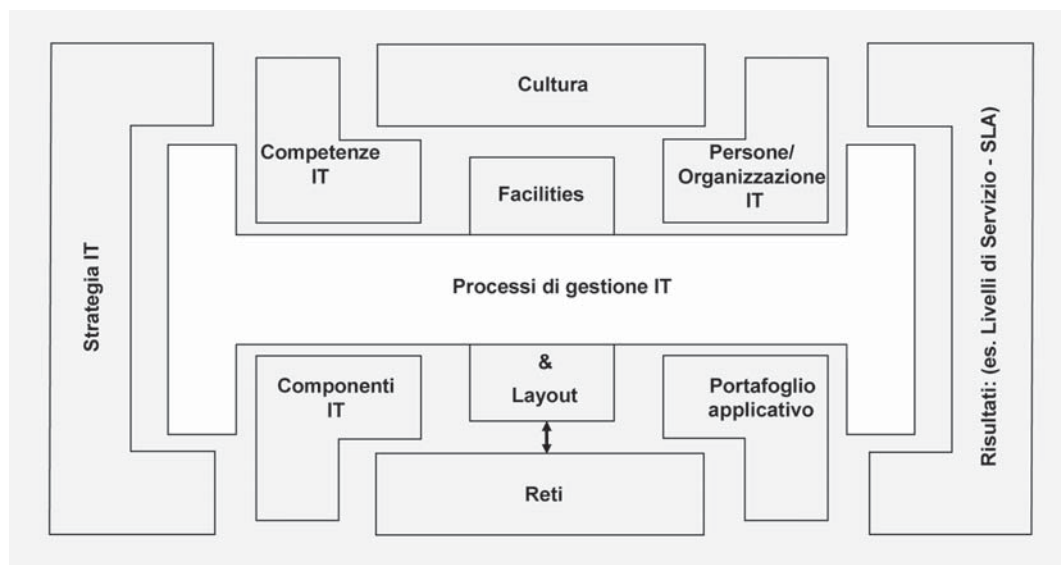
Il *portafoglio IT* è rappresentabile come un *sistema organizzato di dieci elementi* che interagiscono in modo dinamico tra loro e con l'ambiente esterno (Figura 2.4).

I dieci elementi del portafoglio IT sono sinteticamente illustrati di seguito.

Il *primo elemento*, è quello della *strategia IT*, o pianificazione delle tecnologie dell'informazione in un'azienda. Si tratta della definizione di un appropriato framework per indirizzare e guidare gli investimenti in IT e i suoi modi d'utilizzo. Fondamentalmente il processo di pianificazione dell'IT può essere diviso in tre fasi.

1. *Pianificazione strategica* vera e propria. Riguarda la definizione della strategia delle applicazioni, le strategie tecnologiche e le strategie d'organizzazione del servizio.
2. *Il dimensionamento delle risorse* necessarie alla realizzazione degli obiettivi strategici.
3. *La pianificazione e il controllo dei progetti* in cui è stata operazionalizzata la strategia.

Figura 2.4
I dieci elementi
del portafoglio IT.



Queste tre fasi riguardano due step successivi. Il primo step è quello dello *sviluppo del sistema informatico*, ossia la fase realizzativa. Il secondo step è quello dell'*esercizio del sistema*, ossia quello in cui esso è costantemente mantenuto per far sì che operi nelle condizioni di massima efficacia ed efficienza.

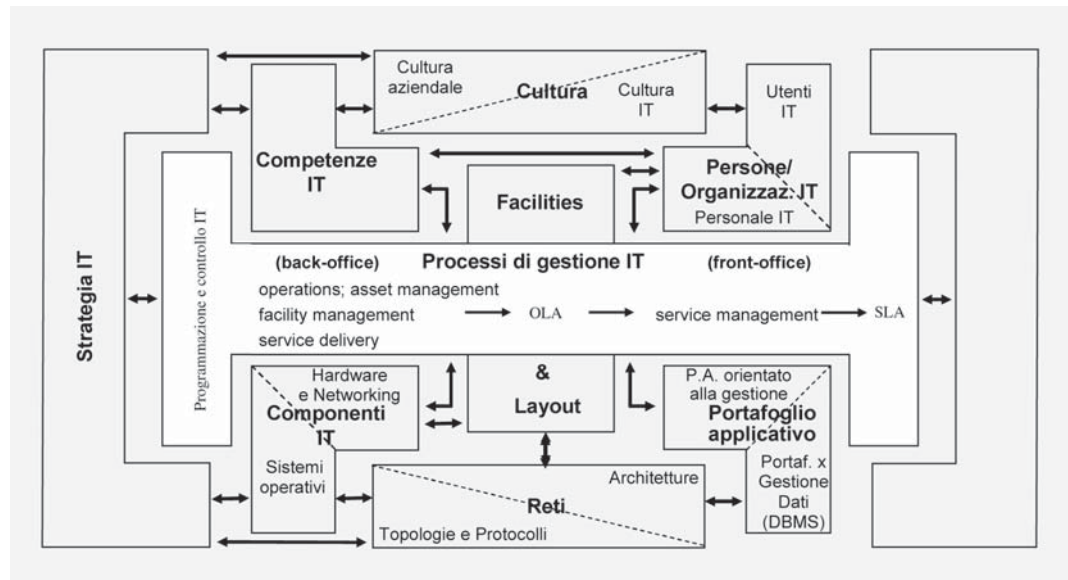
Il *secondo elemento* è quello dei *componenti IT*. Si tratta, sinteticamente, degli elementi “hard” dell’infrastruttura IT. In tale categoria rientrano l’hardware di base (computer, stampanti, scanner e così via; l’hardware di base costituisce la parte materiale o “hard” del sistema informatico e comprende tutti i mezzi informatici aventi natura fisica, cioè concreta e tangibile) e i sistemi operativi.

Il *terzo elemento* è costituito dalle *reti informatiche*. Quando l’IT si sposa con le TLC allora si parla di reti, ossia modalità di collegamento tra elementi hardware anche tra loro eterogenei. Nell’ambito delle reti si parla di *architetture* (modalità di connessione di componenti hardware) e *protocolli di comunicazioni* (linguaggi di comunicazione condivisi tra tutte le risorse hardware).

Il *quarto elemento*, quello del *portafoglio applicativo*, per analogia con il concetto di “portafoglio d’investimento”, indica l’insieme delle applicazioni software, sia gestionali sia per la gestione dei dati (basi di dati o database, Figura 2.5), presenti in azienda a supporto delle diverse competenze/processi di business. A proposito del portafoglio applicativo gestionale, è possibile distinguere tra *portafoglio applicativo centralizzato* e *portafoglio locale* a supporto delle attività specifiche dei singoli utenti. Il portafoglio applicativo centralizzato è la componente più rilevante ed è composto da un set abbastanza stabile di applicazioni tra cui per esempio applicazioni di contabilità, vendite, gestione del personale e così via. In particolare con riferimento alle imprese multibusiness, c’è la tendenza a standardizzare il loro portafoglio applicativo centralizzato con riferimento a tutti i business. A tale fine si stanno diffondendo sistemi informativi integrati o sistemi ERP (*Enterprise Resource Planning*). Si tratta di sistemi che oltre a essere delle applicazioni sono delle vere e proprie soluzioni di business che, oltre ad automatizzare le attività aziendali, favoriscono una vera e propria reingegnerizzazione dei processi.

Il *quinto elemento*, quello dei *processi di gestione IT* (o *servizi IT*), è costituito dall’insieme dei *processi* e delle *strutture* necessarie alla trasformazione degli elementi IT in “servizi IT” erogabili ai diversi utenti. Nell’ambito di tale categoria distinguiamo i *servizi di back-office* e quelli di *front-office*. I *servizi di back-office* (Figura 2.5), si riferiscono a tutti i processi e le strutture per far funzionare e mantenere costantemente efficienti gli elementi IT (operations, asset management, facility management

Figura 2.5
Le relazioni tra i dieci
elementi del portafoglio
IT.



e service delivery). I *servizi di front-office* riguardano le procedure e le strutture necessarie all'assistenza dell'utente del servizio applicativo (service management, Figura 2.5). Tra i processi di gestione IT rientrano anche quelli di gestione economica del portafoglio IT (sistemi di programmazione e controllo, di chargeout, gestione della funzione come centro di costo o di profitto); sinteticamente potremmo parlare di "amministrazione e controllo IT". I servizi di back-office operano per offrire a quelli di front-office un servizio conforme agli OLA (*Operational Level Agreements*), ossia conforme alle indicazioni ricevute dall'utente interno alla funzione sistemi informativi. I servizi di front-office operano cercando di garantire all'utente/cliente dei servizi applicativi (utente finale delle applicazioni aziendali) un livello minimo di servizio o SLA (*Service Level Agreements*).

Il *sesto elemento* è costituito dagli elementi infrastrutturali in cui sono allocate tutte le risorse IT e dalle tecniche/modalità di disposizione delle varie risorse; si parla in questo caso di *facilities & layout*.

Il *settimo, l'ottavo e il nono elemento* riguardano rispettivamente le *competenze, la cultura e l'organizzazione e il personale IT* (Figura 2.5). Si tratta degli *elementi che permettono di trasformare gli elementi IT, seguendo determinate procedure, in funzionalità per gli utenti*. Tale trasformazione è realizzata da *persone*. Si tratta, da un lato, di *personale IT* che possiede conoscenze, competenze ed esperienza per produrre servizi applicativi; dall'altro lato si tratta di *utenti IT* che possono comprendere e utilizzare i servizi applicativi. La produzione e l'utilizzo di tali servizi applicativi è notevolmente influenzato sia dalla *cultura aziendale* sia dalla specifica *cultura IT* dell'azienda. La cultura IT è l'insieme dei valori in cui l'azienda crede con riferimento all'IT. Si tratta dell'orientamento "di fondo" maturato dall'azienda nei riguardi dell'utilità e delle potenzialità dell'IT, vale a dire che cosa sia giusto fare, o meno, per disporre di un sistema informatico efficiente ed efficace a fronte delle proprie possibilità. Tali valori influenzano gli altri elementi del portafoglio IT (sistema informatico) e determinano la combinazione che essi finiscono per assumere nell'azienda in questione. Per esempio, se un'azienda ha una concezione del proprio sistema informatico molto tradizionale, prevalentemente fondata su una scarsa fiducia nell'uso dell'informatica e nell'automazione del sistema, essa sarà portata a non sperimentare tecnologie innovative. Entrambe le culture possono impattare sugli elementi del sistema informatico e dunque sul sistema informativo. Per esempio, facendo riferimento alla concezione del rischio in gene-

rale, non tutte le aziende hanno la medesima propensione a correre dei rischi. A dimostrazione di ciò, quelle meno propense al rischio tenderanno ad adottare sistemi *fault tolerant*, in pratica sistemi che alla presenza di un guasto sono dotati di risorse hardware o software in grado di neutralizzare gli effetti del guasto e di proseguire nell'erogazione del servizio senza degrado.

Il *decimo*, e ultimo, *elemento* del modello è quello dei *risultati* o *prestazioni* del portafoglio IT. Si tratta dell'insieme delle variabili che consentono di "apprezzare" il "valore del portafoglio IT in azienda". A tale proposito, per esempio, si può parlare di allineamento del portafoglio IT alle dinamiche aziendali, del livello di soddisfazione dei clienti/utenti, del grado di evoluzione e cambiamento della funzione sistemi informativi.

Tra tutti questi dieci elementi è possibile individuare una serie di relazioni (Figura 2.5). La strategia IT impatta direttamente e indirettamente sulle altre nove variabili del modello.

Osservando il modello di portafoglio IT, partendo dal basso, un primo gruppo di relazioni è dovuto al fatto che i componenti IT sono interdipendenti con reti, facilities & layout e soprattutto con i processi di gestione IT. I componenti IT e il tipo di rete (la cui topologia influenza facilities & layout) condizionano il portafoglio applicativo (per esempio, certe applicazioni per funzionare necessitano di uno specifico hardware e una determinata architettura di rete) che a sua volta impatta sulle modalità di svolgimento dei tanti processi di gestione IT (per esempio, un determinato portafoglio applicativo richiede un determinato livello di manutenzione applicativa o AM).

Osservando il modello di portafoglio applicativo dall'alto, la strategia impatta direttamente e/o indirettamente sia sulle competenze, sia sulla cultura (IT e aziendale) sia sull'organizzazione IT. Le competenze impattano sia sulla cultura (una certa cultura è figlia anche di determinate competenze) sia sull'organizzazione IT. La cultura (intesa come interazione tra cultura aziendale e cultura IT) impatta sia sulle persone IT (utenti e personale della FSI) sia sull'organizzazione della funzione sistemi informativi. Le persone IT, non foss'altro per la loro numerosità, impattano sulla variabile facilities & layout. Competenze, persone e organizzazione impattano sui modi di esecuzione dei processi IT.

I processi IT, che oltre a essere impattati da tutte le variabili sin ora esaminate, sono influenzati anche dalla strategia IT. L'output dei processi è rappresentato dai risultati o prestazioni del portafoglio applicativo.

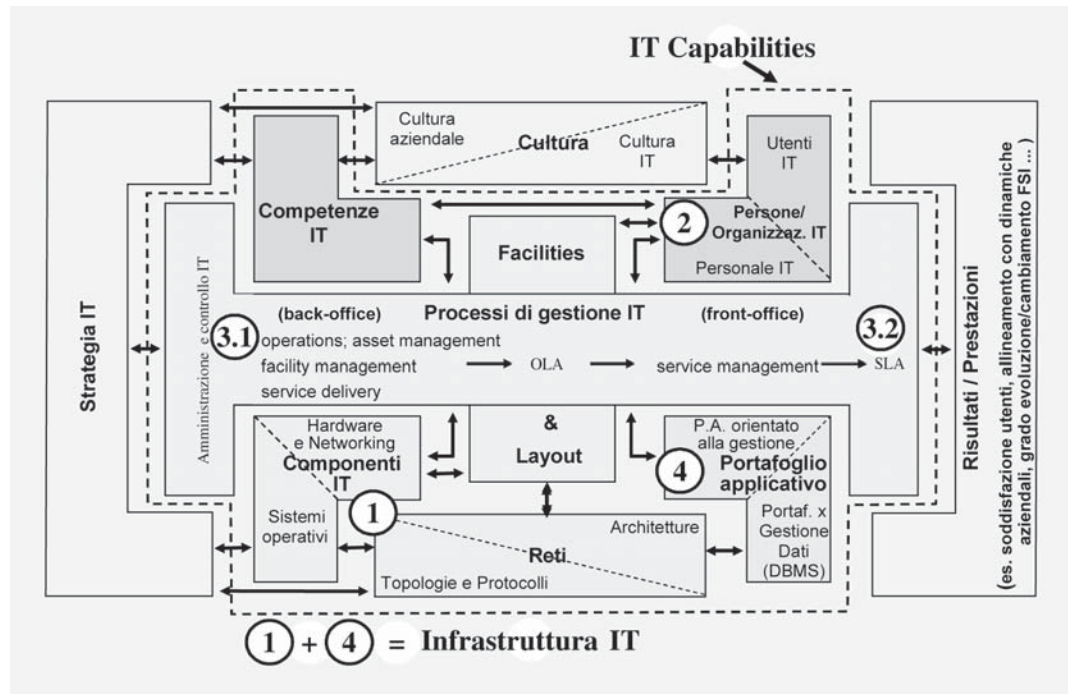
Volendo semplificare il modello, *competenze e personale* IT sono gli *elementi di base o della funzione "Sistemi Informativi"* (punto 2 della Figura 2.6). La predetta funzione si occupa dell'applicazione delle tecnologie informatiche alle attività aziendali. In particolare svolge almeno tre ruoli principali.

- Applica la tecnologia ai sistemi informativi, al fine di automatizzarli rendendoli più tempestivi, precisi ed efficienti.
- Tiene sotto controllo le nuove tecnologie informatiche, al fine di favorire l'introduzione di quelle che risultano di maggiore interesse per l'azienda, cioè quelle più efficaci per la realizzazione dei propri prodotti o servizi. Nelle aziende più evolute ci si attende che l'unità in questione sappia intravedere l'impatto dell'informatica sulle attività caratteristiche dell'azienda, cioè sul suo core business, fornendo un contributo di idee e di proposte anche con riferimento alla strategia dell'azienda.
- Studia gli effetti organizzativi dell'introduzione delle nuove tecnologie informatiche.

Nell'ambito di questi due elementi la cultura, in particolare quella IT, rappresenta una fonte di vantaggio competitivo quando è vista come elemento:

- abilitante per gestire le relazioni con i clienti;
- utilizzato per integrare, trasferire e fornire conoscenza;
- utilizzato per condividere risorse tra le diverse aree dell'impresa realizzando sinergie.

Figura 2.6
Le IT capabilities.

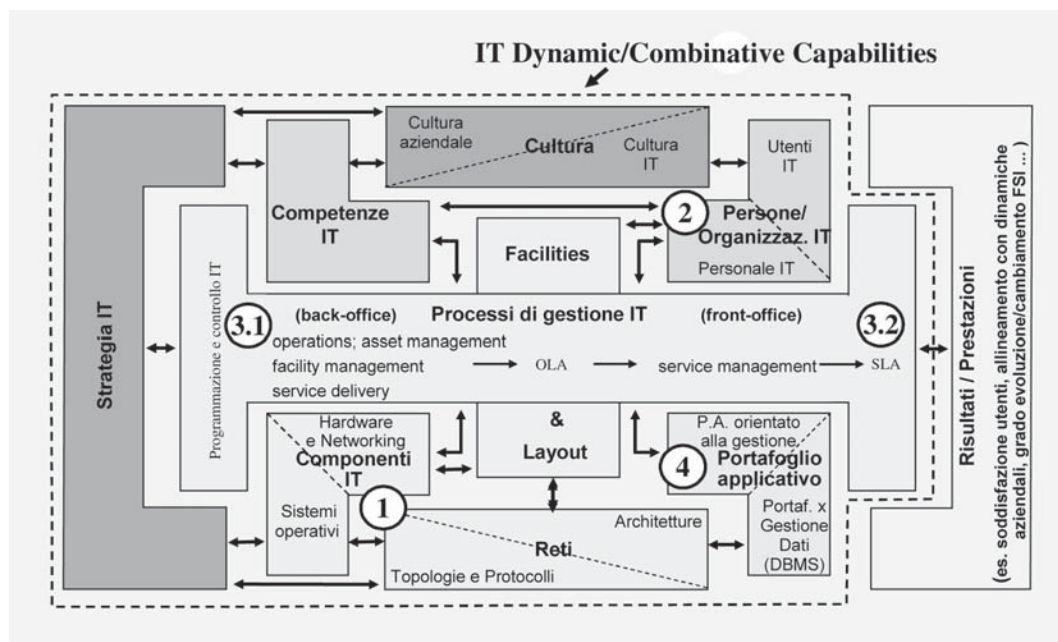


Alla base del concetto di portafoglio IT c'è quella definibile come *infrastruttura IT* o *risorse IT* (punti 1 e 4 della Figura 2.6). L'*infrastruttura IT* si compone dei tre elementi costituiti dai componenti IT, dalle reti e dal portafoglio applicativo. L'*infrastruttura IT* costituisce la componente "hard" del portafoglio IT. Per lungo tempo l'*infrastruttura IT* è stata rappresentata come una delle maggiori risorse di business e come una delle possibili fonti di vantaggio competitivo (Keen, 1991; Mc Kenney, 1995). Come Keen (1991, p. 184) fece notare, *"it is the IT platform that determines the business degrees of freedom a firm enjoys in its business plan"*. Un'*infrastruttura IT* non integrata e caratterizzata da un'elevata incompatibilità restringe le scelte di business dell'impresa. La creazione di un'*infrastruttura IT* integrata richiede in ogni caso tempo e competenze. In particolare è richiesta una continua e chiara attività di pianificazione e distribuzione delle diverse risorse così da favorire la rapida messa in esercizio delle applicazioni di business, la condivisione delle informazioni tra diverse sedi e l'implementazione di sistemi di coordinamento transazionale lungo la catena del valore dell'azienda e del settore.

Nella parte centrale troviamo i *processi IT* (Figura 2.6), ossia la parte intangibile dell'IT che è costituita da tutta la conoscenza esplicita (cioè codificata) e tacita (insita nelle persone) legata alla trasformazione di risorse IT in servizi IT.

In sintesi, l'insieme delle *risorse IT* sinteticamente definite come *infrastruttura IT*, i processi IT e gli elementi della funzione sistemi informativi costituiscono le *IT capabilities* dell'azienda. Si tratta di risorse che, in ogni caso, possono essere acquisite e/o imitate, quindi il vantaggio competitivo deriverà dalla capacità dell'impresa di combinare e utilizzare le risorse per realizzare soluzioni informatiche e implementazioni organizzative originali. In questo ultimo caso, altri due elementi entrano in gioco: la strategia IT e la cultura (IT e aziendale) cioè gli elementi abilitatori di soluzioni "firm-specific". L'inclusione di questi due elementi ci consente di parlare di *IT Dynamic Capabilities* (Figura 2.7).

Figura 2.7
IT Dynamic Capabilities.



L'analisi dell'ultimo elemento, quello dei *risultati/prestazioni* del sistema informativo o portafoglio IT, richiede il riferimento ai concetti di *coerenza interna ed esterna*, introdotti dalla teoria generale dei sistemi.

La teoria generale dei sistemi di Von Bertalanffy (1968) indica che questi devono essere ispirati al principio di *coerenza*. Essa deve intendersi sia come *coerenza esterna* di un sistema verso il suo ambiente, sia come *coerenza interna* che deve sussistere tra i diversi elementi che lo costituiscono.

La coerenza esterna fa quindi riferimento all'efficacia con cui il sistema svolge i propri compiti (confronto tra risultati attesi e risultati effettivamente raggiunti). Nel caso specifico, un sistema efficace erogherà le informazioni agli utenti interessati nei modi e nei tempi previsti.

La coerenza interna riguarda, invece, l'efficienza con cui il sistema informativo svolge i suoi compiti. Tenedo presente il principio di coerenza esterna, il sistema informativo sarà tanto più efficiente quanto minori saranno le risorse da esso utilizzate.

La coerenza interna riguarda anche il grado in cui i dieci elementi risultano tra loro complementari, equilibrati e sinergici. Se un elemento del sistema riesce a esaltare le potenzialità di un altro, si parla di *sinergia*. L'effetto sinergico si svilupperà qualora gli elementi si caratterizzino per un equilibrato livello evolutivo. Per poter essere complementari, i dieci elementi devono essere "integrabili" senza difficoltà derivanti da disallineamenti riferibili a un diverso grado di evoluzione. Così, per esempio i componenti hardware del sistema informatico dovrebbero essere coerenti con i sistemi operativi e le applicazioni selezionate. Questi tre dovrebbero essere coerenti con l'infrastruttura di rete e dovrebbero essere supportati da adeguati processi operativi e da un personale IT dotato delle opportune competenze. Il tutto dovrebbe essere coerente con la cultura IT e con la cultura aziendale nel complesso. Qualora i diversi elementi non si caratterizzino per un eguale livello d'evoluzione, per avere un sistema informatico sinergico si dovrà agire sull'elemento più debole. I risultati del sistema informatico, alla presenza di un elemento meno sviluppato, si allineeranno alle prestazioni di quest'ultimo. Così, per esempio, è inutile avere computer all'avanguardia se il personale IT non ha le competenze per farli funzionare e/o mante-

nere. Alla stessa stregua sarà inutile possedere applicazioni gestionali evolute se i componenti IT non consentono di utilizzarle così come si dovrebbe.

La ricerca dell'efficienza e dell'efficacia non è un processo da svolgere *una tantum*. Il contesto economico in cui si trovano a competere le imprese è in continua evoluzione e il progresso tecnologico mette a disposizione mezzi sempre più sofisticati che consentono possibilità precedentemente precluse.

I principi di coerenza si devono intendere in senso *dinamico*; bisogna ricercare, infatti, una situazione di costante equilibrio fra i vari elementi del sistema, non una *sintonia contingente* (gli elementi non devono essere coerenti solo in un determinato momento).

Tale equilibrio dinamico va ricercato sia nei confronti dell'ambiente sia con riferimento a tutti gli altri elementi. Il sistema informatico/portafoglio IT sarà in equilibrio se:

- eroga prestazioni corrispondenti alle aspettative dell'azienda, cioè produce le informazioni desiderate dagli utenti secondo le modalità richieste (coerenza esterna);
- i componenti del sistema sono in sintonia tra loro nel produrre i risultati, hanno cioè livelli di sviluppo concettualmente corrispondenti, che consentono l'espressione completa delle proprie potenzialità.

La coerenza interna ed esterna può essere raggiunta attraverso la definizione di un'opportuna *strategia IT*. Il tempo richiesto per realizzare nuove iniziative di business dipende solo parzialmente dalle *IT Capabilities* (ITC) ossia dalle risorse di cui si dispone, ma dipende dalle *IT Dynamic Capabilities*, ossia dalla elasticità del portafoglio IT dell'azienda. Il livello di elasticità è frutto del livello di strategicità del fattore IT nell'ambito dell'azienda e dalla cultura (IT e aziendale) prevalente. Per esempio, se una grande banca volesse realizzare un sistema di erogazione di prestiti Web-based avrebbe bisogno di utilizzare le capacità di calcolo dei propri mainframe e server, il database in cui sono contenute le informazioni sui clienti, le proprie reti di comunicazioni centrali e locali e adeguate procedure e sistemi di sicurezza. Avere dei sistemi *scalabili* e *integrabili* ossia pensati, in anticipo (strategia IT) per evolvere verso situazioni in cui è richiesta maggiore capacità (scalabili), o in cui è richiesta la capacità di “dialogare” con altri sistemi (integrabili), sicuramente facilita e *velocizza* la concretizzazione della “business idea” e ne *riduce i costi* d'avviamento. La questione diventerebbe molto più complessa e difficile, qualora, per esempio, i sistemi e le procedure di sicurezza della banca non fossero stati inizialmente progettati per supportare l'utilizzo d'applicazioni Web-based dirette all'integrazione tra i dati del cliente che richiede il prestito e il suo rating di credito. Nel caso in cui la struttura del sistema di sicurezza della banca non fosse stata progettata per essere “aperta” a future evoluzioni del business, l'avvio della nuova iniziativa dovrebbe essere ritardato fino al momento dell'avvenuto adeguamento dell'infrastruttura di sicurezza.

2.4 Il sistema informativo o portafoglio IT: il punto di vista interno

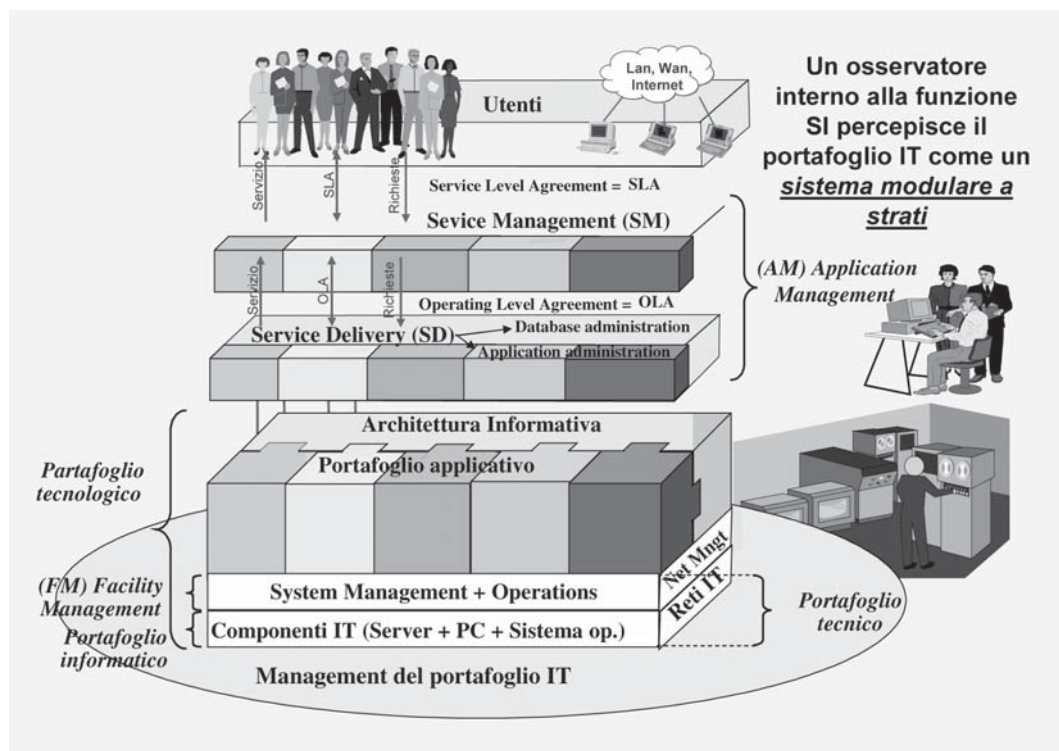
Il secondo punto d'osservazione del portafoglio IT è quello di colui che fa parte della funzione sistemi informativi. Tale osservatore percepisce il *portafoglio IT come un sistema modulare a strati*.

La “modularità” del portafoglio IT si concretizza nel fatto che i sottosistemi di cui è composto non devono sempre esistere ed essere identici, ma è possibile modificare il contenuto di un sottosistema e l'altro esisterà ugualmente (salvo incompatibilità tecnologiche), *la loro relazione è a livello di servizi forniti*.

Il concetto di “strato” si riferisce alla presenza di numerosi “livelli” per la produzione ed erogazione di servizi IT (Figura 2.8).

In particolare, oltre allo strato o utente, s'individuano altri tre strati fondamentali, alcuni dei quali a loro volta scomponibili in sub-strati e/o moduli.

Figura 2.8
Il portafoglio IT
dal punto di vista
interno alla FSI.



Il concetto di modularità e di scomponibilità

Il concetto di *modularità* è diverso da quello di *scomponibilità* di Simon (1962). La scomponibilità è intesa come decomposizione di un sistema complesso in un ordine strutturato di insiemi successivi di sottosistemi caratterizzati da interazioni deboli. La modularità è riferibile alla standardizzazione dei “punti di collegamento” (interfacce qualora ci si trovi nell’ambito del portafoglio applicativo) ossia alla relativa indipendenza dei diversi moduli o strati a prescindere dal loro contenuto.

- *Management del portafoglio IT.*
- *Portafoglio tecnologico.*
- *Application management.*

I diversi livelli non sono percepiti come indipendenti, ma come strettamente inter dipendenti (Figure 2.8 e 2.10). In particolare il livello degli *utenti* risulta inter dipendente con quello dell’*application management (AM)* e quest’ultimo con quello del *facility management (FM)*. Gli utenti finali e intermedi (utenti interni) dei servizi applicativi richiederanno, rispettivamente, a chi eroga/gestisce le applicazioni, un determinato livello di servizio (Figure 2.8 e 2.9) o SLA (*Service Level Agreement*)/OLA (*Operational Level Agreement*). Lo SLA è definibile come *l’accordo (evidentemente interno) che definisce il livello di servizio, e dunque la capacità di funzionamento delle applicazioni, da fornire agli utenti finali dei servizi applicativi.*

Figura 2.9
SLA e OLA.

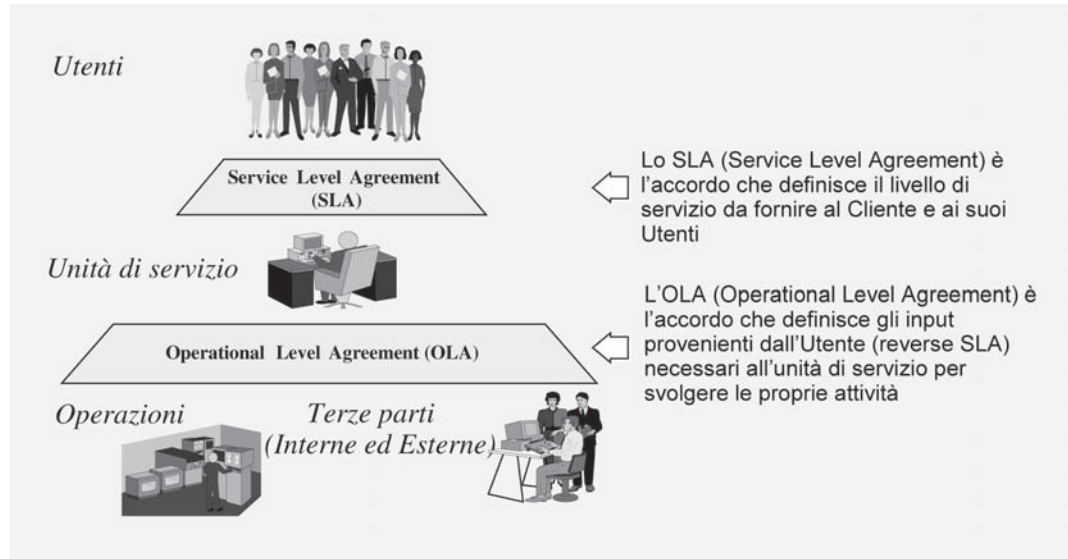
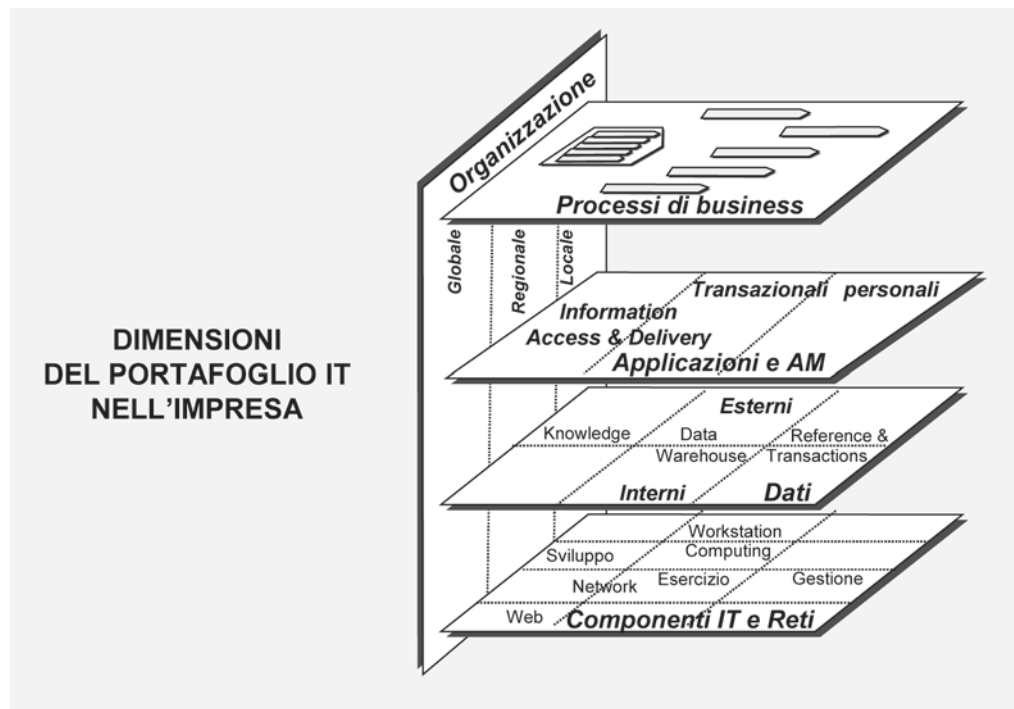


Figura 2.10
Il portafoglio IT come una serie di strati sotto i processi di business.



A sua volta, l'unità di servizio che gestisce le applicazioni richiede che l'infrastruttura hardware funzioni secondo determinati standard. In sostanza, chiede che sia rispettato un certo OLA (Operational Level Agreement), cioè livello di servizio operativo, vale a dire il livello di funzionalità minimo garantito dell'infrastruttura hardware.

A sua volta nell'ambito dell'impresa il portafoglio IT è visto come una serie di strati sottostanti i processi di business di cui diventa un "abilitatore" (Figura 2.10).

2.4.1 Il portafoglio tecnologico

Il portafoglio tecnologico è lo strato fondamentale per la produzione dei servizi IT; senza software e senza computer funzionanti non si potrebbe pensare a nessun servizio.

Il portafoglio tecnologico è definibile come l'insieme dell'hardware (in senso allargato) e del software idoneo a essere utilizzato per supportare lo svolgimento delle attività aziendali.

Il portafoglio tecnologico è scomponibile nei seguenti strati.

- 1. Portafoglio tecnico.
- 2. Portafoglio applicativo.
- 3. Architettura informativa.

2.4.2 Il portafoglio tecnico

Il portafoglio tecnico (Figura 2.11) è costituito dall'insieme degli asset e dei servizi riferibili alle componenti hardware del portafoglio IT. Fondamentalmente il portafoglio tecnico è composto dal portafoglio informatico e dall'attività di facility management.

Il portafoglio informatico non equivale al concetto di computer, bensì è qualcosa di ben più vasto. Si riferisce, infatti, a una rete di computer idonea a supportare, nel tempo, le applicazioni e l'architettura informativa aziendale. Gli elementi del portafoglio informatico sono quattro (Figura 2.12): l'hardware, il software di base, il networking e le architetture tramite cui i primi tre elementi interagiscono.

L'hardware costituisce la parte materiale ("hard") del sistema informatico e comprende tutti gli strumenti tecnici/informatici aventi natura fisica, cioè concreta e tangibile (computer, stampanti, mouse).

Figura 2.11
Il portafoglio tecnico.

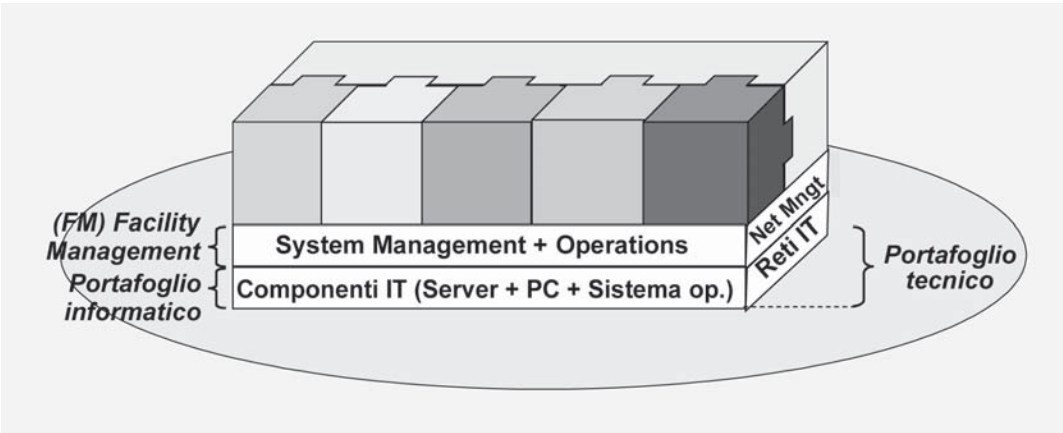
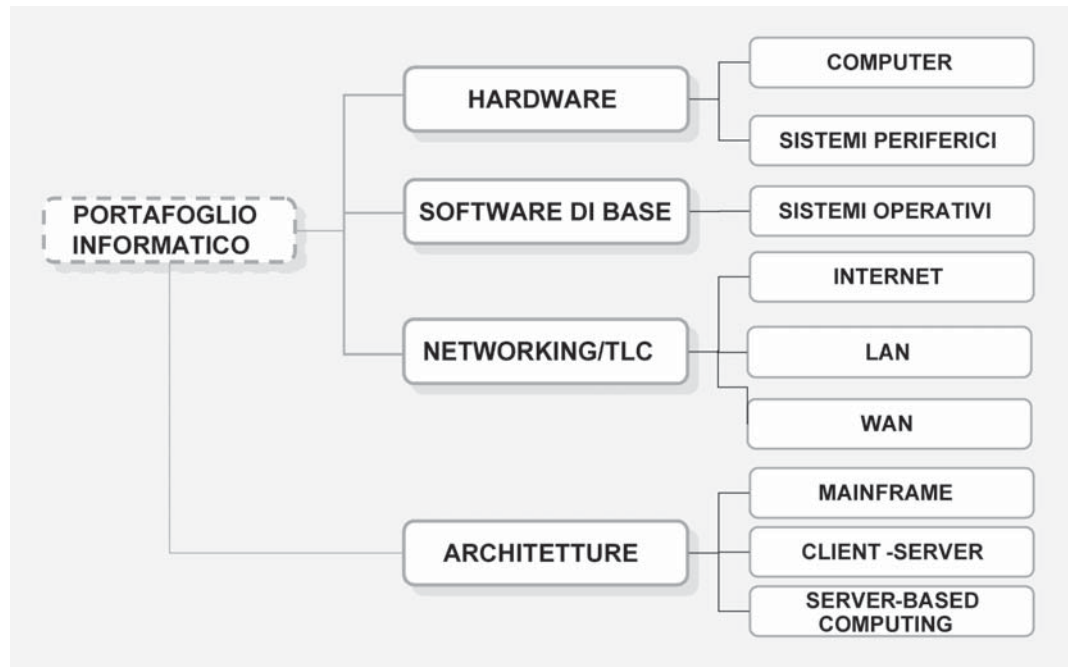


Figura 2.12
Il portafoglio
informatico.



Il *software di base* è la categoria di software che *interagisce in modo diretto con l'hardware*. È quello che *permette al computer di avviarsi e di svolgere alcune funzioni di controllo* (per esempio, Windows, Unix e Linux per le macchine piccole; AS400 e MSV per le macchine grandi). In sostanza tale categoria di software permette il controllo di tutto l'hardware.

Una delle caratteristiche dell'informazione è quella di essere al momento giusto al posto giusto: è necessario che l'IT si sposi con le TLC. Si parla di ICT e di reti o networking.

Il *networking* è costituito, quindi, da tutte le componenti informatiche (tra cui ricordiamo hub e router) che permettono di connettere “fisicamente” tutte le diverse componenti hardware.

Non è sufficiente che tutto l'hardware sia connesso, ma è necessario che le componenti possano effettivamente dialogare interagendo a livello software. Le diverse architetture rappresentano quindi i diversi linguaggi attraverso cui le varie componenti hardware si combinano, si correlano e si presentano agli utenti.

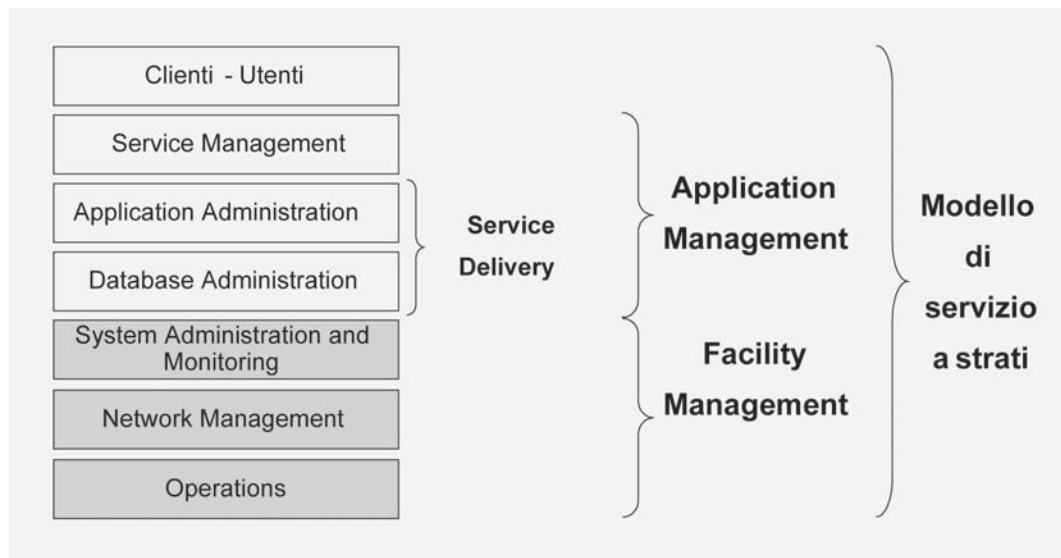
Il *portafoglio informatico per essere sempre in condizioni d'efficienza necessita di essere mantenuto, gestito e adeguato. È necessario, inoltre, eseguire una serie di operazioni per la gestione e la messa in sicurezza del data center*. Queste attività rientrano nel campo del *facility management* (Figura 2.13).

Il *facility management* comprende le attività di:

1. *system administration and monitoring*;
2. *network management*;
3. *operations* (questa attività talvolta non viene fatta rientrare nell'ambito del facility management e viene considerata isolatamente).

Figura 2.13

Il modello di servizio a strati.



Il livello definito *operations* comprende una serie di sottoattività tutte connesse alla gestione di tutti i servizi relativi al data center, al management, al monitoring e alla sicurezza sia del contesto ambientale sia dell'hardware. Nell'ambito dell'attività di messa in sicurezza dell'hardware, una particolare attenzione è riservata alla sicurezza fisica del sistema, alla gestione delle copie di sicurezza (compreso l'off-site storage, ossia la gestione delle copie in luogo diverso dall'origine) e alle attività di backup e restore anche con schedulazione di tipo batch.

Il livello di *system administration and monitoring* comprende le sottoattività che permettono di rendere funzionante e mantenere in efficienza l'hardware del sistema informatico. Tra queste attività ricordiamo le seguenti.

- Setup e configurazione dell'hardware, ossia tutte le attività per l'avvio e il funzionamento dell'attrezzatura informatica.
- Setup e configurazione del sistema operativo (UNIX/NT) dei prodotti per strati aggiuntivi (layered product, per esempio Novell).
- Upgrade del software esistente a mezzo dell'upgrade con patch, per esempio tramite thc.
- Amministrazione del sistema hardware e supervisione del sistema operativo.
- Monitoraggio della capacità e delle performance del sistema.
- Setup e manutenzione del software di monitoraggio e controllo del sistema.
- Attività di manutenzione e programmazione dell'attività.
- Monitoraggio del sistema e degli eventi 24x7.
- Supporto del sistema sia on-site sia remoto.
- Management delle relazioni con le terze parti (i vendor).
- Supporto telefonico 24x7.

Lo strato di *network management* comprende le *sottoattività che permettono di rendere funzionante e mantenere in efficienza le componenti di networking del sistema informatico*. Tra queste subattività rientrano le seguenti.

- Installazione e configurazione delle componenti di networking (per LAN, WAN e Internet).
- Setup e manutenzione del software di monitoraggio e controllo della rete.
- Monitoraggio della capacità e delle performance della rete.
- Assistenza e manutenzione dei componenti di rete.
- Setup, aggiornamento e controllo delle policy di sicurezza della rete.
- Creazione della necessaria elasticità, sia di banda sia di componenti di rete.
- Management delle relazioni con le terze parti (i vendor).
- Supporto telefonico 24×7.

2.4.3 Il portafoglio applicativo

Il portafoglio applicativo è definibile come l'insieme di applicazioni software di cui gli utenti dispongono per lo svolgimento delle diverse attività aziendali.

Scopo del portafoglio IT è quello di raccogliere, elaborare, archiviare dati per produrre e distribuire le informazioni necessarie, al momento e posto giusto, ai soggetti che ne hanno bisogno, per assumere decisioni e compiere azioni (adempiere a compiti), durante lo svolgimento delle loro attività aziendali.

Diversi studiosi hanno preso in esame le attività aziendali cercando di classificarle. Secondo lo schema di Antony (1965), in base al livello gerarchico delle persone nella struttura organizzativa, sono individuabili tre livelli di attività aziendali: attività di tipo strategico (di pianificazione e controllo strategico), tattico (di programmazione e controllo) e operativo o transazionale, cioè di svolgimento delle attività correnti. Nella realtà le attività non rientrano in maniera netta in una sola categoria. Ogni tipo di attività genera le proprie esigenze informative e queste a loro volta condizionano la natura e la tipologia delle informazioni (il supporto informativo) che le devono soddisfare.

Secondo Simon (1979) le esigenze informative dipendono dal livello di programmabilità delle decisioni relative ai compiti da svolgere. In base a tale caratteristica si configurano tre tipi di attività differenti.

- *Processi di tipo fisico produttivi e distributivi*, cioè i processi che generano le transazioni aziendali (per esempio la spedizione di una cassa da un magazzino a un altro). Si tratta di attività che sono normalmente certificate dalla creazione di un documento, cartaceo o elettronico che ne testimonia l'avvenuta esecuzione.
- *Processi decisionali programmati*, orientati alla gestione dei processi di tipo fisico e distributivo.
- *Processi decisionali non programmati*, rivolti alla riprogettazione dei processi di tipo fisico e distributivo.

Ogni persona, a qualunque livello, è chiamata a svolgere un mix di compiti relativi ad attività programmabili e non. Anche secondo Simon ogni tipo d'attività genera le proprie esigenze informative, e queste a loro volta condizionano il supporto informativo che le deve soddisfare.

In ogni azienda dovrebbe, dunque, esistere un portafoglio applicativo composto da diverse applicazioni in relazioni alla classe di attività da supportare (Figura 2.14). Il termine *portafoglio applicativo*, introdotto da Nolan (Nolan e Croson, 1995) per analogia con il concetto finanziario di “portafoglio d’investimento”, indica *l’insieme delle applicazioni software (software gestionale) presenti in azienda*. L’analisi di portafoglio ha quindi lo scopo di caratterizzare le opportunità d’investimento e utilizzo di queste applicazioni nei diversi processi aziendali.

Figura 2.14

Gli strati del portafoglio applicativo.



A una prima approssimazione, in base a quanto sin qui detto, potremmo dire che il portafoglio applicativo di un’azienda generica può essere suddiviso in almeno tre insiemi o “strati” di applicazioni principali (Figura 2.14).

1. *Applicazioni per il supporto alle decisioni*, che comprendono le applicazioni software a supporto dei cicli di pianificazione strategica.
2. *Applicazioni per il controllo direzionale*, che comprendono le applicazioni software per il controllo delle risorse aziendali secondo gli schemi del controllo di gestione.
3. *Applicazioni operative*, che comprendono le applicazioni software per lo svolgimento sia dei processi di supporto (contabilità, amministrazione risorse umane) sia quelle per i cosiddetti processi primari.

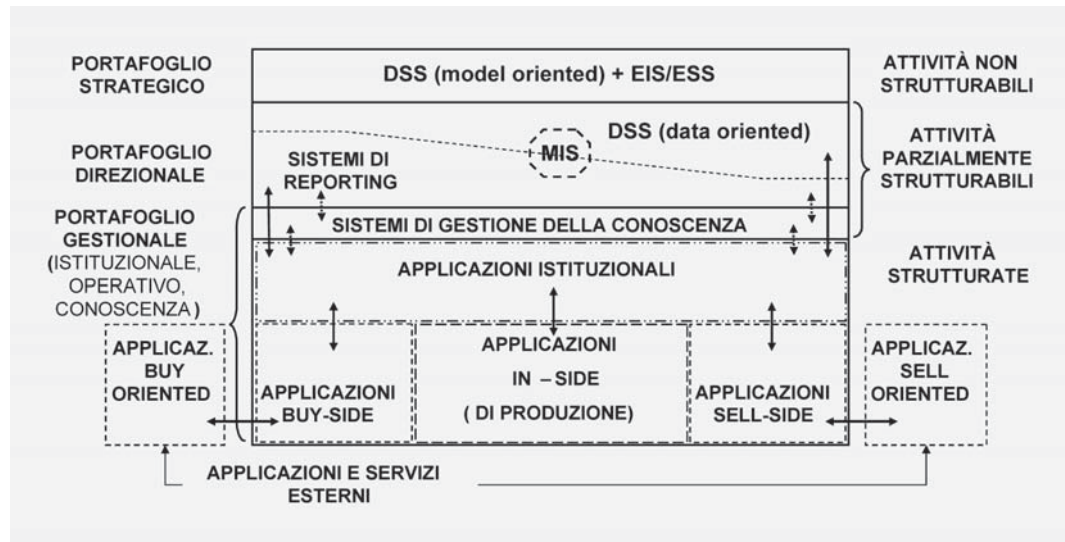
Nell’ambito di questo modello logico è possibile fare rientrare un quarto strato applicativo che normalmente si riscontra nelle aziende. Si tratta dello strato applicativo deputato alla *gestione della conoscenza aziendale*. Si tratta di uno strato che si potrebbe collocare tra quello operativo e quelli strategici. Le applicazioni che consentono di proceduralizzare e standardizzare la gestione della conoscenza aziendale, pur essendo di supporto alle attività operative, hanno anche e soprattutto una valenza strategica e da queste considerazioni emerge la dignità di una loro autonoma collocazione nell’ambito del panorama applicativo.

Nella realtà dei fatti le aziende non procedono alla creazione di più portafogli applicativi ma ragionano in termini di un unico portafoglio applicativo (Figura 2.15) che al suo interno può essere segmentato in più strati, precisamente:

- *portafoglio strategico;*
- *portafoglio direzionale;*
- *sistemi di gestione della conoscenza e lavoro d'ufficio;*
- *portafoglio gestionale.*

Figura 2.15

Il modello di rappresentazione del portafoglio applicativo.



Il *portafoglio gestionale* può essere a sua volta considerato composto dal *portafoglio istituzionale* e da quello *operativo*.

Il *portafoglio strategico*, quello *direzionale*, quello *istituzionale* e quello per la *gestione della conoscenza* sono definiti *orizzontali*, in considerazione della loro relativa indipendenza dalle specifiche caratteristiche dei settori d'attività. Il *portafoglio operativo* è invece definito *verticale*, poiché rispecchia le caratteristiche tipiche del settore. Ogni settore è infatti caratterizzato da un proprio portafoglio operativo; sono tuttavia ravvisabili delle limitate sovrapposizioni per alcuni settori. Nell'ambito del portafoglio strategico troviamo due principali tipologie di applicazioni software: i *Decision Support System* o *DSS*, prevalentemente *model oriented*, e gli *Executive Support System* o *ESS*.

I *DSS model oriented* sono dei software applicativi basati su algoritmi proposti dai metodi quantitativi per la descrizione di fenomeni il cui andamento è rilevante per il decisore, o intesi a riprodurre gli effetti del processo decisionale medesimo consentendo a chi deve prendere la decisione di valutarne le conseguenze. Sono disponibili sul mercato anche dei *DSS* basati su tecniche d'intelligenza artificiale; si tratta di sistemi meglio conosciuti come *Knowledge Based System* se orientati alla creazione di conoscenze, e di *Expert System* se tendono a emulare il comportamento umano. Un esempio di *DSS* è il sistema di pianificazione dei viaggi di un'impresa che offre servizi di trasporto dei metalli prevalentemente-

mente a servizio delle diverse imprese del gruppo, gruppo attivo nel settore dei metalli, che la possiede. L'impresa è proprietaria di diversi cargo e, a ogni richiesta di carico, deve rispondere a diverse richieste di trasporto. Il DSS in questo caso consisterà in un modello capace di calcolare i dati economici e tecnici relativi la richiesta. I dati economici includeranno il carburante, il lavoro, l'ammortamento del cargo, le spese di attracco e così via. I dati tecnici includeranno la capacità del cargo, la velocità, la distanza, i dati di consumo, l'attuale distanza dal luogo di carico e così via. Le domande a cui i manager dell'azienda hanno bisogno di rispondere sono, per esempio, "Dati i vincoli derivanti dalle esigenze del cliente in termini di date di consegna e di tariffa offerta, quale cargo dovrebbe essere assegnato alla richiesta 'x' per massimizzare il profitto?", "Qual è la velocità ottimale a cui un determinato cargo dovrebbe viaggiare per ottimizzare i profitti aziendali e rispettare la data di consegna?". Si tratta di sistemi che richiedono grandi capacità di calcolo, ma che sono idonei a girare anche su sistemi desktop sufficientemente potenti. Sono sistemi sviluppati anche per facilitare l'inserimento dei dati e la comprensione dei risultati.

Con le sigle *EIS* (*Executive Information System*), o *ESS* (*Executive Support System*), ci si riferisce agli applicativi software a supporto dell'attività strategica e, dunque, capaci di supportare nell'indirizzare problemi non strutturabili. Tali sistemi *non si identificano con una soluzione applicativa* ma piuttosto con un *ambiente di lavoro capace di supportare questo tipo di decisioni*. Si tratta di software progettati per incorporare dati esterni come vincoli imposti da norma di legge, azioni e dati dei concorrenti, così come dati forniti dai sistemi di reporting e dai DSS. Si tratta di sistemi capaci di selezionare i dati critici per affrontare un determinato problema fornendo le informazioni il più possibile sintetiche e utili ai manager dell'impresa. In tali sistemi sono presenti *funzionalità grafiche avanzate* con possibilità di definire diversi tipi di grafici da un'unica fonte dati.

Gli applicativi EIS/ESS supportano il management aziendale nel dare delle risposte il più possibili razionali, per esempio, alle seguenti domande: "In quale business dovremmo operare?", "Quale tipo di acquisizioni dovremmo fare per stabilizzare le nostre performance?", "Quale divisione o società dovremmo cedere per disporre della liquidità necessaria al nostro piano di acquisizioni?". Considerato che si tratta di sistemi ideati a uso e consumo del top management aziendale, solitamente poco orientato all'utilizzo diretto di strumenti informativi, sono dotati di interfacce grafiche molto usabili.

Il *portafoglio direzionale* comprende la serie di applicazioni, sinteticamente identificabili con la sigla *MIS* o *Management Information System*, orientate a supportare le decisioni manageriali o tattiche. Essi consistono prevalentemente in due principali categorie applicative: *sistemi di reporting* e *DSS data oriented*.

I *DSS data oriented* sono sistemi basati sull'interrogazione dei diversi database aziendali da parte di un decisore che si propone di reperire input utili per il proprio processo decisionale. Si tratta fondamentalmente di *motori di calcolo e di data mining*, basati su specifiche tecnologie evolute per la gestione dei dati, primariamente *data warehouse* e *OLAP* (*On Line Analytical Processing*). Il *data warehouse*, concetto nato negli anni '90, ha definito una famiglia di basi di dati distinte da quelle dedicate all'elaborazioni delle transazioni operative. Estendendo la tecnologia delle basi di dati relazionali, i *data warehouse* forniscono lo strumento pratico per concentrare masse grandissime di informazioni e per strutturarle secondo le esigenze di consultazione del management. La tecnologia OLAP, presente sin dagli anni '80, permette, invece, una segmentazione multipla delle informazioni, evitando costose e lentissime interrogazioni. La disponibilità di tecnologie di *data warehouse* e OLAP, oltre alle sofisticate tecnologie di analisi dei dati (*data mining*) hanno promosso lo sviluppo di una nuova serie di sistemi informatici direzionali per esempio di marketing, finalizzati prevalentemente all'analisi del cliente. Si tratta di sistemi che hanno l'obiettivo di gestire un rapporto uno a uno con il consumatore, anche e soprattutto nel caso di aziende con un numero particolarmente elevato di clienti, come le compagnie telefoniche, della luce, del gas e costituiscono potentissimi strumenti di controllo delle prestazioni critiche verso il cliente.

I *sistemi di reporting* possono essere considerati l'operazionalizzazione informatica della teoria della *Balanced Scorecard (BSC)* di Kaplan e Norton (1996), che integra i concetti di KPI (*Key Performance Indicators*) e di controllo di gestione. Si tratta in sostanza di suite software che permettono di ottenere *report predefiniti e a differente livello di analisi per il controllo sistematico dei fattori competitivi dei processi gestionali*.

Il *portafoglio operativo* è costituito dall'*insieme delle applicazioni software utilizzate nell'ambito dei processi aziendali, sia primari sia di supporto*.

L'*insieme delle applicazioni utilizzate all'interno dei processi di supporto* è il cosiddetto *portafoglio istituzionale* mentre le singole applicazioni sono definite *applicazioni istituzionali*. Si tratta di applicazioni che eseguono adempimenti di legge (contabilità, retribuzioni previdenza) o supportano l'amministrazione di infrastrutture (immobili) o fattori di produzione (scorte, personale, denaro, impianti). Si tratta di attività tutte orizzontali, vale a dire poco variabili rispetto al settore industriale in cui l'azienda opera. Il *portafoglio istituzionale* ha da sempre rappresentato un'area molto informatizzabile, sia per i grandi volumi sia per la standardizzazione, la ripetitività e la periodicità delle singole operazioni. Sin dai primi anni '70 sono presenti sul mercato i primi pacchetti di software applicativo per l'informatizzazione delle retribuzioni, della fatturazione, delle procedure contabili, creando isole di automazione localmente efficienti. L'integrazione fra queste sotto-aree automatizzate restava tuttavia limitata: per esempio, le anagrafiche dei fornitori della contabilità e degli acquisti erano separate, con conseguenti incoerenze e potenziali inefficienze. Il passaggio dei dati da un'applicazione a un'altra avveniva, infatti, con procedure di interfaccia di tipo batch (cioè non in tempo reale), da realizzare su misura, cioè, non incluse nei pacchetti di mercato, e da mantenere nel tempo. L'integrazione inizia realmente negli anni '90 con i sistemi ERP, che hanno adottato un unico schema di riferimento per la gestione amministrativa e delle risorse umane. I sistemi ERP moderni superano anche il problema della diversità di legislazione attraverso la cosiddetta localizzazione, un programmato piano di parametrizzazione e adattamento che incorpora nel pacchetto le norme delle diverse regioni (per esempio, le diversità dei trattamenti previdenziali UE verso USA) e nazioni (per esempio, le diversità delle norme contabili e fiscali fra paesi diversi della UE).

I *benefici dell'applicazione delle TI nel portafoglio istituzionale* derivano in primo luogo dalla riduzione, spesso notevole, dei costi d'elaborazione, poiché l'elaboratore, diventando quasi un impiegato elettronico, sostituisce parzialmente il lavoro manuale a un costo inferiore. In secondo luogo i benefici derivano dalle superiori capacità di elaborazione delle informazioni dei sistemi informatici, che permettono di elaborare a costi bassi una maggiore gamma d'informazioni e quindi di rendere più efficaci i processi di pianificazione controllo e ottimizzazione. Tali vantaggi sono tangibili, per esempio, nelle applicazioni di gestione delle risorse umane, dove divengono fattibili la pianificazione della carriera, la programmazione degli utilizzi, il bilanciamento delle politiche di promozione, incentivazione e assunzione proprio grazie alle superiori capacità elaborative dei sistemi informatici. Analoghi benefici sono ottenibili nella gestione degli impianti, degli immobili e delle risorse finanziarie.

Il *portafoglio gestionale* comprende anche quello definito *portafoglio operativo o transazionale*. Tale concetto comprende le applicazioni gestionali utilizzate dai processi primari della catena del valore (Porter, 1979) ed è specifico di ciascun settore industriale o *industry* (industria manifatturiera: meccanica, alimentare; banche; assicurazioni; utilità: aziende telefoniche, elettriche, gas, acqua e pubblica amministrazione).

Scopo generale dell'informatizzazione dei processi operativi è l'integrazione dei processi sia orizzontale (vendite con produzione, produzione con fornitori) sia verticale (per esempio, la pianificazione delle operazioni con il controllo di avanzamento delle operazioni). Un sistema di gestione degli ordini clienti deve, per esempio, verificare la disponibilità a magazzino dei prodotti, scambiando informazioni con le applicazioni che elaborano le movimentazioni di magazzino e le transazioni di gestione dei ma-

teriali. Questa integrazione fra flusso degli ordini, dei materiali e delle operazioni è indispensabile, per esempio, per la tracciabilità degli ordini via Internet da parte dei clienti. Tale tracciabilità ha anche un impatto strategico, considerato che si è dimostrato essere un fattore competitivo per ottenere un rapporto di fiducia con la clientela (Cronin, 1996; Davenport e Klahr, 1998). L'integrazione ha trasformato negli anni '90 il modello di business delle aziende, che hanno iniziato a produrre su specifica del cliente, a operare con scorte ridotte o nulle e a diminuire drasticamente le durate dei processi eliminando i tempi morti. Si è così gradualmente superata la logica parziale e limitata dei preesistenti sistemi che informatizzavano in modo non integrato parti dei processi, seguendo spesso una logica funzionale, ovvero supportando e ottimizzando le attività delle funzioni organizzative piuttosto che dei processi. Con queste isole di automazione; infatti, il guadagno d'efficienza delle operazioni locali è annullato dalla perdita di efficienza ed efficacia del processo globale (Hammere Stanton, 1999, Scheere Habermann, 2000).

Fondamentalmente le applicazioni software riferibili al portafoglio operativo o transazionale possono raggrupparsi in tre grandi categorie: *applicazioni in-side*, *applicazioni buy-side* e *applicazioni sell-side*.

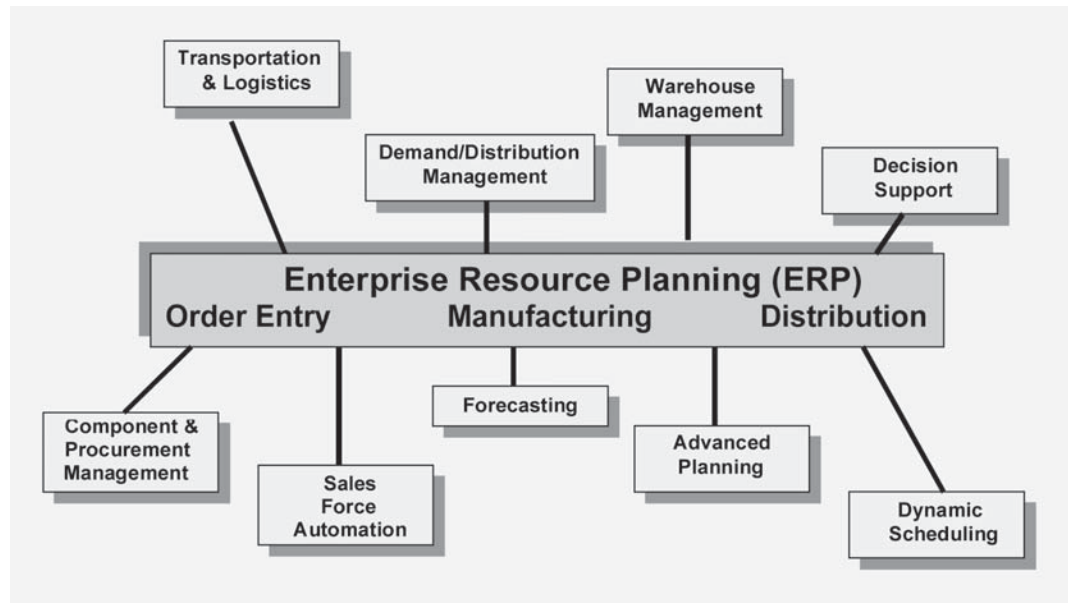
Le *applicazioni in-side* sono quelle incentrate sull'automazione delle attività produttive, per esempio nel caso delle imprese manifatturiere quelle riferibili all'automazione di fabbrica, che include l'automazione fisica delle operazioni e l'informatica a supporto della pianificazione delle operazioni. Si parla per esempio di CIM (*Computer Integrated Manufacturing*), che non corrisponde in realtà a un pacchetto, ma rappresenta uno schema di riferimento concettuale per lo sviluppo di un approccio unitario alla gestione della produzione che è precondizione per l'implementazione di pacchetti MRP (*Manufacturing Resource Planning*) ossia dei pacchetti finalizzati alla produzione e gestione dei materiali. Rientrano in questa categoria anche i pacchetti applicativi di automazione della progettazione come il CAD (*Computer Aided Design*) e le numerose tipologie di automazione della fabbricazione e dei processi di trasformazione.

Le *applicazioni buy-side* sono quelle che informatizzano i processi di acquisto delle imprese. Tra queste sicuramente rientrano le applicazioni di *e-procurement*. Si tratta di sistemi per la creazione di mercati elettronici interaziendali, che favoriscono l'incontro di domanda e offerta. Si realizza una focalizzazione del personale addetto agli acquisti sulla definizione delle condizioni contrattuali e sulla selezione dei fornitori potenziali, mentre il reparto utente svolge direttamente e in tempo reale il processo di acquisto, con un rilevante abbattimento dei tempi e dei costi della transazione.

Le *applicazioni sell-side* riguardano l'automazione dei processi orientati al cliente. Per esempio, la crescente attenzione al cliente ha promosso lo sviluppo dei sistemi CRM (*Customer Relationship Management*), che sono finalizzati ad abbattere i costi di transazione fra azienda e cliente. I sistemi CRM, nati come soluzioni su misura (*Custom Software*) sono, per esempio, nei due casi Cisco e Amazon, divenuti nel tempo vere e proprie soluzioni integrate. Nel caso delle aziende manifatturiere, per esempio, i sistemi CRM supportano, in parziale sovrapposizione con altri pacchetti, i processi di vendita via Web, i call-center, gli agenti e alcuni processi di assistenza post-vendita.

Oltre alle applicazioni riferibili specificatamente a una delle aree prima individuate, esistono e sono disponibili sul mercato delle "suite applicative" basate sul concetto di modularità e composte da applicazioni riferibili sia al portafoglio istituzionale sia a quello operativo o transazionale. Si parla di applicazioni ERP (*Enterprise Resource Planning*), cioè software gestionali sviluppati per integrare tutti i processi aziendali, da quelli primari a quelli di supporto. Gli ERP (Figura 2.16) sono software, progettati per automatizzare le "routine organizzative" (Nelson e Winter, 1982) e per modellizzare i processi dell'impresa, dalla finanza alle vendite, con l'obiettivo di realizzare l'integrazione delle informazioni possedute all'interno dell'impresa e la condivisione delle stesse tra l'impresa e i suoi partner. Si tratta di sistemi che, grazie all'integrazione nativa tra i diversi moduli applicativi, riducono la complessità organizzativa e quindi abbassano i costi di transazione interni o costi di coordinamento.

Figura 2.16
Rappresentazione
logica di un sistema
ERP.



La rilevanza, l'importanza e, dunque, l'attrattività del portafoglio applicativo e dell'IT in generale cambiano da settore a settore. I contenuti informativi dei prodotti e dei processi, infatti, sono variabili. Per esempio, sono molto alti nelle banche e molto bassi nelle manifatture di tabacco. Ciò in considerazione della standardizzazione, ripetitività, volumi e semplicità delle attività.

Nelle aziende manifatturiere, le applicazioni supportano l'esecuzione dei processi primari, la pianificazione e il controllo del ciclo di trasformazione e della logistica in entrata e in uscita e le attività a monte della produzione.

Nei servizi, la tecnologia è spesso parte del servizio stesso e il suo controllo diventa strategico. Con riferimento alla tecnologia Internet, siamo in presenza di un'innovazione IT che oltre a essere un vero e proprio canale di vendita e distribuzione (come nel caso di Amazon) innova profondamente il portafoglio applicativo disponibile sul mercato, richiedendo la sua "riscrittura" per adeguarlo alla tecnologia Internet.

In certi settori, come per esempio nelle telecomunicazioni la tecnologia informatica non solo è parte del servizio, ma diviene tecnologia di produzione vera e propria, diventando il vero motore di cambiamento del modello di business delle TLC sempre più orientato a essere un unico processo di elaborazione delle informazioni. La digitalizzazione della voce, della TV e l'arrivo della tecnologia UMTS stanno definitivamente infrangendo la barriera fra informatica di gestione dei contenuti e informatica di supporto ai processi.

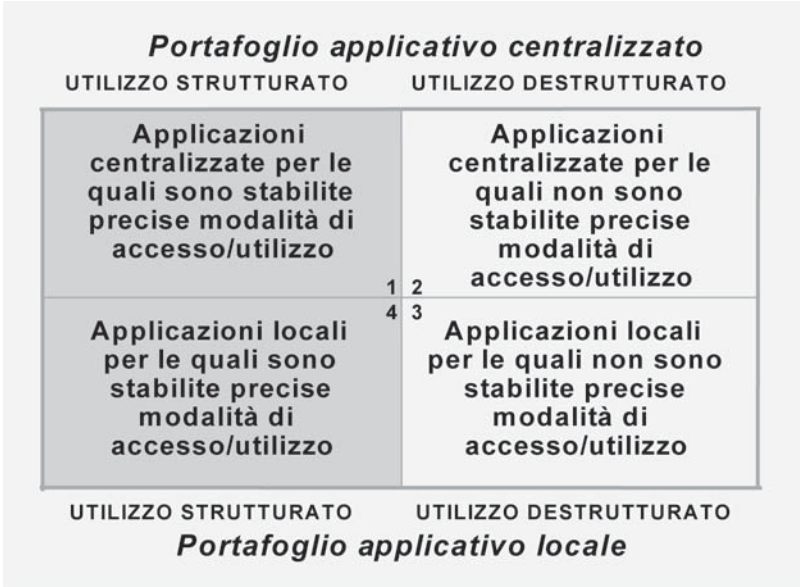
Il *portafoglio applicativo a supporto della conoscenza e dell'attività d'ufficio (KOWS)*, infine, comprende gli applicativi diretti alla gestione della conoscenza aziendale (supporta i cosiddetti "knowledge workers") e quelli orientati all'aumento della produttività individuale (supporta i cosiddetti "data workers", ossia persone di livello culturale medio orientate allo svolgimento di attività operative); in quest'ultimo caso si parla di applicativi di "office automation", quali elaboratori di testi, fogli elettronici o database di piccola potenza. Nel caso di applicazioni per la gestione della conoscenza, ci si riferisce ad applicazioni, come per esempio Lotus Notes, o a taluni database per esempio per la gestione delle immagini, utilizzati dai "knowledge workers" (persone che generalmente hanno un diploma di laurea,

il cui lavoro consiste nuove informazioni e conoscenza) per la strutturazione, gestione e accesso alla conoscenza creata.

Le *applicazioni sell e buy oriented* sono costituite dai servizi applicativi offerti da aziende esterne (chiamate *e-firm*, cioè imprese che utilizzano strutturalmente l'IT come tecnologia di produzione) a supporto e/o rafforzamento delle capacità applicative interne.

Un'ultima distinzione del portafoglio applicativo è fattibile in relazione al fatto che debba rispondere a necessità informative definite unitariamente dall'azienda oppure individualmente e soggettivamente dalle persone che considerano le informazioni. Nel primo caso ci troveremo di fronte a un *portafoglio applicativo centralizzato*, nel secondo caso a uno *locale* (Figura 2.17), ossia installato sui sistemi informatici direttamente posseduti dall'utente. Le applicazioni si caratterizzeranno, inoltre, per la presenza o meno di differenti modalità di utilizzo e accesso. Le applicazioni locali si caratterizzeranno per un uso più destrutturato, mentre quelle centrali per un uso più strutturato (Figura 2.18). L'utilizzo delle applicazioni centralizzate prevarrà sia nell'ambito dello svolgimento delle attività operative sia di quelle strategiche; l'utilizzo strutturato sarà massimo nello svolgimento delle attività operative, mentre sarà minimo nell'ambito dello svolgimento delle attività strategiche.

Figura 2.17
Caratteristiche del portafoglio applicativo.



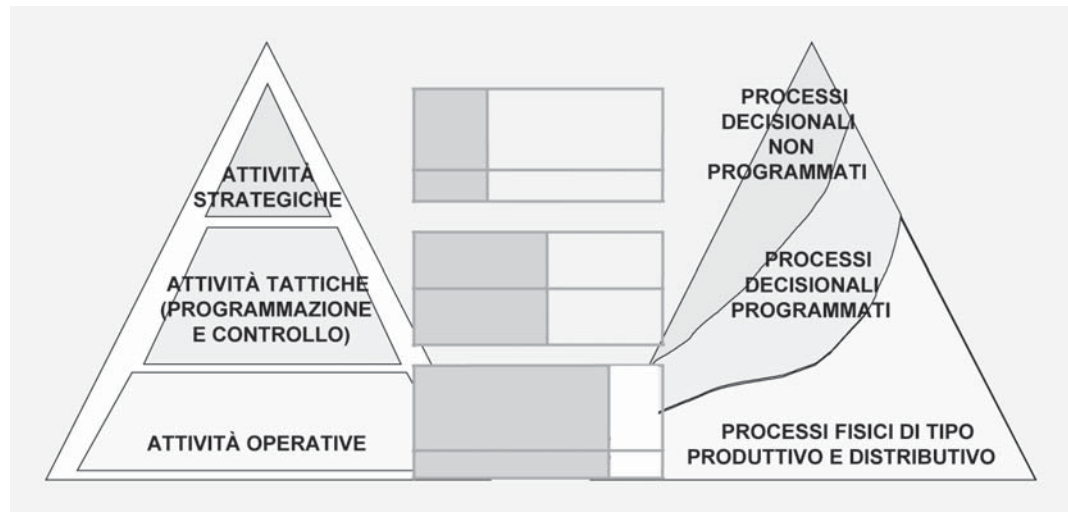
2.4.4 L'architettura informativa

L'*architettura informativa* (Figura 2.19) definisce i *dati rilevanti* e la loro *architettura* (database design), le *strategie e metodologie di proprietà, distribuzione e accesso*, in relazione sia al *tipo di applicazione*, sia al *soggetto cui afferisce l'applicazione* (centrale o locale), sia all'*organizzazione* dell'azienda e sia al *tipo di dato* (storico, di supporto alle decisioni, operative e statiche).

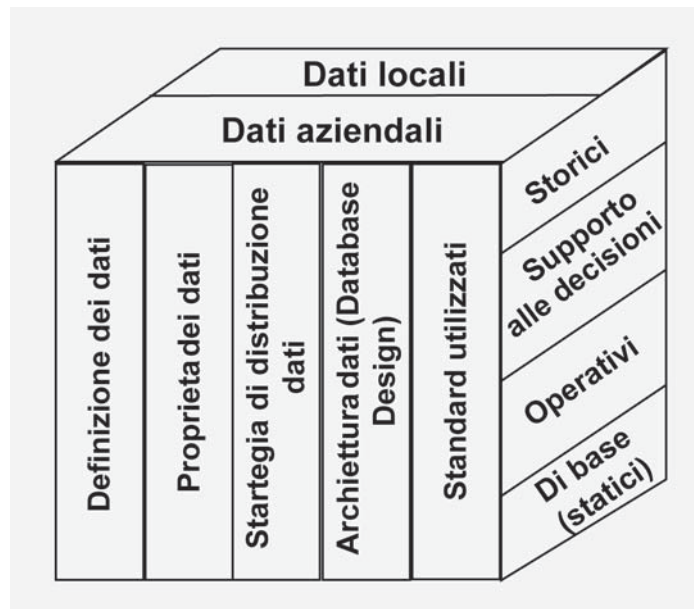
L'architettura informativa può essere vista in relazione a *quattro differenti livelli* (Figura 2.20): livello dati, livello database, livello applicativo e livello di presentazione.

Figura 2.18

Caratteristiche del portafoglio applicativo in base all'attività.

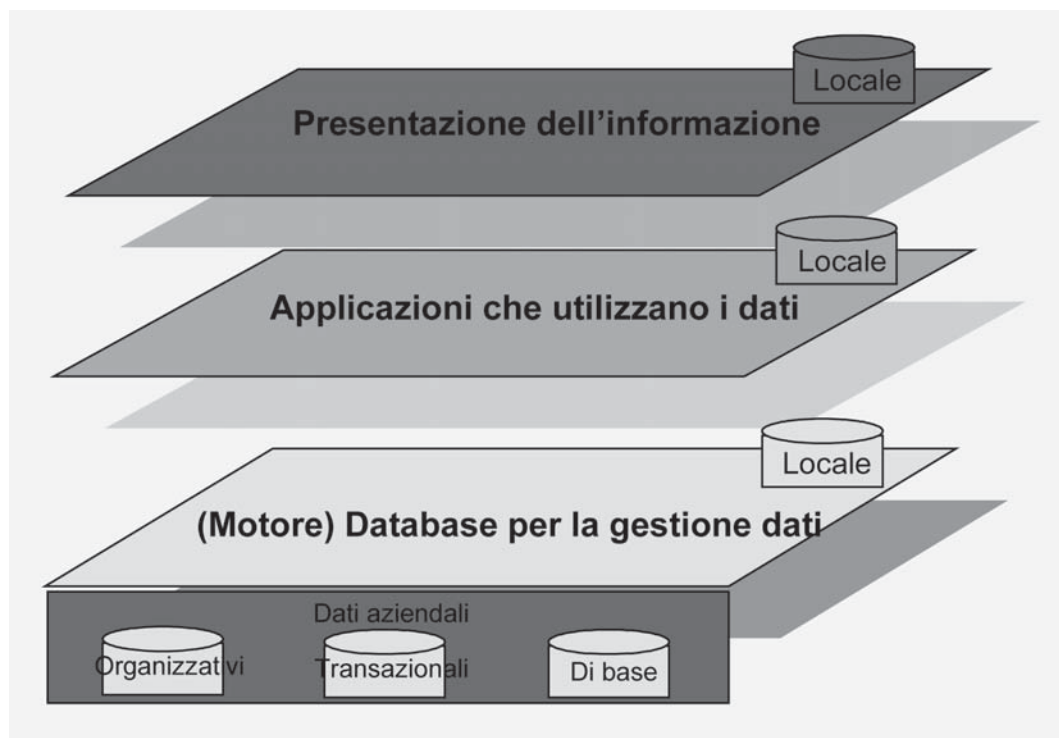
**Figura 2.19**

L'architettura informativa.



Il *livello dati* è quello fisico e riguarda sostanzialmente l'architettura dei dati, ossia come i dati sono relazionati tra loro. A questo livello è rilevante anche la qualità, la stabilità, l'affidabilità e l'integrità dei dati. Il *livello database* riguarda il "motore" utilizzato per la gestione dei dati, in sostanza il tipo, la potenza, la scalabilità e l'apertura della base di dati utilizzata per gestire i dati aziendali. I programmi database permettono sia di integrare dati da fonti eterogenee sia di separare gli strati di utilizzo e archiviazione dei dati.

Figura 2.20
Livelli dell'architettura
informativa.



Il *livello applicativo* è quello che concerne le applicazioni che utilizzano i dati inoltrando opportune “richieste” al livello database.

Il *livello di presentazione* riguarda le modalità e i supporti con i cui i dati e le informazioni sono presentati.

2.4.5 Service delivery

L'attività di *service delivery* riguarda le attività necessarie a garantire l'erogazione di un servizio applicativo efficace ed efficiente agli utenti del portafoglio tecnologico. In sostanza due sono le attività che si riferiscono a quella di service delivery: *database management* e *application management*.

L'attività di *database management (DAM)* comprende tutte le attività per il *setup*, la *configurazione*, la *gestione* e la *manutenzione dell'applicazione per la base dati aziendale*. Fondamentalmente l'attività di database management comprende le seguenti attività.

- Setup e configurazione del database.
- Amministrazione e manutenzione del database.
- Monitoraggio della capacità e delle performance del database.
- Gestione del software di monitoraggio del database.
- Management delle relazioni con le terze parti (i vendor).
- Supporto telefonico 24x7.

L'attività di *application management (AM)* include le attività di *setup, manutenzione ed evoluzione delle applicazioni software e tutto quanto necessario ad assicurare che queste rimangano permanentemente allineate con il business durante tutto il loro ciclo di vita*. Il servizio di application management deve operare nel rispetto del principio di “miglioramento continuo” mediante il quale si devono ricercare e porre in essere le azioni più opportune per migliorare nel tempo il livello e la qualità del servizio applicativo.

Le attività previste per l'application management sono le seguenti.

- Supporto funzionale all'utente.
- Training e testing.
- Manutenzione correttiva (bug fix).
- Manutenzione adattiva.
- Manutenzione evolutiva.
- Configurazione e gestione delle release.
- Distribuzione delle nuove release.
- Help desk di secondo livello.
- Supporto richieste saltuarie.
- Gestione del servizio (pianificazione, controllo e reporting).

Per ognuna di queste attività sono solitamente definiti degli indicatori di misurazione dei livelli di servizio (KPI, *Key Performance Indicator*).

2.4.6 Service management

L'attività di *service management* comprende tutte le attività svolte a supporto dell'utente per facilitare e favorire l'utilizzo e la piena operatività delle applicazioni. Tra le attività di service management, oltre all'help desk di primo livello, ossia al supporto utente circa le funzionalità e l'uso dell'applicazione, sono da comprendere le seguenti attività.

- Escalation management, cioè l'attività di indirizzamento e gestione delle richieste di assistenza più complesse.
- Management dei componenti IT, in particolare quelli distribuiti ossia utilizzati dagli utenti dei servizi applicativi.

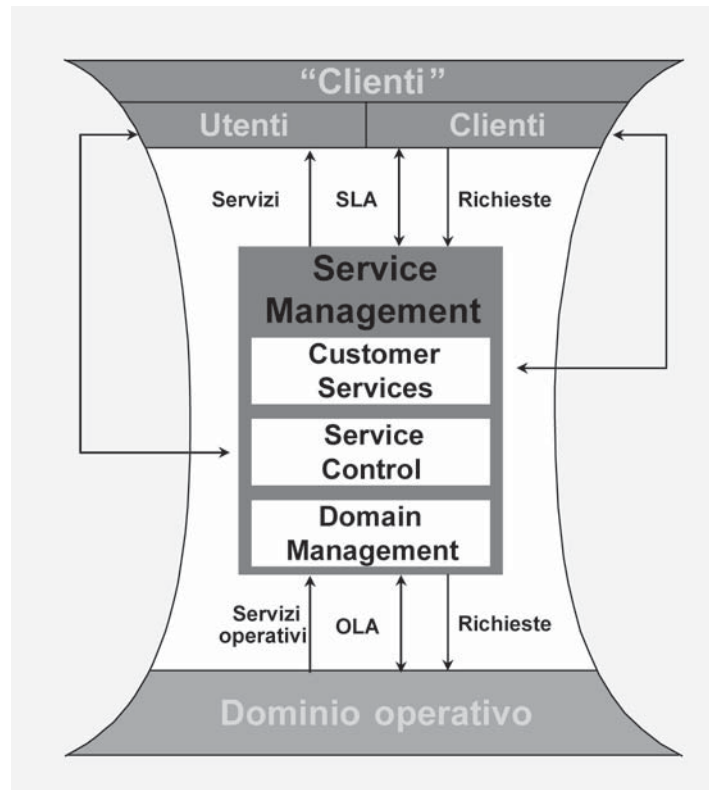
L'attività di help desk di primo livello è solitamente arricchita da un servizio di gestione dei problemi con specializzazione funzionale e da un supporto 24x7. Allo scopo di monitorare con attenzione questo tipo di attività solitamente sono previsti dei reporting sul livello di servizio offerto.

Volendo sintetizzare è possibile schematizzare il *modello di servizio* come nella Figura 2.21. In sostanza risultano evidenti tre aree.

1. *L'area degli utenti o “clienti”* in senso ampio del servizio. Tale area si caratterizza per una chiara segmentazione dei “clienti” e delle loro relative esigenze.
2. *L'area del service management*. Si tratta dell'interfaccia unica verso “clienti” e fornitori di risorse. Tale area è:

Figura 2.21

Modello di service management.



- responsabile per la gestione del servizio e interfaccia verso “clienti”;
 - link tra “service” e “resource” management. Si focalizza sull’operatività giornaliera;
 - si focalizza sulla gestione delle risorse e sugli aspetti operativi.
3. Il *dominio operativo* in cui rientrano le attività di produzione vera e propria dei servizi applicativi: application management, facility management e operation management.

Si tratta di un modello basato su una serie di principi, in particolare è possibile individuarne almeno quattro:

- unico punto di contatto con il “cliente”;
- gestione dell’operatività giornaliera separata dalla introduzione e dalla modifica dei servizi;
- chiara distinzione tra i costi fissi connessi al servizio e quelli variabili connessi ai singoli progetti;
- chiara definizione degli SLA (*Service Level Agreements*) e dei livelli di operatività definiti negli OLA (*Operational Level Agreements*).

Il modello, basato sui principi suddetti, permette di conseguire una serie di benefici, e in particolare:

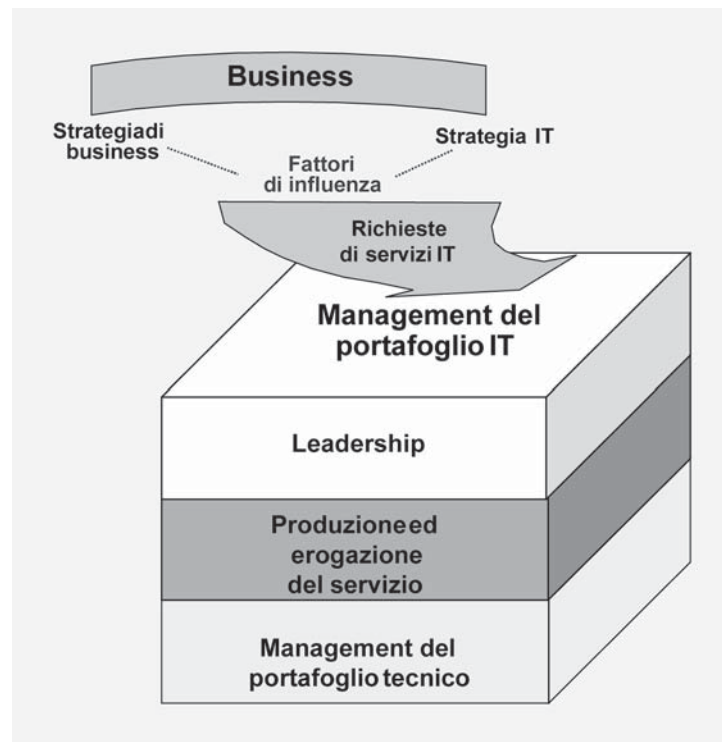
- crea una relazione diretta con i “clienti”;

- permette di concentrarsi sui processi a valore aggiunto;
- definisce chiaramente le responsabilità sul livello di operatività grazie alla definizione degli OLA;
- gli SLA migliorano la comprensione del valore di business dei servizi IT e gli OLA permettono di gestire i fornitori esterni e interni;
- risulta possibile introdurre nuovi servizi senza impatti sul livello di operatività normale.

2.4.7 Management del portafoglio IT

L'attività di *management del portafoglio IT* (o funzione sistema informativo) è, in base a quanto detto, divisa in “strati di servizio” così da individuare e soddisfare le richieste IT derivanti dalla strategia di business (Figura 2.22).

Figura 2.22
Strati di servizio.



Il livello *leadership* è costituito dai manager della funzione SI. Si tratta di coloro che lavorano con i manager dell'area business per identificare e schedare iniziative IT, definire e raggiungere la qualità del servizio e abilitare nuove iniziative di business fornendo consulenza IT.

Il livello di *produzione ed erogazione del servizio* riguarda tutte le persone coinvolte nelle attività di service delivery e service management, ossia delle persone coinvolte nell'attività di produzione ed erogazione dei servizi applicativi.

Il livello di *management del portafoglio tecnico* riguarda tutte le persone coinvolte nella gestione dell’hardware, del software di base, delle reti nell’attività di supporto tecnico e nei servizi operativi.

Ogni livello risulta composto da quattro aree principali (Figura 2.23): capabilities, sourcing, processi e strategia IT/governance.

Con il termine *capabilities* ci riferiamo alle risorse umane con competenze e conoscenze adeguate, alle misure dell’impatto di business e ai criteri per la gestione delle performance del portafoglio tecnico e per la produzione ed erogazione del servizio.

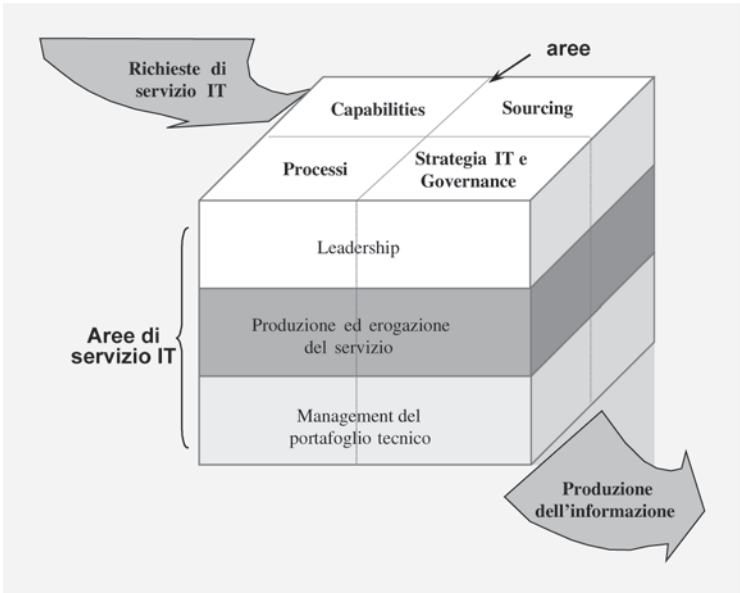
L’area di *sourcing* riguarda la valutazione e definizione d’adeguate opzioni di acquisizione e gestione delle capabilities IT necessarie.

A ogni livello sono presenti specifici *processi*, ossia individuare e utilizzare le migliori pratiche per definire la sequenza delle attività sia di IT management sia operative.

L’ultima area presente è quella relativa alla *Strategia IT e Governance* ossia la definizione delle linee guida per le relazioni interne (compresa la definizione di ruoli e responsabilità) ed esterne per assicurare l’impatto desiderato e l’allineamento dei processi.

Nella Figura 2.24 riportiamo un organigramma di una società generica costituita per la gestione del portafoglio IT di un gruppo di imprese. Risultano evidenti sia i *tre strati* prima evidenziati sia la suddivisione funzionale in relazione al modello di rappresentazione del sistema informativo secondo una visione del manager interno alla funzione.

Figura 2.23
Aree dello strato di servizio.



2.5 La catena del valore dell’IT

Tutto quanto detto sino a questo punto può essere anche rappresentato secondo la classica rappresentazione strategica: la catena del valore (Figura 2.25). Si tratta della visione che considera il sistema informativo o portafoglio IT come un insieme di processi operativi e di supporto.

Figura 2.24
Esempio di organigramma di funzione sistemi informativi.

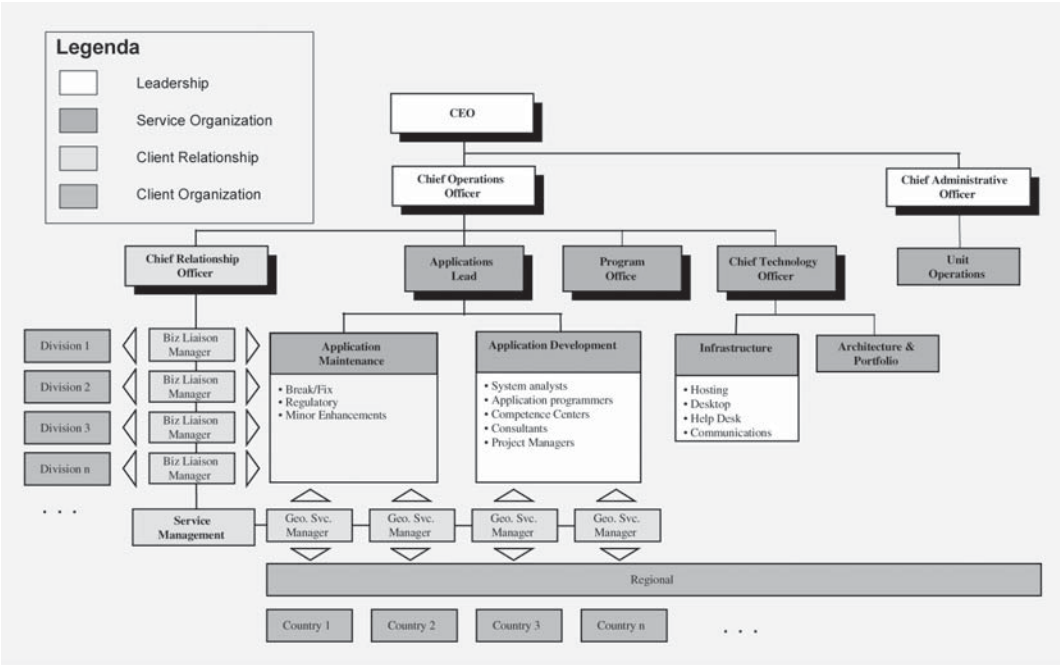
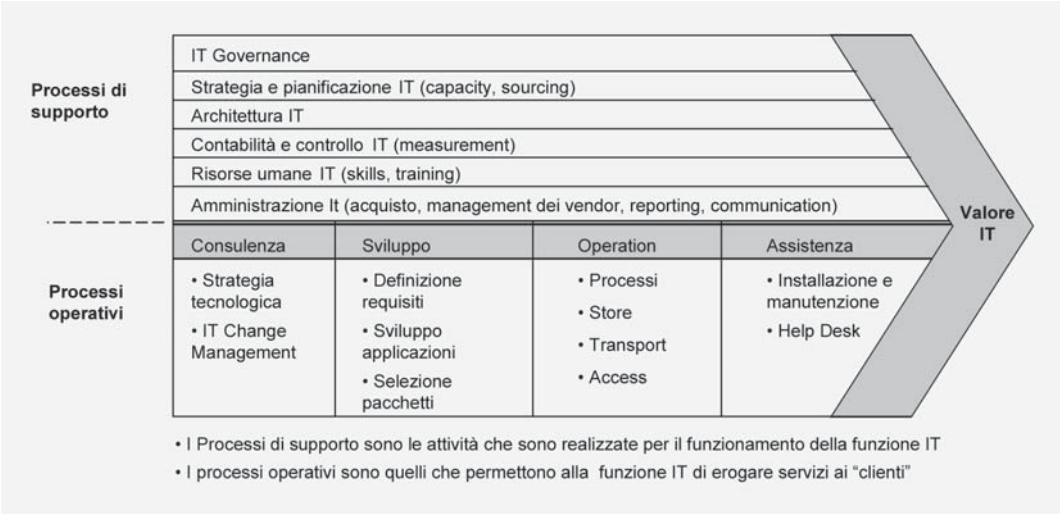


Figura 2.25
Catena del valore dell'IT in azienda.



I processi di supporto sono quelli che permettono il funzionamento della funzione IT. I processi operativi sono quelli che permettono alla funzione sistemi informativi di erogare i servizi ai propri "clienti".

In linea generale, e senza entrare nei dettagli, i processi di supporto sono i seguenti.

1. IT Governance.

- 2. Strategia e pianificazione IT.
- 3. Architettura IT.
- 4. Contabilità e controllo IT.
- 5. Risorse umane.
- 6. Amministrazione, con particolare riferimento alle attività di acquisto, di management dei vendor, di reporting e comunicazione.

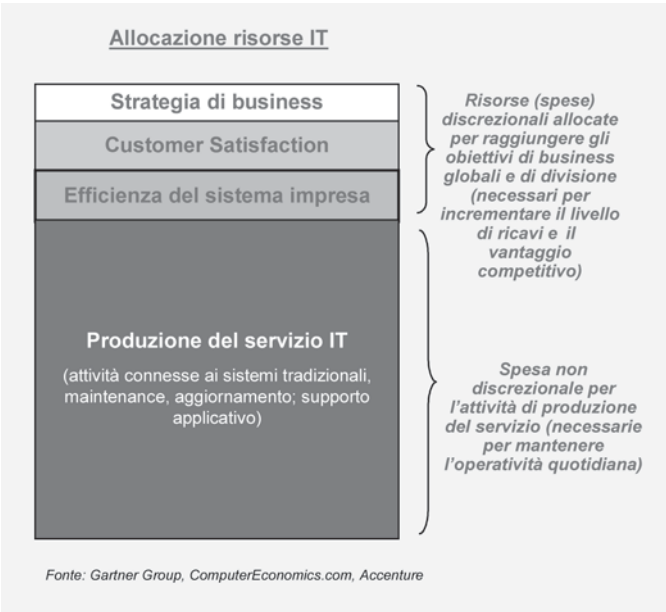
I processi operativi possono essere raggruppati, invece, in *quattro principali* sequenze: *consulenza IT* (strategia tecnologica e IT change management), *sviluppo* (attività legate alla creazione e selezione del portafoglio applicativo e tecnologico), *operations* (attività legate alla produzione dei servizi IT) e *assistenza* (attività legate al service management).

2.6 L'analisi della spesa in IT (is spending)

Un tema di grande interesse solitamente trascurato è quello della spesa IT. Poche sono le analisi dell'allocazione delle risorse destinate al suo mantenimento e sviluppo (Figura 2.26). Fondamentalmente la spesa IT può essere classificata in *spesa non discrezionale*, costituita dalle risorse che necessariamente devono essere allocate per il mantenimento del livello di servizio corrente, e *spesa discrezionale*, costituita dalle risorse indirizzabili verso attività di sviluppo e innovazione.

La prima considerazione da fare è che, dal punto di vista economico, *il budget IT è visto come un centro di costo ed è allocato per la maggior parte sui costi non discrezionali* (produzione e maintenance).

Figura 2.26
Analisi dell'allocazione
delle risorse IT.



Le risorse destinate alla produzione del servizio IT rappresentano tipicamente il 70-80% del budget (con punte del 95%) dell'area IT. Si tratta di budget caratterizzati da:

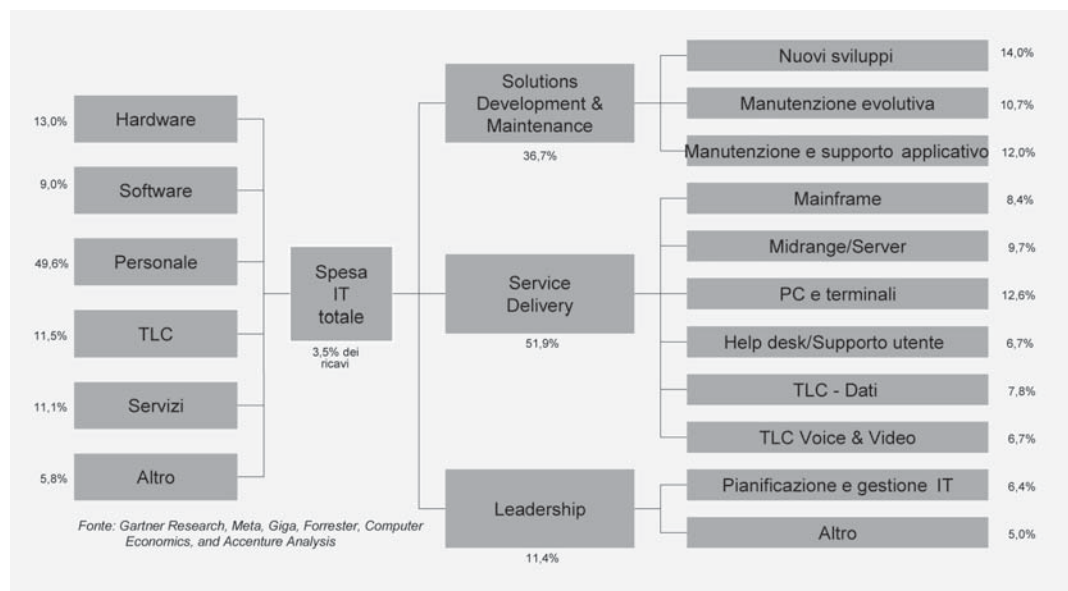
- voci di spesa predefinite con poca varianza nell'entità;
- orientati alla manutenzione e all'evoluzione dei sistemi legacy.

La parte del budget di spesa cosiddetta *non discrezionale* ha una entità del 20-30% del budget totale (con punte minime del 5%). Le risorse investite sono orientate prevalentemente all'allineamento alla strategia d'impresa e all'aumento del valore percepito.

- Strategia di business.
- Customer satisfaction.
- Efficienza globale o di divisione.

In linea molto generale, ponendo attenzione sul fatto che il modello va personalizzato sia in base al settore sia in base al profilo della singola impresa, è possibile ipotizzare una struttura dei costi media dell'area IT (Figura 2.27).

Figura 2.27
Struttura media
della spesa IT.



La spesa IT può essere analizzata sia con riferimento alla *natura* sia con riferimento alla *destinazione*. Con riferimento alla *natura* la spesa IT è per circa il 50% riferibile a costi del personale impegnato nella funzione SI. L'hardware è la seconda voce di spesa con un'incidenza del 13%; seguono i costi di TLC con l'11,5% e i servizi IT con l'11,1%. Il software pesa per il 9%; tutto il resto pesa per circa il 6%.

Con riferimento alla *destinazione*, la spesa IT è per circa il 52% destinata a supportare le attività di produzione dei servizi IT, per circa il 37% destinata alle attività di sviluppo e manutenzione (di cui solo il 14% della spesa totale destinata a nuovi progetti di sviluppo) e per l'11,4% destinata all'attività di management del portafoglio IT ossia alla leadership.

Riepilogo

Il capitolo ha definito e analizzato il concetto di sistema informativo aziendale. Il punto di partenza è stata la considerazione che in un'azienda la tecnologia (T), gli aspetti di business/processi (B) e l'organizzazione (O) sono mantenuti in continuo equilibrio grazie alla presenza di un adeguato "collante" informativo.

I tre elementi interagiscono tra loro grazie alle informazioni che transitano da uno strato all'altro; l'insieme delle informazioni, che costituiscono il collante che unisce organizzazione, processi e tecnologie, è definito *sistema delle informazioni*. Tale sistema ha l'obiettivo di soddisfare il fabbisogno di conoscenza manifestato dal *sistema delle decisioni* e dal *sistema delle operazioni*. Il sistema delle informazioni, si presenta, quindi, come un insieme di flussi d'informazioni aventi lo scopo di rendere le decisioni più rispondenti ai vincoli d'economicità.

In un'ottica sistemica e ricorrendo a una rappresentazione semplificata del sistema aziendale secondo una *visione informativa* o *prospettiva informativa* che considera l'informazione come una risorsa organizzativa, e le tecnologie come il mezzo per supportare l'elaborazione di tale risorsa, questi tre sistemi generali (poiché si riferiscono all'azienda nel suo complesso) possono essere rappresentati come tra loro interconnessi. In dettaglio il sistema delle informazioni, alimenta, ed è alimentato, dal sistema delle decisioni. Il sistema delle decisioni alimenta, ed è alimentato, dal sistema delle operazioni. Ogni operazione, infatti, per essere eseguita necessita di una decisione, mentre non è vero il contrario. Il sistema delle decisioni e quello delle operazioni contribuiscono ad alimentare la base dati aziendale e questa, arricchita dai dati esterni, è elaborata dal sistema tecnologico (o sistema IT nel caso di uso prevalente della tecnologia informatica) che, quindi, provvede ad alimentare il sistema delle informazioni.

Il concetto di *sistema delle informazioni* non è da confondere con quello di *sistema informativo*. Quest'ultimo concetto costituisce il "mezzo" di produzione delle informazioni e non si identifica con esse.

Quando il sistema tecnologico corrisponde a quello informatico, allora parliamo di *sistema informatizzato* o più in generale di *portafoglio IT*, ossia dell'*insieme delle risorse (asset materiali e immateriali) e delle capabilities IT possedute dall'azienda*. Il *portafoglio IT* include, per esempio, hardware, software, apparati di telecomunicazione, apparati di archiviazione dati, device di raccolta e rappresentazione dati e informazioni, persone che forniscono servizi operativi e di supporto. Il *portafoglio IT* include sia le *capabilities IT* realizzate in modalità *in-sourced* (o *modalità make*), sia quelle *acquisite all'esterno (modalità buy)* da terze parti che svolgono attività di *outsourcing*.

Il *portafoglio IT*, inteso quindi come l'*insieme delle risorse e delle capabilities IT posseduti dall'azienda* ha lo scopo di *raccogliere, elaborare, archiviare dati allo scopo di produrre e distribuire le informazioni necessarie, al momento e posto giusto, ai soggetti che ne hanno bisogno, per assumere decisioni e compiere azioni (adempiere a compiti) durante lo svolgimento delle loro attività aziendali*.

Osservando il *portafoglio IT* dal *punto di vista dello studioso*, questo è rappresentabile come un sistema di dieci elementi che interagiscono in modo dinamico tra loro e con l'esterno.

Tali elementi devono essere ispirati al principio di *coerenza*. Essa deve intendersi sia come *coerenza esterna* di un sistema verso il suo ambiente, sia come *coerenza interna* che deve sussistere tra i diversi elementi che lo costituiscono.

Osservando il *portafoglio IT* dal *punto di vista del manager della funzione sistemi informativi* questo è percepibile *come un sistema modulare a strati*.

La "modularità" del *portafoglio IT* si concretizza nel fatto che i sottosistemi di cui sono composti non devono sempre esistere ed essere identici, ma è possibile modificare il contenuto di un sottosistema e l'altro esisterà ugualmente (salvo incompatibilità tecnologiche); la loro relazione è a livello di servizi forniti.

Il concetto di “strato” si riferisce alla presenza di numerosi “livelli” per la produzione ed erogazione dei servizi IT (Figura 2.3).

In particolare, oltre a quello utente, si individuano tre strati fondamentali, alcune dei quali sono a loro volta scomponibili in sub-strati e/o moduli.

1. Management del portafoglio IT.
2. Portafoglio tecnologico.
3. Application management.

I diversi livelli non sono percepiti come indipendenti, ma come strettamente interdipendenti. In particolare il livello degli *utenti* risulta interdipendente con quello dell'*application management* e questo con quello del *facility management*. L'interdipendenza è costituita dai livelli di servizio o SLA (*Service Level Agreement*) e dagli OLA (*Operational Level Agreement*).

Bibliografia

- Ansoff, H. I., 1987, “Concept of Strategy”, in Quinn, J. B., *Strategic Management*, 1987.
- Anthony, R. N., 1965, *Planning and Control: A Framework for Analysis*, Harvard University Press, Cambridge, MA.
- Bertalanffy Von, 1971, *Teoria generale dei sistemi*, I.L.I., Milano.
- Bracchi, G., Francalanci, C., Motta, G., 2001, *Sistemi Informativi e aziende in rete*, McGraw-Hill, Milano.
- Cronin, M. J., 1996, *The Internet Strategy Handbook*, Boston (MA), HBS Press.
- Davenport, T. H., Klahr, P., 1998, “Managing customer support knowledge”, *California Management Review*, 40 (3), pp. 195–208.
- De Marco, M., 2000, *I sistemi informativi aziendali*, Franco Angeli, Milano.
- Kaplan, R. S., Norton, D. P., 1996, *Balanced Scorecard*, Boston, MA, Harvard Business School Press.
- Keen, P. G. W., 1991, *Shaping the Future: Business Design Through Information Technology*, Harvard Business Press, Cambridge.
- Hammer, M., Stanton, S., 1999, “How process enterprises really work”, *Harvard Business Review*, 77 (6), 1999, pp. 108–118.
- McKenney, J. D., 1995, *Waves of Change: Business Evolution Through Information Technology*, Harvard Business School Press, Cambridge, MA.
- Keen, P. G. W., 1991, *Shaping the Future: Business Design through Information Technology*, Harvard Business School Press, Cambridge.
- Mintzberg, H., Waters, J. A., “Of Strategies, deliberate and emergent”, in *Strategic Management Journal*, luglio/settembre, 1985.
- Mella, P., 1997, *Dai sistemi al pensiero sistemico*, Franco Angeli, Milano.
- Morabito, V., 2000, *Il Valore Organizzativo dei Sistemi ERP*, WP Serie Ph.D, Università Bocconi.
- Nelson, R., Winter, S. G., 1982, *An evolutionary theory of economic change*, Cambridge Mass., Harvard University Press.
- Nolan, R. L., Croson, D. C., 1995, *Creative Destruction*, Boston (MA), Harvard Business School Press.
- Optener Stanford, L., 1965, *System analysis for business and industrial problem solving*, Prentice-Hall, Englewood Cliff, New York.
- Pontiggia, A., 2001, *L'impiego efficiente delle tecnologie d'informazione*, Egea, Milano.
- Porter, M. E., 1979, *How Competitive Forces Shape Strategy*, Harvard Business Review, 52 (2), pp. 137–146.
- Quinn, J. B., “Strategies for change”, in Quinn, J. B., *Strategic Management*, 1987.
- Rugiadini, A., 1970, *I sistemi informativi d'impresa*, Giuffrè, Milano.
- Scheer, A. W., Habermann, F., 2000, “Making ERP a Success”, *Communications of the ACM*, 43 (4), pp. 57–61.



Parte seconda

Il contesto della gestione delle basi dati



CAPITOLO

3

L'ambiente delle basi di dati

Al termine di questo capitolo sarete in grado di fare quanto segue.

1. Definire i seguenti termini chiave: base di dati (database), dati, sistema di gestione delle basi di dati, informazioni, metadati, modello di dati aziendale, sistema per la gestione delle risorse aziendali (ERP), extranet, intranet, dati legacy, applicazione di base di dati (database application), data warehouse, indipendenza dei dati, repository (archivio dei dati), vista utente e vincolo.
2. Spiegare perché le basi di dati continueranno a crescere di numero nel prossimo secolo.
3. Citare i numerosi limiti dei sistemi tradizionali di elaborazione dei file.
4. Identificare cinque categorie di basi di dati e le decisioni chiave da prendere per ciascuna categoria.
5. Spiegare sei vantaggi dell'approccio a basi di dati rispetto alla normale elaborazione dei file.
6. Identificare costi e rischi dell'approccio a basi di dati.
7. Elencare e descrivere brevemente nove componenti di un ambiente di base di dati tipico.
8. Descrivere brevemente l'evoluzione dei sistemi di basi di dati.

Introduzione

Negli ultimi due decenni si è assistito a un'enorme crescita del numero e dell'importanza delle applicazioni di basi di dati. Le basi di dati sono utilizzate per memorizzare, manipolare e recuperare i dati in quasi ogni tipo di azienda, comprese le ditte, l'istruzione, la pubblica amministrazione (sanità, scuola e così via) e le biblioteche. La tecnologia delle basi di dati viene solitamente utilizzata da singoli individui su personal computer, da gruppi di lavoro che accedono alle basi di dati tramite server di rete e da tutti i dipendenti che utilizzano applicazioni distribuite a livello della propria azienda.

A fronte di questo periodo di rapida crescita, la domanda di basi di dati e della relativa tecnologia si stabilizzerà? No di certo. Nell'ambiente fortemente competitivo dei primi anni 2000 vi sono più segnali che indicano che la tecnologia delle basi di dati assumerà un'importanza sempre maggiore. Per esempio, dettagliate basi di dati delle vendite possono essere analizzate per determinare i pattern di acquisto dei clienti come riferimento per le campagne pubblicitarie e di marketing. Per questo tipo di applicazioni di supporto alle decisioni, molte aziende creano basi di dati separate chiamate *data warehouse* (Lambert, 1996).

L'uso di basi di dati per il supporto alla gestione delle relazioni con il cliente (CRM, *Custom Relationship Management*), lo shopping on-line e la gestione delle relazioni tra i dipendenti di un'azienda sta diventando sempre più importante. Le basi di dati sono fondamentali per la maggior parte dei sistemi informativi moderni, da quelle piccole utilizzate nei PDA e negli apparecchi informatici a quelle molto grandi che sostengono i sistemi informativi su scala aziendale.

Sebbene il futuro della basi di dati sia assicurato, rimane ancora molto da fare. Molte aziende registrano un proliferare di basi di dati incompatibili sviluppate per far fronte a esigenze immediate e non basate su una strategia pianificata o un'evoluzione ben gestita. Molti dei dati sono intrappolati in sistemi legacy vecchi e spesso sono di qualità scadente. Per progettare nuovi data warehouse sono richieste nuove competenze, che scarseggiano anche in ambiti come l'analisi, la progettazione e l'amministrazione dei dati e delle basi di dati. Questo testo si occupa di questi e di altri aspetti importanti.

Un corso in gestione delle basi di dati è ormai diventato fra i più importanti nei percorsi accademici dei sistemi informativi. In quanto professionisti dei sistemi informativi, occorre essere preparati ad analizzare i requisiti e la progettazione delle basi di dati e implementare queste ultime nel contesto dello sviluppo dei sistemi informativi. Occorre anche essere in grado di consultarsi con gli utenti finali e di mostrare loro come utilizzare le basi di dati o i data warehouse per creare sistemi di supporto alle decisioni e sistemi informativi a supporto dell'attività manageriale per ottenere vantaggi di competitività. L'ampio uso di basi di dati presenti nei siti Web che restituiscono informazioni dinamiche agli utenti del sito richiedono che si capisca non solo come collocare basi di dati sul Web, ma anche come proteggerle, in modo che il loro contenuto venga visualizzato ma non compromesso da utenti esterni.

In questo capitolo presenteremo i concetti fondamentali delle basi di dati e dei sistemi per la loro gestione (DBMS). Descriveremo i sistemi di gestione dei file tradizionali e alcuni dei loro svantaggi, che hanno condotto all'approccio a basi di dati. Descriveremo l'ampia gamma di applicazione delle basi di dati, dai PC ai PDA per i gruppi di lavoro, le funzioni e le imprese e analizzeremo i vantaggi, i costi e i rischi di questo approccio. Concluderemo il capitolo con un breve riepilogo dell'evoluzione dei sistemi di basi di dati e delle tecnologie utilizzate per crearle, utilizzarle e gestirle. Il capitolo è concepito come una panoramica iniziale sugli argomenti approfonditi nel resto del libro.

3.1 Concetti di base e definizioni

Si definisce base di dati (o database) una raccolta organizzata di dati correlati logicamente. Una base di dati può essere di qualsiasi dimensione e complessità. Per esempio, un agente di vendita può gestire sul proprio computer portatile una piccola base di dati di contatti di clienti costituita da pochi megabyte di dati. Una grande società può costruire una base di dati molto estesa costituita da numerosi terabyte di dati (un terabyte corrisponde a un trilione di byte) su un grande computer mainframe utilizzato per applicazioni di supporto alle decisioni (Winter, 1997). Data warehouse molto grandi contengono più di un petabyte di dati (un petabyte corrisponde a un quadrilione di byte). Nel testo si presume che tutte le basi di dati siano basate su computer.

3.1.1 Dati

Storicamente, il termine *dati* faceva riferimento a fatti noti che potevano esser registrati e memorizzati su computer. Per esempio, nella base di dati di un agente di vendita, i dati includono fatti come il cognome, l'indirizzo e il numero di telefono dei clienti. Oggi questa definizione deve essere estesa per riflettere una nuova realtà. Le basi di dati vengono ormai utilizzate per memorizzare oggetti come documenti, immagini fotografiche, suoni e anche parti di video, oltre a dati testuali e numerici convenzionali.

Possono anche includere una registrazione audio o un video clip della conversazione più recente che si è svolta con il cliente. Per rispecchiare questa realtà, si utilizza una definizione estesa: i dati sono costituiti da fatti, testi, immagini, suoni e video che hanno un significato nell'ambiente degli utenti.

Abbiamo definito una base di dati come una raccolta organizzata di dati correlati. Con il termine “organizzata” si intende che i dati sono strutturati in modo da essere facilmente memorizzati, manipolati e recuperati dagli utenti. Con il termine “correlati” si intende che i dati descrivono un campo di interesse per un gruppo di utenti e che gli utenti possono utilizzare i dati per rispondere a domande su quel campo. Per esempio, una base di dati di un'officina meccanica contiene i dati che identificano i clienti (cognome, indirizzo, numero di telefono dell'ufficio e di casa e numero di carta di credito di ciascun cliente), le automobili che appartengono a questi clienti (produttore, modello e anno di fabbricazione), le riparazioni precedenti per ciascun veicolo (data del servizio, cognome della persona che ha lavorato sul veicolo, tipo di riparazione eseguita, importo del lavoro).

3.1.2 Dati e informazioni a confronto

I termini *dati* e *informazioni* sono strettamente legati e vengono spesso utilizzati in modo intercambiabile. È tuttavia utile fare una distinzione tra i due. Le informazioni sono dati che sono stati elaborati in modo da poter aumentare la conoscenza della persona che li utilizza. Si consideri per esempio il seguente elenco di fatti.

Baker, Kenneth D.	324917628
Doyle, Joan E.	476193284
Finkle, Clive R.	548429344
Lewis, John C.	551742186
McFerran, Debra R.	409723145
Sisneros, Michael	392416582

Questi fatti soddisfano la nostra definizione di “dati”, ma è piuttosto chiaro che in questa forma sono pressoché inutili. Sebbene si deduca facilmente che si tratta di una lista di nomi, non si ha un'idea del loro significato. Si noti cosa accade se si collocano gli stessi dati in un contesto, come mostrato nella Figura 3.1a. Aggiungendo poche altre voci di dati e fornendo una struttura, si riconosce facilmente il registro di un corso universitario. Queste sono informazioni utili per alcuni utenti, per esempio per l'insegnante del corso e per l'ufficio esami.

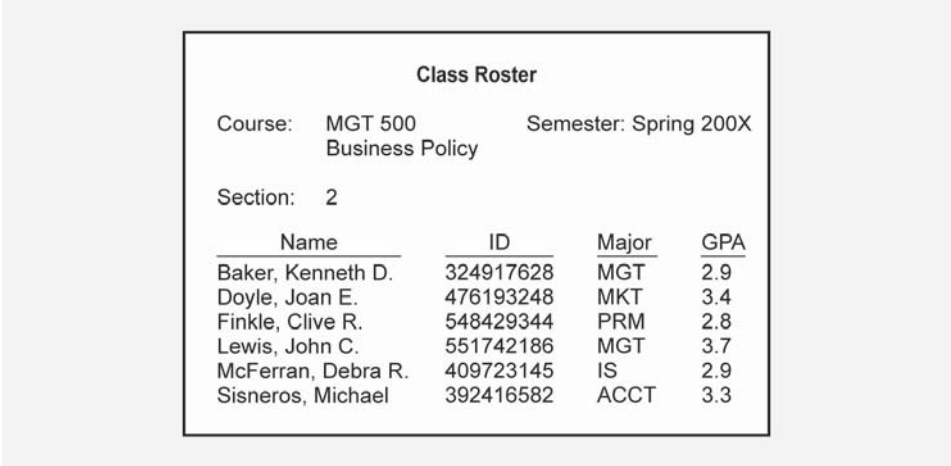
Un altro modo per convertire i dati in informazioni è quello di riepilgarli o di elaborarli in altro modo e poi presentarli a un'interpretazione umana. Per esempio, la Figura 3.1b mostra i dati di immatricolazione degli studenti sotto forma grafica. Queste informazioni possono essere utilizzate come base per decidere se attivare nuovi corsi o assumere nuovi docenti.

Nella pratica, le basi di dati moderne contengono dati o informazioni (o entrambi) così come li abbiamo definiti. Per esempio, una base di dati può contenere un'immagine del documento di registro mostrato nella Figura 3.1a. I dati vengono spesso pre-elaborati e memorizzati in una forma riepilogata nelle basi di dati che vengono utilizzate per il supporto alle decisioni. In questo libro utilizzeremo il termine “base di dati” senza distinguere il suo contenuto, se di dati o di informazioni.

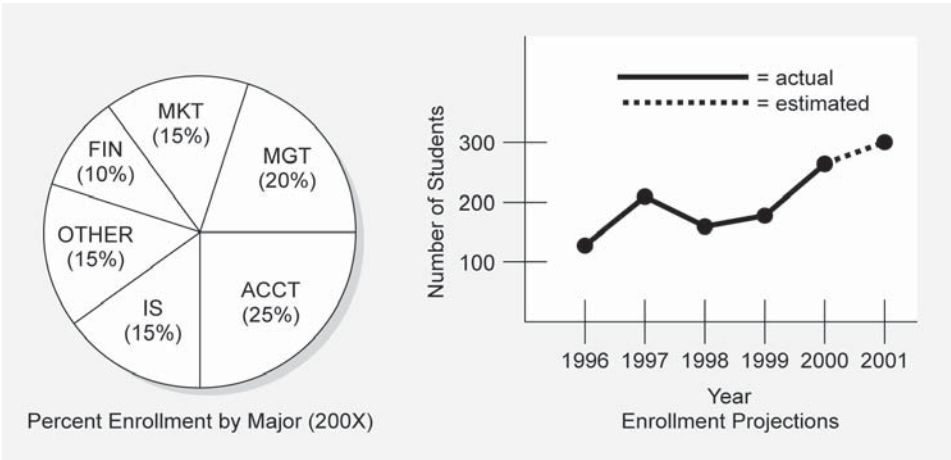
3.1.3 Metadati

Come abbiamo spiegato, i dati diventano utili quando vengono collocati in un contesto. Il meccanismo principale per fornire un contesto per i dati è quello dei metadati. I metadati sono dati che descrivono le proprietà o le caratteristiche di altri dati. Alcune di queste proprietà includono le definizioni dei dati, le strutture dei dati e le regole o i vincoli.

Figura 3.1
Convertire i dati in informazioni.
(a) I dati in un contesto.



(b) I dati riepilogati.



Alcuni metadati di esempio del registro mostrato nella Figura 3.1a sono elencati nella Tabella 3.1. Per ognuna delle voci che appaiono nel registro, i metadati mostrano il nome della voce di dati, il tipo di dati, la lunghezza, i valori minimo e massimo consentiti (se opportuno) e una breve descrizione della voce di dati. Si noti la distinzione tra dati e metadati: i metadati vengono eliminati dai dati. I metadati descrivono cioè le proprietà dei dati ma non comprendono i dati. I metadati mostrati nella Tabella 3.1 non includono i dati di esempio del registro mostrato nella Figura 3.1a. I metadati consentono ai progettisti e agli utenti delle basi di dati di capire quali dati sono presenti, cosa significano e che differenze ci sono tra voci di dati apparentemente simili. La gestione dei metadati è tanto cruciale quanto quella dei dati associati, poiché i dati che non hanno un significato chiaro possono confondere, essere fraintesi o risultare errati.

Tabella 3.1 Metadati di esempio per il registro di classe.

	Voce di dati			Valore	
Nome	Tipo	Lunghezza	Minimo	Massimo	Descrizione
Course	Alfanumerico	30			ID e nome del corso
Section	Intero	1	1	9	Numero del seminario
Semester	Alfanumerico	10			Semestre e anno
Name	Alfanumerico	30			Cognome dello studente
ID	Intero	9			ID studente
Major	Alfanumerico	4			Specializzazione dello studente
GPA	Decimale	3	0.0	4.0	Media dei voti dello studente

3.2 Sistemi tradizionali di elaborazione dei dati (a file)

All’inizio dell’elaborazione dei dati basata su computer, non esistevano basi di dati. I computer, che erano notevolmente meno potenti di quelli odierni, occupavano grandi stanze ed erano utilizzati quasi esclusivamente per calcoli scientifici e ingegneristici. Gradualmente, i computer vennero introdotti nel mondo delle aziende. Per essere utili alle applicazioni aziendali, i computer devono essere in grado di memorizzare, manipolare e recuperare grossi file di dati. I sistemi di elaborazione dei dati sono nati con questo scopo. Sebbene nel tempo si siano evoluti, la loro struttura di base e il loro obiettivo sono cambiati solo di poco nel corso dei decenni.

Man mano che le applicazioni aziendali diventavano più complesse, diventava evidente che i sistemi tradizionali di elaborazione dei dati presentavano numerosi svantaggi e limiti, come si vedrà successivamente. La conseguenza è che attualmente questi sistemi sono stati sostituiti da sistemi di elaborazione delle basi di dati nelle applicazioni aziendali più importanti. È tuttavia utile avere un po’ di familiarità con i sistemi di elaborazione dei file, per i motivi illustrati di seguito.

1. Attualmente i sistemi di elaborazione dei file sono ancora piuttosto diffusi, specialmente per il backup dei sistemi di basi di dati.
2. La comprensione dei problemi e dei limiti dei sistemi di elaborazione dei file può aiutare a evitare gli stessi problemi durante la progettazione dei sistemi di basi di dati.

Nel resto di questo paragrafo si descriveranno i sistemi di elaborazione dei file e se ne mostreranno i limiti mediante un esempio realistico. Nel prossimo paragrafo si utilizzerà lo stesso esempio per introdurre e confrontare i sistemi di elaborazione delle basi di dati.

Sistemi di elaborazione dei dati per Pine Valley Furniture Company

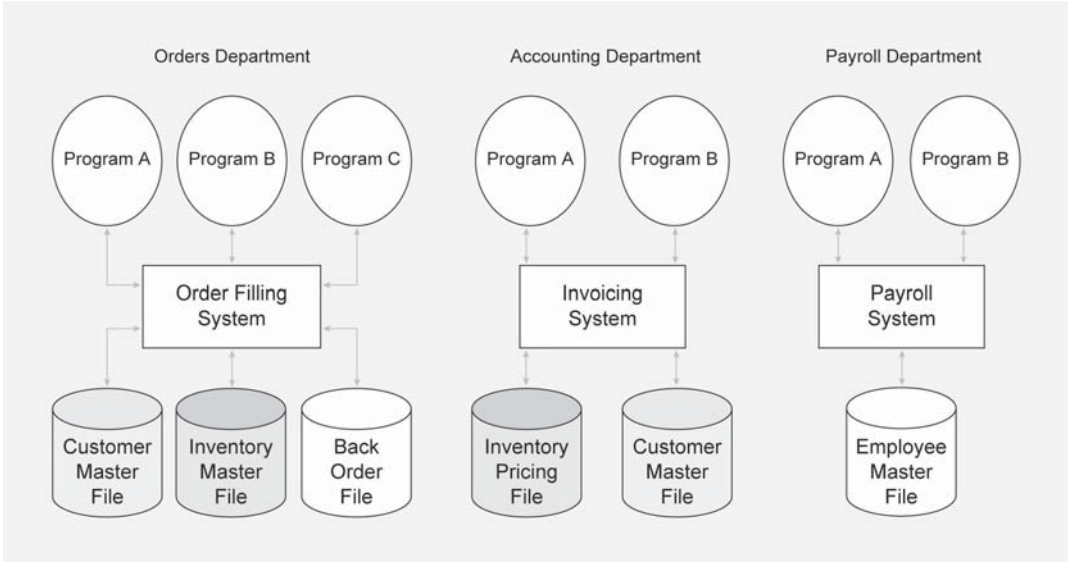
Pine Valley Furniture Company produce mobili in legno di alta qualità e li distribuisce a negozi in tutto il Paese. Tra le numerose linee di prodotti dell’azienda vi sono i tavoli da computer, gli arredamenti per i centri ricreativi, i set da sala da pranzo, le librerie e i mobili da parete. I clienti possono effettuare gli ordini con vari mezzi: telefono, posta, fax o form elettroniche in Internet. La società impiega attualmente un totale di circa 100 persone e sta attraversando una fase di crescita.

Le prime applicazioni informatiche della società utilizzavano l’approccio di elaborazione dei file tradizionale. Questo approccio si concentrava sulle esigenze di elaborazione dei dati delle singole funzioni invece di valutare le esigenze informative globali dell’azienda. La funzione SI risponde solitamente

agli utenti che necessitano di nuovi sistemi sviluppando o acquisendo nuovi programmi per le singole applicazioni, come il controllo delle giacenze, la fatturazione o la gestione delle risorse umane. Ogni programma applicativo o sistema sviluppato è concepito per soddisfare le esigenze della funzione o del gruppo di utenti specifici che lo richiedono. Non esiste quindi una mappa complessiva, un piano o un modello che guidi la crescita delle applicazioni.

Tre delle applicazioni che si basano sull'approccio dell'elaborazione dei file sono mostrate nella Figura 3.2. I sistemi illustrati sono Order Filling, Invoicing e Payroll. La figura mostra anche i file di dati principali associati a ciascuna applicazione. Un file è una raccolta di record correlati. Per esempio, Order Filling System ha tre file: Custom Master, Inventory e Back Order. Si noti che è presente una duplicazione di alcuni dei file utilizzati nelle tre applicazioni, com'è tipico dei sistemi di elaborazione dei dati di tipo tradizionale.

Figura 3.2
Tre sistemi di elaborazione dei file dell'azienda Pine Valley Furniture.



3.2.1 Svantaggi dei sistemi tradizionali di elaborazione dei file

I sistemi di elaborazione dei file convenzionali presentano alcuni svantaggi. La Tabella 3.2 li elenca e ciascuno di essi è descritto brevemente di seguito.

Tabella 3.2 Svantaggi dei sistemi di elaborazione dei file.

Dipendenza dai dati dal programma
Duplicazione dei dati
Condivisione dei dati limitata
Tempi di sviluppo prolungati
Manutenzione eccessiva del programma

Dipendenza dei dati dal programma

Le descrizioni dei file vengono memorizzate in ciascun programma applicativo che accede a un dato file. Per esempio, nell'Invoicing System della Figura 3.2, Program A accede sia a Inventory Pricing File sia a Customer Master File. Di conseguenza, il programma contiene una descrizione di file dettagliata per entrambi questi file. Qualsiasi modifica alla struttura di un file richiede quindi una modifica alle descrizioni di file di tutti i programmi che accedono al file.

Si noti che nella Figura 3.2 Customer Master File è utilizzato sia nell'Order Filling System sia nell'Invoicing System. Se si decide, per esempio, di modificare la lunghezza del campo dell'indirizzo dei clienti nei record di questo file da 30 a 40 caratteri, le descrizioni del file in ciascuno dei programmi interessati (fino a cinque) dovrà essere modificata di conseguenza. Spesso è difficile individuare tutti i programmi interessati da tali modifiche. Quel che è peggio è che durante le modifiche vengono spesso introdotti degli errori.

Duplicazione dei dati

Poiché le applicazioni vengono spesso sviluppate indipendentemente nei sistemi di elaborazione dei file, i file di dati duplicati e non previsti sono più la regola che l'eccezione. Per esempio, nella Figura 3.2, l'Order Filling System contiene un Inventory Master File, mentre l'Invoicing System contiene un Inventory Pricing File. Questi file contengono entrambi senza dubbio dati che descrivono i prodotti di Pine Valley Furniture, come la descrizione dei prodotti, il prezzo unitario e la quantità disponibile. Questa duplicazione è eccessiva, poiché richiede un maggiore spazio di memorizzazione e un maggiore sforzo per mantenere tutti i file aggiornati. Sfortunatamente i file di dati duplicati comportano spesso la perdita dell'integrità dei dati, poiché i formati dei dati possono non essere omogenei o i valori di dati possono non corrispondere (o entrambe le cose). Una ridondanza non pianificata e non controllata di questo tipo può anche portare alla perdita dell'integrità dei metadati. Per esempio, la stessa voce di dati può avere nomi diversi in file diversi o, al contrario, lo stesso nome di file può essere utilizzato per voci di dati diverse in file diversi.

Condivisione dei dati limitata

Secondo l'approccio tradizionale dell'elaborazione dei file, ogni applicazione ha i propri file privati e gli utenti hanno poche possibilità di condividere i dati all'esterno delle applicazioni proprietarie. Per esempio, nella Figura 3.2, gli utenti dell'Accounting Department hanno accesso all'Invoicing System e ai suoi file, ma probabilmente non all'Order Filling System o al Payroll System e ai loro file. Spesso è frustrante per i manager scoprire che un report richiederà uno sforzo di programmazione maggiore per ottenere i dati da più file incompatibili in sistemi separati. Un altro sforzo potrebbe inoltre essere richiesto laddove unità organizzative diverse contengono file diversi.

Tempi di sviluppo prolungati

Con i sistemi di elaborazione dei file tradizionali, c'è poca possibilità di trarre profitto da sforzi di sviluppo precedenti. Ogni nuova applicazione richiede che lo sviluppatore parta fondamentalmente da zero progettando nuovi formati e descrizioni di file e quindi scrivendo la logica di accesso ai file per ciascun programma nuovo. I tempi di sviluppo prolungati necessari si scontrano spesso con l'ambiente aziendale veloce di oggi, nel quale il mercato (o il tempo di produzione di un sistema informativo) è un fattore chiave del successo dell'azienda.

Manutenzione eccessiva del programma

I fattori finora elencati si combinano a creare un carico di gestione di manutenzione del programma molto pesante in quelle aziende che si basano sui sistemi di elaborazione dei file tradizionali. In aziende di questo tipo, almeno l'80 per cento del budget dei sistemi informativi deve infatti essere riservato alla manutenzione dei programmi. Questo lascia ovviamente poco spazio allo sviluppo di nuove applicazioni.

È importante notare che molti degli svantaggi citati dell'elaborazione dei file possono essere anche dei limiti per le basi di dati qualora un'azienda non applichi nel modo opportuno l'approccio a basi di dati. Per esempio, se un'azienda sviluppa separatamente più basi di dati di gestione (per esempio una per ogni reparto o funzione aziendale) coordinando poco o non coordinando i metadati, si possono avere una duplicazione incontrollata dei dati, una condivisione limitata dei dati, tempi di sviluppo prolungati e una manutenzione del programma eccessiva. L'approccio a basi di dati, spiegato nel prossimo paragrafo, è soprattutto un modo per gestire i dati organizzativi e una tecnologia per definire, creare, mantenere e utilizzare questi dati.

3.3 L'approccio a basi di dati (database)

L'approccio a basi di dati enfatizza l'integrazione e la condivisione dei dati nell'azienda, o almeno per le sue aree principali. Questo approccio richiede una modifica dell'orientamento di base o un'evoluzione nel processo di progettazione, a partire dai livelli dirigenziali superiori. Questa modifica dell'orientamento è difficile per la maggior parte delle aziende; molte stanno effettuando il passaggio di recente, imparando che le informazioni possono essere utilizzate come armi nella concorrenza.

3.3.1 L'approccio a basi di dati per Pine Valley Furniture Company

A partire dai primi anni '90 la concorrenza nella produzione dei mobili si è intensificata e gli altri produttori sembravano rispondere più prontamente di Pine Valley Furniture alle nuove opportunità d'affari. Sebbene numerose ragioni spiegassero questa tendenza, i manager si sono resi conto che i sistemi informativi da loro utilizzati fino a quel momento (basati sull'elaborazione tradizionale dei dati) erano ormai datati. L'azienda ha così avviato uno sforzo di sviluppo che ha portato all'adozione di un approccio a basi di dati. Questo approccio viene descritto brevemente in questo capitolo e nel dettaglio nel Capitolo 4.

Modello di dati aziendale

Il primo passo di Pine Valley Furniture nel passaggio a un approccio a basi di dati è stato quello di sviluppare un elenco di entità di alto livello che supportassero le attività dell'azienda. Un'entità è un oggetto o un concetto importante per l'azienda. Alcune delle identità di alto livello di Pine Valley Furniture sono CUSTOMER, PRODUCT, EMPLOYEE, CUSTOMER ORDER e DEPARTMENT.

Una volta che queste entità sono state identificate e definite, l'azienda ha sviluppato un modello di dati aziendale. Un modello di dati aziendale è un modello grafico che mostra le entità di alto livello per l'azienda e le associazioni tra queste entità. Un segmento del modello di dati aziendale che contiene quattro entità e tre associazioni pertinenti è mostrato nella Figura 3.3. Le entità mostrate in questo segmento di modello sono le seguenti.

- **CUSTOMER.** Le persone e le aziende che acquistano o potrebbero potenzialmente acquistare prodotti da Pine Valley Furniture.
- **ORDER.** L'acquisto di uno o più prodotti da parte di un cliente.
- **PRODUCT.** Gli articoli prodotti e venduti da Pine Valley Furniture.
- **ORDER LINE.** I dettagli su ciascun prodotto venduto in base a un ordine specifico di un cliente (come quantità e prezzo).

Le tre associazioni mostrate nella figura, e rappresentate da tre linee che collegano le entità, identificano le tre regole aziendali fondamentali seguenti.

1. Ogni CUSTOMER effettua (*Places*) un determinato numero di ORDER. Per contro, ogni ORDER viene eseguito (*Is placed by*) in corrispondenza esatta di un CUSTOMER.
2. Ogni ORDER contiene (*Contains*) un determinato numero di ORDER LINE. Per contro, ogni ORDER LINE è contenuta (*Is contained in*) esattamente in un ORDER.
3. Ogni PRODUCT ha (*Has*) un numero di ORDER LINE. Per contro, ogni ORDER LINE riguarda (*Is for*) un PRODUCT.

Places, Contains e Has sono associazioni uno-a-molti perché, per esempio, un cliente effettua potenzialmente più ordini e un ordine corrisponde esattamente a un cliente.

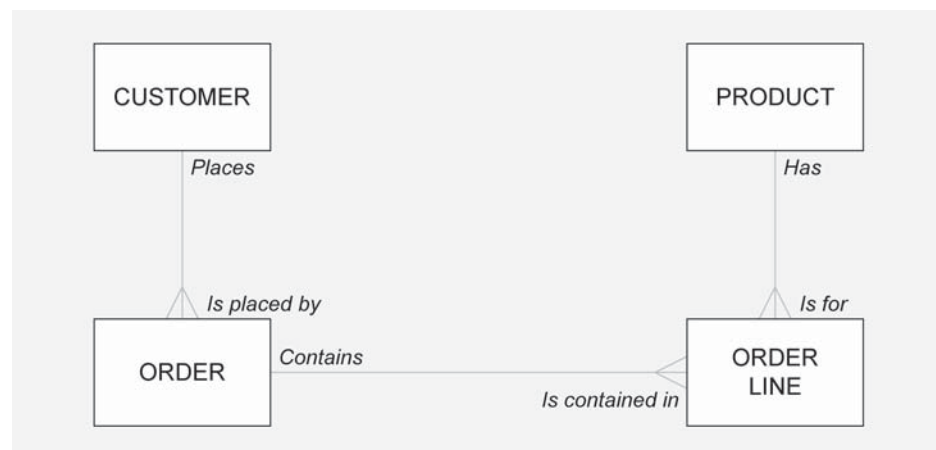
Il tipo di diagramma mostrato nella Figura 3.3 viene definito *diagramma di entità-associazione*. I diagrammi di entità-associazione (o diagrammi E-R, da *entity-relationship*) sono talmente importanti nelle applicazioni di basi di dati che vi dedicheremo due capitoli, il 5 e il 6. Il Capitolo 5 descrive i diagrammi E-R di base e la modellazione dei dati, mentre il Capitolo 6 descrive la modellazione di dati avanzata. Entrambi questi capitoli descrivono anche i vari tipi di regole aziendali e come vengono individuate e modellate.

Si notino le seguenti caratteristiche del modello di dati aziendale.

1. È un modello dell'azienda che fornisce informazioni importanti su come questa funziona, oltre a vincoli importanti.
2. Il modello di dati aziendale accentua l'integrazione dei dati e dei processi focalizzandosi sulle entità, le associazioni e le regole aziendali.

Figura 3.3

Segmento del modello di dati aziendale (Pine Valley Furniture Company).



Basi di dati relazionali

I risultati degli studi preliminari hanno convinto il management dei vantaggi potenziali dell'approccio a basi di dati. L'azienda ha deciso di implementare un sistema moderno di base di dati relazionale che visualizzi tutti i dati sotto forma di tabelle (descriveremo le basi di dati relazionali nel Capitolo 7). La Figura 3.4 mostra le quattro tabelle Customer, Product, Order e Order_Line contenenti alcuni dati di esempio. Si noti che queste tabelle rappresentano le quattro entità mostrate nel modello di dati aziendale della Figura 3.3.

Figura 3.4
Quattro relazioni (Pine Valley Furniture Company).
(a) Tabelle Order e Order_Line.

The screenshot shows the Microsoft Access interface with two tables displayed in Datasheet View. The 'ORDER' table has columns for Order_id, Order_Date, and Customer_ID. The 'Order line' table has columns for Order_ID, Product_ID, and Quantity. Both tables show a list of records with navigation controls at the bottom.

Order_id	Order_Date	Customer_ID
1	9/8/2000	4
2	10/1/2000	3
3	7/19/2000	1
4	11/1/2000	6
5	7/28/2000	4
6	8/17/2000	6
7	8/17/2000	3
(AutoNumber)		0

Order_ID	Product_ID	Quantity
1	2	2
1	10	2
2	3	1
2	8	2
2	14	2
3	6	2
4	3	1
4	7	3
5	6	2
5	10	1
0	0	0

(b) Tabelle Product e Customer.

The screenshot shows the Microsoft Access interface with two tables displayed in Datasheet View. The 'PRODUCT' table has columns for Product ID, Product Description, Product Finish, Unit Price, and On Hand. The 'CUSTOMER' table has columns for Customer ID, Customer Name, Address, City, State, and Postal Code. Both tables show a list of records with navigation controls at the bottom.

Product ID	Product Description	Product Finish	Unit Price	On Hand
1	End Table	Cherry	\$175.00	8
2	Coffe Table	Birch	\$200.00	4
3	Computer Desk	Oak	\$375.00	5
4	Entertainment Center	Maple	\$650.00	3
5	Writer's Desk	Oak	\$325.00	0
6	8-Drawer Dresser	Birch	\$750.00	5
7	48" Bookcase	Walnut	\$150.00	5
8	48" Bookcase	Oak	\$175.00	2
9	96" Bookcase	Walnut	\$225.00	4
10	96" Bookcase	Oak	\$200.00	4
11	4-Drawer Dresser	Oak	\$500.00	3
12	8-Drawer Dresser	Oak	\$800.00	2
13	Nightstand	Cherry	\$150.00	5
14	Writer's Desk	Birch	\$300.00	2

Customer ID	Customer Name	Address	City	State	Postal Code
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-
3	Home Furnishings	1900 Allard Ave	Albany	NY	12209-
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-

Ogni colonna di tabella rappresenta un attributo, o caratteristica, di un'entità. Per esempio, gli attributi mostrati per Customer sono Customer_ID, Customer_Name, Address, City, State e Postal_Code. Ogni riga della tabella rappresenta un'istanza, o occorrenza, dell'entità. Una proprietà importante del modello relazionale è che rappresenta le associazioni tra le entità con valori memorizzati nelle colonne delle tabelle corrispondenti. Per esempio, si noti che Customer_ID è un attributo sia della tabella Customer sia della tabella Order. Di conseguenza possiamo collegare facilmente un ordine al suo cliente associato. Per esempio, possiamo determinare che l'Order_ID 3 è associato al Customer_ID 1. Si è in grado di determinare quali Product_ID sono associati all'Order_ID 2? Nei prossimi capitoli si vedrà come recuperare i dati da queste tabelle utilizzando un linguaggio di interrogazione potente che sfrutta queste connessioni.

Implementare basi di dati relazionali

L'approccio a basi di dati offre diversi vantaggi rispetto ai vecchi sistemi di elaborazione dei file. Alcuni di questi vantaggi verranno sfruttati se l'azienda implementerà una serie di basi di dati indipendenti, poiché questo approccio non permette la condivisione dei dati da parte di più membri. Le basi di dati indipendenti hanno in effetti molti degli svantaggi comuni ai sistemi di elaborazione dei file.

Per facilitare la condivisione dei dati e delle informazioni, Pine Valley Furniture Company utilizza una rete LAN (*Local Area Network*) che collega le stazioni di lavoro degli impiegati nelle varie funzioni a un server di base di dati (o *database server*), come mostrato nella Figura 3.5. Per migliorare la comunicazione all'interno dell'azienda, è stata installata anche una intranet che utilizza la tecnologia Internet basata sul Web e che è accessibile solo all'interno dell'azienda. Ogni stazione di lavoro può essere utilizzata come browser Web per ottenere un accesso rapido a informazioni aziendali quali le directory telefoniche, le specifiche di progettazione dei mobili, la posta elettronica e così via. La stazione di lavoro può essere utilizzata anche come personal computer e per connettersi al server della base di dati quando necessario attraverso la LAN.

Pine Valley Furniture Company anticipa l'aggiunta di un'interfaccia Web alle sue applicazioni aziendali, come una voce di ordine, così che più attività aziendali interne possano essere condotte attraverso la sua intranet. L'azienda intende inoltre diventare più accessibile ai suoi clienti attraverso Internet. Molti dei rivenditori delle varie linee di mobili vogliono verificare attraverso Internet la disponibilità degli articoli nell'inventario di Pine Valley Furniture Company. I rivenditori possono così fornire queste informazioni ai propri clienti.

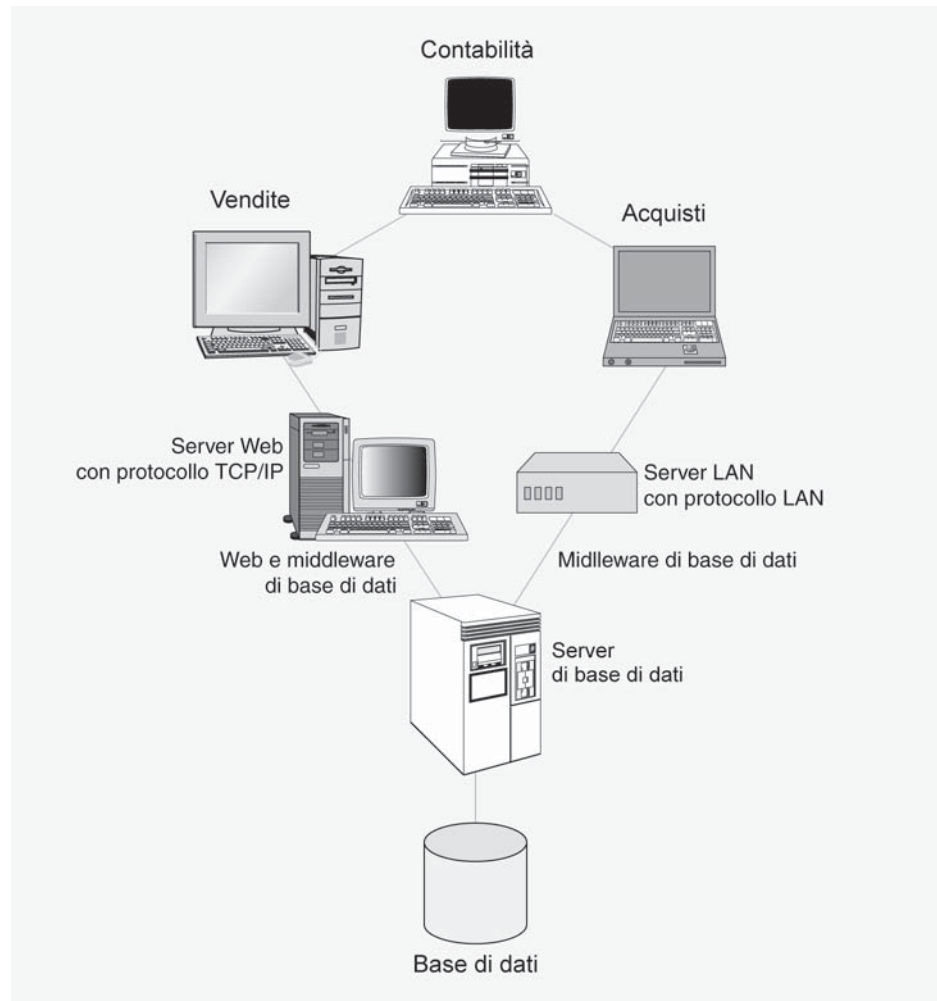
Sebbene la base di dati supporti in modo adeguato le operazioni quotidiane dell'azienda, i manager hanno intuito immediatamente che la stessa base di dati è spesso inadeguata per le applicazioni di supporto alle decisioni. Quelle che seguono sono alcune domande a cui non è semplice dare risposta.

1. Qual è il modello di vendita dei mobili per quest'anno rispetto allo stesso periodo dell'anno scorso?
2. Chi sono i 10 clienti più grandi e quali sono i loro modelli d'acquisto?
3. Quali tipi di mobili vengono solitamente venduti insieme (per esempio, i tavoli con le sedie)?

Per rispondere a queste e ad altre domande, un'azienda deve spesso creare una base di dati separata che contenga informazioni storiche e riepilogate. Una base di dati di questo tipo è solitamente un data warehouse o, in qualche caso, un data mart. Gli analisti necessitano inoltre di strumenti specifici per il supporto alle decisioni per interrogare e analizzare la base di dati. Una classe di strumenti utilizzati a questo scopo è quella degli strumenti di elaborazione analitica on-line (OLAP).

Figura 3.5

Sistema di computer per Pine Valley Furniture Company.



Un'applicazione di base di dati

Un'applicazione di base di dati (o database application) è un programma applicativo (o un insieme di programmi correlati) utilizzato per eseguire una serie di attività per conto degli utenti della base di dati. Ogni applicazione di base di dati esegue una combinazione delle seguenti operazioni fondamentali.

1. *Creare*, ovvero aggiungere nuovi dati alla base di dati.
2. *Leggere*, ovvero leggere i dati correnti della base di dati, spesso presentati in un formato utile sullo schermo di un computer o su un report stampato.
3. *Aggiornare*, ovvero aggiornare o modificare i dati correnti della base di dati.
4. *Eliminare*, ovvero eliminare i dati correnti dalla base di dati.

Una delle applicazioni di base di dati più importanti di Pine Valley Furniture Company è quella che crea le fatture per i clienti. Quando gli articoli ordinati sono pronti per essere spediti al cliente, viene

preparata una fattura che riepiloga gli articoli inclusi nell'ordine e che calcola l'importo dovuto per l'ordine. Una fattura tipica viene mostrata nella Figura 3.6. A meno che l'ordine originale non sia stato modificato, la fattura viene preparata automaticamente in base ai dati memorizzati nella base di dati (Figura 3.4).

3.4 I tipi di applicazioni di base di dati

Le basi di dati spaziano da quelle per un singolo utente con un computer da scrivania o un PDA a quelle sui mainframe con migliaia di utenti. Possono essere suddivise in cinque categorie: personali, di gruppo di lavoro, di funzione, aziendali e Internet/intranet/extranet. Presenteremo ciascuna categoria con un esempio e a seguire esporremo alcuni dei problemi che possono sorgere per la categoria d'uso specifica.

3.4.1 Basi di dati personali

Le basi di dati personali sono concepite per supportare un solo utente e sono rimaste a lungo relegate sui personal computer (PC) e sui portatili. L'introduzione recente dei PDA ha permesso di incorporare le basi di dati personali in dispositivi palmari che funzionano non solo come computer, ma anche come telefoni cellulari, fax e browser Web. Le applicazioni di base di dati semplici che memorizzano informazioni sui clienti e dettagli dei contatti presso i quali possono essere rintracciati i clienti possono essere utilizzate da un PC o da un PDA ed essere quindi trasferite facilmente da un dispositivo all'altro per scopi di backup o di lavoro. Per esempio, si consideri una società che impiega diversi agenti di

Figura 3.6
Fattura per un cliente (Pine Valley Furniture Company).

vendita che contattano clienti attuali o potenziali. Se ogni agente utilizza anche altre applicazioni, come presentazioni grafiche delle vendite e programmi di listino prezzi che aiutano l'agente a determinare la combinazione migliore tra quantità e tipo di articoli da ordinare, un computer portatile può essere adatto in quanto a requisiti di memoria e di prestazioni. Se l'agente deve semplicemente tenere un elenco dei contatti, un PDA con un'applicazione di gestione dei contatti che utilizza una piccola base di dati può essere la soluzione migliore. Un elenco di contatti di clienti tipico è mostrato nella Figura 3.7.

Alcune delle decisioni principali da prendere quando si sviluppano basi di dati personali sono le seguenti.

- 1. L'applicazione deve essere acquistata da un produttore esterno o deve essere sviluppata all'interno dell'azienda?
- 2. Se l'applicazione di base di dati viene sviluppata internamente, deve essere sviluppata dall'utente finale o da un professionista dei sistemi informativi (IS) interni?
- 3. Quali dati sono richiesti dall'utente e come deve essere strutturata la base di dati?
- 4. Quale prodotto di sistema di gestione delle basi di dati (DBMS) deve essere utilizzato per questa applicazione?

Figura 3.7
Dati tipici di una base di dati personale.

Customer

Customer Name:
Multi Media, Inc.

Address:
1000 River Road

City:
San Antonio

State:
TX

Zip:
76235

Phone:
(219) 864-2000

Next Contact Date:
10/17/2000

Time:
10:30 AM

Contact History for Customer

Date	Time	Contact	Comments
08/04/2000	10:00 AM	Roberts	Review proposal
08/19/2000	08:00 AM	Roberts	Revise schedule
09/10/2000	09:00 AM	Pearson	Sign contract
09/21/2000	02:00 PM	Roberts	Follow up

- 5. Come devono essere sincronizzati i dati della base di dati personale con i dati nelle altre basi di dati?
- 6. Chi è responsabile dell'accuratezza dei dati nella base di dati personale?

Le basi di dati personali sono ampiamente utilizzate perché spesso possono migliorare la produttività personale. Tuttavia presentano un rischio: i dati non possono essere facilmente condivisi con altri utenti. Per esempio, si supponga che il manager delle vendite voglia avere una vista consolidata dei contatti dei clienti. Questa non può essere fornita facilmente e rapidamente dalle basi di dati dei singoli agenti di vendita. Il problema è molto comune: i dati che sono importanti per una persona, spesso diventano o diventeranno importanti anche per altre. Per questo motivo le basi di dati personali dovrebbero essere limitate a situazioni particolari, come un'azienda molto piccola, nelle quali l'esigenza di condividere i dati fra utenti delle basi di dati personali può sorgere difficilmente.

3.4.2 Basi di dati di gruppo di lavoro

Un *gruppo di lavoro* è una team relativamente piccolo di persone che collaborano sullo stesso progetto o applicazione o su un gruppo di progetti o applicazioni simili. Un gruppo di lavoro comprende solitamente meno di 25 persone. Queste possono essere impegnate, per esempio, in un progetto di costruzione o nello sviluppo di una nuova applicazione. Una base di dati di gruppo di lavoro è concepita per supportare gli sforzi collaborativi di un team di questo tipo.

Si consideri un gruppo di lavoro che sviluppa oggetti o componenti software sia standard sia personalizzati che vengono venduti tanto a produttori di software quanto a utenti finali. La Tabella 3.3 contiene un elenco di alcuni oggetti software sviluppati di recente. Solitamente, una o più persone lavorano su un determinato oggetto o componente in un tempo dato. Al gruppo di lavoro occorre una base di dati che tenga traccia di ogni elemento man mano che viene sviluppato e che consenta una facile condivisione dei dati tra i membri del team.

Tabella 3.3 Elenco degli oggetti software in una base di dati di gruppo di lavoro di esempio.

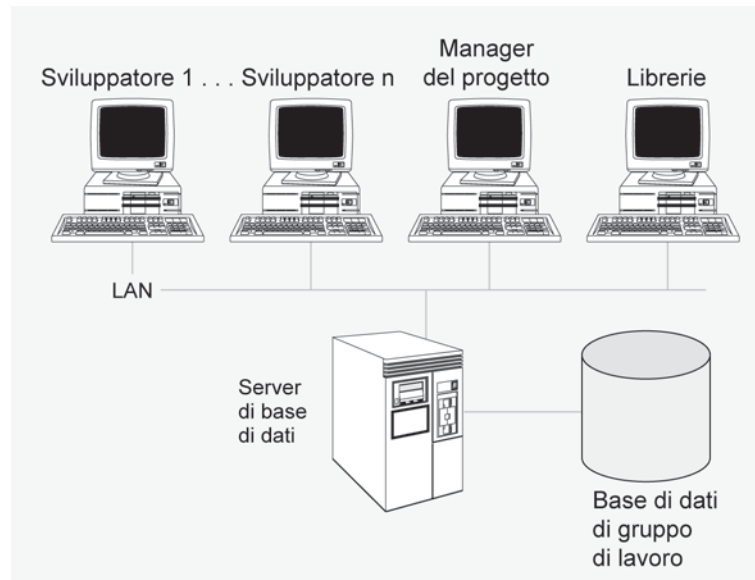
Object Place, Inc.
400 Magnolia St.
Atlanta, GA 02103

Nome	Linguaggio	Descrizione	Prezzo in \$
123Xender	Visual Basic	Wrapper di foglio di calcolo	595
DSSObjects	C++	Generatore di supporto alle decisioni	595
ObjectSuite	Smalltalk	Set di 6 oggetti generici	5000
OrderObject	Smalltalk	Oggetto di ordine generico	1000
PatientObject	Smalltalk	Oggetto di paziente generico	1000

Il metodo di condivisione dei dati in questa base di dati è mostrato nella Figura 3.8. Ogni membro del gruppo di lavoro ha un computer da scrivania e i computer sono collegati tramite una LAN. La base di dati è memorizzata su un dispositivo centrale chiamato *server di base di dati*, anch'esso collegato alla rete. Ogni membro del gruppo di lavoro ha quindi accesso ai dati condivisi. Diversi tipi di membri del gruppo (per esempio lo sviluppatore o il manager del progetto) possono avere viste utente diverse di questa base di dati condivisa. Si noti che questa disposizione aggira l'obiezione principale rispetto

Figura 3.8

Base di dati di gruppo di lavoro con LAN.



alle basi di dati su PC, ovvero che i dati non vengono condivisi facilmente (cosa che invece qui accade, almeno all'interno del gruppo di lavoro). Questa disposizione introduce tuttavia diversi problemi di gestione dei dati non presenti nelle basi di dati personali (a utente singolo), come quelli della protezione e dell'integrità dei dati nel momento dell'aggiornamento contemporaneo da parte di più utenti. Inoltre, poiché un'azienda è composta da più gruppi di lavoro e che un singolo può far parte di gruppi di lavoro diversi in momenti diversi, è possibile generare molte basi di dati, come nel caso delle basi di dati personali.

Quando si crea una base di dati di gruppo di lavoro, l'azienda deve rispondere alle stesse domande che si applicano alle basi di dati personali. Sorgono tuttavia altre domande di gestione specifiche.

1. Come può essere ottimizzata la struttura della base di dati a fronte della diversità delle esigenze informative dei membri del gruppo di lavoro?
2. Come possono i vari membri utilizzare la base di dati contemporaneamente senza comprometterne l'integrità?
3. Quali operazioni di elaborazione della base di dati devono essere eseguite sulle stazioni di lavoro e quali sul server?

3.4.3 Basi di dati di funzione

Una *funzione* è un'unità organizzativa in un'azienda. Esempi tipici di funzioni sono il personale, il marketing, la produzione e la contabilità. Una funzione è solitamente più grande di un gruppo di lavoro (tra le 25 e le 100 persone) ed è responsabile di più attività molto diverse tra loro.

Le basi di dati di funzione sono concepite per supportare le varie funzioni e attività della funzione e sono il tipo più comune di base di dati tra i cinque qui descritti. Per esempio, si consideri una base di dati del personale che è concepita per tenere traccia dei dati riguardanti i dipendenti, i ruoli, le competenze

e le assegnazioni di lavoro. Una volta che i dati più importanti sono stati memorizzati nella base di dati, gli utenti possono interrogare quest'ultima per ottenere le risposte alle seguenti domande.

1. Per una particolare mansione (per esempio, ingegnere del software), quali opportunità professionali esistono attualmente nella società?
2. Per la stessa mansione, quali competenze sono richieste?
3. Quali sono le competenze di un particolare dipendente? Al contrario, quali dipendenti possiedono una competenza specifica (per esempio, saper programmare in C++)?
4. A quali dipendenti sono stati assegnati determinati compiti? Al contrario, qual è la storia professionale di un determinato dipendente?
5. Quali dipendenti rientrano sotto la supervisione di un determinato manager?

Tra le domande tipiche da affrontare quando si progettano e implementano basi di dati di funzione, oltre a quelle già elencate, vi sono le seguenti.

1. Come la base di dati e il suo ambiente possono essere progettati in modo da produrre prestazioni adeguate, dato il gran numero di utenti e di transazioni di utenti?
2. Come si può fornire una protezione adeguata contro le rivelazioni non autorizzate o la diffusione dei dati sensibili?
3. Quali strumenti di base di dati o di sviluppo di applicazioni devono essere utilizzati in questo ambiente complesso?
4. Le altre funzioni mantengono lo stesso tipo di dati e, in caso affermativo, come si possono gestire al meglio la ridondanza e la coerenza dei dati e dei metadati?
5. Gli utenti della base di dati sono geograficamente dispersi o la dimensione della base di dati è tale che i dati devono essere memorizzati su più sistemi di computer, creando così una *base di dati distribuita*?

3.4.4 Basi di dati aziendali

Una base di dati aziendale ha come ambito l'intera azienda (o almeno, molte funzioni diverse). Basi di dati di questo tipo sono concepite per supportare operazioni e decisioni a livello di azienda. Si noti che un'azienda può avere più basi di dati aziendali, e che una base di dati di questo tipo non include quindi tutti i dati dell'azienda. Una base di dati aziendale singola e operativa è poco adatta a contesti organizzativi medi e grandi a causa delle difficoltà di prestazioni delle basi di dati molto estese, delle esigenze diverse di utenti diversi e della difficoltà di raggiungere una singola definizione di dati (metadati) per tutti gli utenti della base di dati. Una base di dati aziendale supporta tuttavia le esigenze informative per più funzioni. Nell'ultimo decennio, l'evoluzione delle basi di dati aziendali ha comportato due linee di sviluppo principali.

1. I sistemi ERP (*Enterprise Resource Planning*).
2. Le implementazioni di data warehouse.

I sistemi ERP (**Enterprise Resource Planning**) sono un'evoluzione dei sistemi MRP (*Materials Requirements Planning*) e MRP-II (*Manufacturing Resources Planning*) degli anni '70 e '80. Laddove i sistemi MRP programmano i materiali e i componenti grezzi e i requisiti di sotto-assemble per i

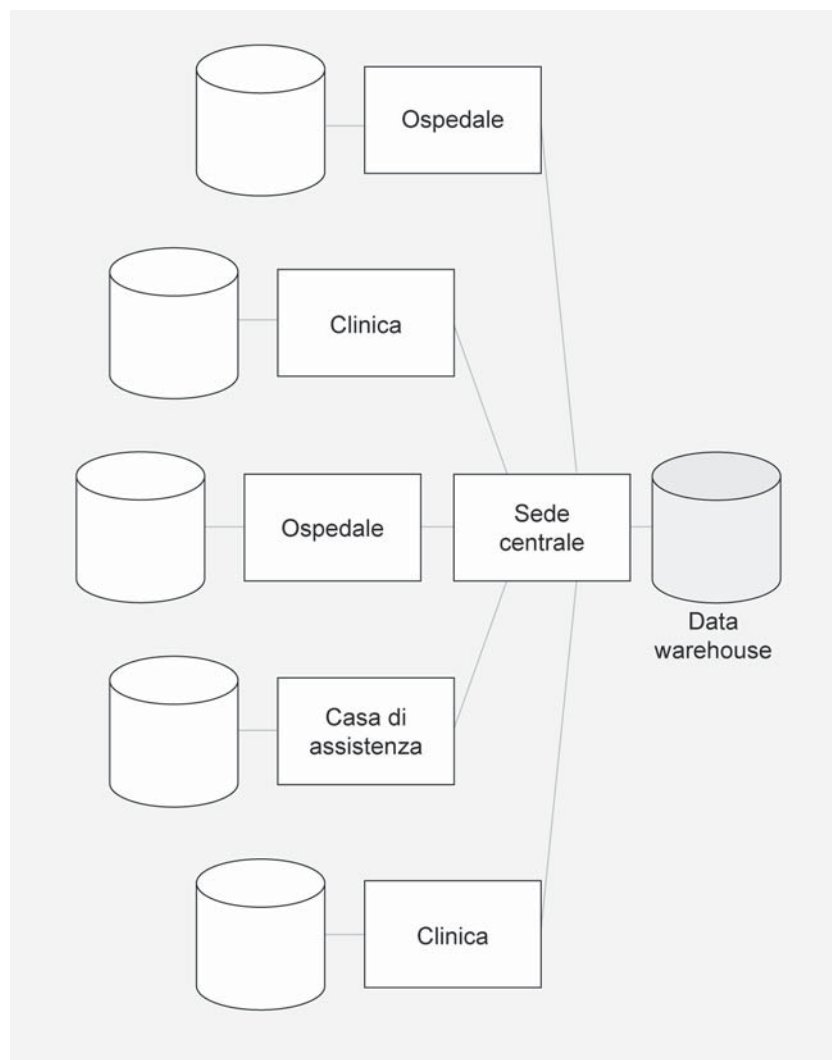
processi di produzione, i sistemi MRP-II programmavano anche spazi di vendita e di distribuzione dei prodotti. L'estensione alle funzioni aziendali rimanenti si è risolta successivamente nei sistemi di gestione aziendali e nei sistemi di pianificazione delle risorse. Tutti i sistemi ERP sono fortemente dipendenti dalle basi di dati per quanto riguarda la memorizzazione dei dati richiesta dalle applicazioni ERP.

Laddove i sistemi ERP lavorano con i dati operazionali correnti dell'impresa, i **data warehouse** raccolgono il loro contenuto dalle varie basi di dati operazionali, comprese quelle personali, di gruppo di lavoro e di funzione. I data warehouse forniscono agli utenti la possibilità di lavorare con i dati storici per identificare pattern, tendenze e risposte alle domande aziendali strategiche. Descriveremo i data warehouse nel dettaglio nel Capitolo 12.

Si consideri una grande azienda sanitaria che gestisce un gruppo di centri medici che comprendono ospedali, cliniche e case di assistenza. Come mostrato nella Figura 3.9, ognuno di questi centri medici ha una o più basi di dati separate che supportano le varie operazioni presso i singoli servizi. Queste basi

Figura 3.9

Un data warehouse aziendale.



di dati contengono dati riguardanti i pazienti, i medici, i servizi, le operazioni aziendali e altre entità correlate.

Le basi di dati forniscono un supporto adeguato per la maggior parte delle funzioni in ogni singolo centro medico. L'azienda rileva tuttavia la necessità di una vista unica e consolidata dell'intera azienda, per esempio per vedere le attività totali riguardanti un singolo paziente o fornitore. Un'efficienza operativa può essere raggiunta, per esempio, ordinando centralmente le forniture per tutte le unità mediche e programmando il lavoro del personale e dei servizi attraverso tutte le unità. Un sistema ERP rende possibile questi approcci. Le decisioni aziendali, le trattative con i fornitori esterni (per esempio le compagnie assicurative) e il reporting alle varie agenzie richiedono dati e informazioni storici compilati. Per soddisfare queste esigenze, l'azienda ha creato un data warehouse che è gestito dalla sede centrale. I dati nel data warehouse vengono derivati estraendo e riepilogando periodicamente i dati delle singole basi di dati e trasmettendoli per mezzo di una rete di telecomunicazioni al data warehouse aziendale.

Nel contesto di una base di dati aziendale, sorgono numerose domande.

1. Come devono essere distribuiti i dati tra le varie posizioni nella struttura aziendale?
2. Come può l'azienda sviluppare e mantenere gli standard riguardanti i nomi dei dati, le definizioni, i formati e gli aspetti correlati?
3. Quali azioni devono essere intraprese per poter integrare con successo i numerosi sistemi, compresi i dati legacy dei sistemi precedenti, necessari per le analisi?

3.4.5 Basi di dati Internet, intranet ed extranet

Il cambiamento più recente che riguarda l'ambiente delle basi di dati è l'affermarsi di Internet, una rete mondiale che connette utenti di più piattaforme attraverso un'interfaccia nota come *browser Web*. L'accettazione di Internet da parte delle aziende ha comportato modifiche radicali nei modelli aziendali di vecchia data. Società di successo sono state scosse dalla concorrenza di nuove aziende che hanno impiegato Internet per fornire informazioni e servizi avanzati ai clienti, eliminando i canali tradizionali di marketing e di distribuzione e implementando una nuova gestione dei rapporti con i clienti (per esempio, i clienti configurano e ordinano i loro computer nuovi direttamente presso il produttore; vengono accettate prenotazioni di voli aerei nel giro di pochi minuti, con un risparmio fino al 40 per cento; informazioni su posizioni aperte e attività aziendali sono rese disponibili da numerose società).

Ognuna di queste applicazioni richiede un supporto di base di dati e molte applicazioni richiedono un accesso universale. La facile connessione di più piattaforme permette alle aziende di riorganizzare le proprie operazioni e di sviluppare nuove applicazioni più rapidamente a costi inferiori. Un'interfaccia standard consente agli utenti di essere produttivi con una formazione ridotta e richiede meno supporto. Centrale per lo sviluppo di applicazioni utili è la possibilità di unirvi una base di dati da cui possano essere recuperate le informazioni correnti. Quando una base di dati è abilitata per il Web, l'interfaccia del browser permette agli utenti di porre domande univoche e specifiche e di ricevere risposte in base alle informazioni correnti. La risposta alla domanda è automatizzata; non è necessario seguire una particolare procedura telefonica né attendere l'aiuto di un operatore. Le basi di dati abilitate per il Web sono indispensabili per lo sviluppo dei siti di shopping on-line. Le società fanno di tutto per raccogliere informazioni sui clienti (pattern di acquisto, navigazione sul sito, permanenza su ogni pagina e così via) per poter migliorare gli sforzi di gestione delle relazioni con essi (CRM).

La maggior parte degli esempi precedentemente citati riflette una relazione *business-to-customer* (BtoC, ossia azienda-cliente), sebbene alcuni clienti dell'azienda siano altri clienti (e in questo caso si parla di relazioni *business-to-business*, o BtoB). Internet viene utilizzato per facilitare le relazioni BtoC

perché i clienti sono forzatamente esterni all'azienda e la loro possibilità di accedere ai dati e alle informazioni dell'azienda è fondamentale per il successo della relazione. Consentire questo tipo di accesso esterno a una base di dati aziendale solleva problemi di protezione e di integrità dei dati che sono una novità per i sistemi di gestione delle informazioni, nei quali i dati rimanevano tradizionalmente protetti e sigillati all'interno di ogni azienda.

Per anni le società si sono scambiate le informazioni per mezzo dello scambio elettronico di dati (EDI). Molte società continuano a utilizzare i loro sistemi EDI per condurre le proprie relazioni BtoB. Alcune di esse, soprattutto quelle che sono nate da poco o che non hanno precedentemente utilizzato EDI per lo scambio di informazioni tra aziende, hanno creato delle extranet. Una extranet utilizza la tecnologia Internet, ma l'accesso all'extranet non è universale come nel caso di un'applicazione Internet. L'accesso viene infatti limitato ai fornitori e ai clienti dell'azienda con i quali è stato raggiunto un accordo sull'uso e l'accesso legittimi ai dati e alle informazioni delle due parti. Solitamente questi fornitori e clienti possono accedere a una parte dell'intranet dell'azienda, come si vedrà in seguito. Questo accesso facilita le relazioni aziendali fornendo un'elaborazione e un accesso più veloci ed efficienti alle informazioni.

Come detto in precedenza, molte società hanno utilizzato la tecnologia Internet per creare reti private concepite per la gestione delle informazioni a livello interno. A prima vista una pagina intranet non è diversa da una Internet, ma il suo accesso è limitato ai soli dipendenti interni all'azienda. L'accesso alle basi di dati aziendali è quindi limitato. Le intranet possono anche stabilire una connessione Internet, ma questa connessione sarà protetta da un firewall, che impedisce a utenti esterni di connettersi all'intranet.

3.4.6 Riepilogo delle applicazioni di base di dati

La Tabella 3.4 mostra un riepilogo dei vari tipi di applicazioni di base di dati descritti in questo paragrafo. Per ogni tipo di base di dati vengono mostrati il numero tipico di utenti, l'architettura specifica (architettura server/client, descritta nel Capitolo 11) e le dimensioni.

Tabella 3.4 Riepilogo delle applicazioni di base di dati (adattato da White, 1995).

Tipo di base di dati	Numero tipico di utenti	Architettura tipica	Dimensione tipica
Personale	1	Computer desktop/portatile	Megabyte
Gruppo di lavoro	5-25	Client/server (a due livelli)	Megabyte-gigabyte
Funzione	25-100	Client/server (a tre livelli)	Gigabyte
Impresa	>100	Client/server (server distribuito o parallelo)	Gigabyte-terabyte
Internet	>1000	Server Web e application server	Megabyte-gigabyte

3.5 Vantaggi dell'approccio a basi di dati

L'approccio a basi di dati offre numerosi vantaggi potenziali, se confrontato ai sistemi tradizionali di elaborazione dei dati. I vantaggi principali sono riassunti nella Tabella 3.5 e vengono descritti di seguito.

Tabella 3.5 Vantaggi dell'approccio a basi di dati

Indipendenza dei dati dal programma
Ridondanza dei dati minima
Coerenza dei dati migliorata
Condivisione dei dati migliorata
Produttività migliorata dello sviluppo dell'applicazione
Applicazione degli standard
Qualità dei dati migliorata
Accessibilità e risposta dei dati migliorate
Manutenzione dei programmi ridotta

3.5.1 Indipendenza dei dati dal programma

La separazione delle descrizioni dei dati (metadati) dai programmi applicativi che utilizzano i dati è chiamata **indipendenza dei dati**. Con l'approccio a basi di dati, le descrizioni dei dati vengono memorizzate in un punto centrale chiamato *repository* (deposito). Questa proprietà dei sistemi di basi di dati permette ai dati di un'azienda di mutare ed evolvere, entro certi limiti, senza modificare i programmi applicativi che li elaborano.

3.5.2 Minima ridondanza dei dati

L'obiettivo di progettazione dell'approccio a basi di dati è quello di integrare file di dati precedentemente separati e ridondanti in un'unica struttura logica. Ogni fatto principale viene idealmente registrato in un punto della base di dati. Per esempio, il fatto che il prodotto con Product_ID 3 sia una tavola da computer di quercia che costa 375 \$ viene registrato in un punto della tabella Product (Figura 3.4b). L'approccio a basi di dati non elimina completamente la ridondanza, ma permette al progettista di controllarne attentamente il tipo e l'entità. Per esempio, ogni ordine nella tabella Order (Figura 3.4a) contiene un Customer_ID per stabilire la relazione tra gli ordini e i clienti. In altri momenti si può desiderare di includere una ridondanza limitata per migliorare le prestazioni della base di dati, come vedremo nei capitoli successivi.

3.5.3 Maggiore coerenza dei dati

Eliminando o controllando la ridondanza dei dati si riduce notevolmente l'eventualità della loro incoerenza. Per esempio, se l'indirizzo di un cliente è memorizzato una sola volta, non si potrà contestare il valore memorizzato. L'aggiornamento dei valori di dati viene decisamente semplificata quando ogni valore è memorizzato in un unico punto. Infine, si evita lo spreco di spazio di memoria tipico della memorizzazione di dati ridondanti.

3.5.4 Condivisione dei dati migliorata

Una base di dati è concepita come risorsa aziendale condivisa. Agli utenti interni ed esterni autorizzati è concesso il permesso di utilizzare la base di dati e a ogni utente, o gruppo di utenti, vengono fornite una o più viste che facilitino questo uso. Una **vista utente** è una descrizione logica di una parte della base di dati che viene richiesta da un utente per eseguire una determinata operazione. Per esempio, ogni tabella

della base di dati di Pine Valley Furniture Company (Figura 3.4) può costituire una vista utente. Una vista utente è spesso una maschera o un report che comprendono i dati di una o più tabelle. Per esempio, anche la fattura di un cliente (Figura 3.6) è una vista utente della stessa base di dati; un catalogo di prodotti disponibile sul sito Web di Pine Valley Furniture Company è un'altra vista utente.

3.5.5 Maggiore produttività dello sviluppo dell'applicazione

Un vantaggio fondamentale dell'approccio a basi di dati è che riduce enormemente i costi e i tempi di sviluppo delle nuove applicazioni aziendali. Ci sono due motivi importanti per i quali le applicazioni di basi di dati possono essere sviluppate più rapidamente di quelle di file convenzionali.

1. Assumendo che la base di dati e le applicazioni di individuazione e manutenzione dei dati correlati siano già state progettate e implementate, il programmatore può concentrarsi sulle funzioni specifiche richieste per la nuova applicazione, senza doversi preoccupare dei dettagli della progettazione o dell'implementazione al livello più basso.
2. Il sistema di gestione delle basi di dati fornisce alcuni strumenti di produttività di alto livello, come generatori di maschere e report e linguaggi di alto livello, che automatizzano alcune delle attività di progettazione e di implementazione della base di dati. Descriveremo molti di questi strumenti nei prossimi capitoli.

3.5.6 Applicazione degli standard

Quando l'approccio a basi di dati viene implementato con un supporto gestionale completo, la funzione di amministrazione della base di dati deve stabilire un'unica autorità responsabile che definisca e applichi gli standard dei dati. Questi standard includono le convenzioni di denominazione, gli standard di qualità dei dati e le procedure uniformi per l'accesso, l'aggiornamento e la protezione dei dati. Il repository di dati fornisce agli amministratori della base di dati un insieme di strumenti potenti per lo sviluppo e l'applicazione di questi standard. Sfortunatamente, il fallimento nell'implementazione di una funzione di amministrazione di base di dati solida è forse la fonte più comune dei fallimenti delle basi di dati nelle aziende.

3.5.7 Maggiore qualità dei dati

La preoccupazione per la qualità scadente dei dati è oggi un tema comune nell'amministrazione delle basi di dati (Ballou e Tayi, 1999). L'approccio a basi di dati fornisce alcuni strumenti e processi per migliorare la qualità dei dati. Due dei più importanti sono i seguenti.

1. I progettisti della base di dati possono specificare i vincoli di integrità applicati dal DBMS. Un vincolo è una regola che non può essere violata dagli utenti della base di dati. Descriveremo i numerosi tipi di vincoli, chiamati anche *regole aziendali*, nei Capitoli 5 e 6.
2. Uno degli obiettivi di un ambiente di data warehouse è quello di ripulire (*scrub*) i dati operazionali prima che vengano collocati nel data warehouse (Jordan, 1996). Descriveremo i data warehouse e il potenziale di miglioramento della qualità dei dati nel Capitolo 12.

3.5.8 Accessibilità e risposta dei dati migliorate

Con una base di dati relazionale, gli utenti finali senza esperienza di programmazione possono recuperare e visualizzare i dati anche quando questi superano i confini tradizionali della funzione. Per esempio, un impiegato può visualizzare le informazioni sui tavoli da computer di Pine Valley Furniture Company con la seguente query:

```
SELECT *
FROM PRODUCT
WHERE Product_Name = "Computer Desk";
```

Il linguaggio utilizzato in questa interrogazione è chiamato *Structured Query Language*, o SQL (lo studieremo nel dettaglio nei Capitoli 9 e 10).

3.5.9 Ridotta manutenzione dei programmi

I dati memorizzati possono essere modificati di frequente per svariate ragioni: vengono aggiunte nuove voci di dati, i formati dei dati sono cambiati e così via. Un celebre esempio è il famoso “problema dell’anno 2000”, in cui i campi comuni di due cifre dell’anno venivano estesi a quattro cifre per assecondare il passaggio dall’anno 1999 all’anno 2000.

In un ambiente di elaborazione a file, la descrizione dei dati e la logica per accedervi sono costruiti in singoli programmi applicativi (si tratta della dipendenza dei dati trattata precedentemente). Come risultato, le modifiche al formato dei dati e ai metodi di accesso comportano inevitabilmente la necessità di modificare i programmi applicativi. In un ambiente a base di dati, i dati sono maggiormente indipendenti dai programmi applicativi che li utilizzano. Entro certi limiti, è possibile modificare i dati o gli applicativi che li utilizzano senza apportare modifiche all’altro fattore. Di conseguenza, in un ambiente di base di dati moderno la manutenzione del programma può essere significativamente ridotta.

3.5.10 Avvertenze sui vantaggi delle basi di dati

Questo paragrafo ha identificato nove benefici potenziali principali dell’approccio a basi di dati. Occorre tuttavia essere consapevoli del fatto che molte aziende sono rimaste frustrate nel tentativo di realizzare alcuni di questi vantaggi. Per esempio, l’obiettivo dell’indipendenza dei dati (e quindi della manutenzione ridotta dei programmi) si è rivelato sfuggente a causa dei limiti dei vecchi modelli di dati e del software di gestione delle basi di dati. Fortunatamente, il modello relazionale (così come il modello più recente orientato agli oggetti) fornisce un ambiente decisamente migliore per il raggiungimento di questi benefici. Un’altra ragione del fallimento risiede in una pianificazione organizzativa e in un’implementazione delle basi di dati scadenti; nemmeno il miglior software di gestione delle basi di dati può superare questi limiti. Ecco perché in questo testo si insiste molto sulla fase di pianificazione e progettazione della base di dati.

3.6 Costi e rischi dell’approccio a basi di dati

Come accade per qualsiasi decisione aziendale, l’approccio a basi di dati implica alcuni costi e rischi supplementari che devono essere individuati e gestiti al momento dell’implementazione (Tabella 3.6).

Tabella 3.6 Costi e rischi dell'approccio a basi di dati.

Personale nuovo e specializzato
Costi e complessità di installazione e gestione
Costi di conversione
Esigenze di backup e ripristino
Conflitti organizzativi

3.6.1 Personale nuovo e specializzato

Le aziende che adottano l'approccio a basi di dati devono impiegare o formare qualcuno che progetti e implementi le basi di dati, fornisca servizi di amministrazione delle basi di dati e gestisca del personale nuovo. Inoltre, a causa dei rapidi cambiamenti nella tecnologia, il personale nuovo dovrà essere continuamente e regolarmente formato e aggiornato. Questo aumento del personale può essere compensato da altre entrate della produttività, ma un'azienda non dovrebbe mai minimizzare la necessità di competenze specializzate, richieste per trarre il massimo dai vantaggi potenziali.

3.6.2 Costi e complessità di installazione e gestione

Un sistema di gestione di base di dati multiutente è una suite estesa e complessa di software che ha un costo iniziale elevato, che richiede un personale formato che la installi e la renda operativa e che implica costi annuali di manutenzione e supporto. L'installazione di un tale sistema può anche richiedere aggiornamenti all'hardware dei sistemi di comunicazione dei dati dell'azienda. La formazione è solitamente continua, così da adeguarsi alle nuove release e ai nuovi aggiornamenti. Un software supplementare più sofisticato e costoso può essere necessario per fornire sicurezza e per garantire un aggiornamento costante dei dati condivisi.

3.6.3 Costi di conversione

Il termine *sistemi legacy* è ampiamente utilizzato per fare riferimento alle applicazioni meno recenti di un'azienda basate sull'elaborazione a file e/o su una tecnologia di basi di dati vecchia. Il costo per convertire questi sistemi in una tecnologia a basi di dati moderna (misurato in soldi, tempo e impegno organizzativo) può spesso apparire proibitivo. Come vedremo nel Capitolo 12, l'uso dei data warehouse è una delle strategie per continuare a utilizzare i sistemi vecchi sfruttando contemporaneamente la tecnologia e le tecniche delle basi di dati moderne.

3.6.4 Esigenze di backup e ripristino

Una base di dati aziendale condivisa deve essere sempre precisa e disponibile in qualsiasi momento. Questo richiede che siano sviluppate e utilizzate procedure articolate per fornire copie di backup dei dati e per ripristinare una base di dati in caso di danneggiamenti. Un sistema di gestione moderno automatizza solitamente molte più operazioni di backup e di ripristino di un sistema di elaborazione di file.

3.6.5 Conflitti organizzativi

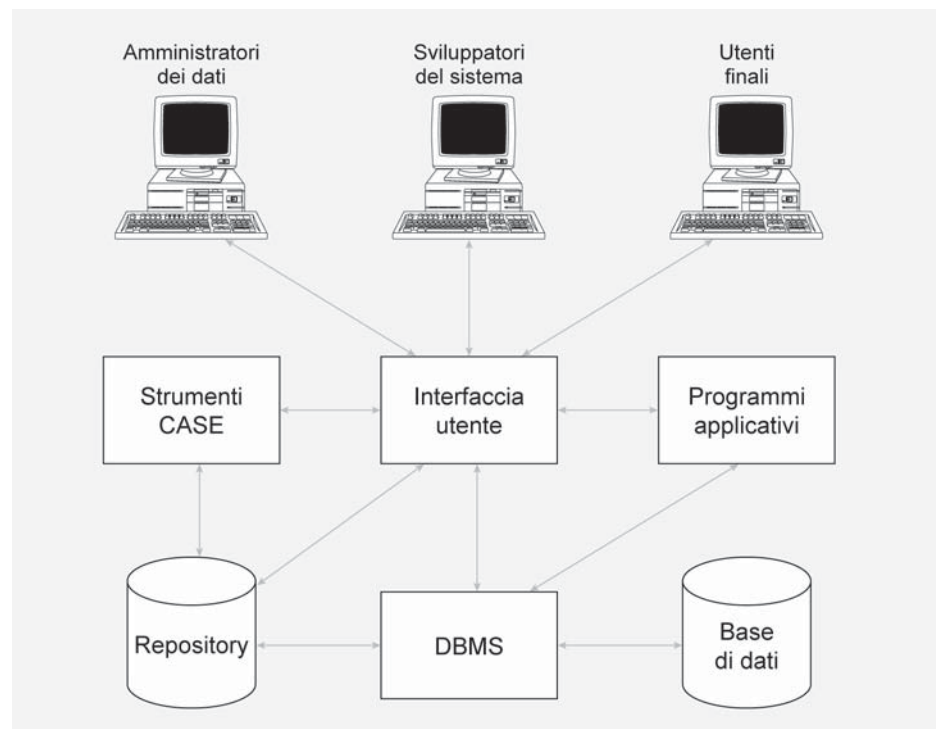
Una base di dati condivisa richiede un accordo sulle definizioni dei dati e la proprietà, oltre che sulle responsabilità della manutenzione accurata dei dati. L'esperienza ha mostrato che i conflitti sulle definizioni dei dati, sui formati e il codice, sui diritti di aggiornamento dei dati condivisi e su problemi correlati sono frequenti e spesso difficili da risolvere. La gestione di questi aspetti richiede un impegno organizzativo verso l'approccio a basi di dati, amministratori delle basi di dati scaltri dal punto di vista gestionale e un approccio solido ed evolutivo allo sviluppo delle basi di dati. Se all'approccio a basi di dati manca il supporto del top management e se vi sono dedicati pochi sforzi, si rischia di assistere a un proliferare di basi di dati indipendenti sviluppate dagli utenti finali. Queste basi di dati non seguono l'approccio generale che è stato descritto e non forniscono i vantaggi precedentemente illustrati.

3.7 Componenti dell'ambiente della base di dati

I componenti principali di un tipico ambiente di base di dati e delle relazioni tra di essi sono mostrati nella Figura 3.10. Alcuni di essi sono già stati presentati nei paragrafi precedenti. Quella che segue è una breve descrizione dei nove componenti mostrati nella Figura 3.10.

1. **Strumenti CASE (*Computer-Aided Software Engineering*).** Gli strumenti automatizzati utilizzati per progettare le basi di dati e i programmi applicativi. Descriveremo l'uso degli strumenti CASE per la progettazione e lo sviluppo delle basi di dati lungo tutto il testo.

Figura 3.10
Componenti dell'ambiente della base di dati.



2. **Repository.** Base di conoscenza centralizzata per tutte le definizioni dei dati, le associazioni, i formati delle schermate e dei report e altri componenti del sistema. Un repository contiene un insieme esteso di metadati importanti per la gestione delle basi di dati e di altri componenti di un sistema informativo.
3. **Sistema di gestione delle basi di dati (DBMS).** Un sistema software (e a volte anche hardware e firmware) commerciale utilizzato per definire, creare, mantenere e fornire un accesso controllato alla base di dati e al repository.
4. **Base di dati.** Una raccolta organizzata di dati correlati logicamente, solitamente concepiti per soddisfare le necessità di informazioni di più utenti in un'azienda. È importante distinguere tra la base di dati e il repository. Il repository contiene *definizioni* dei dati, mentre la base di dati contiene *occorrenze* dei dati. Descriveremo le attività di progettazione nei Capitoli 7 e 8 e l'implementazione nei Capitoli da 9 a 12.
5. **Programmi applicativi.** Programmi utilizzati per creare e mantenere la base di dati e fornire informazioni agli utenti. Le caratteristiche di programmazione fondamentali per una base di dati sono descritte nei Capitoli da 9 a 12.
6. **Interfaccia utente.** Linguaggi, menu e altri elementi in base ai quali gli utenti interagiscono con i vari componenti del sistema, come gli strumenti CASE, i programmi applicativi, il DBMS e il repository. Le interfacce utente vengono descritte lungo tutto il testo.
7. **Amministratori dei dati.** Le persone responsabili delle risorse informative complessive dell'azienda. Gli amministratori dei dati utilizzano gli strumenti CASE per migliorare la produttività della progettazione e della pianificazione della base di dati.
8. **Sviluppatori del sistema.** Persone come gli analisti di sistema e i programmatori che progettano nuovi programmi applicativi. Utilizzano spesso gli strumenti CASE per l'analisi dei requisiti del sistema e per la progettazione dei programmi.
9. **Utenti finali.** Le persone dell'azienda che aggiungono, eliminano e modificano i dati nella base di dati e che richiedono o ricevono informazioni da essa. Tutte le interazioni degli utenti con la base di dati devono passare attraverso il DBMS.

Con i miglioramenti del software, l'interfaccia utente diventa sempre più user-friendly. Esempi di questi miglioramenti sono i sistemi basati su menu, i sistemi abilitati per il Web, l'uso del mouse e i sistemi di riconoscimento vocale. Questi sistemi promuovono l'uso del computer da parte degli utenti finali, che anche se non sono esperti di computer, possono quindi impostare i propri report, le proprie visualizzazioni e le applicazioni più semplici. Naturalmente, in questo tipo di ambiente l'amministrazione della base di dati deve garantire l'applicazione di misure di sicurezza adeguate per proteggere la base di dati.

Riassumendo, l'ambiente operativo DBMS mostrato nella Figura 3.10 è un sistema integrato di hardware, software e persone concepito per facilitare la memorizzazione, il recupero e il controllo delle risorse informative e per migliorare la produttività dell'azienda.

3.8 Evoluzione dei sistemi di basi di dati

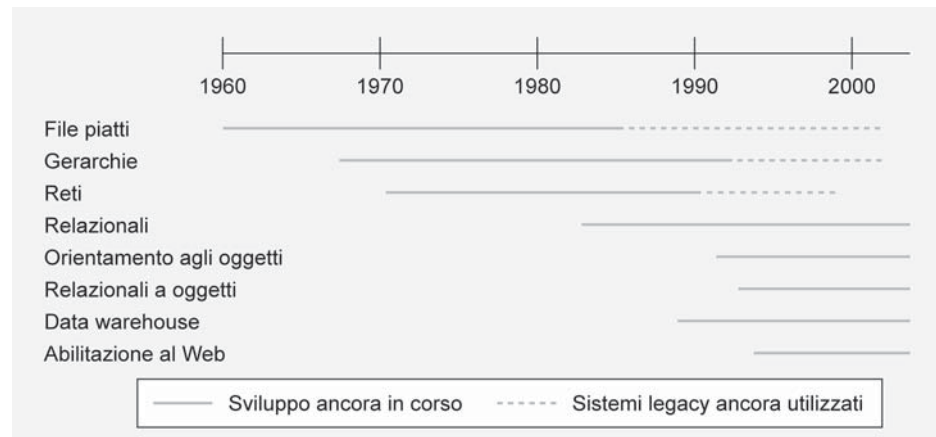
I sistemi di gestione a basi di dati sono stati introdotti per la prima volta negli anni '60 e hanno continuato a evolversi nei decenni seguenti. La Figura 3.11 delinea questa evoluzione evidenziando le tecnologie di basi di dati in primo piano in ciascun decennio. Nella maggior parte dei casi, il periodo di introdu-

zione è stato piuttosto lungo e la tecnologia è stata introdotta nel decennio precedente a quello mostrato nella figura. Per esempio, il modello relazionale è stato definito per la prima volta da E. F. Codd, un ricercatore di IBM, in una pubblicazione del 1970 (Codd, 1970). Il modello relazionale non ha tuttavia avuto un vasto successo commerciale prima degli anni '80.

I sistemi di gestione a basi di dati sono stati sviluppati per superare i limiti dei sistemi di elaborazione dei file, descritti nel paragrafo precedente. Per riassumere, lo sviluppo e l'evoluzione della tecnologia delle basi di dati sono sempre stati guidati da una combinazione dei tre obiettivi seguenti.

1. La necessità di fornire una maggiore indipendenza tra i programmi e i dati riducendo i costi di manutenzione.
2. Il desiderio di gestire strutture e tipi di dati sempre più complessi.
3. Il desiderio di fornire un accesso sempre più facile e veloce ai dati agli utenti che non hanno conoscenze di linguaggi di programmazione né una comprensione dettagliata di come i dati vengono memorizzati in una base di dati.

Figura 3.11
Evoluzione delle tecnologie
delle basi di dati.



3.8.1 1960

In questo periodo i sistemi di elaborazione a file erano ancora dominanti. I primi sistemi di gestione delle basi di dati furono tuttavia introdotti in questo decennio ed erano utilizzati soprattutto in imprese particolarmente impegnative e complicate, come il progetto di allunaggio dell'Apollo. Si può guardare a questi anni come a un periodo sperimentale, in cui si è dimostrata la fattibilità della gestione di grandi quantità di dati con un DBMS. I primi sforzi di creazione di standard furono intrapresi proprio alla fine del decennio grazie al Data Base Task Group.

3.8.2 1970

Durante questo decennio l'uso dei DBMS diventò una realtà commerciale. I sistemi di gestione gerarchici e di rete furono ampiamente sviluppati con strutture di dati sempre più complesse molto difficili da gestire con i metodi convenzionali di elaborazione dei file. I modelli gerarchici e di rete sono gene-

ralmente considerati come i DBMS di prima generazione. Entrambi gli approcci erano ampiamente utilizzati e molti di tali sistemi continuano a essere utilizzati ancora oggi. Ci sono tuttavia alcuni evidenti svantaggi.

1. Accesso difficile ai dati, basato sulle procedure di navigazione di un record alla volta. I programmi devono quindi essere scritti per rispondere a interrogazioni ancora molto semplici.
2. Indipendenza dei dati molto limitata, con i programmi ancora legati alle modifiche ai formati dei file.
3. Nessun fondamento teorico del modello ampiamente accettato, a differenza del modello di dati relazionale.

3.8.3 1980

Per superare questi limiti, E. F. Codd e altri svilupparono negli anni '70 il modello di dati relazionale. Questo modello, considerato il DBMS di seconda generazione, ebbe un'accoglienza commerciale e un'ampia diffusione durante gli anni '80. Con il modello relazionale, tutti i dati vengono rappresentati sotto forma di tabelle. Un linguaggio relativamente semplice di quarta generazione chiamato SQL (*Structured Query Language*) viene utilizzato per il recupero dei dati. Il modello relazionale fornisce così una facilità d'accesso per i non programmatori, superando una delle obiezioni principali ai sistemi di prima generazione.

3.8.4 1990

Gli anni '90 hanno introdotto una nuova era nel campo dei computer, innanzitutto con la tecnologia client/server e poi con i data warehouse e le sempre più importanti applicazioni Internet. Laddove negli anni '80 i dati gestiti dai DBMS erano ampiamente strutturati (come nel caso dei dati contabili), negli anni '90 si assiste a una diffusione sempre più crescente di dati multimediali, comprese immagini, audio e video. Per fronteggiare questi dati sempre più complessi, alla fine degli anni '80 (Grimes, 1998) vennero introdotte le basi di dati orientate agli oggetti (considerate di terza generazione).

Poiché le aziende devono gestire grandi quantità di dati sia strutturati sia non strutturati, le basi di dati relazionali e orientate agli oggetti rivestono oggi un'enorme importanza, tanto che alcuni produttori stanno sviluppando DBMS combinati relazionali-a oggetti che possano gestire entrambi i tipi di dati. Descriveremo questo tipo di basi di dati nell'Appendice D.

3.8.5 2000 e oltre

È naturale cercare di ipotizzare quali direzioni prenderà la tecnologia delle basi di dati nel prossimo decennio. Sebbene si avranno indubbiamente diverse sorprese, possiamo aspettarci la prosecuzione di alcune tendenze ormai definite.

1. La capacità di gestire tipi di dati sempre più complessi. Questi tipi includono i dati multidimensionali, che hanno già assunto un'importanza considerevole nelle applicazioni di data warehouse. Tratteremo i dati multidimensionali nel Capitolo 12.
2. Lo sviluppo continuato dei "server universali". Basati sui DBMS relazionali-a oggetti, sono server di basi di dati che possono gestire un'ampia gamma di tipi di dati in modo trasparente per gli utenti. Diventeranno particolarmente importanti per le applicazioni Internet.

3. Sebbene le basi di dati completamente distribuite siano ormai diventate una realtà, proseguirà comunque la tendenza attuale verso la centralizzazione delle basi di dati. Con la diminuzione dei costi delle comunicazioni a fronte dell'aumento dei dati, il costo per trovare e accedere a dati centralizzati sta subendo un calo. Anche il minor costo dell'elaborazione a prestazioni elevate incoraggia la centralizzazione.
4. La memorizzazione in base al contenuto diventerà sempre più diffusa. Con questo approccio, gli utenti possono recuperare i dati specificando quali dati desiderano invece che come recuperarli. Per esempio, un utente può effettuare la scansione di una fotografia ed eseguire una ricerca dell'elemento che più corrisponde a quella foto.
5. Le basi di dati e le altre tecnologie, come l'intelligenza artificiale e i servizi informativi simili a quelli televisivi, renderanno l'accesso alle basi di dati più facile per gli utenti che non hanno una formazione specifica. Per esempio, gli utenti potranno richiedere i dati in un linguaggio più naturale, e la tecnologia delle basi di dati anticiperà le esigenze di dati degli utenti in base alle interrogazioni precedenti e alle modifiche importanti avvenute nella base di dati.
6. Il lavoro di sviluppo di algoritmi di data mining che possano gestire insieme di dati molto ampi farà sì che le aziende possano analizzare in modo efficiente i propri enormi archivi. La capacità migliorata di individuare pattern, tendenze e correlazioni tra clienti, dipendenti, prodotti e fornitori influenzerà le decisioni strategiche. Questa capacità verrà applicata anche alla grossa mole di dati raccolta dall'attività del sito Web, come nel caso dell'analisi clickstream (Riggs, 2000).
7. All'altra estremità della scala, il proliferare di PDA porterà a un miglioramento della sincronizzazione delle basi di dati più piccole e delle frequenze delle trasmissioni wireless. Lo standard wireless Bluetooth accelererà enormemente lo sviluppo di PDA wireless che si connettono a Internet. Questo sviluppo accentuerà l'importanza della protezione e della sicurezza dei dati in un mondo sempre più "senza fili".

Riepilogo

Gli ultimi due decenni hanno visto una crescita esponenziale del numero e dell'importanza delle basi di dati. Le basi di dati sono utilizzate per memorizzare, manipolare e recuperare i dati in qualsiasi tipo di azienda. Nell'ambiente fortemente competitivo degli anni 2000, vi sono diversi segnali del fatto che la tecnologia a base di dati assumerà un'importanza ancora maggiore. Un corso sulla gestione delle basi di dati moderne è uno dei più importanti in un curriculum sui sistemi informativi.

Una base di dati è una raccolta organizzata di dati correlati. I dati sono definiti come fatti noti o oggetti che hanno un significato nell'ambiente dell'utente. Il termine "organizzata" indica che i dati sono strutturati in modo da poter essere facilmente memorizzati, manipolati e recuperati dagli utenti. Il termine "correlati" indica che i dati descrivono un campo di interesse per un gruppo di utenti e che quegli utenti possono utilizzare i dati per rispondere a domande riguardanti quel contesto. Le informazioni sono dati che sono stati collocati in un contesto o che sono stati elaborati e presentati in una forma adatta all'interpretazione umana.

I metadati sono dati che descrivono le proprietà o le caratteristiche di altri dati. Un sistema di gestione delle basi di dati è un sistema di software commerciale general-purpose utilizzato per definire, creare, mantenere e fornire un accesso controllato alla base di dati. Un DBMS memorizza i metadati in un repository, che è un luogo centrale dove sono raccolte tutte le definizioni dei dati, le relazioni tra i dati, i formati delle schermate e dei report e altri componenti di sistema.

I sistemi di elaborazione a file sono stati sviluppati all'inizio dell'era informatica perché i computer potessero memorizzare, manipolare e recuperare grandi file di dati. Questi sistemi, a tutt'oggi ancora in uso, hanno alcuni limiti importanti, come la dipendenza tra programmi e dati, la duplicazione dei dati, la condivisione limitata dei dati e i tempi di sviluppo prolungati. L'approccio a basi di dati è stato sviluppato per superare questi limiti. Questo approccio enfatizza l'integrazione e la condivisione dei dati nell'azienda. I suoi vantaggi includono l'indipendenza dei dati dai programmi, la condivisione migliorata dei dati, la ridondanza minima dei dati e una produttività migliorata dello sviluppo delle applicazioni.

Le applicazioni a basi di dati possono essere raggruppate nelle seguenti categorie: basi di dati personali, di gruppo di lavoro, di funzione e Internet. Oggigiorno le basi di dati aziendali includono i data warehouse, ossia basi di dati integrate di supporto alle decisioni il cui contenuto è tratto dalle varie basi di dati operazionali. I sistemi ERP si basano sulle basi di dati aziendali.

La tecnologia delle basi di dati si è evoluta costantemente fin dalla sua introduzione, avvenuta negli anni '60. Gli obiettivi principali che hanno guidato questa evoluzione hanno incluso il desiderio di ottenere una maggiore indipendenza dei dati dai programmi, la necessità di gestire strutture di dati sempre più complesse e il bisogno di fornire un accesso più veloce e semplice a tutti gli utenti. Nella fase attuale, le basi di dati relazionali a oggetti vengono sempre più utilizzate per soddisfare questi obiettivi.

Ripasso del capitolo

Parole chiave

Applicazione a base di dati	Repository	Metadati	Sistemi ERP (<i>Enterprise Resource Planning</i>)
Base di dati	Extranet	Modello di dati aziendale	Vincolo
Data warehouse	Indipendenza dei dati	Sistema di gestione delle basi di dati (DBMS)	Vista utente
Dati	Informazioni		
Dati legacy	Intranet		

Domande di ripasso

1. Definire ciascuno dei seguenti termini chiave:

 - a. dati
 - b. informazioni
 - c. metadati
 - d. applicazione di base di dati
 - e. data warehouse
 - f. vincolo
 - g. base di dati
 - h. dati legacy

2. Unire i seguenti termini alle rispettive definizioni:

___ dati; ___ applicazione di base di dati; ___ vincolo;

___ repository; ___ metadati; ___ data warehouse;

___ informazioni; ___ vista utente; ___ base di dati.

 - a. dati collocati in un contesto o riepilogati
 - b. programmi applicativi
 - c. fatti, testi, immagini e così via
 - d. base di dati integrata di supporto alle decisioni
 - e. raccolta organizzata di dati correlati
- f. include definizioni di dati e vincoli
 - g. luogo centralizzato di raccolta di tutte le definizioni di dati
 - h. separazione delle descrizioni dei dati dai programmi
 - i. regola che non può essere violata
 - j. descrizione logica di una parte della base di dati

3. Confrontare i seguenti termini:

 - a. dipendenza dei dati; indipendenza dei dati
 - b. data warehouse; data mining
 - c. dati; informazioni
 - d. repository; base di dati
 - e. entità; modello di dati aziendale
 - f. data warehouse; sistema ERP

4. Elencare e descrivere brevemente cinque categorie di basi di dati e fornire un esempio di ciascuna.

5. Perché la definizione di "dati" è stata estesa negli ambienti di basi di dati moderni?

6. Elencare i nove componenti principali di un ambiente di sistema di base di dati.

7. Come vengono espresse le relazioni tra tabelle in una base di dati relazionale?
8. Elencare alcune domande chiave a cui si deve rispondere per i seguenti tipi di basi di dati: personale, di gruppo di lavoro, di funzione e Internet.
9. Cosa significa l'espressione "indipendenza dei dati" e perché è un obiettivo importante?
10. Elencare sei vantaggi principali dell'approccio a basi di dati rispetto ai sistemi di file convenzionali.
11. Elencare cinque costi supplementari o rischi associati all'approccio a basi di dati.
12. Per ciascun decennio dal 1960 al 1990, elencare le tecnologie delle basi di dati predominanti. Indicare il risultato generato associato a ogni tecnologia.
13. Con il migliorare della capacità di gestire grandi quantità di dati, descrivere tre ambiti aziendali nei quali vengono utilizzate basi di dati estese.

Problemi ed esercizi

1. Per ciascuno dei casi seguenti, indicare se, in situazioni tipiche di un'università, esiste un'associazione uno-a-molti o molti-a-molti. Utilizzando la notazione introdotta nel testo, disegnare un diagramma per ciascuna delle associazioni.
 - a. STUDENT e COURSE (gli studenti si iscrivono ai corsi)
 - b. BOOK e BOOK COPY (i libri hanno delle copie)
 - c. COURSE e SECTION (i corsi hanno dei seminari)
 - d. SECTION e ROOM (i seminari si tengono in alcune aule)
 - e. INSTRUCTOR e COURSE
2. Tenendo come riferimento la Figura 3.4, rispondere alle seguenti domande:
 - a. Quali ordini risultano insoluti per Customer_ID 6? (Fornire Order_ID e Order_Date.)
 - b. Quali prodotti sono inclusi in Order_ID 2? (Fornire Product_ID e Quantity per ciascun elemento.)
 - c. Quali prodotti sono inclusi in Order_ID 2? (Fornire Product_ID, Product_Name e Unit_Price per ciascun elemento.)
3. Esaminare la base di dati personale dei contatti dei clienti (Figura 3.7). L'associazione tra CUSTOMER e CONTACT HISTORY è di tipo uno-a-molti o molti-a-molti? Utilizzando la notazione introdotta nel testo, disegnare un diagramma che mostri questa associazione.
4. Rileggere la definizione data in questo capitolo per "dati" e "base di dati". I sistemi di gestione delle basi di dati hanno incluso solo di recente la capacità di memorizzare e recuperare più dati numerici e di testo. Quali capacità di memorizzazione, recupero e manutenzione richiedono le immagini, l'audio, il video e altri tipi di dati avanzati che non sono richiesti o che sono più semplici con i dati numerici e di testo?
5. La Tabella 3.1 mostra alcuni metadati di esempio per un insieme di elementi di dati. Identificare altre tre colonne di questi dati (ossia altre tre caratteristiche di metadati per gli attributi elencati) e completare le immissioni della Tabella 3.1 con queste tre colonne supplementari.
6. Nel paragrafo "Svantaggi dei sistemi tradizionali di elaborazione dei file", si dice che gli svantaggi dei sistemi di elaborazione dei file possono costituire un limite anche alle basi di dati, a seconda di come l'azienda gestisce le sue basi di dati. Innanzitutto, perché le aziende creano basi di dati multiple e non un'unica base di dati che supporta tutte le esigenze di elaborazione dei dati? In secondo luogo, quali fattori aziendali e personali possono portare un'azienda ad avere più basi di dati gestite indipendentemente (e quindi, non seguendo completamente l'approccio a basi di dati)?
7. Si consideri un club o un'associazione di studenti di cui si è membri. Quali sono le entità di dati di questa associazione? Elencare e definire ciascuna entità. Sviluppare quindi un modello di dati aziendale (come quello della Figura 3.3) che mostri queste entità e le associazioni più importanti tra di esse.
8. Nel paragrafo "I tipi di applicazioni a base di dati" sono presentate cinque diverse categorie di basi di dati. Per ciascuna categoria, vengono delineate alcune decisioni chiave. Successivamente nel testo si vedrà come affrontare queste e altre decisioni; per ora si dovrebbe essere in grado di prevedere risposte parziali a queste decisioni dalla lettura di questo capitolo e grazie alla propria familiarità con la tecnologia informatica. Per le categorie e decisioni elencate di seguito, spiegare quale scelta effettuare o quali fattori dovrebbero essere considerati nella scelta in qualsiasi tipo di azienda.
 - a. *Base di dati personale.* Chi è responsabile dell'accuratezza dei dati in una base di dati personale?
 - b. *Base di dati di gruppo di lavoro.* Quali operazioni di elaborazione della base di dati dovrebbero essere eseguite dalla stazione di lavoro o quali dovrebbero avvenire sul server?
 - c. *Base di dati di funzione.* Come si può fornire una protezione adeguata dalle rivelazioni o distribuzioni non autorizzate dei dati sensibili?
 - d. *Base di dati aziendale.* Come l'azienda deve sviluppare e mantenere gli standard relativi ai nomi dei dati, alle definizioni, ai formati e a quanto correlato?
 - e. *Base di dati Internet, intranet o extranet.* Con gli applet e i moduli di codice su server e browser diversi, come possono tutti i componenti avere una comprensione condivisa del significato dei dati?
9. Una delle sfide maggiori nella creazione di aziende dot.com di commercio elettronico è stata quella di stabilire la possibilità dei clienti di ricevere la merce che ordinano rapidamente sul Web. Da quanto appreso finora riguardo alle basi di dati e in base alla propria esperienza sul Web, si può capire perché le società stanno passando alle soluzioni di basi di dati per avere un aiuto nella gestione della catena delle forniture e nella velocizzazione del completamento e dell'evasione degli ordini?

Esercitazioni

1. Scegliere un'azienda con un sistema informativo piuttosto esteso e un set di applicazioni di sistemi informativi. Verificare se l'azienda segue un approccio di elaborazione dei file o uno a base di dati per l'organizzazione dei dati. Quante basi di dati ha l'azienda? Provare a disegnare uno schema, simile a quella della Figura 3.2, per rappresentare tutti i file e le basi di dati aziendali.
2. Per la stessa azienda dell'esercitazione precedente o per un'altra, parlare con un amministratore o un progettista di basi di dati. Quale tipo di metadati l'azienda mantiene nelle proprie basi di dati? Perché l'azienda ha deciso di tenere una traccia di questi metadati e non di altri? Quali strumenti vengono utilizzati per mantenere questi metadati?
3. Per la stessa azienda delle due esercitazioni precedenti o per un'altra, identificare, se possibile, una base di dati per ciascuno dei cinque tipi elencati nel capitolo: personale, di gruppo di lavoro, di funzione, aziendale e Internet. L'azienda ha un data warehouse? Se sì, come viene creato dai dati di tutte le altre basi di dati? L'azienda utilizza un sistema ERP?
4. Per la stessa azienda delle tre esercitazioni precedenti, determinare l'uso di processi aziendali intranet, extranet o abilitati per il Web dell'azienda. Per ciascun processo, determinare lo scopo e il sistema di gestione della base di dati che viene utilizzato in combinazione con le reti. Chiedere quali sono i piani dell'azienda per l'anno successivo rispetto all'uso di intranet, extranet e del Web nelle proprie attività. Chiedere quali nuove competenze sono necessarie per implementare questi piani.
5. Potrebbe essere utile tenere un diario personale delle idee e delle osservazioni sulla gestione della base di dati mentre si studia su questo libro. Utilizzare questo diario per registrare i commenti ascoltati, i riepiloghi delle nuove storie o gli articoli specialistici letti, le idee originali o le ipotesi, gli URL e i commenti sui siti Web relativi alle basi di dati e le domande che necessitano di un'analisi più approfondita. Si presti attenzione a qualsiasi cosa riguardi la gestione della base di dati. L'insegnante può chiedere di vedere di tanto in tanto una copia del diario per fornire dei riscontri. Il diario è un insieme non strutturato di note personali che integra gli appunti presi in classe e che può aiutare a pensare su altri argomenti oltre quelli trattati nei limiti di tempo della maggior parte dei corsi.

Bibliografia

- Ballou, D. P. e Thay, G. K., 1999, "Enhancing Data Quality in Data Warehouse Environments", *Communications of the AMC* 42 (gennaio), pp. 73–78.
- Codd, E. F., 1970, "A Relational Model of Data for Large Relational Databases", *Communications of the AMC* 16 (giugno), pp. 377–378.
- Grimes, S., 1999, "Object/Relational Reality Check", *Database Programming and Design*, 11, 7 (luglio), pp. 26–33.
- Jordan, A., 1996, "Data Warehouse Integrity: How Long and Bumpy the Road?", *Data Management Review* 6, 3 (marzo), pp. 35–37.
- Lambert, B., 1996, "Data Warehousing Fundamentals: What You Need to Succeed", *Data Management Review* 6, 3 (marzo), pp. 24–30.
- Riggs, S., "Collecting Webdata", *Teradata Review* 3, 1 (primavera), pp. 12–19.
- Ritter, D., "Don't Neglect Your Legacy", *Intelligent Enterprise* 2, 5 (30 marzo), pp. 70, 72.
- White, C., 1995, "Database Technology: Sorting Out the Options", *Supplement to Database Programming & Design*, 8, dicembre, pp. 41–44, 46.
- Winter, R., 1997, "What, After All, Is a Very Large Database?", *Database Programming & Design*, 10, 1, gennaio, pp. 23–26.

Per approfondire

- Date, C. J., "The Birth of the Relational Model, Part 3", *Intelligent Enterprise* 1, 4 (dicembre), pp. 45–48.
- Hoffer, J. A., George, J. F. e Valacich, J. S., 2002, *Modern Systems Analysis and Design*, terza edizione, Upper Saddle River, NJ, Prentice Hall.
- Ritter, D., 1999, "The Long View", *Intelligent Enterprise* 2, 12 (24 agosto), pp. 58, 63, 67.

Project Case



MOUNTAIN VIEW COMMUNITY HOSPITAL

Introduzione

Questo project case fornisce una possibilità di applicare i concetti e le tecniche appresi nel capitolo. Abbiamo scelto il caso di un ospedale perché è un tipo di azienda con cui la maggior parte delle aziende ha familiarità e perché le istituzioni sanitarie rivestono molta importanza nella società odierna. Una parte del progetto viene inclusa al termine di ogni capitolo del libro. Ogni segmento include una breve descrizione del caso per quanto fa riferimento al materiale del capitolo seguito dalle rispettive domande e dalle esercitazioni.

Descrizione del progetto

Mountain View Community Hospital è un ospedale generico non profit per malattie gravi a degenza breve. È un ospedale relativamente piccolo, attualmente con 150 posti letto. Il suo scopo è quello di fornire assistenza medica di alta qualità alla comunità che lo circonda contenendo i costi che sono cresciuti in base alle tendenze nazionali negli anni recenti.

Mountain View Community Hospital ha applicazioni informatiche che supportano le seguenti aree di intervento (tra le altre): amministrazione delle cure, servizi clinici, gestione dei fondi e servizi amministrativi. La maggior parte di queste applicazioni è stata acquistata da fornitori esterni, mentre alcune applicazioni sono state sviluppate internamente. Le applicazioni si basano sull'uso della tecnologia client/server, come illustrato nella Figura 3.5. Molte delle applicazioni vengono implementate utilizzando la tecnologia a basi di dati relazionali. Gran parte delle basi di dati, così come tutte le applicazioni, sono di funzione (in base alla classificazione introdotta in questo capitolo).

Le basi di dati relazionali di Mountain View Community Hospital contengono diverse tabelle. Due di queste, con alcuni dati di esempio, vengono mostrate nella Figura 1. La tabella PATIENT contiene dati che riguardano i pazienti attuali o passati, mentre la tabella PATIENT CHARGES contiene i dati sulle cure prescritte ai pazienti. In questo capitolo tali tabelle sono state semplificate, ma verranno ampliate nei prossimi.

Domande sul progetto

1. Quali sono alcuni dei vantaggi che Mountain View Community Hospital dovrebbe cercare utilizzando le basi di dati? Per quanto possibile, fare riferimento all'ambiente dell'ospedale.
2. Quali sono alcuni dei costi e dei rischi dell'utilizzo delle basi di dati che devono essere gestiti con maggiore attenzione dall'ospedale?
3. Attualmente Mountain View Community Hospital utilizza la tecnologia a basi di dati relazionali. Sebbene questa tecnologia sia opportuna per dati strutturati come i nomi dei pazienti e i dati della contabilità, è più adatta per dati complessi con informazioni grafiche e immagini. Quali possono essere alcuni tipi di dati mantenuti da un ospedale che rientrano nell'ultima categoria? Quali tipi di tecnologie di basi di dati possono essere più adatti a questi tipi di dati invece di quelle relazionali?
4. Come sono correlati i dati nelle tabelle PATIENT e PATIENT CHARGES? Nel concreto, come può un utente trovare le cure importanti per un paziente specifico?
5. Quali sono i modi in cui un ospedale può utilizzare Internet?

Esercizi sul progetto

1. Utilizzando la notazione introdotta in questo capitolo, disegnare un diagramma che mostri l'associazione tra PATIENT e PATIENT CHARGES.
2. Sviluppare un grafico di metadati (Tabella 3.1) per gli attributi di dati nelle tabelle PATIENT e PATIENT CHARGES. Utilizzare le colonne mostrate nella Tabella 3.1, ma includere le altre caratteristiche dei metadati che si ritengono appropriate per la gestione dei dati di Mountain View Community Hospital.
3. Una delle viste di base di dati importante per l'ospedale è Patient Bill. La Figura 2 è una versione estremamente semplificata di questa vista. Completare i dati mancanti in questa vista utilizzando i dati dalla Figura 1.
4. Utilizzando la notazione introdotta in questo capitolo, disegnare un singolo diagramma che rappresenta le seguenti associazioni nell'ambiente dell'ospedale.

Figura 1
Due tabelle di base di dati.

PATIENT		
Patient Name	Patient Number	Patient Address
Dimas, Salena	8379	617 Valley Vista
Dolan, Mark	4238	818 River Run
Larreau, Annette	3047	127 Sandhill
Wiggins, Brian	5838	431 Walnut
Thomas, Wendell	6143	928 Logan

PATIENT CHARGES			
Item Description	Item Code	Patient Number	Amount
Room Semi-Priv	200	4238	1600
Speech Therapy	350	3047	750
Radiology	275	4238	150
Physical Therapy	409	5838	600
EKG Test	500	8379	200
Room Semi-Priv	200	3047	800
Standard IV	470	8379	150
EEG Test	700	4238	200

Figura 2
Versione semplificata della vista
Patient Bill.

Patient Name: Dolan, Mark		
Patient Number: _____		
Patient Address: _____		
Item Code	Item Description	Amount
_____	_____	_____
_____	_____	_____
_____	_____	_____

- Un HOSPITAL ha nel suo personale uno o più PHYSICIAN. Un PHYSICIAN è nel personale di un solo HOSPITAL.
 - Un PHYSICIAN può ammettere uno o più PATIENT. Un PATIENT viene ammesso da un solo PHYSICIAN.
 - A ogni PATIENT possono essere prescritte più CHARGES. Una particolare CHARGE può essere prescritta a più PATIENT.
 - Un HOSPITAL ha uno o più WARD (reparto). Ogni WARD si trova esattamente in un HOSPITAL.
- Un WARD ha un certo numero di EMPLOYEE. Un EMPLOYEE può lavorare in uno o più WARD.
5. Lo scopo di Mountain View Community Hospital è quello di fornire assistenza medica di alta qualità alla comunità che lo circonda contenendo i costi che sono cresciuti in base alle tendenze nazionali negli anni recenti. Come può una base di dati ben gestita aiutare l'ospedale a raggiungere questo obiettivo? Fornire alcuni esempi di come l'uso delle basi di dati dell'ospedale può migliorare l'assistenza medica o contenere i costi.

CAPITOLO

4

Processo di sviluppo delle basi di dati

Al termine di questo capitolo sarete in grado di fare quanto segue.

1. Definire i seguenti termini chiave: modellazione dei dati aziendali, architettura dei sistemi informativi (ISA), ingegneria delle informazioni, progettazione top-down, funzioni aziendali, scomposizione funzionale, ciclo di vita dello sviluppo dei sistemi, prototipazione, strumenti CASE (*Computer-Aided Software Engineering*), repository, progetto, conferma progressiva (*incremental commitment*), schema concettuale, schema fisico e architettura client/server.
2. Descrivere il ciclo di vita di un progetto di sviluppo di un sistema, con un accento sullo scopo dell'analisi della base di dati sull'attività di progettazione e le attività di implementazione.
3. Spiegare l'approccio a prototipi per lo sviluppo delle basi di dati e delle applicazioni.
4. Spiegare i ruoli dei singoli individui che progettano, implementano, utilizzano e amministrano le basi di dati.
5. Spiegare la differenza tra schemi concettuali, esterni e fisici e descrivere le ragioni di un'architettura a tre schemi per le basi di dati.
6. Spiegare l'architettura a tre livelli nell'ambito della progettazione delle basi di dati e l'elaborazione di queste ultime.
7. Spiegare l'obiettivo di un progetto di ideazione e sviluppo di una base di dati.
8. Disegnare semplici modelli di basi di dati per illustrare l'obiettivo di un "database".

Introduzione

Il Capitolo 3 ha introdotto l'approccio a basi di dati per i sistemi informativi e un ambiente applicativo delle basi di dati rintracciabile in molte aziende. È stata utilizzata un'azienda d'esempio, Pine Valley Furniture Company, per illustrare molti dei principi e dei concetti di questo approccio. Lo stesso esempio verrà utilizzato anche in questo e nei prossimi capitoli come guida per l'applicazione della gestione delle basi di dati.

Il Capitolo 4 presenta una panoramica dei passaggi generali da seguire nell'analisi, progettazione, implementazione e amministrazione di una base di dati. Poiché una base di dati è una parte di un sistema informativo, in questo capitolo si vedrà come il processo di sviluppo di una base di dati si adatti a quello complessivo del sistema informativo. Il capitolo sottolinea la necessità di coordinare lo sviluppo

della base di dati con tutte le altre attività nello sviluppo di un sistema informativo completo ed esamina alcuni aspetti salienti di un ipotetico processo di sviluppo di una base di dati per Pine Valley Furniture Company. Vengono introdotti gli strumenti per lo sviluppo di basi di dati su personal computer e il processo di estrazione dei dati dalle basi di dati dell'azienda per un uso nelle applicazioni indipendenti.

Ci sono molte ragioni per trattare lo sviluppo delle basi di dati in questo punto del testo. Innanzitutto, sebbene sia possibile che siano già state utilizzate le funzioni fondamentali di un sistema di gestione di basi di dati, come Microsoft Access®, probabilmente non si è ancora colto appieno come viene sviluppata una base di dati. Lo scopo principale di questo libro è quello di introdurre i concetti e molte delle tecniche necessarie per progettare e creare applicazioni a basi di dati. Utilizzando esempi semplici, il capitolo illustra brevemente cosa si sarà in grado di fare dopo aver completato un corso sulle basi di dati utilizzando questo testo. Il capitolo aiuta quindi a sviluppare una visione e un contesto per ciascun argomento che verrà sviluppato nel dettaglio nei capitoli successivi.

In secondo luogo, molti studenti imparano meglio con un testo ricco di esempi concreti. Sebbene tutti i capitoli di questo libro contengano numerosi esempi, illustrazioni, progetti e codici reali per la gestione di una base di dati, ogni capitolo si concentra su un aspetto specifico. Questo capitolo, per esempio, è stato concepito per aiutare a capire la correlazione esistente tra i singoli aspetti della gestione di una base di dati. Sebbene il dettaglio tecnico sia ridotto al minimo, si spera di stimolare l'interesse verso la modellazione dei dati e uno sviluppo delle competenze presentate nei capitoli successivi.

Infine, molti docenti preferiscono che gli studenti eseguano il prima possibile le prime fasi di un progetto di sviluppo di una base di dati, sia esso in un corso di gruppo o individuale. Grazie alla progressione logica degli argomenti trattati nel libro, occorrerà studiare parecchio prima di riuscire a vedere il progetto finale. Questo capitolo accenna come strutturare un progetto di sviluppo di base di dati in modo sufficiente a iniziare un'esercitazione del corso. Ovviamente, poiché siamo solo al Capitolo 4, molti degli esempi e delle notazioni utilizzati saranno più semplici di quelli necessari per il progetto, per le assegnazioni del corso o per un'azienda vera e propria.

Attenzione: con questo capitolo, che è stato volutamente mantenuto a un livello introduttivo e semplificato, non si imparerà a progettare e sviluppare le basi di dati. Molte delle notazioni utilizzate nel capitolo non sono esattamente identiche a quelle spiegate nei capitoli successivi. Lo scopo del Capitolo 4 è quello di offrire una comprensione generale dei passi chiave e del tipo di operazioni, non quello di insegnare tecniche specifiche. Sarà tuttavia possibile sviluppare un'intuizione e una motivazione per le tecniche e il contenuto presentati nei prossimi capitoli.

4.1 Sviluppo della base di dati all'interno dello sviluppo dei sistemi informativi

In molte aziende, lo sviluppo di una base di dati inizia con la modellazione dei dati aziendali, che stabilisce la gamma e il contenuto generale delle basi di dati dell'azienda. Questo passaggio si verifica solitamente durante la pianificazione dei sistemi informativi di un'azienda. Il suo scopo è quello di creare un'immagine complessiva o una spiegazione dei dati dell'azienda, e non di progettare una base di dati specifica. Quest'ultima fornisce i dati per uno o più sistemi informativi, mentre un modello di dati aziendale, che può comprendere più basi di dati, descrive l'ambito dei dati gestiti dall'azienda. Nella modellazione dei dati aziendali si rivedono i sistemi correnti, si analizza la natura dei settori aziendali da supportare, si descrivono i dati necessari con un livello elevato di astrazione e si pianificano uno o più progetti di sviluppo di basi di dati. La Figura 4.1, un duplicato della Figura 3.3, mostra un segmento di un modello di dati aziendale per Pine Valley Furniture Company che utilizza una versione semplificata della notazione che verrà introdotta nei Capitoli 5 e 6.

Figura 4.1

Segmento del modello di dati aziendale (Pine Valley Furniture Company).



4.1.1 Architettura informativa

Un modello di dati di alto livello, come quello mostrato nella Figura 4.1, è solo una parte di un'**architettura informativa (ISA)**, o *modello informativo aziendale (blueprint)*. Secondo Zachman (1987) e Sowa e Zackman (1992), un'architettura di sistemi informativi è costituita da sei componenti chiave.

1. *Dati* (che possono essere rappresentati come nella Figura 4.1 o in altri modi), alcuni dei quali sono illustrati nel paragrafo sulla pianificazione dei sistemi informativi.
2. *Processi* che manipolano i dati (possono essere rappresentati con diagrammi di flusso, modelli a oggetto con metodi o altre notazioni).
3. *Reti* che trasportano i dati nell'azienda e tra questa e i suoi partner principali (possono essere rappresentate con uno schema dei collegamenti e della topologia di rete).
4. *Persone* che eseguono i processi e che sono la fonte e i destinatari dei dati e delle informazioni (nei modelli di processo possono essere rappresentate come mittenti e destinatari dei dati).
5. *Eventi e punti nel tempo* in corrispondenza dei quali i processi vengono eseguiti (possono essere rappresentati con diagrammi di transizioni di stato o in altro modo).
6. *Ragioni* degli eventi e delle regole che governano l'elaborazione dei dati (sono spesso rappresentate in una forma testuale, ma anche sotto forma di diagrammi o di alberi decisionali).

4.1.2 Ingegneria delle informazioni

Un'architettura di sistemi informativi viene sviluppata da alcuni progettisti che seguono una particolare metodologia di progettazione. Una delle più formali e diffuse è l'ingegneria delle informazioni. **L'ingegneria delle informazioni** è una metodologia orientata ai dati per la creazione e la manutenzione dei sistemi informativi. A causa del suo orientamento, una breve spiegazione dell'ingegneria delle informazioni può essere utile per iniziare a capire come le basi di dati vengono identificate e definite. **L'ingegneria delle informazioni segue un approccio di progettazione top-down** (dall'alto verso il basso), nella quale i sistemi informativi specifici vengono dedotti da una comprensione generale delle esigenze

informative (per esempio, occorrono dati sui clienti, i prodotti, i fornitori, gli agenti di vendita, i centri di produzione e così via) invece che da un consolidamento di più richieste di informazioni (come una schermata di immissione per gli ordini o un riepilogo delle vendite in base al territorio). La progettazione top-down offre i vantaggi di una prospettiva ampia, e costituisce un modo per osservare l'integrazione dei singoli componenti del sistema e per capire il rapporto tra i sistemi informativi e gli obiettivi dell'azienda, e per capire l'impatto dei sistemi informativi sull'intera azienda.

L'ingegneria delle informazioni include quattro passaggi: pianificazione, analisi, progettazione e implementazione. La fase della pianificazione sfocia in un'architettura dei sistemi informativi, che comprende un modello dei dati aziendale. Esamineremo questa fase nel prossimo paragrafo. Sebbene questo libro non segua di per sé la metodologia dell'ingegneria delle informazioni, i capitoli successivi prendono in considerazione le attività che rientrano nelle altre tre fasi.

4.1.3 Pianificazione dell'architettura informativa

Lo scopo della pianificazione è quello di allineare l'architettura informativa alle strategie aziendali. Un allineamento di questo tipo è importante per poter trarre il massimo beneficio dall'investimento nei sistemi informativi e nelle loro tecnologie. Come illustrato nella Tabella 4.1, la fase di pianificazione della metodologia dell'ingegneria informativa include tre passaggi, che vengono analizzati nei paragrafi successivi.

Tabella 4.1 Fase di pianificazione nell'ingegneria informativa.

Passaggio	Spiegazione
1.	Identificazione dei fattori di pianificazione strategici a. Obiettivi b. Fattori critici di successo c. Aree problematiche
2.	Identificazione degli "oggetti" di pianificazione aziendale a. Unità organizzative b. Unità fisiche c. Funzioni aziendali d. Altri tipi di entità
3.	Sviluppo di un modello aziendale a. Scomposizione funzionale b. Diagramma entità-associazione c. Matrici di pianificazione

Identificare i fattori di pianificazione strategici

I fattori di pianificazione strategici sono gli obiettivi dell'azienda, i fattori critici di successo (CSF) e le aree problematiche. Lo scopo dell'identificazione di questi fattori è quello di sviluppare il contesto di pianificazione e di collegare i piani dei sistemi informativi a quelli strategici aziendali. La Tabella 4.2 mostra alcuni fattori di pianificazione strategici di Pine Valley Furniture Company. Questi fattori aiutano i manager dei sistemi informativi (nuove esigenze informative) a impostare le priorità per affrontare le esigenze dei nuovi sistemi informativi e quindi lo sviluppo delle basi di dati. Per esempio, la presenza di un'area problematica nella funzione vendite a causa di previsioni imprecise può far sì che i manager

richiedano l’inserimento nelle basi di dati dell’azienda dati storici di vendita supplementari, nuovi di dati di ricerca di mercato o dati dei risultati dei test sui nuovi prodotti.

Tabella 4.2 Esempi di risultati della fase di pianificazione nell’ingegneria informatica (Pine Valley Furniture Company).

Fattore di pianificazione	Esempi
Obiettivi	Mantenere il 10% di tasso di crescita annuale Mantenere il 15% di reinvestimento dei capitali Evitare il licenziamento dei dipendenti Essere un dipendente responsabile all'interno dell'azienda
Fattori critici di successo	Prodotti di alta qualità Consegne puntuali dei prodotti finiti Elevata produttività dei dipendenti
Aree problematiche	Previsioni di vendita imprecise Concorrenza crescente Esaurimento delle scorte di prodotti finiti

Identificare gli “oggetti” di pianificazione aziendale

Gli oggetti di pianificazione aziendale definiscono l’ambito dell’azienda. L’ambito pone dei limiti all’analisi di sistema successiva e ai punti nei quali possono avvenire delle modifiche nel sistema. Di seguito vengono riportati cinque oggetti di pianificazione fondamentali (la Tabella 4.3 ne contiene alcuni esempi per Pine Valley Furniture Company).

- *Unità dell’azienda*: le varie funzioni dell’azienda.
- *Unità fisiche*: i luoghi in cui avvengono le operazioni dell’azienda.
- *Funzioni aziendali*: gruppi correlati di processi aziendali che supportano la missione dell’azienda. Si noti che le funzioni aziendali sono diverse dalle unità dell’azienda; una funzione può essere assegnata a più di un’unità (per esempio, una funzione come lo sviluppo di un prodotto può essere di responsabilità congiunta dell’Ufficio vendite e della funzione Produzione).
- *Tipi di entità*: categorie principali di dati riguardanti persone, luoghi e cose gestiti dall’azienda.
- *Sistemi informativi*: il software applicativo e le procedure di supporto per la gestione degli insiemi di dati.

Sviluppo di un modello aziendale

Un modello aziendale articolato è costituito dai modelli funzionali di scomposizione, di ogni funzione aziendale, dal modello di dati aziendale e da varie matrici di pianificazione. La scomposizione funzionale è il processo di frammentazione delle funzioni di un’azienda in livelli di dettaglio sempre maggiore. È un processo classico utilizzato nell’analisi dei sistemi per semplificare i problemi, isolare l’attenzione e identificare i componenti.

Un esempio di scomposizione di una funzione di evasione degli ordini per Pine Valley Furniture Company è mostrato nella Figura 4.2. Spesso sono necessarie più basi di dati per gestire l’insieme globale delle funzioni aziendali e di supporto (per esempio, tutte le funzioni e sottofunzioni elencate nella Tabella 4.3), mentre una base di dati specifica può supportare solo un sottoinsieme di funzioni (per

Tabella 4.3 Oggetti di pianificazione aziendale di esempio (Pine Valley Furniture Company).

Oggetto di pianificazione	Esempi
Unità dell'organizzazione	Ufficio vendite
	Ufficio ordini
	Ufficio contabilità
	Produzione
Luoghi dell'organizzazione	Reparto fabbricazione
	Reparto assemblaggio
	Reparto rifiniture
	Sede centrale della società
	Durango Plant
Funzioni aziendali	Western Regional Sales Office
	Lumber Mill
	Pianificazione aziendale
	Sviluppo dei prodotti
	Gestione dei materiali
	Marketing e vendite
	Evasione degli ordini
	Consegne
	Riepilogo delle vendite
	Operazioni di produzione
Tipi di entità	Fondi e contabilità
	CUSTOMER (cliente)
	PRODUCT (prodotto)
	RAW MATERIAL (materiale grezzo)
	ORDER (ordine)
	WORK CENTER (centro di produzione)
	INVOICE (fattura)
	EQUIPMENT (attrezzatura)
	EMPLOYEE (impiegato)
	Sistemi di elaborazione delle transazioni
Sistemi informativi	Traccia degli ordini
	Elaborazione degli ordini
	Programmazione delle forniture
	Pagamenti
	Sistemi informativi di gestione
	Gestione delle vendite
	Controllo dell'inventario
	Programmazione della produzione

esempio, quelli mostrati nella Figura 4.2). È tuttavia utile avere una visione globale e ad alto livello per poter ridurre al minimo la ridondanza e rendere i dati pregnanti.

Un modello di dati aziendale viene descritto utilizzando una particolare notazione. La Figura 4.1 illustra una notazione generale che può essere tranquillamente utilizzata, sebbene uno dei formati più diffusi sia il diagramma entità-associazione, che viene spiegato nei Capitoli 5 e 6. Oltre a questa descrizione grafica dei tipi di entità, un modello di dati aziendale esauriente dovrebbe anche includere le descrizioni di ciascun tipo di entità e un compendio di varie definizioni di come l'azienda opera, chiamate *regole aziendali*, che governano la validità dei dati. Anche le regole aziendali vengono descritte nei Capitoli 5 e 6.

Figura 4.2

Esempio di un processo di scomposizione di una funzione di evasione degli ordini (Pine Valley Furniture Company).



Un modello di dati aziendale mostra non solo i tipi di entità ma anche le associazioni tra queste. Durante la modellazione dell'impresa vengono definite anche altre associazioni tra i vari oggetti della pianificazione. Un formato comune che mostra le interrelazioni tra gli oggetti è quello delle matrici. Le matrici di pianificazione svolgono una funzione importante, perché forniscono un approccio esplicito per descrivere le esigenze dell'azienda senza richiedere che la base di dati venga modellata esplicitamente. Originate spesso dalle regole aziendali, le matrici di pianificazione aiutano a impostare le priorità di sviluppo, a stabilire una sequenza delle attività di sviluppo e a programmare queste attività dalla vista top-down che deriva da un approccio su scala aziendale. Le matrici di pianificazione che si possono utilizzare sono numerose; alcune delle più comuni vengono riportate di seguito.

- *Luogo-a-funzione*: indica quali funzioni aziendali vengono eseguite in quali luoghi.
- *Unità-a-funzione*: identifica quali funzioni aziendali vengono eseguite o sono di responsabilità di quale unità aziendale.
- *Sistema informativo (applicazione)-a-entità di dati*: spiega come ciascun sistema informativo interagisce con ciascuna entità di dati (per esempio, se ciascun sistema crea, recupera, aggiorna o elimina i dati in ciascuna entità).
- *Funzione supportata-a-entità di dati*: identifica quali dati vengono individuati, utilizzati, aggiornati o eliminati in ciascuna funzione.
- *Sistema informativo (applicazione)-a-obiettivo*: mostra quale sistema informativo supporta ciascun obiettivo aziendale.

La Figura 4.3 illustra una possibile matrice funzione-a-entità di dati. Una matrice di questo tipo può essere utilizzata per vari scopi, compresi i seguenti.

1. *Identificare gli orfani*: indicare quali entità di dati non sono utilizzate da alcuna funzione o quali funzioni non utilizzano alcuna entità.
2. *Individuare le entità mancanti*: gli impiegati coinvolti in ciascuna funzione che esaminano la matrice sono in grado di identificare le entità che possono essere state omesse.
3. *Stabilire priorità di sviluppo*: se una data funzione ha una priorità alta per lo sviluppo dei sistemi (per esempio perché è correlata a obiettivi importanti dell'azienda), allora le entità utilizzate in questo contesto avranno una priorità alta nello sviluppo della base di dati.

Figura 4.3
Esempio di matrice funzione
di supporto-a-entità di dati.

Funzioni aziendali	Tipi di entità di dati								
	Cliente	Prodotti	Materiale grezzo	Ordine	Centro di produzione	Ordine di lavoro	Fattura	Attrezzatura	Dipendente
Pianificazione aziendale	X	X						X	X
Sviluppo della produzione		X	X		X			X	
Gestione dei materiali		X	X	X	X	X		X	
Evasione degli ordini	X	X	X	X	X	X	X	X	X
Consegna	X	X		X	X		X		X
Riepilogo delle vendite	X	X		X			X		X
Operazioni di produzione		X	X	X	X	X		X	X
Fondi e contabilità	X	X	X	X	X		X	X	X
X = l'entità di dati (colonna) è utilizzata nella funzione aziendale (riga)									

Si veda Hoffer, George e Valacich (2002) per una descrizione più articolata di come utilizzare le matrici di pianificazione nell'ingegneria delle informazioni e nella pianificazione dei sistemi.

4.2 Processo di sviluppo della base di dati

La pianificazione dei sistemi informativi, basata sull'ingegneria delle informazioni, è una delle fonti dei progetti di sviluppo di una base di dati. Tali progetti sviluppano nuove basi di dati per rispondere alle esigenze strategiche dell'azienda, come il supporto migliorato ai clienti, una gestione migliore della produzione e dell'inventario o previsioni di vendite più accurate. Molti progetti di sviluppo nascono tuttavia in una forma *bottom-up* (dal basso verso l'alto). In questo caso, i progetti sono richiesti dagli utenti dei sistemi informativi, che necessitano di determinate informazioni per compiere il proprio lavoro, o da altri professionisti dei sistemi informativi che riscontrano la necessità di migliorare la gestione dei dati dell'azienda. Anche nel caso bottom-up, la modellazione dei dati aziendali deve essere eseguita in ordine, così da capire se le basi di dati esistenti possono fornire i dati desiderati e, se no, quali basi di dati, entità di dati e attributi nuovi devono essere aggiunti alla risorsa di dati corrente dell'azienda.

Che derivi da esigenze informative operazionali o strategiche, ogni progetto di sviluppo di una base di dati si focalizza solitamente su una sola base di dati. Alcuni progetti si concentrano esclusivamente sulla definizione, progettazione e implementazione di una base di dati come fondamenta per lo sviluppo dei sistemi informativi successivi. Nella maggior parte dei casi, una base di dati e le funzioni di elaborazione delle informazioni associate vengono tuttavia sviluppate congiuntamente come parte di un unico progetto articolato.

4.2.1 Ciclo di vita dello sviluppo dei sistemi informativi (applicazioni e relative basi di dati)

Un procedimento tradizionale per la conduzione di un progetto di sviluppo dei sistemi informativi è chiamato **ciclo di vita dello sviluppo dei sistemi (SDLC)**. Si tratta di una serie completa di passaggi seguiti da una team di professionisti dei sistemi informativi, compresi i progettisti della base di dati e i programmatori, per specificare, sviluppare, mantenere e sostituire i sistemi informativi. Il processo viene spesso rappresentato come una “cascata”, come illustrato nella Figura 4.4 (Hoffer, George e Valacich, 2002). La metafora della cascata deriva dal fatto che ogni fase fluisce in quella successiva; la specifica totale del sistema informativo si sviluppa quindi in varie fasi, e il prodotto di una fase è utilizzato come input di quella successiva. Come mostrato nella figura, i passaggi non sono lineari: si sovrappongono nel tempo (alcuni di essi possono quindi essere condotti in parallelo) ed è possibile tenere una traccia dei passaggi precedenti qualora sia necessario riconsiderare alcune decisioni già prese (si può andare contro corrente!).

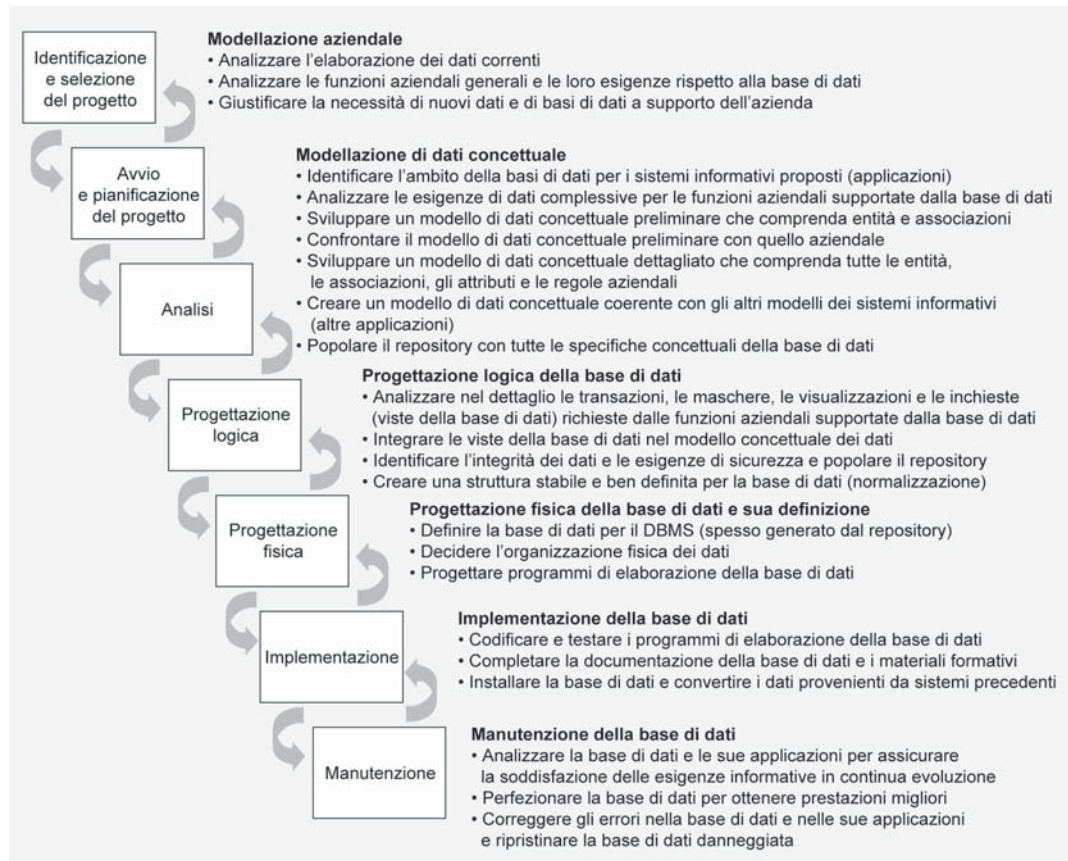
La Figura 4.4 contiene una breve spiegazione dello scopo e dei servizi risultanti da ogni fase SDLC. Poiché ogni fase SDLC include attività correlate allo sviluppo della base di dati, i problemi di gestione della base di dati sono presenti lungo tutto il processo di sviluppo. Gli stessi sette passaggi SDLC

Figura 4.4
Ciclo di vita dello sviluppo dei sistemi (SDLC).



Figura 4.5

Attività di sviluppo della base dei dati durante il ciclo di vita dello sviluppo dei sistemi (SDLC).



vengono ripetuti nella Figura 4.5, questa volta delineando le attività di sviluppo della base di dati solitamente incluse in ciascuna fase. Si noti che non esiste una corrispondenza uno-a-uno tra le fasi SDLC e i passaggi dello sviluppo della base di dati: la *modellazione concettuale dei dati* è presente solo in due fasi SDLC. Successivamente nel capitolo illustreremo questi passaggi di sviluppo della base di dati per Pine Valley Furniture Company.

Modellazione aziendale

Il processo di sviluppo della base di dati inizia con la modellazione aziendale (parte della fase Identificazione e selezione del progetto in SDLC) per impostare la gamma e i contenuti generali delle basi di dati dell'azienda. Si ricordi che la modellazione aziendale avviene durante la pianificazione del sistema informativo e di altre attività, quando la necessità di un sistema informativo cambia, quando sono identificati i miglioramenti e quando viene delineato l'ambito di tutti i dati dell'azienda. In questa fase si devono rivedere le basi di dati e i sistemi informativi correnti, analizzare la natura del contesto aziendale che è oggetto del progetto di sviluppo e descrivere, in termini molto generici, i dati necessari per ciascun sistema informativo preso in considerazione nello sviluppo. Solo i progetti selezionati passano alla fase successiva in base alla loro importanza potenziale per l'azienda.

Modellazione dei dati concettuale

Per un progetto di sistema avviato, si analizzano nella modellazione di dati concettuale le esigenze di dati complessive del sistema informativo proposto. Per farlo, occorrono due passaggi. Nella fase Avvio del progetto e pianificazione, si sviluppi un diagramma simile a quello mostrato nella Figura 4.1, oltre a una documentazione che delinea l'ambito dei dati coinvolti in questo progetto di sviluppo specifico a prescindere dalle basi di dati esistenti. In questa fase vengono incluse solo categorie di dati (entità) di alto livello e le associazioni principali. Durante la fase di Analisi di SDLC, si produce un modello di dati dettagliato che identifichi tutti i dati dell'azienda che devono essere gestiti per questo sistema informativo. Si definiscono tutti gli attributi dei dati, si rappresentano ogni associazione tra le entità di dati e si specificano le regole che permettono l'integrità dei dati. È durante la fase di analisi che si verifica la coerenza del modello di dati concettuale (che successivamente verrà chiamato anche *schema concettuale*) rispetto ad altri tipi di modelli sviluppati per illustrare altre dimensioni del sistema informativo finale, come le fasi dell'elaborazione, le regole per la gestione dei dati e la temporizzazione degli eventi. Tuttavia, anche questo modello di dati concettuale è preliminare, poiché le attività SDLC successive possono riscontrare elementi mancanti o errori durante la progettazione di transazioni specifiche, report, visualizzazioni e inchieste. Si dice spesso che la modellazione dei dati concettuale avviene in una forma top-down, derivata dalla comprensione generale del contesto aziendale e non dalle attività di elaborazione specifiche delle informazioni.

Progettazione logica della base di dati

La progettazione logica considera la base di dati da due punti di vista. Innanzitutto trasforma il modello di dati concettuale in una notazione standard nota con il nome di relazioni, basata sulla teoria delle basi di dati relazionali (verrà descritta nel Capitolo 7). In secondo luogo, ogni programma nel sistema informativo è progettato, anche nei suoi formati di input e output, per eseguire un controllo dettagliato delle transazioni, dei report, delle visualizzazioni e delle inchieste supportate dalla base di dati. Durante questa analisi cosiddetta *bottom-up*, si verifica esattamente quali dati devono essere mantenuti nella base di dati e la loro natura in ogni transazione, report e così via.

L'analisi di ogni singolo report, transazione o altro prende in considerazione una vista specifica e limitata, seppur articolata, della base di dati. Se necessario, per esempio, può essere possibile modificare il modello di dati concettuale man mano che ciascun report, transazione o altro viene analizzato. Soprattutto nel caso di progetti più grandi, i diversi team di analisti e sviluppatori di sistema possono lavorare indipendentemente su diversi programmi o gruppi di programmi. I dettagli del loro lavoro possono anche non essere rivelati fino alla fase di progettazione logica. In questo caso si dovrà combinare o integrare il modello concettuale originale con le singole *viste utente* in una progettazione logica della base di dati articolata. È inoltre possibile che durante la progettazione logica dei sistemi informativi vengano identificate esigenze supplementari di elaborazione delle informazioni, che dovranno quindi essere integrate nella progettazione logica precedentemente identificata.

La fase finale della progettazione logica della base di dati è la trasformazione delle specifiche di dati combinate e riconciliate in elementi di base, o atomici, che seguano le regole stabilite dalla teoria delle basi di dati relazionali e da un processo chiamato normalizzazione, che verrà descritto nel Capitolo 7. Il risultato è un'immagine globale della base di dati senza alcun riferimento a un sistema di gestione specifico dei dati. Una volta implementata la progettazione logica della base di dati, si può iniziare a specificare la logica dei programmi e delle interrogazioni necessarie per mantenere il contenuto dei dati e per generare i relativi report.

Definizione e progettazione fisica della base di dati

Nella definizione e progettazione fisica della base di dati si prendono alcune decisioni sull'organizzazione della memorizzazione della base di dati su computer (solitamente su disco) e si definisce la struttura fisica del sistema di gestione della base di dati. Vengono delineati i programmi che devono elaborare le transazioni e generare informazioni di gestione anticipate e report di supporto alle decisioni. Lo scopo è quello di progettare una base di dati che gestisca in modo efficace e sicuro l'elaborazione di tutti i dati a fronte di questo sistema. La progettazione fisica viene quindi impostata in stretta combinazione con la struttura di tutti gli altri aspetti fisici del sistema informativo: programmi, hardware, sistemi operativi e reti di comunicazione dei dati.

Implementazione della base di dati

Nell'implementazione della base di dati, si scrivono, testano e installano i programmi che elaboreranno la base di dati. Si può programmare con linguaggi di programmazione standard (come COBOL, C o Visual Basic), con linguaggi speciali per le basi di dati (come SQL) o si possono utilizzare linguaggi non procedurali *special-purpose* per produrre report e visualizzazioni con uno stile, che possibilmente includano dei grafici. Durante l'implementazione, si definisce inoltre tutta la documentazione della base di dati, si formano gli utenti e si abilitano le procedure per l'assistenza continua agli utenti (e alle basi di dati) del sistema informativo. L'ultimo passaggio consiste nel caricare i dati dalle fonti informative esistenti (file e basi di dati provenienti da applicazioni legacy ai quali vengono aggiunti i nuovi dati che si sono resi necessari). Il caricamento viene spesso eseguito trasferendo prima i dati dai file e dalle basi di dati esistenti in un formato neutro (come file di testo o binari) e poi caricando questi dati nella nuova base di dati. Infine si abilitano la base di dati e le applicazioni associate per la gestione e il recupero dei dati da parte degli utenti veri e propri. Durante la produzione, si dovranno eseguire un backup della base di dati e un suo eventuale ripristino in caso di danneggiamento o distruzione.

Manutenzione della base di dati

La base di dati si evolve durante la sua manutenzione. In questa fase si aggiungono, eliminano o modificano le caratteristiche della sua struttura per assecondare condizioni aziendali mutevoli, per correggere gli errori nella sua progettazione o per migliorare la velocità di elaborazione delle sue applicazioni. Potrebbe anche essere necessario ricostruire la base di dati qualora questa venisse danneggiata o distrutta a causa del malfunzionamento di un programma o di un sistema. Questa è solitamente la fase più lunga dello sviluppo, poiché si estende lungo tutta la vita della base di dati e delle applicazioni a essa associate. Ogni evoluzione della base di dati può essere considerata una sorta di sviluppo abbreviato della base di dati, nel quale la modellazione di dati concettuale, la progettazione logica e fisica e l'implementazione si adattano alle modifiche sovvenute.

4.2.2 Approcci alternativi di sviluppo dei sistemi informativi

Il ciclo di vita dello sviluppo dei sistemi o le lievi modifiche su di esso vengono spesso utilizzati per guidare lo sviluppo dei sistemi informativi e delle basi di dati. SDLC è un approccio metodico molto strutturato, che include numerose verifiche e adeguamenti per garantire che ogni passaggio produca risultati precisi e che il sistema informativo nuovo o sostitutivo sia coerente con i sistemi esistenti, con i quali deve comunicare o per i quali occorrono definizioni di dati coerenti. Come conseguenza, SDLC viene spesso criticato per la lunghezza dei tempi necessari prima di ottenere un sistema funzionante, cosa che si verifica solo alla fine del processo. Le aziende preferiscono sempre più utilizzare metodi

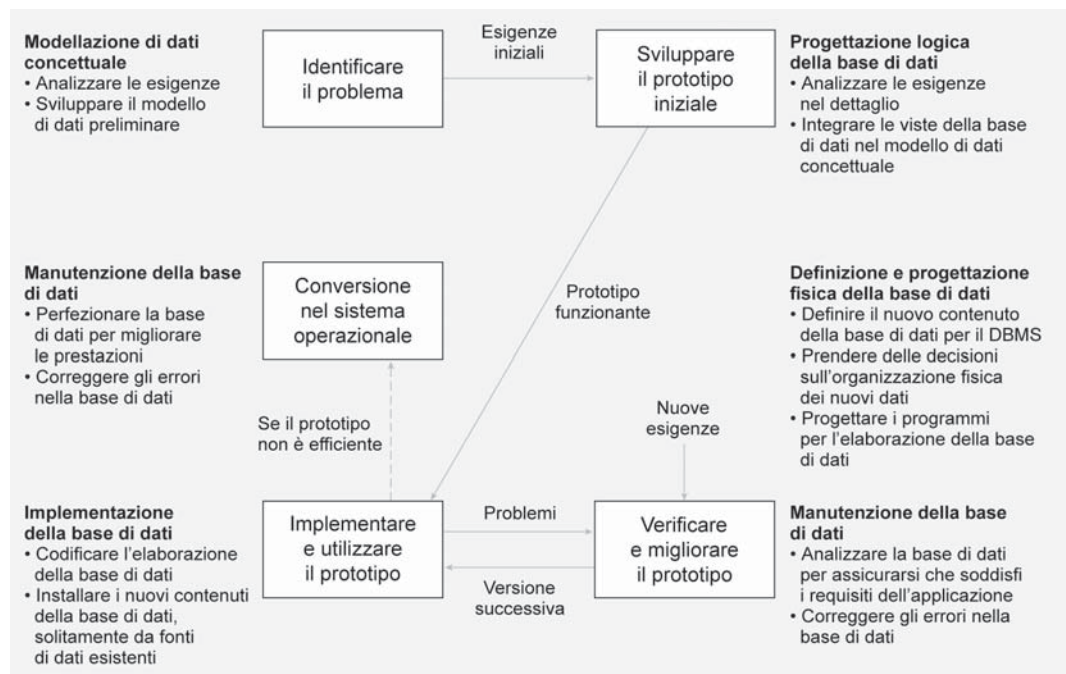
rapidi di sviluppo delle applicazioni (RAD), che seguono un processo iterativo di ripetizione veloce delle fasi di analisi, progettazione e implementazione fino a convergere nel sistema informativo desiderato dall'utente. I metodi RAD funzionano al meglio quando la maggior parte delle basi di dati è già esistente, e quindi soprattutto con i sistemi che recuperano i dati piuttosto che con quelli che popolano e verificano le basi di dati.

Uno dei metodi RAD più diffusi è quello della **creazione di prototipi (prototipazione)**. Si tratta di un processo iterativo di sviluppo dei sistemi in cui le esigenze vengono convertite in un sistema funzionante che viene continuamente verificato mediante una stretta collaborazione tra analisti e utenti. La Figura 4.6 illustra questo metodo. Nella figura sono indicate approssimativamente le attività di sviluppo della base di dati che hanno luogo in ogni fase della creazione del prototipo. Solitamente, quando viene identificato un problema nel sistema informativo, si esegue un tentativo molto rapido di intervento sulla modellazione di dati concettuale. Durante lo sviluppo del prototipo iniziale, si progettano le visualizzazioni e i report richiesti dall'utente accogliendo contemporaneamente le nuove esigenze e definendo una base di dati che deve essere utilizzata dal prototipo. Questo è in genere una nuova base di dati, che è una copia delle porzioni delle basi di dati esistenti popolate con il nuovo contenuto. Se sono necessari nuovi contenuti, questi proverranno solitamente da fonti di dati esterne, come dati di ricerche di mercato, indicatori economici generali o standard industriali.

L'implementazione e l'attività di manutenzione della base di dati vengono quindi ripetute ogni volta che viene prodotta una nuova versione del prototipo. I controlli di sicurezza e integrità sono spesso ridotti al minimo, poiché la discriminante risiede proprio nell'ottenimento di versioni di prototipo funzionanti nel minor tempo possibile. Anche la documentazione tende a essere ritardata fino alla fine del progetto e la formazione degli utenti avviene durante l'uso. Infine, una volta che un prototipo accettato viene creato, lo sviluppatore e l'utente decidono se il prototipo finale e la sua base di dati possono essere

Figura 4.6

La metodologia di creazione del prototipo e il processo di sviluppo della base di dati.



messi in produzione così come sono. Se il sistema e la base di dati non sono efficienti, verranno riprogrammati e riorganizzati perché soddisfino le aspettative di prestazione.

Con l'aumento della diffusione degli strumenti di programmazione visiva (come Visual Basic, Java, Visual C++ e i linguaggi di quarta generazione), nei quali è semplice modificare l'interfaccia tra utente e sistema, la creazione di prototipi sta diventando la metodologia di sviluppo privilegiata. Con i prototipi è relativamente facile modificare il contenuto e il layout dei report e delle visualizzazioni. Durante questo processo, possono essere identificate nuove esigenze della base di dati, e le basi di dati esistenti possono essere utilizzate dall'applicazione in evoluzione che deve essere modificata. È inoltre possibile utilizzare la creazione di prototipi per un sistema che richiede una nuova base di dati. In questo caso, vengono acquisiti dati campione per costruire e ricostruire il prototipo della base di dati man mano che le esigenze evolvono durante il processo di sviluppo iterativo.

4.2.3 Il ruolo degli strumenti CASE e del repository

In più punti del paragrafo precedente abbiamo citato il ruolo degli strumenti CASE nello sviluppo dei sistemi informativi. Gli strumenti CASE (*Computer-Aided Software Engineering*) sono un software che fornisce un supporto automatizzato in una o più fasi del processo di sviluppo dei sistemi informativi.

Per i nostri scopi nello studio dello sviluppo di una base di dati, ci soffermeremo su tre funzioni importanti degli strumenti CASE. La prima è l'aiuto che offrono nel disegnare modelli di dati utilizzando notazioni entità-associazione e di altro tipo (la Figura 4.1 mostra un esempio della notazione E-R). Le funzioni di disegno degli strumenti CASE sono "intelligenti", poiché ciascun simbolo rappresenta costrutti specifici della modellazione di dati, e i simboli possono essere utilizzati solo in modo coerente con le proprietà dei costrutti associati. Gli strumenti di disegno delle basi di dati vengono utilizzati nella modellazione aziendale e concettuale, nella progettazione logica della base di dati e nella modellazione di dati fisica. Gli strumenti CASE contribuiscono a garantire la coerenza tra i diagrammi. Per esempio, uno strumento CASE assicura che ogni uso dell'entità PRODUCT in qualsiasi diagramma abbia esattamente lo stesso significato. Questi strumenti assegnano nomi univoci a ciascun oggetto di dati e possono ridisegnare automaticamente un modello di dati quando le caratteristiche dell'oggetto cambiano.

Strumenti di disegno diversi e i metodi a essi associati sono spesso utilizzati in fasi diverse del processo di sviluppo della base di dati. Per esempio, uno strumento può essere utilizzato nella modellazione aziendale per disegnare diagrammi e matrici di alto livello, un altro nella modellazione di dati concettuale e un altro ancora nella progettazione logica.

La seconda funzione importante degli strumenti CASE è la loro capacità di generare il codice. La maggior parte delle volte, questo codice contiene i comandi di definizione della base di dati da passare al sistema di gestione. Durante l'implementazione della base di dati, lo strumento CASE fa riferimento a tutte le specifiche dei dati concettuali, logiche e fisiche e genera comandi SQL per creare tabelle relazionali, definire ciascun attributo di ciascuna tabella e definire gli indici di chiave. Alcuni strumenti CASE possono generare eccezionalmente un codice C o in un altro linguaggio per fornire i rudimenti del recupero della base di dati e l'aggiornamento dei programmi.

Come nel caso degli strumenti di disegno, molte aziende utilizzano strumenti separati (i migliori) sia nella generazione del codice sia nel disegno. Perché questi strumenti e metodi siano efficaci, devono essere integrati. In particolare, devono essere in grado di condividere i metadati che vengono sviluppati durante ogni fase del processo. Sfortunatamente, la capacità di condividere questo tipo di informazioni tra gli strumenti CASE non è ancora diffusa, soprattutto se provengono da produttori diversi. Uno strumento cosiddetto *CASE-integrato* (o *I-CASE*) fornisce un supporto lungo tutto il ciclo di vita, ma questi strumenti sono raramente utilizzati perché sono molto potenti in alcune fasi e molto deboli in altre. Gli strumenti che supportano SDLC dalla fase di Identificazione e selezione del progetto fino alla Proget-

tazione fisica sono chiamati *strumenti upper-CASE*; quelli che supportano le fasi di Implementazione e Manutenzione sono chiamati *strumenti lower-CASE*.

L'integrazione di uno strumento dipende dall'architettura formale dettagliata per la creazione e manutenzione dei sistemi informativi. Questa architettura include le definizioni formali delle interfacce fra strumenti, gli standard dei modelli di dati tra gli stessi e i controlli comuni lungo il ciclo di vita. Questo conduce alla terza funzione degli strumenti CASE, ovvero quella del repository delle informazioni (o semplicemente *deposito*). Un **repository** è una base di conoscenze di informazioni sui fatti a cui l'impresa deve poter accedere e che questa deve poter elaborare per avere successo (Moriarty, 1991). Un esempio è quello di tutte le informazioni raccolte durante le sei fasi dello sviluppo della base di dati che vengono mantenute in un repository. In senso lato, un repository è anch'esso una base di dati che contiene le informazioni necessarie a generare diagrammi, definizioni di maschere e report e altre documentazioni del sistema. Un repository aiuta gli analisti di sistema e della base di dati a ottenere un'integrazione senza soluzione di continuità dei dati grazie ai numerosi strumenti CASE.

4.3 Gestire le persone coinvolte nello sviluppo della base di dati

Come si evince dalla Figura 4.5, una base di dati viene sviluppata come parte di un progetto. Un progetto è un insieme pianificato di attività correlate per raggiungere un obiettivo e che ha un inizio e una fine. Un progetto inizia con i primi passaggi della fase di Avvio e pianificazione del progetto e termina con gli ultimi passaggi della fase di Implementazione. Al responsabile del progetto viene assegnato un capo analista di sistema o di basi di dati. Questa persona è responsabile della creazione di piani di progetto dettagliati, del personale e della supervisione del team del progetto. Un responsabile di progetto valido possiede una buona leadership e competenze nella gestione, nelle relazioni e comunicazioni con i clienti, nelle tecniche di soluzione dei problemi, nella gestione dei conflitti, nella formazione di un team e nella gestione dei rischi e dei cambiamenti.

Un progetto viene avviato e pianificato nella fase di Avvio e Pianificazione del progetto, eseguito nelle fasi di Analisi, Progettazione logica, Progettazione fisica e Implementazione, e chiuso alla fine dell'implementazione. Il team viene formato durante l'avvio. Un team di sviluppo di sistema o di base di dati può includere una o più persone tra quelle elencate di seguito.

- *Analisti di sistema*: analizzano la situazione aziendale e identificano la necessità di informazioni e di servizi informativi per rispondere ai problemi e alle opportunità dell'azienda.
- *Analisti di basi dati*: si concentrano sulla definizione delle esigenze e del progetto per il componente "base di dati" del sistema informativo.
- *Utenti*: indicano le proprie esigenze informative e controllano che il sistema sviluppato soddisfi le loro necessità.
- *Programmatore*: progettano e scrivono i programmi che incorporano i comandi per mantenere e accedere ai dati nella base di dati.
- *Amministratori della base di dati e dei dati*: sono responsabili delle basi di dati esistenti e future e garantiscono la coerenza e l'integrità tra di esse; in quanto esperti delle tecnologie delle basi di dati, forniscono consulenza e formazione agli altri membri del team.
- *Altri esperti tecnici*: per esempio, esperti di rete, dei sistemi operativi, dei test e della documentazione.

È di competenza del responsabile del progetto selezionare e gestire tutte le persone come un team. Si veda Hoffer, George e Valacich (2002) per i dettagli su come gestire un team di progetto di sviluppo di un sistema.

La chiusura del progetto si verifica quando questo viene terminato, più o meno naturalmente. Si ha una chiusura non naturale quando le prestazioni del sistema o del team di sviluppo non sono accettabili o quando il progetto supera i limiti stabiliti di tempo, finanziamento o supporto. Per determinare se un progetto sta progredendo entro i termini e il budget previsti, il responsabile del progetto sviluppa piani dettagliati delle attività. Questi vengono spesso rappresentati in una forma grafica, come mostrato nei diagrammi di esempio della Figura 4.7. I diagrammi mostrano quando le attività del progetto iniziano e finiscono, chi è incaricato di svolgere ciascuna fase, quanto impegno è richiesto per portare a termine ogni fase e i rapporti di precedenza tra le attività (ossia quali attività dipendono dal risultato di altre attività).

Una caratteristica dei progetti di successo è la presenza di alcuni momenti di verifica frequenti, durante i quali i membri del team relazionano sui risultati del progetto in corso. I risultati vengono spesso presentati a persone esterne al team, compresi altri utenti, a chi fornisce i fondi per il progetto, ai responsabili dei sistemi informativi ed eventualmente al management generale. Lo scopo di questi momenti di revisione sono i seguenti.

- Convalidare il fatto che il progetto sta procedendo in modo soddisfacente.
- Astrarsi dai dettagli delle attività quotidiane e verificare globalmente che tutte le parti del progetto si combinino.
- Richiedere una conferma progressiva rinnovata a tutte le parti coinvolte nel progetto (molto importante nel caso di progetti che durano molti mesi).

Centrale al terzo punto è il concetto di conferma progressiva. La conferma progressiva (*incremental commitment*) consente alle parti interessate al progetto di impegnarsi solo nella fase successiva (con costi e tempi limitati) e quindi di riconfermare un nuovo impegno e nuove risorse (persone e denaro) dopo aver considerato i risultati raggiunti. In questo modo non vengono sprecate risorse qualora il concetto originario che soggiace al sistema non si dimostri più valido. La conferma progressiva permette anche a un progetto di venire facilmente reindirizzato o definitivamente interrotto.

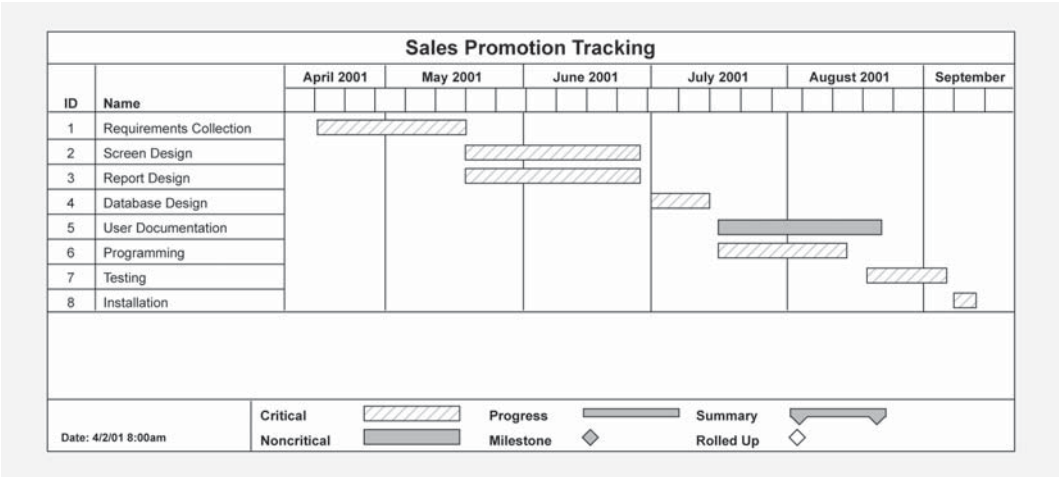
4.4 Architettura a tre schemi per lo sviluppo di una base di dati

La spiegazione del processo di sviluppo di una base di dati esposta precedentemente nel capitolo fa riferimento a modelli o viste diversi, ma correlati, di basi di dati sviluppate in un progetto di sviluppo complessivo dei sistemi informativi aziendali.

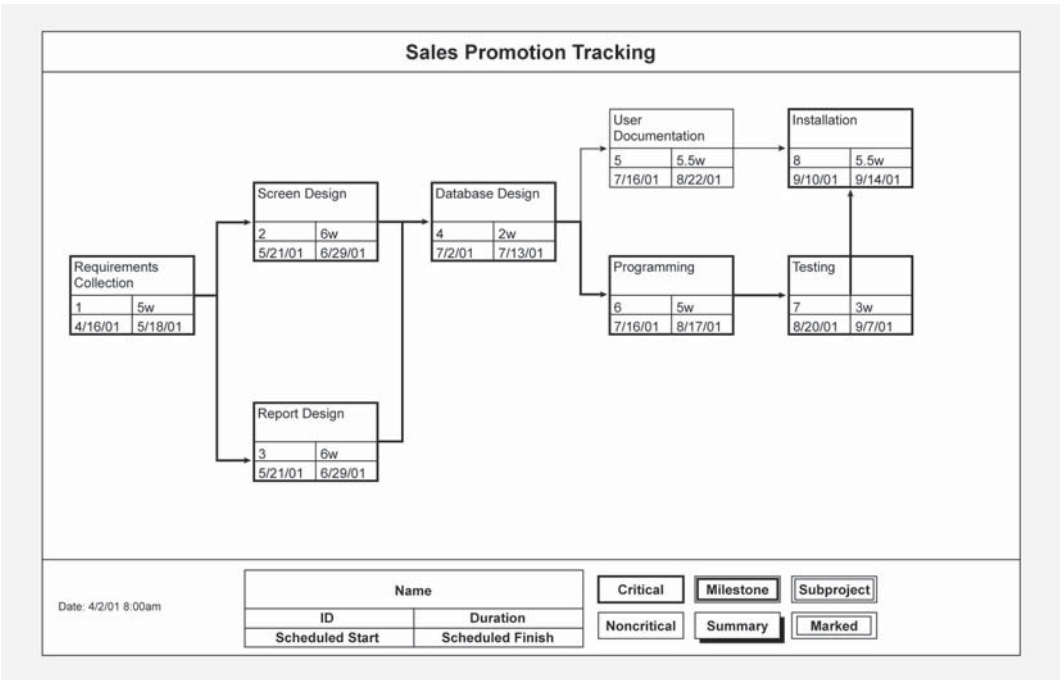
- Schema concettuale (durante la fase di Analisi).
- Schema esterno, o vista utente (durante le fasi di Analisi e di Progettazione logica).
- Schema fisico o interno (durante la fase di Progettazione fisica).

La Figura 4.8 illustra il rapporto tra queste tre viste di una base di dati. È importante ricordare che si tratta di viste o modelli della stessa base di dati; ogni base di dati di un'azienda ha cioè uno schema fisico e uno concettuale e una o più viste. L'architettura a tre schemi definisce quindi una base di dati con più modi diversi di guardare questo unico insieme di dati.

Figura 4.7
Diagrammi grafici per
l'illustrazione dei piani
del progetto.
(a) Diagramma
di Gantt.



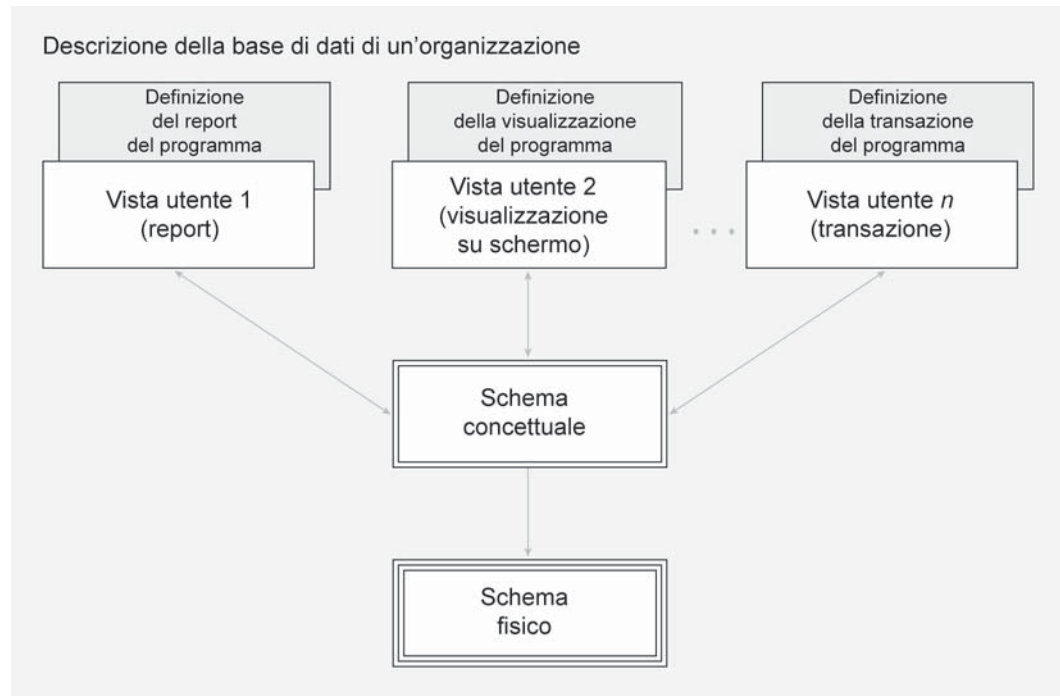
(b) Diagramma
di PERT.



Uno **schema concettuale** è una specifica dettagliata della struttura complessiva dei dati dell'azienda indipendente da qualsiasi tecnologia di gestione della base di dati. Uno schema concettuale definisce l'intera base di dati senza fare riferimento a come i dati sono memorizzati nella memoria secondaria di un computer. Uno schema di questo tipo viene solitamente rappresentato graficamente utilizzando le notazioni di entità-associazione (E-R) o di modellazione dei dati; abbiamo chiamato questo tipo di schema concettuale *modello di dati*. Inoltre, le specifiche dello schema concettuale vengono memorizzate come metadati in un repository o dizionario dei dati.

Figura 4.8

L'architettura a tre schemi della base di dati.



Una *vista utente* è una descrizione logica di una porzione della base di dati richiesta da un utente per eseguire una determinata operazione. Una vista utente, o schema esterno, è quindi indipendente dalla tecnologia della base di dati ma contiene solitamente un sottoinsieme di schemi concettuali associati, importanti per un particolare utente o gruppo di utenti (per esempio, un responsabile degli inventari o l'ufficio contabilità). Una vista utente è anche la versione dello schema utilizzata in un particolare programma (per esempio, un programma di immissione ordini), perché un programma è utilizzato da un particolare utente o gruppo di utenti. Una vista utente è definita sia in termini logici (indipendenti dalla tecnologia) sia in termini di linguaggio di programmazione (coerenti cioè con la sintassi del linguaggio del programma). La rappresentazione tradizionale di una vista utente è una visualizzazione su schermo, una transazione aziendale (come una form di rinnovo di un'iscrizione) o un report, poiché tutti rappresentano i dati necessari per il programma che li elabora. Una versione logica di una vista utente può essere rappresentata da un oggetto o diagramma E-R o sotto forma di relazioni. Il processo di conversione di un diagramma E-R in relazioni e della loro unione (o integrazione della vista) in una descrizione completa e razionale della base di dati è descritta nel Capitolo 7.

Uno schema fisico contiene le specifiche del modo in cui i dati di uno schema concettuale sono memorizzati nella memoria secondaria del computer. Un ruolo in un progetto di sviluppo di una base di dati può riguardare esclusivamente le operazioni associate a una sola di queste tre viste. Per esempio, all'inizio della carriera, un progettista può progettare schemi esterni utilizzati in uno o più programmi; con il tempo e dopo aver acquisito esperienza, progetterà schemi esterni o concettuali. Gli aspetti problematici della progettazione di una base di dati variano a seconda dei livelli. Per questa ragione, questo libro è organizzato in modo da concentrare l'attenzione sui singoli problemi di rilievo nella progettazione di ciascuna delle tre rappresentazioni di una base di dati.

- *Concettuale*: i Capitoli 5 e 6 trattano la modellazione entità-associazione, una delle rappresentazioni grafiche degli schemi concettuali o dei modelli di dati.
- *Esterna*: il Capitolo 7 tratta le basi di dati relazionali, la normalizzazione, la traduzione dei diagrammi E-R in relazioni e l'unione di insiemi di relazioni; tutti questi argomenti supportano lo sviluppo e l'analisi degli schemi esterni.
- *Fisica*: il Capitolo 8 tratta le decisioni da prendere quando si progetta una base di dati fisica, mentre i Capitoli 9 e 10 riepilogano come specificare queste decisioni in un linguaggio utilizzato per definire gli schemi e le basi di dati fisiche, ovvero SQL. L'Appendice D si occupa delle basi di dati relazionali a oggetti.

Gli schemi concettuale, esterno e fisico formano l'architettura a tre schemi delle basi di dati. Si noti che, sebbene la Figura 4.8 mostri lo schema esterno in alto, questo non viene sviluppato necessariamente prima di quello concettuale. Solitamente, infatti, i due tipi di schema vengono sviluppati iterativamente (Figura 4.9). Prima viene data una impostazione allo schema concettuale in base al modello di dati aziendale dell'organizzazione e alla comprensione generale delle esigenze della base di dati in un progetto. Quindi vengono sviluppati gli schemi esterni (viste utente) per transazioni, report, visualizzazioni su schermo e altri usi del sistema. Nella maggior parte dei casi, un'analisi degli schemi esterni produce nuovi attributi ed eventualmente entità e associazioni non mostrati nello schema concettuale. Questo viene così argomentato con le esigenze identificate dalle fonti cosiddette bottom-up, rendendo gli schemi concettuale ed esterno coerenti. Quando le viste utente vengono identificate, il processo di evoluzione dei due tipi di schema viene ripetuto.

Per iniziare a sviluppare la base di dati e i programmi applicativi a essa associati, si scrivono le specifiche per lo schema fisico associato. Oltre agli schemi concettuale ed esterno, si considereranno anche le caratteristiche hardware e software e le aspettative degli utenti sulle prestazioni della base di dati mentre si progetta quest'ultima con lo schema fisico associato. È possibile che nella progettazione della base di dati fisica si presentino delle incoerenze o altri problemi legati allo schema concettuale o alle viste utenti; è quindi possibile tornare a queste fasi della progettazione (non mostrate nella Figura 4.9). Se in seguito si riscontrano nuove esigenze degli utenti, il processo inizierà daccapo. Solitamente le nuove esigenze vengono trattate e prese in considerazione in un nuovo ciclo di progettazione dello schema esterno e concettuale, così che la base di dati non sia sottoposta continuamente a modifiche. Revisioni periodiche della base di dati si effettuano durante la fase di Manutenzione dello sviluppo dei sistemi.

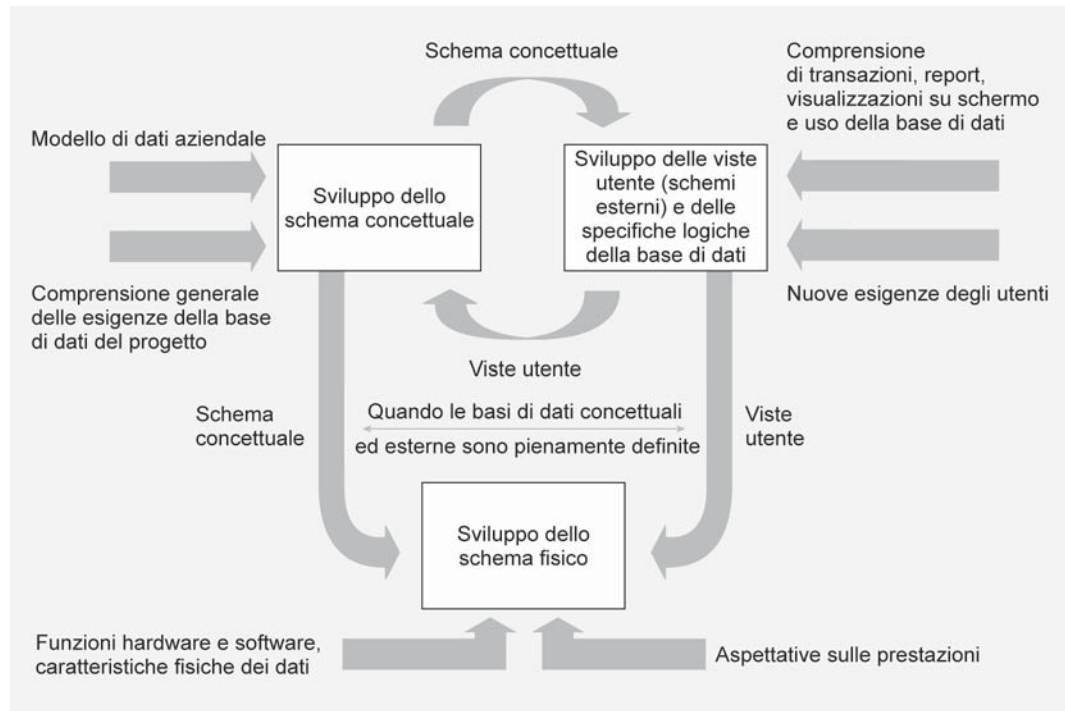
4.5 Architettura fisica a tre livelli

Quando si progetta una base di dati, e in particolare durante la fase di progettazione fisica, sia ha la possibilità di decidere dove memorizzare i dati. Nel Capitolo 8 prenderemo in considerazione un'ampia gamma di decisioni di progettazione fisica, ma in questo paragrafo delineeremo le possibilità principali legate all'architettura della base di dati che possono influire sul processo di sviluppo.

Nel Capitolo 3 abbiamo visto che esistono basi di dati personali, di gruppo di lavoro, di funzione, aziendale e Internet. Le basi di dati personali sono spesso concepite e sviluppate dagli stessi utenti finali, con la sola formazione e consulenza dei professionisti della base di dati, se questa contiene dati che sono di interesse esclusivo dei suoi utenti finali. A volte una base di dati personale è un estratto di una base di dati di gruppo di lavoro o aziendale, nel qual caso i professionisti spesso scrivono le routine di estrazione per la creazione della base di dati locale. Queste basi di dati sono di frequente il frutto di una collaborazione tra gli utenti finali, i professionisti dei sistemi che lavorano nelle unità aziendali e

Figura 4.9

Architettura a tre schemi per il progetto di sviluppo di una base di dati.



i professionisti della base di dati centrale. Questa combinazione di persone è indispensabile, in quanto nella progettazione delle basi di dati condivise devono essere bilanciati diversi aspetti: la velocità di elaborazione, la facilità d'uso, le differenze nelle definizioni dei dati e così via. Le basi di dati aziendali e Internet, a causa del loro impatto su scala aziendale e della loro dimensione estesa, sono solitamente sviluppate da professionisti specializzati molto preparati, che lavorano spesso in gruppi di sviluppo di basi di dati centralizzate.

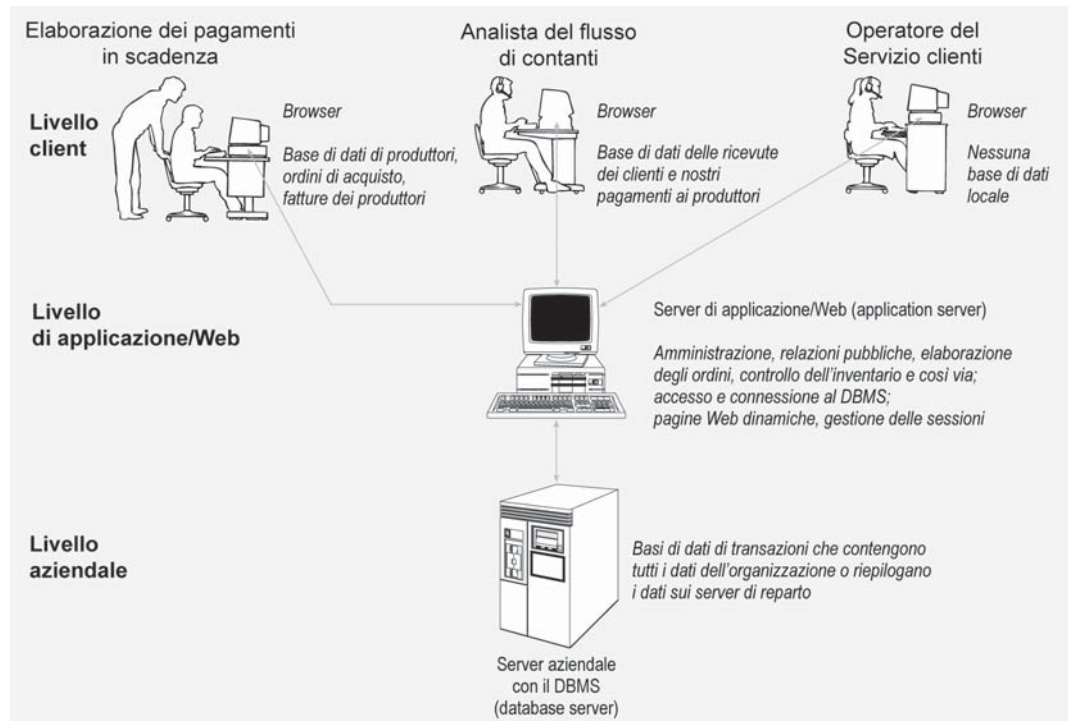
Questa vista della collocazione della base di dati, che implica che una base di dati si trovi forzosamente a uno di questi quattro livelli, risulta parzialmente semplificata in molte aziende moderne. I dati di un sistema informativo possono infatti risiedere in più posizioni o livelli di computer, allo scopo di equilibrare i diversi fattori tecnici e dell'azienda. Diversi tipi di elaborazione dei dati possono anche trovarsi in posizioni diverse per permettere di godere dei vantaggi di velocità di elaborazione e facilità d'uso e per semplificare la programmazione su piattaforme diverse. L'uso esteso di interfacce basate su browser nelle basi di dati aziendali ha comportato una crescita nel mercato degli "application server" (*server di applicazioni*), che gestiscono il software aziendale e le transazioni tra i browser Web e le basi di dati back-end. La maggior parte dei produttori di server di applicazioni, compresi Sun, IBM e Oracle, ha adottato Java 2 Enterprise Edition (J2EE) come proprio standard di server. Sono possibili quattro livelli: dati su un server client, su un application server, su un server Web e su un server di base di dati (*database server*). Sono tuttavia considerati comuni tre livelli.

1. **Livello client:** un computer desktop o laptop specializzato nella gestione dell'interfaccia utente-sistema e dei dati individuati. Viene chiamato anche *livello di presentazione*. Le operazioni di scripting Web possono essere eseguite a questo livello.

2. *Livello di applicazione/Web*: elabora il protocollo HTTP e le attività di scripting, esegue calcoli e fornisce un accesso ai dati. Viene chiamato anche *livello dei servizi*.
3. *Livello di server aziendale (minicomputer o mainframe)*: esegue calcoli complessi e gestisce l'unione di dati provenienti da più fonti dell'azienda. Viene chiamato anche *livello dei servizi di dati*.

Questa architettura a tre livelli è illustrata nella Figura 4.10. Una vista limitata dell'architettura client/server considera solo il livello client e un livello di server generale.

Figura 4.10
L'architettura client/server a tre livelli della base di dati.



L'architettura a livelli per le basi di dati e i sistemi informativi si basa sul concetto di un'architettura client/server per l'uso distribuito dei computer tipico di un'azienda. In un'architettura client/server, che si basa su una rete locale, il software della base di dati su un server (chiamato *server della base di dati* o *motore della base di dati*) esegue i comandi della base di dati inviati dalle stazioni di lavoro client, mentre i programmi applicativi su ciascun client si concentrano sulle funzioni dell'interfaccia utente. Nella pratica, le basi di dati totalmente concettuali, così come le routine di elaborazione delle applicazioni che accedono alla base di dati, possono essere distribuite sulla stazione di lavoro locale, su un server intermedio (di gruppo di lavoro o di funzione) e su un server centralizzato (di funzione o aziendale) come basi di dati distribuite o come basi di dati fisiche separate ma correlate. Hoffer, George e Valacich (2002) e Thompson (1997) spiegano perché questo tipo di architettura stia diventando sempre più popolare. Le ragioni per l'uso di un'architettura client/server sono elencate di seguito.

- Permette l'elaborazione contemporanea della stessa applicazione su più processori, migliorando così il tempo di risposta e la velocità di elaborazione dei dati.

- È possibile trarre il massimo beneficio dalle funzioni migliorate di elaborazione dei dati di ciascuna piattaforma (per esempio, le funzioni avanzate di interfaccia utente dei PC rispetto alla velocità di calcolo dei minicomputer o dei mainframe).
- È possibile combinare tecnologie client (personal computer con processori Intel o Motorola, computer in rete, kiosk informativi e così via) e condividere allo stesso tempo dati comuni. È inoltre possibile modificare le tecnologie in corrispondenza di ciascun livello con un impatto limitato ai moduli di sistema sugli altri livelli.
- L'elaborazione può essere effettuata vicino alla fonte dei dati elaborati, migliorando di conseguenza i tempi di risposta e riducendo il traffico di rete.
- Permette e incoraggia l'accettazione degli standard di sistema aperti.

La conseguenza più significativa dello sviluppo di una base di dati derivante dall'uso di un'architettura client/server a tre livelli è la facilità di separazione tra lo sviluppo e i moduli che mantengono la gestione della base di dati dai moduli dei sistemi informativi che presentano il contenuto della base di dati agli utenti finali. Le routine di presentazione possono interagire tra i livelli per accedere alle routine che portano ai dati richiesti e per analizzare questi dati per formare le informazioni necessarie. In quanto sviluppatori di basi di dati e analisti, si può lavorare in corrispondenza di uno qualsiasi di questi livelli, sviluppando il software richiesto.

4.6 Sviluppare un'applicazione di base di dati per Pine Valley Furniture Company

Pine Valley Furniture Company è stata presentata nel Capitolo 3. La società ha utilizzato la tecnologia di gestione delle basi di dati relazionali fin dal 1990, con diverse basi di dati su personal computer, minicomputer di funzione e un server aziendale (si veda la Figura 3.6 per uno schema generale della rete di computer di Pine Valley Furniture Company).

Helen Jarvis, product manager di mobili per ufficio di Pine Valley Furniture Company, è consapevole della forte concorrenza su questa linea di prodotti in crescita. Per Pine Valley è quindi importante che Helen sia in grado di analizzare approfonditamente le vendite dei prodotti. Queste analisi sono spesso ad hoc, e si basano su condizioni aziendali in cambiamento e non previste, su commenti provenienti dai negozi, su indiscrezioni dell'industria o sull'esperienza. Helen ha chiesto che le venga dato un accesso diretto ai dati di vendita attraverso un'interfaccia facile da utilizzare, così da poter cercare le risposte alle varie domande di marketing che sorgeranno nel tempo.

Chris Martin è un analista di sistemi dell'ufficio di sviluppo dei sistemi informativi di Pine Valley Furniture Company. Chris ha lavorato per l'azienda per cinque anni e ha avuto diverse esperienze sui sistemi informativi in diversi ambiti aziendali. Grazie ai progetti sui cui ha lavorato, ai corsi frequentati all'università e alla formazione avanzata ricevuta in Pine Valley, è diventato uno dei migliori analisti di sistema di Pine Valley. Ha competenze sulla modellazione dei dati e ha familiarità con numerosi sistemi di gestione delle basi di dati relazionali utilizzati all'interno della ditta. Grazie a questa esperienza, competenza e disponibilità, i capi dei sistemi informativi hanno deciso che Chris lavori con Helen, su sua richiesta, su un sistema di supporto al marketing.

Poiché Pine Valley Furniture ha sempre prestato molta cura nello sviluppo dei propri sistemi, specialmente dopo aver adottato l'approccio a basi di dati, l'azienda ha ora un'architettura di sistemi informativi piuttosto completa, che comprende basi di dati per tutte le sue funzioni operazionali. È quindi

probabile che Chris sarà in grado di estrarre i dati di cui Helen necessita dalle basi di dati esistenti. L'architettura dei sistemi informativi di Pine Valley richiede che i sistemi come quelli chiesti da Helen siano costruiti su basi di dati indipendenti, così che l'uso di dati non strutturati e imprevedibili non interferisca con l'accesso alle basi di dati operazionali necessario a supportare sistemi di elaborazione delle transazioni efficienti.

Poiché le esigenze di Helen riguardano l'analisi dei dati, e non la loro creazione e gestione, e sono personali, e non istituzionali, Chris decide di adottare una combinazione degli approcci del prototipo e del ciclo di vita per sviluppare il sistema richiesto da Helen. Con "combinazione", Chris intende che seguirà i passaggi del ciclo di vita, ma eseguirà in modo rapido quei passi che non sono parte integrante della creazione di un prototipo. Considererà quindi velocemente l'avvio e la pianificazione (compresa la collocazione del progetto all'interno dell'architettura del sistema informativo dell'azienda), e poi utilizzerà un ciclo iterativo di analisi, progettazione e implementazione per lavorare da vicino con Helen e sviluppare un prototipo funzionante del sistema che le occorre. Poiché il sistema sarà personale e probabilmente richiederà una base di dati con un ambito limitato, Chris ha deciso di sviluppare il sistema utilizzando Microsoft Access, la tecnologia privilegiata da Pine Valley per le basi di dati personali. Nella maggior parte dei casi, tuttavia, verrà anche illustrato il codice SQL necessario per definire la struttura della base di dati, poiché questi comandi possono essere utilizzati per definire una base di dati utilizzando qualsiasi sistema di gestione delle basi di dati relazionali.

4.6.1 Soddisfare le esigenze dell'utente con l'architettura informativa esistente

Chris inizia il progetto di sviluppo della base di dati e del sistema associato di supporto al marketing per il reparto dei mobili per ufficio ponendo alcune domande a Helen. Chris le chiede del suo reparto, prende appunti sugli obiettivi del reparto, sulle funzioni aziendali, sui tipi di entità di dati e su altri oggetti aziendali con cui Helen ha a che fare. In questa fase, Chris più che parlare ascolta, così da potersi concentrare sulla comprensione del lavoro di Helen; interviene con domande e si assicura che Helen non cerchi di passare oltre per parlare di ciò che ritiene dovrebbe essere presente nelle schermate e nei report. Chris pone domande molto generiche, utilizzando il più possibile una terminologia aziendale e di marketing. Per esempio, chiede a Helen quali sono i problemi che le capita di incontrare nella gestione dei prodotti per ufficio, quali persone, luoghi e cose sono di interesse per il suo lavoro, a quanto tempo addietro devono risalire i dati perché l'analisi le risulti utile e quali sono gli eventi aziendali che influiscono sul suo lavoro.

La Tabella 4.4 riepiloga quello che Chris ha tratto dal primo colloquio con Helen. La tabella elenca i vari oggetti aziendali citati da Helen e indica quali di questi sono già presenti nell'architettura dei sistemi informativi di Pine Valley Furniture Company. Come si può notare, sono molto poche le voci non presenti nell'architettura. Tutte le categorie di dati principali citate da Helen sono trattate nell'architettura e sono gestite nei sistemi informativi esistenti. Chris è quindi fiducioso del fatto che la maggior parte, se non tutti, i dati necessari a Helen siano già presenti nelle basi di dati dell'azienda.

Prima di parlare nuovamente con Helen, Chris compie ancora due rapide analisi. Innanzitutto identifica tutte le basi di dati che contengono i dati associati alle entità di dati menzionate da Helen. Da queste, Chris crea un elenco di tutti gli attributi di dati tratti da queste entità che ritiene possano essere importanti per Helen nelle analisi del mercato dei mobili per ufficio. Il coinvolgimento precedente di Chris in altri progetti che hanno sviluppato un sistema standard di tracciamento e previsione di vendite di Pine Valley e il sistema di contabilità lo aiutano a interpretare i dati della Tabella 4.4 per ipotizzare i dati che potrebbero servire a Helen. Per esempio, il superare l'obiettivo fissato per ogni categoria di

Tabella 4.4 Oggetti aziendali per il sistema a supporto del marketing della linea di prodotti.

Oggetto di pianificazione	Oggetti per il tracciamento della linea di prodotti di mobili per ufficio	L'oggetto è già presente nell'architettura dei sistemi informativi di Pine Valley?
Obbiettivi	Aumento annuale delle vendite dei prodotti per ufficio almeno del 16% (in \$)	No
	Aumento annuale del margine di profitto della linea di prodotti per ufficio almeno del 10%	No
	Aumento delle vendite ripetute allo stesso cliente dei prodotti per ufficio almeno del 5% (in \$)	No
	I risultati dei prodotti per ufficio devono superare la media dei risultati di tutti i prodotti di Pine Valley	No
	Tempo di evasione degli ordini di prodotti per ufficio ridotto almeno del 5%	No
	Tempo di ricevimento del pagamento finale sulle fatture dei prodotti ridotto del 5%	No
Unità organizzative	Ufficio marketing	Sì
	Gestione della linea di prodotti di mobili per ufficio	No
Luoghi aziendali	Ufficio contabilità	Sì
	Ufficio ordini	Sì
	Uffici vendite di zona	Sì
	Sviluppo del prodotto	–
	Analisi demografica	No
	Analisi di mercato dei destinatari	Sì
	Marketing e vendite	–
	Evasione degli ordini	Sì
	Riepilogo delle vendite	Sì
Tipi di entità	Conferma degli ordini	Sì
	CUSTOMER	Sì
	PRODUCT	Sì
	PRODUCT LINE	Sì
	ORDER	Sì
	INVOICE	Sì
	PAYMENT	Sì
Sistemi informativi (applicazioni)	Elaborazione degli ordini	Sì
	Gestione delle vendite	Sì

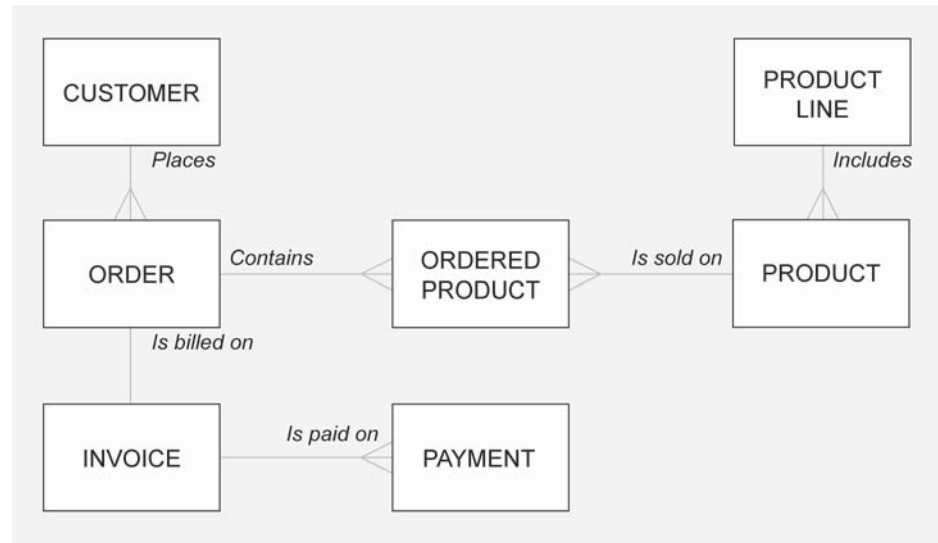
prodotti dei mobili per ufficio suggerisce che Helen desideri avere nel suo sistema gli obiettivi annuali di vendita dei prodotti; anche l'obiettivo di raggiungere una crescita minima annuale delle vendite del 16% significa che dovranno essere inclusi gli ordini per ciascun prodotto dell'anno precedente. Chris deduce anche che la base di dati di Helen dovrà includere tutti i prodotti, e non solo quelli nella linea dei mobili per ufficio, poiché Helen vorrà confrontare la sua linea con quella degli altri. Chris è tuttavia in grado di eliminare molti attributi di dati mantenuti su ciascuna entità di dati. Per esempio, a Helen non sembrano servire i vari dati dei clienti come indirizzo, numero di telefono, contatto, dimensioni del negozio e agente di vendita. Chris aggiunge invece alcuni attributi, come il tipo di cliente e il codice di avviamento postale, che non risultano ovvi dalla Tabella 4.4, perché sono attributi importanti nel sistema di previsione delle vendite.

In secondo luogo, da questo elenco Chris traccia un modello di dati grafico che rappresenta le entità di dati con gli attributi associati e con le associazioni principali tra di esse. La sua speranza è quella di poter ridurre il tempo dedicato alla fase di analisi del processo di sviluppo (e quindi quello della model-

lazione concettuale) presentando il modello di dati a Helen. La Figura 4.11 mostra un rappresentazione grafica del modello di dati per la base di dati preliminare prodotta da Chris per Helen. La Tabella 4.5 elenca gli attributi di dati relativi a ciascuna entità che secondo Chris possono essere utili a Helen nel sistema; la tabella elenca solo gli attributi fondamentali delle basi di dati esistenti, poiché Helen vorrà probabilmente combinarli in vari modi nelle analisi che dovrà fare.

Figura 4.11

Modello di dati preliminare per il sistema di supporto al marketing della linea di prodotti.



4.6.2 Analizzare le caratteristiche della base di dati

Il primo incontro di Chris con Helen è durato solo 45 minuti, ma Chris pensa che siano stati sufficienti per far nascere numerose idee da condividere con Helen rispetto ai dati che ha raccolto, utili per il suo lavoro successivo. Chris programma quindi una sessione più lunga, di due ore, con Helen per discutere su questi spunti. Prima dell'incontro, manda a Helen una bozza di progetto che delinea i passaggi che intende seguire e la loro durata approssimativa. Poiché la creazione di un prototipo è guidata dall'utente, che indica quando interrompere l'iterazione sulle nuove versioni del prototipo, Chris fornisce solo alcune stime approssimative della durata di alcuni passaggi. Per questa ragione, il capo di Chris ha deciso che questo progetto dovrà essere fatturato a Helen in base al tempo e non in base a un costo fisso.

Nel secondo incontro, Chris fa qualcosa di più che parlare. Illustra metodicamente ciascuna entità di dati riportata nella Figura 4.11 spiegando cosa significa, qual è il senso di ogni attributo di dati associato (Tabella 4.5) e quali politiche e procedure aziendali sono rappresentate da ciascuna linea tra le entità. Per esempio, Chris spiega che ogni ordine viene calcolato su una fattura e che viene effettuato da parte di un cliente. Altri dati che Chris ritiene che Helen possa voler conoscere includono l'ultima data in cui è stato effettuato l'ordine e quello in cui è stato evaso (questa dovrebbe essere la data di consegna più recente dei prodotti ordinati). Chris spiega anche che l'attributo *Payment_Date* rappresenta la data più recente in cui un cliente ha effettuato un pagamento, totale o parziale, per l'ordine.

Durante la discussione, Helen comunica a Chris alcuni dati supplementari che le sono necessari (il numero di anni nei quali un cliente ha acquistato prodotti di Pine Valley Furniture Company e il nume-

Tabella 4.5 Attributi di dati per le entità nel modello di dati preliminare.

Tipo di entità	Attributo
CUSTOMER	Customer_Identifier
	Customer_Name
	Customer_Type
	Customer_ZIPCODE
PRODUCT	Product_Identifier
	Product_Finish
	Product_Price
	Product_Cost
	Product_Annual_Sales_Goal
	Product_Line_Name
PRODUCT LINE	Product_Line_Name
	Product_Line_Annual_Sales_Goal
ORDER	Order_Number
	Order_Placement_Date
	Order_Fulfillment_Date
	Customer_Identifier
ORDERED PRODUCT	Order_Number
	Product_Identifier
	Order_Quantity
INVOICE	Invoice_Number
	Order_Number
	Invoice_Date
PAYMENT	Invoice_Number
	Payment_Date
	Payment_Amount

ro di spedizioni necessarie a evadere ciascun ordine). Helen nota anche Chris prevede solo un anno di obiettivi di vendita indicati per una linea di prodotti e gli ricorda che le occorrono questi dati anche per l’anno precedente e per l’anno corrente. Di fronte alle reazioni di Helen al modello di dati, Chris chiede a Helen come intende utilizzare i dati che chiede. Chris non approfondisce questo punto, perché sa che Helen non ha lavorato con un insieme di informazioni come quello in corso di sviluppo e non può quindi sapere con certezza quali dati le occorrono e per che cosa. Lo scopo di Chris è invece quello di capire i modi nei quali Helen intende utilizzare i dati, così da poter sviluppare un prototipo iniziale, che comprenda la base di dati e più visualizzazioni su computer o report. L’elenco finale degli attributi sulla cui necessità concorda anche Helen sono riportati nella Tabella 4.6.

4.6.3 Progettare la base di dati

Poiché Chris sta seguendo una metodologia di prototipo e poiché le prime due sessioni con Helen hanno identificato rapidamente i dati che le possono occorrere, Chris è in grado di iniziare immediatamente a creare il prototipo. Innanzitutto crea alcuni estratti dalla base di dati aziendale per isolare le entità di dati e gli attributi indicati da Helen. Chris può creare tutti questi file utilizzando il linguaggio di interrogazione SQL. Alcuni dei dati desiderati da Helen vengono calcolati da dati grezzi operazionali (per esempio, Customer_Years), ma SQL facilita a Chris la specificazione di questi calcoli. Questa estrazione genera

Tabella 4.6 Attributi di dati per le entità nel modello di dati finale.

Tipo di entità	Attributo (*)
CUSTOMER	Customer_Identifier
	Customer_Name
	Customer_Type
	Customer_ZIPCODE
	<i>Customer_Years</i>
PRODUCT	Product_Identifier
	Product_Finish
	Product_Price
	Product_Cost
	<i>Product_Prior_Year_Sales_Goal</i>
	<i>Product_Current_Year_Sales_Goal</i>
PRODUCT LINE	Product_Line_Name
	<i>Product_Prior_Year_Sales_Goal</i>
	<i>Product_Current_Year_Sales_Goal</i>
ORDER	Order_Number
	Order_Placement_Date
	Order_Fulfillment_Date
	<i>Order_Number_of_Shipments</i>
	Customer_Identifier
ORDERED PRODUCT	Order_Number
	Product_Identifier
	Order_Quantity
INVOICE	Invoice_Number
	Order_Number
	Invoice_Date
PAYMENT	Invoice_Number
	Payment_Date
	Payment_Amount

(*) Le modifiche rispetto all'elenco di attributi precedente appaiono in corsivo.

un singolo file ASCII per ciascuna entità di dati; ogni riga del file contiene tutti gli attributi di dati associati a quella entità di dati nel modello, e le righe sono istanze diverse dell'entità. Per esempio, ogni riga di entità di dati PRODUCT LINE del file ASCII contiene i dati per il nome della linea di prodotti e gli obiettivi di vendita annuali per l'anno passato e per quello corrente.

In secondo luogo, Chris traduce il modello di dati finale tratto dalla sua discussione con Helen in un insieme di tabelle nelle quali le colonne corrispondono agli attributi di dati e le righe ai diversi insiemi di valori per questi attributi. Le tabelle sono i mattoni di una base di dati relazionale (che è poi lo stile tipico di Microsoft Access). Le definizioni delle tabelle PRODUCT LINE e PRODUCT create da Chris, compresi gli attributi di dati associati, sono mostrate nelle Figure 4.12 e 4.13 (la Figura 4.13 contiene entrambe le definizioni SQL e di Access). Chris ha effettuato questa traduzione così che ogni tabella abbia un attributo, chiamato *chiave primaria*, che sarà univoco per ciascuna riga nella tabella. Le altre caratteristiche principali di ogni tabella sono che c'è un solo valore per ogni attributo in ciascuna riga, e se si conosce il valore dell'identificatore, può esserci un solo valore per ciascuno degli altri attributi. Per esempio, in qualsiasi linea di prodotti, può esserci un solo valore per l'obiettivo di vendita dell'anno corrente.

Figura 4.12

Definizione SQL
della tabella
PRODUCT_LINE.

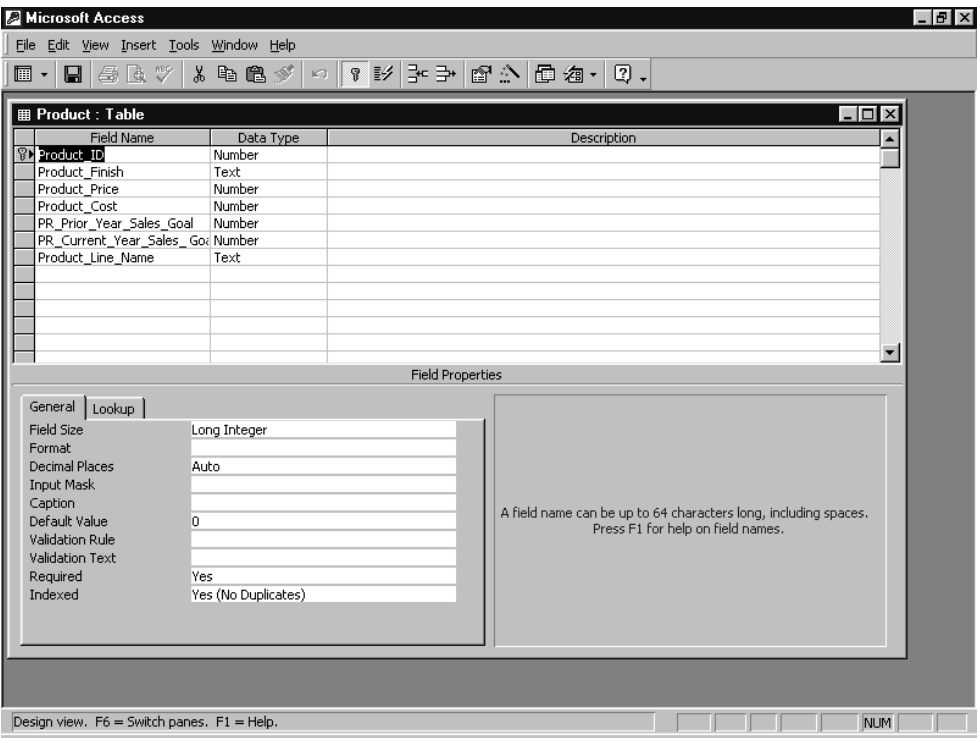
```
CREATE TABLE PRODUCT_LINE
  (PRODUCT_LINE_NAME          VARCHAR (40)  NOT NULL PRIMARY KEY,
   PL_PRIOR_YEAR_GOAL          DECIMAL,
   PL_CURRENT_YEAR_GOAL        DECIMAL);
```

Figura 4.13

Definizione della
tabella PRODUCT
utilizzando SQL
o Access.

```
CREATE TABLE PRODUCT
  (PRODUCT_ID                  INTEGER NOT NULL PRIMARY KEY,
   PRODUCT_FINISH              VARCHAR (20),
   PRODUCT_PRICE               DECIMAL,
   PRODUCT_COST                DECIMAL,
   PR_PRIOR_YEAR_GOAL          DECIMAL,
   PR_CURRENT_YEAR_GOAL        DECIMAL,
   PRODUCT_LINE_NAME           VARCHAR (40),
  FOREIGN KEY (PRODUCT_LINE_NAME) REFERENCES
    PRODUCT_LINE (PRODUCT_LINE_NAME));
```

(a) Definizione SQL.
(b) Definizione
di Access.



Il progetto della base di dati include la specifica del formato, o proprietà, per ogni attributo (che in Access è chiamato *campo*). In questo decidere è facile, dato che la maggior parte degli attributi è già specificata nel dizionario dei dati dell'azienda. Nella Figura 4.13b, vengono mostrate le proprietà di un campo, Product_ID, indicate con la freccia a sinistra del nome del campo. Alcuni attributi della tabella ORDER, come Order_Number_of_Shipments sono stati calcolati da dati grezzi delle basi di dati di Pine Valley, quindi Chris ha dovuto creare le specifiche per ognuno di essi.

L'altra decisione importante che Chris ha dovuto prendere nella progettazione della base di dati è come organizzarla fisicamente per rispondere nel modo più veloce possibile alle interrogazioni che Helen scriverà. Poiché la base di dati verrà utilizzata nel supporto alle decisioni, né Chris né Helen possono prevedere tutte le interrogazioni che sorgeranno; Chris deve quindi decidere in base alla propria esperienza più che alla conoscenza precisa del modo in cui la base di dati verrà utilizzata. La decisione fondamentale che Microsoft Access e SQL permettono di prendere riguarda gli attributi sui quali creare gli indici (un *indice* è come la scheda dell'archivio di una biblioteca, attraverso la quale si possono trovare rapidamente i volumi con caratteristiche simili). Tutti gli attributi di chiave primaria (come Order_Number nel caso della tabella ORDER), quelli con valori univoci tra le righe della tabella, vengono indicizzati. Chris utilizza inoltre una regola generale, ovvero quella di creare un indice per ciascun attributo che ha più di 10 valori diversi e che Helen potrebbe utilizzare per segmentare la base di dati. Per esempio, Helen ha indicato che uno dei modi nei quali vorrebbe utilizzare la base di dati è quello di cercare le vendite in base alla rifinitura del prodotto. Può quindi avere senso creare un indice della tabella PRODUCT utilizzando l'attributo Product_Finish. Tuttavia Pine Valley utilizza solo sei rifiniture, o tipi di legno, quindi questo non è un candidato utile per un indice. D'altra parte, Order_Placement_Date (chiamata *chiave secondaria*, poiché ci può essere più di una riga nella tabella ORDER con lo stesso valore di questo attributo), che Helen vuole utilizzare per analizzare le vendite in periodi diversi, è un buon candidato per un indice. Si osservi la Figura 4.13a per vedere la clausola di chiave primaria, e la Figura 4.13b per trovare un'immissione nel campo Indexed in basso nello schermo e per vedere se l'attributo selezionato è indicizzato e, in caso positivo, se l'indice consente duplicati (una chiave primaria non può avere duplicati).

La Figura 4.14 mostra il modello di dati di prototipo sviluppato da Chris per la base di dati dei mobili per ufficio. Ogni riquadro rappresenta una tabella nella base di dati; gli attributi di una tabella sono elencati all'interno del riquadro associato. Sebbene la notazione per le relazioni sia leggermente diversa da quella della Figura 4.11, il significato è lo stesso. Per esempio, la linea che va dall'entità CUSTOMER all'entità ORDER nella Figura 4.12 mostra un 1 vicino a CUSTOMER e un segno di infinito accanto a ORDER, che ha lo stesso significato dell'associazione *Places* nella Figura 4.11.

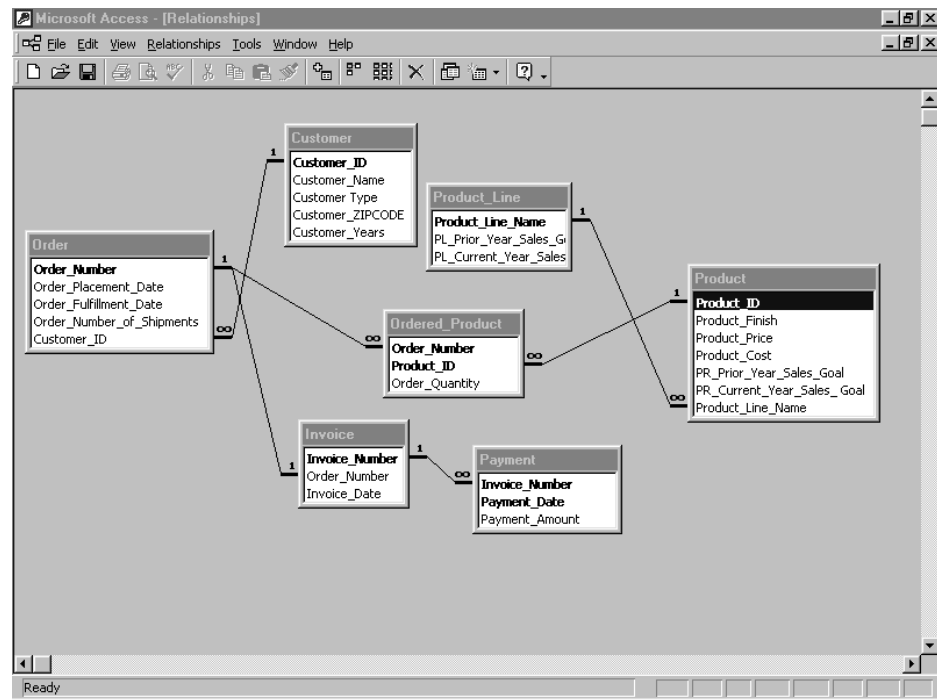
4.6.4 Utilizzare la base di dati

Helen utilizzerà la base di dati che Chris ha costruito soprattutto per domande ad hoc, quindi Chris le fornirà una preparazione su Microsoft Access, e in particolare sulle funzioni di interrogazione di una base di dati. Chris vuole rimandare questa formazione a dopo che il sistema sarà meglio specificato e che Helen avrà imparato qualcosa sulla base di dati e su Microsoft Access dal processo di creazione del prototipo. Helen ha tuttavia indicato alcune domande standard che prevede di porre periodicamente alla base di dati. Microsoft Access fornisce alcune funzioni che consentono a Chris di sviluppare diversi tipi di routine pre-scritte per permettere a Helen di rispondere facilmente a queste domande standard (così che non debba programmarle ogni volta da zero).

- **Maschera (o form):** un insieme di attributi, in un formato predeterminato, tutti basati su un singolo record della base di dati. Una maschera può contenere i dati su un cliente o su un ordine specifico e l'attributo del cliente associato a quell'ordine.

Figura 4.14

La definizione della base di dati per il sistema di supporto alle decisioni di marketing della linea di prodotti di mobili per ufficio (prototipo di modello costruito in Access).



- **Report**: un insieme di attributi, in un formato predeterminato, basati su più record non correlati. Un report contiene solitamente gli stessi attributi su ciascun record. Per esempio, un report può elencare l'identificatore del prodotto, le vendite dell'anno corrente e gli obiettivi di vendita per l'anno corrente di tutti i prodotti per i quali le vendite non raggiungono gli obiettivi.
- **Interrogazione** (o *query*): una domanda a cui deve rispondere la base di dati, posta nel particolare linguaggio di interrogazione Query-By-Example. Il risultato di un'interrogazione è una tabella nella quale le colonne sono gli attributi che l'utente vuole vedere e le righe sono istanze diverse di questi attributi che soddisfano i criteri di ricerca immessi dall'utente.

Durante il processo di sviluppo del prototipo, Chris può aver sviluppato diversi esempi di ciascuna di queste routine man mano che Helen ha comunicato con più chiarezza cosa vuole che il sistema sia in grado di fare. In questa fase iniziale, tuttavia, Chris vuole sviluppare una routine per creare il primo prototipo. Uno degli insiemi di informazioni desiderati da Helen è un elenco di prodotti della linea di mobili per ufficio nel quale vengano mostrate le vendite totali per ciascun prodotto confrontate con gli obiettivi di vendita dell'anno corrente rispettivo. Un'interrogazione di Access può produrre questo risultato. Helen può desiderare che il risultato dell'interrogazione venga visualizzato in un modo gradevole (un'opportunità per utilizzare un report), ma per il momento Chris presenterà a Helen una semplice interrogazione.

L'interrogazione per produrre questo elenco di prodotti è mostrata nella Figura 4.15, con un output di esempio nella Figura 4.16. L'interrogazione nella Figura 4.15 mostra l'associazione del modello di dati necessario per questa interrogazione in alto e il codice Query-By-Example in basso. Se nella casella Show è selezionato un campo, allora quel campo verrà incluso nel risultato. Due campi, Product_Line_Name e Order_Placement_Date, vengono utilizzati solo per selezionare i mobili per ufficio e gli ordini

Figura 4.15
L'interrogazione di confronto
sull'obiettivo di vendita dei mobili
per ufficio.

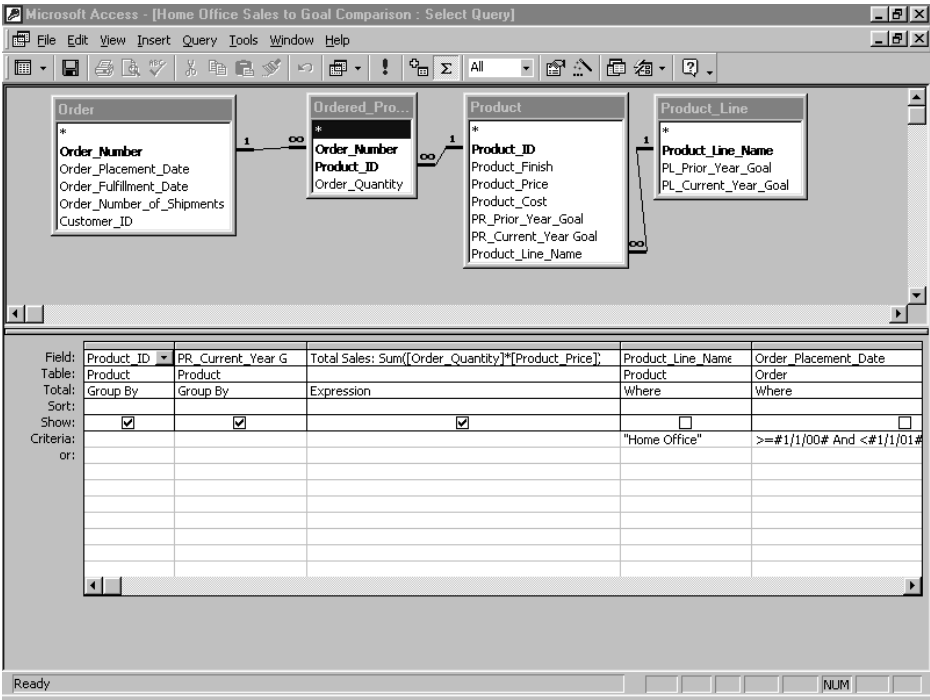


Figura 4.16
Il confronto delle vendite per la
linea di prodotti di mobili per ufficio.

	Product_ID	PR_Current_Year_Sales	Total Sales
▶	3	\$23,500.00	9375
	5	\$26,500.00	4550
	7	\$17,000.00	2250
	10	\$22,500.00	4400

per l'anno corrente, il 2000. La frase Group By su PR_Current_Year_Goal e il campo Product_ID dicono ad Access di sommare le vendite totali per ciascun prodotto. In questo esempio sono inclusi solo dati limitati, quindi il risultato delle vendite totali (Total Sales) mostrato nella Figura 4.16 è piuttosto piccolo, ma il formato è il risultato dell'interrogazione nella Figura 4.15.

Ora Chris è pronto per incontrare nuovamente Helen per vedere se il prototipo inizia a soddisfare le sue esigenze. Chris mostra a Helen il sistema (avviando Access e visualizzando schermate come quelle mostrate nelle figure dalla 4.14 alla 4.16). Man mano che Helen dà alcuni suggerimenti, Chris è in grado di apportare qualche modifica on-line, ma molte delle sue osservazioni necessiteranno di un lavoro più approfondito.

Lo spazio non ci permette di rivedere l'intero progetto del sistema di supporto al marketing per i mobili per ufficio. Chris e Helen si sono incontrati una decina di volte prima che lei fosse certa che tutti gli attributi che le occorreavano fossero nella base di dati, che le interrogazioni standard, le maschere e i report scritti da Chris potessero esserle utili e che fosse in grado di scrivere interrogazioni per le domande potenziali. Chris sarà sempre disponibile per fornire a Helen una consulenza qualora lei si

trovasse in difficoltà, eventualmente anche per scrivere interrogazioni, maschere e report più complessi. L'ultima decisione presa da Chris e Helen riguarda il fatto che le prestazioni del prototipo finale erano sufficientemente buone da non richiedere che questo fosse riscritto o riprogettato. Ora Helen è pronta per utilizzare il sistema.

4.6.5 Amministrare la base di dati

L'amministrazione del sistema di supporto al marketing per i mobili per ufficio è piuttosto semplice. Helen ha stabilito di poter iniziare con alcuni download settimanali di dati dalle basi di dati di Pine Valley nella sua base di dati di Access. Chris ha scritto un programma in C incorporando alcuni comandi SQL così da eseguire le estrazioni necessarie e ha scritto un programma Access in Visual Basic per ricostruire le tabelle di Access da queste estrazioni; ha quindi programmato queste operazioni perché vengano avviate ogni domenica sera. Infine ha aggiornato il modello di architettura del sistema informativo aziendale in modo da includere il supporto al marketing dei mobili per ufficio. Questo passaggio è molto importante: quando si verificano delle modifiche ai formati per i dati inclusi anche nel sistema di Helen, lo strumento CASE aziendale può avvertire Chris che queste modifiche devono essere apportate anche nel sistema di Helen.

Riepilogo

Questo capitolo ha trattato il processo di sviluppo delle basi di dati e delle applicazioni che le utilizzano. Lo sviluppo di una base di dati inizia con la modellazione dei dati aziendale, dove vengono stabiliti l'ambito e i contenuti generali di un'azienda. La modellazione dei dati aziendale è parte di un processo globale che sviluppa un'architettura dei sistemi informativi (compresi dati, processi, reti, eventi e ragionieri) di un'azienda. Una metodologia di sviluppo diffusa per sviluppare questa architettura è l'ingegneria delle informazioni, un approccio top-down di pianificazione dei sistemi informativi.

La pianificazione dei sistemi informativi deve considerare gli obiettivi dell'azienda, i fattori critici di successo e le aree problematiche dell'azienda. Durante la pianificazione, le entità di dati devono essere correlate ad altri oggetti di pianificazione: unità dell'azienda, funzioni aziendali e sistemi informativi (applicazioni). Le funzioni aziendali possono essere rappresentate a livelli diversi di dettaglio suddividendole mediante un processo chiamato scomposizione. Le associazioni tra le entità di dati e gli altri oggetti possono essere rappresentate a un livello più alto pianificando le matrici, che possono essere manipolate per capire i pattern delle associazioni.

Una volta identificata la necessità di una base di dati sia dalla pianificazione dei sistemi informativi sia da una richiesta specifica (come quella di Helen Jarvis per un sistema di supporto al marketing dei mobili per ufficio), viene formato un team di progetto che svilupperà tutti gli elementi. Il team segue un processo di sviluppo del sistema, come il ciclo di vita di sviluppo dei sistemi o la creazione di un prototipo. Il ciclo di vita di sviluppo dei sistemi può essere rappresentato da sette passaggi metodici: 1) identificazione e selezione del progetto; 2) avvio e pianificazione del progetto; 3) analisi; 4) progettazione logica; 5) progettazione fisica; 6) implementazione; 7) manutenzione. Le attività di sviluppo di una base di dati avvengono in ciascuna di queste fasi, che si sovrappongono, e si possono avere dei feedback che riportano il progetto a una fase precedente. Nella creazione di un prototipo, una base di dati e le sue applicazioni vengono raffinate ripetutamente attraverso una stretta interazione tra sviluppatori del sistema e utenti. La creazione di un prototipo funziona al meglio quando l'applicazione della base di dati è piccola e indipendente e quando gli utenti sono pochi.

Nel processo di sviluppo, vengono utilizzati gli strumenti CASE per sviluppare modelli di dati e per mantenere i metadati per la base di dati e le applicazioni. Un repository mantiene tutta la documentazione. Più persone possono utilizzare gli strumenti CASE e il repository associato durante il progetto di sviluppo; analisti di sistema, analisti di basi di dati, utenti, programmatori, amministratori dei dati e delle basi di dati e altri tecnici. Quando una nuova parte significativa del progetto viene completata e vengono effettuate delle immissioni nel repository, è necessario un momento di verifica che permette a chi lavora sul progetto e a chi finanzia le risorse umane ed economiche del progetto di confermarne il progresso e di rinnovare l'impegno su di esso in base al completamento delle varie tappe.

Chi lavora su un progetto di sviluppo di una base di dati ha a che fare con tre viste, o schemi, della base di dati: 1) schema concettuale, che fornisce un'immagine della base di dati completa e tecnologicamente indipendente; 2) uno schema fisico o interno, che specifica la base di dati completa così come verrà memorizzata nella memoria secondaria del computer; 3) schemi esterni, o viste utente, che descrivono un sottoinsieme della base di dati importante per un determinato insieme di utenti.

Una base di dati moderna e le applicazioni che la utilizzano possono trovarsi su più computer. Sebbene possano esistere più livelli da 1 in avanti, tre livelli di computer si rifanno all'architettura client/server per l'elaborazione della base di dati: 1) il livello client, dove il contenuto della base di dati viene presentato all'utente; 2) il livello applicazione/server Web (*application server/Web server*), dove vengono effettuate le analisi sul contenuto della base di dati e dove vengono gestite le sessioni degli utenti; 3) il livello del server di dati (*database server*), dove i dati dell'azienda vengono uniti in un'unica posizione.

Il capitolo si chiude con un progetto ipotetico di sviluppo di una base di dati per Pine Valley Furniture Company. Questo sistema di supporto al marketing della linea di produzione di mobili per ufficio illustra il sistema di gestione di una base di dati personale e la codifica Microsoft Access e SQL per lo sviluppo di una base di dati di solo recupero. La base di dati di questa applicazione contiene i dati estratti dalle basi di dati dell'azienda e li memorizza in una base di dati separata sul livello client. Per sviluppare questa applicazione di base di dati è stata utilizzata la creazione di un prototipo, poiché l'utente, Helen Jarvis, ha esigenze non strutturate che potrebbero essere meglio definite attraverso il processo iterativo di sviluppo e raffinamento del sistema.

Ripasso del capitolo

Parole chiave

Architettura client/server
Architettura informativa (ISA)
Ciclo di vita dello sviluppo
dei sistemi (SDLC)

*Computer-Aided Software
Engineering (CASE)*
Creazione di un prototipo
Repository
Funzione aziendale

Conferma progressiva
Ingegneria delle informazioni
Modellazione dei dati aziendali
Pianificazione top-down

Progetto
Schema concettuale
Schema fisico
Scomposizione funzionale

Domande di ripasso

- Definire ciascuno dei seguenti termini chiave:
 - architettura informativa (ISA)
 - ciclo di vita dello sviluppo dei sistemi (SDLC)
 - architettura client/server
 - conferma progressiva
 - modello dei dati aziendali
 - modellazione dei dati concettuale
- Unire i seguenti termini alle rispettive definizioni:

___ schema concettuale; ___ funzione aziendale;
___ creazione di un prototipo; ___ ciclo di vita dello sviluppo
dei sistemi; ___ scomposizione funzionale; ___ pianificazione
top-down; ___ conferma progressiva; ___ schema fisico.

 - suddividere ripetutamente una funzione in un dettaglio sempre maggiore
 - momenti periodici di verifica in un progetto di sviluppo dei sistemi

- c. un approccio rapido allo sviluppo dei sistemi
 - d. una descrizione completa dei dati aziendali
 - e. ottenere una comprensione estesa delle esigenze dei sistemi informativi
 - f. un gruppo correlato di processi aziendali
 - g. la struttura dei dati memorizzati nella memoria secondaria
 - h. un approccio strutturato e progressivo allo sviluppo dei sistemi
3. Confrontare i seguenti termini:
 - a. schema fisico; schema concettuale
 - b. ciclo di vita dello sviluppo dei sistemi; creazione di un prototipo
 - c. pianificazione top-down; scomposizione funzionale
 - d. modellazione dei dati aziendali; ingegneria delle informazioni
 - e. repository; Computer-Aided Software Engineering
 4. Elencare e spiegare i quattro passaggi dell'ingegneria delle informazioni.
 5. Descrivere i tre passaggi della fase di pianificazione dell'ingegneria delle informazioni.
 6. Elencare e spiegare i tre fattori di pianificazione strategici dell'ingegneria delle informazioni.
 7. Elencare e definire i cinque oggetti principali della pianificazione aziendale.
 8. Spiegare il significato della scomposizione funzionale nello sviluppo dei sistemi informativi e delle basi di dati.
 9. Spiegare l'uso delle matrici di pianificazione dei sistemi informativi nello sviluppo di questi ultimi.
 10. Citare le sette fasi del ciclo di vita di sviluppo dei sistemi tradizionale e spiegare lo scopo del risultato di ogni fase.
 11. In quale delle sette fasi di SDLC si hanno le attività di sviluppo di una base di dati?
 12. Definire i passaggi del processo di sviluppo dei sistemi nella creazione di un prototipo. Quali attività si verificano in ciascuna fase della creazione di un prototipo?
 13. Spiegare le differenze tra le viste utenti, uno schema concettuale e uno schema fisico e le diverse viste della stessa base di dati.
 14. La progettazione degli schemi esterni e di uno schema concettuale avvengono in una sequenza particolare? Perché?
 15. Definire l'architettura a tre livelli di una base di dati.
 16. Nell'architettura a tre livelli di una base di dati è possibile che non ci siano basi di dati a un particolare livello? Se no, perché? Se sì, fornire un esempio.
 17. Elencare i sei componenti chiave di un'architettura informativa(ISA). Quali di questi componenti si riferiscono alla progettazione della base di dati? Perché?
 18. Qual è il significato della metafora della cascata in SDLC?

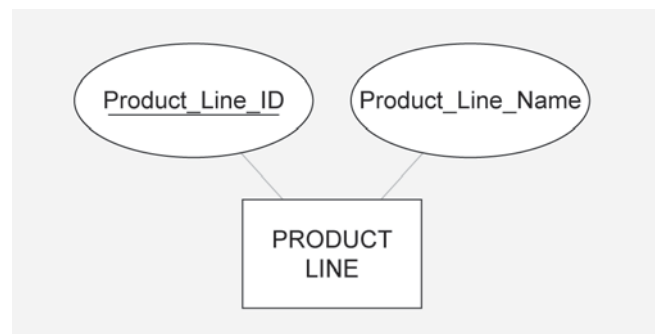
19. Durante il processo di sviluppo di una base di dati, quando deve essere disegnato il diagramma entità-associazione? In cosa differiscono questi diagrammi tra loro anche quando rappresentano una stessa base di dati?

20. Quali sono le ragioni per implementare un'architettura di base di dati client/server?

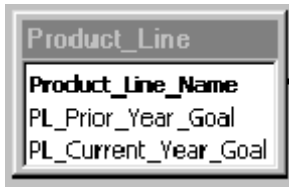
Problemi ed esercizi

1. La metafora della cascata è una buona rappresentazione del ciclo di vita di sviluppo dei sistemi? Spiegare gli eventuali problemi legati a questa metafora.
2. Modificare la disposizione delle righe e delle colonne della Figura 4.3 in una sequenza più utile. Perché si è scelta questa nuova disposizione? A quale scopo si può ora utilizzare la nuova matrice?
3. Elencare tre entità supplementari che possono apparire in un modello di dati aziendale per Pine Valley Furniture Company (Figura 4.1).
4. Si prenda la propria università come esempio di azienda.
 - a. Definire numerose funzioni ed effettuare una scomposizione funzionale ad almeno tre livelli.
 - b. Definire più tipi di entità importanti e disegnare un modello di dati aziendale preliminare (simile nella notazione alla Figura 4.1).
 - c. Sviluppare una matrice di pianificazione con le funzioni di livello più basso della parte a. dell'esercizio come righe e i tipi di entità di dati della parte b. come colonne. Riempire le celle di questa matrice in modo simile alla Figura 4.3.
 - d. Definire quattro fattori critici di successo (CFS) per l'università.
 - e. L'università trarrà beneficio da un'architettura dei dati a livelli multipli? Perché?
5. Considerare un club di studenti di cui si fa parte.
 - a. Definire più sistemi informativi (manuali o automatizzati) utilizzati dal club.
 - b. Definire più tipi di entità importanti e disegnare un modello di dati aziendale preliminare (simile nella notazione alla Figura 4.1).
 - c. Sviluppare una matrice di pianificazione sistemi informativi-a-entità di dati. Riempire le celle per mostrare come ogni sistema informativo interagisce con ogni entità di dati (codificare la cella con una C per creare nuove istanze dell'entità, con una R per recuperare i dati su quell'entità, con una U per i valori di aggiornamento dei dati dell'entità e con una D per eliminare istanze dell'entità).
 - d. Riorganizzare le righe e le colonne della risposta data al punto c. di questo esercizio per creare una matrice con celle con immissioni lungo la diagonale principale e per svuotare le celle lungo la stessa diagonale. Cosa emerge dalla matrice?

6. Si immagini una matrice di pianificazione funzioni aziendali-a-entità di dati. Si supponga che studiando questa matrice si determini che tre delle funzioni aziendali forniscono il cuore dell'uso di cinque delle entità di dati. Quali sono le possibili implicazioni sull'identificazione delle basi di dati per lo sviluppo?
7. Si consideri la Tabella 4.3. Sviluppare una matrice ipotetica sistemi informativi-a-entità di dati dalle informazioni contenute nella tabella. Si assuma quanto i sistemi creano, recuperano, aggiornano ed eliminano da ciascuna entità di dati. Codificare le celle per indicare la creazione (C), il recupero (R), l'aggiornamento (U) e l'eliminazione (D). Poiché i sistemi informativi elencati includono sia sistemi di elaborazione delle transazioni sia di gestione, quali pattern di uso delle entità emerge?
8. Si consideri la Tabella 4.3. Sviluppare una scomposizione funzionale ipotetica della funzione aziendale Product simile a quella mostrata nella Figura 4.2. Nella figura e nel vostro diagramma appaiono alcune sotto-funzioni? Perché?
9. Spiegare le differenze tra un modello di dati aziendale e uno concettuale. Quante basi di dati vengono rappresentate da ciascuno? A quale ambito dell'azienda si rivolgono? Quali sono altre differenze salienti?
10. È possibile che durante la progettazione fisica della base di dati e la fase della creazione si possa voler tornare all'attività di progettazione logica della base di dati? Perché? Se è possibile, fornire un esempio di cosa potrebbe accadere durante la progettazione fisica e la creazione che potrebbe richiedere di dover riconsiderare i progetti concettuali ed esterni dei passaggi precedenti.
11. Confrontare la natura top-down dello sviluppo di una base di dati durante la modellazione concettuale con la natura bottom-up durante la fase di progettazione logica. Quali differenze principali esistono nel tipo di informazioni considerate in ciascuna di queste due fasi di sviluppo della base di dati?
12. L'obiettivo della metodologia di creazione di un prototipo è quello di costruire e ricostruire rapidamente un sistema informativo man mano che l'utente e gli analisti di sistema apprendono dall'uso del prototipo quali funzioni dovrebbero essere incluse nel sistema informativo in via di evoluzione. Poiché il prototipo finale non deve diventare il sistema funzionante, qual è la posizione ideale in cui svilupparlo: un personal computer, un computer di un gruppo di lavoro, un computer di reparto o il server dell'impresa? Da cosa dipende la risposta?
13. Si consideri una qualsiasi azienda con la quale si interagisce di frequente, come una banca, una società di carte di credito, un'università o una compagnia di assicurazioni, dalla quale si ricevono numerosi messaggi generati dal computer, come comunicazioni mensili, estratti conto e così via. Rappresentare i dati inclusi in ciascun messaggio ricevuto come propria vista utente utilizzando la notazione della Figura 4.1. Combinare quindi queste viste in un unico modello di dati concettuale, utilizzando sempre la stessa notazione. Cosa si può notare dalla combinazione di viste utente diverse? Si verificano incoerenze tra le viste? Una volta creato il modello di dati concettuale, risulta preferibile modificare qualche aspetto di una qualsiasi di queste viste utente?
14. Si consideri la Figura 4.10, che illustra un'architettura ipotetica di base di dati a tre livelli. Identificare duplicazioni potenziali dei dati tra le basi di dati elencate nella figura. Quali problemi possono sorgere dalla duplicazione?
15. Si consideri la Figura 4.11. Spiegare il significato della linea che connette ORDER a INVOICE e di quella che connette INVOICE a PAYMENT. Cosa ci dice sul modo in cui Pine Valley fa affari con i suoi clienti?
16. Rispondere alle seguenti domande riguardanti le Figure 4.12 e 4.13.
 - a. Quale sarà la dimensione del campo Product_Line_Name nella tabella PRODUCT? Perché?
 - b. Nella Figura 4.13b, perché si specifica che è necessario il campo Product_ID nella tabella PRODUCT?
 - c. Nella Figura 4.13b, perché accanto a Product_ID nella tabella PRODUCT c'è una chiave?
 - d. Qual è lo scopo dell'ultima riga dell'istruzione SQL nella Figura 4.13a?
17. Si consideri l'interrogazione di Access nella Figura 4.15.
 - a. Perché è necessario includere il campo Order_Placement_Date in questa interrogazione anche se non viene visualizzato nei risultati dell'interrogazione?
 - b. Come cambierebbe l'interrogazione se Helen Jarvis volesse vedere i risultati di tutte le linee di prodotti e non solo di quella dei mobili per ufficio?
18. Nella Figura 5.22 nel Capitolo 5, si può vedere un diagramma entità-associazione per una parte della base di dati di Pine Valley. L'entità PRODUCT LINE è rappresentata come segue:



La bozza preparata da Chris per Helen mostra quanto segue per PRODUCT LINE:



Spiegare le differenze tra le due definizioni di PRODUCT LINE.

Esercitazioni

1. Intervistare gli analisti di sistema e di basi di dati di più aziende. Chiedere loro di descrivere il loro processo di sviluppo dei sistemi. È più simile a quello dei sistemi del ciclo di vita o a quello di creazione di prototipi? Utilizzano metodologie simili a entrambe? Quando utilizzano metodologie diverse? Studiare la metodologia impiegata per sviluppare applicazioni da utilizzare attraverso il Web. Come hanno modificato la loro metodologia per adattarsi a questo nuovo processo di sviluppo dei sistemi?
2. Scegliere un'azienda con cui si ha familiarità, come quella in cui si lavora, in cui si studia o in cui lavora qualche amico. Descrivere la sua ISA. L'azienda ha un'architettura preparata formalmente o è necessario crearne una? Intervistare i manager dei sistemi informativi dell'azienda per scoprire se hanno o non hanno un'architettura informativa formalmente riconosciuta.
3. Scegliere un'azienda con cui si ha familiarità, come quella in cui si lavora, in cui si studia o in cui lavora qualche amico. Si prenda in considerazione una base di dati principale di questa azienda, come quella che supporta le interazioni dei clienti, la contabilità o la produzione. Qual è l'architettura di questa base di dati? L'azienda utilizza un'architettura client/server in qualche forma? Intervistare i manager dei sistemi informativi dell'azienda per scoprire perché hanno scelto questa architettura per la base di dati.
4. Scegliere un'azienda con cui si ha familiarità, come quella in cui si lavora, in cui si studia o in cui lavora qualche amico. Intervistare un analista dei sistemi o delle basi di dati e porgli delle domande sulla composizione tipica di un team di sviluppo dei sistemi informativi. Nello specifico, quale ruolo riveste un

analista delle basi di dati nelle squadre di progetto? Ci si serve di lui in tutto il processo di sviluppo o solo in alcune fasi?

5. Scegliere un'azienda con cui si ha familiarità, come quella in cui si lavora, in cui si studia o in cui lavora qualche amico. Intervistare un analista dei sistemi o delle basi di dati e porgli delle domande su come l'azienda utilizza gli strumenti CASE nel processo di sviluppo. Concentrare le domande su come gli strumenti CASE vengono utilizzati per supportare la modellazione dei dati e la progettazione della base di dati e su come il loro repository mantiene le informazioni raccolte sui dati, le caratteristiche dei dati e l'uso degli stessi. Se vengono utilizzati più strumenti CASE in uno o più progetti, chiedere come l'azienda cerca di integrare i modelli di dati e le definizioni dei dati. Infine, verificare quanto gli analisti dei sistemi e delle basi di dati sono soddisfatti del supporto degli strumenti CASE alla modellazione dei dati e alla progettazione delle basi di dati.

Bibliografia

- Hoffer, J. A., George, J. F. e Valacich, J. S., 2002, *Modern Systems Analysis and Design*, terza edizione, Upper Saddle River, NJ, Prentice Hall.
- Moriarty, T., 1991, "Framing Your System", *Database Programming & Design* 4 (giugno), pp. 38–43.
- Sowa, J. F. e Zachman, A., 1992, "Extending and Formalizing the Framework for Information Systems Architecture", *IBM Systems Journal* 31 (3), pp. 590–616.
- Thompson, C., 1997, "Committing to Three-Tier Architecture", *Database Programming & Design* 10 (agosto), pp. 26–30, 32, 33.
- Zachman, J. A., 1987, "A Framework for Information Systems Architecture", *IBM Systems Journal* 26 (marzo), pp. 276–292.

Per approfondire

- Finkelstein, C., 1989, *An Introduction to Information Engineering*, Reading, MA, Addison-Wesley.
- Shank, M. E., Boynton, A. C. e Zmud, R. W., 1985, "Critical Success Factor Analysis as a Methodology for IS Planning", *MIS Quarterly* 9 (giugno), 121–129.

Project Case



MOUNTAIN VIEW COMMUNITY HOSPITAL

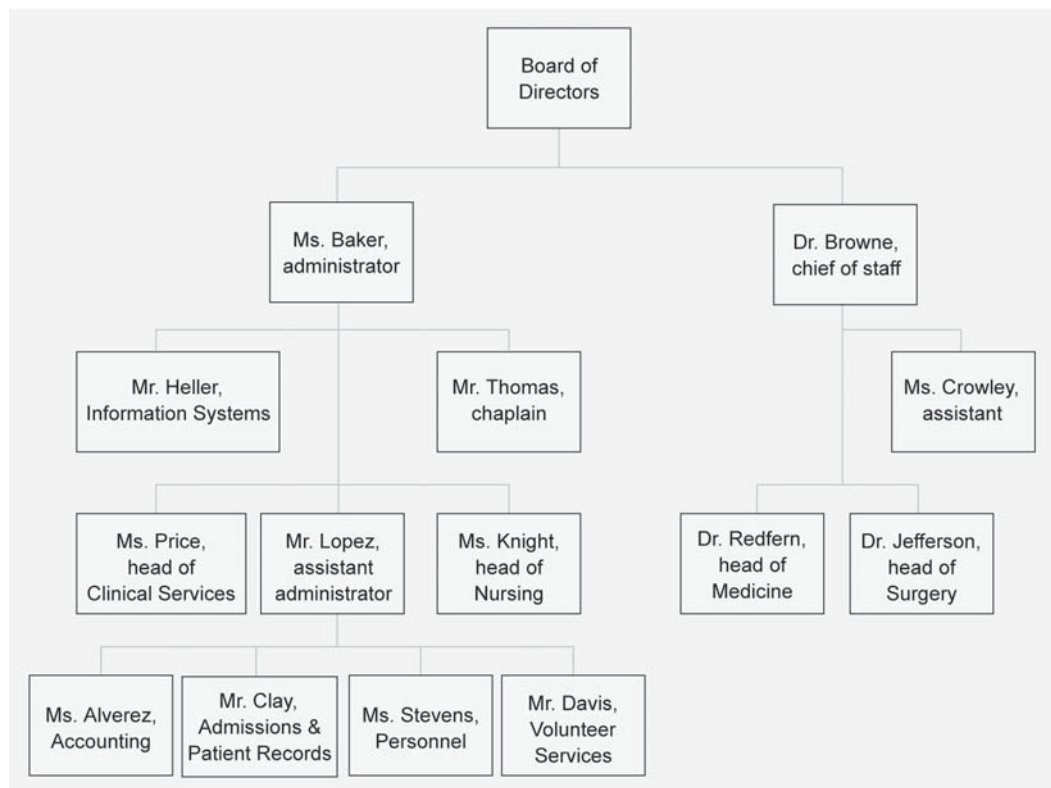
Descrizione del progetto

Nel capitolo precedente è stato presentato il Mountain View Community Hospital. La Figura 1 mostra un organigramma dell'ospedale aggiornato al 1 gennaio 2001. Come molti ospedali, anche questo è diviso in due divisioni principali. I medici, capeggiati dal Dr. Browne (capo del personale), sono responsabili della qualità delle cure mediche fornite ai pazienti. Il gruppo capeggiato da Ms. Baker (amministratrice) fornisce il supporto assistenziale, clinico e amministrativo ai medici che devono occuparsi dei pazienti.

Obiettivi e fattori critici di successo

Come detto nel Capitolo 3, lo scopo principale di Mountain View Community Hospital è quello di fornire assistenza medica di alta qualità alla comunità che lo circonda contenendo i costi che sono cresciuti in base alle tendenze nazionali negli anni recenti. L'ospedale serve una comunità di circa 50.000 persone con un tasso di crescita del 10 per cento, una tendenza che si prevede continuerà, considerato che la zona sta attirando numerosi pensionati. L'obiettivo dell'ospedale è quindi quello di ampliarsi, aggiungendo altri

Figura 1
Organigramma
al 1 gennaio 2001.



50 posti letto in cinque anni, e di aprire un centro di assistenza agli anziani con appartamenti indipendenti e servizi assistiti. Accanto all'ospedale è presente un'area edificabile che permette questo ampliamento. A causa di questo ampliamento, l'ospedale prevede di aumentare il proprio personale medico e amministrativo e di creare un nuovo reparto geriatrico diretto dal Dr. Browne.

In risposta a piani di espansione e crescita stabili, uno speciale team di studio che comprende Mr. Heller, Mr Lopez, il Dr. Jefferson e un consulente, sta sviluppando un piano aziendale a lungo termine che comprende anche un progetto per i sistemi informativi dell'ospedale. Il loro lavoro non è ancora stato completato, ma hanno iniziato a identificare molti degli elementi necessari a creare il piano. Per soddisfare gli obiettivi della cura di altro livello, del contenimento dei costi e dell'espansione in nuovi servizi, il team ha concluso che l'ospedale ha quattro fattori critici di successo (CTF): la qualità della cura medica, il controllo dei costi operativi, il controllo dei costi di capitale e l'assunzione e il mantenimento di personale specializzato (soprattutto infermieri). Lo sviluppo di un sistema informativo migliorato è di aiuto nel raggiungimento di tutti questi fattori.

Il team è attualmente al lavoro per generare da due a quattro obiettivi a breve o lungo termine per ciascun fattore. Ha quindi sviluppato i seguenti quattro obiettivi rispetto al fattore del controllo dei costi operativi.

1. Ridurre i costi per gli articoli acquistati.
2. Programmare più efficientemente il lavoro del personale.
3. Abbassare i costi delle assicurazioni sulle responsabilità.
4. Incrementare il servizio dei volontari.

Il team di studio ha descritto ciascuno di questi fattori e obiettivi in modo più dettagliato e ha memorizzato le descrizioni in un repository gestito dallo strumento CASE utilizzato dai sistemi informativi.

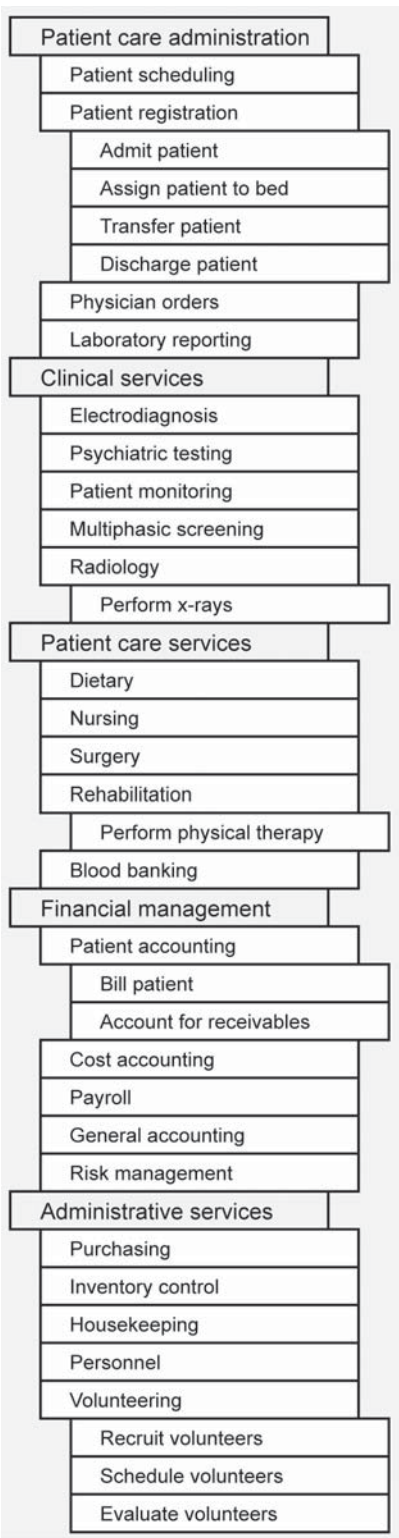
Modellazione aziendale

Il team di studio ha anche sviluppato un elenco preliminare delle funzioni aziendali che descrivono le attività amministrative e mediche dell'ospedale. Queste funzioni considerano gli obiettivi dell'azienda e i fattori importanti spiegati nel paragrafo precedente. Il team ha quindi identificato cinque funzioni aziendali principali che attraversano tutte le unità dell'azienda.

1. *Amministrazione della cura dei pazienti.* Gestire gli aspetti logistici e di registrazione della cura dei pazienti.
2. *Servizi clinici.* Fornire analisi di laboratorio e test, oltre al monitoraggio e allo screening dei pazienti.
3. *Servizi di cura dei pazienti.* Fornire ai pazienti la cura medica e i servizi di assistenza.
4. *Gestione finanziaria.* Gestire le risorse finanziarie e le operazioni dell'ospedale.
5. *Servizi amministrativi.* Fornire una gestione generale e servizi di supporto non legati direttamente alla cura dei pazienti.

Il team di studio è stato in grado di suddividere ciascuna di queste funzioni di alto livello in elenchi di funzioni più dettagliate (Figura 2), ma sono elenchi non ancora completi né ben definiti.

Figura 2
Funzioni aziendali.



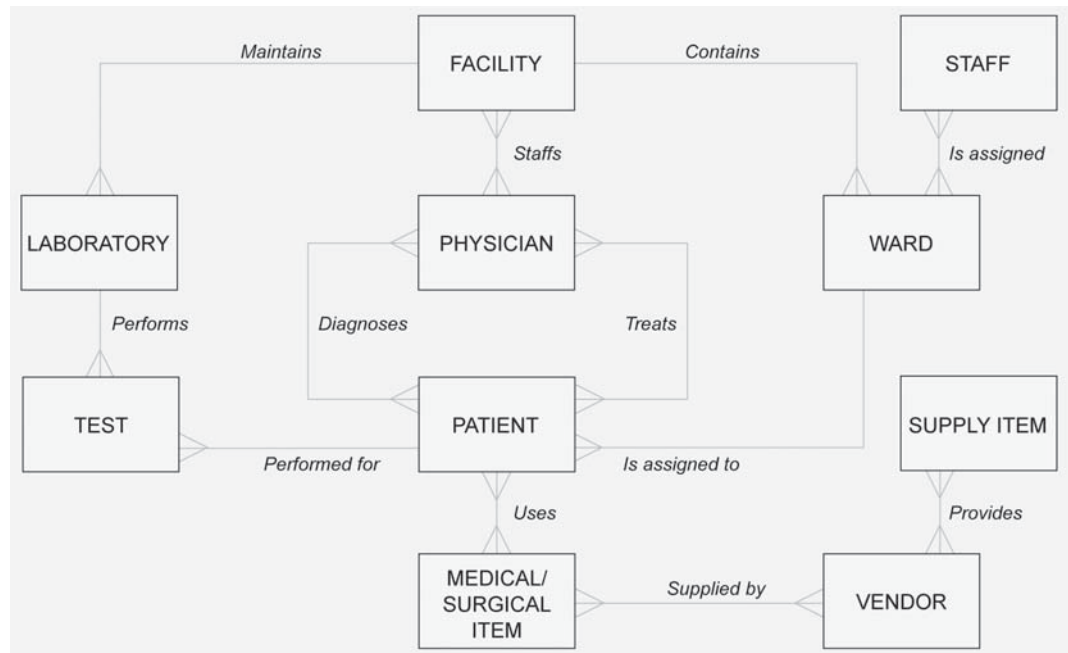
Il team di studio ha individuato inizialmente un insieme preliminare di 10 tipi di entità che descrivono i dati richiesti dall'ospedale e dall'amministrazione: FACILITY, PHYSICIAN, PATIENT, WARD, STAFF, LABORATORY, TEST, MEDICAL/SURGICAL ITEM, SUPPLY ITEM e VENDOR. Dalle discussioni con il personale, dalla revisione dei documenti dell'ospedale e dallo studio dei sistemi informativi esistenti, il team ha sviluppato un elenco di regole aziendali che descrivono le politiche dell'ospedale e la natura delle operazioni che governano le associazioni tra queste entità. Alcune di queste regole sono elencate di seguito.

1. Un FACILITY mantiene un certo numero di LABORATORY: radiologia, elettrodiagnostica, ematologia e così via.
2. Un FACILITY contiene un certo numero di WARD: ostetricia, pronto soccorso, riabilitazione, geriatria e così via.
3. A ciascun WARD è assegnato un certo numero di membri dello STAFF (infermieri, segretarie e così via); un membro dello STAFF può essere assegnato a più WARD.
4. Il FACILITY combina il proprio personale medico con un certo numero di PHYSICIAN. Un PHYSICIAN può essere nel personale di più di un FACILITY.
5. Un PHYSICIAN tratta i PATIENT, e un PATIENT è trattato da un certo numero di PHYSICIAN.

6. Un PHYSICIAN effettua la diagnosi di un PATIENT, e un PATIENT viene sottoposto a diagnosi da un numero qualsiasi di PHYSICIAN.
7. Un PATIENT può essere assegnato a un WARD (i pazienti esterni non vengono assegnati a un WARD). L'ospedale si occupa solo del WARD a cui un paziente è assegnato (se vi sono pazienti).
8. Un PATIENT utilizza MEDICAL/SURGICAL ITEM che vengono forniti dai VENDOR. Un VENDOR può anche fornire SUPPLY ITEM utilizzati per lavori di pulizia o manutenzione.
9. Un LABORATORY effettua dei TEST sui PATIENT.

Il team di studio riconosce che determinate funzioni aziendali, come la gestione dei rischi e i volontari, non sono rappresentati adeguatamente nell'insieme di entità di dati e nelle regole aziendali, ma decide di affrontare questi aspetti successivamente. Le descrizioni delle entità di dati e le regole aziendali vengono memorizzate nel repository CASE per un'analisi successiva. Utilizzando le entità e le regole aziendali identificate, il team ha sviluppato un modello dei dati aziendali preliminare (Figura 3). Anche questo modello è preliminare perché non segue le convenzioni utilizzate dai sistemi informativi per disegnare i modelli di dati, ma il suo scopo è quello di offrire solo una panoramica generale dei dati dell'azienda.

Figura 3
Modello preliminare dei
dati aziendali.



Sviluppare le matrici di pianificazione

Il team di studio ha utilizzato lo strumento CASE per produrre una prima versione di una matrice di tipo funzione aziendale-a-tipo di

entità (Figura 4). La matrice mappa le funzioni aziendali della Figura 2 con le 10 entità di dati. Le celle della matrice sono codificate come segue:

Figura 4
Matrice di tipo funzione aziendale-a-tipo di entità.

Business Functions Data Entity Types	Facility	Physician	Patient	Ward	Staff	Laboratory	Test	Medical/ Surgical Item	Supply Item	Vendor
Patient Scheduling		R	R			R				
Patient Registration			M	R						
Physician Orders		R	R				R			
Laboratory Reporting			R			R	R			
Electrodiagnosis							M			
Psychiatric Testing							M			
Patient Monitoring			R							
Multiphasic Screening			R			R	M			
Radiology						R	M			
Dietary			R							
Nursing			R							
Surgery		R	R							
Rehabilitation			R							
Blood Banking			R			R				
Patient Accounting			R	R		R	R			
Cost Accounting	R		R	R			R			
Payroll		R		R	R					
General Accounting	M			M		M				
Risk Management	R	R	R					R		R
Purchasing								M	M	M
Inventory Control								R	R	
Housekeeping									R	
Personnel		M			M				R	
Volunteering	R		R	R		R				

M = l'entità di dati (colonna) è mantenuta da una funzione aziendale (riga)
R = l'entità di dati (colonna) è utilizzata da una funzione aziendale (riga)

M = la funzione mantiene i dati dell'entità (crea, aggiorna ed elimina)

R = la funzione utilizza i dati sull'entità

Come passaggio iniziale di questa matrice, il team di sviluppo ha deciso di non codificare separatamente le attività di creazione, aggiornamento ed eliminazione.

Inoltre ha utilizzato lo strumento CASE per correlare le funzioni aziendali ai quattro fattori di successo (la Figura 5 mostra il risultato di questo lavoro). In questa figura le celle indicano quanto segue:

E = l'esecuzione corretta della funzione è essenziale per il raggiungimento del fattore di successo

D = l'esecuzione corretta della funzione contribuisce al raggiungimento del fattore di successo

Il team di studio deve cercare i modi per combinare i risultati mostrati nelle Figure 4 e 5 per avere un aiuto nell'impostare alcune proprietà nello sviluppo dei sistemi informativi.

Domande sul progetto

1. Quali matrici di pianificazione dei sistemi informativi (tra quelle mostrate in questo case study o tra altre non mostrate) possono aiutare il team di studio del Mountain View Community Hospital nel terminare un piano di posizionamento a livelli della base di dati? Perché?
2. Nella descrizione di progetto sopra riportata, vengono elencati gli obiettivi per uno dei CFS, ossia il controllo dei costi operativi. Scrivere oggetti plausibili, dati la struttura e i piani dell'ospedale, per gli altri tre CFS.
3. Quali attività supplementari possono avvenire nella funzione aziendale "servizio dei volontari"?
4. Quali attività supplementari possono avvenire nella funzione aziendale "gestione dei rischi"?
5. Quali attività supplementari, oltre alle 10 entità citate in questo caso, possono essere necessarie alla funzione di gestione dei rischi?
6. Si ritiene che il team di studio sarà in grado di identificare tutte le entità di dati con questo tipo di analisi? In caso contrario, quali altri passaggi di sviluppo della base di dati sarà necessario effettuare e quando?
7. Perché si ritiene che per il team di studio siano stati scelti questi membri? Sarebbe stato più utile sceglierne di diversi o degli altri?
8. Nella descrizione di progetto sopra riportata, sono elencate nove regole aziendali. Il team di studio le ha utilizzate per sviluppare la Figura 3. Oltre a queste nove, quali altre regole aziendali sono coinvolte o rappresentate nella Figura 3?
9. Nonostante il controllo dei costi del capitale sia un CFS, solo due delle funzioni aziendali nella Figura 5 lo citano. Perché? Cosa si dovrebbe fare per perfezionare questa figura in modo che più elementi indichino una relazione tra le funzioni aziendali e questo CFS?

Critical Success Factors \ Business Functions	Business Functions			
	Quality Care	Operating Cost Control	Capital Cost Control	Staff Recruitment and Retention
Patient Scheduling	D			
Patient Registration				
Physician Orders	D			
Laboratory Reporting	D			
Electrodiagnosis			D	
Psychiatric Testing	D			
Patient Monitoring	E			
Multiphasic Screening	E			
Radiology	D			
Dietary	E			
Nursing	E	E		E
Surgery	D	D	E	
Rehabilitation	E			
Blood Banking	D			
Patient Accounting	D	E		
Cost Accounting		E		
Payroll				D
General Accounting				
Risk Management		E		D
Purchasing		E		
Inventory Control		E		
Housekeeping		D		
Personnel				E
Volunteering		E		E

E = la funzione aziendale (riga) è essenziale per il raggiungimento del CSF (colonna)
D = la funzione aziendale (riga) è auspicabile per il raggiungimento del CSF (colonna)

Figura 5
Matrice di tipo funzione aziendale-a-CFS.

Esercizi sul progetto

1. Modificare la disposizione delle righe e delle colonne della Figura 4 in modo che le immissioni formino un pattern diagonale a blocchi. Quali conclusioni si raggiungono analizzando questo pattern?
2. Ridisegnare la Figura 1 per riflettere la struttura dell'ospedale dopo l'espansione e la crescita pianificate. Data questa crescita, cos'altro potrebbe essere necessario per modificare le Figure 2, 3 e 4?
3. In base alle Figure 4 e 5, determinare quali entità di dati sono essenziali per il controllo dei costi operativi. Quali implicazioni ha questo risultato nello sviluppo dei sistemi informativi?
4. Nella domanda 5 sul progetto sono state elencate le entità di dati supplementari necessarie per la funzione aziendale di gestione dei rischi. Modificare la Figura 3 per includere queste entità supplementari e le associazioni importanti tra tutte le entità di dati.
5. Uno degli output importanti della funzione aziendale "conti paziente" è Patient Bill. L'esercizio 2 sul progetto del Capitolo 3 ha introdotto una versione ridotta di questa vista, che viene mostrata di seguito. In quell'esercizio veniva chiesto di aggiungere i dati mancanti che apparirebbero sulla fattura di un paziente. Utilizzando il risultato di quell'esercizio, verificare che il modello dei dati aziendali della Figura 3 contenga i dati necessari a generare un conto per il paziente. Spiegare cosa occorre fare per eseguire questa verifica. Cosa emerge dall'analisi?
6. Le attività del team di studio descritte in questo case study si riferivano ai primissimi stadi dello sviluppo dei sistemi informativi e della base di dati. Delineare i passaggi successivi che dovrebbero essere seguiti nell'unità dei sistemi informativi per allineare i sistemi e le basi di dati correnti alle esigenze dei sistemi informativi futuri dell'ospedale.
7. Il manager del settore della gestione dei rischi è ansioso di ricevere un supporto computerizzato alle sue attività. L'ospedale è costretto a fronteggiare lamentele sempre più numerose sul cattivo servizio fornito, e il manager non crede di poter aspettare che i sistemi informativi e la base di dati vengono configurati per avere servizi di informazione migliorati. Ciò che il manager vuole è un sistema che tenga traccia di lamentele, cause legali, avvocati, giudici, personale medico, rimborsi e sentenze. Come affrontare questa richiesta di servizi di informazione migliorati? Quale metodologia si potrebbe applicare per progettare i sistemi e le basi di dati richiesti dal manager? Perché?
8. Si consideri nuovamente la richiesta del manager del settore della gestione dei rischi esposta al punto precedente. A quale o quali livelli è consigliabile sviluppare il sistema e la base di dati richiesti dal manager? Perché?

Patient Name: Dolan, Mark		
Patient Number: _____		
Patient Address: _____		
Item Code	Item Description	Amount
_____	_____	_____
_____	_____	_____
_____	_____	_____

Parte terza
Analisi delle basi di dati



CAPITOLO

5

Modellare i dati dell'azienda

Al termine di questo capitolo sarete in grado di fare quanto segue.

1. Definire i seguenti termini chiave: regola aziendale, termine, fatto, modello entità-associazione (modello E-R), diagramma entità-associazione (diagramma E-R), entità, tipo di entità, istanza di entità, tipo di entità forte, tipo di entità debole, possessore identificativo, associazione identificativa, attributo, attributo composto, attributo semplice, attributo multivalore, attributo derivato, identificatore, identificatore composto, tipo di associazione, istanza di associazione, entità associativa, grado di associazione, associazione unaria, associazione binaria, associazione ternaria, vincolo di cardinalità, cardinalità minima, cardinalità massima e marcatura temporale (*time stamp*).
2. Spiegare le ragioni per cui molti sviluppatori di sistema ritengono che la modellazione dei dati sia la parte più importante del processo di sviluppo del sistema.
3. Scrivere nomi e definizioni validi per le entità, le associazioni e gli attributi.
4. Distinguere tra relazioni unarie, binarie e ternarie e proporre un esempio per ciascuna di esse.
5. Modellare ciascuno dei seguenti costrutti in un diagramma E-R: attributo composto, attributo multivalore, attributo derivato, entità associativa, associazione identificativa e vincoli di cardinalità minima e massima.
6. Disegnare un diagramma E-R per rappresentare situazioni aziendali comuni.
7. Convertire un'associazione multi-a-molti in un tipo di entità associativa.
8. Modellare dati semplici dipendenti dal tempo utilizzando la marcatura temporale in un diagramma E-R.

Introduzione

Nei due capitoli precedenti sono stati introdotti la modellazione dei dati e il modello entità-associazione con alcuni esempi semplificati. In questo capitolo formalizzeremo la modellazione dei dati in base al concetto fondamentale delle regole aziendali e descriveremo il modello di dati entità-associazione (E-R) nel dettaglio.

Le regole aziendali vengono derivate da politiche, procedure, eventi, funzioni, altri oggetti aziendali e vincoli di stato sull'azienda. Le regole aziendali sono importanti nella modellazione dei dati perché governano il modo in cui questi vengono gestiti e memorizzati. Le regole aziendali di base sono i nomi e

le definizioni dei dati. Questo capitolo spiega le linee guida per la denominazione e la definizione chiara degli oggetti di base nell'azienda. In termini di modellazione dei dati concettuali, i nomi e le definizioni devono essere forniti per i tipi di entità, gli attributi e le relazioni. Altre regole aziendali possono definire vincoli su questi oggetti di base, vincoli che possono essere rappresentati in un modello di dati, come un diagramma entità-associazione, e nella documentazione associata.

Dopo alcuni anni d'uso, il modello E-R rimane l'approccio principale per la modellazione di dati concettuale. La sua popolarità nasce da fattori quali una relativa facilità d'uso, un ampio supporto di strumenti CASE e la convinzione che le entità e le associazioni siano concetti di modellazione naturale del mondo reale.

Il modello E-R viene utilizzato soprattutto come strumento di comunicazione tra progettisti di basi di dati e utenti finali durante la fase di analisi dello sviluppo (descritta nel Capitolo 4). Il modello E-R viene utilizzato per creare un modello di dati concettuale, che è una rappresentazione della struttura e dei vincoli di una base di dati che è indipendente dal software (come un sistema di gestione di base di dati) e del modello di dati associato che verrà utilizzato per implementare la base di dati.

Quando trattano la modellazione E-R, alcuni autori introducono termini e concetti peculiari del modello di dati relazionale. In particolare, raccomandano che il modello E-R sia completamente normalizzato, con una soluzione completa delle chiavi primarie ed esterne. A nostro parere, questo costringe a una conferma prematura del modello di dati relazionale. Negli ambienti di basi di dati moderni, la base di dati può essere implementata con una tecnologia orientata agli oggetti o con una combinazione di tecnologie orientate agli oggetti e relazionali. Rimandiamo quindi la trattazione dei concetti di normalizzazione al Capitolo 7.

Il modello entità-associazione è stato introdotto in un articolo fondamentale di Chen (1976), nel quale si descrivono i costrutti principali del modello E-R (da *entity-relationship*) e degli attributi associati. Il modello è stato successivamente esteso da Chen e altri fino a includere costrutti supplementari; si vedano per esempio Teorey, Yang e Fry (1986) e Storey (1991). Il modello E-R continua a evolversi, ma sfortunatamente non esiste ancora una notazione standard per la modellazione E-R. Song, Evans e Park (1995) presentano una comparazione di 10 notazioni di modellazione E-R, spiegando i vantaggi e gli svantaggi principali di ciascun approccio.

Molti sviluppatori di sistemi ritengono che la modellazione di dati sia la parte più importante dello sviluppo dei sistemi informativi. Questa convinzione si basa su tre ragioni principali (Hoffer, George e Valacich, 2002).

1. Le caratteristiche dei dati individuati durante la modellazione sono cruciali nella progettazione delle basi di dati, dei programmi e degli altri componenti del sistema. I fatti e le regole individuati durante il processo di modellazione sono essenziali nell'assicurare l'integrità dei dati in un sistema informativo.
2. I dati, più che i processi, sono l'aspetto più complesso di molti sistemi informativi moderni e ricoprono quindi un ruolo centrale nella strutturazione del sistema. Spesso l'obiettivo è quello di fornire una risorsa ricca di dati che possa supportare qualsiasi tipo di inchiesta, analisi o riepilogo.
3. I dati tendono a essere più stabili dei processi aziendali che li utilizzano. Un progetto di sistema informativo che si basa sull'orientamento ai dati dovrebbe quindi avere una vita utile più lunga di uno basato sull'orientamento ai processi.

In questo capitolo presenteremo le funzioni principali della modellazione E-R utilizzando le notazioni e le convenzioni più diffuse. Inizieremo definendo i costrutti di base del modello E-R: entità, attributi e associazioni. Definiremo tre tipi di entità che sono comuni nel modello E-R: entità forti, entità deboli ed entità associative. Definiremo anche diversi tipi di attributi importanti, compresi quelli

a valore singolo e multivalore, quelli derivati e quelli composti. Introduciamo quindi tre concetti legati alle associazioni: il grado di associazione, la cardinalità di un'associazione e i vincoli di partecipazione in un'associazione. Concluderemo con un esempio articolato di un diagramma E-R per Pine Valley Furniture Company.

5.1 Modellare le regole aziendali

In questo capitolo e nel prossimo vedremo come utilizzare i modelli di dati, e in particolare la notazione entità-associazione, per documentare le regole e le politiche di un'azienda. Le regole di documentazione e le politiche di un'azienda che governano i dati sono infatti l'oggetto della modellazione dei dati. Le regole e le politiche aziendali creano, aggiornano ed eliminano i dati in un sistema informativo di elaborazione e memorizzazione. Per esempio, una politica secondo la quale “ogni studente dell'università deve avere un tutor”, fa sì che i dati in una base di dati su ciascuno studente vengano associati ai dati di un tutor. Anche l'affermazione in base alla quale “uno studente è una persona qualsiasi che ha fatto richiesta di iscrizione, che ha frequentato un corso o un programma propedeutico che dà o non dà crediti in una qualsiasi unità universitaria” non solo definisce il concetto di “studente”, ma stabilisce anche una politica dell'università (per esempio, implicitamente, gli iscritti sono studenti, e uno studente delle superiori che ha partecipato a una fiera universitaria ma non si è iscritto non è uno studente, presumendo che una fiera universitaria sia un programma propedeutico che non dà crediti).

Le regole e le politiche aziendali non sono universali; diverse università possono avere politiche diverse per l'assistenza agli studenti e possono includere tipi diversi di persone come studenti. Inoltre le regole di un'azienda possono cambiare nel tempo, seppur lentamente; un'università potrebbe decidere che a uno studente non deve essere assegnato un tutor fino a quando non sceglie la propria specializzazione.

Il lavoro di un analista di base di dati consiste in tre compiti.

- Identificare e capire le regole che governano i dati.
- Rappresentare queste regole in modo tale che siano capite senza ambiguità dagli sviluppatori e dagli utenti dei sistemi informativi.
- Implementare queste regole in una tecnologia di base di dati.

La modellazione dei dati è uno strumento importante in questo processo. Poiché il suo scopo è quello di documentare le regole aziendali sui dati, introdurremo la sua trattazione e la notazione entità-associazione con una panoramica sulle regole aziendali. I modelli di dati non sono in grado di rappresentare tutte le regole aziendali (e non devono nemmeno farlo, poiché non tutte le regole aziendali governano i dati); i modelli di dati, con la documentazione associata e gli altri tipi di modelli di sistemi informativi (per esempio i modelli che documentano l'elaborazione dei dati) rappresentano tutte le regole aziendali che devono essere applicate attraverso i sistemi informativi.

5.1.1 Panoramica sulle regole aziendali

Una **regola aziendale** è “un'affermazione che definisce o vincola alcuni aspetti dell'azienda. Il suo scopo è quello di dichiarare la struttura aziendale o di controllare o influenzare il comportamento dell'azienda... le regole impediscono, provocano o suggeriscono gli eventi” (GUIDE Business Rules Project, 1997). Per esempio, le due affermazioni seguenti sono espressioni comuni di regole aziendali che influiscono sull'elaborazione e la memorizzazione dei dati.

- “Uno studente può iscriversi a un seminario di un corso solo se ha completato con successo quanto richiesto per quel corso”.
- “Un cliente privilegiato si candida per un sconto del 10%, a meno che non abbia insoluti”.

La maggior parte delle aziende moderne, e dei loro dipendenti, è guidata da migliaia di combinazioni di questo tipo di regole. Le regole influenzano il comportamento dell'aggregato e determinano come l'azienda risponde al suo ambiente (Gottesdiener, 1997). L'individuazione e la documentazione delle regole aziendali è un'operazione importante e complessa. Individuare e strutturare le regole e applicarle attraverso la tecnologia delle basi di dati aiuta a garantire che i sistemi informativi funzionino correttamente e che gli utenti delle informazioni capiscano cosa immettono e vedono.

Il paradigma delle regole aziendali

Il concetto di regole aziendali è stato utilizzato nei sistemi informativi per diverso tempo. Quando ci si riferisce a queste regole, è tuttavia diventato più comune, soprattutto nel mondo delle basi di dati, utilizzare il termine “vincolo di integrità”. In realtà questa distinzione è in qualche modo più limitata, poiché si riferisce solitamente al mantenimento di valori di dati e di associazioni validi nella base di dati.

Attualmente il termine “regole aziendali” ha un ambito più ampio, e include tutte le regole (come quelle di esempio elencate precedentemente) che hanno un impatto sulle basi di dati di un'azienda. In realtà l'approccio delle regole di dati è stato indicato da numerosi autori come nuovo paradigma per la specifica dei requisiti dei sistemi informativi (von Halle, 1997). Tale approccio si basa sulle seguenti premesse.

- Le regole aziendali sono un concetto centrale in un'impresa poiché sono un'espressione della politica aziendale e guidano un comportamento individuale o collettivo. Regole aziendali ben strutturate possono essere definite in un linguaggio naturale per gli utenti finali e in un modello di dati per gli sviluppatori di sistema.
- Le regole aziendali possono essere espresse in termini familiari agli utenti finali. Questi possono quindi definire e mantenere le proprie regole.
- Le regole aziendali sono altamente mantenibili. Sono memorizzate in un repository centrale e ogni regola viene espressa una sola volta e quindi condivisa nell'azienda.
- L'applicazione delle regole aziendali può essere automatizzata attraverso l'uso di un software che interpreta le regole e le applica utilizzando i meccanismi di integrità del sistema di gestione della base di dati (Moriarty, 2000).

Sebbene la maggior parte dei progressi sia stata ormai compiuta, l'industria non ha ancora realizzato tutti gli obiettivi. La ricerca e lo sviluppo di strumenti prosegue (specialmente per quanto riguarda l'automazione delle regole aziendali) e l'approccio delle regole aziendali è ancora ricco di potenzialità. La premessa con il beneficio probabilmente più alto è che “le regole aziendali sono altamente mantenibili”. La capacità di specificare e mantenere le esigenze di un sistema informativo come insieme di regole diventa molto forte se combinata con la capacità di generare automaticamente i sistemi informativi da un repository di regole. La generazione e manutenzione automatiche dei sistemi non semplificano solo il processo di sviluppo dei sistemi, ma ne migliorano anche la qualità.

5.1.2 Ambito delle regole aziendali

In questo capitolo e nel prossimo ci si concentrerà su quelle regole aziendali che hanno un impatto solo sulle basi di dati dell'azienda. La maggior parte delle aziende ha una serie di regole e/o politiche che non rientrano in questa definizione. Per esempio, la regola "Venerdì è il giorno in cui si può adottare un abbigliamento casual" può essere un'importante indicazione di politica, ma non ha un impatto immediato sulle basi di dati. D'altra parte, la regola "Uno studente può iscriversi a un seminario di un corso solo se ha completato con successo quanto richiesto per quel corso" rientra nel nostro ambito, poiché vincola le transazioni che possono essere elaborate nella base di dati. In particolare, fa sì che venga rifiutata qualsiasi transazione che cerchi di iscrivere uno studente che non ha i pre-requisiti necessari. Alcune regole aziendali non possono essere rappresentate con la notazione comune della modellazione dei dati; le regole che non possono essere illustrate in una variante del diagramma entità-associazione vengono definite con un linguaggio naturale, e alcune di esse possono essere rappresentate nel modello di dati relazionale, come vedremo nel Capitolo 7.

Regole aziendali ben definite

Che sia definita con un linguaggio naturale, in un modello di dati strutturato o in un altro tipo di documentazione dei sistemi informativi, una regola aziendale avrà determinate caratteristiche se dovrà essere coerente con le premesse precedentemente delineate. Queste caratteristiche, riepilogate nella Tabella 5.1, avranno una possibilità maggiore di essere soddisfatte se la regola aziendale è definita e se è di proprietà di persone, e non tecnici, dell'azienda. Le persone dell'azienda diventano gli accompagnatori delle regole aziendali, mentre gli analisti delle basi di dati facilitano l'emergere delle regole e la trasformazione di quelle mal definite in regole che soddisfino le caratteristiche desiderate.

Tabella 5.1 Caratteristiche di una buona regola aziendale.

Caratteristica	Spiegazione
Deve essere dichiarativa	Una regola aziendale è <u>l'affermazione di una politica e non di come questa viene applicata e condotta</u> ; la regola non descrive un processo o un'implementazione, ma descrive cosa viene convalidato da un processo.
Deve essere precisa	<u>Nell'azienda, la regola deve avere una sola interpretazione per tutte le persone coinvolte, e il suo significato deve essere chiaro.</u>
Deve essere atomica	Una regola aziendale <u>indica una sola istruzione</u> ; nessuna parte della regola può essere una regola di per sé (ossia, la regola è indivisibile e sufficiente).
Deve essere coerente	Una regola aziendale <u>deve essere internamente coerente</u> (ossia, <u>non contenere istruzioni contraddittorie</u>) e deve essere <u>coerente con altre regole</u> (che non deve contraddire).
Deve essere espressiva	Una regola aziendale <u>deve poter essere espressa in un linguaggio naturale</u> , ma verrà affermata in linguaggio strutturato, così da evitare fraintendimenti.
Deve essere distintiva	Le regole aziendali <u>non sono ridondanti</u> , ma una regola aziendale può fare riferimento ad altre regole (e in particolare alle definizioni).
Deve essere orientata all'azienda	Una regola aziendale <u>viene affermata in termini che possono essere compresi dal personale dell'azienda</u> , e poiché è espressione di una politica aziendale, solo il personale dell'azienda può modificarla o invalidarla; una regola aziendale è quindi di proprietà dell'azienda.

Adattamento di Gottesdiener (1999) e Plotkin (1999).

Raccogliere le regole aziendali

Le regole aziendali appaiono nelle descrizioni di funzioni, eventi, politiche, unità, partecipazioni e nei loro oggetti. Queste descrizioni si possono trarre da raccolte di sessioni di interviste singole o di gruppo sulle esigenze informative (per esempio, manuali del personale, polizze, contratti, dépliant del marketing e istruzioni tecniche) e da altre fonti. Le regole vengono identificate ponendo domande sui chi, cosa, quando, dove e perché dell'azienda. L'analista dei dati deve solitamente insistere molto per poter fare chiarezza nelle affermazioni delle regole, poiché spesso le asserzioni iniziali sono vaghe o imprecise. Le regole esatte vengono quindi formulate attraverso il processo iterativo dell'inchiesta. Si deve essere preparati a porre domande come "È sempre vero?", "Ci sono situazioni speciali in cui si verificano casi diversi?", "Ci sono generi diversi per quel tipo di ruolo?", "Ce n'è uno solo o ce ne sono molti?" oppure "Occorre tenere una storia di questi dati o sono sufficienti quelli correnti?". Riporteremo domande specifiche che possano essere utili per far emergere le regole per ciascun tipo di costrutto di modellazione dei dati man mano che presenteremo ciascun costrutto in questo capitolo e nei prossimi.

5.1.3 Nomi e definizioni degli "oggetti" di base

Fondamentali per la comprensione e la modellazione dei dati aziendali sono la denominazione e la definizione degli oggetti di base. Gli oggetti di base devono essere denominati e definiti prima di poter essere utilizzati senza ambiguità in un modello di dati dell'azienda. Nella notazione entità-associazione descritta in questo capitolo, è necessario assegnare a entità, associazioni e attributi nomi chiari e definizioni distintive.

Nomi degli "oggetti" di base

Approfondiremo la denominazione specifica di entità, relazioni e attributi man mano che svilupperemo il modello di dati entità-associazione, ma esistono comunque alcune linee guida sulla denominazione di tutti gli oggetti di base che, per Salin (1990), devono avere le caratteristiche descritte di seguito.

- Fare riferimento a caratteristiche aziendali (non tecniche, hardware e software): quindi, Cliente è un buon nome, mentre File10, Bit7 e ChiaveOrdinamentoReportStipendi non lo sono.
- Essere significativi, quasi fino al punto di auto-documentarsi (ossia, la definizione precisa spiega il nome senza dover descrivere l'essenza del significato del nome dell'oggetto); è preferibile evitare di utilizzare parole generiche come "ha", "è", "persona" o "esso".
- Essere univoci rispetto al nome utilizzato per tutti gli altri; nel nome di un dato dovrebbero essere incluse le parole che distinguono gli oggetti di base da altri oggetti di base simili (per esempio, IndirizzoCasa rispetto a IndirizzoUfficio).
- Essere leggibili, così che il nome sia strutturato come il concetto verrebbe pronunciato naturalmente (per esempio, MediaVoti è un buon nome, mentre MediaVotiRelativaClasseA, per quanto preciso, è un nome poco elegante).
- Essere composti da parole tratte da una lista di parole approvata: ogni azienda sceglie spesso un vocabolario da cui prendere le parole significative da adottare nella denominazione dei dati (per esempio, si preferisce "massimo" a "limite superiore", "tetto" o "più alto"). Nel set della documentazione della base di dati possono essere incluse anche alternative o alias; le parole nel dizionario possono anche avere abbreviazioni approvate (come CLI per Cliente) e si può essere incoraggiati a utilizzare le abbreviazioni, così che i nomi dei dati siano sufficientemente corti da assecondare i limiti di lunghezza della tecnologia della base di dati.

- Essere ripetibili: significa che persone diverse o la stessa persona in momenti diversi possono sviluppare esattamente lo stesso nome, o quasi lo stesso nome; questo significa spesso che esiste una gerarchia standard o un pattern per i nomi (per esempio, la data di nascita di uno studente potrebbe essere DataNascitaStudente, mentre quella di un impiegato potrebbe essere DataNascita).

Salin (1990) suggerisce di sviluppare i nomi dei dati seguendo alcune linee guida.

1. Preparare una definizione dei dati (argomento trattato successivamente).
2. Eliminare parole senza significato o non valide (parole non presenti nell'elenco approvato). Si noti che la presenza di AND e OR nella definizione può implicare che due o più oggetti di base sono combinati; si potrebbero voler separare gli oggetti e assegnare nomi diversi.
3. Disporre le parole in modo significativo e ripetibile.
4. Assegnare un'abbreviazione standard a ogni parola.
5. Determinare se il nome è già esistente e, in questo caso, aggiungere altri qualificatori che rendano il nome univoco.

Nel capitolo, vedremo alcuni esempi di nomi di dati validi man mano che svilupperemo la notazione di modellazione dei dati.

Definizione degli oggetti di base (del modello di dati aziendale)

Una definizione (chiamata a volte asserzione strutturale) è considerata un tipo di regola aziendale (GUIDE Business Rules Project, 1997). Una definizione è una spiegazione di un termine o fatto. Un termine è una parola o una frase che ha un significato specifico per l'azienda. Esempi di termini sono corso, seminario, auto a noleggio, volo, prenotazione e passaggero. I termini sono spesso le parole chiave utilizzate per formare i nomi di dati. I termini devono essere definiti con attenzione e in modo conciso. Non è tuttavia necessario definire termini comuni come giorno, mese, persona o televisione, poiché vengono capiti da tutti senza ambiguità.

Un fatto è un'associazione tra due o più termini. Esempi di fatti che sono definizioni sono i seguenti (i termini definiti appaiono sottolineati).

- “Un corso è un modulo didattico in una particolare materia”. Questa definizione associa due termini: modulo didattico e materia. Si presume che questi due termini non debbano essere ulteriormente definiti.
- “Un cliente può richiedere un modello di automobile a un'agenzia di noleggio in una particolare data”. Questo fatto, che è una definizione di richiesta di noleggio di automobile associa i quattro termini sottolineati (GUIDE Business Rules Project, 1997). Tre di questi termini sono termini specifici per l'azienda che dovrebbero essere ulteriormente definiti singolarmente (“data” è un termine comune).

Definizioni di dati valide

Illustreremo definizioni valide per entità, associazioni e attributi man mano che svilupperemo la notazione entità-associazione in questo capitolo e nei prossimi. Ci sono tuttavia alcune linee guida da seguire (Aranow, 1989).

- Le definizioni (e tutti i tipi di regole aziendali) vengono raccolte dalle stesse fonti, così come tutte le esigenze dei sistemi informativi. In questo modo, gli analisti di sistema e dei dati dovrebbero cercare gli oggetti di base e le loro definizioni studiando le fonti e i sistemi informativi.

■ Le definizioni sono solitamente accompagnate da diagrammi, come quelli entità-associazione. La definizione non deve ripetere necessariamente quanto è presente nel diagramma, ma piuttosto integrarlo. Una definizione definirà queste caratteristiche degli oggetti di base:

- sottotitoli;
- condizioni speciali o eccezionali;
- esempi;
- dove, quando e come i dati vengono creati o calcolati nell'azienda;
- se i dati sono statici o cambiano nel tempo;
- se i dati sono singolari o plurali nella loro forma atomica;
- chi determina i valori dei dati;
- chi possiede i dati (ossia, chi ne controlla la definizione e l'uso);
- se i dati sono facoltativi o se sono consentiti dati vuoti (nulli);
- se i dati possono essere scomposti in parti ancora più atomiche o se sono spesso combinati con altri dati in una forma più composta o aggregata.

Se queste caratteristiche non sono incluse nella definizione dei dati, allora devono essere documentate in qualche altro punto, dove vengono memorizzati gli altri metadati.

■ Un oggetto di base non dovrebbe essere aggiunto al modello di dati, come una diagramma entità-relazione, fino a quando non è stato attentamente definito e denominato e fino a che non è stato raggiunto un accordo su questa definizione. Ci si può tuttavia aspettare che la definizione cambi una volta che l'oggetto è stato inserito nel diagramma, poiché il processo di sviluppo del modello di dati testa la comprensione del significato dei dati (in altre parole, la modellazione dei dati è un processo iterativo).

Nella modellazione degli oggetti di base del modello dei dati aziendali c'è una frase che evidenzia l'importanza di definizioni di dati valide. Questa frase è "Chi controlla il significato dei dati, controlla i dati". Può sembrare che ottenere un accordo in un'azienda sulle definizioni da utilizzare per i vari termini e fatti sia relativamente semplice. In realtà non lo è. Probabilmente è una delle sfide più difficili da affrontare nella modellazione dei dati (e anche in altri contesti). Non è insolito per un'azienda avere definizioni multiple (anche più di una decina) per termini comuni come "cliente" o "ordine".

Per illustrare i problemi inerenti lo sviluppo delle definizioni, si consideri un oggetto di base *Stu-dente* che si può trovare in una qualsiasi università. Una definizione tipica di *Stu-dente* è "una persona che si è iscritta almeno a un corso nell'ultimo anno". Questa definizione verrà sicuramente messa in discussione, poiché è troppa ristretta. Una persona che è studente passa probabilmente attraverso varie fasi del suo rapporto con la scuola, come quelle elencate di seguito.

1. Prospettiva: ha alcuni contatti formali che indicano un interesse per la scuola.
2. Richiesta: richiede l'ammissione.
3. Richiesta accettata: viene ammesso alla scuola e a un piano di studi.
4. *Stu-dente* immatricolato: si iscrive almeno a un corso.
5. *Stu-dente* costante: si iscrive a più corsi su una base continuativa (senza interruzioni significative).

6. **Studente passato:** non si iscrive ai corsi durante il periodo stabilito (può ripresentare la richiesta di iscrizione).
7. **Laureato:** completa in modo soddisfacente il piano di studi (può iscriversi nuovamente per seguire un altro piano di studi).

Si immagini la difficoltà di ottenere un consenso su ognuna di queste singole definizioni! Le alternative possibili sono tre.

1. Utilizzare definizioni multiple per coprire tutte le situazioni. Questo può creare molta confusione qualora ci sia solo un tipo di entità, e non è consigliato (le definizioni multiple non sono buone definizioni). Potrebbe essere possibile creare tipi di entità multipli, uno per ciascuna situazione in cui si trova lo studente. Poiché è probabile che ci sia una notevole somiglianza tra i tipi di entità, le distinzioni minime tra i tipi di entità può creare confusione e il modello di dati potrebbe mostrare molti costrutti.
2. Utilizzare una definizione molto generica che copra tutte le situazioni. Questo approccio può richiedere l'aggiunta di alcuni dati supplementari sugli studenti per registrare lo stato corrente di uno studente. Per esempio, i dati per lo stato di uno studente, con i valori sulla prospettiva, la richiesta e così via possono essere sufficienti. D'altra parte, se lo stesso studente può avere più stati (per esempio, la prospettiva per un corso di laurea e l'immatricolazione per un altro), questo metodo può non funzionare.
3. Si prenda in considerazione l'ipotesi di utilizzare oggetti di base multipli, ma correlati, per Studente. Per esempio, si può creare un tipo di entità generale per Studente e poi altri tipi di entità per generi di studenti con caratteristiche univoche. Descriveremo le condizioni che suggeriscono questo approccio nel Capitolo 6.

5.2 Il modello E-R

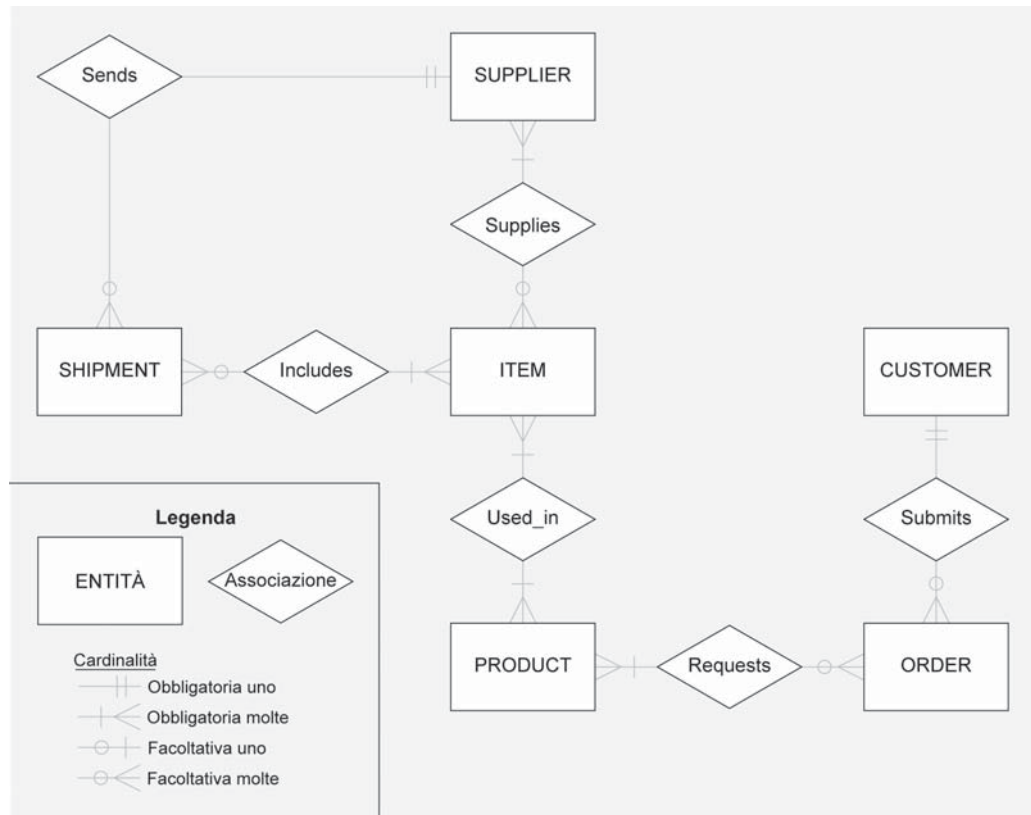
Il modello entità-associazione (o modello E-R) è una rappresentazione logica e dettagliata degli oggetti e delle dinamiche (dei dati) di un'azienda o di un'area aziendale. Il modello E-R viene espresso in termini di entità nell'ambiente aziendale, di associazioni tra le entità e di attributi (o proprietà) sia delle entità sia delle loro associazioni. Un modello E-R viene espresso solitamente come un **diagramma entità-associazione** (o **diagramma E-R**), che è una rappresentazione grafica di un modello E-R.

5.2.1 Esempio di diagramma E-R

Per iniziare da subito a capire i diagrammi E-R, si osservi la Figura 5.1, che presenta un diagramma E-R semplificato per una piccola società di mobili in legno chiamata Pine Valley Furniture Company. La società ha acquistato articoli da diversi fornitori, che li hanno quindi consegnati al produttore. Gli articoli vengono assemblati in prodotti che vengono venduti ai clienti che li ordinano. Ogni ordine può includere una o più linee che corrispondono ai prodotti che appaiono nell'ordine.

Il diagramma nella Figura 5.1 mostra le entità e le associazioni di questa società (per semplicità, per ora vengono omessi gli attributi). Le entità sono rappresentate dai rettangoli, mentre le associazioni tra di esse sono rappresentate dai rombi connessi da linee alle entità correlate. Le entità mostrate nella Figura 5.1 sono le seguenti.

Figura 5.1
Esempio di diagramma
E-R.



- **CUSTOMER:** una persona o azienda che ha ordinato o potrebbe ordinare prodotti. *Esempio:* L. L. Fish Furniture.
- **PRODUCT:** uno tipo di fornitura realizzato da Pine Valley che può essere ordinata dai clienti. Si noti che un prodotto non è una libreria specifica, fino a quando le singole librerie non devono essere individuate. *Esempio:* una libreria di quercia alta 1,8 m a 6 scaffali chiamata O600.
- **ORDER:** la transazione associata alla vendita di uno o più prodotti a un cliente e identificata da un numero di transazione fornito dalle vendite o dalla contabilità. *Esempio:* l'evento di L. L. Fish che acquista un prodotto O600 e quattro prodotti O623 il 10 settembre 2001.
- **ITEM:** un tipo di componente che entra nella fabbricazione di uno o più prodotti e che può essere fornito da più fornitori. *Esempio:* rotelle con un diametro di 10,15 cm.
- **SUPPLIER:** un'altra azienda che può fornire articoli a Pine Valley Furniture. *Esempio:* Sure Fasteners, Inc.
- **SHIPMENT:** la transazione associata agli articoli ricevuti in un unico collo da Pine Valley Forniture da un fornitore. Tutti gli articoli in una consegna appaiono su un unico documento di trasporto. *Esempio:* la ricezione degli articoli 300 I-27-4375 e 200 I-27-4380 da Sure Fasteners, Inc il 9 settembre 2001.

Si noti che è importante definire con chiarezza ciascuna entità, così come per i metadati. Per esempio, è importante sapere che l'entità CUSTOMER include persone o aziende che non hanno ancora acquistato prodotti da Pine Valley Furniture Company. È comune che le varie funzioni dell'azienda assegnino un significato diverso allo stesso termine (omonimia). Per esempio, l'ufficio contabilità può indicare come clienti solo quelle persone o aziende che hanno effettuato un acquisto, mentre l'ufficio marketing indica come clienti chiunque abbia contattato la società o abbia effettuato un acquisto o chiunque sia un concorrente noto. Un diagramma E-R articolato e preciso senza metadati chiari può essere interpretato in modi diversi da persone diverse.

I simboli alle estremità di ciascuna linea del diagramma E-R rappresentano le cardinalità delle associazioni. Esaminando la Figura 5.1 si può rilevare che tali simboli esprimono le seguenti regole aziendali.

1. Un SUPPLIER può fornire più ITEM (per “può fornire” si intende che il fornitore potrebbe anche non fornire alcuni elementi). Ogni ITEM viene fornito a qualsiasi numero di SUPPLIER (per “viene fornito” si intende che deve essere fornito almeno da un produttore).
2. Ogni ITEM deve essere utilizzato nell'assemblaggio di almeno un PRODUCT e può essere utilizzato in molti prodotti. Per contro, ogni PRODUCT deve utilizzare uno o più ITEM.
3. Un SUPPLIER può effettuare molte SHIPMENT. D'altra parte, ogni consegna deve essere effettuata esattamente da un SUPPLIER. Si noti che “effettuare una consegna” e “fornire” sono due concetti separati. Un SUPPLIER può essere in grado di fornire un articolo ma potrebbe non aver ancora effettuato alcuna consegna per quell'articolo.
4. Una SHIPMENT deve includere uno o più ITEM. Un ITEM può essere incluso in più SHIPMENT.
5. Un CUSTOMER può effettuare un numero qualsiasi di ORDER. Tuttavia ciascun ORDER deve essere effettuato esattamente da un CUSTOMER.
6. Un ORDER deve richiedere uno o più PRODUCT. Un PRODUCT dato potrebbe non essere richiesto su nessun ORDER o su uno o più ORDER.

Si noti che ciascuna di queste regole aziendali segue approssimativamente una sintassi:

<entità><cardinalità minima><associazione><cardinalità massima><entità>

Per esempio, la regola 5 è:

<CUSTOMER><può><effettuare><qualsiasi numero di><ORDER>

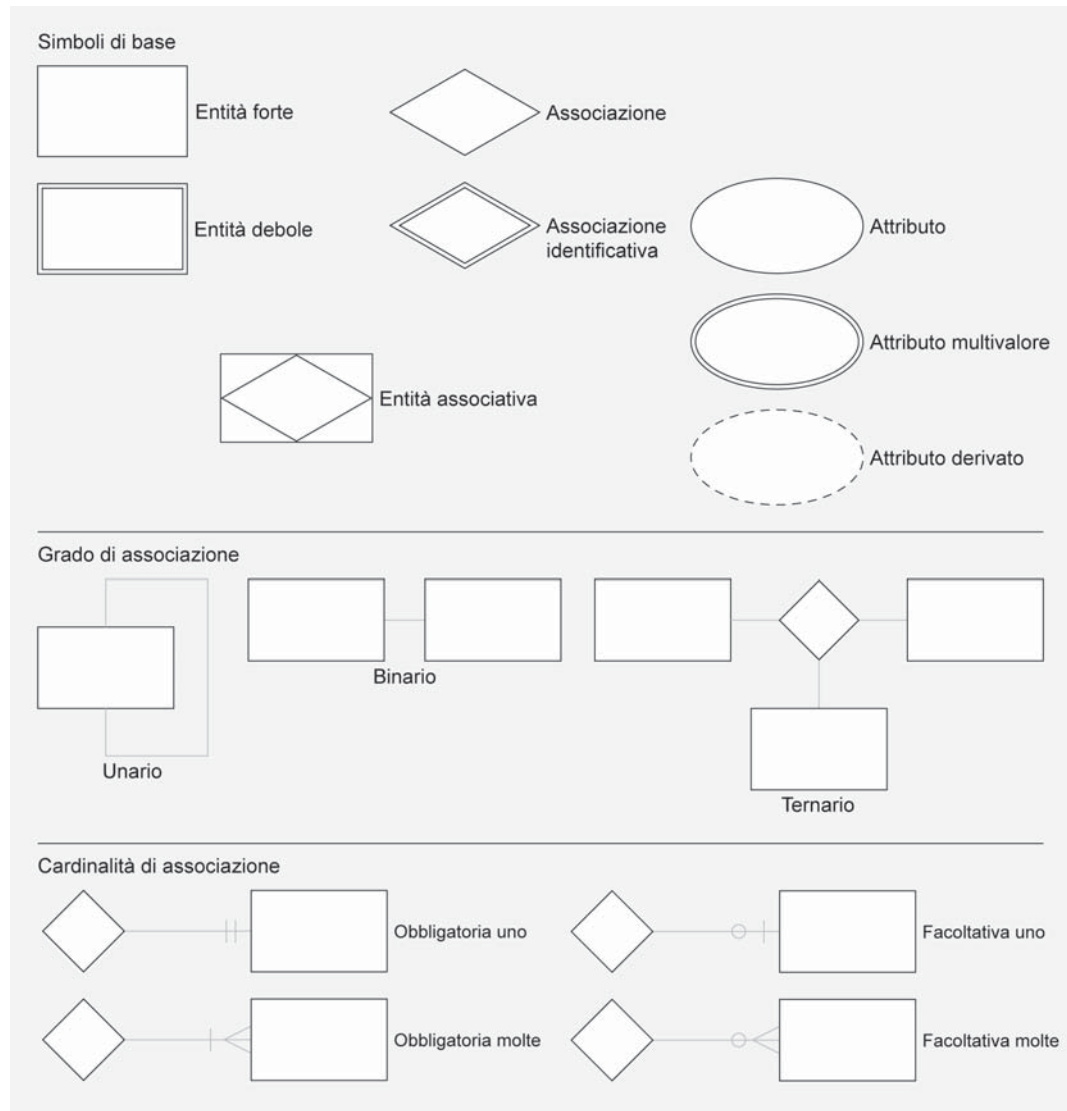
Questa sintassi offre un modo standard per tradurre ciascuna associazione in un'istruzione di regola aziendale in un linguaggio naturale.

5.2.2 Notazione del modello E-R

La notazione di base utilizzata per i diagrammi E-R è mostrata nella Figura 5.2. Come indicato nel paragrafo precedente, non esiste una notazione industriale standard. Tuttavia, la notazione mostrata nella figura combina la maggior parte delle caratteristiche auspicabili delle diverse notazioni che sono solitamente utilizzate e permette anche di modellare con precisione la maggior parte delle situazioni che si incontrano nella pratica. Presenteremo la notazione supplementare per i modelli entità-associazione avanzati (comprese le relazioni classe-sottoclasse) nel Capitolo 6. L'Appendice A illustra la notazione E-R tratta da diverse linee guida e strumenti di progettazione comunemente utilizzati. Questa appendice può aiutare a convertire le notazioni nel testo con quelle utilizzate nel concreto.

Figura 5.2

Notazione E-R di base.



5.3 Costrutti del modello entità-associazione

I costrutti di base del modello entità-associazione sono le entità, le associazioni e gli attributi. Come mostrato nella Figura 5.2, il modello consente numerose varianti a ciascuno di questi costrutti. La ricchezza del modello E-R permette ai progettisti di modellare le situazioni del mondo reale in modo preciso ed esauriente, cosa che favorisce la diffusione del modello.

5.3.1 Entità

Un'entità è una persona, luogo, oggetto, evento o concetto nell'ambiente dell'utente riguardo al quale l'azienda desidera mantenere dei dati. Alcuni esempi di ciascun tipo di entità sono elencati di seguito.

- Persona: EMPLOYEE, STUDENT, PATIENT
- Luogo: STORE, WAREHOUSE, STATE
- Oggetto: MACHINE, BUILDING, AUTOMOBILE
- Evento: SALE, REGISTRATION, RENEWAL
- Concetto: ACCOUNT, COURSE, WORK CENTER

Tipo di entità contro istanza di entità

Esiste una distinzione importante tra tipi di entità e istanze di entità. Un tipo di entità è una raccolta di entità che condividono proprietà o caratteristiche comuni. A ogni tipo di entità in un modello E-R viene assegnato un nome. Poiché il nome rappresenta una raccolta (o insieme) di voci, è sempre al singolare. Per i nomi di tipi di entità si utilizzano le lettere maiuscole. In un diagramma E-R l'entità viene collocata nel rettangolo che rappresenta il tipo di entità (Figura 5.1).

Un'istanza di entità è una singola occorrenza di un tipo di entità. La Figura 5.3 illustra la distinzione tra un tipo di entità e due delle sue istanze. Un'entità viene descritta solo una volta (utilizzando i metadati) in una base di dati, mentre più istanze di quel tipo di entità possono essere rappresentate dai dati memorizzati nella base di dati. Per esempio, nella maggior parte delle aziende c'è un tipo di entità EMPLOYEE, ma ci sono centinaia (a volte migliaia) di istanze di questo tipo di entità memorizzate nella base di dati. Spesso si utilizza il termine singolo "entità" invece che la sua istanza, ma solo quando il significato che si evince dal contesto è inequivocabile.

Figura 5.3
Tipo di entità (EMPLOYEE)
con due istanze.

Tipo di entità: EMPLOYEE	
Attributi:	
EMPLOYEE NUMBER	CHAR (10)
NAME	CHAR (25)
ADDRESS	CHAR (30)
CITY	CHAR (20)
STATE	CHAR (2)
ZIP	CHAR (9)
DATE HIRED	DATE
BIRTHDATE	DATE
Due istanze di EMPLOYEE:	
642-17-8360	534-10-1971
Michelle Brady	David Johnson
100 Pacific Avenue	450 Redwood Drive
San Francisco	Redwood City
CA	CA
98173	97142
03-21-1992	08-16-1994
06-19-1968	09-04-1975

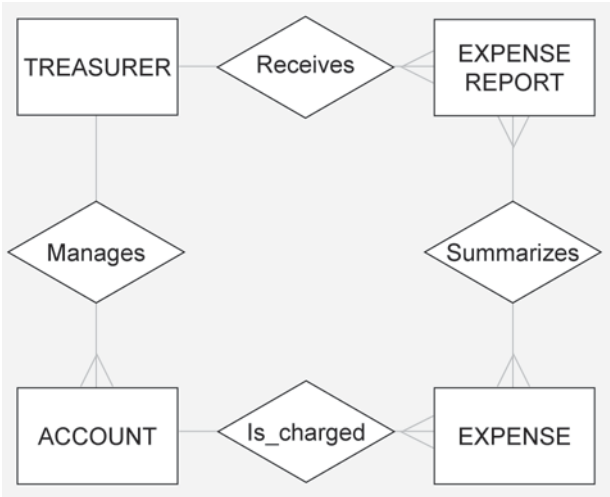
Tipo di entità contro input, output o utente di sistema

Un errore comune che si compie quando si imparano i diagrammi E-R, soprattutto se si ha già familiarità con la modellazione dei processi dei dati (come la creazione di diagrammi di flusso), è quello di confondere le entità di dati con altri elementi di un modello complessivo dei sistemi informativi. Una regola semplice per evitare tale confusione è quella di ricordare che un'entità di dati autentica avrà più istanze possibili, ognuna con una caratteristica specifica, così come una o più altre parti di dati descrittive.

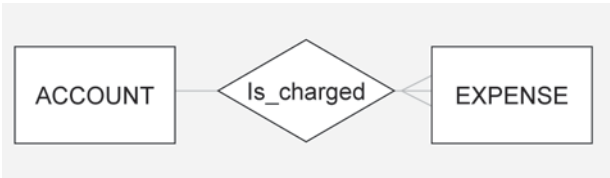
La Figura 5.4a potrebbe essere disegnata per rappresentare una base di dati necessaria per il sistema di spese di un club di studentesse di un'università. In questa situazione, il tesoriere dell'associazione gestisce conti, riceve report delle spese e registra le transazioni delle spese confrontandole con ciascun conto. Quello che ci occorre, tuttavia, è tenere una traccia dei dati sul tesoriere (il tipo di entità TREASURER), della sua supervisione dei conti (l'associazione Manages) e della ricezione dei report (l'associazione Receives). Il tesoriere è la persona che immette i dati sui conti e le spese e che riceve i report sulle spese. Detto più chiaramente, è l'utente della base di dati. Poiché c'è un solo tesoriere, i dati di TREASURER non devono essere considerati. L'entità EXPENSE REPORT è necessaria? Poiché un report di spese viene calcolato dalle transazioni di spesa e dai bilanci, è il risultato dell'estrazione dei dati dalla base di dati. Sebbene ci saranno più istanze dei report di spesa dati al tesoriere nel tempo, i dati necessari per calcolare ogni volta il contenuto dei report sono già rappresentati nei tipi di entità ACCOUNT ed EXPENSE.

Un'altra chiave per capire perché il diagramma nella Figura 5.4a possa essere errata è la natura dei nomi delle associazioni Receives e Summarizes. Questi nomi fanno riferimento alle attività aziendali che trasferiscono o traducono i dati e non semplicemente all'associazione di un tipo di dati con un altro. Il modello E-R semplice nella Figura 5.4b mostra entità e una relazione che sarebbero sufficienti per gestire il sistema di spese del club come qui descritto.

Figura 5.4
Esempio di entità non appropriate.
(a) Utente del sistema (TREASURER) e output (EXPENSE REPORT) mostrati come entità.



(b) Modello E-R con le sole entità necessarie.



Tipi di entità forti contro tipi di entità deboli

La maggior parte dei tipi di entità che identificano un'azienda è classificata come forte. Un tipo di entità forte è un tipo che esiste indipendentemente dagli altri tipi di entità. Alcuni esempi sono STUDENT, EMPLOYEE, AUTOMOBILE e COURSE. Anche le istanze dei tipi di entità forti hanno una caratteristica univoca (chiamata *identificatore*), ossia un attributo o una combinazione di attributi che distinguono ciascuna occorrenza dell'entità.

Per contro, un tipo di entità debole è un'entità la cui esistenza dipende da altri tipi di entità. Un tipo di entità debole non ha un significato aziendale nel diagramma E-R senza l'entità da cui dipende. Il tipo di entità da cui dipende l'entità debole è chiamato **possessore identificativo** (o semplicemente **possessore**). Un tipo di entità debole non ha il proprio identificatore. In genere, in un diagramma E-R un tipo di entità debole ha un attributo che serve come identificatore *parziale*. Nella fase di progettazione successiva (descritta nel Capitolo 7), per l'entità debole verrà creato un identificatore completo combinando l'identificatore parziale con quello del suo possessore.

Un esempio di tipo di entità debole con una relazione identificativa è mostrata nella Figura 5.5. EMPLOYEE è un tipo di entità forte con un identificatore Employee_ID (indicheremo l'attributo dell'indicatore con la sottolineatura). DEPENDENT è un tipo di entità debole, come segnalato dal rettangolo con il contorno doppio. L'associazione tra un tipo di entità debole e il suo possessore è chiamata **associazione identificativa**. Nella Figura 5.5 "Has" è l'associazione identificativa (indicata dal rombo con il contorno doppio). L'attributo Dependent_Name serve come identificatore *parziale* (Dependent_Name è un attributo composto che può essere scomposto in parti significative, come vedremo successivamente); la doppia sottolineatura indica un identificatore parziale. Durante una fase di progettazione successiva, Dependent_Name verrà combinato con Employee_ID (l'identificatore del possessore) per formare un identificatore completo per DEPENDENT.

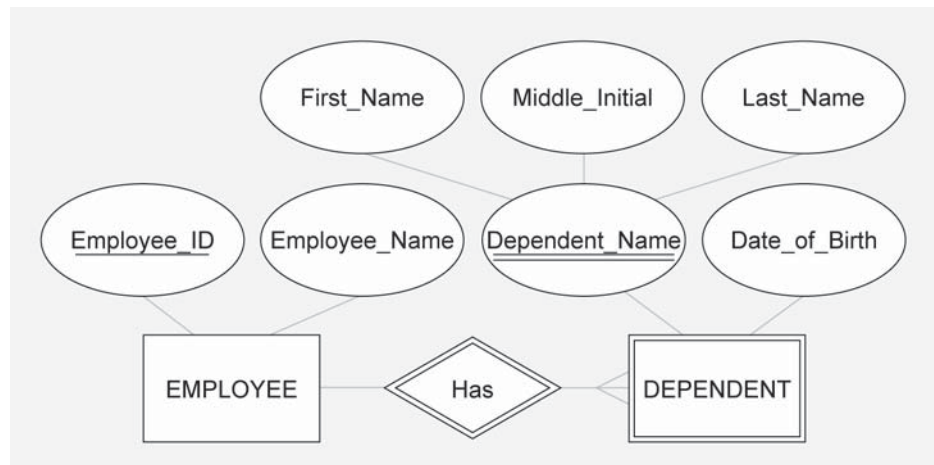
Denominare e definire i tipi di entità

Oltre alle linee guida generali per la denominazione e la definizione dei dati di oggetto, ci sono alcune linee guida per la denominazione dei tipi di entità.

- Un nome di tipo di entità è un *nome singolare* (come CUSTOMER, STUDENT o AUTOMOBILE); un'entità è una persona, luogo, oggetto, evento o concetto e il nome per il tipo di entità, che rappresenta un *insieme* di istanze di entità (per esempio, STUDENT rappresenta gli studenti Hank Finley,

Figura 5.5

Esempio di un'entità debole e delle sue associazioni identificative.



Jean Krebs e così via). È comune specificare anche la forma plurale (verosimilmente in un repository degli strumenti CASE che accompagna il diagramma E-R), poiché alcune volte il diagramma E-R viene letto meglio utilizzando i plurali. Per esempio, nella Figura 5.1 si può affermare che un SUPPLIER fornisce più ITEM. (Poiché in inglese i plurali non vengono formati sempre aggiungendo una “s” al nome singolare, è consigliabile documentare la forma plurale esatta).

- Un nome di tipo di entità deve essere specifico per l'azienda. Un'azienda potrebbe utilizzare il nome di tipo di entità CUSTOMER, mentre un'altra potrebbe utilizzare il termine CLIENT. Il nome dovrebbe essere descrittivo per l'azienda e distinto da tutti gli altri nomi di tipi di entità al suo interno. Per esempio, un PURCHASE ORDER per ordini effettuati presso fornitori è diverso da un CUSTOMER ORDER per ordini effettuati dai propri clienti. Nessuno di questi due tipi di entità può essere denominato ORDER.
- Un nome di tipo di entità deve essere conciso, e utilizzare il minor numero di parole possibili. Per esempio, nella base di dati di un'università, un tipo di entità REGISTRATION per l'evento di iscrizione di uno studente a un corso è probabilmente sufficiente un nome per questo tipo di entità; STUDENT REGISTRATION FOR CLASS, sebbene preciso, è probabilmente troppo lungo, considerato che l'utente può distinguere REGISTRATION dagli altri tipi di entità.
- Un'abbreviazione deve essere specificata per ciascun nome di tipo di entità, e l'abbreviazione può essere sufficiente per essere utilizzata nel diagramma E-R; tutte le abbreviazioni devono seguire le stesse regole dei nomi di entità completi.
- I tipi di entità di evento devono essere denominati per il risultato dell'evento, e non per l'attività o processo dell'evento. Per esempio, l'evento di un project manager che assegna un lavoro a un impiegato nei risultati del progetto è un ASSIGNMENT, mentre l'evento di uno studente che contatta il proprio tutor per avere alcune informazioni è un CONTACT.
- Il nome utilizzato per lo stesso tipo di entità dovrebbe essere lo stesso in tutti i diagrammi E-R nei quali appare quel tipo di entità. Oltre a essere specifico per l'azienda, il nome utilizzato per il tipo di entità dovrebbe quindi essere uno standard, adottato dall'azienda per tutti i riferimenti allo stesso tipo di dati. Alcuni tipi di entità avranno tuttavia degli *alias*, o nomi alternativi, che sono sinonimi utilizzati in parti diverse dell'azienda. Per esempio, il tipo di entità ITEM può avere alcuni alias per MATERIAL (per la produzione) e DRAWING (per l'ingegneria). Gli alias sono specificati nella documentazione della base di dati, come il repository di uno strumento CASE.

Esistono infine alcune linee guida specifiche per la definizione dei tipi di entità.

- Una definizione di tipo di entità inizia solitamente con “Una X è ...”. È il modo più chiaro e diretto per definire il significato di un'entità di dati.
- Una definizione di tipo di entità deve includere la definizione della caratteristica univoca per ciascuna istanza del tipo di entità. In molti casi, definire l'identificatore per un tipo di entità aiuta a comunicare il significato dell'entità. Per esempio, nella Figura 5.4b, “Una spesa è un pagamento per l'acquisto di beni o servizi. Una spesa identificata ha un numero di immissione progressivo”.
- Una definizione di tipo di entità include spesso una descrizione di quando un'istanza del tipo di entità viene creata ed eliminata. Per esempio, al punto precedente di questo elenco, un'istanza di cliente viene creata implicitamente quando la persona o l'azienda effettua il suo primo ordine; poiché questa definizione non specifica altrimenti, implicitamente un'istanza di cliente non viene mai eliminata o viene eliminata in base a regole generali specificate sulla rimozione dei dati dalla base di dati. Un'istruzione su quando eliminare un'istanza di entità viene a volte chiamata conservazione

del tipo di entità. Un'istruzione di eliminazione possibile di una definizione di tipo di entità può essere "Un cliente cessa di essere tale se non ha effettuato alcun ordine per più di tre anni".

- Per alcuni tipi di entità, la definizione deve specificare *quando un'istanza dovrebbe cambiare in un'istanza di un altro tipo di entità*. Per esempio, si consideri la situazione di una ditta di costruzioni per cui le offerte accettate da clienti potenziali diventano contratti. In questo caso, un'offerta può essere definita come "un'offerta legale della nostra azienda a effettuare del lavoro per un cliente. Un'offerta si crea quando un responsabile della nostra azienda firma il documento di offerta; un'offerta diventa istanza di contratto quando riceviamo una copia dell'offerta firmata da un superiore del cliente". Questa definizione è anche un buon esempio di come una definizione può utilizzare altri nomi di tipi di entità (in questo caso, la definizione di offerta utilizza il nome di tipo di entità Customer).
- Per alcuni tipi di entità, la definizione deve specificare *quale storia deve essere mantenuta rispetto alle istanze del tipo di entità*. Per esempio, le caratteristiche di un ITEM nella Figura 5.1 possono cambiare nel tempo, e potrebbe essere necessario tenere una storia completa dei singoli valori e di quando sono in vigore. Come vedremo in alcuni esempi successivi, tali istruzioni sul mantenimento della storia possono avere ramificazioni nel modo in cui si rappresenta il tipo di entità in un diagramma E-R e di come si memorizzano i dati per le istanze di entità.

5.3.2 Attributi

Ogni tipo di entità è associata a un insieme di attributi. Un **attributo** è una proprietà o caratteristica del sistema di un tipo di entità che è di interesse per l'azienda. Quelli che seguono sono alcuni tipi di entità e degli attributi associati.

STUDENT	Student_ID, Student_Name, Home_Address, Phone_Number, Major
AUTOMOBILE	Vehicle_ID, Color, Weight, Horsepower
EMPLOYEE	Employee_ID, Employee_Name, Payroll_Address, Skill

Nella denominazione degli attributi si utilizza un'iniziale maiuscola seguita da lettere minuscole. Se il nome di un attributo è costituito da due parole, si utilizza il carattere di underscore (_) per connettere le parole e si inizia ogni parola con una lettera maiuscola, come in Employee_Name. Nei diagrammi E-R, possiamo rappresentare un attributo collocando il suo nome in un'ellisse con una linea che lo connette all'entità associata. Gli attributi possono anche essere collegati ad associazioni, come descritto di seguito. Si noti che un attributo è associato esattamente a un'entità o associazione.

Nella Figura 5.5 si può notare che tutti gli attributi di DEPENDENT sono caratteristiche di un solo DEPENDENT di EMPLOYEE, e non caratteristiche di un impiegato. Un tipo di entità, sia essa debole o meno, include attributi di entità a cui è correlata (quelli che si potrebbero chiamare *attributi esterni*). Per esempio, DEPENDENT non include alcun attributo che indica a quale impiegato è associato questo dipendente. La funzione non ridondante del modello E-R è coerente con la proprietà di dati condivisa delle basi di dati. A causa delle associazioni, che discuteremo brevemente, chi accede ai dati da una base di dati sarà in grado di associare gli attributi dalle entità correlate (per esempio, visualizzare sullo schermo un Dependent_Name e l'Employee_Name associato).

Ciascuna entità (o istanza di un tipo di entità) ha un valore associato a ciascuno degli attributi di quel tipo di entità. Per esempio, la Figura 5.6 mostra due entità con i rispettivi valori di attributo. Una base di dati è fondamentalmente una raccolta di tutti questi valori di attributo per tutte le entità. Di conseguenza, ogni entità ha un attributo identificativo oltre a uno o più attributi. Se si cerca di creare un'entità che ha solo un identificatore, quell'entità non sarà probabilmente valida. Tale struttura di dati può semplice-

Figura 3.6

Due istanze di entità con valori di attributo.

Entità 1
Student_ID = 638124289
Student_Name = Allison Smith
Home_Address = 482 Walnut St. Denver CO 86391
Phone = 303-839-1479
Major = Psychology
Entità 2
Student_ID = 726153972
Student_Name = Cedric Thomas
Home_Address = 944 Maple St. Houston, TX 51269
Phone = 631-347-9847
Major = English

mente contenere un elenco di valori validi per alcuni attributi, che è preferibile tenere all'esterno della base di dati.

Attributi semplici contro attributi composti

Alcuni attributi possono essere scomposti in parti significative. L'esempio più comune è Address, che può solitamente essere scomposto nei componenti Street_Address, City, State e Postal_Code. **Un attributo composto** è un attributo (come Address) che può essere scomposto in parti componenti. La Figura 5.7 mostra la notazione utilizzata per gli attributi applicati a questo esempio. In un diagramma E-R gli attributi di componente possono apparire sopra o sotto l'attributo composto.

Gli attributi composti forniscono una notevole flessibilità agli utenti, che possono fare riferimento all'attributo composto come a un'unità singola o che possono fare riferimento ai singoli componenti di quell'attributo. Per esempio, un utente può fare riferimento ad Address o a uno dei suoi componenti, come Street_Address. La decisione di suddividere o meno un attributo nei suoi componenti dipende dal fatto che all'utente occorra fare riferimento a questi singoli componenti. Il progettista deve naturalmente cercare di prevedere i futuri pattern d'uso della base di dati.

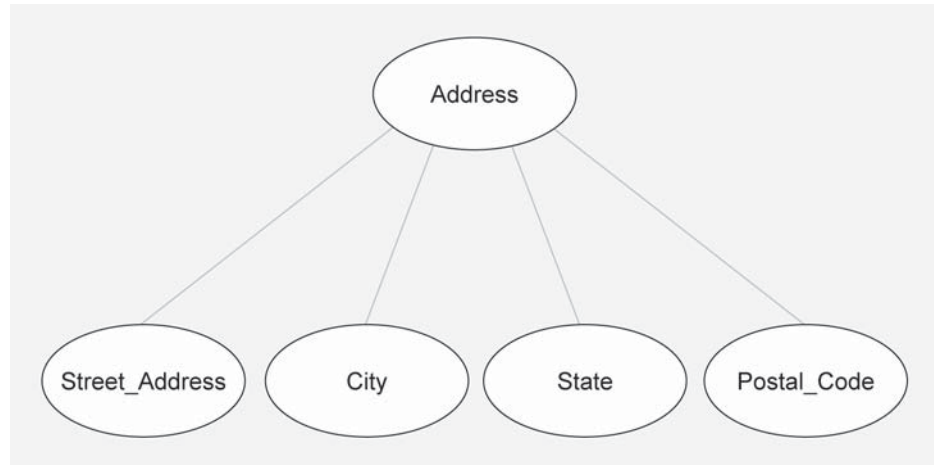
Un attributo semplice (o atomico) è un attributo che non può essere scomposto in componenti più piccoli. Per esempio, tutti gli attributi associati ad AUTOMOBILE sono semplici: Vehicle_ID, Color, Weight e Horsepower.

Attributi a valore singolo contro attributi multivalore

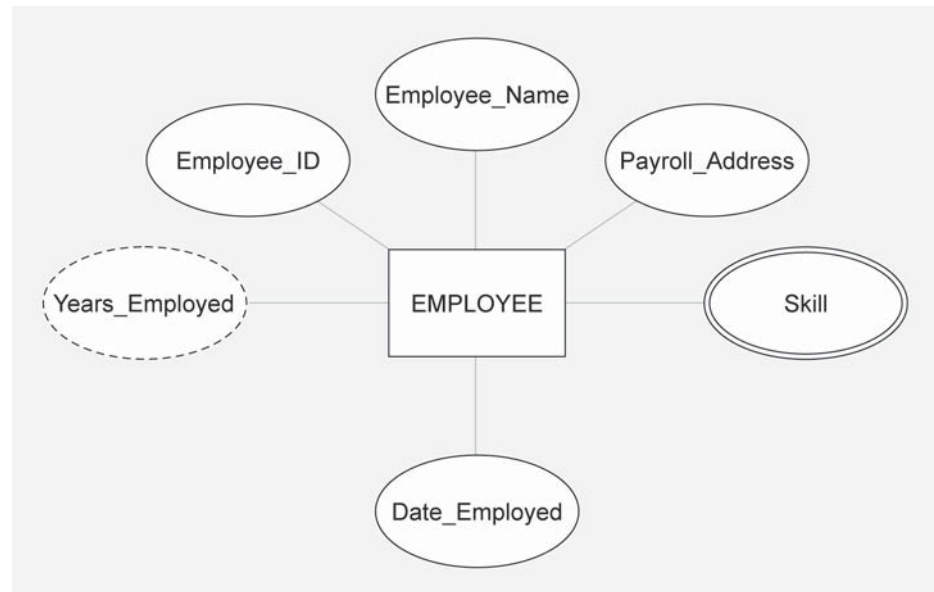
La Figura 5.6 mostra due entità con i rispettivi valori di attributo. Per ciascuna istanza di entità, ognuno degli attributi nella figura ha un valore. Accade spesso che ci sia un attributo che può avere più di un valore per una data istanza. Per esempio, il tipo di entità EMPLOYEE nella Figura 5.8 ha un attributo chiamato Skill i cui valori registrano le competenze di un impiegato. Naturalmente alcuni impiegati possono avere più di una competenza (come i programmatori C++ e Cobol). **Un attributo multivalore** è un attributo che può assumere più di un valore per un'istanza di entità data. Indicheremo un attributo multivalore con un'ellisse dal contorno doppio, come mostrato per l'attributo Skill nell'esempio di EMPLOYEE della Figura 5.8.

Figura 5.7

Un attributo composto.

**Figura 5.8**

Entità con un attributo multivalore (Skill) e un attributo derivato (Years_Employed).

**Attributi memorizzati contro attributi derivati**

Alcuni valori di attributo che sono di interesse per gli utenti possono essere calcolati o derivati da altri valori di attributo correlati che sono memorizzati nella base di dati. Per esempio, si supponga che per un'azienda il tipo di entità EMPLOYEE abbia un attributo Date_Employed. Se gli utenti devono sapere per quanti anni una persona ha lavorato per l'azienda, potranno calcolare questo valore utilizzando Date_Employed e la data odierna. Un attributo derivato è un attributo i cui valori possono essere calcolati da valori di attributo correlati (oltre eventualmente da dati non presenti nella base di dati, come la data odierna, l'ora corrente o un codice di sicurezza fornito da un utente del sistema). In un diagramma E-R indicheremo un attributo utilizzando un'ellisse con un contorno tratteggiato, come mostrato nella Figura 5.8.

In alcune situazioni, il valore di un attributo può essere derivato da attributi nelle entità correlate. Per esempio, si consideri la fattura creata per ciascun cliente di Pine Valley Furniture Company (Figura 3.6). Order_Total è un attributo dell'entità INVOICE, che indica l'importo totale in dollari che viene fatturato al cliente. Il valore di Order_Total può essere calcolato sommando i valori di Extended_Price per gli articoli delle varie linee che sono conteggiati nella fattura. Le formule per il calcolo di valori come questi sono un tipo di regola aziendale.

Attributo identificatore

Un **identificatore** è un attributo (o combinazione di attributi) che identifica in modo univoco istanze individuali di un tipo di entità. L'identificatore del tipo di entità STUDENT introdotto precedentemente è Student_ID, mentre quello per AUTOMOBILE è Vehicle_ID. Si noti che un attributo come Student_Name non è un identificatore candidato, poiché potenzialmente molti studenti possono avere lo stesso nome e poiché gli studenti, come tutti, possono cambiare il loro nome. Per essere un identificatore candidato, ogni istanza di entità deve avere un singolo valore per l'attributo e l'attributo deve essere associato all'entità. Nel diagramma E-R i nomi degli identificatori appaiono sottolineati, come mostrato nella Figura 5.9a.

Nel caso di alcuni tipi di entità, non c'è un attributo singolo (o atomico) che può servire come identificatore (che può cioè assicurare univocità). Tuttavia, due o più attributi utilizzati in combinazione possono servire da identificatori. Un **identificatore composto** è un identificatore costituito da un attributo composto. La Figura 5.9b mostra l'entità FLIGHT con l'identificatore composto Flight_ID. A sua volta, Flight_ID possiede gli attributi di componente Flight_Number e Date. Questa combinazione è necessaria per identificare in modo univoco le singole occorrenze di FLIGHT. Per convenzione, l'attributo composto (Flight_ID) appare sottolineato per indicare che è l'identificatore, mentre gli attributi del componente non sono sottolineati.

Alcune entità possono avere più di un identificatore candidato. In questo caso, il progettista deve sceglierne uno come identificatore. Bruce (1992) suggerisce i seguenti criteri per selezionare gli identificatori.

1. Scegliere un identificatore che non modificherà il proprio valore nel ciclo di vita di ciascuna istanza di tipo di entità. Per esempio, la combinazione di Employee_Name e Payroll_Address (anche se univoca) sarebbe una scelta scadente come identificatore per EMPLOYEE, perché i valori di Employee_Name e Payroll_Address possono cambiare facilmente durante il periodo di assunzione di un dipendente.
2. Scegliere un identificatore tale per cui per ogni istanza dell'entità garantisca che l'attributo avrà valori validi e non nulli (o non noti). Se l'identificatore è un attributo composto, (come Flight_ID nella Figura 5.9), assicurarsi che tutte le sue parti abbiano valori validi.
3. Evitare l'uso dei cosiddetti identificatori intelligenti (o chiave), la cui struttura indica classificazioni, luoghi e così via. Per esempio, le prime due cifre di un identificatore possono indicare il luogo di un magazzino. Tali codici vengono spesso modificati al cambiare delle condizioni, cosa che rende i valori degli identificatori non validi.
4. Prendere in considerazione l'ipotesi di sostituire identificatori grandi e composti con identificatori surrogati ad attributo singolo. Per esempio, un attributo chiamato Game_Number può essere utilizzato per il tipo di entità GAME al posto della combinazione di Home_Team e Visiting_Team.

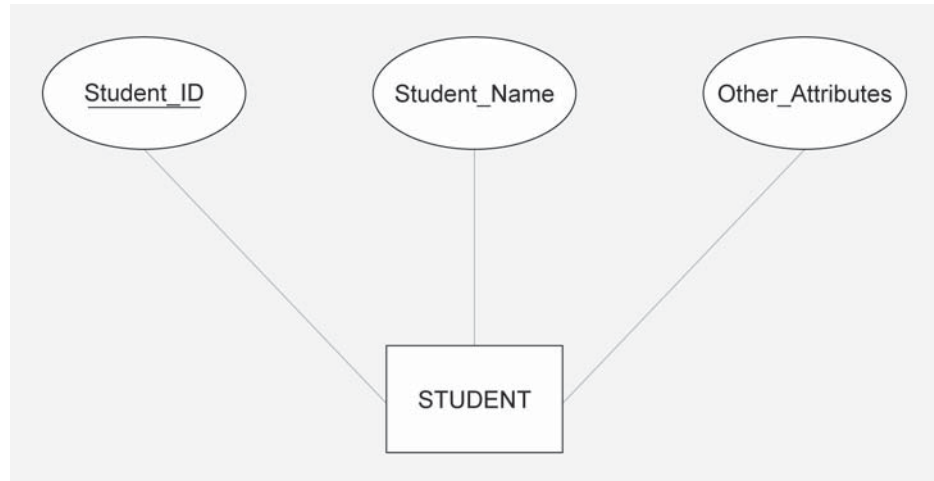
Denominare e definire gli attributi

Oltre alle linee guida generali per la denominazione degli oggetti di base, ci sono alcune linee guida per la denominazione degli attributi.

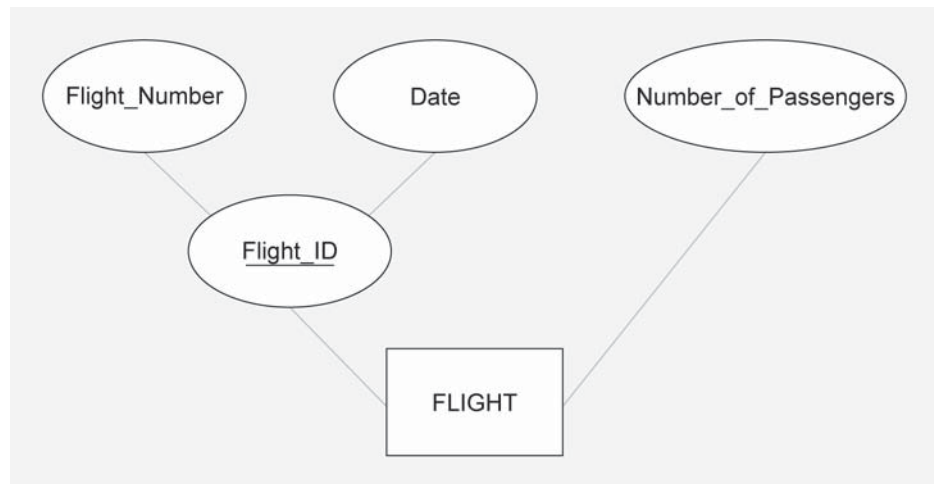
Figura 5.9

Attributi di chiave semplici e composti.

(a) Attributo di chiave semplice.



(b) Attributo di chiave composto.



- Un nome di attributo è un nome in senso grammaticale (come Customer_ID, Age, Product, Product_Minimum_Price o Major). Gli attributi, che si concretizzano in valori di dati, sono concetti o caratteristiche fisiche delle entità. Tali concetti e caratteristiche vengono descritti da nomi.
- Un attributo deve essere univoco. Due attributi dello stesso tipo di entità non possono avere lo stesso nome ed è preferibile, per ragioni di chiarezza, che due attributi tra tutti i tipi di entità non abbiano lo stesso nome.
- Per rendere un nome univoco e per ragioni di chiarezza, ogni nome di attributo deve seguire un formato standard. Per esempio, l'università può stabilire Student_GPA invece di GPA_Student come esempio di standard di denominazione degli attributi. Il formato da utilizzare verrà stabilito da ciascuna azienda. Un formato comune è {[_Qualificatore]}_Class, dove [...] è una clausola facoltativa e {...} indica che la clausola può ripetersi. Il nome di tipo di entità è il nome dell'entità a cui è associato l'attributo. Il nome del tipo di entità può essere utilizzato per rendere il nome dell'attributo esplicito. Viene utilizzato quasi sempre per l'attributo identificatore (per esempio, Customer_ID) di

ciascun tipo di entità. *Class* è una frase tratta da un elenco di frasi definite dall'azienda che corrispondono alle caratteristiche consentite delle entità (o alle abbreviazioni di queste caratteristiche). Per esempio, valori consentiti (e abbreviazioni associate approvate) per Class possono essere Name (Nm), Identificativo (ID), Date (Dt) o Amount (Amt). Ovviamente è necessario anche Class. *Qualificatore* è una frase tratta da un elenco di frasi definite dall'azienda che vengono utilizzate per porre vincoli sulle classi. Possono essere necessari uno o più qualificatori per rendere univoco ogni attributo di un tipo di entità. Per esempio, un qualificatore può essere Maximum (Max), Hourly (Hrly) o State (St). Un qualificatore può anche non essere necessario: Employee_Age e Student_Major sono entrambi nomi di attributi espliciti. A volte un qualificatore è necessario; per esempio, Employee_Birth_Date ed Employee_Hire_Date sono due attributi di Employee che richiedono un qualificatore. Può essere necessario anche più di un qualificatore; per esempio, Employee_Residence_City_Name (o Emp_Res_Cty_Nm) è il nome della città di residenza di un impiegato, mentre Employee_Tax_City_Name (o Emp_Res_Tax_Nm) è il nome della città nella quale un impiegato paga le tasse comunali.

- Attributi simili di tipi di entità diversi devono utilizzare gli stessi qualificatori e le stesse classi fino a quando questi sono nomi utilizzati nell'azienda. Per esempio, la città di residenza degli insegnanti e degli studenti dovrebbe essere rispettivamente Faculty_Residence_City_Name e Student_Residence_City_Name. Utilizzare nomi simili fa sì che gli utenti capiscano più facilmente che i valori di questi attributi derivano probabilmente dallo stesso insieme di valori, che chiameremo *domini*. Gli utenti potrebbero voler approfittare dei vantaggi dei domini comuni nelle interrogazioni (per esempio, trovare gli utenti che vivono nella stessa città dei loro tutor) e riconosceranno facilmente che questo tipo di corrispondenza può essere possibile se vengono utilizzati lo stesso qualificatore e le stesse frasi.

Esistono anche alcune linee guida specifiche per la definizione degli attributi.

- Una definizione di attributo afferma *cos'è l'attributo e perché è importante*. La definizione riprende spesso in parallelo il nome dell'attributo; per esempio, Student_Residence_City_Name può essere definito come "Il nome della città in cui uno studente ha la propria residenza definitiva".
- Una definizione di attributo rende chiaro *cosa è incluso e cosa non è incluso* nel valore di un attributo. Per esempio, "Employee_Monthly_Salary_Amount è l'importo pagato ogni mese nella valuta del paese di residenza dell'impiegato esclusi benefit, bonus, rimborsi o pagamenti speciali".
- Qualsiasi *alias*, o nome alternativo, dell'attributo deve essere specificato nella definizione o può essere incluso altrove nella documentazione specifica, eventualmente memorizzata in un repository dello strumento CASE utilizzato per mantenere le definizioni dei dati.
- È preferibile includere nella definizione *la fonte dei valori dell'attributo*. La definizione della fonte può rendere più chiaro il significato. Per esempio, "Customer_Standard_Industrial_Code è un'indicazione del tipo di azienda del cliente. I valori di questo codice derivano da un insieme di valori standard forniti dalla Federal Trade Commission e si trovano su un CD SIC che abbiamo acquistato e che viene fornito annualmente da FTC".
- Una definizione di attributo (o altra specifica in un repository di strumenti CASE) deve anche indicare *se il valore dell'attributo è richiesto o è facoltativo*. La regola aziendale sull'attributo è importante per mantenere l'integrità dei dati. L'attributo identificativo di un tipo di entità viene richiesto per definizione. Se un valore di attributo viene richiesto, allora per creare un'istanza del tipo di entità deve essere fornito un valore di questo attributo. "Richiesto" significa che un'istanza di entità deve sempre avere un valore per questo attributo, e non solo quando l'istanza viene creata. "Facoltativo" significa che un valore deve sempre esistere. Per esempio, "Employee_Department_ID è l'identi-

ficatore della funzione a cui l'impiegato è assegnato. Un impiegato può non essere assegnato ad alcuna funzione quando viene assunto (quindi inizialmente l'attributo è facoltativo), ma una volta che viene assegnato a una funzione, l'impiegato dovrà sempre essere assegnato a una funzione”.

- Una definizione di attributo (o altra specifica in un repository di strumenti CASE) deve anche indicare *se il valore dell'attributo può cambiare* una volta che il valore è fornito e prima che l'istanza di entità venga eliminata. Questa regola aziendale controlla anche l'integrità dei dati. Gli identificatori non intelligenti possono non cambiare i valori nel tempo. Per assegnare un identificatore non intelligente a un'istanza di entità, quell'istanza deve prima essere eliminata e poi ricreata.
- Nel caso di un attributo multivalore, la definizione dell'attributo deve indicare *il numero massimo e minimo di occorrenze di un valore di attributo per un'istanza di entità*. Per esempio, “Employee_Skill_Name è il nome di una competenza di un impiegato. Ogni impiegato deve possedere almeno una competenza, e un impiegato può scegliere da un elenco di 10 competenze al massimo”. La ragione di un attributo multivalore può essere che occorre conservare una storia dell'attributo. Per esempio, “Employee_Yearly_Absent_Days_Number è il numero di giorni in un anno di calendario durante il quale l'impiegato è stato assente dal lavoro. Un impiegato è considerato assente se lavora meno del 50 percento delle ore previste in un giorno. Un valore di questo attributo deve essere conservato per ogni anno in cui l'impiegato lavora per l'azienda”.
- Una definizione di attributo deve anche *indicare qualsiasi associazione dell'attributo con altri attributi*. Per esempio, “Employee_Vacation_Days_Number è il numero di giorni di vacanze pagate all'impiegato. Se l'impiegato ha un valore di ‘Exempt’ per Employee_Type, allora il valore massimo di Employee_Vacation_Days_Number verrà determinato da una formula che implica il numero di anni di servizio dell'impiegato”.

5.4 Associazioni

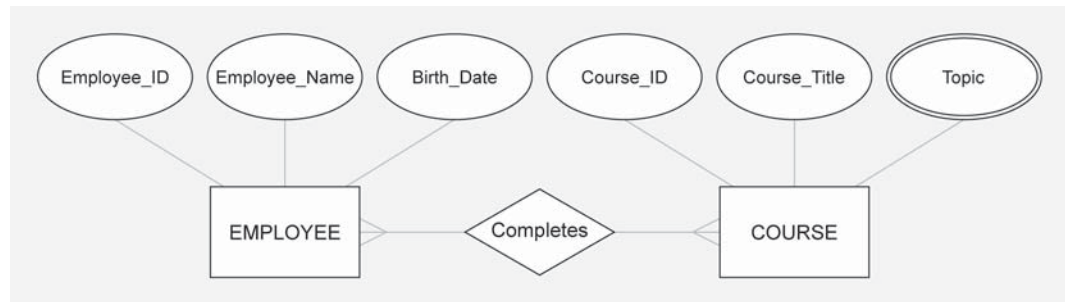
Le associazioni sono il collante che tiene insieme i vari componenti di un modello E-R. Nello specifico, un'associazione lega tra loro le istanze di uno o più tipi di entità che sono di interesse per l'azienda. Per capire più chiaramente questa definizione, occorre distinguere tra i tipi di associazioni e le istanze di associazione. Si considerino i tipi di entità EMPLOYEE e COURSE, dove COURSE rappresenta i corsi di formazione che devono essere seguiti dagli impiegati. Per rintracciare i corsi che sono stati completati da un determinato impiegato, si definisce un'associazione Completes tra due tipi di entità (Figura 5.10a). Si tratta di un'associazione multi-a-molti, poiché ogni impiegato può completare un certo numero di corsi, mentre un dato corso può essere seguito da più impiegati. Per esempio, nella Figura 5.10b, l'impiegato Melton ha completato tre corsi (C++, COBOL e Perl). Il corso su SQL è stato completato da due impiegati (Celko e Gosling).

In questo esempio, ci sono due tipi di entità (EMPLOYEE e COURSE) che partecipano all'associazione denominata Completes. In generale, qualsiasi numero di tipi di entità (da uno a molti) possono partecipare a un'associazione. In questo capitolo e in quelli successivi utilizzeremo la convenzione di un rombo con un'etichetta che contiene il verbo singolo che rappresenta l'associazione. Poiché le associazioni si verificano spesso a causa di un evento dell'azienda, le istanze di evento sono correlate, perché è stata intrapresa un'azione; un verbo singolo è quindi adatto a questo tipo di etichetta. Il verbo deve essere al tempo presente e deve essere descrittivo. Ci sono tuttavia molti modi per rappresentare un'associazione. Nel Capitolo 3 (Figura 3.3) è stata utilizzata un'altra notazione diffusa, con una linea semplice senza rombo, ma con due nomi di associazioni, uno per ciascun nome di relazione. Alcuni modellatori di dati preferiscono il formato con due nomi di associazioni. Queste due notazioni hanno lo stesso significato strutturale, quindi possono essere utilizzate a piacere.

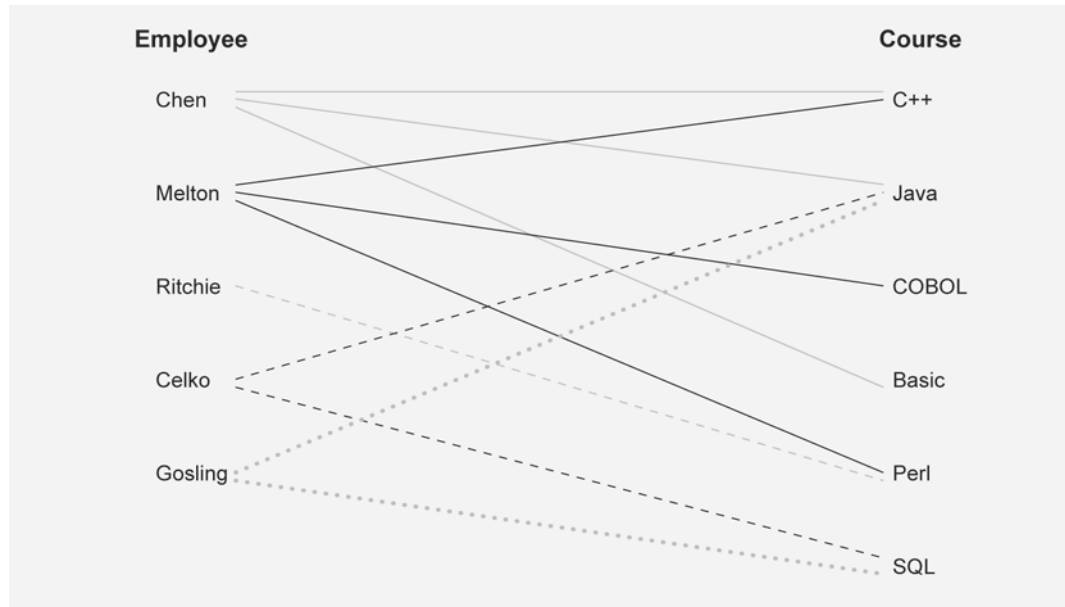
Figura 5.10

Le associazioni: tipi e istanze.

(a) Tipo di associazione (Completes).



(b) Istanze di associazione.



5.4.1 Concetti e definizioni di base nelle associazioni

Un tipo di associazione è una relazione significativa tra due o più tipi di entità. La frase “relazione significativa” implica che l’associazione permette di rispondere alle domande alle quali non si può rispondere con i soli tipi di entità. Un tipo di associazione è indicata da un rombo che contiene il nome dell’associazione, come nell’esempio mostrato nella Figura 5.10a. Nella denominazione dell’associazione, è preferibile utilizzare un verbo singolo e breve che abbia senso per l’utente (diremo qualcosa di più sulla denominazione e definizione delle associazioni successivamente nel paragrafo).

Un’istanza di associazione è una relazione tra una o più istanze di entità, dove ciascuna istanza di associazione include esattamente un’entità da ciascun tipo di entità coinvolto (Elmasri e Navathe, 1994). Per esempio, nella Figura 5.10b, ogni linea nella figura rappresenta un’istanza di associazione tra un impiegato e un corso, indicando che l’impiegato ha completato quel corso.

Attributi sulle associazioni

Gli attributi possono essere associati ad associazioni multi-a-molti (o uno-a-uno), così come a un'entità. Per esempio, si supponga che l'azienda voglia registrare la data (mese e anno) in cui un impiegato completa ciascun corso. Questo attributo è chiamato Date_Completed. Per avere alcuni dati di esempio, si faccia riferimento alla Tabella 5.2.

Tabella 5.2 Istanze che mostrano Date_Completed.

Employee_Name	Course Title	Date_Completed
Chen	C++	06/2000
Chen	Java	09/2000
Chen	Basic	10/2000
Melton	C++	06/2000
Melton	COBOL	02/2001
Melton	SQL	03/2000
Ritchie	Perl	11/2000
Celko	Java	03/2000
Celko	SQL	03/2001
Gosling	Java	09/2000
Gosling	Perl	06/2000

Dove deve essere collocato l'attributo Date_Completed nel diagramma E-R? Osservando la Figura 5.10a, si può notare che Date_Completed non è stato associato né all'entità EMPLOYEE né all'entità COURSE. Questo perché Date_Completed è una proprietà dell'associazione Completes e non una proprietà di una delle due entità. In altre parole, per ciascuna istanza della relazione Completes, c'è un valore per Date_Completed. Una di queste istanze mostra che l'impiegato chiamato Melton ha completato il corso su C++ nel giugno 2000.

Una versione rivista del diagramma E-R di questo esempio è mostrata nella Figura 5.11a. In questo diagramma, l'attributo Date_Completed è connesso al simbolo dell'associazione che collega le due entità. Se opportuno, a questa associazione possono essere aggiunti altri attributi, come Course_Grade, Instructor e Room_Location.

Entità associative

La presenza di uno o più attributi nell'associazione suggerisce al progettista che l'associazione dovrebbe invece essere rappresentata come tipo di entità. Un'entità associativa è un tipo di entità che associa le istanze di uno o più tipi di entità e che contiene attributi specifici all'associazione tra queste istanze di entità. L'entità associativa CERTIFICATE è rappresentata dal simbolo di associazione del rombo racchiuso nel rettangolo dell'entità, come mostrato nella Figura 5.11b.

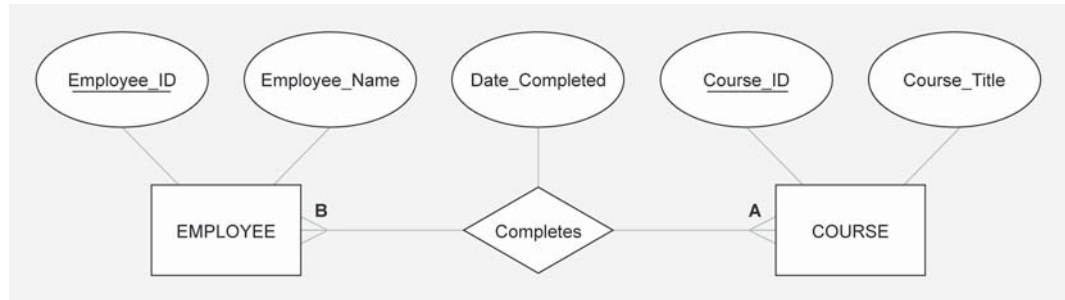
Lo scopo di questo simbolo speciale è quello di conservare l'informazione che l'entità era inizialmente specificata come associazione nel diagramma E-R. Le entità associative vengono anche chiamate gerundi, poiché il nome dell'associazione (un verbo) viene solitamente convertito in questa forma.

Nella Figura 5.11b si nota un rombo di associazione sulla linea tra un'entità associativa e un'entità forte. Questo accade perché l'entità associativa rappresenta l'associazione.

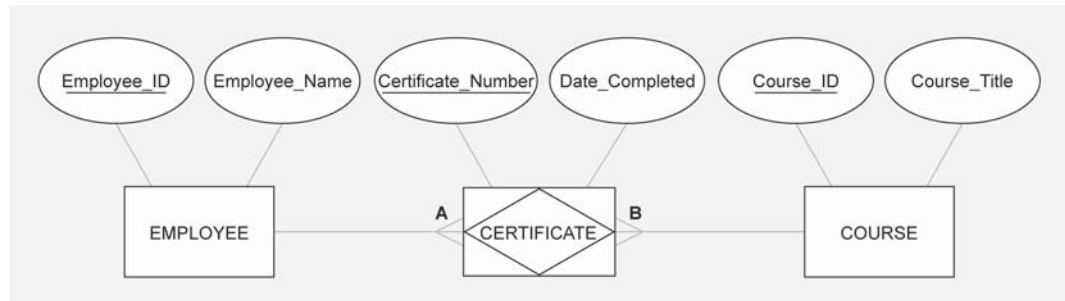
Come si può capire se convertire o meno un'associazione in un tipo di entità associativa? Devono esistere quattro condizioni.

Figura 5.11

Un'entità associativa.
(a) Attributo in un'associazione.



(b) Un'entità associativa associativa (CERTIFICATE).



1. Tutte le associazioni di partecipazione dei tipi di entità sono associazioni del tipo “molti”.
2. Il tipo di entità associativa risultante ha un significato indipendente per gli utenti finali ed è preferibilmente identificato con un identificatore ad attributo singolo.
3. L'entità associativa ha uno o più attributi, oltre all'identificatore.
4. L'entità associativa partecipa a una o più associazioni indipendenti delle entità correlate nella relazione associata.

La Figura 5.11b mostra l'associazione *Completes* convertita in un tipo di entità associativa. In questo caso, l'ufficio formazione dell'azienda ha deciso di conferire un attestato a ciascun impiegato che completa un corso. L'entità viene quindi chiamata *CERTIFICATE*, che ha un significato indipendente per gli utenti finali. Ogni attestato ha un numero (*Certificate_Number*) che serve come identificatore. Viene anche incluso l'attributo *Date_Completed*.

Si noti che la conversione di un'associazione a un'entità associativa ha comportato uno spostamento della notazione dell'associazione. La cardinalità “molti” ora termina infatti in corrispondenza dell'entità associativa e non più del tipo di entità. Nella Figura 5.11, un impiegato, che può completare uno o più corsi (notazione A nella Figura 5.11a) può ricevere più di un attestato (notazione A nella Figura 5.11b); inoltre, un corso, che può essere completato da uno o più impiegati (notazione B nella Figura 5.11a) può rilasciare uno o più certificati (notazione B nella Figura 5.11b).

5.4.2 Grado delle associazioni

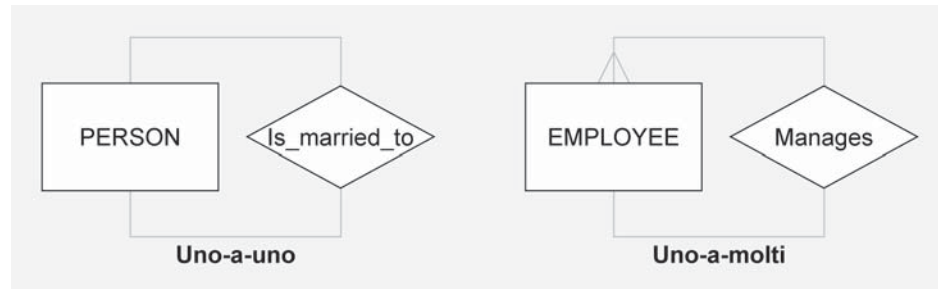
Il **grado** di un'associazione è il numero di tipi di entità che partecipano a quell'associazione. Per esempio, l'associazione *Completes* è di grado 2, poiché ci sono due tipi di entità, *EMPLOYEE* e *COURSE*. I tre gradi di associazione più comuni sono quello unario (grado 1), quello binario (grado 2) e quello

ternario (grado 3). Sono possibili associazioni di grado anche più alto, ma sono piuttosto rare. Esempi dei tre tipi di gradi delle associazioni sono mostrati nella Figura 5.12.

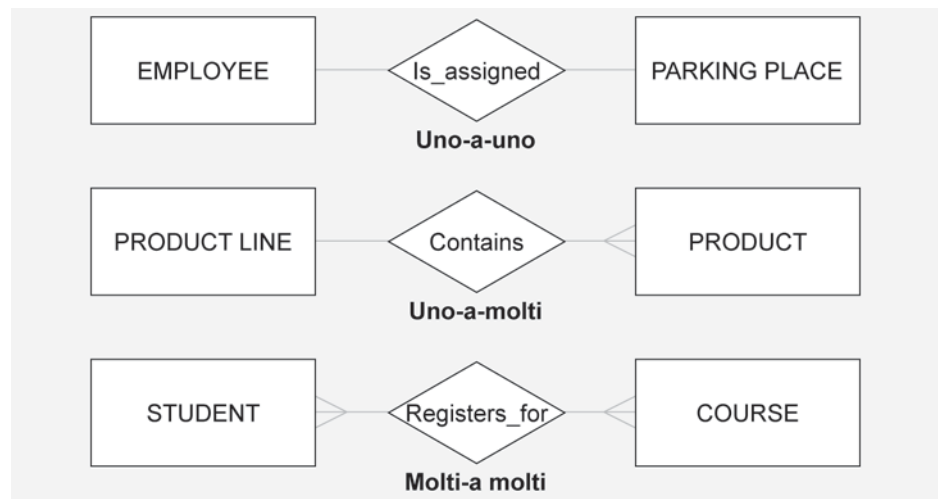
Figura 5.12

Esempi di associazioni di grado diverso.

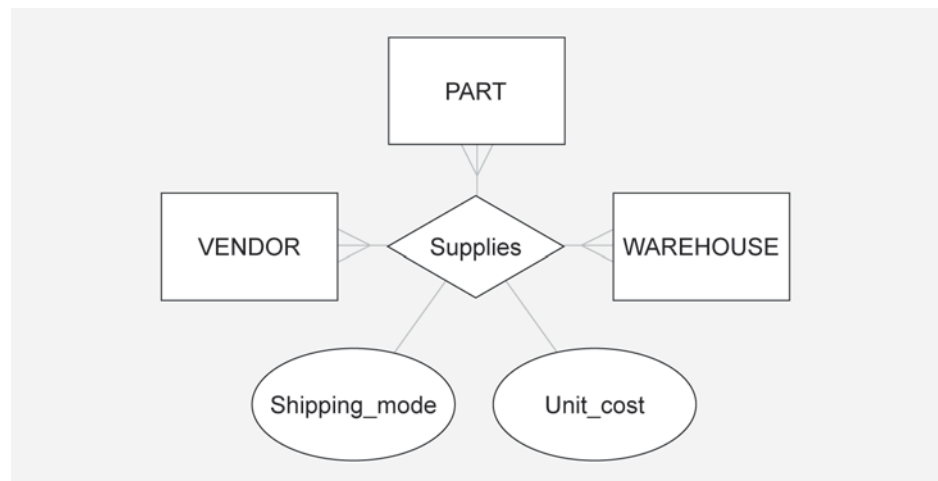
(a) Associazioni unarie.



(b) Associazioni binarie.



(a) Associazioni ternarie.



Osservando la figura, si può capire chiaramente che qualsiasi modello di dati specifico rappresenta una determinata situazione e non una generalizzazione. Per esempio, si consideri l'associazione Managers nella Figura 5.12a. In alcune aziende, può essere possibile che un impiegato sia gestito da molti altri impiegati (come nel caso di un'azienda a matrice). Quando si sviluppa un modello E-R, è importante capire le regole aziendali dell'azienda specifica che si sta modellando.

Associazione unaria

Un'associazione unaria è un'associazione tra le istanze di un *singolo* tipo di entità (le associazioni unarie sono chiamate anche associazioni *ricorsive*). Due esempi sono mostrati nella Figura 5.12a. Nel primo esempio, "Is_married_to" è mostrato come associazione uno-a-uno tra istanze del tipo di entità PERSON. Poiché questa è un'associazione uno-a-uno, questa notazione indica che solo il matrimonio corrente, se ne esiste uno, deve essere mantenuto rispetto a una persona. Nel secondo esempio, "Manages" è mostrata come associazione uno-a-molti tra istanze del tipo di entità EMPLOYEE. Utilizzando questa associazione, è possibile identificare per esempio gli impiegati che fanno riferimento a un determinato manager. (Nota: in questi esempi si ignora se queste associazioni siano a cardinalità obbligatoria [mandatory] o facoltativa, come mostrato nella Figura 5.2; introdurremo questi concetti in un paragrafo successivo.)

La Figura 5.13 mostra un esempio di un'altra associazione unaria, chiamata struttura bill-of-materials. Molti manufatti sono costituiti da assemblaggi, che a loro volta sono costituiti da sotto-assemblaggi, parti e così via. Come mostrato nella Figura 5.13a, si può rappresentare questa struttura come associazione unaria multi-a-molti. Nella figura, il tipo di entità ITEM viene utilizzato per rappresentare tutti i tipi di componenti, e si utilizza "Has_components" per il nome del tipo di associazione che associa gli articoli al livello più basso con quelli al livello più alto.

Due occorrenze di questa struttura vengono mostrate nella Figura 5.13b. Ciascun diagramma mostra i componenti diretti di ciascun articolo e le quantità di ogni componente. Per esempio, l'articolo X è costituito dall'articolo U (quantità 3) e dall'articolo V (quantità 2). Può accadere facilmente che le associazioni siano di tipo multi-a-molti. Molti degli articoli hanno più di un tipo di componente (per esempio, l'articolo A ha tre tipi di componenti diretti: V, X e Y). Alcuni dei componenti vengono inoltre utilizzati in numerosi assemblaggi ad alto livello. Per esempio, l'articolo X è utilizzato sia nell'articolo A sia nell'articolo B. L'associazione multi-a-molti assicura che, per esempio, la stessa struttura di sotto-assemblaggio di X venga utilizzata ogni volta che l'articolo X diventa parte di qualche altro articolo.

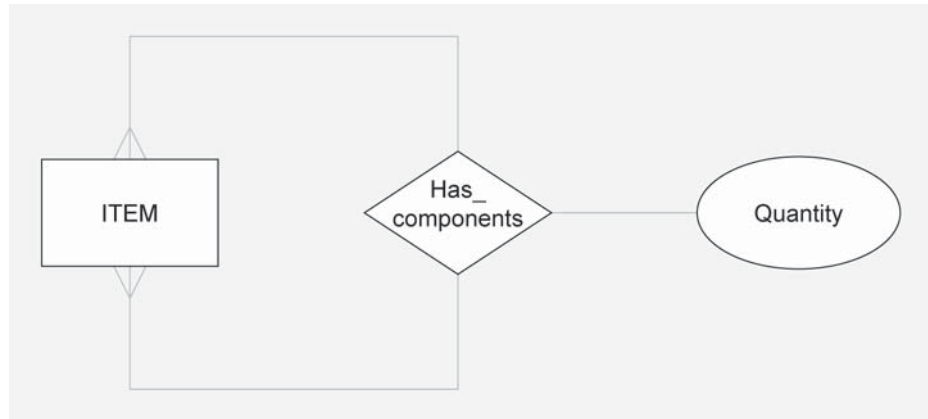
La presenza dell'attributo Quantity nell'associazione suggerisce che l'analista considera la conversione dell'associazione "Has_components" in un'entità associativa. La Figura 5.13c mostra il tipo di entità BOM_STRUCTURE, che forma un'associazione tra istanze del tipo di entità ITEM. Un secondo attributo (Effective_Date) è stato aggiunto a BOM_STRUCTURE per registrare la data in cui questo componente è stato utilizzato per la prima volta nell'assemblaggio correlato. Le date effettive sono spesso necessarie quando, per esempio, si richiede una storia delle valvole.

Associazione binaria

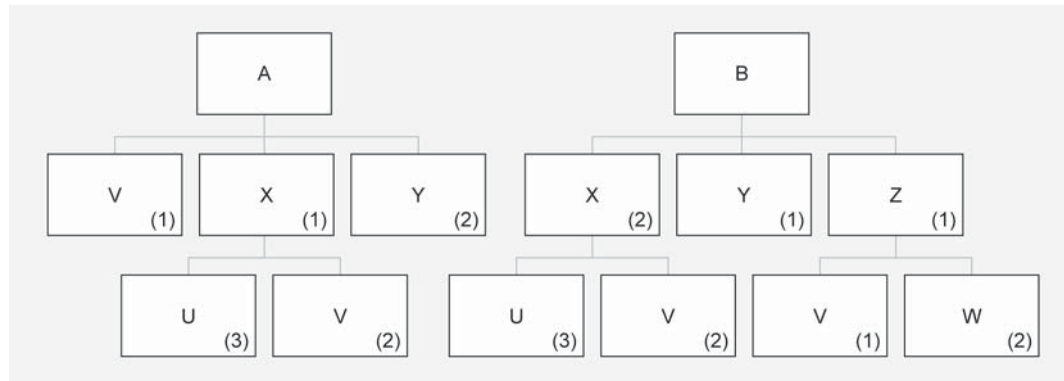
Un'associazione binaria è un'associazione tra le istanze di due tipi di entità ed è il tipo di associazione più comune. La Figura 5.12b mostra tre esempi. Il primo (uno-a-uno) indica che a un impiegato è assegnato un posto auto e che ciascun posto auto è assegnato a un impiegato. Il secondo (uno-a-molti) indica che una linea di prodotti può contenere più prodotti e che ciascun prodotto appartiene a una sola linea di prodotti. Il terzo (molti-a-molti) mostra che uno studente può iscriversi a più di un corso e che a ogni corso possono iscriversi molti studenti.

Figura 5.13

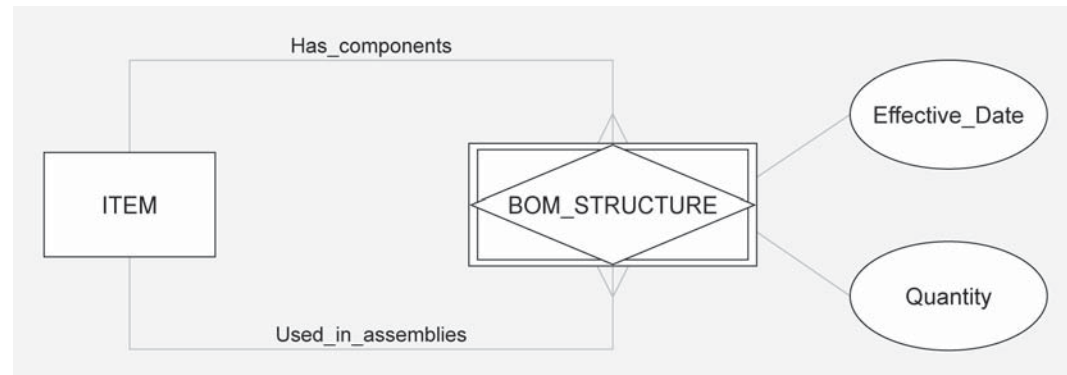
Rappresentazione della struttura bill-of-materials.
(a) Associazione multi-a-molti.



(b) Due istanze.



(c) Entità associativa.



Associazione ternaria

Un'associazione ternaria è un'associazione *contemporanea* tra le istanze di tre tipi di entità. Una situazione aziendale tipica che porta a un'associazione ternaria è mostrata nella Figura 5.12c. In questo esempio, i produttori possono fornire parti specifiche che vengono fornite da un dato produttore a un magazzino specifico. Esistono tre tipi di entità: **VENDOR**, **PART** e **WAREHOUSE**. Nell'associazione Supplies ci sono due attributi: **Shipping Mode** e **Unit Cost**. Per esempio, Supplies può registrare il fatto

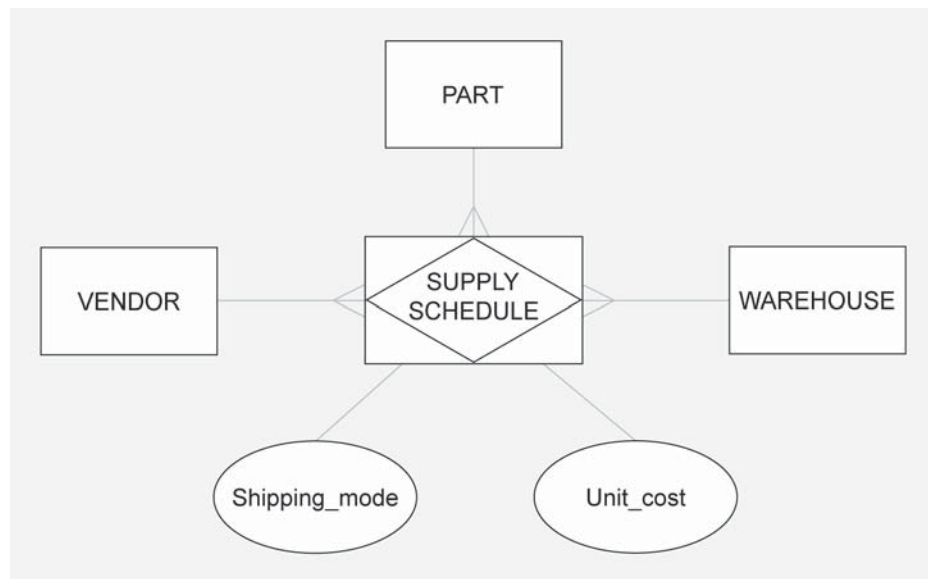
che il produttore X può consegnare la parte C al magazzino Y, che la modalità di spedizione è per via aerea il giorno successivo e che il costo è di 5 \$ per unità.

Si noti che un'associazione ternaria non è la stessa cosa di tre associazioni binarie. Per esempio, nella Figura 5.12c, Unit_Cost è un attributo dell'associazione Supplies. Unit_Cost non può essere associata in modo opportuno ad alcuna delle tre associazioni binarie possibili tra i tre tipi di entità (per esempio tra PART e WAREHOUSE). Così, per esempio, se si dicesse che il produttore X può consegnare la parte C per un costo unitario di 8 \$, i dati sarebbero incompleti, poiché non indicano a quale magazzino vengono consegnate le parti.

Come di consueto, la presenza di un attributo nell'associazione Supplies nella Figura 5.12c suggerisce la conversione dell'associazione in un tipo di entità associativa. La Figura 5.14 mostra una rappresentazione alternativa, egualmente corretta, dell'associazione ternaria mostrata nella Figura 5.12c. Nella Figura 5.14 il tipo di entità associativa SUPPLY SCHEDULE viene utilizzato per sostituire l'associazione Supplies della Figura 5.12c. Il tipo di entità SUPPLY SCHEDULE è chiaramente di interesse indipendente per gli utenti. Si noti tuttavia che non è stato ancora assegnato un identificatore a SUPPLY SCHEDULE. Questo è accettabile. Se durante la modellazione E-R non si assegna un identificatore a un'entità associativa, un identificatore (o chiave) verrà assegnato durante la modellazione logica (trattata nel Capitolo 8). Questo sarà un identificatore composto i cui componenti saranno costituiti dall'identificatore di ciascun tipo di entità che partecipa all'associazione (in questo caso PART, VENDOR e WAREHOUSE). Possono esistere altri attributi associati a SUPPLY SCHEDULE?

Figura 5.14

Associazione ternaria come entità associativa.



Come detto in precedenza, non si utilizzano simboli a rombo sulle linee che collegano SUPPLY SCHEDULE alle tre entità. Questo perché queste linee non rappresentano associazioni binarie. Per mantenere lo stesso significato dell'associazione ternaria della Figura 5.12c, non si può suddividere l'associazione Supplies in tre associazioni binarie.

Si consiglia vivamente di convertire tutte le associazioni ternarie (o di grado più elevato) in entità associative, come in questo esempio. Song, Evans e Park (1995) mostrano che i vincoli particolari

(descritti nel prossimo paragrafo) non possono essere rappresentati accuratamente nel caso di un'associazione ternaria, data la notazione utilizzata in questo capitolo. Questo può invece avvenire nella conversione di un'entità associativa. Inoltre, numerosi strumenti di disegno di diagrammi E-R, compresi molti strumenti CASE, non possono rappresentare associazioni ternarie. Quindi, sebbene il risultato non sia semanticamente accurato, è possibile rappresentare l'associazione ternaria con un'entità associativa e tre associazioni binarie.

5.4.3 Attributi o associazioni?

La Figura 5.15 include tre esempi di situazioni in cui un attributo può essere rappresentato mediante un'associazione. Nella Figura 5.15a, i pre-requisiti di un corso sono essi stessi dei corsi (e un corso può essere un pre-requisito per altri corsi). Il pre-requisito può quindi essere visualizzato come associazione unaria tra corsi e non come attributo multivalore di COURSE. La rappresentazione dei pre-requisiti mediante un'associazione unaria significa anche che trovare i pre-requisiti di un corso e trovare i corsi per i quali un corso è un pre-requisito ha a che fare con le associazioni tra i tipi di entità. Quando i pre-requisiti sono un attributo multivalore di COURSE, trovare i corsi per i quali un corso è un pre-requisito significa cercare un valore specifico per il pre-requisito in tutte le istanze di COURSE. Nella Figura 5.15b, gli impiegati hanno competenze, ma queste possono essere visualizzate come un tipo di entità riguardo al quale l'azienda vuole mantenere i dati (il codice univoco per identificare ciascuna competenza, un titolo descrittivo e il tipo di competenza, per esempio tecnica o manageriale). Un impiegato ha competenze, ma non come attributi, bensì come istanze di un tipo di dati correlato. Nei casi delle Figure 5.15a e 5.15b, la rappresentazione dei dati come attributo multivalore invece che mediante un'associazione con un altro tipo di entità può, per alcuni, semplificare il diagramma.

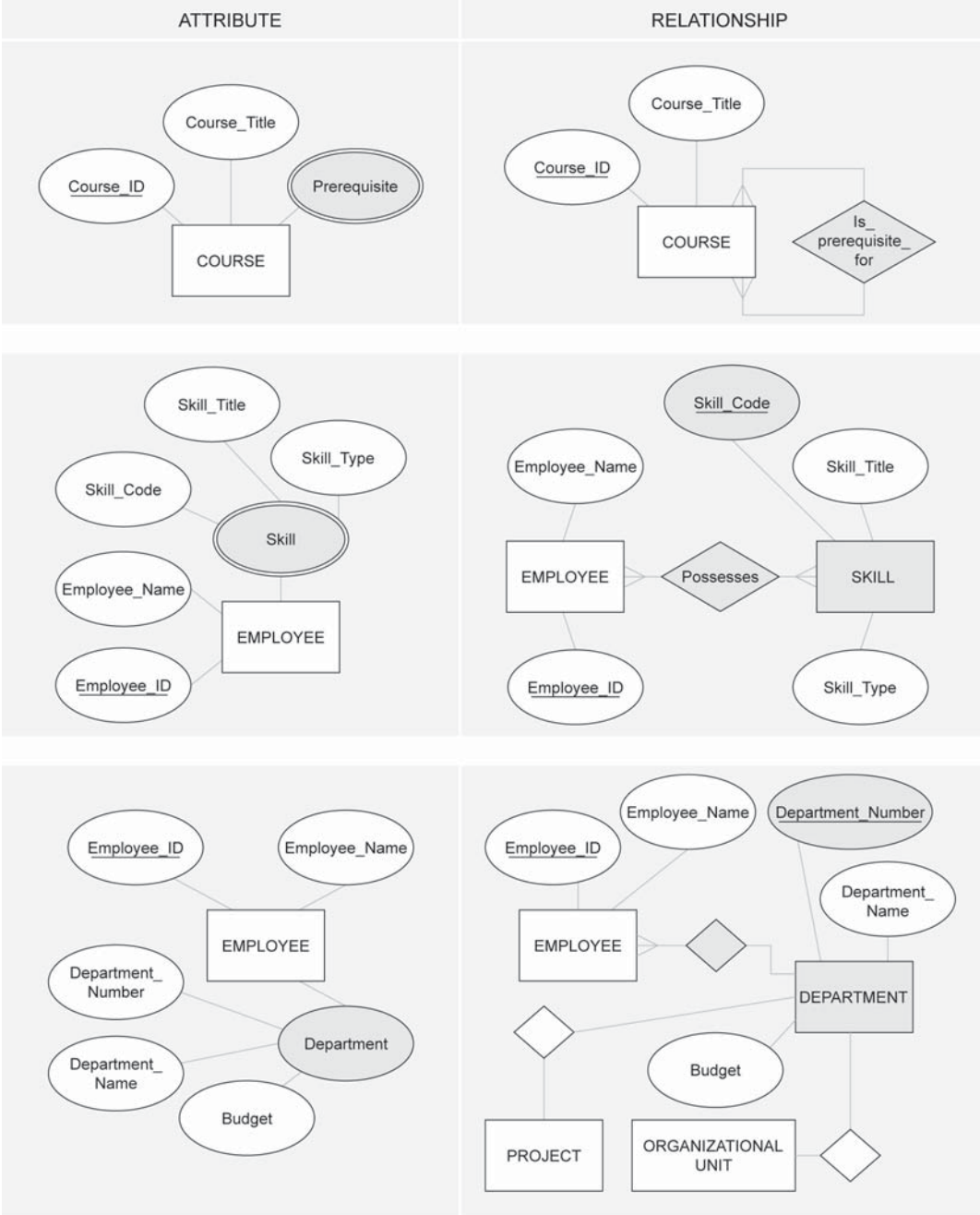
In definitiva, quando un attributo *dovrebbe* essere collegato a un tipo di entità mediante un'associazione? La risposta è: quando l'attributo è l'identificatore o qualche altra caratteristica di un tipo di entità in un modello di dati e le istanze di entità multiple devono condividere gli stessi attributi. La Figura 5.15c rappresenta un esempio di questa regola. In questo esempio, EMPLOYEE ha un attributo composto di Department. Poiché Department è un concetto dell'azienda e più impiegati condivideranno gli stessi dati di funzione, questi potranno essere rappresentati (in modo non ridondante) in un tipo di entità DEPARTMENT, con attributi per i dati sulle funzioni che tutte le altre istanze di entità correlate devono conoscere. Con questo approccio, non solo diversi impiegati condividono la memorizzazione degli stessi dati di funzione, ma anche i progetti (che sono assegnati a una funzione) e le unità dell'azienda (che sono composte da funzioni) possono condividere la memorizzazione degli stessi dati di funzione.

5.4.4 Vincoli di cardinalità

Si supponga che ci siano due tipi di entità, A e B, connessi da un'associazione. Un **vincolo di cardinalità** specifica il numero di istanze dell'entità B che possono o devono essere associate a ogni istanza dell'entità A. Per esempio, si consideri un negozio di video che noleggia videocassette e film. Poiché il negozio può avere a disposizione più di una videocassetta per ogni film, si avrà ovviamente un'associazione "uno-a-molti", come mostrato nella Figura 5.16a. È anche vero che il negozio potrebbe non avere a disposizione videocassette di un dato film in un momento particolare (per esempio, tutte le copie potrebbero essere state noleggiate). Servirà quindi una notazione più precisa per indicare il *grado* di cardinalità di un'associazione. Questa notazione è stata introdotta nella Figura 5.2, che si può tenere come riferimento.

Figura 5.15

Utilizzare le associazioni per collegare attributi correlati.
(a) Attributo multivalore contro associazione.
(b) Attributo multivalore composto contro associazione.
(c) Attributo composto di dati condivisi con altri tipi di entità.



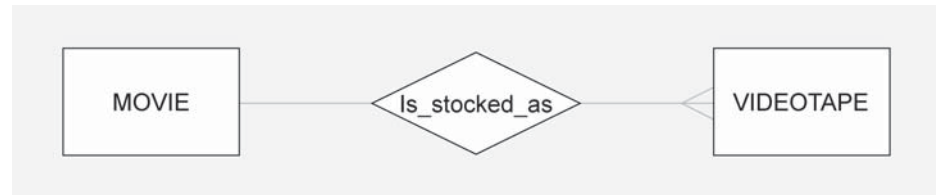
Cardinalità minima

La **cardinalità minima** di un'associazione è il numero minimo di istanze dell'entità B che possono essere associate a ogni istanza dell'entità A. Nell'esempio del negozio, il numero di videocassette di un film è zero. Quando il numero minimo di partecipanti è zero, si dice che il tipo di entità B è un partecipante *facoltativo* all'associazione. In questo esempio, VIDEOTAPE è un partecipante facoltativo nell'associazione "Is_stocked_as". Questo fatto è indicato dal simbolo zero nella freccia vicino all'entità VIDEOTAPE nella Figura 5.16b.

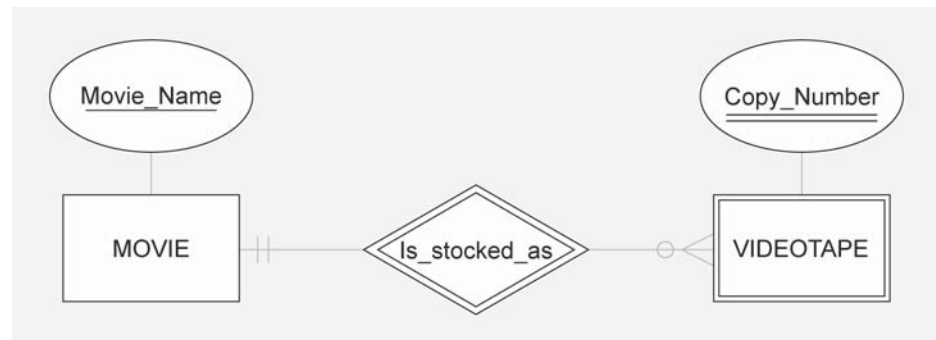
Figura 5.16

Presentazione dei vincoli di cardinalità.

(a) Associazione di base.



(b) Associazione con vincoli di cardinalità.



Cardinalità massima

La **cardinalità massima** di un'associazione è il numero massimo di istanze dell'entità B che possono essere associate a ogni istanza dell'entità A. Nell'esempio del negozio, la cardinalità massima per il tipo di entità VIDEOTAPE è "molti", ossia un numero non specificato maggiore di uno. Questo fatto è indicato dal simbolo della "zampa di gallina" nella freccia vicino all'entità VIDEOTAPE nella Figura 5.16b.

Un'associazione è naturalmente bidirezionale, quindi è presente anche la notazione di cardinalità accanto all'entità MOVIE. Si noti che il minimo e il massimo sono entrambi 1. In questo caso si parla di cardinalità *obbligatoria uno*. In altre parole, ogni videocassetta di film deve essere una copia esattamente di un film. In generale, la partecipazione a un'associazione può essere facoltativa o obbligatoria per le entità coinvolte. Se la cardinalità minima è zero, la partecipazione è facoltativa; se la cardinalità minima è 1, la partecipazione è obbligatoria.

Nella Figura 5.16b, sono stati aggiunti alcuni attributi a ciascun tipo di entità. Si noti che VIDEOTAPE è rappresentato da un'entità debole. Questo perché una videocassetta non può esistere a meno che non esista anche il filmato possessore. L'identificatore di MOVIE è Movie_Name. VIDEOTAPE non ha un identificatore univoco. Tuttavia Copy_Number è un identificatore *parziale* che, con Movie_Name, identifica in modo univoco un'istanza di VIDEOTAPE.

Alcuni esempi

Esempi di tre associazioni che mostrano tutte le combinazioni possibili di cardinalità minima e massima appaiono nella Figura 5.17. Ogni esempio definisce la regola aziendale per ciascun vincolo di cardinalità e mostra la notazione E-R associata. Gli esempi mostrano anche alcune istanze per chiarire la natura dell'associazione. È necessario studiare con attenzione ciascuno di questi esempi. Quelle che seguono sono le regole aziendali per ciascuno degli esempi nella Figura 5.17.

1. PATIENT Has PATIENT HISTORY (Figura 5.17a). Ogni paziente ha una o più storie di pazienti (la visita iniziale del paziente è sempre registrata come un'istanza di PATIENT HISTORY). Ogni istanza di PATIENT HISTORY “appartiene” esattamente a un PATIENT.
2. EMPLOYEE Is_assigned_to PROJECT (Figura 5.17b). A ogni PROJECT è assegnato almeno un EMPLOYEE (qualche progetto può averne anche più di uno). Ogni EMPLOYEE può o, facoltativamente, può non essere assegnato ad alcun PROJECT esistente (per esempio, impiegato “Pete”) o può essere assegnato a uno o più PROJECT.
3. PERSON Is_married_to PERSON (Figura 5.17c). Questo è uno zero facoltativo o una cardinalità in entrambe le direzioni, poiché una persona può essere o non essere sposata.

È possibile che la cardinalità massima sia un numero fisso, e non un valore “molti” arbitrario. Per esempio, si supponga che la politica aziendale stabilisca che un impiegato può lavorare su un massimo di cinque progetti contemporaneamente. Si può rappresentare questa regola aziendale collocando un 5 sopra o sotto il simbolo di zampa di gallina accanto all'entità PROJECT nella Figura 5.17b.

Associazione ternaria

Abbiamo mostrato l'associazione ternaria con il tipo di entità associativa SUPPLY SCHEDULE nella Figura 5.14. Ora aggiungeremo i vincoli di cardinalità a questo diagramma in base alle regole aziendali per questa situazione. Il diagramma E-R, con le regole aziendali importanti, è mostrato nella Figura 5.18. Si noti che PART e WAREHOUSE sono partecipanti obbligatori nell'associazione, mentre VENDOR è un partecipante facoltativo. La cardinalità di ciascuna delle entità partecipanti è obbligatoria mente uno, poiché ogni istanza di SUPPLY SCHEDULE deve essere correlata esattamente a una istanza di ciascuna con i questi tipi di entità “partecipanti”.

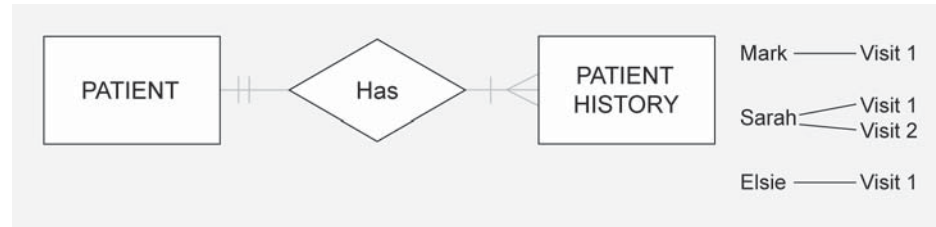
Come notato in precedenza, un'associazione ternaria non equivale a tre associazioni binarie. Sfortunatamente, con molti strumenti CASE non è possibile disegnare associazioni ternarie; si è invece costretti a rappresentare le associazioni ternarie come tre binarie. In questo caso, è sufficiente disegnare le associazioni binarie con i rombi e accertarsi che la cardinalità accanto alle tre entità forti sia obbligatoria uno.

5.4.5 Modellare dati dipendenti dal tempo

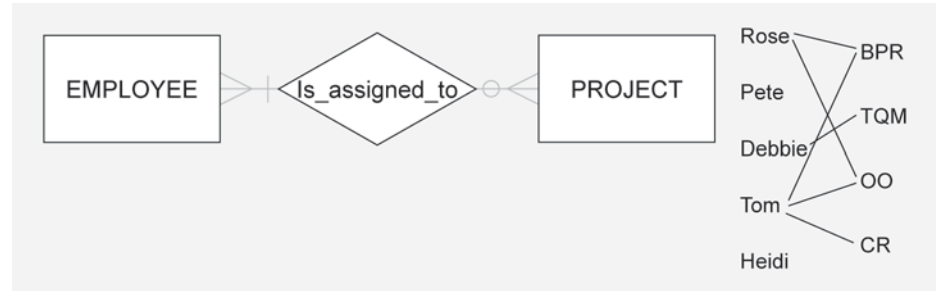
I contenuti di una base di dati variano nel tempo. Per esempio, in una base di dati che contiene informazioni sui prodotti, il prezzo unitario di ciascun prodotto può essere modificato laddove cambiano i costi dei materiali e delle lavorazioni. Se viene richiesto solo il prezzo corrente, allora Price può essere modellato come attributo a valore singolo. Per casi di contabilità, fatturazione e altri casi, può essere tuttavia più probabile che occorra una storia dei prezzi e del periodo di tempo nel quale erano in vigore. Come mostra la Figura 5.19, si possono concettualizzare queste esigenze come una serie di prezzi e di data effettivi per ciascun prezzo. Questo dà come risultato l'attributo multivalore (composto) Price_History. I componenti di Price_History sono Price ed Effective_Date. Una caratteristica importante di un attributo composto e multivalore è che gli attributi del componente vengono collegati. Così, nella Figura 5.19, ogni valore è accoppiato alla Effective_Date corrispondente.

Figura 5.17

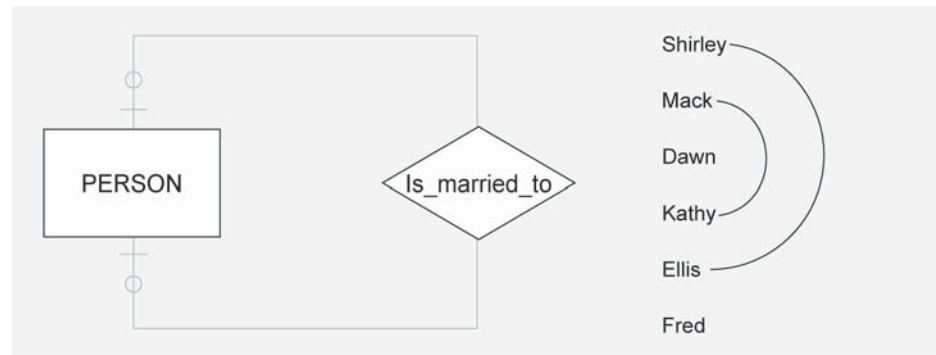
Esempi di vincoli di cardinalità.
(a) Cardinalità obbligatorie.



(b) Cardinalità facoltativa uno, obbligatoria uno.



(c) Cardinalità facoltative.

**Figura 5.18**

Vincoli di cardinalità
in un'associazione
ternaria.

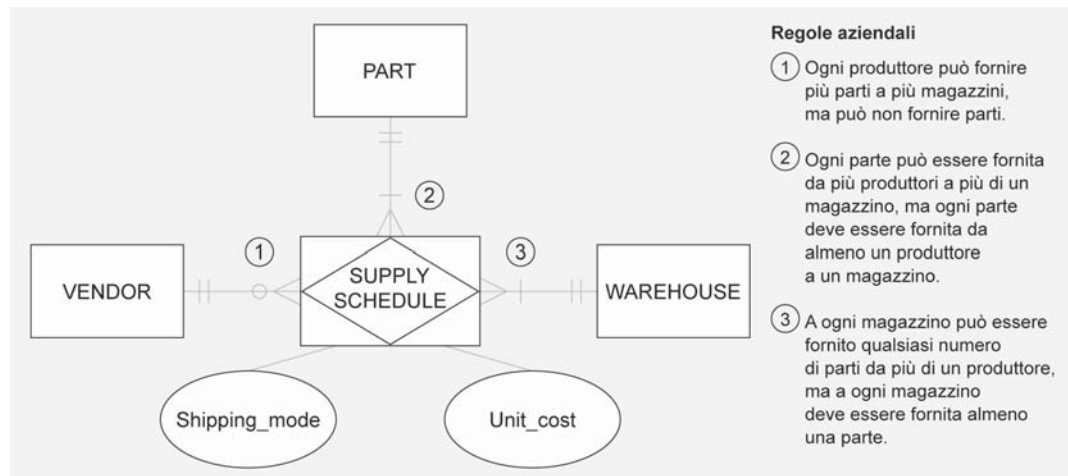
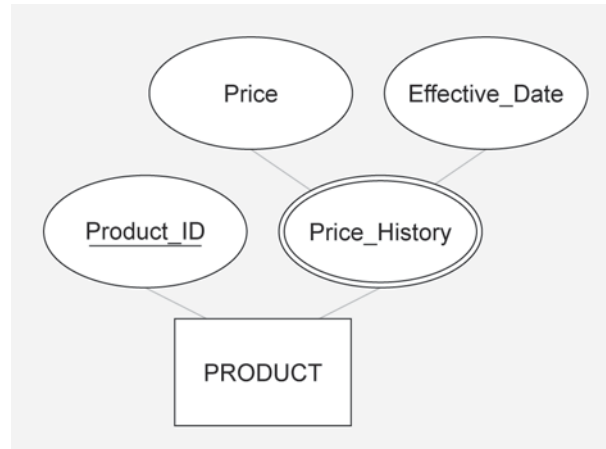


Figura 5.19

Un esempio semplice di marcatura temporale.



Nella Figura 5.19, ogni valore dell'attributo Price ha una marcatura temporale con la data effettiva. Una **marcatura temporale** (*time stamp*) è semplicemente un valore di tempo (come data e ora) che è associato a una data o valore. Una marcatura temporale può essere associata a qualsiasi valore di dato che cambia nel tempo quando è necessario mantenere una storia di questi valori. Le marcature temporali possono essere registrate per indicare l'ora in cui un valore è stato immesso (ora della transazione), l'ora in cui è diventato valido o ha cessato di esserlo o l'ora in cui sono state eseguite operazioni importanti (come aggiornamenti, correzioni o verifiche).

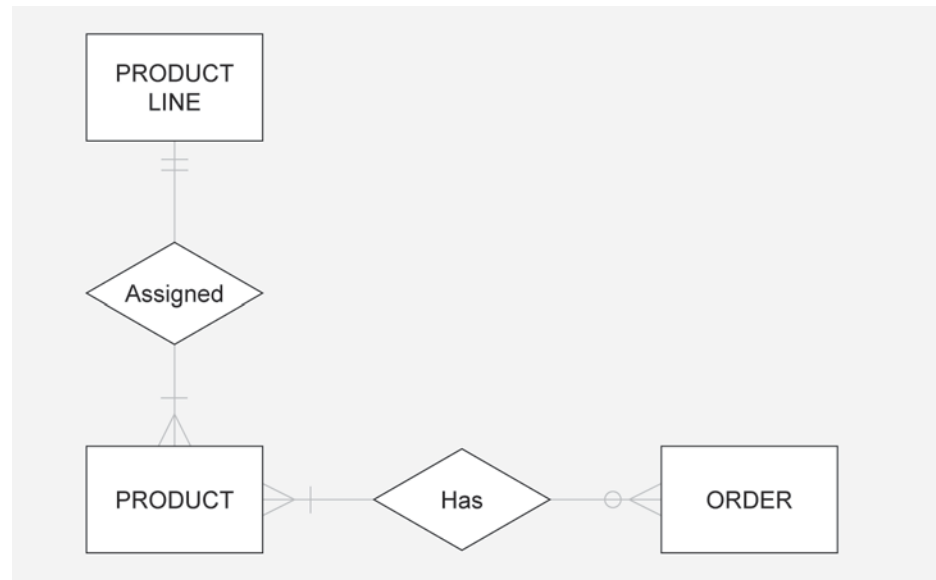
L'uso di una marcatura temporale semplice, come quella dell'esempio precedente, è spesso adatta nella modellazione di dati che dipendono dal tempo. La variabile "tempo" può tuttavia introdurre alcune complessità insidiose nella modellazione dei dati. Per esempio, la Figura 5.20a rappresenta una parte di un diagramma E-R per Pine Valley Furniture. Ogni prodotto è assegnato a una linea di prodotti (o gruppo di prodotti correlati). Gli ordini dei clienti sono elaborati lungo l'anno e vengono stilati riepiloghi mensili in base alla linea di prodotti e al prodotto in quella linea.

Si supponga che a metà anno, in base a una riorganizzazione della funzione delle vendite, alcuni prodotti vengano riassegnati a linee di prodotto diverse. Il modello mostrato nella Figura 5.20a non è concepito per tenere traccia della riassegnazione del prodotto a una nuova linea. Tutti i report di vendita mostreranno quindi vendite cumulative per un prodotto in base alla sua linea *corrente* e non in base a quella del momento della vendita. Per esempio, un prodotto può avere un totale di vendite annuale di 50.000 \$ ed essere associato alla linea di prodotti B, ma 40.000 \$ di queste vendite possono essere avvenute mentre il prodotto era assegnato alla linea di prodotti A. Questa situazione va persa utilizzando il modello della Figura 5.20a. Il semplice cambiamento di progettazione mostrato nella Figura 5.2b comporta il riconoscimento corretto delle riassegnazioni dei prodotti. Una nuova associazione, chiamata *Sales_for_product_line*, è stata aggiunta tra ORDER e PRODUCT LINE. Quando gli ordini di un cliente vengono elaborati, vengono accreditati sia al prodotto corrente sia alla linea di prodotti corretta al momento della vendita.

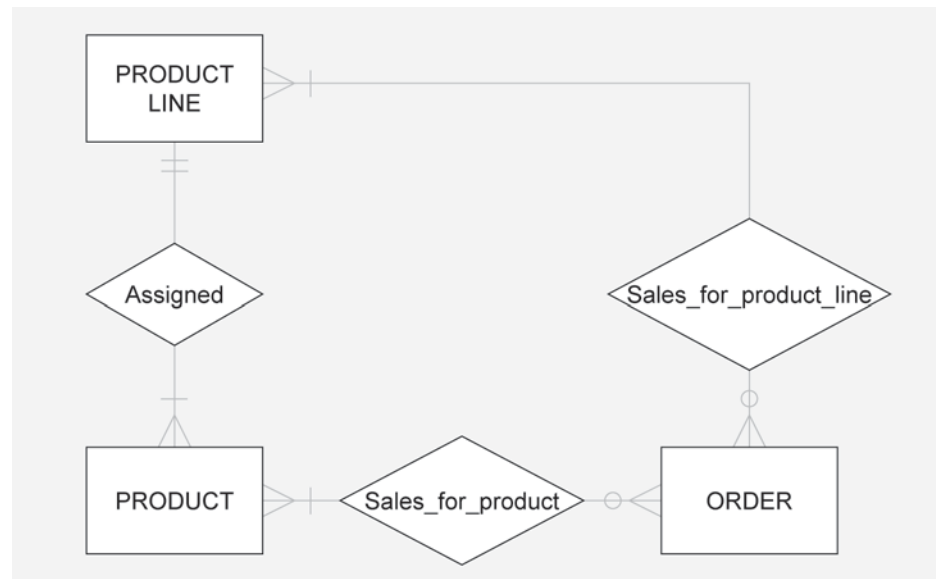
Abbiamo discusso il problema dei dati dipendenti dal tempo con i manager di numerose aziende leader nell'uso della modellazione dei dati e della gestione delle basi di dati. Queste discussioni hanno rivelato che i modelli di dati correnti (e i sistemi di gestione delle basi di dati basati su questi modelli) sono generalmente inadatti a gestire i dati dipendenti dal tempo e che le aziende ignorano spesso questo problema e sperano che le imprecisioni risultanti si compensino in qualche modo. Le applicazioni di data warehouse sono concepite per eliminare molte di queste incertezze fornendo progetti espliciti per

Figura 5.20

Base di dati dei prodotti di Pine Valley Furniture.
 (a) Diagramma E-R in cui non si riconosce la riassegnazione del prodotto.



(b) Diagramma E-R in cui si riconosce la riassegnazione del prodotto.



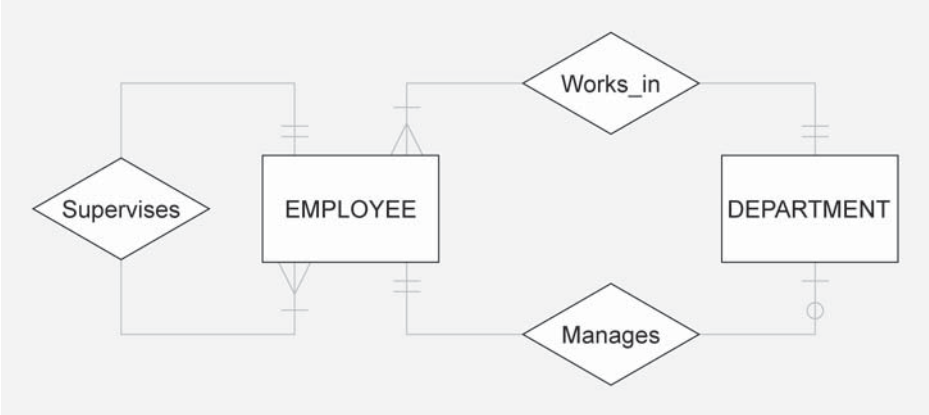
dati dipendenti dal tempo. È quindi necessario avvertire delle complessità poste da questo tipo di dati man mano che si sviluppano i modelli di dati di un'azienda.

5.4.6 Associazioni multiple

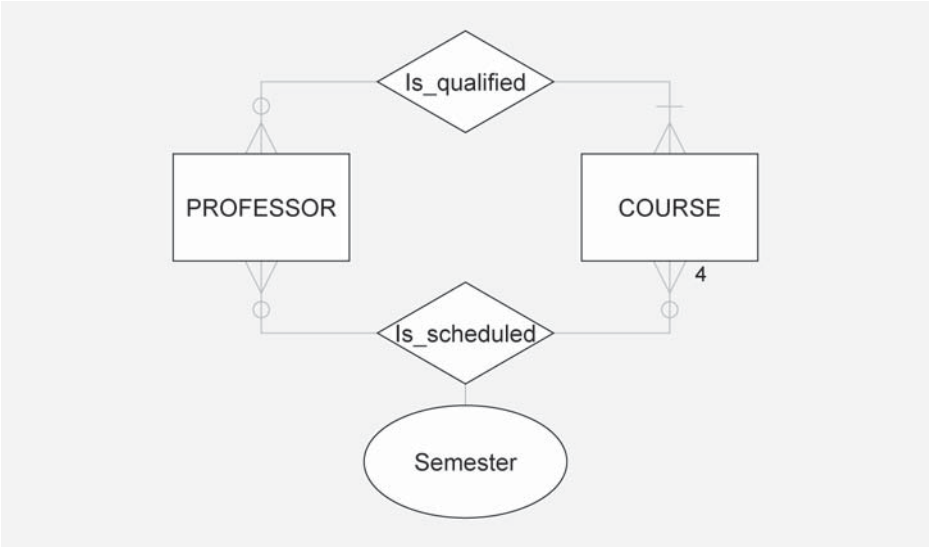
In alcuni casi, un'azienda può voler modellare più di un'associazione tra gli stessi tipi di entità. Due esempi sono mostrati nella Figura 5.21. La Figura 5.21a mostra due associazioni tra i tipi di entità EM-

PLOYEE e DEPARTMENT. La relazione Works_in associa gli impiegati alla funzione in cui lavorano. Questa associazione è di tipo uno-a-molti e obbligatoria in entrambe le direzioni. La funzione deve cioè avere almeno un impiegato che vi lavora (per esempio, il manager) e ogni impiego deve essere assegnato esattamente a una funzione. (Nota: queste sono le regole aziendali specifiche per questa illustrazione. Quando si sviluppa un diagramma E-R per una particolare situazione, è cruciale capire le regole aziendali che si applicano a quella impostazione. Per esempio, se EMPLOYEE dovesse includere i pensionati, allora ogni impiegato potrebbe non essere correntemente associato esattamente a una funzione; il modello E-R nella Figura 5.21 presume inoltre che l'azienda voglia ricordare in quale DEPARTMENT lavora correntemente ciascun EMPLOYEE, piuttosto che ricordare la storia delle assegnazioni alla funzione. Anche in questo caso la struttura del modello di dati riflette le informazioni che l'azienda ha la necessità di ricordare.)

Figura 5.21
Esempi di relazioni multiple.
(a) Impiegati e reparti.



(b) Professori e corsi (stabiliti su un vincolo).



La seconda associazione tra EMPLOYEE e DEPARTMENT (denominata “Manages”) associa ogni funzione all’impiegato che gestisce la funzione. L’associazione da EMPLOYEE e DEPARTMENT è obbligatoria uno, indicando che una funzione deve avere esattamente un manager. Da EMPLOYEE a DEPARTMENT l’associazione è facoltativa uno, poiché un dato impiegato è o non è un manager di funzione.

La Figura 5.21a mostra anche l’associazione unaria Supervises che associa ciascun impiegato al suo supervisore e viceversa. L’associazione registra la regola aziendale in base alla quale qualsiasi impiegato deve avere esattamente un supervisore. Per contro, ogni impiegato può essere supervisore di un numero qualsiasi di impiegati, o può non essere un supervisore.

L’esempio nella Figura 5.21b mostra le due associazioni tra i tipi di entità PROFESSOR e COURSE. L’associazione Is_qualified associa gli insegnanti ai corsi per i quali sono qualificati a insegnare. Un dato corso può avere più di un insegnante qualificato o, facoltativamente, può non avere insegnanti qualificati. Questo può accadere, per esempio, se il corso è nuovo ed è appena entrato nell’ordinamento. Per contro, ciascun insegnante deve essere qualificato per tenere almeno un corso (un’aspettativa ragionevole!).

La seconda associazione nella figura collega i professori ai corsi che questi, secondo il programma, devono tenere in un dato semestre. Si noti che la cardinalità massima per i corsi programmati per gli insegnanti durante un semestre è quattro. Questo è un esempio di come si può registrare un vincolo fisso (superiore o inferiore) in un diagramma E-R. L’attributo Semester (che potrebbe essere un attributo composto con i componenti Semester_Name e Year) si trova nell’associazione Is_scheduled.

Il diagramma E-R nella Figura 5.12b rappresenta i vincoli di cardinalità di base. Spesso ci sono tuttavia altre regole aziendali che possono non essere facilmente rappresentate in un diagramma E-R di base. Si consideri la regola seguente per l’esempio mostrato nella Figura 5.12b: “Un insegnante che in base al programma deve tenere un corso deve essere qualificato per insegnare in quel corso” (un’altra regola ragionevole!). Nel Capitolo 6 tratteremo le tecniche per la rappresentazione di queste regole come parte del processo di modellazione complessivo.

5.4.7 Denominare e definire le associazioni

Oltre alle linee guida generali per la denominazione degli oggetti di base, esistono alcune linee guida speciali per la denominazione delle associazioni.

- Il nome di un’associazione deve essere una *frase verbale* (come Assigned_to, Supplies o Teaches). Le associazioni rappresentano azioni da intraprendere, solitamente al tempo presente. Il nome di un’associazione definisce l’azione intrapresa e non il risultato dell’azione (per esempio, si utilizza Assigned_to e non Assignment). Il nome definisce l’essenza dell’interazione tra i tipi di entità coinvolti e non il processo (per esempio, si dice che un Employee è Assigned_to a un progetto, e non che un Employee sta Assigning un progetto).

- È preferibile evitare nomi vaghi, come Has o Is_related_to. Utilizzare invece frasi verbali descrittive, traendole dai verbi di azione che si trovano nella definizione dell’associazione.

Esistono anche linee guida specifiche per la definizione delle associazioni.

- Una definizione di associazione spiega quale azione deve essere intrapresa ed eventualmente perché è importante. Può essere importante definire chi o cosa esegue l’azione, ma non è importante spiegare perché l’azione viene eseguita. Affermare gli oggetti aziendali implicati nell’associazione è naturale, ma poiché il diagramma E-R mostra quali tipi di entità sono implicati nell’associazione e altre definizioni spiegano i tipi di entità, descrivere gli oggetti aziendali non è necessario.

- Può essere importante *fornire esempi per chiarire l'azione*. Per esempio, per una relazione Registered_for tra studente e corso, può essere utile spiegare che questa copre sia l'iscrizione classica sia quella on-line e che include le iscrizioni effettuate durante il periodo di frequenza/assenza.
- La definizione deve spiegare la *partecipazione facoltativa*. Si dovrebbe spiegare quali condizioni portano a istanze associate pari a zero, se questo può accadere solo quando un'istanza di entità viene creata per la prima volta o in qualsiasi momento. Per esempio, "Registered_for collega un corso agli studenti che si sono iscritti al corso e ai corsi a cui uno studente si è iscritto. Gli studenti non possono iscriversi al corso prima che inizi il periodo di iscrizione e non si potranno mai avere studenti iscritti. Uno studente non verrà iscritto ad alcun corso prima che inizi il periodo di iscrizione e non potrà iscriversi ad alcuna lezione (o potrà iscriversi alle lezioni e poi ritirarsi da una o da tutte le lezioni)".
- Una definizione di associazione deve anche *spiegare la ragione di un'eventuale cardinalità massima* diversa da molte. Per esempio, "Assigned_to collega un impiegato ai progetti a cui è assegnato e gli impiegati assegnati a un progetto. Per un accordo sindacale, un impiegato non può essere assegnato a più di quattro progetti contemporaneamente". Questo esempio, tipico di numerose regole aziendali di limiti superiori, suggerisce che le cardinalità massime tendono a non essere definitive. In questo esempio, il prossimo accordo sindacale potrà aumentare o ridurre questo limite. Di conseguenza, l'implementazione delle cardinalità massime dovrà essere impostata in modo da consentire queste modifiche.
- Una definizione di associazione deve *spiegare le eventuali associazioni rispettivamente esclusive*. Queste associazioni sono tali per cui un'istanza di entità può partecipare solo a un'associazione fra più associazioni alternative. Mostriamo alcuni esempi di questa situazione nel Capitolo 6. Per ora, si consideri questo esempio: "Plays_on collega una squadra sportiva inter-universitaria ai suoi giocatori studenti e indica in quale squadra gioca uno studente. Gli studenti che giocano in squadre inter-universitarie non possono anche fare un lavoro all'interno del campus (ossia, uno studente non può essere collegato sia a una squadra sportiva inter-universitaria attraverso Plays_on sia a un lavoro nel campus attraverso l'associazione Works_on)". Un altro esempio di limite reciprocamente esclusivo è quello di un impiegato che non può essere Supervised_by e Married_to con lo stesso impiegato.
- Una definizione di associazione deve *spiegare le eventuali restrizioni alla partecipazione nell'associazione*. L'esclusività reciproca è una restrizione, ma ve ne possono essere altre. Per esempio, "Supervised_by collega un impiegato ad altri impiegati che sono suoi supervisori. Un impiegato non può essere supervisore di se stesso e non può essere supervisore di altri impiegati se il suo livello è inferiore al quarto".
- Una definizione di associazione deve *spiegare l'estensione della storia che deve essere conservata nell'associazione*. Per esempio "Assigned_to collega un letto di ospedale a un paziente. Solo l'assegnazione corrente viene memorizzata. Quando un paziente non viene accettato, quel paziente non viene assegnato a un letto, e un letto può essere libero in un determinato momento nel tempo". Un altro esempio di descrizione della storia in un'associazione è "Places collega un cliente agli ordini da lui effettuati presso la nostra società e collega un ordine al cliente associato. Nella base di dati sono mantenuti solo due anni di ordini, quindi non tutti gli ordini possono partecipare all'associazione".
- Una definizione di associazione deve *spiegare se un'istanza di entità implicata in un'istanza di associazione può trasferire la partecipazione a un'altra istanza di associazione*. Per esempio, "Places collega un cliente agli ordini da lui effettuati presso la nostra società e collega un ordine al cliente associato. Un ordine non è trasferibile a un altro cliente". Un altro esempio è "Categorized_as col-

lega una linea di prodotti ai prodotti venduti in quella categoria e collega un prodotto alla sua linea di prodotti associata. A causa delle modifiche nella struttura dell'azienda e nelle caratteristiche di progettazione dei prodotti, i prodotti possono essere spostati di categoria in un'altra linea di prodotti. Categorized_as tiene una traccia solo della linea di prodotti correnti a cui un prodotto è collegato".

5.5 Un esempio di modellazione E-R: Pine Valley Furniture Company

Lo sviluppo di un diagramma entità-associazione può partire da una di due prospettive (o da entrambe). Secondo una prospettiva top-down, il progettista parte dalle descrizioni di base dell'azienda, e delle sue politiche, e dall'ambiente. Questo approccio è particolarmente adatto per sviluppare un diagramma E-R di alto livello che contenga solo le entità e associazioni principali e un insieme ristretto di attributi (per esempio solo gli identificatori di entità). Secondo un approccio bottom-down, il progettista parte dai colloqui dettagliati con gli utenti e da uno studio approfondito di documenti, schermate e altre fonti di dati. Questo approccio è necessario per sviluppare un diagramma E-R dettagliato e completo di tutti gli attributi.

In questo paragrafo svilupperemo un diagramma E-R di alto livello per Pine Valley Furniture Company, basato ampiamente sul primo di questi approcci (Figura 5.22). Faremo riferimento alla fattura mostrata nel Capitolo 3 per aggiungere qualche dettaglio e per illustrare come i due approcci possono essere combinati.

Da uno studio dei processi aziendali di Pine Valley Furniture Company, abbiamo identificato i seguenti tipi di entità. Viene suggerito anche un identificatore per ciascuna identità, con gli attributi più importanti scelti.

- La società vende diversi tipi di mobili. Questi prodotti sono raggruppati in numerose linee di prodotti. L'identificatore di un prodotto è Product_ID, mentre quello di una linea di prodotti è Product_Line_ID. Facendo riferimento alla fattura (Figura 3.6), identifichiamo alcuni attributi supplementari per il prodotto: Product_Description, Product_Finish e Standard_Price (la fattura nella Figura 3.6 visualizza l'intestazione Unit Price, ma noi abbiamo scelto il nome di attributo Standard Price; in futuro, il prezzo unitario effettivo potrebbe infatti essere scontato, quindi abbiamo deciso di memorizzare il prezzo standard). Un altro attributo della linea di prodotti è Product_Line_Name. Una linea di prodotti può raggruppare qualsiasi numero di prodotti, ma deve contenere almeno un prodotto. Ogni prodotto deve appartenere esattamente a una linea di prodotti.
- I clienti effettuano ordini per i prodotti. L'identificatore per un ordine è Order_ID e un altro attributo è Order_Date. Un cliente può effettuare qualsiasi numero di ordini, ma non può non effettuare un ordine. Ogni ordine viene effettuato esattamente da un cliente. L'identificatore per un cliente è Customer_ID. Altri attributi includono Customer_Name, Customer_Address e Postal_Code.
- Un dato ordine di un cliente può richiedere almeno un prodotto e solo un prodotto per voce d'ordine. Qualsiasi prodotto venduto da Pine Valley Furniture Company può non apparire in qualsiasi voce d'ordine o può apparire in una o più voci d'ordine. Un attributo associato a ciascuna voce d'ordine è Order_Quantity, che è un numero delle unità richieste.
- Pine Valley Furniture Company ha definito alcune zone di vendita per i suoi clienti. Ogni cliente ha un'attività in una o più zone di vendita. L'identificatore per una zona di vendita è Territory_ID e un attributo è Territory Name. Una zona di vendita può avere qualsiasi numero di clienti o può non avere clienti che hanno un'attività.

Figura 5.22

Diagramma E-R per
Pine Valley Furniture
Company.



- Pine Valley Furniture Company ha numerosi agenti di vendita. L'identificatore per un agente di vendita è Salesperson_ID. Altri attributi includono Salesperson_Name, Salesperson_Telephone e Salesperson_Fax. Un agente opera esattamente in una zona di vendita. Ogni zona di vendita è servita da uno o più agenti.

- Ogni prodotto viene assemblato da una quantità specifica di uno o più materiali grezzi. L'identificatore per l'entità del materiale grezzo è `Material_ID`. Altri attributi includono `Unit_of_Measure`, `Material_Name` e `Standard_Cost`. Ogni materiale grezzo è assemblato in uno o più prodotti, utilizzando una quantità specifica di materiale grezzo per ciascun prodotto.
- I materiali grezzi sono forniti dai produttori. L'identificatore di un produttore è `Vendor_ID`. Altri attributi includono `Vendor_Name` e `Vendor_Address`. Ogni materiale grezzo può essere fornito da uno o più produttori. Un produttore può fornire qualsiasi numero di materiali grezzi o può non fornire alcun materiale grezzo a Pine Valley Furniture Company. Un attributo dell'associazione tra produttore e materiale grezzo è `Supply_Unit_Price`.
- Pine Valley Furniture ha stabilito alcuni centri di produzione. L'identificatore per un centro di produzione è `Work_Center_ID`. Un altro attributo è `Work_Center_Location`. Ogni prodotto è prodotto in uno o più centri di produzione. Un centro di produzione può essere utilizzato per produrre qualsiasi numero di prodotti o può non essere utilizzato per produrre alcun prodotto.
- La società ha più di 100 impiegati. L'identificatore per l'impiegato è `Employee_ID`. Altri attributi includono `Employee_Name`, `Address` e `Skill`. Un impiegato può avere più di una competenza. Ogni impiegato può lavorare in uno o più centri di produzione. Un centro di produzione deve avere almeno un impiegato che vi lavora, ma può avere un numero qualsiasi di impiegati.
- Ogni impiegato ha esattamente un solo supervisore, a eccezione del presidente, che non ha supervisori. Un impiegato che è supervisore può essere supervisore di un numero qualsiasi di impiegati, ma non tutti gli impiegati sono supervisori. (*Nota:* questa regola aziendale è effettivamente ambigua, ma la notazione del diagramma E-R introdotta finora maschera questa ambiguità. Non si può dedurre dalla regola o dal diagramma se un impiegato può essere supervisore ma senza impiegati di cui esserlo o se alcuni impiegati non sono supervisori. La notazione che verrà introdotta nel Capitolo 6 può essere utilizzata per chiarire questa distinzione.)

5.6 Elaborazione della base di dati di Pine Valley Furniture Company

Lo scopo del diagramma E-R mostrato nella Figura 5.22 è quello di fornire un progetto concettuale della base di dati di Pine Valley Furniture Company. È importante controllare la qualità di questo progetto attraverso un'interazione frequente con le persone che utilizzeranno la base di dati dopo che è stata implementata. Un tipo importante di controllo sulla qualità è quello che determina se il modello E-R può soddisfare facilmente le esigenze degli utenti su dati e informazioni. Gli impiegati di Pine Valley Furniture Company hanno più esigenze di recupero e report dei dati. In questo paragrafo mostreremo come alcune di queste esigenze informative possono essere soddisfatte grazie all'elaborazione della base di dati mostrata nella Figura 5.22.

Per definire queste interrogazioni utilizziamo il linguaggio SQL di elaborazione delle basi di dati (che verrà spiegato nei Capitoli 9 e 10). Per comprendere appieno queste interrogazioni, sarà necessario capire i concetti introdotti nel Capitolo 7. Tuttavia, alcune interrogazioni semplici in questo capitolo dovrebbero aiutare a comprendere le capacità di una base di dati di rispondere alle domande importanti dell'azienda e di passare poi a capire le interrogazioni SQL nel Capitolo 7 e nei capitoli successivi.

5.6.1 Visualizzare le informazioni sui prodotti

Più utenti diversi possono voler vedere i dati sui prodotti di Pine Valley Furniture Company, per esempio, agenti di vendita, responsabili degli inventari e product manager. Una esigenza specifica riguarda un agente di vendita che vuole rispondere alla richiesta di un cliente di un elenco di prodotti di un determinato tipo. Un esempio di interrogazione è:

Elenca tutti i dettagli dei vari tavoli da computer venduti dall'azienda.

I dati di questa interrogazione sono mantenuti nell'entità **PRODUCT** (Figura 5.22). L'interrogazione analizza questa entità e visualizza tutti gli attributi per i prodotti che contengono la descrizione "Computer Desk".

Il codice SQL per l'interrogazione è il seguente:

```
SELECT *
FROM PRODUCT
WHERE Product_Description LIKE "Computer Desk%";
```

Un tipico output per questa interrogazione è il seguente:

Product_ID	Product_Description	Product_Finish	Standard_Price
3	Computer Desk 48"	Oak	375.00
8	Computer Desk 64"	Pine	450.00

`SELECT * FROM PRODUCT` dice di visualizzare tutti gli attributi delle entità **PRODUCT**. La clausola `WHERE` dice di limitare la ricerca ai soli prodotti la cui descrizione inizia con la frase "Computer Desk".

5.6.2 Visualizzare le informazioni sui clienti

Un'altra esigenza di informazione comune è quella di visualizzare i clienti di Pine Valley Furniture Company. Un tipo specifico di persona che può aver bisogno di questa informazione è il responsabile degli agenti di vendita. Ecco un esempio tipico:

Elenca i dettagli dei clienti nella zona di vendita Nord-est.

I dati di questa interrogazione sono mantenuti nell'entità **CUSTOMER**. Come spiegheremo nel Capitolo 7, l'attributo `Territory_ID` verrà aggiunto all'entità **CUSTOMER** quando il diagramma E-R mostrato nella Figura 5.22 viene tradotto in una base di dati a cui si può accedere attraverso SQL. L'interrogazione analizza quell'entità e visualizza tutti gli attributi per i clienti che si trovano nell'area selezionata.

```
SELECT *
FROM CUSTOMER
WHERE Territory_ID = "Northwest";
```

Un tipico output per questa interrogazione è il seguente:

Customer_ID	Customer_Name	Customer_Address	Territory_ID
5	Value Furniture	349 Rainbow Dr., Seattle WA 97954	Northwest
9	Furniture Gallery	816 Peach Rd., Santa Clara, CA, 96915	Northwest

La spiegazione di questa interrogazione è simile a quella precedente.

5.6.3 Visualizzare le informazioni sui clienti

Le due interrogazioni mostrate in precedenza erano relativamente semplici, e invocavano i dati da una sola tabella per ciascun caso. In una richiesta di informazioni, occorrono spesso dati tratti da più tabelle. Sebbene l'interrogazione fosse piuttosto semplice, abbiamo dovuto cercare in tutta la base di dati per trovare le entità e gli attributi necessari a soddisfare la richiesta.

Per semplificare la scrittura delle interrogazioni e per altre ragioni, molti sistemi di gestione delle basi di dati supportano la creazione di viste ristrette di una base di dati in base alle esigenze di un determinato utente. Per le interrogazioni relative allo stato degli ordini, Pine Valley Furniture Company utilizza la vista di "ordini dei clienti" mostrata nella Figura 5.23a. Questa vista utente permette agli utenti di vedere solo le entità CUSTOMER e ORDER nella base di dati e solo gli attributi di queste entità mostrati nelle figure. Come si vedrà nel Capitolo 7, l'attributo Customer_ID verrà aggiunto all'entità ORDER (come mostrato nella Figura 5.23a). Una tipica interrogazione sullo stato degli ordini è la seguente:

Quanti ordini abbiamo ricevuto da "Value Furniture"?

Il codice SQL per questa interrogazione può essere scritto in molti modi. Qui si è scelto di inserire un'interrogazione in un'altra, chiamata sotto-interrogazione. L'interrogazione avviene in due passaggi. Innanzitutto, la sotto-interrogazione (o interrogazione interna) analizza l'entità CUSTOMER per determinare il Customer_ID del cliente chiamato "Value Furniture" (l'ID del cliente è 5, come mostrato nell'output dell'interrogazione precedente). Quindi, l'interrogazione (o interrogazione esterna) analizza l'entità ORDER e conta le istanze di ordine per quel cliente.

Il codice SQL per questa interrogazione rispetto alla vista utente "ordini dei clienti" è il seguente:

```
SELECT COUNT (Order_ID)
FROM ORDER
WHERE Customer_ID =
  (SELECT Customer_ID
   FROM CUSTOMER
   WHERE Customer_Name = "Value Furniture");
```

Un output tipico di questa interrogazione è il seguente:

```
COUNT(Order_ID)
-----
4
```

5.6.4 Visualizzare le vendite dei prodotti

Gli agenti di vendita, i responsabili delle zone, i product manager, i responsabili della produzione e altri devono conoscere lo stato delle vendite dei prodotti. Un tipo di domanda sulle vendite è quali prodotti registrano un mese di vendite particolarmente eccezionale. Una tipica interrogazione è la seguente:

Quali prodotti hanno avuto una vendita totale che supera i 25.000 \$ nel mese scorso (giugno 2001)?

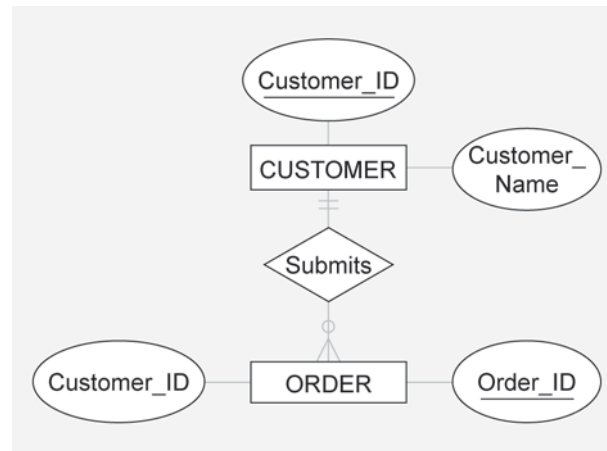
Questa interrogazione può essere scritta utilizzando la vista utente "ordini dei prodotti" mostrata nella Figura 5.23b. I dati che rispondono all'interrogazione si ottengono dalle seguenti fonti.

- Order_Date dall'entità ORDER (per trovare solo gli ordini nel mese desiderato).
- Quantity per ogni prodotto in ogni ordine dall'entità associativa ORDER LINE per un'entità ORDER nel mese desiderato.

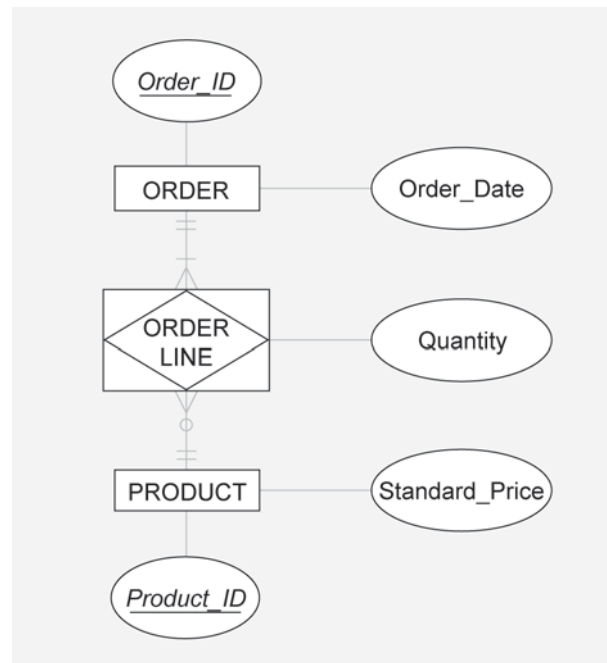
Figura 5.23

Due viste utente per Pine Valley Furniture Company.

(a) Vista utente 1: ordini dei clienti.



(b) Vista utente 2: ordini dei prodotti.



■ Standard_Price per il prodotto ordinato dall'entità **PRODUCT** associata all'entità **ORDER LINE**.

Per ciascun articolo che è stato ordinato nel mese di giugno 2001, l'interrogazione deve moltiplicare Quantity per Standard_Price per ottenere l'importo in dollari delle vendite. L'importo totale per quell'articolo si ottiene quindi sommando tutti gli ordini. I dati vengono visualizzati solo se il totale supera i 25.000 \$.

Il codice SQL per questa interrogazione va oltre lo scopo di questo capitolo, poiché richiede tecniche che verranno introdotte nel Capitolo 7. Introduciamo questa interrogazione ora solo per indicare la capacità di una base di dati come quella mostrata nella Figura 5.22 di trovare le informazioni per la gestione

di dati dettagliati. In molte aziende moderne, gli utenti possono utilizzare un browser Web per ottenere le informazioni come descritto in precedenza. Il codice di programmazione associato a una pagina Web invoca quindi i comandi SQL richiesti per ottenere le informazioni necessarie.

Riepilogo

Questo capitolo ha descritto i fondamenti della modellazione dei dati in un'azienda. Le regole aziendali, derivate dalle politiche, le procedure, gli eventi, le funzioni e altri oggetti aziendali definiscono i vincoli che governano l'azienda e, quindi, il modo in cui i dati sono gestiti e memorizzati. Le regole aziendali sono un modo potente per descrivere le esigenze di un sistema informativo, e soprattutto di una base di dati. La potenza delle regole aziendali emerge dalle regole aziendali che sono il concetto cuore dell'azienda, potendo essere espresse in termini familiari per gli utenti finali, altamente gestibili e applicabili in modo automatizzato, molte delle quali attraverso una base di dati. Le buone regole aziendali sono dichiarative, precise, atomiche, coerenti, espressive, distinte e orientate all'azienda.

Le regole aziendali di base sono i nomi e le definizioni dei dati. Questo capitolo ha spiegato le linee guida per una denominazione e definizione chiara degli oggetti di base di un'azienda. In termini di modellazione di dati concettuale, i nomi e le definizioni devono essere forniti per i tipi di entità, gli attributi e le associazioni. Altre regole aziendali possono definire dei vincoli sugli oggetti di base. I vincoli possono essere rappresentati in un modello di dati e nella documentazione associata.

La notazione della modellazione dei dati attualmente utilizzata più spesso è il modello entità-associazione. Un modello E-R è una rappresentazione logica e dettagliata dei dati di un'azienda. Un modello E-R è solitamente espresso nella forma grafica di un diagramma E-R. Il modello E-R è stato introdotto da Chen nel 1976; ciononostante, attualmente non è ancora disponibile una notazione standard per la modellazione E-R.

I costrutti di base di un modello E-R sono i tipi di entità, le associazioni e gli attributi correlati. Un entità è una persona, un luogo, un oggetto, un evento o un concetto nell'ambiente dell'utente del quale l'azienda desidera mantenere dei dati. Un tipo di entità è una raccolta di entità che condividono proprietà comuni, mentre un'istanza di entità è una singola occorrenza di un tipo di entità. Un tipo di entità forte è un'entità che ha il proprio identificatore e che può esistere senza altre entità. Le entità deboli non hanno il proprio identificatore, sebbene abbiano solitamente un identificatore parziale. Le entità deboli vengono identificate attraverso un'associazione identificativa con il proprio tipo di entità.

Un attributo è una proprietà o caratteristica di un'entità o associazione che è di interesse per un'azienda. Esistono numerosi tipi di attributi. Un attributo semplice non ha parti componenti. Un attributo composto può essere scomposto in parti significative. Per esempio, `Person_Name` può essere scomposto nelle parti `First_Name`, `Middle_Initial` e `Last_Name`. Un attributo multivalore può avere più valori per una singola istanza di un'entità. Per esempio, l'attributo `College_Degree` può avere più valori per un singolo individuo. Un attributo derivato è un attributo i cui valori possono essere calcolati da altri valori di attributo. Per esempio, `Average_Salary` può essere calcolato dai valori di `Salary` per tutti gli impiegati.

Un identificatore è un attributo che identifica in modo univoco le singole istanze di un tipo di entità. Gli identificatori devono essere scelti con attenzione per garantire stabilità e facilità d'uso. Possono essere attributi semplici o composti.

Un tipo di associazione è un'associazione significativa tra due o più tipi di entità. Un'istanza di associazione è un'associazione tra due o più istanze di entità. Il grado di associazione è il numero di tipi di entità che partecipano all'associazione. I tipi di associazioni più comuni sono quella unaria (grado 1), binaria (grado 2) e ternaria (grado 3).

Quando si sviluppano diagrammi E-R si possono incontrare associazioni multi-a-molti (o uno-a-uno) che hanno uno o più attributi associati all'associazione invece di uno solo. Questo suggerisce la conversione dell'associazione in un'entità associativa. Questo tipo di entità associa le istanze di uno o più tipi di entità e contiene attributi specifici di quella associazione. I tipi di entità associativi possono avere il proprio identificatore semplice o può essere assegnato loro un identificatore composto durante la progettazione logica.

Un vincolo di cardinalità è un vincolo che specifica il numero di istanze dell'entità B che possono (o devono) essere associate a ogni istanza dell'entità A. I vincoli di cardinalità specificano solitamente il numero minimo e massimo di istanze. I vincoli possibili sono obbligatorio uno, obbligatorio molti, facoltativo uno e facoltativo molti. Il vincolo di cardinalità minima viene anche chiamato vincolo di partecipazione. Una cardinalità minima pari a zero specifica una partecipazione facoltativa, mentre una pari a 1 specifica una partecipazione obbligatoria.

Ripasso del capitolo

Parole chiave

Associazione binaria	Attributo semplice	Grado	Possessore identificativo
Associazione identificativa	Cardinalità massima	Identificatore	Regola aziendale
Associazione ternaria	Cardinalità minima	Identificatore composto	Termine
Associazione unaria	Diagramma entità-associazione	Istanza di associazione	Tipo di associazione
Attributo	(diagramma E-R)	Istanza di entità	Tipo di entità
Attributo composto	Entità	Marcatura temporale	Tipo di entità debole
Attributo derivato	Entità associativa	Modello entità-associazione	Tipo di entità forte
Attributo multivalore	Fatto	(modello E-R)	Vincolo di cardinalità

Domande di ripasso

1. Definire ciascuno dei seguenti termini chiave:

a. tipo di entità

b. modello entità-associazione

c. istanza di entità

d. attributo

e. tipo di associazione

f. identificatore

g. attributo multivalore

h. entità associativa

i. vincolo di cardinalità

j. entità debole

k. associazione identificativa

l. attributo derivato

m. attributo multivalore

n. regola aziendale
2. Unire i seguenti termini alle rispettive definizioni:

___ attributo composto; ___ entità associativa;

___ associazione unaria; ___ entità debole; ___ attributo;

___ entità; ___ tipo di associazione; ___ vincolo di cardinalità;

___ grado; ___ identificatore; ___ tipo di entità;

___ ternaria; ___ bill-of-materials.
- a. identifica in modo univoco le istanze di entità
- b. fa riferimento a istanze di un singolo tipo di entità
- c. specifica il numero minimo e massimo di istanze
- d. associazione modellata come tipo di entità
- e. associazione tra tipi di entità
- f. raccolta di entità simili
- g. numero di tipi di entità che partecipano a un'associazione
- h. proprietà di un'entità
- i. può essere scomposto in parti componenti
- j. dipende dall'esistenza di un altro tipo di entità
- k. associazione di grado tre
- l. associazione unaria multi-a-molti
- m. persona, luogo, oggetto, concetto, evento
3. Confrontare i seguenti termini:

a. attributo memorizzato; attributo derivato

b. tipo di entità; istanza di entità

c. attributo semplice; attributo composto

d. tipo di entità; tipo di associazione

e. tipo di entità forte; tipo di entità debole

f. grado; cardinalità

4. Elencare tre ragioni per le quali molti progettisti di sistemi ritengono che la modellazione dei dati sia la parte più importante del processo di sviluppo di un sistema.
5. Elencare quattro ragioni per le quali l'approccio delle regole aziendali è considerato come nuovo paradigma per la specifica dei requisiti di un sistema informativo.
6. Spiegare dove si possono trovare le regole aziendali in un'azienda.
7. Citare sei linee guida generali per denominare gli oggetti di base in un modello di dati.
8. Citare quattro criteri per la selezione degli identificatori delle entità.
9. Citare tre condizioni che suggeriscono che il progettista dovrebbe modellare un'associazione come tipo di entità associativa.
10. Elencare i quattro tipi di vincoli di cardinalità e disegnare un esempio di ciascuno.
11. Fornire un esempio di un tipo di entità debole diverso da quelli descritti in questo capitolo. Perché è necessario identificare un'associazione identificativa?
12. Cos'è il grado di un'associazione? Elencare i tre tipi di gradi di associazione descritti nel capitolo e fornire un esempio di ciascuno.
13. Fornire un esempio (diverso da quelli presenti in questo capitolo) per ciascuno dei seguenti attributi:
 - a. attributo derivato
 - b. attributo multivalore
 - c. attributo composto
14. Fornire un esempio (diverso da quelli presenti in questo capitolo) per ciascuna delle seguenti associazioni:
 - a. associazione ternaria
 - b. associazione unaria
15. Fornire un esempio dell'uso di dati veri e propri come attributi di un'entità.
16. Definire una regola che dice quando estrarre un attributo da un tipo di entità e collocarlo in un tipo di entità collegata.

Problemi ed esercizi

1. Confrontare le parole "termine" e "fatto" nella loro applicazione alle regole aziendali. Fornire un esempio di una regola aziendale e indicare quali sono i termini e i fatti nella regola.
2. Nel paragrafo 5.6 è presente un elenco puntato che descrive la Figura 5.22. Per ognuno dei 10 punti nell'elenco, identificare il sottoinsieme della figura descritto da quel punto.
3. Disegnare un diagramma E-R per ciascuna delle situazioni riportate di seguito (se si ritiene che sia necessario stabilire alcuni presupposti supplementari, definirli chiaramente per ogni situazione).
 - a. Una società ha un certo numero di impiegati. Gli attributi di EMPLOYEE includono Employee_ID (identificatore),

Name, Address e Birthdate. La società ha numerosi progetti. Tra gli attributi di PROJECT vi sono Project_ID (identificatore), Project_Name e Start_Date. Ogni impiegato può essere assegnato a un progetto. A un progetto deve essere assegnato almeno un impiegato e può esservi assegnato un numero qualsiasi di impiegati. La percentuale di fatturazione di un impiegato può variare in base al progetto, e la società vuole registrare la percentuale di fatturazione applicabile (Billing_Rate) a ogni impiegato quando questi viene assegnato a un progetto. I nomi degli attributi in questa descrizione seguono le linee guida per la denominazione degli attributi? In caso contrario, si suggeriscano dei nomi più adatti.

- b. Un'università offre un gran numero di corsi. Gli attributi di COURSE includono Course_Number (identificatore), Course_Name e Units. Ogni corso può avere uno o più corsi diversi come pre-requisiti o può non avere pre-requisiti. Analogamente, un determinato corso può essere un pre-requisito per un certo numero di corsi o può non essere un pre-requisito per alcun corso. Fornire una buona definizione di COURSE. Perché questa definizione può essere considerata buona?
- c. Un laboratorio ospita numerosi chimici che lavorano su uno o più progetti. I chimici possono anche utilizzare determinati tipi di attrezzature in ciascun progetto. Gli attributi di CHEMIST sono Employee_ID (identificatore), Name e Phone_No. Gli attributi di PROJECT includono Project_ID (identificatore) e Start_Date. Gli attributi di EQUIPMENT includono Serial_No e Cost. L'azienda desidera registrare l'Assign_Date, ovvero la data in cui un dato elemento di un'attrezzatura è stato assegnato a un determinato chimico che lavora su un progetto specifico. Un chimico deve essere assegnato almeno a un progetto e deve essergli assegnato almeno un elemento di un'attrezzatura. Fornire buone definizioni di tutte le associazioni in questa situazione.
- d. Un corso universitario può avere uno o più seminari programmati o può non aver alcun seminario programmato. Gli attributi di COURSE includono Course_ID, Course_Name e Units. Gli attributi di SECTION sono Section_Number e Semester_ID. Semester_ID è composto da due parti: Semester e Year. Section_Number è un intero (come "1" o "2") che distingue un seminario da un altro nello stesso corso ma che non identifica in modo univoco un seminario. Come si deve modellare SECTION? Perché si è scelto questo modo alternativo e non altri per modellare SECTION?
- e. Un ospedale ospita un gran numero di medici assunti. Gli attributi di PHYSICIAN includono Physician_Name (identificatore) e Specialty. I pazienti vengono accettati nell'ospedale dai medici. Gli attributi di PATIENT sono Patient_ID (identificatore) e Patient_Name. Qualsiasi paziente che viene accettato deve avere esattamente un medico accettante. Un medico può accettare facoltativamente un numero qualsiasi di pazienti. Una volta accettato, un dato paziente deve

essere trattato almeno da un medico. Un medico specifico può trattare un numero qualsiasi di pazienti o può non trattare alcun paziente. Ogni volta che un paziente è trattato da un medico, l'ospedale vuole registrare i dettagli del trattamento (Treatment_Detail). Componenti di Treatment_Detail includono Date, Time e Results. È necessario disegnare più di un'associazione tra medico e paziente? Perché?

4. La Figura 5.24 mostra un pagellino che viene inviato agli studenti alla fine di ciascun semestre. Preparare un diagramma E-R che rispecchi i dati contenuti nel pagellino. Si assuma che ciascun corso sia tenuto da un insegnante.
5. Il tipo di entità STUDENT ha i seguenti attributi: Student_Name, Address, Phone, Age, Activity e No_of_Years. Activity rappresenta alcune attività degli studenti che si svolgono al campus, mentre No_of_Years rappresenta il numero di anni durante i quali lo studente è stato impegnato nell'attività. Un dato studente può impegnarsi in una o più attività. Disegnare un diagramma E-R per questa situazione.
6. Preparare un diagramma E-R per un'agenzia immobiliare che elenca alcune proprietà in vendita. Preparare anche una definizione per ciascun tipo di entità, attributo e associazione nel diagramma. Quelli che seguono descrivono l'azienda.
 - L'agenzia ha alcune filiali in più Stati. Gli attributi delle filiali includono Office_Number (identificatore) e Location.
 - A ogni filiale sono assegnati uno o più impiegati. Gli attributi dell'impiegato includono Employee_Name. Un impiegato deve essere assegnato a una sola filiale.
 - Per ogni filiale c'è sempre un impiegato addetto alla sua gestione. Un impiegato può gestire solo la filiale a cui è assegnato.
 - L'agenzia elenca una proprietà in vendita. Gli attributi della proprietà includono Property_ID (identificatore) e Location. I componenti di Location includono Address, City, State e Zip_Code.

- Ogni unità della proprietà deve essere elencata con una (e solo una) delle filiali. Una filiale può elencare un numero qualsiasi di proprietà o può non elencare alcuna proprietà.
- Ogni proprietà ha uno o più proprietari. Gli attributi dei possessori sono Owner_ID (identificatore) e Owner_Name. Un proprietario può possedere una o più unità della proprietà. un attributo dell'associazione tra proprietà e proprietario è Percent_Owned.

7. Aggiungere la notazione di cardinalità massima e minima a ciascuna delle seguenti figure come opportuno:
 - a. Figura 5.5
 - b. Figura 5.10a
 - c. Figura 5.11b
 - d. Figura 5.12 (tutte le parti)
 - e. Figura 5.13c
 - f. Figura 5.14
8. Dopo aver completato un corso sulla gestione delle basi di dati, verrà richiesto di sviluppare un diagramma E-R preliminare per un'orchestra sinfonica. I tipi di entità da includere sono i seguenti.

- CONCERT SEASON. La stagione durante la quale viene eseguita una serie di concerti. L'identificatore è Opening_Date, che include Month, Date e Year.
- CONCERT. Un'esecuzione data di una o più composizioni. L'identificatore è Concert_Number. Un altro attributo importante è Concert_Date, che è costituito da Month, Date, Year e Time. Ogni concerto ha solitamente più di una data.
- COMPOSITION. Le composizioni da eseguire in ogni concerto. L'identificatore è Composition_ID, che è costituito da Composer_Name e Composition_Name. Un altro attributo è Movement_ID, che è costituito da due parti: Movement_Number e Movement_Name. Molte composizioni, ma non tutte, hanno più movimenti.

Figura 5.24
Pagellino.

MILLENNIUM COLLEGE GRADE REPORT FALL SEMESTER 200X				
NAME:		Emily Williams	ID: 268300458	
CAMPUS ADDRESS:		208 Brooks Hall		
MAJOR:		Information Systems		
COURSE ID	TITLE	INSTRUCTOR NAME	INSTRUCTOR LOCATION	GRADE
IS 350	Database Mgt.	Codd	B104	A
IS 465	System Analysis	Parsons	B317	B

■ **CONDUCTOR.** Persona che dirige una data composizione in un particolare concerto. L'identificatore è `Conductor_ID`. Un altro attributo è `Conductor_Name`.

■ **SOLOIST.** Un artista solista che esegue una data composizione in un particolare concerto. L'identificatore è `Soloist_ID`. Un altro attributo è `Soloist_Name`.

Nelle trattazioni successive di scoprirà quanto segue.

■ Una stagione concertistica programma uno o più concerti. Un particolare concerto è programmato per una sola stagione concertistica.

■ Un concerto include l'esecuzione di una o più composizioni. Una composizione può essere eseguita in uno o più concerti o può non essere eseguita.

■ Per ogni concerto c'è un direttore d'orchestra. Un direttore può dirigere un numero qualsiasi di concerti o può non dirigere alcun concerto.

■ Ogni composizione può richiedere uno o più solisti o può non richiedere alcun solista. Un solista può eseguire una o più composizioni in un dato concerto o può non eseguire alcuna composizione. L'orchestra sinfonica vuole registrare la data in cui un solista ha eseguito più di recente una data composizione (`Date_Last_Performed`).

Disegnare un diagramma E-R per rappresentare quanto scoperto, come sopra descritto. Identificare una regola aziendale in questa descrizione e spiegare come questa viene modellata nel diagramma E-R.

9. Acquisire una vista utente comune come una ricevuta di carta di credito, una bolletta telefonica o qualche altro documento comune. Preparare un diagramma E-R per questo documento.
10. Disegnare un diagramma E-R per un'azienda con la quale si ha familiarità (boy scout, squadra sportiva e così via).
11. Disegnare un diagramma E-R per le seguenti situazioni (Batra, Hoffer e Bostrom, 1998). Sviluppare anche un elenco di parole per i qualificatori e di classi da utilizzare per formare i nomi degli attributi. Spiegare perché si sono scelte le parole della lista. Projects, Inc., è una ditta ingegneristica con circa 500 impiegati. È necessaria una base di dati per tenere traccia di tutti gli impiegati, delle loro competenze, dei progetti assegnati e dei reparti in cui lavorano. Ogni impiegato ha un numero univoco assegnato dalla ditta necessario per memorizzare il suo nome e la sua data di nascita. Se un impiegato è attualmente sposato con un altro impiegato di Projects, Inc., devono essere memorizzati la data del matrimonio e il nome del coniuge; nessun record sul matrimonio è tuttavia richiesto se il coniuge di un impiegato non è anch'esso un impiegato. A ogni impiegato è assegnato un titolo (per esempio, ingegnere, segretaria e così via). Un impiegato svolge un solo tipo di lavoro in un dato momento e la ditta vuole conservare le informazioni sul lavoro corrente dell'impiegato.

La ditta è suddivisa in 11 reparti, ognuno con un nome univoco. Un impiegato può fare rapporto a un solo reparto. Ogni reparto ha un numero di telefono.

Per procurarsi vari tipi di attrezzatura, ogni reparto tratta con più produttori. Un produttore fornisce solitamente l'attrezzatura a molti dipartimenti. Viene richiesto di memorizzare il nome e l'indirizzo di ciascun produttore e la data dell'ultima riunione tra un reparto e un produttore.

Molti impiegati possono lavorare su un unico progetto. Un impiegato può lavorare su molti progetti (per esempio, Southwest Refinery, California Petrochemicals e così via), ma può essere assegnato al massimo a un progetto in una data città. Per ogni città, la ditta è interessata al suo Stato e alla popolazione. Un impiegato può avere più competenze (preparare gli ordini per il materiale, controllare i disegni e così via), ma può utilizzare solo un certo insieme di competenze su un determinato progetto (per esempio, un impiegato MURPHY può preparare richieste per il progetto Southwest Refinery e preparare richieste e controllare i disegni per il progetto California Petrochemicals). Gli impiegati utilizzano ciascuna competenza che possiedono almeno in un progetto. A ogni competenza è assegnato un numero e la ditta vuole memorizzare una breve descrizione di ciascuna competenza. I progetti vengono distinti con un numero di progetto e la ditta vuole memorizzare il costo previsto per ciascun progetto.

12. Disegnare un diagramma E-R per la seguente situazione (definire ogni presupposto che si ritiene utile per poter sviluppare un diagramma completo): Stillwater Antiques acquista e vende pezzi unici di antiquariato di tutti i tipi (per esempio mobili, gioielli, porcellane e abiti). Ogni articolo è identificato in modo univoco da un numero di articolo ed è caratterizzato anche da una descrizione, un prezzo di richiesta, una condizione e da alcuni commenti. Stillwater lavora con numerosi individui singoli, chiamati clienti, che vendono articoli al negozio e che acquistano articoli dal negozio. Alcuni clienti vendono soltanto articoli a Stillwater, altri acquistano soltanto, altri vendono e acquistano. Un cliente è identificato da un numero di cliente ed è anche descritto da un nome e da un indirizzo. Quando Stillwater vende un articolo a un cliente, i proprietari vogliono registrare la commissione pagata, il prezzo di vendita effettivo, le tasse sulla vendita (una tassa pari a zero indica una vendita esente da tasse) e la data di vendita. Quando Stillwater acquista un articolo da un cliente, i proprietari vogliono registrare il costo di acquisto, la data di acquisto e la condizione al momento dell'acquisto.
13. Disegnare un diagramma E-R per la seguente situazione (definire ogni presupposto che si ritiene utile per poter sviluppare un diagramma completo): H. I. Topi School of Business realizza programmi aziendali internazionali in 10 città sparse in tutta Europa. L'istituto ha diplomato i suoi primi 9000 studenti nel 1965. L'istituto tiene traccia del numero di ciascun diplomato, del suo nome da studente, del Paese di nascita, del Paese di cittadinanza attuale, del nome attuale, dell'indirizzo attuale e del nome di ciascuna specializzazione completata dallo studente (ogni studente ha una o più specializzazioni). Per poter mantenere dei legami stretti con i suoi studenti, l'istituto organiz-

za numerosi eventi nel mondo. Gli eventi hanno un titolo, una data, un luogo e un tipo (per esempio, sottoscrizione, pranzo o seminario). L'istituto deve tenere una traccia di quali diplomati hanno partecipato a quali eventi. Per ogni partecipazione di un diplomato a un evento, viene registrato un commento sulle informazioni che i funzionari dell'istituto hanno ottenuto dal diplomato che ha partecipato all'evento. L'istituto tiene anche i contatti con i diplomati per posta, per e-mail, per telefono e per fax. Quando un funzionario dell'istituto sa che incontrerà o parlerà con un diplomato, produce un report che mostra le informazioni apprese negli ultimi due anni su quel diplomato da tutti i contatti e gli eventi a cui questi ha partecipato.

14. Si assuma che in Pine Valley Furniture Company ogni prodotto (descritto da numero di prodotto, descrizione e costo) comprenda almeno tre componenti (descritti da numero di componente, descrizione e unità di misura), che i componenti vengano utilizzati per produrre altri componenti e che anche i materiali grezzi siano considerati componenti. In entrambi i casi di componenti, si dovrà tenere una traccia di quanti componenti entrano a far parte di qualcos'altro. Disegnare un diagramma E-R per questa situazione e inserire nel diagramma le cardinalità massima e minima.
15. I grossisti vendono le scorte e i prezzi cambiano continuamente. Disegnare un diagramma E-R che consideri la natura mutevole del prezzo delle scorte.
16. Ogni semestre, a ogni studente deve essere assegnato un tutor che gli fornisca dei consigli sui requisiti necessari a laurearsi e che lo aiuti a iscriversi alle lezioni. Ogni studente deve iscriversi alle lezioni con l'aiuto di un tutor, ma se il tutor assegnato allo studente non è disponibile, lo studente può registrarsi senza alcun tutor. Si dovrà tenere traccia degli studenti, dei tutor assegnati a ciascuno di essi e del nome del tutor con il quale lo studente è iscritto nel semestre corrente. Rappresentare questa situazione di studenti e tutor in un diagramma E-R.
17. Nella Figura 5.22, si consideri ogni associazione multi-a-molti (M:N). Per ciascuna di esse, si dovrebbe invece rappresentare un'entità associativa. Spiegare. Per le relazioni M:N che dovrebbero essere entità associative, ridisegnare quel segmento del diagramma E-R.

Esercitazioni

1. Intervistare un analista di sistema o di basi di dati per capire come ha scelto i nomi per gli oggetti di base nei modelli di dati. L'azienda in cui questa persona lavora prevede delle linee guida di denominazione? Se sì, descrivere il pattern utilizzato. Se non ci sono linee guida, chiedere al contatto se ha mai avuto problemi a causa di questo.
2. Visitare due piccole aziende locali, una nel settore dei servizi (come una lavanderia, un'autofficina, un veterinario o una libreria) e una nella produzione manifatturiera. Intervistare gli impiegati di queste aziende per cercare di capire le entità, gli

attributi e le associazioni che si incontrano solitamente in questo tipo di aziende. Utilizzare queste informazioni per costruire diagrammi entità-associazione. Quali differenze e somiglianze ci sono tra i diagrammi delle aziende orientate ai servizi e quelle orientate ai prodotti? La tecnica dei diagrammi entità-associazione gestisce bene entrambe le situazioni? Perché sì o no?

3. Chiedere a un analista di sistema o di basi di dati di fornire alcuni esempi di associazioni unarie, binarie e ternarie con cui ha avuto a che fare personalmente presso la propria azienda. Chiedere qual è il tipo di associazione più comune e perché.
4. Chiedere a un analista di sistema o di basi di dati di un'azienda locale di mostrare un diagramma E-R di una delle basi di dati principali dell'azienda. Porre domande per assicurarsi di capire bene cosa significhino ogni entità, attributo o associazione. Questa azienda utilizza la stessa notazione E-R utilizzata in questo testo? In caso contrario, quali altri simboli vengono utilizzati e cosa significano? L'azienda modella le entità associative nel diagramma E-R? In caso contrario, come vengono modellate le entità associative?
5. Per lo stesso diagramma E-R utilizzato nell'esercitazione 4 o per una base di dati diversa nella stessa o in un'altra azienda, identificare gli usi della marcatura temporale o di altri mezzi per modellare i dati indipendenti dal tempo. Perché questo tipo di dati è necessario per coloro che utilizzano la base di dati? Il diagramma E-R sarebbe più semplice se non fosse necessario rappresentare la storia dei valori di attributo?

Bibliografia

- Aranow, E. B., 1989, "Developing Good Data Definitions", *Database Programming & Design* 2 (8), pp. 36–39.
- Batra, D., Hoffer, J. A. e Bostrom, R. B., 1988, "A Comparison of User Performance Between the Relational and Extended Relationship Model in the Discovery Phase of Database Design", *Proceedings of the Ninth International Conference on Information Systems*, Minneapolis, 30 novembre–3 dicembre, pp. 295–306.
- Bruce, T. A., 1992, *Designing Quality Databases with IDEFIX Information Models*, New York, Dorset House.
- Chen, P. P.-S., 1976, "The Entity-Relationship Model—Toward a Unified View of Data", *ACM Transactions on Database Systems*, 1 (marzo), pp. 9–36.
- Elmasri, R. e Navathe, S. B., 1994, *Fundamentals of Database Systems*, seconda edizione, Menlo Park, CA, Benjamin/Cummings.
- Gottesdiener, E., 1999, "Turning Rules into Requirements", *Application Development Trends*, 7 (7), pp. 37–50.
- GUIDE, "GUIDE Business Rules Project", Final Report, revisione 1.2, ottobre, 1997.
- Hoffer, J. A., George, J. F. e Valacich, J. S., 2002, *Modern Systems Analysis and Design*, terza edizione, Upper Saddle River, NJ, Prentice Hall.
- Moriarty, T., 2000, "The Right Tool for the Job", *Intelligent Enterprise* 3 (9), pp. 68, 70–71.

- Plotkin, D., 1999, "Business Rules Everywhere", *Intelligent Enterprise* 2, (4), pp. 37–44.
- Salin, T., "What's in a Name?", *Database Programming & Design* 3 (3), pp. 55–58.
- Song, I-Y., Evans, M. e Park, E. K., "A Comparative Analysis of Entity-Relationship Diagrams", *Journal of Computer & Software Engineering* 3 (4), pp. 427–459.
- Teorey, T. J., Yang, D. e Fry, J. P., 1986, "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model", *Computing Surveys* 18 (giugno), pp. 197–221.
- Von Halle, B., 1997, "Digging for Business Rules", *Database Programming & Design*, 8 (11), pp. 11–13.

Per approfondire

- Batini, C., Ceri, S. e Navathe, B., 1992, *Conceptual Database Design: An Entity-Relationship Approach*, Menlo Park, CA, Benjamin/Cummings.
- Keuffel, W., 1996, "Battle of the Modeling Techniques", *DBMS* 9, (agosto), pp. 83, 84, 86, 97.
- Moody, D., 1996, "The Seven Habitus of Highly Effective Data Modelers", *Database Programming & Design* 9 (ottobre), pp. 57, 58, 60–62, 64.
- Teorey, T., 1999, *Database Modeling & Design*, terza edizione, San Francisco, CA, Morgan Kaufman.
- Tillman, G., 1994, "Should You Model Derived Data?", *DBMS* 8 (agosto), pp. 70, 72, 74, 76, 80–82, 87.

Project Case



MOUNTAIN VIEW COMMUNITY HOSPITAL

Descrizione del progetto

Dopo aver completato un corso sulla gestione delle basi di dati, si immagini di venire assunti come tirocinanti estivi presso il Mountain View Community Hospital. Il primo compito da svolgere è quello di lavorare come parte di un team di tre persone che deve sviluppare un diagramma E-R di alto livello per l'ospedale. Si inizierà quindi effettuando alcune interviste ad alcuni amministratori dell'ospedale e del personale per identificare i tipi di entità chiave per l'ospedale. Dopo breve tempo, il team ha identificato i seguenti tipi di entità.

- **Care center:** un'unità di trattamento all'interno dell'ospedale. Alcuni esempi di queste unità sono la maternità e il pronto soccorso. Ogni unità di cura ha un ID (identificatore) e un nome.
- **Patient:** una persona che viene accettata nell'ospedale o che viene iscritta a un programma per pazienti esterni. Ogni paziente ha un numero (identificatore) e un nome.
- **Physician:** un membro del personale medico dell'ospedale che può accettare i pazienti nell'ospedale e che può effettuare trattamenti medici. Ogni medico ha un ID (identificatore) e un nome.
- **Bed:** un posto letto nell'ospedale che può essere assegnato a un paziente accettato nell'ospedale. Ogni posto letto ha un numero (identificatore), un numero di stanza e un ID di unità di cura.
- **Item:** qualsiasi articolo medico o chirurgico che può essere utilizzato per trattare un paziente. Ogni articolo ha un numero (identificatore), una descrizione e un costo unitario.
- **Employee:** qualsiasi persona che fa parte del personale dell'ospedale. Ogni impiegato ha un numero e un nome.
- **Treatment:** qualsiasi analisi o test effettuati da un medico su un paziente. Ogni trattamento ha un ID costituito da due parti: il numero del trattamento e il suo nome.

Il team ha quindi registrato le seguenti informazioni sulle associazioni.

- Ogni impiegato dell'ospedale viene assegnato a lavorare in una o più unità di cura. Ogni unità di cura ha almeno un impiegato e può avere un numero qualsiasi di impiegati. L'ospedale registra il numero di ore per settimana durante le quali un dato impiegato lavora in un'unità di cura specifica.

- Ogni unità di cura ha esattamente un impiegato che è capo-infermiere per quell'unità.
- Un dato paziente può essere o non essere assegnato a un posto letto (poiché alcuni pazienti possono essere esterni). Un posto letto può essere o non essere assegnato a un paziente.
- Un paziente fa riferimento all'ospedale esattamente attraverso un medico. Un medico può fare riferimento a un numero qualsiasi di pazienti o a nessun paziente.
- I medici possono effettuare un numero qualsiasi di trattamenti su un numero qualsiasi di pazienti o possono non effettuare alcun trattamento. Un paziente può avere trattamenti effettuati da un numero qualsiasi di medici. Per ogni trattamento su un dato paziente da un medico, l'ospedale registra le seguenti informazioni: data del trattamento, ora del trattamento e risultati.
- Un paziente può consumare facoltativamente un numero qualsiasi di articoli. Un dato articolo può essere consumato da uno o più pazienti o può non essere consumato. Per ciascun articolo consumato da un paziente, l'ospedale registra le seguenti informazioni: data, ora, quantità e costo totale (che può essere calcolato moltiplicando la quantità per il costo unitario).

Domande sul progetto

1. Perché Mountain View Community Hospital vuole utilizzare la modellazione entità-associazione per capire le proprie esigenze sui dati? In quali altri modi l'ospedale potrebbe voler modellare le proprie esigenze informative?
2. Appaiono delle entità deboli nella descrizione delle esigenze sui dati di Mountain View Community Hospital? Se sì, quali sono?
3. Nello sviluppo di un diagramma E-R per Mountain View Community Hospital, qual è il significato della terza entità dell'elenco precedente, che stabilisce che alcuni pazienti sono assegnati a un posto letto, ma che i pazienti esterni non sono assegnati ad alcun posto letto?
4. Mountain View Community Hospital è anch'esso un tipo di entità in un modello di dati che ne rappresenta le esigenze informative?

Esercizi sul progetto

1. Studiare la descrizione del progetto con molta attenzione. Quali altre domande si potrebbero porre per capire meglio le esigenze informative dell'ospedale?
 2. Sviluppare un diagramma E-R per Mountain View Community Hospital. Definire i presupposti utilizzati per sviluppare il diagramma.
 3. La descrizione del progetto descrive un tipo di entità chiamata Item. Una volta risposto all'esercizio sul progetto 2, questo tipo di entità è in grado di rappresentare gli apparecchi TV nelle stanze come elementi fatturabili ai pazienti? Perché?
 4. Si supponga che l'attributo del numero del letto sia di tipo composto, costituito dall'ID dell'unità di cura, dal numero della stanza e dal numero del singolo letto. Ridisegnare quelle parti della risposta all'esercizio sul progetto 2 che dovrebbero essere modificate per gestire questo attributo composto.
 5. Si consideri il nuovo diagramma E-R per Mountain View Community Hospital dopo le modifiche all'esercizio sul progetto 2 per gestire la situazione emersa nell'esercizio sul progetto 4. Si presuma ora che un'unità di cura abbia molte stanze e che ogni stanza possa contenere articoli che vengono fatturati ai pazienti assegnati a quella stanza. Ridisegnare il diagramma E-R per adeguarsi a questi nuovi presupposti.
 6. La risposta data all'esercizio sul progetto 2 permette che più di un medico effettui contemporaneamente un trattamento su un paziente? Se no, ridisegnare la risposta all'esercizio sul progetto 2 per adeguarsi a questa nuova situazione. Aggiungere eventuali nuovi presupposti che si considerano necessari per rappresentare questa situazione.
 7. La risposta data all'esercizio sul progetto 2 permette che lo stesso trattamento venga effettuato più di una volta sullo stesso paziente dallo stesso medico? Se no, ridisegnare la risposta all'esercizio sul progetto 2 per adeguarsi a questa nuova situazione. Aggiungere eventuali nuovi presupposti che si considerano necessari per rappresentare questa situazione.
-



CAPITOLO

6

Il modello E-R esteso e le regole aziendali

Al termine di questo capitolo sarete in grado di fare quanto segue.

1. Definire i seguenti termini chiave: modello entità-associazione esteso (EER), sottotipo, supertipo, ereditarietà di attributo, generalizzazione, specializzazione, regola della specializzazione totale, regola della separazione, regola della sovrapposizione, discriminante di sottotipo, gerarchia di specializzazione, cluster di entità, asserzione strutturale, asserzione di azione e fatto derivato.
2. Riconoscere quando utilizzare le associazioni di specializzazione nella modellazione dei dati.
3. Utilizzare la specializzazione e la generalizzazione come tecniche per definire le associazioni supertipo/sottotipo.
4. Specificare i vincoli di completezza e di separazione nelle associazioni supertipo/sottotipo.
5. Sviluppare una gerarchia supertipo/sottotipo per una situazione aziendale realistica.
6. Sviluppare un cluster di entità per semplificare la presentazione di un diagramma E-R.
7. Denominare le varie categorie di regole aziendali.
8. Definire un vincolo operativo semplice utilizzando un modello grafico o un'istruzione in un linguaggio naturale.

Introduzione

Il modello E-R di base descritto nel Capitolo 5 è stato introdotto per la prima volta a metà degli anni Settanta. Per molto tempo si è rivelato adatto per la modellazione della maggior parte dei problemi aziendali comuni e ha avuto un'ampia diffusione. Da allora, tuttavia, l'ambiente aziendale ha subito un cambiamento drastico e anche le relazioni tra aziende sono diventate sempre più complesse. Per esempio, le aziende devono essere preparate a segmentare il proprio mercato e a personalizzare i propri prodotti, cosa che implica l'interrogazione delle basi di dati con domande sempre più articolate.

Per affrontare al meglio questi cambiamenti, i ricercatori hanno continuato a migliorare il modello E-R di modo che potesse rappresentare più accuratamente i dati complessi incontrati nell'ambiente aziendale di oggi. Il termine **modello entità-associazione esteso (EER)** viene utilizzato per identificare il modello che è nato dall'estensione del modello E-R originario con questi nuovi costrutti di modellazione.

Il nuovo costrutto di modellazione più importante incorporato nel modello EER è l'associazione supertipo/sottotipo. Questo costrutto permette di modellare un tipo di entità generica (chiamata *supertipo*) e di suddividerla in più tipi di entità specializzati (chiamati *sottotipi*). Così, per esempio, il tipo di entità CAR può essere modellato come supertipo con i sottotipi SEDAN, SPORTS CAR, COUPE e così via. Ogni sottotipo eredita gli attributi del suo supertipo e contemporaneamente può avere i propri attributi speciali. L'aggiunta di una nuova notazione per la modellazione dell'associazione supertipo/sottotipo ha notevolmente migliorato la flessibilità del modello E-R di base.

I diagrammi E-R, e soprattutto quelli EER, possono diventare grandi e complessi e richiedere più pagine, o font molto piccoli, per la visualizzazione. Alcune basi di dati commerciali includono centinaia di entità. Molti utenti e manager che specificano le esigenze per una base di dati non necessitano di vedere tutte le entità, le associazioni e gli attributi per capire la parte della base di dati a cui sono interessati. L'accorpamento delle entità è un modo per convertire una parte di un modello entità-associazione in una vista a macro-livello degli stessi dati. L'accorpamento delle entità è una tecnica di scomposizione gerarchica che può rendere i diagrammi E-R più facili da leggere. Raggruppando le entità e le associazioni, si può delineare un diagramma E-R che permette di concentrare l'attenzione sui dettagli del modello che più contano in una determinata attività di modellazione dei dati.

I diagrammi E-R estesi sono utilizzati per rappresentare regole aziendali importanti come i vincoli nelle associazioni sottotipo/supertipo. La maggior parte delle aziende utilizza tuttavia più regole aziendali per guidare il comportamento, e molte di queste non possono essere espresse con diagrammi E-R di base né con diagrammi EER. Attualmente sono in corso ricerche promettenti nella direzione dello sviluppo di nuovi modi di espressione delle regole aziendali, che dovrebbero permettere agli utenti di definire, con un po' di formazione, le proprie regole per il sistema. Queste regole diventano quindi automaticamente dei vincoli che sono applicati dal sistema di gestione della base di dati e che sono utilizzati per gestire la base di dati in modo coerente e valido. L'approccio delle regole aziendali consente alle aziende di rispondere al cambiamento delle condizioni aziendali più rapidamente rispetto agli approcci precedenti ai sistemi di sviluppo.

6.1 Rappresentare supertipi e sottotipi

Nel Capitolo 5 è stato spiegato che un tipo di entità è una raccolta di entità che condividono proprietà e caratteristiche comuni. Sebbene le istanze di entità che compongono un tipo di entità siano simili, non ci si può aspettare che siano identiche. Una delle sfide principali nella modellazione dei dati è quella di riconoscere e rappresentare senza ambiguità le entità che sono quasi simili, ossia i tipi di entità che condividono proprietà comuni ma che hanno anche alcune proprietà distintive che sono di interesse per l'azienda.

Per questa ragione, il modello E-R è stato esteso fino a includere le associazioni supertipo/sottotipo. Un **sottotipo** è un sottoraggruppamento delle entità in un tipo di entità significativo per l'azienda. Per esempio, STUDENT è un tipo di entità in un'università. I due sottotipi di STUDENT sono GRADUATE STUDENT e UNDERGRAD STUDENT. In questo esempio, STUDENT è il supertipo. Un **supertipo** è un tipo di entità generico che ha un'associazione con uno o più sottotipi.

Nella creazione di diagrammi E-R affrontata finora, i supertipi e i sottotipi sono stati nascosti. Per esempio, si consideri nuovamente la Figura 5.22, che è il diagramma E-R per Pine Valley Furniture Company. Si noti che tutti i clienti hanno un'attività in una zona di vendita. Perché? Una ragione possibile è che ci sono due tipi di clienti, quelli nazionali e quelli di zona, e solo quelli di zona sono assegnati a una zona di vendita. Nella Figura 5.22, la ragione della cardinalità facoltativa accanto a CUSTOMER nell'associazione Does_business viene nascosta. Il disegno esplicito di un supertipo di entità di cliente e

di numerosi sottotipi di entità permette di rendere il diagramma E-R più significativo. Successivamente nel capitolo mostreremo un diagramma E-R rivisto per Pine Valley Furniture, che illustrerà numerose notazioni EER che rendono più espliciti gli aspetti ambigui della Figura 5.22.

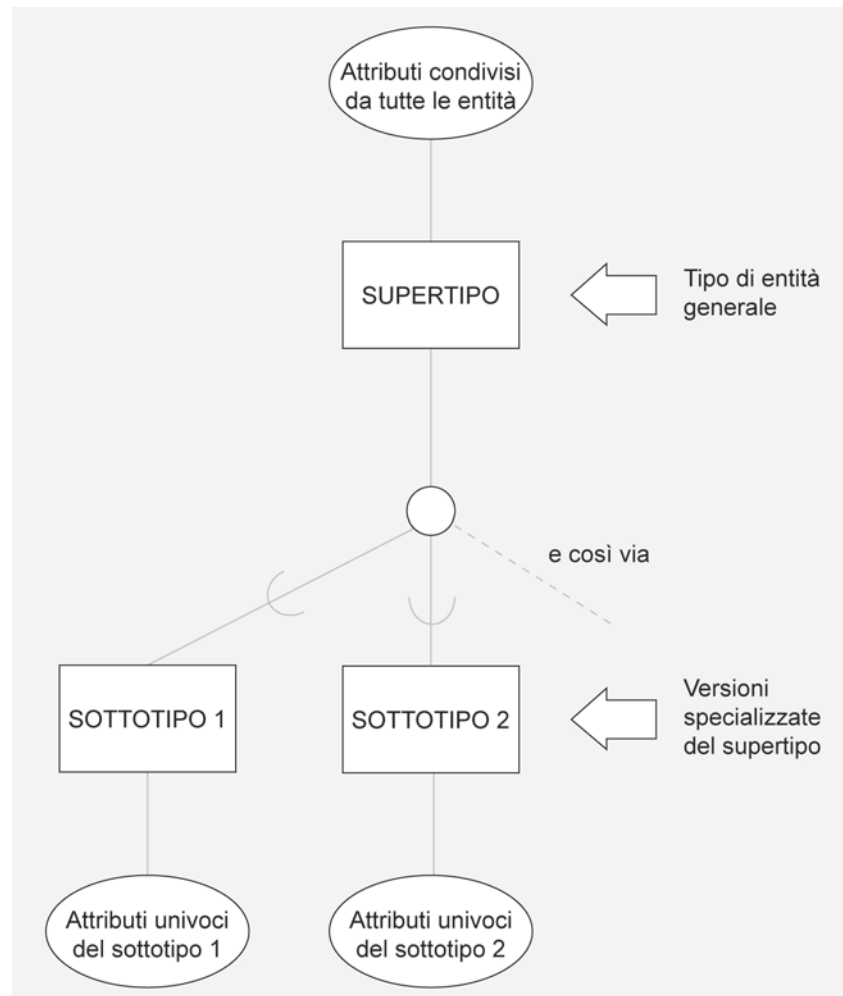
6.1.1 Concetti di base e notazione

La notazione di base utilizzata per le associazioni supertipo/sottotipo è mostrata nella Figura 6.1. Il supertipo è connesso con una linea a un cerchio, che a sua volta è connesso a ciascun sottotipo che è stato definito. Il simbolo a forma di U sulle linee che connettono un sottotipo al cerchio indica che il sottotipo è un sottoinsieme del supertipo e indica anche la direzione dell'associazione supertipo/sottotipo.

Gli attributi che sono condivisi da tutte le entità (compreso l'identificatore) sono associati al supertipo. Gli attributi che sono univoci per un particolare tipo di sottotipo sono associati a quel sottotipo. A questa notazione verranno aggiunti altri componenti per fornire un significato più articolato alle associazioni supertipo/sottotipo man mano che si procede nel capitolo.

Figura 6.1

La notazione di base delle associazioni supertipo/sottotipo.



Un esempio

Utilizziamo un esempio semplice per illustrare le associazioni supertipo/sottotipo. Si supponga che un'azienda abbia tre tipi di impiegati di base: impiegati a ore, impiegati salariati e consulenti a contratto. Alcuni degli attributi importanti per ciascuno di questi tipi di impiegati sono i seguenti.

- *Impiegati a ore*: Employee_Number, Employee_Name, Address, Data_Hired, Hourly_Rate.
- *Impiegati salariati*: Employee_Number, Employee_Name, Address, Data_Hired, Annual_Salary, Stock_Option.
- *Consulenti a contratto*: Employee_Number, Employee_Name, Address, Data_Hired, Contract_Number, Billing_Rate.

Si noti che tutti i tipi di impiegati hanno numerosi attributi in comune: Employee_Number, Employee_Name, Address e Data_Hired. Inoltre, ogni tipo ha uno o più attributi distinti dagli attributi di altri tipi (per esempio, Hourly_Rate è univoco per gli impiegati a ore). Se si sta sviluppando un modello di dati concettuale in questa situazione, si possono considerare tre opzioni.

1. Definire un singolo tipo di entità chiamato EMPLOYEE. Sebbene sia concettualmente semplice, questo approccio ha lo svantaggio che EMPLOYEE dovrebbe contenere tutti gli attributi dei tre tipi di impiegati. Per esempio, per un'istanza di un impiegato a ore, attributi come Annual_Salary e Contract_Number non potrebbero essere applicati e sarebbero nulli o rimarrebbero inutilizzati. Se utilizzati in un ambiente di sviluppo, i programmi che si servono di questo tipo di entità dovrebbero essere necessariamente complessi per poter gestire i numerosi cambiamenti.
2. Definire un'entità separata per ciascun tipo delle tre entità. Questo approccio non riuscirebbe a sfruttare le proprietà comuni degli impiegati e gli utenti dovrebbero fare attenzione a selezionare il tipo di entità corretto quando utilizzano il sistema.
3. Definire un supertipo chiamato EMPLOYEE con sottotipi per HOURLY EMPLOYEE, SALARIED EMPLOYEE e CONSULTANT. Questo approccio sfrutta le proprietà comuni di tutti gli impiegati e riconosce le proprietà distinte di ciascun tipo.

La Figura 6.2 mostra una rappresentazione del supertipo EMPLOYEE con i suoi tre sottotipi utilizzando la notazione E-R estesa. Gli attributi condivisi da tutti gli impiegati sono associati al tipo di entità EMPLOYEE. Gli attributi che sono specifici di ciascun sottotipo sono inclusi con il solo sottotipo.

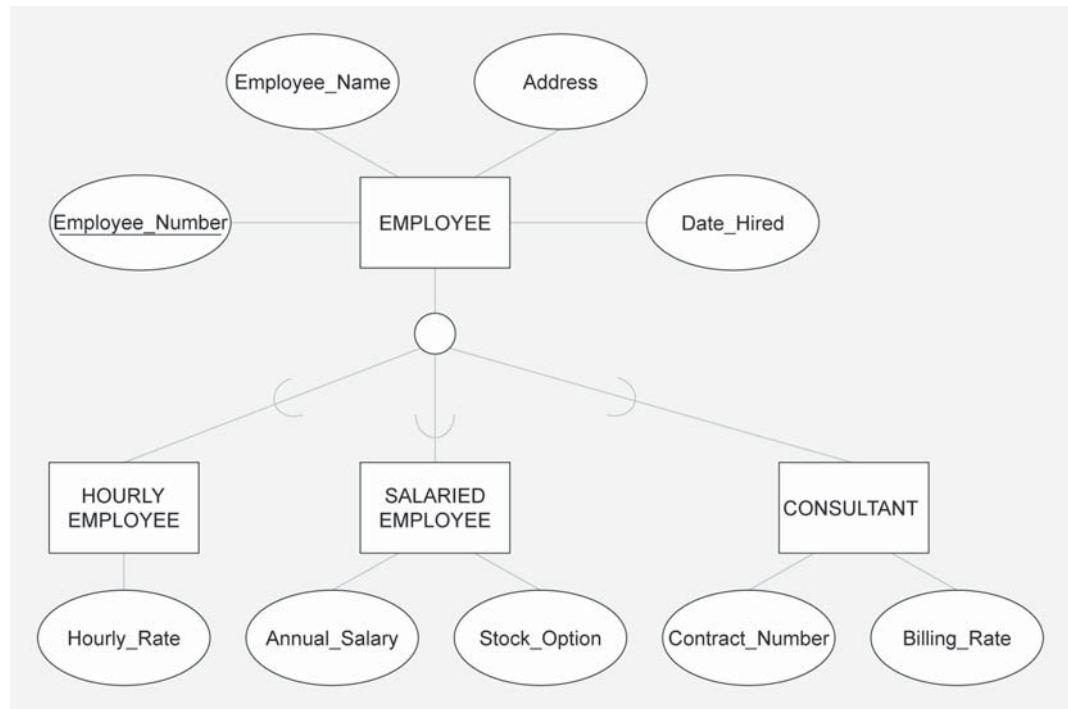
Ereditarietà degli attributi

Un sottotipo è un tipo di entità di per sé. Un'istanza di sottotipo rappresenta la *stessa* istanza di entità del supertipo. Per esempio, se 'Therese Jones' è un'occorrenza del sottotipo CONSULTANT, allora la stessa persona è necessariamente un'occorrenza del supertipo EMPLOYEE. Di conseguenza, un'entità in un sottotipo deve possedere non solo i valori per i propri attributi, ma anche i valori per i suoi attributi come membro del supertipo.

L'ereditarietà degli attributi è la proprietà in base alla quale le entità del sottotipo ereditano i valori di tutti gli attributi del supertipo. Questa proprietà importante rende non necessario includere attributi di supertipo in modo ridondante con i sottotipi. Per esempio, Employee_Name è un attributo di EMPLOYEE (Figura 6.2) ma non dei sottotipi di EMPLOYEE. Il fatto che il nome dell'impiegato sia 'Therese Jones', quindi, viene ereditato dal supertipo EMPLOYEE. Tuttavia, il Billing_Rate per questo stesso impiegato è un attributo del sottotipo CONSULTANT.

Abbiamo stabilito che un membro di un sottotipo *deve* essere un membro del supertipo. È vero anche il contrario (ossia, un membro del supertipo è anche membro di uno o più sottotipi)? Può essere vero o meno a seconda della situazione aziendale. Vedremo le varie possibilità successivamente nel capitolo.

Figura 6.2
Il supertipo Employee
con tre sottotipi.



Quando utilizzare le associazioni supertipo/sottotipo

Se utilizzare o meno le associazioni supertipo/sottotipo è una decisione che deve essere presa dal modellatore dei dati a seconda della situazione. Si può ipotizzare l'uso dei sottotipi quando è presente una delle seguenti condizioni (o entrambe).

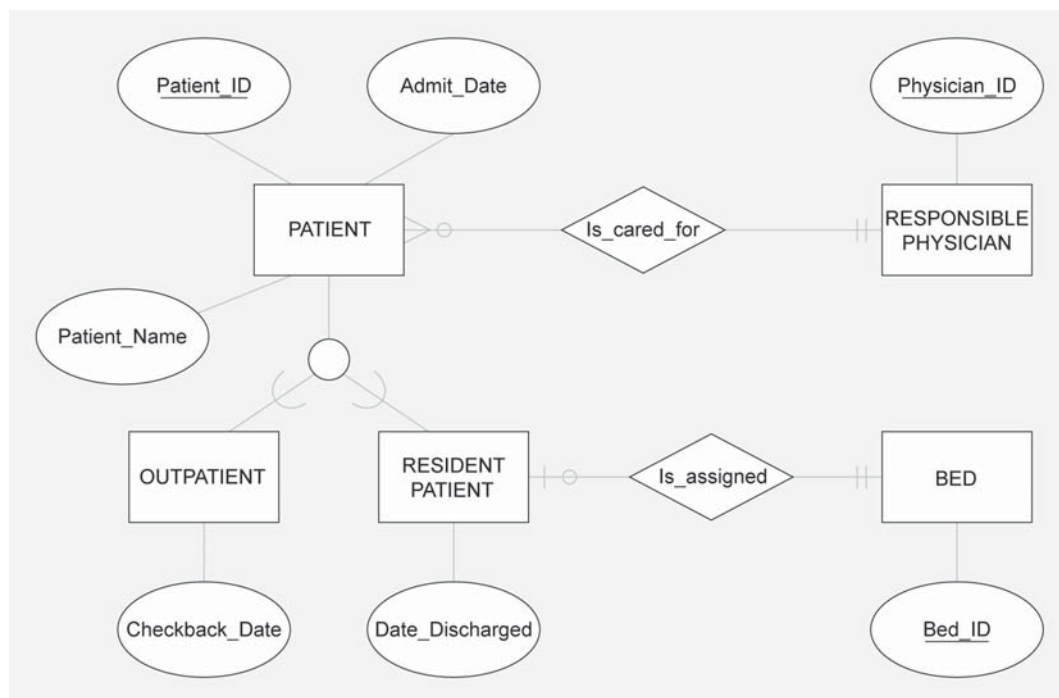
1. Ci sono attributi che si applicano ad alcune (ma non a tutte) le istanze di un tipo di entità. Si veda per esempio il tipo di entità EMPLOYEE nella Figura 6.2.
2. Le istanze di un sottotipo partecipano a un'associazione univoca per quel sottotipo.

La Figura 6.3 è un esempio dell'uso delle associazioni di sottotipo che illustra entrambe queste situazioni. Il tipo di entità dell'ospedale PATIENT ha due sottotipi: OUTPATIENT e RESIDENT PATIENT (la chiave primaria è Patient_ID). Tutti i pazienti hanno un attributo Admit_Date, oltre a un Patient_Name. Ogni paziente viene inoltre curato da un RESPONSIBLE PHYSICIAN che sviluppa un piano di trattamento per il paziente.

Ogni sottotipo ha un attributo che è univoco per quel sottotipo. I pazienti esterni hanno una Check-back_Date, mentre i pazienti interni hanno una Date_Discharged. I pazienti interni hanno inoltre un'associazione univoca che assegna ciascun paziente a un letto (si noti che è un'associazione obbligatoria). Ogni letto può essere o non essere assegnato a un paziente.

In precedenza abbiamo discusso la proprietà dell'ereditarietà degli attributi. Ogni paziente esterno e ogni paziente interno eredita gli attributi del supertipo genitore PATIENT: Patient_ID, Patient_Name e Admit_Date. La Figura 6.3 illustra anche il principio di ereditarietà dell'associazione. OUTPATIENT e RESIDENT PATIENT sono istanze di PATIENT; quindi, ogni paziente Is_cared_for da un RESPONSIBLE PHYSICIAN.

Figura 6.3
Associazione
supertipo/sottotipo
in un ospedale.



6.1.2 Rappresentare la specializzazione e la generalizzazione

Finora abbiamo descritto e illustrato i principi di base delle associazioni supertipo/sottotipo, comprese le caratteristiche dei sottotipi “buoni”. Quando si sviluppano modelli di dati nel mondo reale, però, come è possibile riconoscere le opportunità giuste per sfruttare queste associazioni? Esistono due modelli, la specializzazione e la generalizzazione, che servono come modelli mentali nello sviluppo di associazioni supertipo/sottotipo.

Generalizzazione

Un aspetto unico dell'intelligenza umana è la capacità e la propensione a classificare gli oggetti e le esperienze e a generalizzarne le proprietà. Nella modellazione dei dati, la **generalizzazione** è il processo di definizione di un tipo di entità di livello più generico da un insieme di tipi di entità più specializzati. È quindi un processo bottom-up.

Un esempio di generalizzazione è mostrata nella Figura 6.4. Nella Figura 6.4a, sono stati definiti tre tipi di entità: CAR, TRUCK e MOTORCYCLE. In questa fase, il modellatore dei dati intende rappresentarli separatamente in un diagramma E-R. Osservando con più attenzione, si nota però che i tre tipi di entità hanno alcuni attributi in comune: Vehicle_ID (identificatore), Vehicle_Name (con i componenti Make e Model), Price ed Engine_Displacement. Questo fatto, rafforzato dalla presenza di un identificatore comune, suggerisce che ciascuna delle tre entità sia effettivamente una versione di un tipo di entità generale.

Questo tipo di entità generale (VEHICLE) con il risultato delle associazioni supertipo/sottotipo viene mostrato nella Figura 5.4b. L'entità CAR ha l'attributo specifico No_of_Passengers, mentre TRUCK

ha due attributi specifici: Capacity e Cab_Type. La generalizzazione ha quindi permesso di raggruppare i tipi di entità con i loro attributi comuni e allo stesso tempo di conservare attributi specifici che sono peculiari per ciascun sottotipo.

Si noti che il tipo di entità MOTORCYCLE non è incluso nell'associazione. Si tratta di un'omissione? No. L'entità non è stata inclusa volontariamente perché non soddisfa le condizioni per un sottotipo discusso in precedenza. Confrontando le parti a e b della Figura 6.4, si può notare che i soli attributi di MOTORCYCLE sono quelli comuni a tutti i veicoli; non ci sono attributi specifici per le motociclette. Non occorre quindi creare un sottotipo MOTORCYCLE.

Il fatto che non esista un sottotipo MOTORCYCLE suggerisce che deve essere possibile avere un'istanza di VEHICLE che non è membro di alcuno dei suoi sottotipi. Tratteremo questo tipo di vincolo nel paragrafo sulla specifica dei vincoli.

Specializzazione

Come abbiamo visto, la generalizzazione è un processo bottom-up. La specializzazione è un processo top-down, l'esatto inverso della generalizzazione. Si supponga di aver definito un tipo di entità con i suoi attributi. La specializzazione è il processo di definizione di uno o più sottotipi del supertipo e di formazione di associazioni supertipo/sottotipo. Ogni sottotipo viene formato in base ad alcune caratteristiche distintive come attributi o associazioni specifici del sottotipo.

Un esempio di specializzazione è mostrato nella Figura 6.5. La Figura 6.5a mostra un tipo di entità chiamata PART con i suoi numerosi attributi. L'identificatore è Part_No e altri attributi comprendono Description, Unit_Price, Location, Qty_on_Hand, Routing_Number e Supplier (l'ultimo attributo è multivalore, poiché ci può essere più di un fornitore con il prezzo unitario per una parte).

Parlando con gli utenti, si scopre che ci sono due fonti possibili per le parti: alcune parti sono prodotte internamente, mentre altre vengono acquistate da fornitori esterni. Si scopre inoltre che alcune parti provengono da entrambe le fonti. In questo caso la scelta dipende da fattori come la capacità di produzione, il prezzo unitario delle parti e così via.

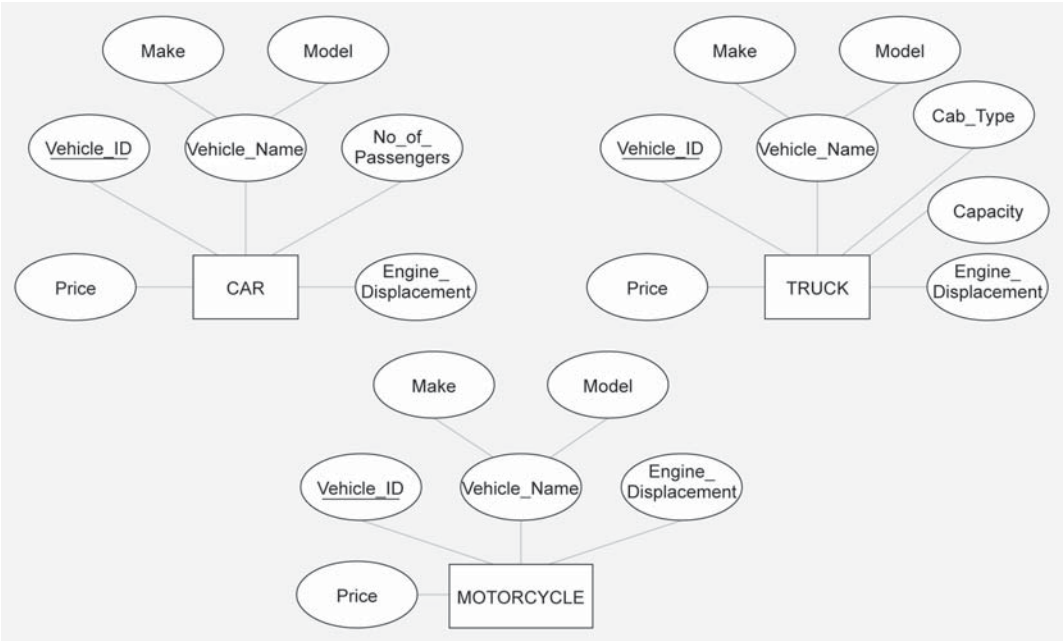
Alcuni degli attributi nella Figura 6.4a si applicano a tutte le parti, a prescindere dalla fonte. Altri dipendono invece dalla fonte. Per esempio, Routing_Number si applica solo alle parti prodotte, mentre Supplier_ID e Unit_Price si applicano solo alle parti acquistate. Questi fattori suggeriscono che l'entità PART debba essere specializzata definendo i sottotipi MANUFACTURED PART e PURCHASED PART (Figura 6.5b).

Nella Figura 6.5b, Routing_Number è associato a MANUFACTURED PART. Il modellatore dei dati ha pianificato inizialmente di associare Supplier_ID e Unit_Price a PURCHASED PART. In discussioni successive con gli utenti, emerge invece che si crea una nuova associazione tra PURCHASED PART e SUPPLIER. L'associazione (chiamata Suppliers nella Figura 6.5b) consente agli utenti di associare più facilmente le parti acquistate ai loro fornitori. Si noti che l'attributo Unit_Price è ora associato alla relazione Suppliers, così che il prezzo unitario di una parte possa variare da un fornitore all'altro. In questo esempio, la specializzazione ha permesso la rappresentazione preferita del contesto problematico.

Combinare specializzazione e generalizzazione

La specializzazione e la generalizzazione sono entrambe tecniche valide per lo sviluppo di associazioni supertipo/sottotipo. La tecnica da utilizzare in un determinato momento dipende da numerosi fattori, come la natura del contesto problematico, degli sforzi di modellazione precedenti e dalle preferenze personali. Si dovrebbe quindi essere preparati a utilizzare entrambi gli approcci e a muoversi avanti e indietro tra i due in base ai fattori appena citati.

Figura 6.4
Esempio di
generalizzazione.
(a) Tre tipi di entità:
CAR, TRUCK e
MOTORCYCLE.



(b) Generalizzazione
sul supertipo VEHICLE.

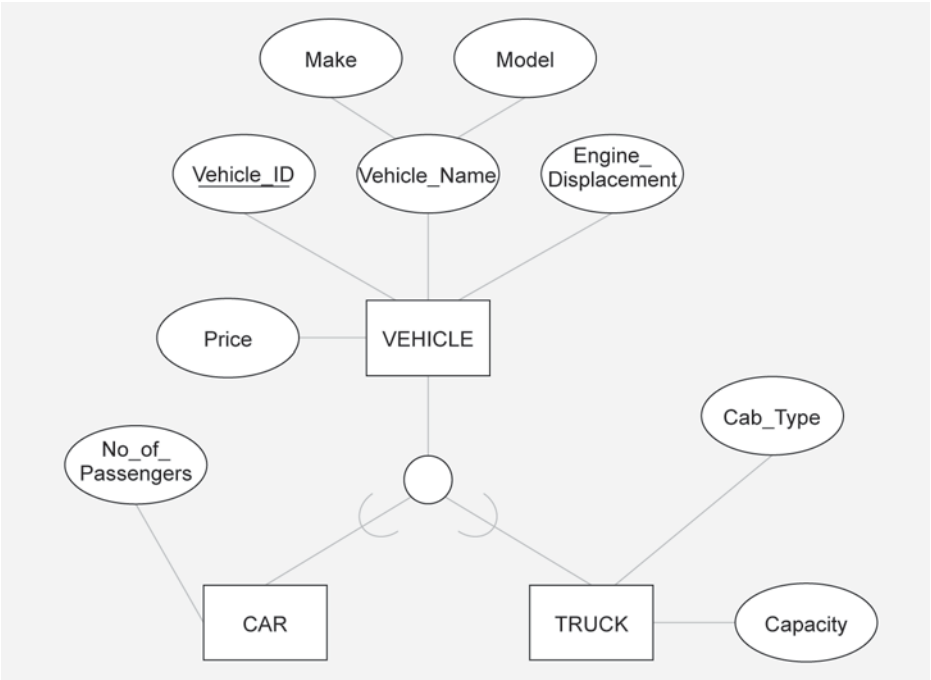
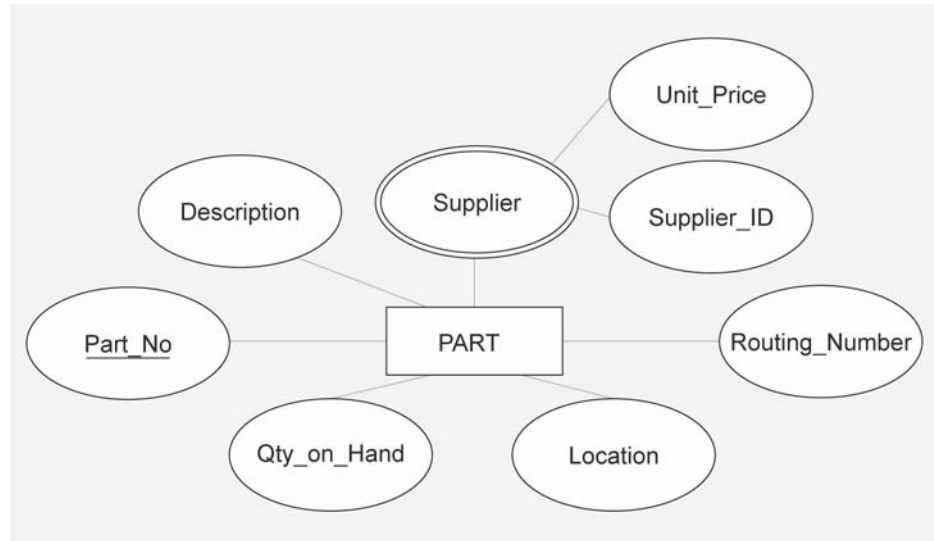
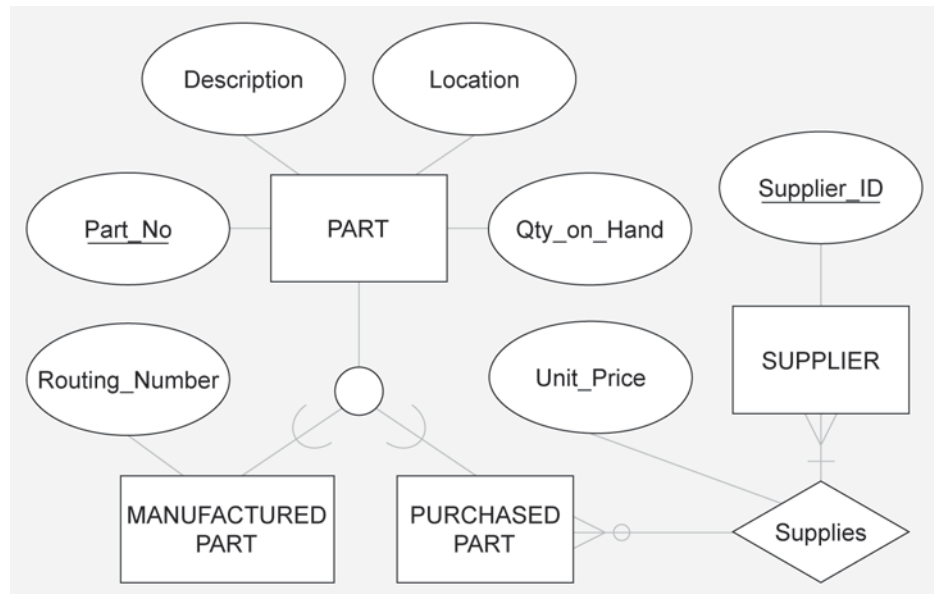


Figura 6.5

Esempio di specializzazione.
(a) Tipo di entità PART.



(b) Specializzazione su
MANUFACTURED PART e
PURCHASED PART.



6.2 Specificare i vincoli nelle associazioni supertipo/sottotipo

Finora abbiamo trattato i concetti di base delle associazioni supertipo/sottotipo e abbiamo introdotto una notazione di base per questi concetti. Abbiamo inoltre descritto i processi di generalizzazione e specializzazione, che aiutano il modellatore dei dati a riconoscere le opportunità di sfruttare queste associazioni. In questo paragrafo introdurremo la notazione supplementare per rappresentare i vincoli sulle associazioni supertipo/sottotipo. Questi vincoli permettono di illustrare alcune delle regole aziendali più

importanti che si applicano a queste associazioni. I due tipi di vincoli più importanti che sono descritti in questo paragrafo sono quelli di completezza e di separazione (Elmasri e Navathe, 1994).

6.2.1 Specificare i vincoli di completezza

Un **vincolo di completezza** affronta la questione se un'istanza di un supertipo deve essere anche un membro di almeno un sottotipo. Il vincolo di completezza ha due regole possibili: la specializzazione totale e la specializzazione parziale. La regola della specializzazione totale specifica che ciascuna istanza di entità del supertipo deve essere un membro di un sottotipo nell'associazione. La regola della specializzazione parziale specifica che un'istanza di entità del supertipo non appartiene ad alcun sottotipo. Illustriamo ognuna di queste regole nei primi esempi di questo capitolo (Figura 6.6).

Regola della specializzazione totale

La Figura 6.6a riprende l'esempio di PATIENT (Figura 6.3) e introduce la notazione per la specializzazione totale. In questo esempio, la regola aziendale è la seguente: "Un paziente deve essere un paziente esterno o un paziente interno (non ci sono altri tipi di pazienti nell'ospedale)". La specializzazione totale è indicata dalla linea *doppia* che va dal tipo di entità PATIENT al cerchio.

In questo esempio, ogni volta che una nuova istanza di PATIENT viene inserita nel supertipo, un'istanza corrispondente è inserita in OUTPATIENT o in RESIDENT PATIENT. Se l'istanza è inserita in RESIDENT PATIENT, viene creata un'istanza dell'associazione Is_assigned per assegnare il paziente a un letto dell'ospedale.

Regola della specializzazione parziale

La Figura 6.6b riprende l'esempio di VEHICLE e dei suoi sottotipi CAR e TRUCK della Figura 6.4. Si ricorderà che in questo esempio la motocicletta è un tipo di veicolo, ma che non è rappresentato come sottotipo nel modello di dati. Così, se un veicolo è un automobile, deve apparire come un'istanza di CAR, e se è un camion, deve apparire come un'istanza di TRUCK. Tuttavia, se il veicolo è una motocicletta, non può apparire come istanza di alcun sottotipo. Questo è un esempio di specializzazione parziale ed è specificato della linea *singola* che va dal supertipo VEHICLE al cerchio.

6.2.2 Specificare i vincoli di separazione

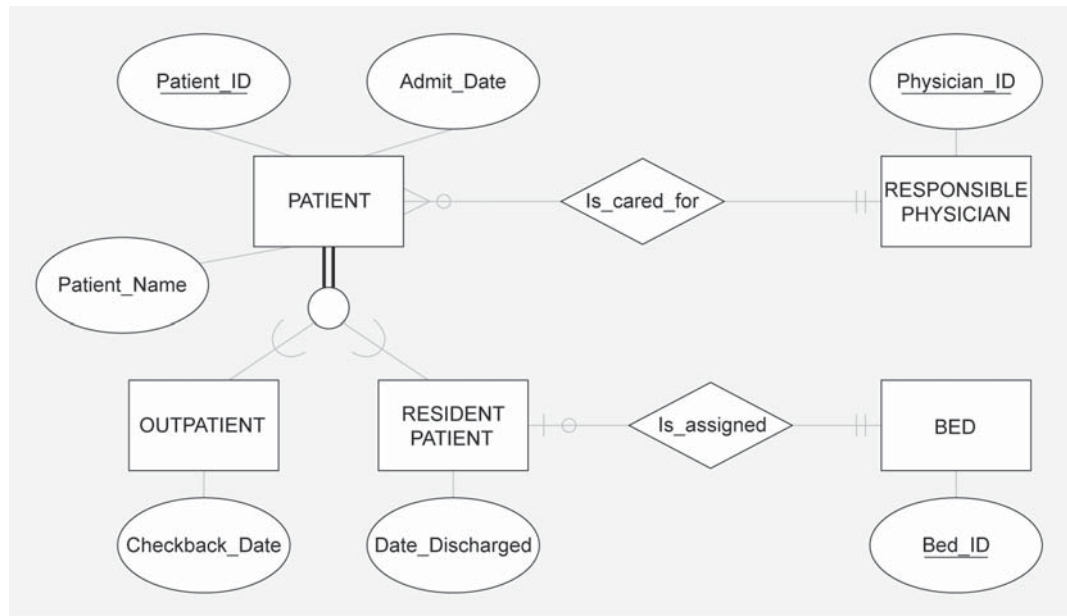
Un **vincolo di separazione** affronta la questione se un'istanza di un supertipo può essere contemporaneamente un membro di uno (o più) sottotipi. Il vincolo di separazione ha due regole possibili: la regola di separazione e la regola di sovrapposizione. La regola della separazione specifica che se un'istanza di entità (o del supertipo) è un membro di un sottotipo, non può essere contemporaneamente un membro di un altro sottotipo. La regola della sovrapposizione specifica che un'istanza di entità può essere contemporaneamente un membro di due (o più) sottotipi. Un esempio di queste regole è mostrato nella Figura 6.7.

Regola della separazione

La Figura 6.7 mostra l'esempio di PATIENT della Figura 6.4a. In questo caso, la regola aziendale è la seguente: "In un dato momento, un paziente deve essere un paziente interno o un paziente esterno ma non può essere entrambi". Questa è la regola della separazione, specificata dalla lettera **d** nel cerchio che unisce il supertipo ai suoi sottotipi. Si noti che in questa figura la sottoclasse di PATIENT può cambiare nel tempo, ma in un momento dato, un PATIENT deve essere di un solo tipo.

Figura 6.6

Esempi di vincoli di completezza.
(a) Regola della specializzazione totale.



(b) Regola della specializzazione parziale.

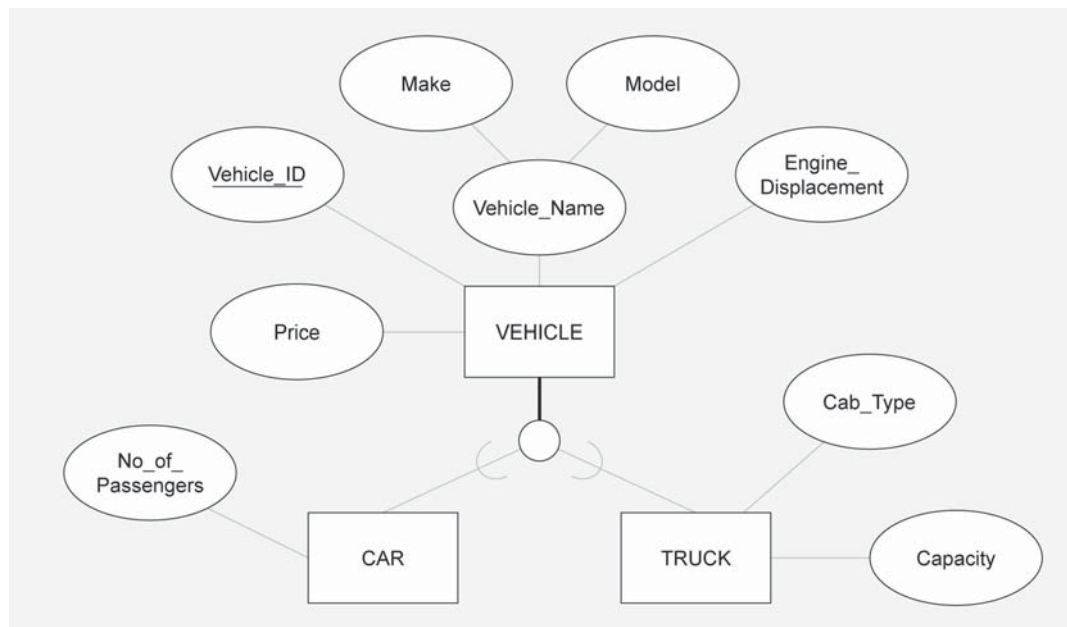
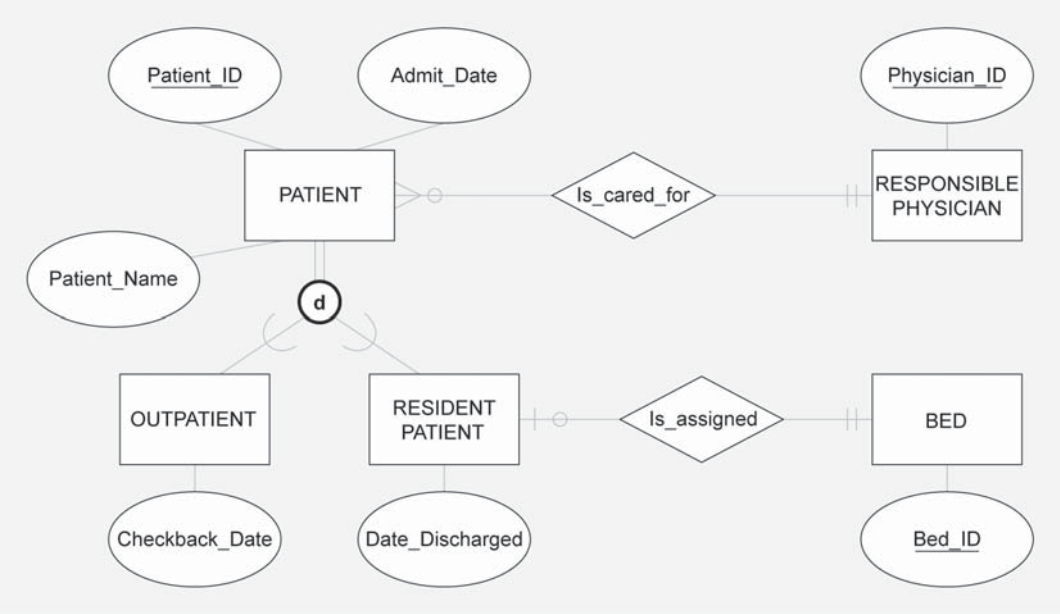
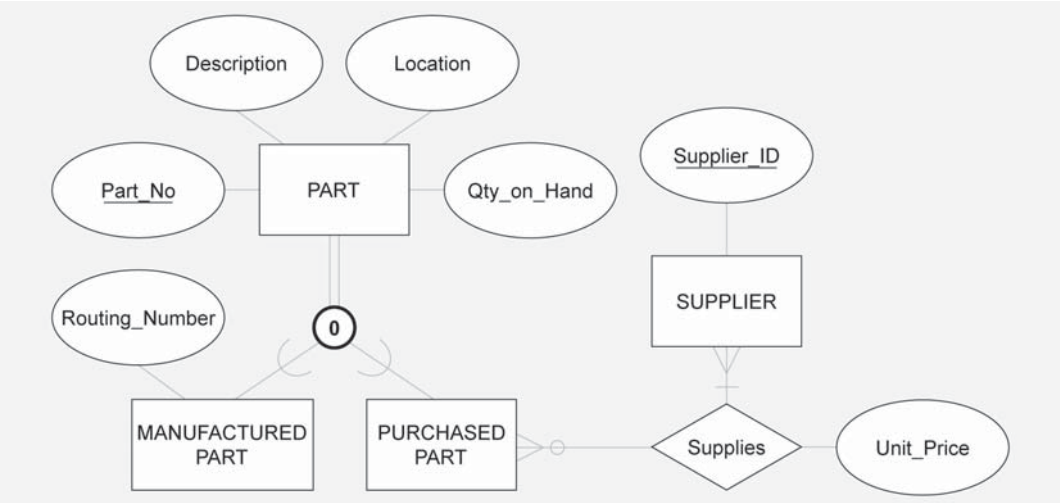


Figura 6.7

Esempi di vincoli di separazione.
(a) Regola della separazione.



(b) Regola della sovrapposizione.



Regola della sovrapposizione

La Figura 6.7 mostra il tipo di entità **PART** con i suoi due sottotipi **MANUFACTURED PART** e **PURCHASED PART** (dalla Figura 6.4b). Dalla discussione su questo esempio, si ricorderà che alcune parti sono sia prodotte sia acquistate. Questa affermazione deve essere chiarita. In questo esempio, un'istanza di **PART** è un numero di parte particolare (ossia un *tipo di parte*), e non una parte individuale (questa è indicata dall'identificatore, che è Part_No). Per esempio, si consideri il numero di parte 400. In un dato momento, la quantità a disposizione per questa parte può essere di 250, di cui 100 sono prodotte e le rimanenti 150 sono acquistate. In questo caso non è importante tenere traccia delle singole parti. Quando invece è importante farlo, a ogni parte viene assegnato un numero identificativo seriale e la quantità a disposizione è 1 o 0, a seconda che la parte esista o meno.

La regola della sovrapposizione è specificata collocando la lettera **o** nel cerchio, come mostrato nella Figura 6.7b. Si noti che nella figura è specificata anche la regola della specializzazione totale, come indicato dalla linea doppia. Qualsiasi parte può quindi essere una parte acquistata o una parte prodotta, oppure può essere contemporaneamente entrambe le cose.

6.2.3 Definire le discriminanti di sottotipo

Data un'associazione supertipo/sottotipo, si consideri il problema dell'inserimento di una nuova istanza del supertipo. In quale dei sottotipi (se ce ne sono) dovrebbe essere inserita questa istanza? Abbiamo già visto le varie regole possibili che si applicano in questa situazione. Occorrerà un meccanismo semplice per implementare queste regole, se ne è disponibile uno. Spesso si può ottenere questo risultato utilizzando una discriminante di sottotipo. Una discriminante di sottotipo è un attributo del supertipo i cui valori determinano il sottotipo o i sottotipi di destinazione.

Sottotipi di separazione

Un esempio dell'uso di una discriminante di sottotipo è mostrato nella Figura 6.8. L'esempio riguarda il supertipo **EMPLOYEE** e i suoi sottotipi, presentati nella Figura 6.2. Si noti che nella figura sono stati aggiunti i seguenti vincoli: specializzazione totale e sottotipi di separazione.

Un nuovo attributo (**Employee_Type**) è stato aggiunto al supertipo come discriminante di sottotipo. Quando un nuovo impiegato viene aggiunto al supertipo, l'attributo viene codificato con uno di tre valori: "H" (per Hourly), "S" (per Salaried) o "C" (per Consultant). A seconda di questo codice, l'istanza viene quindi assegnata al sottotipo appropriato.

La notazione utilizzata per specificare la discriminante di sottotipo è mostrata anche nella Figura 6.8. L'espressione **Employee_Type** (che si trova sul lato sinistro di un'istruzione di condizione) viene collocata vicino alla linea che va dal supertipo al cerchio. Il valore dell'attributo che seleziona l'attributo opportuno (in questo esempio, "H", "S" o "C") viene collocato adiacente alla linea che porta a quel sottotipo. Così, per esempio, la condizione "**Employee_Type**= "S" fa sì che un'istanza di entità venga inserita nel sottotipo **SALARIED EMPLOYEE**.

Sovrapporre i sottotipi

Quando i sottotipi si sovrappongono, deve essere applicato un approccio leggermente modificato della discriminante di sottotipo. La ragione è che una data istanza di sottotipo può richiedere la creazione di un'istanza in più di un sottotipo.

Un esempio di questa situazione è mostrato nella Figura 6.9 per **PART** e i suoi sottotipi che si sovrappongono. Un nuovo attributo chiamato **Part_Type** è stato aggiunto a **PART**. **Part_Type** è un attributo composto con i componenti **Manufactured?** e **Purchased?**. Ognuno di questi attributi è un valore booleano (ossia, adotta solo i valori Sì, "Y", e No, "N"). Quando viene aggiunta una nuova istanza a **PART**, questi componenti vengono codificati come segue.

Tipo di parte	Manufactured?	Purchased?
Solo prodotto	"Y"	"N"
Solo acquistato	"N"	"Y"
Acquistato e prodotto	"Y"	"Y"

Il metodo per specificare la discriminante di sottotipo per questo esempio è mostrato nella Figura 6.9. Si noti che questo approccio può essere utilizzato per qualsiasi numero di sottotipi che si sovrappongono.

Figura 6.8
Introduzione di una discriminante di sottotipo (regola della separazione).

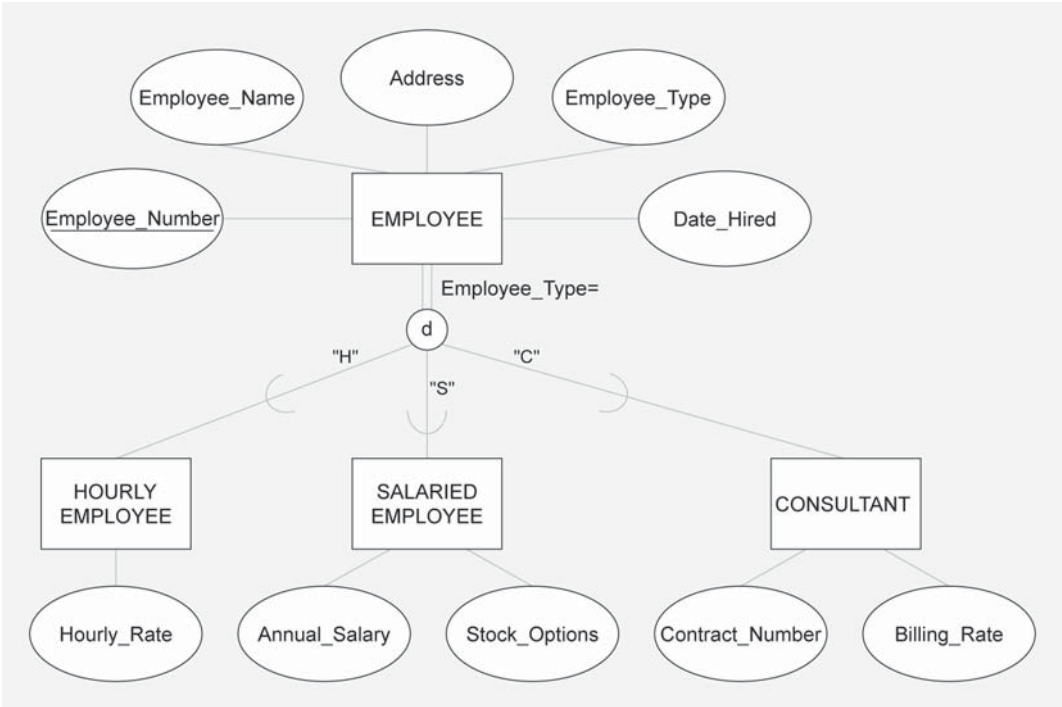
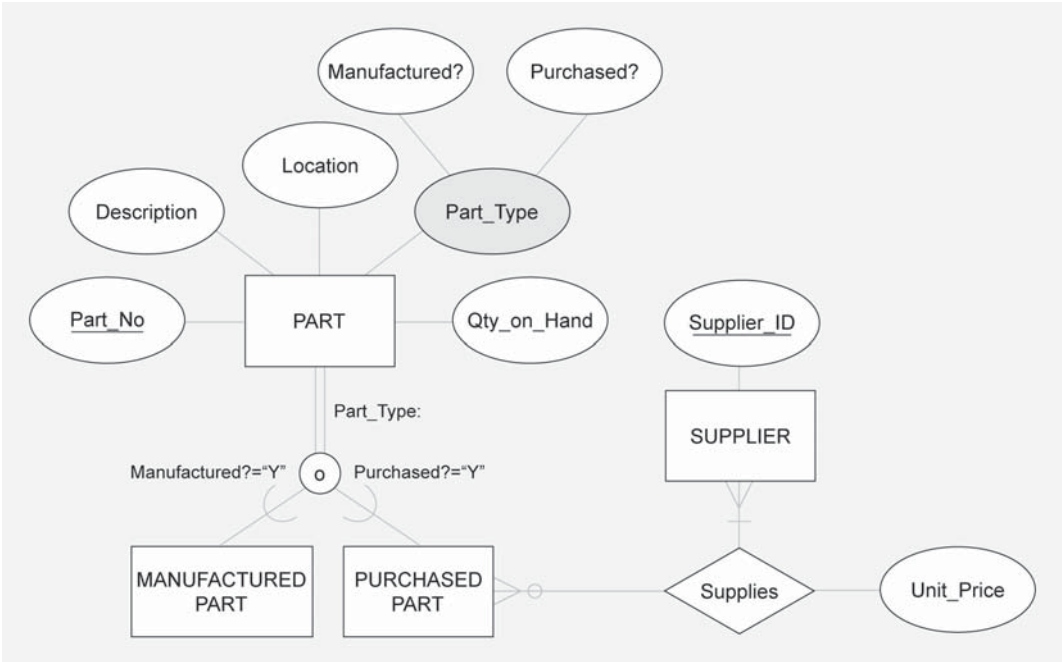


Figura 6.9
Discriminante di sottotipo (regola della sovrapposizione).



6.2.4 Definire le gerarchie di supertipo/sottotipo

Finora abbiamo visto alcuni esempi di associazioni supertipo/sottotipo. Per ognuno dei sottotipi in questi esempi è possibile avere altri sottotipi definiti su di essi (in questo caso il sottotipo diventa un supertipo dei nuovi sottotipi definiti). Una gerarchia di supertipo/sottotipo è una disposizione gerarchica dei supertipi e dei sottotipi, dove ogni sottotipo ha un solo supertipo (Elmasri e Navathe, 1994).

In questo paragrafo presenteremo un esempio di gerarchia di supertipo/sottotipo (Figura 6.10) che include la maggior parte dei concetti e delle notazioni utilizzati fino a questo momento nel capitolo. L'esempio presenta anche una metodologia, basata sulla specializzazione, che si può utilizzare in molte situazioni di modellazione dei dati.

Un esempio

Si supponga di dover modellare le risorse umane in un'università. Utilizzando la specializzazione (approccio top-down), si può procedere come descritto di seguito. Partendo dall'alto della gerarchia, modellare innanzitutto il tipo di entità più generale. In questo caso si tratta del tipo di entità PERSON. Elencare e associare tutti gli attributi di PERSON. Gli attributi mostrati nella Figura 6.10 sono SSN (identificatore), Name, Address, Gender e Date_of_Birth. Il tipo di entità all'inizio della gerarchia viene anche chiamato *radice*.

Si prosegue definendo tutti i sottotipi principali della radice. In questo esempio ci sono tre sottotipi di PERSON: EMPLOYEE (persone che lavorano per l'università), STUDENT (persone che partecipano alle lezioni) e ALUMNUS (persone che si sono laureate). Si presuma che non ci siano altri tipi di persone di interesse per l'università alle quali si applica la regola della specializzazione totale, come mostrato nella figura. Poiché una persona può appartenere a più di un sottotipo (per esempio, ALUMNUS ed EMPLOYEE), viene utilizzata la regola della sovrapposizione. Si noti che la sovrapposizione permette qualsiasi tipo di sovrapposizione (una PERSON può essere contemporaneamente in qualsiasi coppia o in tutte e tre i sottotipi). Se determinate combinazioni non sono permesse, allora si dovrà sviluppare una gerarchia di supertipo/sottotipo più raffinata per eliminare le combinazioni vietate.

Gli attributi che si applicano specificatamente a ognuno di questi sottotipi sono mostrati nella figura. Ogni istanza di EMPLOYEE ha dunque un valore per Date_Hired e Salary. Major_Dept è un attributo di STUDENT e Degreee (con i componenti Year, Designation e Date) è un attributo di ALUMNUS.

Il passo successivo consiste nel valutare se qualcuno dei sottotipi già definiti si qualifica per un'ulteriore specializzazione. In questo esempio, EMPLOYEE è suddiviso in due sottotipi: FACULTY e STAFF. FACULTY ha l'attributo specifico Rank, mentre STAFF ha l'attributo specifico Position. Si noti che in questo esempio, il sottotipo EMPLOYEE diventa un supertipo di FACULTY e STAFF. Poiché ci possono essere tipi di impiegati diversi dal corpo docente e dal personale (come gli assistenti), viene indicata la regola della specializzazione parziale. Un impiegato non può tuttavia essere parte sia del corpo docente sia del personale. La regola della separazione è quindi indicata nel cerchio.

Per STUDENT sono definiti anche due sottotipi: GRADUATE STUDENT e UNDERGRAD STUDENT. UNDERGRAD STUDENT ha l'attributo Class_Standing, mentre GRADUATE STUDENT ha l'attributo Test_Score. Si noti che sono specificate la regola della specializzazione totale e della separazione; si dovrebbe essere in grado di definire le regole aziendali per questi vincoli.

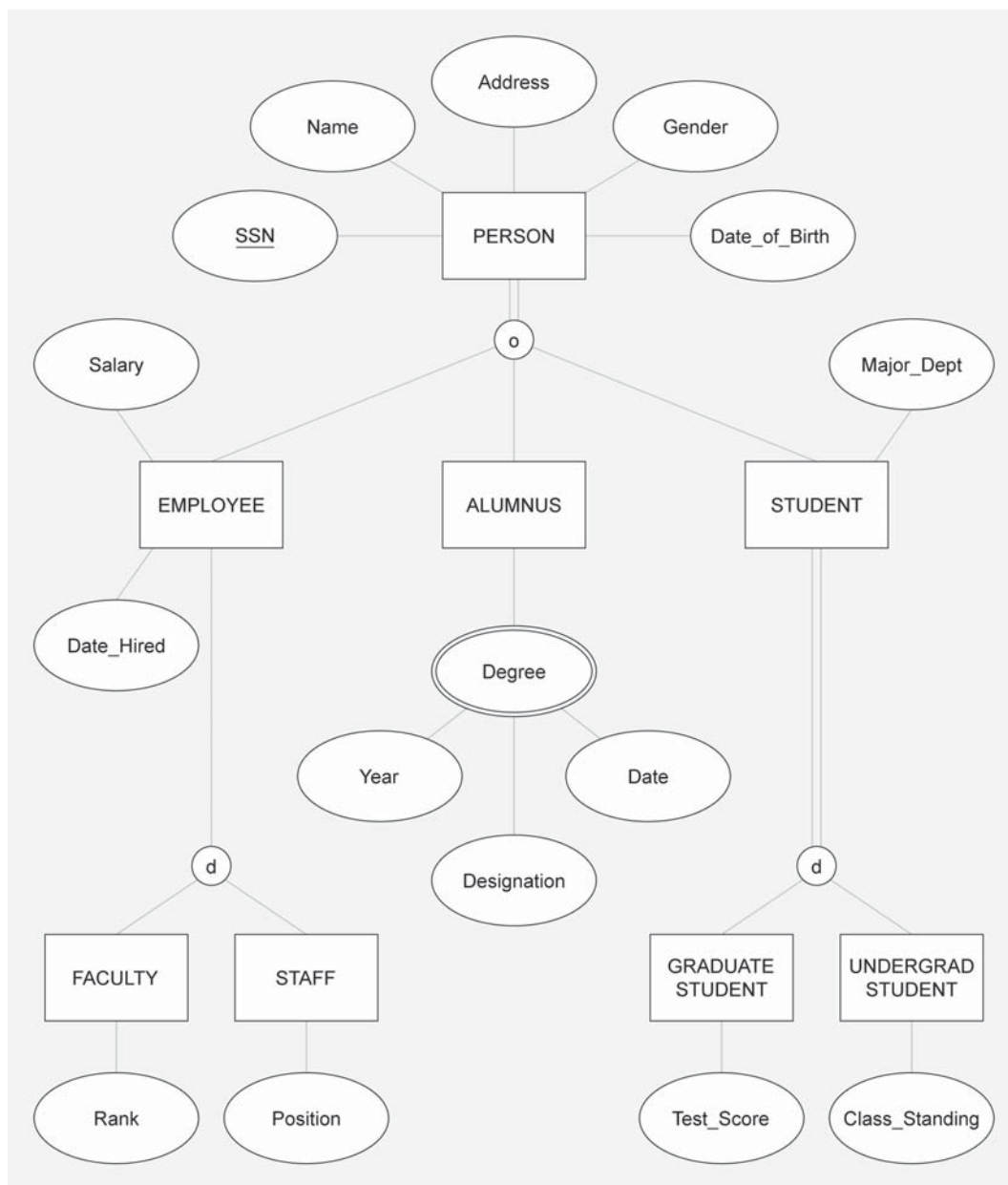
Riepilogo delle gerarchie di supertipo/sottotipo

Nella Figura 6.10, si notano due caratteristiche che riguardano gli attributi contenuti nella gerarchia.

1. Gli attributi vengono assegnati al livello logico più alto possibile nella gerarchia. Per esempio, poiché SSN (il numero della previdenza sociale) si applica a tutte le persone, viene assegnato alla radice.

Figura 6.10

Esempio di gerarchia di supertipo/sottotipo.



D'altra parte, Data_Hired si applica solo agli impiegati, quindi viene assegnato a EMPLOYEE. Questo approccio garantisce che gli attributi possano essere condivisi da quanti più sottotipi possibili.

2. I sottotipi che sono più in basso nella gerarchia ereditano gli attributi non solo del loro supertipo diretto, ma anche di tutti i sottotipi più in alto nella gerarchia, fino alla radice. Così, per esempio, un'istanza di Faculty ha valori per tutti gli attributi seguenti: SSN, Name, Address, Gender e Date_of_Birth (da PERSON), Data_Hired e Salary (da EMPLOYEE) e Rank (da FACULTY).

Nel project case alla fine del capitolo si chiederà di sviluppare un diagramma E-R esteso per Mountain View Community Hospital utilizzando la stessa procedura delineata in questo paragrafo.

6.3 Esempio di modellazione EER: Pine Valley Furniture Company

Nel Capitolo 5 abbiamo presentato un esempio di diagramma E-R per Pine Valley Furniture Company (questo diagramma viene ripetuto nella Figura 6.11). Dopo averlo studiato, ci si può servire di alcune domande che aiutino a chiarire il significato delle entità e delle associazioni. Tre ambiti di queste domande sono i seguenti.

1. Perché tutti i clienti non hanno attività in una o più zone di vendita?
2. Perché alcuni impiegati sono supervisori di altri e perché non sono oggetto della supervisione di un altro impiegato? E perché alcuni impiegati non lavorano in un centro di produzione?
3. Perché tutti i produttori non forniscono materiali grezzi a Pine Valley Furniture?

Ci possono essere altre domande, ma per ora ci concentreremo su queste tre per illustrare come le associazioni supertipo/sottotipo possono essere utilizzate per rappresentare un modello di dati più specifico (cioè ricco dal punto di vista semantico).

Dopo alcune indagini partendo dalle domande precedenti, si sono scoperte le seguenti regole aziendali che si applicano all'attività di Pine Valley Furniture.

1. Ci sono due tipi di clienti: di zona e nazionali. Solo i clienti di zona hanno un'attività nelle zone di vendita. Una zona di vendita esiste solo se a essa è associato almeno un cliente di zona. Un cliente nazionale è associato a un account manager. È possibile per un cliente essere sia un cliente di zona sia nazionale.
2. Esistono due tipi di impiegati: gestionali e di unità. Solo gli impiegati di unità lavorano nei centri di produzione e un impiegato della gestione è supervisore degli impiegati di unità. Ci sono poi altri tipi di impiegati. Un impiegato di unità può essere promosso al management, allorché cessa di essere un impiegato di unità.
3. Pine Valley Furniture tiene traccia di numerosi produttori diversi, ma non di tutti quelli che hanno fornito materiali grezzi all'azienda. Un produttore è associato a un numero di contratto una volta che diventa un fornitore ufficiale di materiali grezzi.

Le regole aziendali citate sono state utilizzate per modificare il diagramma E-R della Figura 6.11 nel diagramma EER della Figura 6.12 (abbiamo omissso la maggior parte degli attributi dal diagramma a eccezione di quelli essenziali per vedere i cambiamenti che si sono verificati).

La regola 1 dice che c'è una specializzazione totale di sovrapposizione di CUSTOMER in REGULAR CUSTOMER e in NATIONAL ACCOUNT CUSTOMER. Un attributo composto di CUSTOMER, Customer_Type, viene utilizzato per indicare se un'istanza di cliente è un cliente di zona, uno nazionale o entrambi. Poiché solo i clienti di zona hanno un'attività nelle zone di vendita, solo questi sono coinvolti nell'associazione Does_business_in e la cardinalità minima vicino al cliente di zona è uno, perché una zona di vendita per esistere deve avere almeno un cliente di zona.

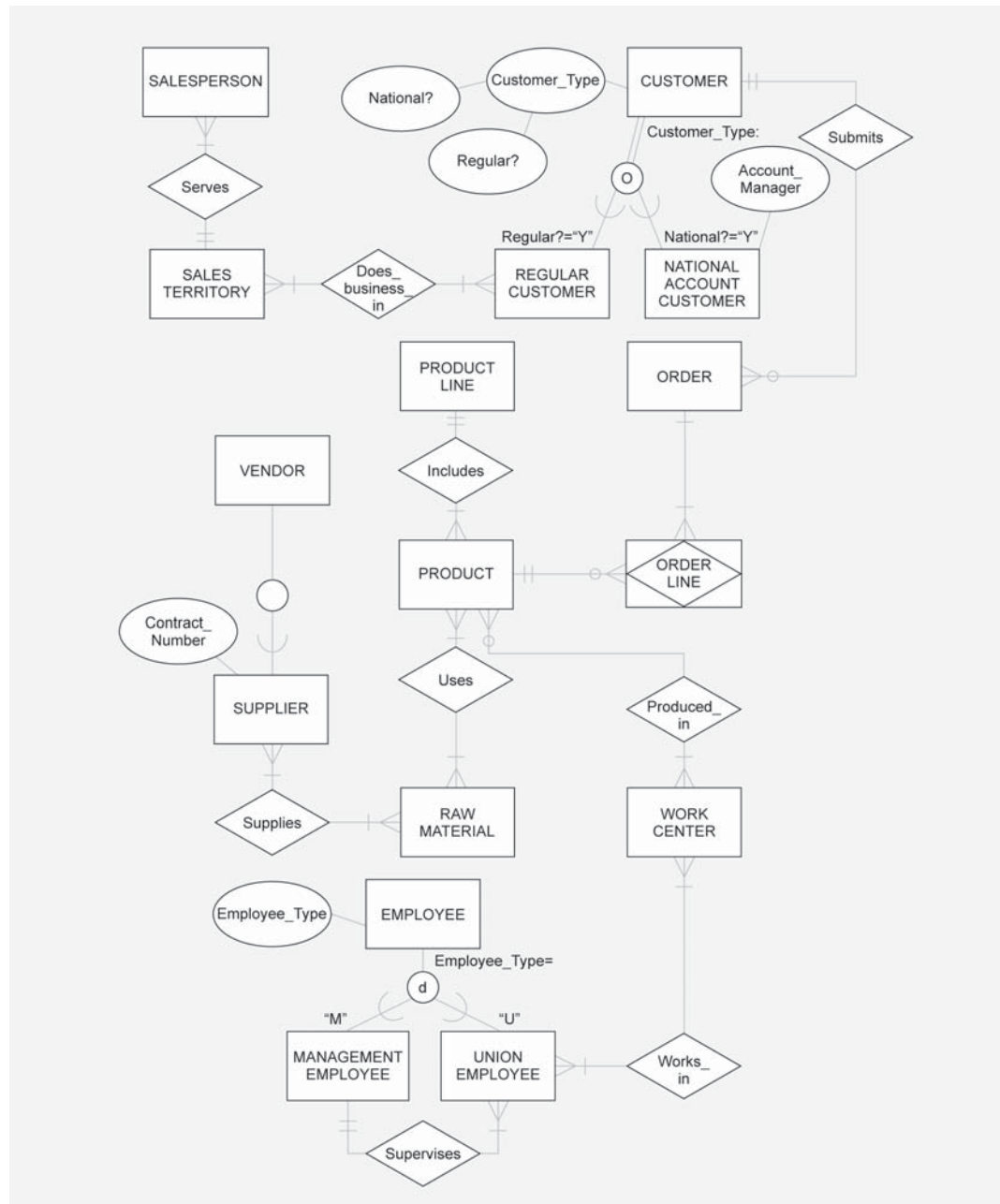
La regola 2 significa che c'è una specializzazione parziale di separazione di EMPLOYEE in MANAGEMENT EMPLOYEE e UNION EMPLOYEE. Un attributo di EMPLOYEE, Employee_Type, è la discriminante tra due tipi speciali di impiegati. La specializzazione è parziale perché ci sono altri tipi

Figura 6.11
Diagramma E-R per
Pine Valley Furniture
Company.



di impiegati oltre a questi due. Solo gli impiegati di unità sono coinvolti nell'associazione Works_in, ma tutti gli impiegati di unità lavorano in un centro produttivo; la cardinalità minima di Works in accanto a WORK CENTER è ora obbligatoria. Poiché un impiegato non può essere contemporaneamente di unità e di management, la specializzazione è di separazione.

Figura 6.12
Diagramma EER per
Pine Valley Furniture
Company.



La regola 3 indica che c'è una specializzazione parziale di **VENDOR** in **SUPPLIER** perché solo alcuni produttori diventano anche fornitori. Un fornitore, non un produttore, ha un numero di contratto. Poiché c'è solo un sottotipo di **VENDOR**, non c'è ragione di specificare una regola di separazione o di sovrapposizione. Poiché tutti i fornitori forniscono del materiale grezzo, la cardinalità minima accanto a **RAW MATERIAL** nell'associazione **Supplies** è uno.

Questi esempi mostrano come un diagramma E-R possa essere trasformato in un diagramma EER una volta che è stata compresa la generalizzazione/specializzazione delle entità. Non solo le entità di supertipo e sottotipo sono ora nel diagramma, ma vengono aggiunti anche attributi supplementari, compresi quelli discriminanti, le cardinalità minime cambiano (da facoltative a obbligatorie) e le associazioni si spostano dal supertipo al sottotipo.

Questo è il punto adatto in cui sottolineare una puntualizzazione fatta in precedenza sulla modellazione dei dati. Un modello di dati è un'immagine concettuale dei dati richiesti da un'azienda. Un modello di dati non mappa uno per uno gli elementi di una base di dati implementata. Per esempio, per chi ha familiarità con un prodotto di basi di dati come Microsoft Access, sebbene SUPPLIER sia un tipo di entità e Supplies sia un'associazione, in Access entrambi questi elementi appariranno come tabelle in una base di dati. Tali dettagli ricoprono una grande importanza. Ora lo scopo è quello di spiegare tutte le regole che governano i dati, e non come i dati verranno memorizzati e come vi si accederà per ottenere un'elaborazione efficiente delle informazioni richieste. Affronteremo la tecnologia e i problemi di efficienza nei prossimi capitoli, quando tratteremo la progettazione e l'implementazione delle basi di dati.

Sebbene il diagramma EER nella Figura 6.12 chiarisca alcune domande e renda il modello di dati nella Figura 6.11 più esplicito, per qualcuno può essere ancora difficile capire il tutto. Alcune persone non saranno interessate a tutti i tipi di dati, mentre altre non avranno bisogno di vedere tutti i dettagli nel diagramma EER per capire cosa copre una determinata base di dati. Il prossimo paragrafo accenna come si può semplificare un modello di dati completo ed esplicito per la presentazione a gruppi di utenti e manager specifici.

6.4 Accorpamento di entità

Alcuni sistemi informativi aziendali possono annoverare oltre mille tipi di entità e associazioni. Come presentare una quantità così voluminosa dei dati organizzativi agli sviluppatori e agli utenti? Un approccio consiste nel creare più diagrammi E-R, ciascuno dei quali deve mostrare i dettagli dei diversi segmenti che potrebbero sovrapporsi o dei sottoinsiemi del modello di dati (per esempio, segmenti diversi che si applicano a reparti, applicazioni dei sistemi informativi, processi aziendali o divisioni della società). Il metodo, seppur funzionale, è inadeguato, poiché non si otterrebbe una rappresentazione complessiva.

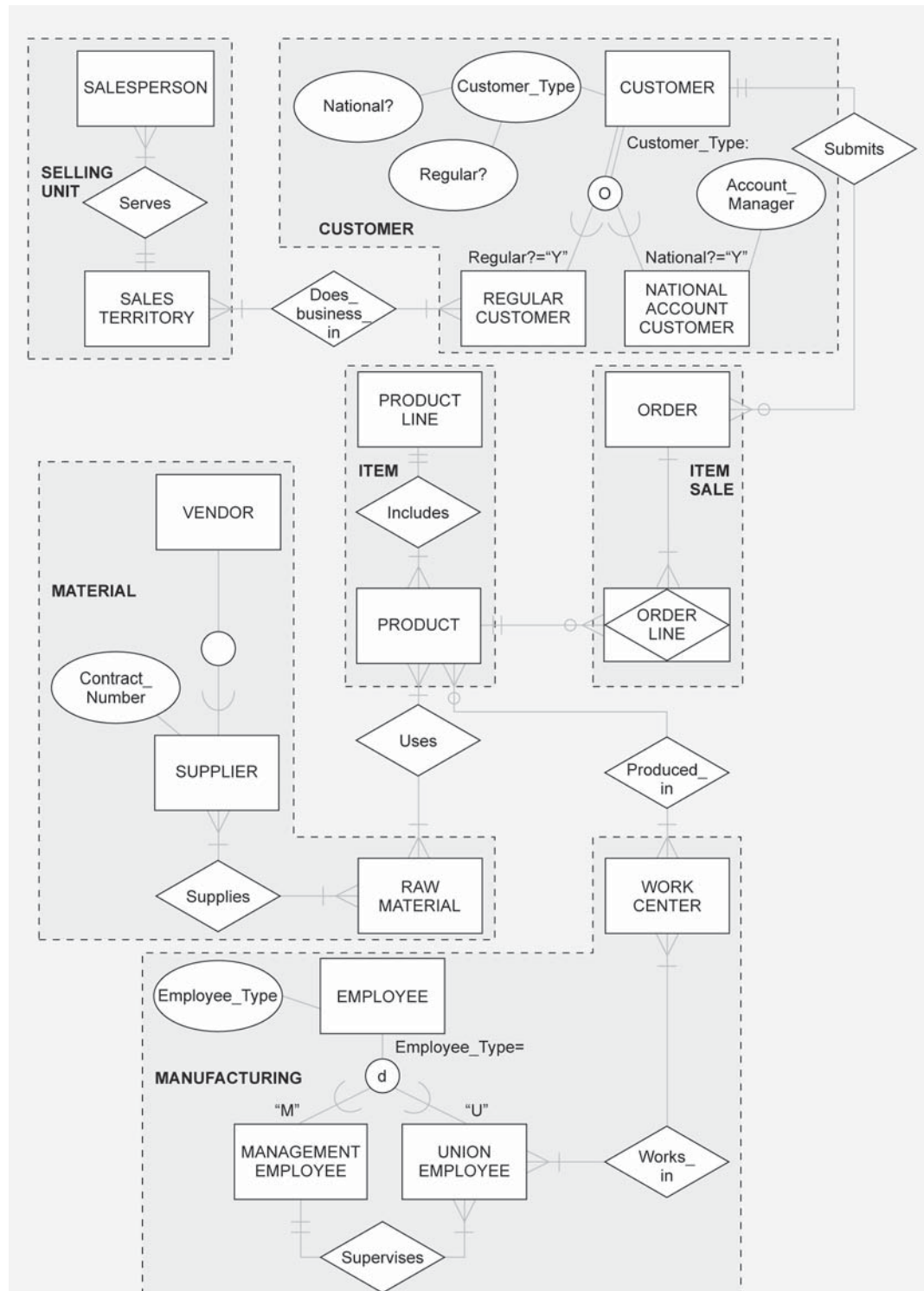
L'accorpamento delle entità, detto anche *clustering* (Teorey, 1999), è un modo utile per presentare un modello di dati di un'azienda vasta e complessa. Un **cluster di entità** è un insieme di uno o più tipi di entità e delle associazioni accorpati in un unico tipo di entità astratta.

Poiché un cluster di entità si comporta come un tipo di entità, i cluster di entità e di tipo possono essere ulteriormente accorpati a formare un cluster di entità di livello superiore. L'accorpamento di entità è una scomposizione di una vista a macro-livello del modello di dati in viste sempre più raffinate, che può generare il modello di dati completo e dettagliato.

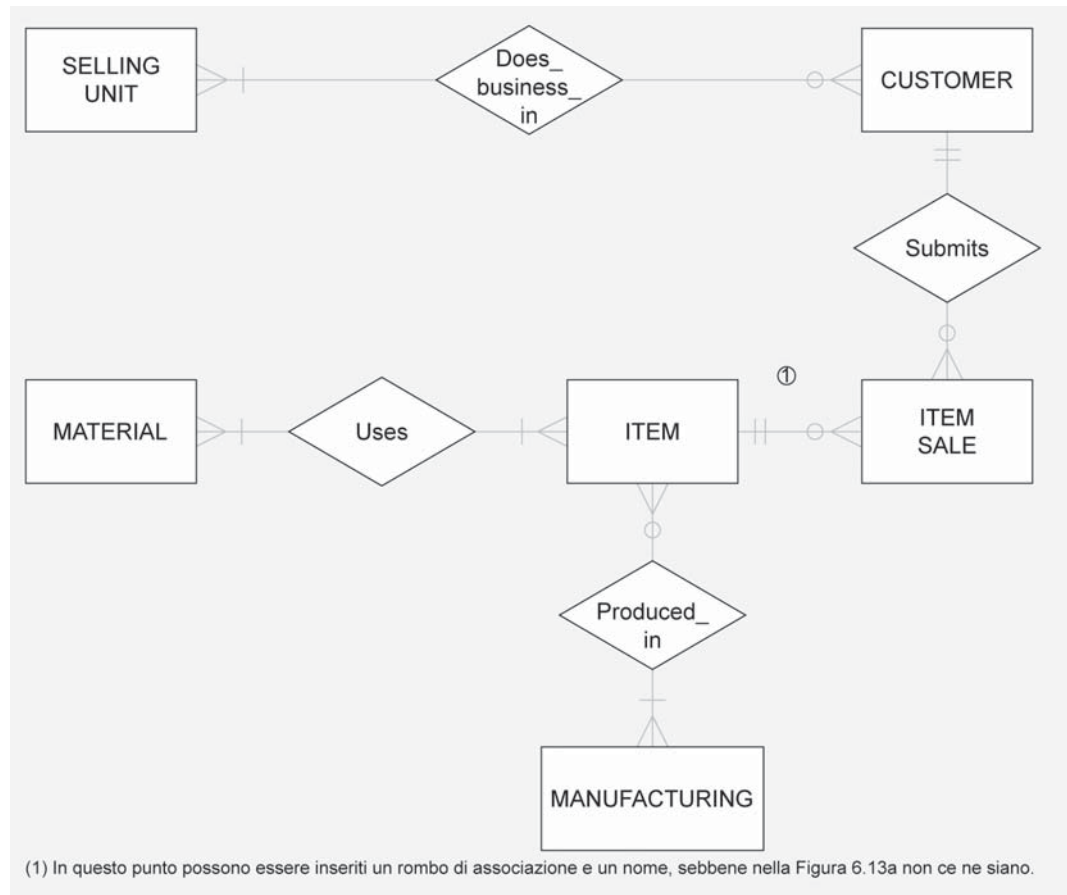
La Figura 6.13 illustra uno dei possibili risultati dell'accorpamento di entità per il modello di dati di Pine Valley Furniture mostrato nella Figura 6.12. La Figura 6.13 mostra il modello di dati completo con alcune aree tratteggiate che indicano i cluster di entità possibili. La Figura 6.13b mostra il risultato finale della trasformazione del diagramma EER dettagliato in un diagramma EER dei soli cluster di entità e delle associazioni (un diagramma EER può includere sia i cluster di entità sia i tipi di entità, ma questo diagramma include solo i primi). In questa figura, il cluster di entità rappresenta quanto segue.

Figura 6.13

Accorpamento di entità per il modello di dati di Pine Valley Furniture.
(a) Cluster di entità possibili.



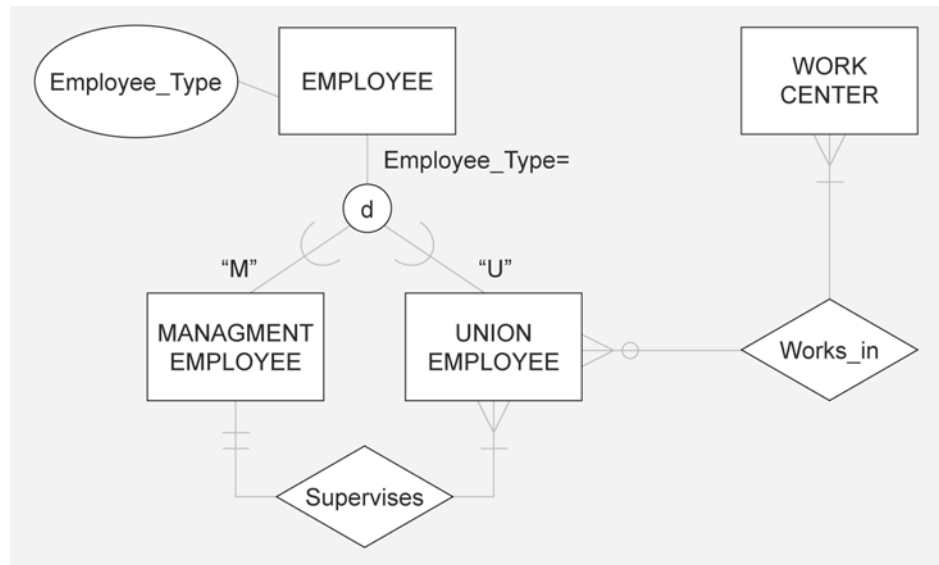
(b) Diagramma EER di cluster di entità.



- **SELLING UNIT**: rappresenta i tipi di entità SALESPERSON e SALES TERRITORY e l'associazione Serves.
- **CUSTOMER**: rappresenta il supertipo di entità CUSTOMER, i suoi sottotipi e l'associazione tra il supertipo e i sottotipi.
- **ITEM SALE**: rappresenta il tipo di entità ORDER e l'entità associativa ORDER LINE, oltre all'associazione tra le due.
- **ITEM**: rappresenta i tipi di entità PRODUCT LINE e PRODUCT e l'associazione Includes.
- **MANUFACTURING**: rappresenta l'entità dei supertipi WORK CENTER ed EMPLOYEE e i suoi sottotipi, oltre alle associazioni Works-in e Supervises e all'associazione tra il supertipo e i suoi sottotipi (la Figura 6.14 mostra un'esplosione del cluster di entità MANUFACTURING nei suoi componenti).
- **MATERIAL**: rappresenta i tipi di entità RAW MATERIAL e VENDORS, il sottotipo SUPPLIER, l'associazione Supplies e l'associazione supertipo/sottotipo tra VENDOR e SUPPLIER.

I diagrammi E-R mostrati nelle Figure 6.13 e 6.14 possono essere utilizzati per illustrare i dettagli a coloro che si occupano dei processi di assemblaggio e che sono particolarmente interessati alle informa-

Figura 6.14
Cluster di entità
MANUFACTURING.



zioni necessarie a supporto dell'attività stessa. Per esempio, osservando la Figura 6.13, un responsabile della gestione delle scorte può vedere che i dati relativi alla produzione possono essere messi in relazione ai dati sugli articoli (l'associazione Produced_in). Più sotto, la figura mostra il livello di dettaglio mantenuto riguardo al processo di produzione che coinvolge i centri di produzione e gli impiegati. Questa persona probabilmente non ha bisogno di vedere, per esempio, i dettagli sulla struttura delle vendite che è incorporata nel cluster di entità SELLING UNIT.

I cluster di entità mostrati nella Figura 6.13 sono stati creati astruendo un supertipo e il suo sottotipo (si veda il cluster di entità CUSTOMER) e combinando direttamente i tipi di entità correlati e le loro associazioni (si vedano i cluster di entità SELLING UNIT, ITEM, MATERIAL e MANUFACTURING). Un cluster di entità può anche essere creato combinando un'entità forte e i tipi di entità deboli a essa associati (non illustrati in questo esempio). Poiché l'accorpamento delle entità è gerarchico, qualora fosse necessario, si potrebbe disegnare un altro diagramma EER in cui combinare i cluster di entità SELLING UNIT e CUSTOMER con l'associazione Does_business_in in un unico cluster di entità, poiché questi sono cluster di entità direttamente correlati.

Un cluster di entità dovrebbe focalizzarsi su un'area di interesse per una determinata comunità di utenti, sviluppatori o manager. Quali tipi di entità e associazioni debbano essere accorpati per formare un cluster di entità dipende dall'obiettivo che ci si è prefissati. Nell'esempio dell'accorpamento di entità del modello di dati Pine Valley Furniture, il tipo di entità ORDER potrebbe essere accorpati con il cluster di entità CUSTOMER, mentre il tipo di entità ORDER LINE potrebbe essere accorpati con il cluster di entità ITEM. Questo accorpamento eliminerebbe il cluster ITEM SALE, che potrebbe non essere di interesse per il gruppo di utenti considerato. È inoltre possibile effettuare numerosi e diversi accorpamenti di entità del modello di dati, ciascuno con un focus differente.

6.5 Regole aziendali rivedute e corrette

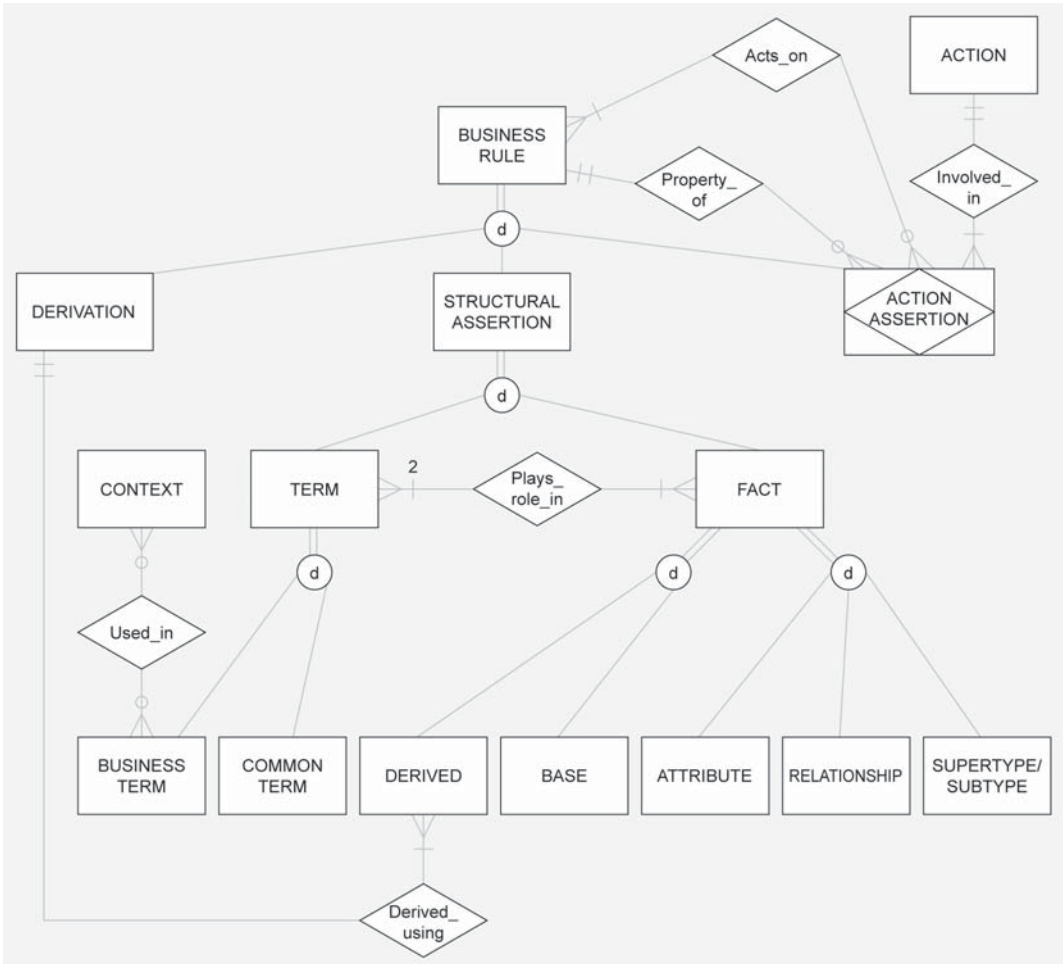
Come si è visto, i diagrammi E-R (e i diagrammi E-R estesi) sono uno strumento utile per esprimere determinati tipi di regole aziendali. Così, per esempio, i vincoli di partecipazione e di separazione associati

a supertipi e sottotipi trattati in questo capitolo sono un'espressione delle regole aziendali associate a tali associazioni. In un'azienda, esistono tuttavia molti altri tipi di regole che non possono essere espresse secondo questa notazione. In questo paragrafo svilupperemo un framework generico per le regole aziendali e mostreremo come esprimere alcuni tipi di regole importanti che eludono le notazioni standard del modello E-R ed E-R esteso.

6.5.1 Classificazione delle regole aziendali

Esistono molti tipi diversi di regole aziendali. Nel Capitolo 5 e in questo capitolo ne sono già stati mostrati alcuni esempi, come le associazioni tra tipi di entità (per esempio, i valori di cardinalità), le associazioni supertipo/sottotipo, i fatti relativi agli attributi e i tipi di entità (per esempio, le definizioni). La Figura 6.15, adattata dal GUIDE Business Rules Project, mostra un diagramma E-R che classifica le regole aziendali e le associazioni potenziali tra i diversi tipi di regole.

Figura 6.15
Diagramma E-R che
descrive le regole
aziendali (adattato dal
GUIDE Business Rules
Project, 1997).



I tre tipi di regole aziendali principali sono la derivazione, l'asserzione strutturale e l'asserzione di azione. Una **derivazione** è un'istruzione derivata da altre conoscenze nell'azienda. Solitamente si tratta di un'inferenza matematica o logica che implica letterali e fatti. Poiché nei modelli di dati si è particolarmente interessati ai fatti derivati, che si ottengono utilizzando le derivazioni, tratteremo le derivazioni solo indirettamente quando discuteremo i fatti derivati in uno dei prossimi paragrafi. Un'asserzione strutturale è un'istruzione che esprime alcuni aspetti della struttura statica dell'azienda.

I diagrammi sono un modo comune per descrivere le asserzioni strutturali. Come mostra la Figura 6.15, un'asserzione strutturale viene espressa sia come termine sia come fatto (i termini e i fatti vengono definiti nel Capitolo 5), e i termini giocano un ruolo nell'istruzione di un fatto. Un termine può essere comune, cioè di comprensione diffusa, o aziendale, nel qual caso può essere compreso solo nel contesto di una data struttura. Un'asserzione di azione è un'istruzione di vincolo o controllo sulle azioni dell'azienda. Un'asserzione di azione è la proprietà di una data regola aziendale e definisce in quali condizioni una data azione può essere eseguita su determinate regole aziendali (si ricordi che una regola aziendale può essere semplice come la definizione di un tipo di entità, un'associazione o un attributo). Per esempio, un'asserzione di azione può definire in quali condizioni è possibile creare un nuovo cliente o quando può essere scritto un nuovo ordine. Un altro tipo comune di asserzione di azione determina quali valori possono essere contenuti in un determinato attributo (si parla anche di *vincolo di dominio*).

Nei paragrafi che seguono descriveremo le varie forme di asserzioni strutturali, che vengono solitamente illustrate in diagrammi E-R ed EER e nella documentazione della base di dati associata. Nel prossimo paragrafo spiegheremo le asserzioni di azione, che possono essere implementate in una base di dati come trigger e stored procedure, come vedremo successivamente nel testo. Per illustrare queste asserzioni, utilizzeremo un modello di dati semplice per un programma di lezioni. Il modello E-R per questo esempio è mostrato nella Figura 6.16. Il modello E-R mostrato nella Figura 6.16 contiene i quattro tipi di entità FACULTY, COURSE, SECTION e STUDENT e mostra le associazioni che intercorrono tra di essi. Si noti che SECTION è un tipo di entità debole, poiché non può esistere senza il tipo di entità COURSE. L'associazione identificativa di SECTION è Is_scheduled, mentre l'identificatore parziale è Section_ID (che è un attributo composto). Per semplificare il diagramma, vengono mostrati solo gli attributi selezionati per i diversi tipi di entità.

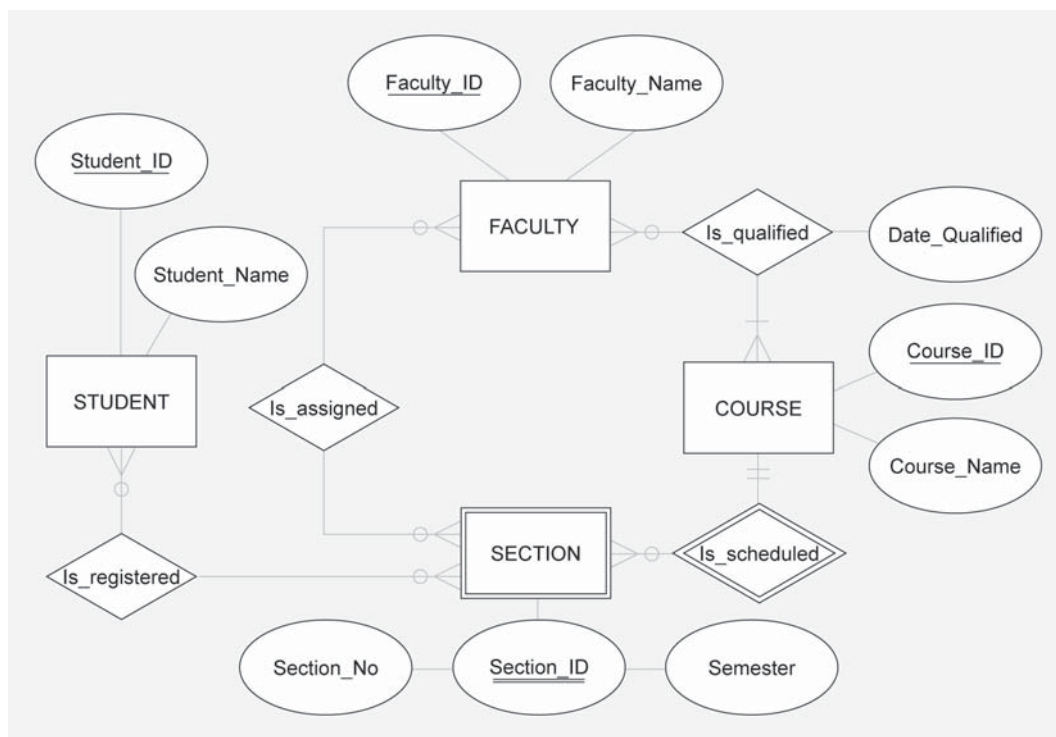
6.5.2 Dichiarare un'asserzione strutturale

Un'istruzione strutturale dichiara che un elemento importante per l'azienda esiste o esiste in relazione ad altri elementi di interesse. Può essere semplice come la definizione di un termine, o di un fatto, che è un'istruzione di un'associazione tra termini. Di seguito vengono riportati quattro esempi di fatti.

- Un corso è un modulo di insegnamento in una determinata materia. Questo corso *definizione del termine* associa due termini: il modulo di insegnamento e la materia. Presumiamo qui che si tratti di termini comuni che non necessitano di un'ulteriore definizione, né di una collocazione in un contesto aziendale.
- Il nome dello studente è un *attributo* dello studente. Tale fatto viene mostrato nella Figura 6.16 nell'ovale dell'attributo Student_Name.
- Uno studente può iscriversi a più seminari, e in un seminario possono esserci iscrizioni di numerosi studenti. Questo fatto definisce la partecipazione dei tipi di entità in un'associazione.
- Un corpo docente (*faculty*) è formato da dipendenti dell'università. Sebbene non venga mostrato nella Figura 6.16, questo fatto indica un'associazione *supertipo/sottotipo* tra il sottotipo "corpo docente" e il suo sottotipo "dipendente".

Figura 6.16

Segmento di modello di dati per un programma di lezioni.



Fatti derivati

Il tipo di fatto precedentemente descritto viene chiamato *fatto di base*; si tratta infatti di fatti fondamentali che non possono essere derivati da altri termini o fatti. Un altro tipo di fatto (mostrato nella Figura 6.15) viene chiamato fatto derivato. Un fatto derivato è un fatto che viene dedotto dalle regole aziendali utilizzando un algoritmo o un'inferenza e che può essere trattato come un fatto di base (per esempio, poiché un fatto derivato è un tipo di regola aziendale, un'asserzione di azione può essere una proprietà di un fatto derivato). Un attributo derivato, definito nel Capitolo 5, è un esempio di fatto derivato. Di seguito sono riportati due esempi di fatti derivati.

- $\text{Student\_GPA} = \frac{\text{Quality\_Points}}{\text{Total\_Hours\_Taken}}$
 where $\text{Quality\_Points} = \text{sum} [\text{per tutti i corsi frequentati}]$
 $(\text{Credit\_Hours} * \text{Numerical\_Grade})$

In questo esempio, Student_GPA è derivato da altri fatti di base e derivati. Anche Quality_Points e Credit_Hours sono fatti derivati. Student_GPA può essere rappresentato nella Figura 6.16 in un ovale tratteggiato connesso a STUDENT; Numerical_Grade può apparire come attributo di Is_Registered e Credit_Hours come attributo di COURSE.

- Uno studente viene indirizzato dal corpo docente assegnato ai seminari ai quali si è iscritto. Per vedere come questo fatto può essere dedotto, si può seguire l'associazione Is_registered da STUDENT o da SECTION e poi da SECTION a FACULTY nel diagramma E-R nella Figura 6.16.

6.5.3 Dichiarare un'asserzione di azione

Laddove un'asserzione strutturale riguarda la struttura statica di un'azienda, un'asserzione di azione riguarda gli aspetti dinamici dell'azienda. Le asserzioni di azione impongono vincoli “deve (non deve)” e “dovrebbe (non dovrebbe)” sulla gestione dei dati. Un'asserzione di azione è la proprietà di alcune regole aziendali (chiamate **oggetti ancora**); nel caso di un'azione (per esempio, creare, aggiornare, eliminare o leggere), dichiara come le altre regole aziendali (chiamate **oggetti corrispondenti**) agiscono sull'oggetto ancora.

Alcuni esempi di asserzioni di azione sono i seguenti.

- Un corso (oggetto ancora) deve avere un nome (oggetto corrispondente). In questo esempio, l'azione è l'aggiornamento della proprietà “nome del corso” di un corso.
- Uno studente (oggetto ancora) deve avere una media pari a 2.0 o superiore per Student_GPA (oggetto corrispondente) perché possa laurearsi (azione). In questo esempio, l'oggetto ancora è un'asserzione strutturale; è possibile che l'oggetto ancora sia un'altra asserzione di azione.
- Uno studente non può iscriversi (l'oggetto ancora è l'associazione Is_registered) a un seminario di corso per il quale non esiste corpo docente qualificato (l'oggetto corrispondente è l'associazione Is_qualified).

Tipi di asserzioni di azione

Per semplicità, la Figura 6.15 non mostra i vari tipi di asserzioni di azione esistenti. Per classificare le asserzioni di azione esistono tre modi.

1. Le asserzioni di azione possono essere classificate in base al tipo di risultato derivante dall'asserzione. In quest'ottica, esistono tre tipi di asserzione.
 - a) *Di condizione*, che dichiara se qualcosa è vero, così che sia possibile applicarlo a un'altra regola aziendale. Il terzo esempio di asserzione di azione sopra citato può essere dichiarato come condizione nella forma: “Se un corso ha un corpo docente qualificato, allora gli studenti possono iscriversi a un seminario di quel corso”.
 - b) *Di vincolo di integrità*, che dichiara se qualcosa deve sempre essere vero. Il primo esempio di asserzione di azione sopra citato illustra un vincolo di integrità. Un altro esempio potrebbe essere “La data in cui un corpo docente diventa qualificato per un corso non può essere successiva al semestre nel quale al corpo docente è assegnato quel corso”.
 - c) *Di autorizzazione*, che dichiara un privilegio. Per esempio, solo i direttori di dipartimento (un tipo di utente) possono qualificare un corpo docente perché tenga un corso.
2. Le asserzioni di azione possono essere classificate in base alla forma dell'asserzione. In quest'ottica, esistono tre tipi di asserzione.
 - a) *Di attivazione*, che, se vera, permette o porta all'esistenza dell'oggetto corrispondente. Un esempio di asserzione di attivazione è: “Un corpo docente può essere creato una volta che è qualificato per tenere almeno un corso”.
 - b) *Di temporizzazione*, che abilita (disabilita) o crea (elimina) un'azione. Un esempio di un'asserzione di temporizzazione è: “Quando uno studente ha una media (GPA) superiore a 2.0 e un totale di ore di frequenza superiore a 125, allora può laurearsi”. Si noti che l'azione di laurea non si verifica grazie a questo tipo di azione, ma piuttosto perché l'asserzione di temporizzazione abilita l'azione perché possa verificarsi.

- c) *Di esecuzione*, che comporta l'esecuzione di una o più azioni. Un'azione di esecuzione può essere considerata come un trigger. Un esempio di asserzione di esecuzione è: "Quando uno studente ha una media inferiore a 2.0, viene sottoposto a un esame". Questa azione di esecuzione può dare origine a un attributo di stato dello studente che deve essere aggiornato in base al valore dell'esame.
3. Le asserzioni di azione possono essere classificate in base al rigore dell'asserzione. In quest'ottica, esistono due tipi di asserzioni.
- a) *Di controllo*, che dichiara che qualcosa deve o non deve verificarsi. Tutti gli esempi finora utilizzati rientrano in questa categoria.
 - b) *Di influenza*, che sono le linee guida o gli elementi di interesse per i quali deve avvenire una notifica. Un esempio di asserzione di influenza è: "Quando il numero di studenti iscritti a un seminario supera del 90 per cento la capacità dello stesso, si deve avvertire il direttore di dipartimento". In questa situazione, niente viene controllato (gli studenti possono continuare a iscriversi al seminario e non vengono creati ulteriori seminari), ma l'amministrazione vuole sapere che si è verificata una particolare condizione.

6.5.4 Rappresentazione e applicazione delle regole aziendali

La maggior parte delle aziende ha centinaia, se non migliaia, di regole. Le asserzioni di azione sono state tradizionalmente implementate nella logica procedurale che soggiace ai singoli programmi applicativi in una forma virtualmente irriconoscibile, ingestibile e incoerente. Questo approccio sposta un peso notevole sul programmatore, che deve conoscere tutti i vincoli che un'azione può violare e che deve includere controlli per ciascun tipo di vincolo. Un'omissione, un equivoco o un errore da parte del programmatore lascerà probabilmente la base di dati in uno stato di invalidità.

L'approccio più moderno consiste nel dichiarare le asserzioni a livello concettuale senza specificare come la regola verrà implementata. Si deve quindi creare un linguaggio di specifiche per le regole aziendali. Abbiamo visto che la notazione EER funziona bene quando si tratta di specificare molti tipi di regole aziendali. In effetti la notazione EER è stata inventata per consentire che le regole aziendali potessero essere mostrate in una forma grafica più di quanto non fosse possibile con la notazione E-R, più semplice. Un'alternativa alla notazione grafica può essere una grammatica strutturata (come una forma limitata di una determinata lingua). A prescindere dal tipo di notazione, è comunque auspicabile che il linguaggio delle specifiche delle regole aziendali abbia due caratteristiche.

1. Deve essere relativamente semplice, così che gli utenti non solo capiscano le istruzioni delle regole, ma possano anche definire le regole autonomamente.
2. Deve essere sufficientemente strutturato di modo da poter essere convertito automaticamente nel codice di computer che applica le specifiche.

Nel prossimo paragrafo illustreremo la specifica delle regole aziendali sia mediante l'approccio grafico sia mediante quello della grammatica strutturata. L'approccio grafico è adattato da Ross (1997), un leader nello sviluppo delle specifiche delle regole aziendali.

Esempi di regole aziendali

In questo paragrafo prenderemo in considerazione il modello di dati del programma di lezioni mostrato nella Figura 6.16 aggiungendo due nuove regole aziendali. Al termine del capitolo, si trovano problemi ed esercizi relativi all'aggiunta di regole aziendali supplementari.

Regola aziendale 1. Perché un membro del corpo docente sia assegnato a un seminario di un corso, deve essere qualificato a tenere il corso per il quale il seminario è programmato.

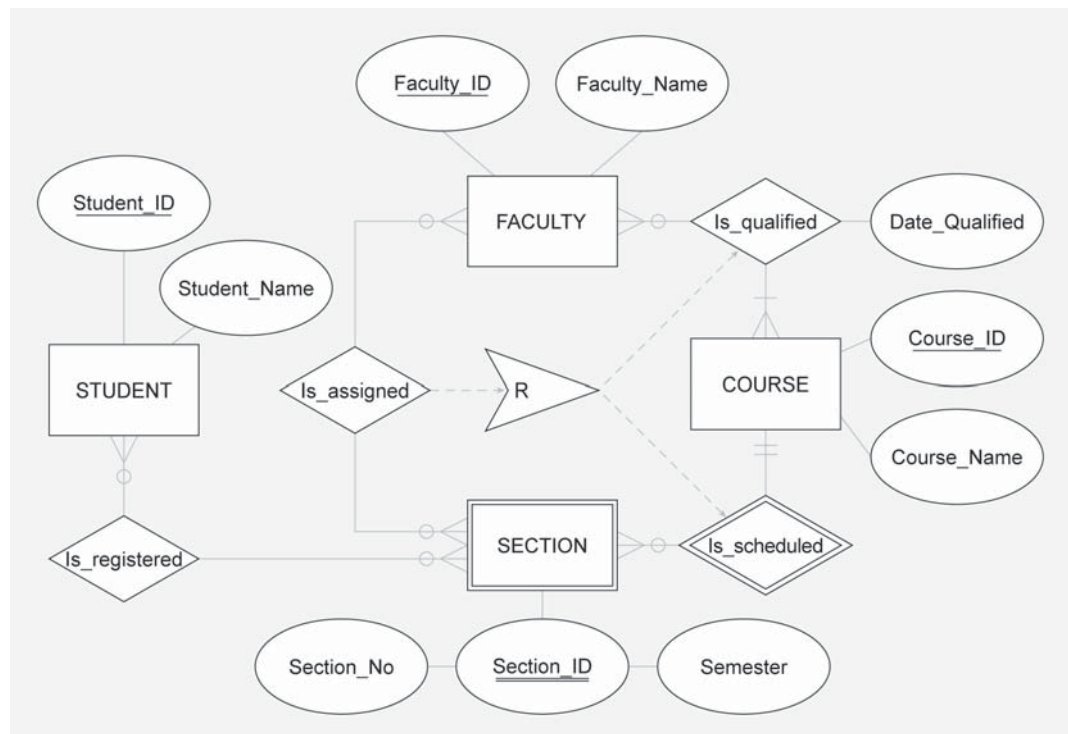
Questa regola fa riferimento a tre tipi di entità mostrati nella Figura 6.16: FACULTY, SECTION e COURSE. La domanda è se un membro del corpo docente possa essere assegnato all'insegnamento in un seminario (di un corso). L'oggetto ancora è quindi l'associazione Is_assigned. Il vincolo non è né sull'insegnante né sul seminario, bensì sull'assegnazione del membro del corpo docente al seminario. La Figura 6.17 mostra una linea tratteggiata tra Is_assigned e il simbolo dell'asserzione di azione. Quali sono gli oggetti corrispondenti in questa regola? Perché un membro del corpo docente venga assegnato a un seminario di corso, sono necessarie due condizioni.

1. Il membro del corpo docente deve essere qualificato per tenere il corso (questa informazione è registrata dall'associazione Is_qualified).
2. Il seminario deve essere programmato per quel corso (questa informazione è registrata dall'associazione Is_scheduled).

In questo caso ci sono quindi due oggetti corrispondenti. Tale fatto è rappresentato dalle linee tratteggiate tra il simbolo dell'asserzione di azione e le due associazioni.

Figura 6.17

Regola aziendale 1. Perché un membro del corpo docente sia assegnato a un seminario di un corso, deve essere qualificato a tenere il corso per il quale il seminario è programmato (la lettera R all'interno del simbolo sta per "Restricted" [limitato], che è una delle numerose opzioni per i vincoli).



Regola aziendale 2. Perché un membro del corpo docente sia assegnato a un seminario di un corso, non deve essere assegnato all'insegnamento in più di tre seminari di corso.

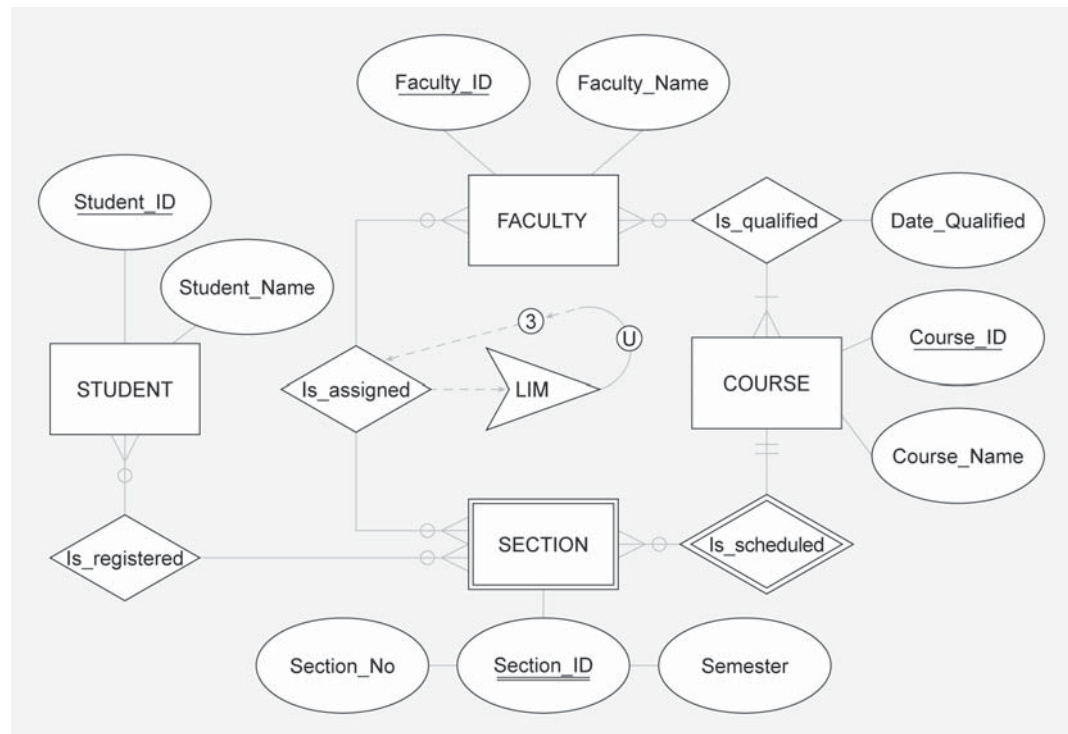
Questa regola impone un limite al numero *totale* dei seminari che un membro del corpo docente può tenere in un dato periodo di tempo. Poiché la regola implica un totale, richiede la modifica della notazione precedente. Come mostrato nella Figura 6.18, l'oggetto ancora è nuovamente l'associazione *Is_assigned*. In questo caso, tuttavia, l'oggetto corrispondente è anche l'associazione *Is_assigned*. Nello specifico, è un conteggio dei seminari totali assegnati al membro del corpo docente. Le lettere "LIM" nel simbolo di asserzione di azione stanno per "limit" (*limite*). La freccia parte dal simbolo e punta a un cerchio con la lettera "U", che sta per "upper" (*superiore*). Il secondo cerchio contiene il numero 3, che è il limite superiore. Il vincolo viene quindi letto come segue: "L'oggetto corrispondente è il conteggio del numero di seminari assegnati al membro del corpo docente, che è un limite superiore pari a tre". Se a un membro del corpo docente sono già stati assegnati tre seminari, verrà rifiutata qualsiasi transazione che tenti di aggiungere un altro seminario.

È possibile implementare regole aziendali come queste utilizzando il linguaggio SQL e memorizzando le regole come parte delle definizioni della base di dati. Un modo per implementare una regola mediante SQL è quello di utilizzare l'istruzione **CREATE ASSERTION**, che è inclusa nella versione più recente del linguaggio (SQL2). Si consideri per esempio la regola aziendale 2. L'istruzione riportata di seguito crea un'asserzione denominata "Overload_Protect":

```
CREATE ASSERTION Overload_Protect
CHECK (SELECT COUNT(*)
FROM ASSIGNED
WHERE Faculty_ID = '12345') <= 3;
```

Figura 6.18

Regola aziendale 2. Perché un membro del corpo docente sia assegnato a un seminario di un corso, non deve essere assegnato all'insegnamento in più di tre seminari di corso.



Questa istruzione controlla se il numero totale dei seminari del corso assegnati a un determinato membro del corpo docente è inferiore o uguale al limite (3). Il sistema di gestione della base di dati ha la responsabilità di garantire che questo vincolo non venga violato.

Al di là degli esempi di questo paragrafo, sono possibili molte altre regole aziendali. Si veda Ross (1997) per un elenco completo con la notazione associata.

6.5.5 Identificazione e verifica delle regole aziendali

In questo capitolo è stata presentata un'ampia gamma di regole aziendali ed è stato mostrato come possono essere rappresentate con una grammatica strutturata o con diagrammi EER ed estensioni. Il compito dell'analista di dati è quello di documentare le specifiche delle regole aziendali e di garantire che queste vengano applicate il più possibile attraverso le tecnologie delle basi di dati. Prima di essere specificate, le regole devono però essere identificate, dopodiché potranno essere verificate prima di essere implementate nella tecnologia.

Un approccio all'identificazione e verifica delle regole aziendale è quello degli *scenari* (Moriarty, 1993). Uno scenario è un brano o uno script che descrive come un'azienda reagisce in una determinata situazione. Uno scenario è simile al concetto di caso d'uso, che è uno strumento importante nella modellazione degli oggetti. Per esempio, uno scenario relativo al diagramma E-R mostrato nella Figura 6.16 potrebbe concentrarsi sugli eventi che riguardano uno studente. Gli eventi includono l'ammissione, l'iscrizione alle lezioni (e il ritiro da queste), l'ottenimento di voti e la laurea. Uno scenario potrebbe seguire il percorso tra questi eventi per un determinato individuo. Infine, potrebbe essere utilizzato per identificare gli elementi (regole aziendali) per il modello di dati o per verificare che il modello di dati gestisca tutte le situazioni possibili per questi eventi (per esempio, uno studente che prova a iscriversi a una lezione inesistente, uno sotto esame che prova a iscriversi o uno laureato che prova a iscriversi).

Lo scenario di uno studente potrebbe essere il seguente: "Missi Davies accede al sistema di iscrizione alle lezioni per iscriversi al prossimo semestre. Missi immette il proprio ID studente e la propria password per avere l'autorizzazione a iscriversi. Missi si sta specializzando in MIS (*Management Information System*) e ha una buona situazione accademica. Vuole essere certa di entrare nei corsi richiesti per la sua specializzazione, perciò cerca nell'elenco dei corsi MIS il corso sulle basi di dati richiesto. Missi scopre che ci sono due seminari di questo corso nel prossimo semestre, ma uno è già completo. Per il sistema di iscrizione on-line, Missi deve quindi iscriversi al seminario ancora aperto. In base al proprio programma di lezioni complessivo, può chiedere al direttore di dipartimento di essere iscritta al seminario già chiuso e di annullare l'iscrizione a quello ancora aperto". Se un insieme di regole aziendali non è ancora stato identificato, questo e altri scenari possono essere utilizzati per identificare termini, fatti e asserzioni di azione da includere nel modello di dati. Se invece esiste già un insieme di regole aziendali, le istruzioni negli scenari possono essere utilizzate per verificare e confermare che le regole aziendali siano in grado di gestire tutte le situazioni possibili.

Riepilogo

Questo capitolo ha descritto come il modello E-R di base sia stato esteso per includere le associazioni supertipo/sottotipo. Un supertipo è un tipo di entità generico che ha un'associazione con uno o più sottotipi. Un sottotipo è un accorpamento di entità in un tipo di entità significativo per l'azienda. Per esempio, il tipo di entità PERSON viene frequentemente modellato come sottotipo. I sottotipi di PER-

SON possono includere EMPLOYEE, VOLUNTEER e CLIENT. I sottotipi ereditano gli attributi e le associazioni associati al loro supertipo.

Nella modellazione dei dati, le associazioni supertipo/sottotipo dovrebbero solitamente essere considerate con una o entrambe queste condizioni: la prima è che ci sono attributi che si applicano ad alcune, ma non a tutte, le istanze di un tipo di entità; la seconda è che le istanze di un sottotipo partecipano a un'associazione univoca per quel sottotipo.

Le tecniche di generalizzazione e specializzazione sono guide importanti nello sviluppo delle associazioni supertipo/sottotipo. La generalizzazione è il processo bottom-up di definizione di un tipo di entità generalizzato da un insieme di tipi di entità più specializzati. La specializzazione è il processo top-down di definizione di uno o più sottotipi che sono già stati definiti.

La notazione EER consente di illustrare le regole aziendali importanti che si applicano alle associazioni supertipo/sottotipo. Il vincolo di completezza permette di specificare se un'istanza di un supertipo debba essere anche membro di almeno un sottotipo. Esistono due situazioni: con la specializzazione totale, un'istanza di un supertipo deve essere membro di almeno un sottotipo; con la specializzazione parziale, un'istanza di un supertipo può essere o non essere membro di un sottotipo. Il vincolo di separazione consente di specificare se un'istanza di un supertipo possa essere contemporaneamente membro di due o più sottotipi. Anche in questo caso esistono due situazioni: con la regola della separazione, un'istanza può essere membro di un solo sottotipo in un determinato momento; con la regola della sovrapposizione, un'istanza di entità può essere contemporaneamente membro di due o più sottotipi.

Una discriminante di sottotipo è un attributo di un supertipo i cui valori determinano a quale sottotipo o sottotipi appartiene un'istanza. Una gerarchia supertipo/sottotipo è una disposizione gerarchica di supertipi e sottotipi, nella quale un sottotipo ha un solo supertipo.

La notazione E-R prevede altre estensioni oltre quelle delle associazioni supertipo/sottotipo. Una delle tante e utili estensioni è l'aggregazione, che rappresenta il modo in cui alcune entità fanno parte di altre entità (per esempio, un PC è composto da un'unità disco, da una RAM, da una scheda madre e così via). Per ragioni di spazio, non abbiamo qui trattato tali estensioni. La maggior parte di esse, come l'aggregazione, è anche parte della modellazione dei dati orientata agli oggetti.

I diagrammi E-R possono diventare grandi e complessi e comprendere centinaia di entità. Molti utenti e manager non hanno la necessità di vedere tutte le entità, le associazioni e gli attributi per comprendere la parte della base di dati che più li riguarda. L'accorpamento delle entità è un modo per convertire una parte del modello di dati entità-associazione in una vista a macro-livello degli stessi dati. Un cluster di entità è un insieme di uno o più tipi di entità e delle associazioni associate accorpati in un unico tipo di entità astratta. Più cluster di entità e di associazioni associate possono essere ulteriormente accorpati in un cluster di entità maggiore. L'accorpamento delle entità è quindi una tecnica di scomposizione gerarchica. Accorpare le entità e le associazioni è possibile predisporre un diagramma E-R che consenta di focalizzarsi sui dettagli del modello che più contano in un dato compito di modellazione dei dati.

Una regola aziendale è un'istruzione che definisce o vincola alcuni aspetti del lavoro. È concepita per definire la struttura aziendale o per controllare l'influenza del comportamento dell'azienda. Le regole possono essere divise in tre categorie principali: derivazioni, asserzioni strutturali e asserzioni di azione. Le asserzioni strutturali definiscono la struttura statica dell'azienda, mentre quelle di azione sono regole che vincolano le operazioni dinamiche della stessa. Un'asserzione strutturale è un termine o un fatto. Un termine è una definizione di un concetto riguardante l'azienda, mentre un fatto è un'istruzione che implica l'associazione di due o più termini. I fatti possono essere di base (o fondamentali) o derivati da manipolazioni matematiche o logiche di altri fatti; possono avere a che fare con un attributo, un'associazione o un'associazione supertipo/sottotipo tra fatti.

Le asserzioni di azione dichiarano che le azioni (come creare, eliminare, aggiornare e leggere) su una determinata regola aziendale, chiamata oggetto ancora, vengano vincolate da altri oggetti corrispondenti. I vincoli possono essere di tipo if ... then ... else, o possono essere condizioni, regole di integrità o privilegi di autorizzazione. Un'asserzione di azione può permettere l'esistenza dell'oggetto corrispondente, abilitando o disabilitando un'azione, o facendo in modo che l'azione venga intrapresa. Altri tipi di asserzioni di azione sono semplici linee guida che attivano la notifica di condizioni speciali.

Ripasso del capitolo

Parole chiave

Asserzione di azione	Fatto derivato	Regola della separazione	Sottotipo
Asserzione strutturale	Generalizzazione	Regola della sovrapposizione	Specializzazione
Azione	Gerarchia supertipo/sottotipo	Regola della specializzazione parziale	Supertipo
Cluster di entità	Modello entità-associazione esteso (EER)	Regola della specializzazione totale	Vincolo di completezza
Derivazione	Oggetto ancora		Vincolo di separazione
Discriminante di sottotipo	Oggetto corrispondente		
Ereditarietà di attributo			

Domande di ripasso

- Definire ciascuno dei seguenti termini chiave:
 - supertipo
 - sottotipo
 - specializzazione
 - cluster di entità
 - asserzione strutturale
 - oggetto ancora
 - discriminante di sottotipo
 - regola di specializzazione totale
 - generalizzazione
 - regola della separazione
 - regola della sovrapposizione
 - asserzione di azione
- Unire i seguenti termini alle rispettive definizioni:

___ supertipo; ___ cluster di entità; ___ asserzione strutturale;
 ___ sottotipo; ___ specializzazione; ___ oggetto ancora;
 ___ azione; ___ discriminante di sottotipo;
 ___ ereditarietà di attributo; ___ regola della sovrapposizione;
 ___ oggetto corrispondente; ___ fatto derivato.

 - sottoinsieme del supertipo
 - si riferisce a istanze di un singolo tipo di entità
 - entità che appartiene a due sottotipi
 - sottotipo che prende gli attributi del supertipo
 - regola in base alla quale le azioni sono limitate
 - creare, eliminare, aggiornare o leggere
 - calcolato utilizzando un algoritmo
 - tipo di entità generalizzata
- struttura statica di un'azienda
 - creare sottotipi di un tipo di entità
 - un gruppo di tipi di entità e associazioni associate
 - trova il sottotipo di destinazione per un'entità
- Confrontare i seguenti termini:
 - supertipo; sottotipo
 - tipo di entità; istanza di entità
 - generalizzazione; specializzazione
 - regola della separazione; regola della sovrapposizione
 - asserzione strutturale; asserzione di azione
 - regola della specializzazione parziale/regola della specializzazione totale
- Citare due condizioni che indicano quando un progettista dovrebbe prendere in considerazione l'uso di associazioni supertipo/sottotipo.
- Spiegare la ragione del raggruppamento di entità.
- Fornire un esempio (diverso da quelli presenti in questo capitolo) di un'associazione supertipo/sottotipo.
- Fornire un esempio (diverso da quelli presenti in questo capitolo) di quanto segue:
 - asserzione strutturale
 - asserzione di azione
- Cos'è l'ereditarietà di attributo? Perché è importante?
- Fornire un esempio di quanto segue:
 - associazione supertipo/sottotipo a cui si applica la regola della separazione
 - associazioni supertipo/sottotipo a cui si applica la regola della sovrapposizione

10. Quali tipi di regole aziendali vengono solitamente rappresentate in un diagramma EER?
11. Qual è lo scopo di una discriminante di sottotipo?

Problemi ed esercizi

1. Esaminare la gerarchia del digramma EER dell'università (Figura 6.10). In quanto studenti, si è un'istanza di uno dei sottotipi UNDERGRAD STUDENT o GRADUATE STUDENT. Elencare i nomi di tutti gli attributi che si applicano allo studente e, per ciascun attributo, registrare il valore di dati che si applicano allo studente.
2. Aggiungere una discriminante di sottotipo per ognuno dei sottotipi mostrati nella Figura 6.10. Mostrare i valori di discriminante che assegnano istanze a ciascun sottotipo. Utilizzare i seguenti nomi e valori di discriminante.
 - a. PERSON: Person_Type (Employee?, Alumnus?, Student?)
 - b. EMPLOYEE: Employee_Type (Faculty, Staff)
 - c. STUDENT: Student_Type (Grad, Undergrad)
3. Disegnare un diagramma EER per il seguente problema. Un'azienda non-profit dipende da alcuni tipi di persone diverse per la sua operatività. L'azienda è interessata ai seguenti attributi per tutte queste persone: SSN, Name, Address, City/State/Zip e Telephone. Tre tipi di persone sono particolarmente interessanti: impiegati, volontari e donatori. Gli impiegati hanno solo un attributo Date_Hired, mentre i volontari hanno solo un attributo Skill. I donatori hanno solo un'associazione (chiamata Donates) con il tipo di entità Item. Un donatore deve aver donato uno o più articoli e un articolo può non avere donatori o può avere uno o più donatori.
 Ci sono altre persone, diverse dagli impiegati, dai volontari e dai donatori, che interessano all'azienda, cosicché una persona non deve appartenere obbligatoriamente a uno di questi tre gruppi. D'altra parte, in un dato momento una persona può appartenere a due o più di questi gruppi (per esempio, impiegati e donatori).
4. Aggiungere una discriminante di sottotipo (chiamata Person_Type) al diagramma EER creato nel problema ed esercizio 3.
5. Un'agenzia di noleggio auto classifica i veicoli che noleggia in quattro categorie: compatti, di dimensioni medie, di dimensioni grandi e sportivi. L'agenzia vuole registrare i dati seguenti per tutti i veicoli: Vehicle_ID, Make, Model, Year e Color. Non ci sono attributi univoci per nessuna delle quattro classi di veicoli. Il tipo di entità veicolo ha un'associazione (chiamata Rents) con un tipo di entità cliente. Nessuna delle quattro classi di veicoli ha un'associazione univoca con un tipo di entità. Si può ipotizzare di creare un'associazione supertipo/sottotipo per questo problema? Perché sì o no?
6. Durante un ritiro che avviene nel fine settimana, il tipo di entità PERSON ha tre sottotipi: CAMPER, BIKER e RUNNER. Disegnare un segmento separato del diagramma EER per ciascuna delle seguenti situazioni:

- a. In un dato momento, una persona deve essere esattamente uno di questi sottotipi.
 - b. Una persona può essere o non essere uno di questi sottotipi. Tuttavia, una persona che è uno dei sottotipi non può essere contemporaneamente uno degli altri sottotipi.
 - c. Una persona può essere o non essere uno di questi sottotipi. D'altra parte, una persona può essere contemporaneamente uno qualsiasi di questi due (o anche tre) sottotipi.
 - d. In un dato momento, una persona deve essere almeno uno di questi sottotipi.
7. Una banca ha tre tipi di conti: correnti, di risparmio e di mutuo. Quelli che seguono sono gli attributi per ciascun tipo di conto:
 CHECKING: Acct_No, Date_Opened, Balance, Service_Charge
 SAVINGS: Acct_No, Date_Opened, Balance
 LOAN: Acct_No, Date_Opened, Balance, Interest_Rate, Payment
 Si assuma che ciascun conto bancario debba essere un membro esattamente di uno di questi sottotipi. Utilizzando la generalizzazione, sviluppare un segmento di modello EER per rappresentare questa situazione. Si ricordi di includere una discriminante di sottotipo.
 8. Fare riferimento al diagramma EER dell'impiegato mostrato (Figura 6.2) ed eseguire questo esercizio. Considerare tutti i presupposti che si ritengono necessari.
 - a. Sviluppare una definizione di esempio per ciascun tipo di entità, attributo e associazione del diagramma.
 - b. Sviluppare un'asserzione di azione di vincolo d'integrità di esempio per tutti gli attributi nella figura.
 9. Fare riferimento al modello di dati per la programmazione delle lezioni (Figura 6.16). Aggiungere la notazione grafica a questa figura per esprimere ciascuna delle seguenti regole aziendali.
 - a. Per un seminario del corso che deve essere programmato, deve esserci un membro del corpo docente che è qualificato per insegnare in quel corso (*suggerimento*: si veda la Figura 6.17).
 - b. Per uno studente che deve iscriversi a un seminario di corso, quello studente non può essere iscritto a un totale di più di sei seminari di corso in un dato momento (*suggerimento*: si veda la Figura 6.18).
 10. Fare riferimento al diagramma EER per le associazioni di un ospedale (Figura 6.3). Aggiungere la notazione per esprimere la seguente regola aziendale: "Un paziente interno può essere assegnato a un letto solo se il paziente è stato assegnato a un medico responsabile". In questo esempio, quali sono:
 - a. l'oggetto ancora?
 - b. gli oggetti corrispondenti?
 11. Si consideri la seguente regola "aziendale": "Uno studente può assistere a un concerto solo se ha finito i compiti".
 - a. Disegnare un segmento di diagramma EER che rappresenti le entità e le associazioni implicate in questa regola.
 - b. Aggiungere la notazione adatta per esprimere la regola aziendale.

- c. Identificare questi elementi: i) oggetto ancora; ii) oggetti corrispondenti.
12. Fare riferimento al diagramma EER per i pazienti (Figura 6.3) ed eseguire questo esercizio. Considerare tutti i presupposti che si ritengono necessari.
 - a. Sviluppare definizioni di esempio per ciascun tipo di entità, attributo e associazione del diagramma.
 - b. Sviluppare inserimenti di asserzioni di azione di vincolo d'integrità di esempio per tutti gli attributi nel diagramma.
13. La Figura 6.13 mostra i cluster di entità per il diagramma E-R di Pine Valley Furniture Company. Nella Figura 6.13, spiegare quanto segue.
 - a. Perché la cardinalità minima vicino a CUSTOMER per l'associazione Does_business_in è 0.
 - b. Quali dovrebbero essere gli attributi di ITEM (fare riferimento alla Figura 5.22).
 - c. Quali dovrebbero essere gli attributi di MATERIAL (fare riferimento alla Figura 5.23).
10. Fare riferimento alla risposta al problema ed esercizio 6 nel Capitolo 5. Sviluppare cluster di entità per questo diagramma E-R e ridisegnare il diagramma utilizzando i cluster di entità. Spiegare perché si è scelto di utilizzare quei cluster di entità.
11. Fare riferimento alla risposta al problema ed esercizio 11 nel Capitolo 5. Sviluppare cluster di entità per questo diagramma E-R e ridisegnare il diagramma utilizzando i cluster di entità. Spiegare perché si è scelto di utilizzare quei cluster di entità.

Esercitazioni

1. Intervistare un amico o un membro della propria famiglia per cercare di ottenere alcuni esempi semplici degli elementi riportati di seguito rispetto al posto di lavoro.
 - a. *Associazioni supertipo/sottotipo*. Spiegare il significato di questo termine alla persona che si intervista e fornire un esempio comune, come PROPERTY: RESIDENTIAL, COMMERCIAL o BONDS: CORPORATE, MUNICIPAL. Utilizzare le informazioni fornite dalla persona per costruire un segmento di diagramma EER per poi presentarglielo. Rivedere il diagramma, se necessario, fino a quando non risulta soddisfacente.
 - b. *Regole aziendali*. Fare qualche esempio di regola aziendale alla persona che si intervista utilizzando quelli contenuti in questo capitolo (Figure 6.17 e 6.18). Quando la persona fornisce una regola relativa al proprio ambiente, ristrutturarla, se necessario, per adeguarla alla sintassi esposta nel capitolo.

2. Visitare due piccole aziende locali, una nel settore dei servizi e una della produzione manifatturiera. Intervistare gli impiegati di queste aziende per ottenere esempi sia di associazioni supertipo/sottotipo sia di regole aziendali (per esempio, "Un cliente può restituire la merce solo presentando uno scontrino valido"). In quali di questi ambienti è più facile trovare esempi di questo costruito? Perché?
3. Chiedere a un amministratore o analista di sistema o di basi di dati di un'azienda locale di vedere un diagramma EER (o E-R) per una delle basi di dati principali dell'azienda. L'azienda modella le associazioni supertipo/sottotipo? Se sì, quale notazione viene utilizzata? Lo strumento CASE impiegato dall'azienda supporta queste associazioni? Quali tipi di regole aziendali sono inclusi nella modellazione EER? Come vengono rappresentate le regole e come e dove sono memorizzate?
4. Leggere il riepilogo delle regole aziendali pubblicato dal GUIDE Business Rules Project (1997) e l'articolo di Gottesdiener (1997). Cercare sul Web altre informazioni sulle regole aziendali. Scrivere quindi un riepilogo esecutivo di tre pagine delle direzioni correnti delle regole e del loro impatto potenziale sullo sviluppo e la manutenzione dei sistemi.

Bibliografia

- Elmasri, R. e Navathe, S. B., 1994, *Fundamentals of Database Systems*, Menlo Park, CA, Benjamin/Cummings.
- Gottesdiener, E., 1997, "Business Rules Show Power Promise", *Application Development Trends* 4 (3) (marzo), pp. 36-54.
- "GUIDE Business Rules Project", Final Report, revisione, 1.2, ottobre 1997.
- Moriarty, T., "Using Scenarios in Information Modeling: Bringing Business Rules to Life", *Database Programming & Design* 6 (8), pp. 65-67.
- Ross, R. G., 1997, *The Business Rules Book*, versione 4, Boston, Business Rules Solution, Inc.
- Teorey, T., 1999, *Database Modeling & Design*, San Francisco, CA, Morgan Kauffman Publishers.

Per approfondire

- Schmidt, B., 1997, "A Taxonomy of Domains", *Database Programming & Design* 10 (9) (settembre), pp. 95, 96, 98, 99.
- Von Halle, B., 1996, "Object-Oriented Lessons", *Database Programming & Design* 9 (1) (gennaio), pp. 13-16.
- Von Halle, B. e Kaplan, R., "Is IT Falling Short?", *Database Programming & Design* 10 (6) (giugno), pp. 15-17.

Project Case



MOUNTAIN VIEW COMMUNITY HOSPITAL

Descrizione del progetto

In quanto azienda estesa di servizi, Mountain View Community Hospital dipende da un gran numero di persone per la sua operatività. Ci sono quattro gruppi di persone dalle quali l'ospedale dipende maggiormente: impiegati, medici, pazienti e volontari. Naturalmente alcuni attributi comuni sono condivisi da tutte queste persone: Person_ID (identificatore), Name, Address, City/State/Zip, Birth_Date e Phone.

Ognuno dei quattro gruppi di persone ha almeno un attributo univoco esclusivo. Gli impiegati hanno Data_Hired, i volontari hanno Skill, i medici hanno Specialty e Pager# (numero del cercapersona) e i pazienti hanno Contact_Date (la data del loro primo contatto con l'ospedale).

Altre persone nella comunità dell'ospedale non appartengono ad alcuno di questi gruppi (il loro numero è relativamente piccolo). Tuttavia, una determinata persona può appartenere a due o più di questi gruppi in un dato momento (per esempio, può essere paziente e volontario).

Ogni paziente ha un, e solo un, medico responsabile. Un dato medico può non essere responsabile di un paziente in un dato momento o può essere responsabile di uno o più pazienti. I pazienti sono suddivisi in due gruppi: interni ed esterni. Ogni paziente interno ha un attributo Date_Admitted. Ogni paziente esterno è programmato per una o zero visite. L'entità VISIT ha due attributi: Date (identificatore parziale) e Comments. Si noti che un'istanza di visita non può esistere senza un'entità possessore di paziente esterno.

Gli impiegati sono suddivisi in tre gruppi: infermieri, staff e tecnici. Solo gli infermieri hanno l'attributo Certificate, che indica la qualifica (RN, LPN e così via). Solo lo staff ha l'attributo Job_Class e solo i tecnici hanno l'attributo Skill. Ogni infermiere è assegnato a una (e solo una) unità di cura. Esempi di unità di cura sono la maternità, il pronto soccorso e la cardiologia. Attributi dell'unità di cura sono Name (identificatore) e Location. A un'unità di cura possono essere assegnati uno o più infermieri. Inoltre, uno degli infermieri assegnati a quell'unità di cura è nominato nurse_in_charge. Un infermiere non può essere nurse_in_charge di un'unità di cura a meno che non abbia un diploma RN. Ogni tecnico è assegnato a uno o più

laboratori. Gli attributi di un laboratorio includono Name (identificatore) e Location. A un laboratorio deve essere assegnato almeno un tecnico e può avere un numero di qualsiasi di tecnici assegnati. Possono non esserci letti assegnati a un'unità di cura o a un'unità di cura possono essere assegnati uno o più letti (fino a un numero qualsiasi). L'unico attributo del letto è Bed_ID (identificatore). Bed_ID è un attributo composto, con i componenti Bed# e Room#. Ogni paziente interno deve essere assegnato a un letto. A un letto può essere o non essere assegnato un paziente interno in un dato momento.

Domande sul progetto

1. La capacità di modellare associazioni supertipo/sottotipo può essere importante in un ambiente come Mountain View Community Hospital? Perché?
2. Il paradigma delle regole aziendali e la possibilità di definire, implementare e mantenere le regole aziendali possono essere utilizzati come un vantaggio competitivo in un ambiente come Mountain View Community Hospital? Perché sì o no?
3. Nella descrizione delle esigenze di dati di questo segmento di progetto appare qualche entità debole?
4. Si possono ipotizzare delle regole aziendali, diverse da quelle descritte esplicitamente nel project case, da utilizzare nell'ambiente di un ospedale?

Esercizi sul progetto

1. Disegnare un diagramma EER per rappresentare accuratamente questo insieme di esigenze, seguendo con attenzione la notazione presentata in questo capitolo.
2. Sviluppare una definizione per ciascuno dei tipi di oggetti nel diagramma EER che è stato costruito nell'esercizio sul progetto 1. Se possibile, contattare qualche membro dell'ospedale; in caso contrario, considerare qualche presupposto ragionevole in base a quanto già si conosce e in base alla propria esperienza.
 - a. Tipi di entità
 - b. Attributi
 - c. Associazioni

3. L'asserzione "Un infermiere non può essere nurse_in_charge di un'unità di cura a meno che non abbia un diploma RN" dovrebbe essere considerata come asserzione di una regola aziendale. Rispondere alle seguenti domande.
 - a. Qual è l'oggetto àncora?
 - b. Qual è l'oggetto corrispondente (o gli oggetti, se ce n'è più di uno)? È un'entità, un'associazione o qualche altro oggetto?
 4. Confrontare il diagramma EER sviluppato in questo capitolo con il diagramma E-R sviluppato nel Capitolo 5 nell'esercizio sul progetto 2. Quali sono le differenze tra questi due diagrammi? Perché ci sono differenze?
 5. Unire il diagramma dell'esercizio sul progetto 4 e il diagramma dell'esercizio sul progetto 2 nel Capitolo 5 in un unico diagramma. Spiegare le decisioni prese durante l'unione.
-



Parte quarta

Progettazione delle basi di dati



CAPITOLO

7

Progettazione logica delle basi di dati e modello relazionale

Al termine di questo capitolo sarete in grado di fare quanto segue.

1. Definire i seguenti termini chiave: relazione, chiave primaria, chiave composta, chiave esterna, nullo, regola di integrità dell'entità, vincolo di integrità referenziale, relazione ben strutturata, anomalia, chiave esterna ricorsiva, normalizzazione, forma normale, dipendenza funzionale, determinante, chiave candidata, prima forma normale, seconda forma normale, dipendenza funzionale parziale, terza forma normale, dipendenza transitiva, sinonimo, alias, omonimo e chiave aziendale.
2. Elencare cinque proprietà delle relazioni.
3. Definire due proprietà che sono essenziali per una chiave candidata.
4. Fornire una definizione concisa di prima forma normale, seconda forma normale e terza forma normale.
5. Descrivere brevemente quattro problemi che possono sorgere quando si uniscono le relazioni.
6. Trasformare un diagramma ER (o EER) in un insieme di relazioni logicamente equivalenti.
7. Creare tabelle relazionali che incorporano i vincoli di integrità di entità e di integrità referenziale.
8. Utilizzare la normalizzazione per scomporre una relazione che contiene anomalie in relazioni ben strutturate.

Introduzione

In questo capitolo descriveremo la progettazione logica di una base di dati, ponendo particolare attenzione sul modello di dati relazionale. La progettazione logica di una base di dati è il processo di trasformazione del modello di dati concettuale (descritto nei Capitoli 5 e 6) in un modello di dati logico. Sebbene esistano altri tipi di modelli, ci sono due ragioni per enfatizzare quello relazionale in questo capitolo. Innanzitutto, il modello di dati relazionale viene utilizzato più diffusamente nelle applicazioni di basi di dati moderne. In secondo luogo, alcuni dei principi della progettazione logica per il modello relazionale si applicano anche ad altri modelli logici.

Nei capitoli precedenti abbiamo introdotto informalmente il modello di dati relazionale attraverso esempi semplici. In questo capitolo definiremo i termini e i concetti importanti per questo modello. Per comodità, quando faremo riferimento al modello di dati relazionale utilizzeremo spesso la forma abbreviata *modello relazionale*. Successivamente, descriveremo e illustreremo il processo di trasforma-

zione di un modello E-R nel modello relazionale. Oggigiorno molti strumenti CASE supportano questa trasformazione; tuttavia, è importante capire i principi e le procedure che soggiacciono al modello. Descriveremo quindi nel dettaglio i concetti della normalizzazione. La normalizzazione, che è il processo di progettazione di relazioni ben strutturate, è un componente importante della progettazione logica per il modello relazionale. Infine, descriveremo come unire le relazioni evitando contemporaneamente le trappole comuni di questo processo.

L'obiettivo di una progettazione logica di una base di dati è quello di tradurre il progetto concettuale (che rappresenta le esigenze di un'azienda rispetto ai dati) in un progetto logico che possa essere implementato in un sistema scelto di gestione della base di dati. Le basi di dati che ne risultano devono soddisfare le esigenze dei clienti riguardo alla condivisione dei dati, alla flessibilità e alla facilità di accesso. I concetti presentati in questo capitolo sono essenziali per capire il processo di sviluppo di una base di dati.

7.1 Il modello di dati relazionale

Il modello di dati relazionale è stato introdotto per la prima volta nel 1970 da E. F. Codd, allora membro di IBM (Codd, 1970). Due progetti di ricerca iniziali erano stati avviati per verificare la fattibilità del modello di dati relazionale e per sviluppare i sistemi prototipo. Il primo di questi progetti, realizzato presso il San Jose Research Laboratory di IBM, portò allo sviluppo di System R (un DBMS-RDBMS relazionale prototipo) verso la fine degli anni Settanta. Il secondo progetto, realizzato presso la University of California a Berkeley, portò allo sviluppo di Ingres, un RDBMS orientato all'università. I prodotti RDBMS commerciali di numerosi produttori iniziarono ad apparire più o meno nel 1980 (si visiti il sito Web di questo libro per trovare qualche collegamento a produttori di RDBMS e di altri DBMS). Oggigiorno i RDBMS sono diventati la tecnologia dominante per la gestione delle basi di dati e ci sono letteralmente centinaia di prodotti RDBMS per computer che vanno dai PC ai mainframe.

7.1.1 Definizioni di base

Il modello di dati relazionale rappresenta i dati sotto forma di tabelle. Si basa su una teoria matematica e ha quindi dei solidi fondamenti teorici. Tuttavia, sono sufficienti solo pochi semplici concetti per descrivere questo tipo di modello, che è quindi facilmente comprensibile e utilizzabile anche da chi non ha familiarità con la teoria su cui si basa. Il modello di dati relazionale è costituito dai tre componenti seguenti (Fleming e von Halle, 1989).

1. *Struttura dei dati.* I dati sono organizzati sotto forma di tabelle con righe e colonne.
2. *Manipolazione dei dati.* Operazioni potenti (utilizzando il linguaggio SQL) vengono utilizzate per manipolare i dati memorizzati nelle relazioni.
3. *Integrità dei dati.* Vengono incluse alcune funzioni per specificare le regole aziendali che mantengono l'integrità dei dati quando questi sono manipolati.

Discuteremo la struttura dei dati e l'integrità dei dati in questo paragrafo. La manipolazione dei dati verrà trattata nei Capitoli 9 e 10.

Struttura dei dati relazionale

Una **relazione** è una tabella di dati denominata e a due dimensioni. Ogni relazione (o tabella) è costituita da un insieme di colonne denominate e da un numero arbitrario di righe senza nome. Un attributo, coerentemente con la sua definizione data nel Capitolo 3, è una colonna denominata di una relazione. Ogni riga di una relazione corrisponde a un record che contiene valori (attributi) di dati per una singola entità. La Figura 7.1 mostra un esempio di una relazione chiamata EMPLOYEE1. Questa relazione contiene i seguenti attributi che descrivono gli impiegati: Emp_ID, Name, Dept_Name e Salary. Le cinque righe della tabella corrispondono a cinque impiegati. È importante capire che i dati di esempio nella Figura 7.1 sono concepiti per illustrare la struttura della relazione EMPLOYEE1 e non sono parte della relazione stessa. Anche se si aggiunge un'altra riga di dati, si avrà sempre la stessa relazione EMPLOYEE1. Nemmeno l'eliminazione di una riga modifica la relazione. In effetti, anche se si eliminano tutte le righe mostrate nella figura, la relazione EMPLOYEE1 continuerà a esistere. La Figura 7.1 è un'istanza della relazione EMPLOYEE1.

Figura 7.1
La relazione EMPLOYEE1
con alcuni dati di esempio.

EMPLOYEE1			
Emp_ID	Name	Dept_Name	Salary
100	Margaret Simpson	Marketing	48,000
140	Allen Beeton	Accounting	52,000
110	Chris Lucero	Info Systems	43,000
190	Lorenzo Davis	Finance	55,000
150	Susan Martin	Marketing	42,000

La struttura di una relazione può essere espressa con una notazione nella quale il nome di una relazione è seguito (tra parentesi) dai nomi degli attributi nella relazione. Nel caso di EMPLOYEE1, si dovrebbe avere:

EMPLOYEE1(Emp_ID,Name,Dept_Name)

Chiavi relazionali

Occorre essere in grado di memorizzare e recuperare una riga di dati in una relazione in base ai valori di dati memorizzati in quella riga. Per raggiungere questo obiettivo, ogni relazione deve avere una chiave primaria. Una **chiave primaria** è un attributo (o una combinazione di attributi) che identifica in modo univoco ogni riga in una relazione. Indicheremo una chiave primaria sottolineando il nome dell'attributo. Per esempio, la chiave primaria per la relazione EMPLOYEE1 è Emp_ID. Si noti che nella Figura 7.1 questo attributo appare sottolineato. Nella notazione, la relazione si esprime come segue:

EMPLOYEE1(Emp_ID,Name,Dept_Name,Salary)

Il concetto di chiave primaria è correlato al termine “identificatore” definito nel Capitolo 5. Gli stessi attributi indicati come identificatori di entità in un diagramma E-R possono essere gli stessi attributi che compongono la chiave primaria per la relazione che rappresenta quella entità. Esistono alcune eccezioni; per esempio, le entità associative non hanno un identificatore e l'identificatore di un'entità debole forma solo parte di una chiave primaria di entità debole. Inoltre possono esserci più attributi di un'entità che possono servire come chiave primaria della relazione associata. Tutte queste situazioni verranno illustrate successivamente in questo capitolo.

Una **chiave composta** è una chiave primaria costituita da più di un attributo. Per esempio, la chiave primaria per una relazione **DEPENDENT** potrebbe essere costituita dalla combinazione di **Emp_ID** e **Dependent_Name**. Mostriamo numerosi esempi di chiavi composte successivamente nel capitolo.

Spesso può essere necessario rappresentare l'associazione tra due tabelle o relazioni. Per farlo, si possono utilizzare le chiavi esterne. Una **chiave esterna** è un attributo (eventualmente composto) in una relazione di una base di dati che serve come chiave primaria di un'altra relazione nella stessa base di dati. Per esempio, si considerino le relazioni **EMPLOYEE1** e **DEPARTMENT**:

```
EMPLOYEE1(Emp_ID, Name, Dept_Name, Salary)
DEPARTMENT(Dept_Name, Location, Fax)
```

L'attributo **Dept_Name** è una chiave esterna in **EMPLOYEE1**, che permette a un utente di associare qualsiasi impiegato alla funzione a cui è assegnato. Alcuni autori sottolineano il fatto che un attributo è una chiave esterna utilizzando una sottolineatura tratteggiata, come in:

```
EMPLOYEE1(Emp_ID, Name, Dept_Name, Salary)
```

Forniremo numerosi esempi di chiavi esterne nel resto del capitolo e discuteremo le proprietà delle chiavi esterne nel paragrafo sull'integrità referenziale.

Proprietà delle relazioni

Abbiamo definito le relazioni come tabelle di dati a due dimensioni. Non tutte le tabelle sono però relazioni. Le relazioni hanno numerose proprietà che le distinguono dalle tabelle non relazionali, e che elenchiamo di seguito.

1. Ogni relazione (o tabella) in una base di dati ha un nome univoco.
2. Un'immissione all'intersezione di ogni riga e colonna è atomica (o a valore singolo).
3. Ogni riga è unica; in una relazione non esistono due righe identiche.
4. Ogni attributo (o colonna) in una tabella ha un nome univoco.
5. La sequenza delle colonne (da sinistra a destra) non è significativa. Le colonne di una relazione possono essere scambiate tra loro senza modificare il significato o l'uso della relazione.
6. La sequenza delle righe (dall'alto al basso) non è significativa. Come nel caso delle colonne, le righe di una relazione possono essere scambiate tra loro o essere memorizzate in qualsiasi ordine.

Eliminare gli attributi multivalore dalle tabelle

La seconda proprietà delle relazioni stabilisce che non ci possono essere attributi multivalore in una relazione. Una tabella che contiene uno o più attributi multivalore non è quindi una relazione. La Figura 7.2a mostra i dati degli impiegati della relazione **EMPLOYEE1** estesa fino a includere i corsi che questi possono aver frequentato. Poiché un dato impiegato può aver partecipato a più di un corso, gli attributi **Course_Title** e **Data_Completed** sono attributi multivalore. Per esempio, l'impiegato **Emp_ID 100** ha partecipato a due corsi. Se un impiegato non ha partecipato ad alcun corso, i valori di attributo **Course_Title** e **Data_Completed** saranno nulli (si veda l'impiegato **Emp_ID 190** per un esempio).

Nella figura si mostra come eliminare gli attributi multivalore nella Figura 7.2b inserendo i valori di dati importanti nelle celle precedentemente vuote nella Figura 7.2a. Come risultato, la tabella nella Figura 7.2b ha solo attributi a valore singolo e soddisfa ora la proprietà atomica delle relazioni. A questa relazione viene dato il nome **EMPLOYEE2** per distinguerla da **EMPLOYEE1**. Tuttavia, come si vedrà di seguito, questa nuova relazione ha alcune proprietà non desiderate.

Figura 7.2

Eliminare gli attributi multivalore.
(a) Tabella con gruppi che si ripetono.

<u>Emp_ID</u>	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
				Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	SPSS	1/12/200X
				C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/16/200X
				Java	8/12/200X

(b) Relazione EMPLOYEE2.

EMPLOYEE2					
<u>Emp_ID</u>	Name	Dept_Name	Salary	<u>Course_Title</u>	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	SPSS	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

7.1.2 Base di dati di esempio

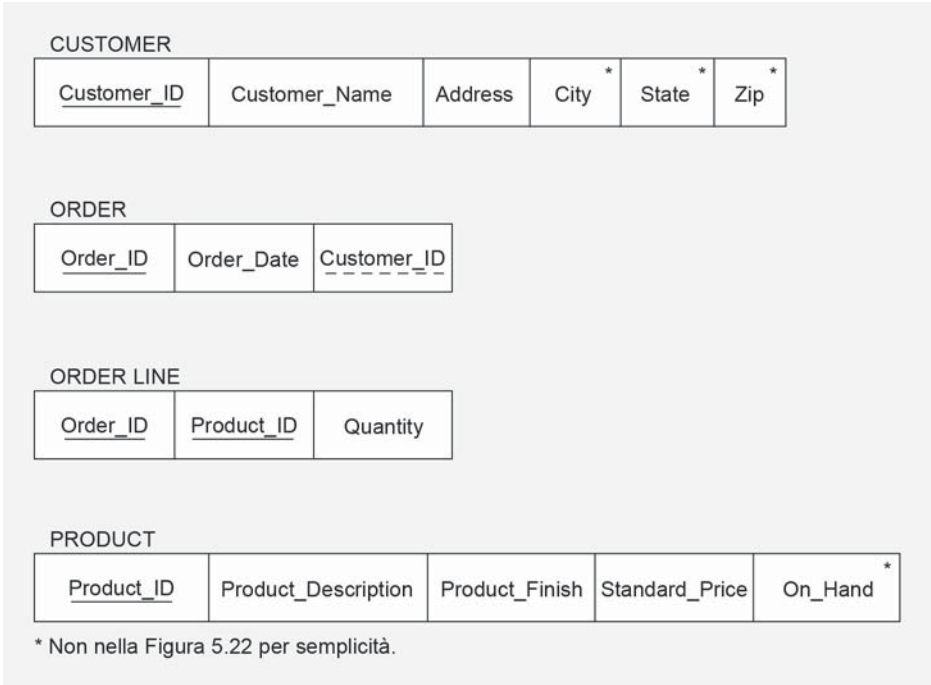
Una base di dati relazionale è costituita da un certo numero di relazioni. La struttura della base di dati viene descritta attraverso l'uso di uno schema concettuale (definito nel Capitolo 4), che è una rappresentazione della struttura logica complessiva della base di dati. Per esprimere uno schema concettuale ci sono due metodi comuni.

- Brevi istruzioni di testo, nelle quali ogni relazione ha un nome e i nomi dei suoi attributi seguono tra parentesi (si vedano le relazioni EMPLOYEE1 e DEPARTMENT definite in precedenza nel capitolo).
- Una rappresentazione grafica, in cui ciascuna relazione è rappresentata da un rettangolo che contiene gli attributi per la relazione.

Le istruzioni di testo hanno il vantaggio della semplicità. D'altra parte, una rappresentazione grafica fornisce un mezzo migliore per esprimere i vincoli di integrità referenziale (come si vedrà tra breve). In questo paragrafo utilizziamo entrambe le tecniche per esprimere uno schema, così che sia possibile confrontarle.

Nel Capitolo 5 è stato sviluppato un diagramma entità-associazione per Pine Valley Furniture Company (Figura 5.22). Nel Capitolo 3 sono state mostrate le seguenti quattro relazioni della base di dati (Figura 3.4): CUSTOMER, ORDER, ORDER LINE e PRODUCT. In questo paragrafo mostreremo uno schema per queste quattro relazioni. Una rappresentazione grafica è illustrata nella Figura 7.3.

Figura 7.3
Schema per quattro relazioni
(Pine Valley Furniture).



Quella che segue è una descrizione di testo delle relazioni:

CUSTOMER(Customer_ID,Customer_Name,Address,City,State,Zip)
ORDER(Order_ID,Order_Date,Address,Customer_ID)
PRODUCT(Product_ID, Product_Description, Product_Finish,Standard_Price,On_Hand)

Si noti che la chiave primaria per ORDER LINE è una chiave composta costituita da Order_ID e Product_ID. Anche Customer_ID è una chiave esterna nella relazione ORDER; questo consente all'utente di associare un ordine al cliente che lo ha effettuato. ORDER LINE ha due chiavi esterne: Order_ID e Product_ID. Queste chiavi permettono all'utente di associare ogni riga in un ordine con l'ordine e il prodotto principali.

Un'istanza di questa base di dati è mostrata nella Figura 7.4. La figura mostra quattro tabelle con alcuni dati di esempio. Si noti come le chiavi esterne permettono di associare le varie tabelle. Creare un'istanza dello schema relazionale con dati di esempio è una buona idea per tre ragioni.

1. I dati di esempio forniscono un modo comodo per controllare l'accuratezza del progetto.
2. I dati di esempio aiutano a migliorare la comunicazione con gli utenti nella discussione del progetto.
3. Si possono utilizzare i dati di esempio per sviluppare applicazioni prototipo e per testare le interrogazioni.

Figura 7.4

Istanza di uno schema relazionale.

CUSTOMER Table

Customer ID	Customer Name	Address	City	State	Postal Code
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-
3	Home Furnishings	1900 Allard Ave	Alb		
4	Eastern Furniture	1925 Beltline Rd.	Cart		
5	Impressions	5585 Westcott Ct.	Sac		
6	Furniture Gallery	325 Flatiron Dr.	Bou		

ORDER Table

Order_id	Order Date	Customer ID
1	9/8/2000	4
2	10/4/2000	3
3	7/19/2000	1
4	11/1/2000	6
5	7/28/2000	4
6	8/17/2000	6
7	8/17/2000	3

PRODUCT Table

Product ID	Product Description	Product Finish	Unit Price	On Hand
1	End Table	Cherry	\$175.00	8
2	Coffe Table	Birch	\$200.00	4
3	Computer Desk	Oak	\$375.00	5
4	Entertainment Center	Maple	\$650.00	3
5	Writer's Desk	Oak	\$325.00	0
6	8-Drawer Dresser	Birch	\$750.00	5
7	48" Bookcase	Walnut	\$150.00	5
8	48" Bookcase	Oak	\$175.00	2
9	96" Bookcase	Walnut	\$225.00	4
10	96" Bookcase	Oak	\$200.00	4
11	4-Drawer Dresser	Oak	\$500.00	3
12	8-Drawer Dresser	Oak	\$800.00	2
13	Nightstand	Cherry	\$150.00	5
14	Writer's Desk	Birch	\$300.00	2

Order_line Table

Order_ID	Product_ID	Quantity
1	2	2
1	10	2
2	3	1
2	8	2
2	14	2
3	6	3
4	3	1
4	7	3
5	6	2
5	10	1

7.2 Vincoli di integrità

Il modello di dati relazionale include numerosi tipi di vincoli, o regole aziendali, il cui scopo è quello di facilitare il mantenimento dell'accuratezza e dell'integrità dei dati nella base di dati. I tipi principali di vincoli di integrità sono vincoli di dominio, integrità dell'entità, integrità referenziale e asserzioni di azione.

7.2.1 Vincoli di dominio

Tutti i valori che appaiono in una colonna di una relazione devono essere tratti dallo stesso dominio. Un dominio è un insieme di valori che possono essere assegnati a un attributo. La definizione di un dominio è costituita dai seguenti componenti: nome di dominio, significato, tipo di dati, dimensione (o lunghezza) e valori consentiti e intervallo consentito (se applicabili). La Tabella 7.1 mostra le definizioni dei domini per i domini associati agli attributi della Figura 7.3.

7.2.2 Integrità dell'entità

La regola dell'integrità dell'entità è concepita per garantire che ogni relazione abbia una chiave primaria e che i valori di dati per quella chiave primaria siano validi. In particolare, garantisce che ogni attributo di chiave primaria non sia nullo.

In alcuni casi, un particolare attributo non può essere assegnato a un valore di dati. Le situazioni in cui questo può verificarsi sono due: non ci sono valori di dati applicabili o il valore di dati applicabile

Tabella 7.1 Definizioni dei domini per gli attributi selezionati.

Attributo	Nome di dominio	Descrizione	Dominio
Customer_ID	Customer_IDs	Insieme di tutti gli ID di cliente possibili	carattere: dimensione 5
Customer_Name	Customer_Names	Insieme di tutti i nomi di cliente possibili	carattere: dimensione 25
Customer_Address	Customer_Addresses	Insieme di tutti gli indirizzi di cliente possibili	carattere: dimensione 30
Customer_City	Cities	Insieme di tutte le città possibili	carattere: dimensione 20
Customer_State	States	Insieme di tutti gli Stati possibili	carattere: dimensione 2
Customer_Zip	Zips	Insieme di tutti i codici di avviamento postale possibili	carattere: dimensione 10
Order_ID	Order_IDs	Insieme di tutti gli ID di ordine possibili	carattere: dimensione 5
Order_Date	Order_Dates	Insieme di tutte le date di ordine possibili	formato di data mm-gg-aa
Product_ID	Product_IDs	Insieme di tutti gli ID di prodotto possibili	carattere: dimensione 5
Product_Description	Product_Descriptions	Insieme di tutte le descrizioni di prodotto possibili	carattere: dimensione 25
Product_Finish	Product_Finishes	Insieme di tutte le rifiniture di prodotto possibili	carattere: dimensione 12
Standard_Price	Unit_Prices	Insieme di tutti i prezzi unitari possibili	valuta: 6 cifre
On_Hand	On_Hands	Insieme di tutte le disponibilità possibili	intero: 3 cifre

non è noto quando i valori vengono assegnati. Si supponga, per esempio, di riempire un modulo di assunzione che contiene uno spazio riservato al numero di fax. Se non si ha un numero di fax, si lascia questo spazio vuoto perché non si applica al caso particolare. Si supponga, invece, che venga richiesto di inserire il numero di telefono del proprio datore di lavoro precedente. Se non si ricorda il numero, si può lasciare vuoto quello spazio fino a quando l'informazione non sarà nota.

Il modello di dati relazionale consente di assegnare un valore nullo a un attributo nelle situazioni appena descritte. Un nullo è un valore che può essere assegnato a un attributo quando nessun altro valore si applica o quando il valore applicabile non è noto. In realtà, un nullo non è un valore, quanto piuttosto l'assenza di un valore. Per esempio, non è lo stesso di un zero o di una stringa di spazi vuoti. L'inclusione di nulli nel modello relazionale è controversa, poiché può portare a risultati a anomali (Date, 1995). Per contro, Codd difende l'uso dei nulli per i valori mancanti (Codd, 1999).

Un aspetto su cui tutti concordano è che non deve essere permesso che i valori della chiave primaria siano nulli. La regola di integrità dell'entità stabilisce quindi quanto segue: nessun attributo di chiave primaria (o componente di un attributo di chiave primaria) può essere nullo.

7.2.3 Integrità referenziale

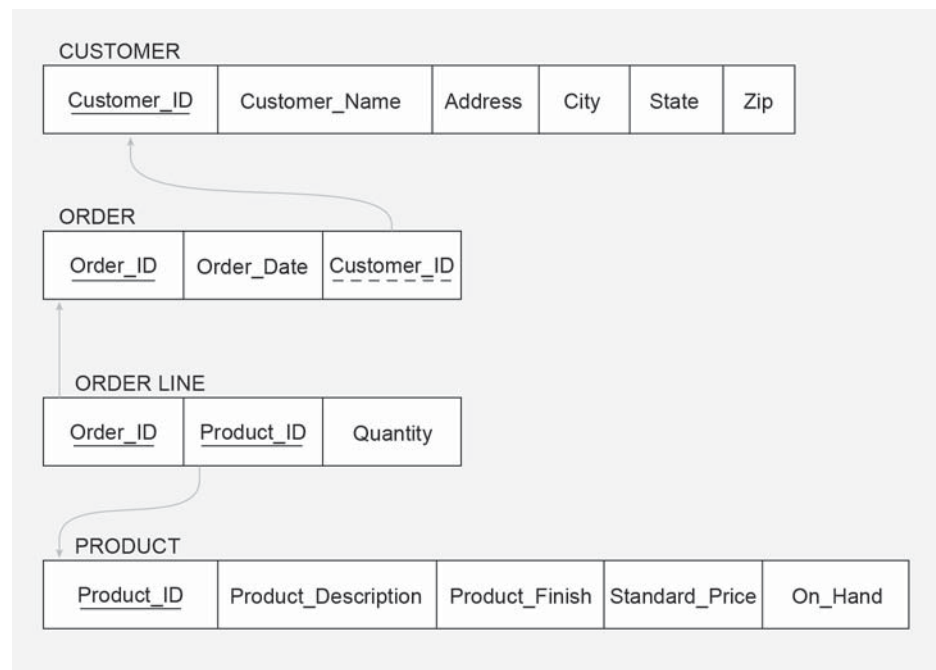
Nel modello di dati relazionale, le associazioni tra tabelle vengono definite attraverso l'uso di chiavi esterne. Per esempio, nella Figura 7.4, l'associazione tra le tabelle CUSTOMER e ORDER è definita includendo l'attributo Customer_ID come chiave esterna in ORDER. Questo implica naturalmente che prima di inserire una nuova riga nella tabella ORDER, il cliente di quell'ordine debba già esistere nella tabella CUSTOMER. Se si esaminano le righe nella tabella ORDER nella Figura 7.4, si vedrà che ogni numero di cliente di un ordine appare già nella tabella CUSTOMER.

Un vincolo di integrità referenziale è una regola che mantiene la coerenza fra le righe di due relazioni. La regola stabilisce che se in una relazione c'è una chiave esterna, ogni valore della chiave esterna deve corrispondere a un valore della chiave primaria in un'altra relazione, o il valore della chiave esterna sarà nullo. Si osservino le tabelle nella Figura 7.4 per verificare se è stata applicata la regola dell'integrità referenziale.

La versione grafica dello schema relazionale fornisce una tecnica semplice per identificare le associazioni nelle quali l'integrità referenziale deve essere applicata. La Figura 7.5 mostra lo schema per le relazioni introdotte nella Figura 7.3. È stata disegnata una freccia da ogni chiave esterna fino alla chiave primaria associata. Un vincolo di integrità referenziale deve essere definito per ognuna delle frecce nello schema.

Come si può sapere se una chiave esterna può essere nulla? Una risposta semplice si può avere nella Figura 5.22, il diagramma E-R associato alle Figure 7.4 e 7.5. Nella Figura 5.22, poiché ogni ordine deve avere un cliente (la cardinalità minima accanto a CUSTOMER sull'associazione Submits è uno), la chiave esterna di Customer_ID non può essere nulla nella relazione ORDER. Se la cardinalità minima fosse stata zero, la chiave esterna avrebbe potuto essere nulla. Se una chiave esterna possa essere nulla, deve essere specificato come proprietà dell'attributo di chiave esterna quando la base di dati viene definita.

Figura 7.5
Vincoli di integrità referenziale
(Pine Valley Furniture).



In effetti, una chiave esterna che può essere nulla diventa più complessa da modellare in un diagramma E-R e diventa più difficile determinare quanto mostrato finora. Per esempio, cosa accade ai dati degli ordini se si decide di eliminare un cliente che ha effettuato degli ordini? Si potrebbe lo stesso voler vedere le vendite anche se il cliente non è più di interesse. Sono possibili tre scelte.

1. Eliminare gli ordini associati (si parla di *eliminazione a cascata*), nel quale caso non si perde solo il cliente, ma anche la storia di tutte le vendite.
2. Impedire l'eliminazione del cliente fino a quando non vengono prima eliminati tutti gli ordini associati (controllo di sicurezza).

3. Collocare un valore nullo nella chiave esterna (un'eccezione che dice che, sebbene un ordine debba avere un valore `Customer_ID` quando l'ordine viene creato, `Customer_ID` può diventare nullo successivamente se il cliente associato viene eliminato).

Vedremo come ciascuna di queste scelte viene implementata quando descriveremo il linguaggio SQL di interrogazione delle basi di dati nel Capitolo 9.

7.2.4 Asserzioni di azione

Nel Capitolo 6 sono state trattate le regole aziendali ed è stata introdotta una nuova categoria di regola aziendale chiamata *asserzione di azione*. Per esempio, un'asserzione di azione tipica può stabilire quanto segue: “Una persona può acquistare un biglietto per la partita solo se possiede un abbonamento per la stagione”. Esistono diverse tecniche per definire e applicare queste regole; alcune di queste verranno trattate nei prossimi capitoli.

7.2.5 Creare tabelle relazionali

In questo paragrafo si creeranno alcune definizioni di tabella per le quattro tabelle mostrate nella Figura 7.5. Queste definizioni vengono create utilizzando le istruzioni **CREATE TABLES** del linguaggio SQL di definizione dei dati. Nella realtà, queste definizioni di tabella vengono create durante la fase di implementazione successivamente nel processo di sviluppo della base di dati. Mostriamo queste tabelle di esempio nel presente capitolo per continuità e soprattutto per illustrare il modo in cui i vincoli di integrità precedentemente descritti sono implementate in SQL.

Le definizioni SQL per le tabelle sono mostrate nella Figura 7.6. Viene creata una tabella per ciascuna delle quattro tabelle mostrate nello schema relazionale (Figura 7.5). Viene quindi definito ciascun attributo per una tabella. Si noti che il tipo di dati e la lunghezza di ogni attributo sono tratti dalle definizioni del dominio (Tabella 7.1). Per esempio, l'attributo `Customer_Name` nella relazione `CUSTOMER` è definito come tipo di dati `VARCHAR` (carattere variabile) con una lunghezza di 25. Specificando **NOT NULL**, ogni attributo può essere vincolato in modo che non gli venga assegnato un valore nullo.

La chiave primaria per ogni tabella è specificata utilizzando la clausola **PRIMARY KEY** alla fine di ogni definizione di tabella. La tabella `ORDER_LINE` illustra come specificare una chiave primaria quando questa è un attributo composto.

In questo esempio, la chiave primaria di `ORDER_LINE` è la combinazione di `Order_ID` e `Product_ID`. Ogni attributo di chiave primaria nelle quattro tabelle è vincolato con **NOT NULL**. Questo applica il vincolo di integrità dell'entità descritto nel paragrafo precedente. Si noti che il vincolo **NOT NULL** può essere utilizzato anche con attributi di chiave non primaria.

I vincoli di integrità referenziale vengono facilmente definiti utilizzando lo schema grafico mostrato nella Figura 7.5. Una freccia parte da ciascuna chiave esterna e punta alla chiave primaria correlata nella relazione associata. Nella definizione SQL per le tabelle, un'istruzione **FOREIGN KEY REFERENCES** corrisponde a ognuna di queste frecce. Per la tabella `ORDER`, quindi, la chiave esterna `CUSTOMER_ID` fa riferimento alla chiave primaria di `CUSTOMER`, che è anche `CUSTOMER_ID`. Sebbene in questo caso la chiave esterna e le chiavi primarie abbiano lo stesso nome, la cosa non è necessaria. Per esempio, l'attributo di chiave esterna potrebbe chiamarsi `CUST_NO` invece che `CUSTOMER_ID`. Tuttavia le chiavi esterne e primarie devono avere lo stesso dominio.

La tabella `ORDER_LINE` fornisce un esempio di una tabella che ha due chiavi esterne. Le chiavi esterne in questa tabella fanno riferimento sia alla tabella `ORDER` sia alla tabella `PRODUCT`.

Figura 7.6
Definizioni SQL
per la tabella.

```
CREATE TABLE CUSTOMER
  (CUSTOMER_ID          VARCHAR(5)      NOT NULL,
   CUSTOMER_NAME        VARCHAR(25)     NOT NULL,
   CUSTOMER ADDRESS     VARCHAR(30)     NOT NULL,
   CUSTOMER_CITY        VARCHAR(20)     NOT NULL,
   CUSTOMER_STATE       CHAR(2)         NOT NULL,
   CUSTOMER_ZIP         CHAR(10)        NOT NULL,
  PRIMARY KEY (CUSTOMER_ID);

CREATE TABLE ORDER
  (ORDER_ID             CHAR(5)         NOT NULL,
   ORDER DATE          DATE            NOT NULL,
   CUSTOMER_ID         VARCHAR(5)      NOT NULL,
  PRIMARY KEY (ORDER_ID),
  FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID);

CREATE TABLE ORDER_LINE
  (ORDER_ID             CHAR(5)         NOT NULL,
   PRODUCT_ID          CHAR(5)         NOT NULL,
   QUANTITY            INT              NOT NULL,
  PRIMARY KEY (ORDER_ID, PRODUCT_ID),
  FOREIGN KEY (ORDER_ID) REFERENCES ORDER (ORDER_ID),
  FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT (PRODUCT_ID);

CREATE TABLE PRODUCT
  (PRODUCT_ID          CHAR(5)         NOT NULL,
   PRODUCT_DESCRIPTION  VARCHAR(25),
   PRODUCT_FINISH      VARCHAR(12),
   STANDARD_PRICE      DECIMAL(8,2)   NOT NULL,
   ON_HAND             INT             NOT NULL,
  PRIMARY KEY (PRODUCT_ID);
```

7.2.6 Relazioni ben strutturate

Per introdurre l’argomento della normalizzazione, occorre affrontare la seguente domanda: cosa costituisce una relazione ben strutturata? Intuitivamente, una **relazione ben strutturata** contiene una ridondanza minima e permette agli utenti di inserire, modificare ed eliminare le righe di una tabella senza errori e incoerenze. Ogni riga della tabella contiene i dati che descrivono un impiegato, e qualsiasi modifica nei dati dell’impiegato (come un cambiamento nello stipendio) è confinato a una riga nella tabella. Per contro, EMPLOYEE2 (Figura 7.2b) non è una relazione ben strutturata. Se si osservano i dati di esempio della tabella, si noter  una notevole ridondanza. Per esempio, i valori per Emp_ID, Name, Dept_Name e Salary appaiono in due righe separate per gli impiegati 100, 110 e 150. Di conseguenza, se lo stipendio dell’impiegato 100 cambia, si dovr  registrare questo fatto in due righe (o in pi  righe per qualche impiegato).

Le ridondanze in una tabella possono comportare errori o incoerenze, chiamate **anomalie**, ogni qualvolta un utente cerca di aggiornare i dati. I tre tipi di anomalie possibili sono quelle di inserimento, eliminazione e modifica.

- 1. *Anomalia di inserimento.* Si supponga di dover aggiungere un nuovo impiegato a EMPLOYEE2. La chiave primaria per questa relazione è la combinazione di Emp_ID e Course_Title (come visto in precedenza). Per inserire una nuova riga, l'utente deve fornire i valori sia per Emp_ID sia per Course_Title (poiché valori di chiave primaria non possono essere nulli o inesistenti). Questa è un'anomalia, poiché l'utente dovrebbe essere in grado di immettere i dati sull'impiegato senza dover fornire quelli sul corso.
- 2. *Anomalia di eliminazione.* Si supponga che i dati per l'impiegato numero 140 vengano eliminati dalla tabella. Questo produrrà una perdita dell'informazione che questo impiegato ha completato un corso (Tax Acc) in data 12/8/200X. Si perde l'informazione che questo corso aveva un frequentante che l'ha completato in quella data.
- 3. *Anomalia di modifica.* Si supponga che l'impiegato numero 100 abbia un aumento di stipendio. Si dovrà registrare l'aumento in ogni riga che riguarda quell'impiegato (due occorrenze nella Figura 7.2); in caso contrario, i dati saranno incoerenti.

Queste anomalie indicano che EMPLOYEE2 non è una relazione ben strutturata. Il problema con questa relazione è che contiene i dati su due entità: EMPLOYEE e COURSE. Utilizzeremo la teoria della normalizzazione (descritta di seguito) per dividere EMPLOYEE2 in due relazioni; una delle relazioni risultanti sarà EMPLOYEE1 (Figura 7.1), mentre l'altra sarà EMP_COURSE, che appare con i dati di esempio nella Figura 7.7. La chiave primaria di questa relazione è la combinazione di Emp_ID e Course_Title; per evidenziare questo fatto, nella Figura 7.7 i nomi degli attributi appaiono sottolineati.

Figura 7.7
EMP_COURSE.

<u>Emp_ID</u>	<u>Course_Title</u>	Date_Completed
100	SPSS	6/19/200X
100	Surveys	10/7/200X
140	Tax Acc	12/8/200X
110	SPSS	1/12/200X
110	C++	4/22/200X
150	SPSS	6/19/200X
150	Java	8/12/200X

7.3 Trasformare i diagrammi EER in relazioni

Durante la progettazione locale, si trasformano i diagrammi E-R ed EER che sono stati sviluppati durante la progettazione concettuale in schemi di basi di dati relazionali. Gli input di questo processo sono i diagrammi entità-associazioni (anche nella forma estesa) studiati nei Capitoli 5 e 6. Gli output sono gli schemi relazionali descritti nei primi due paragrafi di questo capitolo.

La trasformazione (o mappatura) dei diagrammi E-R in relazioni è un processo relativamente diretto con un insieme di regole ben definite. Molti strumenti CASE possono effettuare automaticamente diversi dei passaggi della conversione. È tuttavia importante capire a fondo le fasi di questo processo per tre ragioni.

- 1. Gli strumenti CASE spesso non possono modellare le associazioni di dati più complesse, come quelle ternarie o supertipo/sottotipo. In questi casi si deve eseguire ogni passaggio manualmente.

2. Ci sono alcune alternative valide rispetto al punto in cui si deve scegliere una determinata soluzione.
3. Si deve essere preparati per eseguire un controllo di qualità sui risultati ottenuti con uno strumento CASE.

Nella trattazione che segue illustreremo i passaggi nella trasformazione con alcuni esempi tratti dai Capitoli 5 e 6, dove sono già stati discussi tre tipi di entità.

1. *Entità regolari*: sono entità che hanno un'esistenza indipendente e che rappresentano generalmente oggetti del mondo reale come persone e prodotti. I tipi di entità regolari sono rappresentati da rettangoli con un bordo singolo.
2. *Entità deboli*: sono entità che non possono esistere se non con un'associazione di identificazione con un tipo di entità possessore (regolare). Le entità deboli sono identificate da un rettangolo con un bordo doppio.
3. *Entità associative*: chiamate anche gerundi, sono formate da associazioni multi-a-molti tra altri tipi di entità. Le entità associative sono rappresentate da un rettangolo con un bordo singolo che racchiude il simbolo di associazione a rombo.

7.3.1 Passaggio 1: mappare le entità regolari

Ogni tipo di entità regolare in un diagramma ER viene trasformato in una relazione. Il nome dato alla relazione è solitamente lo stesso del tipo di entità. Ogni attributo semplice del tipo di entità diventa un attributo della relazione. L'identificatore del tipo di entità diventa la chiave primaria della relazione corrispondente. È necessario assicurarsi che questa chiave primaria soddisfi le proprietà desiderabili degli identificatori delineate nel Capitolo 5.

La Figura 7.8a mostra una rappresentazione del tipo di entità CUSTOMER per Pine Valley Furniture Company tratta dal Capitolo 5 (Figura 5.22). La relazione CUSTOMER corrispondente è mostrata in una forma grafica nella Figura 7.8b. Per ragioni di semplicità, in questa figura e in quelle che seguono in questo paragrafo verranno mostrati solo alcuni degli attributi chiave per ogni relazione.

Attributi composti

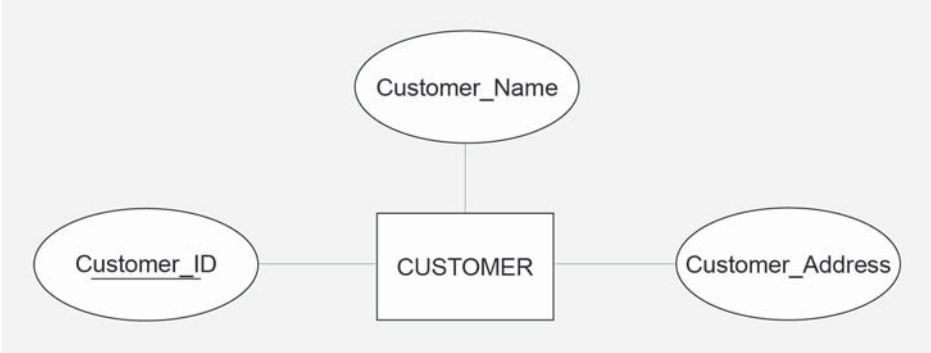
Quando un tipo di entità regolare ha un attributo composto, nella nuova relazione vengono inclusi solo gli attributi semplici dei componenti dell'attributo composto. La Figura 7.9 mostra una variante dell'esempio nella Figura 7.8, dove Customer_Address è rappresentato come attributo composto con i componenti Street, City e Zip (Figura 7.9a). Questa entità è mappata sulla relazione CUSTOMER, che contiene gli attributi semplici dell'indirizzo, come mostrato nella Figura 7.9b.

Attributi multivalore

Quando il tipo di entità regolare contiene un attributo multivalore, vengono create due nuove relazioni (invece di una). La prima relazione contiene tutti gli attributi del tipo di entità a eccezione dell'attributo multivalore. La seconda relazione contiene due attributi che formano la chiave primaria della seconda relazione. Il primo di questi attributi è la chiave primaria della prima relazione, che diventa una chiave esterna nella seconda. La seconda è l'attributo multivalore. Il nome della seconda relazione dovrebbe rappresentare il significato dell'attributo.

Un esempio di questa procedura è mostrato nella Figura 7.10. Si tratta del tipo di entità EMPLOYEE per Pine Valley Furniture Company (Figura 5.22). Come mostrato nella Figura 7.10a, EMPLOYEE ha Skill come attributo multivalore. La Figura 7.10b mostra le due relazioni che vengono create. La seconda

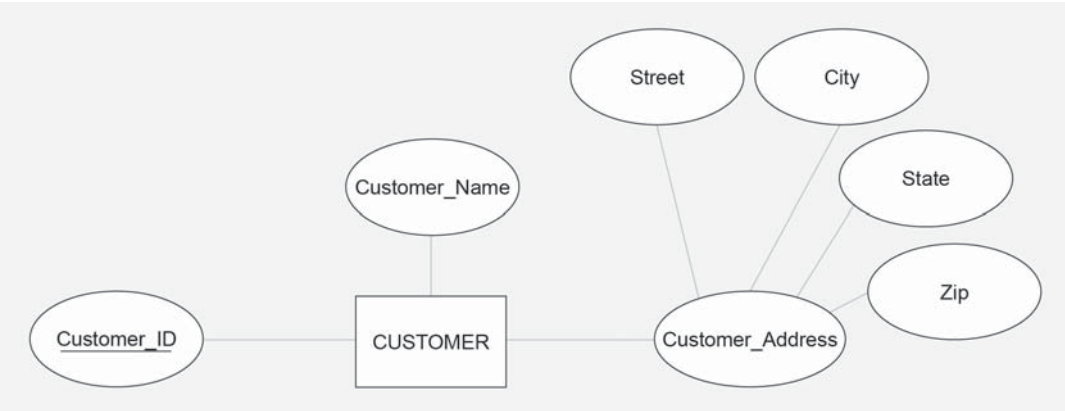
Figura 7.8
Mappare l'entità
regolare CUSTOMER.
(a) Tipo di entità
CUSTOMER.



(b) Relazione
CUSTOMER.



Figura 7.9
Mappare un attributo
composto.
(a) Tipo di entità
CUSTOMER con
attributo composto.



(b) Relazione
CUSTOMER
con i dettagli
sull'indirizzo.



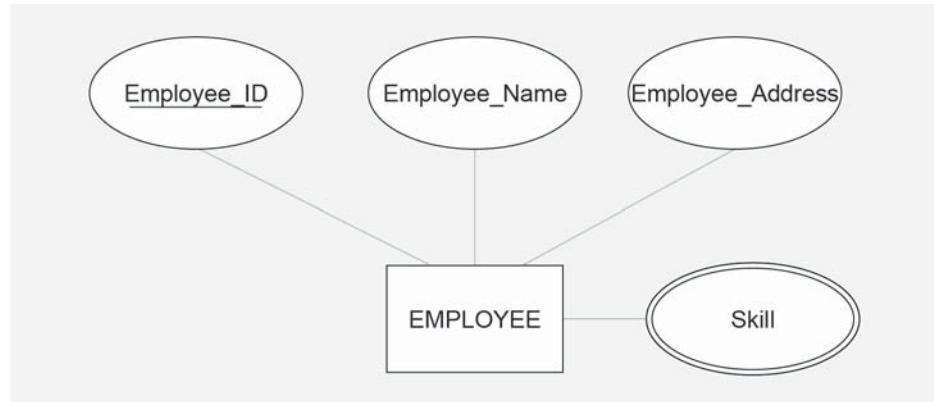
relazione (chiamata EMPLOYEE) ha i due attributi Employee_ID e Skill, che formano la chiave primaria. L'associazione tra chiavi esterne e primarie è indicata nella figura con la freccia.

La relazione EMPLOYEE_SKILL non contiene alcun attributo non di chiave (chiamato anche *descrittore*). Ogni riga registra semplicemente il fatto che un determinato impiegato possiede una determinata competenza. Questo fornisce la possibilità di suggerire agli utenti che a questa relazione possono essere aggiunti nuovi attributi. Per esempio, gli attributi Years_Experience e/o Certification_Date potrebbero essere nuovi valori appropriati da aggiungere a questa relazione.

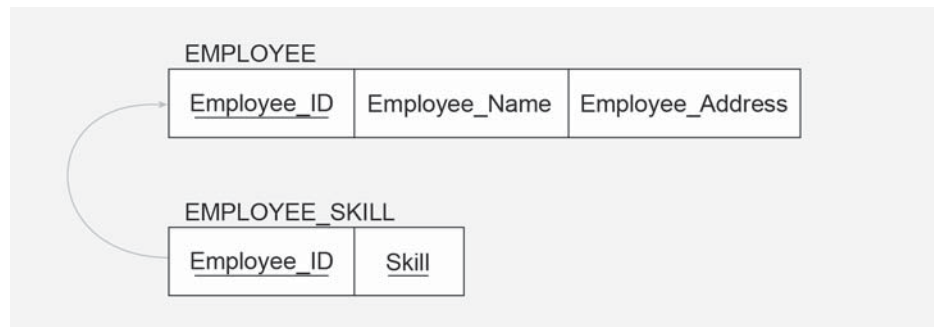
Figura 7.10

Mappare un'entità con un attributo multivalore.

(a) Tipo di entità EMPLOYEE con attributo multivalore.



(b) Mappare un valore multiattributo.



7.3.2 Passaggio 2: mappare le entità deboli

Come si ricorderà, un tipo di entità debole non ha un'esistenza indipendente, ma esiste solo attraverso un'associazione identificativa con un'altra entità chiamata *possessore*. Un tipo di entità debole non ha un identificatore completo, ma deve avere un attributo chiamato *identificatore parziale* che permette di distinguere tra le varie occorrenze dell'entità debole per ciascuna istanza di entità possessore.

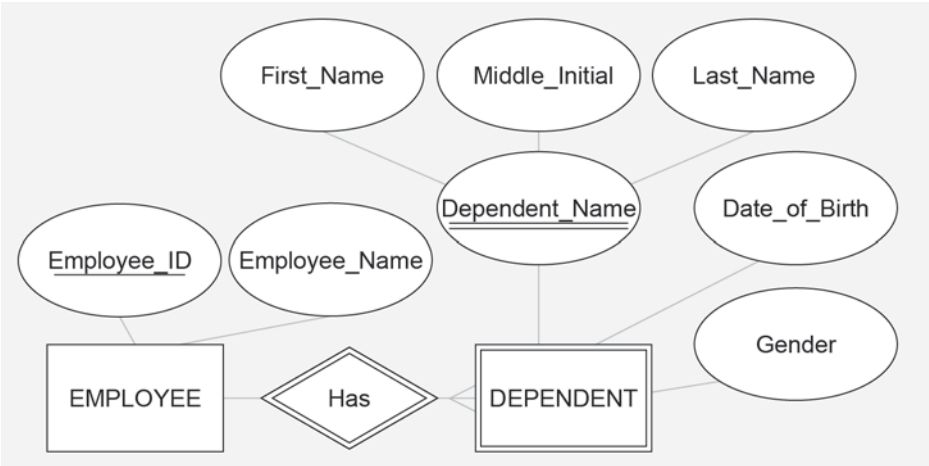
La procedura che segue presume che sia già stata creata una relazione che corrisponde al tipo di entità identificativa durante il passaggio 1. Se non ne esiste una, è necessario crearla utilizzando il processo descritto al passaggio 1.

Per ciascun tipo di entità debole, si crei una nuova relazione e si includono tutti gli attributi semplici (o i componenti semplici degli attributi composti) come attributi di questa relazione. Si includa quindi la chiave primaria della relazione *identificativa* come attributo di chiave esterna in questa nuova relazione. La chiave primaria della nuova relazione è la combinazione di questa chiave primaria dell'identificativo con l'identificatore parziale del tipo di entità debole.

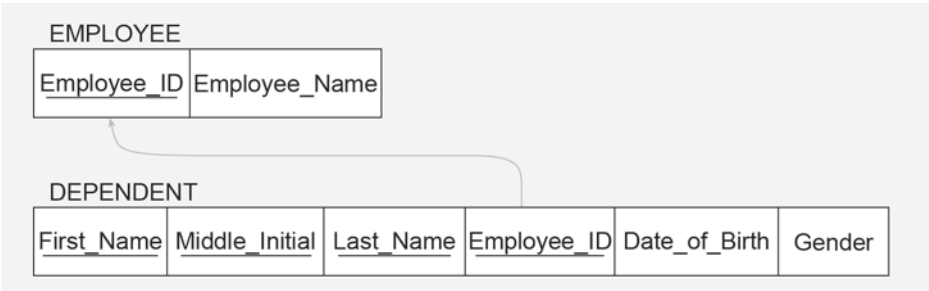
Un esempio di questo processo è mostrato nella Figura 7.11. La Figura 7.11a mostra il tipo di entità debole *DEPENDENT* con il suo tipo di entità identificativa *EMPLOYEE* collegata dall'associazione identificativa *Has* (Figura 5.5). Si noti che l'attributo *Dependent_Name*, che è l'identificatore parziale per questa relazione, è un attributo composto con i componenti *First_Name*, *Middle_Initial* e *Last_Name*. Si presume quindi che, *per un dato impiegato*, queste voci identificheranno in modo univoco un dipendente (un'eccezione significativa sarebbe il caso del campione di boxe George Foreman, che ha chiamato tutti i suoi figli come se stesso!).

La Figura 7.11b mostra le due relazioni che risultano dalla mappatura di questo segmento E-R. La chiave primaria di DEPENDENT è costituita da quattro attributi: Employee_ID, First_Name, Middle_Initial e Last_Name. Date_of_Birth e Gender non sono attributi di chiave. L'associazione di chiave esterna con la sua chiave primaria è indicata dalla freccia.

Figura 7.11
Esempio di mappatura di un'entità debole.
(a) Entità debole DEPENDENT.



(b) Relazioni risultanti dall'entità debole.



7.3.3 Passaggio 3: mappare le associazioni binarie

La procedura per rappresentare le associazioni binarie dipende sia dal grado delle associazioni (unaria, binaria e ternaria) sia dalle loro cardinalità. Descriveremo e illustreremo i casi più importanti di seguito.

Mappare associazioni binarie uno-a-molti

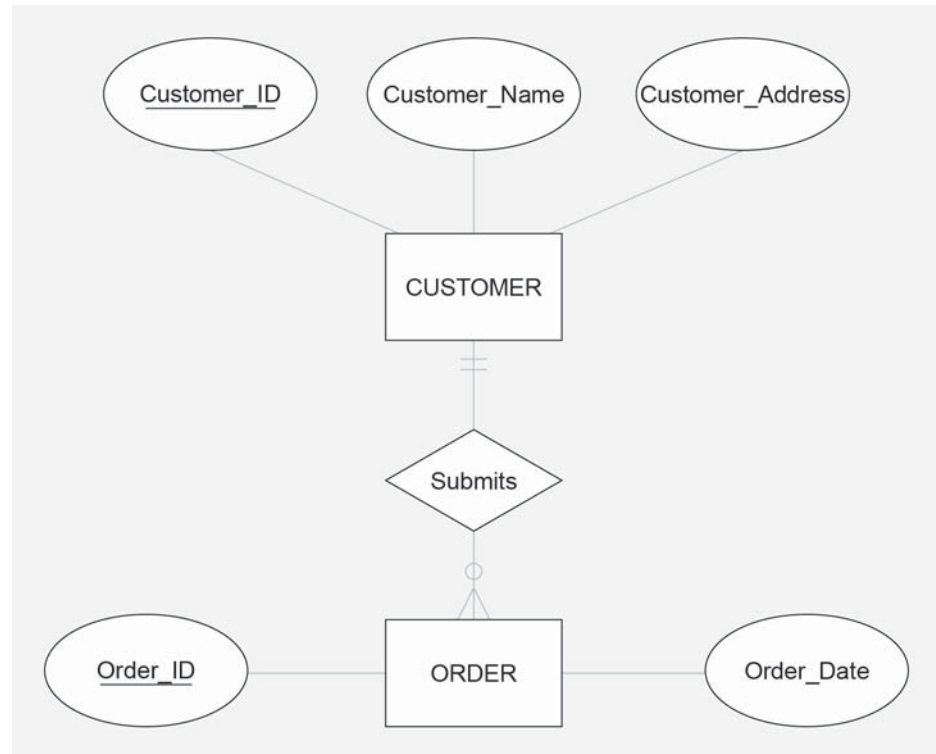
Per ogni associazione binaria 1:M, si crei innanzitutto una relazione per ognuno dei due tipi di entità che partecipano all'associazione utilizzando la procedura descritta nel passaggio 1. Si includa quindi l'attributo (o gli attributi) di chiave primaria sul lato uno dell'associazione come chiave esterna nella relazione che si trova sul lato molti dell'associazione.

Per illustrare questa procedura semplice, utilizziamo l'associazione Submits tra clienti e ordini per Pine Valley Furniture Company (Figura 5.22). Questa associazione 1:M è illustrata nella Figura 7.12a.

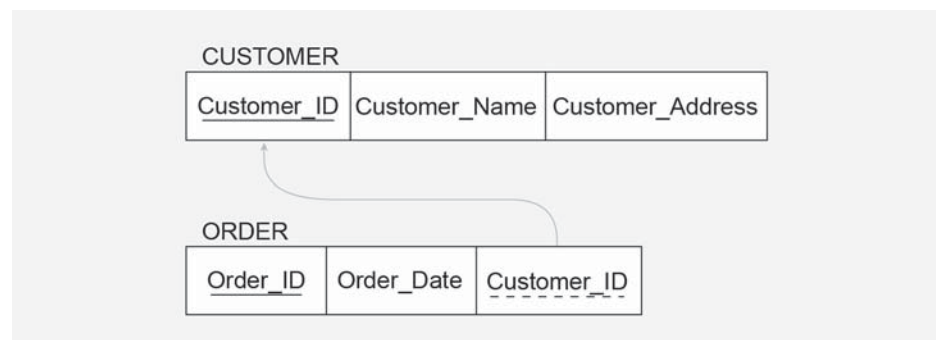
La Figura 7.12b mostra il risultato dell'applicazione di questa regola per mappare i tipi di entità con l'associazione 1:M. La chiave primaria Customer_ID di CUSTOMER (il lato uno) è inclusa come chiave esterna in ORDER (il lato molti). L'associazione di chiave esterna è indicata con una freccia.

Figura 7.12

Esempio di mappatura di un'associazione 1:M.
(a) Associazione tra clienti e ordini.



(b) Mappare l'associazione.

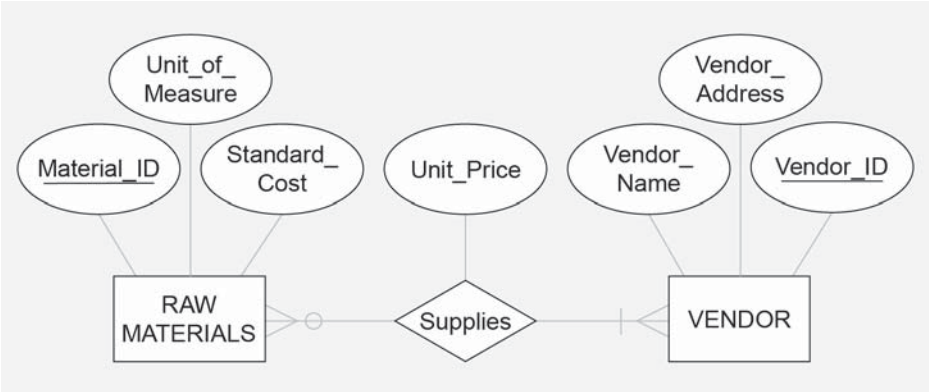


Mappare associazioni binarie multi-a-molti

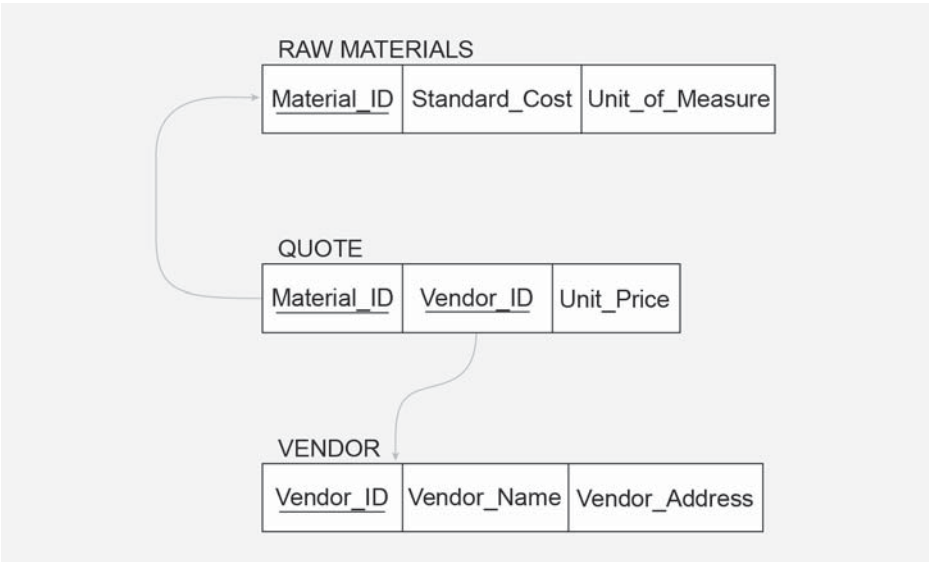
Si supponga che ci sia un'associazione binaria multi-a-molti (M:N) tra i tipi di entità A e B. Per questa associazione, si crei una nuova relazione C. Si includano come attributi di chiave primaria in C la chiave primaria per ciascuno dei tipi di entità partecipanti. Questi attributi diventano la chiave primaria di C. Qualsiasi attributo non di chiave che è associato nell'associazione M:N è incluso nella relazione C.

La Figura 7.13 mostra un esempio dell'applicazione di questa regola. La Figura 7.13a mostra l'associazione Supplies tra i tipi di entità **VENDOR** e **RAW MATERIALS** per Pine Valley Furniture Company (introdotti nella Figura 5.2). La Figura 7.13b mostra le tre relazioni (**VENDOR**, **RAW MATERIALS** e **QUOTE**) che si formano dai tipi di entità e dall'associazione Supplies. Per prima cosa viene creata un'associazione per ognuno dei tipi di entità regolari **VENDOR** e **RAW MATERIALS**. Quindi viene creata una relazione (chiamata **QUOTE** nella Figura 7.13b) per l'associazione Supplies. La chiave primaria di **QUOTE** è la combinazione di **Vendor_ID** e **Material_ID**, che sono le chiavi primarie rispettive di **VENDOR** e **RAW MATERIALS**. Come indicato nel diagramma, questi attributi sono chiavi esterne che puntano alle rispettive chiavi primarie. L'attributo non di chiave **Unit_Price** appare anche in **QUOTES**.

Figura 7.13
Esempio di mappatura
di un'associazione M:N.
(a) Associazioni di richiesta (M:N).



(b) Tre relazioni risultanti.



Mappare associazioni binarie uno-a-uno

Le associazioni binarie uno-a-uno possono essere considerate come un caso speciale di associazioni uno-a-molti. Il processo di mappatura di queste associazioni sulle relazioni si svolge in due fasi. Innanzitutto vengono create due relazioni, una per ogni tipo di entità partecipante. Quindi la chiave primaria di una delle relazioni viene inclusa come chiave esterna nell'altra relazione.

In un'associazione 1:1, l'associazione in una direzione è quasi sempre facoltativa uno, mentre l'associazione nell'altra direzione è obbligatoria uno (si può rivedere la notazione di questi termini nella Figura 7.1). È preferibile includere nella relazione sul lato facoltativo dell'associazione la chiave primaria del tipo di entità che ha la partecipazione obbligatoria nell'associazione 1:1. Questo approccio evita di dover memorizzare valori nulli nell'attributo di chiave esterna. Qualsiasi attributo associato all'associazione stessa è incluso nella stessa relazione come chiave esterna.

Un esempio dell'applicazione di questa procedura è mostrato nella Figura 7.14. La Figura 7.14a mostra un'associazione binaria 1:1 tra i tipi di entità NURSE e CARE CENTER. Ogni unità di cura deve avere un infermiere responsabile per quella unità. L'associazione da CARE CENTER a NURSE è quindi obbligatoria uno, mentre quella da NURSE a CARE CENTER è facoltativa uno (poiché qualsiasi infermiere può essere o non essere responsabile di un'unità di cura). L'attributo Date_Assigned è collegato all'associazione In_charge.

Il risultato della mappatura di questa associazione in un insieme di relazioni è mostrato nella Figura 6.14b. Le due relazioni NURSE e CARE CENTER sono create dai due tipi di entità. Poiché CARE CENTER è un partecipante facoltativo, la chiave esterna viene collocata in questa relazione. In questo caso la chiave esterna è chiamata Nurse_In_Charge e ha lo stesso dominio di Nurse_ID; l'associazione con la chiave primaria è mostrata nella figura. L'attributo Date_Assigned è collocato anche in CARE CENTER e non gli viene concesso di essere nullo.

7.3.4 Passaggio 4: mappare le entità associative

Come spiegato nel Capitolo 5, quando il modellatore dei dati incontra un'associazione multi-a-molti, può decidere di modellare quell'associazione come entità associativa nel diagramma E-R. Questo approccio è particolarmente adatto quando l'utente finale riesce a visualizzare al meglio l'associazione come tipo di entità piuttosto che come associazione M:M. La mappatura dell'entità associativa implica fondamentalmente gli stessi passaggi di quella di un'associazione M:N, come descritto al passaggio 3.

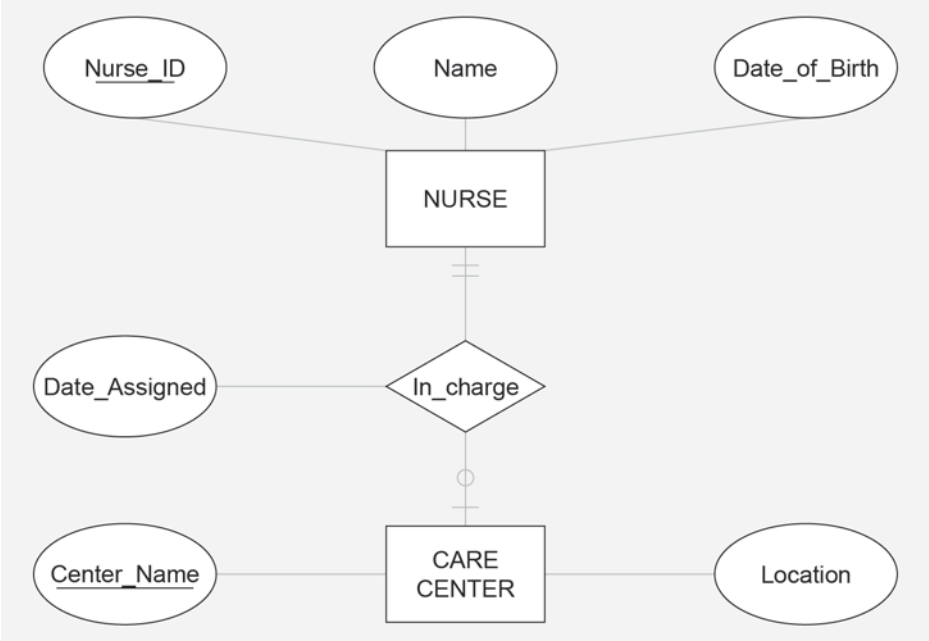
Il primo passaggio consiste nel creare tre relazioni: una per ciascuno dei due tipi di entità partecipanti e la terza per l'entità associativa. La relazione nata dall'entità associativa è definita *relazione associativa*. Il secondo passaggio dipende dal fatto se nel diagramma E-R è stato assegnato un identificatore per l'entità associativa.

Identificatore non assegnato

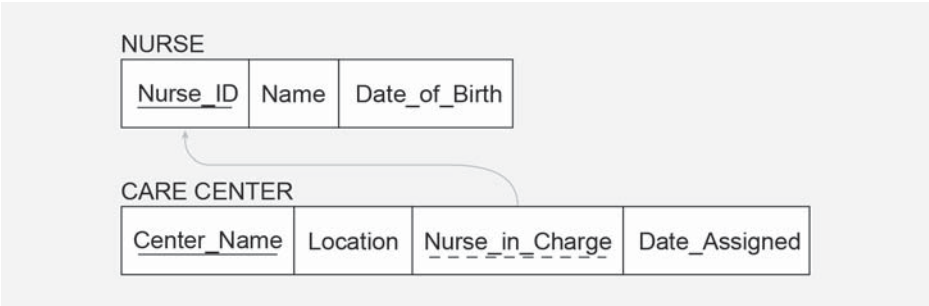
Se non è stato assegnato un identificatore, la chiave primaria predefinita per la relazione associativa sarà costituita dai due attributi di chiave primaria delle altre due relazioni. Questi attributi sono quindi le chiavi esterne che fanno riferimento alle altre due relazioni.

Un esempio di questa situazione è mostrato nella Figura 7.15. La Figura 7.15a mostra l'entità associativa ORDER LINE che collega i tipi di entità ORDER e PRODUCT di Pine Valley Furniture Company (Figura 5.22). Si noti la somiglianza di questo esempio con quello di un'associazione M:N mostrato nella Figura 7.13.

Figura 7.14
Mappare un'associazione
binaria 1:1.
(a) Associazione binaria 1:1.



(b) Relazioni risultanti.



Identificatore assegnato

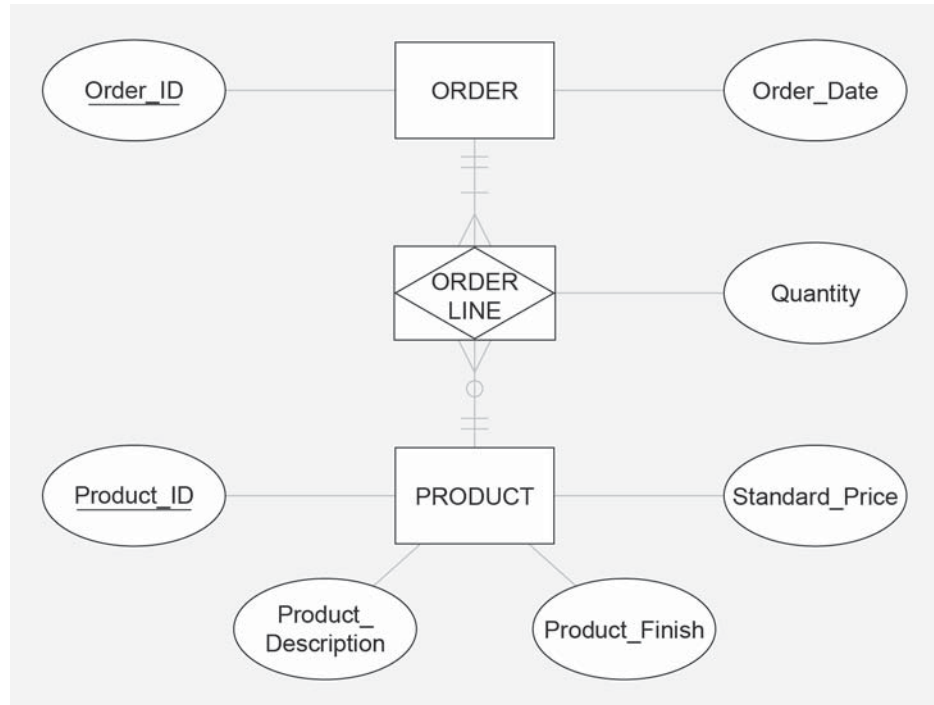
Può accadere che il modellatore dei dati assegni un identificatore (chiamato *identificatore surrogato* o *chiave surrogata*) al tipo di entità associativo nel diagramma E-R. Le motivazioni di questo approccio sono due.

- 1. Il tipo di entità associativa ha un identificatore naturale che è familiare agli utenti finali.
- 2. L'identificatore predefinito (costituito dagli identificatori di ognuno dei tipi di entità partecipanti) può non identificare in modo univoco le istanze dell'entità associativa.

Il processo per mappare l'entità associativa in questo caso viene modificato come segue. Come prima, viene creata una nuova relazione associativa per rappresentare l'entità associativa. Tuttavia, la chiave primaria per questa relazione è l'identificatore assegnato nel diagramma E-R (e non la chiave predefinita). Le chiavi primarie per i due tipi di entità partecipanti vengono quindi incluse come chiavi esterne nella relazione associativa.

Figura 7.15

Mappare un'entità associativa.
(a) Un'entità associativa.



(b) Tre relazioni risultanti.



Un esempio di questo processo è mostrato nella Figura 7.16. La Figura 7.16a mostra il tipo di entità associativa SHIPMENT che collega i tipi di entità CUSTOMER e VENDOR. Ship_No è stato scelto come identificatore di SHIPMENT per due ragioni.

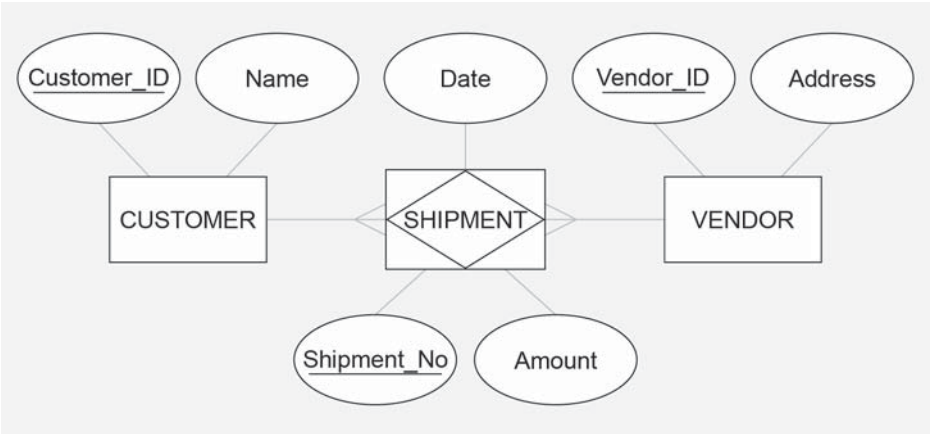
1. Ship_No è un identificatore naturale per questa entità che è molto familiare agli utenti finali.
2. L'identificatore predefinito costituito dalla combinazione di Customer_ID e di Vendor_ID non identifica in modo univoco le istanze di SHIPMENT. In effetti un dato produttore effettuerà delle spedi-

zioni a un dato cliente. Anche includere l'attributo Date non garantisce l'univocità, poiché ci possono essere più spedizioni da un particolare produttore in una determinata data, ma la chiave surrogata Shipment_No identificherà univocamente ogni spedizione.

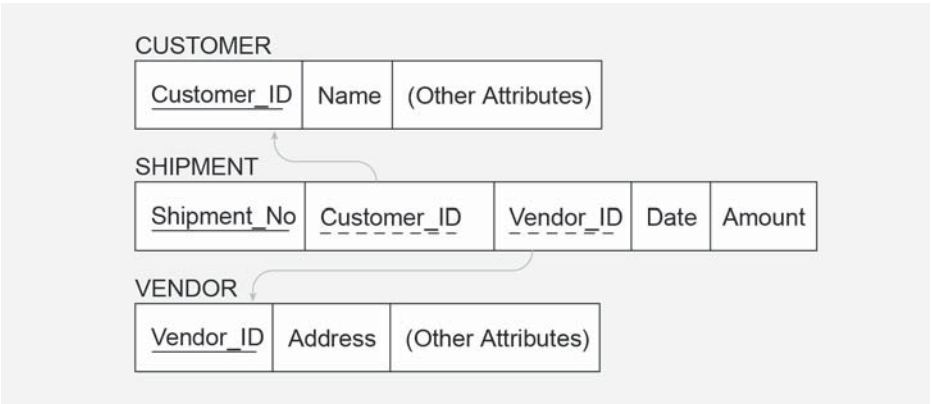
Gli attributi non di chiave associati a SHIPMENT sono Date e Amount.

Il risultato della mappatura di questa entità in un insieme di relazioni è mostrato nella Figura 7.16b. La nuova relazione associativa è chiamata SHIPMENT. La chiave primaria è Shipment_No. Customer_ID e Vendor_ID sono inclusi come chiavi esterne nella relazione, e Date e Amount sono attributi non di chiave.

Figura 7.16
Mappare un'entità associativa con un identificatore.
(a) Entità associativa (SHIPMENT).



(b) Tre relazioni.



7.3.5 Passaggio 5: mappare le associazioni unarie

Nel Capitolo 5 abbiamo definito l'associazione unaria come un'associazione tra le istanze di un singolo tipo di entità. Le associazioni unarie sono anche chiamate *associazione ricorsive*. I due casi di associazioni unarie più importanti sono quella uno-a-molti e quella molti-a-molti. Discuteremo i due esempi separatamente, poiché l'approccio alla mappatura è parzialmente diverso per i due tipi.

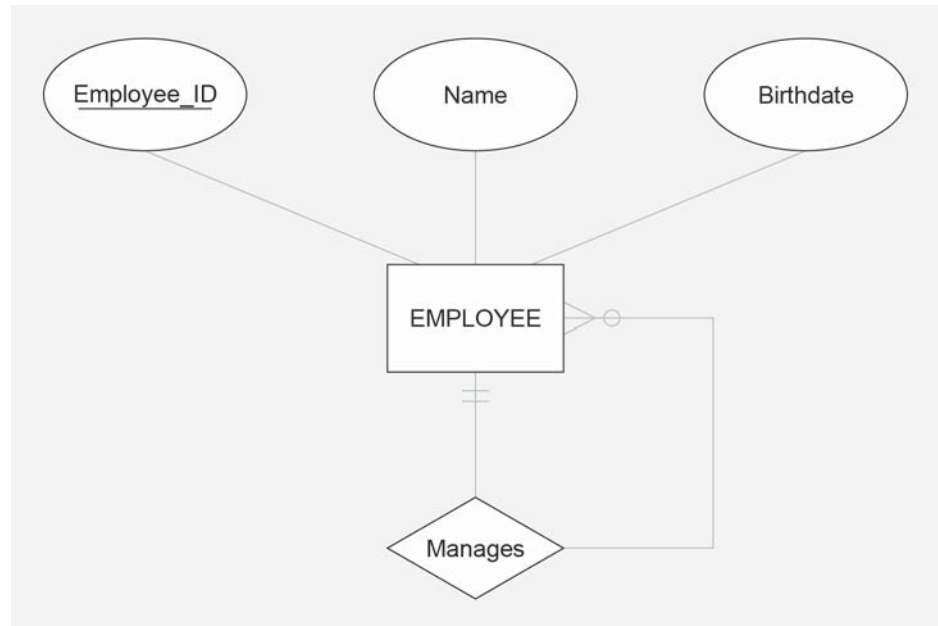
Associazioni unarie uno-a-molti

Il tipo di entità nell'associazione unaria viene mappato in una relazione utilizzando la procedura descritta nel passaggio 1. Viene quindi aggiunto un attributo di chiave esterna *all'interno* della stessa relazione che fa riferimento ai valori di chiave primari (questa chiave esterna deve avere lo stesso dominio di quella primaria). Una **chiave esterna ricorsiva** è una chiave esterna in una relazione che fa riferimento ai valori di chiave primaria della stessa relazione.

La Figura 7.17a mostra un'associazione unaria uno-a-molti chiamata *Manages* che associa ciascun impiegato di un'azienda a un altro impiegato che è il suo manager. Ogni impiegato ha esattamente un manager; un dato impiegato può essere manager di un numero di impiegati da zero a molti.

Figura 7.17

Mappare un'associazione unaria 1:N.
(a) Entità *EMPLOYEE* con associazione *Manages*.



(b) Relazione *EMPLOYEE* con chiave esterna ricorsiva.



La relazione *EMPLOYEE* che risulta dalla mappatura di questa entità e dell'associazione è mostrata nella Figura 7.17b. La chiave esterna ricorsiva nella relazione è chiamata *Manager_ID*. Questo attributo ha lo stesso dominio della chiave primaria *Employee_ID*. Ogni riga di questa relazione memorizza i dati seguenti per un dato impiegato: *Employee_ID*, *Name*, *Birthdate* e *Manager_ID* (ossia *Employee_ID* per il manager di questo impiegato). Si noti che, poiché si tratta di una chiave esterna, *Manager_ID* si riferisce a *Employee_ID*.

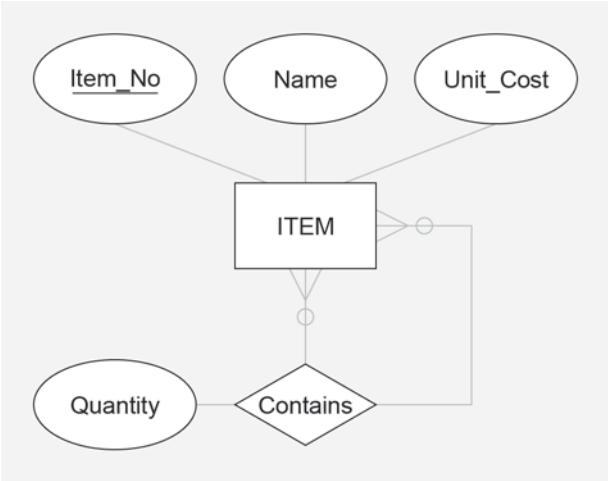
Associazioni unarie multi-a-molti

Con questo tipo di associazioni, vengono create due relazioni: una per rappresentare il tipo di entità nell'associazione e l'altra relazione associativa per rappresentare l'associazione M:N stessa. La chiave primaria della relazione associativa è costituita da due attributi. Questi attributi (che non devono avere per forza lo stesso nome) traggono entrambi i loro valori dalla chiave primaria dell'altra relazione. Un attributo non di chiave dell'associazione è incluso nella relazione associativa.

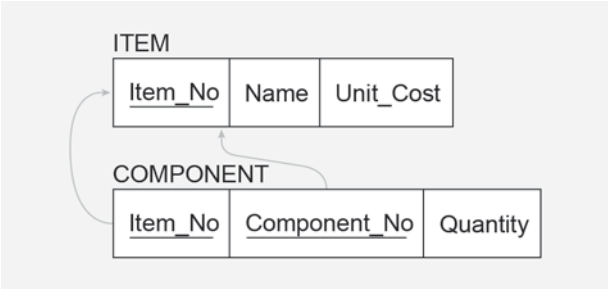
Un esempio di mappatura di un'associazione unaria M:N è mostrato nella Figura 7.18. La Figura 7.18a mostra un'associazione bill-of-materials tra articoli che vengono assemblati da altri articoli o componenti (questa struttura è stata descritta nel Capitolo 5 e un esempio appare nella Figura 5.13). L'associazione, chiamata Contains, è M:N, poiché un dato articolo può contenere numerosi articoli componenti e, all'opposto, un articolo può essere utilizzato come componente in molti altri articoli.

Le relazioni risultanti dalla mappatura di questa entità e delle sue associazioni sono mostrate nella Figura 7.15b. La relazione ITEM è mappata direttamente dallo stesso tipo di entità. COMPONENT è una relazione associativa la cui chiave primaria è costituita da due attributi denominati arbitrariamente Item_No e Component_No. L'attributo Quantity è un attributo non di chiave di questa relazione che, per un dato articolo, registra la quantità di un determinato articolo componente utilizzato in quell'articolo. Si noti che sia _No sia Component_No fanno riferimento alla chiave primaria (Item_No) della relazione ITEM.

Figura 7.18
Mappare un'associazione unaria M:N.
(a) Associazioni bill-of-materials (M:N).



(b) Relazioni ITEM e COMPONENT.



Le relazioni sopra descritte possono essere facilmente interrogate per determinare, per esempio, i componenti di un dato articolo. L'interrogazione SQL seguente elencherà direttamente i componenti (e la loro quantità) per l'articolo numero 100:

```
SELECT Component_No, Quantity
FROM COMPONENT
WHERE Item_No = 100;
```

7.3.6 Passaggio 6: mappare le associazioni ternarie (e *n*-arie)

Nel Capitolo 5 si è spiegato che un'associazione ternaria è un'associazione tra tre tipi di entità. In quel capitolo si è consigliato di convertire un'associazione ternaria in un'entità associativa per rappresentare i vincoli di partecipazione più accuratamente.

Per mappare un tipo di entità associativa che collega tre tipi di entità regolari, si dovrà creare una nuova relazione associativa. La chiave primaria predefinita di questa relazione è costituita dai tre attributi di chiave primaria per i tipi di entità partecipanti (a volte vengono richiesti degli attributi supplementari per formare una chiave primaria univoca). Questi attributi agiscono quindi nel ruolo di chiavi esterne che fanno riferimento alle singole chiavi primarie dei tipi di entità partecipanti. Tutti gli attributi del tipo di entità diventano attributi della nuova relazione.

Un esempio di mappatura di un'associazione ternaria (rappresentata come tipo di entità associativa) è mostrata nella Figura 7.19. La Figura 7.19a è un segmento di un diagramma E-R (o di una vista) che rappresenta un *paziente* che riceve un *trattamento* da un *medico*. Il tipo di entità associativa PATIENT TREATMENT ha gli attributi Date, Time e Results; i valori vengono registrati per questi attributi per ogni istanza di PATIENT TREATMENT.

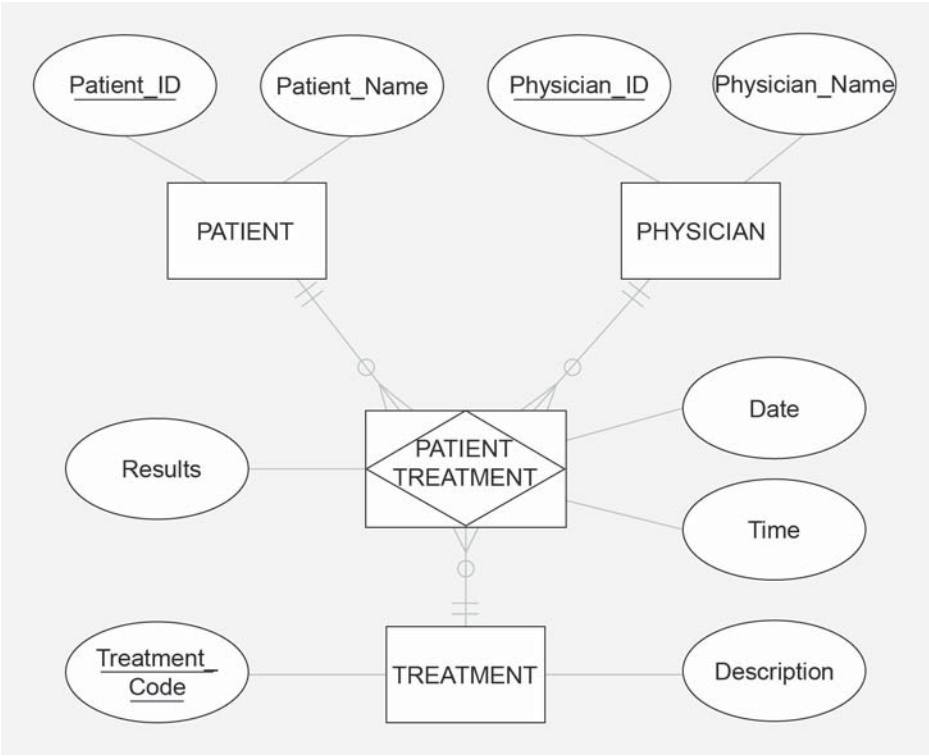
Il risultato della mappatura di questa vista è mostrato nella Figura 7.19b. Gli attributi di chiave primaria Patient_ID, Physician_ID e Treatment_Code diventano chiavi esterne in PATIENT TREATMENT. Questi attributi sono le chiavi esterne di PATIENT TREATMENT. Tuttavia, non identificano in modo univoco un dato trattamento, poiché un paziente può ricevere lo stesso trattamento dallo stesso medico in più di un'occasione. L'inclusione dell'attributo Date come parte della chiave primaria (insieme agli altre tre attributi) dà come risultato una chiave primaria? Sarebbe così se un dato paziente ricevesse solo un trattamento da un particolare medico in una data specifica. Questa è tuttavia una situazione improbabile. Per esempio, un paziente potrebbe ricevere un trattamento al mattino e poi nuovamente nel pomeriggio. Per risolvere questo problema, si include Time come parte della chiave primaria. La chiave primaria di PATIENT TREATMENT è quindi costituita dai cinque attributi mostrati nella Figura 7.19b: Patient_ID, Physician_ID, Treatment_Code, Date e Time. L'unico attributo non di chiave nella relazione è Results.

7.3.7 Passaggio 7: mappare le associazioni supertipo/sottotipo

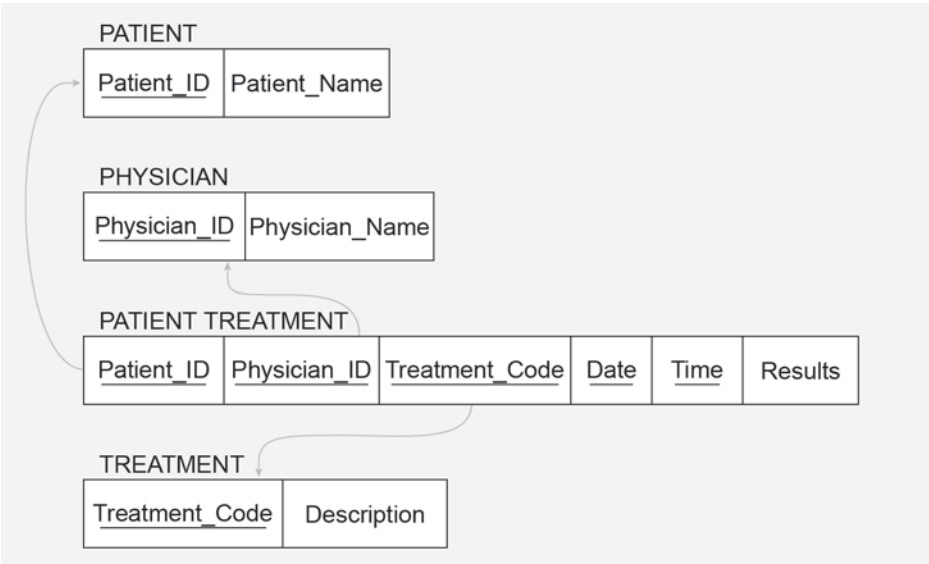
Il modello di dati relazionale non supporta ancora direttamente le associazioni supertipo/sottotipo. Fortunatamente, ci sono varie strategie che i progettisti delle basi di dati possono utilizzare per rappresentare queste associazioni con il modello di dati relazionale (Chouinard, 1898). Per i nostri scopi, utilizzeremo la strategia illustrata di seguito, che è quella impiegata più di frequente.

1. Creare una relazione separata per il supertipo e per ognuno dei suoi sottotipi.

Figura 7.19
Mappare un'associazione ternaria.
(a) Associazione ternaria con entità associativa.



(b) Mappare l'associazione ternaria.



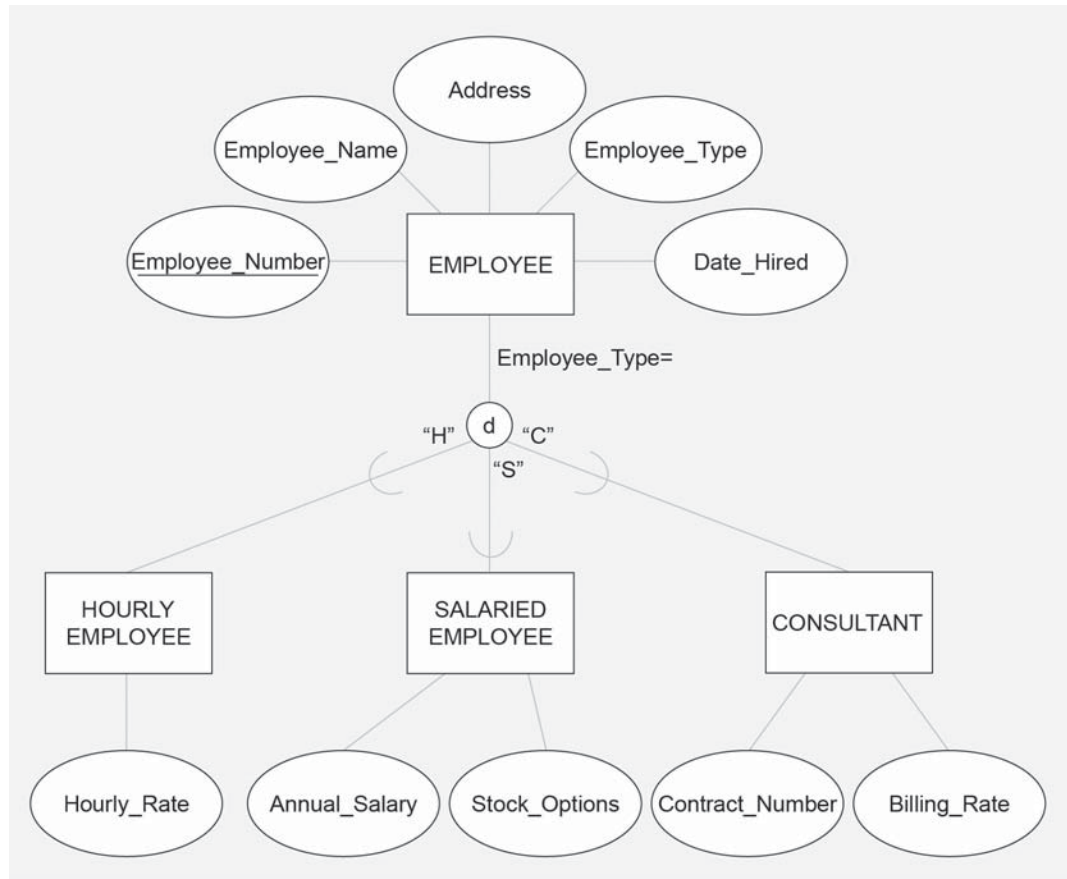
- 2. Assegnare alla relazione creata per il supertipo gli attributi che sono comuni a tutti i membri del supertipo, compresa la chiave primaria.

3. Assegnare alla relazione per ogni sottotipo la chiave primaria del supertipo e solo quegli attributi che sono univoci per quel sottotipo.
4. Assegnare uno o più attributi del supertipo alla funzione come discriminante di sottotipo (il ruolo della discriminante di sottotipo è descritto nel Capitolo 6).

Un esempio dell'applicazione di questa procedura è mostrato nelle Figure 7.20 e 7.21. La Figura 7.20 mostra il supertipo EMPLOYEE con i sottotipi HOURLY EMPLOYEE, SALARIED EMPLOYEE e CONSULTANT (questo esempio è descritto nel Capitolo 6; la Figura 7.20 riprende la Figura 6.8). La chiave primaria di EMPLOYEE è Employee_Number, e l'attributo Employee_Type è la discriminante di sottotipo.

Figura 7.20

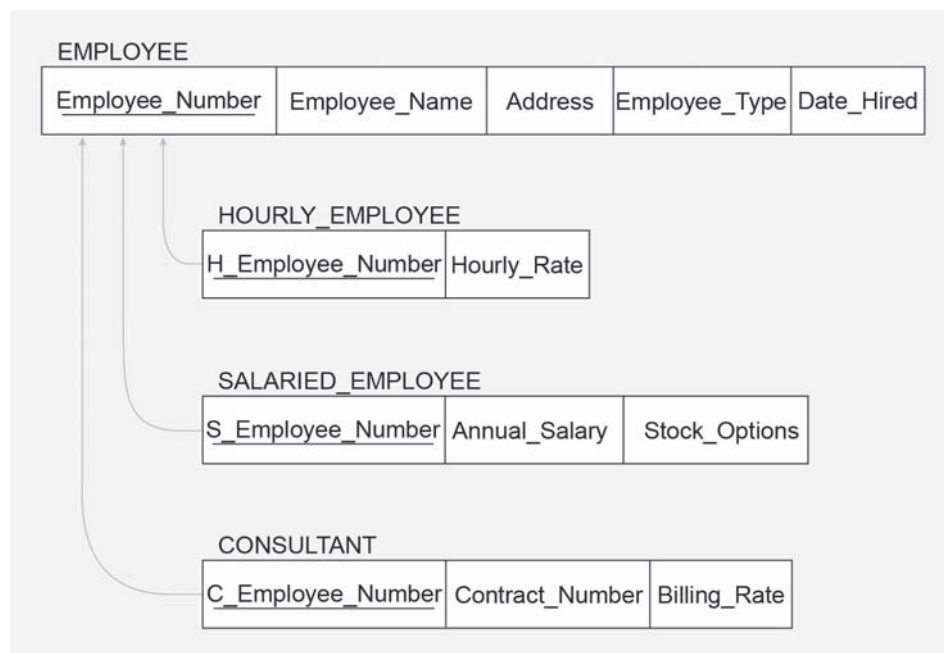
Associazioni supertipo/
sottotipo.



Il risultato della mappatura di questo diagramma in relazioni utilizzando le regole sopra citate è mostrato nella Figura 7.21. C'è una relazione per il supertipo (EMPLOYEE) e una per ognuno dei tre sottotipi. La chiave primaria per ognuna delle quattro relazioni è Employee_Number. Viene utilizzato un prefisso per distinguere il nome della chiave primaria per ciascun sottotipo. Per esempio, S_Employee_Number è il nome della chiave primaria della relazione SALARIED_EMPLOYEE. Ognuno di

Figura 7.21

Mappare le associazioni supertipo/sottotipo in relazioni.



questi attributi è una chiave esterna che fa riferimento alla chiave primaria di supertipo, come indicato dalle frecce nel diagramma. Ogni relazione di sottotipo contiene solo gli attributi peculiari di quel sottotipo.

Per ogni sottotipo, può essere prodotta una relazione che contenga *tutti* gli attributi di quel sottotipo (sia specifici sia ereditati) utilizzando il seguente comando

```

SELECT *
FROM EMPLOYEE, SALARIED_EMPLOYEE
WHERE Employee_Number = S_Employee_Number;
  
```

Sebbene non si siano ancora studiati formalmente questi comandi, si può intuire che tale comando unisce le due tabelle e produce una tabella più grande che contiene tutti gli attributi tratti da entrambe le tabelle.

7.4 Introduzione alla normalizzazione

La normalizzazione è un processo formale per decidere quali attributi dovrebbero essere raggruppati in una relazione. Nei Capitoli 5 e 6 è stato usato un po' di buon senso per raggruppare gli attributi in tipi di entità durante la modellazione concettuale dei dati. Nel paragrafo precedente di questo capitolo si è imparato a mappare i diagrammi E-R in relazioni. Prima di procedere con la progettazione fisica, occorrerà un metodo per convalidare la progettazione logica. La normalizzazione è principalmente uno strumento per la convalida e il miglioramento della progettazione logica, così che questa soddisfi determinati vincoli che evitino la duplicazione non necessaria dei dati.

Finora abbiamo presentato una trattazione intuitiva delle relazioni ben strutturate; tuttavia occorrono alcune definizioni formali di tali relazioni, oltre a un processo per progettarele. La normalizzazione

è il processo di scomposizione delle relazioni che contengono anomalie per produrre relazioni più piccole e ben strutturate. Per esempio, abbiamo già utilizzato i principi della normalizzazione per convertire la tabella EMPLOYEE2 (con la sua ridondanza) in EMPLOYEE1 (Figura 7.1) ed EMP_COURSE (Figura 7.7).

7.4.1 I passaggi della normalizzazione

La normalizzazione può essere ottenuta e capita attraverso una serie di passaggi, ognuno dei quali corrisponde a una forma normale (Figura 7.22). Una forma normale è uno stato di una relazione che deriva dall'applicazione di alcune regole semplici che riguardano le dipendenze funzionali (o le associazioni tra gli attributi) su quella relazione. In questo paragrafo descriveremo brevemente tali regole e le illustreremo nel dettaglio nei prossimi.

1. *Prima forma normale*: ogni attributo multivalore (chiamato anche *gruppo ripetente*) è stato eliminato, creando così un singolo valore (probabilmente nullo) all'intersezione di ogni riga e colonna della tabella (come nella Figura 7.2b).
2. *Seconda forma normale*: ogni dipendenza funzionale parziale è stata eliminata.
3. *Terza forma normale*: ogni dipendenza transitiva è stata eliminata.
4. *Forma normale di Boyce-Codd*: ogni anomalia residua che risulta dalle dipendenze funzionali che sono state eliminate.
5. *Quarta forma normale*: ogni dipendenza multivalore è stata eliminata.
6. *Quinta forma normale*: ogni anomalia residua è stata eliminata.

In questo capitolo descriveremo le forme normali dalla prima alla terza. Le forme normali restanti verranno descritte nell'Appendice B.

7.4.2 Dipendenze funzionali e chiavi

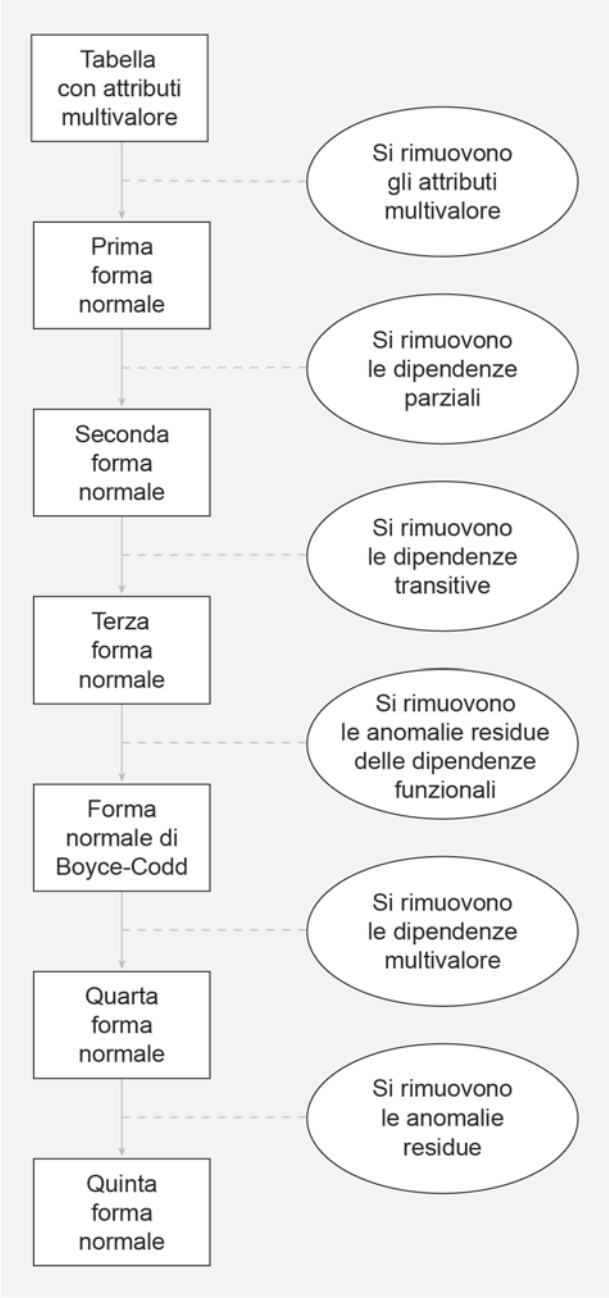
La normalizzazione si basa sull'analisi delle dipendenze funzionali. Una dipendenza funzionale è un vincolo tra due attributi o due insiemi di attributi. Per qualsiasi relazione R, l'attributo B è funzionalmente dipendente dall'attributo A se, per ogni istanza valida di A, quel valore di A determina in modo univoco il valore di B (Dutka e Hansom, 1989). La dipendenza funzionale di B da A è rappresentata da una freccia, come in $A \rightarrow B$. Un attributo può essere funzionalmente dipendente da due o più attributi invece che da uno solo. Per esempio, si consideri la relazione EMP_COURSE (Emp_ID, Course_Title, Date_Completed) mostrata nella Figura 7.7. La dipendenza funzionale in questa relazione è rappresentata come segue:

Emp_ID, Course_Title $\rightarrow$ Date_Completed.

La dipendenza funzionale in questa istruzione implica che la data in cui un corso viene completato è determinata per intero dall'identità dell'impiegato e dal titolo del corso. Esempi comuni di dipendenze funzionali sono le seguenti.

1. SSN $\rightarrow$ Name, Address, Birthdate: il nome, l'indirizzo e la data di nascita di una persona sono funzionalmente dipendenti dal numero della previdenza sociale di quella persona.
2. VIN $\rightarrow$ Make, Model, Color: la marca, il modello e il colore di un veicolo sono funzionalmente dipendenti dal numero di targa del veicolo.

Figura 7.22
Passaggi della normalizzazione.



3. ISBN → Title, First_Author_Name: il titolo di un libro e il nome dell'autore sono funzionalmente dipendenti dal numero ISBN del libro.

Determinanti

L'attributo sul lato sinistro di una freccia in una dipendenza funzionale è chiamato **determinante**. Negli esempi precedenti, SSN, VIN e ISBN sono determinanti. Nella relazione EMP_COURSE (Figura 7.7) la combinazione di Emp_ID e Course_Title è una determinante.

Chiavi candidate

Una **chiave candidata** è un attributo, o una combinazione di attributi, che identifica in modo univoco una riga in una relazione. Una chiave candidata deve soddisfare le seguenti proprietà (Dutka e Hanson, 1989), che sono un sottoinsieme delle sei proprietà di una delle chiavi primarie precedentemente elencate.

1. *Identificazione univoca.* Per ogni riga, il valore della chiave deve identificare quella riga in modo univoco. *Questa proprietà implica che ogni attributo non di chiave sia funzionalmente dipendente da quella chiave.*
2. *Non ridondanza.* Nessun attributo nella chiave può essere eliminato senza distruggere la proprietà dell'identificazione univoca.

Applichiamo ora la definizione precedente per identificare le chiavi candidate in due delle relazioni descritte in questo capitolo. La relazione EMPLOYEE1 (Figura 7.1) ha il seguente schema: EMPLOYEE1(Emp_ID, Name, Dept_Name, Salary). Emp_ID è la sola determinante in questa relazione. Tutti gli altri attributi sono funzionalmente dipendenti da Emp_ID. Emp_ID è quindi una chiave candidata, e poiché non ci sono altre chiavi candidate, è anche la chiave primaria.

Rappresentiamo le dipendenze funzionali per una relazione utilizzando la notazione mostrata nella Figura 7.23. La Figura 7.23a mostra la rappresentazione per EMPLOYEE1. La linea orizzontale nella figura rappresenta le dipendenze funzionali. Una linea verticale scende dalla chiave primaria (Emp_ID) e si connette alla linea orizzontale. Le frecce verticali puntano a ognuno degli attributi non di chiave che sono funzionalmente dipendenti dalla chiave primaria.

Per la relazione EMPLOYEE2 (Figura 7.2b) si noti che, a differenza di EMPLOYEE1, Emp_ID non identifica in modo univoco una riga nella relazione. Per esempio, ci sono due righe nella tabella per Emp_ID numero 100. In questa relazione ci sono due dipendenze funzionali.

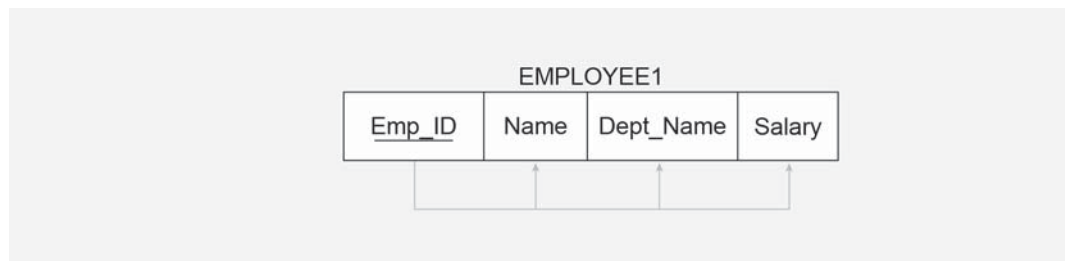
1. Emp_ID → Name, Dept_Name, Salary
2. Emp_ID, Course_Title → Date_Completed

Le dipendenze funzionali indicano che la combinazione di Emp_ID e Course_Title è la sola chiave candidata (e quindi è la chiave primaria) per EMPLOYEE2. In altre parole, la chiave primaria di EMPLOYEE2 è una chiave composta. Né Emp_ID né Course_Title identificano in modo univoco una riga in questa relazione e quindi la proprietà 1 non può essere di per sé una chiave candidata. Si osservino i dati nella Figura 7.2b per verificare che la combinazione di Emp_ID e Course_Title non identifichino in modo univoco ogni riga di EMPLOYEE2. Rappresentiamo le dipendenze funzionali di questa relazione nella Figura 7.23b. Si noti che Date_Completed è il solo attributo che è funzionalmente dipendente dalla chiave primaria costituita dagli attributi Emp_ID e Course_Title.

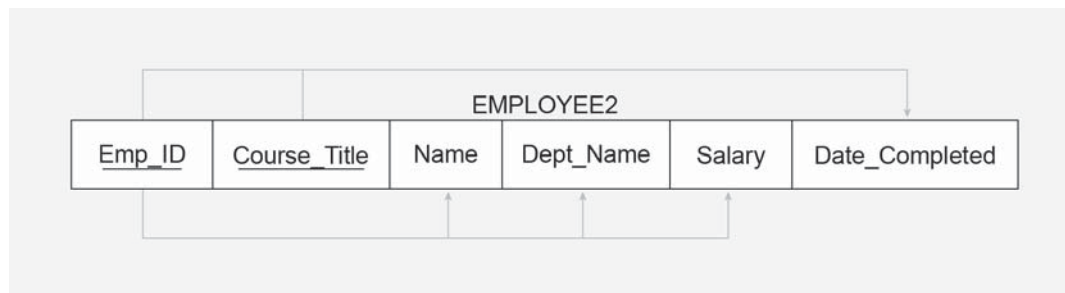
L'associazione tra determinanti e chiavi candidate si può riassumere come segue: una chiave candidata è sempre una determinante, mentre una determinante può essere o non essere una chiave candidata. Per esempio, in EMPLOYEE2, Emp_ID è una determinante ma non una chiave candidata. Una chiave candidata è una determinante che identifica in modo univoco gli attributi (non di chiave) rimanenti in una relazione. Una determinante può essere una chiave candidata (come Emp_ID in EMPLOYEE1), parte di una chiave candidata composta (come Emp_ID in EMPLOYEE2) o un attributo non di chiave. Proporremo alcuni esempi tra breve.

Figura 7.23

Rappresentare le dipendenze funzionali.
(a) Dipendenze funzionali in EMPLOYEE1.



(b) Dipendenze funzionali in EMPLOYEE2.



7.5 Le forme normali di base

Ora che abbiamo esaminato le dipendenze funzionali e le chiavi, siamo pronti per descrivere e illustrare le forme normali dalla prima alla terza. Descriveremo anche la normalizzazione dei dati di riepilogo che appaiono nelle basi di informazioni.

7.5.1 Prima forma normale

Una relazione è nella **prima forma normale (1NF)** se non contiene attributi multivalore. Si ricordi che la prima proprietà di una relazione è che il valore all'intersezione di ogni riga e colonna deve essere atomico. Una tabella che contiene attributi multivalore o gruppi ripetenti non è quindi una relazione.

Nel paragrafo precedente, nel descrivere come mappare i diagrammi E-R in relazioni, è stata descritta una procedura per eliminare gli attributi multivalore dai tipi di entità nel diagramma E-R. Di conseguenza, se è stata sviluppata una progettazione logica trasformando i diagrammi E-R in relazioni, non dovrebbero esserci attributi multivalore rimanenti. Tuttavia, alcuni vecchi sistemi legacy scritti in linguaggi come il COBOL supportavano attributi multivalore. Poiché può accadere di convertire questi vecchi sistemi in basi di dati relazionali, è importante capire come eliminare gli attributi multivalore.

7.5.2 Seconda forma normale

Una relazione è nella **seconda forma normale (2NF)** se è nella prima forma normale e ogni attributo non di chiave è del tutto dipendente funzionalmente dalla chiave primaria. Di conseguenza, nessun attributo non di chiave è funzionalmente dipendente da parte della chiave primaria (ma non da tutta). Una relazione che è nella prima forma normale sarà nella seconda forma normale se si verifica una delle seguenti condizioni.

1. La chiave primaria è costituita solo da un attributo (come l'attributo Emp_ID in EMPLOYEE1).
2. Nella relazione non esistono attributi non di chiave (quindi tutti gli attributi nella relazione sono componenti della chiave primaria).
3. Ogni attributo non di chiave è funzionalmente dipendente dall'intero insieme di attributi di chiave primaria.

EMPLOYEE2 (Figura 7.2b) è un esempio di relazione che non è nella seconda forma normale. La chiave primaria di questa relazione è la chiave composta Emp_ID, Course_Title. Gli attributi non di chiave Name, Dept_Name e Salary sono quindi funzionalmente dipendenti da parte della chiave primaria (Emp_ID) ma non da Course_Title. Tali dipendenze sono mostrate graficamente nella Figura 7.23b. Una funzione parzialmente dipendente è una dipendenza funzionale nella quale uno o più attributi non di chiave (come Name) sono funzionalmente dipendenti da parte della chiave primaria (ma non da tutta). La funzione parzialmente dipendente in EMPLOYEE2 crea ridondanza in quella relazione, cosa che provoca delle anomalie quando la tabella viene aggiornata come indicato nel paragrafo precedente.

Per convertire una relazione nella seconda forma normale, occorre scomporre la relazione in nuove relazioni che soddisfino una o più condizioni tra quelle sopra descritte. EMPLOYEE2 viene scomposta nelle due relazioni seguenti.

1. EMPLOYEE1(Emp_ID,Name,Dept_Name,Salary). Questa relazione soddisfa la prima condizione ed è nella seconda forma normale (i dati di esempio sono mostrati nella Figura 7.1).
2. EMP_COURSE(Emp_ID, Course_Title,Date_Completed). Questa relazione soddisfa la terza condizione ed è anch'essa nella seconda forma normale (i dati di esempio sono mostrati nella Figura 7.7).

Esamineremo queste nuove relazioni per verificare che non contengano le anomalie associate a EMPLOYEE2.

7.5.3 Terza forma normale

Una relazione è nella terza forma normale (3NF) se è nella seconda forma normale e non esistono dipendenze transitive. Una dipendenza transitiva in una relazione è una dipendenza funzionale tra due o più attributi non di chiave. Per esempio, si consideri la seguente relazione:

SALES(Cust_ID,Name,Salesperson,Region)

(I dati di esempio per questa relazione appaiono nella Figura 7.24a).

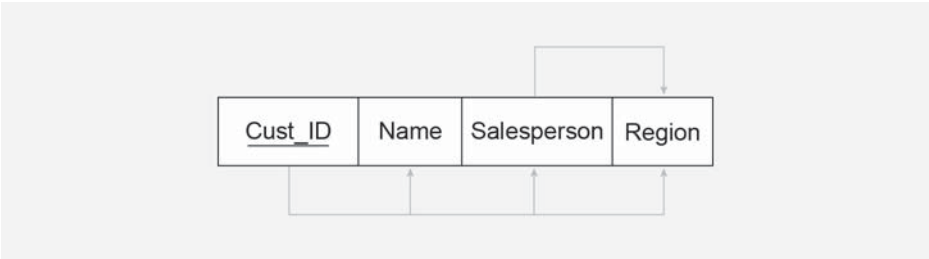
Le dipendenze funzionali nella relazione SALES sono mostrate graficamente nella Figura 7.24b. Cust_ID è la chiave primaria, cosicché tutti gli attributi rimanenti sono funzionalmente dipendenti da questo attributo. C'è tuttavia una dipendenza transitiva (Region è funzionalmente dipendente da Salesperson e Salesperson è funzionalmente dipendente da Cust_ID). Il risultato è che in SALES ci sono tre anomalie di aggiornamento.

1. *Anomalia di inserimento.* Un nuovo agente di vendita (Robinson) assegnato alla zona nord non può essere immesso fino a che un cliente non è stato assegnato a quell'agente (poiché deve essere fornito un valore per Cust_ID per inserire una riga nella tabella).
2. *Anomalia di eliminazione.* Se il cliente 6837 viene eliminato dalla tabella, si perde l'informazione che dice che l'agente di vendita Hernandez è stato assegnato alla zona est.
3. *Anomalia di modifica.* Se l'agente di vendita Smith viene riassegnato alla zona est, dovranno essere modificate numerose righe per riflettere questo fatto (due righe sono mostrate nella Figura 7.24a).

Figura 7.24
Relazione con dipendenza
transitiva.
(a) Relazione SALES con dati
di esempio.

SALES			
<u>Cust_ID</u>	Name	Salesperson	Region
8023	Anderson	Smith	South
9167	Bancroft	Hicks	West
7924	Hobbs	Smith	South
6837	Tucker	Hernandez	East
8596	Eckersley	Hicks	West
7018	Arnold	Faulb	North

(b) Dipendenza transitiva nella
relazione SALES.



Queste anomalie sono il risultato della dipendenza transitiva. La dipendenza transitiva può essere eliminata scomponendo SALES in due relazioni, come mostrato nella Figura 7.25a. Si noti che Salesperson, che è la determinante nella dipendenza transitiva in SALES, diventa la chiave primaria in SPERSON. Salesperson diventa una chiave esterna in SALES1. (Poiché Salesperson è una chiave aziendale intelligente, non è una buona scelta come chiave primaria. Il concetto è spiegato brevemente nel Capitolo 5 e verrà elaborato successivamente in questo capitolo.) Una volta implementate, le relazioni normalizzate della Figura 7.25 utilizzano spesso meno spazio di memoria di quanto non faccia una base di dati che si basa sulla relazione nella Figura 7.24a; questo accade perché i dati dipendenti (Region e altri dati dell’agente di vendita) non devono essere ripetuti per ciascun cliente. La normalizzazione, oltre a risolvere le anomalie, riduce quindi spesso una memorizzazione dei dati ridondante.

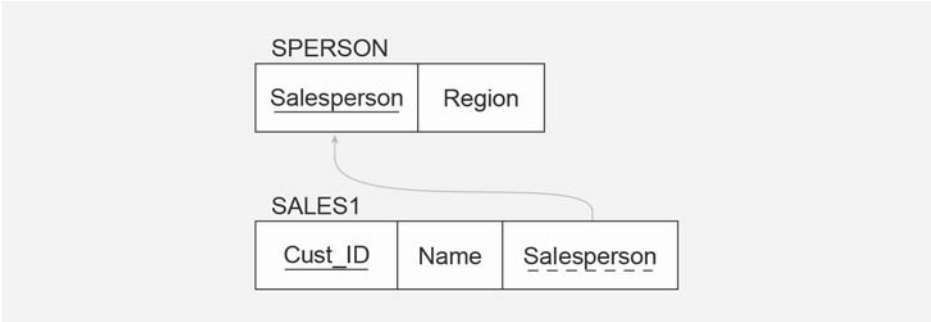
Come mostrato nella Figura 7.25b, le nuove relazioni sono ora nella terza forma normale, poiché non esistono dipendenze transitive. Si dovrebbe controllare che le anomalie esistenti in SALES non siano presenti in SALES1 e SPERSON.

Le dipendenze transitive possono verificarsi anche tra insiemi di attributi in una relazione. Per esempio, la relazione SHIPMENT (Snum,Origin,Destination,Distance) potrebbe essere utilizzata per registrare le spedizioni in base all’origine, alla destinazione e alla distanza (Dutka e Hanson, 1989). I dati di esempio per questa relazione appaiono nella Figura 7.26a. Le dipendenze funzionali nella relazione SHIPMENT sono mostrate nella Figura 7.26b. La chiave primaria della relazione SHIPMENT è l’attributo Snum (Shipment Number). Di conseguenza, si sa che la relazione è nella seconda forma normale (perché?). In questa relazione esiste tuttavia una dipendenza transitiva: l’attributo Distance è funzionalmente dipendente dalla *coppia* di attributi non di chiave Origin e Destination. Come risultato, si hanno alcune anomalie in SHIPMENT (come esercizio, si può esaminare la Figura 7.26a e provare a

Figura 7.25
Rimuovere una dipendenza
transitiva.
(a) Scomporre la relazione SALES.

SALES1			SPERSON	
<u>Cust_ID</u>	Name	<u>Salesperson</u>	<u>Salesperson</u>	Region
8023	Anderson	Smith	Smith	South
9167	Bancroft	Hicks	Hicks	West
7924	Hobbs	Smith	Hernandez	East
6837	Tucker	Hernandez	Faulb	North
8596	Eckersley	Hicks		
7018	Arnold	Faulb		

(B) Relazioni in 3NF.



individuare le anomalie di inserimento, eliminazione e modifica). È possibile rimuovere la dipendenza transitiva in SHIPMENT scomponendola in due relazioni (entrambe in 3NF):

SHIPTO(SNum,Origin,Destination)
DISTANCES(Origin, Destination, Distance)

La prima di queste relazioni fornisce l'origine e la destinazione per qualsiasi spedizione, mentre la seconda fornisce la distanza tra una coppia di origine e destinazione. I dati di esempio per queste relazioni appaiono nella Figura 7.26c.

7.5.4 Normalizzare i dati di riepilogo

Le basi di dati per il supporto delle decisioni gestionali in un'azienda contengono spesso sottoinsiemi e riepiloghi di dati delle basi di dati operazionali. Questi data warehouse utilizzati per supportare livelli di gestione più alti vengono spesso normalizzati (entro un certo limite) per evitare le stesse anomalie che sorgono nelle basi di dati operazionali. Descriveremo i problemi legati alla normalizzazione dei dati nei data warehouse nel Capitolo 13.

7.6 Unire le relazioni

In precedenza abbiamo spiegato come trasformare i diagrammi E-R in relazioni e abbiamo anche descritto come controllare le relazioni per determinare se sono nella terza forma normale e come eseguire i passaggi necessari per la normalizzazione.

Figura 7.26

Un altro esempio di dipendenze transitive.
(a) Relazione SHIPMENT con dati di esempio.

SHIPMENT			
<u>Snum</u>	Origin	Destination	Distance
409	Seattle	Denver	1,537
618	Chicago	Dallas	1,058
723	Boston	Atlanta	1,214
824	Denver	Los Angeles	1,150
629	Minneapolis	St. Louis	587

(b) Dipendenze funzionali in SHIPMENT.



(c) Relazioni in 3NF.

SHIPTO			DISTANCES		
<u>Snum</u>	<u>Origin</u>	<u>Destination</u>	<u>Origin</u>	<u>Destination</u>	Distance
409	Seattle	Denver	Seattle	Denver	1,537
618	Chicago	Dallas	Chicago	Dallas	1,058
723	Boston	Atlanta	Boston	Atlanta	1,214
824	Denver	Los Angeles	Denver	Los Angeles	1,150
629	Minneapolis	St. Louis	Minneapolis	St. Louis	587

Come parte del processo di progettazione logica, le relazioni normalizzate possono essere state create da diversi diagrammi E-R separati ed eventualmente da altre viste utente. Alcune delle relazioni possono essere ridondanti, ossia possono fare riferimento alle stesse entità. In questo caso, è preferibile unire queste relazioni per rimuovere la ridondanza. Questo paragrafo descrive l'unione delle relazioni, chiamata anche *integrazione delle viste*. Capire come unire le relazioni è importante per tre ragioni.

1. Nei progetti grandi, il lavoro di più sottogruppi viene unito durante la progettazione logica; è quindi necessario unire le relazioni.
2. L'integrazione di basi di dati esistenti con nuove esigenze informative porta spesso alla necessità di integrare viste differenti.
3. Durante il ciclo di vita, possono emergere nuove esigenze informative; diventa quindi necessario unire qualsiasi relazione presente con quanto è già stato sviluppato.

7.6.1 Un esempio

Si supponga che la modellazione di una vista utente dia come risultato la seguente relazione 3NF:

EMPLOYEE1(Employee_ID,Name,Address,Phone)

Modellando una seconda vista utente, si può avere come risultato la seguente relazione:

EMPLOYEE2(Employee_ID,Name,Address,Jobocode,No_Years)

Poiché queste due relazioni hanno la stessa chiave primaria (Employee_ID), è probabile che descrivano la stessa entità e possono quindi trovarsi in una relazione. Il risultato dell'unione delle relazioni è la seguente relazione:

EMPLOYEE(Employee_ID,Name,Address,Phone Jobocode,No_Years)

Si noti che un attributo che appare in entrambe le relazioni (come Name in questo esempio) appare solo una volta nella relazione unita.

7.6.2 Problemi di integrazione delle viste

Quando si integrano le relazioni come nell'esempio precedente, l'analista della base di dati deve capire il significato dei dati e deve essere preparato a risolvere gli eventuali problemi che possono sorgere nel processo. In questo paragrafo descriveremo e illustreremo brevemente quattro problemi che sorgono nell'integrazione delle viste: *sinonimi*, *omonimi*, *dipendenze transitive* e *associazioni supertipo/sottotipo*.

Sinonimi

In alcune situazioni, due o più attributi possono avere nomi diversi ma lo stesso significato, come quando descrivono la stessa caratteristica di un'entità. Questi attributi sono chiamati **sinonimi**. Per esempio, Employee_ID ed Employee_No possono essere sinonimi. Quando si uniscono relazioni che contengono sinonimi, ci si dovrebbe accordare, se possibile, con gli utenti su un unico nome standardizzato per l'attributo ed eliminare altri sinonimi (un'alternativa può essere quella di scegliere un terzo nome per sostituire i sinonimi). Per esempio, si considerino le seguenti relazioni:

STUDENT1(Student_ID,Name)

STUDENT2(Matriculation_No,Name,Address)

In questo caso, l'analista riconosce che sia Student_ID sia Matriculation_No sono sinonimi per il numero di previdenza sociale di una persona e che sono identici (un'altra possibilità è che siano entrambi chiavi candidate e che solo uno di essi venga scelto come chiave primaria). Una soluzione possibile è quella di standardizzare uno o due nomi di attributi, come Student_ID. Un'altra possibilità è quella di utilizzare un nuovo nome di attributo, come SSN, per sostituire entrambi i sinonimi. Con quest'ultimo approccio, l'unione di due relazioni produrrebbe il seguente risultato:

STUDENT(SSN,Name,Address)

Quando ci sono dei sinonimi, è necessario permettere ad alcuni utenti della base di dati di fare riferimento alla stessa base di dati con nomi diversi. Gli utenti potrebbero aver bisogno di utilizzare nomi familiari che sono coerenti con la terminologia del loro settore dell'azienda. Un **alias** è il nome alternativo utilizzato per un attributo. Molti sistemi di gestione delle basi di dati permettono la definizione di un alias che può essere utilizzato in modo intercambiabile con l'etichetta di attributo principale.

Omonimi

Un attributo che può avere più di un significato è chiamato omonimo. Per esempio, il termine “conto” può fare riferimento a un conto corrente bancario, a un conto di risparmio, a un conto di mutuo o a un altro tipo di conto (“conto” si riferisce quindi a tipi di dati diversi, a seconda di come viene utilizzato).

Quando si uniscono le relazioni, occorre prestare attenzione agli omonimi. Si consideri il seguente esempio:

```
STUDENT1(Student_ID,Name,Address)
STUDENT2(Student_ID,Name,Phone_No,Address)
```

Parlando con gli utenti, l'analista può scoprire che l'attributo Address in STUDENT1 si riferisce all'indirizzo del campus di uno studente, mentre in STUDENT2 lo stesso attributo si riferisce al domicilio fisso dello studente. Per risolvere questo conflitto, occorrerà probabilmente creare nuovi attributi, così che la relazione unita diventi la seguente:

```
STUDENT(Student_ID,Name,Phone_No,Campus_Address,Permanent_Address)
```

Dipendenze transitive

Quando due relazioni 3NF sono unite in un'unica relazione, possono sorgere delle dipendenze transitive (descritte precedentemente nel capitolo). Per esempio, si considerino le due relazioni seguenti:

```
STUDENT1(Student_ID,Major)
STUDENT2(Student_ID,Advisor)
```

Poiché STUDENT1 e STUDENT2 hanno la stessa chiave primaria, le due relazioni possono essere unite:

```
STUDENT(Student_ID,Major,Advisor)
```

Si supponga tuttavia che ogni specializzazione (*major*) abbia un tutor (*advisor*). In questo caso Advisor dipende funzionalmente da Major:

Major → Advisor

Se è presente questa dipendenza funzionale, allora STUDENT è in 2NF ma non in 3NF, poiché contiene una dipendenza transitiva. L'analista può creare relazioni 3NF rimuovendo la dipendenza transitiva (Major diventa una chiave esterna in STUDENT):

```
STUDENT(Student_ID,Major)
MAJOR ADVISOR(Major,Advisor)
```

Associazioni supertipo/sottotipo

Queste associazioni possono essere nascoste nelle viste utente o nelle relazioni. Si supponga di avere le due relazioni seguenti in un ospedale:

```
PATIENT1(Patient_ID,Name,Address)
PATIENT2(Patient_ID,Room_No)
```

Inizialmente, può sembrare che queste due relazioni possano essere unite in una singola relazione PATIENT. L'analista sospetta tuttavia correttamente che ci siano due tipi diversi di pazienti: interni ed esterni. PATIENT1 contiene effettivamente attributi comuni a *tutti* i pazienti. PATIENT2 contiene un attributo (Room_No) che è una caratteristica dei soli pazienti interni. In questa situazione, l'analista deve creare un'associazione supertipo/sottotipo per queste entità:

PATIENT(Patient_ID,Name,Address)
 RESIDENT PATIENT(Patient_ID, Room_No)
 OUTPATIENT(Patient_ID,Date,Treated)

Per una trattazione estesa dell'integrazione delle viste nella progettazione di una base di dati, si veda Navathe, Elmasri e Larson (1986).

7.7 L'ultima tappa: definire le chiavi relazionali

Nel Capitolo 3 abbiamo fornito alcuni criteri per la scelta degli *identificatori*: non cambiando il valore nel tempo, devono essere univoci, noti e non intelligenti e devono utilizzare un unico surrogato di attributo nel caso di un identificatore composto. In realtà nessuno di questi criteri deve essere applicato fino a che la base di dati non è stata implementata (ossia, fino a quando l'identificatore non diventa una chiave primaria ed è definito come campo nella base di dati fisica). Prima che le relazioni vengano definite come tabelle, le chiavi primarie delle tabelle devono essere modificate, se necessario, in modo da adattarsi a questi criteri.

Recentemente, alcuni esperti delle basi di dati (Johnston, 2000) hanno rafforzato i criteri per la specifica delle chiavi primarie. Inoltre raccomandano che una chiave primaria sia univoca nell'intera base di dati (diventando una cosiddetta **chiave aziendale**) e non solo nella tabella relazionale a cui si applica. Questo criterio rende una chiave primaria più simile a quello che nelle basi di dati orientate agli oggetti è chiamato *identificatore di oggetto*. Con questa raccomandazione, la chiave primaria di una relazione diventa un valore interno al sistema della base di dati e non ha un significato aziendale.

Una chiave primaria candidata, come Emp_ID nella relazione EMPLOYEE1 della Figura 7.1 o Salesperson nella relazione SPERSON della Figura 7.25, se è già stata utilizzata nell'azienda, è chiamata "chiave aziendale" e viene inclusa nella relazione come attributo non di chiave. Le relazioni EMPLOYEE1 e SPERSON (e qualsiasi altra relazione della base di dati) hanno quindi un nuovo attributo di chiave aziendale (chiamato, per esempio, Object_ID), che non ha un significato aziendale.

Perché creare questo attributo supplementare? Una delle ragioni principali che spiega una chiave aziendale è il fatto che una base di dati si evolve (unendo due relazioni in una base di dati una volta che questa viene creata). Per esempio, si considerino le seguenti relazioni:

EMPLOYEE(Emp_ID,Emp_Name,Dept_Name,Salary)
 CUSTOMER(Cust_ID,Cust_Name,Address)

In questo esempio, senza una chiave aziendale, Emp_ID e Cust_ID possono avere o non avere lo stesso formato, lunghezza e tipo di dati, sia che siano intelligenti o non intelligenti. Si supponga che l'azienda evolva le proprie esigenze informative e riconosca che gli impiegati possono essere anche clienti, così che impiegato e cliente siano semplicemente due sottotipi dello stesso supertipo PERSON. L'azienda avrebbe quindi tre relazioni:

PERSON(Person_ID,Person,Name)
 EMPLOYEE(Person_ID,Dept_Name,Salary)
 CUSTOMER(Person_ID,Address)

In questo caso, si suppone che Person_ID sia ovunque lo stesso valore per la stessa persona. Tuttavia, se i valori di Emp_ID e Cust_ID fossero stati scelti prima che la relazione PERSON venisse creata, i valori di Emp_ID e Cust_ID probabilmente non coinciderebbero. Inoltre, se si modificano i valori di Emp_ID e Cust_ID per farli coincidere con il nuovo Person_ID, come si può essere certi che tutti gli Emp_ID e Cust_ID siano univoci nel caso in cui un altro impiegato o cliente abbia già il valore

associato di `Person_ID`? Ancora peggio, se ci sono altre tabelle che fanno riferimento, per esempio, a `EMPLOYEE`, allora le chiavi esterne in queste tabelle devono cambiare, creando un effetto a onda di modifiche delle chiavi esterne. L'unico modo per garantire che ogni chiave primaria di una relazione sia univoca nella base di dati è quello di creare una chiave aziendale fin dall'inizio, così che le chiavi primarie non debbano mai cambiare.

Nel nostro esempio, la base di dati originale (senza `PERSON`) con una chiave aziendale è mostrata nella Figura 7.27a (le relazioni) e nella Figura 7.27b (i dati di esempio). In questa figura, `Emp_ID` e `Cust_ID` sono ora chiavi aziendali e `OBJECT` è il supertipo di tutte le altre relazioni. `OBJECT` può avere attributi come il nome del tipo di oggetto (incluso nell'esempio come attributo `Object_Type`), la data di creazione, la data dell'ultima modifica o qualsiasi altro attributo di sistema interno per un'istanza di oggetto. Quando occorre `PERSON`, quindi, la base di dati si evolve nella struttura mostrata nella Figura 7.27c (le relazioni) e nella Figura 7.27d (i dati di esempio). L'evoluzione della base di dati con `PERSON` richiede ancora alcune modifiche alle tabelle esistenti, ma non ai valori di chiave primaria; l'attributo di nome è spostato su `PERSON`, poiché è comune a entrambi i sottotipi, e viene aggiunta una chiave esterna a `EMPLOYEE` e `CUSTOMER` per puntare all'istanza della persona in comune. Come si vedrà nel Capitolo 9, è facile aggiungere ed eliminare colonne non di chiave, anche esterne, dalle definizioni delle tabelle. Per contro, modificare una chiave primaria di una relazione non è permesso da gran parte dei sistemi di gestione delle basi di dati, a causa del costo estensivo dell'effetto a onda delle chiavi primarie.

Riepilogo

La progettazione logica di una base di dati è il processo di trasformazione del modello di dati concettuale in un modello di dati logico. In questo capitolo si è posto l'accento sul modello di dati relazionale, a causa della sua importanza nei sistemi di gestione di basi di dati più moderni. Il modello di dati relazionale rappresenta i dati sotto forma di tabelle chiamate relazioni. Una relazione è una tabella di dati che ha un nome e due dimensioni. Una proprietà fondamentale delle relazioni è che non possono contenere attributi multivalore.

In questo capitolo sono stati descritti i passaggi principali del processo di progettazione logica della base di dati. Questo processo si basa sulla trasformazione di diagrammi E-R in relazioni normalizzate. I tre passaggi di questo processo sono i seguenti: trasformare i diagrammi E-R ed EER in relazioni, normalizzare le relazioni e unire le relazioni. Il risultato del processo è un insieme di relazioni nella terza forma normale che può essere implementato utilizzando qualsiasi sistema di gestione di basi di dati moderno.

Ogni tipo di entità nel diagramma E-R viene trasformato in una relazione che ha la stessa chiave primaria del tipo di entità. Un'associazione uno-a-molti viene rappresentata aggiungendo una chiave esterna alla relazione che rappresenta l'entità sul lato molti dell'associazione (questa chiave primaria è la chiave primaria dell'entità sul lato uno dell'associazione). Un'associazione multi-a-molti viene rappresentata creando una relazione separata. La chiave primaria di questa relazione è una chiave composta, costituita dalla chiave primaria di ognuna delle entità che partecipa all'associazione.

Il modello relazionale non supporta direttamente le associazioni supertipo/sottotipo, ma è possibile modellare queste associazioni creando una tabella (o relazione) separata per il supertipo e per ogni sottotipo. La chiave primaria di ciascun sottotipo è la stessa per il supertipo (o appartiene almeno allo stesso dominio). Il supertipo deve avere un attributo chiamato discriminante di sottotipo, che indica a quali sottotipi appartiene ciascuna istanza del supertipo. Lo scopo della normalizzazione è quello di ottenere delle relazioni ben strutturate prive di quelle anomalie (incoerenze o errori) che possono sorgere

Figura 7.27

Chiave aziendale.
(a) Relazioni con chiave aziendale.

OBJECT (OID, Object_Type)
EMPLOYEE (OID, Emp_ID, Emp_Name, Dept_Name, Salary)
CUSTOMER (OID, Cust_ID, Cust_Name, Address)

(b) Dati di esempio con chiave aziendale.

OBJECT

<u>OID</u>	Object_Type
1	EMPLOYEE
2	CUSTOMER
3	CUSTOMER
4	EMPLOYEE
5	EMPLOYEE
6	CUSTOMER
7	CUSTOMER

EMPLOYEE

<u>OID</u>	Emp_ID	Emp_Name	Dept_Name	Salary
1	100	Jennings, Fred	Marketing	50000
4	101	Hopkins, Dan	Purchasing	45000
5	102	Huber, Ike	Accounting	45000

CUSTOMER

<u>OID</u>	Cust_ID	Cust_Name	Address
2	100	Fred's Warehouse	Greensboro, NC
3	101	Bargain Bonanza	Moscow, ID
6	102	Jasper's	Tallahassee, FL
7	103	Desks 'R Us	Kettering, OH

(c) Relazioni dopo l'aggiunta della relazione PERSON.

OBJECT (OID, Object_Type)
EMPLOYEE (OID, Emp_ID, Dept_Name, Salary, Person_ID)
CUSTOMER (OID, Cust_ID, Address, Person_ID)
PERSON (OID, Name)

(d) Dati di esempio dopo l'aggiunta della relazione PERSON.

OBJECT		PERSON	
<u>OID</u>	Object_Type	<u>OID</u>	Name
1	EMPLOYEE	8	Jennings, Fred
2	CUSTOMER	9	Fred's Warehouse
3	CUSTOMER	10	Bargain Bonanza
4	EMPLOYEE	11	Hopkins, Dan
5	EMPLOYEE	12	Huber, Ike
6	CUSTOMER	13	Jasper's
7	CUSTOMER	14	Desks 'R Us
8	PERSON		
9	PERSON		
10	PERSON		
11	PERSON		
12	PERSON		
13	PERSON		
14	PERSON		

EMPLOYEE				
<u>OID</u>	Emp_ID	Dept_Name	Salary	<u>Person_ID</u>
1	100	Marketing	50000	8
4	101	Purchasing	45000	11
5	102	Accounting	45000	12

CUSTOMER			
<u>OID</u>	Cust_ID	Address	<u>Person_ID</u>
2	100	Greensboro, NC	9
3	101	Moscow, ID	10
6	102	Tallahassee, FL	13
7	103	Kettering, OH	14

quando le relazioni vengono aggiornate o modificate. La normalizzazione si basa sull'analisi delle dipendenze funzionali, che sono vincoli tra due attributi (o due insiemi di attributi) e può essere realizzata in vari momenti. Le relazioni nella prima forma normale (1NF) non contengono attributi multivalore o gruppi ripetenti. Le relazioni in 2NF non contengono dipendenze parziali, mentre le relazioni 3NF non contengono dipendenze transitive. È possibile utilizzare dei diagrammi per rappresentare le dipendenze funzionali in una relazione per aiutare a scomporre quella relazione, se necessario, e ottenere così relazioni nella terza forma normale. Sono state definite anche forme normali di livello più alto (oltre la 3NF), che verranno trattate nell'Appendice B.

Quando si combinano le relazioni, occorre fare attenzione, poiché ci si può trovare ad affrontare problemi come i sinonimi, gli omonimi, le dipendenze transitive e le associazioni supertipo/sottotipo. Inoltre, prima che le relazioni vengano definite nel sistema di gestione della base di dati, tutte le chiavi primarie devono essere descritte come chiavi non intelligenti di attributo singolo e, preferibilmente, come chiavi aziendali.

Ripasso del capitolo

Parole chiave

Alias	Chiave esterna ricorsiva	Forma normale	Relazione
Anomalia	Chiave primaria	Normalizzazione	Relazione ben strutturata
Chiave aziendale	Determinante	Nulla	Seconda forma normale
Chiave candidata	Dipendenza funzionale	Omonimo	Sinonimo
Chiave composta	Dipendenza funzionale parziale	Prima forma normale	Terza forma normale
Chiave esterna	Dipendenza transitiva	Regola di integrità dell'entità	Vincolo di integrità referenziale

Domande di ripasso

- Definire ciascuno dei seguenti termini chiave:
 - determinante
 - dipendenza funzionale
 - dipendenza transitiva
 - chiave esterna ricorsiva
 - normalizzazione
 - chiave composta
 - relazione
 - forma normale
 - dipendenza funzionale parziale
 - chiave aziendale
- Unire i seguenti termini alle rispettive definizioni:
 - ___ relazione ben strutturata; ___ anomalia;
 - ___ dipendenza funzionale; ___ determinante;
 - ___ chiave composta; ___ 1NF; ___ 2NF; ___ 3NF;
 - ___ chiave esterna ricorsiva; ___ relazione;
 - ___ dipendenza transitiva.
 - vincolo tra due attributi
 - dipendenza funzionale tra attributi non di chiave
 - si riferisce alla chiave primaria nella stessa relazione
 - attributi multivalore eliminati
 - incoerenza o errore
 - contiene una lieve ridondanza
 - contiene due o più attributi
 - non contiene dipendenze funzionali parziali
 - può essere scomposto in parti componenti
 - dipende dall'esistenza di un altro tipo di entità
 - associazione di grado tre
- Confrontare i seguenti termini:
 - forma normale; normalizzazione
 - chiave candidata; chiave primaria
 - dipendenza funzionale; dipendenza transitiva
 - chiave composta; chiave esterna ricorsiva
 - determinante; chiave candidata
 - chiave esterna; chiave primaria
- Citare sei proprietà importanti delle relazioni.
- Descrivere due proprietà che devono essere soddisfatte dalle chiavi candidate.
- Descrivere tre tipi di anomalie che possono sorgere in una tabella.
- Riempire gli spazi in ciascuna delle seguenti frasi.
 - Una relazione che non ha dipendenze funzionali parziali è nella ___ forma normale.
 - Una relazione che non ha attributi multivalore è nella ___ forma normale.
 - Una relazione che non ha dipendenze transitive è nella ___ forma normale.
- Cos'è una relazione ben strutturata? Perché le relazioni ben strutturate sono importanti nella progettazione logica di una base di dati?
- Descrivere come i seguenti componenti di un diagramma E-R vengono trasformati in relazioni:
 - tipo di entità regolare
 - associazione (1:M)
 - associazione (M:N)
 - associazione (supertipo/sottotipo)
 - attributo multivalore
 - entità debole
 - attributo composto

10. Descrivere brevemente quattro problemi tipici che sorgono di frequente quando si uniscono le relazioni, descrivere le tecniche più diffuse per risolverli.
11. Elencare tre condizioni che possono essere applicate per determinare se una relazione che è nella prima forma normale è anche nella seconda forma normale.
12. Spiegare come ciascuno dei tipi seguenti di vincoli di integrità sono applicati nei comandi CREATE TABLE SQL:
 - a. integrità dell'entità
 - b. integrità referenziale
13. Come vengono rappresentate le associazioni tra le entità nel modello di dati relazionale?
14. Come si rappresenta un'associazione unaria 1:M in un modello di dati relazionale?
15. Come si rappresenta un'associazione ternaria M:N in un modello di dati relazionale?
16. Qual è il rapporto tra la chiave primaria di una relazione e le dipendenze funzionali tra tutti gli attributi in una relazione?
17. In quali condizioni una chiave esterna non deve essere nulla?
18. Spiegare come intervenire sulle chiavi primarie per eliminare gli effetti a onda delle chiavi man mano che la base di dati si evolve.
 - a. Figura 6.6b
 - b. Figura 6.7a
 - c. Figura 6.9
 - d. Figura 6.10
 - e. Figura 6.18
3. Per ciascuna di queste relazioni, indicare la relativa forma normale. Se la relazione non è nella terza forma normale, scomporla in relazioni 3NF. Le dipendenze funzionali (diverse da quelle implicate dalla chiave primaria) vengono mostrate dove opportuno.
 - a. CLASS(Course_No,Section_No)
 - b. CLASS(Course_No,Section_No,Room)
 - c. CLASS(Course_No,Section_No,Room,Capacity)Room → Capacity
 - d. CLASS(Course_No,Section_No,Course_Name,Room,Capacity)
Course_No → Course_Name Room → Capacity
4. La Figura 7.28 mostra il registro di una classe del Millenium College. Convertire questa vista utente in un insieme di relazioni 3NF utilizzando una chiave aziendale. Si presuma quanto segue.
 - Un insegnante ha un'unica aula.
 - Uno studente ha un'unica specializzazione.
 - Una corso ha un unico titolo.

Problemi ed esercizi

1. Per ciascuno dei seguenti diagrammi E-R del Capitolo 5, fare quanto segue.
 - I. Trasformare il diagramma in uno schema relazionale che mostri i vincoli di integrità referenziale (si veda la Figura 7.5 per un esempio di questo schema).
 - II. Per ogni relazione, disegnare un diagramma delle dipendenze funzionali (si veda la Figura 7.23 per un esempio).
 - III. Se qualcuna delle seguenti relazioni non è in NF3, trasformarla in NF3.
 - a. Figura 5.8
 - b. Figura 7.8b
 - c. Figura 5.11a
 - d. Figura 5.11b
 - e. Figura 5.15a (versione delle relazioni)
 - f. Figura 5.16
 - g. Figura 5.19
2. Per ciascuno dei seguenti diagrammi E-R del Capitolo 6, fare quanto segue.
 - I. Trasformare il diagramma in uno schema relazionale che mostri i vincoli di integrità referenziale (si veda la Figura 7.5 per un esempio di questo schema).
 - II. Per ogni relazione, disegnare un diagramma delle dipendenze funzionali (si veda la Figura 7.23 per un esempio).
 - III. Se qualcuna delle seguenti relazioni non è in NF3, trasformarla in NF3.
 - a. Convertire questa tabella in una relazione (chiamata PART SUPPLIER) nella prima forma normale. Illustrare la relazione con i dati di esempio nella tabella.
 - b. Elencare le dipendenze funzionali in PART SUPPLIER e identificare una chiave candidata.
 - c. Per la relazione PART SUPPLIER, identificare quanto segue: un'anomalia di inserimento, un'anomalia di eliminazione e un'anomalia di modifica.
 - d. Disegnare uno schema relazionale per PART SUPPLIER (analogamente alla Figura 7.23) e mostrare le dipendenze funzionali.
5. La Figura 7.29 mostra un diagramma E-R per un ambiente semplificato di carte di credito. Ci sono due tipi di conti per le carte: a debito e a credito. I conti a credito accumulano gli importi spesi presso i commercianti. Ogni importo è identificato dalla data e dall'ammontare della spesa.
 - a. Sviluppare uno schema relazionale (analogamente alla Figura 7.5).
 - b. Mostrare le dipendenze funzionali (analogamente alla Figura 7.23).
 - c. Sviluppare un insieme di dipendenze funzionali (analogamente alla Figura 7.23).
6. La Tabella 7.2 contiene alcuni dati di esempio di parti e produttori che forniscono queste parti. Discutendo questi dati con gli utenti, si scopre che i numeri delle parti (ma non le descrizioni) identificano in modo univoco le parti e che i nomi dei produttori identificano in modo univoco i produttori.
 - a. Convertire questa tabella in una relazione (chiamata PART SUPPLIER) nella prima forma normale. Illustrare la relazione con i dati di esempio nella tabella.
 - b. Elencare le dipendenze funzionali in PART SUPPLIER e identificare una chiave candidata.
 - c. Per la relazione PART SUPPLIER, identificare quanto segue: un'anomalia di inserimento, un'anomalia di eliminazione e un'anomalia di modifica.
 - d. Disegnare uno schema relazionale per PART SUPPLIER (analogamente alla Figura 7.23) e mostrare le dipendenze funzionali.

Figura 7.28
Registro di classe
(Millenium College).

MILLENNIUM COLLEGE CLASS LIST FALL SEMESTER 200X			
COURSE NO.: IS 460			
COURSE TITLE: DATABASE			
INSTRUCTOR NAME: NORMA L. FORM			
INSTRUCTOR LOCATION: B 104			
STUDENT NO.	STUDENT NAME	MAJOR	GRADE
38214	Bright	IS	A
40875	Cortez	CS	B
51893	Edwards	IS	A

Figura 7.29
Diagramma E-R
per carte di credito.

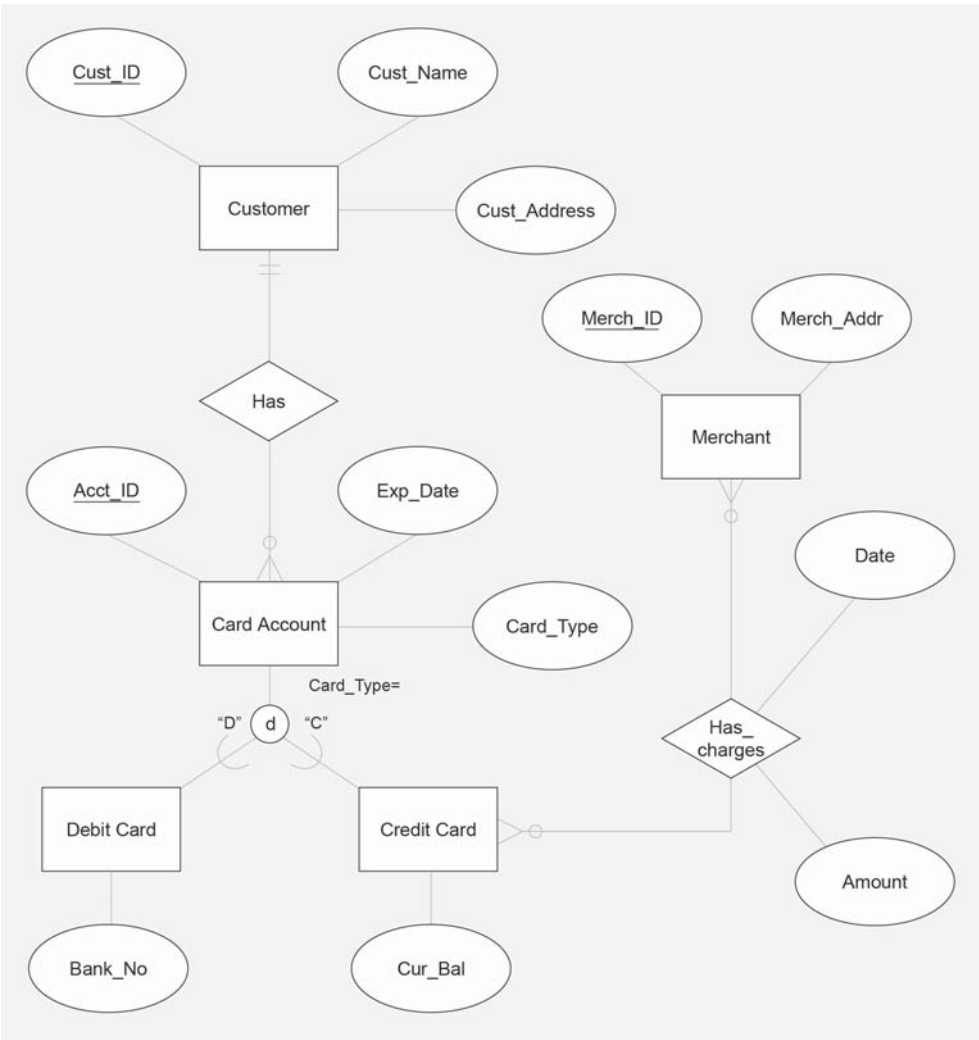


Tabella 7.2 Dati di esempio per parti e produttori.

Part_No.	Description	Vendor_Name	Address	Unit_Cost
1234	Logic chip	Fast Chips	Cupertino	10.00
		Smart Chips	Phoenix	8.00
5678	Memory chip	Fast Chips	Cupertino	3.00
		Quality Chips	Austin	2.00
		Smart Chips	Phoenix	5.00

- e. In quale forma normale è la relazione?
- f. Sviluppare un insieme di relazioni 3NF da PART SUPPLIER.
7. La Tabella 7.3 mostra una relazione chiamata GRADE REPORT per un'università. Quella che segue è una descrizione delle dipendenze funzionali in GRADE REPORT.
- Student_ID → Student_Name, Campus, Address, Major
 - Course_ID → Course_Title, Instructor_Name, Instructor_Location
 - Student_ID, Course_ID → Grade
 - Instructor_Name → Instructor_Location
- Il compito è il seguente.
- a. Disegnare uno schema relazionale (analogamente alla Figura 7.23) e disegnare un diagramma delle dipendenze funzionali nella relazione.
- b. In quale forma normale è la relazione?
- c. Scomporre GRADE REPORT in un insieme di relazioni 3NF.
- d. Disegnare uno schema relazionale per le proprie relazioni 3NF (analogamente alla Figura 7.5) e mostrare in vincoli di integrità referenziale.
8. Trasformare la versione degli attributi della Figura 5.15b in relazioni 3NF. Trasformare la versione delle relazioni della Figura 5.15b in relazioni 3NF. Confrontare i due insiemi di relazioni 3NF con quelli nella Figura 7.10. Quali osservazioni e conclusioni si possono trarre confrontando questi diversi insiemi di relazioni 3NF?

Esercitazioni

1. Intervistare i progettisti di sistema e di basi di dati di diverse aziende. Chiedere loro di descrivere il processo che utilizzano per la progettazione logica. Come trasformano i loro modelli di dati concettuali (per esempio i diagrammi E-R) in schemi relazionali? Qual è il ruolo degli strumenti CASE nel processo? Utilizzano la normalizzazione? Se sì, a quale livello?
2. Procurarsi un documento comune, come uno scontrino, una ricevuta di un meccanico, una ricevuta di una carta di credito e così via. Procedere quindi come segue.
 - a. Elencare i nomi degli attributi.
 - b. Elencare le dipendenze funzionali tra gli attributi (considerare i presupposti che si ritengono necessari).
 - c. Disegnare uno schema per la relazione (analogamente alla Figura 7.23) e disegnare un diagramma delle dipendenze funzionali.
 - d. Scomporre la relazione in un insieme di relazioni 3NF e disegnare lo schema relazionale (analogamente alla Figura 7.5).
3. Procurarsi la documentazione per tre diffusi strumenti CASE basati su PC e confrontare la notazione che utilizzano per rappresentare gli schemi relazionali. Si possono utilizzare i seguenti indirizzi Internet per ottenere informazioni on-line per i tre strumenti seguenti.
 - a. Microsoft Access: www.microsoft.com/Visual Tools/
 - b. Erwin: www.logicworks.com
 - c. EasyER: www.estl.com

Tabella 7.3 Relazione GRADE REPORT.

GRADE REPORT								
Student_ID	Student_Name	Campus_Address	Major	Course_ID	Course_Title	Instructor_Name	Instructor_Location	Grade
168300458	Williams	208 Brooks	IS	IS 350	Database Mgt	Codd	B 104	A
268300458	Williams	208 Brooks	IS	IS 465	Systems Analysis	Parsons	B 317	B
543291073	Baker	104 Phillips	Acctg	IS 650	Database Mgt	Codd	B 104	C
543291073	Baker	104 Phillips	Acctg	Acct 201	Fund Acctg	Miller	H 310	B
543291073	Baker	104 Phillips	Acctg	Mktg 300	Intro Mktg	Bennett	B 212	A

Bibliografia

- Chouinard, P., 1989, “Supertypes, Subtype, and DB2”, *Database Programming & Design* 2 (ottobre), pp. 50–57.
- Codd, E. F., 1970, “A Relational Model of Data for Large Relational Databases”, *Communications of the ACM* 13 (giugno), pp. 77–87.
- Codd, E. F., 1990, *The Relational Model for Database Management Version 2*, Reading, MA, Addison-Wesley.
- Date, C. J., 1995, *An Introduction to Database Systems*, sesta edizione, Reading, MA, Addison-Wesley.
- Dutka, A. F. e Hanson, H. H., 1989, *Fundamentals of Data Normalization*, Reading, MA, Addison-Wesley.
- Fleming, C. C. e Von Halle, B., 1989, *Handbook of Relational Database Design*, Reading, MA, Addison-Wesley.

Johnston, T., 2000, “Primary Key Reengineering Projects: The Problem” e “Primary Key Reengineering Projects: The Solution”, disponibili sul sito www.dmreview.com.

Navathe, S., Elmasri, R. e Larson, J., 1986, “Integrating User Views in Database Design”, *Computer* (gennaio), pp. 50–62.

Per approfondire

Elmasri, R. e Navathe, S., 1994, *Fundamentals of Database Systems*, seconda edizione, Menlo Park, CA, Benjamin Cummings.

Hoffer, J. A., George, J. F. e Valacich, J. S., 1999, *Modern Systems Analysis and Design*, seconda edizione, Reading, MA, Addison-Wesley.

Storey, V. C., “Relational Database Design Based on the Entity-Relationship Model”, *Data and Knowledge Engineering* 7, pp. 47–83.

Project Case



MOUNTAIN VIEW COMMUNITY HOSPITAL

Descrizione del progetto

Nei capitoli precedenti del libro è stato introdotto il project case di Mountain View Community Hospital. In questo capitolo si prosegue con il lavoro sul progetto ponendo l'accento sulla progettazione logica per il modello di dati relazionale. Sebbene l'ospedale prenda in considerazione le tecnologie più nuove orientate agli oggetti e relazionali-a oggetti, è probabile che nei prossimi anni continuerà a utilizzare prevalentemente la tecnologia relazionale dello sviluppo dei sistemi.

Domande sul progetto

1. Perché Mountain View Community Hospital continua a utilizzare la tecnologia relazionale per lo sviluppo dei sistemi, nonostante il sorgere continuo di tecnologie più nuove?
2. Mountain View Community Hospital dovrebbe utilizzare la normalizzazione nella progettazione delle sue basi di dati relazionali? Perché sì o no?
3. Perché l'integrità dell'entità e i vincoli di integrità referenziale sono importanti per l'ospedale?
4. Quali persone nell'ospedale dovrebbero essere coinvolte nella normalizzazione dei dati?

Esercizi sul progetto

Il compito da svolgere consiste nell'eseguire una progettazione logica per le basi di dati di Mountain View Community Hospital. Nei Capitoli 5 e 6 sono stati sviluppati modelli di dati concettuali per l'ospedale, e in particolare si è preparato quanto segue.

■ Un diagramma E-R per le entità e le associazioni principali dell'ospedale (esercizio sul progetto 2 nel Capitolo 5).

■ Un diagramma EER focalizzato sulle risorse umane dell'ospedale (esercizio sul progetto 1 nel Capitolo 6).

Il compito da svolgere in questo capitolo è quello di sviluppare una progettazione logica per uno o entrambi questi progetti logici. Per ogni progetto logico, si esegua quanto segue.

1. Mappare il diagramma E-R (e/o EER) in uno schema relazionale utilizzando le tecniche descritte in questo capitolo. Accertarsi di sottolineare tutte le chiavi primarie, di includere tutte le chiavi esterne necessarie e di indicare i vincoli di integrità referenziale.
2. Disegnare le dipendenze funzionali in ogni relazione.
3. Se qualche relazione non è in 3NF, scomporla in relazioni 3NF e rivedere lo schema relazionale.
4. Creare chiavi aziendali per tutte le relazioni e ridefinire tutte le relazioni.
5. Scrivere comandi CREATE TABLE per ogni relazione di risposta all'esercizio sul progetto 4. Considerare qualche ragionevole presupposto riguardante il tipo di dati per ciascun attributo in ognuna delle relazioni.
6. Se si sviluppa un progetto logico per il diagramma E-R di entrambi i capitoli, unire le relazioni in un unico insieme di relazioni 3NF.

CAPITOLO

8

Progettazione fisica delle basi di dati e prestazioni

Al termine di questo capitolo sarete in grado di fare quanto segue.

1. Definire i seguenti termini chiave: campo, tipo di dati, record fisico, pagina, fattore di blocco, denormalizzazione, partizionamento orizzontale, partizionamento verticale, file fisico, tablespace, estensione, puntatore, organizzazione dei file, organizzazione dei file sequenziale, organizzazione dei file indicizzata, chiave secondaria, indice bitmap, indice di join, organizzazione dei file hash, algoritmo di hashing, tabella di indice hash, *Redundant Array of Inexpensive Disks* (RAID), stripe.
2. Descrivere il processo di progettazione fisica della base di dati, i suoi obiettivi e i risultati.
3. Scegliere i formati di memorizzazione degli attributi di un modello di dati logico.
4. Scegliere un'organizzazione di file opportuna bilanciando vari fattori di progettazione.
5. Descrivere tre tipi importanti di organizzazione dei file.
6. Descrivere lo scopo degli indici e le considerazioni principali nella scelta degli attributi da indicizzare.
7. Tradurre un modello di dati relazionale in strutture di base di dati efficienti, sapendo quando e come denormalizzare il modello di dati logico.

Introduzione

Nei Capitoli dal 5 al 7 si è visto come descrivere e modellare i dati dell'azienda durante le fasi di modellazione concettuale dei dati e di progettazione logica del processo di sviluppo della base di dati. Si è spiegato come utilizzare la notazione EER e sono stati introdotti il modello di dati relazionale e la normalizzazione per sviluppare astrazioni dei dati dell'azienda che rappresentino il significato dei dati; queste notazioni non spiegano tuttavia *come* i dati verranno elaborati o memorizzati. Lo scopo della progettazione fisica della base di dati è quello di tradurre la descrizione logica dei dati in tecniche specifiche per la memorizzazione e il recupero dei dati. Lo scopo è quello di creare un progetto per la memorizzazione dei dati che fornisca prestazioni adeguate e che assicuri l'integrità, la protezione e l'eventuale ripristino della base di dati.

La progettazione fisica della base di dati non include l'implementazione dei file e della base di dati (per esempio, creare i file e caricarli nella base di dati), ma produce le specifiche tecniche che verranno

utilizzate dai programmatori e da altre persone dei sistemi informativi durante la fase di implementazione, che tratteremo nei Capitoli dal 9 al 12.

In questo capitolo si studieranno i passaggi di base necessari per sviluppare una progettazione fisica della base di dati efficiente. In particolare, ci si concentrerà sulla progettazione di un'unica base di dati centralizzata. Si vedrà come valutare l'entità dei dati richiesti dagli utenti in una base di dati e come è probabile che i dati vengano utilizzati. Si vedranno le scelte da compiere nella memorizzazione dei dati e come compierle. Si spiegherà perché le tabelle normalizzate non costituiscono sempre i file di dati fisici ideali e come si possono denormalizzare i dati per migliorare il recupero dei dati. Si esamineranno le diverse organizzazioni dei file e l'uso degli indici, che sono importanti per velocizzare il recupero dei dati. Si vedranno inoltre le differenze principali tra architetture diverse delle basi di dati.

La progettazione fisica della base di dati deve essere effettuata con cura, poiché le decisioni prese in questa fase hanno un impatto fondamentale sull'accessibilità dei dati, sui tempi di risposta, sulla sicurezza, sul rapporto con gli utenti e su altri fattori di progettazione analoghi dei sistemi informativi. Anche l'amministrazione della base di dati gioca un ruolo fondamentale in questo tipo di progettazione.

8.1 Processo di progettazione fisica di una base di dati

Nella maggior parte dei casi, molte decisioni sulla progettazione fisica sono implicite o vengono scremate nel momento in cui si scelgono le tecnologie di gestione della base di dati da utilizzare con i sistemi informativi che si stanno progettando. Poiché molte aziende hanno standard per i sistemi operativi, i sistemi di gestione delle basi di dati e i linguaggi di accesso ai dati, si dovranno affrontare solo quelle scelte non implicite nelle tecnologie date. Tratteremo quindi solo le decisioni che si devono prendere più di frequente, così come le altre decisioni che possono essere fondamentali per alcuni tipi di applicazioni, come la rappresentazione dei dati o il loro recupero on-line.

Lo scopo principale della progettazione fisica di una base di dati è l'efficienza dell'elaborazione dei dati. Oggigiorno, con il ribasso dei prezzi della tecnologia informatica per unità di misura (in termini di velocità e spazio), è molto importante progettare la base di dati fisica per ridurre il tempo necessario agli utenti per interagire con il sistema informativo. Ci concentreremo quindi su come rendere l'elaborazione dei file e delle basi di dati fisici efficiente, ponendo meno attenzione all'uso efficiente dello spazio.

La progettazione di file e basi di dati fisici richiede alcune informazioni che dovrebbero essere state raccolte e prodotte durante le fasi di sviluppo precedenti; tra di esse vi sono le seguenti.

- Relazioni normalizzate, comprese le stime dei volumi.
- Definizione di ciascun attributo.
- Descrizione di dove e quando i dati vengono utilizzati, e cioè immessi, recuperati, eliminati e aggiornati (compresa la frequenza delle operazioni).
- Aspettative o esigenze sui tempi di risposta e la sicurezza, il backup, il recupero, la conservazione e l'integrità dei dati.
- Descrizioni delle tecnologie (sistemi di gestione delle basi di dati) utilizzate per implementare la base di dati.

La progettazione fisica della base di dati richiede numerose decisioni importanti che influenzano l'integrità e le prestazioni del sistema applicativo. Tra di esse vi sono le seguenti.

1. Scegliere il formato di memorizzazione (chiamato *tipo di dati*) per ogni attributo del modello di dati logico. Il formato viene scelto per ridurre al minimo lo spazio di memoria e per elevare al massimo l'integrità dei dati.
2. Raggruppare gli attributi del modello di dati logico in *record fisici*. Si vedrà che, sebbene le colonne di una tabella relazionale siano una definizione naturale per i contenuti di un record fisico, questo non è sempre il raggruppamento di attributi ideale.
3. Disporre record strutturati in modo simile nella memoria secondaria (soprattutto dischi rigidi), così che record singoli o gruppi di record (chiamati *organizzazioni dei file*) possano essere memorizzati, recuperati e aggiornati rapidamente. È necessario considerare anche la protezione e il ripristino dei dati dopo che sono stati individuati degli errori.
4. Scegliere le strutture (chiamate *indici* e *architetture di basi di dati*) per ordinare e connettere i file per rendere più efficiente il recupero dei dati correlati.
5. Preparare strategie per la gestione delle interrogazioni che ottimizzino le prestazioni e traggano vantaggio delle organizzazioni dei file e dagli indici che sono stati specificati. Strutture di basi di dati efficienti saranno vantaggiose solo se le interrogazioni e i sistemi di gestione delle basi di dati che gestiscono queste interrogazioni verranno modificate per un loro uso intelligente.

8.1.1 Volume dei dati e analisi dei pattern d'uso

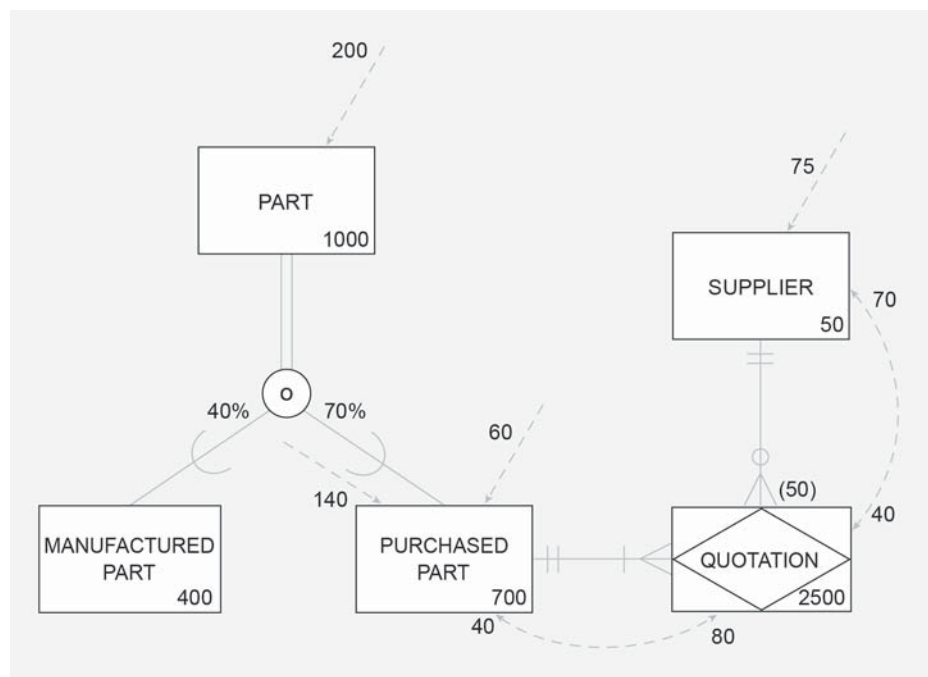
Come detto in precedenza, le statistiche sul volume dei dati e sulla frequenza d'uso sono input importanti nel processo di progettazione fisica di una base di dati. Di conseguenza, il passaggio finale da intraprendere nella progettazione logica o il passaggio iniziale nella progettazione fisica consistono nel valutare le dimensioni e i pattern d'uso della base di dati.

Un modo semplice per rappresentare le statistiche sui volumi dei dati e l'uso è quello di aggiungere la notazione ai diagrammi EER che rappresentano l'insieme finale delle relazioni normalizzate della progettazione logica. La Figura 8.1 mostra il diagramma EER (senza attributi) per una semplice base di dati delle scorte di Pine Valley Furniture Company. Questo diagramma EER rappresenta le relazioni normalizzate costruite durante la progettazione logica dal modello di dati concettuale di questa situazione, illustrato nella Figura 6.7b. L'associazione multi-a-molti Supplies della Figura 6.7b è mostrata nella Figura 8.1 come entità associativa (QUOTATION) corrispondente alla relazione creata da questa associazione.

Il volume dei dati e la frequenza degli accessi sono entrambi mostrati nella Figura 8.1. Per esempio, in questa base di dati ci sono 1000 PART. Il supertipo PART ha due sottotipi, MANUFACTURED (il 40 per cento di tutte le PART viene prodotto) e PURCHASED (il 70 per cento viene acquistato; poiché alcune PART sono di entrambi i sottotipi, le percentuali assommano a oltre il 100 per cento). Gli analisti di Pine Valley stimano che ci siano solitamente 50 SUPPLIER, e Pine Valley riceve in media 50 QUOTATION da ogni SUPPLIER, per un totale di 2500 QUOTATION. Le frecce tratteggiate rappresentano la frequenza degli accessi. Così, per esempio, in tutte le applicazioni che utilizzano questa base di dati, ci sono in media 200 accessi all'ora ai dati di PART e questo può portare, in base alle percentuali del sottotipo, a 140 accessi all'ora ai dati di PURCHASED PART. Del totale dei 200 accessi a PURCHASED PART, 80 richiedono anche i dati di QUOTATION e di questi 80 accessi a QUOTATION, ci sono 70 accessi successivi ai dati di SUPPLIER. Nel caso delle applicazioni on-line e basate sul Web, le mappe d'uso dovrebbero mostrare gli accessi al secondo. Possono essere necessarie numerose mappe d'uso per illustrare pattern d'uso diversi in momenti diversi del giorno. Le prestazioni vengono influenzate anche dalle specifiche di rete.

Figura 8.1

Mapa d'uso composta (Pine Valley Furniture Company).



Le statistiche sul volume e la frequenza vengono generate durante la fase di analisi dei sistemi del processo di sviluppo dei sistemi, quando gli analisti di sistema studiano le elaborazioni di dati e le attività aziendali presenti e quelle proposte. Le statistiche sul volume dei dati rappresentano le dimensioni dell'attività aziendale e dovrebbero essere calcolate considerando la crescita degli affari in un periodo recente di qualche anno. Le frequenze di accesso vengono valutate in base ai tempi degli eventi, al volume delle transazioni, al numero degli utenti simultanei e alle attività di report e di interrogazione. Poiché molte basi di dati supportano accessi ad hoc, e questi accessi possono cambiare notevolmente nel tempo e avere picchi e cali in un giorno, una settimana o un mese, le frequenze di accesso tendono a essere meno prevedibili e regolari di quelle del volume dei dati. Fortunatamente i numeri precisi non sono necessari. Ciò che è cruciale è la dimensione relativa dei numeri, che suggerirà se debba essere prestata più cura per poter ottenere le prestazioni migliori possibili. Per esempio, nella Figura 8.1 si può notare che per ognuna delle 40 volte all'ora in cui si ha avuto accesso a QUOTATION, lo stesso è accaduto per PURCHASED PART. Il diagramma dovrebbe quindi suggerire un'eventuale combinazione di queste due entità in una tabella (o file) di base di dati. L'azione di combinazione delle tabelle normalizzate è un esempio di denormalizzazione, come si vedrà successivamente nel capitolo.

Per le prossime tappe della progettazione fisica può essere utile riuscire a spiegare anche la natura degli accessi nei percorsi d'accesso indicati dalle linee tratteggiate. Per esempio, può servire sapere che dei 200 accessi ai dati di PART, 150 chiedono una parte o un insieme di parti in base alla chiave primaria Part_No (per esempio, accesso a una determinata parte con un numero); gli altri 50 accessi qualificano i dati della parte per l'accesso in base al valore di Qty_on_Hand (queste specifiche non sono mostrate nella Figura 8.1). Questa descrizione più precisa può aiutare nella selezione degli indici, uno degli argomenti che tratteremo successivamente nel capitolo. Può essere utile sapere anche se un accesso comporta la creazione, il recupero, l'aggiornamento o l'eliminazione di dati. Una descrizione così raffinata delle frequenze di accesso può essere gestita da una notazione supplementare in diagramma come quello nella Figura 8.1 o da un testo e da tabelle presenti in un'altra documentazione.

8.2 Progettare i campi

Un **campo** è l'unità più piccola dei dati di un'applicazione riconosciuta da un software di sistema, come un linguaggio di programmazione o un sistema di gestione delle basi di dati. Un campo corrisponde a un attributo semplice del modello di dati logici e rappresenta quindi ogni componente di un attributo composto.

Le decisioni di base da prendere quando si deve specificare un campo riguardano il tipo di dati (o tipo di memorizzazione) utilizzato per rappresentare i valori di quel campo, i controlli di integrità dei dati presenti nella base di dati e il modo in cui il DBMS deve gestire i valori mancanti per quel campo. Ci sono poi altre specifiche di campo, come il formato di visualizzazione, che vanno decise come parte della specifica totale del sistema informativo, ma qui non ci occuperemo di quelle specifiche gestite solitamente dai programmi invece che dai DBMS.

8.2.1 Scegliere il tipo di dati

Un **tipo di dati** è uno schema di codifica dettagliato riconosciuto dal software di sistema, come un DBMS, per rappresentare i dati dell'azienda. Il pattern di bit dello schema di codifica appare solitamente come trasparente, ma lo spazio per memorizzare i dati e la velocità necessaria per accedere ai dati sono una conseguenza della progettazione fisica della base di dati. Il DBMS specifico che si utilizza detta le opzioni disponibili. Per esempio, la Tabella 8.1 elenca i tipi di dati disponibili nel DBMS Oracle 8i, un DBMS tipico che utilizza la definizione dei dati e il linguaggio di manipolazione SQL. In alcuni tipi di DBMS possono essere disponibili altri tipi di dati per la valuta, la voce e le immagini e i vari tipi possono essere definiti dall'utente.

La scelta di un tipo di dati implica quattro obiettivi che hanno importanza diversa a seconda delle applicazioni.

1. Ridurre al minimo lo spazio di memoria.
2. Rappresentare tutti i valori possibili.
3. Migliorare l'integrità dei dati.
4. Supportare tutte le manipolazioni dei dati.

Il tipo di dati corretto per un campo può, in uno spazio minimo, rappresentare ogni valore possibile (ma eliminare i valori non validi) per l'attributo associato e può supportare la manipolazione dei dati richiesta (per esempio, i tipi di dati numerici per l'aritmetica e quelli di carattere per la manipolazione delle stringhe). Qualsiasi vincolo di dominio di attributo del modello di dati concettuale può essere utile nel scegliere il tipo di dati ottimale per quell'attributo. Il raggiungimento di questi quattro obiettivi può essere insidioso. Per esempio, si consideri un DBMS per il quale un tipo di dati ha una grandezza massima di 2 byte. Si supponga che il tipo di dati sia sufficiente per rappresentare un campo per una determinata quantità venduta. Quando i campi delle quantità vendute vengono sommati, la somma può richiedere un numero più grande di 2 byte. Se il DBMS utilizza questo tipo di dati per i risultati delle operazioni su quel campo, la lunghezza di 2 byte non sarà adatta. Alcuni tipi di dati hanno funzionalità di manipolazioni speciali; per esempio, solo il tipo di dati DATE consente l'aritmetica dei dati reali.

Tecniche di codifica e di compressione

Alcuni attributi hanno un insieme di valori sparsi o sono talmente grandi da consumare un notevole spazio di memoria (la distribuzione di dati sparsi comporta un rallentamento nell'elaborazione dei dati).

Tabella 8.1 Tipi di dati in Oracle 8i.

Tipo di dati	Descrizione
VARCHAR2	Dati di carattere a lunghezza variabile con una lunghezza massima di 4000 caratteri; si deve immettere una lunghezza di campo massima (per esempio, VARCHAR2(30) per un campo con una lunghezza massima di 30 caratteri). Un valore inferiore a 30 caratteri utilizza solo lo spazio necessario.
CHAR	Dati di carattere a lunghezza fissa con una lunghezza massima di 2000 caratteri; la lunghezza predefinita è di 1 carattere (per esempio, CHAR(5) per un campo con una lunghezza fissa di 5 caratteri, in grado di contenere un valore lungo da 0 a 5 caratteri).
LONG	In grado di memorizzare fino a 4 gigabyte di un campo di carattere a lunghezza variabile (per esempio, per contenere una prescrizione medica o il commento di un cliente).
NUMBER	Numeri positivi e negativi nell'intervallo da 10^{-130} a 10^{126} , può specificare la precisione (il numero totale di cifre a sinistra e a destra della virgola decimale) e la scala (il numero di cifre a destra della virgola decimale). Per esempio, NUMBER(5) specifica un campo di interi con un massimo di 5 cifre, mentre NUMBER(5,2) specifica un campo con non più di 5 cifre ed esattamente 2 cifre a destra della virgola decimale.
DATE	Qualsiasi data dal 1 gennaio 4712 a.C. al 31 dicembre 4712 d.C.; la data memorizza il secolo, l'anno, il mese, il giorno, l'ora, il minuto e il secondo.
BLOB	Grande oggetto binario (<i>Binary Large Object</i>), in grado di memorizzare fino a 4 gigabyte di dati binari (per esempio, una fotografia o un clip audio).

Un campo con un numero limitato di valori possibili può essere convertito in un codice che richiede meno spazio. Si consideri l'esempio del campo di prodotto Finish illustrato nella Figura 8.2. I prodotti di Pine Valley Furniture sono forniti solo in un numero limitato di tipi di legname: betulla, acero e quercia. Creando un codice, o tabella di conversione, ogni valore del campo Finish può essere sostituito da un codice, un riferimento incrociato alla tabella di ricerca, simile a una chiave esterna. Questo riduce l'entità di spazio per il campo Finish e quindi per il file PRODUCT. Ci sarà uno spazio supplementare per la tabella di ricerca FINISH, e quando servirà il campo Finish, verrà richiesto un accesso supplementare a questa tabella di ricerca. Se il campo Finish non viene utilizzato molto spesso, i vantaggi della codifica possono superare i costi. Si noti che la tabella di codice non appare nel modello concettuale o logico. La tabella di codice è un costrutto fisico per ottenere miglioramenti nelle prestazioni dell'elaborazione dei dati e non un insieme di dati con un valore aziendale.

Un forma di tabella di codice viene utilizzata dalle tecniche di compressione dei dati, come le routine dei file zip. Una tecnica di compressione dei dati cerca i pattern nei dati e quindi codifica i pattern che appaiono più spesso con un numero minore di bit. Sebbene tali routine siano solitamente utilizzate per comprimere un intero file, possono anche essere utilizzate con alcuni DBMS per comprimere file specifici. Tecniche di compressione dei dati correlate sono le tecniche di codifica, che convertono un campo in un formato protetto. In entrambi i casi, perché un utente possa vedere il valore effettivo del file, il software deve conoscere la procedura di conversione inversa.

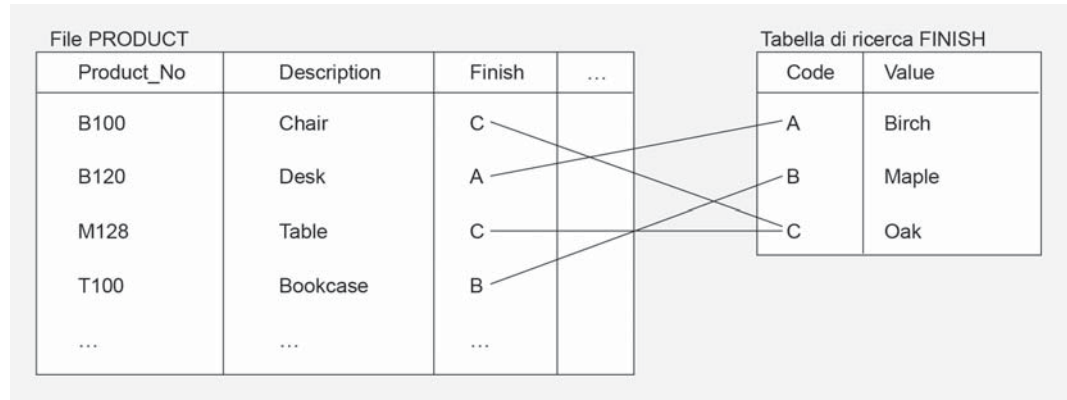
8.2.2 Controllare l'integrità dei dati

In molti DBMS, i controlli dell'integrità dei dati (per esempio, i controlli sul valore possibile che un campo può assumere) possono essere impostati nella struttura fisica dei campi e nei controlli applicati dal DBMS su quei campi. Il tipo di dati applica una forma di controllo dell'integrità dei dati, poiché può limitare il tipo di dati (numerici o di carattere) e la lunghezza di un valore di campo. Alcuni controlli tipici che un DBMS può supportare sono i seguenti.

- *Valore predefinito (default).* Un valore predefinito è un valore che un campo può assumere a meno che un utente non immetta un valore esplicito per un'istanza di quel campo. L'assegnazione di un

Figura 8.2

Esempio di tabella di ricerca di un codice (Pine Valley Furniture).



valore predefinito a un campo può ridurre il tempo di immissione dei dati, poiché l'immissione di un valore può essere saltata e può anche aiutare a ridurre gli errori di immissione per il valore più comune.

- **Controllo di intervallo.** Un controllo di intervallo limita l'insieme dei valori possibili che un campo può assumere. L'intervallo può essere un limite numerico superiore o inferiore o un insieme di valori specifici. I controlli di intervallo devono essere utilizzati con attenzione, perché i limiti dell'intervallo possono cambiare nel tempo. Una combinazione di controlli di intervallo e di codifica è quella che ha portato al problema dell'anno 2000 per molte aziende, nelle quali il campo dell'anno era rappresentato solo da numeri da 00 a 99. È preferibile implementare tali controlli attraverso un DBMS, poiché i controlli di intervallo nei programmi possono essere applicati in modo uniforme, rendendo quindi più difficile indentificarli e modificarli.
- **Controllo dei valori nulli.** Un valore nullo definito nel Capitolo 7 come un valore vuoto. Ogni chiave primaria deve avere un controllo di integrità che impedisce la presenza di un valore nullo. Qualsiasi altro campo richiesto può avere anche un controllo dei valori nulli se questa è la politica dell'azienda. Per esempio, un'università può impedire l'aggiunta di un corso alla sua base di dati a meno che il corso non abbia un titolo e un valore di chiave primaria, *Course_ID*. Molti campi possono avere legittimamente un valore nullo, che dovrà però essere utilizzato solo quando effettivamente richiesto dalle regole aziendali.
- **Integrità referenziale.** Il termine "integrità referenziale" è stato definito nel Capitolo 7. L'integrità referenziale su un campo è una forma di controllo di intervallo nel quale il valore di quel campo deve esistere come valore in qualche campo in un'altra riga della stessa tabella o di un'altra tabella. L'intervallo dei valori validi deriva quindi dal contenuto di un campo in una tabella della base di dati, e non da un insieme di valori precedentemente specificati. Si noti che l'integrità referenziale garantisce che vengano utilizzati solo valori a riferimento incrociato, e non che questi siano quelli corretti.

Gestire i dati mancanti

Quando un campo può essere nullo, non immettere alcun valore non è sufficiente. Per esempio, si supponga che il campo del codice di avviamento postale di un cliente sia nullo e che un report riepiloghi le vendite totali in base al mese e al codice di avviamento postale. Come devono essere gestite le vendite ai clienti il cui codice di avviamento postale non è noto? Due opzioni per gestire o impedire la presenza di

dati mancanti sono già stati citati: l'uso di un valore predefinito e il divieto di immettere valori mancanti (nulli). I dati mancanti sono inevitabili. Secondo Babad e Hoffer (1984), altri metodi per la gestione dei dati mancanti sono i seguenti.

1. Sostituire una stima del valore mancante. Per esempio, per calcolare un valore mancante delle vendite quando si calcolano le vendite di prodotti mensili, si può utilizzare una formula che considera la media dei valori delle vendite mensili per quel prodotto indicizzate in base alle vendite totali per quel mese di tutti i prodotti. Tali stime devono essere contrassegnate, così che gli utenti sappiano che non si tratta dei valori reali.
2. Tenere una traccia dei valori mancanti, così che report speciali e altri elementi di sistema permettano agli utenti di definire velocemente i valori non noti. Per farlo, si imposti un trigger nella definizione della base di dati. Un trigger è una routine che viene eseguita automaticamente quando si verifica un evento o quando passa un determinato periodo di tempo. Un trigger può registrare l'immissione mancante in un file quando viene memorizzato un valore nullo o altri valori mancanti, mentre un altro trigger può essere avviato periodicamente per creare un report dei contenuti di questo file di registro.
3. Eseguire un test di sensibilità, così che i dati mancanti vengano ignorati a meno che non si sappia che un valore possa modificare significativamente un risultato (per esempio, come quando le vendite mensili totali per un agente sono quasi oltre la soglia che costituirebbero una differenza nel compenso a quell'agente). Questo è il metodo più complesso tra quelli citati e richiede una programmazione più sofisticata. Tali routine per la gestione dei dati mancanti possono essere scritte nei programmi applicativi. Molti DBMS moderni hanno ora funzionalità di programmazione più sofisticate, come le espressioni CASE, le funzioni definite dall'utente e i trigger, così che la logica possa essere disponibile nella base di dati per tutti gli utenti senza una programmazione specifica dell'applicazione.

8.3 La progettazione dei record fisici e la denormalizzazione

In un modello di dati logico si raggruppano in una relazione quegli attributi che sono determinati dalla stessa chiave primaria. Per contro, un record fisico è un gruppo di campi memorizzato in posizioni in memoria adiacenti e recuperato e scritto come un'unità da un DBMS. La progettazione di un record fisico implica la scelta di collocare i file in sequenza in posizioni in memoria adiacenti per raggiungere due obiettivi: l'uso efficiente della memoria secondaria e la velocità nell'elaborazione dei dati.

L'uso efficiente della memoria secondaria è influenzato sia della dimensione del record specifico sia dalla struttura della memoria secondaria stessa. I sistemi operativi leggono i dati dai dischi rigidi in unità chiamate pagine, e non dai record fisici. Una pagina è l'entità dei dati letti o scritti da un sistema operativo in un input nella memoria secondaria o in un'operazione di output. La dimensione della pagina è fissata dai programmatori di sistema ed è scelta per utilizzare la RAM con la massima efficienza tra tutte le applicazioni. A seconda del sistema operativo, a un record fisico può o può non essere concesso di estendersi su due pagine. Così, se la lunghezza della pagina non è un intero multiplo della lunghezza del record, può esserci uno spazio vuoto alla fine di una pagina. Il numero di record fisici per pagina è chiamato fattore di blocco. Se lo spazio di memoria è scarso e i record fisici non possono estendersi su più pagine, si possono creare più record fisici da una relazione logica per ridurre al minimo lo spreco di memoria. Alcuni DBMS raggruppano anche i record fisici multipli in un blocco di dati; in questo caso il DBMS gestisce un blocco di dati, mentre il sistema operativo gestisce una pagina.

8.3.1 Denormalizzazione

La discussione precedente sulla progettazione dei record fisici si concentrava su un uso efficace dello spazio di memoria. Nella maggior parte dei casi, il secondo obiettivo della progettazione dei record fisici, ovvero un'elaborazione efficiente dei dati, domina il processo di progettazione. L'elaborazione efficiente dei dati, così come l'accesso efficiente ai volumi in una biblioteca, dipende da quanto vicini sono i dati correlati (o i libri). Spesso tutti gli attributi che appaiono in una relazione non vengono utilizzati insieme, e sono necessari i dati di diverse relazioni per rispondere a un'interrogazione o per produrre un report. Di conseguenza, sebbene le relazioni normalizzate risolvano le anomalie di gestione dei dati e riducano le ridondanze, se implementate una per una come record fisici, possono non dare un'elaborazione dei dati efficiente.

Le differenze nelle prestazioni di elaborazione tra basi di dati completamente e parzialmente normalizzate possono essere radicali. Inmon (1988) cita uno studio fatto per quantificare questi tipi di basi di dati. Una base di dati completamente normalizzata conteneva otto tabelle con circa 50.000 righe, un'altra parzialmente normalizzata aveva quattro tabelle con circa 25.000 righe ciascuna e un'altra base di dati anch'essa parzialmente normalizzata aveva due tabelle. Il risultato mostrava che le basi dati parzialmente normalizzate avevano un ordine di magnitudo più veloce di quella completamente normalizzata. Sebbene questi risultati dipendano in gran parte dalla base di dati e dal tipo di elaborazione a cui è sottoposta, si può anche dedurre che occorre valutare molto attentamente se i record fisici debbano coincidere esattamente con le relazioni normalizzate per una base di dati.

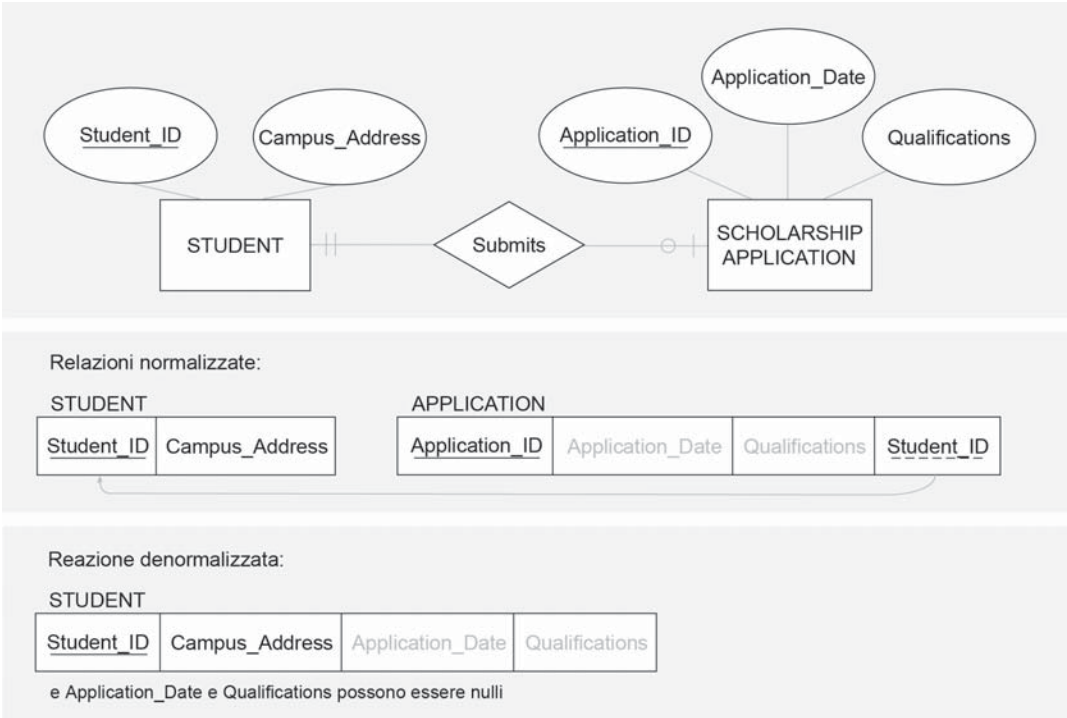
La denormalizzazione è il processo di trasformazione delle relazioni normalizzate in specifiche di record fisici non normalizzati. In questo paragrafo vedremo le varie forme e i motivi della denormalizzazione. In generale, la denormalizzazione può suddividere una relazione in numerosi record fisici, può unire attributi di più relazioni in un unico record fisico o può essere una combinazione delle due cose. Come evidenzia Finkelstein (1988), la denormalizzazione può ridurre l'eventualità di errori e incoerenze e può costringere alla riprogrammazione dei sistemi qualora le regole aziendali cambino. Inoltre, la denormalizzazione ottimizza l'elaborazione di determinati dati rispetto ad altri, così che se le frequenze di varie attività di elaborazione cambiano, i vantaggi della denormalizzazione possono andare perduti. La denormalizzazione porta quasi sempre a uno spazio di memoria maggiore per i dati grezzi e ancora a più spazio per l'overhead delle basi di dati (per esempio gli indici). Dovrebbe quindi essere un atto esplicito per l'ottenimento di una velocità di elaborazione significativa quando altre azioni di progettazioni fisica non sono sufficienti per raggiungere le aspettative di elaborazione.

Rogers (1989) ha introdotto diverse possibilità di denormalizzazione (le Figure dalla 8.3 alla 8.5 mostrano alcuni esempi di relazioni normalizzate e denormalizzate per ognuna di queste tre situazioni).

1. *Due entità con un'associazione uno-a-uno.* Anche se una delle entità è un partecipante facoltativo, se l'entità corrispondente esiste, allora può essere saggio combinare queste due relazioni nella definizione di un record (specialmente se la frequenza di accesso a questi due tipi di entità è alta). La Figura 8.3 mostra i dati dello studente con i dati facoltativi dell'iscrizione standard che uno studente deve completare. In questo caso, un record può essere formato con quattro campi dalle relazioni normalizzate STUDENT e SCHOLARSHIP APPLICATION, presumendo che Application_ID non sia più necessario. (Nota: in questo caso, ai campi dell'entità facoltativa deve essere concessa la presenza di valori nulli.)
2. *Un'associazione multi-a-molti (entità associativa) con attributi non di chiave.* Invece di unire tre file per estrarre i dati dalle due entità di base nell'associazione, può essere consigliabile combinare attributi da una delle entità nel record che rappresenta l'associazione multi-a-molti, evitando così un'operazione di unione in molti moduli di accesso ai dati. Anche in questo caso l'operazione è più

Figura 8.3

Una situazione di denormalizzazione possibile: due entità con un'associazione uno-a-uno. (Nota: si presume che Application_ID non sia necessario quando tutti i campi sono memorizzati in un record, ma questo campo può essere incluso se è un dato richiesto dall'applicazione.)



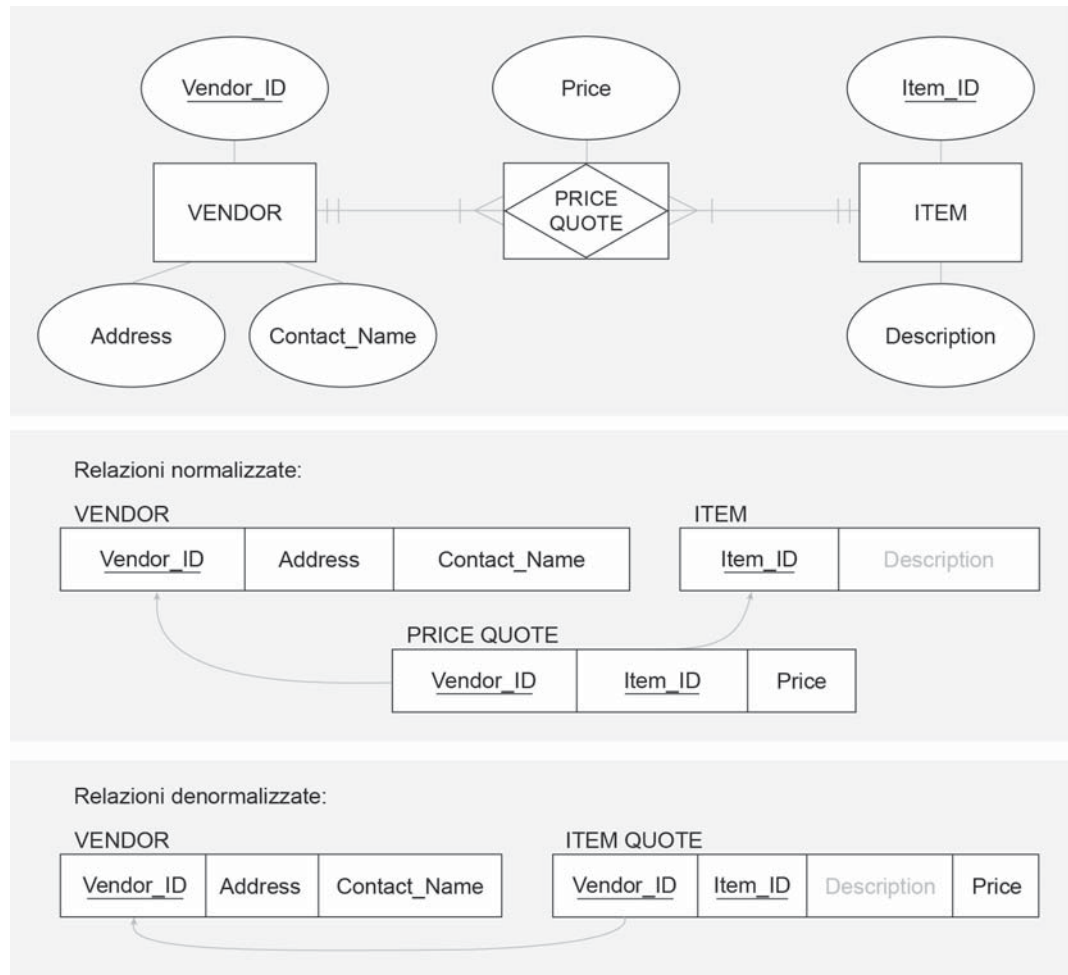
vantaggiosa se l'unione si verifica di frequente. La Figura 8.4 mostra le quote di prezzi per diversi articoli di diversi produttori. In questo caso, le relazioni ITEM e PRICE QUOTE possono essere combinate in un unico record per evitare l'unione di tre file. (Nota: questo può provocare una duplicazione rilevante dei dati; nell'esempio, i campi ITEM, come Description, si ripeterebbero per ciascuna quota di prezzo. Tutto ciò necessiterebbe di un aggiornamento eccessivo nel momento in cui questi dati duplicati cambiassero.)

3. *Dati di riferimento.* I dati di riferimento esistono in un'entità sul lato uno di un'associazione uno-a-molti, e l'entità non partecipa in altre associazioni della base di dati. È consigliabile prendere in considerazione l'ipotesi di unire le due entità in questa situazione nella definizione di un record quando ci sono poche istanze dell'entità sul lato molti per ogni istanza di entità sul lato uno. Nella Figura 8.5, molti ITEM hanno le stesse STORAGE INSTRUCTIONS e le STORAGE INSTRUCTIONS si riferiscono solo agli ITEM. In questo caso i dati delle istruzioni di memorizzazione potrebbero essere memorizzati nel record di ITEM creando ovviamente una ridondanza e un potenziale per una manutenzione supplementare dei dati (Instr_ID non è più necessario).

Le opportunità appena elencate implicano tutte tabelle che si combinano per evitare la creazione di join (unioni). Per contro, la denormalizzazione può essere utilizzata anche per creare più tabelle suddividendo una relazione in tabelle multiple. Sono possibili il partizionamento, o combinazione, verticale e orizzontale. Il partizionamento orizzontale suddivide una relazione in più specifiche di record collocando righe diverse in record diversi in base ai valori di colonna comuni. Seguendo la metafora di una biblioteca, il partizionamento orizzontale è simile a collocare le riviste di affari in una biblioteca aziendale, quelli di scienze in una scientifica e così via. Ogni file generato dal partizionamento ha lo stesso

Figura 8.4

Una possibile situazione di denormalizzazione: un'associazione molti-a-molti con attributi non di chiave.

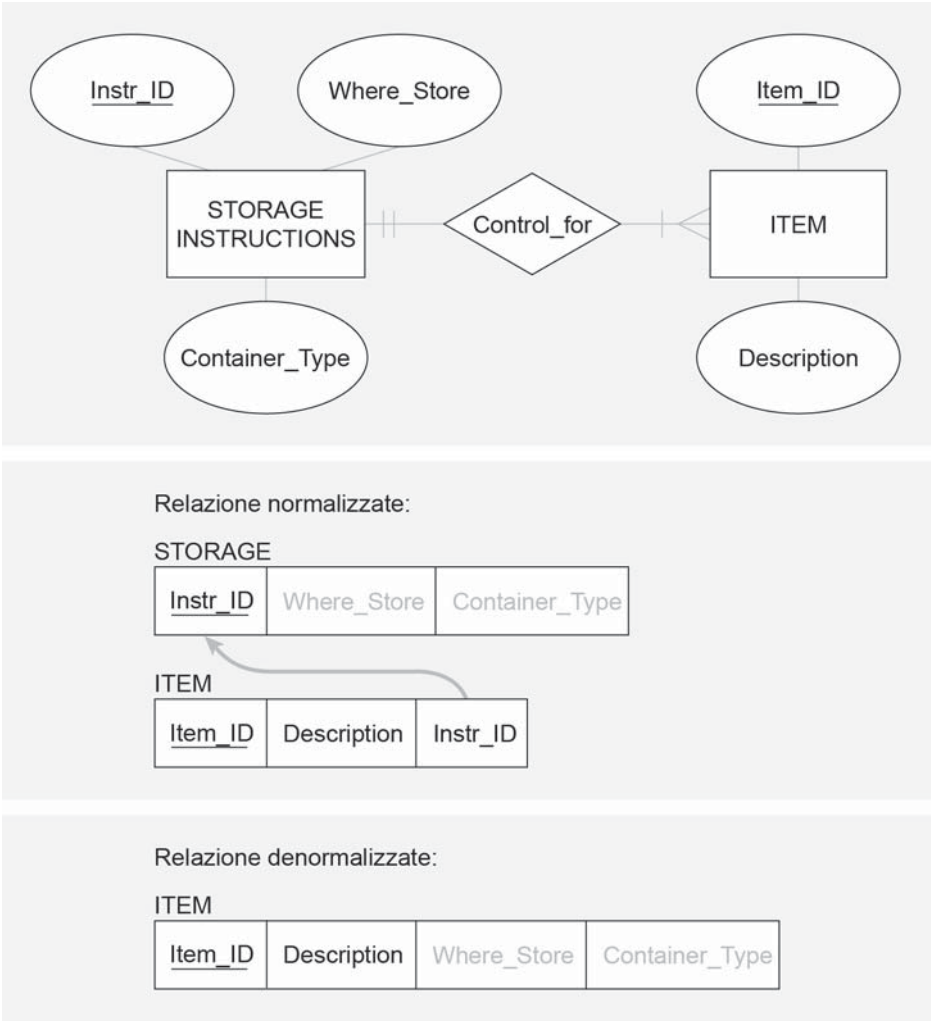


layout di record. Per esempio, una relazione di cliente può essere suddivisa in quattro file di clienti per zona in base al valore di un campo Region.

Il partizionamento orizzontale ha senso quando diverse categorie di righe di una tabella vengono elaborate separatamente, per esempio, per la tabella di clienti appena citata, se gran parte dell'elaborazione dei dati deve funzionare con una sola zona alla volta. Il partizionamento orizzontale è anche più sicuro, poiché il livello di protezione del file può essere utilizzato per impedire agli utenti di vedere determinate righe di dati. Ogni tabella partizionata può anche essere organizzata diversamente in base a come viene utilizzata. Probabilmente è anche più veloce ripristinare uno dei file partizionati piuttosto che uno con tutte le righe. Quando si prende uno dei file partizionati non in uso perché danneggiato per ripristinarlo, l'elaborazione degli altri file partizionati può comunque continuare. Infine, ognuno dei file partizionati può essere collocato su un'unità di disco rigido separata per ridurre il conflitto sulla stessa unità e migliorare quindi le prestazioni nella base di dati. I vantaggi e gli svantaggi del partizionamento orizzontale (e di tutte le forme di partizionamento) sono riassunti nella Tabella 8.2.

Si noti che il partizionamento orizzontale è molto simile alla creazione di un'associazione supertipo/sottotipo, perché i diversi tipi di entità (dove la discriminante di sottotipo è il campo utilizzato per isola-

Figura 8.5
Una possibile
situazione
di denormalizzazione:
dati di riferimento.



re le righe) sono implicati in associazioni diverse, e da qui un’elaborazione diversa. In realtà, quando si ha un’associazione supertipo/sottotipo, occorre decidere se creare tabelle separate per ciascun sottotipo o se unirle in varie combinazioni. L’unione ha senso quando tutti i sottotipi vengono utilizzati più o meno nello stesso modo, mentre il partizionamento dell’entità di supertipo in più file ha senso quando i sottotipi sono gestiti in modo diverso nelle transazioni, nelle interrogazioni e nei report. Quando una relazione viene partizionata orizzontalmente, l’intero insieme di righe può essere ricostruito dall’operatore UNION SQL (descritto nel Capitolo 9); così, per esempio, tutti i dati dei clienti possono essere visualizzati insieme quando desiderato.

Il sistema di gestione di basi di dati di Oracle supporta numerose forme di partizionamento orizzontale, concepite in particolare per gestire tabelle molto grandi (Brobst e altri, 1999). Una tabella viene partizionata quando è definita nel DBMS utilizzando il linguaggio di definizione dei dati SQL (il comando CREATE TABLE verrà trattato nel Capitolo 9). In Oracle c’è solo una tabella con più partizioni e non singole tabelle separate. Oracle 8i prevede tre metodi di partizionamento orizzontale.

Tabella 8.2 Vantaggi e svantaggi del partizionamento dei dati.**Vantaggi**

1. *Efficienza*: i dati utilizzati insieme vengono memorizzati gli uni vicini agli altri e separatamente dai dati non utilizzati insieme.
2. *Ottimizzazione locale*: ogni partizione di dati può essere memorizzata per ottimizzare le prestazioni relative al suo uso specifico.
3. *Protezione*: i dati non importanti per un gruppo di utenti possono essere isolati dai dati che gli stessi utenti hanno il permesso di utilizzare.
4. *Ripristino e aggiornamento*: i file più piccoli vengono ripristinati in meno tempo e gli altri file sono comunque accessibili se uno è danneggiato; gli effetti del danno rimangono quindi isolati.
5. *Bilanciamento del carico*: i file possono essere allocati in aree di memoria diverse (dischi o altri supporti), il che riduce al minimo i conflitti di accesso alla stessa area di memoria o consente un accesso parallelo alle diverse aree.

Svantaggi

1. *Velocità di accesso non uniformi*: le diverse partizioni possono dare velocità di accesso diverse, confondendo gli utenti. Inoltre, quando i dati devono essere combinati tra più partizioni, gli utenti potrebbero dover affrontare tempi di risposta significativamente più lenti che non con un approccio non a partizioni.
2. *Complessità*: il partizionamento è solitamente più trasparente per i programmatori, che si trovano a dover scrivere programmi più complessi quando combinano dati tra più partizioni.
3. *Spazio di memoria extra e tempi di aggiornamento più lunghi*: i dati possono essere duplicati tra le partizioni, occupando uno spazio di memoria extra per memorizzare tutti i dati nei file normalizzati. Gli aggiornamenti che riguardano i dati in partizioni multiple possono impiegare più tempo che non se venisse utilizzato un solo file.

1. *Partizionamento di intervallo di chiave*, nel quale ogni partizione è definita da un intervallo di valori (limiti di valore di chiave superiori e inferiori) per una o più colonne della tabella normalizzata. Una riga di tabella è inserita nella partizione opportuna in base ai suoi valori iniziali per i campi dell'intervallo. Poiché i valori di chiave delle partizioni devono seguire dei pattern, ogni partizione può contenere un numero di righe piuttosto differente. Una chiave di partizione può essere generata dal progettista della base di dati per creare una distribuzione più bilanciata delle righe. È possibile limitare lo spostamento delle righe tra le partizioni quando i valori di chiave vengono aggiornati.
2. *Partizionamento hash*, nel quale i dati vengono distribuiti regolarmente tra le partizioni a prescindere dal valore di chiave della partizione. Il partizionamento hash supera la distribuzione irregolare delle righe che può verificarsi con il partizionamento di intervallo di chiave.
3. *Partizionamento composto*, che combina aspetti del partizionamento di intervallo di chiave e hash. Le partizioni possono essere trasparenti per l'utente della base di dati (si deve fare riferimento a una partizione solo se si vuol costringere il processore dell'interrogazione a vedere in una o più partizioni). La parte del DBMS che ottimizza l'elaborazione di un'interrogazione cercherà la definizione delle partizioni per una tabella coinvolta nell'interrogazione e deciderà automaticamente se determinate partizioni possono essere eliminate quando si recuperano i dati necessari a formare i risultati dell'interrogazione, cosa che può migliorare nettamente le prestazioni di elaborazione delle interrogazioni.

Per esempio, si supponga che la data di una transazione venga utilizzata per definire le partizioni in un partizionamento di intervallo di chiave. Un'interrogazione che chiede solo le transazioni più recenti può essere elaborata più rapidamente se si considera solo la partizione, o le poche partizioni, con le transazioni più recenti invece che l'intera base di dati o anche se si utilizzano gli indici per trovare le righe nell'intervallo desiderato da una tabella non partizionata. Inoltre una partizione sulle date isola gli inserimenti di nuove righe in una partizione, cosa che può ridurre i costi di gestione della base di dati; ignorando le transazioni "vecchie", si ignora semplicemente una partizione. Gli indici possono essere comunque utilizzati con una tabella partizionata e possono migliorare le prestazioni anche più del solo

partizionamento. Si veda Brobst e altri (1999) per ulteriori dettagli sui pro e i contro dell'uso delle date per il partizionamento di intervallo di chiave.

Nel partizionamento hash, le righe vengono distribuite in modo ancora più regolare tra le partizioni. Se le partizioni vengono collocate in aree di memoria diverse che possono essere elaborate in parallelo, allora le prestazioni delle interrogazioni possono migliorare notevolmente rispetto a quando occorre accedere a tutti i dati in modo sequenziale in un'area di memoria per l'intera tabella. Come nel partizionamento di intervallo di chiave, l'esistenza di partizioni può essere trasparente per il programmatore di un'interrogazione. Con il partizionamento composto, le partizioni vengono definite da intervalli di chiave, mentre le sottopartizioni vengono definite dall'hashing di chiave. Il partizionamento composto unisce i vantaggi della separazione degli intervalli con l'elaborazione parallela dell'hashing di chiave.

Il partizionamento verticale distribuisce le colonne di una relazione in record fisici, ripetendo la chiave primaria in ognuno dei record. Un esempio di partizionamento verticale è la suddivisione di una relazione di parti che avviene collocando il numero della parte con i suoi dati tecnici in un'altra specifica di record, e il numero della parte con i suoi dati di vendita in un'altra specifica di record. I vantaggi e gli svantaggi del partizionamento verticale sono simili a quelli del partizionamento orizzontale. Per esempio, quando i dati contabili, tecnici e di vendita delle parti devono essere utilizzati insieme, è possibile unire le rispettive tabelle. Di conseguenza, né il partizionamento orizzontale né quello verticale impediscono di trattare la relazione originaria come un tutt'uno.

Sono anche possibili combinazioni del partizionamento orizzontale con quello verticale. Questa forma di denormalizzazione, chiamata *partizionamento dei record*, è diffusa soprattutto nelle basi di dati i cui file sono distribuiti tra più computer.

Una singola tabella fisica può essere partizionata logicamente e più tabelle possono essere unite logicamente utilizzando il concetto di *vista utente*, che verrà definito nel Capitolo 9. Con una vista utente, si può conferire negli utenti l'impressione che la base di dati contenga tabelle diverse da quelle fisicamente definite; queste tabelle logiche possono essere create eseguendo un partizionamento orizzontale e verticale o altre forme di denormalizzazione. Lo scopo di qualsiasi forma di vista utente, compreso il partizionamento logico mediante le viste, è tuttavia quello di semplificare la scrittura delle interrogazioni e di creare basi di dati più sicure, non quello di migliorare le prestazioni delle interrogazioni. Una forma di vista utente disponibile in Oracle è chiamata Partition View. Con una Partition View, tabelle fisicamente separate che hanno strutture simili possono essere combinate logicamente in un'unica tabella utilizzando l'operatore UNION SQL. Ci sono alcuni limiti a questo tipo di partizionamento. Innanzitutto, poiché esistono effettivamente numerose tabelle fisiche separate, non può esistere un indice globale su tutte le righe combinate. In secondo luogo, ogni tabella fisica deve essere gestita separatamente; la gestione dei dati diventa così più complessa (per esempio, una nuova riga deve essere inserita in una tabella specifica). In terzo luogo, l'ottimizzatore delle interrogazioni ha meno opzioni con una Partition View che non con le partizioni di una tabella singola quando deve creare un piano di elaborazione delle interrogazioni più efficiente.

La forma finale di denormalizzazione è la duplicazione dei dati. Nella duplicazione dei dati, gli stessi dati vengono memorizzati di proposito in più punti della base di dati. Per esempio, si consideri nuovamente la Figura 8.1. In questo paragrafo si è visto che le relazioni possono essere denormalizzate unendo i dati di un'entità associativa con quelli di una delle entità semplici a cui sono associati. Nella figura, i dati di QUOTATION possono essere memorizzati con i dati di PURCHASED PART in una specifica di record fisico PURCHASED PART estesa. Con la duplicazione dei dati, gli stessi dati di QUOTATION possono essere memorizzati anche con i dati di SUPPLIER associati in un'altra specifica di record fisico SUPPLIER. Con questa duplicazione dei dati, una volta che è stato recuperato il record di SUPPLIER o di PURCHASED PART, i dati correlati di QUOTATION saranno disponibili senza dover accedere ulteriormente alla memoria secondaria. Questa velocità migliorata ha senso solo se si

accede spesso ai dati di QUOTATION con i dati di SUPPLIER e di PURCHASED PART e se i costi della memoria secondaria e della manutenzione dei dati non sono eccessivi.

8.4 Progettare i file fisici

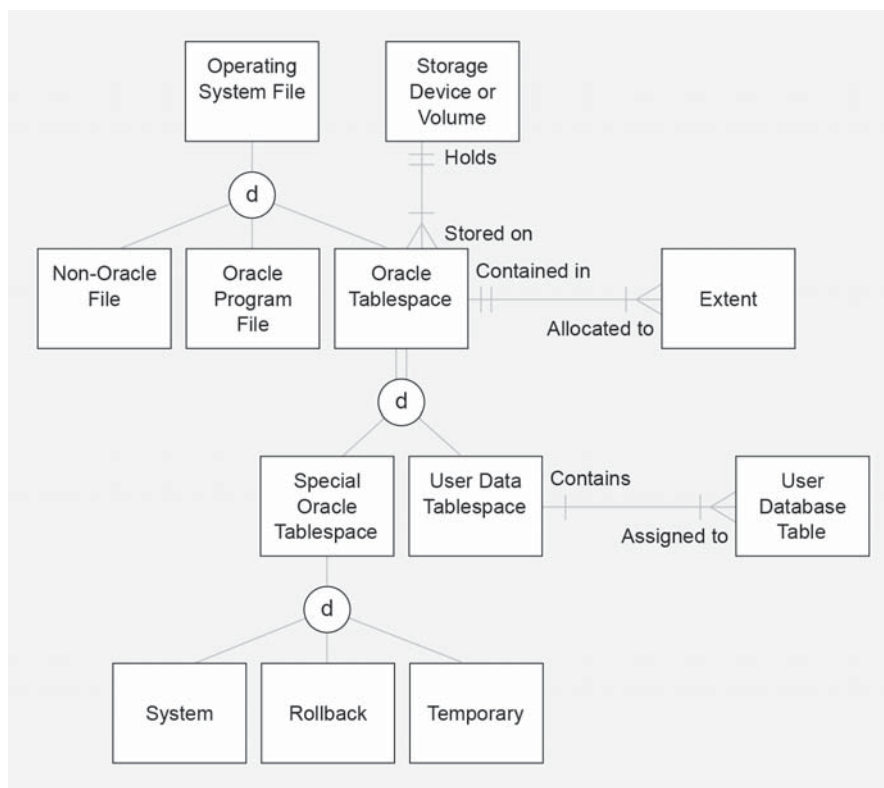
Un file fisico è una porzione denominata della memoria secondaria (come un nastro magnetico o un disco rigido) riservata alla memorizzazione dei record fisici. Alcuni sistemi operativi permettono che un file venga suddiviso in parti separate, chiamate anche *estensioni*. Nei prossimi paragrafi si presumerà che un file fisico non sia suddiviso e che ogni record in un file abbia la stessa struttura. Si vedrà quindi come memorizzare e collegare le righe delle tabelle relazionali di una singola base di dati in uno spazio di memoria fisico. Per poter ottimizzare le prestazioni dell'elaborazione della base di dati, la persona che amministra una base di dati, chiamato amministratore della base di dati, deve conoscere i dettagli di come il DBMS gestisce lo spazio di memoria fisico. Questa conoscenza è specifica del DBMS, ma i principi descritti nei prossimi paragrafi sono i fondamenti per le strutture fisiche dei dati utilizzati dalla maggior parte dei DBMS.

La gran parte dei sistemi di gestione delle basi di dati memorizza molti tipi diversi di dati in un unico file di sistema operativo. Con il termine "file di sistema operativo" si intende un file denominato che appare nell'elenco delle directory di un disco (come l'elenco dei file in una cartella sull'unità C: di un personal computer). Per esempio, una struttura importante per lo spazio di memoria fisico in Oracle è un tablespace. Un tablespace è un insieme denominato di elementi di memorizzazione su disco nel quale possono essere memorizzati i dati di una o più tabelle della base di dati. Un'istanza di Oracle include molti tablespace, per esempio uno per il ripristino della base di dati e diversi per contenere i dati aziendali degli utenti. Uno o più tablespace sono contenuti in un file fisico del sistema operativo. Oracle ha quindi la responsabilità di gestire la memorizzazione dei dati all'interno di un tablespace, laddove il sistema operativo ha la responsabilità della gestione di un tablespace nel complesso come di qualsiasi altro file del sistema operativo (per esempio, la gestione del livello di protezione dei file, l'allocazione dello spazio e la risposta alla lettura dei dischi e agli errori).

Poiché un'istanza di Oracle supporta generalmente molte basi di dati per molti utenti, l'amministratore della base di dati crea solitamente numerosi tablespace utente, che aiutano a raggiungere la sicurezza di una base di dati assegnando a ciascun utente i diritti selezionati per accedere a ogni tablespace. Come visto in precedenza, in quanto file di sistema operativo, un tablespace può essere distribuito su più estensioni, dove un'estensione è una sezione adiacente dello spazio di memoria di un disco. Quando un tablespace deve essere ampliato per contenere più dati, viene assegnata un'altra estensione. A ogni tabella di base di dati è assegnato uno o più tablespace (ogni riga di tabella è un tablespace, e solo uno, ma righe diverse della stessa tabella possono trovarsi in tablespace diversi); un tablespace può contenere i dati di una o più tabelle. La gestione dei tablespace, o dei file di basi di dati fisici, è un lavoro piuttosto impegnativo che in un ambiente Oracle tocca all'amministratore della base di dati. Per esempio, trovando tablespace diversi su dispositivi o canali diversi e distribuendo le tabelle tra i tablespace, un amministratore della base di dati può ridurre il conflitto delle unità disco tra utenti simultanei della base di dati. Poiché questo non è un testo su Oracle, non tratteremo i dettagli della gestione dei tablespace; è comunque giusto sapere che i principi generali della progettazione fisica di una base di dati valgono anche per la progettazione e la gestione dei tablespace Oracle così come di qualsiasi unità di memoria fisica di tutti i sistemi di gestione di una base di dati. La Figura 8.6 è un modello EER che mostra le associazioni tra vari termini fisici e logici di una base di dati correlati alla progettazione fisica di una base di dati in un ambiente Oracle.

Figura 8.6

Terminologia dei file fisici in un ambiente Oracle.



8.4.1 Puntatori

Tutti i file vengono organizzati utilizzando due costrutti di base per collegare un segmento di dati con un altro: la memorizzazione sequenziale e i puntatori. Con la memorizzazione sequenziale un file o record viene memorizzato subito dopo un altro campo o record. Sebbene semplice da implementare e utilizzare, a volte non è il metodo più efficiente per organizzare i dati. Un puntatore è un campo di dati che può essere utilizzato per trovare un campo o record correlato di dati. Nella maggior parte dei casi, un puntatore contiene un indirizzo, o posizione, dei dati associati. I puntatori vengono utilizzati in un'ampia gamma di strutture di memorizzazione; tuttavia, in questo capitolo ne vedremo solo alcune. Il lettore interessato ad approfondire l'argomento può consultare l'Appendice C. I puntatori vengono definiti in questo punto solo perché sono necessari per capire le organizzazioni di file. È probabile che non si lavorerà mai direttamente con i puntatori, dato che il DBMS ne gestisce automaticamente l'uso e la manutenzione.

8.4.2 Organizzazioni di file

Un'organizzazione di file è una tecnica per disporre fisicamente i record di un file su dispositivi di memorizzazione secondari. Con i DBMS moderni, non è più necessario progettare organizzazioni di file, ma si può ancora scegliere un'organizzazione e i suoi parametri per una tabella o un file fisico. Nello scegliere un'organizzazione di file per un determinato file in una base di dati, si devono considerare sette fattori importanti.

- 1. Recupero veloce dei dati.
- 2. Uso elevato delle risorse per l’elaborazione dell’input dei dati e il mantenimento delle transazioni.
- 3. Uso efficace dello spazio di memoria.
- 4. Protezione da fallimenti o perdite di dati.
- 5. Riduzione della necessità di organizzazione.
- 6. Adattamento alla crescita.
- 7. Protezione da un uso non autorizzato.

Questi obiettivi entrano spesso in conflitto tra loro, ed è necessario scegliere un’organizzazione di file che fornisca un equilibrio ragionevole tra i vari criteri nel contesto delle risorse disponibili.

In questo capitolo si considereranno le seguenti categorie di organizzazioni di file di base: sequenziale, indicizzata e di hashing. La Figura 8.7 illustra ciascuna di queste organizzazioni con i nomi di alcune squadre sportive universitarie.

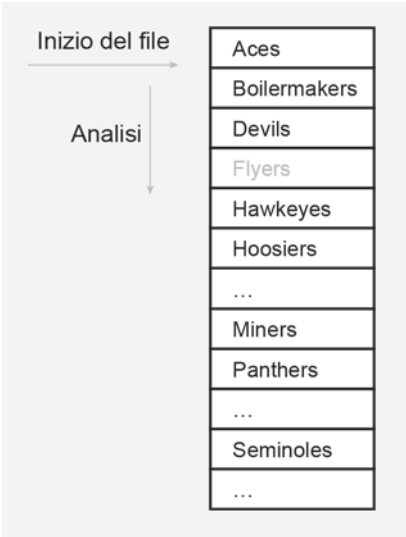
Organizzazioni di file sequenziali

In un’organizzazione di file sequenziale, i record nel file vengono memorizzati in sequenza in base al valore della chiave primaria (Figura 8.7a). Per trovare un determinato record, un programma deve solitamente analizzare il file dall’inizio fino a che il record desiderato non viene individuato. Un esempio comune di file sequenziale è l’elenco alfabetico di persone nelle pagine bianche di una directory telefonica (ignorando qualsiasi indice che può essere incluso nella directory). Un confronto delle potenzialità dei file sequenziali con gli altri due tipi di file è mostrata nella Tabella 8.3. A causa della loro mancanza di flessibilità, i file sequenziali non vengono utilizzati in una base di dati, ma possono essere utilizzati per i file di backup della stessa.

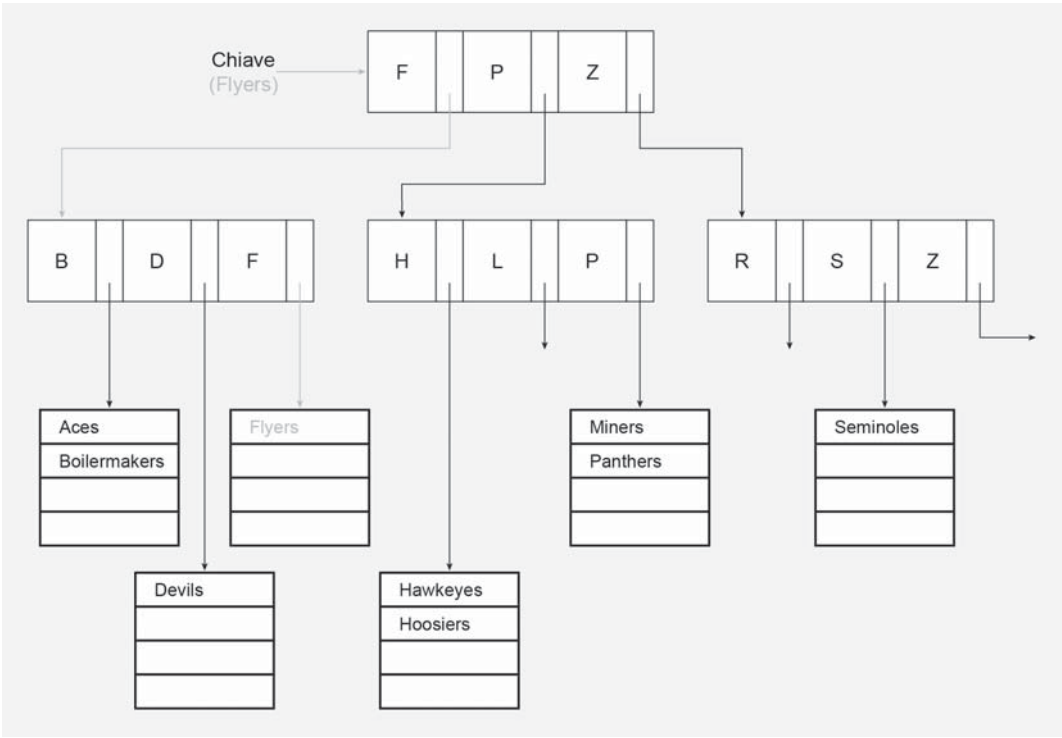
Tabella 8.3 Confronto tra le funzioni di diversi tipi di organizzazioni di file.

Fattore	Organizzazione di file		
	Sequenziale	Indicizzata	Di hashing
Spazio di memoria	Nessuno spreco di spazio	Nessuno spreco di spazio ma spazio extra per l'indice	Deve essere aggiunto uno spazio extra per permettere l’inserimento e l’eliminazione dei record dopo che l’insieme iniziale dei record è stato caricato.
Recupero sequenziale sulla chiave primaria	Molto veloce	Moderatamente veloce	Poco pratico, a meno di non utilizzare un indice hash
Recupero casuale sulla chiave primaria	Poco pratico	Moderatamente veloce	Molto veloce
Recupero di chiave multipla	Possibile, ma richiede l’analisi dell’intero file	Molto veloce con indici multipli	Non possibile, a meno di non utilizzare un indice hash
Eliminazione dei record	Può comportare uno spreco di spazio o richiedere la riorganizzazione	Se lo spazio può essere allocato dinamicamente, è semplice, ma richiede la manutenzione degli indici	Molto semplice
Aggiunta di nuovi record	Richiede la riscrittura del file	Se lo spazio può essere allocato dinamicamente, è semplice, ma richiede la manutenzione degli indici	Molto semplice; le chiavi multiple con lo stesso indirizzo richiedono però un lavoro supplementare
Aggiornamento dei record	Solitamente richiede la riscrittura del file	Semplice, ma richiede la manutenzione degli indici	Molto semplice

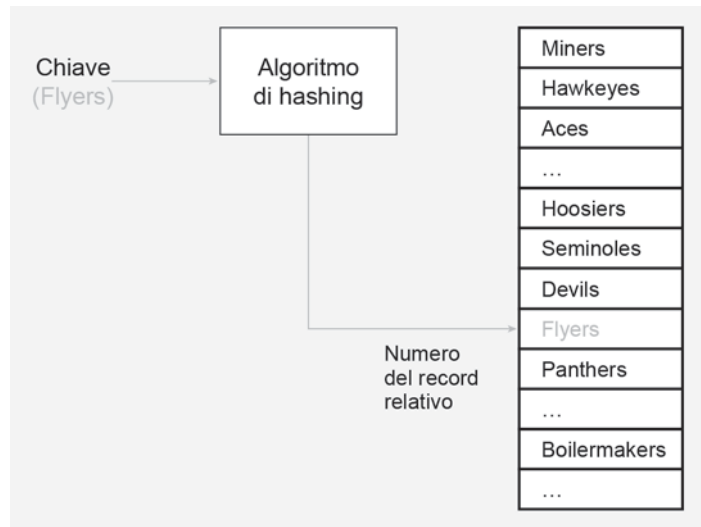
Figura 8.7
Confronto tra
organizzazioni di file.
(a) Sequenziale.



(b) Indicizzata.



(c) Di hashing.



Organizzazioni di file indicizzati

In un'organizzazione di file indicizzata, i record vengono memorizzati sia sequenzialmente sia non sequenzialmente e viene creato un indice che permette al software applicativo di trovare i singoli record (Figura 8.7b). Come uno schedario di una biblioteca, un **indice** è una tabella utilizzata per determinare la posizione delle righe in un file che soddisfano alcune condizioni. Ogni immissione dell'indice corrisponde a un valore di chiave con uno o più record. Un indice può puntare a record univoci (un indice di chiave primaria, come il campo Product_ID del record di un prodotto) o potenzialmente a più di un record. Un indice che permette a ogni immissione di puntare a uno o più record è chiamato **indice di chiave secondaria**. Gli indici di chiave secondaria sono importanti per supportare diverse esigenze per i report e per fornire un rapido recupero ad hoc dei dati. Un esempio è quello di un indice sul campo Finish di un record di prodotto.

Alcune strutture di indice influenzano il punto in cui le righe delle tabelle vengono memorizzate, mentre altre prescindono dalla posizione in cui vengono trovate le righe. Poiché la struttura effettiva di un indice non influisce sulla progettazione della base e non è importante nella scrittura delle interrogazioni, in questo capitolo non ce ne occuperemo. La Figura 8.7b va considerata come una vista logica di come un indice è utilizzato, e non come una vista fisica di come i dati sono memorizzati in una struttura di indice. Alcuni DBMS relazionali utilizzano il termine *indice primario* (diverso da *indice di chiave primaria*) per indicare un indice che determina la posizione fisica dei dati, mentre un *indice secondario* (diverso dall'*indice di chiave secondaria*) è un indice che non gioca alcun ruolo nel determinare la posizione di memorizzazione e può avere molti indici secondari. L'indice primario può utilizzare o non utilizzare la chiave primaria della tabella, mentre un indice secondario può utilizzare un campo che è univoco o non univoco tra le righe della tabella. Quando gli indici di termini primari e secondari vengono utilizzati, si hanno quattro tipi di indici.

- **Indice primario univoco (UPI)**, un indice su un campo univoco, solitamente la chiave primaria della tabella, che non solo viene utilizzato per trovare le righe della tabella in base al valore di quel campo, ma che viene anche utilizzato dal DBMS per determinare dove memorizzare una riga in base al valore del campo di indice primario.

- *Indice primario non univoco (NUPI)*, un indice su un campo non univoco che non solo viene utilizzato per trovare le righe della tabella in base al valore di quel campo, ma che viene anche utilizzato dal DBMS per determinare dove memorizzare una riga in base al valore del campo di indice primario.
- *Indice secondario univoco (USI)*, un indice su un campo univoco che viene utilizzato solo per trovare le righe della tabella in base al valore di quel campo.
- *Indice secondario non univoco (NUSI)*, un indice su un campo non univoco che viene utilizzato solo per trovare le righe della tabella in base al valore di quel campo.

L'esempio nella Figura 8.7b mostra che gli indici possono essere costruiti al livello superiore di altri indici, creando così un insieme gerarchico di indici. Questo può aiutare, poiché un indice è anch'esso un file, e se è molto grande, può essere organizzato indicizzandolo. Ogni immissione dell'indice nella Figura 8.7b ha un valore di chiave e un puntatore per un altro indice o per un record di dati. Per esempio, per trovare il record con la chiave "Flyers", l'organizzazione dei file dovrebbe partire dall'indice superiore e prendere il puntatore dopo l'immissione F, che punta a un altro indice per tutte le chiavi che iniziano con la lettera F in questo indice, che rappresenta tutti quei record con le chiavi che iniziano con le lettere dalla E alla F. La ricerca negli indici può trovare il record desiderato o indicare che quel record non esiste.

Una delle potenzialità maggiori della organizzazione dei file è la capacità di creare indici multipli. Gli indici multipli esistono anche in una biblioteca, dove ce ne sono per titolo, per autore e per argomento, che puntano allo stesso insieme di libri o riviste. È quindi possibile creare virtualmente un altro file creando semplicemente un altro indice sullo stesso insieme di record di dati, così che dati molto voluminosi non vengano duplicati. Gli indici multipli possono essere manipolati. Per esempio, se si utilizza l'indice Finish per trovare gli indirizzi per l'insieme dei record di prodotti con la rifinitura Birch e l'indice Cost per trovare l'insieme dei record di prodotti la cui realizzazione costa meno di 500 \$, è possibile intersecare questi due insiemi per trovare gli indirizzi dei prodotti, e solo dei prodotti, che sono realizzati in betulla e che costano meno di 500 \$. La possibilità di utilizzare indici multipli in un'interrogazione è importante nell'elaborazione delle interrogazioni su una base di dati relazionale. Le operazioni logiche AND, OR e NOT possono essere gestite manipolando semplicemente i risultati di un'analisi dell'indice, evitando così l'accesso più costoso a record di dati che non corrispondono a tutti i criteri dell'interrogazione.

La struttura generale di un indice gerarchico come quello nella Figura 8.7b è chiamato *albero* (con il nodo radice in alto e i nodi foglia in basso; già, è un albero al contrario!). Le prestazioni delle immissioni in un indice gerarchico vengono notevolmente influenzate dalle proprietà della struttura ad albero utilizzata. Un tipo di albero comune utilizzato dai sistemi di gestione delle basi di dati per memorizzare gli indici è quello bilanciato, o *albero B* (da *B-tree*), mentre la forma più diffusa di albero B è un *albero B+*. In un albero B, tutte le foglie (che solitamente contengono i record di dati o i puntatori a ciascun record) sono memorizzate alla stessa distanza dalla radice, come nella Figura 8.7b, nella quale tutti i record di dati sono memorizzati a due livelli dalla radice. Poiché il sistema operativo e il DBMS gestiscono qualsiasi struttura ad albero utilizzata, e poiché in genere si ha poca voce in capitolo sul modo in cui l'indice è strutturato, la gestione dell'albero è trasparente per il progettista della base di dati e per il programmatore. L'Appendice C include un riepilogo delle strutture ad albero dei dati e dei parametri per la loro progettazione sui quali è possibile avere un qualche controllo.

Attualmente gode di crescente popolarità un tipo di indice chiamato indice bitmap. Un indice bitmap (Figura 8.8) assegna una bitmap a ogni valore di chiave. Si supponga che una tabella Product abbia 10 righe e che l'attributo Price abbia 4 valori diversi in queste 10 righe (è un esempio ridotto a scopo illustrativo). Un indice bitmap sul campo Price dovrebbe avere 4 immissioni di 10 bit ognuna. Ogni bit

in ciascuna immissione corrisponde a una riga nella riga 8 della tabella Product. Per esempio, se il bit 8 è attivo per l'immissione di Price di 400 \$, significa che la riga 8 della tabella Price contiene un Price di 400 \$ (ogni colonna di bit avrà solo un bit attivo). Una bitmap è ideale per gli attributi che hanno pochi valori possibili, cosa che non accade negli indici ad albero convenzionali. Una bitmap richiede inoltre meno spazio di memoria di un indice ad albero convenzionale, in genere attorno al 25 per cento (Schumacher, 1997), ma per un attributo con molti valori distinti, un indice bitmap può superare lo spazio di memoria di un normale indice ad albero. In quel caso, poiché la manipolazione dei bit e la ricerca sono veloci, la velocità di elaborazione delle interrogazioni con una bitmap può essere 10 volte maggiore che non in un indice ad albero convenzionale.

Una caratteristica interessante di un indice bitmap è che può essere utilizzato per le chiavi multiple. Per esempio, in fondo alla Figura 8.8 è possibile aggiungere un altro insieme di righe per tutti i valori dell'attributo Room. Ogni colonna di bit avrà quindi due bit attivi, uno per il prezzo della riga Product associata e una per Room. Un'interrogazione che cerca i prodotti da 100 \$ per la sala da pranzo può intersecare il pattern di bit della riga del prezzo di 100 \$ con la riga della sala da pranzo per trovare solo quei prodotti che soddisfano entrambe le condizioni. Una manipolazione di bit di questa natura può essere molto efficiente.

Le applicazioni di elaborazione delle transazioni richiedono una risposta rapida alle interrogazioni che implicano una o più righe di tabella correlate. Per esempio, per immettere un nuovo ordine di cliente, un'applicazione deve poter trovare rapidamente la riga specifica della tabella dei clienti, poche righe della tabella dei prodotti per gli articoli che devono essere acquistati ed eventualmente poche altre righe della tabella dei prodotti in base alle caratteristiche dei prodotti desiderati dai clienti (per esempio la rifinitura); una riga di ordine dei clienti e una riga per la consegna al cliente dovranno essere aggiunte dall'applicazione alle relative tabelle. I tipi di indici trattati finora funzionano molto bene in quelle applicazioni che cercano poche righe specifiche in una tabella.

Un altro tipo di indice molto diffuso, soprattutto nei data warehouse e in altre applicazioni di supporto alle decisioni (Capitolo 12) è l'indice di join. Nelle applicazioni di supporto alle decisioni l'accesso ai dati tende a estendersi su tutte le righe di tabelle molto grandi che sono correlate le une alle altre (per esempio, tutti i clienti che hanno acquistato articoli dallo stesso negozio). Un indice di join è un indice in colonne di dati tratti da due o più tabelle dello stesso dominio di valori. Per esempio, si consideri la Figura 8.9a, che mostra due tabelle, Store e Customer. Ognuna di queste tabelle ha una colonna, City. L'indice di join della colonna City indica gli identificatori di riga per le righe nelle due tabelle che hanno

Figura 8.8
Indice bitmap
sull'attributo Price di
Product.

Price	Numeri di riga della tabella Product									
	1	2	3	4	5	6	7	8	9	10
100	0	0	1	0	1	0	0	0	0	0
200	1	0	0	0	0	0	0	0	0	0
300	0	1	0	0	0	0	1	0	0	1
400	0	0	0	1	0	1	0	1	1	0

I prodotti 3 e 5 hanno un Price di 100 \$

Il prodotto 1 ha un Price di 200 \$

I prodotti 2, 7 e 10 hanno un Price di 300 \$

I prodotti 4, 6, 8 e 9 hanno un Price di 400 \$

lo stesso valore di City. A causa dei numerosi data warehouse progettati, è molto probabile che le interrogazioni trovino i dati (fatti) in comune a un negozio e a un cliente nella stessa città (o intersezioni simili di fatti tra più dimensioni). La Figura 8.9b mostra un'altra applicazione possibile di un indice di join. In questo caso, l'indice di join pre-calcola la corrispondenza di una chiave esterna nella tabella Order con il cliente associato nella tabella Customer. Un indice di join pre-calcola il risultato di un operatore di join relazionale, che verrà trattato nel Capitolo 9. In parole semplici, un join dice che troverà le righe nella stessa tabella o in tabelle diverse che hanno valori che corrispondono a un determinato criterio.

Figura 8.9
Indici di join.
(a) Indice di join per colonne comuni non di chiave.

Customer

RowID	Cust#	CustName	City	State
10001	C2027	Hadley	Dayton	Ohio
10002	C1026	Baines	Columbus	Ohio
10003	C0042	Ruskin	Columbus	Ohio
10004	C3861	Davies	Toledo	Ohio
...				

Store

RowID	Store#	City	Size	Manager
20001	S4266	Dayton	K2	E2166
20002	S2654	Columbus	K3	E0245
20003	S3789	Dayton	K4	E3330
20004	S1941	Toledo	K1	E0874
...				

Indice di join

CustRowID	StoreRowID	Valore comune*
10001	20001	Dayton
10001	20003	Dayton
10002	20002	Columbus
10003	20002	Columbus
10004	20004	Toledo
...		

* Questa colonna può essere o non essere inclusa, a seconda delle necessità. L'indice di join può essere ordinato su una qualsiasi delle tre colonne. A volte vengono creati due indici di join, uno come quello mostrato sopra e uno con le due colonne RowID in negativo.

(b) Indice di join per la corrispondenza con una chiave esterna (FK) e una chiave primaria (PK).

Order

RowID	Order#	Order Date	Cust#(FK)
30001	O5532	10/01/2001	C3861
30002	O3478	10/01/2001	C1062
30003	O8734	10/02/2001	C1062
30004	O9845	10/02/2001	C2027
...			

Customer

RowID	Cust#(PK)	CustName	City	State
10001	C2027	Hadley	Dayton	Ohio
10002	C1026	Baines	Columbus	Ohio
10003	C0042	Ruskin	Columbus	Ohio
10004	C3861	Davies	Toledo	Ohio
...				

Indice di join

CustRowID	OrderRowID	Cust#
10001	30004	C2027
10002	30002	C1062
10002	30003	C1062
10004	30001	C3861
...		

Un indice di join viene creato quando le righe vengono caricate in una base di dati; di conseguenza, l'indice e tutti gli altri indici precedentemente discussi sono sempre aggiornati. Senza un indice di join nella base di dati della Figura 8.9a, qualsiasi interrogazione che vuole trovare i negozi e i clienti nella stessa città dovrebbe calcolare l'equivalente dell'indice di join ogni volta che l'interrogazione viene eseguita. Nel caso di tabelle molto grandi, l'unione di tutte le righe di una tabella con le righe corrispondenti in un'altra tabella eventualmente grande può occupare molto tempo e può ritardare significativamente la risposta a un'interrogazione on-line. Nella Figura 8.9b, l'indice di join fornisce un punto dove il DBMS può trovare le informazioni sulle righe di tabella correlate. Bontempo e Saracco (1996) indicano come combinare gli indici di join con gli indici bitmap per velocizzare l'elaborazione di alcuni tipi di indici di join.

Ballinger (1998) suggerisce un'estensione a un indice di join che possa eliminare la necessità di accedere alle tabelle dei dati effettivi. Per esempio, nella Figura 8.9b, oltre a Cust#, altre colonne di una o entrambe le tabelle potrebbero essere incluse come colonne supplementari dell'indice di join. Di con-

sequenza, se l'interrogazione che utilizza l'indice di join richiede solo le colonne incluse nell'indice, le tabelle della base di dati non saranno necessarie per elaborare le interrogazioni. Quando può avere senso implementare questa estensione? La risposta è semplice: quando la base di dati è abbastanza statica, così che l'indice di join, una volta creato, abbia una vita piuttosto lunga. Questa condizione è vera in molti ambienti di data warehouse, nei quali le righe esistenti non sono aggiornate, vengono aggiunte nuove righe e le righe obsolete vengono eliminate solo periodicamente, per esempio una volta al mese. In questo caso, quando ci sono modifiche poco frequenti alla base di dati ma anche interrogazioni frequenti e complesse su basi di dati molto grandi, un indice di join esteso può essere una struttura molto produttiva per il miglioramento delle prestazioni delle interrogazioni.

L'uso di basi di dati per le nuove applicazioni, come nei data warehouse e nel supporto alle decisioni on-line, sta comportando lo sviluppo di nuovi tipi di indici. È consigliabile indagare sulle potenzialità di indicizzazione del sistema di gestione che si utilizza per poter comprendere appieno quando applicare ciascun tipo di indice e come perfezionare le prestazioni delle strutture degli indici.

Organizzazioni di file di hashing

In un'organizzazione di file di hashing, l'indirizzo di ogni record è determinato utilizzando un algoritmo di hashing (Figura 8.7c). Un algoritmo di hashing è una routine che converte un valore di chiave primaria in un indirizzo di record. Sebbene ci siano numerose varianti di file di hashing, nella maggior parte dei casi i record vengono trovati in modo non sequenziale come dettato dall'algoritmo di hashing. L'elaborazione di dati sequenziali non è quindi pratica.

Un tipico algoritmo di hashing utilizza la tecnica di dividere ogni valore di chiave primaria in base a un numero primo opportuno e poi di utilizzare il resto della divisione come posizione di memoria relativa. Per esempio, si supponga che un'azienda abbia un insieme di 1000 record di impiegati da memorizzare su un disco magnetico. Un numero primo adatto potrebbe essere 997, che è vicino a 1000. Si consideri ora il record per l'impiegato 12396. Quando si divide il numero per 997, si ha un resto di 432. Questo record viene quindi memorizzato nella posizione 432 nel file. Un'altra tecnica (non trattata) deve essere utilizzata per risolvere i duplicati (o l'overflow) che possono generarsi con il metodo divisione/resto quando due o più chiavi ordinano lo stesso indirizzo (*clash di hashing*).

Uno dei numerosi limiti dell'hashing risiede nel fatto che, poiché le posizioni delle righe delle tabelle sono dettate dall'algoritmo di hashing, è possibile utilizzare solo una chiave per il recupero (e la memorizzazione) dei dati in base all'hashing. Per superare questo limite, l'hashing e l'indicizzazione possono essere uniti nella cosiddetta tabella di indice hash. Una tabella di indice hash utilizza l'hashing per mappare una chiave in una posizione in un indice (chiamato a volte *tabella di indice a dispersione*), dove c'è un puntatore al record dei dati effettivi che corrispondono alla chiave hash. L'indice è la destinazione dell'algoritmo di hashing, ma i dati effettivi vengono memorizzati separatamente dagli indirizzi generati dall'hashing. A causa dei risultati dell'hashing in una posizione di indice, le righe della tabella possono essere memorizzate indipendentemente dall'indirizzo di hashing, utilizzando qualsiasi organizzazione di file che abbia senso per la tabella di dati (per esempio, quella sequenziale o il primo spazio disponibile). Come negli altri schemi di indicizzazione ma a differenza degli schemi di hashing più puri, possono esserci quindi numerose chiavi primarie e secondarie, ognuna con il proprio algoritmo di hashing e la propria tabella di indice, che condividono una tabella di dati. Inoltre, poiché una tabella di indice è molto più piccola di una tabella di dati, l'indice può essere progettato più facilmente per ridurre la probabilità di conflitti, o overflow, tra le chiavi, di quanto non sia possibile fare con una tabella di dati. Lo spazio di memoria supplementare per l'indice aggiunge flessibilità e velocità al recupero dei dati, di pari passo con dei costi aggiuntivi per la memorizzazione e la manutenzione dello spazio dell'indice. Un altro uso della tabella di indice hash si trova in alcune tecnologie di data warehouse che utilizzano l'elaborazione parallela. In questa situazione, il DBMS può distribuire regolarmente le righe

della tabella di dati tra tutti i dispositivi di memorizzazione utilizzando l'hashing e l'indicizzazione per trovare rapidamente il processore sul quale i dati sono memorizzati.

Un aspetto da non trascurare con tutte le tecniche di indicizzazione è la difficoltà o il costo dello spostamento dei dati una volta che una riga di tabella è stata memorizzata. Se una riga deve essere spostata, anche il puntatore a essa in tutti gli indici sulla tabella dovrà essere aggiornato. Questo può comportare costi notevoli qualora i dati debbano essere aggiornati di frequente (può accadere per esempio nel caso di un'eliminazione eccessiva di righe e nella creazione di spazi vuoti, o buchi, al centro del file fisico riservato alle righe della tabella di dati). D'altra parte, l'hashing puro non permette lo spostamento dei dati se non come parte della gestione degli overflow di hashing. Così, una tabella di indice hash fornisce i vantaggi di un recupero molto veloce dei dati in base ai valori di chiave e la capacità di spostare i dati, con gli svantaggi di costi per l'aggiornamento degli indici, se la riorganizzazione dello spazio dei dati diventa importante.

Come detto in precedenza, il DBMS gestisce la gestione di qualsiasi organizzazione di file di hashing. Non ci si deve quindi preoccupare degli overflow di gestione, dell'accesso agli indici o dell'algoritmo di hashing. Ciò che è importante per un progettista di base di dati è capire le proprietà delle diverse organizzazioni di file, così da poter scegliere quella più adatta al tipo di elaborazione necessaria nella base di dati e nell'applicazione che si sta progettando. La comprensione delle proprietà delle organizzazioni di file utilizzata dal DBMS può aiutare i programmatori a scrivere interrogazioni che traggano il massimo dalle proprietà dell'organizzazione dei file. Come si vedrà nei Capitoli 9 e 10, molte interrogazioni possono essere scritte in più modi utilizzando SQL; strutture di interrogazione molto diverse possono tuttavia comportare passaggi altrettanto diversi per il DBMS che risponde all'interrogazione. Se è noto come opera il DBMS sull'uso delle organizzazioni di file (per esempio, quali indici utilizza quando e come, e quando utilizza un algoritmo di hashing), si potranno progettare basi di dati migliori e interrogazioni più efficienti.

8.4.3 Riepilogo delle organizzazioni di file

Le tre categorie di organizzazioni di file trattate coprono la maggior parte di quelle che si hanno a disposizione quando si progettano basi di dati e file fisici. Sebbene si possano costruire strutture più complesse utilizzando le strutture di dati delineate nell'Appendice C, è improbabile che si sia in grado di utilizzarle con un sistema di gestione delle basi di dati.

La Tabella 8.3 precedentemente riportata riassume le funzioni di confronto delle organizzazioni di file sequenziali, indicizzate e di hashing. Si studi questa tabella e la Figura 8.7 per vedere un'applicazione e una conferma di ciascuna delle funzioni di confronto.

8.4.4 Accorpamento dei file

Alcuni sistemi di gestione delle basi di dati permettono la presenza di uno spazio contiguo nella memoria secondaria per poter contenere righe di più tabelle. In questo caso, un file fisico non contiene record con strutture identiche. Per esempio, in Oracle, le righe di una, due o più tabelle correlate che vengono spesso unite tra loro possono essere memorizzate nella stessa area del disco. Un cluster viene definito dalle tabelle e dalla colonna o colonne in base alle quali le tabelle sono solitamente unite. Per esempio, una tabella Customer e una tabella Customer_Order verrebbero unite dal valore comune di Customer_ID; le righe di una tabella Price_Quote (che contiene prezzi sugli articoli acquistati dai produttori) potrebbero invece essere raggruppate in base a valori comuni di Item_ID. L'accorpamento riduce il tempo di accesso ai record correlati rispetto all'allocazione normale di file diversi su aree diverse di un disco.

Il tempo è ridotto perché i record correlati sono molto più vicini tra loro che non se fossero memorizzati in file separati su aree separate del disco. Definire una tabella in un solo cluster riduce le prestazioni di recupero solo per quelle tabelle memorizzate nello stesso cluster.

I comandi di definizione di una base di dati Oracle che seguono mostrano come un cluster viene definito e quali sono le tabelle ad esso assegnate. Innanzitutto viene specificato il cluster (spazio contiguo su disco), come nel seguente esempio:

```
CREATE CLUSTER ORDERING (CLUSTERKEY CHAR(25));
```

Il termine ORDERING denomina lo spazio del cluster; il termine CLUSTERKEY è necessario ma non viene ancora utilizzato.

Come secondo passo, le tabelle vengono assegnate al cluster quando sono create:

```
CREATE TABLE CUSTOMER (
  CUSTOMER_ID          VARCHAR2(25)  NOT NULL,
  CUSTOMER_ADDRESS     VARCHAR(15)
)
CLUSTER ORDERING (CUSTOMER_ID);

CREATE TABLE ORDER (
  ORDER_ID              VARCHAR2(20)  NOT NULL,
  CUSTOMER_ID           VARCHAR2(25)   NOT NULL,
  ORDER_DATE            DATE
)
CLUSTER ORDERING (CUSTOMER_ID);
```

L'accesso ai record in un cluster può essere specificato in Oracle come effettuabile attraverso un indice sulla chiave di cluster o attraverso una funzione di hashing sulla chiave di cluster. Le ragioni per scegliere un indice piuttosto che un cluster hash sono simili a quelle per la scelta tra file indicizzati e hash (Tabella 8.3). L'accorpamento dei record viene utilizzato al meglio quando i record sono tendenzialmente statici; quando i record vengono aggiunti, eliminati e modificati di frequente, si può creare dello spazio vuoto e può diventare difficile trovare record correlati vicini tra loro dopo il loro caricamento iniziale, che definisce i cluster. L'accorpamento è tuttavia una possibilità per il progettista dei file di migliorare le prestazioni delle tabelle che sono spesso utilizzate insieme nelle stesse interrogazioni e negli stessi report.

8.4.5 Progettare il controllo dei file

Un aspetto supplementare di un file di base di dati sul quale si hanno varie opzioni di progettazione riguarda i tipi di controlli che si possono utilizzare per proteggere il file dalla distruzione o dal danneggiamento o per ricostruirlo qualora venga danneggiato. Poiché un file di base di dati viene memorizzato in un formato proprietario dal DBMS, c'è un livello di controllo dell'accesso di base. Si possono però richiedere altri controlli di protezione sui campi, i file o le basi di dati. Queste opzioni verranno esaminate nel dettaglio nel Capitolo 10. In breve, si vedrà come i file possano essere danneggiati e come sia importante avere la possibilità di ripristinarli rapidamente. Le procedure di backup forniscono una copia del file e delle transazioni che lo hanno modificato. Quando un file viene danneggiato, la copia del file o il file corrente con il registro delle transazioni vengono utilizzati per riportare il file a uno stato di integrità. In termini di sicurezza, il metodo più efficace è quello di codificare i contenuti del file; in questo modo, solo i programmi con un accesso alla routine di decodifica sono in grado di vedere il contenuto dei file. Questi argomenti importanti verranno trattati successivamente quando si studierà il linguaggio SQL di manipolazione dei dati nel Capitolo 10.

8.5 Utilizzare e scegliere gli indici

Gran parte delle manipolazioni delle basi di dati richiede di trovare una riga, o un insieme di righe, che soddisfino una determinata condizione. Per esempio, si potrebbero voler recuperare tutti i clienti con un dato codice di avviamento postale o tutti gli studenti con una particolare specializzazione. L'analisi di ogni riga in una tabella alla ricerca delle righe desiderate può essere inaccettabilmente lenta, soprattutto quando le tabelle sono molto grandi, come accade spesso nelle applicazioni del mondo reale. L'uso degli indici, descritti in precedenza, può velocizzare notevolmente questo processo e la definizione degli indici è una parte importante della progettazione fisica di una base di dati.

Come descritto nel paragrafo precedente, gli indici su un file possono essere creati per una chiave primaria, per una secondaria o per entrambe. Solitamente viene creato un indice per la chiave primaria di ogni tabella. L'indice è anch'esso una tabella con due colonne: la chiave e l'indirizzo del record o dei record che contengono il valore di chiave. Nel caso di una chiave primaria, ci sarà una sola immissione nell'indice per ogni valore di chiave.

8.5.1 Creare un indice di chiave univoca

La tabella Customer definita nel paragrafo sull'accorpamento ha una chiave primaria di Customer_ID. Un indice di chiave univoca potrebbe essere creato su questo campo utilizzando il seguente comando SQL:

```
CREATE UNIQUE INDEX CUSTINDEX ON CUSTOMER(CUSTOMER_ID);
```

In questo comando, CUSTINDEX è il nome del file di indice creato per memorizzare le immissioni dell'indice. La clausola ON specifica quale tabella viene indicizzata e la colonna (o le colonne) che forma la chiave di indice. Se ci sono valori duplicati di Customer_ID, il comando CREATE INDEX fallisce. Una volta che un indice viene creato, il DBMS rifiuta tutti gli inserimenti o aggiornamenti di dati nella tabella CUSTOMER che potrebbero violare il vincolo di univocità su Customer_ID.

Quando è presente una chiave univoca composta, si elencano semplicemente tutti gli elementi della chiave univoca nella clausola ON. Per esempio, una tabella di articoli di una linea su un ordine di cliente potrebbe avere una chiave univoca composta da Order_ID e Product_ID. Il comando SQL per creare questo indice per la tabella Order_Line è il seguente:

```
CREATE UNIQUE INDEX LINEINDEX ON ORDER_LINE(ORDER_ID,PRODUCT_ID);
```

8.5.2 Creare un indice di chiave secondaria (non univoca)

Gli utenti delle basi di dati desiderano spesso recuperare le righe di una relazione in base ai valori dei vari attributi piuttosto che in base alla chiave primaria. Per esempio, in una tabella Product, gli utenti potrebbero voler recuperare i record che soddisfano una combinazione qualsiasi tra le seguenti.

- Tutti i prodotti nella tabella (Description = 'Table')
- Tutti i mobili in quercia (Finish = 'Oak')
- Tutti i mobili per la sala da pranzo (Room = 'DR')
- Tutti i mobili di costo inferiore ai 500 \$ (Price<500)

Per velocizzare tali recuperi, si può definire un indice su ciascun attributo da utilizzare per qualificare un recupero. Per esempio, si potrebbe creare un indice non univoco sul campo Description della tabella Product con il seguente comando SQL:

```
CREATE INDEX DESCINDX ON PRODUCT(DESCRIPTION);
```

Si noti che il termine UNIQUE non dovrebbe essere utilizzato con gli attributi di chiave secondaria (non univoca), poiché ogni valore dell'attributo può essere ripetuto. Come con le chiavi univoche, un indice di chiave univoca può essere creato su una combinazione di attributi.

Per creare un indice bitmap, si segue una struttura di comando analoga. Per esempio, se si volesse un indice bitmap per il campo Description, il comando dovrebbe essere il seguente:

```
CREATE BITMAP DESCBITIDX ON PRODUCT(DESCRIPTION);
```

8.5.3 Quando utilizzare gli indici

Durante la progettazione fisica di una base di dati, occorre scegliere quali attributi utilizzare per creare gli indici. Esiste un compromesso tra le prestazioni migliorate nel recupero dei dati attraverso gli indici e le prestazioni degradate nell'inserimento, l'eliminazione e l'aggiornamento dei record in un file. Gli indici dovrebbero essere ampiamente utilizzati per le basi di dati concepite principalmente per il supporto del recupero dei dati, come nel caso delle applicazioni di supporto alle decisioni o di data warehouse. Gli indici dovrebbero essere utilizzati con parsimonia nelle basi di dati che supportano l'elaborazione delle transazioni e di altre applicazioni con esigenze di aggiornamento notevoli, poiché impongono un costo supplementare.

Quelle che seguono sono alcune linee guida per la scelta degli indici per le basi di dati relazionali.

1. Gli indici sono più utili sulle tabelle grandi.
2. Specificare un indice univoco per la chiave primaria di ogni tabella.
3. Gli indici sono più utili per le colonne che appaiono spesso nelle clausole WHERE dei comandi SQL per qualificare le righe da selezionare (per esempio, WHERE FINISH = 'Oak', nel quale un indice su Finish velocizzerebbe il recupero) o per il collegamento, mediante join, delle tabelle (per esempio, WHERE PRODUCT.PRODUCT_ID = ORDER_LINE.PRODUCT_ID, nel quale un indice di chiave secondaria su Product_ID nella tabella Order_Line e un indice di chiave primaria su Product_ID nella tabella Product migliorerebbero le prestazioni di recupero). Nel secondo caso, l'indice è su una chiave esterna nella tabella Order_Line che viene utilizzata per unire le tabelle.
4. Utilizzare un indice per gli attributi a cui si fa riferimento nelle clausole ORDER BY (ordinamento) e GROUP BY (classificazione). Occorre prestare molta attenzione a queste clausole. È infatti necessario accertarsi che il DBMS utilizzi gli indici sugli attributi elencati in queste clausole (per esempio, Oracle utilizza gli indici sugli attributi nelle clausole ORDER BY ma non in quelle GROUP BY).
5. Utilizzare un indice quando c'è una varietà significativa dei valori di un attributo. Oracle suggerisce che un indice non è utile quando ci sono meno di 30 valori diversi per un attributo, mentre un indice è chiaramente utile quando ci sono 100 o più valori diversi per un attributo. Analogamente, un indice è utile solo se i risultati di un'interrogazione che utilizza quell'indice non superano approssimativamente il 20 per cento del numero totale di record nel file (Schumacher, 1997).
6. Verificare nel DBMS eventuali limiti al numero degli indici consentiti per tabella. Molti sistemi non permettono più di 16 indici e possono limitare la dimensione di un valore di chiave di indice (per esempio, non più di 2000 byte per ogni valore composto). Se il proprio sistema prevede tale limite, si

dovranno scegliere quelle chiavi secondarie che probabilmente porteranno a un miglioramento delle prestazioni.

7. Occorre fare molta attenzione quando si indicizzano attributi che hanno valori nulli. Per molti DBMS, le righe con un valore nullo non verranno referenziate nell'indice (così che non possano essere trovate con un *indice di ricerca* di `ATTRIBUTE = NULL`). Una ricerca di questo tipo dovrà essere eseguita analizzando il file.

La selezione degli indici è probabilmente la decisione più importante da prendere quando si progetta una base di dati, ma non è il solo modo attraverso il quale si possono migliorare le prestazioni di una base di dati. Altri metodi risolvono problemi come la riduzione dei costi per ricollocare i record, l'ottimizzazione dell'uso dello spazio extra o cosiddetto libero nei file e l'ottimizzazione degli algoritmi di elaborazione delle interrogazioni (si veda Viehman, 1994, per una discussione di questi metodi alternativi per migliorare la progettazione e l'efficienza delle basi di dati fisiche). Tratteremo l'ottimizzazione delle interrogazioni in uno degli ultimi paragrafi di questo capitolo; l'ottimizzazione può essere utilizzata per invalidare il modo in cui il DBMS utilizza determinate opzioni di progettazione della base di dati, grazie al miglioramento che nella maggior dei casi queste apportano nelle prestazioni di elaborazione.

8.6 RAID: migliorare le prestazioni di accesso ai file tramite l'elaborazione parallela

Nei paragrafi precedenti di questo capitolo sono stati introdotti la denormalizzazione e l'accorpamento, due schemi per collocare i dati utilizzati insieme gli uni vicini agli altri nella memoria su disco. La denormalizzazione e l'accorpamento sono ideali per ridurre il tempo di accesso ai dati se i dati utilizzati insieme in un programma possono essere trovati all'interno di una pagina di dati fisica. Nel caso di record piccoli, questo avviene senza problemi. Tuttavia, i dati utilizzati insieme o uno subito dopo l'altro possono essere stati memorizzati su pagine separate; in questo caso, saranno necessarie più operazioni consecutive di input/output su disco per leggere e scrivere i dati. La denormalizzazione e l'accorpamento possono non essere schemi sufficienti per migliorare in modo significativo le prestazioni di lettura e scrittura.

Due delle tendenze inarrestabili nella tecnologia informatica sono la riduzione costante dei costi e delle dimensioni delle tecnologie. Queste due tendenze rendono la ridondanza dei componenti dei computer, e quindi la tolleranza degli errori, economicamente e fisicamente accettabili. La possibilità di utilizzare più componenti di computer piccoli consente anche l'elaborazione parallela dei dati. Il risultato dell'elaborazione parallela dei dati è che, per esempio, se si effettuano operazioni di input/output in parallelo, è come se si effettuasse un'operazione sola. Queste caratteristiche possono essere decisive per il successo di una base di dati molto grande in un'applicazione on-line, per esempio una di commercio elettronico.

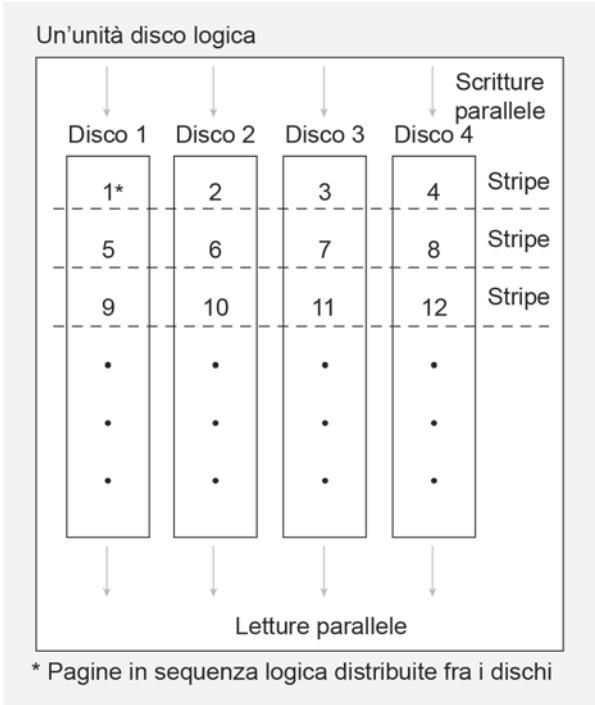
L'elaborazione parallela di una base di dati e la tolleranza degli errori possono essere ottenute da un progettista di basi di dati utilizzando una tecnologia hardware o software chiamata ***Redundant Array of Inexpensive Disks (RAID)***. La memorizzazione RAID utilizza un insieme, o array, di unità disco fisiche che appaiono all'utente e ai programmi della base di dati come se formassero un'unica grande unità logica di memoria. RAID non modifica quindi la struttura logica o fisica dei programmi applicativi o delle interrogazioni di base di dati.

Per ottimizzare le prestazioni di input/output di RAID, tutte le unità disco devono essere mantenute occupate. Lo *striping* permette di distribuire in modo equilibrato il carico di lavoro tra le unità disco. I segmenti di dati, chiamati *stripe* (*strisce*), si distribuiscono tra tutte le unità disco, come illustrato nella

Figura 8.10. Pagine di dati logicamente sequenziali (per esempio, più pagine per un record, un cluster di record o più record logicamente sequenziali tratti da un file) sono memorizzate in modo circolare tra le unità disco fisiche. Così, per la memorizzazione RAID ipotetica illustrata nella Figura 8.10, sarebbe possibile leggere in parallelo quattro pagine di dati logicamente sequenziali nel tempo necessario a leggere una sola pagina. Un record di dati molto lungo (che contiene per esempio un campo audio o video) che è memorizzato su più pagine, può essere letto in una frazione del tempo che sarebbe necessario senza RAID. Oppure, più record correlati (un cluster) che si estendono su più pagine possono essere letti tutti contemporaneamente.

Oggigiorno, la maggior parte dei sistemi operativi multiutente, compresi Microsoft NT, UNIX e

Figura 8.10
RAID con quattro dischi e striping.



Novell NetWare, supporta alcune forme di RAID. Tali sistemi operativi avviano operazioni di lettura parallele e concatenano i risultati tra loro in un record logico per il programma richiedente. Inoltre permettono il *multithreading*, nel quale le operazioni di input/output parallele provengono da diversi programmi o attività utente. RAID può anche essere implementato nell'hardware utilizzando adattatori speciali, sollevando così il computer host dal lavoro di suddivisione delle operazioni di input e output in passaggi paralleli.

RAID presenta un rischio da non sottovalutare, ovvero l'aumento della probabilità del fallimento di un'unità disco nell'intera base di dati. Se il tempo medio tra i fallimenti su una singola unità disco è, per esempio, di 1,2 milioni, allora il tempo medio tra fallimenti di un RAID con, per esempio, quattro unità disco, sarà di 1,2 milioni divisi per 4, o di ogni 300.000 operazioni su disco parallele. Per gestire questo rischio e rendere la memorizzazione su disco tollerante agli errori, molti tipi di tecnologie RAID

memorizzano i dati in modo ridondante, così che almeno una copia dei dati sia accessibile quando richiesto, o memorizzano i codici di correzione degli errori più comuni così che quelli danneggiati o persi possano essere ricostruiti.

8.6.1 Scegliere tra i livelli di RAID

In quanto progettisti di file e basi di dati, si avranno probabilmente sei o più tipi di architetture RAID ridondanti o di correzione degli errori tra le quali scegliere. Anche RAID-0, che non utilizza una memorizzazione ridondante, viene comunque utilizzato, poiché permette le prestazioni più veloci utilizzando il grado di parallelismo più elevato. La Figura 8.11 confronta sei forme di RAID ed elenca alcune funzioni chiave di ogni forma. Unità disco specifiche e sistemi operativi possono fornire alternative supplementari.

RAID-0

I tempi più veloci di lettura/scrittura e una memorizzazione più veloce dei dati di qualsiasi tipo di array di disco si ottengono con RAID-0 (Figura 8.11a). In RAID-0, chiamato anche *striping di disco*, ogni disco nell'array è utilizzato in parallelo per leggere e scrivere i dati delle applicazioni. RAID-0 viene utilizzato solitamente quando l'applicazione di base di dati accede a record molto grandi (richiedendo quindi gran parte di una o più stripe) o a un gruppo di record correlati o quando è sottoposta a un input/output molto intenso. Poiché RAID-0 non utilizza alcuna forma di ridondanza dei dati o di codici di correzione degli errori nelle stripe, un fallimento in una qualsiasi delle unità disco può danneggiare la base di dati. Inoltre RAID-0 permette che venga elaborato un solo comando I/O alla volta (e quindi non I/O multithread). Il collo di bottiglia in RAID-0 è il controller del disco, che può risultare saturato da numerose operazioni di I/O parallele. Idealmente, possono essere utilizzati controlli multipli tra le varie unità.

RAID-1

Un modo per migliorare la tolleranza degli errori è quello di memorizzare i dati delle applicazioni in modo ridondante. Una memorizzazione completamente ridondante, o mirror di disco, si ottiene con RAID-1 (Figura 8.11b), nel quale ciascuna pagina viene scritta in modo ridondante. Sebbene lo spazio di memoria totale del disco venga effettivamente ridotto della metà, la base di dati ha un alto grado di tolleranza degli errori, poiché è improbabile che esattamente la stessa pagina venga danneggiata in entrambe le copie. Un singolo array RAID-1 di due unità disco non utilizza lo striping, ma è possibile unire più array RAID-1 duali di disco per formare un array più grande che utilizzi lo striping (si veda RAID 0+1 di seguito). In RAID-1, ogni operazione di scrittura della pagina viene eseguita in parallelo su una coppia di unità disco, e questo non permette alcun miglioramento nelle prestazioni. Due operazioni di lettura, anche se generate da interrogazioni di base di dati separate in programmi utente diversi (multithread), possono essere eseguite in parallelo per ciascun array duale. RAID-1 offre quindi sia prestazioni sia una tolleranza degli errori eccellenti in ambienti a utente singolo o multiutente. La diminuzione dei prezzi dei dischi ha reso RAID-1 la forma più diffusa di RAID attualmente in uso.

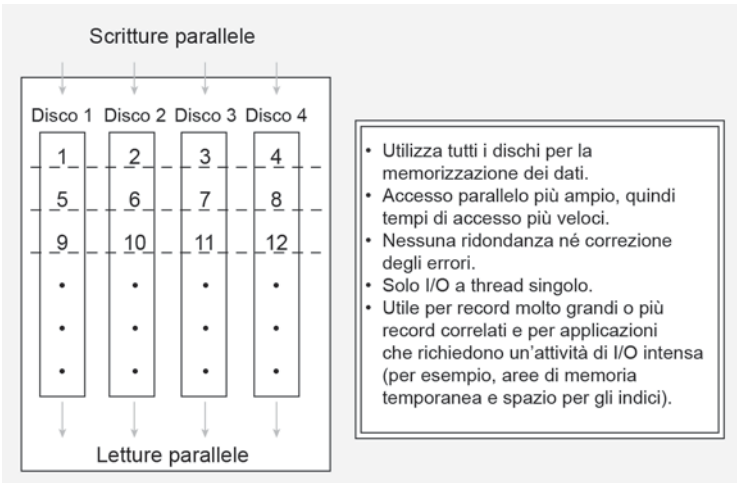
RAID 0+1

Come già visto, RAID-0 enfatizza le alte prestazioni attraverso letture e scritture parallele là dove RAID-1 enfatizza la tolleranza degli errori attraverso una memorizzazione ridondante. Impostando una configurazione RAID-0 e creandone quindi un mirror, si ottiene il massimo sia da RAID-0 sia da RAID-1. Questa configurazione è chiamata RAID 0+1 o RAID-10 (Compaq) o ancora RAID-6 (Hew-

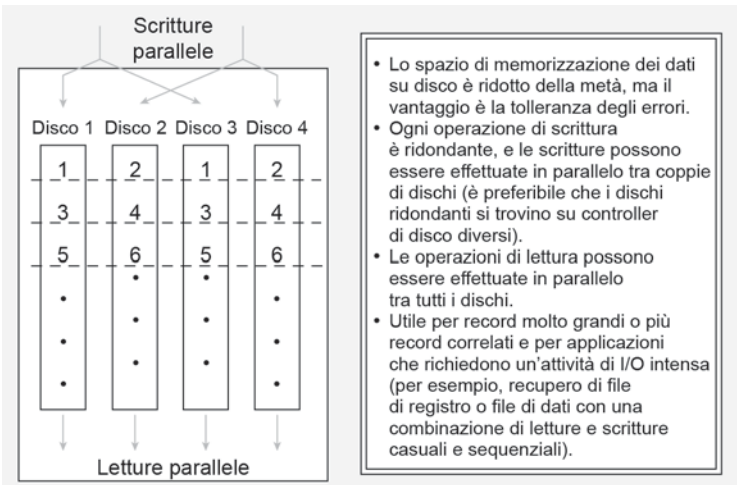
Figura 8.11

Sei livelli di RAID. (Nota: i numeri sono numeri di blocchi di dati sequenziali. Le lettere indicano i segmenti di un blocco di dati.)

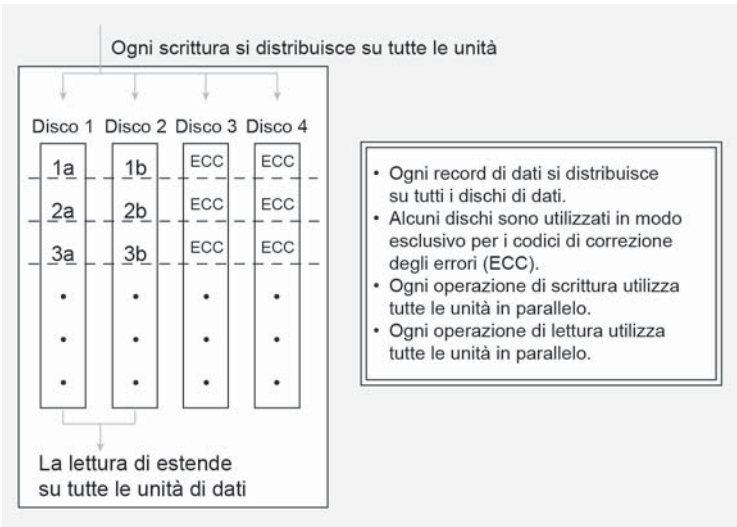
(a) RAID-0 (striping).



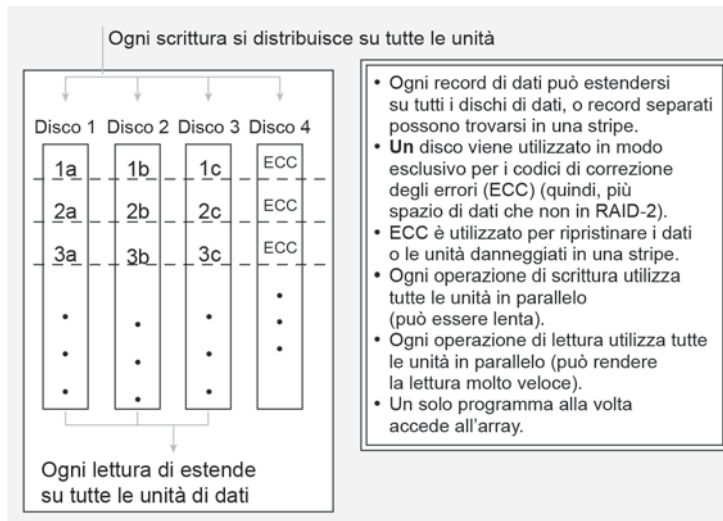
(b) RAID-1 con due coppie di dischi mirror.



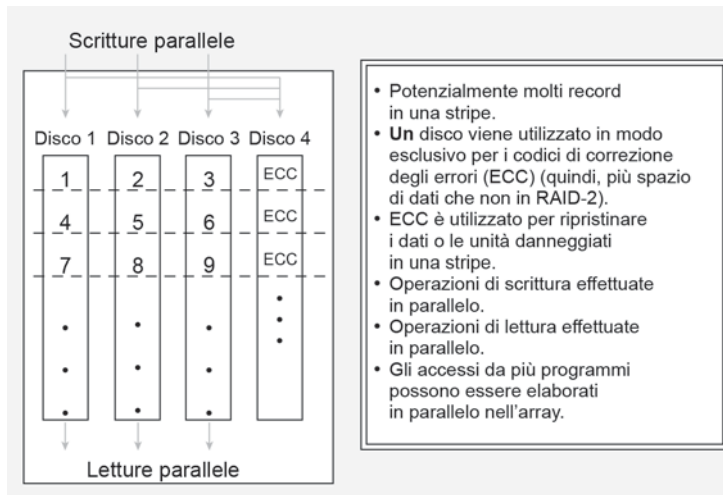
(c) RAID-2 con metà delle unità utilizzate per i dati.



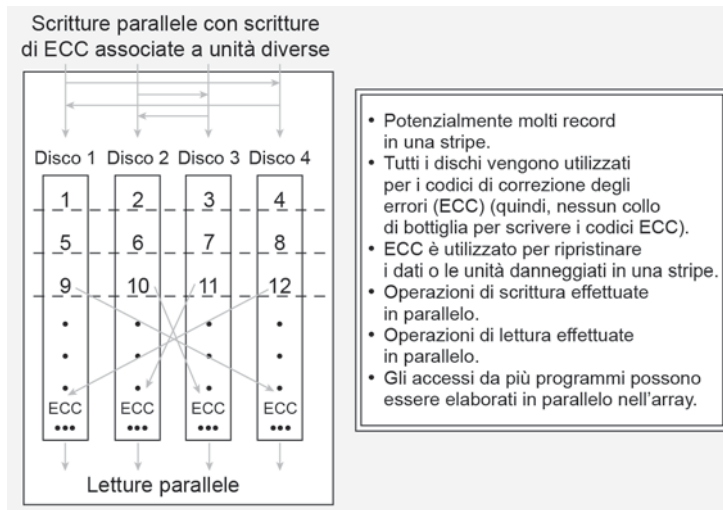
(d) RAID-3.



(e) RAID-4.



(f) RAID-5.



lett Packard). (RAID 1+0 non è mostrato nella Figura 8.11.) RAID 1+0 fornisce la memorizzazione di grandi volumi di dati, un'alta affidabilità e una buona protezione dai fallimenti dell'unità (Musciano, 1999). Tuttavia, è molto costoso, e per questo viene utilizzato in ambienti che richiedono percentuali di uptime molto elevate con risposte molto rapide, come accade nel caso di applicazioni in tempo reale per il comando e il controllo militare, il controllo del traffico aereo, il controllo dei processi delle utility e alcune situazioni di commercio elettronico (per esempio gli investimenti on-line).

RAID 2

In RAID-2 (Figura 8.11c), alcuni dischi sono utilizzati con lo striping dei dati per l'elaborazione parallela dei dati (senza memorizzazione ridondante), mentre altri sono riservati alla memorizzazione dei codici di correzione degli errori. Questi codici vengono utilizzati per individuare gli errori nelle stripe e per ricostruire i dati quando le pagine nelle stripe di dati risultano danneggiate. Laddove RAID-1 utilizza la ridondanza per permettere il recupero dei dati danneggiati, RAID-2, e i livelli successivi, utilizzano codici di correzione, o parità, degli errori più efficienti. Tuttavia, poiché la maggior parte delle unità disco dei computer moderni memorizza già codici di correzione o informazioni di parità a livello di pagina, in ciascuna pagina fisica è preferibile utilizzare RAID-3 piuttosto che RAID-2.

RAID 3

Come RAID-2, anche RAID-3 distribuisce i dati tra le unità, ma solo un'unità memorizza i codici di correzione a livello di stripe (Figura 8.11d). I codici di correzione incorporati in ogni pagina vengono utilizzati per individuare gli errori, e la pagina ECC viene utilizzata per ripristinare una pagina danneggiata. Si immagini che l'unità 3 nella Figura 8.11 vada in crash. La pagina 3 in ciascuna stripe può essere recuperata dalle altre pagine di dati e dalla pagina di correzione degli errori nella sua stripe mediante un calcolo OR esclusivo all'interno della stripe. Perché questo sia possibile, un aggiornamento su una pagina in una stripe richiede che vengano lette tutte le pagine nella stripe, così che la pagina ECC possa essere ricalcolata e riscritta. RAID-3 viene utilizzato solitamente quando ci sono record molto lunghi, che richiedono cioè una o più stripe. Di conseguenza, ogni operazione logica di input/output utilizza (o si estende) tutte le unità nell'array; non è quindi possibile elaborare richieste in parallelo da più utenti o programmi. RAID-3 funziona dunque al meglio in ambienti a utente singolo e a programmazione unica, con un volume di attività di lettura alto e uno di scrittura basso. Lo spazio di overhead di RAID-3 diminuisce con l'aumentare del numero dei dischi in una strip. L'unità ECC diventa un collo di bottiglia.

RAID-4

In RAID-4, i codici di correzione degli errori a livello di stripe sono memorizzati in un'unica unità, come in RAID-3, ma in una stripe sono memorizzati più record (Figura 8.11e). Le operazioni di lettura su record diversi dello stesso programma o di programmi diversi possono quindi essere eseguite in parallelo tra le unità. Il collo di bottiglia è l'unità di parità, poiché deve essere possibile accedervi per aggiornare una pagina di correzione degli errori quando un record qualsiasi viene riscritto. Nelle basi di dati con aggiornamenti frequenti dei record, come nelle elaborazioni delle transazioni on-line, si può creare un notevole conflitto (e quindi un ritardo) nell'unità di parità, riducendo i vantaggi dell'elaborazione parallela. Questo collo di bottiglia può essere evitato utilizzando RAID-5. *Nota:* alcuni autori non fanno distinzione tra RAID-3 e RAID-4.

RAID-5

Chiamato anche *Rotating Parity Array*, RAID-5 (Figura 8.11f) non utilizza un'unità di parità dedicata. Ogni unità contiene sia pagine di dati sia di parità. Le operazioni di lettura possono essere eseguite in

parallelo su tutte le unità. Un'operazione di scrittura accederà alla propria unità o a più unità per scrivere il nuovo record e l'unità di parità *per quel record*. Record diversi memorizzano i codici di correzione, per scopi di ripristino, su unità diverse. Le operazioni di scrittura possono quindi anche essere eseguite in parallelo con meno probabilità di ritardo rispetto a RAID-4, poiché gli aggiornamenti casuali dei record generano aggiornamenti della pagina dei codici di correzione distribuiti casualmente tra le unità.

8.6.2 Prestazioni di RAID

Come si è visto nei paragrafi precedenti, RAID-1, RAID-3 e RAID-5 dominano le altre forme di RAID tolleranti degli errori. Poiché RAID-3 è concepito per ambienti a utente singolo o con un'intensa attività sui dati, questa forma di RAID ha un'applicabilità specializzata. Nella maggior parte delle applicazioni di basi di dati, questo significa che molto probabilmente si opterà per RAID-1, RAID 0+1 o RAID-5.

Un produttore di RAID, Adaptec (<http://www.dpt.com>), ha messo a confronto le prestazioni di RAID-1 e di RAID-5 nelle operazioni sia di lettura sia di scrittura con diverse unità disco nell'array e in unità disco con frequenze di trasferimento diverse. Come ci si può aspettare, RAID-5 utilizza in modo più efficiente lo spazio di memoria, poiché non memorizza i dati in modo ridondante. RAID-5 supera RAID-1 nelle operazioni di lettura, poiché permette l'elaborazione parallela con lo stesso numero di unità disco nell'array. Tuttavia, i tempi di scrittura di RAID-5 possono essere da 1,67 a 3 volte superiori di quelli di RAID-1 (a seconda dell'applicazione benchmark e della configurazione dell'array). RAID-5 è quindi meno adatto di RAID-1 per i file molto volatili. Adaptec ha anche stimato che per rendere le operazioni di scrittura due volte più veloci come se si utilizzasse una sola unità, RAID-1 richiede un array di quattro unità disco, mentre RAID-5 ne richiede da sette a dodici.

Qual è dunque la forma migliore di RAID? RAID-1 è ideale per le applicazioni di manutenzione di basi di dati tolleranti agli errori (quelle che richiedono un'alta percentuale di uptime) o quando sono possibili solo due controller di unità disco. RAID-1 può essere costoso, ma è ideale per un'ampia gamma di carichi di lavoro. RAID-5 è adatto per le applicazioni ad alta intensità di lettura su insiemi di dati molto grandi quando sono possibili almeno tre (più tipicamente cinque) unità disco. RAID-5 diventa meno adatto man mano che le unità disco diventano più grandi.

I dispositivi di memorizzazione sono un ambito in continuo cambiamento. Le nuove tecnologie *Storage Area Networks* (SAN) e *Network-Attached Storage* (NAS) stanno emergendo per i grandi ambienti aziendali di memorizzazione (Shah, 1999).

8.7 Progettare una base di dati

La maggior parte dei sistemi informativi moderni utilizza tecnologie delle basi di dati, siano essi sistemi di gestione o di data warehouse, per la memorizzazione e il recupero dei dati. Si ricordi che una base di dati è una raccolta di dati logicamente correlati, progettati per soddisfare le esigenze informative di più utenti in un'azienda. I rapporti tra i file in una base di dati dipendono dalle associazioni identificate nei modelli di dati logico e concettuale. Le associazioni implicano percorsi di accesso tra i dati. Ogni tipo tecnologia delle basi di dati consente percorsi di accesso diversi. Il processo per scegliere il tipo di tecnologia di DBMS o di data warehouse adatta consiste quindi nel combinare i percorsi di accesso necessari con le potenzialità della tecnologia.

8.7.1 Scegliere l'architettura della base di dati

Esistono diversi stili di sistemi di gestione delle basi di dati e di data warehouse, ognuno caratterizzato dal modo in cui i dati sono definiti e strutturati, chiamati *architetture delle basi di dati*. La scelta dell'architettura più adatta alla propria base di dati è fondamentale per la progettazione di quest'ultima. Un determinato sistema di gestione della basi di dati o di data warehouse supporta una di cinque architetture diverse. La Figura 8.12 le mette a confronto.

1. *Modello di base di dati gerarchico*. In questo modello, i file sono disposti in una struttura dall'alto verso il basso che somiglia a un albero genealogico. I dati vengono correlati in un insieme di associazioni nidificate uno-a-molti. Il file al livello più alto si chiama *radice*, quelli al livello più basso si chiamano *foglie* e quelli intermedi hanno un file *genitore*, o possessore, e uno o più file *figlio*. Essendo una delle architetture più vecchie, si può trovare ancora oggi nelle aziende più grandi. Si applica al meglio quando anche il modello di dati concettuale somiglia a un albero e quando la maggior parte degli accessi ai dati inizia con lo stesso file radice. Questa tecnologia viene utilizzata per le applicazioni di elaborazione di grandi volumi di transazioni e MIS. Le basi di dati moderne che utilizzano questo modello sono poche, poiché le applicazioni nuove tendono ad avere necessità più ampie che non la semplice elaborazione delle transazioni o il riepilogo dei dati delle transazioni.
2. *Modello di base di dati di rete*. In questo modello, ogni file può essere associato a un numero arbitrario di file. Sebbene sia molto flessibile data la possibilità di implementazione di qualsiasi associazione (una gerarchia è un tipo speciale di rete), la sua forma di implementazione, che utilizza puntatori tra record correlati in file diversi, crea un overhead significativo nello spazio di memoria e nei tempi di manutenzione. Solitamente i sistemi di modello di rete supportano solo associazioni uno-a-molti su ogni arco della rete, ma alcuni supportano anche associazioni multi-a-molti. Questi sistemi sono ancora diffusi su mainframe potenti e nelle applicazioni di elaborazione di grandi volumi di transazioni. Poiché il progettista della base di dati ha un controllo molto elevato sulle organizzazioni dei dati, con questo modello è possibile progettare basi di dati altamente ottimizzate. Per esempio, ogni tipo di record (mostrato nella figura dai rettangoli) può essere organizzato utilizzando algoritmi di hashing o può essere collocato vicino a un altro tipo di record correlato (una forma primitiva di accorpamento). I sistemi di rete supportano un'ampia gamma di esigenze di elaborazione che non i sistemi gerarchici, ma richiedono una notevole programmazione e una grande conoscenza della progettazione delle basi di dati; per questo vengono utilizzati principalmente nelle aziende che hanno una notevole competenza su queste tecnologie.
3. *Modello di base di dati relazionale*. Il modello di base di dati più comune per i nuovi sistemi definisce tabelle semplici per ciascuna relazione e associazioni multi-a-molti. Chiavi a riferimento incrociato collegano le tabelle tra loro, rappresentando le associazioni tra le entità. Gli indici di chiave primaria e secondaria forniscono un rapido accesso ai dati in base alle qualifiche. La maggior parte delle nuove applicazioni viene costruita utilizzando DBMS relazionali ed esistono numerosi prodotti di DBMS. Si vedano il Capitolo 7 per una panoramica delle caratteristiche del modello relazionale, il Capitolo 11 per un'introduzione a Query-by-Example, uno stile di linguaggio di interrogazione relazionale, e i Capitoli 9 e 10 per una trattazione di SQL, linguaggio di manipolazione dei dati relazionali più diffuso.
4. *Modello di base di dati orientato agli oggetti*. In questo modello, gli attributi e i metodi che operano su questi attributi sono incapsulati in strutture chiamate *classi di oggetto*. Le associazioni tra le classi di oggetto sono mostrate, in parte, nidificando o incapsulando un oggetto di classe in un altro. Le nuove classi di oggetto vengono definite da classi di oggetto più generali. Un vantaggio principale di

questo modello di dati è che i tipi di dati complessi, come immagini, video e audio, sono supportati facilmente come se fossero tipi di dati più semplici. Si tratta della tecnologia DBMS più recente e le aziende più grandi stanno prendendo sempre più familiarità con essa, utilizzandola selettivamente quando i dati complessi o una programmazione guidata dagli eventi sono opportuni per l'applicazione.

5. *Modello di base di dati multidimensionale.* Questo modello di base di dati è utilizzato nelle applicazioni di data warehouse. Esistono due modi per visualizzarlo. Il primo visualizza i dati come un cubo a più dimensioni nel quale ogni cella contiene uno o più attributi semplici e le dimensioni sono un modo per classificare i dati grezzi. Queste categorie, o dimensioni, sono i fattori sui quali gli utenti desiderano riepilogare o segmentare i dati, come periodi di tempo, aree geografiche, linee aziendali o persone. Una cella contiene i dati importanti all'intersezione di tutti i suoi valori di dimensione. Per esempio, una cella potrebbe contenere l'attributo numero-di-unità-vendute per un periodo di tempo, un luogo, una linea aziendale e un agente di vendita specifici. La seconda vista (equivalente) è chiamata *schema a stella*. Al centro dello schema si trova la tabella dei fatti, equivalente alla cella della vista multidimensionale. Le tabelle di dimensione circostanti definiscono ciascuno dei modi in cui si definiscono i dati in categorie, come tutti i dati di descrizione per ogni agente di vendita. Il problema di strutturazione principale del modello di base di dati multidimensionale è quello di identificare in anticipo il denominatore comune di dati più basso e quindi tutte le dimensioni e le categorie in base alle quali gli utenti vogliono regolare i fatti grezzi. Sebbene questo tipo di modello sia un caso speciale dei modelli di dati relazionale e di rete, sono stati sviluppati alcuni DBMS speciali che ottimizzano l'elaborazione dei dati multidimensionali. Il Capitolo 12 tratta i data warehouse e il modello di dati multidimensionale.

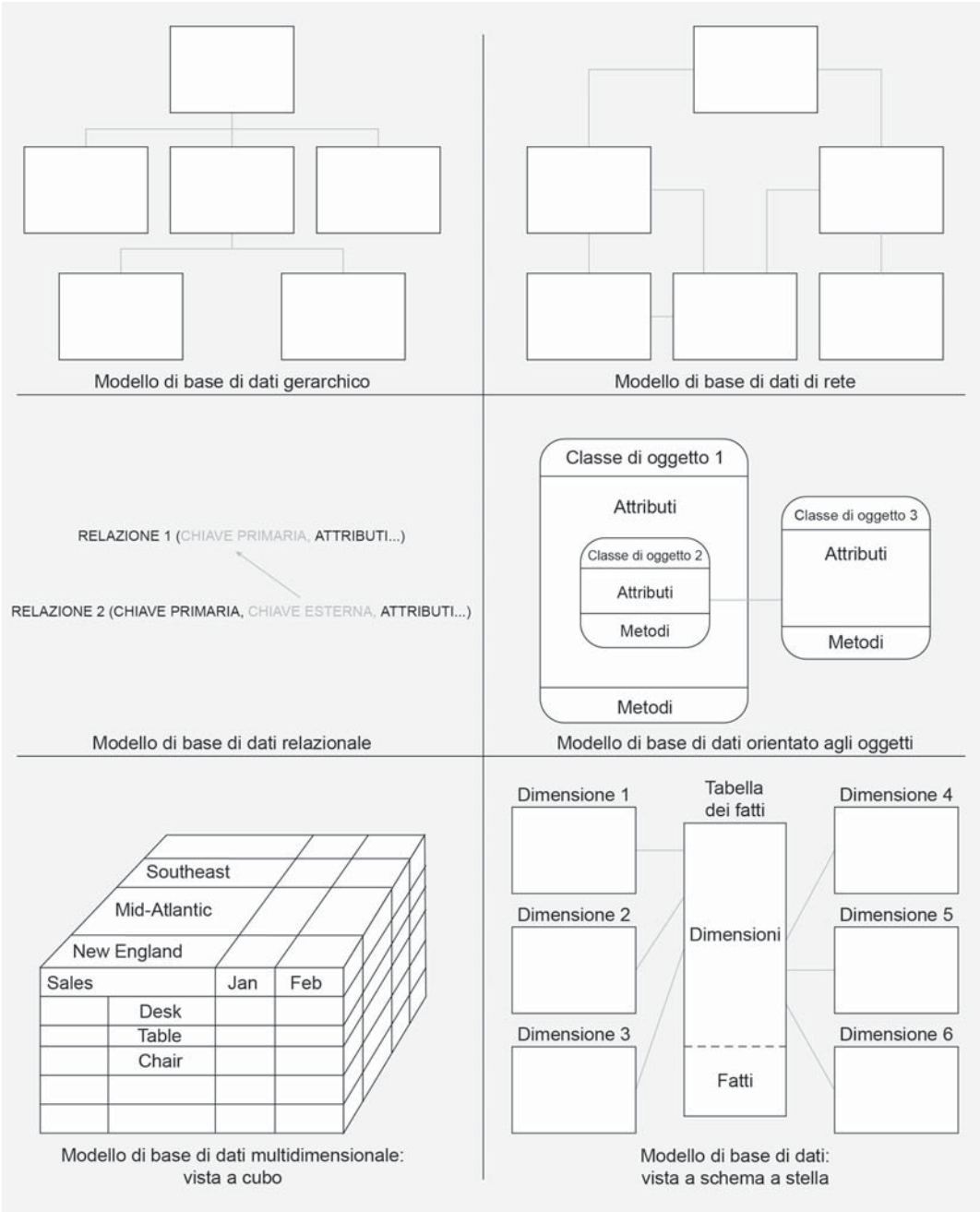
In questo libro, si è scelto di trattare nel dettaglio solo i tre modelli di basi di dati più recenti: quello relazionale, quello orientato agli oggetti e quello multidimensionale. Per avere ulteriori informazioni sui modelli gerarchico e di rete, si consultino i testi proposti nella Bibliografia al termine di questo capitolo.

8.8 Ottimizzare per le prestazioni di interrogazione

L'obiettivo principale della progettazione di basi di dati moderne è quello di ottimizzare le prestazioni di elaborazione. Questa include l'aggiunta, l'elaborazione e la modifica di una base di dati, così come diverse attività di recupero dei dati. Nel caso di basi di dati che hanno un traffico di recupero più intenso di quello di manutenzione, l'ottimizzazione delle prestazioni delle interrogazioni (che genera in anticipo schermate on-line e off-line e ad hoc e report per gli utenti finali) è l'obiettivo primario. Questo capitolo ha già considerato la maggior parte delle decisioni che si devono prendere per perfezionare la progettazione della base di dati per soddisfare le richieste di interrogazione (accorpamento, indici, organizzazioni dei file e così via). In questo ultimo paragrafo del capitolo verranno introdotte alcune opzioni supplementari di progettazione e di elaborazione incluse nei DBMS più moderni.

L'entità del lavoro di un progettista che deve ottimizzare le prestazioni di interrogazione dipende in gran parte dal DBMS. A causa del costo elevato degli sviluppatori di basi di dati esperti, meno è il lavoro da fare, meno costerà lo sviluppo e l'uso di una base di dati. Alcuni DBMS non permettono un grande controllo ai progettisti e agli scrittori di interrogazioni sul modo in cui un'interrogazione viene elaborata o sulla posizione fisica dei dati per l'ottimizzazione della scrittura e della lettura dei dati. Altri

Figura 8.12
Architetture delle basi di dati.



sistemi offrono invece un controllo maggiore e spesso richiedono un lavoro più esteso per perfezionare la progettazione delle basi di dati e la struttura delle interrogazioni per ottenere prestazioni accettabili. A volte il carico di lavoro varia a tal punto e le opzioni di progettazione sono talmente sofisticate che delle prestazioni buone sono tutto quanto si può ottenere. Quando il carico di lavoro è ben indirizzato, per esempio in un data warehouse nel quale ci sono pochi aggiornamenti in batch e nel quale interrogazioni molto complesse richiedono grandi segmenti della base di dati, le prestazioni possono essere perfezionate con alcuni ottimizzatori veloci nel DBMS, con una progettazione intelligente della base di dati e delle interrogazioni o con una combinazione di entrambi gli elementi. Per esempio, il DBMS NCR Teradata è perfezionato per l'elaborazione parallela in un ambiente di data warehouse. In questo caso, è difficile che un progettista di base di dati o uno scrittore di interrogazioni possano apportare dei miglioramenti nelle funzionalità del DBMS per la memorizzazione e l'elaborazione dei dati. Poiché questa situazione è rara, è importante che un progettista di basi di dati prenda in considerazione le opzioni per il miglioramento delle prestazioni di elaborazione della base di dati.

8.8.1 Elaborazione parallela delle interrogazioni

Una delle modifiche principali apportate all'architettura dei computer negli ultimi anni è l'incremento nell'uso di processori multipli nei server di base di dati. Questi server utilizzano spesso la tecnologia dei processori simmetrici, o SMP (Schumacher, 1997). Per approfittare dei vantaggi delle funzioni di elaborazione parallela, alcuni dei DBMS più sofisticati includono strategie per suddividere le interrogazioni in moduli che possano essere pre-elaborati in parallelo da ognuno dei processori correlati. L'approccio più comune consiste nel duplicare l'interrogazione, così che ogni copia lavori su una porzione diversa della base di dati, solitamente una partizione orizzontale (insieme di righe). Le partizioni devono essere definite in anticipo dal progettista della base di dati. La stessa interrogazione viene eseguita in ogni porzione in parallelo su processori diversi e i risultati intermedi di ogni processore vengono uniti per creare il risultato finale come se l'interrogazione fosse stata eseguita sull'intera base di dati.

Si supponga di avere una tabella Order con molti milioni di righe per la quale le prestazioni di interrogazione sono state lente. Per assicurarsi che analisi successive di questa tabella vengano eseguite in parallelo utilizzando almeno tre processori, si può modificare la struttura della tabella con il seguente comando SQL:

```
ALTER TABLE ORDER PARALLEL 3
```

Poiché necessario regolare ciascuna tabella al grado di parallelismo migliore, non è raro dover modificare una tabella più volte fino a trovare il grado giusto.

La velocità dell'elaborazione parallela delle interrogazioni può essere stupefacente. Schumacher (1997) cita un test nel quale il tempo di esecuzione di un'interrogazione è stato ridotto della metà grazie all'elaborazione parallela rispetto alla normale analisi della tabella. Poiché un indice è una tabella, anche agli indici può essere conferita una struttura parallela, così che anche le analisi degli indici possano essere veloci. Schumacher (1997) cita un altro esempio nel quale il costo nel tempo necessario per creare un indice mediante elaborazione parallela è stato ridotto da circa sette minuti a cinque secondi!

Oltre alle analisi delle tabelle, altri elementi di un'interrogazione possono essere elaborati in parallelo, come alcuni tipi di unione tra tabelle correlate, raggruppamenti di risultati delle interrogazioni in categorie, *unione* di più parti di risultati delle interrogazioni, ordinamento delle righe e calcolo dei valori aggregati. Anche le operazioni di aggiornamento, eliminazione e aggiunta di righe possono essere elaborate in parallelo. Inoltre, le prestazioni di alcuni comandi di creazione di una base di dati possono essere migliorate grazie all'elaborazione parallela; fra questi, la creazione e la ricostruzione di un indice e la creazione di una tabella dai dati in una base di dati. L'ambiente Oracle può essere pre-configurato

specificando il numero dei server di basi di dati paralleli che possono esistere. Una volta effettuata questa operazione, l'elaboratore dell'interrogazione deciderà quale ritiene sia l'uso migliore dell'elaborazione parallela per ogni comando.

8.8.2 Aggirare l'ottimizzazione automatica delle interrogazioni

Può accadere che lo scrittore delle interrogazioni conosca, o possa scoprire, le informazioni chiave sull'interrogazione che possono essere trascurate o che non sono note al modulo ottimizzatore delle interrogazioni del DBMS. Con queste informazioni, uno scrittore di interrogazioni può farsi un'idea del modo migliore di elaborare un'interrogazione. Prima di ciò, è tuttavia necessario sapere come l'interrogazione verrà elaborata dall'ottimizzatore (che solitamente sceglie quel piano di elaborazione che riduce al minimo i tempi o i costi di elaborazione previsti). Questo vale soprattutto per un'interrogazione mai presentata in precedenza. Fortunatamente, con la maggior parte dei DBMS si può conoscere il piano dell'ottimizzatore rispetto all'elaborazione dell'interrogazione ancora prima di eseguire quest'ultima. Un comando come EXPLAIN o EXPLAIN PLAN (il comando esatto varia a seconda del DBMS) visualizza il modo in cui l'ottimizzatore intende accedere agli indici, utilizzare server paralleli e unire le tabelle per preparare il risultato dell'interrogazione. Se si fa precedere il comando relazionale effettivo dalla clausola di spiegazione, il processore visualizzerà i passaggi logici per elaborare l'interrogazione e interromperà l'elaborazione prima di accedere effettivamente alla base di dati. L'ottimizzatore sceglie il piano migliore in base alle statistiche su ciascuna tabella, come la lunghezza media delle righe o il numero delle righe. Può essere necessario costringere il DBMS a calcolare statistiche aggiornate sulla base di dati (si veda, per esempio, il comando Analyze di Oracle) per ottenere una stima accurata dei costi dell'interrogazione. È possibile associare diversi comandi di spiegazione all'interrogazione scritta in modi diversi per vedere se l'ottimizzatore prevede prestazioni differenti. Si può quindi scegliere la forma di interrogazione per la quale sono previsti i tempi di elaborazione migliore, oppure si può decidere di non eseguire l'interrogazione se è troppo costosa.

C'è un modo per migliorare le prestazioni di elaborazione delle interrogazioni. Con alcuni DBMS, è possibile fare in modo che il DBMS esegua i passaggi in modo diverso o che utilizzi le proprie potenzialità, come i server paralleli, diversamente da quello che l'ottimizzatore ritiene sia il piano migliore.

Per esempio, si supponga di voler contare il numero di ordini elaborati da un determinato agente di vendita, Smith. In Oracle, l'elaborazione parallela delle tabelle funziona solo quando una tabella viene analizzata e non quando vi si accede tramite un indice. In Oracle si può quindi imporre sia un'analisi completa della tabella sia un'analisi in parallelo. Il comando SQL per questa interrogazione è il seguente:

```
SELECT /*+ FULL(ORDER) PARALLEL(ORDER,3) */COUNT(*)
FROM ORDER
WHERE SALESPERSON = "SMITH";
```

La clausola tra i delimitatori /* */ è il suggerimento per Oracle. Questo suggerimento aggira qualsiasi piano che Oracle crea per questa interrogazione. I suggerimenti sono quindi specifici per ciascuna interrogazione, ma il loro uso deve essere anticipato modificando la struttura delle tabelle che devono essere gestite con l'elaborazione parallela.

8.8.3 Scegliere la dimensione dei blocchi di dati

Come detto in precedenza, i dati vengono trasferiti tra la RAM e la memoria su disco in blocchi, o pagine. La dimensione dei blocchi di dati può influenzare notevolmente le prestazioni delle interrogazioni. Una dimensione troppo piccola può generare molte operazioni fisiche di I/O per ogni riga di tabella a cui si accede o può dare come risultato numerose righe. Un blocco di dati troppo grande può comportare il trasferimento di dati extra, con un notevole spreco di tempo. La dimensione di blocco minima è solitamente di 2 kilobyte; il limite superiore è determinato dal sistema operativo del computer, ma in genere è di 32 kilobyte o superiore. Una volta stabilita la dimensione del blocco di dati, questa può essere modificata solo scaricando i dati, ridefinendo la base di dati e ricaricando nuovamente i dati.

Quando si passa da dimensioni di blocco piccole a dimensioni più grandi, si devono considerare cinque fattori (Yuhanna, 2000).

- *Conflitto di blocco.* È la misura della necessità di non accedere in modo simultaneo allo stesso blocco di dati attraverso numerosi comandi di I/O. Blocchi più piccoli creano meno conflitti. I conflitti riguardano la maggior parte degli ambienti nei quali vengono eseguite più operazioni sulla stessa base di dati, come quando ci sono molti utenti on-line.
- *Velocità di accesso casuale alle righe.* È la velocità alla quale si può accedere a una riga di una tabella. Anche in questo caso i blocchi più piccoli sono l'ideale. L'accesso casuale alle righe è comune nelle applicazioni di elaborazione delle transazioni on-line.
- *Velocità di accesso sequenziale alle righe.* È la velocità alla quale una tabella viene analizzata. In questo caso, è meglio che i blocchi di dati siano più grandi. Blocchi grandi consentono che molte righe vengano collocate nella RAM con un'operazione di I/O, così che siano necessarie meno operazioni di I/O per recuperare tutte le righe di tabella necessarie per l'interrogazione. Un blocco di dati grandi si adatta bene all'accorpamento (trattato in precedenza nel capitolo), perché un intero cluster può essere memorizzato in un'unica operazione di I/O. Le analisi sequenziali possono avvenire nelle applicazioni di supporto alle decisioni e di data warehouse e in quelle che producono report riepilogativi delle transazioni.
- *Dimensione di riga.* È la lunghezza di tutti i campi in una riga di tabella. Solitamente è preferibile cercare di far coincidere la dimensione del blocco con la dimensione di riga della tabella fisica o con un multiplo della dimensione di riga.
- *Overhead.* È il costo nel tempo che il DBMS deve sopportare per poter gestire tutte le operazioni di I/O necessarie a produrre il risultato di un'interrogazione o di un'altra operazione della base di dati. Dimensioni di blocco piccole producono un costo maggiore che non dimensioni grandi.

Questi fattori si riferiscono raramente a una sola dimensione di blocco di dati. In genere, dimensioni di blocco più piccole vengono utilizzate per le applicazioni di elaborazione delle transazioni on-line, mentre dimensioni di blocco più grandi vengono utilizzate per le basi di dati con un sistema di supporto alle decisioni o di data warehouse. Gli ambienti con un carico di lavoro misto possono essere molto difficili da regolare in termini di dimensioni di blocco di dati.

8.8.4 Equilibrare l'I/O tra i controller di disco

Un controller di disco gestisce le operazioni di I/O per le unità disco associate a quel controller. Più controller sono meglio di pochi (come in precedenza, un parallelismo maggiore significa prestazione migliori), ma il costo può limitare il numero di controller che si hanno a disposizione. I vantaggi dei

controller multipli si hanno quando tutti questi sono tenuti impegnati. È quindi buona norma allocare i file di sistema del DBMS e della base di dati su unità, così che ci sia più o meno lo stesso carico di lavoro su ciascun controller e, ancora meglio, su ciascun disco.

Le tabelle possono essere spostate facilmente da un disco a un altro dopo le tabelle che contengono dati. L'assegnazione iniziale dei file ai dischi può quindi essere modificata. Per migliorare le prestazioni dell'elaborazione dei file si possono seguire alcune linee guida.

- Capire quali file sono su quali unità e quali unità sono associate a quali controller.
- Capire i programmi predefiniti e la natura delle interrogazioni ad hoc eseguite nella base di dati (potrebbe essere necessario concentrarsi sui programmi più importanti o sui giorni e le ore più problematiche).
- Raccogliere statistiche sull'uso dei dischi e dei controller e sull'accesso alle tabelle (partizionamento).
- Riequilibrare il carico di lavoro spostando le tabelle tra unità e controller.

In generale, quando non è possibile raccogliere i dati necessari in un'interrogazione (o tra interrogazioni eseguite simultaneamente) memorizzata nello stesso blocco di dati, è preferibile collocare questi dati su dischi e controller diversi, così che sia possibile accedervi in parallelo. Anche in questo caso è difficile, se non impossibile, ottimizzare le prestazioni quando c'è un pattern di carico di lavoro molto vario. Potrebbe essere necessario concentrarsi su alcune delle applicazioni più importanti (o sulle applicazioni che lavorano con prestazioni oggi inaccettabili), eseguire l'analisi precedente su di esse ed equilibrare le assegnazioni dei file in modo da ottenere le prestazioni migliori per queste applicazioni senza avere un decadimento nelle altre.

8.8.5 Linee guida per una progettazione migliore delle interrogazioni

Nei paragrafi precedenti di questo capitolo sono state presentate numerose tecniche e approcci per la progettazione delle basi di dati e delle interrogazioni che comportano un'elaborazione rapida di queste ultime. Diversi esperti di basi di dati hanno sviluppato linee guida che non fanno direttamente riferimento agli argomenti già trattati. Si vedano DeLoach (1987), Holmes (1996) e Kurka (1999) per alcuni suggerimenti sul miglioramento delle prestazioni delle interrogazioni con varie impostazioni. Ecco alcuni dei loro suggerimenti che si applicano a diverse situazioni.

- *Capire come gli indici sono utilizzati nell'elaborazione delle interrogazioni.* Molti DBMS utilizzano solo un indice per tabella in un'interrogazione. È necessario sapere come il DBMS sceglie l'indice da utilizzare e come controlla l'accesso agli indici e poi eliminare gli indici che vengono poco utilizzati. Questo migliora le prestazioni delle operazioni di aggiornamento della base di dati. In generale, le interrogazioni che hanno criteri equivalenti per la scelta delle righe di tabella (per esempio, WHERE Finish = "Birch" OR "Walnut") comportano un'elaborazione più veloce che non quelle che implicano qualifiche più complesse (per esempio, WHERE Finish NOT = "Walnut"), perché i criteri di uguaglianza possono essere valutati attraverso gli indici. Anche in questo caso, è necessario sapere come il DBMS tratta i vari tipi di clausole nelle interrogazioni.
- *Utilizzare tipi di dati compatibili per i campi con letterali nelle interrogazioni.* "Tipi di dati compatibili" significa probabilmente che il DBMS può evitare di dover convertire i dati durante l'elaborazione dell'interrogazione.

- *Scrivere interrogazioni semplici.* La forma di interrogazione più semplice è solitamente la più facile da elaborare per il DBMS. Per esempio, poiché i DBMS relazionali si basano sulla teoria degli insiemi, è preferibile scrivere interrogazioni che manipolano insiemi di righe e letterali.
- *Suddividere le interrogazioni complesse in più parti semplici.* Poiché il DBMS può utilizzare solo un indice per interrogazione, è preferibile suddividere un'interrogazione complessa in più parti semplici (ognuna delle quali utilizza un indice) e quindi unire i risultati delle interrogazioni più piccole. Per esempio, poiché un DBMS relazionale lavora con gli insiemi, è molto semplice per il DBMS unire (con il comando UNION) due insiemi di righe che sono il risultato di due interrogazioni semplici indipendenti.
- *Non nidificare un'interrogazione in un'altra.* Come si vedrà nei Capitoli 9 e 10, il linguaggio SQL permette di scrivere un'interrogazione in un'altra (che viene chiamata *sotto-interrogazione*). Interrogazioni di questo tipo sono solitamente meno efficienti di quelle che evitano le sotto-interrogazioni per produrre lo stesso risultato.
- *Non unire una tabella a se stessa.* Si eviti, se possibile, di utilizzare la stessa tabella in due o più ruoli diversi nella stessa interrogazione (si parla in questo caso di *self-join*, come si vedrà nel Capitolo 9). Solitamente è preferibile creare una copia temporanea della tabella e poi fare riferimento alla tabella originaria con quella temporanea (un metodo più efficiente per elaborare l'interrogazione).
- *Creare tabelle temporanee per gruppi di interrogazioni.* Se possibile, si riutilizzino i dati utilizzati in una sequenza di interrogazioni. Per esempio, se tutte le interrogazioni di una serie fanno riferimento allo stesso sottoinsieme di dati della base di dati, può essere più utile memorizzare il sottoinsieme in una o più tabelle temporanee e poi fare riferimento a queste tabelle temporanee in una serie di interrogazioni. Questo impedisce la combinazione ripetuta degli stessi dati o l'analisi ripetuta della base di dati per trovare lo stesso segmento di base di dati per ciascuna interrogazione. Il risvolto negativo è che le tabelle temporanee non cambiano quando le tabelle originali vengono aggiornate durante l'esecuzione delle interrogazioni.
- *Combinare operazioni di aggiornamento.* Se possibile, si uniscano più comandi di aggiornamento in un unico comando. Questo ridurrà l'overhead dell'elaborazione delle interrogazioni e permetterà al DBMS di cercare un modo per elaborare gli aggiornamenti in parallelo.
- *Recuperare solo i dati necessari.* In questo modo si riduce il numero di blocchi di dati a cui si accede e che vengono trasferiti. Può sembrare ovvio, ma esistono alcune scorciatoie per scrivere interrogazioni che violano questa linea guida. Per esempio, in SQL, il comando SELECT * di EMP recupera tutti i campi da tutte le righe della tabella EMP. Se però l'utente vuole vedere solo alcune delle colonne della tabella, trasferire le colonne supplementari aumenta i tempi di elaborazione.
- *Non permettere che il DBMS effettui l'ordinamento senza un indice.* Se i dati devono essere visualizzati secondo un ordinamento e non esiste un indice nel campo chiave di ordinamento, si ordinino i dati all'esterno del DBMS dopo che i risultati sono stati recuperati. In genere un'utility di ordinamento agisce più velocemente che non se si ordinano i dati senza l'aiuto di un indice del DBMS.
- *Informarsi e formarsi.* Si tenga una traccia dei tempi di elaborazione, si rivedano i piani delle interrogazioni con il comando EXPLAIN e si migliori la propria comprensione del modo in cui il DBMS determina come elaborare le interrogazioni. Si partecipi alla formazione specializzata proposta dal produttore del proprio DBMS sulla scrittura di interrogazioni efficienti, che fornirà tutte le delucidazioni sull'ottimizzatore delle interrogazioni.

- *Considerare i tempi complessivi di elaborazione delle interrogazioni ad hoc.* Il tempo complessivo include il tempo impiegato dal programmatore (o dall'utente finale) per scrivere l'interrogazione, così come il tempo necessario a elaborare l'interrogazione. Molto spesso, per le interrogazioni ad hoc è preferibile che il DBMS esegua il lavoro extra per consentire all'utente di scrivere più velocemente un'interrogazione. Questo è esattamente lo scopo della tecnologia: permettere alle persone di essere più produttive. Non è quindi opportuno passare troppo tempo a scrivere interrogazioni efficienti, soprattutto nel caso di quelle ad hoc. Si scriva invece un'interrogazione che sia corretta logicamente (che produca cioè i risultati desiderati), e si lasci che il DBMS esegua il lavoro (ovviamente si esegua prima un comando EXPLAIN per essere sicuri di non aver scritto un'interrogazione che comporti un ritardo nei tempi di elaborazione per tutti gli utenti). Questo suggerisce un corollario: quando possibile, si esegua l'interrogazione quando il carico sulla base di dati è leggero, poiché il tempo complessivo di elaborazione di un'interrogazione include i ritardi provocati dall'altro carico sul DBMS e sulla base di dati.

Questo elenco conclude la trattazione delle opzioni avanzate per il perfezionamento delle prestazioni di una base di dati. Non tutte le opzioni sono disponibili in tutti i DBMS, e ogni DBMS ha spesso opzioni uniche dovute alla sua progettazione. Si consultino i manuali del proprio DBMS per sapere quali sono le opzioni specifiche disponibili.

Riepilogo

Durante la progettazione fisica di una base di dati, il progettista traduce la descrizione logica dei dati in specifiche tecniche per la memorizzazione e il recupero dei dati. L'obiettivo è quello di creare un progetto per la memorizzazione dei dati che fornisca prestazioni adeguate e che garantisca l'integrità della base di dati, la sua protezione e il suo ripristino. Nella progettazione fisica di una base di dati si considerano le relazioni normalizzate e le stime dei volumi di dati, le definizioni dei dati, le prestazioni di elaborazione e la loro frequenza, le aspettative degli utenti e le caratteristiche della tecnologia di base di dati per stabilire le specifiche di campo, la progettazione dei record, l'organizzazione dei file e l'architettura della base di dati.

Un campo è l'unità più piccola dei dati dell'applicazione e corrisponde a un attributo nel modello di dati logico. Tra gli altri fattori, è necessario determinare il tipo di dati, i controlli di integrità e il modo in cui gestire i valori mancanti per ciascun campo. Un tipo di dati è uno schema di codifica dettagliato per la rappresentazione dei dati di un'azienda. I dati possono essere codificati o compressi per ridurre lo spazio di memoria. Il controllo dell'integrità dei dati comprende la specifica di un valore predefinito, un intervallo di valori permessi, l'eventualità di un valore nullo e l'integrità referenziale.

Un record fisico è un gruppo di campi memorizzati in posizioni adiacenti in memoria e recuperati insieme come unità. I record fisici sono solitamente memorizzati in una pagina (o blocco di dati), che è l'entità dei dati letti o scritti con un'operazione di I/O nella memoria secondaria. Il numero di record in una pagina è chiamato fattore di blocco. Per ragioni di efficienza, gli attributi di una relazione possono non essere memorizzati in un record fisico, mentre gli attributi di più relazioni possono esserlo. Un processo di denormalizzazione trasforma le relazioni normalizzate in specifiche del record fisico denormalizzato. La denormalizzazione è applicata per collocare in un record fisico quegli attributi che spesso vengono richiesti insieme in un'operazione di I/O. La denormalizzazione include il partizionamento orizzontale, che suddivide una relazione in più specifiche di record collocando righe diverse in record diversi in base ai valori di colonna comuni. La denormalizzazione include anche il partizionamento verticale, che distribuisce le colonne di una relazione in file separati, ripetendo la chiave primaria in ognuno dei file.

Un file fisico è una porzione denominata di memoria secondaria allocata per la memorizzazione dei record fisici. I dati in un file fisico sono organizzati attraverso una combinazione di memorizzazione sequenziale e puntatori. Un puntatore è un campo di dati che può essere utilizzato per trovare un campo correlato o un record di dati.

Un'organizzazione di file dispone i record di un file su un dispositivo di memorizzazione secondaria. Le tre categorie principali di organizzazioni di file sono: 1) sequenziale, che memorizza i record uno dopo l'altro in base al valore di chiave primaria; 2) indicizzata, nella quale i record sono memorizzati sequenzialmente o non sequenzialmente e un indice viene utilizzato per tenere traccia di dove i record vengono memorizzati; 3) di hash, nella quale l'indirizzo di ogni record viene determinato utilizzando un algoritmo che converte un valore di chiave primaria in un indirizzo di record. Record fisici di tipo diverso possono essere accorpati in un file fisico per poter collocare i record utilizzati di frequente gli uni vicini agli altri in un'altra memoria secondaria.

L'organizzazione di file indicizzata è attualmente una delle più diffuse. Un indice può essere basato su una chiave univoca o non univoca (secondaria), cosa che permette che più di un record venga associato allo stesso valore di chiave. La forma più nuova di un indice, quello bitmap, crea una tabella di bit nella quale un bit attivo indica che il record correlato ha il relativo valore di chiave. Un indice di join indica le righe di due o più tabelle che hanno valori comuni per i campi correlati. Una tabella di indice di hashing rende la collocazione dei dati indipendente dall'algoritmo di hashing e permette che si possa accedere agli stessi dati attraverso numerose funzioni di hashing su campi diversi. Gli indici sono importanti per velocizzare il recupero dei dati, soprattutto quando vengono utilizzate condizioni multiple per selezionare, ordinare o correlare i dati. Gli indici sono utili in molte situazioni, per esempio con le tabelle grandi, con le colonne utilizzate di frequente per qualificare i dati da recuperare, quando un campo ha un gran numero di valori distinti e quando l'elaborazione dei dati è dominata dal recupero dei dati piuttosto che dalla loro manutenzione.

L'efficienza dell'accesso ai file e l'affidabilità dei file possono essere migliorate dall'uso di un RAID (*Redundant Array of Inexpensive Disk*), che permette che i blocchi di dati vengano letti e scritti in parallelo su dischi diversi, riducendo così i ritardi di input/output delle operazioni di I/O tradizionali su una singola unità disco. I vari livelli di RAID permettono a un progettista di file e di base di dati di scegliere la combinazione tra efficienza di accesso, uso dello spazio e tolleranza degli errori più adatta alle applicazioni della base di dati.

Le architetture di base di dati attualmente in uso sono quelle gerarchica, di rete, relazionale, orientata agli oggetti e multidimensionale. Quelle gerarchica e di rete si trovano soprattutto nelle applicazioni legacy, mentre quelle relazionale, orientata agli oggetti e multidimensionale sono utilizzate nello sviluppo dei sistemi più nuovi.

L'introduzione dei server di base dati multiprocessore ha reso possibili nuove potenzialità nei sistemi di gestione delle basi di dati. Una delle caratteristiche principali è la capacità di suddividere un'interrogazione e di elaborarla in parallelo su più segmenti di una tabella. L'elaborazione parallela può migliorare notevolmente la velocità di elaborazione delle interrogazioni. Inoltre i programmatori possono migliorare notevolmente le prestazioni fornendo al DBMS alcuni suggerimenti sulla sequenza da seguire nello svolgimento delle operazioni sulle tabelle. Questi suggerimenti aggirano l'ottimizzatore costoso del DBMS. Sia il DBMS sia i programmatori possono esaminare le statistiche sulla base di dati per determinare come elaborare un'interrogazione. Il capitolo ha presentato un'ampia gamma di linee guida per una buona progettazione delle interrogazioni.

Questo capitolo conclude la parte del libro sulla progettazione delle basi di dati. Avendo sviluppato specifiche complete dei dati fisici, si è pronti per iniziare a implementare la base di dati con la relativa tecnologia. Implementare significa definire le routine client e server della base di dati e di programmazione che gestiscono le interrogazioni, i report e le transazioni. Questi sono gli argomenti principali dei prossimi cinque capitoli, che trattano l'implementazione di basi di dati relazionali su piattaforme client, piattaforme server, ambienti client/server e tecnologie di data warehouse.

Ripasso del capitolo

Parole chiave

Algoritmo di hashing	Indice	Organizzazione di file sequenziale	RAID (<i>Redundant Array of Inexpensive Disks</i>)
Campo	Indice bitmap	Pagina	Record fisico
Chiave secondaria	Indice di join	Partizionamento orizzontale	Stripe
Denormalizzazione	Organizzazione di file	Partizionamento verticale	Tabella di indice hash
Estensione	Organizzazione di file hash	Puntatore	Tablespace
Fattore di blocco	Organizzazione di file indicizzata		Tipo di dati
File fisico			

Domande di ripasso

1. Definire ciascuno dei seguenti termini chiave:
a. organizzazione di file
b. organizzazione di file sequenziale
c. organizzazione di file indicizzata
d. organizzazione di file hash
e. indice
f. chiave secondaria
g. tipo di dati
h. indice bitmap
i. RAID
j. indice di join
l. stripe

2. Unire i seguenti termini alle rispettive definizioni:
___ indice bitmap; ___ algoritmo di hashing;
___ record fisico; ___ puntatore; ___ file fisico.
a. dati letti in un'operazione di I/O
b. il numero di record in una pagina
c. un'area denominata della memoria secondaria
d. una tabella di zeri e uno
e. un campo che non contiene dati aziendali
f. converte un valore di chiave in un indirizzo
g. campi adiacenti

3. Confrontare i seguenti termini:
a. partizionamento orizzontale; partizionamento verticale
b. file fisico; tablespace
c. record fisico; file fisico
d. pagina; record fisico
e. chiave secondaria; chiave primaria

4. Quali sono gli input principali nella progettazione fisica di una base di dati?

5. Quali sono le decisioni chiave nella progettazione fisica di una base di dati?

6. Quali informazioni sono mostrate in una mappa d'uso composta?

7. Quali decisioni è necessario prendere per sviluppare una specifica di campo?

8. Quali sono gli obiettivi della scelta di un tipo di dati per un campo?

9. Perché i valori di campo vengono codificati o compressi?

10. Quali sono le opzioni disponibili per controllare l'integrità dei file?

11. Descrivere tre modi per gestire i valori di campo mancanti.

12. Spiegare perché le relazioni normalizzate possono non essere record fisici efficienti.

13. Elencare tre situazioni comuni che suggeriscono la denormalizzazione delle relazioni per formare record fisici.

14. Quali sono i vantaggi e gli svantaggi del partizionamento orizzontale e verticale?

15. Elencare sette criteri importanti nella scelta di un'organizzazione di file.

16. In quali casi può essere utile un indice bitmap?

17. Quali sono i vantaggi di una tabella di indice hash?

18. Qual è lo scopo dell'accorpamento dei dati in un file?

19. Citare sette linee guida per la scelta degli indici.

20. Confrontare i due modi di visualizzare le basi di dati multidimensionali.

21. Come il comando EXPLAIN può essere d'aiuto nella scrittura di un'interrogazione più efficiente?

22. Descrivere quattro opzioni per l'ottimizzazione delle prestazioni di interrogazione.

23. Qual è il livello di RAID migliore? Perché?
- Problemi ed esercizi**
1. Si considerino le due relazioni seguenti per il Millenium College.

STUDENT (Student_ID, Student_Name, Campus_Adress, GPA)
REGISTRATION (Student_ID, Course_ID, Grade)

Quella che segue è un'interrogazione frequente su queste relazioni.

SELECT STUDENT.STUDENT_ID, STUDENT_NAME,
COURSE_ID, GRADE
FROM STUDENT, REGISTRATION

```

WHERE      STUDENT.STUDENT_ID =
           REGISTRATION.STUDENT_ID
AND        GPA > 3.0
ORDER BY STUDENT_NAME;

```

- a. Su quali attributi dovrebbero essere definiti gli indici per velocizzare questa interrogazione? Motivare la scelta di ciascun attributo.
 - b. Scrivere comandi SQL per creare gli indici per ciascun attributo identificato al punto a.
2. Si consideri la mappa d'uso composta nella Figura 8.1. Dopo un certo periodo di tempo, i presupposti per questa mappa sono cambiati come segue.
- a. C'è una media di 40 preventivi (invece che 50) per ciascun fornitore.
 - b. Le parti prodotte rappresentano solo il 30 per cento di tutte le parti; le parti acquistate rappresentano il 75 per cento.
 - c. Il numero di accessi diretti alle parti acquistate aumenta a 75 all'ora (invece che 60).
- Disegnare una nuova mappa d'uso composta che rifletta queste nuove informazioni per sostituire quella della Figura 8.1.
3. Scegliere i tipi di dati Oracle per gli attributi nelle relazioni normalizzate della Figura 8.4.
4. Si supponga di dover progettare un valore predefinito per il campo dell'età in un record di studente di un'università. Quali sono i valori possibili da considerare e perché? Come può variare il valore predefinito in base alle altre caratteristiche dello studente, come l'istituto all'interno dell'università o il voto?
5. Quando uno studente non ha scelto una specializzazione universitaria, l'università immette spesso un valore "Undecided" nel campo della specializzazione. "Undecided" è un valore predefinito o un modo per rappresentare il valore nullo?
6. Si considerino le seguenti relazioni normalizzate di una base di dati di una grande catena di rivenditori.
- ```

STORE (Store_ID, Region, Manager_ID, Square_Feet)
EMPLOYEE (Employee_ID, Where_Work, Employee_Name, Employee_Address)
DEPARTMENT (Department_ID, Manager_ID, Sales_Goal)
SCHEDULE (Department_ID, Employee_ID, Date)

```
- Quali possibilità di normalizzazione di queste relazioni esistono quando si definiscono i record fisici per questa base di dati? In quali circostanze si può ipotizzare di creare tali record denormalizzati?
7. Si presuma di avere un file di disco rigido concepito per contenere un massimo di 1000 record di 240 byte ciascuno. Si presuma che una pagina sia lunga 4000 byte e che i record non possano estendersi su più pagine. Quanti byte saranno necessari per questo file?
  8. Quali problemi possono sorgere dal partizionamento verticale di una relazione? Considerati questi problemi, quali condizioni generali influenzano il momento in cui partizionare una relazione verticalmente?

9. Con un'organizzazione di file sequenziale, è possibile permettere l'analisi sequenziale dei dati basati su più ordini sui quali è stato effettuato un ordinamento? Perché sì o no?
10. Si supponga che ogni record di un file sia connesso ai suoi record precedente e successivo in una sequenza di chiave mediante l'uso di puntatori. Ogni record dovrebbe quindi avere il seguente formato:  
Chiave primaria, altri attributi, puntatore al record precedente, puntatore al record successivo.
  - a. Quali sono i vantaggi di questa organizzazione di file rispetto a un'organizzazione sequenziale?
  - b. Diversamente da un'organizzazione di file sequenziale, sarebbe possibile mantenere i record in sequenze multiple? Perché sì o no?
11. Si presuma che uno studente nella base di dati di un'università abbia un indice su Student\_ID (la chiave primaria) e alcuni indici su Major, Age, Marital\_Status e Home\_Zipcode (tutte chiavi secondarie). Si presuma inoltre che l'università voglia un elenco di studenti che si specializzano in MIS o in scienze dell'informazione, che abbiano 25 anni e che siano sposati OR di studenti che si specializzano in scienze dell'informazione, che non siano sposati e il cui codice di avviamento postale sia 45462. Come si possono utilizzare questi indici in modo tale da accedere solo ai record che soddisfano questi criteri?
12. Si consideri nuovamente il file degli studenti descritto nel problema ed esercizio 11. Per quali indici e in quali condizioni può avere senso utilizzare un indice bitmap? Scegliere una delle chiavi che è una candidata probabile per un indice bitmap e disegnare una figura simile alla Figura 8.8 che ne mostri la struttura.
13. Creare un indice di join sui campi Customer\_ID delle tabelle dei clienti e degli ordini della Figura 7.4.
14. Si consideri la Figura 8.7c. Presumendo che le righe vuote nelle foglie di questo indice mostrino lo spazio in cui i nuovi record possono essere memorizzati, spiegare dove il record di 'Sooners' viene memorizzato. Dove viene memorizzato il record di 'Flashes'? Cosa può accadere quando una delle foglie è completa e occorre aggiungere un nuovo record a quella foglia?
15. Si consideri la Figura 8.12, che mette a confronto le caratteristiche generali delle architetture di basi di dati più diffuse. Scrivere relazioni equivalenti al modello di base di dati con la vista di tabelle multidimensionali.
16. L'accorpamento dei file può verificarsi dopo che i file sono stati popolati con i record? Perché sì o no?
17. Come descritto in questo capitolo, l'elaborazione parallela delle interrogazioni fa sì che la stessa interrogazione venga eseguita su più processori e che ogni processore acceda in parallelo a un sottoinsieme diverso della base di dati. Un'altra forma di elaborazione parallela delle interrogazioni, non trattata in questo capitolo, può suddividere l'interrogazione in modo che ognuna delle sue parti venga eseguita su un processore diverso ma

che acceda comunque a qualsiasi parte della base di dati che le occorre. La maggior parte delle interrogazioni implica una clausola di qualifica che seleziona i record di interesse per l'interrogazione. In generale questa clausola ha la forma seguente: (condizione OR condizione OR ...) AND (condizione OR condizione OR ...) AND...

Data questa forma generale, come può essere suddivisa un'interrogazione in modo tale che ciascun processore ne gestisca un sottoinsieme e poi unisca ogni parte elaborata?

18. Nella Figura 8.11f, si supponga che i record 2, 3, 7 e 9 debbano essere aggiornati contemporaneamente da quattro utenti diversi. Quali di questi record possono essere aggiornati in parallelo? Definire la programmazione di questi aggiornamenti per ottimizzare al massimo l'elaborazione parallela.

## Esercitazioni

1. Trovare quali sono i sistemi di gestione delle basi di dati disponibili nella propria università per un uso da parte degli studenti. Informarsi su quali tipi di dati sono supportati da questi DBMS. Confrontare questi DBMS basati sui tipi di dati supportati e, in base a questo confronto, suggerire i tipi di applicazioni per i quali ogni DBMS è più adatto.
2. Utilizzando il sito Web del libro e altre risorse Internet, informarsi sulle funzioni di elaborazione parallela dei DBMS più diffusi. In cosa differiscono le varie funzioni?
3. Utilizzando il sito Web del libro e altre risorse Internet, informarsi sulle funzioni di prodotti delle basi di dati orientate agli oggetti. Quanto costano questi DBMS? Per quali tipi di applicazioni i produttori suggeriscono di utilizzare i loro prodotti?
4. Contattare un progettista o amministratore di basi di dati di un'azienda con cui si ha familiarità. Chiedergli quali sono le organizzazioni di file disponibili nei vari DBMS utilizzati nell'azienda. Cercare di capire quali fattori prende in considerazione quando sceglie un'organizzazione per i file della base di dati e, nel caso di file indicizzati, come decide quali indici creare. Gli indici vengono anche eliminati? Perché?
5. Contattare un progettista o amministratore di basi di dati di un'azienda con cui si ha familiarità. Chiedergli se utilizza una tecnologia RAID nelle basi di dati dell'azienda. Perché sì o no? In caso affermativo, quali livello di RAID utilizza e perché?

## Bibliografia

- Babad, Y. M. e Hoffer, J. A., 1984, "Even No Data Has a Value", *Communications of the ACM* 27 (agosto), pp. 748-756.
- Ballinger, C., 1998, "Introducing the Join Index", *Teradata Review* 1 (autunno), pp. 18-23. Anche in [www.teradatareview.com/Fall98/Ballinger.html](http://www.teradatareview.com/Fall98/Ballinger.html).
- Bontempo, C. J. e Saracco, C. M., 1996, "Accelerating Indexed Searching", *Database Programming & Design* 9 (luglio), pp. 37-43.
- Brobst, S., Gant, S. e Thompson, F., 1999, "Partitioning Very Large Database Tables with Oracle8", *Oracle Magazine* (marzo/aprile), pp. 123-126.
- DeLoach, A., 1987, "The Path to Writing Efficient Queries in SQL/DS", *Database Programming & Design* 1 (gennaio), pp. 26-32.
- Finkelstein, R., 1988, "Breaking the Rules Has a Price", *Database Programming & Design* 1 (giugno), pp. 11-14.
- Holmes, J., "More Paths to Better Performance", *Database Programming & Design* 9 (febbraio), pp. 47-48.
- Inmon, W. H., 1988, "What Price Normalization", *ComputerWorld* (17 ottobre), pp. 27, 31.
- Kurka, A., "Rules for Better SELECTS", *SAP Technical Journal* 1 (2), pp. 75-79. Anche in [www.saptechjournal.com](http://www.saptechjournal.com).
- Musciano, C., 1999, "0, 1, 0+1...RAID Basics, Part 1", in *UnixInsider*, presente sul Web all'indirizzo [www.unixinsider.com/unixinsideronline/swol-06-1999/swol-06-1999-raid1\\_p.html](http://www.unixinsider.com/unixinsideronline/swol-06-1999/swol-06-1999-raid1_p.html) (indirizzo verificato al 25 dicembre 2000).
- Rogers, U., 1989, "Denormalization: Why, What, and How?", *Database Programming & Design* 2 (dicembre), pp. 46-53.
- Schumacher, R., 1997, "Oracle Performance Strategies", *DBMS 10* (maggio), pp. 89-93.
- Shah, R., 1999, "Storage Beyond RAID", in *UnixInsider*, presente sul Web all'indirizzo [www.unixinsider.com/unixinsideronline/swol-07-1999/swol-07-connectivity\\_p.html](http://www.unixinsider.com/unixinsideronline/swol-07-1999/swol-07-connectivity_p.html) (indirizzo verificato al 25 dicembre 2000).
- Viehman, P., 1994, "24 Ways to Improve Database Performance", *Database Programming & Design* 7 (febbraio), pp. 32-41.
- Yuhanna, N., 2000, *Oracle8i Database Administration*, Greenwich, CT, Manning Publications.

## Per approfondire

- Elmasri, R. e Navathe, S., 1994, *Fundamentals of Database Systems*, seconda edizione, Menlo Park, CA, Benjamin/Cummings.
- Loney, K., Aronoff, E. e Sonawalla, N., 1996, "Big Tips for Big Tables", *Database Programming & Design* 9 (novembre), pp. 58-62.
- Roti, S., 1996, "Indexing and Access Mechanisms", *DBMS 9* (maggio), pp. 65-70.

## Project Case



# MOUNTAIN VIEW COMMUNITY HOSPITAL

Nei capitoli precedenti è stata sviluppata una base di dati per Mountain View Community Hospital. In questo project case si prenderanno alcune decisioni sulla sua progettazione fisica in base alle basi di dati logica e concettuale sviluppate nei Capitoli 5, 6 e 7.

### Descrizione del progetto

Mountain View Community Hospital ha deciso di utilizzare un sistema di gestione di basi di dati (DBMS) Oracle per questa base di dati. Il progetto della base di dati fisica deve quindi adattarsi alle potenzialità del DBMS. (Se non si ha familiarità con Oracle, si può presumere che Mountain View Community Hospital abbia scelto un altro DBMS con cui si ha più familiarità, e si risponderà alle domande tenendone conto.)

### Domande sul progetto

1. Quali tipi di informazioni supplementari, oltre alle relazioni 3NF sviluppate nel project case del Capitolo 7 per la base di dati concettuale del Capitolo 5, sono necessarie per progettare fisicamente questa base di dati?
2. Ci sono possibilità di un partizionamento orizzontale o verticale per questa base di dati? Se non si è certi, quali altre informazioni potrebbero essere utili per rispondere con più sicurezza a questa domanda?
3. La Figura 1 mostra una mappa d'uso composta iniziale per una porzione della base di dati di Mountain View Community Hospital. Da quando la mappa d'uso è stata sviluppata, sono cambiati alcuni presupposti sull'uso dei dati.
  - C'è una media di 12 (invece che 10) consumi di articoli per paziente.
  - Ai dati delle prestazioni ai pazienti si accede una media di 40 (invece che 30) volte all'ora e ogni volta c'è un accesso anche ai dati dei trattamenti corrispondenti.

Disegnare una nuova versione della Figura 1 che rifletta queste nuove informazioni.

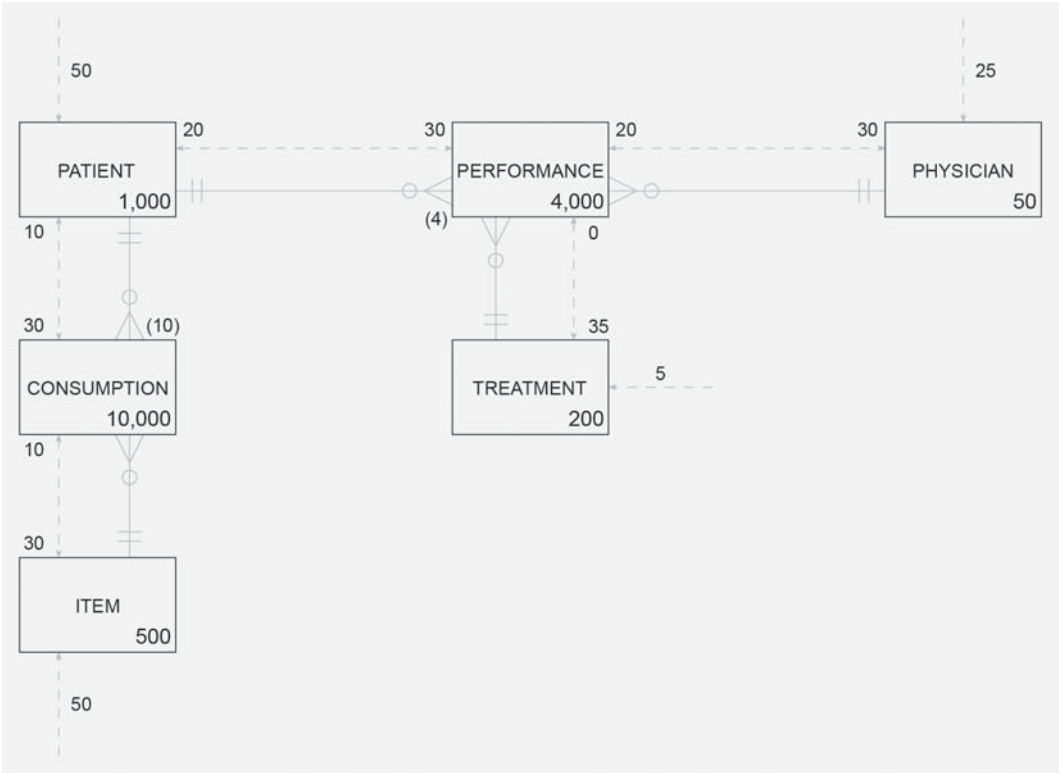
### Esercizi sul progetto

1. Nell'esercizio sul progetto 5 del Capitolo 7 sono stati scritti dei comandi CREATE TABLE per ciascuna relazione logica

nella base di dati di Mountain View Community Hospital della base di dati concettuale nel Capitolo 5. Tuttavia, non sono state comprese appieno le scelte di progettazione fisica messe a disposizione da Oracle (o dal DBMS che si utilizza per questo progetto). Si riconsiderino i comandi CREATE TABLE precedenti mentre si risponde alle seguenti domande.

- a. Ora si sceglierebbero tipi di dati diversi per qualche campo? Perché?
  - b. I campi sono candidati per la codifica? Se sì, quale schema di codifica si potrebbe utilizzare per ognuno di questi campi.
  - c. Quali campi possono avere un valore nullo?
  - d. Si supponga che le date per l'esecuzione dei trattamenti non siano state immesse. Quali procedure si possono utilizzare per gestire questi dati mancanti? Si può e si dovrebbe utilizzare un valore predefinito per questo campo? Perché sì o no?
2. Nella domanda sul progetto 2 è stato chiesto di verificare l'eventualità di un partizionamento. Oltre al partizionamento, ci sono altre possibilità di denormalizzare le relazioni di questa base di dati? Se no, perché? Se sì, dove e come si può eseguire la denormalizzazione?
  3. Nella domanda sul progetto 3 è stata aggiornata la Figura 1, una mappa d'uso composta per una porzione della base di dati di Mountain View Community Hospital. Facendo riferimento alla figura aggiornata, si vede qualche opportunità di accorpare le righe di due o più tabelle? Perché sì o no?
  4. Scrivere comandi CREATE INDEX per gli indici di chiave primaria per ogni tabella nella base di dati di Mountain View Community Hospital.
  5. Si consideri la seguente interrogazione sulla base di dati di Mountain View Community Hospital: "Per ogni trattamento effettuato nelle ultime due settimane, elencare in ordine di ID del trattamento, e ogni ID in ordine di data cronologicamente inverso, i medici che eseguono ogni trattamento (raggruppati per trattamento) e il numero di volte in cui il medico ha effettuato quel trattamento in quel giorno". Creare indici di chiave secondaria per ottimizzare le prestazioni dell'interrogazione. Si presuma di dover rispondere a questa domanda.

**Figura 1**  
Mappa d'uso composta  
per una porzione della  
base di dati di Mountain  
View Community  
Hospital.



Parte quinta  
Implementazione



**Al termine di questo capitolo sarete in grado di fare quanto segue.**

1. Definire i seguenti termini chiave: DBMS relazionale (RDBMS), catalogo, schema, linguaggio di definizione dei dati, linguaggio di manipolazione dei dati, linguaggio di controllo dei dati, tabella di base, vista dinamica, vista materializzata, integrità referenziale, aggregato scalare e aggregato vettoriale.
2. Interpretare la storia e il ruolo di SQL nello sviluppo di una base di dati.
3. Definire una base di dati utilizzando il linguaggio di definizione dei dati SQL.
5. Scrivere interrogazioni su tabelle singole utilizzando i comandi SQL.
6. Stabilire l'integrità referenziale utilizzando SQL.
7. Discutere gli standard SQL-92 e SQL-99.

## Introduzione

SQL è diventato di fatto il linguaggio standard per creare e interrogare le basi di dati relazionali. Lo scopo principale di questo capitolo è quello di analizzare approfonditamente SQL. SQL è il linguaggio più diffuso per i sistemi relazionali. È stato accettato come standard per gli Stati Uniti dall'ANSI (*American National Standards Institute*) ed è un FIPS (*Federal Information Processing Standard*). È anche uno standard internazionale riconosciuto dall'ISO (*International Organization for Standardization*).

Gli standard SQL ANSI sono stati pubblicati per la prima volta nel 1986 e poi aggiornati nel 1989, nel 1992 (SQL-92) e nel 1999 (SQL-99). SQL-99 è un'estensione significativa di SQL-92, che era strutturato in tre livelli: Entry, Intermediate e Full. SQL-99 stabilisce la conformità al livello Core che deve essere raggiunta prima di poter ottenere qualsiasi altro livello di conformità. Da allora sono stati specificati altri otto tipi di conformazioni avanzate, comprese Active Database, Enhanced Integrity Management e Basic Object Support. Al momento della stesura di questo libro, la maggior parte dei sistemi di gestione delle basi di dati è di livello Entry compatibile con SQL-92 ed è predisposta per la compatibilità con Core SQL-99. Per esempio Oracle 8i supporta Entry SQL-92 ed è in gran parte compatibile con Core SQL-99. Gli esempi di questo capitolo sono conformi allo standard SQL, tranne quando indicati come SQL di Oracle e SQL di Microsoft Access.

SQL è stato implementato nei sistemi sia mainframe sia di personal computer; il capitolo è quindi importante per entrambi gli ambienti. Sebbene molti dei pacchetti di basi di dati per PC utilizzino un'in-



terfaccia QBE (*Query-by-Example*), includono anche un'opzione SQL. In Microsoft Access, per esempio, è possibile spostarsi avanti e indietro tra le due interfacce; un'interrogazione che è stata costruita utilizzando un'interfaccia QBE può essere visualizzata in SQL facendo semplicemente clic su un pulsante. Questa funzione può aiutare il lettore a imparare la sintassi SQL. Nelle architetture client/server, i comandi SQL vengono eseguiti sul server e i risultati sono restituiti sulla stazione di lavoro client.

Il primo DBMS commerciale che ha supportato SQL è stato Oracle, nel 1979. Oracle è ora disponibile in piattaforme mainframe, client/server e PC di molti sistemi operativi, compresi OS/390 (MVS), diversi sistemi operativi UNIX, Linux, Microsoft Windows e Windows NT, VAX/VMS e VM/CMS. Anche DB2 di IBM, Informix e Sybase sono disponibili per questa gamma di sistemi operativi. Microsoft SQL Server 2000 lavora in Windows NT e Windows 2000; la Personal Edition lavora in Windows 98, Windows NT, Windows 2000 e Windows Me.

Core SQL-99 contiene Entry SQL-92 e alcune funzionalità supplementari e questo è il motivo per il quale i produttori stanno passando da Entry SQL-92 alla compatibilità con Core SQL-99. Lo standard è stato pubblicato in molti documenti; questi specificano i livelli avanzati che possono essere implementati oltre a Core SQL-99 e abilitano finalmente SQL affinché venga utilizzato con oggetti complessi e persistenti nelle basi di dati di oggetti, oltre che in quelle relazionali correntemente supportate. In questo capitolo si utilizzerà Oracle 8i SQL\*Plus, che è compatibile con Entry SQL-92 e che sta passando a Core SQL-99. Al momento della stesura di questo libro, non è ancora compatibile con Core SQL-99, ma Oracle ha annunciato che lo sarà presto.

## 9.1 Storia dello standard SQL

I concetti della tecnologia delle basi di dati sono stati articolati per la prima volta nel 1970 in un documento scritto da E. F. Codd intitolato *A Relational Model of Data for Large Shared Data Banks*. I tecnici del Research Laboratory IBM di San Jose, California, intrapresero lo sviluppo di System R, un progetto il cui scopo era quello di dimostrare la fattibilità dell'implementazione del modello relazionale in un sistema di gestione delle basi di dati. I tecnici utilizzavano un linguaggio chiamato Sequel, anch'esso sviluppato presso lo stesso laboratorio di ricerca. Sequel fu rinominato SQL durante lo svolgimento del progetto, che si prolungò dal 1974 al 1979. Quanto appreso venne applicato allo sviluppo di SQL/DS, il primo sistema di gestione di basi di dati reso commercialmente disponibile da IBM. SQL/DS fu reso disponibile per la prima volta nel 1981, e lavorava sul sistema operativo DOS/VSE. Seguì una versione VM nel 1982 e nel 1983 fu annunciata la versione MVS, DB2.

Poiché System R si era ben adattato ai siti utente sui quali era stato installato, altri produttori iniziarono a sviluppare prodotti relazionali che utilizzavano SQL. Uno di questi prodotti, Oracle, di Relational Market, fu immesso sul mercato prima di SQL/DS (1979). Altri prodotti includevano INGRES di Relational Technology (1981), IDM di Britton-Lee (1982), DG/SQL di Data General Corporation (1984) e Sybase di Sybase Inc. (1986). Per fornire alcune indicazioni per lo sviluppo dei RDBMS, ANSI e ISO approvarono uno standard per il linguaggio (funzioni e sintassi) di interrogazione relazionale SQL, proposto originariamente dall'X3H2 Technical Committee on Database (Technical Committee X3H2, Database, 1986; ISO, 1987), spesso chiamato SQL/86. Gli standard del 1986 sono stati estesi per includere una funzione IEF (*Integrity Enhancement Feature*) facoltativa, chiamata anche SQL/89. Uno standard correlato, Database Language Embedded SQL, fu adottato negli Stati Uniti nel 1989. I comitati ISO e ANSI crearono SQL-92 (Technical Committee X3H2, Database, 1989; ISO, 1989, 1991), che era un'ulteriore estensione di SQL/86. Questo standard fu ratificato nell'ultima parte del 1992 ed è ora noto come International Standard ISO/IEC 9075, 1992, Database Language SQL. Fu revisionato nel 1994 e nel 1996. SQL-99 fu ratificato nel luglio 1999.

Attualmente sono disponibili numerosi prodotti che supportano SQL e lavorano su macchine di tutte le dimensioni, da piccoli personal computer a grandi mainframe. Il mercato delle basi di dati sta crescendo e la frequenza delle modifiche apportate ai prodotti potrà subire un rallentamento, anche se le basi di dati continueranno a essere fondate su SQL. Nel 1999, Oracle, IBM e Microsoft raccolsero insieme il 73 per cento del mercato complessivo delle basi di dati (Gartner Dataquest, 3 maggio 1999), con Oracle e IBM che controllavano ognuna circa il 30 per cento del mercato. La crescita recente delle applicazioni di commercio elettronico e dei sistemi di gestione delle relazioni con i clienti (CRM) ha favorito la crescita dei produttori principali di basi di dati, ma secondo Gartner Dataquest, esistono ancora delle opportunità per i produttori più piccoli di prosperare attraverso sistemi specifici per i diversi settori o applicazioni di nicchia. Le prossime release dei prodotti potrebbero modificare la forza dei sistemi di gestione delle basi di dati già nel momento in cui questo libro viene pubblicato. Ciononostante, tutte continueranno a utilizzare SQL e si adegueranno, entro certi limiti, agli standard descritti di seguito.

## 9.2 Il ruolo di SQL in un'architettura di base di dati

Con i DBMS e i generatori di applicazioni più moderni, gli utenti delle applicazioni possono anche non rilevare immediatamente l'importanza di SQL nell'architettura delle basi di dati. Molti utenti accedono alle applicazioni delle basi di dati senza alcuna conoscenza di SQL. Per esempio, i siti Web consentono agli utenti di navigare nel catalogo del sito che si visita (si veda, per esempio, <http://www.llbean.com>).

Le informazioni su ogni articolo presentato, come dimensione, colore, descrizione o disponibilità, vengono memorizzate in una base di dati. Le informazioni vengono recuperate utilizzando un'interrogazione SQL, ma l'utente non ha eseguito direttamente un comando SQL.

Un'applicazione di base di dati relazionale SQL implica un'interfaccia utente, un insieme di tabelle nella base di dati e una sistema di gestione di basi di dati relazionali con una funzionalità SQL. All'interno del RDBMS, SQL verrà utilizzato per creare tabelle, tradurre richieste degli utenti, mantenere il dizionario dei dati e il catalogo di sistema, aggiornare e mantenere le tabelle, stabilire la protezione ed eseguire le procedure di backup o ripristino.

Un DBMS relazionale (RDBMS) è un sistema di gestione dei dati che implementa un modello di dati relazionale, nel quale i dati sono memorizzati in una raccolta di tabelle e le relazioni tra i dati sono rappresentate da valori comuni, non da collegamenti. Questa vista dei dati è stata illustrata nel Capitolo 5 per il sistema di base di dati di Pine Valley Furniture e verrà utilizzata in tutte le interrogazioni SQL di esempio di questo capitolo.

Quelli che seguono sono gli obiettivi originari dello standard SQL.

1. Specificare la sintassi e la semantica dei linguaggi SQL di definizione e manipolazione dei dati.
2. Definire le strutture dei dati e le operazioni di base per la progettazione, l'accesso, il mantenimento, il controllo e la protezione di una base di dati SQL.
3. Fornire un veicolo per la portabilità della definizione della base dei dati e dei moduli applicativi tra DBMS conformi.
4. Specificare gli standard minimo (Livello 1) e completo (Livello 2), che permettono gradi diversi di adozione nei prodotti.

5. Fornire uno standard iniziale, per quanto incompleto, che verrà migliorato in seguito per includere specifiche per la gestione di argomenti quali l'integrità referenziale, la gestione delle transazioni, le funzioni definite dall'utente, gli operatori di join oltre l'equi-join e i set di caratteri nazionali (tra gli altri).

Lo standard SQL-92 è stato ampiamente accettato e la maggior parte dei nuovi prodotti è compatibile con SQL-92 e sta passando alla compatibilità con Core SQL-99. Tuttavia, la versione SQL di ciascun produttore include anche miglioramenti, funzioni e potenzialità che estendono le diverse versioni oltre gli standard di base di SQL-92. Per esempio, Oracle include un comando DESCRIBE, che elenca tutti gli attributi, i loro tipi di dati e i vincoli per una tabella. Si tratta di una funzione molto utile, soprattutto per gli studenti che devono familiarizzare con una base di dati dimostrativa, ma è un comando che non si trova se si utilizza una versione SQL di un altro produttore, come Microsoft Access. Gli standard SQL sono sempre stati dotati di un insieme di funzionalità ridotto al minimo piuttosto che di uno completamente specificato. Quali sono i vantaggi e gli svantaggi dell'avere uno standard SQL?

Tra i vantaggi di un linguaggio relazionale standardizzato vi sono i seguenti.

- *Costi di formazione ridotti.* La formazione in un'azienda può concentrarsi su un linguaggio. Avere un gruppo di professionisti dei sistemi informativi tutti formati con un linguaggio comune può ridurre i tempi della formazione dei nuovi assunti.
- *Produttività.* I professionisti dei sistemi informativi possono imparare a fondo SQL e diventare sempre più esperti continuando a utilizzarlo. L'azienda può permettersi di investire in strumenti che aiutino i professionisti dei sistemi informativi a diventare più produttivi. E poiché hanno familiarità con il linguaggio nel quale sono scritti i programmi, i programmatori possono mantenere in modo più rapido i programmi esistenti.
- *Portabilità delle applicazioni.* Le applicazioni possono essere spostate da macchina a macchina se ogni macchina utilizza SQL. Inoltre, è economico anche per l'industria del software sviluppare programmi applicativi standardizzati quando c'è un linguaggio standard.
- *Longevità delle applicazioni.* Un linguaggio standard tende a rimanere tale per lungo tempo; non si subiranno quindi pressioni per riscrivere applicazioni vecchie. Le applicazioni verranno invece semplicemente aggiornate man mano che il linguaggio standard viene migliorato o che nuove versioni di DBMS vengono introdotte.
- *Dipendenza ridotta da un unico produttore.* Quando si utilizza un linguaggio non proprietario, è più facile rivolgersi a produttori diversi per il DBMS, i servizi di formazione, il software applicativo e la consulenza; il mercato di questi produttori diventa così più competitivo, cosa che può ridurre i prezzi e migliorare i servizi.
- *Comunicazione a sistemi incrociati.* DBMS e programmi applicativi diversi possono comunicare e collaborare più facilmente nella gestione dei dati e nell'elaborazione dei programmi degli utenti.

Il rovescio della medaglia è che uno standard può soffocare la creatività e l'innovazione; uno standard non è mai sufficiente a soddisfare tutte le esigenze, e uno standard di settore può essere molto lontano dall'ideale, poiché può essere il frutto di compromessi tra più parti. Uno standard può essere difficile da modificare (dato che molti produttori vi investono parecchio), quindi correggere le eventuali lacune può richiedere molti sforzi. Un altro svantaggio dello standard è che utilizzare le funzioni speciali aggiunte a SQL da un determinato produttore può comportare la perdita di alcuni vantaggi, come la portabilità delle applicazioni.

Lo standard SQL originale è stato ampiamente criticato per la mancanza di regole di integrità referenziale e di alcuni operatori. Date e Darwen (1997) esprimono la preoccupazione che SQL sembra

essere stato concepito senza aderire ai principi stabiliti della progettazione dei linguaggi; il risultato è che “il linguaggio contiene numerose restrizioni, costrutti ad hoc e regole speciali fastidiose” (p. 8). I due autori temono che lo standard non sia sufficientemente esplicito e che il problema delle implementazioni dell'SQL standard continuerà a esistere. Alcuni di questi limiti verranno messi in evidenza in questo capitolo.

## 9.3 L'ambiente SQL

La Figura 9.1 è una rappresentazione schematica semplificata di un ambiente SQL, compatibile con gli standard SQL-92. Come illustrato, un ambiente SQL include un'istanza di un DBMS SQL insieme alle basi di dati accessibili da quel DBMS e agli utenti e programmi che possono utilizzare quel DBMS per accedere alle basi di dati. Ogni base di dati è contenuta in un **catalogo**, che descrive qualsiasi oggetto che fa parte della base di dati, a prescindere da quale utente ha creato l'oggetto. La Figura 9.1 mostra due cataloghi: DEV\_C e PROD\_C. La maggior parte delle società mantiene almeno due versioni di tutte le basi di dati utilizzate. La versione di produzione, in questo caso PROD\_C, è la versione *live*, che rappresenta i dati aziendali reali e che deve quindi essere strettamente controllata e monitorata. La versione di sviluppo, in questo caso DEV\_C, è utilizzata quando la base di dati è stata costruita e continua a operare con strumenti di sviluppo nei quali è possibile testare approfonditamente gli sforzi di miglioramento e di manutenzione prima che vengano applicati alla base di dati di produzione. Questa base di dati non è solitamente monitorata né tenuta sotto stretto controllo, poiché non contiene dati aziendali *live*. Ogni base di dati ha degli schemi denominati associati a un catalogo. Uno **schema** è una raccolta di oggetti correlati, compresi tabelle di base e viste, domini, vincoli, set di caratteri, trigger, ruoli e così via.

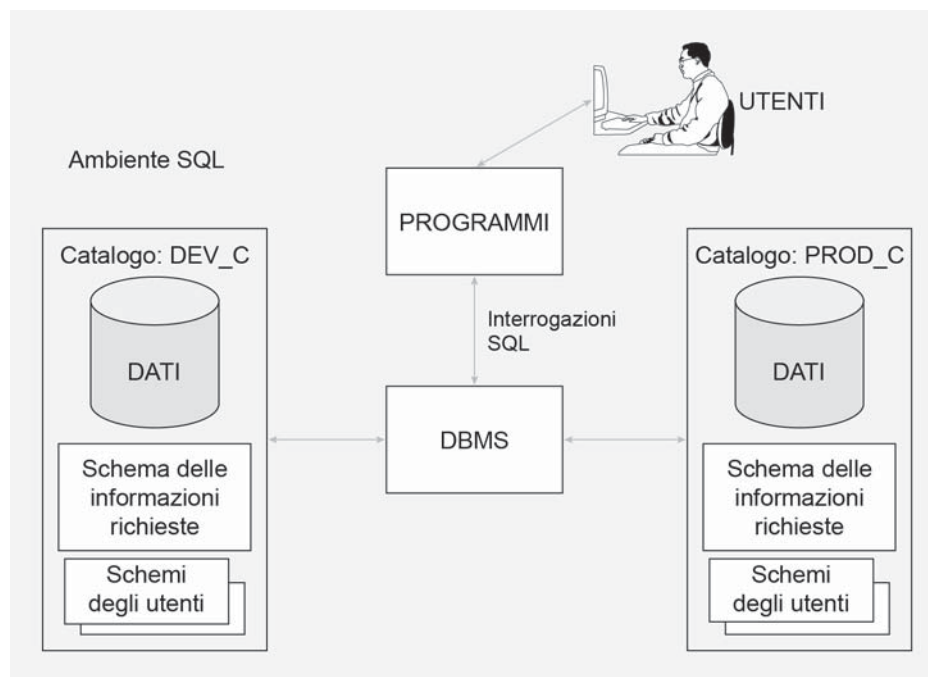
Se più di un utente ha creato oggetti nella base di dati, combinare le informazioni sugli schemi di tutti gli utenti fornirà informazioni per l'intera base di dati. Ogni catalogo deve contenere anche uno **schema delle informazioni, che contiene descrizioni di tutti gli schemi nel catalogo, le tabelle, le viste, gli attributi, i privilegi, i vincoli e i domini, insieme ad altre informazioni importanti per la base di dati.** Le informazioni nel catalogo sono mantenute dal DBMS come risultato dei comandi SQL eseguiti dagli utenti e non richiedono un'azione diretta dell'utente per la loro costruzione. Fa parte della potenza del linguaggio SQL che la capacità di produrre comandi SQL sintatticamente semplici possa far sì che le attività di gestione dei dati complessi vengano portate avanti dal software DBMS. Gli utenti possono sfogliare il contenuto del catalogo utilizzando istruzioni SQL di selezione.

I comandi SQL possono essere classificati in tre tipi. Ci sono innanzitutto **i comandi del linguaggio di definizione dei dati (DDL).** Questi comandi sono utilizzati per creare, modificare ed eliminare le tabelle e verranno trattati per primi in questo capitolo. In una base di dati di produzione, la capacità di utilizzare viene solitamente riservata a uno o più amministratori della base di dati, così da proteggere la struttura della base di dati da modifiche non previste. Nelle basi di dati di sviluppo o di esercizio, i privilegi DDL vengono concessi a più utenti.

In secondo luogo ci sono **i comandi del linguaggio di manipolazione dei dati (DML).** Molti considerano i comandi DML il cuore dei comandi SQL. Sono infatti utilizzati per aggiornare, inserire, modificare e interrogare i dati nella base di dati. Possono essere eseguiti interattivamente, così che il risultato venga restituito immediatamente dopo l'esecuzione dell'istruzione, oppure possono essere inclusi nei programmi scritti in un linguaggio di terza generazione, come C o COBOL. L'incorporazione dei comandi SQL può fornire al programmatore un controllo maggiore sui tempi della generazione dei report, sull'aspetto dell'interfaccia, sulla gestione degli errori e sulla protezione della base di dati. Gran parte di questo capitolo è dedicata ai comandi DML di base nel formato interattivo.

**Figura 9.1**

Una rappresentazione schematica semplificata di un ambiente SQL, come descritto dallo standard SQL-92.



La sintassi generale del comando SELECT SQL utilizzato in DML è mostrata nella Figura 9.2.

**Figura 9.2**

Sintassi generale dell'istruzione SELECT utilizzata nel linguaggio di manipolazione dei dati.

```

SELECT [ALL DISTINCT] elenco_colonne
FROM elenco_tabelle
[WHERE espressione condizionale]
[GROUP BY raggruppa_per_elenco_colonne]
[HAVING espressione condizionale]
[ORDER BY ordina_per_elenco_colonne]

```

Gli ultimi **comandi sono quelli del linguaggio di controllo dei dati (DCL)**. Questi comandi aiutano l'amministratore della base di dati a controllare la base di dati; includono comandi che concedono e revocano i privilegi di accesso alla base di dati o a particolari oggetti al suo interno e di memorizzare o eliminare quelle transazioni che potrebbero influire sulla base di dati.

Ogni DBMS ha un elenco definito di tipi di dati che può gestire. Tutti contengono variabili numeriche, di stringa e di data/ora. Alcuni contengono anche tipi di dati grafici, spaziali o di immagine, cosa che aumenta notevolmente la flessibilità della manipolazione dei dati. Quando una tabella viene creata, deve essere specificato il tipo di dati per ciascun attributo. Un prezzo unitario dovrà essere memorizzato in un formato numerico, poiché sono previste manipolazioni matematiche come la moltiplicazione di un prezzo unitario per il numero di unità ordinate. Un numero di telefono può essere memorizzato come dati di stringa, soprattutto se nell'insieme di dati dovranno essere inclusi numeri telefonici este-

ri. Sebbene i numeri di telefono contengano solo cifre, operazioni matematiche come la somma o la moltiplicazione non hanno alcun senso. Poiché i dati di carattere vengono elaborati più rapidamente, i dati numerici sui quali non si devono effettuare operazioni matematiche possono essere memorizzati come dati di carattere. La selezione di un campo invece che di una stringa permette allo sviluppatore di sfruttare i vantaggi delle funzioni di calcolo dell'intervallo data/ora che non possono essere applicate a un campo di carattere. Si veda la Tabella 9.1.

**Tabella 9.1** Tipi di dati di esempio di Oracle8.

|          |                               |                                                                                                      |
|----------|-------------------------------|------------------------------------------------------------------------------------------------------|
| Stringa  | CHAR( <i>n</i> )              | Dati di carattere a una lunghezza fissa di <i>n</i> caratteri. La lunghezza massima è 2000.          |
|          | VARCHAR2( <i>n</i> )          | Dati di carattere a lunghezza variabile. Dimensione massima di 4000 byte.                            |
|          | LONG                          | Dati di carattere a lunghezza variabile. Dimensione massima di 4 GB. Al massimo uno per tabella.     |
| Numerici | NUMBER( <i>p</i> , <i>q</i> ) | Numero decimale con <i>p</i> cifre e alcune cifre presunte <i>q</i> alla virgola decimale da destra. |
|          | INTEGER( <i>p</i> )           | Intero, decimale o binario, grande <i>p</i> cifre.                                                   |
|          | FLOAT( <i>p</i> )             | Numero a virgola mobile nella notazione scientifica di precisione di <i>p</i> cifre binarie.         |
| Data/ora | DATE                          | Data e ora a lunghezza fissa nella forma <i>gg-mm-aa</i> .                                           |

Con l'avvento dei vari tipi di dati grafici e di immagine, è necessario considerare le esigenze aziendali rispetto a dove memorizzare i dati. Per esempio, il colore potrebbe essere memorizzato come campo di carattere descrittivo, come 'sand drift' o 'beige'. Tali descrizioni possono però variare da un produttore a un altro e non contengono l'entità delle informazioni che potrebbero essere contenute in un tipo di dati spaziale che include i valori di intensità esatti di rosso, verde e blu. Questo tipo di dati è ora disponibile sui server universali, che gestiscono i data warehouse, ed è molto probabile che appaiano presto anche nei RDBMS. SQL-92 e SQL-99 supportano numerosi tipi di dati che non sono mostrati nella Tabella 9.1, come `TIMESTAMP`, `REAL` e `INTERVAL`, mentre altri RDBMS hanno incluso dati come `CURRENCY` e `LOGICAL`. Sarà necessario prendere confidenza con i tipi di dati disponibili per ogni RDBMS con cui si lavora, così da trarre il massimo vantaggio dalle sue potenzialità.

Siamo quasi pronti per illustrare qualche comando SQL di esempio. I dati di esempio che verranno utilizzati sono mostrati nella Figura 9.3. Ogni nome di tabella segue uno standard di denominazione che colloca un underscore e la lettera `t` al termine del nome di ciascuna tabella, come in `Order_t` o `Product_t`. Mentre si osservano queste tabelle, si consideri quanto segue.

1. Ogni ordine deve avere un numero di cliente valido incluso nella tabella `Order_t`.
2. Ogni articolo in un ordine deve avere sia un numero di prodotto valido sia un numero di ordine valido associato nella tabella `Order_Line_t`.
3. Queste quattro tabelle rappresentano una versione semplificata di uno degli insiemi di relazioni più diffusi nei sistemi di basi di dati aziendali, quello degli ordini dei clienti per i prodotti. I comandi SQL necessari a creare le tabelle `Customer` e `Order` sono inclusi nel Capitolo 5 e sono approfonditi in questo capitolo.

Il resto del capitolo illustrerà i comandi DDL, DML e DCL. La Figura 9.4 offre una panoramica dei vari tipi di comandi utilizzati nel processo di sviluppo della base di dati. Nei comandi SQL di esempio utilizzeremo la seguente notazione.

1. Le parole in maiuscolo indicano la sintassi del comando. Devono essere digitate esattamente come mostrato, sebbene la capitalizzazione possa non essere richiesta dal RDBMS.

**Figura 9.3**  
Dati di esempio  
dell'azienda Pine Valley  
Furniture.

Microsoft Access - [CUSTOMER\_t : Table]

File Edit View Insert Format Records Tools Window Help

| Customer_ID    | Customer_Name            | Customer_Address     | City          | State | Postal_Code |
|----------------|--------------------------|----------------------|---------------|-------|-------------|
| 1              | Contemporary Casuals     | 1355 S Hines Blvd    | Gainesville   | FL    | 32601-      |
| 2              | Value Furniture          | 15145 S.W. 17th St.  | Plano         | TX    | 75094-      |
| 3              | Home Furnishings         | 1900 Allard Ave.     | Albany        | NY    | 12209-      |
| 4              | Eastern Furniture        | 1925 Beltline Rd.    | Carteret      | NJ    | 07008-      |
| 5              | Impressions              | 5585 Westcott Ct.    | Sacramento    | CA    | 94206-      |
| 6              | Furniture Gallery        | 325 Flatiron Dr.     | Boulder       | CO    | 80514-      |
| 7              | Period Furniture         | 394 Rainbow Dr.      | Seattle       | WA    | 97954-      |
| 8              | California Classics      | 816 Peach Rd.        | Santa Clara   | CA    | 96915-      |
| 9              | M & H Casual Furniture   | 3709 First Street    | Clearwater    | FL    | 34620-      |
| 10             | Seminole Interiors       | 2400 Rocky Point Dr. | Seminole      | FL    | 34646-      |
| 11             | American Euro Lifestyles | 2424 Missouri Ave N. | Prospect Park | NJ    | 07508-      |
| 12             | Battle Creek Furniture   | 345 Capitol Ave. SW  | Battle Creek  | MI    | 49015-      |
| 13             | Heritage Furnishings     | 66789 College Ave.   | Carlisle      | PA    | 17013-      |
| 14             | Kaneohe Homes            | 112 Kiowai St.       | Kaneohe       | HI    | 96744-      |
| 15             | Mountain Scenes          | 4132 Main Street     | Ogden         | UT    | 84403-      |
| * (AutoNumber) |                          |                      |               |       |             |

Record: 1 of 15

Unique number to identify customer

Customer\_t

|     | Order_Id | Product_Id | Ordered_Quan |
|-----|----------|------------|--------------|
| ▶ + | 1001     | 1          | 2            |
| +   | 1001     | 2          | 2            |
| +   | 1001     | 4          | 1            |
| +   | 1002     | 3          | 5            |
| +   | 1003     | 3          | 3            |
| +   | 1004     | 6          | 2            |
| +   | 1004     | 8          | 2            |
| +   | 1005     | 4          | 4            |
| +   | 1006     | 4          | 1            |
| +   | 1006     | 5          | 2            |
| +   | 1006     | 7          | 2            |
| +   | 1007     | 1          | 3            |
| +   | 1007     | 2          | 2            |
| +   | 1008     | 3          | 3            |
| +   | 1008     | 8          | 3            |
| +   | 1009     | 4          | 2            |
| +   | 1009     | 7          | 3            |
| +   | 1010     | 8          | 10           |
| *   | 0        | 0          | 0            |

Order\_Line\_t

|     | Order_Id | Order_Date | Customer_ID |
|-----|----------|------------|-------------|
| ▶ + | 1001     | 10/21/2000 | 1           |
| +   | 1002     | 10/21/2000 | 8           |
| +   | 1003     | 10/22/2000 | 15          |
| +   | 1004     | 10/22/2000 | 5           |
| +   | 1005     | 10/24/2000 | 3           |
| +   | 1006     | 10/24/2000 | 2           |
| +   | 1007     | 10/27/2000 | 11          |
| +   | 1008     | 10/30/2000 | 12          |
| +   | 1009     | 11/5/2000  | 4           |
| +   | 1010     | 11/5/2000  | 1           |
| *   | 0        |            | 0           |

Order\_t

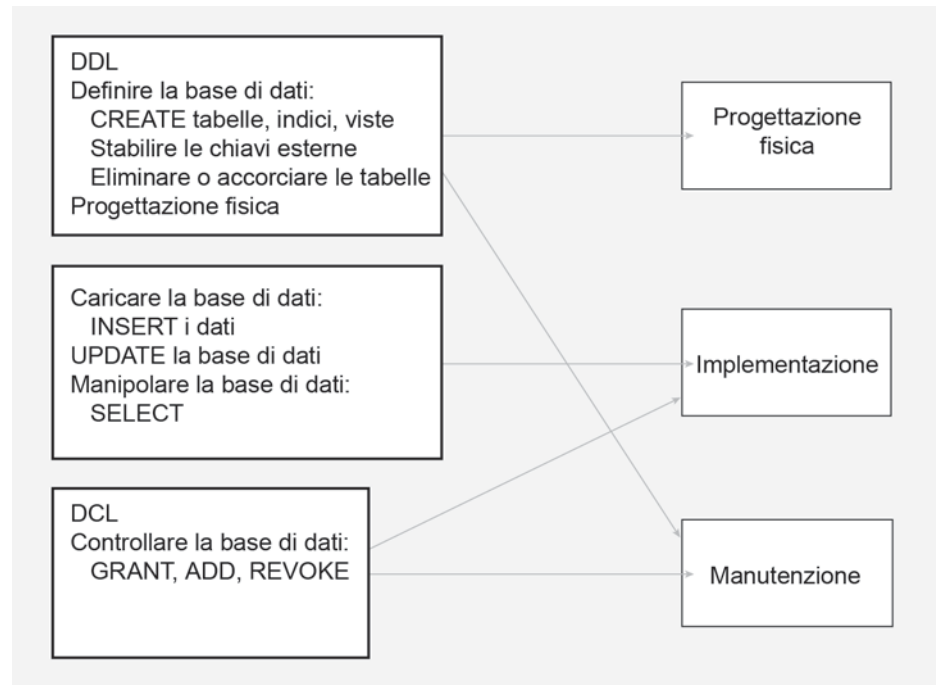
|     | Product_ID   | Product_Description  | Product_Finish | Standard_Price | Product_Line_Id |
|-----|--------------|----------------------|----------------|----------------|-----------------|
| ▶ + | 1            | End Table            | Cherry         | \$175.00       | 10001           |
| +   | 2            | Coffee Table         | Natural Ash    | \$200.00       | 20001           |
| +   | 3            | Computer Desk        | Natural Ash    | \$375.00       | 20001           |
| +   | 4            | Entertainment Center | Natural Maple  | \$650.00       | 30001           |
| +   | 5            | Writer's Desk        | Cherry         | \$325.00       | 10001           |
| +   | 6            | 8-Drawer Dresser     | White Ash      | \$750.00       | 20001           |
| +   | 7            | Dining Table         | Natural Ash    | \$800.00       | 20001           |
| +   | 8            | Computer Desk        | Walnut         | \$250.00       | 30001           |
| *   | (AutoNumber) |                      |                | \$0.00         |                 |

Product\_t

- 2. Le parole in minuscolo indicano i valori che devono essere forniti dall'utente.
- 3. Le parentesi racchiudono la sintassi facoltativa.
- 4. Le ellissi (...) indicano che la clausola sintattica associata può essere ripetuta, se necessario.
- 5. Ogni comando SQL termina con un punto e virgola (;). Nella modalità interattiva, quando l'utente preme il tasto INVIO, il comando SQL viene eseguito. Possono esserci delle convenzioni alternati-

**Figura 9.4**

DDL, DML, DCL e il processo di sviluppo della base di dati.



ve, come digitare GO o includere un simbolo di prosecuzione, come un trattino, alla fine di ogni riga utilizzata nel comando. Le spaziature e i rientri qui mostrati sono inclusi per favorire la leggibilità e non sono richiesti come parte della sintassi SQL.

## 9.4 Definire una base di dati in SQL

La maggior parte dei sistemi alloca uno spazio di memoria che deve contenere tabelle, viste, vincoli, indici e altri oggetti della base di dati quando questa viene creata. Per questa ragione, il privilegio di creare le basi di dati può essere riservato all'amministratore, al quale ci si può rivolgere perché ne sia creata una per le proprie esigenze. Agli studenti di un'università può essere assegnato un account che permetta di accedere a una base di dati esistente o possono essere autorizzati a creare la propria base di dati. In ogni caso, la sintassi di base per creare una base di dati è `CREATE SCHEMA nome_base di dati; AUTHORIZATION id utente_proprietario`. La base di dati sarà posseduta dall'utente autorizzato, sebbene sia possibile per altri utenti specifici lavorare con essa o anche trasferirne la proprietà. La memorizzazione fisica della base di dati dipende dall'ambiente hardware e software e di solito è compito dell'amministratore di sistema. L'entità del controllo sulla memorizzazione fisica esercitato da un amministratore dipende dal RDBS utilizzato. Con Microsoft Access 2000 il controllo possibile è molto ridotto, mentre un amministratore di una base di dati Oracle 8i può esercitare un controllo considerevole sul posizionamento dei dati, dei file dei controlli, dei file di indice e così via, perfezionando la base di dati in modo che lavori in modo più efficiente.



## 9.4.1 Generare definizioni SQL per la base di dati

I comandi CREATE DDL SQL sono inclusi in SQL-92 livello Entry.

|               |                                                                                                                                                                                                                                                                                                                                        |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREATE SCHEMA | Utilizzato per definire la porzione di una base di dati posseduta da un determinato utente. Gli schemi sono dipendenti da un catalogo e contengono oggetti di schema, come tabelle di base e viste, domini, vincoli, asserzioni, set di caratteri, collazioni e così via.                                                              |
| CREATE TABLE  | Definisce una nuova tabella e le sue colonne. La tabella può essere di base o derivata. Le tabelle dipendono da uno schema. Le tabelle derivate vengono create eseguendo un'interrogazione che utilizza una o più tabelle o viste.                                                                                                     |
| CREATE VIEW   | Definisce una tabella logica o più tabelle o viste. Le viste possono non essere indicizzate. Esistono alcuni limiti all'aggiornamento dei dati in una vista. Quando le viste possono essere aggiornate, le modifiche possono essere trasferite alle tabelle sottostanti a cui si è fatto inizialmente riferimento per creare la vista. |

Ognuno di questi comandi CREATE può essere invertito utilizzando un comando DROP. Così DROP TABLE distruggerà una tabella, comprese le sue definizioni, il contenuto e gli eventuali vincoli, viste o indici associati. Solitamente solo il creatore della tabella può eliminarla. DROP SCHEMA e DROP VIEW distruggono lo schema o la vista denominati. ALTER TABLE può essere utilizzato per modificare la definizione di una base di dati esistente aggiungendo, eliminando o modificando una colonna o eliminando un vincolo.

Ci sono altri cinque comandi CREATE inclusi nello standard SQL-92. A nessuno di questi è richiesto di soddisfare gli standard dei livelli Entry o Intermediate di SQL-92.

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREATE CHARACTER SET | Permette all'utente di definire un set di caratteri per le stringhe di testo e favorisce la globalizzazione di SQL abilitando l'uso di lingue diverse dall'inglese. Ogni set di caratteri contiene un insieme di caratteri, un modo di rappresentare ciascun carattere internamente, un formato di dati utilizzato per questa rappresentazione e una collazione, ovvero un modo per ordinare il set di caratteri. |
| CREATE COLLATION     | Un oggetto di schema denominato che specifica l'ordine che verrà assunto dal set di caratteri. Le collazioni esistenti possono essere manipolate per creare una nuova collazione.                                                                                                                                                                                                                                 |
| CREATE TRANSLATION   | Un insieme denominato di regole che mappa i caratteri da un set di caratteri d'origine in uno di destinazione per scopi di traduzione o conversione.                                                                                                                                                                                                                                                              |
| CREATE ASSERTION     | Un oggetto di schema che stabilisce un vincolo CHECK che viene violato se il vincolo è falso.                                                                                                                                                                                                                                                                                                                     |
| CREATE DOMAIN        | Un oggetto di schema che stabilisce un dominio, o un insieme di valori validi, per un attributo. Verrà specificato il tipo di dati e, se si desidera, possono essere specificati anche un valore predefinito, una collazione o altri vincoli.                                                                                                                                                                     |

## 9.4.2 Creare le tabelle

Una volta che il modello di dati è stato progettato e normalizzato, è possibile definire le colonne necessarie per ogni tabella utilizzando il comando CREATE TABLE SQL. La sintassi generale per questo

**Figura 9.5**

Sintassi generale dell'istruzione  
CREATE TABLE utilizzata nel  
linguaggio di definizione dei dati.

```
CREATE TABLE nometabella
({definizione di colonna [vincolo di tabella] } . . .
[ON COMMIT {DELETE | PRESERVE} ROWS]);

dove definizione di colonna ::=
nome_colonna
{nome dominio | tipo di dati [(dimensione)] }
[clausola_vincolo_colonna...]
[valore predefinito]
[clausola di collazione]
e vincolo di tabella ::=
[CONSTRAINT nome_vincolo]
Tipo_vincolo [attributi_vincolo]
```

comando è mostrata nella Figura 9.5. Ecco una serie di passaggi da seguire quando ci si prepara a creare una tabella.

1. Identificare il tipo di dati opportuno, comprendendo la lunghezza, la precisione e la scala, se necessari, per ciascun attributo.
2. Identificare quelle colonne che potranno accettare dei valori nulli, come spiegato nel Capitolo 8. I controlli di colonna che indicano che questa non può essere nulla vengono impostati quando una tabella viene creata e sono applicati a ogni aggiornamento della tabella quando i dati vengono immessi.
3. Identificare le colonne che devono essere univoche. Quando un controllo di univocità (UNIQUE) viene stabilito per una colonna, i dati in quella colonna devono avere un valore diverso (non possono cioè essere duplicati) per ogni riga di dati nella tabella. Una colonna o insieme di colonne contrassegnate come UNIQUE diventano chiavi candidate, come visto nel Capitolo 7. Sebbene ogni tabella di base possa avere più chiavi candidate, solo una di queste può essere quella primaria (PRIMARY KEY). Quando una o più colonne sono specificate come PRIMARY KEY, si presume anche che queste non siano nulle, anche se NOT NULL non è specificato. UNIQUE e PRIMARY KEY sono entrambi vincoli di colonne.
4. Identificare tutte le coppie chiave primaria-chiave esterna, come discusso nel Capitolo 7. Le chiavi esterne possono essere stabilite immediatamente, quando una tabella viene creata, o successivamente, modificando la tabella. La tabella genitore in questa relazione genitore-figlio dovrebbe essere creata per prima, così che la tabella figlio possa fare riferimento a una tabella genitore esistente quando viene creata. Il vincolo di colonna REFERENCES può essere utilizzato per applicare l'integrità referenziale.
5. Determinare i valori da inserire nelle colonne per le quali è preferibile un valore predefinito. In SQL-92, DEFAULT può essere utilizzato per definire un valore che viene inserito automaticamente quando non viene inserito alcun valore durante l'immissione dei dati. Nella Figura 9.6, il comando che crea la tabella ORDER\_T ha definito un valore predefinito di SYSDATE (il nome di Oracle per la data corrente) per l'attributo DATE.
6. Identificare le colonne per le quali possono essere definite delle specifiche di dominio che sono maggiormente vincolate di quelle stabilite dal tipo di dati. Utilizzando CHECK come vincolo di colonna in SQL-92, può essere possibile stabilire delle regole di convalida da inserire nella base di dati. Nella Figura 9.6, la tabella PRODUCT\_T include un vincolo di controllo, che elenca i valori possibili per

**Figura 9.6**

Comandi SQL di definizione della base di dati per Pine Valley Furniture Company (Oracle 8i).

```
CREATE TABLE CUSTOMER_T
(CUSTOMER_ID NUMBER(11, 0) NOT NULL,
CUSTOMER_NAME VARCHAR2(25) NOT NULL,
CUSTOMER_ADDRESS VARCHAR2(30),
CITY VARCHAR2(20),
STATE VARCHAR2(2),
POSTAL_CODE VARCHAR2(9),
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID));

CREATE TABLE ORDER_T
(ORDER_ID NUMBER(11, 0) NOT NULL,
ORDER_DATE DATE DEFAULT SYSDATE,
CUSTOMER_ID NUMBER(11, 0),
CONSTRAINT ORDER_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDER_FK FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID));

CREATE TABLE PRODUCT_T
(PRODUCT_ID INTEGER NOT NULL,
PRODUCT_DESCRIPTION VARCHAR2(50),
PRODUCT_FINISH VARCHAR2(20)
 CHECK (PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'White Ash',
 'Red Oak', 'Natural Oak', 'Walnut')),
STANDARD_PRICE DECIMAL(6,2),
PRODUCT_LINE_ID INTEGER,
CONSTRAINT PRODUCT_PK PRIMARY KEY (PRODUCT_ID));

CREATE TABLE ORDER_LINE_T
(ORDER_ID NUMBER(11,0) NOT NULL,
PRODUCT_ID NUMBER(11,0) NOT NULL,
ORDERED_QUANTITY NUMBER(11,0),
CONSTRAINT ORDER_LINE_PK PRIMARY KEY (ORDER_ID, PRODUCT_ID),
CONSTRAINT ORDER_LINE_FK1 FOREIGN KEY(ORDER_ID) REFERENCES ORDER_T(ORDER_ID),
CONSTRAINT ORDER_LINE_FK2 FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_T(PRODUCT_ID));
```

PRODUCT\_FINISH. Così, sebbene l'immissione di 'White Maple' potrebbe soddisfare i vincoli dei tipi di dati di carattere, verrebbe rifiutata, poiché 'White Maple' non è nella lista di controllo (*checklist*).

7. Creare la tabella e il numero di indici desiderati utilizzando le istruzioni CREATE TABLE e CREATE INDEX (CREATE INDEX non fa parte dello standard SQL-92 perché l'indicizzazione viene utilizzata per risolvere problemi di prestazione; è tuttavia disponibile nella maggior parte dei RDBMS).

Nei Capitoli 5 e 7, i comandi SQL di definizione della base di dati per creare le tabelle Customer e Order per Pine Valley Furniture includevano la definizione dei vincoli di colonna all'interno dei comandi CREATE TABLE. Nella Figura 9.6, i comandi di definizione della base di dati utilizzando Oracle 8i includono alcuni vincoli di colonna supplementari, e alle chiavi primarie ed esterne sono stati assegnati dei nomi. Per esempio, la chiave primaria della tabella Customer è CUSTOMER\_ID. Il vincolo di chiave primaria è chiamato CUSTOMER\_PK. Ora, per esempio, un amministratore che guarda la tabella DBA\_CONSTRAINTS in Oracle identificherà facilmente il vincolo di chiave primaria sulla tabella dei clienti, poiché il suo nome CUSTOMER\_PK, darà il valore della colonna *nome\_vincolo*. Senza il nome di vincolo, verrebbe assegnato automaticamente un identificatore di sistema, che sarebbe difficile da leggere.

### 9.4.3 Utilizzare e definire le viste

La sintassi SQL mostrata nella Figura 9.6 illustra la creazione di quattro **tabelle di base** in uno schema di base di dati. Queste tabelle, che vengono utilizzate per memorizzare fisicamente i dati nella base di dati, corrispondono alle entità nello schema concettuale. È possibile utilizzare interrogazioni SQL per creare tabelle virtuali, o viste dinamiche, il cui contenuto si materializza quando vi si fa riferimento. Queste viste possono spesso essere manipolate nello stesso modo di una tabella, attraverso le interrogazioni SELECT SQL. In alternativa possono essere utilizzate anche **viste materializzate**, che sono memorizzate fisicamente su un disco e che vengono aggiornate a intervalli opportuni o in corrispondenza di determinati eventi.

Lo scopo ben noto di una vista è quello di semplificare i comandi di interrogazione, ma una vista può anche fornire una protezione significativa dei dati e migliorare notevolmente la produttività nella programmazione di una base di dati. Per sottolineare l'utilità di una vista, si consideri la fattura di Pine Valley mostrata nella Figura 3.6. La costruzione di questa fattura richiede l'accesso alle quattro tabelle di base della base di dati di Pine Valley della Figura 9.3: CUSTOMER\_T, ORDER\_T, ORDER\_LINE\_T e PRODUCT\_T. Un utente principiante di una base di dati potrebbe compiere degli errori o non essere produttivo nel formulare interrogazioni efficienti che implicano così tante tabelle. Una vista permette di predefinire questa associazione in una singola tabella virtuale come parte della base di dati. Con una vista di questo tipo, un utente che vuole vedere solo i dati di una fattura di un cliente non deve ricostruire i join tra le tabelle per produrre il report o i suoi sottoinsiemi. La Tabella 9.2 riepiloga i pro e i contro dell'uso delle viste.

**Tabella 9.2** Pro e contro dell'uso delle viste dinamiche.

| Aspetti positivi                                                | Aspetti negativi                                                                                    |
|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| Semplificano i comandi di interrogazione                        | Utilizzano un lungo tempo di elaborazione per ricreare la vista ogni volta che vi si fa riferimento |
| Aiutano a fornire protezione e segretezza sui dati              |                                                                                                     |
| Migliorano la produttività del programmatore                    | Possono o possono non essere direttamente aggiornabili                                              |
| Contengono i dati correnti più aggiornati delle tabelle di base |                                                                                                     |
| Utilizzano poco spazio di memoria                               |                                                                                                     |
| Forniscono una vista personalizzata all'utente                  |                                                                                                     |
| Stabiliscono l'indipendenza fisica dei dati                     |                                                                                                     |

Una vista, E\_V, viene definita specificando un'interrogazione SQL (SELECT ... FROM ... WHERE) che ha la vista come risultato.

**Interrogazione:** Quali sono gli elementi di dati necessari a creare una fattura per un cliente? Salvare l'interrogazione come una vista chiamata INVOICE\_V.

```
CREATE VIEW INVOICE_V AS
 SELECT CUSTOMER_T.CUSTOMER_ID, CUSTOMER_ADDRESS,ORDER_T.
 ORDER_ID, PRODUCT_T.PRODUCT_ID, ORDERED_QUANTITY,
 e altre colonne necessarie
 FROM CUSTOMER_T, ORDER_T, ORDER_LINE_T, PRODUCT_T
 WHERE CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID
 AND ORDER_T.ORDER_ID = ORDER_LINE_T.ORDER_ID
 AND PRODUCT_T.PRODUCT_ID = ORDER_LINE_T.PRODUCT_ID;
```

La clausola **SELECT** specifica, o programma, quali elementi di dati (colonne) devono essere inclusi nella tabella della vista. La clausola **FROM** elenca le tabelle e le viste implicate nello sviluppo della vista. La clausola **WHERE** specifica i nomi delle colonne comuni utilizzate per unire **CUSTOMER\_T** a **ORDER\_T** a **ORDER\_LINE\_T** a **PRODUCT\_T**. Poiché una vista è una tabella, e una delle proprietà relazionali delle tabelle è che l'ordine delle righe non è materiale, le righe in una vista non possono essere ordinate. Tuttavia, le interrogazioni che si riferiscono a questa vista possono visualizzare i loro risultati in qualsiasi sequenza desiderata.

Possiamo vedere la potenza di una vista di questo tipo quando si crea un'interrogazione per generare una fattura per il numero d'ordine 1004. Invece di dover specificare l'unione delle quattro tabelle, l'interrogazione può includere tutti gli elementi di dati importanti dalla tabella della vista, **INVOICE\_V**.

**Interrogazione:** Quali sono gli elementi di dati necessari per creare una fattura per il numero d'ordine 1004?

```
SELECT CUSTOMER_ID, CUSTOMER_ADDRESS, PRODUCT_ID, QUANTITY,
e altre colonne necessarie
FROM INVOICE_V
WHERE ORDER_ID = 1004;
```

Una vista dinamica è una tabella virtuale; è costruita automaticamente come richiesto dal DBMS e non viene mantenuta come i dati reali. Per creare una tabella si può utilizzare qualsiasi istruzione **SELECT SQL**. I dati reali sono memorizzati nelle tabelle di base, quelle definite dai comandi **CREATE TABLE**. Una vista dinamica contiene sempre i valori derivati correnti più aggiornati ed è quindi superiore, in termini di aggiornamento dei dati, alla costruzione di una tabella reale temporanea da più tabelle di base. Rispetto a una tabella reale temporanea, inoltre, una vista consuma uno spazio di memoria molto ridotto. Una vista è tuttavia costosa, perché i suoi contenuti devono essere calcolati ogni volta che vengono richiesti. Attualmente sono disponibili le viste materializzate, che possono risolvere questo problema.

Una vista può unire più tabelle o viste tra loro e può contenere colonne derivate (o virtuali). Per esempio, se un utente della base di dati di Pine Valley Furniture vuole conoscere solo il valore totale degli ordini effettuati per ogni prodotto, è possibile creare una vista adatta da **INVOICE\_V** in Oracle **SQL\*Plus**.

**Interrogazione:** Qual è il valore totale degli ordini effettuati per ogni prodotto?

```
CREATE VIEW ORDER_TOTALS_V AS
SELECT PRODUCT_ID PRODUCT, SUM (UNIT_PRICE*QUANTITY) TOTAL
FROM INVOICE_V
GROUP BY PRODUCT_ID;
```

A una colonna della vista è possibile assegnare un nome diverso (un alias) di quello della tabella di base associata o di quello della colonna delle espressioni. In questo esempio, **PRODUCT** è l'alias di **PRODUCT\_ID**, ma solo per questa vista. **TOTAL** è il nome di colonna dato all'espressione delle vendite totali per ciascun prodotto. Nelle interrogazioni successive, è quindi possibile fare riferimento all'espressione attraverso questa vista invece che attraverso un'espressione derivata. Definire le viste in base ad altre viste può far sorgere dei problemi. Per esempio, se si ridefinisce **INVOICE\_V** di modo che **UNIT\_PRICE** non sia incluso, allora **ORDER\_TOTALS\_V** non funzionerà più, perché non sarà in grado di trovare i prezzi unitari.

Le viste possono anche aiutare a impostare una protezione dei dati. Le tabelle e le colonne che non sono incluse non saranno, ovviamente, visibili all'utente della vista. Limitare l'accesso a una vista con le istruzioni **GRANT** e **REVOKE** aggiunge un altro livello di protezione. Per esempio, se si concedono

ad alcuni utenti i privilegi di accesso ai dati aggregati (come le medie) in una vista, ma si nega loro l'accesso alla tabella di base, allora i dati dettagliati non permetteranno loro di visualizzare i dati della tabella di base.

La riservatezza e la segretezza dei dati possono essere ottenute creando viste che limitano il lavoro degli utenti ai soli dati necessari per svolgere i compiti loro assegnati. Se un impiegato deve lavorare con gli indirizzi dei dipendenti ma non gli è permesso di accedere ai loro stipendi, è possibile concedergli un accesso a un vista che non contenga informazioni sulla retribuzione.

Alcune persone prediligono la creazione di una vista per ogni singola tabella di base, anche se questa vista è identica alla tabella di base. Questo approccio nasce dal fatto che le viste possono contribuire a una maggiore produttività nella programmazione man mano che la base di dati si evolve. Si consideri una situazione nella quale 50 programmi utilizzano tutti la tabella CUSTOMER\_T. Si supponga che la base di dati di Pine Valley Furniture Company evolva fino a supportare nuove funzioni che richiedono che la tabella di base CUSTOMER\_T sia rinormalizzata in due tabelle. Se questi 50 programmi fanno direttamente riferimento alla tabella di base CUSTOMER\_T, dovranno essere tutti modificati per fare riferimento a una o due nuove tabelle o a tabelle unite. Se questi programmi utilizzano tutti la vista su questa tabella di base, allora dovrà essere ricreata solo la vista, risparmiando un notevole lavoro di riprogrammazione. Le viste dinamiche richiedono tuttavia una notevole elaborazione di run-time, poiché la tabella virtuale di una vista viene ricreata ogni volta che vi si fa riferimento. Fare riferimento a una tabella di base mediante una vista piuttosto che direttamente può aumentare considerevolmente il tempo di elaborazione di un'interrogazione. Questo costo operativo supplementare deve essere bilanciato con il risparmio potenziale nella riprogrammazione di una vista.

Nel resto del capitolo si creeranno e utilizzeranno viste dinamiche che duplicano le tabelle di Pine Valley Furniture che verranno utilizzate per illustrare la sintassi SQL. Le istruzioni SQL che seguono impostano le viste della tabella di base.

```
CREATE VIEW CUSTOMER_V AS SELECT * FROM CUSTOMER_T;
CREATE VIEW ORDER_V AS SELECT * FROM ORDER_T;
CREATE VIEW ORDER_LINE_V AS SELECT * FROM ORDER_LINE_T;
CREATE VIEW PRODUCT_V AS SELECT * FROM PRODUCT_T;
```

L'aggiornamento dei dati direttamente da una vista invece che dalle tabelle di base è possibile entro certi limiti. In generale, le operazioni di aggiornamento ai dati in una vista sono permessi se l'aggiornamento non presenta ambiguità in termini di modifica dei dati nella tabella di base. Al contrario, quando l'istruzione CREATE VIEW contiene una delle situazioni riportate di seguito, la vista potrebbe non essere aggiornata direttamente.

1. La clausola SELECT include la parola chiave DISTINCT (le clausole SELECT sono trattate nel dettaglio successivamente nel paragrafo "Elaborare tabelle singole").
2. La clausola SELECT contiene espressioni, comprese colonne derivate, aggregati, funzioni statistiche e così via.
3. La clausola FROM, una sotto-interrogazione o una clausola UNION fanno riferimento a più di una tabella.
4. La clausola FROM, o una sotto-interrogazione fanno riferimento a un'altra vista che non è aggiornabile.
5. Il comando CREATE VIEW contiene una clausola GROUP BY o HAVING.

Può accadere che un aggiornamento a un'istanza comporti la sparizione dell'istanza dalla vista. Facciamo un esempio. Si crei una vista chiamata EXPENSIVE\_STUFF\_V, che elenca tutti i prodotti

che hanno un UNIT\_PRICE superiore ai 300 \$. Questi includono PRODUCT\_ID 5, una scrivania, che ha un prezzo unitario di 325 \$. Se si aggiornano i dati utilizzando EXPENSIVE\_STUFF\_V e si riduce il prezzo unitario della scrivania a 295 \$, allora la scrivania non apparirà più nella tabella virtuale EXPENSIVE\_STUFF\_V, perché il suo prezzo unitario è ora inferiore ai 300 \$. Se si desidera tenere una traccia di tutti gli articoli con un prezzo originale superiore ai 300 \$, si deve includere una clausola WITH CHECK OPTION dopo la clausola SELECT per creare il comando CREATE VIEW. WITH CHECK OPTION farà sì che le istruzioni UPDATE o INSERT vengano rifiutate quando causano l'aggiornamento o l'inserimento di righe che devono essere eliminate dalla vista. Questa opzione può essere utilizzata solo con viste aggiornabili. Quella che segue è l'istruzione CREATE VIEW per EXPENSIVE\_STUFF\_V.

**Interrogazione:**     Elencare tutti i prodotti che hanno avuto un UNIT\_PRICE superiore ai 300 \$.

```
CREATE VIEW EXPENSIVE_STUFF_V
AS
 SELECT PRODUCT_ID, PRODUCT_NAME, UNIT_PRICE
 FROM PRODUCT_T
 WHERE UNIT_PRICE > 300
 WITH CHECK OPTION;
```

Quando si prova ad aggiornare il prezzo unitario di una scrivania di 295 \$, se si utilizza questa sintassi SQL in Oracle SQL\*Plus:

```
UPDATE EXPENSIVE_STUFF_V
 SET UNIT_PRICE = 295
 WHERE PRODUCT_ID = 5;
```

Oracle visualizzerà questo messaggio d'errore:

```
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

L'aumento di prezzo della scrivania a 350 \$ verrà applicato senza che appaiano messaggi d'errore perché la vista è aggiornabile e le condizioni specificate nella vista non vengono violate.

Le informazioni sulle viste vengono memorizzate nelle tabelle di sistema del DBMS. Per esempio, in Oracle8, il testo di tutte le viste viene memorizzato in DBA\_VIEWS. Gli utenti con privilegi di sistema possono trovare queste informazioni.

**Interrogazione:**     Quali informazioni sono disponibili sulla vista chiamata EXPENSIVE\_STUFF?

```
SELECT * FROM DBA_VIEWS WHERE VIEW_NAME = 'EXPENSIVE_STUFF_V';
```

**Risultato:**

| OWNER                                       | VIEW_NAME         | TEXT_LENGTH | TEXT |
|---------------------------------------------|-------------------|-------------|------|
| MPRESCOTT                                   | EXPENSIVE_STUFF_V | 103         |      |
| SELECT PRODUCT_ID, PRODUCT_NAME, UNIT_PRICE |                   |             |      |
| FROM PRODUCT_T                              |                   |             |      |
| WHERE UNIT_PRICE > 300                      |                   |             |      |

**Viste materializzate**

Come le viste dinamiche, le viste materializzate possono essere costruite in modi diversi per scopi diversi. Le tabelle possono essere duplicate per intero o in parte, possono essere aggiornate secondo un intervallo prestabilito o possono essere attivate quando è necessario accedervi. Le viste materializzate possono essere basate su interrogazioni di una o più tabelle. In Oracle 8i, è possibile creare tabelle di

riepilogo in base alle aggregazioni di dati. Le copie di dati remoti che utilizzano i dati distribuiti possono essere memorizzate localmente come viste materializzate. Nonostante si abbia un costo di manutenzione dovuto alla necessità di mantenere la vista locale sincronizzata con le tabelle di base remote o con il data warehouse, l'uso delle viste materializzate può migliorare le prestazioni delle interrogazioni distribuite, specialmente se i dati nella vista materializzata sono relativamente statici e non devono essere aggiornati di frequente.

#### 9.4.4 Creare controlli di integrità dei dati

Abbiamo già visto la sintassi che stabilisce le chiavi esterne nella Figura 9.6. Per poter stabilire l'integrità referenziale tra due tabelle con un'associazione 1:M nel modello di dati relazionale, la chiave primaria della tabella sul lato uno deve essere referenziata da una colonna nella tabella sul lato molti dell'associazione. **Integrità referenziale** significa che un valore nella colonna corrispondente sul lato molti deve corrispondere a un valore nella chiave primaria per una riga nella tabella sul lato uno, oppure essere NULL. La clausola REFERENCES SQL impedisce che un valore di chiave esterna venga aggiunto se non è già un valore valido nella colonna di chiave primaria referenziata, ma ci sono altri problemi di integrità.

Se un valore CUSTOMER\_ID viene modificato, la connessione tra quel cliente e gli ordini da lui effettuati verrà alterata. La clausola REFERENCES impedisce di apportare tali modifiche in un valore di chiave esterna, ma non in quello di chiave primaria. Questo problema può essere gestito asserendo che i valori di chiave primaria non possono essere modificati una volta che sono stati stabiliti. In questo caso gli aggiornamenti alla tabella dei clienti verranno gestiti dalla maggior parte dei sistemi includendo un clausola ON UPDATE RESTRICT. Qualsiasi aggiornamento che elimina o modifica un valore di chiave primario verrà quindi rifiutato, a meno che nessuna chiave esterna faccia riferimento a quel valore in una tabella figlio. La Figura 9.7 mostra la sintassi associata agli aggiornamenti.

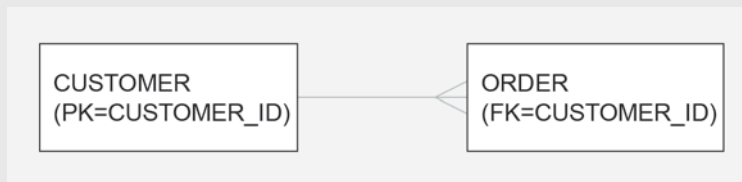
Un'altra soluzione è quella di passare la modifica attraverso la tabella o le tabelle figlio utilizzando l'opzione ON UPDATE CASCADE. Se un numero ID cliente viene modificato, questa modifica fluisce a cascata nella tabella figlio, ORDER\_T, e l>ID cliente viene aggiornato anche nella tabella ORDER\_T.

Una terza soluzione è quella di consentire l'aggiornamento su CUSTOMER\_T ma di impostare il valore CUSTOMER\_ID in oggetto nella tabella ORDER\_T a NULL utilizzando l'opzione ON UPDATE SET NULL. In questo caso, l'uso dell'opzione SET NULL comporta un'alterazione della connessione tra ordini e clienti, un effetto non desiderabile. L'opzione più flessibile da utilizzare è CASCADE. Nel caso si debba eliminare un record di cliente, sono disponibili anche ON DELETE RESTRICT, CASCADE o SET NULL. Con DELETE RESTRICT, il record del cliente non potrebbe essere eliminato a meno che non ci fossero ordini effettuati da quel cliente nella tabella ORDER\_T. Con DELETE CASCADE, l'eliminazione di un cliente rimuove da ORDER\_T tutti i record di ordine associati. Con DELETE SET NULL, i record di ordine per quel cliente vengono impostati come nulli prima che il record del cliente venga eliminato. Con DELETE SET DEFAULT, i record degli ordini per quel cliente vengono impostati a un valore predefinito prima che il record del cliente venga eliminato. DELETE RESTRICT è probabilmente l'opzione che ha più senso. Non tutti i RDBMS SQL forniscono un'integrità referenziale di chiave primaria. In questo caso i permessi di aggiornamento ed eliminazione sulla colonna della chiave primaria devono essere revocati.



**Figura 9.7**

Garantire l'integrità dei dati attraverso gli aggiornamenti.



**Aggiornamento limitato:** un ID cliente può essere eliminato solo se non si trova nella tabella ORDER.

```

CREATE TABLE CUSTOMER_T
 (CUSTOMER_ID INTEGER DEFAULT 'C999' NOT NULL,
 CUSTOMER_NAME VARCHAR(40) NOT NULL,
 ...
 CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID),
 ON UPDATE RESTRICT);

```

**Aggiornamento a cascata:** la modifica di un ID cliente nella tabella CUSTOMER comporta la modifica di quel valore nella tabella ORDER per creare la corrispondenza.

```
... ON UPDATE CASCADE);
```

**Impostare l'aggiornamento nullo:** quando un ID cliente viene modificato, qualsiasi ID cliente nella tabella ORDER che corrisponde all'ID cliente vecchio è impostato a NULL.

```
... ON UPDATE SET NULL);
```

**Impostare l'aggiornamento predefinito:** quando un ID cliente viene modificato, qualsiasi ID cliente nella tabella ORDER che corrisponde all'ID cliente vecchio è impostato a un valore predefinito.

```
... ON UPDATE SET DEFAULT);
```

## 9.4.5 Modificare le definizioni di tabella

Le definizioni di tabella possono essere modificate applicando i comandi ALTER alle specifiche di colonna. Il comando ALTER TABLE può essere utilizzato per aggiungere nuove colonne a una tabella CUSTOMER.

**Comando:** Aggiungere una colonna del tipo di cliente alla tabella CUSTOMER.

```

ALTER TABLE CUSTOMER_T
 ADD (TYPE VARCHAR (2));

```

Il comando ALTER TABLE può includere parole chiave come ADD, DROP o ALTER e permettere la modifica dei nomi di colonna, del tipo di dati, della lunghezza e dei vincoli. Quando si aggiunge una nuova colonna, il suo stato è solitamente NULL perché sia possibile lavorare con i dati che sono già stati immessi nella tabella. Quando la nuova colonna viene creata, viene aggiunta a tutte le istanze nella

tabella e un valore di NULL potrebbe essere più ragionevole. Il comando ALTER è insuperabile quando si deve adattare una base di dati alle modifiche inevitabili dovute ai cambiamenti delle esigenze, alla creazione di prototipi, allo sviluppo evolutivo e agli errori. Questo comando non può essere utilizzato per modificare una vista.

### 9.4.6 Eliminare le tabelle

Per eliminare una tabella da una base di dati, il suo proprietario può utilizzare il comando DROP TABLE. Le viste vengono eliminate utilizzando il comando analogo DROP VIEW.

**Comando:** Eliminare una tabella da uno schema di base di dati.

```
DROP TABLE CUSTOMER_T;
```

Questo comando elimina le tabelle e salva le eventuali modifiche residue alla base di dati. Per eliminare una tabella, è necessario esserne proprietari o perlomeno godere di un privilegio di sistema DROP ANY TABLE. L'eliminazione di una tabella comporta l'eliminazione anche degli indici e dei privilegi associati. Il comando DROP TABLE può essere qualificato dalle parole chiave RESTRICT o CASCADE. Se RESTRICT è specificato, il comando fallirà e la tabella non verrà eliminata se ci sono oggetti dipendenti, come viste o vincoli, che fanno correntemente riferimento alla tabella. Se CASCADE è specificato, tutti gli oggetti dipendenti verranno eliminati quando la tabella viene eliminata. Oracle permette di conservare la struttura di una tabella ma di eliminare tutti i dati che sono stati immessi nella tabella con il suo comando TRUNCATE TABLE.

## 9.5 Inserire, aggiornare ed eliminare i dati

Una volta che le tabelle e le viste sono state create, è necessario popolarle con i dati e mantenere questi dati prima che le interrogazioni possano essere scritte. Il comando SQL utilizzato per popolare le tabelle è INSERT. Se si immette un valore per ogni colonna della tabella, si può utilizzare un comando come quello riportato di seguito, che è stato utilizzato per aggiungere la prima riga di dati alla tabella CUSTOMER\_T. Si noti che i valori dei dati devono essere ordinati nello stesso ordine delle colonne nella tabella.

**Comando:** Inserire una riga di dati in una tabella nella quale verrà inserito un valore per ciascun attributo.

```
INSERT INTO CUSTOMER_T VALUES
(001, 'Contemporary Casuals', '1355 S. Himes Blvd.', 'Gainesville', 'FL', 32601);
```

Quando i dati non vengono immessi in ciascuna colonna della tabella, non si deve immettere il valore NULL per i campi vuoti né specificare le colonne nelle quali devono essere aggiunti i dati. Anche in questo caso, i valori dei dati devono essere nello stesso ordine delle colonne specificate nel comando INSERT. Per esempio, la seguente istruzione è stata utilizzata per inserire una riga di dati nella tabella PRODUCT\_T, poiché non c'è una descrizione dei prodotti per la tabella finale.

**Comando:** Inserire una riga di dati in una tabella nella quale alcuni attributi sono lasciati come nulli.

```
INSERT INTO PRODUCT_T (PRODUCT_ID, PRODUCT_DESCRIPTION,
PRODUCT_FINISH, STANDARD_PRICE, PRODUCT_ON_HAND)
VALUES (1, 'End Table', 'Cherry', 175, 8);
```

In generale, il comando INSERT colloca una nuova riga in una tabella in base ai valori forniti nell'istruzione, copia una o più righe tratte da altri dati della base di dati in una tabella, o estrae i dati da una tabella e li inserisce in un'altra. Quando si desidera popolare una tabella CA\_CUSTOMER\_T, che ha la stessa struttura della tabella CUSTOMER\_T, con i clienti di Pine Valley della sola California, si potrà utilizzare il seguente comando INSERT.

**Comando:** Popolare una tabella utilizzando un sotto-insieme di un'altra tabella con la stessa struttura.

```
INSERT INTO CA_CUSTOMER_T
 SELECT * FROM CUSTOMER_T
 WHERE STATE = 'CA';
```

La tabella identificata nel comando INSERT può essere una vista. La vista può tuttavia essere aggiornabile, così che i dati inseriti attraverso di essa vengano inseriti anche nella tabella di base su cui è fondata la vista. Se la definizione della vista è inclusa in WITH CHECK OPTION, i tentativi di inserimento dei dati attraverso la vista verranno respinti quando i valori non soddisfano le specifiche di WITH CHECK OPTION.

### 9.5.1 Input in batch

Il comando INSERT è utilizzato per immettere una riga di dati alla volta o per aggiungere più righe come risultato di un'interrogazione. Alcune versioni di SQL hanno un comando speciale o un'utilità per l'immissione di più righe di dati in batch: il comando INPUT. Oracle include un programma, SQL\*Loader, che si esegue dalla riga di comando e che può essere utilizzato per caricare i dati da un file in una base di dati. Questo celebre programma è piuttosto complesso da utilizzare e non rientra negli scopi di questo testo.

### 9.5.2 Eliminare il contenuto della base di dati

Le righe possono essere eliminate singolarmente o in gruppi. Si immagini che Pine Valley Furniture decida di non trattare più con i clienti delle Hawaii. Le righe di CUSTOMER\_T per i clienti con l'indirizzo alle Hawaii possono essere tutti eliminati con il seguente comando.

**Comando:** Eliminare le righe che soddisfano determinati criteri dalla tabella CUSTOMER.

```
DELETE FROM CUSTOMER_T
 WHERE STATE = 'HI';
```

La forma più semplice di DELETE elimina tutte le righe di una tabella.

**Comando:** Eliminare tutte le righe dalla tabella CUSTOMER.

```
DELETE FROM CUSTOMER_T;
```

Questa forma del comando deve essere utilizzata con molta attenzione!

L'eliminazione deve essere eseguita con attenzione anche quando implica le righe di più relazioni. Per esempio, se si elimina una riga di CUSTOMER\_T prima di eliminare le righe di ORDER\_T associate, si avrà una violazione dell'integrità referenziale (*Nota:* l'inclusione di una clausola ON DELETE in un campo può attenuare questo problema. Si veda il paragrafo precedente "Creare controlli di integrità dei dati" se si vuole ripassare il comando ON DELETE). SQL elimina a tutti gli effetti i record selezionati con un comando DELETE. Si consiglia quindi di eseguire prima un comando SELECT per visualizzare i record da eliminare e per verificare visivamente che siano incluse solo le righe desiderate.

### 9.5.3 Modificare il contenuto della base di dati

Per aggiornare i dati in SQL si deve informare il DBMS sulle relazioni, le colonne e le righe interessate. Se nella tabella `PRODUCT_T` è presente un prezzo non corretto per il tavolo da pranzo, l'istruzione `UPDATE SQL` definirà la correzione.

**Comando:** Modificare il prezzo unitario del prodotto 7 nella tabella `PRODUCT` a 775.

```
UPDATE PRODUCT_T
 SET UNIT PRICE = 775
 WHERE PRODUCT_ID = 7;
```

Il comando `SET` può anche modificare un valore a `NULL`; la sintassi è `SET nomecolonna = NULL`. Come accade in `DELETE`, la clausola `WHERE` in un comando `UPDATE` può contenere una sotto-interrogazione, ma la tabella da aggiornare può non essere referenziata nella sotto-interrogazione. Le sotto-interrogazioni sono trattate nel Capitolo 8.

## 9.6. Definizione dello schema interno nei RDBMS

Lo schema interno di una base di dati relazionale può essere controllato per ottenere un'efficienza di elaborazione e di memorizzazione. Alcune tecniche utilizzate per perfezionare le prestazioni operazionali del modello di dati interno di una base di dati relazionale includono le seguenti.

1. Scegliere una chiave primaria e/o secondaria di indice per aumentare la velocità della selezione delle righe, l'unione delle tabelle e l'ordinamento delle righe. L'eliminazione degli indici aumenta la velocità di aggiornamento delle tabelle. Per un ripasso sulla scelta degli indici, si veda il paragrafo specifico nel Capitolo 8.
2. Scegliere organizzazioni di file per le tabelle di base che soddisfino il tipo di attività di elaborazione su quelle tabelle (per esempio, mantenendo una tabella memorizzata fisicamente ordinata in base a una chiave di ordinamento di report utilizzata di frequente).
3. Scegliere organizzazioni di file per gli indici, che sono anche tabelle, adatte al modo in cui gli indici sono utilizzati, e allocando uno spazio supplementare per un file di indice, così che un indice possa crescere senza dover essere riorganizzato.
4. Accorpate i dati, così che le righe relative di tabelle unite frequentemente siano memorizzate le une vicine alle altre nella memoria secondaria, in modo da ridurre i tempi di recupero.
5. Mantenere statistiche sulle tabelle e i loro indici, così che il DBMS possa trovare i modi più efficienti per eseguire le varie operazioni sulla base di dati.

Non tutte queste tecniche sono disponibili in tutti i sistemi SQL. L'indicizzazione e l'accorpamento sono però quasi sempre disponibili e per questo li tratteremo di seguito.

### 9.6.1 Creare indici

Gli indici vengono creati dalla maggior parte dei RDBMS per fornire un rapido accesso casuale e sequenziale ai dati delle tabelle di base. Poiché gli standard ISO per SQL generalmente non risolvono i problemi di prestazione, non è disponibile una sintassi standard per la creazione degli indici. Gli esempi forniti utilizzano la sintassi Oracle e danno un'idea di come gli indici vengono gestiti dalla maggior par-

te dei RDBMS. Si noti che, sebbene gli utenti non facciano direttamente riferimento agli indici quando scrivono i comandi SQL, il DBMS riconosce quali indici esistenti possono migliorare le prestazioni di interrogazione. Gli indici vengono creati solitamente per le chiavi primarie e secondarie e per le chiavi concatenate (a colonna multipla). In alcuni sistemi, si può scegliere tra sequenze di chiavi crescenti o decrescenti.

Nell'esempio che segue, viene creato un indice alfabetico su CUSTOMER\_NAME nella tabella CUSTOMER\_T in Oracle.

**Comando:** Creare un indice alfabetico sui nomi dei clienti nella tabella CUSTOMER.

```
CREATE INDEX NAME_IDX ON CUSTOMER_T (CUSTOMER_NAME);
```

Gli indici possono essere creati o eliminati in qualsiasi momento. Se nelle colonne chiave esistono già dei dati, l'indice verrà popolato automaticamente per i dati esistenti. Se un indice è definito come UNIQUE (utilizzando la sintassi CREATE UNIQUE INDEX...) e i dati esistenti violano questa condizione, la creazione dell'indice fallirà. Una volta che un indice è stato creato, verrà aggiornato man mano che i dati vengono immessi, aggiornati o eliminati.

Quando non occorrono più tabelle, viste o indici, si utilizzano le istruzioni associate DROP. Nell'esempio che segue, viene eliminato l'indice NAME\_IDX creato con il comando precedente.

**Comando:** Eliminare l'indice sui nomi dei clienti nella tabella CUSTOMER.

```
DROP INDEX NAME_IDX;
```

Sebbene sia possibile indicizzare ogni colonna in una tabella, si deve prestare molta attenzione quando si decide di creare un nuovo indice. Ogni indice consuma uno spazio di memoria supplementare e richiede anche tempi e costi di manutenzione ogni volta che i dati indicizzati cambiano di valore. Questi costi possono rallentare notevolmente i tempi di risposta e comportare ritardi fastidiosi agli utenti on-line. Un sistema può utilizzare solo un indice anche se ce ne sono diversi disponibili per le chiavi in una qualifica complessa. Il progettista della base di dati deve sapere esattamente come gli indici sono utilizzati dal RDBMS per poter fare delle scelte oculute sull'indicizzazione. Oracle include uno strumento di pianificazione e spiegazione che può essere utilizzato per capire l'ordine con il quale un'istruzione SQL verrà elaborata e gli indici utilizzati. L'output include anche una stima dei costi che può essere confrontata con le stime relative all'esecuzione dell'istruzione con indici diversi, così che sia possibile determinare qual è la procedura più efficiente.

## 9.7 Elaborare tabelle singole

In SQL vengono utilizzati quattro comandi del linguaggio di manipolazione dei dati. Ne abbiamo già affrontati brevemente tre (UPDATE, INSERT e DELETE) e abbiamo visto numerosi esempi del quarto, SELECT. Laddove i comandi UPDATE, INSERT e DELETE consentono la modifica dei dati nelle tabelle, è il comando SELECT, con le sue varie clausole, che permette di interrogare i dati contenuti nelle tabelle, di rispondere a molte domande diverse o di creare interrogazioni ad hoc. La costruzione di base di un comando SQL è piuttosto semplice e facile da imparare. Non ci si lasci però imbrogliare; SQL è uno strumento potente che permette di specificare processi complessi di analisi dei dati. Poiché la sintassi di base è relativamente semplice da imparare, è facile anche scrivere interrogazioni SELECT che siano sintatticamente corrette ma che non rispondano esattamente alla domanda prevista. Prima di eseguire interrogazioni su una grande base di dati di produzione, è consigliabile testare attentamente le interrogazioni su un piccolo insieme di dati di prova per assicurarsi che restituiscano i risultati corretti. Oltre al controllo manuale dei risultati, è spesso possibile suddividere le interrogazioni in parti più

piccole, esaminare i risultati di queste interrogazioni semplici e poi riunirli. Questo garantisce che le interrogazioni agiscano insieme nel modo previsto. Inizieremo a esplorare le interrogazioni SQL che agiscono su una singola tabella. Nel Capitolo 10 uniremo le tabelle e utilizzeremo interrogazioni che richiedono più di una tabella.

## 9.7.1 Clausole dell'istruzione SELECT

La maggior parte delle istruzioni SQL per il recupero dei dati include le tre clausole seguenti.

|        |                                                                                                                                                                                                                 |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SELECT | Elenca le colonne (comprese le espressioni che implicano colonne) delle tabelle di base o le viste previste nella tabella che risulterà dal comando.                                                            |
| FROM   | Identifica le tabelle o le viste da cui verranno scelte le colonne che dovranno apparire nella tabella risultante e include le tabelle o le viste necessarie a unire le tabelle per elaborare l'interrogazione. |
| WHERE  | Include le condizioni per la selezione delle righe in una singola tabella o vista e le condizioni tra le tabelle o le viste per l'unione.                                                                       |

Le prime due clausole sono sempre necessarie, mentre la terza è richiesta solo quando devono essere recuperate determinate righe di tabella o quando si devono unire più tabelle. Gli esempi in questo paragrafo sono tratti dai dati mostrati nella Figura 9.3. Per iniziare, possiamo visualizzare il nome e la quantità di prodotti disponibili nella vista PRODUCT per tutti i prodotti che hanno un prezzo standard o inferiore ai 275 \$.

**Interrogazione:** Quali prodotti hanno un prezzo standard o inferiore ai 275 \$?

```
SELECT PRODUCT_NAME, STANDARD_PRICE
FROM PRODUCT_V
WHERE STANDARD_PRICE <275;
```

**Risultato:**

| PRODUCT_NAME  | STANDARD_PRICE |
|---------------|----------------|
| End Table     | 175            |
| Computer Desk | 250            |
| Coffee Table  | 200            |

Ogni istruzione SELECT restituisce una tabella quando viene eseguita. Due parole chiavi speciali, DISTINCT e \*, possono essere utilizzate con l'elenco delle colonne da visualizzare. Se l'utente non vuole vedere righe duplicate nel risultato, può utilizzare SELECT DISTINCT. Nell'esempio precedente, se anche *altri* tavoli da computer di Pine Valley Furniture costassero 275 \$, i risultati dell'interrogazione presenterebbero delle righe duplicate. SELECT DISTINCT PRODUCT\_NAME visualizza un risultato senza le righe duplicate. SELECT \*, dove \* è utilizzato come carattere jolly per indicare tutte le colonne, visualizza tutte le colonne di tutte le tabelle o viste nella clausola FROM.

Si noti che la clausole dell'istruzione SELECT devono essere mantenute nel loro ordine; in caso contrario, verranno visualizzati messaggi d'errore e l'interrogazione non verrà eseguita. Potrebbe anche essere necessario qualificare i nomi degli oggetti della base di dati in base alla versione di SQL utilizzata. Se c'è qualche ambiguità in un comando SQL, si deve indicare esattamente da quale tabella o vista devono provenire i dati richiesti. Per esempio, nella Figura 9.3 CUSTOMER\_ID è una colonna sia in CUSTOMER\_T sia in ORDER\_T. Quando si possiede la base di dati utilizzata (ossia, se si sono create le tabelle) e si vuole che CUSTOMER\_ID sia tratto da CUSTOMER\_T, si dovrà specificarlo richiedendo CUSTOMER\_T.CUSTOMER\_ID. Se si vuole che CUSTOMER\_ID provenga da ORDER\_T,

allora si dovrà richiedere ORDER\_T.CUSTOMER\_ID. Anche se all'utente non interessa sapere da quale tabella il CUSTOMER\_ID proviene, dovrà essere comunque preciso, perché SQL non è in grado di risolvere l'ambiguità senza l'aiuto dell'utente. Quando si utilizzano i dati creati da qualcun altro, si deve anche specificare il proprietario della tabella aggiungendone l'ID utente. La richiesta di selezionare il CUSTOMER\_ID da CUSTOMER\_T può quindi apparire nella forma *ID\_PROPRIETARIO*. CUSTOMER\_T.CUSTOMER\_ID. Gli esempi nel libro presumono che il lettore possieda le tabelle o le viste utilizzate, così che sia più facile leggere le istruzioni SELECT senza i qualificatori. Questi possono essere immessi dove necessario e possono sempre essere inclusi nelle istruzioni, se lo si desidera. L'omissione dei qualificatori può comportare qualche problema, ma nessun problema sorge dal loro inserimento.

Se la digitazione dei qualificatori e dei nomi delle colonne è impegnativa, o se i nomi delle colonne non hanno senso per chi legge i report, si possono stabilire degli alias per le colonne, le tabelle o le viste, che verranno utilizzati per il resto dell'interrogazione.

**Interrogazione:**    Qual è l'indirizzo del cliente chiamato Home Furnishings? Utilizzare un alias, NAME, per il nome del cliente.

```
SELECT CUST.CUSTOMER_NAME AS NAME, CUST.CUSTOMER_ADDRESS
FROM idproprietario.CUSTOMER_V CUST
WHERE NAME = 'Home Furnishings';
```

Questa istruzione di recupero visualizza il risultato mostrato di seguito in molte versioni di SQL. In Oracle SQL\*Plus, l'alias per la colonna non può essere utilizzato nel resto dell'istruzione SELECT, tranne che in una clausola HAVING, così che nell'ultima riga verrebbe utilizzato CUSTOMER\_NAME invece che NAME per l'esecuzione. Si noti che l'intestazione della colonna viene visualizzata come NAME invece che come CUSTOMER\_NAME, e che l'alias della vista può essere utilizzato nella clausola SELECT anche se non viene definito fino alla clausola FROM.

**Risultato:**

| NAME             | CUSTOMER_ADDRESS |
|------------------|------------------|
| Home Furnishings | 1900 Allard Ave. |

Quando si utilizza la clausola SELECT per scegliere le colonne di una tabella risultante, le colonne possono essere ridisposte in modo che in questa tabella abbiano un ordinamento diverso da quello della tabella originale. Per la precisione, le colonne verranno visualizzate nello stesso ordine in cui erano incluse nell'istruzione SELECT. Si osservi la tabella PRODUCT\_T nella Figura 9.3 per vedere l'ordinamento differente della tabella di base da quella che risulta dall'interrogazione.

**Interrogazione:**    Elencare il prezzo unitario, il nome del prodotto e l'ID del prodotto per tutti i prodotti nella tabella PRODUCT.

```
SELECT STANDARD_PRICE, PRODUCT_DESCRIPTION, PRODUCT_ID
FROM PRODUCT_T;
```

**Risultato:**

| STANDARD_PRICE | PRODUCT_DESCRIPTION  | PRODUCT_ID |
|----------------|----------------------|------------|
| 175            | End Table            | 1          |
| 200            | Coffee Table         | 2          |
| 375            | Computer Desk        | 3          |
| 650            | Entertainment Center | 4          |
| 325            | Writer's Desk        | 5          |

|     |               |   |
|-----|---------------|---|
| 750 | 8-Drawer Desk | 6 |
| 800 | Dining Table  | 7 |
| 250 | Computer Desk | 8 |

8 righe selezionate.

## 9.7.2 Utilizzare le espressioni

Con le clausole di base `SELECT ... FROM ... WHERE` si possono fare molte altre cose in una singola tabella. Si possono, per esempio, creare delle *espressioni*, che sono manipolazioni matematiche dei dati della tabella, oppure si può approfittare dei vantaggi delle funzioni memorizzate, come `SUM` o `AVG`, per manipolare le righe di dati selezionate da una tabella. Può accadere, per esempio, di voler conoscere il prezzo medio standard di ciascun articolo in giacenza. Per ottenere un valore medio, si utilizza la funzione memorizzata `AVG` e si denomina `AVERAGE` l'espressione risultante. Ecco l'interrogazione e i risultati utilizzando `SQL*Plus`.

**Interrogazione:** Qual è il prezzo standard medio per ogni prodotto in giacenza?

```
SELECT AVG (STANDARD_PRICE, AS AVERAGE
FROM PRODUCT_V;
```

Le manipolazioni matematiche possono essere costruite utilizzando i segni `+` per l'addizione, `-` per la sottrazione, `*` per la moltiplicazione e `/` per la divisione. Questi operatori possono essere utilizzati con un numero qualsiasi di colonne. Alcuni sistemi hanno anche un operando chiamato *modulo*, indicato dal segno `%`. Un modulo è il resto intero che risulta dalla divisione di due interi. Per esempio, `14 % 4` è uguale a `2` perché `14/4` è `3`. Lo standard supporta gli intervalli anno-mese e giorno-ora, cosa che rende possibile eseguire calcoli sulla data e l'ora. Si noti che il risultato dell'interrogazione precedente mostra ora le espressioni `AVERAGE`.

**Risultato:**

```
AVERAGE
440.625
```

Le regole di precedenza per l'ordine in base al quale vengono valutate le espressioni complesse sono le stesse di quelle utilizzate nei linguaggi di programmazione e in algebra. Le espressioni tra parentesi vengono calcolate per prime. Dove le parentesi non stabiliscono un ordine, vengono calcolate prima la moltiplicazione e la divisione, da sinistra a destra, seguite dall'addizione e dalla sottrazione, anche queste da sinistra a destra. Per evitare confusione, si utilizzano le parentesi per stabilire l'ordine. Se le parentesi sono nidificate, vengono effettuate per prime le operazioni più interne.

## 9.7.3 Utilizzare le funzioni

Funzioni come `COUNT`, `MIN`, `MAX`, `SUM` e `AVG` di colonne specifiche nell'elenco delle colonne di un comando `SELECT` possono essere utilizzate per specificare che la tabella risultante deve contenere dati aggregati invece che dati a livello di riga. L'uso di una qualsiasi di queste funzioni aggregate darà una risposta a una riga.

**Interrogazione:** Quanti articoli diversi sono stati ordinati nel numero d'ordine 1004?

```
SELECT COUNT (*)
FROM ORDER_LINE_V
WHERE ORDER_ID = 1004;
```



**Risultato:**

```

COUNT (*)
 2

```

Sembrerebbe che elencare l'ordine numero 1004 modificando l'interrogazione precedente sia piuttosto semplice.

**Interrogazione:** Quanti articoli diversi sono stati ordinati sul numero d'ordine 1004 e quali sono?

```

SELECT PRODUCT_ID, COUNT (*)
FROM ORDER_LINE_V
WHERE ORDER_ID = 1004;

```

In Oracle, il risultato è il seguente.

**Risultato:**

```

ERROR at line 1:
ORA-00937: not a single-group group function

```

Il problema è che `PRODUCT_ID` restituisce due valori, 6 e 8, per le due righe selezionate, mentre `COUNT` restituisce un valore aggregato, 2, per l'insieme di righe con `ID = 1004`. Nella maggior parte delle implementazioni, SQL non è in grado di restituire un valore di riga e uno dell'insieme; bisogna quindi eseguire due interrogazioni separate, una che restituisca le informazioni di riga e una quelle dell'insieme.

Può essere facile confondere le funzioni `COUNT (*)` e `COUNT`. La funzione `COUNT (*)`, utilizzata sopra, conta tutte le righe selezionate da un'interrogazione a prescindere dal fatto che una qualsiasi di queste righe contenga valori nulli. `COUNT` conta solo quelle righe che contengono un valore e ignora tutti i valori nulli.

`SUM` e `AVG` possono essere utilizzati solo con le colonne numeriche. `COUNT`, `COUNT (*)`, `MIN` e `MAX` possono essere utilizzati con qualsiasi tipo di dati. Per esempio, `MIN` utilizzato su una colonna di testo trova il valore più basso nella colonna, quello cioè la cui colonna è più vicina all'inizio dell'alfabeto. Le implementazioni SQL interpretano l'ordine alfabetico in modo diverso. Per esempio, alcuni sistemi possono iniziare con A-Z, e proseguire con a-z, con 0-9 e con i caratteri speciali. Altri trattano le lettere maiuscole e minuscole come equivalenti. Altri ancora iniziano con alcuni caratteri speciali e poi proseguono con numeri, lettere e altri caratteri speciali. In Oracle, il set di caratteri di un lingua viene determinato quando la base di dati viene creata e non può essere modificato senza ricreare la base di dati. Quando un terminale client utilizza un set di caratteri diverso, Oracle converte automaticamente il set di caratteri in entrambe le direzioni, modificando il set della base di dati del terminale client e viceversa. Quando una conversione diretta tra i set non è possibile, viene utilizzata la sostituzione dei caratteri. Ecco l'interrogazione che chiede il primo `PRODUCT_NAME` in `PRODUCT_T` in ordine alfabetico utilizzando il set di caratteri `AMERICAN` in Oracle 8i.

**Interrogazione:** Qual è, in ordine alfabetico, il primo nome di prodotto nella tabella `PRODUCT`?

```

SELECT MIN (PRODUCT_DESCRIPTION)
FROM PRODUCT_V;

```

Il risultato è illustrato di seguito, e mostra che in questo set di caratteri i numeri sono ordinati prima delle lettere.

**Risultato:**

```

MIN(PRODUCT_DESCRIPTION)
8-Drawer Desk

```

### 9.7.4 Utilizzare i caratteri jolly

L'uso dell'asterisco (\*) come carattere jolly in un'istruzione SELECT è già stato mostrato in precedenza. I caratteri jolly possono essere utilizzati anche nella clausola WHERE dove una corrispondenza esatta non è possibile. La parola chiave LIKE è accoppiata ai caratteri jolly e a una stringa che contiene i caratteri di corrispondenza presunti. Il carattere jolly % è utilizzato per rappresentare qualsiasi raccolta di caratteri. Se si utilizza LIKE '%Desk' quando si effettua una ricerca in PRODUCT\_DESCRIPTION, si troveranno tutti i tipi diversi di scrivanie prodotti da Pine Valley Furniture. L'underscore (\_) è utilizzato come carattere jolly per rappresentare un carattere esatto invece che una raccolta di caratteri. Quindi, se si utilizza LIKE '\_-drawer' quando si effettua una ricerca in PRODUCT\_NAME, si troveranno tutti i prodotti con il numero di cassetti specificati (per esempio, 3-drawer, 5-drawer o 8-drawer).

### 9.7.5 Operatori di confronto

A eccezione del primo esempio di SQL di questo paragrafo, negli altri esempi con le clausole WHERE è stato utilizzato un confronto di uguaglianza. Nel primo esempio è stato utilizzato l'operatore maggiore (minore) di. Gli operatori di confronto più comuni per le implementazioni SQL sono elencati nella Tabella 9.3. Si può essere portati a pensare che gli operatori di confronto valgano solo con i dati numerici, ma in realtà in SQL possono essere utilizzati con i dati di carattere e le date. L'interrogazione mostrata di seguito richiede tutti gli ordini effettuati dopo il 24 ottobre 2000.

**Tabella 9.3** Operatori di confronto in SQL.

| Operatore | Significato            |
|-----------|------------------------|
| =         | Uguale a               |
| >         | Maggiore di            |
| > =       | Maggiore di o uguale a |
| <         | Minore di              |
| < =       | Minore di o uguale a   |
| <>        | Diverso da             |
| !=        | Diverso da             |

**Interrogazione:** Quanti ordini sono stati effettuati dopo il 24 ottobre 2000?

```
SELECT ORDER_ID, ORDER_DATE
FROM ORDER_V
WHERE ORDER_DATE > '24-OCT-2000';
```

Si noti che la data è racchiusa tra virgolette semplici e che il suo formato è diverso da quello mostrato nella Figura 9.3, tratta da Microsoft Access. L'interrogazione è stata eseguita in SQL\*Plus.

**Risultato:**

| ORDER_ID | ORDER_DATE |
|----------|------------|
| 1007     | 27-NOV-00  |
| 1008     | 30-OCT-00  |
| 1009     | 05-NOV-00  |
| 1010     | 05-NOV-00  |

**Interrogazione:**     Quali sono i mobili prodotti da Pine Valley non in ciliegio?

```
SELECT PRODUCT_DESCRIPTION, PRODUCT_FINISH
FROM PRODUCT_V
WHERE PRODUCT_FINISH != 'Cherry';
```

**Risultato:**

| PRODUCT              | PRODUCT_FINISH |
|----------------------|----------------|
| Coffee Table         | Natural Ash    |
| Computer Desk        | Natural Ash    |
| Entertainment Center | Natural Maple  |
| 8-Drawer Desk        | White Ash      |
| Dining Table         | Natural Ash    |
| Computer Desk        | Walnut         |

6 rows selected.

### 9.7.6    Utilizzare gli operatori booleani

Perfezionando ulteriormente la clausola WHERE, si possono rispondere a domande ancora più complesse. Gli operatori booleani o logici AND, OR e NOT possono essere utilizzati a questo scopo.

- AND    Unisce due o più condizioni e restituisce un risultato solo quando tutte le condizioni sono vere.
- OR     Unisce due o più condizioni e restituisce un risultato solo quando qualsiasi condizione è vera.
- NOT    Nega un'espressione.

Se in un'istruzione SQL vengono utilizzati più operatori booleani, viene valutato per primo NOT, poi AND e quindi OR. Si consideri per esempio la seguente interrogazione.

**Interrogazione:**     Elencare il nome del prodotto, la rifinitura e il prezzo unitario per tutti i tavoli che costano più di 300 \$ nella vista PRODUCT.

```
SELECT PRODUCT_DESCRIPTION, PRODUCT_FINISH, STANDARD_PRICE
FROM PRODUCT_V
WHERE PRODUCT_DESCRIPTION LIKE '%Desk'
 OR PRODUCT_DESCRIPTION LIKE '%Table'
 AND UNIT_PRICE > 300;
```

Vengono elencate tutte le scrivanie, anche quelle da computer, che costano meno di 300 \$. Viene elencato un solo tavolo; i meno cari che costano meno di 300 \$ non sono inclusi. Nell'interrogazione, AND verrà elaborato per primo, che restituisce tutti i tavoli che hanno un prezzo unitario maggiore di 300 \$. Quindi viene elaborato OR, che restituisce tutte le scrivanie, a prescindere dal loro costo, e tutti i tavoli che costano più di 300 \$.

**Risultato:**

| PRODUCT_DESCRIPTION | PRODUCT_FINISH | STANDARD_PRICE |
|---------------------|----------------|----------------|
| Computer Desk       | Natural Ash    | 375            |
| Writer's Desk       | Cherry         | 325            |
| 8-Drawer Desk       | White Ash      | 750            |
| Dining Table        | Natural Ash    | 800            |
| Computer Desk       | Walnut         | 250            |

Se avessimo voluto avere come risultato solo le scrivanie e i tavoli che costano più di 300 \$, avremmo dovuto mettere le parentesi dopo WHERE e prima di AND.

**Interrogazione:** Elencare il nome del prodotto e il prezzo unitario per tutte le scrivanie e i tavoli nella vista PRODUCT che costano più di 300 \$.

```
SELECT PRODUCT_DESCRIPTION, PRODUCT_FINISH, UNIT_PRICE
FROM PRODUCT_V
WHERE (PRODUCT_NAME LIKE '%Desk'
 OR PRODUCT_NAME LIKE '%Table')
AND UNIT_PRICE > 300;
```

I risultati restituiti sono mostrati di seguito. Solo i prodotti con un prezzo unitario maggiore di 300 \$ vengono inclusi.

**Risultato:**

| PRODUCT_DESCRIPTION | PRODUCT_FINISH | STANDARD_PRICE |
|---------------------|----------------|----------------|
| Computer Desk       | Natural Ash    | 375            |
| Writer's Desk       | Cherry         | 325            |
| 8-Drawer Desk       | White Ash      | 750            |
| Dining Table        | Natural Ash    | 800            |

## 9.7.7 Intervalli

Gli operatori di confronto < e > sono utilizzati per stabilire un intervallo di valori. In alternativa, si possono utilizzare le parole chiave BETWEEN o NOT BETWEEN. Per esempio, per trovare i prodotti con un prezzo standard compreso tra 200 e 300 \$, si può utilizzare l'interrogazione seguente.

**Interrogazione:** Quali prodotti nella vista PRODUCT hanno un prezzo standard compreso tra 200 e 300 \$?

```
SELECT PRODUCT_DESCRIPTION, STANDARD_PRICE
FROM PRODUCT_V
WHERE STANDARD_PRICE > 199 AND STANDARD_PRICE < 301;
```

**Risultato:**

| PRODUCT_NAME  | STANDARD_PRICE |
|---------------|----------------|
| Coffee Table  | 200            |
| Computer Desk | 250            |

Lo stesso risultato viene restituito da questa interrogazione.

**Interrogazione:** Quali prodotti nella vista PRODUCT hanno un prezzo standard compreso tra 200 e 300 \$?

```
SELECT PRODUCT_DESCRIPTION, STANDARD_PRICE
FROM PRODUCT_V
WHERE STANDARD_PRICE BETWEEN 200 AND 300;
```

**Risultato:** Uguali all'interrogazione precedente.

Aggiungendo NOT prima di BETWEEN in questa interrogazione, si avranno tutti gli altri prodotti in PRODUCT\_V, poiché i loro prezzi sono inferiori ai 200 \$ o superiori ai 300.

## 9.7.8 DISTINCT

Quando nei risultati si hanno righe che non includono la chiave primaria, possono essere restituite righe duplicate. Per esempio, si osservi questa interrogazione e i risultati che restituisce.

**Interrogazione:** Quali sono i numeri d'ordine nella tabella ORDER\_LINE?

```
SELECT ORDER_ID
FROM ORDER_LINE_T;
```

Vengono restituite 18 righe e molte di queste sono duplicate, poiché la tabella contiene molti ordini per più articoli.

**Risultato:**

| <u>ORDER_ID</u> |
|-----------------|
| 1001            |
| 1001            |
| 1001            |
| 1002            |
| 1003            |
| 1004            |
| 1004            |
| 1005            |
| 1006            |
| 1006            |
| 1006            |
| 1007            |
| 1007            |
| 1008            |
| 1008            |
| 1009            |
| 1009            |
| 1010            |

18 rows selected.

Tuttavia, se si aggiunge la parola DISTINCT, verrà restituita una sola occorrenza di ciascun ORDER\_ID, una per ognuno dei dieci ordini rappresentati nella tabella.

**Interrogazione:** Quali sono i numeri d'ordine inclusi nella tabella ORDER\_LINE?

```
SELECT DISTINCT ORDER_ID
FROM ORDER_LINE_V;
```

**Risultato:**

| <u>ORDER_ID</u> |
|-----------------|
| 1001            |
| 1002            |
| 1003            |
| 1004            |
| 1005            |
| 1006            |
| 1007            |
| 1008            |

1009  
1010

10 rows selected.

DISTINCT e la sua controparte, ALL, possono essere utilizzate solo in un’istruzione di selezione. Vengono dopo SELECT e prima che le colonne o le espressioni vengano elencate. Se un’istruzione SELECT seleziona più di una colonna, verranno eliminate solo le righe che sono identiche in ogni riga. Per esempio, se l’istruzione precedente contenesse anche QUANTITY, verrebbero restituite 14 righe, perché ora ci sono solo quattro righe duplicate invece di otto. Per esempio, entrambi gli articoli ordinati su ORDER\_ID 1004 riguardavano due articoli, quindi il secondo accoppiamento di 1004 e 2 verrà eliminato.

**Interrogazione:** Quali sono le combinazioni univoche di numero d’ordine e quantità d’ordine nella tabella ORDER\_LINE?

```
SELECT DISTINCT ORDER_ID, ORDERED_QUANTITY
FROM ORDER_LINE_V;
```

**Risultato:**

| ORDER_ID | QUANTITY |
|----------|----------|
| 1001     | 1        |
| 1001     | 2        |
| 1002     | 5        |
| 1003     | 3        |
| 1004     | 2        |
| 1005     | 4        |
| 1006     | 1        |
| 1006     | 2        |
| 1007     | 2        |
| 1007     | 3        |
| 1008     | 3        |
| 1009     | 2        |
| 1009     | 3        |
| 1010     | 10       |

14 rows selected.

### 9.7.9 Liste IN e NOT IN

Per far corrispondere una lista di valori, si prenda in considerazione l’uso di IN.

**Interrogazione:** Elencare tutti i clienti che vivono negli Stati più caldi.

```
SELECT CUSTOMER_NAME, CITY, STATE
FROM CUSTOMER_V
WHERE STATE IN ('FL', 'TX', 'CA', 'HI');
```

**Risultati:**

| CUSTOMER_NAME        | CITY        | ST |
|----------------------|-------------|----|
| Contemporary Casuals | Gainesville | FL |
| Value Furniture      | Plano       | TX |
| Impressions          | Sacramento  | CA |

|                          |             |    |
|--------------------------|-------------|----|
| California Classics      | Santa Clara | CA |
| M and H Casual Furniture | Clearwater  | FL |
| Seminole Interiors       | Seminole    | FL |
| Kaneohe Homes            | Kaneohe     | HI |

7 rows selected.

IN è particolarmente utile nelle istruzioni SQL che utilizzano le sotto-interrogazioni, che verranno trattate nel Capitolo 10.

### 9.7.10 Ordinare i risultati: la clausola ORDER BY

Osservando i risultati precedenti, può sembrare che abbia più senso elencare i clienti della California, seguiti da quelli della Florida, delle Hawaii e del Texas. Questo ci porta ad altre tre parti fondamentali dell’istruzione SQL.

- ORDER BY    Ordina i risultati finali in ordine crescente e decrescente.
- GROUP BY    Raggruppa le righe in una tabella di risultati intermedi nella quale i valori in quelle righe sono gli stessi per una o più colonne.
- HAVING       Può essere utilizzata solo dopo una clausola GROUP BY e agisce come clausola WHERE secondaria, restituendo solo quei gruppi che soddisfano una condizione specifica.

Possiamo quindi ordinare i clienti aggiungendo una clausola ORDER BY.

**Interrogazione:**    Elencare il cliente, la città e lo Stato per tutti i clienti nella vista CUSTOMER il cui indirizzo è in Florida, Texas, California o Hawaii. Elencare alfabeticamente i clienti in base allo Stato e poi in base al cliente in uno Stato, sempre alfabeticamente.

```
SELECT CUSTOMER_NAME, CITY, STATE
FROM CUSTOMER_V
WHERE STATE IN ('FL', 'TX', 'CA', 'HI')
ORDER BY STATE, CUSTOMER_NAME;
```

Ora i risultati sono più facili da leggere.

**Risultato:**

| CUSTOMER_NAME            | CITY        | ST |
|--------------------------|-------------|----|
| California Classics      | Santa Clara | CA |
| Impressions              | Sacramento  | CA |
| Contemporary Casuals     | Gainesville | FL |
| M and H Casual Furniture | Clearwater  | FL |
| Seminole Interiors       | Seminole    | FL |
| Kaneohe Homes            | Kaneohe     | HI |
| Value Furniture          | Plano       | TX |

7 rows selected.

Si noti che tutti i clienti di ogni Stato sono elencati insieme e, all’interno di ogni Stato, i nomi dei clienti sono in ordine alfabetico. L’ordinamento è determinato dall’ordine in cui le colonne sono elencate nella clausola ORDER BY; in questo caso, sono stati ordinati alfabeticamente prima gli Stati e poi i nomi dei clienti. Se si ordina dall’alto in basso, si collochi una parola chiave DESC dopo la colonna utilizzata per l’ordinamento.

Come sono ordinati i valori NULL? SQL-92 stabilisce che i valori nulli debbano essere collocati per primi o ultimi, prima o dopo le colonne che hanno valori. Il punto in cui i valori nulli sono collocati dipende dall'implementazione SQL. SQL\*Plus ordina i NULL per ultimi.

### 9.7.11 Raggruppare i risultati: la clausola GROUP BY

GROUP BY è particolarmente utile se accoppiata a funzioni aggregate, come SUM o COUNT. GROUP BY divide una tabella in sottoinsiemi (in base ai gruppi), dopodiché può essere utilizzata per fornire informazioni di riepilogo per quel gruppo. Il valore singolo restituito dalla funzione aggregata degli esempi precedenti è chiamata **aggregato scalare**. Quando le funzioni aggregate sono utilizzate in una clausola GROUP BY e vengono restituiti più valori, si hanno gli **aggregati vettoriali**.

**Interrogazione:** Contare il numero dei clienti con indirizzi in ogni Stato in cui l'azienda effettua consegne.

```
SELECT STATE, COUNT (STATE)
 FROM CUSTOMER_V
 GROUP BY STATE;
```

**Risultato:**

| ST | COUNT(STATE) |
|----|--------------|
| CA | 2            |
| CO | 1            |
| FL | 3            |
| HI | 1            |
| MI | 1            |
| NJ | 2            |
| NY | 1            |
| PA | 1            |
| TX | 1            |
| UT | 1            |
| WA | 1            |

11 rows selected.

È anche possibile nidificare i gruppi dentro ad altri gruppi; la stessa logica viene utilizzata quando si ordinano più articoli.

**Interrogazione:** Contare il numero dei clienti con indirizzi in ogni città in cui l'azienda effettua le consegne. Elencare le città in base allo Stato.

```
SELECT STATE, CITY, COUNT (CITY)
 FROM CUSTOMER_V
 GROUP BY STATE, CITY;
```

Sebbene la clausola GROUP BY sembri semplice e diretta, può produrre risultati inattesi se ne viene ignorata la logica. Quando una clausola GROUP BY viene inclusa, le colonne che è permesso specificare nella clausola SELECT vengono limitate. Possono infatti essere incluse solo le colonne che hanno un valore singolo per ciascun gruppo. Nell'interrogazione precedente, ogni gruppo è costituito da una città e dal suo Stato. L'istruzione SELECT include entrambe le colonne della città e dello Stato. Questo può accadere perché ogni combinazione di città e Stato è un valore. Se la clausola SELECT della prima interrogazione in questo paragrafo avesse incluso anche 'city' l'istruzione sarebbe fallita perché GROUP BY raggruppa solo per Stato. Poiché gli Stati possono avere più di una città, il requisito che ogni valore



nella clausola `SELECT` ha solo un valore nel gruppo `GROUP BY` non viene soddisfatto e SQL non sarà in grado di presentare le informazioni sulla città in un modo sensato. In generale, ogni colonna che viene referenziata nell'istruzione `SELECT` deve essere referenziata anche nella clausola `GROUP BY`, a meno che la colonna non sia un argomento per una funzione aggregata inclusa nella clausola `SELECT`.

### 9.7.12 Qualificare i risultati in base alle categorie: la clausola `HAVING`

La clausola `HAVING` agisce come una clausola `WHERE`, ma identifica i gruppi che soddisfano un criterio piuttosto che le righe. Per questo, accade spesso di vedere una clausola `HAVING` che segue una clausola `GROUP BY`.

**Interrogazione:**     Trovare solo gli Stati che hanno più di un cliente.

```
SELECT STATE, COUNT (STATE)
 FROM CUSTOMER_V
 GROUP BY STATE
 HAVING COUNT (STATE) > 1;
```

Questa interrogazione restituisce un risultato che ha eliminato tutti quegli Stati con un cliente. Si ricordi che l'uso di `WHERE` in questo caso non può funzionare, poiché `WHERE` non consente aggregati; inoltre `WHERE` qualifica le righe, mentre `HAVING` qualifica i gruppi.

**Risultato:**

| ST | COUNT(STATE) |
|----|--------------|
| CA | 2            |
| FL | 3            |
| NJ | 2            |

Per includere più di una condizione nella clausola `HAVING`, si utilizzino `AND`, `OR` e `NOT` esattamente come nella clausola `WHERE`. Di seguito, ecco un comando che include tutte le sei clausole qui considerate; si ricordi che devono essere utilizzate in questo ordine.

**Interrogazione:**     Elencare tutte le rifiniture dei prodotti e il prezzo standard per ciascuna rifinitura per le rifiniture selezionate dove il prezzo standard medio è inferiore a 750.

```
SELECT PRODUCT_FINISH, AVG (STANDARD_PRICE)
 FROM PRODUCT_V
 WHERE PRODUCT_FINISH IN ('Cherry', 'Natural Ash', 'Natural Maple', 'White Ash')
 GROUP BY PRODUCT_FINISH
 HAVING AVG (STANDARD_PRICE) < 750
 ORDER BY PRODUCT_FINISH;
```

**Risultato:**

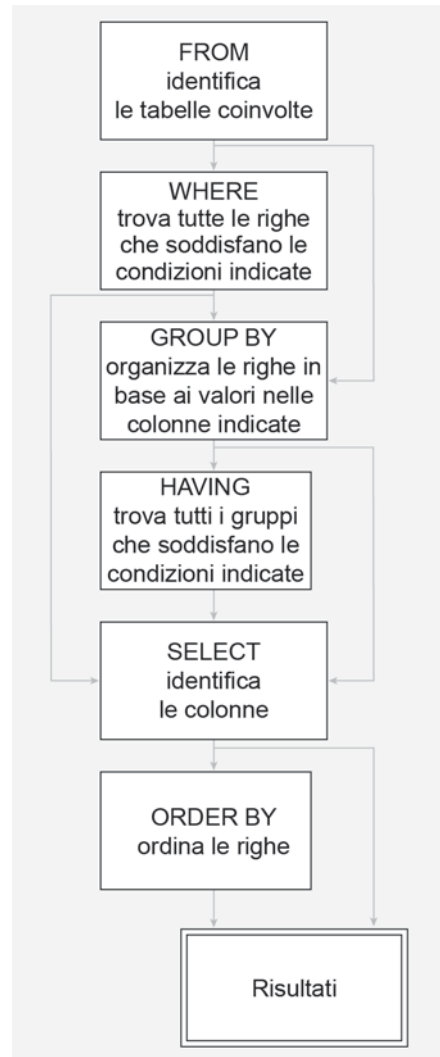
| PRODUCT_FINISH | AVG(UNIT_PRICE) |
|----------------|-----------------|
| Cherry         | 250             |
| Natural Ash    | 458.333333      |
| Natural Maple  | 650             |

La Figura 9.8 mostra l'ordine in cui SQL elabora le clausole di un'istruzione. Le frecce indicano i percorsi che possono o possono non essere eseguiti. Si ricordi che solo le clausole `SELECT` e `FROM` sono obbligatorie. Si noti che l'ordine di elaborazione è diverso da quello della sintassi utilizzata per

creare l'istruzione. Man mano che ogni clausola viene elaborata, viene prodotta una tabella di risultati intermedi che viene utilizzata per creare la clausola successiva. Gli utenti non vedono i risultati intermedi, ma solo quelli finali. È possibile eseguire il debugging di un'interrogazione ricordando l'ordine mostrato nella Figura 9.8: si estraggono le clausole facoltative e poi si aggiungono nuovamente una alla volta nell'ordine in cui verranno elaborate. In questo modo si potranno vedere i risultati intermedi e, spesso, si potranno anche individuare i problemi.

**Figura 9.8**

Ordine di elaborazione di un'istruzione SQL (adattato da van der Lans, p. 100).



## Riepilogo

Questo capitolo ha introdotto il linguaggio SQL per la definizione (DDL), la manipolazione (DML) e il controllo (DCL) delle basi di dati relazionali, comunemente utilizzati per definire e interrogare i sistemi di gestione delle basi di dati relazionali (RDBMS). Questo standard è stato criticato per i suoi molti difetti; per reazione, e per aumentare la potenza del linguaggio, le estensioni vengono riviste continuamente dal comitato X3H2 ANSI e da ISO/IEC JTC1/SC 21/WG3 DBL. Lo standard corrente è noto come SQL-99.

La definizione degli standard SQL e dei test di certificazione di conformità hanno contribuito a fare che i sistemi relazionali diventassero la forma predominante nello sviluppo delle basi di dati. I vantaggi degli standard SQL includono la riduzione dei costi di formazione, una produttività migliorata, la portabilità e la longevità delle applicazioni, una dipendenza ridotta da produttori singoli e una comunicazione migliorata tra i sistemi.

L'ambiente SQL include un'istanza di un DBMS SQL con basi di dati accessibili e utenti e programmi associati. Ogni base di dati è inclusa in un catalogo e ha uno schema che ne descrive gli oggetti. Le informazioni contenute nel catalogo sono mantenute dal DBMS stesso invece che dagli utenti del DBMS.

I comandi del linguaggio di definizione dei dati (DDL) SQL vengono utilizzati per definire una base di dati, compresa la sua creazione e la creazione delle sue tabelle, dei suoi indici e delle sue viste. Anche l'integrità referenziale viene stabilita attraverso i comandi DDL. I comandi di manipolazione dei dati (DML) sono utilizzati per caricare, aggiornare e interrogare la base di dati attraverso l'uso del comando SELECT. I comandi del linguaggio di controllo dei dati (DCL) sono utilizzati per stabilire l'accesso degli utenti alla base di dati.

I comandi SQL possono influenzare direttamente le tabelle di base, che contengono i dati grezzi, o una vista che è stata creata. Le modifiche e gli aggiornamenti alle viste possono o possono non essere trasferite alle tabelle di base. La sintassi fondamentale di un'istruzione SELECT SQL contiene le seguenti parole chiave: SELECT, FROM, WHERE, ORDER BY, GROUP BY e HAVING. SELECT determina quali attributi verranno visualizzati nella tabella dei risultati dell'interrogazione. FROM determina quali tabelle o viste verranno utilizzate nell'interrogazione. WHERE imposta il criterio dell'interrogazione, compresi gli eventuali join di tabelle multiple, se necessari. ORDER BY determina l'ordine con il quale vengono visualizzati i risultati. GROUP BY è utilizzata per raggruppare i risultati in categorie e può restituire aggregati sia scalari sia vettoriali. HAVING definisce i risultati per categorie.

La comprensione della sintassi SQL di base presentata in questo capitolo dovrebbe consentire al lettore di iniziare a utilizzare SQL nella pratica e di cogliere meglio tutte le possibilità di creazione di interrogazioni complesse con un esercizio continuo. Una trattazione avanzata di SQL è presentata nel Capitolo 10.

## Ripasso del capitolo

### Parole chiave

|                          |                                  |                               |                      |
|--------------------------|----------------------------------|-------------------------------|----------------------|
| Aggregato scalare        | Integrità referenziale           | dei dati (DML)                | Tabella di base      |
| Aggregato vettoriale     | Linguaggio di controllo dei dati | Linguaggio di definizione dei | Vista dinamica       |
| Catalogo                 | (DCL)                            | dati (DDL)                    | Vista materializzata |
| DBMS relazionale (RDBMS) | Linguaggio di manipolazione      | Schema                        |                      |

**Domande di ripasso**

1. Definire ciascuno dei seguenti termini chiave:
  - a. tabella di base
  - b. linguaggio di definizione dei dati
  - c. linguaggio di manipolazione dei dati
  - d. vista dinamica
  - e. vista materializzata
  - f. integrità referenziale
  - g. DBMS relazionale (RDBMS)
  - h. schema
2. Unire i seguenti termini alle rispettive definizioni:
 

\_\_\_\_\_ vista; \_\_\_\_\_ integrità referenziale;  
 \_\_\_\_\_ vista dinamica; \_\_\_\_\_ vista materializzata;  
 \_\_\_\_\_ SQL-99; \_\_\_\_\_ valore nullo;  
 \_\_\_\_\_ aggregato scalare; \_\_\_\_\_ aggregato vettoriale;  
 \_\_\_\_\_ catalogo; \_\_\_\_\_ schema;  
 \_\_\_\_\_ linguaggio host.

  - a. elenco di valori
  - b. descrizione di una base di dati
  - c. vista materializzata come risultato di un'interrogazione SQL che utilizza questa vista
  - d. tabella logica
  - e. valore mancante o inesistente
  - f. descrizioni di oggetti della base di dati per la stessa
  - g. linguaggio di programmazione di terza generazione in cui sono incorporati comandi SQL
  - h. stabilito in modelli di dati relazionali mediante l'uso di chiavi esterne
  - i. vista che esiste come tabella
  - j. linguaggio standard di definizione e interrogazione relazionale
  - k. valore singolo
3. Confrontare i seguenti termini:
  - a. tabella di base; vista
  - b. vista dinamica; vista materializzata
  - c. catalogo; schema
4. Cosa sono SQL-92 e SQL-99? Descrivere brevemente la differenza tra i due.
5. Descrivere un DBMS relazionale (RDBMS), il suo modello di dati, le strutture di memorizzazione dei dati e un modo per stabilire le associazioni tra i dati.
6. Elencare sei vantaggi potenziali del raggiungimento di uno standard SQL che sia ampiamente accettato.
7. Descrivere i componenti e la struttura di un tipico ambiente SQL.
8. Distinguere tra i comandi di definizione, di manipolazione e di controllo dei dati.
9. Spiegare come si stabilisce l'integrità referenziale in una base di dati compatibile con SQL-92. Spiegare le differenze tra le clausole ON UPDATE RESTRICT, ON UPDATE CASCADE e ON UPDATE SET NULL. Cosa accade se viene impostata la clausola ON DELETE CASCADE?
10. Spiegare alcuni obiettivi possibili della creazione di una vista utilizzando SQL. In particolare, spiegare come si può utilizzare una vista per rafforzare la protezione dei dati.
11. Spiegare perché è necessario limitare i tipi di aggiornamento eseguiti sui dati quando si fa riferimento ai dati attraverso una vista.
12. Descrivere un insieme di circostanze nelle quali l'uso di una vista può risparmiare sforzi di riprogrammazione.
13. In base a quanto trattato nei capitoli precedenti, spiegare i fattori da prendere in considerazione quando si deve decidere se creare un indice chiave per una tabella in SQL.
14. Spiegare e fornire almeno un esempio di come si può qualificare la proprietà di una tabella in SQL.
15. Come si modifica l'ordine in cui gli attributi appaiono in una tabella di risultati? Come si possono modificare le intestazioni di colonna?
16. Qual è la differenza tra COUNT, COUNT DISTINCT e COUNT(\*) in SQL? Quando questi tre comandi generano risultati uguali o diversi?
17. Qual è l'ordine di esecuzione degli operatori booleani (AND, OR, NOT) in un comando SQL? Come si può essere certi che gli operatori lavorino nell'ordine desiderato invece che in quello prescritto?
18. Se un'istruzione SQL include una clausola GROUP BY, gli attributi che possono essere richiesti nell'istruzione SELECT verranno limitati. Spiegare in cosa consiste la limitazione.

**Problemi ed esercizi**

I problemi ed esercizi da 1 a 9 si basano sul diagramma E-R illustrato nella Figura 6.16 nel Capitolo 6. Le relazioni 3NF con alcuni dati di esempio vengono ripetute nella Figura 9.9. Per le domande dalla 4 alla 9, si disegni un diagramma di istanza e lo si contrassegni in modo che mostri i dati che ci si aspetta vengano restituiti dall'interrogazione.

1. Scrivere una descrizione di base di dati per ciascuna delle relazioni mostrate utilizzando comandi DDL (eventualmente, ridurre, abbreviare o modificare i nomi dei dati come richiesto dalla propria versione di SQL). Si presumano i seguenti tipi di dati.

STUDENT\_ID (intero, chiave primaria)  
 STUDENT\_NAME (25 caratteri)  
 FACULTY\_ID (intero, chiave primaria)  
 FACULTY\_NAME (25 caratteri)  
 COURSE\_ID (8 caratteri, chiave primaria)  
 COURSE\_NAME (15 caratteri)

DATE\_QUALIFIED (data)  
SECTION\_ID (intero, chiave primaria)  
SEMESTER (7 caratteri)

2. Utilizzare SQL per definire la vista seguente.

| STUDENT_ID | STUDENT_NAME |
|------------|--------------|
| 38214      | Letersky     |
| 54907      | Altwater     |
| 54907      | Altwater     |
| 66324      | Aiken        |

...

3. Prima che una riga qualsiasi venga immessa nella tabella SECTION, il COURSE\_ID da immettere deve già essere presente nella tabella COURSE (integrità referenziale). Si scriva un'asserzione SQL che applichi questo vincolo.
4. Scrivere dei comandi di definizione SQL per ognuna delle seguenti interrogazioni.
- Come si può aggiungere un attributo, CLASS, alla tabella STUDENT?
  - Come si può eliminare la tabella IS\_REGISTERED?
  - Come si può portare il campo FACULTY\_NAME da 25 a 40 caratteri?
5. a. Creare due forme diverse del comando INSERT per aggiungere un ID studente 65798 e il cognome *Lopez* alla tabella STUDENT.  
b. Scrivere poi un comando che rimuova Lopez dalla tabella STUDENT.  
c. Creare un comando SQL che modifichi il nome del corso ISM 4212 da 'Database' a 'Introduction to Relational Databases'.
6. Scrivere alcune interrogazioni SQL che rispondano alle seguenti domande.
- Quali studenti hanno un numero ID inferiore a 50000?
  - Qual è il nome del membro del corpo docente il cui ID è 4756?
  - Qual è il numero di seminario più piccolo nel primo semestre del 2001?
7. Scrivere alcune interrogazioni SQL che rispondano alle seguenti domande.
- Quanti studenti sono iscritti al seminario 2714 nel primo semestre del 2001?
  - Quali membri del corpo docente sono qualificati per tenere un corso fin dal 1993? Elencare l'ID del membro, il corso e la data della qualifica.
8. Scrivere alcune interrogazioni SQL che rispondano alle seguenti domande.
- Quanti studenti sono iscritti al corso 'Database and Networking'? (*Suggerimento*: Utilizzare il SECTION\_ID per ciascuna lezione, così da poter determinare la risposta dalla tabella IS\_REGISTERED stessa.)
  - Quali docenti non possono insegnare sia Syst Analysis sia Syst Design?
9. Scrivere alcune interrogazioni SQL che rispondano alle seguenti domande.
- Quali sono i corsi inclusi nella tabella SECTION? Elencare ogni corso una volta sola.
  - Elencare tutti gli studenti in ordine alfabetico in base a STUDENT\_NAME.
  - Elencare gli studenti che sono iscritti a ciascun corso nel Semester I del 2001. Raggruppare gli studenti in base ai seminari a cui sono iscritti.
  - Elencare i corsi disponibili. Raggrupparli in base al prefisso del corso (ISM è l'unico prefisso mostrato, ma nell'università ce ne sono molti altri).
- I problemi ed esercizi da 10 a 19 si basano sull'intera base di dati di Pine Valley Furniture.
10. Modificare la tabella PRODUCT\_t aggiungendo un attributo QTY\_ON\_HAND che possa essere utilizzato per tenere traccia delle scorte delle merci disponibili. Il campo deve essere un intero di cinque caratteri e deve accettare solo numeri positivi.
11. Immettere dati di esempio a scelta in QTY\_ON\_HAND nella tabella PRODUCT\_t. Verificare la modifica apportata nel problema ed esercizio 10 cercando di aggiornare un prodotto modificando le giacenze a 10.000 unità. Testarla nuovamente modificando le giacenze per il prodotto a -10 unità. Se non vengono visualizzati messaggi d'errore e le modifiche hanno successo, significa che nel problema ed esercizio 10 non sono stati stabiliti i vincoli opportuni.
12. Aggiungere un ordine alla tabella ORDER\_t e includere un valore d'esempio per ogni attributo.
- Si considerino innanzitutto i dati nella tabella CUSTOMER\_t e si immetta un ordine per uno qualsiasi dei clienti.
  - Si immetta quindi un ordine effettuato da un nuovo cliente. A meno che non si siano già inserite informazioni sul nuovo cliente nella tabella CUSTOMER\_t, l'immissione del dato dell'ordine verrà rifiutato. I vincoli di integrità referenziale dovrebbero impedire l'immissione di un ordine se non ci sono informazioni sul cliente.
13. a. Quanti centri di produzione ha Pine Valley?  
b. Dove si trovano?
14. Quali prodotti hanno una qualche rifinitura in quercia?
15. Elencare i clienti che vivono in California o Washington. Ordinarli in base al codice di avviamento postale, dall'alto al basso.
16. Determinare il prezzo medio standard per ogni linea di prodotti.
17. Elencare gli impiegati il cui cognome inizia con una 'L'.
18. Quali impiegati sono stati assunti durante il 1999?
19. Per ogni prodotto che è stato ordinato, determinare la quantità totale che è stata ordinata. Elencare per primo il prodotto più popolare e per ultimo quello meno popolare.

**Figura 9.9**  
Relazioni nella  
programmazione  
delle lezioni (manca  
IS\_ASSIGNED).

|                                             |              |                                                                       |                  |                |
|---------------------------------------------|--------------|-----------------------------------------------------------------------|------------------|----------------|
| STUDENT ( <u>STUDENT_ID</u> , STUDENT_NAME) |              | IS_QUALIFIED ( <u>FACULTY_ID</u> , <u>COURSE_ID</u> , DATE_QUALIFIED) |                  |                |
| <u>STUDENT_ID</u>                           | STUDENT_NAME | <u>FACULTY_ID</u>                                                     | <u>COURSE_ID</u> | DATE_QUALIFIED |
| 38214                                       | Letersky     | 2143                                                                  | ISM 3112         | 9/1988         |
| 54907                                       | Altvater     | 2143                                                                  | ISM 3113         | 9/1988         |
| 66324                                       | Aiken        | 3467                                                                  | ISM 4212         | 9/1995         |
| 70542                                       | Marra        | 3467                                                                  | ISM 4930         | 9/1996         |
| ...                                         |              | 4756                                                                  | ISM 3113         | 9/1991         |
|                                             |              | 4756                                                                  | ISM 3112         | 9/1991         |
|                                             |              | ...                                                                   |                  |                |

|                                             |              |                                                  |                  |
|---------------------------------------------|--------------|--------------------------------------------------|------------------|
| FACULTY ( <u>FACULTY_ID</u> , FACULTY_NAME) |              | SECTION ( <u>SECTION_ID</u> , <u>COURSE_ID</u> ) |                  |
| <u>FACULTY_ID</u>                           | FACULTY_NAME | <u>SECTION_ID</u>                                | <u>COURSE_ID</u> |
| 2143                                        | Birkin       | 2712                                             | ISM 3113         |
| 3487                                        | Berndt       | 2713                                             | ISM 3113         |
| 4756                                        | Collins      | 2714                                             | ISM 4212         |
| ...                                         |              | 2715                                             | ISM 4930         |
|                                             |              | ...                                              |                  |

|                                          |               |                                                                   |                   |          |
|------------------------------------------|---------------|-------------------------------------------------------------------|-------------------|----------|
| COURSE ( <u>COURSE_ID</u> , COURSE_NAME) |               | IS_REGISTERED ( <u>STUDENT_ID</u> , <u>SECTION_ID</u> , SEMESTER) |                   |          |
| <u>COURSE_ID</u>                         | COURSE_NAME   | <u>STUDENT_ID</u>                                                 | <u>SECTION_ID</u> | SEMESTER |
| ISM 3113                                 | Syst Analysis | 38214                                                             | 2714              | I-2001   |
| ISM 3112                                 | Syst Design   | 54907                                                             | 2714              | I-2001   |
| ISM 4212                                 | Database      | 54907                                                             | 2715              | I-2001   |
| ISM 4930                                 | Networking    | 66324                                                             | 2713              | I-2001   |
| ...                                      |               | ...                                                               |                   |          |

Esercitazioni

1. Fissare un colloquio con un amministratore di basi di dati in un'azienda della propria zona. Quando lo si intervista, ci si prepari su un'applicazione che viene utilizzata attivamente nell'azienda. Si concentrino le domande su come gli utenti finali vengono coinvolti nell'applicazione e si cerchi di capire il livello di familiarità con SQL che devono avere. Per esempio, se gli utenti finali utilizzano SQL, quale formazioni ricevono? Utilizzano una forma interattiva di SQL o una versione incorporata?
2. Fissare un colloquio con un altro amministratore di basi di dati in un'azienda della propria zona. Concentrare l'intervista sulla comprensione dell'ambiente all'interno del quale SQL viene utilizzato nell'azienda. Informarsi sulla versione di SQL utiliz-

- zata e determinare se la stessa versione viene utilizzata in tutte le posizioni. Se vengono utilizzate versioni diverse, indagare sulle difficoltà che il DBA ha avuto nell'amministrazione della base di dati. Informarsi anche sugli eventuali linguaggi proprietari, come Oracle PL\*SQL, utilizzati. Informarsi inoltre sulle differenze possibili tra le versioni utilizzate nelle diverse posizioni e sulle difficoltà che possono verificarsi se sono installate versioni diverse.
3. Parlare con uno sviluppatore di basi di dati di un'azienda locale. Informarsi sull'uso che questi fa di SQL nel suo lavoro. Lo sviluppatore utilizza utility di generazione del codice o scrive tutto il codice SQL autonomamente? Ha a disposizione uno strumento CASE, come Designer 2000? Se possibile, chiedere

allo sviluppatore di effettuare una dimostrazione dello strumento CASE sviluppato.

## Bibliografia

Codd, E. F., 1970, "A Relational Model of Data for Large Relational Databases", *Communications of the ACM* 13 (giugno), pp. 77–87.  
 Date, C. J. e Darwen, H., 1997, *A Guide to the SQL Standard*, Reading, MA, Addison-Wesley.  
 Gruber, M., 2000, *Mastering SQL*, Alameda, CA, SYBEX.  
 Gulutzan, P. e Petzer, T., 1999, *SQL-99 Complete, Really*, Lawrence, KS, R&D Books.  
 Melton, J., 1997, "A Case for SQL Conformance Testing", *Database Programming & Design* 10 (7), pp. 66–69.  
 Van der Lans, R. F., 1993, *Introduction to SQL*, seconda edizione, Workingham, England, Addison-Wesley.

## Per approfondire

American National Standards Institute, <http://www.ansi.org>, 1997–2000 (informazioni su ANSI e sugli standard nazionali e internazionali più recenti).  
 Bowman, J. S., Emerson, S. L. e Darnovsky, M., 1996, *The Practical SQL Handbook*, terza edizione, Reading, MA, Addison-Wesley.

Celko, J., 1997, *Joe Celko's SQL Puzzles & Answers*, San Francisco, Morgan Kaufmann.  
 IEEE Standards Systems/Network Staff, *IEEE Standards Home Page*, <http://standards.ieee.org>, 2000 (informazioni sull'associazione degli standard IEEE).  
 ODMG Home Page, <http://www.odmg.org/>, 1993–2000 (consorzio dei produttori di sistemi di gestione delle basi di orientate agli oggetti e delle parti interessate).  
 OMG Home Page, <http://www.omg.org/>, 1997–2000 (*Object Management Group*: informazioni su CORBA).  
 SW Consulting, SA, IEC—*International Electrotechnical Commission—Home Page* (inglese), <http://www.iec.ch>, 2000 (ente per la definizione di standard e conformità in tutti i campi dell'elettrotecnologia).  
 Tibbs, A., *Welcome to NCITS*, <http://www.ncits.org/>, senza data (ultimo accesso: 28 dicembre 2000). (*National Committee for Information Technology Standards*, in precedenza Accredited Standard Committee X3).  
*Welcome to ISO Online*, <http://www.iso.ch>, 11 dicembre 2000 (informazioni sull'*International Organization for Standardization*).

## Project Case



# MOUNTAIN VIEW COMMUNITY HOSPITAL

Si utilizzi il modello di dati SQL costruito nel Capitolo 6 per Mountain View Community Hospital per completare le domande e gli esercizi sul progetto.

### Domande sul progetto

1. Quale versione di SQL verrà utilizzata per gli esercizi sul progetto?
2. Quali strumenti CASE sono disponibili per gli esercizi sul progetto?

### Esercizi sul progetto

1. Creare in SQL la base di dati di Mountain View Community Hospital per il modello di dati concettuale costruito nei capitoli precedenti. Si utilizzino le informazioni fornite nei paragrafi del project case alla fine dei Capitoli 5, 6, 7 e 8 per avere un aiuto nel scegliere i tipi di dati di colonna, le lunghezze, gli indici e così via.
2. Se nella domanda precedente si è dimenticato di stabilire le chiavi primarie ed esterne, si creino le asserzioni SQL necessarie a farlo.
3. Selezionare almeno una porzione della base di dati e popolarla con dati di esempio. Per esempio, si potrebbe voler lavorare con le parti della base di dati relative allo staff, all'unità di cura o ai pazienti; oppure si potrebbe lavorare con le parti relative ai produttori, agli articoli medici/chirurgici, agli esami e ai pazienti. Ci si prepari a sostenere il test dei dati d'esempio che si immettono nella base di dati. Come i valori che si utilizzano possono aiutare a testare la funzionalità della base di dati?
4. Scrivere e testare alcune interrogazioni che funzioneranno utilizzando i dati di esempio. Scrivere interrogazioni che:
  - a. selezionino le informazioni da una sola delle tabelle;
  - b. aggregino informazioni da un attributo in una tabella;
  - c. testino le varie funzioni come MIN, MAX e AVG.





**Al termine di questo capitolo sarete in grado di fare quanto segue.**

1. Definire i seguenti termini chiave: join, equi-join, join naturale, join esterno, sotto-interrogazione correlata, tipo di dati definito dall'utente, *Persistent Stored Modules* (SQL/PSM), trigger, funzione, procedura, SQL incorporato e SQL dinamico.
2. Scrivere interrogazioni di tabella singole e multiple utilizzando i comandi SQL.
3. Definire tre tipi di comandi di unione e utilizzare SQL per scriverli.
4. Scrivere sotto-interrogazioni non correlate e correlate e sapere quando scriverle.
5. Stabilire l'integrità referenziale utilizzando SQL.
6. Capire gli usi comuni dei trigger e delle stored procedure di una base di dati.
7. Discutere lo standard SQL-99 e spiegarne i miglioramenti e le estensioni rispetto a SQL-92.

## Introduzione

Il capitolo precedente ha introdotto SQL e ne ha esplorato le potenzialità nell'interrogazione di una tabella. La vera forza del modello relazionale deriva dalla sua memorizzazione dei dati in più entità correlate. Per trasferire questo approccio alla memorizzazione dei dati è necessario stabilire le associazioni e costruire interrogazioni che utilizzino i dati provenienti da più tabelle. Questo capitolo mostrerà le interrogazioni su tabelle multiple nel dettaglio. Vengono inoltre illustrati alcuni approcci diversi per ottenere risultati da più di una tabella, compreso l'uso delle sotto-interrogazioni e dei join interni, esterni e di unione.

Una volta compresa la sintassi di base di SQL, è importante capire come SQL viene utilizzato nella creazione delle applicazioni. I trigger, piccoli moduli di codice che includono SQL, vengono eseguiti automaticamente quando si verifica una particolare condizione definita all'interno del trigger stesso. Le procedure sono moduli di codice simili, ma devono essere chiamate per poter essere eseguite. I comandi SQL sono spesso incorporati in moduli scritti in un linguaggio host, come C o Java. SQL dinamico crea istruzioni SQL al volo, inserendo i valori di parametro necessari, ed è essenziale nelle applicazioni Web. In questo capitolo sono inclusi una presentazione e alcuni esempi di ciascuno di questi metodi. Vengono inoltre trattati alcuni degli miglioramenti e delle estensioni inclusi in SQL-99.

Al termine del capitolo viene offerta agli studenti una panoramica di SQL e di alcuni dei modi in cui viene utilizzato. Ci sono molte funzioni supplementari, spesso definite come "oscuri" nei testi speciali-

stici su SQL, che si rendono necessarie in alcune situazioni particolari. Esercitarsi con la sintassi inclusa in questo capitolo è un buon punto di partenza per acquisire padronanza di SQL.

## 10.1 Elaborare le tabelle multiple

Esaminate le possibilità per lavorare con una singola tabella, si passerà ora a lavorare con più tabelle contemporaneamente. La potenza del RDBMS si realizza appieno proprio in questa situazione. Se tra le tabelle esistono delle relazioni, queste possono essere collegate tra loro nelle interrogazioni. Come si ricorderà, nel Capitolo 7 si è visto che queste relazioni sono stabilite includendo una o più colonne comuni in ciascuna tabella dove un'associazione è necessaria. Questo si ottiene solitamente impostando una relazione chiave primaria-chiave esterna, dove la chiave esterna in una tabella fa riferimento alla chiave primaria in un'altra e i valori in entrambe derivano da un dominio comune. Si possono utilizzare queste colonne per stabilire il collegamento tra le due tabelle trovando i valori comuni nelle colonne. Per esempio, nella Figura 9.3, `Customer_ID` in `Order_t` corrisponde a `Customer_ID` in `Customer_t`. Se le confrontiamo, vedremo che Contemporary Casuals ha effettuato gli ordini numero 1001 e 1010 perché il `Customer_ID` di Contemporary Casuals è 1, e `Order_t` mostra che gli `Order_ID` 1001 e 1010 sono stati effettuati dal cliente numero 1. In un sistema relazionale, i dati provenienti da tabelle correlate sono uniti in una tabella di risultati, o vista, e quindi visualizzati o utilizzati come input per la definizione di una maschera o di un report.

Il collegamento delle tabelle correlate varia a seconda dei diversi tipi di sistemi relazionali. In SQL, la clausola `WHERE` del comando `SELECT` è utilizzata anche per operazioni su tabelle multiple; in effetti `SELECT` può includere riferimenti a due, tre o più tabelle nello stesso comando. Come si vedrà di seguito, SQL ha due modi per utilizzare `SELECT` per combinare i dati di tabelle correlate.

L'operazione relazionale più utilizzata, che raccoglie i dati da due o più tabelle correlate in un'unica tabella risultante, è chiamata **join** (unione). SQL specifica un join in modo implicito facendo riferimento in una clausola `WHERE` alla corrispondenza di colonne comuni nelle tabelle che sono unite. Due tabelle possono essere unite quando ognuna delle due contiene una colonna che condivide un dominio comune con l'altra. Ogni riga restituita contiene dati tratti dalle righe in diverse tabelle di input dove i valori delle colonne comuni corrispondono.

Una regola importante nel formare condizioni di unione è la seguente: ci deve essere una condizione nella clausola `WHERE` per ogni coppia di tabelle che devono essere unite. Così, se devono essere unite due tabelle, sarà necessaria una condizione; ma se devono essere unite tre tabelle (A, B e C), allora saranno necessarie due condizioni, perché ci sono due coppie di tabelle (A-B e B-C) e via di questo passo.

Nelle interrogazioni delle basi di dati relazionali sono possibili molti tipi di join, sebbene ciascuna implementazione SQL ne supporti solo alcuni. In questo capitolo ne vengono descritti quattro tipi: equi-join, join naturali, join esterni e join di unione.

### 10.1.1 Equi-join

Con un **equi-join**, la condizione di join si basa sull'uguaglianza tra i valori nelle colonne comuni. Per esempio, se si volessero conoscere i nomi dei clienti che hanno effettuato degli ordini, si dovrebbero consultare le due tabelle che conservano queste informazioni, ovvero `CUSTOMER_T` e `ORDER_T`. (Nota: utilizzeremo solo lettere maiuscole per i nomi di tabelle e attributi per indicare l'uso di elementi di una base di dati Oracle.) È necessario far corrispondere i clienti ai rispettivi ordini e raccogliere quindi le informazioni sul nome e il numero d'ordine in una tabella per rispondere alla seguente domanda.

**Interrogazione:** Quali sono i nomi di tutti i clienti che hanno effettuato degli ordini?

```
SELECT CUSTOMER_T.CUSTOMER_ID, ORDER_T.CUSTOMER_ID,
CUSTOMER_NAME, ORDER_ID
FROM CUSTOMER_T, ORDER_T
WHERE CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID;
```

**Risultato:**

| CUSTOMER_ID | CUSTOMER_ID | CUSTOMER_NAME            | ORDER_ID |
|-------------|-------------|--------------------------|----------|
| 1           | 1           | Contemporary Casuals     | 1001     |
| 8           | 8           | California Classics      | 1002     |
| 15          | 15          | Mountain Scenes          | 1003     |
| 5           | 5           | Impressions              | 1004     |
| 3           | 3           | Home Furnishings         | 1005     |
| 2           | 2           | Value Furniture          | 1006     |
| 11          | 11          | American Euro Lifestyles | 1007     |
| 12          | 12          | Battle Creek Furniture   | 1008     |
| 4           | 4           | Eastern Furniture        | 1009     |
| 1           | 1           | Contemporary Casuals     | 1010     |

10 rows selected.

Le colonne CUSTOMER\_ID ridondanti, una per ciascuna tabella, dimostrano che c'è una corrispondenza per gli ID cliente e che questa corrispondenza assegna una riga a ciascun ordine effettuato. L'importanza di raggiungere una corrispondenza tra le tabelle emerge chiaramente se la clausola WHERE viene omessa. Quel tipo di interrogazione restituirà tutte le combinazioni e gli ordini, o 150 righe, che non sono un risultato utile o significativo. Il numero delle righe è uguale al numero delle righe in ciascuna tabella, moltiplicate tra loro (10 ordini × 15 clienti = 150 righe). Questo tipo di join è detto *cartesiano*.

Per illustrare la differenza tra Oracle SQL e Access SQL, mostriamo di seguito la sintassi per questa interrogazione generata da Access.

**Interrogazione:** Quali sono i nomi di tutti i clienti che hanno effettuato degli ordini?

```
SELECT Customer_t.Customer_ID, Order_t.Customer_ID,
Customer_t.Customer_name, Order_t.Order_ID
FROM Customer_t INNER JOIN Order_t ON Customer_t.Customer_ID =
Order_t.Customer_ID;
```

**Risultato:** Come nell'interrogazione precedente.

Il join denota un join interno tra Customer\_t e Order\_t. La clausola WHERE è rappresentata dalla clausola ON Customer\_t.Customer\_ID = Order\_t.Customer\_ID. INNER JOIN ... ON è la sintassi SQL-92, che probabilmente diventerà sempre più diffusa. Si noti che la sintassi del join è stata spostata dalla clausola FROM, lasciando la clausola WHERE sulla funzione solo come filtro.

## 10.1.2 Join naturali

Un **join naturale** è uguale a un equi-join tranne per il fatto che viene eseguito sulle colonne corrispondenti che sono state definite con lo stesso nome e che prevede che una delle colonne venga eliminata. Il join naturale è la forma di operazione di join più utilizzata. Il comando SQL per trovare i nomi dei clienti e i numeri degli ordini tralascia un CUSTOMER\_ID. Si noti che CUSTOMER\_ID deve sempre essere

qualificato perché c'è ancora un'ambiguità; CUSTOMER\_ID esiste infatti sia in CUSTOMER\_T sia in ORDER\_T; occorre quindi specificare da quale tabella SQL\*Plus deve scegliere il CUSTOMER\_ID.

**Interrogazione:** Per ogni cliente che ha effettuato un ordine, qual è il nome del cliente e il numero d'ordine?

```
SELECT CUSTOMER_T.CUSTOMER_ID, CUSTOMER_NAME, ORDER_ID
FROM CUSTOMER_T, ORDER_T
WHERE CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID;
```

Si noti che l'ordine dei nomi delle tabelle nella clausola FROM non è materiale. L'ottimizzatore di interrogazioni del DBMS decide in quale sequenza elaborare ciascuna tabella. L'esistenza di indici sulle colonne comuni influisce sulla sequenza con la quale le tabelle vengono elaborate, così come stabilisce quale tabella è sul lato 1 e quale è sul lato M dell'associazione 1:M. Se un'interrogazione impiega molto tempo a essere elaborata in base all'ordine con cui le tabelle sono elencate nella clausola FROM, significa che il DBMS non ha un buon ottimizzatore di interrogazioni.

### 10.1.3 Join esterni

Quando si uniscono due tabelle, accade spesso che una riga in una tabella non abbia una riga corrispondente nell'altra. Per esempio, molti numeri CUSTOMER\_ID non appaiono nella tabella ORDER\_T (Figura 9.3), e lo si può desumere dal fatto che questi clienti non hanno effettuato ordini dopo il 21 ottobre 2000 o dal fatto che i loro ordini non sono inclusi nella nostra breve tabella ORDER\_T d'esempio. Come risultato, l'equi-join e il join naturale mostrati in precedenza non includono tutti i clienti mostrati in CUSTOMER\_T.

Un'azienda può essere ovviamente molto interessata a identificare quei clienti che non hanno effettuato ordini. Potrebbe volerli contattare per incoraggiare nuovi ordini oppure potrebbe voler analizzare questi clienti per cercare di capire perché non effettuano ordini. Utilizzando un **join esterno** si ottengono queste informazioni; nella tabella dei risultati vengono incluse anche le righe che non hanno valori corrispondenti nelle colonne comuni. I valori nulli appaiono nelle colonne dove non c'è una corrispondenza tra le tabelle.

I join esterni possono essere gestiti dai produttori di RDBMS principali, ma la sintassi utilizzata per generare un join esterno varia tra i produttori. L'esempio qui fornito utilizza la sintassi ANSI standard. Quando un join esterno non viene utilizzato esplicitamente, si utilizzino UNION e NOT EXISTS (trattati successivamente nel capitolo). Ecco un esempio di join esterno.

**Interrogazione:** Elencare il nome cliente, il numero identificativo e il numero d'ordine per tutti i clienti elencati nella tabella CUSTOMER. Includere il numero di identificazione cliente e il nome anche se non ci sono ordini disponibili per quel cliente.

```
SELECT CUSTOMER_T.CUSTOMER_ID, CUSTOMER_NAME, ORDER_ID
FROM CUSTOMER_T LEFT OUTERJOIN ORDER_T
WHERE CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID;
```

È stata scelta la sintassi LEFT OUTER JOIN perché la tabella CUSTOMER\_T è stata citata per prima, e perché è la tabella da cui si vogliono ottenere tutte le righe, a prescindere dall'esistenza di una corrispondenza d'ordine nella tabella ORDER\_T. Invertendo l'ordine in cui vengono elencate le tabelle, si otterrebbero gli stessi risultati richiedendo un RIGHT OUTER JOIN. È anche possibile richiedere un FULL OUTER JOIN; in questo caso, tutte le righe sarebbero corrispondenti e verrebbero restituite, comprese quelle che non hanno una corrispondenza nell'altra tabella. I join interni sono molto più dif-

fusi di quelli esterni, poiché questi ultimi sono necessari solo quando si prevede che possano mancare delle informazioni se si utilizza un join interno.

Si noti inoltre che la sintassi di OUTER JOIN non si applica facilmente a una condizione di join di due o più tabelle. I risultati restituiti variano a seconda del produttore; ci assicuri quindi di testare qualsiasi sintassi di join esterno che implica più di due tabelle, così da capire come verrà interpretata dal DBMS utilizzato.

La tabella dei risultati di un join esterno può anche indicare NULL come valore delle colonne nella seconda tabella qualora non si raggiunga alcuna corrispondenza. Se queste colonne possono avere NULL come valore di dati, allora non è possibile sapere se la riga restituita è una riga che ha una corrispondenza, a meno di non eseguire un'altra interrogazione che cerchi i valori nulli nella tabella di base o nella vista. A una colonna definita come NOT NULL può essere assegnato un valore NULL nella tabella dei risultati di un OUTER JOIN.

**Risultato:**

| CUSTOMER_ID | CUSTOMER_NAME            | ORDER_ID |
|-------------|--------------------------|----------|
| 1           | Contemporary Casuals     | 1001     |
| 1           | Contemporary Casuals     | 1010     |
| 2           | Value Furniture          | 1006     |
| 3           | Home Furnishings         | 1005     |
| 4           | Eastern Furniture        | 1009     |
| 5           | Impressions              | 1004     |
| 6           | Furniture Gallery        |          |
| 7           | Period Furnishings       |          |
| 8           | California Classics      | 1002     |
| 9           | M & H Casual Furniture   |          |
| 10          | Seminole Interiors       |          |
| 11          | American Euro Lifestyles | 1007     |
| 12          | Battle Creek Furniture   | 1008     |
| 13          | Heritage Furnishings     |          |
| 14          | Kaneohe Homes            |          |
| 15          | Mountain Scenes          | 1003     |

16 rows selected.

Il vantaggio del join esterno è che le informazioni non vanno perse. In questo caso sono stati restituiti tutti i nomi dei clienti anche se questi non hanno effettuato alcun ordine. La richiesta di un join RIGHT OUTER farà sì che vengano restituiti tutti gli ordini (poiché l'integrità referenziale richiede che ogni ordine sia associato a un ID cliente valido, il join interno destro assicurerà solo che l'integrità referenziale venga applicata).

**Interrogazione:** Elencare il nome cliente, il numero identificativo e il numero d'ordine per tutti i clienti elencati nella tabella ORDER. Includere il numero d'ordine anche se non sono disponibili un nome cliente e un numero identificativo.

```
SELECT CUSTOMER_T.CUSTOMER_ID, CUSTOMER_NAME, ORDER_ID
FROM CUSTOMER_T RIGHT OUTER JOIN ORDER_T ON
CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID;
```

### 10.1.4 Join di unione

SQL-99 consente anche un UNION JOIN, che non è ancora implementato in tutti i prodotti DBMS. Il risultato di un join di unione è una tabella che include tutti i dati provenienti da ogni tabella che viene unita. La tabella dei risultati contiene tutte le colonne di ogni tabella e conterrà un'istanza per ogni riga di dati inclusa da ogni tabella. Un join di unione della tabella CUSTOMER\_T (15 clienti e 6 attributi) e della tabella ORDER\_T (10 ordini e 3 attributi) restituirà una tabella di 25 righe e 9 attributi. Presumendo che la tabella originale non contenesse nulli, ogni riga dei clienti nella tabella dei risultati conterrà 3 attributi con dei valori nulli assegnati e ogni riga degli ordini conterrà 6 attributi con dei valori nulli assegnati. Non si confonda questo comando con il comando UNION che unisce più istruzioni SELECT e che verrà trattato successivamente nel capitolo.

### 10.1.5 Join multipli d'esempio con quattro tabelle

Gran parte della potenza del modello relazionale deriva dalla sua capacità di lavorare con le relazioni tra gli oggetti in una base di dati. Progettare una base di dati in modo che i dati su ciascun oggetto siano mantenuti in tabelle separate semplifica la manutenzione e l'integrità dei dati. La capacità di correlare gli oggetti gli uni agli altri unendo le tabelle fornisce alcune informazioni aziendali e di report importanti per gli impiegati. Sebbene gli esempi nei Capitoli 7 e 8 siano semplici e siano stati concepiti per fornire una comprensione di base di SQL, è importante sapere che questi comandi possono essere costruiti in interrogazioni molto più complesse che forniscono esattamente le informazioni per un report o una elaborazione.

Ecco un'interrogazione d'esempio che implica un join su quattro di tabelle. Questa interrogazione produce una tabella di risultati che include le informazioni necessarie a creare una fattura per l'ordine numero 1006. Quello che si vuole ottenere sono le informazioni sui clienti, le informazioni sugli ordini e la linea d'ordine e le informazioni sui prodotti; sarà quindi necessario unire quattro tabelle.

**Interrogazione:** Assemblare tutte le informazioni necessarie a creare una fattura per il numero d'ordine 1006.

```
SELECT CUSTOMER_T.CUSTOMER_ID, CUSTOMER_NAME, CUSTOMER_ADDRESS,
CITY, STATE, POSTAL_CODE, ORDER_T.ORDER_ID, ORDER_DATE, QUANTITY,
PRODUCT_NAME, UNIT_PRICE, (QUANTITY * UNIT_PRICE)
FROM CUSTOMER_T, ORDER_T, ORDER_LINE_T, PRODUCT_T
WHERE CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID
AND ORDER_T.ORDER_ID = ORDER_LINE_T.ORDER_ID
AND ORDER_LINE_T.PRODUCT_ID = PRODUCT_T.PRODUCT_ID
AND ORDER_T.ORDER_ID = 1006;
```

I risultati dell'interrogazione sono mostrati nella Figura 10.1. Si ricordi che, poiché il join implica quattro tabelle, ci saranno tre condizioni di join di tabella.

### 10.1.6 Sotto-interrogazioni

Gli esempi SQL precedenti illustrano uno dei due approcci di base per l'unione di due tabelle: la tecnica di unione. SQL fornisce anche la tecnica delle sotto-interrogazioni, che implica la collocazione di un'interrogazione interna (SELECT, FROM, WHERE) in una clausola WHERE o HAVING di un'altra interrogazione esterna. L'interrogazione interna fornisce i valori per la condizione di ricerca di quella esterna. Tali interrogazioni sono chiamate *sotto-interrogazioni* o *sotto-interrogazioni nidificate* e possono essere nidificate più volte.

**Figura 10.1**  
I risultati di un join  
su quattro tabelle  
(modificati  
per la leggibilità).

| CUSTOMER_ID | CUSTOMER_NAME   | CUSTOMER_ADDRESS    | CITY                 | ST         | POSTAL_CODE           |
|-------------|-----------------|---------------------|----------------------|------------|-----------------------|
| 2           | Value Furniture | 15145 S.W. 17th St. | Plano                | TX         | 75094                 |
| 2           | Value Furniture | 15145 S.W. 17th St. | Plano                | TX         | 75094                 |
| 2           | Value Furniture | 15145 S.W. 17th St. | Plano                | TX         | 75094                 |
| ORDER_ID    | ORDER-DAT       | QUANTITY            | PRODUCT_NAME         | UNIT_PRICE | (QUANTITY*UNIT_PRICE) |
| 1006        | 24-OCT-00       | 1                   | Entertainment Center | 650        | 650                   |
| 1006        | 24-OCT-00       | 2                   | Writer's Desk        | 325        | 650                   |
| 1006        | 24-OCT-00       | 2                   | Dining Table         | 800        | 1600                  |

A volte le tecniche di unione e delle sotto-interrogazioni possono essere utilizzate per ottenere lo stesso risultato, e persone diverse possono avere preferenze diverse sulla tecnica da utilizzare. Altre volte solo una delle due tecniche funziona. La tecnica di unione è utile quando i dati di più relazioni devono essere recuperati e visualizzati e le relazioni non sono necessariamente nidificate. Si confrontino due interrogazioni che restituiscono lo stesso risultato; entrambe rispondono alla domanda: “Qual è il nome e l’indirizzo del cliente che ha effettuato l’ordine numero 1008?”. Iniziamo con un’interrogazione di join.

**Interrogazione:** Qual è il nome e l’indirizzo del cliente che ha effettuato l’ordine numero 1008?

```
SELECT CUSTOMER_NAME, CUSTOMER_ADDRESS, CITY, STATE, POSTAL_CODE
FROM CUSTOMER_T, ORDER_T
WHERE CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID AND
ORDER_ID = 1008;
```

Ecco invece l’interrogazione equivalente che utilizza la tecnica delle sotto-interrogazioni.

**Interrogazione:** Qual è il nome e l’indirizzo del cliente che ha effettuato l’ordine numero 1008?

```
SELECT CUSTOMER_NAME, CUSTOMER_ADDRESS, CITY, STATE, POSTAL_CODE
FROM CUSTOMER_T
WHERE CUSTOMER_T.CUSTOMER_ID =
(SELECT ORDER_T.CUSTOMER_ID
FROM ORDER_T
WHERE ORDER_ID = 1008);
```

Si noti che la sotto-interrogazione, racchiusa tra parentesi, segue la sintassi già nota della costruzione delle interrogazioni SQL e che può essere un’interrogazione indipendente. In questo caso l’approccio delle sotto-interrogazioni può essere utilizzato perché occorre solo visualizzare i dati della tabella nell’interrogazione esterna. Il valore di ORDER\_ID non appare nel risultato dell’interrogazione, ma è utilizzato come criterio di selezione nell’interrogazione esterna. Per includere i dati della sotto-interrogazione nel risultato, si deve utilizzare la tecnica dei join, poiché i dati di una sotto-interrogazione non possono essere inclusi nei risultati finali.

Sappiamo già in anticipo che la sotto-interrogazione precedente restituirà al massimo un valore, il CUSTOMER\_ID associato all’ORDER\_ID numero 1008. Il risultato sarà vuoto se un ordine con quell’ID non esiste. Una sotto-interrogazione può anche restituire un elenco di valori (con zero, una o molte immissioni) utilizzando la parola chiave IN. Per esempio, quali clienti hanno effettuato degli ordini? Ecco un’interrogazioni che può rispondere a questa domanda.

**Interrogazione:** Quali clienti hanno effettuato degli ordini?



```

SELECT CUSTOMER_NAME
 FROM CUSTOMER_T
 WHERE CUSTOMER_ID IN
 (SELECT DISTINCT CUSTOMER_ID
 FROM ORDER_T);

```

Questa interrogazione produce il risultato riportato di seguito. Nella sotto-interrogazione viene utilizzato DISTINCT perché non ci interessa tanto sapere quanti ordini sono stati effettuati da un cliente, quanto sapere se ne hanno effettuati. Per ogni cliente identificato nella tabella ORDER\_T, quel nome di cliente è stato restituito da CUSTOMER\_T (questa interrogazione verrà ripresa nella Figura 10.2a.)

**Risultato:**

```

CUSTOMER_NAME
Contemporary Casuals
Value Furniture
Home Furnishings
Eastern Furniture
Impressions
California Classics
American Euro Lifestyles
Battle Creek Furniture
Mountain Scenes

```

9 rows selected.

I qualificatori NOT, ANY e ALL possono essere utilizzati davanti a IN o con operatori logici come =, > e <. Poiché IN funziona con zero, uno o molti valori dell'interrogazione interna, molti programmatori utilizzano semplicemente IN invece di = per tutte le interrogazioni, anche se il segno uguale funzionerebbe comunque. Il prossimo esempio mostra l'uso di NOT e dimostra che un join può essere utilizzato in un'interrogazione interna.

**Interrogazione:** Quali clienti non hanno effettuato ordini per tavoli da computer?

```

SELECT CUSTOMER_NAME
 FROM CUSTOMER_T
 WHERE CUSTOMER_ID NOT IN
 (SELECT CUSTOMER_ID
 FROM ORDER_T, ORDER_LINE_T, PRODUCT_T
 WHERE ORDER_T.ORDER_ID =
 ORDER_LINE_T.ORDER_ID AND
 ORDER_LINE_T.PRODUCT_ID =
 PRODUCT_T.PRODUCT_ID
 AND PRODUCT_NAME = 'Computer Desk');

```

**Risultato:**

```

CUSTOMER_NAME
Value Furniture
Home Furnishings
Eastern Furniture
Furniture Gallery
Period Furniture
M H Casual Furniture
Seminole Interiors

```

American Euro Lifestyles  
Heritage Furnishings  
Kaneohe Homes

10 rows selected.

Il risultato mostra che 10 dei clienti non hanno ancora ordinato tavoli da computer. L'interrogazione interna ha restituito un elenco di tutti i clienti che hanno ordinato tavoli da computer, mentre l'interrogazione esterna ha elencato i nomi di quei clienti che non sono stati restituiti da quella interna.

Altre due condizioni associate all'uso delle sotto-interrogazioni sono EXISTS e NOT EXISTS. Queste parole chiave sono incluse in un'interrogazione SQL nello stesso punto in cui si troverebbe IN, appena prima dell'inizio della sotto-interrogazione. EXISTS è vera se la sotto-interrogazione restituisce una tabella di risultati intermedi che contiene una o più righe, e falsa se non vengono restituite righe. NOT EXISTS è vera se non vengono restituite righe, e falsa se vengono restituite una o più righe. Si consideri la seguente istruzione SQL, che include EXISTS.

**Interrogazione:** Quali sono i numeri d'ordine per tutti gli ordini in cui sono inclusi mobili con rifiniture in frassino naturale?

```
SELECT DISTINCT ORDER_ID FROM ORDER_LINE_T
WHERE EXISTS
 (SELECT *
 FROM PRODUCT_T
 WHERE PRODUCT_ID = ORDER_LINE_T.PRODUCT_ID
 AND PRODUCT_FINISH = 'Natural Ash');
```

La sotto-interrogazione controlla se la rifinitura del prodotto su una linea d'ordine è in frassino naturale. L'interrogazione principale estrae i numeri d'ordine per tutti gli ordini in cui sono inclusi mobili rifiniti in frassino naturale. Come mostra il risultato, di questi ordini ne sono stati effettuati sette (discuteremo questa interrogazione successivamente nella Figura 10.2b).

**Risultato:**

| <u>ORDER_ID</u> |
|-----------------|
| 1001            |
| 1002            |
| 1003            |
| 1006            |
| 1007            |
| 1008            |
| 1009            |

7 rows selected.

Quando EXISTS o NOT EXISTS sono utilizzate in una sotto-interrogazione, l'elenco di selezione della sotto-interrogazione selezionerà semplicemente tutte le colonne (SELECT \*) come un segnaposto, poiché non interessa sapere quali colonne vengono restituite. Lo scopo della sotto-interrogazione è quello di verificare se c'è qualche colonna che soddisfa le condizioni e non quello di restituire valori dalle colonne. Le colonne che verranno visualizzate sono determinate dall'interrogazione esterna. Una sotto-interrogazione EXISTS è quasi sempre una sotto-interrogazione correlata, come si vedrà di seguito. Le interrogazioni che contengono la parola chiave NOT EXISTS restituiscono una tabella di risultati in cui non si trovano righe che soddisfano la sotto-interrogazione.

Per riassumere, è preferibile utilizzare l'approccio delle sotto-interrogazioni quando le qualifiche sono nidificate o quando sono più comprensibili in una forma nidificata. Laddove SQL\*Plus permette a

una sotto-interrogazione di restituire più di una colonna, la maggior parte dei sistemi consente l'unione a coppie di una e solo una colonna in un'interrogazione interna con una colonna in un'interrogazione esterna. Un'eccezione a questa regola è quando una sotto-interrogazione viene utilizzata con la parola chiave EXISTS. I dati possono essere visualizzati solo dalle tabelle referenziate nell'interrogazione esterna. Solitamente vengono supportati fino a 16 livelli di annidamento. Le interrogazioni vengono elaborate dalla più interna alla più esterna, sebbene esistano le sotto-interrogazioni correlate, che vengono elaborate in senso inverso.

**Figura 10.2**  
Elaborazione delle  
sotto-interrogazioni.  
(a) Elaborazione di una  
sotto-interrogazione  
non correlata.

SELECT CUSTOMER\_NAME  
FROM CUSTOMER\_T  
WHERE CUSTOMER\_ID IN  

(SELECT DISTINCT CUSTOMER\_ID  
FROM ORDER\_T);

1. La sotto-interrogazione (mostrata nel riquadro) viene elaborata per prima; viene quindi creata una tabella di risultati intermedi.

CUSTOMER\_ID

1  
8  
15  
5  
3  
2  
11  
12  
4

9 rows selected.

2. L'interrogazione esterna restituisce le informazioni richieste sui clienti per ogni cliente incluso nella tabella dei risultati intermedi.

CUSTOMER\_NAME

Contemporary Casuals  
Value Furniture  
Home Furnishings  
Eastern Furniture  
Impressions  
California Classics  
American Euro Lifestyles  
Battle Creek Furniture  
Mountain Scenes

9 rows selected.

(b) Elaborazione di una sotto-interrogazione correlata.

```
SELECT DISTINCT ORDER_ID FROM ORDER_LINE_T
WHERE EXISTS
 (SELECT *
 FROM PRODUCT_T
 WHERE PRODUCT_ID = ORDER_LINE_T.PRODUCT_ID
 AND PRODUCT_FINISH = 'Natural Ash');
```

| Order_Id | Product_Id | Ordered_Quantity |
|----------|------------|------------------|
| 1001     | 1          | 1                |
| 1001     | 2          | 2                |
| 1001     | 4          | 1                |
| 1002     | 3          | 5                |
| 1002     | 3          | 3                |
| 1004     | 6          | 2                |
| 1004     | 8          | 2                |
| 1005     | 4          | 4                |
| 1006     | 4          | 1                |
| 1006     | 5          | 2                |
| 1007     | 1          | 3                |
| 1007     | 2          | 2                |
| 1009     | 3          | 3                |
| 1009     | 8          | 3                |
| 1009     | 4          | 2                |
| 1009     | 7          | 3                |
| 1010     | 8          | 10               |
| 0        | 0          | 0                |

| Product_ID | Product_Description  | Product_Finish | Standard Price | Product_Line_Id |
|------------|----------------------|----------------|----------------|-----------------|
| 1          | End Table            | Cherry         | \$175.00       | 10001           |
| 2          | Coffee Table         | Natural Ash    | \$200.00       | 20001           |
| 3          | Computer Desk        | Natural Ash    | \$375.00       | 20001           |
| 4          | Entertainment Center | Natural Maple  | \$650.00       | 30001           |
| 5          | Writer's Desk        | Cherry         | \$325.00       | 10001           |
| 6          | 8-Drawer Dresser     | White Ash      | \$750.00       | 20001           |
| 7          | Dining Table         | Natural Ash    | \$800.00       | 20001           |
| 8          | Computer Desk        | Walnut         | \$250.00       | 30001           |
| *          | (AutoNumber)         |                | \$0.00         |                 |

1. Il primo ID d'ordine viene selezionato da ORDER\_LINE\_T: ORDER\_ID = 1001.
2. La sotto-interrogazione verifica se qualche prodotto nell'ordine ha una rifinitura in frassino naturale. Product 2 ce l'ha, e fa parte dell'ordine. EXISTS viene valutata come vera e l'ID d'ordine viene aggiunto alla tabella dei risultati.
3. L'ID d'ordine successivo viene selezionato da ORDER\_LINE\_T: ORDER\_ID = 1002.
4. La sotto-interrogazione verifica se il prodotto ordinato ha una rifinitura in frassino naturale. Ce l'ha. EXISTS viene valutata come vera e l'ID d'ordine viene aggiunto alla tabella dei risultati.
5. L'elaborazione continua su tutti gli altri ID d'ordine. Gli ordini 1004, 1005 e 1010 non sono inclusi nella tabella dei risultati perché non hanno rifiniture in frassino naturale. La tabella dei risultati finali è l'ultima mostrata nel paragrafo "Sotto-interrogazioni".

10.1.7 Sotto-interrogazioni correlate

Nel primo esempio di sotto-interrogazione era necessario esaminare l'interrogazione interna prima di considerare quella esterna. Il suo risultato era infatti utilizzato per limitare l'elaborazione dell'interrogazione esterna. Per contro, le sotto-interrogazioni correlate utilizzano il risultato dell'interrogazione esterna per determinare l'elaborazione di quella interna. L'interrogazione interna è in qualche modo diversa per ciascuna riga referenziata nell'interrogazione esterna. In questo caso, l'interrogazione interna deve essere calcolata per ciascuna riga esterna, mentre negli esempi precedenti era calcolata solo

una volta per tutte le righe elaborate nell’interrogazione esterna. Le Figure 10.2a e 10.2b illustrano il diverso ordine di elaborazione per ciascuno degli esempi riportati nel paragrafo precedente sulle sotto-interrogazioni.

L’interrogazione riportata di seguito elenca i dettagli del prodotto che ha un prezzo unitario più alto di qualsiasi altro prodotto in PRODUCT\_T.

**Interrogazione:**    Elencare i dettagli del prodotto che ha il prezzo unitario più alto.

```
SELECT PRODUCT_NAME, PRODUCT_FINISH, UNIT_PRICE
FROM PRODUCT_T PA
WHERE UNIT_PRICE > ALL
 (SELECT UNIT_PRICE FROM PRODUCT_T PB
 WHERE PB.PRODUCT_ID != PA.PRODUCT_ID);
```

Ecco il risultato: il tavolo da pranzo ha un prezzo unitario più alto di qualsiasi altro prodotto.

**Risultato:**

| PRODUCT_NAME | PRODUCT_FINISH | UNIT_PRICE |
|--------------|----------------|------------|
| Dining Table | Natural Ash    | 800        |

La logica di questa istruzione SQL è che la sotto-interrogazione viene eseguita una volta per ogni prodotto per essere certi che nessun altro prodotto abbia un prezzo unitario più alto. Si noti che si sta confrontando una tabella a sé stessa e che è possibile farlo perché alla tabella sono stati assegnati due alias, PA e PB. PRODUCT\_ID 1, la tabella finale, viene considerata per prima. Quando la sotto-interrogazione viene eseguita, restituisce i prezzi di ciascun prodotto a eccezione di quello considerato nell’interrogazione esterna. Quest’ultima controllerà se il prezzo unitario per il prodotto considerato è maggiore di tutti quelli restituiti dalla sotto-interrogazione. In caso affermativo, il prezzo verrà restituito come risultato dell’interrogazione. In caso negativo, verrà considerato il valore successivo nell’interrogazione esterna, e l’interrogazione interna restituirà un elenco di tutti i prezzi unitari per gli altri prodotti. Questo elenco cambia quando ogni prodotto nell’interrogazione esterna cambia; è questo che crea una sotto-interrogazione correlata. Si è in grado di identificare un insieme speciale di prezzi unitari per i quali questa interrogazione non darà il risultato desiderato?

### 10.1.8 Utilizzare le tabelle derivate

Le sotto-interrogazioni non si limitano all’inclusione della clausola WHERE. Possono essere utilizzate anche nella clausola FROM, creando una tabella derivata temporanea che viene utilizzata nell’interrogazione. La creazione di una tabella derivata che contiene un valore aggregato, come MAX, AVG o MIN, consente che l’aggregato venga utilizzato nella clausola WHERE. Nell’esempio che segue, sono elencati alcuni prodotti che superano il prezzo standard medio.

**Interrogazione:**    Quali prodotti hanno un prezzo standard che è superiore a quello medio?

```
SELECT PRODUCT_DESCRIPTION, STANDARD_PRICE, AVGPRICE
FROM
 (SELECT AVG(STANDARD_PRICE) AVGPRICE FROM PRODUCT_T),
PRODUCT_T
WHERE STANDARD_PRICE > AVGPRICE;
```

**Risultato:**

| PRODUCT_DESCRIPTION  | STANDARD_PRICE | AVGPRICE |
|----------------------|----------------|----------|
| Entertainment Center | 650            | 440.625  |
| 8-Drawer Dresser     | 750            | 440.625  |
| Dining Table         | 800            | 440.625  |

**10.1.9 Unire le interrogazioni**

La clausola UNION viene utilizzata per unire l'output di più tabelle in un'unica tabella di risultati. Per poter utilizzare questa clausola, tutte le interrogazioni coinvolte devono generare lo stesso numero di righe e devono essere compatibili con l'unione. Questo significa che l'output di ogni interrogazione per ciascuna colonna deve essere costituito da tipi di dati compatibili. La compatibilità riconosciuta ai tipi di dati varia a seconda dei prodotti DBMS. Quando si esegue un'unione in cui una colonna unisce due tipi di dati diversi, è più sicuro utilizzare il comando CAST per controllare direttamente la conversione del tipo di dati. Per esempio, il tipo di dati DATE in ORDER\_T potrebbe dover essere convertito in un tipo di dati di testo. Il comando per svolgere l'operazione è il seguente:

```
SELECT CAST(ORDER_DATE AS CHAR) FROM ORDER_T;
```

L'interrogazione d'esempio mostrata di seguito determina i clienti che hanno acquistato la quantità più grande di qualsiasi prodotto di Pine Valley e i clienti che hanno acquistato la quantità più piccola, e restituisce i risultati in un'unica tabella.

**Interrogazione:**

```
SELECT C1.CUSTOMER_ID,CUSTOMER_NAME,ORDERED_QUANTITY, 'LARGEST
QUANTITY' QUANTITY
 FROM CUSTOMER_T C1,ORDER_T O1, ORDER_LINE_T Q1
 WHERE C1.CUSTOMER_ID =O1.CUSTOMER_ID
 AND O1.ORDER_ID =Q1.ORDER_ID
 AND ORDERED_QUANTITY =
 (SELECT MAX(ORDERED_QUANTITY)
 FROM ORDER_LINE_T)
UNION
SELECT C1.CUSTOMER_ID,CUSTOMER_NAME,ORDERED_QUANTITY, 'SMALLEST
QUANTITY'
 FROM CUSTOMER_T C1,ORDER_T O1, ORDER_LINE_T Q1
 WHERE C1.CUSTOMER_ID =O1.CUSTOMER_ID
 AND O1.ORDER_ID =Q1.ORDER_ID
 AND ORDERED_QUANTITY =
 (SELECT MIN(ORDERED_QUANTITY)
 FROM ORDER_LINE_T)
ORDER BY ORDERED_QUANTITY
```

Si noti che è stata creata un'espressione QUANTITY nella quale le stringhe 'Smallest Quantity' e 'Largest Quantity' sono state inserite per leggibilità. La clausola ORDER BY è stata utilizzata per organizzare l'ordine con il quale vengono elencate le righe di output.

**Risultato:**

| CUSTOMER_ID | CUSTOMER_NAME        | ORDERED_QUANTITY | QUANTITY          |
|-------------|----------------------|------------------|-------------------|
| 1           | Contemporary Casuals | 1                | Smallest Quantity |
| 2           | Value Furniture      | 1                | Smallest Quantity |
| 1           | Contemporary Casuals | 10               | Largest Quantity  |

### 10.1.10 Espressioni condizionali

Per stabilire un'elaborazione logica IF-THEN-ELSE in un'istruzione SQL statement si può ora utilizzare la parola chiave CASE in un'istruzione. La Figura 10.3 illustra la sintassi di CASE, che ha quattro forme. La forma di CASE può essere costruita utilizzando un'espressione che equivale sia a un valore sia a un predicato. La forma del predicato si basa su una logica a tre valori (vero, falso, non si sa), ma consente operazioni più complesse. La forma dell'espressione del valore richiede una corrispondenza con il valore dell'espressione. NULLIF e COALESCE sono le parole chiave associate ad altre due forme dell'espressione CASE.

**Figura 10.3**

Sintassi condizionale di CASE.

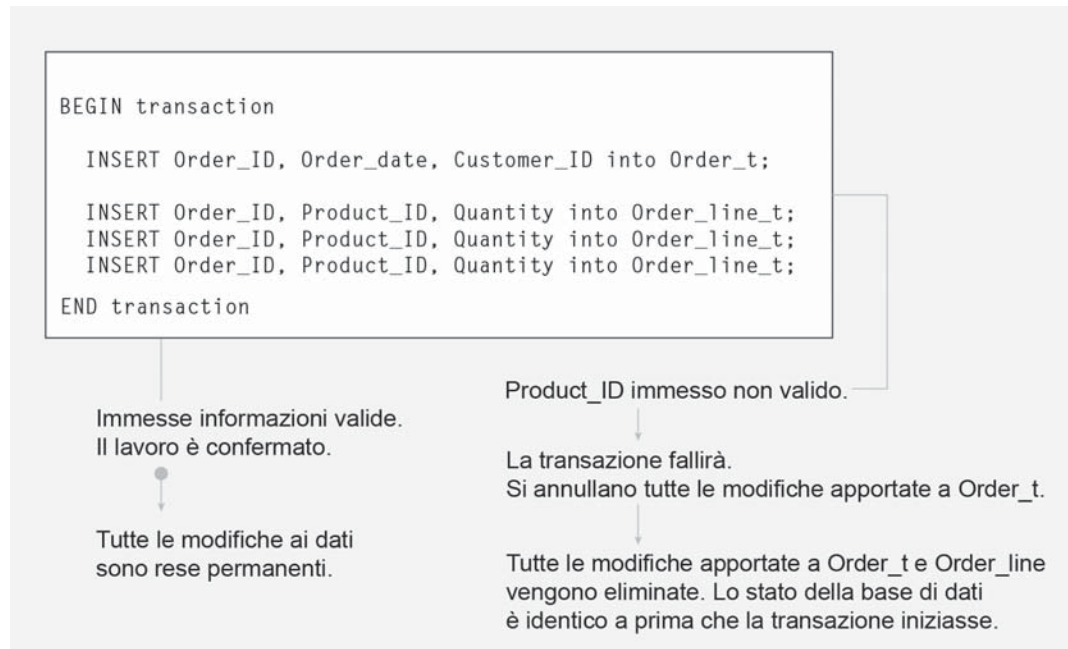
```
{CASE espressione
{WHEN espressione
THEN { espressione | NULL}}...
| {WHEN predicato
THEN { espressione | NULL}}...
[ELSE { espressione NULL}]
END }
| (NULLIF (espressione, espressione))
| (COALESCE (espressione...) }
```

## 10.2 Garantire l'integrità delle transazioni

I DBMS relazionali non sono diversi da altri tipi di gestori di basi di dati per quanto riguarda le responsabilità principali nel garantire che la manutenzione dei dati sia gestita in modo adeguato e completo. La manutenzione dei dati è definita in unità di lavoro chiamate *transazioni*, che implicano uno o più comandi di manipolazione. Una transazione è l'insieme complessivo di comandi di aggiornamento strettamente correlati che devono essere eseguiti, o non devono essere eseguiti, perché la base di dati rimanga valida. Si consideri la Figura 10.4. Per esempio, quando un ordine viene immesso nella base di dati di Pine Valley, tutti gli articoli ordinati dovrebbero essere immessi nello stesso momento. In questo modo devono essere immesse anche tutte le righe di ORDER\_LINE\_T, insieme alle informazioni in ORDER\_T, oppure non dovrebbe essere immessa alcuna riga. Ciò che occorre avere sono dei comandi per definire i confini di una transazione, così da confermare il lavoro di una transazione come modifica permanente alla base di dati, e per interrompere nel modo più corretto una transazione, se necessario. Inoltre, occorrono anche dei servizi di ripristino dei dati per risolvere l'interruzione irregolare dell'elaborazione di una base di dati a metà di una transazione. Può infatti accadere che l'ordine sia preciso, ma che a metà dell'immissione di un ordine, si verifichino dei malfunzionamenti nel computer o si verifichi un calo di energia. In questo caso, si può desiderare che vengano apportate alcune modifiche e non altre. L'obiettivo è quello di avere comunque una base di dati valida.

Quando un singolo comando SQL costituisce una transazione, alcuni RDBMS confermano o annullano automaticamente l'operazione dopo che il comando è stato eseguito. Nel caso di una transazione definita dall'utente, tuttavia, quando devono essere eseguiti, confermati o annullati più comandi SQL, sono necessari comandi espliciti che gestiscano la transazione. Molti sistemi hanno i comandi BEGIN TRANSACTION ed END TRANSACTION, che vengono utilizzati per contrassegnare i confini di

**Figura 10.4**  
Una sequenza  
SQL di transazione  
(pseudocodifica).



un'unità logica di lavoro. END TRANSACTION e COMMIT WORK prendono i contenuti del file di registro e li applicano alla base di dati (rendendo perciò definitive le modifiche), quindi svuotano il file di registro. ROLLBACK WORK chiede a SQL di svuotare il file di registro. In alcuni RDBMS c'è anche un comando AUTOCOMMIT (ON/OFF) che specifica se le modifiche devono essere rese definitive dopo ogni comando di modifica dei dati (ON) o solo quando il lavoro è reso esplicitamente definitivo (OFF) dal comando COMMIT WORK.

Le transazioni definite dagli utenti possono migliorare le prestazioni del sistema perché vengono elaborate come insiemi invece che come singole transazioni, riducendo quindi l'overhead del sistema. Quando AUTOCOMMIT è impostato a OFF, le modifiche non vengono apportate automaticamente fino al termine di ciascuna istruzione SQL; questo non consente che le transazioni definite dagli utenti vengano confermate o annullate solo come un intero.

SET AUTOCOMMIT è un comando interattivo, quindi una data sessione utente può essere controllata dinamicamente per quanto riguarda l'integrità. Ogni comando INSERT, UPDATE e DELETE SQL funziona solitamente su una sola tabella alla volta. La manutenzione dei dati può richiedere l'aggiornamento di più tabelle perché il lavoro sia completo. Questi comandi di integrità delle transazioni sono quindi importanti per definire chiaramente le modifiche alle unità intere della base di dati che devono essere completate per intero perché la base di dati mantenga la sua integrità.

Alcuni sistemi SQL hanno anche dei controlli di concomitanza che gestiscono l'aggiornamento di una base di dati condivisa simultaneamente da più utenti. Tali controlli possono tenere una sorta di "diario" delle modifiche della base di dati, così che questa possa essere recuperata qualora si verificino delle interruzioni improvvise a metà di una transazione. I controlli possono anche annullare transazioni errate. Per esempio, in un'applicazione bancaria, l'aggiornamento di un conto da parte di due utenti simultanei dovrebbe essere cumulativo. Tali controlli sono trasparenti perché in SQL; non è quindi necessaria alcuna programmazione da parte dell'utente per garantire il controllo opportuno dell'accesso



simultaneo ai dati. Per garantire l'integrità di una determinata base di dati, si presti particolare attenzione all'integrità delle transazioni e ai problemi di ripristino e ci si assicuri che i programmatori dell'applicazione siano opportunamente informati su quando tali comandi devono essere utilizzati.

## 10.3 I dizionari dei dati

I RDBMS memorizzano le informazioni dei dati in tabelle create dal sistema; queste tabelle di sistema possono essere considerate come *dizionari dei dati*. Acquisire familiarità con le tabelle di sistema per i RDBMS utilizzati permette di fornire informazioni valide, sia come utenti sia come amministratori del sistema. Poiché le informazioni sono memorizzate in tabelle, è possibile accedervi utilizzando istruzioni SELECT SQL che generano report sull'uso del sistema, i privilegi degli utenti, i vincoli e così via. Un utente che comprende la struttura delle tabelle di sistema può estendere le tabelle esistenti o costruire altre tabelle per migliorare le funzionalità predefinite (per esempio, per includere informazioni su chi è responsabile dell'integrità dei dati). Tuttavia, accade spesso che a un utente non venga permesso di modificare del tutto e direttamente la struttura o le tabelle di sistema, poiché è il DBMS che le mantiene e che dipende da esse per l'interpretazione e l'analisi delle interrogazioni.

Ogni RDBMS mantiene diverse tabelle interne per queste definizioni. In Oracle8i ci sono 98 viste di dizionari dei dati utilizzabili dagli amministratori della base di dati. Molte di queste viste, o sottoinsiemi della vista del DBA (quelle informazioni importanti per un singolo utente), sono messe a disposizione anche degli utenti che non possiedono privilegi di amministratore. I nomi di queste tabelle iniziano con USER o ALL invece che con DBA. Di seguito è riportato un elenco delle tabelle accessibili ai DBA che mantengono le informazioni sulle tabelle, i cluster, le colonne e la protezione. Esistono anche tabelle relative alla memorizzazione, agli oggetti, agli indici, ai blocchi, alle verifiche, alle esportazioni e agli ambienti distribuiti.

|                  |                                                                  |
|------------------|------------------------------------------------------------------|
| DBA_TABLES       | Descrive tutte le tabelle nella base di dati.                    |
| DBA_TAB_COMMENTS | Commenti su tutte le tabelle nella base di dati.                 |
| DBA_CLUSTERS     | Descrive tutti i cluster nella base di dati.                     |
| DBA_TAB_COLUMNS  | Descrive le colonne di tutte le tabelle, le viste e i cluster.   |
| DBA_COL_PRIVS    | Include tutti i privilegi sulle colonne nella base di dati.      |
| DBA_COL_COMMENTS | Commenti su tutte le colonne nelle tabelle e nelle viste.        |
| DBA_CONSTRAINTS  | Definizioni dei vincoli su tutte le tabelle nella base di dati.  |
| DBA_CLU_COLUMNS  | Mappa le colonne di tabella in colonne di cluster.               |
| DBA_CONS_COLUMNS | Informazioni su tutte le colonne nelle definizioni dei vincoli.  |
| DBA_USERS        | Informazioni su tutti gli utenti della base di dati.             |
| DBA_SYS_PRIVS    | Descrive i privilegi di sistema concessi agli utenti e ai ruoli. |
| DBA_ROLES        | Descrive tutti i ruoli che esistono nella base di dati.          |
| DBA_PROFILES     | Include i limiti delle risorse assegnati a ciascun profilo.      |
| DBA_ROLE_PRIVS   | Descrive i ruoli concessi agli utenti e ad altri ruoli.          |
| DBA_TAB_PRIVS    | Descrive tutti i privilegi sugli oggetti nella base di dati.     |

Per avere un'idea del tipo di informazioni che si trovano nelle tabelle di sistema, si consideri DBA\_USERS. Questa tabella contiene le informazioni sugli utenti validi della base di dati, compresi i nomi utente, gli ID utente, le password codificate, il tablespace predefinito, il tablespace temporaneo, la data

di creazione e il profilo assegnato. DBA\_TAB\_COLUMNS ha 15 attributi, che comprendono, tra gli altri, il proprietario di ogni tabella, il nome della tabella, il nome delle colonne, il tipo di dati, la lunghezza dei dati, la precisione e la scala. Di seguito è riportata un'interrogazione SQL su DBA\_TABLES che trova chi è il proprietario di PRODUCT\_T.

**Interrogazione:** Chi è il proprietario della tabella PRODUCT?

```
SELECT OWNER, TABLE_NAME
FROM DBA_TABLES
WHERE TABLE_NAME = 'PRODUCT_T';
```

**Risultato:**

| OWNER     | TABLE_NAME |
|-----------|------------|
| MPRESCOTT | PRODUCT_T  |

## 10.4 Miglioramenti ed estensioni SQL-99

I paragrafi precedenti di questo capitolo e il Capitolo 9 hanno dimostrato la potenza di SQL. I lettori particolarmente interessati alle analisi aziendali possono tuttavia porsi delle domande sulle funzioni di statistica disponibili, che per ora sono piuttosto limitate. I programmatori che hanno familiarità con altri linguaggi potrebbero invece chiedersi come vengono definite le variabili, come viene stabilito il controllo sui flussi o come vengono creati i *tipi di dati definiti dagli utenti* (UDT). E poiché la programmazione è sempre più orientata agli oggetti, come si adatterà SQL? SQL-99 ha esteso SQL fornendo ulteriori funzionalità di programmazione. È stata proposta un'integrazione che renderà standard le funzioni statistiche aggiunte. È così che si inizia a capire come lo standard SQL verrà modificato per adeguarsi ai concetti dell'orientamento agli oggetti.

### 10.4.1 Funzioni analitiche proposte

Una modifica proposta a SQL-99, chiamata Amendment 1, dovrebbe aggiungere una serie di funzioni analitiche chiamate OLAP (*On-Line Analytical Processing*) come estensioni del linguaggio SQL. La maggior parte delle funzioni è già stata implementata in Oracle8i R2, DB2 6.2 e Teradata V2R4. Poiché agli inizi del 2001 questo standard non era ancora stato definito, non è detto che le implementazioni di queste funzioni seguiranno esattamente lo standard proposto. L'integrazione è uno sviluppo importante perché affronta la necessità di avere delle funzionalità analitiche nel motore della base di dati. Le regressioni lineari, le correlazioni e le medie mobili non possono ora essere calcolate senza spostare i dati all'esterno della base di dati. Nel tempo, una volta che la rettifica sarà stata resa ufficiale, le implementazioni dei produttori dovranno aderire strettamente allo standard e diventare sempre più simili tra loro.

Sono state proposte 32 nuove funzioni, elencate nella Tabella 10.1. Tra di esse vi sono funzioni sia statistiche sia analitiche. Funzioni come ROW\_NUMBER e RANK permetteranno di lavorare in modo molto più flessibile con un risultato ordinato. Per le applicazioni di gestione delle basi di dati delle relazioni con i clienti o del marketing, la possibilità di considerare solo le prime  $n$  righe o di suddividere il risultato in gruppi in base alle percentuali sarebbe un'aggiunta apprezzata. Gli utenti possono sperare di raggiungere anche un'elaborazione più efficiente, poiché le funzioni sono portate nel motore della base di dati e sono ottimizzate. Una volta che le funzioni verranno definite come standard, i produttori potranno dipendere da esse e potranno includerne l'uso nelle proprie applicazioni, evitando di dover creare funzioni specifiche all'esterno della base di dati.

Se l'integrazione verrà accettata così come è stata proposta, l'istruzione SQL acquisirà una clausola supplementare, chiamata WINDOW. Questa clausola è costituita da un elenco di definizioni di finestra, ognuna delle quali definisce un nome e una specifica per la finestra. Le specifiche includono il partizionamento, l'ordine e il raggruppamento delle aggregazioni.

Ecco un'interrogazione di esempio tratta dal documento della proposta di integrazione (Zemke e altri, 1999, p. 4):

```
SELECT SH.TERRITORY, SH.MONTH, SH.SALES,
 AVG (SH.SALES) OVER W1 AS MOVING_AVERAGE
FROM SALES_HISTORY AS SH
WINDOW W1 AS (PARTITION BY (SH.TERRITORY)
 ORDER BY (SH.MONTH ASC)
 ROWS 2 PRECEDING);
```

Il nome della finestra è W1, che è definita nella clausola WINDOW che segue la clausola FROM. La clausola PARTITION suddivide le righe in SALES\_HISTORY in base a TERRITORY. In ogni partizione per zona, le righe verranno ordinate in senso crescente in base al mese. Infine, un gruppo di aggregazioni viene definito come la riga corrente e le due righe precedenti della partizione, seguendo l'ordine imposto dalla clausola ORDER BY. Infine, una media mobile delle vendite di ciascuna zona verrà restituita come MOVING\_AVERAGE.

**Tabella 10.1** Funzioni OLAP proposte per l'aggiunta a SQL-99.

|            |                 |             |
|------------|-----------------|-------------|
| CEILING    | PERCENTILE_CONT | REGR_SLOPE  |
| CORR       | PERCENT_RANK    | REGR_SXX    |
| COVAR_POP  | POWER           | REGR_SXY    |
| COVAR_SAMP | RANK            | REGR_SYY    |
| CUME_DIST  | RANGE           | ROW_NUMBER  |
| DENSE_RANK | REGR_AVG        | SQRT        |
| EXP        | REGR_AVGX       | STDDEV_POP  |
| FLOOR      | REGR_AVGY       | STDDEV_SAMP |
| LN         | REGR_COUNT      | VAR_POP     |
| MOVING_AVG | REGR_INTERCEPT  | VAR_SAMP    |
| MOVING_SUM | REGR_R2         |             |

Tratto da "The Extra Mile" di R. Winter, 2000, *Intelligent Enterprise* 3 (10), p. 63.

### 10.4.2 Estensioni di programmazione

Gli standard SQL-92 e precedenti hanno sviluppato le funzionalità di SQL come linguaggio di recupero e manipolazione dei dati e non come linguaggio per la creazione di applicazioni. Il risultato è che SQL è stato utilizzato in combinazione con linguaggi completi a livello di calcolo come C o Java per poter creare programmi aziendali applicativi, procedure o funzioni. SQL-99 estende tuttavia SQL aggiungendo alcune funzioni di programmazione nel Core di SQL, SQL/PSM e SQL/OLB.

Le estensioni che rendono SQL completo a livello di calcolo includono le funzionalità di controllo dei flussi, come le istruzioni IF-THEN, FOR, WHILE, e i cicli (*loop*), che sono contenuti in un pacchetto di estensioni per le specifiche SQL essenziali. Il pacchetto è chiamato **Persistent Stored Modules (SQL/PSM)** perché le funzionalità di creazione ed eliminazione di moduli di programma sono memo-

rizzate al suo interno. “Persistent” significa che un modulo di codice verrà conservato in memoria fino alla sua eliminazione, rendendolo così disponibile per l’esecuzione in più sessioni utente, così come le tabelle vengono conservate fino a che non vengono esplicitamente eliminate. Ogni modulo è memorizzato in uno schema come oggetto di schema. Uno schema non deve avere moduli di programmi, altrimenti avrà dei moduli multipli.

Ogni modulo deve avere un nome, un ID di autorizzazione, un’associazione con un determinato schema, un indicazione del set di caratteri da utilizzare e le eventuali dichiarazioni di tabelle temporanee che saranno necessari nel momento in cui il modulo viene eseguito. Ogni modulo deve contenere una o più procedure SQL, chiamate *programmi*, che eseguono un’istruzione SQL ciascuna quando vengono chiamate. Ogni procedura deve includere una dichiarazione SQLSTATE che agisce come parametro dello stato e che indica se un’istruzione SQL è stata o meno eseguita con successo.

SQL/PSM può essere utilizzato per creare applicazioni o per incorporare procedure e funzioni utilizzando direttamente i tipi di dati SQL. L’uso di SQL/PSM introduce un’impostazione procedurale in SQL, poiché le istruzioni sono elaborate in sequenza. Si ricordi che SQL è un linguaggio non procedurale e non implica alcuna sequenza di esecuzione delle istruzioni. SQL/PSM include numerose istruzioni di controllo di SQL, tra cui le seguenti.

|         |                                                                                                                                                                                                                                                                                                                                    |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CASE    | Un’istruzione che esegue insieme diversi di sequenze SQL, in base a un valore di confronto o al valore di una clausola WHEN, utilizzando condizioni di ricerca o espressioni di valore. La logica è simile a un’espressione CASE SQL, ma termina con END CASE invece che con END e non ha un equivalente della clausola ELSE NULL. |
| IF      | Se un predicato è TRUE, un’istruzione SQL verrà eseguita. L’istruzione termina con ENDIF e contiene le istruzioni ELSE ed ELSEIF per la gestione del controllo del flusso nelle diverse condizioni.                                                                                                                                |
| LOOP    | Fa sì che un’istruzione venga eseguita ripetutamente fino a che esiste una condizione che termina in un’uscita.                                                                                                                                                                                                                    |
| LEAVE   | Istruzione utilizzata per impostare una condizione che termina con l’uscita da un ciclo.                                                                                                                                                                                                                                           |
| FOR     | Istruzione che viene eseguita una volta per ogni riga di un insieme di risultati.                                                                                                                                                                                                                                                  |
| WHILE   | Istruzione che viene eseguita finché è presente una determinata condizione. Incorpora una logica che funziona come un’istruzione LEAVE.                                                                                                                                                                                            |
| REPEAT  | Simile all’istruzione WHILE, ma la condizione viene testata dopo l’esecuzione dell’istruzione SQL.                                                                                                                                                                                                                                 |
| ITERATE | Istruzione utilizzata per riavviare un ciclo.                                                                                                                                                                                                                                                                                      |

SQL/PSM promette di risolvere molte dei ben noti limiti dell’SQL di base. È ancora troppo presto per sapere se i programmatori adotteranno SQL/PSM o se continueranno a utilizzare linguaggio host, invocando SQL attraverso SQL incorporato o attraverso CLI. Il nuovo standard può comunque rendere possibile quanto segue.

- Creare procedure e funzioni all’interno di SQL, rendendo così possibile l’accettazione di parametri di input e di output o la restituzione diretta di un valore.
- Individuare e gestire gli errori all’interno di SQL invece che dover gestire gli errori attraverso un altro linguaggio.
- Creare variabili utilizzando l’istruzione DECLARE che resta entro lo stesso ambito in tutta la procedura, il metodo o la funzione in cui è contenuta.
- Passare gruppi di istruzioni SQL invece che singole istruzioni, migliorando le prestazioni.
- Gestire il problema della mancata corrispondenza di impedenza, nel quale SQL elabora gli insiemi di dati all’interno di moduli, laddove i linguaggi procedurali elaborano le singole righe di dati.

SQL/PSM non è stato ancora ampiamente implementato e per questo non abbiamo incluso esempi estesi della sua sintassi in questo capitolo. PL/SQL di Oracle ha qualche somiglianza con il nuovo standard nei suoi moduli di codice, le istruzioni BEGIN ... END, LOOP e WHILE. Sebbene SQL/PSM non sia ancora molto diffuso, la situazione può cambiare rapidamente.

## 10.5 Trigger e routine

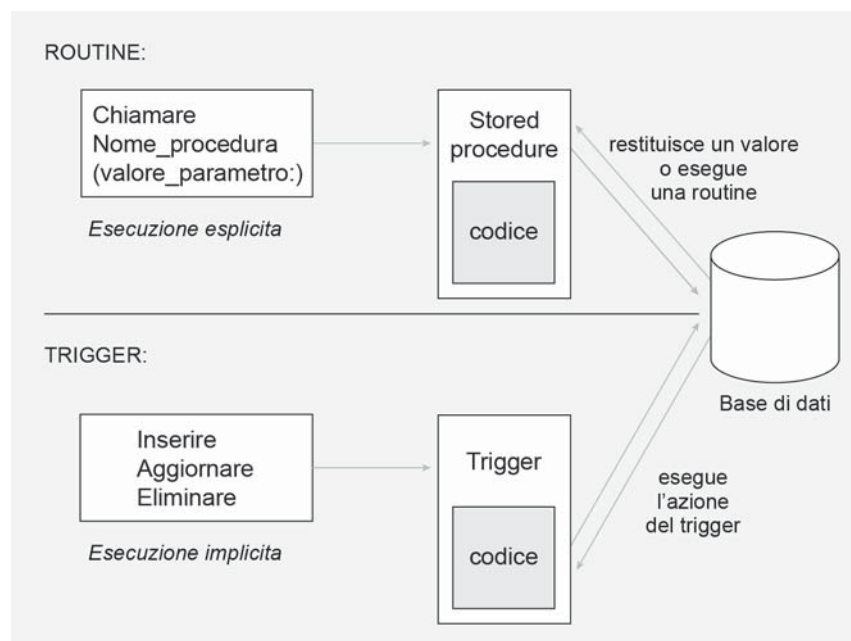
Prima di SQL-99, gli standard SQL non includevano alcun supporto alle funzioni definite dagli utenti né procedure. I prodotti commerciali, riconoscendo la necessità di questo tipo di funzionalità, le hanno fornite per un certo periodo e ci si aspetta di assistere a qualche modifica della loro sintassi nel tempo per un adeguamento ai requisiti di SQL-99, così come ci si aspetta l'inclusione degli standard SQL/PSM.

I *trigger* e le routine sono oggetti molto potenti in una base di dati, perché sono memorizzati al suo interno e perché sono controllati dal DBMS. Il codice necessario per crearli è quindi memorizzato in un unico punto ed è gestito centralmente. Questo favorisce una maggiore integrità dei dati e una coerenza d'uso all'interno della base di dati. Poiché sono memorizzati una sola volta, anche la manutenzione del codice risulta semplificata (Mullins, 1995).

Sia i trigger sia le routine sono costituiti da blocchi di codice procedurale. Il codice dei trigger è memorizzato nella base di dati e viene eseguito automaticamente ogni volta che si verifica un evento di attivazione, come un aggiornamento con UPDATE. Per contro, le routine non vengono eseguite automaticamente. Le routine sono blocchi di codice memorizzati che devono essere chiamati per poter operare (Figura 10.5).

**Figura 10.5**

Differenza tra trigger e stored procedure (adattato da Mullins, 1995).



## 10.5.1 Trigger

Poiché i trigger sono memorizzati ed eseguiti nella base di dati, vengono eseguiti su tutte le applicazioni che accedono alla base di dati. I trigger possono essere anche eseguiti “a cascata”, attivando a loro volta altri trigger. Così, per esempio, una singola richiesta da un client può comportare l’esecuzione di una serie di controlli logici o di integrità sul server senza generare un traffico di rete eccessivo tra client e server. I trigger possono essere utilizzati per garantire l’integrità referenziale, applicare le regole aziendali, creare test di verifica, duplicare le tabelle o attivare una procedura (Rennhackkamp, 1996).

I vincoli possono essere considerati come un tipo speciale di trigger. Vengono applicati (avviati) automaticamente come risultato dei comandi di modifica dei dati, ma la loro sintassi specifica è determinata dal DBMS e non ha la flessibilità di un trigger.

I trigger sono costituiti da tre parti: l’*evento*, la *condizione* e l’*azione*. Queste parti si riflettono nella struttura di codifica dei trigger (si veda la Figura 10.6 per una sintassi di trigger d’esempio). Di seguito è riportato un semplice esempio di un trigger scritto in PL/SQL:

```
CREATE TRIGGER ORDER_ID_BIR
 BEFORE INSERT ON ORDER_T
 FOR EACH ROW
BEGIN
 SELECT ID_SEQUENCE.NEXTVAL
 INTO :NEW.ORDER_ID
 FROM DUAL;
END ORDER_ID_BIR;
```

**Figura 10.6**

Sintassi di trigger in Oracle PL/SQL semplificata.

```
CREATE [OR REPLACE] TRIGGER nome_trigger
 {BEFORE AFTER} {INSERT|DELETE|UPDATE} ON nome_tabella
 [FOR EACH ROW [WHEN (condizione_trigger)]]
 inserire_qui_corpo_trigger;
```

Questo trigger inserisce automaticamente il numero d’ordine ogni volta che viene aggiunto un nuovo ordine. BIR fa parte di una convenzione di denominazione dei trigger nota come *Before Insert Row*. I trigger possono attivarsi prima o dopo l’esecuzione dell’istruzione che ha avviato il trigger. Possono venire attivati in corrispondenza dei comandi INSERT, UPDATE o DELETE; possono essere attivati ogni volta che una riga viene modificata oppure una sola volta per istruzione, a prescindere dal numero di righe modificate. Nel caso illustrato in precedenza, il trigger inserisce il numero d’ordine prima che venga aggiunto l’ordine. A un trigger che controlla il livello delle scorte dopo che queste sono state aggiornate, viene assegnata un’etichetta di denominazione AUR. Il trigger sopra riportato richiede anche che una sequenza chiamata ID\_SEQUENCE sia stata precedentemente definita, poiché il valore successivo nella sequenza verrà utilizzato come numero d’ordine successivo.

Lo sviluppatore che desidera includere i trigger deve prestare molta attenzione. Poiché i trigger si avviano automaticamente, l’utente non saprà che sono stati avviati, a meno che non un prevedano la visualizzazione di un messaggio. I trigger possono anche attivarsi “a cascata” e comportare l’attivazione di altri trigger. Per esempio, un trigger BEFORE UPDATE potrebbe richiedere che una riga venga inserita in un’altra tabella; se quella tabella ha un trigger BEFORE INSERT, questo verrà avviato,

probabilmente con risultati non previsti. È inoltre possibile creare un ciclo infinito di trigger. Poiché i trigger hanno numerose opzioni, compresa l'applicazione di regole aziendali complesse, la creazione di registri di verifica sofisticati e l'applicazione di autorizzazione di protezione elaborate, dovrebbero essere inclusi con attenzione.

## 10.5.2 Routine

Le routine invocate da SQL possono essere procedure o funzioni. I due termini sono utilizzati nello stesso modo in cui sono utilizzati negli altri linguaggi di programmazione. Una **funzione** restituisce un valore e ha solo parametri di input. Una **procedura** può avere parametri di input, di output e parametri che sono sia di input sia di output. Un'unità di codice procedurale può essere dichiarata e denominata utilizzando il codice proprietario del prodotto RDBMS utilizzato o può invocare una routine di libreria del linguaggio host. I prodotti SQL hanno sviluppato proprie versioni di routine precedentemente a SQL-99; occorre quindi avere una certa familiarità con la sintassi e le potenzialità di qualsiasi prodotto che si utilizza. Alcuni di questi linguaggi proprietari, come PL/SQL di Oracle, sono ampiamente utilizzati e continueranno a essere disponibili.

I vantaggi delle routine invocate da SQL includono i seguenti.

- *Flessibilità.* Le routine possono essere utilizzate in più situazioni che non i vincoli o i trigger, che sono limitati ai casi di modifica dei dati. Così come i trigger hanno più opzioni di codice dei vincoli, le routine hanno più opzioni di codice dei trigger.
- *Efficienza.* Le routine possono essere costruite e ottimizzate in modo da operare più velocemente che non le istruzioni SQL generiche, più lente.
- *Condivisione.* Le routine possono essere memorizzate sul server e messe a disposizione di tutti gli utenti, così che non debbano essere riscritte.
- *Applicabilità.* Le routine possono essere applicate all'intera base di dati invece che essere confinate a una sola applicazione. Questo vantaggio è un corollario della condivisione.

La sintassi SQL-99 per la creazione delle procedure e delle funzioni è mostrata nella Figura 10.7. Come si può vedere, la sintassi è complessa e non ci addentreremo nei dettagli di ogni singola clausola. Riportiamo comunque una procedura semplice, che possa dare un'idea di come funziona il codice.

**Figura 10.7**

Sintassi di una routine di creazione (SQL-99).

```
{CREATE PROCEDURE | CREATE FUNCTION} nome_routine
([parametro [{,parametro}...]])
[RETURNS tipo_di_dati_risultato] /* solo per le funzioni */
[LANGUAGE {ADA|C|COBOL|FORTRAN|MUMPS|PASCAL|PLI|SQL}]
[PARAMETER STYLE {SQL|GENERAL}]
[SPECIFIC nome_specifica]
[DETERMINISTIC|NOT DETERMINISTIC]
[NO SQL|CONTAINS SQL|READS SQL DATA|MODIFIES SQL DATA]
[RETURN NULL ON NULL INPUT|CALL ON NULL INPUT]
[DYNAMIC RESULT SETS intero_non_segnato] /* solo per le procedure*/
[STATIC DISPATCH] /* solo per le funzioni*/
corpo_routine
```

Una *procedura* è un raccolta di istruzioni procedurali e SQL a cui è assegnato un nome univoco all'interno dello schema e che viene memorizzata nella base di dati. Quando si desidera eseguire una procedura, occorre chiamarla mediante il suo nome. Quando viene chiamata, tutte le istruzioni al suo interno vengono eseguite. Questa caratteristica delle procedura aiuta a ridurre il traffico di rete, poiché tutte le istruzioni vengono trasmesse contemporaneamente invece che singolarmente.

Per poter costruire una semplice procedura che imposti un prezzo di vendita, si è modificata la tabella PRODUCT\_T esistente aggiungendo una nuova colonna, SALE\_PRICE, che conterrà il prezzo di vendita per i prodotti.

```
ALTER TABLE PRODUCT_T
ADD (SALE_PRICE DECIMAL (6,2));
```

La tabella è modificata.

Questa semplice procedura esegue due istruzioni SQL. I prodotti con uno STANDARD\_PRICE di 400 \$ o superiore vengono scontati del 10 per cento, mentre i prodotti con uno STANDARD\_PRICE inferiore a 400 \$ vengono scontati del 15 per cento. Ecco il modulo di codice Oracle che crea e memorizza la procedura chiamata PRODUCT\_LINE\_SALE:

```
CREATE OR REPLACE PROCEDURE PRODUCT_LINE_SALE
AS BEGIN
 UPDATE PRODUCT_T
 SET SALE_PRICE = .90 * STANDARD_PRICE
 WHERE STANDARD_PRICE >= 400;
 UPDATE PRODUCT_T
 SET SALE_PRICE = .85 * STANDARD_PRICE
 WHERE STANDARD_PRICE < 400;
END;
```

Se la sintassi è stata accettata, Oracle restituisce il commento "Procedure created".

Per eseguire la procedure in Oracle, si utilizza questo comando:

```
SQL> EXEC PRODUCT_LINE_SALE
```

Oracle fornisce questa risposta:

PL/SQL procedure successfully completed. Now PRODUCT\_T contains:

| PRODUCT_LINE | PRODUCT_ID | PRODUCT_DESCRIPTION  | PRODUCT_FINISH | STANDARD_PRICE | SALE_PRICE |
|--------------|------------|----------------------|----------------|----------------|------------|
| 10001        | 1          | End Table            | Cherry         | 175            | 148.75     |
| 20001        | 2          | Coffee Table         | Natural Ash    | 200            | 170        |
| 20001        | 3          | Computer Desk        | Natural Ash    | 375            | 318.75     |
| 30001        | 4          | Entertainment Center | Natural Maple  | 650            | 585        |
| 10001        | 5          | Writers Desk         | Cherry         | 325            | 276.25     |
| 20001        | 6          | 8-Drawer Dresser     | White Ash      | 750            | 675        |
| 20001        | 7          | Dining Table         | Natural Ash    | 800            | 720        |
| 30001        | 8          | Computer Desk        | Walnut         | 250            | 212.5      |

## 10.6 SQL incorporato e SQL dinamico

La funzionalità di SQL/PSM introdotta in SQL-99 fornisce i tipi di estensioni di programmazione che in precedenza sono stati sviluppati utilizzando SQL incorporato o dinamico. Entrambi gli approcci sono stati sviluppati prima di SQL-99 per consentire lo sviluppo di applicazioni di basi di dati. SQL è stato originariamente creato per gestire il solo accesso alle basi di dati e non aveva il controllo del flusso né le altre strutture necessarie per creare un'applicazione. **SQL incorporato** (*Embedded SQL*) si riferisce al



processo di inclusione di istruzioni SQL definite staticamente in un programma scritto con un altro linguaggio, come C, Cobol, Ada o Java. SQL dinamico deduce l'istruzione SQL precisa durante il runtime. I programmatori scrivono su un'API (*Application Programming Interface*) per ottenere un'interfaccia tra i linguaggi.

Al momento della stesura di questo libro, lo standard *Open Database Connectivity* (ODBC) è l'API più utilizzata. Lo standard SQL-99 include la *Call Level Interface/SQL* (SQL/CLI). Entrambe sono scritte in C e si basano sul medesimo standard originario. *Java Database Connectivity* (JDBC) è uno standard industriale utilizzato per la connessione da Java; non è ancora uno standard ISO.

Per incorporare SQL, si devono collocare le istruzioni SQL all'interno del codice sorgente nel programma host, precedute dalla frase EXEC SQL. Un pre-compilatore convertirà le istruzioni SQL che il linguaggio host può interpretare, generando un modulo di richiesta di base di dati (DBRM). Una volta che il file host è compilato nel codice dell'oggetto, un programma linker collega le chiamate del DBMS nel linguaggio host a una libreria del DBMS. Un programma solitamente chiamato BIND elabora le istruzioni SQL nel modulo di accesso, analizzando e convalidando ogni istruzione e generando un piano di accesso che cerca di ottimizzare l'esecuzione delle istruzioni. Con alcuni DBMS, il piano di accesso non viene determinato al momento dell'esecuzione, prendendo in considerazione lo stato corrente della base di dati. Il piano di accesso e le eventuali ulteriori informazioni necessarie sono memorizzati nella base di dati o nel modulo di codice esterno, un'immagine eseguibile. È possibile accedere all'immagine e utilizzarla con qualsiasi istanza della base di dati, mentre la memorizzazione nella base di dati limita l'accesso alla medesima. L'uso di SQL incorporato comporta in ogni caso un'elaborazione più efficiente rispetto a SQL interattivo, perché sfrutta le informazioni memorizzate nella libreria. Una volta scritti, i piani di accesso, le chiamate delle procedure e così via risulteranno trasparenti per lo sviluppatore.

SQL dinamico (*Dynamic SQL*) è utilizzato per generare un codice SQL appropriato al volo, mentre un'applicazione sta elaborando le informazioni. La maggior parte dei programmatori scrive su un'API, come ODBC, che può quindi essere passata a una qualsiasi base di dati compatibile con ODBC. SQL dinamico è fondamentale per la maggior parte delle applicazioni Internet e verrà trattato nel Capitolo 12. Lo sviluppatore è in grado di creare un'applicazione più flessibile perché l'esatta interrogazione SQL viene determinata durante il runtime, includendo il numero di parametri da passare, le tabelle a cui accedere e così via. SQL dinamico è molto utile quando una shell di istruzioni SQL viene utilizzata ripetutamente, con valori di parametro diversi da inserire a ogni esecuzione.

Man mano che SQL-99 verrà implementato in modo più completo, l'uso di SQL incorporato e dinamico diventerà maggiormente standardizzato, poiché lo standard creerà per la prima volta un linguaggio SQL di calcolo completo. Poiché la maggior parte dei produttori si è portata avanti e ha creato queste funzionalità in modo indipendente, i prossimi anni saranno probabilmente un periodo in cui esisteranno prodotti compatibili con SQL-99 in parallelo con versioni più vecchie ma consolidate. Gli utenti dovranno essere consapevoli di queste possibilità e dovranno essere in grado di gestirne l'uso.

## Riepilogo

Questo capitolo è una prosecuzione del Capitolo 9, che ha introdotto il linguaggio SQL. Sono stati considerati gli equi-join i join naturali, quelli esterni e quelli di unione. Gli equi-join si basano su valori uguali nelle colonne comuni di tabelle che devono essere unite e restituiscono tutti i risultati richiesti compresi i valori delle colonne comuni per ciascuna tabella inclusa nel join. I join naturali restituiscono tutti i risultati richiesti, ma i valori delle colonne comuni sono inclusi una volta sola. I join esterni restituiscono tutti i valori in una delle tabelle incluse nel join a prescindere dall'esistenza di una corrispondenza nell'altra tabella. I join di unione restituiscono una tabella che include tutti i dati da ogni tabella che è stata unita.

Le sotto-interrogazioni nidificate, in cui più istruzioni SELECT sono nidificate in un'unica interrogazione, sono utili in situazioni di interrogazione complesse. Una forma speciale di sotto-interrogazione richiede che un valore sia conosciuto dall'interrogazione esterna prima che quella interna venga elaborata. Altre sotto-interrogazioni elaborano l'interrogazione interna, restituiscono un risultato nell'interrogazione esterna successiva, e quindi elaborano l'interrogazione esterna.

Altri argomenti SQL avanzati includono l'uso di SQL incorporato e dei trigger e delle routine. SQL può essere incluso all'interno di molti linguaggi di terza generazione, compresi COBOL, C, Fortran e Ada. L'uso di SQL incorporato consente lo sviluppo di interfacce più flessibili, di prestazioni migliorate e di una protezione migliore della base di dati. Le funzioni definite dagli utenti che vengono eseguite automaticamente quando i record sono inseriti, aggiornati o eliminati sono chiamate trigger. Le procedure sono moduli di codice definiti dagli utenti che devono essere chiamate per poter essere eseguite.

Recentemente è stata proposta un'integrazione a SQL-99 che aggiunge una serie di funzioni analitiche. Le estensioni già incluse rendono ora SQL completo dal punto di vista dei calcoli, e includono funzionalità di controllo del flusso in un insieme di specifiche SQL note come *Persistent Stored Modules* (SQL/PSM). SQL/PSM possono essere utilizzate per creare le applicazioni o per incorporare procedure e funzioni utilizzando direttamente i tipi di dati SQL. Le routine invocate da SQL, compresi i trigger, le funzioni e le procedure, sono incluse anche in SQL-99. È importante sapere che queste funzionalità sono state incluse in precedenza da alcuni produttori come estensioni specifiche e che continueranno a esistere ancora per diverso tempo.

Il Capitolo 10 presenta alcune delle funzioni SQL più complesse e illustra le potenzialità SQL più estese e complesse che devono essere padroneggiate per poter costruire programmi di applicazione di basi di dati.

Ripasso del capitolo

Parole chiave

|           |                                           |                                |                                   |
|-----------|-------------------------------------------|--------------------------------|-----------------------------------|
| Equi-join | Join esterno                              | Procedura                      | Tipo di dati definiti dall'utente |
| Funzione  | Join naturale                             | Sotto-interrogazione correlata | (UDT)                             |
| Join      | <i>Persistent Stored Module</i> (SQL/PSM) | SQL dinamico                   | Trigger                           |
|           |                                           | SQL incorporato                |                                   |

Domande di ripasso

1. Definire ciascuno dei seguenti termini chiave:

a. SQL dinamico

b. sotto-interrogazione correlata

c. SQL incorporato

d. procedura

e. join

f. equi-join

g. join naturale

h. join esterno

i. funzione

j. *Persistent Stored Modules* (SQL/PSM)
2. Unire i seguenti termini alle rispettive definizioni:

\_\_\_\_\_ equi-join; \_\_\_\_\_ natural join;

\_\_\_\_\_ join esterno; \_\_\_\_\_ trigger; \_\_\_\_\_ procedura;
- \_\_\_\_\_ SQL incorporato; \_\_\_\_\_ UDT;

\_\_\_\_\_ COMMIT; \_\_\_\_\_ SQL/PSM;

\_\_\_\_\_ SQL dinamico; \_\_\_\_\_ ROLLBACK.

a. le modifiche alle tabelle vengono annullate

b. tipo di dati definito dall'utente

c. estensione di SQL-99

d. restituisce tutti i record della tabella indicata

e. le colonne ridondanti vengono conservate

f. le modifiche alle tabelle vengono rese definitive

g. processo che include istruzioni SQL in un linguaggio host

h. processo per rendere un'applicazione in grado di generare un codice SQL specifico al volo

i. le colonne ridondanti non vengono conservate

j. insieme di istruzioni SQL che vengono eseguite in determinate condizioni

- k. raccolta memorizzata e denominata di istruzioni procedurali e SQL
- 3. Quando viene utilizzato un join esterno invece di un join naturale?
- 4. Spiegare l'ordine di elaborazione di una sotto-interrogazione correlata.
- 5. Spiegare la seguente affermazione riguardo a SQL: "Qualsiasi interrogazione che può essere scritta utilizzando l'approccio delle sotto-interrogazioni può essere scritta anche utilizzando l'approccio dell'unione, ma non viceversa".
- 6. Qual è lo scopo del comando COMMIT in SQL? In che termini è legato alle transazioni aziendali (come l'immissione di un ordine o l'emissione di una fattura)?
- 7. Quando si scrivono i trigger per una base di dati occorre prestare molta attenzione. Quali sono i problemi che si possono incontrare?
- 8. Spiegare la struttura di un modulo di codice che definisce un trigger.
- 9. In quali casi può essere utilizzata una clausola UNION?
- 10. Discutere l'Amendment 1 proposto per SQL-99.
- 11. Spiegare lo scopo di SQL/PSM.
- 12. Citare quattro vantaggi delle routine invocate da SQL.
- 13. Quando si può pensare di utilizzare SQL incorporato e quando SQL dinamico?

### Problemi ed esercizi

I problemi ed esercizi da 1 a 5 si basano sul diagramma E-R illustrato nella Figura 6.16 nel Capitolo 6. Le relazioni 3NF con alcuni dati di esempio vengono ripetute nella Figura 10.8. Per le domande da 1 a 5, si disegni un diagramma di istanza e lo si contrassegni in modo che mostri i dati che ci si aspetta vengano restituiti dall'interrogazione.

- 1. Scrivere dei comandi SQL di recupero per ognuna delle seguenti interrogazioni.
  - a. Visualizzare l'ID del corso e il nome del corso per tutti i corsi che hanno un prefisso ISM.
  - b. Visualizzare tutti i corsi per i quali il professor Berndt è stato qualificato.
  - c. Visualizzare il registro di classe, includendo il nome di tutti gli studenti iscritti al seminario 2714 del corso ISM 4212.
- 2. Scrivere un'interrogazione SQL per rispondere alla seguente domanda: Quali insegnanti sono qualificati per insegnare nel corso ISM 3113?
- 3. Scrivere un'interrogazione SQL per rispondere alla seguente domanda: C'è qualche insegnante qualificato per insegnare nel corso ISM 3113 e non qualificato per insegnare nel corso ISM 4930?
- 4. Scrivere un'interrogazione SQL per rispondere alle seguenti domande: Quanti studenti sono iscritti al seminario 2714 nel

I semestre del 2001? Quanti studenti sono iscritti al corso ISM 3113 nel I semestre del 2001?

- 5. Scrivere un'interrogazione SQL per rispondere alla seguente domanda: Quali studenti non sono iscritti ad alcun corso nel I semestre del 2001?  
I problemi ed esercizi da 6 a 12 si basano sull'intera base di dati di Pine Valley Furniture.
- 6. Scrivere un comando SQL per visualizzare il numero d'ordine, il numero di cliente, la data d'ordine e gli articoli ordinati per l'ordine numero 1001.
- 7. Scrivere un comando SQL per visualizzare ogni articolo ordinato nell'ordine numero 1001, il suo prezzo standard e il prezzo totale per ogni articolo ordinato.
- 8. Scrivere un comando SQL per visualizzare il costo dell'ordine numero 1001.
- 9. Scrivere un comando SQL che trovi tutti i clienti che non hanno effettuato alcun ordine.
- 10. Scrivere un'interrogazione SQL per produrre un elenco di tutti i prodotti e il numero di volte in cui un articolo è stato ordinato.
- 11. Scrivere un'interrogazione SQL per visualizzare il numero di cliente, il nome e il numero d'ordine per tutti i clienti e i loro ordini.
- 12. Scrivere un'interrogazione SQL per elencare il numero d'ordine e la quantità d'ordine per tutti gli ordini dei clienti nei quali la quantità ordinata è maggiore della quantità media di quel prodotto. *Suggerimento:* l'esercizio implica l'uso di una sotto-interrogazione correlata.

### Esercitazioni

- 1. Eseguire una ricerca sul Web per trovare quanti più collegamenti possibili che trattano gli standard SQL.
- 2. Confrontare due versioni di SQL a cui si ha accesso, come Microsoft Access e Oracle SQL\*Plus. Individuare almeno cinque somiglianze nel codice SQL utilizzato e tre differenze. Le differenze comportano risultati diversi?

### Bibliografia

- Codd, E. F., 1970, "A Relational Model of Data for Large Relational Databases", *Communications of the ACM* 13 (giugno), pp. 77-87.
- Date, C. J. e Darwen, H., 1997, *A Guide to the SQL Standard*, Reading, MA, Addison-Wesley.
- Melton, J., 1997, "A Case for SQL Conformance Testing", *Database Programming & Design* 10 (7), pp. 66-69.
- Mullins, C. S., 1995, "The Procedural DBA", *Database Programming & Design* 8 (12), pp. 40-45.
- Rennhackkamp, M., 1996, "Trigger Happy", *DBMS* 9 (5), pp. 89-91, 95.
- Van der Lans, R. F., 1993, *Introduction to SQL*, seconda edizione, Workingham, England, Addison-Wesley.
- Winter, R., 2000, "SQL-99's New OLAP Functions", *Intelligent Enterprise* 3 (2) (20 gennaio), pp. 62, 64-65.

Winter, R., 2000, “The Extra Mile”, *Intelligent Enterprise* 3 (10) (26 giugno), pp. 62–64.

Zemke, F., Kulkarni, K., Witkowski, A. e Lyle, B., 1999, “Introduction to OLAP Functions”, *ISO/IEC JTC1/SC32 WG3*, YGS, nnn ANSI NCITS H2-99-154.

### **Per approfondire**

American National Standards Institute, <http://www.ansi.org>, 1996 (informazioni su ANSI e sugli standard nazionali e internazionali più recenti).

ANSI Standards Action, Vol. 31, n. 11, 6/2/2000, p. 20, American National Standards Institute, NY (si veda anche il paragrafo “Per approfondire” del Capitolo 9).

Welcome to ISO Online, <http://www.iso.ch>, 1997 (informazioni sull’*International Organization for Standardization*).

Si utilizzi il progetto di base di dati SQL implementato nel Capitolo 9 per Mountain View Community Hospital per completare le domande e gli esercizi sul progetto.

## Project Case



# MOUNTAIN VIEW COMMUNITY HOSPITAL

### Domande sul progetto

1. Quale versione di SQL verrà utilizzata per gli esercizi sul progetto?
2. Il proprio DBMS basato su SQL supporta SQL dinamico, le funzioni, le stored procedure e i tipi di dati definiti dall'utente?

### Esercizi sul progetto

1. Scrivere e testare alcune interrogazioni che verranno utilizzate con i propri dati d'esempio. Scrivere alcune interrogazioni che illustrino le interrogazioni più complesse trattate nel Capitolo 10 e che:
  - a. selezionino informazioni da due o più tabelle;
  - b. utilizzino la sintassi delle sotto-interrogazioni;
  - c. restituiscano una tabella di risultati che possa essere utilizzata per generare un report dell'ospedale, per esempio sul personale infermieristico assegnato a ogni unità di cura;
  - d. utilizzino un'istruzione UNION.
2. Scrivere interrogazioni che determinino quanto segue (visualizzare le colonne che interessano).
  - a. Per un dato medico, quali trattamenti ha somministrato quel medico a ciascun paziente che fa riferimento a quel medico dell'ospedale?
  - b. Per l'interrogazione in a., includere anche i medici a cui non fa riferimento alcun paziente dell'ospedale.
  - c. Per ogni paziente, qual è il numero dei trattamenti somministrati a ciascun paziente da ogni medico che ha trattato il paziente?
  - d. Per ogni infermiere, qual è il numero totale di ore lavorate da tutti i dipendenti che lavorano nell'unità di cura di cui l'infermiere è supervisore?

## CAPITOLO

# 11

## L'ambiente client/server delle basi di dati

**Al termine di questo capitolo sarete in grado di fare quanto segue.**

1. Definire i seguenti termini chiave: sistemi client/server, file server, client spesso, server della base di dati, stored procedure, architettura a tre livelli, client sottile, partizionamento delle applicazioni, *Symmetric Multiprocessing* (SMP), *Massively Parallel Processing* (MPP)/architettura di non condivisione, middleware, *Application Program Interface* (API), *Query-by-Example* (QBE), standard *Open Database Connectivity* (ODBC), *Visual Basic for Applications* (VBA), sistema guidato dagli eventi.
2. Elencare i vantaggi principali dell'architettura client/server rispetto ad altri approcci di calcolo.
3. Descrivere tre componenti della logica delle applicazioni: servizi di presentazione dei dati, servizi di elaborazione e servizi di memorizzazione.
4. Suggerire le possibilità di suddivisione di questi servizi in diverse architetture client/server.
5. Distinguere tra un file server, un database server, un'architettura a tre livelli e una a  $n$  livelli.
6. Descrivere il middleware e spiegare come facilita le architetture client/server.
7. Spiegare le potenzialità dell'interfaccia *Query-by-Example* (QBE), il suo rapporto con Microsoft Access 2000 e i suoi vantaggi e svantaggi rispetto a SQL.
8. Spiegare come collegare tabelle di dati esterne a un'applicazione in un ambiente client/server utilizzando ODBC o JDBC.
9. Spiegare come VBA viene utilizzato con Access 2000 per creare applicazioni client.

## Introduzione

I sistemi client/server operano in ambienti di rete, suddividendo l'elaborazione di un'applicazione tra un client front-end e un processore back-end. L'elaborazione del client richiede solitamente alcune risorse che vengono fornite dal server. I client e i server possono risiedere sullo stesso computer o su computer diversi in rete. Sia i client sia i server sono intelligenti e programmabili, così che il potere di calcolo di entrambi possa essere utilizzato per creare applicazioni efficaci ed efficienti.

È difficile sovrastimare l'impatto che le applicazioni client/server hanno avuto negli ultimi 10 anni. I progressi nella tecnologia dei personal computer e la rapida evoluzione delle interfacce grafiche utente (GUI) di rete e delle comunicazioni hanno cambiato il modo in cui le aziende utilizzano i sistemi infor-

matici per soddisfare le proprie esigenze sempre più articolate. Il commercio elettronico richiede che i browser dei client siano in grado di accedere alle pagine Web dinamiche collegate alle basi di dati che forniscono informazioni in tempo reale. I personal computer collegati in rete che supportano operazioni di gruppi di lavoro sono ormai la norma. Le applicazioni mainframe sono state riscritte per poter essere eseguite in ambienti client/server e approfittano dei vantaggi della grande efficienza delle reti di personal computer e stazioni di lavoro. La necessità di strategie che si adattino ad ambienti aziendali specifici è stata soddisfatta dalle soluzioni client/server, che offrono flessibilità, scalabilità (la capacità di aggiornare un sistema senza doverlo riprogettare) ed estensibilità (la capacità di definire nuovi tipi di dati e operazioni). Man mano che le aziende conferiscono alle proprie operazioni una connotazione più globale, devono optare per sistemi distribuiti, che forniscono spesso architetture client/server.

In questo capitolo si analizzerà l'evoluzione degli ambienti di gestione di basi di dati multiutente che hanno portato allo sviluppo di strategie client/server diverse per l'elaborazione dei dati. Tra queste, i DBMS basati su LAN, i DBMS client/server (che comprendono architetture a tre livelli), le architetture parallele, i DBMS Internet e intranet e il middleware.

Ciò che ha guidato la tendenza verso queste tecnologie di base di dati è un'ampia gamma di opportunità e pressioni legate alla competitività e alla simultaneità. In un certo senso, la rete è diventata "il computer" dal momento in cui gli utenti hanno iniziato a condividere i dati tra le unità di rete e a rivolgersi a Internet per accedere alle informazioni e ai dati desiderati. Le ristrutturazioni aziendali, come le fusioni, le acquisizioni e i consolidamenti, rendono necessario connettere, integrare o sostituire le applicazioni indipendenti già esistenti. Una conseguenza del ridimensionamento di un'azienda è che i singoli manager hanno un controllo maggiore, che necessita di un accesso a una gamma di dati più ampia. Le applicazioni sono state scalate da mainframe costosi a microcomputer e stazioni di lavoro in rete che sono più *user-friendly* e più vantaggiose economicamente. La gestione del traffico di rete, che in alcune architetture può diventare eccessivo, è un aspetto fondamentale nello sviluppo di applicazioni client/server di successo, soprattutto se le aziende collocano le applicazioni chiave della loro missione in ambienti distribuiti. La definizione di un buon equilibrio tra i sistemi centrali e decentrati è l'argomento principale più discusso, poiché le aziende si sforzano di ottenere il massimo vantaggio sia dai DBMS client/server sia da quelli basati su mainframe.

## 11.1 Architetture client/server

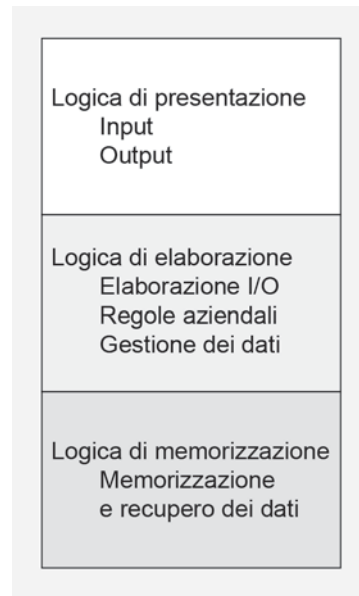
Gli ambienti client/server utilizzano una rete locale (LAN) per supportare una rete di personal computer, ognuno dei quali con la propria capacità di memoria, che è anche in grado di condividere i dispositivi comuni (come un disco rigido o una stampante) e il software (come un DBMS) collegato a una LAN. I PC e le stazioni di lavoro in una LAN sono collocati in genere a tre metri gli uni dalle altre, con una lunghezza totale del cavo di rete solitamente inferiore ai due chilometri. Almeno un PC viene designato come file server, dove è memorizzata la base di dati condivisa. I moduli LAN di un DBMS aggiungono controlli degli accessi simultanei, eventualmente alcune funzioni di protezione supplementari e una gestione delle interrogazioni e delle conversioni delle stesse per supportare l'accesso contemporaneo di più utenti alla base di dati condivisa.

Le architetture server/client che hanno subito un'evoluzione si riconoscono dalla distribuzione dei componenti della logica delle applicazioni tra client e server sono tre (Figura 11.1). Il primo è il componente di input/output (I/O) o di presentazione, responsabile della formattazione e della presentazione dei dati sullo schermo dell'utente o su altri dispositivi di output e della gestione dell'input dell'utente mediante tastiera o altro dispositivo di input. Il secondo componente è quello di elaborazione, che gestisce la logica dell'elaborazione dei dati, quella delle regole aziendali e quella della gestione dei dati.

La logica dell'elaborazione dei dati include attività come la convalida dei dati e l'identificazione degli errori di elaborazione. Le regole aziendali che non sono state codificate a livello del DBMS devono essere codificate in questo componente. La logica di gestione dei dati identifica i dati necessari per l'elaborazione delle transazioni o delle interrogazioni. Il terzo componente è quello della memorizzazione, che è responsabile del salvataggio dei dati e del loro recupero dai dispositivi di memoria fisici associati all'applicazione. Le attività di un DBMS avvengono in questo componente.

**Figura 11.1**

Componenti della logica delle applicazioni.



### 11.1.1 Architetture dei file server

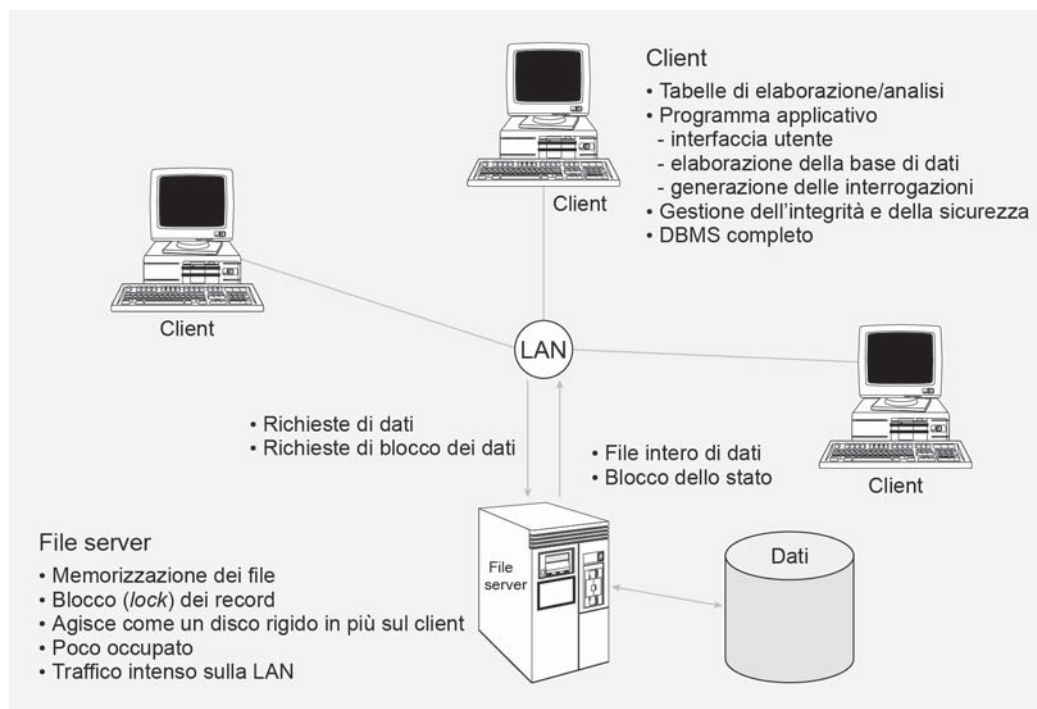
Le prime architetture client/server sviluppate erano file server. In un ambiente di file server (Figura 11.2), tutte le manipolazioni dei dati si verificano sulle stazioni di lavoro nelle quali vengono richiesti i dati. Il client gestisce la logica di presentazione, quella di elaborazione e gran parte di quella di memorizzazione (la parte associata al DBMS). Uno o più file server sono collegati alla LAN. Un file server è un dispositivo che gestisce le operazioni sui file e che è condiviso da ognuno dei PC client collegati alla LAN. Ciascuno di questi file server agisce come un disco rigido supplementare per ognuno dei PC client. Per esempio, il proprio PC può riconoscere un'unità logica F:, che è in realtà un volume di disco memorizzato su un file server sulla LAN. I programmi sul PC fanno riferimento ai file su questa unità utilizzando la classica specifica di tracciato, che include questa unità e le eventuali altre directory, così come il nome di file. Con un file server, ogni PC client può essere chiamato **client spesso** (*fat client*), dove la maggior parte dell'elaborazione avviene sul client invece che su un server.

In un ambiente di file server, ogni PC client è autorizzato a utilizzare il DBMS quando il programma applicativo della base di dati viene eseguito su quel PC. C'è quindi una sola base di dati ma più copie simultanee in esecuzione sul DBMS, una su ciascuno dei PC attivi. La caratteristica principale di un'architettura di file server è che la manipolazione dei dati avviene per intero sul PC client e non sul file server. Questo agisce semplicemente come un dispositivo di memorizzazione dei dati. Il software sul



**Figura 11.2**

Modello di file server.



file server accoda le richieste di accesso, ma sta al programma applicativo sul PC client, che lavora con la copia del DBMS, gestire tutte le funzioni di manipolazione dei dati. Per esempio, in questo ambiente i controlli sulla protezione dei dati e sul blocco dei record vengono avviati sui PC client.

### 11.1.2 Limiti dei file server

Quando si utilizzano i file server sulle reti locali, si devono affrontare tre limiti. Innanzitutto, si genera un notevole spostamento dei dati nella rete. Per esempio, quando un programma in esecuzione su un PC client di Pine Valley Furniture vuole accedere ai prodotti realizzati in quercia, l'intera tabella Product viene trasferita sul PC client e poi analizzata su di esso per trovare i pochi record desiderati. Il server fa quindi una piccola parte del lavoro, laddove il client è impegnato in una manipolazione intensa dei dati e la rete trasferisce grandi blocchi di dati. Di conseguenza, una LAN basata su client sposta sul PC client gran parte delle funzioni che devono essere eseguite da tutti i client, e genera un traffico di rete intenso.

In secondo luogo, ogni stazione di lavoro client deve riservare la memoria a una versione completa del DBMS. Questo significa che c'è meno spazio in memoria per i programmi applicativi sul PC client. L'espansione della RAM sul PC migliora le prestazioni aumentando la quantità totale dei dati che possono risiedere sul PC mentre una transazione viene elaborata. Inoltre, poiché le stazioni di lavoro client fanno la maggior parte del lavoro, ogni client deve essere sufficientemente potente da fornire tempi di risposta adeguati. Per contro, il file server non deve disporre necessariamente di molta RAM e non deve essere per forza un PC potente, poiché non fa granché del lavoro.

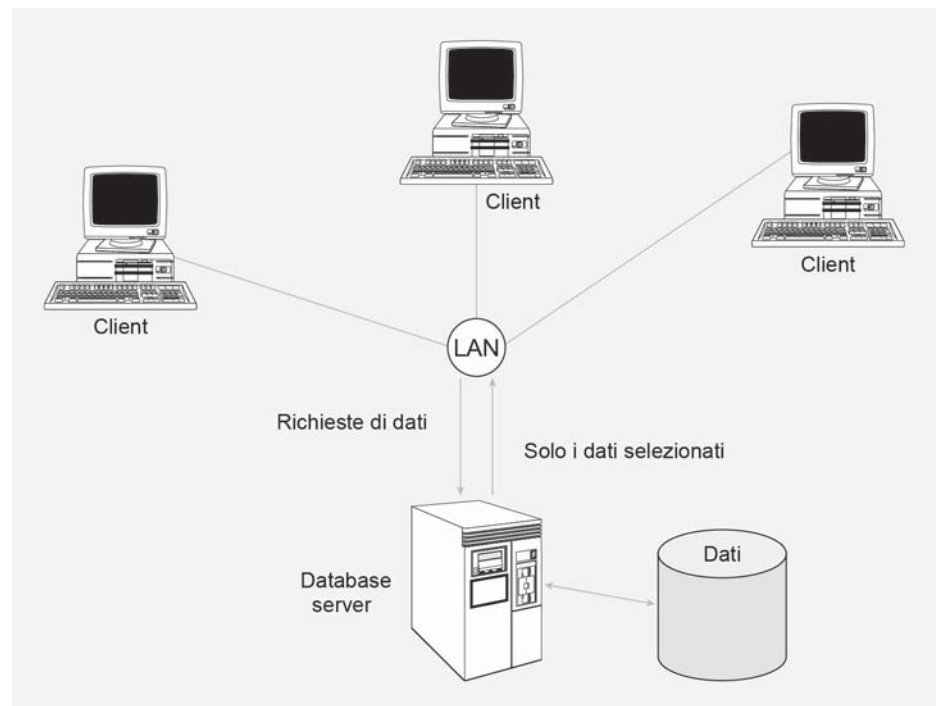
Il terzo limite, e probabilmente il più importante, è che la copia del DBMS su ciascuna stazione di lavoro deve gestire l'integrità della base di dati condivisa. Inoltre, ogni programma applicativo deve riconoscere, per esempio, i blocchi, e preoccuparsi di attivare quelli opportuni. I programmatori delle applicazioni devono quindi essere abbastanza preparati da capire le numerose e insidiose condizioni che possono sorgere in un ambiente di base di dati multiutente. Devono capire come le applicazioni interagiranno con i controlli di simultaneità, di ripristino e di protezione e a volte dovranno programmare tali controlli nelle proprie applicazioni.

### 11.1.3 Architetture di database server

Gli approcci a due livelli alle architetture client/server sono seguiti all'approccio del file server. In questo sistema la stazione di lavoro client è responsabile della gestione dell'interfaccia utente, compresa la logica di presentazione, quella di elaborazione dei dati e quella delle regole aziendali; il server della base di dati è responsabile dell'accesso alle memoria della base di dati e dell'elaborazione. La Figura 11.3 mostra una tipica architettura di server della base di dati. Con il DBMS collocato su questo server, il traffico della LAN è ridotto, perché alla stazione client vengono trasmessi solo i record che corrispondono ai criteri, e non i file di dati interi. Alcune persone parlano delle funzioni centrali del DBMS come di *funzioni di back-end*, mentre chiamano i programmi applicativi sui PC client *programmi front-end*.

Lo spostamento del DBMS sul server della base di dati comporta numerosi vantaggi. Con questa architettura, solo il server della base di dati necessita di una potenza di elaborazione adeguata a gestire la base di dati, e questa viene memorizzata sul server, e non sui client. Il server della base di dati può quindi essere impostato in modo da ottimizzare le prestazioni di elaborazione. Poiché i dati inviati sulla

**Figura 11.3**  
Architettura di database server  
(architettura a due livelli).



LAN sono di meno, il carico di comunicazione è ridotto. L'autorizzazione degli utenti, il controllo dell'integrità, la manutenzione dei dizionari dei dati e l'elaborazione delle interrogazioni e degli aggiornamenti sono tutti eseguiti in un unico punto, il server della base di dati.

I progetti client/server che utilizzano le architetture dei file server e di base di dati tendono a essere applicazioni di funzione, che supportano cioè un numero di utenti relativamente piccolo. Tali applicazioni non sono cruciali per la missione dell'azienda e hanno più successo là dove i volumi delle transazioni sono bassi, la disponibilità immediata non è fondamentale e la protezione dei dati non è la preoccupazione principale. Man mano che le aziende hanno cercato di ottenere i vantaggi previsti dai progetti client/server, come la scalabilità, la flessibilità e i costi ridotti, hanno dovuto sviluppare nuovi approcci alle architetture server/client.

L'uso delle **stored procedure**, moduli di codice che implementano la logica delle applicazioni, ha spinto l'architettura delle basi di dati verso la necessità di saper gestire applicazioni aziendali più cruciali (Quinlan, 1995). Come indicato da Quinlan, le stored procedure presentano i seguenti vantaggi.

- Migliorano le prestazioni per le istruzioni SQL compilate.
- Il traffico di rete diminuisce man mano che l'elaborazione si sposta dal client al server.
- La protezione migliora se si accede alla stored procedure invece che ai dati e se il codice viene spostato sul server, lontano dall'accesso diretto degli utenti.
- L'integrità dei dati migliora man mano che più applicazioni accedono alla stessa stored procedure.
- Le stored procedure permettono un client più sottile e un server della base di dati più spesso.

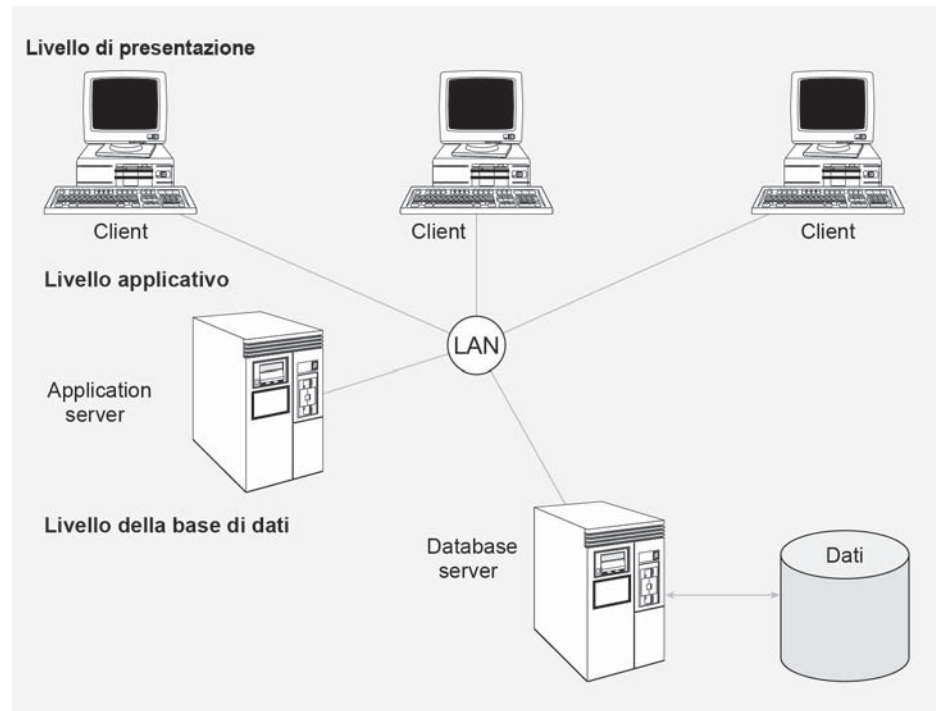
La scrittura di stored procedure occupa tuttavia più tempo che non l'uso di Visual Basic o Powerbuilder per creare un'applicazione. La natura proprietaria delle stored procedure riduce inoltre la loro portabilità e può rendere difficile cambiare DBMS senza doverle riscrivere. Inoltre, su ogni client devono essere caricate tutte le applicazioni che verranno utilizzate in quella posizione. Le prestazioni tendono a degradare con l'aumentare del numero degli utenti. Gli aggiornamenti a un'applicazione richiedono che ciascun client venga aggiornato separatamente. Questi svantaggi delle architetture di server della base di dati hanno portato alla popolarità dell'architettura a tre livelli.

## 11.2 Architetture a tre livelli

In generale, un'architettura a tre livelli include un livello di server supplementare nel client e alcuni dei livelli di server delle basi di dati precedentemente citati (Figura 11.4). Tali configurazioni sono anche dette architetture a *n*-livelli, *multilivello* o *client/server avanzate*. Il server supplementare in un'architettura a tre livelli può essere utilizzato per più scopi. I programmi applicativi risiedono spesso sul server supplementare, nel qual caso si parla di *application server*. Lo stesso tipo di server può anche contenere una base di dati locali, mentre un altro server contiene la base di dati aziendale. È probabile che ognuna di queste configurazioni venga definita "architettura a tre livelli", ma la funzionalità di ciascuna di esse cambia a seconda della situazione. I vantaggi di un'architettura a tre livelli rispetto a una a due livelli, come la scalabilità aumentata, la flessibilità, le prestazioni e la riusabilità, hanno reso le architetture a tre livelli la scelta più diffusa per le applicazioni Internet e i sistemi informativi basati sulla rete. I vantaggi vengono trattati nel dettaglio di seguito.

In alcune architetture a tre livelli, la maggior parte del codice delle applicazioni è memorizzato sull'application server. Questa impostazione fornisce gli stessi vantaggi del collocare le stored procedure su un server della base di dati in un'architettura a due livelli. L'uso di un application server può anche

**Figura 11.4**  
Architettura a tre livelli.



migliorare le prestazioni grazie a un codice macchina reale, a una maggiore portabilità del codice applicativo su altre piattaforme e alla minore dipendenza da linguaggi proprietari come SQL/PLUS (Quinlan, 1995). In molti casi, gran parte dell'elaborazione dei dati aziendali si verifica sull'application server piuttosto che sulla stazione di lavoro client o sul server della base di dati, cosa che permette di avere un client sottile. L'uso di browser Internet sui client per accedere al Web fornisce un esempio di architettura di client sottile. Le applicazioni che risiedono su un server e che lì vengono eseguite senza essere scaricate sul client stanno diventando sempre più diffuse. L'aggiornamento dei programmi applicativi richiede quindi il caricamento della nuova versione solo sull'application server invece che sulle stazioni di lavoro client.

Le architetture a tre livelli possono offrire numerosi vantaggi (Thompson, 1997).

- **Scalabilità.** Le architetture a tre livelli sono maggiormente scalabili di quelle a due livelli. Per esempio, il livello intermedio può essere utilizzato per ridurre il carico su un server della base di dati utilizzando un controllo *Transaction Processing* (TP) per diminuire il numero delle connessioni a un server, e alcuni application server supplementari possono essere aggiunti per distribuire l'elaborazione delle applicazioni. Un controllo TP è un programma che controlla il trasferimento dei dati tra client e server per fornire un ambiente coerente per l'elaborazione delle transazioni on-line (OLTP).
- **Flessibilità tecnologica.** Con un'architettura a tre livelli, è più semplice cambiare i motori del DBMS, sebbene i trigger e le stored procedure debbano essere riscritti. Il livello intermedio può anche essere spostato su una piattaforma diversa. I servizi di presentazione semplificati facilitano l'implementazione delle varie interfacce desiderate, come i browser Web o i kiosk.

- *Costi ridotti sul lungo termine.* L'uso di componenti o servizi standardizzati nel livello intermedio può ridurre i costi, così come il sostituire i moduli in un'applicazione invece dell'intera applicazione.
- *Migliore corrispondenza dei sistemi alle esigenze aziendali.* È possibile costruire nuovi moduli per supportare esigenze aziendali specifiche piuttosto che creare applicazioni complete più generiche.
- *Servizio per i clienti migliorato.* Interfacce diverse su client diversi possono accedere agli stessi processi aziendali.
- *Vantaggio di competitività.* L'abilità di reagire rapidamente ai cambiamenti aziendali modificando piccoli moduli di codice invece che applicazioni intere può essere utilizzata per ottenere un vantaggio di competitività.
- *Rischio ridotto.* Anche in questo caso, la capacità di implementare moduli di codice più piccoli e di combinarli con il codice acquistato dai vari produttori limita i rischi legati allo sviluppo di un progetto su vasta scala.

Le architetture a tre e a  $n$  livelli sono lo sviluppo più recente degli approcci client/server. Le sfide associate al passaggio ad ambienti a tre livelli o più complessi includono le seguenti (Thompson, 1997).

- *Costi più alti sul breve termine.* L'implementazione di un'architettura a tre livelli richiede che il componente della presentazione venga separato da quello dell'elaborazione. Il raggiungimento di questa separazione implica una maggiore programmazione con un linguaggio di terza generazione, come C, che non quando si implementa un'architettura a due livelli.
- *Strumenti e formazione.* Poiché le architetture a tre livelli sono relativamente recenti, gli strumenti per implementarle non sono ancora pienamente sviluppati. Analogamente, poiché i programmi di formazione non sono ancora molto diffusi, le aziende dovranno sviluppare le competenze di implementazione al proprio interno.
- *Esperienza.* Analogamente, sono poche le persone che hanno già un'esperienza nella creazione di sistemi a tre livelli.
- *Standard incompatibili.* Finora sono stati proposti pochi standard per i controlli TP. Non è ancora chiaro quale dei numerosi standard suggeriti per gli oggetti distribuiti prevarrà.
- *Mancanza di strumenti per gli utenti finali che funzionino con i servizi del livello intermedio.* Gli strumenti generici ampiamente utilizzati, come fogli di calcolo e strumenti di reporting, non operano ancora attraverso i servizi del livello intermedio. Questo problema verrà spiegato nel dettaglio successivamente nel capitolo, nel paragrafo sul middleware.

## 11.3 Partizionamento di un'applicazione

Come si può immaginare, non esiste un'architettura client/server ottimale che rappresenti una soluzione migliore per tutti i problemi aziendali. Per contro, la flessibilità intrinseca delle architetture client/server offre alle aziende la possibilità di adattare le loro configurazioni alle proprie esigenze di elaborazione. La Figura 11.1 ha mostrato la logica di calcolo che deve essere distribuita tra il client e i server. La logica della presentazione risiede sul client, dove l'utente interagisce con il sistema. La logica dell'elaborazione può essere suddivisa tra i client e i server, come indicato nella trattazione sui file server e le architetture client/server a due e tre livelli. La logica della memorizzazione risiede sul server della base di dati, vicino alla posizione fisica dei dati. Le attività di controllo dell'integrità dei dati, come la

verifica dei vincoli, avvengono solitamente in questo punto. I trigger, che si attivano sempre quando si verificano determinate condizioni, sono associati ai comandi di inserimento, modifica, aggiornamento ed eliminazione. Poiché questi comandi influiscono direttamente sui dati, i trigger vengono solitamente memorizzati anche sul server della base di dati, e lo stesso vale per le stored procedure che utilizzano direttamente i dati. Quelle che lavorano con il risultato di un'interrogazione possono essere memorizzate sul client. A seconda della natura del problema aziendale da affrontare, queste regole generali possono anche non essere seguite per poter ottenere delle prestazioni ottimali.

Il partizionamento delle applicazioni aiuta in questo tipo di adattamento. Offre agli sviluppatori l'opportunità di scrivere il codice delle applicazioni che verrà poi collocato sulla stazione di lavoro client o sul server, a seconda della posizione che offre le prestazioni migliori. Non è necessario includere il codice che collocherà il processo da suddividere né di scrivere il codice che stabilirà le connessioni al processo. Queste attività sono gestite dagli strumenti di partizionamento delle applicazioni.

Gli oggetti creati utilizzando la programmazione orientata agli oggetti sono particolarmente adatti per il partizionamento delle applicazioni. I programmatori possono avere un enorme controllo sul contenuto dell'oggetto ed è più facile separare il codice dell'interfaccia utente, le regole aziendali e i dati. Questa separazione supporta lo sviluppo odierno dei sistemi a  $n$  livelli. Il grosso volume d'affari che spinge verso soluzioni Internet e di commercio elettronico si sta sviluppando in modi più rapidi e nuovi. Le applicazioni Web devono essere a più livelli e suddivise; richiedono componenti che possano essere assemblati al volo, non appena vengono richiesti dal browser, e devono essere compatibili con sistemi operativi, interfacce utente e basi di dati diversi. Nell'ambiente Web, un partizionamento della applicazioni efficace è necessario per poter raggiungere le prestazioni desiderate con un livello di manutenzione, un'integrità dei dati e una sicurezza accettabili in un ambiente distribuito e imprevedibile.

Il codice dell'applicazione può essere sviluppato e testato su una stazione di lavoro client e le decisioni sul partizionamento e il posizionamento del codice possono essere prese in un secondo tempo. Questa funzionalità aumenterà probabilmente la produttività degli sviluppatori. I moduli delle applicazioni possono essere collocati su un client o un server nella fase di progettazione. Lo sviluppatore deve tuttavia capire come e dove ogni processo verrà eseguito per poter sincronizzare correttamente i processi e le transazioni tra le basi di dati e le piattaforme. Le decisioni sul collocamento del codice sull'application server o sul database server dipendono parzialmente dalle funzionalità del DBMS. Per esempio, un DBMS che supporta un codice SQL statico (codice SQL interamente pre-scritto) attraverso le stored procedure e i trigger sul server della base di dati può comportare un decremento delle prestazioni se il codice SQL dinamico (codice creato durante il runtime) è collocato sull'application server. Ogni istruzione di SQL dinamico genererà un legame dinamico (o collegamento agli oggetti della base di dati) al server della base di dati quando viene elaborato. L'impatto delle prestazioni dipende dall'intensità con cui le istruzioni di SQL dinamico vengono utilizzate. Se concentrarsi sull'elaborazione sull'application server o sul database server è una decisione che deve essere presa dallo sviluppatore, che conosce l'ambiente hardware disponibile, le interazioni tra l'hardware e il software del DBMS e le richieste dell'applicazione.

È inoltre possibile aggiungere controlli TP ai sistemi client/server per migliorare le prestazioni. Là dove siano disponibili application server e di base di dati multipli, i controlli TP possono bilanciare il carico di lavoro, indirizzando le transazioni sui server che non sono occupati. I controlli TP sono utili anche negli ambienti distribuiti, dove le transazioni distribuite da una singola unità di lavoro possono essere gestite attraverso un ambiente eterogeneo.

## 11.4 Ruolo dei mainframe

Nell'ultimo decennio il ruolo dei mainframe è diventato meno scontato, a fronte del notevole sviluppo delle funzionalità distribuite client/server e di calcolo. Come detto in precedenza, i sistemi fondamentali per la missione delle aziende, che dieci anni fa risiedevano sui sistemi mainframe, tendono ancora oggi a restare sugli stessi. Sono stati invece sviluppati sistemi meno cruciali, spesso a livello di gruppo di lavoro, utilizzando architetture client/server. La popolarità delle architetture client/server e il desiderio crescente delle aziende di essere più efficienti in ambienti sempre più distribuiti man mano che le prospettive diventavano più ampie e globali, hanno fatto ipotizzare che i sistemi fondamentali venissero spostati dai mainframe alle architetture server/client. Questo spostamento è stato una vera sfida. Un sistema di calcolo distribuito con successo dipende dalla distribuzione del software di lavoro, dalla gestione delle prestazioni e dall'adeguamento dei sistemi di produzione, dalla definizione delle procedure di risoluzione dei problemi e dalla gestione di un codice efficace (Hurwitz, 1996). Le aziende hanno scoperto che gestire queste applicazioni così importanti è molto più complicato se queste vengono convertite in sistemi distribuiti. Tra i problemi di distribuzione del software che si verificano in un ambiente di elaborazione distribuito vi sono i seguenti.

- Determinare quale codice deve essere collocato su quale stazione di lavoro e quale sarà disponibile attraverso il server.
- Identificare i conflitti potenziali con il codice di altre applicazioni.
- Garantire che siano disponibili risorse sufficienti in tutte le posizioni per gestire un carico in anticipo.

A meno che gli sviluppatori non anticipino i problemi di scalabilità risolvendoli man mano che sviluppano il codice, lo spostamento delle applicazioni fondamentali in ambienti client/server distribuiti può causare seri problemi nella transizione fra la fase pilota e la produzione vera e propria. Questo risvolto ha rallentato lo sviluppo di sistemi client/server fondamentali per la missione. L'affermarsi di Internet rende però sempre più necessario che le applicazioni operino in modo incrociato. La risoluzione di questi problemi è difficile, e molte aziende potrebbero trovarsi impreparate. L'uso sempre più frequente di librerie (file di moduli di applicazioni condivisi) aumenta la necessità di gestire le inter-dipendenze che si sviluppano inevitabilmente man mano che queste librerie vengono condivise tra le applicazioni.

Il supporto agli ambienti distribuiti è molto complesso. Comprendere le dinamiche delle posizioni dei diversi ambienti e creare una rete di calcolo distribuita è tutt'altro che semplice. La difficoltà di gestire un ambiente distribuito ha fatto sì che i manager IT riprendessero in considerazione l'idea di spostare i sistemi fondamentali per la missione su soluzioni client/server. Si sono così diffusi dispositivi potenti di elaborazione parallela (discussi nel prossimo paragrafo), che rendono più attraente l'ipotesi di mantenere un'impostazione centralizzata su un application server o di base di dati.

Le difficoltà sopra citate hanno scoraggiato molti manager IT, e alcuni si sono completamente ritirati da qualsiasi iniziativa client/server, tornando all'ambiente mainframe. È tuttavia molto probabile che ogni azienda si trovi, prima o poi, a dover cercare l'equilibrio tra piattaforme mainframe e client/server e tra soluzioni centralizzate e distribuite più adatto alla natura dei propri dati e alla posizione degli utenti dei dati dell'azienda. Come ha suggerito Hurwitz (1996), i dati che non devono essere spostati di frequente possono essere centralizzati su un mainframe. I dati a cui gli utenti accedono spesso, le immagini grafiche e l'interfaccia utente dovrebbero invece essere tenuti vicini sulle stazioni di lavoro client. I sistemi ERP (*Enterprise Resource Planning*) hanno affrontato molte sfide. Negli anni '90, i sistemi ERP come quelli di SAP, Baan, Oracle e J. D. Edwards hanno convinto molte aziende che lavoravano

con grandi quantità di dati che acquistare sistemi integrati di risorse finanziarie e umane, di produzione e di approvvigionamento sarebbe stato efficace e conveniente. Passare a una suite di applicazioni compatibili con l'anno 2000 era più attraente che non riscrivere milioni di righe di codice nelle applicazioni esistenti. Tuttavia, i sistemi ERP erano stati sviluppati come sistemi molto chiusi e controllati. Incorporare i sistemi legacy, dove risiedono i dati storici dell'azienda, spesso su un mainframe, è stato uno dei problemi principali. Ora, negli anni 2000, l'attenzione si rivolge all'ottenere una gestione efficiente della catena delle forniture, della gestione dei rapporti con i clienti (CRM) e dell'approvvigionamento *business-to-business* (B2B) attraverso soluzioni di commercio elettronico. I sistemi ERP stabiliscono un'infrastruttura integrata per le esigenze di gestione interne all'azienda, ma devono offrire anche funzionalità di commercio elettronico. I sistemi ERP classici sono stati modificati nell'architettura e sono stati implementati per competere in un ambiente ancora più distribuito e flessibile. Attualmente non è possibile prevedere quali saranno i risultati dei sistemi ERP.

## 11.5 Utilizzare architetture parallele

I tentativi delle aziende di spostare le applicazioni fondamentali su architetture client/server implicano la creazione di sistemi che possano gestire basi di dati più grandi dei progetti client/server originali. I progetti che implicano basi di dati molto grandi (VLDB) hanno beneficiato dell'uso di architetture di computer parallele. In effetti alcuni progetti sono possibili solo perché l'elaborazione parallela della base di dati fornisce tempi di risposta accettabilmente veloci. I data warehouse hanno utilizzato il parallelismo per ottenere un'estrazione, una trasformazione e un caricamento dei dati più veloci lavorando su più server non costosi in un ambiente multilivello.

La capacità di gestire grandi volumi di transazioni, interrogazioni complesse e nuovi tipi di dati si è dimostrata problematica in molti ambienti a processore unico (DeWitt e Gray, 1992). I RDBMS e il linguaggio SQL si prestano tuttavia a creare autonomamente un ambiente parallelo in due modi (Ferguson, 1994).

1. Nella maggior parte delle interrogazioni, SQL agisce come un linguaggio di elaborazione non procedurale. Le interrogazioni possono quindi essere divise in parti, ognuna delle quali può essere eseguita contemporaneamente su una processore parallelo diverso.
2. Le interrogazioni multiple possono essere eseguite in parallelo su processori paralleli (Figura 11.5).

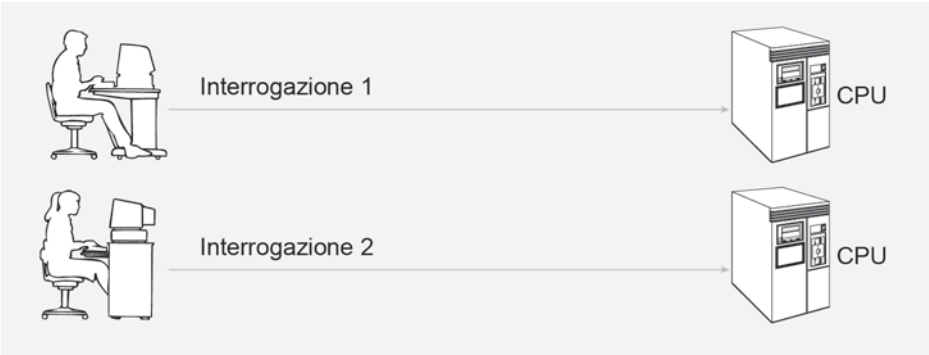
L'uso efficace di architetture di computer parallele dipende anche dalla consapevolezza delle possibilità di elaborazione che possono essere sfruttate in un ambiente di server parallelo, e non dalla semplice modifica degli algoritmi seriali in algoritmi paralleli. Di seguito introdurremo due delle numerose architetture hardware multiprocessore disponibili; ognuna di esse è ideale per un tipo specifico di elaborazione.

### 11.5.1 Architetture hardware multiprocessore

I sistemi multiprocessore strettamente accoppiati hanno una memoria condivisa tra tutti i processori (Figura 11.6). Questa architettura è spesso chiamata SMP (*Symmetric Multiprocessing*) e presenta alcuni vantaggi (Ferguson, 1994). Innanzitutto, una copia singola del sistema operativo risiede nella memoria condivisa e viene quindi condivisa tra tutti i processori. Il sistema operativo deve essere riscritto per operare in un tale ambiente così da poter eseguire parti di sé stesso in parallelo su qualsiasi altro processore senza sviluppare una tendenza a utilizzare un processore più di altri.



**Figura 11.5**  
Transazioni e interrogazioni  
parallele (adattato da Ferguson,  
1994).  
(a) Transazioni parallele.



(b) Interrogazione parallela.

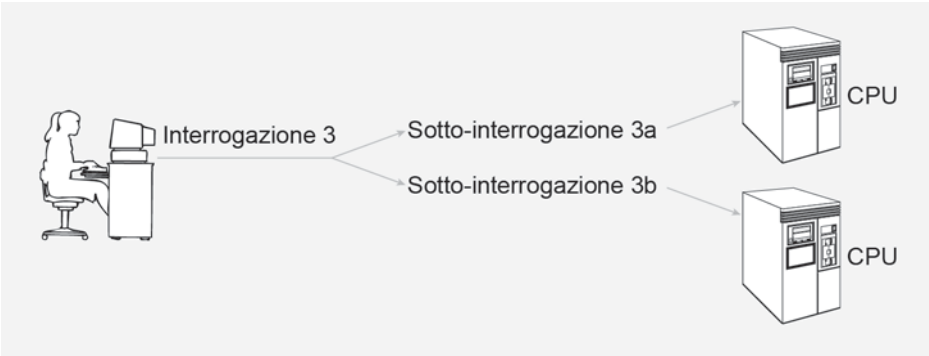
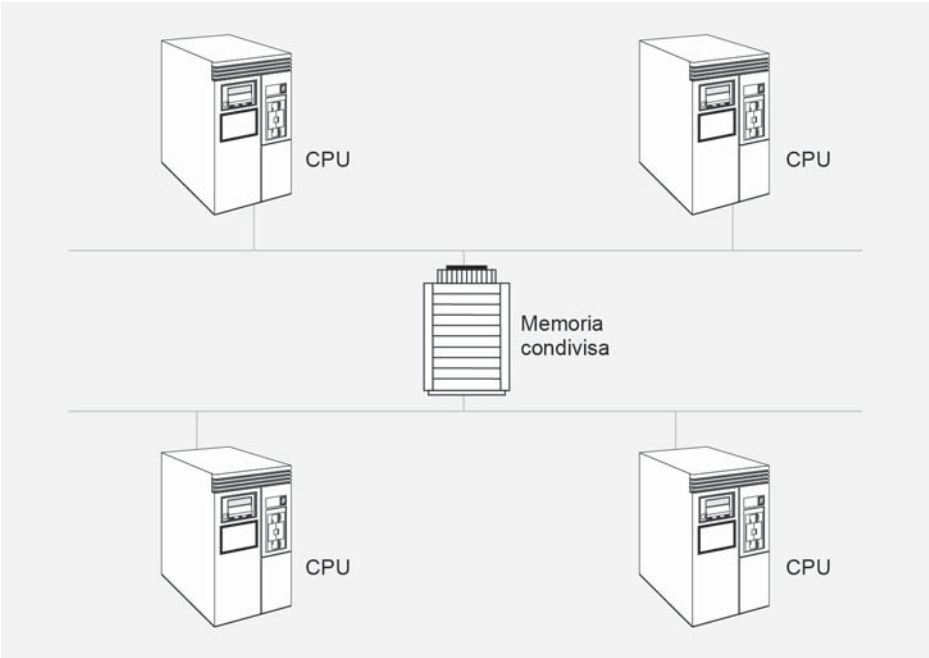


**Figura 11.6**  
Architettura a multiprocessori  
strettamente accoppiati (adattato  
da Ferguson, 1994).



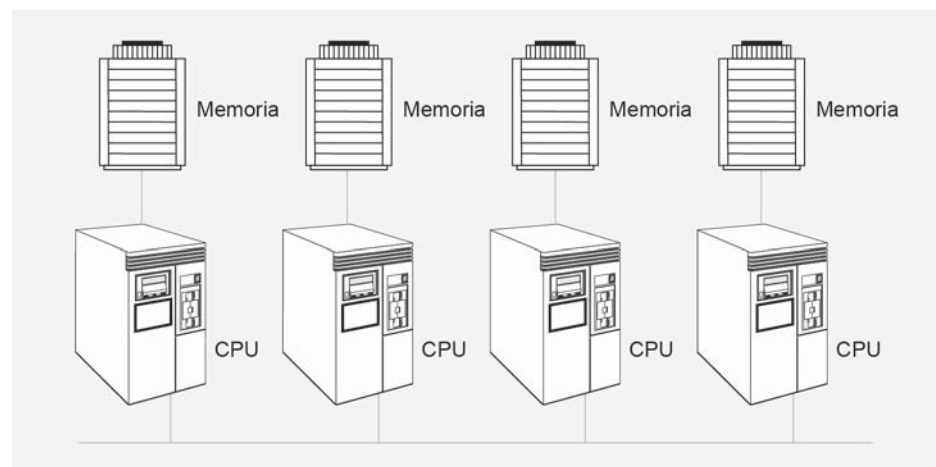
Un secondo vantaggio è che i colli di bottiglia vengono ridotti rispetto ai sistemi a un processore, perché tutti i processori nella multi-elaborazione simmetrica condividono tutte le operazioni. Anche i sistemi SMP possono sviluppare colli di bottiglia di I/O quando il traffico di richiesta della memoria diretto verso la memoria condivisa è molto intenso. Dotare ciascun processore di una memoria cache propria riduce il problema. È quindi possibile limitare il volume delle richieste della memoria condivisa riattivando le transazioni in attesa di essere eseguite sul processore nel momento in cui è iniziata la loro elaborazione. Questo permette che la transazione venga riattivata più velocemente, con meno richiesta di memoria, e che sia possibile utilizzare le informazioni variabili relative alla transazione che sono ancora disponibili nella cache del processore specifico (Ferguson, 1994).

In terzo luogo, le architetture SMP sono utili nelle situazioni in cui i dati devono rimanere in memoria durante l'elaborazione per raggiungere il livello di prestazioni desiderato. I problemi da risolvere includono la possibilità di contenimento della memoria condivisa, l'ampiezza di banda richiesta per poter recuperare i dati nella memoria condivisa e la collocazione di tutta la memoria in un unico punto di fallimento. I problemi di contenimento dell'I/O sono più probabili quando i processori vengono aggiunti in un ambiente di memoria condivisa.

A differenza delle architetture SMP, le architetture non strettamente accoppiate (Figura 11.7), chiamate anche **MPP (Massively Parallel Processing)** o **architetture di non condivisione**, impostano ogni CPU con una memoria dedicata. I sistemi MPP richiedono quindi che una copia completa del sistema operativo risieda in ogni memoria dedicata. Poiché ci sono poche risorse da condividere tra i processori, i problemi di contenimento della memoria che si verificano nei sistemi SMP sono pressoché inesistenti, ed è possibile aggiungere nodi (un altro processore e della memoria dedicata) in singole unità. Questa aggiornabilità lineare consente alle aziende di raggiungere l'elaborazione parallela a costi inferiori, aggiungendo capacità di elaborazione in piccoli incrementi in base alle esigenze. La scalabilità crea un sistema più gestibile nell'ambiente aziendale dinamico, dove i data warehouse crescono rapidamente e in modo esponenziale. Passando i messaggi tra i processori per raggiungere il parallelismo, questi sistemi possono scomporre le operazioni più complesse e ripartirle tra più processori. Sono infatti le applicazioni che eseguono le operazioni più articolate a trarre beneficio dalla suddivisione e dal lavoro simultaneo e che sono più adatte alle architetture MPP piuttosto che le applicazioni che si avvantaggiano dell'uso della memoria condivisa. Lo sviluppo della tecnologia MPP è piuttosto impegnativo e IBM, NCR, Intel, Oracle, DEC (una divisione di Compaq), HP, Encore, Silicon Graphics e altri sono attualmente coinvolti

**Figura 11.7**

Architetture a multiprocessori non strettamente accoppiati (adattato da Ferguson, 1994).



nella ricerca e sviluppo di queste soluzioni. Poiché i sistemi di elaborazione parallela sono diversi da quelli tradizionali, è necessario che le aziende che prendono in considerazione questi sistemi rivalutino la progettazione delle proprie basi di dati e delle proprie applicazioni, la pianificazione della capacità, la gestione delle prestazioni e le procedure operative e di programmazione.

### 11.5.2 Usi aziendali delle SMP e MPP

Le aziende hanno grandi depositi di dati operazionali storici da sfruttare, soprattutto quando si tratta di informazioni importanti su molti aspetti della propria attività, come i clienti, i prodotti, le scorte e le vendite. Fino a poco tempo fa, le aziende non avevano una potenza di calcolo sufficiente per riuscire ad approfittare di questi dati memorizzati (Rudin, 1995). Ora, invece, l'elaborazione parallela sta consentendo alle aziende di iniziare ad analizzare grandi quantità di dati storici attraverso lo sviluppo di data warehouse e data mart. L'urgenza delle funzionalità di elaborazione parallela sfocia in particolare nello sviluppo dei data warehouse, che vengono discussi nel dettaglio nel Capitolo 11. Le aziende di vendita al dettaglio, che devono analizzare grandi quantità di dati, hanno adottato i sistemi di elaborazione parallela.

Le aziende che intendono passare ai sistemi SMP o MPP devono considerare con attenzione quanto riportato di seguito (Rudin, 1995).

- *Esigenze di analisi dei dati.* I sistemi paralleli sono più efficaci quando si tratta di risolvere problemi che implicano grandi quantità di dati (250 GB o volumi superiori), di eseguire interrogazioni complesse o di elaborare dati che sono utilizzati da un gran numero di utenti simultanei. Tali problemi dovrebbero essere affrontati e risolti solo se hanno un'importanza strategica per l'azienda, altrimenti i costi significativi dell'hardware, dello sviluppo del software e della nuova formazione non valgono il gioco. Per esempio, le applicazioni per i rivenditori al dettaglio forniscono risultati di grande valore strategico. Un esempio è il sistema di analisti del paniere di mercato di Catalina Marketing, che tiene traccia non solo dei prodotti che vengono acquistati, ma anche degli assortimenti di prodotti che tendenzialmente vengono acquistati nello stesso momento. I dettaglianti possono utilizzare i risultati di queste applicazioni per pianificare promozioni coordinate di prodotti correlati, per riservare dello spazio sugli scaffali e per prendere decisioni sui prezzi.
- *Giustificazione dei costi.* Si ricordi che i sistemi paralleli tendono a essere complementari piuttosto che sostitutivi dei sistemi operazionali esistenti. Le aziende li impiegano più per aumentare i ricavi che non per ridurre i costi. Le procedure tradizionali di giustificazione dei costi possono quindi non essere appropriate. Poiché molte delle applicazioni sono concepite per migliorare le strategie di marketing di un'azienda, è una buona idea considerare il sistema parallelo come un investimento di marketing.
- *Tecnologie tradizionali.* Se l'applicazione proposta può essere realizzata utilizzando la tecnologia tradizionale esistente nell'azienda, può avere senso continuare a basarsi su tale tecnologia. I sistemi MPP sono tutt'altro che maturi e la loro implementazione comporta sempre una sfida per qualsiasi azienda. Anche quando si è certi che l'elaborazione parallela abbia senso per il problema che si vuole affrontare, è consigliabile intraprendere un'analisi di fattibilità delle tecnologie più mature. Implementare una soluzione per un problema che potrebbe trarre vantaggi dall'elaborazione parallela in un ambiente a monoprocesso può essere fattibile, ma prima di procedere con soluzioni di questo tipo è preferibile escludere le tecnologie tradizionali.

Le applicazioni di data warehouse e data mining sono riservate all'elaborazione delle interrogazioni che accedono a grandi quantità di dati e metadati mediante operazioni di unione e di aggregazione.

Queste interrogazioni implicano numerose operazioni di I/O e generano spesso insiemi di risultati intermedi ridimensionabili (Frazer, 1998). Attualmente non esistono basi di dati relazionali/a oggetti (O/R) abbastanza grandi da qualificarsi come sistemi VLDB (per ulteriori informazioni sulle basi O/R si veda l'Appendice A). Con l'evoluzione della tecnologia O/R, è tuttavia probabile che ci si troverà a gestire basi di dati che abbiano almeno la dimensione delle implementazioni relazionali correnti, in parte perché le basi di dati O/R gestiscono nuovi tipi di dati che sono più voluminosi da memorizzare. Le basi di dati O/R avranno a che fare con associazioni complesse tra istanze di nuovi tipi di dati, associazioni che vanno oltre la semplice struttura a righe e colonne del modello relazionale. I metodi di elaborazione paralleli supporteranno probabilmente questi nuovi sviluppi.

## 11.6 Utilizzare il middleware

Il middleware è spesso definito come la colla che tiene insieme le applicazioni client/server. Il termine viene utilizzato per descrivere qualsiasi componente software tra il PC client e la base di dati relazionale nelle architetture a  $n$  livelli. In parole semplici, il middleware è una qualsiasi delle numerose classi di software che permettono che un'applicazione inter-operi con un altro software senza che sia necessario che l'utente capisca e codifichi le operazioni a livello più basso richieste per raggiungere l'interoperabilità (Hurwitz, 1998). Il middleware esiste da decenni; alcuni esempi sono il middleware di elaborazione delle transazioni di BM, CICS o Tuxedo di BEA Systems per macchine UNIX. L'avvento delle tecnologie client/server e lo sviluppo orientato al Web ha favorito lo sviluppo di un middleware disponibile a livello commerciale. Un ipotetico "middleware universale", un magico pacchetto software che possa integrare e connettere ogni tipo di sistema, sarebbe ovviamente la soluzione ideale. Purtroppo un pacchetto middleware di questo tipo non esiste ancora. La maggior parte delle aziende utilizza diversi pacchetti middleware, a volte anche all'interno di una stessa applicazione.

Un'altra considerazione riguarda la tipologia della comunicazione implicata, se sincrona o asincrona. Con i sistemi sincroni, il sistema in richiesta attende la risposta in tempo reale. Un sistema bancario on-line nel quale l'utente controlla un estratto conto prima di emettere un assegno è un esempio di sistema sincrono. I sistemi asincroni inviano una richiesta ma non attendono la risposta in tempo reale, che invece viene accettata quando viene ricevuta. La posta elettronica è un esempio di un sistema asincrono con cui si ha probabilmente molta familiarità.

Hurwitz (1998) ha fornito un utile sistema di classificazione strutturato in sei categorie che organizzano i numerosi tipi di middleware attualmente disponibili. Le sue classificazioni si basano sulla scalabilità e sul livello di ripristino.

- *RPC (Remote Procedure Call) asincrono*. Il client richiede servizi ma non attende una risposta. Solitamente stabilisce una connessione point-to-point con il server ed esegue altre procedure mentre attende la risposta. Se la connessione cade, il client deve ristabilirla e inviare nuovamente la richiesta. Questo tipo di middleware ha una grande scalabilità ma una bassa ripristinabilità, ed è stato ampiamente sostituito da un RPC sincrono fin dal 1998.
- *RPC sincrono*. Un programma distribuito che utilizza un RPC sincrono può chiamare i servizi disponibili su computer diversi. Questo middleware rende possibile stabilire tale servizio senza avventurarsi nella codifica dettagliata solitamente necessaria per scrivere un RPC. Esempi includono Microsoft Transaction Server e CICS di IBM. L'equivalente Java di un RPC è un *Remote Method Invocation* (RMI).
- *Pubblica/iscrivi*. Questo tipo middleware analizza l'attività e indirizza le informazioni agli iscritti. È un sistema sincrono; i client, o iscritti, eseguono altre attività tra le notifiche provenienti dal server.

Gli iscritti comunicano al pubblicante le informazioni che vogliono ricevere, e quando si verifica un evento che contiene tali informazioni, queste vengono inviate agli iscritti, che possono quindi decidere se riceverle o meno. Per esempio, è possibile fornire alle librerie elettroniche le parole chiave degli argomenti a cui si è interessati. Ogni volta che la libreria aggiunge un titolo la cui parola chiave corrisponde a quella fornita, le informazioni su quel titolo verranno inoltrate automaticamente per essere valutate. Questo tipo di middleware è molto utile per controllare quelle situazioni in cui le azioni devono essere intraprese quando si verifica un determinato evento.

- *Middleware orientato agli oggetti (MOM).* Anche MOM è un software asincrono, che invia i messaggi che vengono raccolti e memorizzati fino a che non vengono richiamati, mentre il client continua con altre operazioni. Le applicazioni di flusso di lavoro come quelle delle assicurazioni, che implicano spesso numerosi passaggi di elaborazione, possono trarre vantaggio da un MOM. È possibile creare un registro della coda in cui memorizzare le richieste, così da fornire un minimo di possibilità di ripristino.
- *Broker di richiesta dell'oggetto (ORB).* Questo tipo di middleware rende possibile alle applicazioni l'invio di oggetti e servizi di richiesta in un sistema orientato agli oggetti. L'ORB tiene traccia della posizione di tutti gli oggetti e indirizza le richieste a ognuno di essi. Gli ORB correnti sono sincroni, ma sono in via di sviluppo anche ORB asincroni.
- *Accesso ai dati orientato a SQL.* La connessione delle applicazioni a basi di dati in rete si ottiene utilizzando un middleware di accesso ai dati orientato a SQL. Questo middleware ha anche la capacità di tradurre il codice SQL generico in quello specifico della base di dati. I produttori e le società di basi di dati che hanno sviluppato il middleware di accesso a basi di dati multiple dominano questo segmento di middleware.

L'elenco precedente fornisce una descrizione elementare delle varianti funzionali del middleware, ma sono disponibili anche numerosi prodotti ibridi che combinano tutte queste funzionalità. Per esempio, i middleware orientati a SQL e RPC aggiungono la consapevolezza che gli oggetti e gli ORB stanno aggiungendo servizi di transazioni, di coda e di messaggio. Controlli di transazione degli oggetti saranno disponibili presto e promettono di unire la funzionalità degli oggetti distribuiti, attualmente gestite da CORBA o DCOM, ai controlli delle transazioni, che forniscono la gestione delle transazioni per le applicazioni, aiutando a ridurre i colli di bottiglia.

Nei sistemi client/server, il middleware orientato alle basi di dati fornisce una sorta di interfaccia del programma applicativo, o **API (Application Program Interface)**, per l'accesso alla base di dati. Le API sono insiemi di routine utilizzati da un programma applicativo per dirigere le prestazioni delle procedure eseguite dal sistema operativo del computer. Per esempio, nell'ottenere l'accesso a una base di dati, un'API chiama le routine delle librerie che inviano in modo trasparente i comandi SQL dall'applicazione front-end al server della base di dati. Un'API può lavorare con un software front-end esistente, come un linguaggio di terza generazione o un generatore di report personalizzati, e può includere utility proprie per costruire le applicazioni. Quando ci sono API per più strumenti di sviluppo di programmi, allora si avrà una notevole indipendenza nello sviluppare applicazioni client nell'ambiente di programmazione front-end più conveniente e nel trarre i dati da una base di dati comune sul server. Questo tipo di middleware rende possibile agli sviluppatori collegare facilmente un'applicazione alle basi di dati più popolari.

*Open Database Connectivity (ODBC)* è simile ad API, ma riguarda solo le applicazioni client/server basate su Windows. Questo tipo di connettività è utile soprattutto per accedere ai dati relazionali, mentre non è adatta per accedere ad altri tipi di dati, come i file ISAM (LaRue, 1997). Sebbene ODBC sia difficile da programmare e implementare, è stata ben accettata perché permette ai programmatori di

creare connessioni con le basi di dati di quasi tutti i produttori, senza dover imparare il codice proprietario specifico. OLE-DB di Microsoft aggiunge valore allo standard ODBC fornendo un unico punto di accesso a più basi di dati (Linthicum, 1997). Microsoft sta progettando di creare uno standard OLE-DB universale di accesso ai dati e ha aggiunto OLE-DB per le applicazioni di data mining e OLE-DB per OLAP. L'accesso ai dati legacy mentre si passa ai sistemi client/server può essere ottenuto grazie a prodotti come EDA/SQL, che tentano di supportare più sistemi operativi, reti e basi di dati diversi.

Le classi di *Java Database Connectivity* (JDBC) possono essere utilizzate per aiutare un'applet ad accedere a un numero qualsiasi di basi di dati senza che sia necessario capirne le funzioni native. JDBC definisce un'interfaccia a livello di chiamata (CLI) per lo sviluppo Java e prende in prestito alcune convenzioni di ODBC. Stabilire un linguaggio comune per definire le interfacce tra i componenti e un meccanismo di mediazione faciliterà lo sviluppo di un middleware universale (Keuffel, 1997). L'*Object Management Group* (OMG), stabilito nel 1989, è un cartello che ha prodotto la *Common Object Request Broker Architecture* (CORBA), che imposta la specifica del middleware universale orientato agli oggetti. Microsoft ha sviluppato un modello competitivo, *Distributed Component Object Model* (DCOM), ma CORBA è una specifica più solida, perché è stata sviluppata per gestire più piattaforme diverse. L'interoperabilità tra i due standard sta lentamente emergendo. Tali standard sono particolarmente importanti sul World Wide Web proprio per la diversità delle piattaforme che connettono.

## 11.7 La protezione dell'ambiente client/server

Un ambiente di base di dati client/server implica l'esistenza di una rete che colleghi tra loro i componenti client/server. In questo ambiente distribuito, stabilire la protezione della base di dati è più complesso che non in un ambiente centralizzato. Le reti sono soggette a brecce nella protezione dovute a intercettazioni, connessioni non autorizzate o recuperi non autorizzati di pacchetti di informazioni che viaggiano attraverso la rete. Le architetture client/server sono quindi più sensibili alle minacce alla sicurezza che non i sistemi centralizzati.

Le misure di protezione che dovrebbero essere intraprese in un ambiente client/server includono quelle comuni a tutti i sistemi, ma dovrebbero prevedere anche quelle specifiche (Bobrowski, 1994).

- *Protezione con password a livello di sistema.* I nomi utente e le password sono solitamente utilizzati per identificare e autorizzare gli utenti che vogliono connettersi a un sistema client/server multiutente. Gli standard di sicurezza dovrebbero includere linee guida per le lunghezze delle password, le convenzioni per la loro denominazione, la frequenza con cui cambiarle e così via. Le utility di gestione delle password dovrebbero essere incluse come parte della rete e dei sistemi operativi.
- *Protezione con password a livello di base di dati.* La maggior parte dei DBMS client/server hanno una protezione con password a livello di base di dati che è simile a quella a livello di sistema. È inoltre possibile adottare con facilità le informazioni di autenticazione tratte dalla funzione di autenticazione del sistema operativo, ma in questo modo anche i tentativi esterni di ottenere un accesso risultano più facili, perché l'uso di questa funzionalità riduce da due a uno il numero dei livelli di protezione mediante password. L'autenticazione che si basa su quella del sistema operativo non è quindi consigliabile.
- *Comunicazione client/server protetta.* La codifica, ovvero la trasformazione dei dati leggibili (testo normale) in non leggibili (testo cifrato) può aiutare a proteggere la comunicazione client/server. La maggior parte dei client invia le password in testo normale degli utenti della base di dati ai database server. I RDBS più grandi, come Oracle, Sybase e Inform, hanno funzionalità di trasmissione sicura

delle password in rete. Ovviamente sarebbe auspicabile la codifica di tutti i dati che passano in rete, ma i costi dei software di codifica sono molto alti. Inoltre, la codifica influisce negativamente sulle prestazioni a causa del tempo necessario per criptare e decrittare i dati.

### 11.7.1 Problemi di sicurezza client/server per basi di dati abilitate per il Web

Il fiorire di siti Web che rendono accessibili i dati correnti agli utenti mediante le connessioni Internet solleva nuovi problemi che vanno oltre le tradizionali questioni di sicurezza di client e server affrontati nel paragrafo precedente. La creazione dinamica di una pagina Web da una base di dati richiede un accesso alla base di dati; se questa non adeguatamente protetta, risulta vulnerabile ad accessi impropri da parte di qualsiasi utente. Questo è un nuovo limite che in precedenza veniva evitato grazie a schemi di autorizzazione di accesso o a un software specializzato di accesso al client.

Se un'azienda desidera solo rendere disponibile le pagine HTML statiche, dovrà stabilire una protezione per i file HTML memorizzati su un server Web. La creazione di una pagina Web statica si basa sull'uso di strumenti client/server tradizionali come Visual Basic o PowerBuilder; in questo modo la creazione delle pagine può essere controllata utilizzando metodi standard di controllo di accesso alla base di dati. Se alcuni dei file HTML caricati sul server Web sono sensibili, possono essere collocati in directory che vengono protette utilizzando la protezione del sistema operativo, oppure possono essere letti ma non pubblicati nella directory. In questo modo l'utente deve conoscere il nome esatto del file per accedere alla pagina HTML sensibile. È anche prassi comune isolare il server Web e limitarne il contenuto a pagine Web che possano essere sfogliate pubblicamente mediante un browser. I file sensibili possono essere conservati su un altro server accessibile attraverso un'intranet dell'azienda. Anche in questo caso saranno necessarie le misure di protezione sopra descritte se non si vuole permettere ad alcuni utenti di accedere a tutti i file memorizzati nell'intranet. La protezione basata sull'autenticazione dell'utente può essere utilizzata per controllare l'accesso ai file sensibili.

Le misure di protezione per la generazione di pagine Web dinamiche sono diverse. Le pagine Web dinamiche sono memorizzate come un modello nel quale i dati opportuni e correnti vengono inseriti dalla base di dati una volta che le interrogazioni associate alla pagina vengono eseguite. Questo significa che il server Web deve essere in grado di accedere alla base di dati. Per poter funzionare correttamente, la connessione necessita solitamente dell'accesso completo alla base di dati. Stabilire un'adeguata protezione del server è quindi cruciale per tenere i dati al sicuro. Il server che possiede la connessione alla base di dati dovrebbe essere fisicamente sicuro, e l'esecuzione dei programmi e degli script CGI (*Common Gateway Interface*) dovrebbe essere monitorata.

L'accesso ai dati può essere controllato attraverso un altro livello di protezione, quello utente. L'uso di una forma di login HTML consente all'amministratore di definire i privilegi concessi a ciascun utente. Ogni sessione può essere tracciata memorizzando un elemento di dati, detto *cookie*, sulla macchina client. Questo dato può essere restituito sul server e fornisce informazioni sulla sessione di login. Per poter eseguire le routine di autenticazione necessarie, occorre stabilire gli script CGI, o altri strumenti.

La protezione della sessioni deve essere stabilita per garantire che i dati privati non vengano compromessi durante una sessione, poiché le informazioni vengono trasmesse attraverso una rete per essere ricevute su una determinata macchina, e sono potenzialmente intercettabili. TCP/IP non è un protocollo molto sicuro, e i sistemi di codifica come quelli trattati nel paragrafo precedente sulla protezione dei client/server sono fondamentali. Molti sistemi Internet utilizzano una codifica che combina una chiave pubblica e una segreta. Sia il server sia il client possono codificare una chiave segreta con le proprie chiavi private. I dati codificati vengono trasmessi con la loro chiave pubblica; la loro chiave pubblica può essere eventualmente conservata anche dal server di certificati che la fornisce. *Secure Sockets Layer*

(SSL) è un metodo di codifica standard utilizzato da numerosi sviluppatori per codificare tutti i dati che viaggiano tra i client e il server durante una sessione. Gli URL che iniziano con il prefisso `http://` utilizzano SSL per la trasmissione. Esistono infine altre misure di protezione, come le firme digitali, i server Kerberos e alcuni strumenti specifici di alcuni produttori.

## 11.8 Problemi di client e server

La nascita delle architetture client/server ha senza dubbio influenzato l'ambiente di calcolo delle basi di dati. Ciononostante, si sente ancora parlare di fallimenti nell'implementazione di client e server e di utenti delusi. Per poter riuscire, i progetti client/server devono risolvere un problema aziendale specifico utilizzando una tecnologia ben definita e considerando i parametri di costo. Per migliorare le opportunità di creazione di applicazioni client/server di successo (Linthicum, 1996), si devono considerare i seguenti aspetti.

- *Analisi accurata del problema aziendale.* Come accade per tutte le architetture di computer, è cruciale sviluppare un progetto di applicazione e di architettura solidi per qualsiasi sistema client/server nuovo. La tendenza degli sviluppatori a scegliere la tecnologia e poi adeguarvi l'applicazione emerge dalla spinta verso gli ambienti client/server che si è verificata nell'ultimo decennio. È invece più appropriato definire accuratamente l'ambito del problema, determinare le esigenze e poi utilizzare queste informazioni per scegliere la tecnologia.
- *Analisi dettagliata dell'architettura.* È importante specificare i dettagli dell'architettura client/server. La creazione di una soluzione di questo tipo implica l'interconnessione di più componenti, cosa che non sempre funziona facilmente. Uno dei vantaggi spesso sollecitati degli ambienti client/server, ossia la capacità di accettare un approccio ai sistemi aperti, può comportare molti problemi se i componenti scelti sono troppo eterogenei e quindi difficili da connettere. Oltre a specificare le stazioni di lavoro client, i server, le reti e i DBMS, gli analisti dovrebbero anche specificare l'infrastruttura della rete, il livello intermedio e gli strumenti di sviluppo da utilizzare per l'applicazione. A ogni passaggio, gli analisti dovrebbero agire in modo tale da garantire che gli strumenti si connetteranno con il middleware, la base di dati, la rete e così via.
- *Evitare architetture guidate dagli strumenti.* Come in precedenza, è preferibile determinare le esigenze del sistema prima di scegliere gli strumenti e non il contrario. Scegliere prima uno strumento e poi applicarlo al problema non garantisce una corrispondenza adeguata tra strumento e problema. Scegliere gli strumenti in questo modo è come sceglierli in base a un criterio emotivo piuttosto che in base alla loro funzionalità.
- *Ottenere una scalabilità adeguata.* Una soluzione multilivello consente di scalare i sistemi client/server a un numero qualsiasi di utenti di gestire carichi di elaborazione diversificati. Le soluzioni multilivello sono però molto costose e difficile da creare; anche gli strumenti per sviluppare un ambiente multilivello sono molto limitati. Gli architetti dovrebbero evitare di passare a una soluzione di questo tipo se non è davvero necessario. Un'impostazione multilivello ha senso negli ambienti con più di 100 utenti simultanei, nei sistemi che elaborano grandi volumi di transazioni o nei casi di elaborazioni in tempo reale. Gli ambienti con un carico di lavoro minore possono essere eseguiti più efficientemente sui sistemi tradizionali a due livelli, soprattutto se l'elaborazione è gestita da trigger e stored procedure.
- *Collocazione appropriata dei servizi.* Anche in questo caso, un'analisi attenta del problema aziendale da risolvere è importante quando si prendono delle decisioni sulla posizione dei servizi di



elaborazione. Il passaggio a client sottili e server spessi non è sempre la soluzione ideale. Spostare la logica delle applicazioni su un server, creando così un server spesso, può influenzarne la capacità, poiché gli utenti finali tenderanno tutti di utilizzare l'applicazione ora collocata sul server. A volte è possibile ottenere una scalabilità migliore spostando l'elaborazione delle applicazioni sul client. I server spessi tendono a ridurre il carico in rete perché l'elaborazione avviene vicino ai dati, e i server spessi riducono la necessità di client potenti. Comprendere in profondità il problema aziendale dovrebbe aiutare l'architetto a distribuire la logica nel modo opportuno.

- *Analisi di rete.* Il collo di bottiglia più comune nei sistemi distribuiti è ancora la rete. È quindi importante che gli architetti non ignorino le capacità dell'ampiezza di banda della rete che il sistema deve utilizzare. Se la rete è non sufficiente per gestire l'entità delle informazioni che devono passare tra client e server, i tempi di risposta ne risentiranno, e il sistema probabilmente fallirà.
- *Consapevolezza dei costi nascosti.* I problemi di implementazione di client/server vanno oltre i problemi di analisi, sviluppo e architettura elencati in precedenza (Atre, 1995). Per esempio, i sistemi che sono concepiti per utilizzare hardware, reti, sistemi operativi e DBMS esistenti sono spesso ostacolati dalle complessità dell'integrazione di questi componenti eterogenei nella costruzione del sistema client/server. Quella della formazione è una spesa significativa e ricorrente che viene spesso sottovalutata. La difficoltà di lavorare in un ambiente multi-produttore può rivelarsi molto costosa.

Se questi problemi vengono affrontati nel modo opportuno, si possono ottenere numerosi vantaggi dal passaggio alle architetture client/server (Atre, 1995).

- La funzionalità può essere consegnata agli utenti finali in più tappe. In questo modo si può ottenere più rapidamente man mano che i primi segmenti del progetto vengono distribuiti.
- Interfacce grafiche utente comuni negli ambienti client/server incoraggiano gli utenti a utilizzare la funzionalità delle applicazioni.
- La flessibilità e la scalabilità delle soluzioni client/server facilitano la nuova progettazione dell'elaborazione aziendale.
- Una maggiore elaborazione può essere eseguita vicino all'origine dei dati da elaborare, migliorando così i tempi di risposta e riducendo il traffico di rete.
- Le architetture client/server consentono lo sviluppo di applicazioni abilitate per il Web, favorendo una comunicazione efficace all'interno dell'azienda e la conduzione di affari con l'esterno su Internet.

## 11.9 Accedere alla base di dati dalle applicazioni client

La Figura 11.1 riportata in precedenza rappresenta i componenti della logica di programma che devono essere distribuiti in un ambiente di applicazione client/server a due o tre livelli. Il partizionamento di un ambiente per creare un'architettura a due, tre o  $n$  livelli comporta che debbano essere prese delle decisioni sul posizionamento della logica di elaborazione. In ogni caso, la logica della memorizzazione (il motore della base di dati), è gestita dal server, mentre quella della presentazione è gestita dal client.

La Figura 11.8a illustra alcuni sistemi a due livelli possibili, collocando la logica dell'elaborazione sul client (creando un client spesso), sul server (creando un client sottile) o suddividendola tra il server e il client (ambiente distribuito). Nei tre scenari si evidenzia la posizione della logica. Nel client spesso,

l'elaborazione delle applicazioni avviene per intero sul client, mentre nel client sottile l'elaborazione avviene sul server. Nell'esempio distribuito, l'elaborazione delle applicazioni è suddivisa tra il client e il server.

La Figura 11.8b presenta una tipica architettura a tre livelli e una a  $n$  livelli. Anche in questo caso, parte della logica dell'elaborazione può essere collocata sul client, se si desidera. Tuttavia, un client tipico in un ambiente client/server abilitato per il Web è un client sottile e utilizza un browser per la sua logica delle applicazioni. I livelli intermedi sono solitamente codificati in linguaggio portatile come C o Java. La flessibilità e la migliore gestibilità degli approcci a  $n$  livelli spiegano la popolarità di questa architettura, nonostante la notevole complessità della gestione della comunicazione tra i livelli. Anche l'ambiente veloce, distribuito ed eterogeneo delle iniziative Internet e di commercio elettronico hanno portato allo sviluppo di numerose architetture a  $n$  livelli.

Il client è quindi responsabile della logica della presentazione. Gli utenti comunicano con il resto del sistema attraverso i loro browser, un'applicazione o un'interfaccia di programmazione. Queste interfacce sono confluite in gran parte nelle *interfacce grafiche utente* (GUI). Le interfacce per accedere alle basi di dati non sono diverse. Sebbene la sintassi SQL venga comunque utilizzata per comunicare direttamente con le basi di dati, anche l'approccio *Query-by-Example* (QBE) per interrogare le basi di dati si è dimostrato molto popolare. La facilità d'uso e l'inclusione di funzionalità di programmazione nei prodotti di base di dati desktop come Microsoft Access hanno portato molte aziende a utilizzare pacchetti front-end associati a basi di dati di back-end più potenti, come Oracle o Informix. Per esempio, utilizzando le interrogazioni pass-through, formulate in Access ma eseguite su una base di dati Oracle di back-end, le aziende possono raggiungere tempi di sviluppo più veloci e utilizzare RDBMS più solidi e potenti per le loro applicazioni cruciali.

## 11.10 Utilizzare Query-by-Example

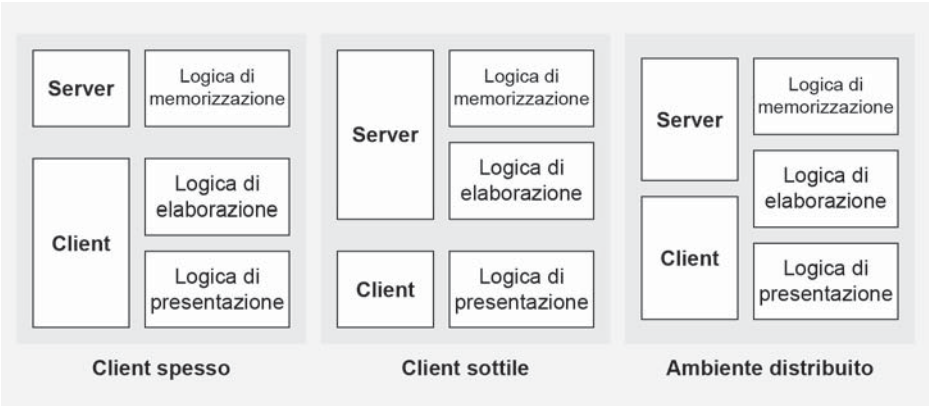
*Query-by-Example (QBE)* è il linguaggio più diffuso di interrogazione di base di dati che si fonda sulla manipolazione diretta dei basi. Sebbene non sia uno standard internazionale come SQL, QBE è stato utilizzato per molti anni, soprattutto nel mercato dei RDBMS per PC. La sua natura di linguaggio di interrogazione basato sulla manipolazione diretta fa sì che gli utenti possano imparare facilmente come interrogare una base di dati. La sua semplicità lo rende inoltre un linguaggio ideale per sviluppare i prototipi. Poiché alcuni sistemi di basi di dati come Microsoft Access traducono le interrogazioni QBE in SQL, QBE può essere utilizzato come primo passo nella creazione di un codice SQL. I sistemi basati su QBE che generano il codice SQL possono poi essere utilizzati per creare moduli di presentazione o di livello client che accedono alle basi di dati sul server. In questo paragrafo verrà illustrata la storia di QBE e si analizzeranno molte delle sue funzionalità di interrogazione.

### 11.10.1 Storia e importanza di QBE

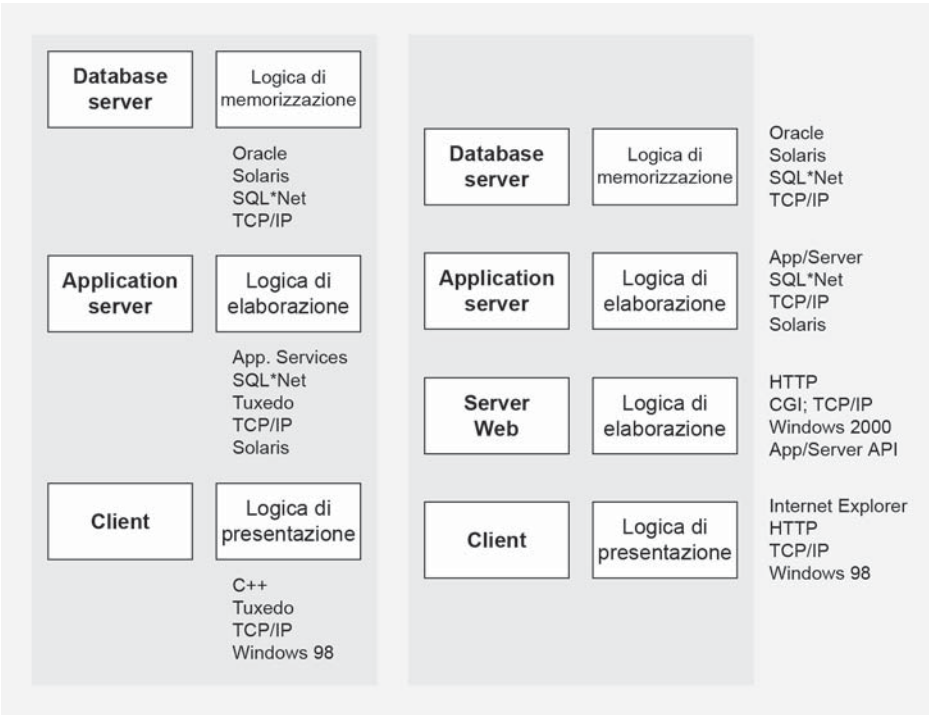
Sebbene QBE (come SQL) sia stato originariamente sviluppato per l'elaborazione delle basi di dati su mainframe, è diventato fondamentale anche nei sistemi di personal computer e client/server. Query-by-Example fu sviluppato per la prima volta da Zloof (1977) e fu reso disponibile per l'uso con SQL/DS e i sistemi di base di dati DB2 su mainframe IBM. Il successo del primo DBMS relazionale per personal computer (PC-RDBMS) completamente basato su QBE, Paradox, incoraggiò altri produttori ad adottare l'interfaccia QBE come opzione. La maggior parte dei sistemi attuali include una variante di QBE.

La trattazione di QBE è essenziale per capire i sistemi di basi di dati moderni, tanto più che QBE è considerato un linguaggio altamente produttivo con il quale programmare. Versioni visuali dei linguaggi

**Figura 11.8**  
Distribuzioni di logica più diffuse.  
(a) Ambienti client/server  
a due livelli.



(b) Ambienti client/server a *n* livelli  
(le possibilità sono numerose;  
questi sono solo alcuni esempi).



di programmazione, come Visual Basic, Visual C e Visual Java hanno trasformato il modo in cui i programmi vengono scritti. Studi e ricerche (per esempio, Greenblatt e Waxman, 1978; Thomas e Gould, 1975) hanno dimostrato che anche con una formazione minima, gli studenti trovano QBE più facile da utilizzare che non SQL o un linguaggio algebrico relazionale. Sebbene questi studi abbiano ormai più di vent'anni, non è stato ancora sviluppato un linguaggio di interrogazione che batta l'usabilità di QBE.

QBE è particolarmente utile per la programmazione delle basi di dati degli utenti finali. L'ambiente di programmazione visuale offre all'utente che non programma una vista singola dei dati a prescindere dall'operazione di base di dati eseguita. Come si vedrà successivamente (e come spiegato nel Capitolo 4), le interrogazioni vengono sviluppate interattivamente su uno schermo CRT in un formato che asso-

miglia all'output desiderato. Nella maggior parte dei programmi, sia le interrogazioni sia i risultati sono visualizzati nello stesso formato, solitamente quello di un foglio elettronico.

Con QBE si possono scrivere applicazioni di basi di dati complete, ma è più comune utilizzare QBE per interrogare o aggiornare interattivamente una base di dati. QBE è quindi particolarmente utile per un'elaborazione ad hoc. Invece di sviluppare un'applicazione completa in QBE, è più frequente utilizzarlo per creare il prototipo di un'applicazione, salvando le interrogazioni sviluppate. Le interrogazioni salvate possono essere poi migliorate utilizzando gli strumenti di generazione di applicazioni come schermate, maschere e report, e aggiungendo moduli di codice utilizzando il linguaggio di base di dati associato al PC-RDBMS per fornire comportamenti e impostazioni personalizzati.

## 11.10.2 QBE: i fondamenti

Per QBE non esiste uno standard ufficiale come per SQL (si veda il Capitolo 10 per una trattazione degli standard di SQL). Per questa ragione non esiste un insieme minimo di funzionalità che un'interfaccia di interrogazione deve soddisfare per poter essere considerata un'implementazione QBE. Tuttavia, poiché QBE si è evoluto e poiché l'interfaccia visiva di Windows è stata standardizzata, e dato che si tratta comunque di un linguaggio visuale, tutti i produttori hanno adottato approcci simili rispetto a QBE.

Sia il recupero sia la modifica dei dati possono essere eseguiti attraverso l'immissione di parole chiave, costanti e dati d'esempio nelle celle di un layout di tabella. Poiché le definizioni dei dati sono memorizzate in tabelle interne, anche la definizione dei dati viene eseguita attraverso un'interfaccia con un layout simile. In Microsoft Access 2000 (utilizzato con Oracle 8i per gli esempi di questo capitolo), facendo clic su un pulsante SQL si apre il codice SQL di Access che è stato generato quando l'interrogazione basata su QBE è stata costruita.

La Figura 11.9 illustra la gerarchia di usabilità di Microsoft Access. La piramide dimostra che Access è utilizzabile a tutti i livelli di abilità di programmazione e complessità. Al livello più basso della piramide ci sono gli oggetti, che permettono la creazione di tabelle, interrogazioni, forme e report senza avere conoscenze di programmazione specifiche. L'uso delle espressioni, o delle funzioni, per eseguire operazioni semplici come la moltiplicazione dei campi, la convalida dei dati o l'applicazione di una regola aziendale specifica, avviene nel livello successivo. A questo livello, quello delle macro, gli utenti traggono vantaggio dai moduli memorizzati di codice VBA (*Visual Basic for Applications*) per automatizzare le proprie applicazioni, anche in questo caso senza che occorra una conoscenza specifica di VBA. Nel livello successivo, gli utenti possono programmare i propri moduli di codice VBA per personalizzare le loro applicazioni. Al livello superiore, Windows API chiama le funzioni o i DLL scritti in linguaggi come C, Java o Visual Basic che possono essere utilizzati per scrivere le interfacce per altri programmi o fonti di dati.

QBE fornisce un metodo semplice e visuale per specificare le interrogazioni qualificate. I dati da visualizzare devono essere limitati ad alcune colonne e record con i valori desiderati, come mostrato nel Capitolo 10 a proposito dell'uso di SQL. In realtà, il codice SQL di Access che visualizza un'interrogazione in un riquadro di QBE può sempre essere visualizzato facendo clic sull'icona SQL.

In Access 2000, il riquadro QBE (Figura 11.10) contiene uno spazio di lavoro superiore in cui sono collocate le rappresentazioni del modello di dati delle tabelle o interrogazioni implicate in un'interrogazione. Anche le relazioni precedentemente stabilite vengono visualizzate automaticamente. Può accadere che occorra un'altra relazione per una singola interrogazione, interrogazione che può essere stabilita anche nel riquadro QBE. La parte inferiore del riquadro, chiamata riquadro di progettazione dell'interrogazione, o griglia, visualizza una maschera con il layout di un foglio elettronico nella quale sono collocati i campi necessari per l'interrogazione, assieme agli eventuali criteri di ordinamento e vincolo. Ogni colonna nella griglia QBE contiene informazioni su un singolo campo di una tabella o

**Figura 11.9**  
La gerarchia di usabilità di Access  
(adattato da Prague e Irwin, 1997).

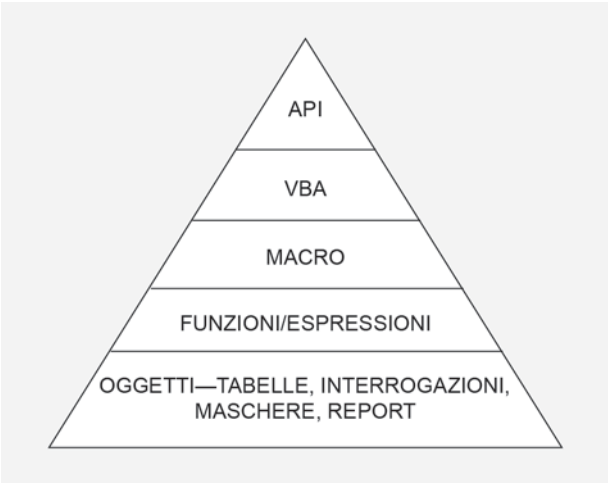
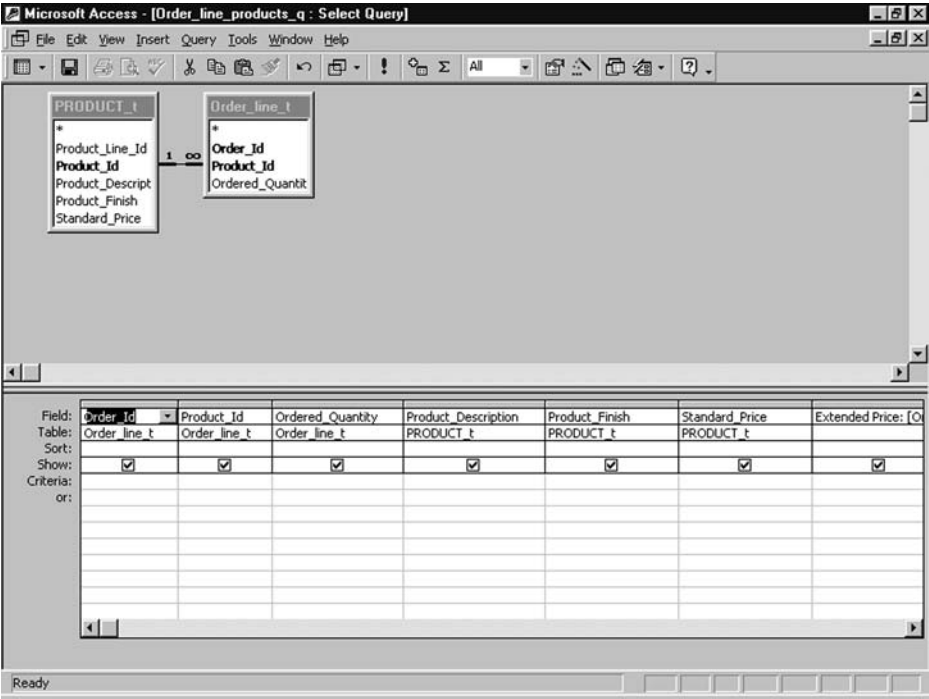


**Figura 11.10**  
Interrogazione su tabella multipla  
nella modalità di progettazione di  
Microsoft Access 2000 (Pine Valley  
Furniture).



interrogazione nel riquadro. La Figura 11.10 mostra un riquadro QBE per un'interrogazione selezionata che riguarda le tabelle dei prodotti e degli ordini e che chiede informazioni sugli ordini per ogni prodotto che è stato ordinato. Il risultato visualizzato nella Figura 11.11 è chiamato *dynaset*. Sebbene il risultato assomigli a una tabella, questa non è una tabella di base come quella delle materie grezze su cui è fondata. È invece un insieme di record *dinamici* o *virtuali*, non memorizzato nella base di dati. La Figura 10.12 mostra la vista SQL di Access di questa interrogazione.

**Figura 11.11**  
Dynaset di Microsoft Access 2000;  
è il risultato dell'interrogazione  
mostrata nella Figura 11.10 (Pine  
Valley Furniture).

| Order Id | Product Id | Ordered Quantity | Product Description  | Product Finish | Standard Price | Extended Price |
|----------|------------|------------------|----------------------|----------------|----------------|----------------|
| 1001     | 1          | 1                | End Table            | Cherry         | \$175.00       | \$175.00       |
| 1007     | 1          | 3                | End Table            | Cherry         | \$175.00       | \$525.00       |
| 1001     | 2          | 2                | Coffee Table         | Natural Ash    | \$200.00       | \$400.00       |
| 1007     | 2          | 2                | Coffee Table         | Natural Ash    | \$200.00       | \$400.00       |
| 1002     | 3          | 5                | Computer Desk        | Natural Ash    | \$375.00       | \$1,875.00     |
| 1003     | 3          | 3                | Computer Desk        | Natural Ash    | \$375.00       | \$1,125.00     |
| 1008     | 3          | 3                | Computer Desk        | Natural Ash    | \$375.00       | \$1,125.00     |
| 1001     | 4          | 1                | Entertainment Center | Natural Maple  | \$650.00       | \$650.00       |
| 1005     | 4          | 4                | Entertainment Center | Natural Maple  | \$650.00       | \$2,600.00     |
| 1006     | 4          | 1                | Entertainment Center | Natural Maple  | \$650.00       | \$650.00       |
| 1009     | 4          | 2                | Entertainment Center | Natural Maple  | \$650.00       | \$1,300.00     |
| 1006     | 5          | 2                | Writer's Desk        | Cherry         | \$325.00       | \$650.00       |
| 1004     | 6          | 2                | 8-Drawer Dresser     | White Ash      | \$750.00       | \$1,500.00     |
| 1009     | 7          | 3                | Dining Table         | Natural Ash    | \$800.00       | \$2,400.00     |
| 1004     | 8          | 2                | Computer Desk        | Walnut         | \$250.00       | \$500.00       |
| 1008     | 8          | 3                | Computer Desk        | Walnut         | \$250.00       | \$750.00       |
| 1010     | 8          | 10               | Computer Desk        | Walnut         | \$250.00       | \$2,500.00     |

**Figura 11.12**  
Interrogazione su tabella multipla  
nella vista SQL di Microsoft Access  
2000 (Pine Valley Furniture).

```
SELECT Order_line_t.Order_Id, Order_line_t.Product_Id, Order_line_t.Ordered_Quantity, PRODUCT_t.Product_Description, PRODUCT_t.Product_Finish, PRODUCT_t.Standard_Price, [Order_line_t].[Ordered_Quantity]*[PRODUCT_t].[Standard_Price] AS [Extended Price]
FROM PRODUCT_t INNER JOIN Order_line_t
ON PRODUCT_t.Product_Id = Order_line_t.Product_Id;
```

I vantaggi di un dynaset sono che il dispositivo di memorizzazione utilizzato richiede meno spazio di memoria e che ogni volta che l'interrogazione viene eseguita, utilizza la copia corrente della base di dati, che include i record che sono stati eventualmente aggiunti o aggiornati dall'ultima esecuzione. Le interrogazioni si comportano quindi come le viste delle relazioni trattate nel Capitolo 10. Se occorre salvare i risultati di un'interrogazione in una nuova tabella, si utilizzi un'interrogazione che crea una tabella.

Un veloce ripasso delle interrogazioni su più tabelle in SQL (Capitolo 8) permette di ricordare la necessità di specificare quali tabelle sono richieste nell'interrogazione (nella clausola FROM) e di impostare le uguaglianze tra ogni tabella necessaria per l'interrogazione (nella clausola WHERE). Se non si è molto veloci a utilizzare la tastiera, la codifica di questi collegamenti in SQL può portare via molto tempo. In QBE di Microsoft Access 2000, questi collegamenti vengono stabiliti quando le relazioni sono rese esplicite nella finestra Relazioni; l'inserimento delle tabelle necessarie nel riquadro QBE attiva automaticamente questi collegamenti, eliminando così la necessità di digitarli uno per uno. Ciò permette di risparmiare tempo e garantisce accuratezza. Si osservi la Figura 11.10, nella quale due delle tabelle da utilizzare nell'interrogazione sono state aggiunte al riquadro QBE e i campi necessari per il

risultato sono stati selezionati e trascinati sulla griglia QBE. La Figura 11.12 mostra la parte di codice SQL che è già stata definita con il clic e il trascinamento. Si noti che, sebbene la griglia QBE non contenga la colonna di chiave primaria della tabella Product\_t, ovvero Product\_ID, che è stata utilizzata per stabilire i collegamenti tra due tabelle, i collegamenti corretti sono stati resi espliciti nel codice SQL. Si osservi anche come il codice SQL di Access sia diverso da quello di Oracle SQL\*Plus utilizzato nel Capitolo 10 (qualora si volesse scrivere l'equivalente Oracle SQL del codice nella Figura 11.12). Le differenze evidenziano che ogni produttore ha i propri gusti in fatto di SQL.

### 11.10.3    Selezionare record qualificati

Cosa accade se si è interessati solo a determinati record? La Figura 11.13 mostra l'approccio di base per immettere le qualifiche per i record che devono essere visualizzati da una tabella. Quello che si deve fare è collocare delle condizioni sotto la colonna associata. Il dynaset restituito è mostrato nella Figura 11.14. La condizione può essere di diversità, come nella Figura 11.13, o può implicare un intervallo o un'uguaglianza. Per esempio, si potrebbe immettere "Between 350 and 700" sotto la colonna Standard\_Price nella Figura 11.13 per sapere quali prodotti hanno un prezzo medio.

Quando si impostano i criteri nelle interrogazioni, si possono utilizzare anche operatori, funzioni ed espressioni. Gli operatori matematici, come la moltiplicazione e la divisione, lavorano sui campi numerici. Gli operatori relazionali, come uguale a, diverso da o minore di, possono essere utilizzati con campi numerici, campi di data e campi di testo. Gli operatori booleani o logici sono utilizzati per impostare le condizioni nelle espressioni.

**Figura 11.13**  
Interrogazione per selezionare  
gli ordini effettuati per tre o più unità  
di un determinato prodotto  
(Pine Valley Furniture).

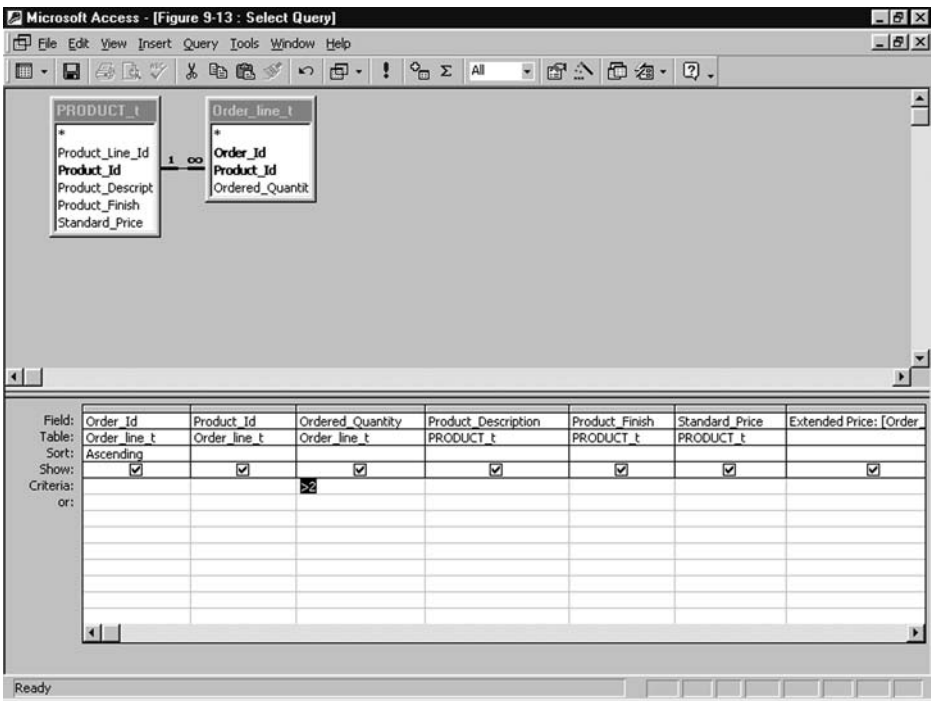


**Figura 11.14**

Dynaset derivato dall'interrogazione nella Figura 11.13 (Pine Valley Furniture).

| Order Id | Product Id | Ordered Quan | Product Description  | Product Finish | Standard Price | Extended Price |
|----------|------------|--------------|----------------------|----------------|----------------|----------------|
| 1002     | 3          | 5            | Computer Desk        | Natural Ash    | \$375.00       | \$1,875.00     |
| 1003     | 3          | 3            | Computer Desk        | Natural Ash    | \$375.00       | \$1,125.00     |
| 1005     | 4          | 4            | Entertainment Center | Natural Maple  | \$650.00       | \$2,600.00     |
| 1007     | 1          | 3            | End Table            | Cherry         | \$175.00       | \$525.00       |
| 1008     | 8          | 3            | Computer Desk        | Walnut         | \$250.00       | \$750.00       |
| 1008     | 3          | 3            | Computer Desk        | Natural Ash    | \$375.00       | \$1,125.00     |
| 1009     | 7          | 3            | Dining Table         | Natural Ash    | \$800.00       | \$2,400.00     |
| 1010     | 8          | 10           | Computer Desk        | Walnut         | \$250.00       | \$2,500.00     |

Le funzioni sono moduli di codice che restituiscono sempre un valore basato su un calcolo o un confronto eseguito dalla funzione. Possono venire restituiti valori di stringa, logici o numerici, a seconda della natura della funzioni. Microsoft Access fornisce numerose funzioni già pronte per l'uso, come quelle di somma o media, o permette di scrivere funzioni particolari in VBA. Le espressioni sono un termine o una serie di termini controllati da operatori. Access fornisce un aiuto nella sintassi delle espressioni, inserendo i caratteri necessari, come le virgolette, qualora vengano omessi.

### 11.10.4 Self-join

Alcune interrogazioni sono più facili da costruire utilizzando un'interfaccia QBE, e questo perché la sua interfaccia visuale è più chiara della sintassi. Un esempio è il caso in cui un'interrogazione richieda una tabella da unire a sé stessa. Questo tipo di interrogazione è chiamata *self-join* o *join ricorsivo*. Per esempio, si supponga di voler sapere quali ordini che includono uno qualsiasi degli stessi prodotti presenti sull'ordine numero 1004 sono stati effettuati. Interrogazioni di questo tipo possono sorgere, per esempio, a causa di un problema di imballaggio dell'ordine, e si vuole sapere se ci sono altri ordini che presentano lo stesso problema. Un'interrogazione QBE di Access 2000 che risponde a questa domanda è mostrata nella Figura 11.15a. Un self-join si crea collocando due copie della tabella Order\_Line\_t nel riquadro QBE, impostando una relazione tra gli ID di prodotto (si veda la linea che unisce i campi Product\_ID nelle due tabelle) e impostando un criterio 1004 in una tabella e uno NOT 1004 nell'altra che restituiranno l'ID di prodotto e i valori di quantità. Nella Figura 11.15b si può vedere che solo il numero 8 è stato ordinato su altri ordini. Ce ne sono tre nel numero d'ordine 1008 e dieci nell'ordine numero 1010. Il codice SQL (che dovrebbe essere eseguito su un server locale o di base di dati) è mostrato nella Figura 11.15c.

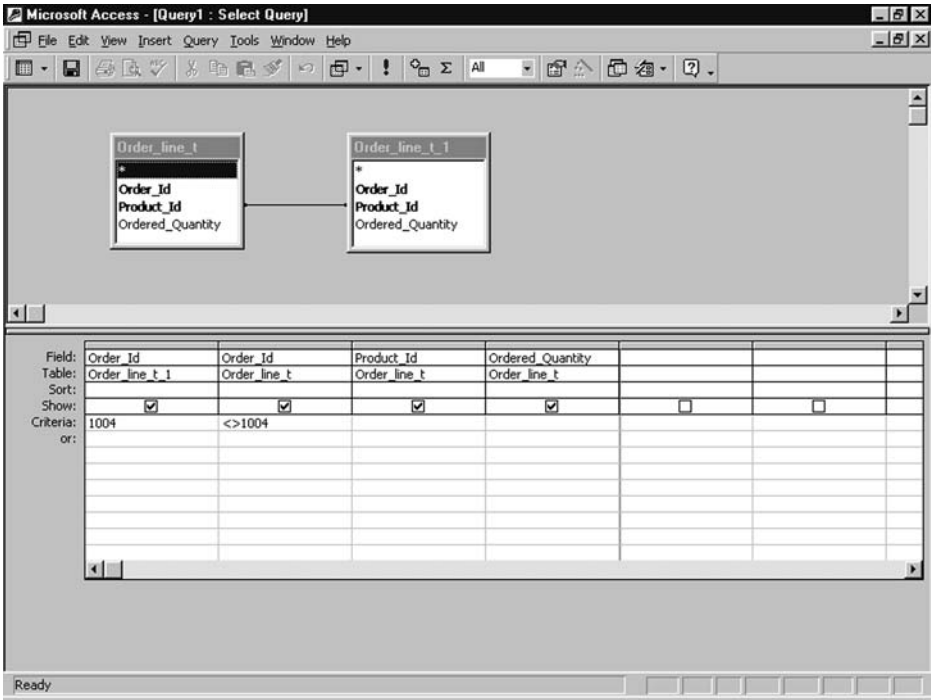
### 11.10.5 Basare un'interrogazione su un'altra interrogazione

Anche in QBE può essere difficile rispondere a una domanda con un'unica interrogazione. Un modo per affrontare queste interrogazioni così difficili è quello di suddividerle in più passaggi, salvare l'interrogazione generata per rispondere a quel passaggio e poi basare l'interrogazione successiva sull'interrogazione salvata invece che su quella di partenza. Tale approccio è parzialmente simile a quando si utilizzano le interrogazioni in SQL. Basare un'interrogazione su un'altra permette di calcolare i valori

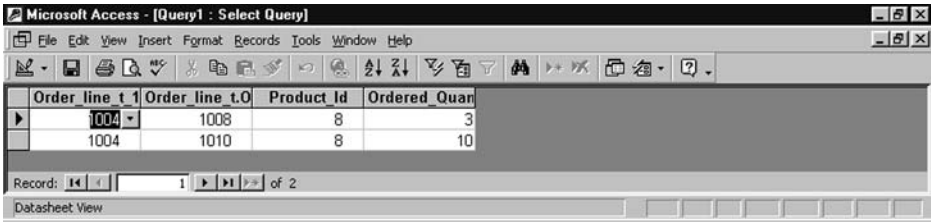


**Figura 11.15**

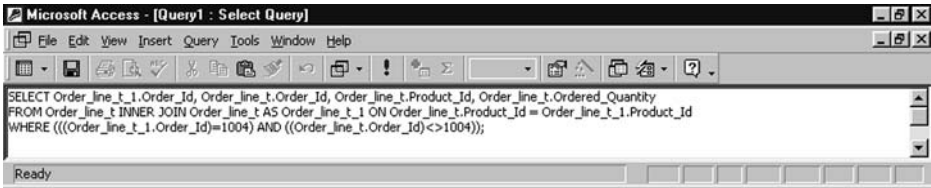
Join ricorsivo d'esempio: quali ordini contengono gli stessi prodotti dell'ordine 1004?  
(a) Join ricorsivo di Microsoft Access 2000 da Pine Valley Furniture.



(b) Dynaset di Microsoft Access 2000 dal join ricorsivo in (a).



(c) Interrogazione di join ricorsivo di Microsoft Access 2000 nella vista SQL (Pine Valley Furniture).



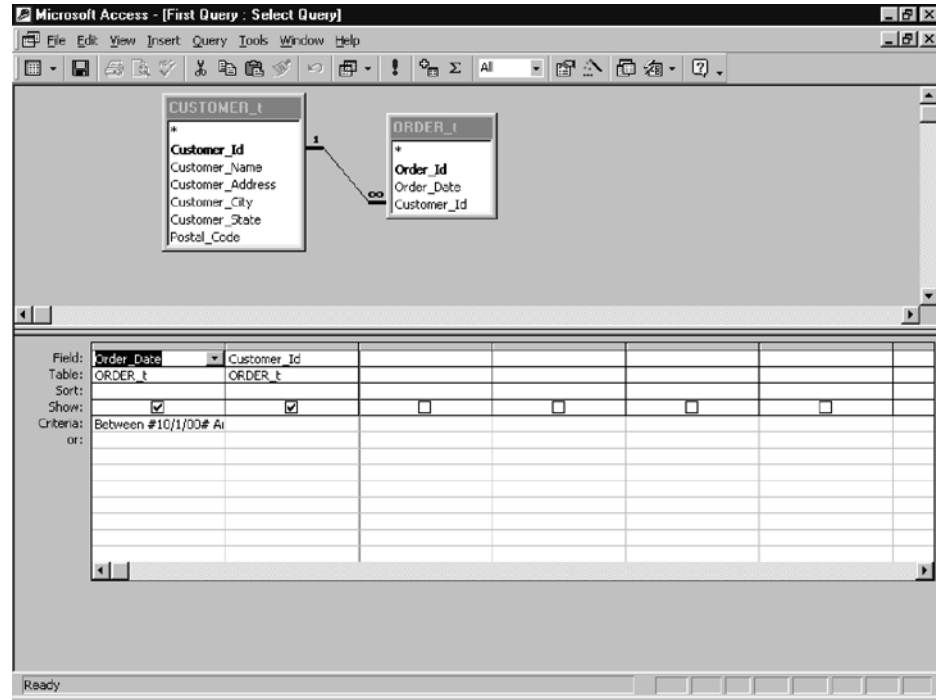
per SUM, COUNT e così via utilizzando il dynaset generato dall'interrogazione salvata, risolvendo così alcune delle difficoltà riguardanti l'uso di funzioni con gruppi di record (Capitolo 10).

Si immagini di voler trovare quei clienti che non hanno acquistato alcun articolo da Pine Valley Furniture dall'ottobre del 2000. Si vogliono cioè conoscere quali numeri di cliente dei record nella tabella Customer\_t non sono elencati nei record della tabella Order\_t per ottobre 2000. Le Figure dalla 11.16a alla 11.16d mostrano come si può rispondere alla domanda costruendo un'interrogazione e poi costruendo una seconda interrogazione che utilizza la prima. La prima interrogazione (QBE nella Figura 11.16a e il dynaset nella Figura 11.16b) restituisce i clienti che hanno effettuato ordini in ottobre.

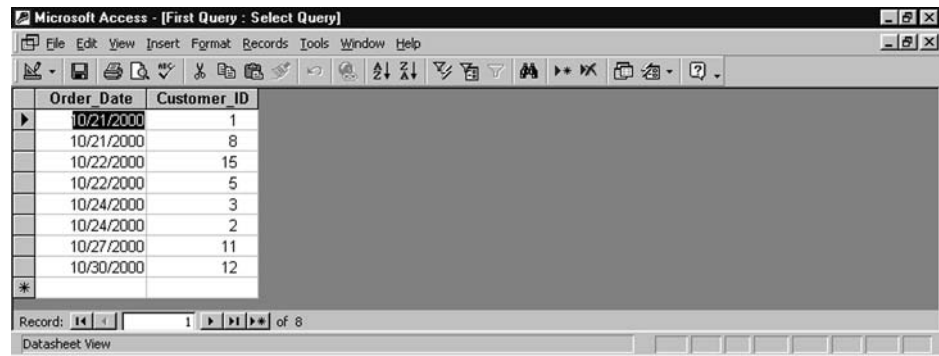
**Figura 11.16**

Interrogazione basata su un'altra interrogazione.

(a) Interrogazione di base di Microsoft Access 2000 da utilizzare per l'interrogazione successiva nella Figura 11.16c.



(b) Dynaset di Microsoft Access 2000 restituito dall'interrogazione nella

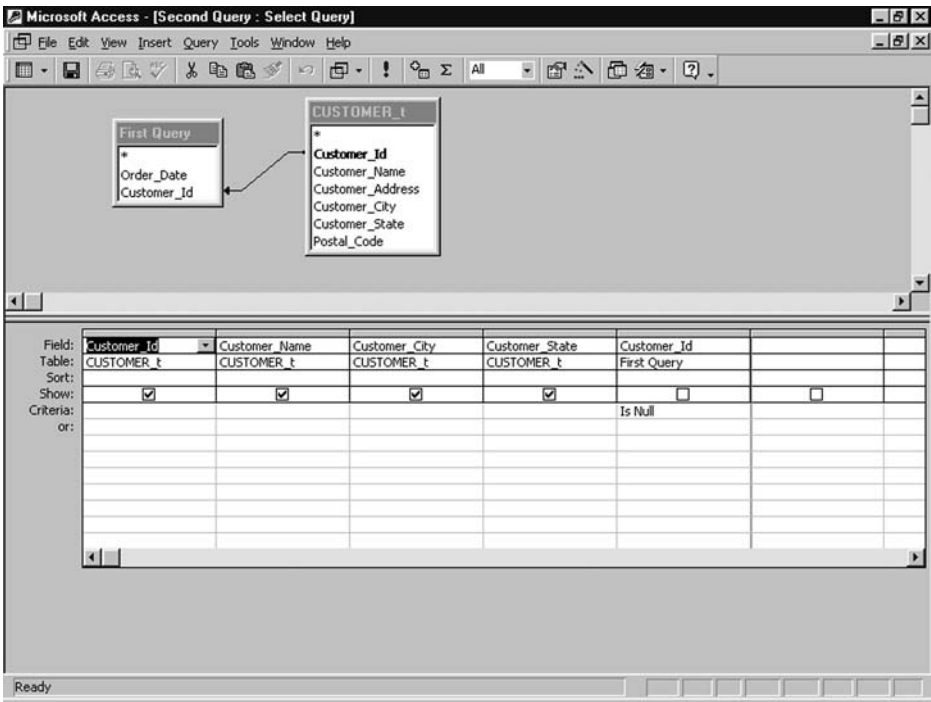


Il QBE per questa interrogazione è stato salvato in un'interrogazione chiamata First Query. La seconda interrogazione (QBE nella Figura 11.16c e il dynaset nella Figura 11.16d) utilizza quindi un join esterno per confrontare l'elenco di tutti i clienti (tabella Customer\_t) con quelli che hanno effettuato ordini (la tabella virtuale definita dalla prima interrogazione) e restituisce un dynaset con i soli clienti che non si trovano nel dynaset risultante dalla prima interrogazione (la qualifica Is Null nella Figura 11.16c).

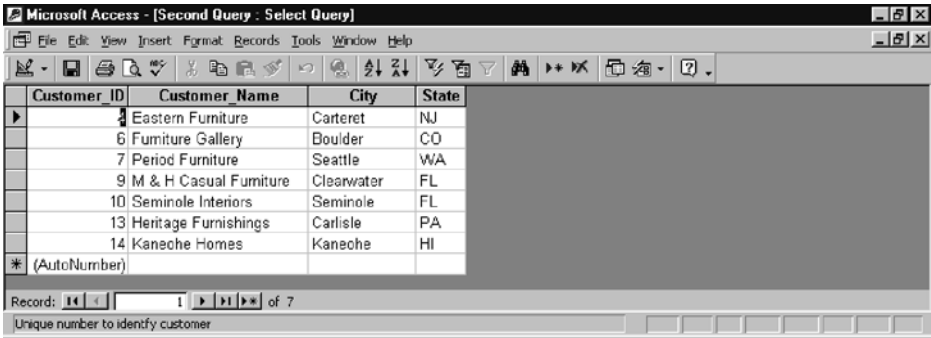
### 11.10.6 Utilizzare interrogazioni pass-through SQL

Una delle ragioni per cui Microsoft Access è diventato popolare come interfaccia client per le applicazioni client/server è la facilità con cui permette di inviare direttamente i comandi a qualsiasi database server ODBC (*Open Database Connectivity*). Utilizzando un'interrogazione pass-through, che è scritta

(b) Dynaset di Microsoft Access 2000 restituito dall'interrogazione nella Figura 11.16a.



(c) Interrogazione di base di Microsoft Access 2000 basata sulla Figura 11.16.a  
(d) Dynaset di Microsoft Access



nel dialetto SQL del database server ODBC invece che con il codice SQL di Access, si lavora direttamente con le tabelle sul server invece di collegarle. Questo aggira il motore di base di dati Jet di Access e garantisce prestazioni più rapide. Il controllo della sintassi, l'interpretazione e la traduzione delle interrogazioni SQL avvengono tutte nel database server. Il traffico di rete viene ridotto perché solo l'interrogazione SQL e i record che sono restituiti devono essere passati tra il server e il client.

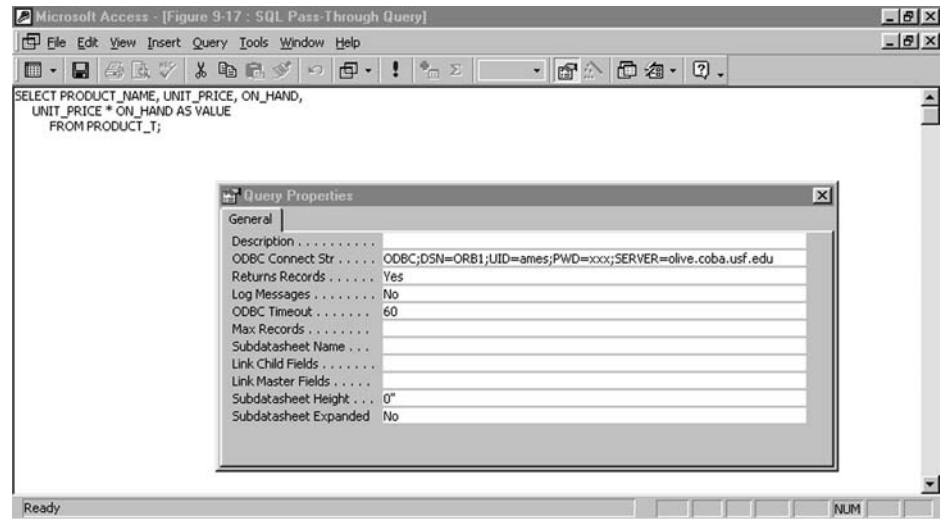
Le interrogazioni pass-through di Microsoft Access possono essere utilizzate per recuperare record, modificare dati o eseguire stored procedure o trigger collocati sul server della base di dati; qui, si possono anche creare nuove tabelle. Si faccia tuttavia attenzione a non eseguire operazioni che influiscano sullo stato della connessione, perché potrebbero verificarsi risultati imprevisti.

Questo metodo non permette di collegare le tabelle della base di dati ODBC ad Access (il collegamento verrà trattato successivamente nel capitolo). Questo significa che non è possibile creare un

insieme di record aggiornabili basato su queste tabelle. L'utente deve avere familiarità con il dialetto SQL utilizzato dalla base di dati ODBC. Laddove ci possono essere notevoli progressi nelle prestazioni e si possa approfittare della potenza dei database server grandi, l'utente deve sapere quale dialetto SQL utilizzare e deve essere preparato a intraprendere più operazioni manuali per raggiungere la funzionalità completa. La creazione di interrogazioni pass-through di Access richiede che venga specificata una stringa di connessione, come proprietà dell'interrogazione stessa o nel momento in cui quest'ultima viene eseguita. La Figura 11.17 mostra la sintassi di Oracle SQL e la finestra delle proprietà di un'interrogazione pass-through di Microsoft Access 2000 contenente un'istruzione Oracle SQL.

**Figura 11.17**

Interrogazione pass-through SQL di Microsoft Access 2000 con la finestra delle proprietà.



È importante sapere che ogni RDBMS, come Oracle, Informix o SQL Server, ha la propria sintassi per la stringa di connessione ODBC che deve essere inserita nella finestra delle proprietà o immessa nel momento dell'esecuzione dell'interrogazione. Per esempio, la sintassi della stringa di connessione Oracle qui utilizzata è la seguente:

ODBC;DSN = *nome connessione* ODBC;UID = *Utente*;PWD = *Password*;SERVER = *Alias di connessione da TnSNames.ora*;

## 11.11 Utilizzare ODBC per collegare tabelle esterne memorizzate su un server di base di dati

Lo standard *Open Database Connectivity (ODBC)* è stato sviluppato all'inizio degli anni '90 dai comitati X/Open e SQL Access Group. ODBC propone diversi livelli di standard che possono essere raggiunti dai RDBMS, consentendo a qualsiasi programma applicativo di utilizzare le interfacce API (*Application Programming Interface*) per accedere ai dati ed elaborarli. Tali RDBMS sono detti *compatibili con ODBC*. Lo standard ha ottenuto ampia accettazione, stimolata inizialmente dall'implementazione

Microsoft di ODBC nei suoi prodotti. ODBC è importante anche per le applicazioni Internet, perché permette lo sviluppo di applicazioni che accedono a prodotti di basi di dati diversi. Per poter raggiungere questa capacità, ODBC utilizza le istruzioni generiche nel codice SQL standard ANSI presentate nel Capitolo 10, ma non è in grado di approfittare dei vantaggi delle estensioni e di funzioni speciali che sono state inserite da ciascun produttore nel proprio motore.

La specifica ODBC permette che le unità siano conformi ai vari livelli della specifica e che influiscano sul livello di funzionalità delle unità. Le differenze nel modo in cui le unità stesse sono scritte possono influenzare le prestazioni. Ogni produttore che desidera una base di dati compatibile con ODBC fornisce un'unità ODBC che può essere installata sulle macchine Windows. Ogni applicazione Windows può quindi comunicare, mediante l'unità opportuna, con la versione desiderata del server della base di dati. Per esempio, un'applicazione Microsoft Access può essere connessa per operare con un database server Oracle. Le tabelle della base di dati sono collegate all'applicazione Microsoft Access attraverso un collegamento ODBC e rimangono nella base di dati Oracle, senza essere portate nella base di dati di Access.

I *database server* Oracle possono essere chiamati anche *server remoti*, *server di back-end* o *server SQL*. Quando un server Microsoft di back-end è chiamato SQL Server, fare riferimento al server SQL può voler dire fare riferimento al tipo di server o a un database server di un produttore specifico. Si può quindi creare confusione.

Per stabilire una connessione ODBC devono essere definiti cinque parametri.

- *Driver* ODBC specifici necessari.
- Nome del *server* di back-end a cui connettersi.
- Nome della *base di dati* a cui connettersi.
- *ID utente* a cui garantire accesso alla base di dati.
- *Password* dell'*ID* utente.

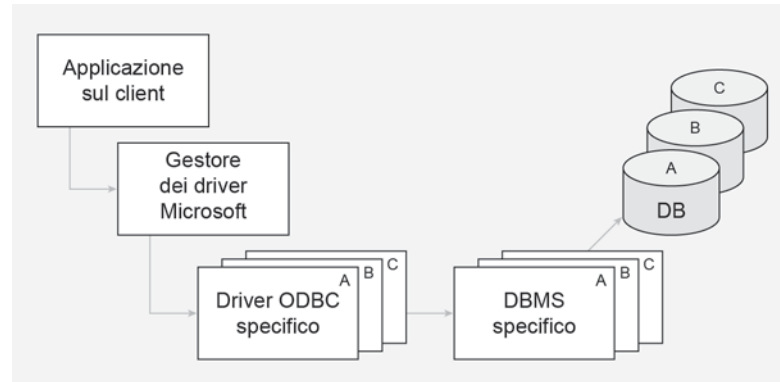
Se si desidera, possono essere fornite alcune informazioni supplementari.

- *Data Source Name* (DSN).
- Nome del computer client Windows.
- Nome dell'eseguibile del programma applicativo del client.

Questi parametri possono essere definiti in più punti e possono essere inclusi nel programma, o attraverso il DSN, o dall'utente quando richiesto. Includere tutti i parametri rende possibile al programma connettersi direttamente alla base di dati senza altre comunicazioni. Il programma dovrà naturalmente essere modificato e i nuovi valori di parametri inseriti per portare il programma su un altro server o RDBMS. Includere il DSN, che comprende alcuni dei valori di parametro, consente agli amministratori locali di collocare la base di dati e di scegliere un RDBMS. L'utente può fornire l'*ID* utente e la *password* quando si collega per utilizzare l'applicazione.

La Figura 11.18 è una rappresentazione di una tipica architettura ODBC. L'applicazione client richiede che venga stabilita una connessione con l'origine dei dati. La richiesta è gestita dal gestore dei driver di Microsoft, che identifica il driver ODBC corretto da utilizzare. Si ricordi che questi driver sono forniti dai produttori; ci può quindi essere un driver SQL Server, un driver Oracle, un driver Informix e così via. Anche le richieste di inizializzazione e la gestione delle richieste ODBC sono gestite dal gestore dei driver. Il driver scelto elaborerà la richiesta ricevuta dal client e invierà le interrogazioni

**Figura 11.18**  
Architettura *Open Database Connectivity* (ODBC).



al RDBMS scelto con la sintassi SQL di quel RDBS specifico. L'entità dell'elaborazione richiesta per creare quell'interrogazione dipende dalla capacità del RDBMS a cui si accede.

Il livello di conformità di ODBC verrà determinato dalle potenzialità del driver che è stato fornito dal produttore. I livelli di conformità definiti sono tre: Core API, API di Livello 1 e API di Livello 2. Questi livelli sono stati definiti dai comitati degli standard citati in precedenza nel capitolo. La maggior parte dei driver fornisce le funzionalità Core API e API di Livello 1, che includono quanto segue.

- Capacità di connettere le origini dei dati con informazioni specifiche del driver.
- Preparare ed eseguire istruzioni SQL.
- Recuperare i dati da un insieme di risultati.
- Confermare o annullare le transazioni.
- Recuperare le informazioni degli errori e di catalogo.
- Inviare e ricevere risultati parziali.
- Recuperare le informazioni sui driver.

Di seguito sono riportate le caratteristiche specifiche delle API di Livello 2.

- Capacità di sfogliare origini di dati e connessioni potenziali.
- Recuperare il codice SQL nativo (dialetto di back-end).
- Chiamare una libreria di traduzione.
- Elaborare e visualizzare un cursore scorrevole.

Le applicazioni determinano il livello del supporto API disponibile quando chiamano il driver. Se occorre un'API di Livello 2 e l'applicazione e un'API di Livello 1, allora la sequenza dell'esecuzione può essere terminata in modo controllato, così che non si verifichi una corruzione dei dati. Alcune applicazioni sono scritte per funzionare sia a Livello 1 sia a Livello 2, a seconda delle capacità del driver per il RDBMS che viene utilizzato.

## 11.12 Utilizzare JDBC per collegare tabelle esterne memorizzate su un server di base di dati

L'API *Java Database Connectivity* (JDBC) permette ai programmi Java di eseguire istruzioni SQL e di connettersi ai database server. JDBC è simile a ODBC, ma è concepito specificatamente per le applicazioni Java. ODBC dipende dal linguaggio. Java è un buon linguaggio da utilizzare negli ambienti client/server perché è orientato alle reti, garantisce un'ottima protezione ed è portabile. Oracle ha adottato Java ed è probabile che il linguaggio proprietario di Oracle, PL/SQL, verrà sostituito da Java per fornire le funzionalità di programmazione necessarie oltre a SQL per costruire le applicazioni delle basi di dati. Lo standard JDBC è concettualmente simile all'ODBC di Microsoft. In base alla X/Open SQL Call Level Interface, JDBC è costituito da due livelli principali. Un livello, l'API JDBC, supporta le comunicazioni da un'applicazione Java al gestore dei driver JDBC. L'altro livello, l'API JDBC Driver, supporta direttamente le comunicazioni dal gestore dei driver JDBC ai driver JDBC, ai driver di rete e ai driver basati su ODBC.

La Figura 11.19 include un semplice esempio del codice necessario per utilizzare JDBC per accedere a una base di dati compatibile con JDBC. Si noti che il codice è stato suddiviso in quattro riquadri per sottolineare le diverse operazioni che vengono eseguite.

Innanzitutto vengono importati nove pacchetti Java predefiniti. Ognuno di essi fornisce le implementazioni di classe per le classi virtuali, come `java.sql.Connection`, `java.sql.Statement`, `java.sql.PreparedStatement` e `java.sql.CallableStatement`. Queste classi virtuali descrivono l'API, ma devono essere specifiche per ciascun prodotto di base di dati che viene connesso.

L'esempio nella Figura 11.19 mostra una connessione a una base di dati Oracle; il riquadro successivo include le informazioni sulla struttura molto semplice della tabella utilizzata per l'esempio e la sequenza di chiave primaria e trigger che può essere utilizzata per inserire il valore di chiave primaria successivo. In questa fase, non è fondamentale capire il codice del trigger, ma è stato incluso per coloro a cui può interessare.

**Figura 11.19**

Utilizzare JDBC per accedere a una base di dati compatibile con JDBC.

```
//package.com.iteamsolutions.eis.tools.sql.sample; Pacchetti Java predefiniti
```

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.SQLException;
import java.sql.Types;
```

```
/**
```

```
* Questa classe è una semplice dimostrazione di come utilizzare JDBC per accedere
* a una base di dati compatibile con JDBC.
```

**Figura 11.19**  
(continua)

### Struttura di tabella Oracle, sequenza di chiave primaria e trigger

\*La seguente struttura di base di dati verrà utilizzata per tutti gli esempi

\* pre

\*

TABLE temp\_demo

\*Name Null? Type

\*

\*ID NOT NULL NUMBER

\*NAME VARCHAR2(20)

\*

\*Sequenza utilizzata per popolare la chiave primaria

\*SEQUENCE temp\_demo\_seq

\*

\*—Trigger utilizzato per popolare la chiave primaria

\*

\*CREATE OR REPLACE TRIGGER temp\_demo\_trig

\*BEFORE INSERT ON temp\_demo

\*FOR EACH ROW

\*BEGIN

\*—Consente che la chiave si auto-popoli

\*—può essere recuperato utilizzando la clausola RETURNING

\*—dell'istruzione INSERT

\*—IF(:NEW.id IS NULL)

\*THEN

\* SELECT temp\_demo\_seq.nextval

\* INTO :NEW.id

\* FROM dual;

\*END IF;

\*END;

\* /pre )

\* (br )

### Inizializzazione

```
public class OracleSqlTest
```

```
{
```

```
 /**costante utilizzata per recuperare l'ID di colonna della tabella temp_demo*/
```

```
 public static final String ID = "id";
```

```
 /**costante utilizzata per recuperare il nome di colonna della tabella temp_demo*/
```

```
 public static final String NAME = "name";
```

```
 /**nome dell'account della base di dati*/
```

```
 private String NAME = "nome utente";
```

```
 /**password dell'utente*/
```

```
 private String m_password = "password utente";
```

```
 /**classe del driver da utilizzare per la connessione alla base di dati*/
```

```
 private String m_driver = "oracle.jdbc.driver.OracleDriver";
```

```
 /**tipo di driver da utilizzare*/
```

```
 private String m_drivertype = "jdbc:oracle:thin:@";
```



**Figura 11.19***(continua)*

```

/**base di dati a cui connettersi*/
private String m_database = "nomeserver:1521:nomebasedati";

/**inserire un nuovo record in temp_demo*/
private String m_insert = "INSERT INTO temp_demo (id, name) values (?,?)";

/**inserire un nuovo record in temp_demo*/
private String m_insertReturning = "BEGIN\n" +
 "INSERT INTO temp_demo (name) values (?)\n" +
 "RETURNING id INTO ?;\n" +
 "END;";

/**connessione utilizzata da questa classe per fare tutto il lavoro*/
private Connection m_conn = null;

/**
Costruttore predefinito; può essere modificato per fornire i valori precedenti
* o per trarli da un file delle proprietà.
*
* @throws eccezione; il fallimento è una condizione terminale
*/
public OracleSqlTest()
 throws Exception
{
 m_conn = buildConnection();
}

/**
*inserisce i valori passati nella tabella temp_demo
*
* @param id - numero da utilizzare come chiave primaria nel record
* @param name - nome da memorizzare nella colonna dei nomi
*/
public void insert(int id, String name)
 throws SQLException
{
 PreparedStatement stmt = null;

 try
 {
 //costruisce un'istruzione preparata; consente che i valori con "?"
 //siano sostituiti nella stringa passata
 stmt = m_conn.prepareStatement(m_insert);

 //lega il valore all'argomento (?)
 //il primo argomento è la posizione di ? da sostituire.
 //Il conteggio della posizione inizia da 1
 stmt.setInt(1, id);
 stmt.setString(2, name);

 //esegue l'istruzione
 stmt.execute();
 }
}

```

**Figura 11.19**  
(continua)

```

catch (SQLException e)
{
 throw e;
}
finally
{
 //garantisce il rilascio di tutte le risorse
 //se non avviene, la base di dati genera un'eccezione
 //perché sono aperti
 //troppi cursori
 if(stmt! = null)
 stmt.close();
 }
}

```

#### Costruzione dell'oggetto per l'accesso a una base di dati Oracle

```

/**
 *si occupa di costruire l'oggetto di connessione per l'accesso a una base di dati Oracle;
 */
private Connection buildConnection()
 throws SQLException
{
 //occorre assicurarsi che il driver per la base di dati sia caricato
 //ogni eccezione generata viene catturata e viene generata una SQLException
 //che deve essere catturata dal client.
 try
 {
 DriverManager.registerDriver
 ((Driver) Class.forName(m_driver).newInstance());
 }
 catch (InstantiationException e)
 {
 throw new SQLException("Unable to instantiate the Oracle Driver");
 }
 catch (IllegalAccessException e)
 {
 throw new SQLException("IllegalAccessException thrown when attempting to
 load the Oracle Driver class");
 }
 catch (ClassNotFoundException e)
 {
 throw new SQLException("Oracle Driver class not found");
 }
 //se non ci sono problemi, il driver viene registrato e restituisce una connessione
 return (DriverManager.getConnection (m_drivertype + m_database,
 m_username,
 m_password));
}

```

Successivamente si verifica l'inizializzazione e vengono specificati i valori come l'ID utente, la password, la classe del driver, il tipo di driver, la stringa di connessione alla base di dati e così via. Il riquadro più grande include il codice che mostra come stabilire la connessione alla base di dati (utilizzando `buildConnection()`) e come inserire i valori passati nella tabella `temp_demo`.

L'ultimo riquadro include il codice necessario a costruire l'oggetto di connessione per poter aver accesso alla base di dati Oracle, includendo i messaggi d'errore che verranno visualizzati se la connessione, per una ragione o per un'altra, non viene stabilita. Un esempio più completo di JDBC viene incluso anche sul sito Web del libro. Include un altro metodo per inserire il valore di chiave primaria e il codice non solo per creare un nuovo record, ma anche per modificare, aggiornare o eliminare i record.

## 11.13 Utilizzare Visual Basic for Applications (VBA) nelle applicazioni client

Access 2000 permette all'utente neofita di creare un prototipo di un'applicazione che comprenda il progetto della base di dati, i menu, le maschere e i report. Esiste tuttavia un limite a quanto può essere ottenuto utilizzando le macro di Access e lo sviluppatore viene ostacolato quando tenta di includere tutte le funzionalità desiderate. Mentre alcune di queste richiedono le macro e non possono essere ottenute attraverso *Visual Basic for Applications (VBA)*, molte altre possono essere ottenute o migliorate utilizzando VBA in unione con Access 2000. Ecco alcune delle ragioni per cui un utente di Access 2000 può voler imparare VBA una volta che ha raggiunto familiarità con Access 2000 (Smith e Sussman, 1997).

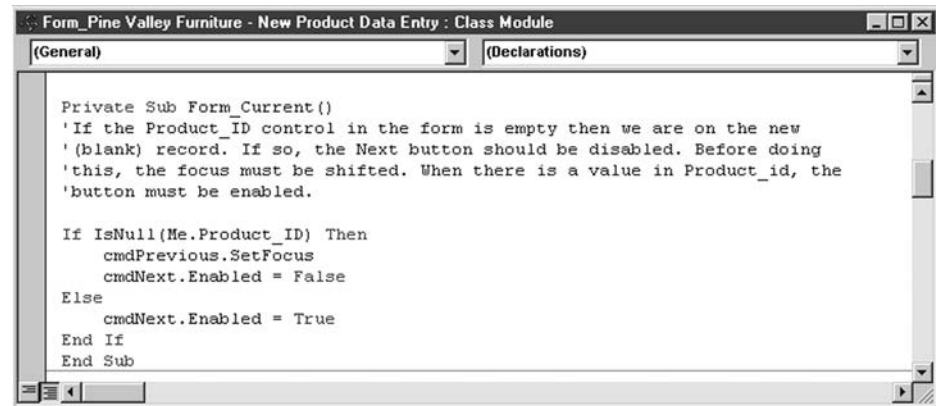
- Le *funzionalità più complesse* possono essere ottenute solo utilizzando VBA. La visualizzazione di messaggi d'errore, l'attenuazione del colore grigio dei pulsanti che non devono essere selezionati e così via richiedono il codice VBA.
- La *gestione degli errori* può essere ottenuta solo utilizzando VBA. Basarsi sulle macro in un'applicazione finita è pericoloso, perché non c'è modo per l'utente di ripristinare i dati se una macro fallisce.
- Un'*esecuzione più veloce* viene raggiunta quando l'applicazione utilizza moduli VBA invece che macro. Il codice viene eseguito più velocemente delle macro; questo vantaggio emerge più chiaramente quando l'applicazione diventa grande e complessa.
- La *manutenzione è più semplice* perché i moduli VBA sono memorizzati con le maschere e i report. Le macro sono memorizzate separatamente dalle relative maschere e report. Lo spostamento dell'applicazione su un'altra base di dati sarà più semplice perché i moduli VBA si sposteranno con le maschere e i report, che sono memorizzati al loro interno.
- L'*automazione OLE* può essere utilizzata in modo più completo.
- Un *maggiore controllo di programmazione* può essere ottenuto utilizzando VBA. Le macro non possono passare le variabili come parametri per un'altra variabile e non si può controllare facilmente la sequenza dell'azione.
- *Leggere* il codice VBA è più facile che leggere gli argomenti delle macro, perché si può vedere tutto il modulo nella vista a modulo intero, che ha il testo codificato con il colore.

Per aprire una base di dati, deve essere creata una macro a esecuzione automatica e le macro devono essere utilizzate anche per isolare alcune combinazioni da tastiera nelle applicazioni. Inoltre, la conversione di un prototipo dalle macro in moduli VBA per una versione di produzione dell'applicazione genererà un'applicazione più solida con numerose funzionalità.

Il sistema operativo e le applicazioni Windows sono **guidati dagli eventi**. Tutto quanto accade in un'applicazione di Access 2000 è una risposta a un evento, come un clic del mouse, che Access ha individuato. Un evento accade, viene individuato e viene generata una risposta a un evento. Access 2000 non è quindi procedurale; l'evento successivo può infatti provenire da un punto qualsiasi dell'interfaccia o derivare dalla pressione di un tasto. La programmazione in un ambiente guidato dagli eventi è costituita dalla creazione di oggetti e dalla modifica delle loro proprietà in modo che si comportino come desidera il programmatore per ogni evento che influisce sull'oggetto.

La Figura 11.20 mostra il codice VBA che fa sì che il pulsante NEXT sulla maschera Product Entry si disattivi (diventi grigio) quando viene visualizzato un nuovo record. I record per la maschera sono tratti dalla tabella Product\_t. Product\_ID è la chiave primaria della tabella. Il valore del Product\_ID del record corrente viene determinato utilizzando Me.Product\_ID. Se non c'è un valore di chiave primaria per il record corrente, dovrà esserci un nuovo record, e il pulsante NEXT verrà disabilitato. Poiché la disattivazione del pulsante che ha il focus causa un errore di runtime, il focus deve prima essere spostato in un altro punto. Il focus viene spostato sul pulsante PREVIOUS e il pulsante NEXT viene disabilitato. Se il record corrente non è un nuovo record, il pulsante NEXT rimane abilitato.

**Figura 11.20**  
Modulo VBA di esempio.



## Riepilogo

Le architetture client/server hanno offerto alle aziende l'opportunità di adattare al meglio i loro sistemi di computer alle proprie esigenze. Stabilire un equilibrio appropriato tra DBMS server/client e mainframe è oggetto di grande discussione. Le architetture client/server si distinguono nel fornire applicazioni Internet, compreso l'accesso ai dati dinamici. Queste architetture sono trattate in questo capitolo, insieme alle architetture dei file server, dove il file server gestisce le operazioni di file ed è condiviso da tutti i PC client connessi alla sua LAN. I file server generano un carico di rete molto pesante, necessitano di una versione completa del DBMS su ciascun client e richiedono una programmazione complessa per poter gestire l'integrità della base di dati.

Un altro approccio, l'architettura del database server, fa sì che il client gestisca l'interfaccia utente mentre il server della base di dati gestisce la memorizzazione e l'accesso. Questa architettura riduce il traffico di rete e la potenza richiesta da ciascun client e centralizza l'autorizzazione degli utenti, il controllo dell'integrità, la manutenzione dei dizionari dei dati e l'elaborazione delle interrogazioni e degli aggiornamenti sul server della base di dati.

Le architetture a tre livelli, che includono un altro server oltre ai livelli del client e del database server, permettono che il codice delle applicazioni venga memorizzato sul server supplementare, dove avviene anche l'elaborazione, generando così un client sottile. I vantaggi dell'architettura a tre livelli possono includere la scalabilità, la flessibilità tecnologica, la riduzione dei costi sul lungo termine, una migliore corrispondenza tra sistemi ed esigenze aziendali, un servizio clienti migliorato, vantaggi competitivi e rischi ridotti. Gli svantaggi sono però i costi più alti sul breve termine, la necessità di strumenti avanzati e di formazione, la scarsità di personale qualificato, la presenza di standard non compatibili e la mancanza di strumenti per gli utenti finali.

Il partizionamento delle applicazioni assegna porzioni del codice delle applicazioni alle partizioni client o server dopo che il codice è stato scritto, così da ottenere prestazioni migliori e inter-operabilità. Uno dei risultati del partizionamento è un aumento della produttività dello sviluppatore, ma questi deve capire il processo in profondità per poterlo collocare correttamente.

Sebbene l'ampia diffusione delle architetture client/server abbia portato a ipotizzare l'abbandono del mainframe, le applicazioni cruciali per le aziende tendono ancora a rimanere sui mainframe. Convertire queste applicazioni complesse in ambienti client/server distribuiti non è semplice. La disponibilità di soluzioni di elaborazione parallela favorisce la tendenza alla centralizzazione su un application server o di base di dati.

Poiché SQL è un linguaggio di elaborazione non procedurale, si è rivelato perfettibile quando utilizzato in ambienti paralleli. Sono stati affrontati due tipi di elaborazione parallela. SMP è un sistema multiprocessore strettamente accoppiato con una memoria comune condivisa tra tutti i processori. I sistemi SMP tendono a sviluppare colli di bottiglia di I/O quando il traffico di richiesta sulla memoria condivisa è molto elevato.

Le architetture non strettamente accoppiate (MPP) impostano ciascuna CPU con la propria memoria dedicata. Questo tipo di architettura è chiamata anche architettura di non condivisione. Poiché la condivisione delle risorse tra i processori MPP è scarsa, i problemi di contenimento della memoria che si verificano nei sistemi SMP sono molto rari; è inoltre possibile aggiungere nodi in unità singole, rendendo le architetture MPP particolarmente scalabili.

Il middleware è una delle numerose classi di software che permettono a un'applicazione di inter-operare con altri software senza che l'utente debba capire e codificare le operazioni al livello più basso necessarie per ottenere l'inter-operabilità. Sono state trattate sei categorie di middleware, basate sui criteri di scalabilità e di ripristino: RPC asincrono, pubblica/iscrivi, MOM, ORB, middleware di accesso ai dati orientato a SQL e RPC sincrono. Gli sviluppi recenti nel middleware orientato alle basi di dati includono ODBC, JDBC di Java e OLE-DB di Microsoft.

Di recente ha suscitato molto interesse la connessione delle basi di dati al Web, che permette ai browser di interagire con il Web per effettuare ordini, accedere a informazioni aggiornate sui prezzi e così via. Il Web sta cambiando i pattern di distribuzione dei dati, spostando la logica delle applicazioni su server più centralizzati nel momento in cui vengono utilizzate interfacce di browser.

La protezione dei dati è più complessa in un ambiente client/server che non in un ambiente centralizzato, poiché oltre alle stazioni di lavoro e ai server, devono essere protette anche le reti. Le misure di protezione da includere in un ambiente client/server includono password a livello di sistema e a livello di base di dati e comunicazioni sicure tra client e server.

Tra i problemi da affrontare per migliorare le opportunità di costruire un'applicazione client/server di successo vi sono i seguenti: analisi accurata dei problemi aziendali, analisi dettagliata dell'architettura, non adozione di architetture guidate dagli strumenti, ottenimento di una scalabilità appropriata, collocazione opportuna dei servizi, analisi adeguata della rete e consapevolezza dei costi potenziali nascosti.

I vantaggi legati al passaggio a un ambiente client/server sono la distribuzione delle funzionalità in più passaggi, la flessibilità, la scalabilità, il traffico di rete ridotto e lo sviluppo di applicazioni abilitate per il Web.

I servizi di presentazione della maggior parte delle architetture client/server utilizzano interfacce Query-by-Example (QBE). QBE fornisce un ambiente di programmazione visuale utilizzato per lo sviluppo delle interrogazioni. Non esistendo uno standard per QBE, le interfacce variano a seconda dei prodotti, ma la maggior parte di esse utilizza un approccio di modello di dati combinato con un layout di foglio elettronico per visualizzare i risultati. L'interfaccia QBE di Microsoft Access 2000 utilizza ampiamente questo layout, includendo tanto la definizione di tabelle e interrogazioni quanto quella delle relazioni tra i dati.

I pacchetti di basi di dati relazionali meno solidi hanno guadagnato popolarità grazie alle loro interfacce GUI e QBE. Per esempio, Microsoft Access 2000 utilizza interrogazioni pass-through per inviare direttamente i comandi a qualsiasi database server che sia compatibile con ODBC. Le tabelle della base di dati possono essere collegate all'interfaccia di un'applicazione di Access se la base di dati è compatibile con ODBC o con JDBC. Per poter stabilire una connessione ODBC o JDBC devono essere definiti numerosi parametri; tra questi, i driver ODBC e JDBC necessari, il nome del database server di back-end, il nome della base di dati e l'ID e la password dell'utente della base di dati. Con lo standard ODBC esistono livelli di conformità diversi, e ciò permette che la maggior parte degli applicativi raggiunga almeno un livello di connessione minimo alla base di dati desiderata.

Il linguaggio di programmazione associato ad Access 2000 è Visual Basic for Applications (VBA). L'uso di macro di Access 2000, che sono moduli memorizzati di codice VBA, è ideale per la creazione di prototipi, ma la creazione di un'applicazione robusta richiede la conversione di queste macro in moduli VBA. VBA può essere utilizzato anche per ottenere funzionalità più complesse, per gestire errori, per ottenere un'esecuzione più veloce, una manutenzione più facile, una maggiore automazione OLE e un controllo migliorato della programmazione.

## Ripasso del capitolo

### Parole chiave

|                                            |                                            |                                                   |                                            |
|--------------------------------------------|--------------------------------------------|---------------------------------------------------|--------------------------------------------|
| <i>Application Program Interface</i> (API) | Database server                            | Partizionamento delle applicazioni                | Stored procedure                           |
| Architettura a tre livelli                 | File server                                | <i>Query-by-Example</i> (QBE)                     | <i>Symmetric Multiprocessing</i> (SMP)     |
| Architettura client/server                 | Guidato dagli eventi                       | Standard <i>Open Database Connectivity</i> (ODBC) | <i>Visual Basic for Applications</i> (VBA) |
| Client sottile                             | <i>Massively Parallel Processing</i> (MPP) |                                                   |                                            |
| Client spesso                              | Middleware                                 |                                                   |                                            |

### Domande di ripasso

- Definire ciascuno dei seguenti termini chiave:
  - partizionamento delle applicazioni
  - Application Program Interface* (API)
  - architettura client/server
  - client spesso
  - file server
  - middleware
- Unire i seguenti termini alle rispettive definizioni:
 

|                                   |                                                           |
|-----------------------------------|-----------------------------------------------------------|
| _____ architettura client/server; | _____ <i>Application Program Interface</i> (API);         |
|                                   | _____ client spesso; _____ database server;               |
|                                   | _____ file server; _____ middleware;                      |
|                                   | _____ architettura a tre livelli;                         |
|                                   | _____ <i>Symmetric Multi-Processing</i> (SMP);            |
|                                   | _____ <i>Massively Parallel Processing</i> (MPP);         |
|                                   | _____ client sottile; _____ sistema guidato dagli eventi; |
|                                   | _____ interrogazione pass-through; _____ QBE.             |

3. Elencare i vantaggi principali delle architetture server/client rispetto ad altri approcci.
4. Confrontare i seguenti termini.
  - a. *Symmetric Multi-Processing* (SMP); architettura di non condivisione (MPP)
  - b. file server; server della base di dati; architettura a tre livelli
  - c. ambiente client/server; ambiente mainframe
  - d. client spesso; client sottile
  - e. Query-by-Example; interfaccia lineare
5. Descrivere i limiti dei file server.
6. Descrivere i vantaggi e gli svantaggi dei database server.
7. Descrivere i vantaggi e gli svantaggi delle architetture a tre o a *n* livelli.
8. Come può il partizionamento delle applicazioni aiutare gli sviluppatori ad adattare il sistema a una particolare situazione aziendale?
9. Descrivere sei categorie di middleware.
10. In che modo il Web sta cambiando i pattern di distribuzione dei dati?
11. Spiegare i vantaggi e gli svantaggi di QBE rispetto a SQL.
12. Qual è lo scopo della specifica ODBC?
13. Quali problemi di protezione emergono quando si costruisce un'applicazione abilitata per il Web?
14. Spiegare i rapporti tra Access 2000 e VBA.
8. Visualizzare il numero d'ordine, il numero di cliente, il numero di prodotto, la data d'ordine e la quantità per gli ordini di più di quattro unità dei prodotti 3 o 8.
9. Visualizzare il numero di cliente, il nome e i numeri d'ordine per tutti gli ordini dei clienti e includere nell'elenco qualsiasi cliente che non abbia ordini nella base di dati.
10. Scrivere un'interrogazione che visualizzi ogni prodotto e la quantità totale ordinata per ciascun prodotto, compresi i prodotti che non sono mai stati ordinati.
11. Visualizzare un elenco di tutti i prodotti e del numero di volte in cui ogni prodotto è stato ordinato.
12. Visualizzare il numero d'ordine, il numero di prodotto e la quantità d'ordine per tutti gli ordini dei clienti la cui quantità ordinata è maggiore della quantità media ordinata di quel prodotto. *Suggerimento*: potrebbe essere necessaria più di un'interrogazione.

### Esercitazioni

#### Problemi ed esercizi

1. Si immagini di dover preparare un report che valuti le possibili soluzioni client/server per gestire un nuovo sistema di applicazioni per i clienti per tutte le filiali. Quali caratteristiche aziendali e tecnologiche occorre valutare? Perché?
2. Quali problemi manageriali potrebbero diventare importanti qualora si introducesse una nuova architettura client/server?
3. Illustrare i diversi livelli di protezione che dovrebbero essere stabiliti in un sistema di base di dati client/server.
4. Che tipo di influenza ha il Web sui sistemi di basi di dati client/server?
5. Perché ODBC è importante? Quali altri standard di connettività sono in corso di sviluppo?
6. Storicamente, quali tipi di applicazioni sono passate più rapidamente ai sistemi di basi di dati client/server? Quali invece hanno visto un passaggio più lento e perché? Cosa si ritiene possa accadere in futuro per equilibrare i sistemi di basi di dati client/server con quelli mainframe?
7. Quali sono i vantaggi e gli svantaggi del middleware utilizzato in combinazione con i sistemi di basi di dati?  
I problemi ed esercizi da 8 a 12 si basano sulla base di dati di Pine Valley Furniture e sono concepiti per esercitarsi con un'interfaccia QBE.
1. Indagare sull'architettura di computer della propria università. Tracciarne la storia e determinare qual è il percorso seguito dall'università per arrivare alla configurazione corrente. Alcune università hanno cominciato fin dall'inizio come ambienti mainframe, mentre altre hanno adottato i PC. Come l'ambiente di calcolo iniziale della propria università ha influito su quello attuale?
2. Su una scala ridotta, indagare sull'architettura di computer di un dipartimento della propria università. Si cerchi di capire come il sistema corrente soddisfa le esigenze di elaborazione delle informazioni del dipartimento.
3. Intervistare un professionista dei sistemi di un'azienda locale che ha uffici in più luoghi. Determinare l'architettura di computer dell'azienda e capire come il sistema corrente soddisfa le esigenze di elaborazione delle informazioni.
4. Individuare tre siti Web a cui sono collegati dei sistemi di basi di dati interattivi. Valutare la funzionalità dei siti e considerare come il sistema delle basi di dati interattivo influisce sulla base di dati. Se si desidera un punto di partenza, si visiti il sito <http://www.amazon.com>.
5. Individuare due piccole aziende simili nella propria zona che utilizzano sistemi di computer diversi per mantenere i record. Confrontare la funzionalità e la facilità d'uso dei due sistemi.
6. Contattare un analista di basi di dati o uno sviluppatore di sistemi di un'azienda con cui si ha familiarità. Indagare sull'uso dei sistemi di basi di dati di server e PC. Quali fattori vengono considerati quando si sceglie la piattaforma per una base di dati? L'azienda utilizza una piattaforma mista, nella quale i moduli delle applicazioni basate sul client sono scritti in un linguaggio (per esempio, QBE) e i moduli delle applicazioni basate sul server sono scritti in un altro linguaggio (per esempio, SQL)? Perché sì o no?

## Bibliografia

- Anderson, G. e Armstrong, B., 1995, "Client/Server: Where Are We Really?", *Health Management Technology* 16, 6 (maggio), pp. 34, 36, 38, 40, 44.
- Atre, S., 1995, "The Hidden Costs of Client/Server", *DBMS* 8, 7 (giugno), pp. 71, 72, 74.
- Bobrowski, S., 1994, "Database Security in a Client/Server World", *DBMS* 7, 10 (ottobre), pp. 48–50, 54, 58.
- DeWitt, D. e Gray, J., 1992, "Parallel Database Systems: The Future of High Performance Database Systems", *Communications of the ACM* 35, 6 (giugno), pp. 85–98.
- Ferguson, M., 1994, "Parallel Database the Shape of Things to Come", *Database Programming & Design* 7, 10 (ottobre), pp. 32–44.
- Frazer, W. D., 1998, "Object/Relational Grows Up", *Database Programming & Design* 11, 1 (gennaio), pp. 22–28.
- Hurwitz, J., 1996, "Managing Complexity", *DBMS* 9, 5 (maggio), pp. 12, 80.
- Hurwitz, J., 1998, "Sorting Out Middleware", *DBMS* 11, 1 (gennaio), pp. 10–12.
- Keuffel, W., 1997, "CORBA Masterminds Object Management", *DBMS* 10, 3 (marzo), pp. 42–50, 71.
- Greenblatt, D. e Waxman, J., 1978, "A Study of Three Database Query Languages" in *Database: Improving Usability and Responsiveness*, ed. B. Schneiderman, New York, Academic Press.
- LaRue, M., 1997, "Database Doorways", *Database Programming & Design* 10, 12 (dicembre), pp. 62–67.
- Linthicum, D. S., 1996, "Client/Server Collapse", *DBMS* 9, 13 (dicembre), pp. 24, 26, 28.
- Linthicum, D. S., 1997, "Next-Generation Middleware", *DBMS* 10, 10 (ottobre), pp. 69–78.
- Prague, C. e Irwin, M., 1997, *Access 97 Bible*, Foster City, CA, IDG Books.
- Quinlan, T., 1995, "The Second Generation of Client/Server", *Database Programming & Design* 8, 5 (maggio), pp. 31–39.
- Rudin, K., 1995, "The Practical Side of Going Parallel", *Database Programming & Design* 8, 12 (dicembre), pp. 26–33.
- Schneiderman, B., 1983, "Direct Manipulation: A Step Beyond Programming Languages", *IEEE Computer* 16, pp. 57–69.
- Smith, R. e Sussman, D., 1997, *Beginning Access97 VBA Programming*, Birmingham, UK, Wrox Press.
- Thomas, J. C. e Gould, J. D., 1975, "A Psychological Study of Query by Example", *Proceedings of National Computer Conference*, New York, AFIPS Press.
- Thompson, C., 1997, "Committing to Three-Tier Architecture", *Database Programming & Design* 10, 8 (agosto), pp. 26–33.
- Zloof, M. M., 1977, "Query-by-Example: A Data Base Language", *IBM Systems Journal* 16 (4), pp. 324–343.



## Project Case



# MOUNTAIN VIEW COMMUNITY HOSPITAL

### Descrizione del progetto

Al termine del Capitolo 4, si è visto che il team speciale di Mountain View Community Hospital sta sviluppando un piano aziendale sul lungo termine. Il team, formato da Mr. Heller, Mr. Lopez, dal Dr. Jefferson e da un consulente, ha considerato le future esigenze tecnologiche dell'ospedale. Attualmente, i pazienti devono affrontare un labirinto di piani di cura, amministratori, medici e cliniche simile a quello mostrato nella Figura 1. Ciò che vorrebbero è un sistema che integri tutti questi dati (dei piani di cura, dei medici e dei sistemi ospedalieri), così che siano disponibili informazioni precise in tempo reale.

Il team è venuto a conoscenza che molti ospedali hanno iniziato a convertire i propri sistemi aziendali cruciali da sistemi legacy su mainframe a nuovi sistemi client/server. Hanno saputo che tali sistemi forniscono una flessibilità, integrazione e funzionalità migliori. Ci si sta quindi chiedendo se sia il caso di integrare i sistemi correnti di contabilità, gestione delle cure e assicurazione dell'ospedale per migliorare l'efficienza e per supportare sistemi avanzati. Il consulente ha steso una bozza di una potenziale architettura client/server distribuita per l'ospedale, mostrata nella Figura 2. Per realizzare il passaggio all'ambiente client/server, il team ha considerato tre strategie diverse, discusse nei paragrafi che seguono.

Una possibilità è quella di ottenere pacchetti di applicazioni client/server esistenti e di adattarli a Mountain View Community Hospital. Questo approccio permette allo staff MIS esistente di acquisire familiarità con porzioni cruciali dell'ambiente client/server, come la gestione delle reti, della configurazione dei PC e dei sistemi distribuiti, senza dover gestire un progetto di sviluppo del tutto nuovo e su vasta scala oltre al lavoro corrente. La maggior parte dei pacchetti esaminati richiede tuttavia l'uso di un middleware proprietario per le connessioni client/server. L'ospedale prevede quindi di poter proseguire la connessione ai sistemi legacy attraverso i server proposti e non sa con certezza quanti sono i programmi disponibili che potrebbero permettere di passare all'ambiente auspicato.

Un'altra possibilità considerata dal team è quella di uno sviluppo personalizzato. Alcuni grandi ospedali hanno intrapreso uno sviluppo personalizzato dei propri sistemi client/server. Nonostante i

rischi legati a questa iniziativa, gli ospedali più grandi hanno accettato di correre questo rischio pur di ottenere i dati e le informazioni in tempo reale. Il team di Mountain View team è piuttosto preoccupato dei rischi legati a questo tipo di progetto e alla necessità di ingrandire lo staff dei sistemi informativi che ne conseguirebbe.

L'ultima possibilità considerata è quella di collocare le applicazioni client/server al livello superiore dei sistemi esistenti. Le applicazioni di flusso di lavoro potrebbero essere utilizzate per facilitare la definizione dell'interfaccia front-end. La necessità di connettersi a un data warehouse emerge con forza sempre maggiore man mano che il team lavora sui piani a lungo termine e per questo il team vuole pianificare le infrastrutture in questa fase, così che la definizione di un data warehouse sia il più semplice possibile.

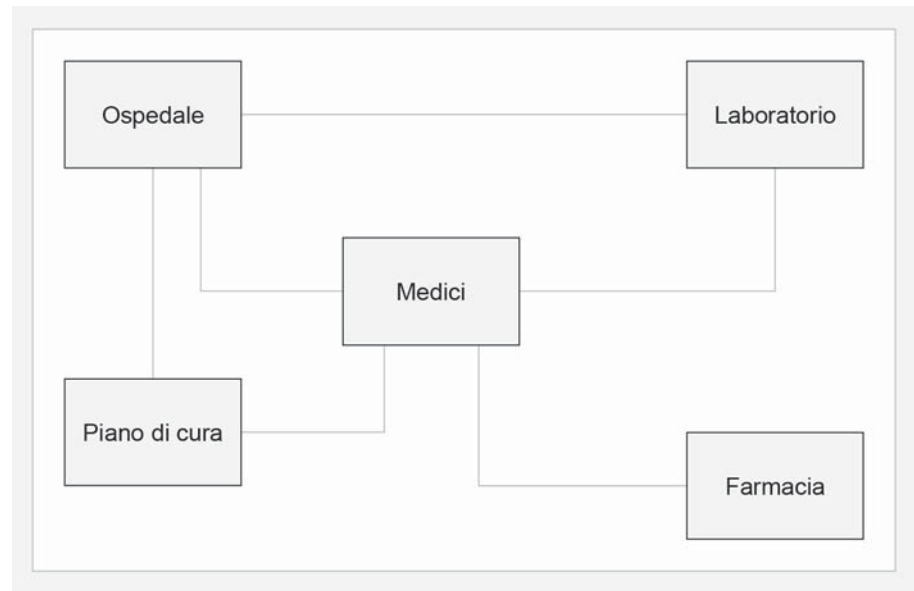
Mountain View Community Hospital può adottare naturalmente anche un sistema abilitato per il Web. Le diverse unità di cura stanno estendendo i servizi dei loro membri al di là dei confini dell'azienda, nei luoghi di lavoro, nelle scuole e nelle case. I membri del piano di cura possono verificare lo stato dei pagamenti, inviare messaggi ai rappresentanti dei servizi e rivedere la destinazione dei fondi. I pazienti possono anche prendere appuntamento accedendo a programmi di appuntamento specifici. I membri del team non sono certi dell'impatto che questi nuovi servizi avranno sulla loro zona, ma si aspettano che altri ospedali adottino e implementino rapidamente queste stesse funzioni. Stanno quindi valutando le altre opzioni, tra le quali il Web.

### Domande sul progetto

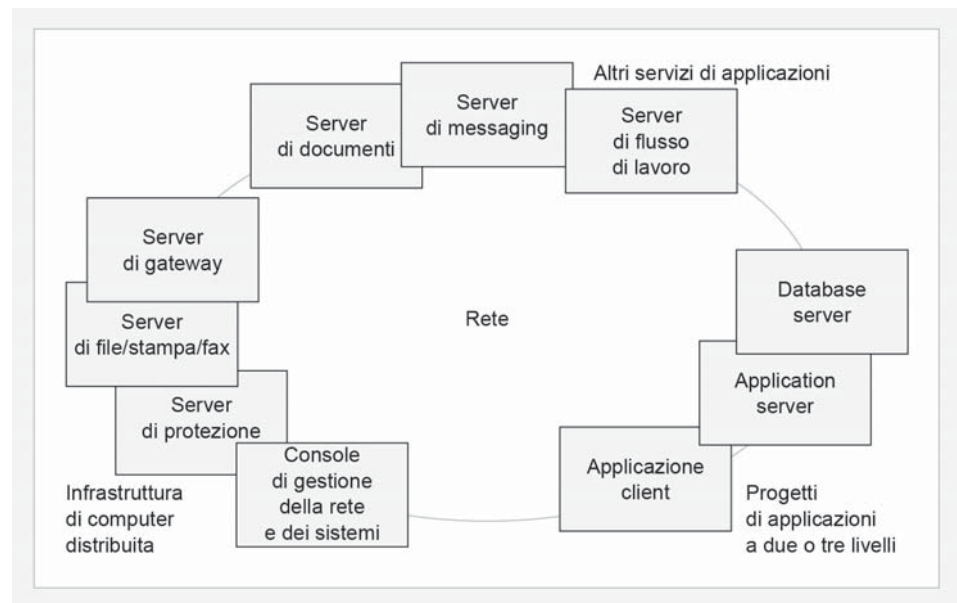
1. Perché il team di Mountain View ritiene che costruire applicazioni client/server personalizzate possa essere rischioso?
2. Quali sono alcune delle possibilità per elaborare e fornire informazioni che potrebbero sorgere qualora Mountain View passasse a un ambiente client/server?
3. Quali attività dovrebbero essere pianificate se Mountain View decidesse di adottare e implementare applicazioni client/server standard? Si scelga un'attività, come il sistema delle fatture, e si consideri come potrebbe essere coinvolto nella scelta, installazione e uso di tale sistema.

**Figura 1**

La base tecnologica corrente di Mountain View Community Hospital (adattato da Anderson e Armstrong, 1995).

**Figura 2**

Una potenziale architettura client/server distribuita (adattato da Anderson e Armstrong, 1995).



4. Quali vantaggi e svantaggi sono legati al terzo approccio, che colloca le applicazioni client/server al livello superiore dei sistemi esistenti?
5. Quali vantaggi e svantaggi si potrebbero avere qualora Mountain View decidesse di mantenere i sistemi attuali per i prossimi due anni?

### Esercizi sul progetto

1. Indicare i pro e i contro di ciascuno dei quattro approcci del passaggio a un ambiente client/server: personalizzazione di sistemi standard (COTS), sviluppo di sistemi personalizzati, modifica dei livelli e nessun provvedimento per un certo periodo di tempo.
2. Utilizzando gli elenchi sviluppati nel primo progetto sull'esercizio, indicare la soluzione che si ritiene ideale per Mountain View Community Hospital. Motivare la propria risposta. Indicare le informazioni supplementari eventualmente necessarie a sostenere questa scelta.
3. Cercare di procurarsi dei sistemi client/server standard per ospedali, come Symphonia di Orion, Meridia Managed Care Information System o Facets di Erisco Managed Care Technologies. Informarsi sulle potenzialità di ciascun sistema, sul numero di installazione, sul sistema operativo utilizzato e così via. Determinare quali dei sistemi individuati e studiati potrebbe essere di qualche utilità per Mountain View.

**Al termine di questo capitolo sarete in grado di fare quanto segue.**

1. Definire i seguenti termini chiave: data warehouse, sistema operativo, sistema direzionale, data mart, data mart indipendente, data mart dipendente, data warehouse aziendale, data store operativo, @ctive warehouse, dati riconciliati, dati derivati, evento, dati transienti, dati persistenti, estratto statico, estratto incrementale, data scrubbing, modalità di refresh, modalità di aggiornamento, trasformazione dei dati, selezione, unione, aggregazione, schema a stella, grana, analisi del paniere di mercato, dimensione conforme, schema a snowflake, OLAP, ROLAP, MOLAP, data mining e visualizzazione dei dati.
2. Citare due motivi importanti per cui spesso si verifica un “gap informativo” tra le esigenze del manager delle informazioni e le informazioni generalmente disponibili.
3. Elencare due ragioni importanti per cui la maggior parte delle aziende moderne necessita di un data warehouse.
4. Elencare e descrivere brevemente i tre livelli dell’architettura di un data warehouse.
5. Elencare i quattro passaggi principali della riconciliazione dei dati.
6. Descrivere i due componenti principali di uno schema a stella.
7. Calcolare il numero di righe e la dimensione totale in byte di una tabella di fatti, dati alcuni presupposti sulle dimensioni della base di dati.
8. Progettare un data mart utilizzando schemi diversi per normalizzare e denormalizzare le dimensioni e per rappresentare la cronologia dei fatti e i valori mutevoli dell’attributo della dimensione.

## Introduzione

Chiunque concorda sul fatto che nel mondo degli affari è ormai diventato vitale avere a disposizione informazioni di alta qualità. Si considerino, per esempio, i commenti recenti di due esperti della gestione dei dati.

Nell’ambiente degli affari moderno, le informazioni sono diventate centrali. Il successo dipende da un loro uso precoce e decisivo. Una mancanza di informazioni è segno certo di fallimento. L’ambiente continuamente mutevole in cui operano le aziende richiede un accesso ai dati sempre più immediato. (Devlin, 1997)

Molte aziende sono alla ricerca di nuove tecnologie che le assistano nel diventare più redditizie e competitive. L'ottenimento di un vantaggio competitivo richiede che le aziende accelerino il loro processo decisionale così da poter rispondere rapidamente ai cambiamenti. Uno dei fattori chiave di questa accelerazione nella decisioni è la disponibilità di informazioni corrette, nel momento giusto, che siano facilmente accessibili. (Poe, 1996)

Alla luce di questa enfasi sulle informazioni e dei progressi più recenti nella tecnologia delle informazioni, ci si potrebbe aspettare che la maggior parte delle aziende abbia sviluppato sistemi per la visualizzazione delle informazioni ai manager e ad altri utenti. In realtà ciò è accaduto raramente. Nonostante abbiano grandi quantità di dati, e spesso molte basi di dati, sono poche le aziende che utilizzano più di una piccola frazione delle informazioni necessarie. I manager sono spesso frustrati dall'impossibilità di accedere o utilizzare i dati desiderati.

La realtà è che le aziende moderne sono sommerse dai dati ma mancano delle informazioni. Qual è la ragione di questa situazione? I motivi principali e correlati che hanno permesso la creazione di questo "gap informativo" sono due.

La prima ragione del gap informativo risiede nel modo frammentato con cui le aziende hanno sviluppato per anni i sistemi direzionali e le basi di dati a supporto. In questo testo si pone molta enfasi su un approccio di architettura dello sviluppo dei sistemi che sia attentamente pianificato e che produca un insieme di basi di dati integrate. Tuttavia, nella realtà molti vincoli di tempo e risorse fanno sì che la maggior parte delle imprese ricorra a un approccio "una cosa alla volta" per sviluppare isole di sistemi direzionali. Questo approccio produce inevitabilmente un insieme non coordinato e spesso incoerente di basi di dati, che si basano su un'ampia gamma di piattaforme hardware e software. In questo ambiente per i manager è estremamente difficile, se non impossibile, trovare e utilizzare informazioni accurate, che devono essere sintetizzate tra i vari sistemi di archiviazione.

La seconda ragione del gap informativo è che la maggior parte dei sistemi è sviluppata per supportare l'elaborazione operativa, con poca o nessuna attenzione agli strumenti di informazione o analitici necessari per prendere le decisioni. L'elaborazione operativa, chiamata anche *elaborazione delle transazioni*, acquisisce, memorizza e manipola i dati per supportare le operazioni quotidiane dell'azienda. L'elaborazione delle informazioni è l'analisi dei dati o di altre forme di informazioni che supportano le decisioni e può essere molto diversificata. La maggior parte dei sistemi che sono sviluppati internamente o che vengono acquistati da produttori esterni è concepita per supportare l'elaborazione operativa, riservando poca attenzione a quella delle informazioni.

Il compito di colmare il gap informativo spetta ai data warehouse, che consolidano e integrano le informazioni da più sorgenti interne ed esterne e le integrano in un formato significativo che aiuta a prendere le decisioni (Martín, 1997a). I data warehouse supportano gli analisti e i manager che devono prendere decisioni aziendali attraverso applicazioni come l'analisi delle tendenze, il marketing dei destinatari, l'analisi competitiva, la gestione dei rapporti con i clienti e così via. La tecnologia dei data warehouse si è evoluta per soddisfare queste esigenze senza alterare l'elaborazione operativa esistente.

Questo capitolo fornisce una panoramica sui data warehouse. L'argomento, molto vasto, richiederebbe un intero testo; in realtà la maggior parte dei testi sui data warehouse coprono un singolo aspetto, come la progettazione o l'amministrazione. In questo caso ci concentreremo sui due contesti più importanti per un testo sulla gestione delle basi di dati: l'architettura dei dati e la progettazione della base di dati. Vedremo innanzitutto come i data warehouse si relazionano con i sistemi operazionali esistenti, e descriveremo l'architettura a tre livelli, che è la più adatta in quasi tutti gli ambienti di data warehouse. Affronteremo poi il problema dell'estrazione dei dati dai sistemi operazionali esistenti e il loro caricamento nel data warehouse. Descriveremo gli elementi di progettazione delle basi di dati utilizzati

frequentemente nei data warehouse e vedremo infine come gli utenti interagiscono con i data warehouse attraverso l'elaborazione analitica on-line, il data mining e la visualizzazione dei dati.

## 12.1 Concetti di base dei data warehouse

Un data warehouse è una raccolta di dati orientata agli oggetti, integrata, legata al tempo e non aggiornabile, utilizzata nel processo di supporto alle decisioni sulla gestione e nelle informazioni aziendali (Inmon e Hackathorn, 1994). Di seguito viene definito il significato dei termini chiave dei data warehouse.

1. *Orientato agli oggetti.* Un data warehouse è organizzato attorno a degli oggetti chiave (o entità di alto livello) dell'impresa. Gli oggetti principali possono includere i clienti, i pazienti, gli studenti, i prodotti e i tempi.
2. *Integrato.* I dati raccolti in un data warehouse vengono definiti utilizzando convenzioni coerenti di denominazione, formati, strutture di codice e caratteristiche correlate tratti da più sistemi interni di registrazione e spesso da altre sorgenti esterne all'azienda.
3. *Legato al tempo.* I dati in un data warehouse contengono una dimensione temporale, così che possano essere utilizzati per studiare tendenze e cambiamenti.
4. *Non aggiornabile.* I dati in un data warehouse vengono caricati e aggiornati dai sistemi operazionali, ma non possono essere aggiornati dagli utenti finali.

Un data warehouse non è un semplice consolidamento di tutte le basi di dati operazionali di un'azienda. Poiché è centrato sulle informazioni aziendali, sui dati esterni e sui dati legati al tempo (e non solo allo stato corrente), è un tipo di base di dati unico.

Il *data warehousing* è il processo con il quale le aziende estraggono il significato e le decisioni sulle informazioni dai propri elementi informativi attraverso l'uso dei data warehouse (Barquin, 1996). Il data warehousing è un'iniziativa recente nella tecnologia delle informazioni. Avviata circa 15 anni fa, si è evoluta così rapidamente da diventare uno degli argomenti cruciali dei sistemi direzionali. Gli studi mostrano continuamente che oltre il 90 per cento delle grandi società ha un data warehouse o sta per avviarne uno. Uno studio del 1996 su 62 progetti di data warehouse ha mostrato un ritorno medio degli investimenti del 321 per cento, con un periodo medio di recupero di 2,73 anni.

### 12.1.1 Breve storia

Il data warehousing è emerso come risultato dei progressi nel campo dei sistemi direzionali avvenuti nei decenni, tra cui quelli elencati di seguito.

- Miglioramenti nella tecnologia delle basi di dati, e in particolare lo sviluppo del modello di dati relazionale e dei sistemi di gestione delle basi di dati relazionali (RDBMS).
- Progressi nell'hardware del computer, e in particolare l'emergere di spazi di memoria capienti e di architetture di computer parallele.
- L'affermarsi del lavoro di calcolo compiuto dall'utente, facilitato da interfacce e strumenti potenti e intuitivi.
- Progressi nei prodotti middleware che consentono la connessione delle basi di dati aziendali tra piattaforme eterogenee (Hackathorn, 1993).

La scoperta fondamentale che ha avviato lo sviluppo dei data warehouse è stata il riconoscere (e quindi il definire) le differenze principali tra i sistemi operazionali (o di elaborazione delle transazioni, chiamati a volte sistemi *di registrazione* perché il loro ruolo è quello di tenere una registrazione ufficiale e legale dell'azienda) e i sistemi direzionali. Nel 1998 Devlin e Murphy pubblicarono il primo articolo che descriveva l'architettura di un data warehouse proprio in base a questa distinzione. Nel 1992 Inmon pubblicò il primo libro che descriveva il data warehousing, diventando in seguito uno degli autori più prolifici sull'argomento.

## 12.1.2 La necessità del data warehousing

Attualmente sono due i fattori che stanno alla base della necessità del data warehousing nelle aziende più moderne.

1. Un'impresa necessita di informazioni integrate su scala aziendale e di alta qualità.
2. I responsabili dei sistemi informativi devono separare i sistemi direzionali da quelli operazionali per poter migliorare drasticamente le prestazioni della gestione dei dati aziendali.

### Necessità di una vista a livello aziendale

I dati nei sistemi operazionali sono solitamente frammentati e incoerenti e spesso sono anche distribuiti su un'ampia gamma di piattaforme hardware e software non compatibili. Per esempio, un file che contiene i dati dei clienti può essere collocato su un server UNIX che esegue un DBMS Oracle, mentre un altro può essere collocato su un mainframe IBM che esegue un DBMS DB2. È spesso necessario fornire una vista singola e aziendale delle informazioni anche per scopi decisionali.

Per capire meglio la difficoltà di ricavare un'unica vista aziendale, si osservi il semplice esempio mostrato nella Figura 12.1. La figura mostra tre tabelle di tre sistemi di registrazione distinti, ognuna contenente alcuni dati sugli studenti. Ci sono una tabella STUDENT\_DATA tratta del sistema di registrazione delle lezioni, una tabella STUDENT\_EMPLOYEE tratta dal sistema del personale e una tabella STUDENT\_HEALTH tratta da un sistema sanitario. Ogni tabella contiene alcuni dati univoci riguardanti gli studenti, ma sono memorizzati anche dati comuni (come i cognomi degli studenti) utilizzando formati diversi.

Si immagini di voler sviluppare un profilo per ogni studente, consolidando tutti i dati in un unico formato di file. Ci si potrebbe trovare a dover risolvere alcuni problemi.

- *Strutture di chiave incoerenti.* La chiave primaria delle prime due tabelle è una versione del numero della previdenza sociale dello studente, mentre la chiave primaria di STUDENT\_HEALTH è Student\_Name.
- *Sinonimi.* In STUDENT\_DATA la chiave primaria è Student\_No, mentre in STUDENT\_EMPLOYEE è Student\_ID (i sinonimi sono trattati nel Capitolo 7).
- *Campi a forma libera e campi strutturati.* In STUDENT\_HEALTH, Student\_Name è un campo singolo. In STUDENT\_DATA, Student\_Name (un attributo composto) è scomposto nelle sue parti significative Last\_Name, MI e First\_Name.
- *Valori di dati incoerenti.* Elaine Smith ha un numero di telefono in STUDENT\_DATA, ma un numero diverso in STUDENT\_HEALTH. È un errore o questa persona ha due numeri di telefono?
- *Dati mancanti.* Il valore di Insurance è mancante (o nullo) per Elaine Smith nella tabella STUDENT\_HEALTH. Come si può trovare questo valore?

**Figura 12.1**  
Esempi di dati eterogenei.

| STUDENT_DATA |           |    |            |           |        |     |
|--------------|-----------|----|------------|-----------|--------|-----|
| Student_No   | Last_Name | MI | First_Name | Telephone | Status | ... |
| 123-45-6789  | Enright   | T  | Mark       | 483-1967  | Soph   |     |
| 389-21-4062  | Smith     | R  | Elaine     | 283-4195  | Jr     |     |

| STUDENT_EMPLOYEE |                                   |      |       |     |
|------------------|-----------------------------------|------|-------|-----|
| Student_ID       | Address                           | Dept | Hours | ... |
| 123-45-6789      | 1218 Elk Drive, Phoenix, AZ 91304 | Soc  | 8     |     |
| 389-21-4062      | 134 Mesa Road, Tempe, AZ 90142    | Math | 10    |     |

| STUDENT_HEALTH  |           |            |             |     |
|-----------------|-----------|------------|-------------|-----|
| Name            | Telephone | Insurance  | ID          | ... |
| Mark T. Enright | 483-1967  | Blue Cross | 123-45-6789 |     |
| Elaine R. Smith | 555-7828  | ?          | 389-21-4062 |     |

Questo semplice esempio illustra la natura del problema dello sviluppo di un'unica vista aziendale, ma non spiega ancora la complessità dell'operazione. Uno scenario della vita reale avrebbe decine (se non centinaia) di file, e migliaia (o milioni!) di record. Torneremo sui temi della complessità e delle dimensioni successivamente nel capitolo.

Perché le aziende devono raggruppare i dati da vari sistemi di registrazione? La ragione più ovvia è per essere redditizie, più competitive o per crescere assumendo più valore per i clienti. Questo obiettivo può essere raggiunto aumentando la velocità e la flessibilità delle decisioni da prendere, migliorando i processi aziendali o acquisendo una comprensione più chiara del comportamento dei clienti. Per l'esempio degli studenti precedente, gli amministratori dell'università potrebbero voler capire se lo stato di salute o il numero di ore in cui gli studenti hanno lavorato al campus sono legati al rendimento scolastico, se frequentare determinate lezioni ha un rapporto con la salute degli studenti o se gli studenti con un profitto scarso comportano dei costi in più, a causa, per esempio, di un aumento delle spese per le cure o di altri costi. In generale, determinate tendenze nelle aziende incoraggiano la necessità dei data warehouse, e tra queste le seguenti.



- *Mancanza di un unico sistema di record.* Quasi nessuna azienda ha una sola base di dati. Strano, vero? Nel Capitolo 3 si sono viste le ragioni per scegliere una base di dati rispetto a un sistema di elaborazione dei file. A causa delle esigenze eterogenee di dati nelle impostazioni operazionali, delle fusioni e acquisizioni aziendali e delle dimensioni variabili di molte aziende, esistono più basi di dati operazionali.
- *I sistemi multipli non sono sincronizzati.* È difficile, se non impossibile, rendere coerenti basi di dati separate. Anche se i metadati sono controllati e resi omogenei da un unico amministratore dei dati, i valori dei dati per gli stessi attributi non coincideranno. Questo accade a causa dei diversi cicli di aggiornamento e dei punti separati in cui gli stessi dati vengono acquisiti in ciascun sistema. Per avere un'unica vista dell'azienda, è quindi necessario che i dati provenienti da sistemi diversi vengano periodicamente consolidati e sincronizzati in una base di dati supplementare. Vedremo che esistono due tipi di basi di dati consolidate, una chiamata "data store operazionale" e un'altra chiamata "data warehouse aziendale", entrambe esaminate con il data warehousing.
- *Le aziende vogliono analizzare le attività in modo equilibrato.* Molte aziende hanno implementato la tecnologia delle *balanced scorecard*, che mostrano simultaneamente i risultati nella soddisfazione finanziaria, umana e dei clienti, nella qualità dei prodotti e in altri termini. Per garantire che questa vista multidimensionale dell'azienda mostri risultati coerenti, è necessario un data warehouse. Quando durante la valutazione emergono delle domande, il software di analisi che lavora con il data warehouse può essere utilizzato per il "drill-down", lo "slice and dice" e la visualizzazione delle informazioni aziendali estratte.
- *Gestione dei rapporti con i clienti.* Le aziende in tutti i settori stanno realizzando che è importante avere un'immagine complessiva delle proprie interazioni con i clienti in tutti i punti di contatto. I diversi punti di contatto (per esempio, in una banca i punti di contatto includono il bancomat, gli sportelli automatici, i trasferimenti elettronici di fondi, la gestione del portafoglio degli investimenti e i prestiti) sono supportati da sistemi operazionali diversi. Senza un data warehouse, uno sportello automatico potrebbe non sapere come gestire una transazione automatica di deposito atipica molto grande che appare sul suo schermo. Oppure, un ufficio di servizi di CRM potrebbe voler assegnare una certa priorità alle chiamate in entrata a un help desk in base alla redditività del cliente e lo staff delle vendite potrebbe voler stabilire piani di contatto con i clienti in base alla loro redditività. Un'immagine complessiva dell'attività con un dato cliente richiede il consolidamento dei dati provenienti da vari sistemi operazionali.
- *Gestione dei rapporti con i fornitori.* La gestione della catena delle forniture diventa un elemento cruciale nella riduzione dei costi e nell'innalzamento della qualità dei prodotti per molte aziende. Le aziende vogliono creare rapporti strategici con i fornitori in base a un'immagine complessiva delle loro attività con essi, dalla fatturazione, al rispetto delle date di consegna, al controllo qualità, ai prezzi e all'assistenza. I dati sulle varie attività possono essere bloccati all'interno di sistemi operazionali separati (per esempio, crediti, consegne e ricevimenti, programmazione della produzione e manutenzione). I sistemi ERP hanno migliorato questa situazione portando molti di questi dati in un'unica base di dati. Tali sistemi tendono tuttavia a essere concepiti per ottimizzare l'elaborazione operazionale e non quella delle informazioni o analitica, come vedremo di seguito.

### **Necessità di separare i sistemi operazionali da quelli direzionali**

Un sistema operazionale è un sistema utilizzato per condurre un'azienda in tempo reale in base ai dati correnti. Esempi di sistemi operazionali sono l'elaborazione degli ordini di vendita, i sistemi di pre-

notazione e la registrazione dei pazienti. I sistemi operazionali devono elaborare grandi volumi di dati di transazioni di lettura/scrittura relativamente semplici, fornendo contemporaneamente una risposta veloce. I sistemi operazionali sono detti anche *sistemi di registrazione*.

I sistemi direzionali sono concepiti come supporto alle decisioni in base a un punto storico nel tempo e a dati di previsione. Sono concepiti anche per interrogazioni complesse o applicazioni di data mining. Esempi di sistemi direzionali sono le analisi delle tendenze di vendita, la segmentazione dei clienti e la pianificazione delle risorse umane.

Le differenze principali tra i due tipi di sistemi sono esposte nella Tabella 12.1. I due tipi di elaborazione hanno caratteristiche molto diverse a quasi tutti i livelli. In particolare, si noti che hanno comunità di utenti piuttosto diverse. I sistemi operazionali sono utilizzati da impiegati, amministratori, agenti di vendita e altri che devono elaborare le transazioni aziendali. I sistemi direzionali sono utilizzati da manager, dirigenti, analisti aziendali e, sempre più, da clienti che cercano informazioni sullo stato o che devono prendere decisioni.

**Tabella 12.1** Confronto tra sistemi operazionali e delle informazioni.

|                            |                                                                                    |                                                                                         |
|----------------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| Caratteristica             | Sistemi operazionali                                                               | Sistemi direzionali                                                                     |
| Scopo principale           | Condurre l'azienda in base alla situazione corrente                                | Supportare le decisioni prese a livello di management                                   |
| Tipo di dati               | Rappresentare lo stato corrente dell'azienda                                       | Un punto storico nel tempo (istantanea) e previsioni                                    |
| Utenti principali          | Impiegati, amministratori, agenti di vendita                                       | Manager, analisti aziendali, clienti                                                    |
| Ambito d'uso               | Aggiornamenti e interrogazioni circoscritti, pianificati e semplici                | Analisi e interrogazioni estese, ad hoc e complesse                                     |
| Obiettivo di progettazione | Volume delle prestazioni, disponibilità                                            | Accesso e uso flessibili                                                                |
| Volume                     | Numerosi e costanti aggiornamenti e interrogazioni su una o poche righe di tabella | Aggiornamenti e interrogazioni periodici in batch che richiedono molte o tutte le righe |

La ragione per separare i due tipi di sistemi si basa su tre fattori principali.

1. Un data warehouse centralizza i dati che sono distribuiti su sistemi operazionali disparati e li rende immediatamente disponibili per le applicazioni di supporto alle decisioni.
2. Un data warehouse opportunamente progettato aggiunge valore ai dati migliorandone qualità e omogeneità.
3. Un data warehouse separato elimina il contenimento delle risorse che si genera quando le applicazioni informative sono confuse con l’elaborazione operativa.

## 12.2 Architetture dei data warehouse

Le architetture utilizzate più spesso con i data warehouse sono tre: un’architettura generica a due livelli di base; un’architettura a tre livelli espansa utilizzata in ambienti più complessi; un’architettura di dati a tre livelli associata a un’architettura fisica a tre livelli.

### 12.2.1    Architettura generica a due livelli

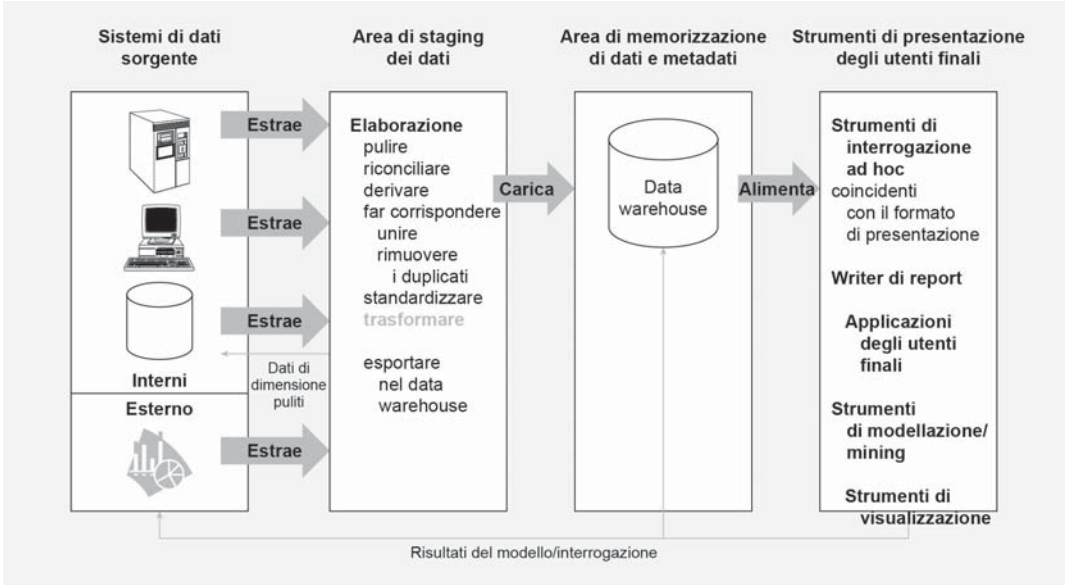
Un'architettura generica di un data warehouse è mostrata nella Figura 12.2. La costruzione di questa architettura richiede quattro passaggi fondamentali.

- 1. I dati vengono estratti da diversi file di sistema e basi di dati sorgente interni ed esterni. In una grande azienda ci possono essere decine o centinaia di tali file o basi di dati.
- 2. I dati tratti dai vari sistemi sorgente sono trasformati e integrati prima di essere caricati nel data warehouse. Le transazioni possono essere inviate ai sistemi sorgente per correggere gli errori scoperti nello staging dei dati.
- 3. Il data warehouse è una base di dati organizzata per il supporto alle decisioni. Contiene dati sia dettagliati sia riepilogativi.
- 4. Gli utenti accedono al data warehouse attraverso un'ampia gamma di linguaggi di interrogazione e strumenti analitici. I risultati (per esempio, le previsioni) possono essere rinviati al data warehouse e alle basi di dati operazionali.

Nei prossimi paragrafi discuteremo nel dettaglio gli importanti processi di estrazione, trasformazione e caricamento dei dati (ETL) dai sistemi sorgente nel data warehouse. Vedremo anche diversi strumenti di presentazione degli utenti finali.

L'estrazione e il caricamento si verificano su una base periodica, a volte quotidiana, settimanale o mensile. Il data warehouse, quindi, spesso non ha, o non ha bisogno di avere, dati correnti. Si ricordi che il data warehouse non supporta l'elaborazione delle transazioni operazionali, sebbene possa contenere dati transazionali (ma più spesso riepiloghi di transazioni e istantanee delle variabili dello stato, come bilanci di conto e livelli delle scorte). In gran parte delle applicazioni di data warehouse, gli utenti non cercano una risposta a una singola transazione quanto alle tendenze e ai pattern nello stato dell'azienda in un grande sottoinsieme di data warehouse. In un data warehouse sono conservati un minimo di cin-

**Figura 12.2**  
Architettura di data warehouse generica a due livelli.



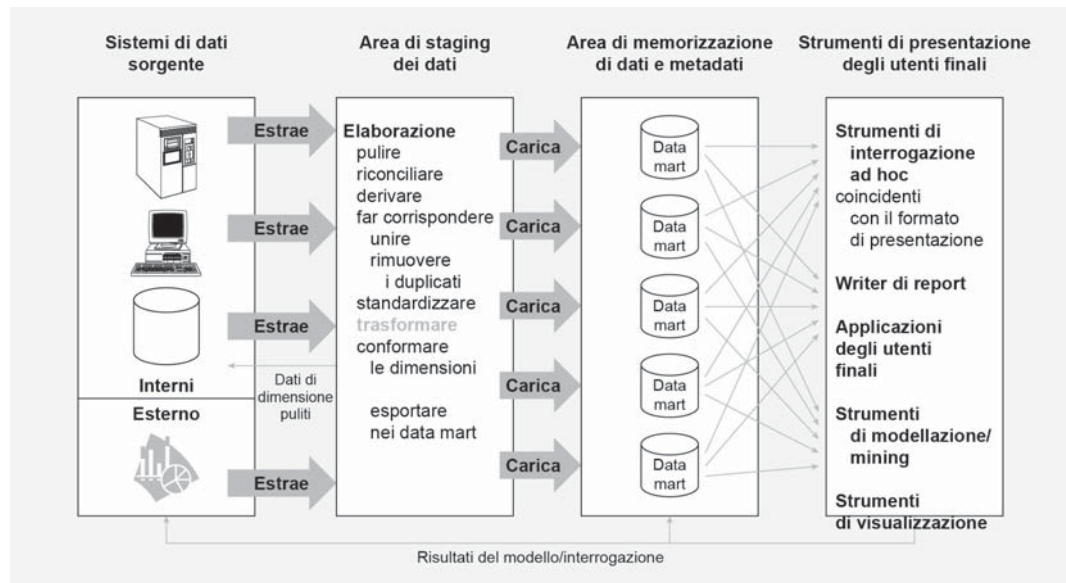
que trimestri fiscali, così che sia possibile discernere almeno le tendenze e i pattern annuali. I dati più vecchi possono essere eliminati o archiviati. Vedremo in seguito come un'architettura di data warehouse avanzata, chiamata *active data warehousing*, si basi su un presupposto diverso rispetto alla necessità di dati correnti.

## 12.2.2 Ambienti di data warehouse con data mart indipendenti

Contrariamente a molti dei principi discussi finora, alcune aziende non creano un data warehouse, bensì più data mart separati, ognuno basato sul data warehousing, e non sulle tecnologie di elaborazione delle transazioni o sulle basi di dati. Un **data mart** è un data warehouse il cui ambito è limitato. I suoi contenuti vengono ottenuti sia da processi ETL indipendenti, come mostrato nella Figura 12.3 per un **data mart indipendente**, sia dal data warehouse, come vedremo nei prossimi due paragrafi. Un data mart viene personalizzato per le applicazioni di decisione di un particolare gruppo di utenti finali. È quindi concepito per ottimizzare le prestazioni per usi ben definiti e prevedibili, a volte ridotti a una o due interrogazioni. Per esempio, un'azienda può avere un data mart per il marketing, uno finanziario, uno per la catena delle forniture e così via per supportare l'elaborazione analitica nota.

Forniremo un confronto tra le varie architetture di data warehouse in seguito, ma fin da ora si può notare una delle caratteristiche più ovvie della strategia dei data mart indipendenti, ovvero la complessità per gli utenti finali di accedere ai dati in data mart separati (evidenziata dalle linee incrociate che connettono tutti i data mart agli strumenti di presentazione degli utenti finali). Questa complessità non deriva solo dal fatto di dover accedere ai dati da basi di dati distinte, ma anche dalla presenza di sistemi di dati di nuova generazione non coerenti, i data mart appunto. Se un insieme di metadati è presente in tutti i data mart e se i dati sono resi coerenti tra di essi attraverso le attività nell'area dello staging (per esempio mediante le cosiddette "dimensioni conformi" nel riquadro dell'area dello staging nella Figura 12.3), allora la complessità per gli utenti si riduce. Nella Figura 12.3 non è però così ovvia la complessità

**Figura 12.3**  
Architettura di data warehouse con data mart indipendenti.



sità per i processi ETL, poiché per creare ciascun data mart indipendente sono necessari trasformazioni e caricamenti separati.

La ragione per cui le aziende adottano questa architettura è la convenienza. Può essere più facile, sia dal punto di vista organizzativo sia da quello delle politiche, avere piccoli data warehouse separati che non fare in modo che tutte le parti dell'azienda si accordino su un'unica vista in un data warehouse centrale. Alcune tecnologie di data warehousing pongono inoltre alcuni limiti alle dimensioni del data warehouse che possono supportare; è il cosiddetto problema della scalabilità, che vedremo successivamente. Se si aderisce a un determinato insieme di tecnologie di data warehousing prima di capire le esigenze del data warehousing, si rischia che sia la tecnologia, più che l'azienda, a dettare l'architettura del data warehouse. Nel prossimo paragrafo vedremo i pro e i contro dell'architettura dei data mart indipendenti rispetto alla principale architettura concorrente.

### 12.2.3 Data mart dipendenti e architetture di data store operazionali

L'architettura dei data mart indipendenti mostrata nella Figura 12.3 ha numerosi e importanti limiti (Meyer, 1997).

1. Processi ETL separati vengono sviluppati per ciascun data mart, cosa che può generare dati ridondanti e sforzi di elaborazione eccessivi.
2. I data mart possono non essere coerenti tra loro, e questo impedisce una vista aziendale chiara dei dati che riguardano aspetti fondamentali, quali i clienti, i fornitori e i prodotti.
3. Non è offerta la capacità del drill down dettagliato o nei fatti correlati in altri data mart o in un repository di dati condiviso; l'analisi è quindi limitata, o perlomeno molto difficile (per esempio, quando si effettuano delle unioni tra piattaforme separate per data mart diversi).

Il valore dei data mart indipendenti è stato oggetto di grandi discussioni. Kimball (1997) sostiene con forza lo sviluppo di data mart indipendenti come strategia fattibile per uno sviluppo in fasi dei sistemi di supporto alle decisioni. Armstrong (1997) e Inmon (1997 e 2000) evidenziano questi limiti e molti altri ancora. Attualmente la discussione sui data mart indipendenti verte su due aspetti.

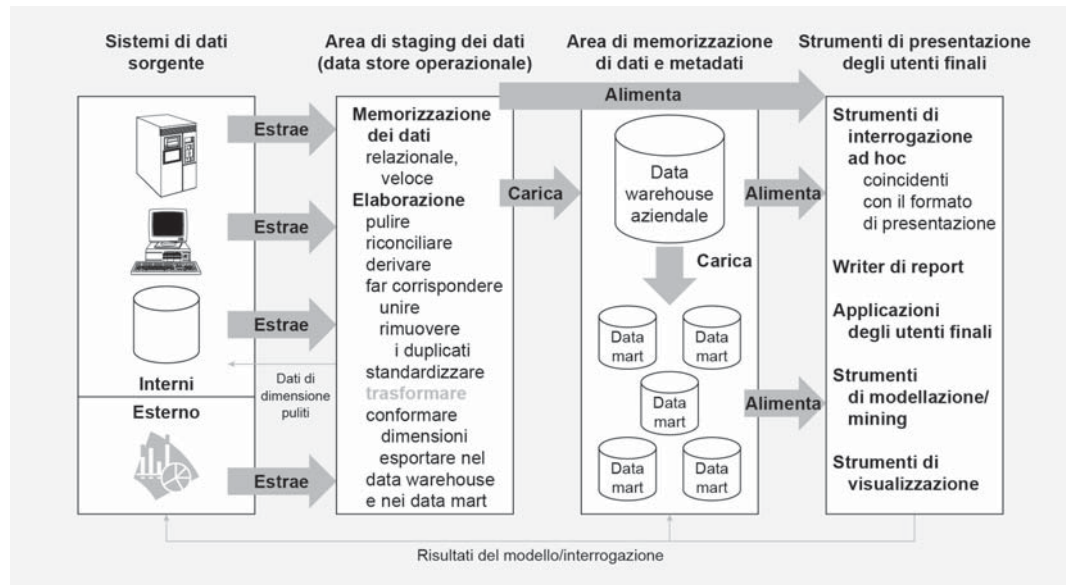
1. Un dibattito riguarda la natura dell'approccio in fasi all'implementazione dell'ambiente del data warehouse. In particolare, si discute se ogni data mart debba o meno evolversi come sottoinsieme di una definizione a livello di impresa dei dati di supporto alle decisioni.
2. L'altro dibattito riguarda l'architettura più adatta per l'elaborazione analitica. In particolare, si concentra sull'estensione alla quale ciascun data mart dovrebbe essere normalizzato.

I due aspetti vengono presi in considerazione in tutto il resto del capitolo; al termine forniremo un esercizio che permetterà di approfondirli.

Uno dei primi e più diffusi approcci per risolvere i limiti dei data mart indipendenti è quello che utilizza data mart dipendenti e un'architettura di data store operazionali (Figura 12.4). Il primo e il secondo limite vengono risolti caricando **data mart dipendenti** da un **data warehouse aziendale (EDW)**, che è un data warehouse centrale e integrato che fa da punto di controllo e da unica “versione della verità” per gli utenti finali rispetto alle applicazioni di supporto alle decisioni. I data mart dipendenti hanno ancora lo scopo di fornire un ambiente semplificato e ad alte prestazioni adattato alle esigenze decisionali dei gruppi di utenti. Un gruppo di utenti può accedere al suo data mart, e quando necessita di altri dati, può accedere all'EDW. La ridondanza tra data mart indipendenti è pianificata e i dati ridondanti sono coerenti.

**Figura 12.4**

Data mart dipendenti e architettura dei data store operazionali.



ti perché ogni data mart è caricato in modo sincronizzato da una sorgente di dati comune. L'integrazione dei dati è responsabilità dello staff IT che gestisce il data warehouse aziendale, e non degli utenti finali che devono integrare i dati tra data mart indipendenti per ogni interrogazione o applicazione. L'architettura operazionale dei data mart dipendenti è spesso chiamata approccio *hub and spoke*, dove l'EDW è l'hub e i sistemi dei dati sorgente e i data mart sono agli estremi dei spoke di input e output.

Il terzo limite viene risolto fornendo una sorgente integrata di tutti i dati operazionali in un data store operazionale. Un data store operazionale (ODS) è una base di dati integrata, orientata agli oggetti, aggiornabile, corrente e dettagliata concepita per supportare gli utenti operativi quando elaborano a supporto delle decisioni (Imhoff, 1998; Inmon, 1998). Un ODS è solitamente una base di dati relazionale normalizzata come quelle dei sistemi di record, ma adattata per le applicazioni di supporto alle decisioni. Per esempio, gli indici e altri elementi di una base di dati relazionale sono adattati per le interrogazioni che recuperano grandi gruppi di dati invece che per l'elaborazione delle transazioni o per l'interrogazione di singoli record direttamente correlati (per esempio, gli ordini dei clienti). Un ODS offre un'immagine corrente di tutte le attività a livello dell'impresa. Poiché contiene dati volatili e correnti, la stessa interrogazione su un ODS restituirà molto probabilmente risultati diversi in momenti diversi. Un ODS solitamente non contiene una storia, mentre un EDW contiene la storia delle istantanee dello stato dell'azienda. Un ODS può essere alimentato dalla base di dati di un'applicazione ERP, ma poiché la maggior parte delle aziende non ha una sola base di dati ERP e non esegue tutte le operazioni al suo interno, un ODS è solitamente diverso da una base di dati ERP. Inoltre un ODS non è concepito per l'elaborazione di grandi volumi di transazioni, bensì per il supporto alle decisioni, e assume il ruolo dell'area dello staging per caricare i dati nell'EDW. L'ODS può ricevere i dati dai sistemi transazionali immediatamente o con un po' di ritardo, cosa che è pratica e accettabile per le esigenze di supporto alle decisioni.

L'architettura dei data store operazionali e dei data mart dipendenti è chiamata anche *Corporate Information Factory*, o CIF (Imhoff, 1999), ed è considerata una vista comprensiva dei dati dell'azienda a supporto di tutte le esigenze degli utenti.

I più autorevoli esperti su questo argomento sostengono approcci diversi al data warehousing. Quelli che prediligono l'approccio dei data mart indipendenti, ne evidenziano due vantaggi significativi.

1. Questo approccio permette che il concetto di data warehouse sia verificato lavorando su una serie di piccoli progetti.
2. I tempi lunghi necessari ad avere alcuni benefici dai data warehouse sono ridotti, poiché l'azienda non viene ritardata per attendere che tutti i dati siano centralizzati.

I sostenitori dell'approccio della CIF sollevano molte riserve sull'approccio dei data mart indipendenti (Inmon, 1999; Armstrong, 2000).

1. Ridondanza tra i data mart.
2. La necessità che siano gli utenti stessi a fornire la prospettiva integrata dell'azienda tra i data mart.
3. Il lavoro supplementare per molti processi ETL.
4. L'uso probabile delle tecnologie dei data warehouse per ogni data mart che non si scala in un EDW a causa dei limiti tecnici o perché nei vari data mart sono utilizzate tecnologie diverse.
5. I costi elevati per mantenere i data mart sincronizzati.

Armstrong (2000) e altri vanno oltre, fino ad affermare che i vantaggi proposti dai sostenitori dei data mart indipendenti sono in realtà i vantaggi legati all'adozione di un approccio di sviluppo in fasi dei data warehouse. Questo tipo di approccio può essere realizzato anche nel framework della CIF, ed è facilitato dall'architettura di data warehouse finale che presentiamo nel prossimo paragrafo.

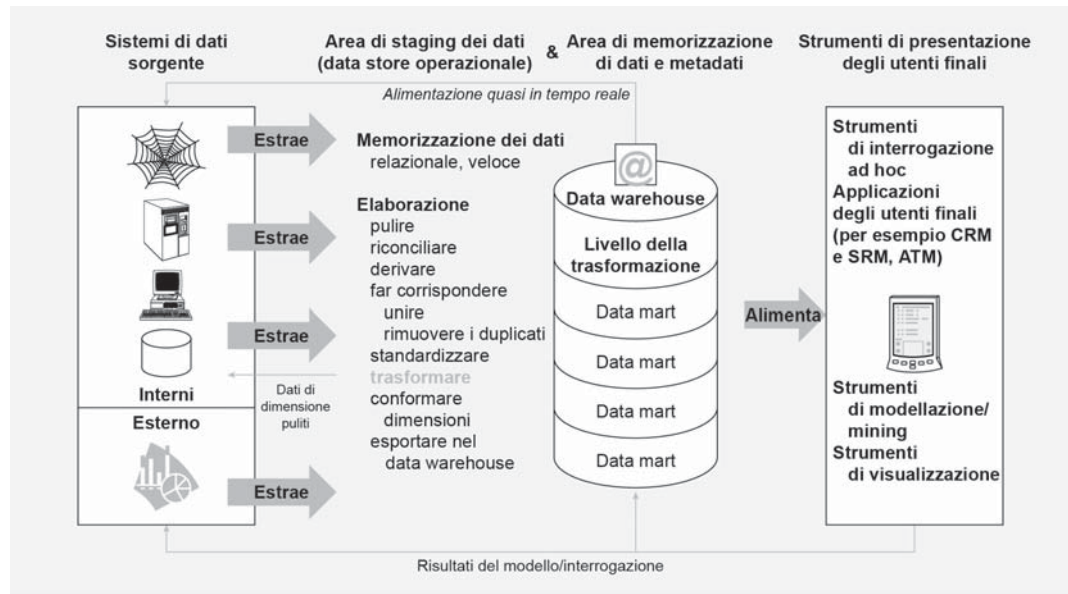
### 12.2.4 Architettura @ctive warehouse dei data mart logici

L'architettura @ctive warehouse dei data mart logici è ideale per data warehouse di dimensioni modeste o quando si utilizza una tecnologia di data warehouse a prestazioni elevate, come il sistema NCR Teradata. Come si può vedere dalla Figura 12.5, questa architettura ha alcune caratteristiche uniche.

1. I data mart logici non sono basi di dati fisicamente separate, bensì viste relazionali diverse di un data warehouse relazionale leggermente denormalizzato (si veda il Capitolo 9 per un ripasso del concetto di vista).
2. I dati vengono spostati nel data warehouse invece che in un'area di staging separata per sfruttare la potenza di calcolo ad altre prestazioni della tecnologia per eseguire i passaggi di pulizia e trasformazione.
3. I nuovi data mart possono essere creati rapidamente perché non è necessario creare alcuna base di dati fisica o tecnologia di base di dati né acquisire o caricare routine scritte.
4. I data mart sono sempre aggiornati, perché i dati in una vista vengono creati quando si fa riferimento a quest'ultima; le viste possono essere materializzate se un utente ha una serie di interrogazioni o analisi che devono utilizzare la stessa istanza del data mart.

Siano essi logici o fisici, i data mart o i data warehouse giocano ruoli diversi in un ambiente di data warehouse; questi ruoli sono riepilogati nella Tabella 12.2. Sebbene limitato come ambito, un data mart può non essere piccolo; la tecnologia scalabile diventa quindi cruciale. Per quanto meno complessi da

**Figura 12.5**  
Architettura @ctive  
warehouse dei data  
mart logici.



costruire a causa del loro ambito ridotto, i data mart sollevano sempre problemi quando si tratta dei processi ETL. Inoltre, un carico e un costo significativi sono posti sugli utenti quando loro stessi devono integrare i dati tra data mart fisici separati (sempre che ciò sia possibile). Così come i data mart vengono aggiunti, un data warehouse può essere costruito in più fasi; il modo più facile perché questo accada è seguire un'architettura @ctive warehouse dei data mart logici.

L'aspetto di @ctive data warehouse dell'architettura mostrata nella Figura 12.5 indica che i sistemi di dati sorgente e quelli di scambio del data warehouse avvengono quasi in tempo reale, poiché occorre una risposta rapida a un'immagine corrente e articolata dell'azienda. Per esempio, un professionista di un help desk che risponde alle domande e registra i problemi avrà un'immagine complessiva dei contatti di vendita più recenti, delle transazioni di fatturazione e pagamento, delle attività di manutenzione e degli ordini. Con queste informazioni, il sistema che supporta l'help desk può generare automaticamente uno script per il professionista che deve vendere un contratto di assistenza, un prodotto aggiornato o un altro prodotto acquistato da clienti che hanno un profilo analogo. Un evento cruciale, come l'immissione di un nuovo ordine di prodotto, può essere preso immediatamente in considerazione così che l'azienda sappia della relazione con il suo cliente almeno tanto quanto ne sa il cliente. Il principio è che ogni evento, per esempio con un cliente, è un'opportunità potenziale per stabilire una comunicazione personalizzata che si basa sulla decisione strategica di come rispondere a un cliente che ha un determinato profilo. Alcune applicazioni vantaggiose dell'@ctive data warehousing includono le seguenti.

- Consegna immediata delle merci in base a livelli delle scorte aggiornati.
- Commercio elettronico, dove, per esempio, un carrello della spesa abbandonato può attivare un messaggio promozionale di posta elettronica prima che l'utente chiuda la sessione.
- Individuazione delle frodi nelle transazioni con carte di credito, dove pattern di transazione insolite possono avvertire un commesso o una routine di carrello della spesa on-line di prendere precauzioni supplementari.



**Tabella 12.2** Data warehouse e data mart.

| Data warehouse                                      | Data mart                                              |
|-----------------------------------------------------|--------------------------------------------------------|
| <i>Ambito</i>                                       | <i>Ambito</i>                                          |
| ■ Indipendente dall'applicazione                    | ■ Applicazione DSS specifica                           |
| ■ Centralizzato, eventualmente a livello di impresa | ■ Decentralizzato dall'area degli utenti               |
| ■ Pianificato                                       | ■ Organico, eventualmente non pianificato              |
| <i>Dati</i>                                         | <i>Dati</i>                                            |
| ■ Storici, dettagliati e riepilogati                | ■ Parzialmente storici, dettagliati e riepilogati      |
| ■ Leggermente denormalizzati                        | ■ Fortemente denormalizzati                            |
| <i>Oggetti</i>                                      | <i>Oggetti</i>                                         |
| ■ Oggetti multipli                                  | ■ Un oggetto centrale di interesse per gli utenti      |
| <i>Sorgenti</i>                                     | <i>Sorgenti</i>                                        |
| ■ Molte sorgenti interne ed esterne                 | ■ Poche sorgenti interne ed esterne                    |
| <i>Altre caratteristiche</i>                        | <i>Altre caratteristiche</i>                           |
| ■ Flessibile                                        | ■ Restrittivo                                          |
| ■ Orientato ai dati                                 | ■ Orientato al progetto                                |
| ■ Vita lunga                                        | ■ Vita breve                                           |
| ■ Grande                                            | ■ Parte piccolo, diventa grande                        |
| ■ Unica struttura complessa                         | ■ Più strutture semi-complesse, nell'insieme complesse |

Adattato da Strange (1997)

Tali applicazioni sono caratterizzate da un accesso on-line degli utenti, 7 giorni su 7, 24 ore al giorno, per 365 giorni all'anno. Gli utenti delle architetture di data warehouse possono essere dipendenti, clienti o partner d'affari.

Con computer ad alte prestazioni e tecnologie di data warehouse, può rendersi necessario un ODS separato dal data warehouse aziendale. Quando l'ODS e l'EDW sono uniti, è più facile che i clienti effettuino il drill-down e il drill-up quando lavorano con una serie di domande ad hoc nella quale una domanda porta a un'altra. Si tratta inoltre di un'architettura più semplice, perché un livello dell'architettura dei data mart e dei data store operazionali è stata eliminata.

### 12.2.5    Architettura dei dati a tre livelli

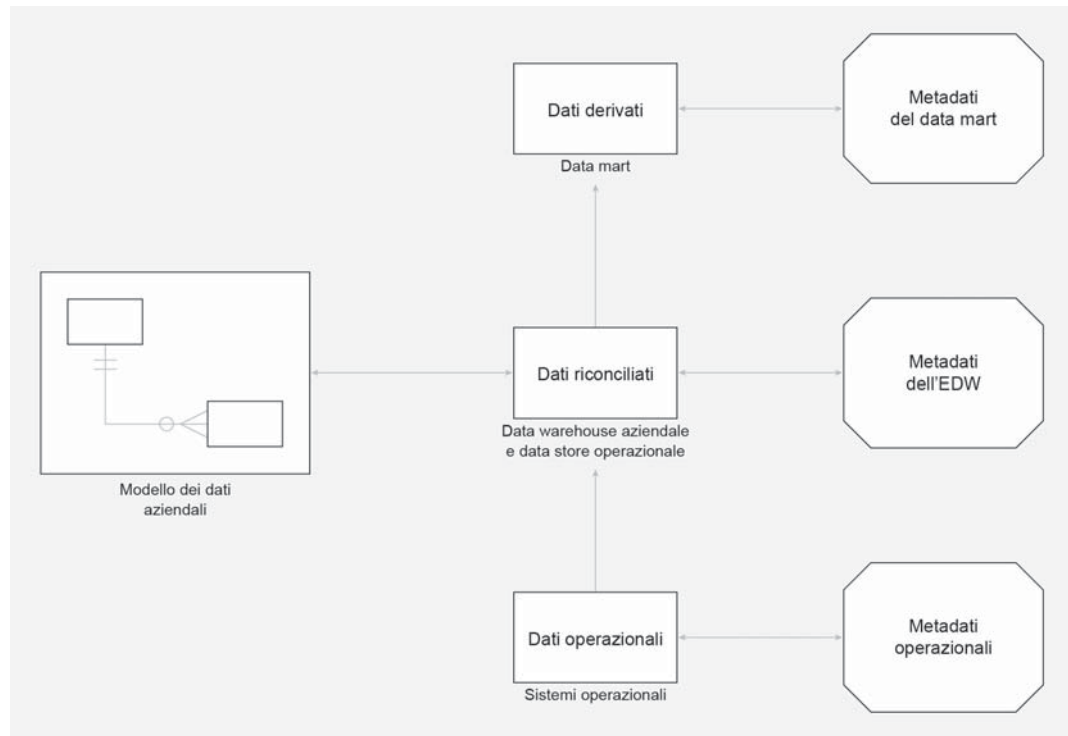
La Figura 12.6 mostra un'architettura di dati a tre livelli per un data warehouse.

1. I dati operazionali sono memorizzati nei vari sistemi operazionali in tutta l'azienda (e a volte in sistemi esterni).
2. I dati riconciliati sono il tipo di dati memorizzati nel data warehouse aziendale e nel data store operazionale.
3. I dati derivati sono il tipo di dati memorizzati in ciascun data mart.

I **dati riconciliati** sono i dati dettagliati e correnti che devono rappresentare l'unica e autorevole sorgente per tutte le applicazioni di supporto alle decisioni. I **dati derivati** sono i dati che sono stati selezionati, formattati e aggregati per le applicazioni di supporto alle decisioni degli utenti finali.

Discuteremo questi due tipi di dati nei paragrafi successivi. Nella Figura 12.6 sono inoltre mostrati due componenti che giocano un ruolo fondamentale nell'architettura dei dati: il modello dei dati aziendali e i metadati.

**Figura 12.6**  
Architettura a tre livelli.



### **Ruolo del modello dei dati aziendali**

Nella Figura 12.6 è mostrato il livello dei dati riconciliati collegato al modello dei dati aziendali. Nel Capitolo 4 è stato spiegato che il modello dei dati aziendali presenta un'immagine complessiva che definisce i dati richiesti da un'azienda. Se il livello dei dati riconciliati deve essere unico, come sorgente autorevole di tutti i dati necessari per il supporto alle decisioni, deve essere conforme al progetto specificato nel modello dei dati aziendale. Questo controlla quindi l'evoluzione in fasi del data warehouse. Solitamente il modello evolve man mano che si affrontano nuovi problemi e applicazioni di decisione. Sviluppare un modello dei dati aziendale in un unico passo è un'operazione molto lunga ed è probabile che le necessità dinamiche per le decisioni cambino prima ancora che il warehouse venga costruito.

### **Ruolo dei metadati**

La Figura 12.6 mostra un livello di metadati collegato a ognuno dei tre livelli di dati. Nel Capitolo 3 è stato spiegato che i metadati sono dati che descrivono le proprietà o caratteristiche dei dati. A seguire si riporta una breve descrizione dei tre tipi di metadati mostrati nella Figura 12.6.

1. I *metadati operazionali* descrivono i dati nei vari sistemi operazionali (così come i dati esterni) che alimentano il data warehouse aziendale. I metadati operazionali si presentano solitamente in vari formati diversi e, sfortunatamente, sono spesso di scarsa qualità.
2. I *metadati dell'EDW* sono tratti dal modello dei dati aziendali (o perlomeno sono coerenti con esso) e descrivono il livello dei dati riconciliati così come le regole per la trasformazione dei dati operazionali in dati riconciliati.

- 3. I *metadati del data mart* descrivono il livello dei dati derivati e le regole per la trasformazione dei dati riconciliati in dati derivati.

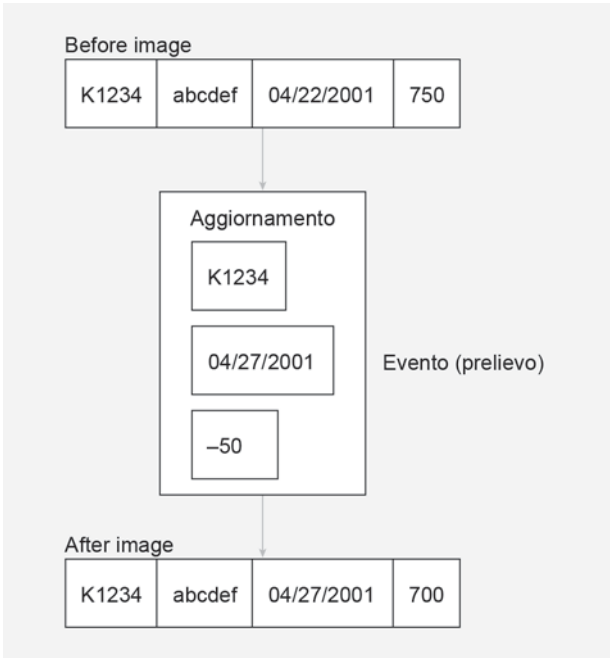
## 12.3 Carattersitiche dei dati dei data warehouse

Per poter capire meglio i dati presenti in ciascuno di questi livelli, occorre conoscere alcune caratteristiche dei dati che vengono memorizzati nelle basi di dati dei data warehouse.

### 12.3.1 Dati di stato e dati di evento

La differenza tra i dati di stato e di evento è mostrata nella Figura 12.7. La figura illustra una tipica immissione di registro registrata da un DBMS quando elabora una transazione aziendale per un'applicazione bancaria. Questa immissione contiene sia dati di stato sia dati di evento. La *before image* e l'*after image* rappresentano lo stato del conto bancario prima e dopo un prelievo. I dati che rappresentano il prelievo (o evento di aggiornamento) sono al centro della figura.

**Figura 12.7**  
Esempio di immissione di registro di un DBMS.



Le transazioni sono attività aziendali che fanno sì che uno o più eventi aziendali si verifichino a livello della base di dati (Devlin, 1997). Un evento è un'azione nella base di dati (creazione, aggiornamento o eliminazione) che risulta da una transazione. Si noti che una transazione può portare a uno o più eventi. La transazione del prelievo nella Figura 12.7 porta a un singolo evento, che è la riduzione del conto da 750 a 700. Il trasferimento di denaro da un conto a un altro produrrebbe due eventi: un evento

di prelievo e uno di deposito. A volte anche le non-transazioni, come un carrello della spesa abbandonato, un segnale di occupato, la caduta di una connessione di rete, o un articolo collocato nel carrello e poi rimosso prima del conto finale, possono essere eventi importanti.

Sia i dati di stato sia quelli di evento possono essere memorizzati in una base di dati. Nella pratica, tuttavia, gran parte dei dati memorizzati nelle basi di dati, data warehouse compresi, sono dati di stato. Un data warehouse conterrà probabilmente una storia delle istantanee dei dati di stato o un riepilogo (per esempio, un totale orario) dei dati di transazione o di evento. I dati di evento, che rappresentano transazioni, possono essere conservati per un certo periodo di tempo, dopodiché vengono eliminati o archiviati per risparmiare spazio di memoria. Entrambi i tipi di evento sono solitamente memorizzati in registri della base di dati (come rappresentato nella Figura 12.7) per scopi di backup e di ripristino. Come si vedrà successivamente, il registro della base di dati gioca un ruolo importante nell'alimentazione del data warehouse.

### 12.3.2 Dati transienti e dati persistenti

Nei data warehouse è spesso necessario mantenere una registrazione di quando gli eventi si sono verificati nel passato. Può servire, per esempio, per confrontare le vendite o i livelli delle scorte in una data specifica o durante un determinato periodo con le vendite dell'anno precedente nella stessa data o durante lo stesso periodo.

La maggior parte dei sistemi operazionali si basa sull'uso di dati transienti. I dati transienti sono dati in cui le modifiche ai record esistenti sono scritte sopra ai record precedenti, distruggendo quindi il contenuto esistente. La Figura 12.7 aiuta a individuare i dati transienti. Se l'after image sovrascrive la before image, questa (che contiene il bilancio precedente) va perduta. Tuttavia, poiché si tratta di un registro di base di dati, solitamente entrambe le immagini vengono conservate.

I dati persistenti sono dati che non vengono mai modificati né eliminati fisicamente una volta che sono stati aggiunti allo store. La before image e l'after image nella Figura 12.7 rappresentano i dati persistenti. Si noti che ogni record contiene una marcatura temporale che indica la data (e l'ora, se necessario) in cui si è verificato l'evento di aggiornamento più recente (le marcature temporali sono trattate nel Capitolo 5).

### 12.3.3 Un esempio di dati transienti e persistenti

Un esempio dettagliato del confronto tra dati transienti e persistenti è mostrato nelle Figure 12.8 e 12.9.

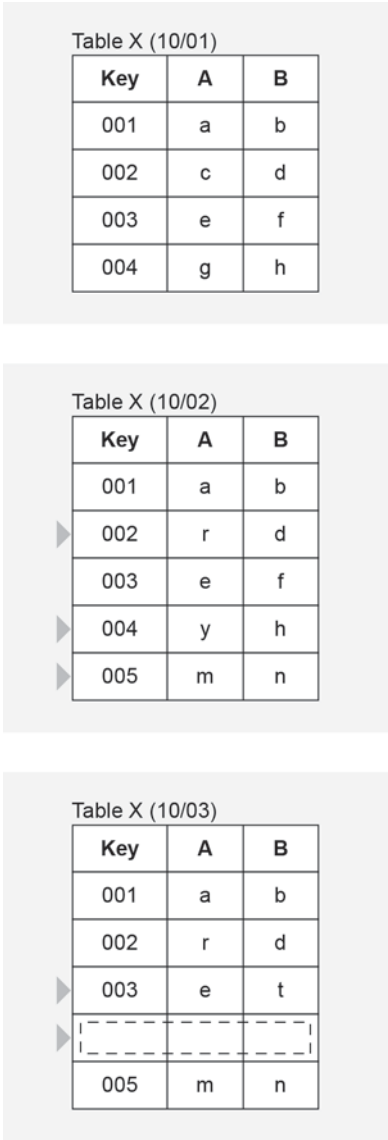
#### Dati transienti

La Figura 12.8 mostra una relazione (Table X) che contiene inizialmente quattro record. La tabella ha tre attributi: una chiave primaria e due attributi non di chiave A e B. I valori per ciascuno di questi attributi sulla data 10/01 sono mostrati nella figura. Per esempio, per il record 001, il valore dell'attributo A in questa data è *a*.

In data 10/02 vengono apportate tre modifiche alla tabella (le modifiche alle righe sono indicate dalle frecce a sinistra della tabella). Il record 002 viene aggiornato, così che il valore di A venga modificato da *c* a *r*. Anche il record 004 viene aggiornato, così che il valore di A venga modificato da *g* a *y*. Infine, viene inserito un nuovo record (con la chiave 005) nella tabella.

Si noti che quando i record 002 e 004 vengono aggiornati, i nuovi record sostituiscono quelli precedenti. I valori precedenti vanno quindi perduti e non ne esiste una registrazione storica. Questa è una caratteristica dei dati transienti.

**Figura 12.8**  
Dati operazionali transienti.



In data 10/03 vengono apportate più modifiche ai record (per semplificare l’esempio, si immagini che possa essere apportata una sola modifica a un dato record in una data specifica). Il record 003 è aggiornato e il record 004 eliminato. Si noti che niente indica che il record 004 sia mai stato memorizzato nella base di dati. Il modo in cui i dati sono elaborati nella Figura 12.7 è una caratteristica dei dati transienti tipici dei sistemi operazionali.

**Dati persistenti**

Un tipico obiettivo di un data warehouse è quello di mantenere una registrazione storica degli eventi chiave o di creare una serie temporale per determinate variabili, come le vendite. Questo richiede spesso

**Figura 12.9**  
Dati di warehouse persistenti.

Table X (10/01)

| Key | Date  | A | B | Action |
|-----|-------|---|---|--------|
| 001 | 10/01 | a | b | C      |
| 002 | 10/01 | c | d | C      |
| 003 | 10/01 | e | f | C      |
| 004 | 10/01 | g | h | C      |

Table X (10/02)

| Key   | Date  | A | B | Action |
|-------|-------|---|---|--------|
| 001   | 10/01 | a | b | C      |
| 002   | 10/01 | c | d | C      |
| ▶ 002 | 10/02 | r | d | U      |
| 003   | 10/01 | e | f | C      |
| 004   | 10/01 | g | h | C      |
| ▶ 004 | 10/02 | y | h | U      |
| ▶ 005 | 10/02 | m | n | C      |

Table X (10/03)

| Key   | Date  | A | B | Action |
|-------|-------|---|---|--------|
| 001   | 10/01 | a | b | C      |
| 002   | 10/01 | c | d | C      |
| 002   | 10/02 | r | d | U      |
| 003   | 10/01 | e | f | C      |
| ▶ 003 | 10/03 | e | t | U      |
| 004   | 10/01 | g | h | C      |
| 004   | 10/02 | y | h | U      |
| ▶ 004 | 10/03 | y | h | D      |
| 005   | 10/02 | m | n | C      |

la memorizzazione di dati persistenti anziché transienti. La Figura 12.9 mostra la tabella utilizzata nella Figura 12.8 modificata per rappresentare i dati persistenti. Nella Figura 12.9 sono state apportate le seguenti modifiche.

1. Sono state aggiunte due nuove colonne alla Table X.
  - a. La colonna Date è una marcatura temporale che registra la data più recente in cui una riga è stata modificata.
  - b. La colonna Action è utilizzata per registrare il tipo di modifica. I valori possibili per questo attributo sono C (crea), U (aggiorna) e D (elimina).
2. Una volta che un record è stato memorizzato nella tabella, non viene più modificato. Quando questo accade, la before image e l'after image vengono memorizzate nella tabella. Sebbene un record possa essere eliminato logicamente, una versione storica del record eliminato viene conservata nella base di dati per il periodo cronologico necessario (almeno cinque trimestri) ad analizzare le tendenze.

Esaminiamo ora lo stesso insieme di azioni che si sono verificate nella Figura 12.8. Si presuma che tutti e quattro i record siano stati creati in data 10/01, come mostrato nella prima tabella.

Nella seconda tabella (del 10/02), sono state aggiornate le righe 002 e 004. La tabella contiene ora sia la versione vecchia (del 10/01) sia quella nuova (del 10/02) di queste righe. La tabella contiene anche la nuova riga (005) che è stata creata in data 10/02.

La terza tabella (del 10/03) mostra l'aggiornamento alla riga 003, con la versione sia vecchia sia nuova. Nella tabella è stata eliminata anche la riga 004. La tabella contiene ora tre versioni della riga 004: quella originale (del 10/01), quella aggiornata (del 10/02) e quella eliminata (del 10/03). La lettera D nell'ultima riga del record 004 indica che questo record è stato eliminato logicamente, così che non sia più disponibile per gli utenti o per le loro applicazioni.

Se si osserva la Figura 12.9, si può vedere perché i data warehouse tendono a crescere rapidamente. La memorizzazione di dati persistenti può imporre la necessità di un'area di memoria molto grande. Un corollario è che gli utenti devono scegliere con molta attenzione i dati chiave che richiedono questa forma di elaborazione.

## 12.4 Il livello dei dati riconciliati

Come indicato nella Figura 12.6, si utilizza il termine *dati riconciliati* per fare riferimento al livello dei dati associato al data store operativo e al data warehouse aziendale. Questo è il termine utilizzato in un documento di IBM del 1993 che descriveva le architetture di data warehouse. Sebbene non sia ampiamente utilizzato, descrive con precisione la natura dei dati che dovrebbero apparire nel data warehouse aziendale e il modo in cui vengono derivati. Più comunemente, i dati riconciliati sono definiti come il risultato del processo ETL. Un EDW o un ODS sono solitamente basi di dati relazionali normalizzate, perché necessitano della flessibilità per supportare un'ampia gamma di esigenze di supporto alle decisioni.

### 12.4.1 Caratteristiche dei dati dopo le operazioni ETL

Le operazioni di estrazione, trasformazione e caricamento (ETL) sono pensate per fornire un'unica sorgente autorevole per i dati che supportano le decisioni.

Idealmente, questo livello di dati è dettagliato, storico, normalizzato, comprensivo, corrente e di qualità.

1. *Dettagliato*. I dati sono dettagliati (e non riepilogativi), e forniscono la massima flessibilità a varie comunità di utenti per strutturare i dati in modo che soddisfino al meglio le loro esigenze.
2. *Storico*. I dati sono persistenti (o di un punto nel tempo), così da offrire una prospettiva cronologica.
3. *Normalizzato*. I dati sono totalmente normalizzati, ossia nella terza forma normale o in forme superiori (la normalizzazione è trattata nel Capitolo 7). I dati normalizzati offrono maggiore integrità e flessibilità rispetto a quelli denormalizzati. La denormalizzazione non è necessaria per migliorare le prestazioni, poiché ai dati riconciliati si accede solitamente con frequenza periodica mediante processi in batch. Vedremo tuttavia che alcune delle strutture dei dati nei data warehouse più diffuse sono denormalizzate.
4. *Comprensivo*. I dati riconciliati riflettono una prospettiva a livello aziendale, la cui struttura è conforme al modello dei dati aziendali.
5. *Corrente*. A eccezione dell'@ctive warehousing, i dati non devono essere in tempo reale, ma devono essere comunque sufficientemente aggiornati così che si possano prendere le decisioni in tempo e al momento giusto.
6. *Qualità controllata*. I dati riconciliati devono essere di indubbia qualità e integrità, poiché sono riepilogati nei data mart e utilizzati per il supporto alle decisioni.

Si noti che queste caratteristiche dei dati riconciliati sono piuttosto diverse da quelle dei dati operazionali da cui derivano. I dati operazionali sono solitamente dettagliati, ma differiscono notevolmente nelle altre quattro dimensioni sopra descritte.

1. I dati operazionali sono transienti anziché storici.
2. Non sono normalizzati. A seconda delle loro radici, i dati operazionali potrebbero non essere mai stati normalizzati o potrebbero essere stati denormalizzati per ragioni di prestazioni.
3. Invece di essere comprensivi, i dati operazionali hanno solitamente un ambito circoscritto a una determinata applicazione.
4. I dati operazionali sono spesso di qualità scadente, con molti tipi di incoerenze ed errori.

Il processo di riconciliazione dei dati è responsabile della trasformazione dei dati operazionali in dati riconciliati. A causa delle differenze tra i due tipi di dati, la riconciliazione è chiaramente la parte più difficile e tecnologicamente impegnativa della costruzione di un data warehouse. Fortunatamente sono disponibili numerosi prodotti software sofisticati che assistono in questa attività.

## **12.4.2 Il processo ETL**

La riconciliazione dei dati avviene in due passaggi durante il processo di riempimento del data warehouse aziendale.

1. Un caricamento iniziale, quando l'EDW viene creato per la prima volta.
2. Aggiornamenti successivi (eseguiti solitamente con frequenza periodica) per mantenere l'EDW aggiornato e/o per espanderlo.



La riconciliazione dei dati può essere visualizzata come un processo, mostrato nella Figura 12.10, costituito da quattro passaggi: acquisizione, scrub, trasformazione e caricamento e indicizzazione. Per esempio, l'acquisizione dei dati e lo scrub oppure lo scrub e la trasformazione possono essere combinati in un unico processo. Discuteremo i singoli passaggi di seguito, con un'attenzione particolare alla trasformazione.

**Figura 12.10**

I passaggi della riconciliazione.



### Estrazione

L'acquisizione di dati importanti dai file sorgente e dalle basi di dati utilizzati per popolare l'EDW è chiamata *estrazione*. Solitamente non è necessario acquisire tutti i dati contenuti nei vari sistemi operazionali, ma è sufficiente un sottoinsieme. L'estrazione di un sottoinsieme di dati si basa su un'analisi estesa dei sistemi sorgente e di destinazione, che viene eseguita al meglio da un team diretto dall'amministratore dei dati e costituito tanto dagli utenti finali quanto dai professionisti del data warehouse.

I due tipi generici di estrazioni di dati sono l'estrazione statica e quella progressiva. L'estrazione statica viene utilizzata per il riempimento iniziale del data warehouse, mentre quella progressiva è utilizzata per la manutenzione continuativa del data warehouse.

**L'estrazione statica è un metodo di acquisizione di un'istantanea dei dati sorgente in un punto nel tempo.** La vista dei dati sorgente è indipendente dal momento in cui questi sono stati creati.

**L'estrazione progressiva acquisisce solo le modifiche che si sono verificate nei dati sorgente dall'ultima acquisizione.** Il metodo più comune è l'acquisizione del registro. Come si ricorderà, il registro della base di dati contiene alcune after image che registrano le modifiche più recenti apportate ai record della base di dati (Figura 12.7). Con l'acquisizione del registro, solo le after image che sono registrate dopo l'ultima acquisizione vengono selezionate dal registro.

English (1999) e White (2000) descrivono nel dettaglio i passaggi necessari a qualificare quali sistemi di record e altre sorgenti di dati utilizzare per l'estrazione nell'area dello staging. La qualità dipende dai seguenti fattori.

- Chiarezza della denominazione dei dati, così che i progettisti del warehouse sappiano esattamente quali dati sono presenti in un sistema sorgente.
- Completezza e accuratezza delle regole aziendali applicate da un sistema, che influiscono direttamente sulla precisione dei dati; le regole aziendali nel sistema sorgente devono corrispondere a quelle utilizzate nel data warehouse.
- Formato dei dati (formati comuni tra le sorgenti per favorire la corrispondenza dei dati correlati).

È anche importante stabilire degli accordi con i proprietari dei sistemi sorgente, così che questi possano informare gli amministratori dei data warehouse quando vengono apportate delle modifiche nei metadati del sistema. Poiché i sistemi delle transazioni cambiano spesso per adattarsi alle nuove esigenze aziendali e per utilizzare tecnologie hardware e software nuove e migliori, gestire le modifiche nei sistemi sorgente è una delle sfide principali del processo di estrazione. Le modifiche nel sistema richiedono una riconsiderazione della qualità dei dati e delle procedure per l'estrazione e la trasformazione dei dati. Queste procedure mappano i dati nei sistemi sorgente sui dati nel warehouse di destinazione (o data mart). Una mappa indica per ogni elemento di dati nel data warehouse quali dati da quali sistemi sorgente utilizzare per derivare i dati; le regole di trasformazione, che tratteremo in un paragrafo a parte, stabiliscono quindi come eseguire la derivazione. Nel caso di sistemi sorgente personalizzati, un amministratore di data warehouse dovrà sviluppare mappe personalizzate e routine di estrazione; modelli di mappe predefiniti possono essere acquistati con alcuni pacchetti software, come i sistemi ERP.

L'estrazione può essere eseguita da routine scritte con strumenti associati al sistema sorgente, per esempio uno strumento per esportare i dati. I dati vengono solitamente estratti in un formato di dati neutro, come ANSI o delimitato da virgole. A volte il comando `SELECT...INTO SQL` può essere utilizzato per creare una tabella che possa essere portata nell'area dello staging.

Una volta che le sorgenti dei dati sono state selezionate e che le routine di estrazione sono state scritte, i dati possono essere spostati nell'area dello staging, dove inizia il processo di pulizia.

### **Pulizia**

I dati nei sistemi operazionali qualificati sono spesso di qualità scadente. Alcuni degli errori e incoerenze tipici di questi dati sono i seguenti.

1. Nomi e indirizzi errati
2. Date di nascita impossibili o errate
3. Campi utilizzati per scopi diversi da quelli previsti
4. Corrispondenze errate tra indirizzi e codici di avviamento postale
5. Dati mancanti
6. Dati duplicati
7. Discrepanze (per esempio, indirizzi diversi) tra le sorgenti

Vediamo alcuni esempi di tali errori. I nomi dei clienti sono utilizzati frequentemente come chiavi primarie o come criteri di ricerca nei file dei clienti. Questi nomi sono tuttavia spesso errati o scritti in modi diversi nei vari file. Per esempio, "The Coca-Cola Company" è il nome corretto per la celebre casa produttrice di bibite. Questo nome può essere immesso nei record personalizzati come "Coca-Cola", "Coca Cola", "TCCC" e così via. Uno studio ha dimostrato che il nome "McDonald's" può essere scritto in 100 modi diversi!

Un altro tipo di inquinamento dei dati si verifica quando un campo viene utilizzato per scopi diversi da quelli previsti. Per esempio, in una banca un campo di record è stato concepito per contenere un numero di telefono. I manager di filiale non utilizzano il campo per questo scopo e vi memorizzano invece i tassi di interesse. Un altro esempio, riportato da una grande banca inglese, è ancora più bizzarro. Il programma di data scrubbing ha immesso nei file un cliente la cui occupazione era “cameriere sul Titanic”! (Devlin, 1997.)

Ci si potrebbe chiedere perché tali errori siano così diffusi nei dati operazionali. La qualità di questi dati è ampiamente determinata dal loro valore per l’azienda che è responsabile della loro raccolta. Sfortunatamente, accade spesso che l’azienda di raccolta dei dati sottovaluti i dati la cui precisione è invece importante per le applicazioni di downstream, come i data warehouse.

Assodato il verificarsi frequente degli errori, la cosa peggiore che un’azienda può fare è quella di copiare semplicemente i dati operazionali nel data warehouse. È invece importante migliorare la qualità dei dati sorgente attraverso una tecnica chiamata data scrubbing. Il data scrubbing (chiamato anche data cleansing, o pulizia dei dati) è una tecnica che utilizza il riconoscimento dei pattern e altre tecniche per aggiornare la qualità dei dati grezzi prima di trasformare i dati e di spostarli nel data warehouse.

Data warehouse di successo richiedono che venga implementato un programma formale di gestione della qualità totale (*Total Quality Management*, TQM). Un TQM si concentra sulla prevenzione dei difetti più che sulla loro correzione. Sebbene il data scrubbing possa aggiornare la qualità dei dati, non è una soluzione ideale sul lungo termine. Per una buona trattazione del TQM applicato alla gestione dei dati, si veda English (1999).

Il tipo di pulizia dei dati necessario dipende dalla qualità dei dati nel sistema sorgente. Oltre a correggere i tipi di problemi precedentemente identificati, esistono altre operazioni di pulizia comuni.

- Decodifica dei dati per renderli comprensibili alle applicazioni di data warehouse.
- Riformattazione e modifica dei tipi di dati ed esecuzione di altre funzioni per collocare i dati tratti da ciascuna sorgente nel formato di dati standard del data warehouse, pronti per la trasformazione.
- Aggiunta di marcature temporali che distinguano i valori dello stesso attributo nel tempo.
- Conversione tra unità di misura diverse.
- Generazione di chiavi primarie per ogni riga di una tabella (affronteremo la creazione delle chiavi primarie ed esterne di una tabella di data warehouse successivamente nel capitolo).
- Corrispondenza e unione di estrazione separate in un’unica tabella o file e corrispondenza dei dati che devono riempire la stessa riga della tabella generata (può essere un processo molto difficile quando vengono utilizzate chiavi diverse in sistemi sorgente differenti, quando le convenzioni di denominazione sono diverse e quando i sistemi sorgente sono errati).
- Registrazione degli errori individuati, loro correzione e rielaborazione dei dati corretti senza dover creare immissioni duplicate.
- Individuazione dei dati mancanti per completare il batch dei dati necessari per il caricamento successivo.

L’ordine in cui le varie sorgenti dei dati sono elaborate può fare la differenza. Per esempio, potrebbe essere necessario elaborare i dati dei clienti da un sistema di vendite prima che i dati demografici di un nuovo cliente tratti da un sistema esterno possano essere associati ai clienti.

Una volta che i dati vengono puliti nell’area dello staging, sono pronti per la trasformazione. Prima di discutere il processo di trasformazione nel dettaglio, vedremo rapidamente le procedure utilizzate per caricare i dati nel data warehouse o nei data mart.

### Caricamento e indicizzazione

L'ultimo passaggio nel riempimento del data warehouse aziendale è quello di caricare i dati selezionati nel data warehouse di destinazione e di creare gli indici necessari. Le due modalità di base per caricare i dati nell'EDW sono il refresh e l'aggiornamento.

La modalità di refresh è un approccio al completamento del data warehouse che utilizza una riscrittura massiccia dei dati di destinazione a intervalli periodici. I dati di destinazione vengono cioè scritti all'inizio per riempire il warehouse, che viene poi riscritto a intervalli regolari, sostituendo il contenuto precedente. Questa modalità non è più molto diffusa.

La modalità di aggiornamento è un approccio con il quale nel data warehouse vengono scritte solo le modifiche nei dati sorgente. Per supportare la natura periodica dei dati nel warehouse, i nuovi record vengono solitamente scritti nel data warehouse senza sovrascrivere o eliminare i record precedenti (si veda la Figura 12.9).

Come ci si può aspettare, la modalità di refresh è utilizzata generalmente per riempire il warehouse quando viene creato la prima volta. La modalità di aggiornamento è invece utilizzata per la manutenzione progressiva del warehouse di destinazione. La modalità di refresh è utilizzata in combinazione con l'acquisizione dei dati statici, mentre quella di aggiornamento è utilizzata con l'acquisizione progressiva dei dati.

Con entrambe le modalità, è necessario creare e mantenere gli indici che sono utilizzati per gestire i dati del warehouse. In un ambiente di data warehouse viene spesso utilizzata un tipo di indicizzazione chiamata *indicizzazione bitmap*. L'indicizzazione è stata trattata nel Capitolo 8; la prenderemo nuovamente in considerazione quando esamineremo la progettazione della base di dati per i data warehouse e i mart.

Basandosi sui celebri ed eccezionali data warehouse costruiti per Wal-Mart Corporation, Westerman (2001) esamina i fattori che determinano la frequenza degli aggiornamenti di un data warehouse. La sua indicazione è quella di aggiornare il data warehouse tutte le volte che può tornare utile. Aggiornamenti poco frequenti comportano carichi massicci e lunghe attese da parte degli utenti. I carichi quasi in tempo reale sono indispensabili per l'@ctive data warehousing, ma possono non essere efficaci e necessari per la maggior parte delle applicazioni di data mining e di analisi. Tuttavia, aggiornamenti quotidiani rendono impossibile reagire ad alcune condizioni mutevoli, come l'applicazione di nuovi prezzi o la modifica degli ordini d'acquisto per gli articoli a spostamento lento. Wal-Mart aggiorna il data warehouse continuamente, cosa che ha un senso per il tipo di tecnologia di data warehouse altamente parallela che utilizza.

Il caricamento dei dati nel warehouse consiste solitamente nell'aggiungere nuove righe alle tabelle nel warehouse. Significa anche aggiornare le righe esistenti con dati nuovi (per esempio, completare i valori mancanti da una sorgente di dati supplementare) o eliminare i dati identificati che sono diventati obsoleti o che sono stati caricati in modo errato in una precedente operazione di caricamento. I dati possono essere caricati dall'area dello staging nel warehouse attraverso:

- comandi SQL (come INSERT o UPDATE);
- utility di caricamento speciali fornite dal data warehouse o da produttori di terze parti;
- routine personalizzate codificate dagli amministratori del warehouse.

In ogni caso, queste routine non devono solo aggiornare il data warehouse ma anche generare report di errore che visualizzino i dati rifiutati (per esempio, quando si cerca di aggiungere una riga con una chiave duplicata o di aggiornare una riga che non esiste in una tabella del data warehouse).

Le utility di caricamento funzionano in batch in modalità continuativa. Con una utility, si scrive uno script che definisce il formato dei dati nell'area dello staging e quale area dello staging mappare su quali

campi del data warehouse. L'utility può essere in grado di convertire i tipi di dati per un campo nell'area dello staging nel campo di destinazione, e può eseguire la logica IF...THEN...ELSE per gestire l'area dello staging area in vari formati o dirigere i dati di input a tabelle diverse nel data warehouse. Può eliminare tutti i dati in una tabella (DROP TABLE) prima del caricamento dei dati (modalità di refresh) o può aggiungere nuove righe (modalità di aggiornamento); può inoltre essere in grado di ordinare i dati di input, così che le righe vengano aggiunte prima di essere aggiornate. Il programma di utility si esegue come qualsiasi altra procedura per il DBMS; idealmente funzioneranno anche tutti i controlli di concomitanza del DBMS, oltre al riavvio e al ripristino in caso di fallimento del DBMS durante il caricamento. Poiché l'esecuzione di un caricamento può occupare molto tempo, è fondamentale poter riavviare un caricamento da un punto di controllo qualora il DBMS fallisse a metà dell'operazione.

## 12.5 Trasformazione dei dati

La trasformazione dei dati (o semplicemente trasformazione) è al centro del processo di riconciliazione dei dati. La trasformazione dei dati è il componente della riconciliazione dei dati che converte i dati dal formato dei sistemi operazionali sorgente in quello del data warehouse aziendale: accetta i dati dal componente di acquisizione dei dati (dopo l'eventuale data scrubbing), mappa i dati nel formato del livello dei dati riconciliati e quindi li passa al componente di caricamento e indicizzazione.

La trasformazione dei dati può variare dalla semplice modifica del formato o della rappresentazione dei dati a un esercizio più complesso di integrazione dei dati. A seguire riportiamo tre esempi.

1. L'agente di vendita richiede di scaricare sul proprio computer portatile i dati dei clienti da una base di dati su mainframe. In questo caso la trasformazione richiesta è la semplice mappatura dei dati dalla rappresentazione EBCDIC a quella ASCII, che può essere facilmente eseguita con software dedicati.
2. Un'azienda manifatturiera ha memorizzato i dati sui prodotti in tre diversi sistemi legacy: un sistema di produzione, uno di marketing e uno di ingegneria. L'azienda deve sviluppare una vista consolidata dei dati sui prodotti. In questo caso la trasformazione dei dati implica numerose funzioni diverse, comprese la risoluzione di strutture di chiave diverse, la conversione in un insieme di codici comune e l'integrazione dei dati da più sorgenti. Le funzioni sono piuttosto dirette e gran parte del software necessario può essere generato utilizzando un pacchetto software commerciale standard con un'interfaccia grafica.
3. Una grande azienda sanitaria gestisce un gruppo geograficamente distribuito di ospedali, cliniche e altri centri di cura. Poiché molte di queste unità di cura sono state incorporate mediante acquisizioni nel tempo, i dati sono eterogenei e non coordinati. Per una serie di ragioni, l'azienda deve sviluppare un data warehouse che fornisca un'unica vista aziendale. Questo compito richiederà l'intera gamma di funzioni di trasformazione descritte di seguito, compreso lo sviluppo parziale di un software personalizzato.

Le funzioni eseguite nel data scrubbing e nella trasformazione dei dati si fondono tra loro. In generale, l'obiettivo del data scrubbing è quello di correggere gli errori nei valori dei dati sorgente, mentre quello della trasformazione è convertire il formato dei dati dal sistema sorgente a quello di destinazione. Si noti che è essenziale eseguire lo scrubbing dei dati prima della trasformazione; in caso contrario, se ci sono errori nei dati che devono essere trasformati, gli errori rimarranno anche dopo la trasformazione.

## 12.5.1 Funzioni della trasformazione dei dati

La trasformazione dei dati prevede un'ampia gamma di funzioni diverse, che possono essere suddivise in due categorie; funzioni a livello di record e funzioni a livello di campo. Nella maggior parte delle applicazioni di data warehouse, è richiesta una combinazione di alcune o anche di tutte queste funzioni.

### Funzioni a livello di record

Le funzioni principali che operano a livello di un insieme di record, come un file o una tabella, sono la selezione, l'unione, la normalizzazione e l'aggregazione.

La **selezione** (chiamata anche *subsetting*) è il processo di partizionamento dei dati in base a criteri predefiniti. Nel caso di applicazioni di data warehouse, è utilizzata per estrarre i dati importanti dai sistemi sorgente che verranno utilizzati per riempire il data warehouse. La selezione è infatti una parte della funzione di acquisizione descritta in precedenza.

Quando i dati sorgente sono relazionali, le istruzioni SELECT SQL possono essere utilizzate per la selezione (si veda il Capitolo 9 per una discussione dettagliata). Per esempio, si ricorderà che l'acquisizione progressiva è spesso implementata selezionando le after image dal registro della base di dati che sono state create dopo l'ultima acquisizione. Una tipica after image è mostrata nella Figura 12.7. Si supponga che le after image per questa applicazione siano memorizzate in una tabella chiamata ACCOUNT\_HISTORY. Le after image che sono state create dopo il 31 dicembre 2001 possono quindi essere selezionate con le seguenti istruzioni:

```
SELECT *
FROM ACCOUNT_HISTORY
WHERE Create_Date > 12/31/2001;
```

L'**unione** combina i dati da più sorgenti in un'unica tabella o vista. L'unione dei dati è una funzione importante nelle applicazioni di data warehouse, poiché è spesso necessario consolidare i dati provenienti da varie sorgenti. Per esempio, una compagnia di assicurazioni può avere i dati sui clienti distribuiti in più file e basi di dati. Quando i dati sorgente sono relazionali, le istruzioni SQL possono essere utilizzate per eseguire un'operazione di unione (si veda il Capitolo 9 per i dettagli).

L'unione è spesso complicata da fattori come i seguenti.

1. La sorgente dei dati spesso non è relazionale, nel qual caso le istruzioni SQL non possono essere utilizzate. Dovranno invece essere codificate istruzioni di linguaggio procedurale.
2. Anche nel caso di dati relazionali, le chiavi primarie per le tabelle da unire provengono spesso da domini diversi (per esempio, i numeri delle parti rispetto ai numeri di catalogo). Queste chiavi devono essere riconciliate prima che un'unione SQL possa essere eseguita.
3. I dati sorgente possono contenere errori, cosa che rende le operazioni di unione pericolose.

La **normalizzazione** è il processo con cui si scompongono le relazioni che contengono anomalie per produrre relazioni più piccole e ben strutturate (si veda il Capitolo 7 per i dettagli). Come indicato in precedenza, i dati sorgente nei sistemi operazionali sono spesso denormalizzati (o semplicemente non normalizzati). I dati devono essere prima normalizzati come parte della trasformazione dei dati.

L'**aggregazione** è il processo di trasformazione dei dati da un livello dettagliato a uno di riepilogo. Per esempio, nelle aziende al dettaglio, le singole transazioni di vendita possono essere riepilogate per produrre vendite totali in base al negozio, al prodotto, alla data e così via. Poiché nel nostro modello il data warehouse aziendale contiene solo dati dettagliati, l'aggregazione non è solitamente associata

a questo componente; tuttavia, resta una funzione importante nel riempimento dei data mart, come si vedrà in seguito.

Poiché la selezione, l'unione e l'aggregazione possono essere spesso realizzate in SQL, alcuni esperti di data warehousing sostengono che la trasformazione debba essere eseguita in un data store operativo o addirittura nel data warehouse stesso.

### **Funzioni a livello di campo**

Una funzione a livello di campo converte i dati da un formato dato del record sorgente in un formato diverso in un formato diverso del record di destinazione. Le funzioni a livello di campo sono di due tipi: a campo singolo e multicampo.

Una trasformazione a campo singolo converte i dati da un singolo campo sorgente in un singolo campo di destinazione. La Figura 12.11a è una rappresentazione di questo tipo di trasformazione (indicata dalla lettera T nel diagramma). Un esempio di trasformazione a campo singolo è la conversione di un'unità di misura da imperiale a metrica.

Come mostrato nelle Figure 12.11b e c, esistono due metodi di base per eseguire una trasformazione a campo singolo: quello algoritmico e quello della tabella di ricerca. Una trasformazione algoritmica viene eseguita utilizzando una formula o un'espressione logica. La Figura 12.11b mostra la conversione di una temperatura da gradi Fahrenheit a Celsius mediante una formula. Quando l'algoritmo non è adatto, si può utilizzare una tabella di ricerca. La Figura 12.11c mostra l'uso di una tabella per convertire le sigle degli Stati in nomi (un tipo di conversione molto comune nelle applicazioni di data warehouse).

Una trasformazione multicampo converte i dati da uno o più campi sorgente in uno o più campi di destinazione. La Figura 12.12 mostra due trasformazioni multicampo.

La Figura 12.12a è un esempio di trasformazione multi-a-uno (in questo caso due campi sorgente sono mappati su un campo di destinazione). Nel record sorgente, la combinazione del nome dell'impiegato con il numero di telefono è utilizzata come chiave primaria. Questa combinazione è subdola e può non identificare in modo univoco una persona. Quando si crea un record di destinazione, la combinazione viene quindi mappata su un identificativo di impiegato univoco (Emp\_ID). Per supportare questa informazione, potrebbe essere creata una tabella di ricerca, mentre un programma di data scrubbing potrebbe essere utilizzato per identificare i duplicati nei dati sorgente.

La Figura 12.12b è un esempio di trasformazione uno-a-molti (in questo caso, un campo sorgente è stato convertito in due campi di destinazione). Nel record sorgente, un codice di prodotto è stato utilizzato per codificare la combinazione tra nome della marca e nome del prodotto (l'uso di tali codici è molto comune nei dati operazionali). Tuttavia, nel record di destinazione, si preferisce visualizzare il testo completo che descrive i nomi dei prodotti e delle marche. Anche in questo caso si può utilizzare una tabella di ricerca a questo scopo.

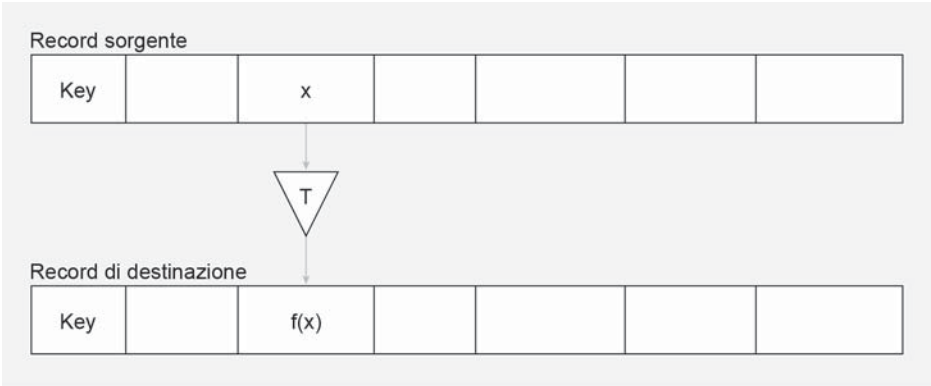
### **Trasformazioni più complesse**

Nella Figura 12.12, le trasformazioni multicampo includono entrambe un record sorgente e uno di destinazione. Come regola generale, questo tipo di trasformazione implica più di un record sorgente e/o più di uno record sorgente e/o più di uno di destinazione. Nei casi più complessi, questi record possono anche generare sistemi operazionali diversi in zone con fusi orari diversi (Devlin, 1997).

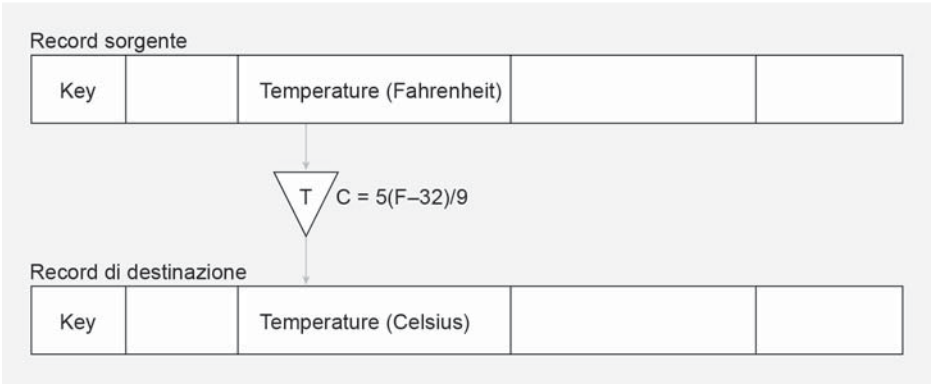
## **12.5.2 Strumenti di supporto alla riconciliazione dei dati**

La riconciliazione dei dati è un processo molto complesso e impegnativo, per il cui supporto è necessario sviluppare (o acquistare) un'ampia gamma di applicazioni software strettamente integrate. Fortuna-

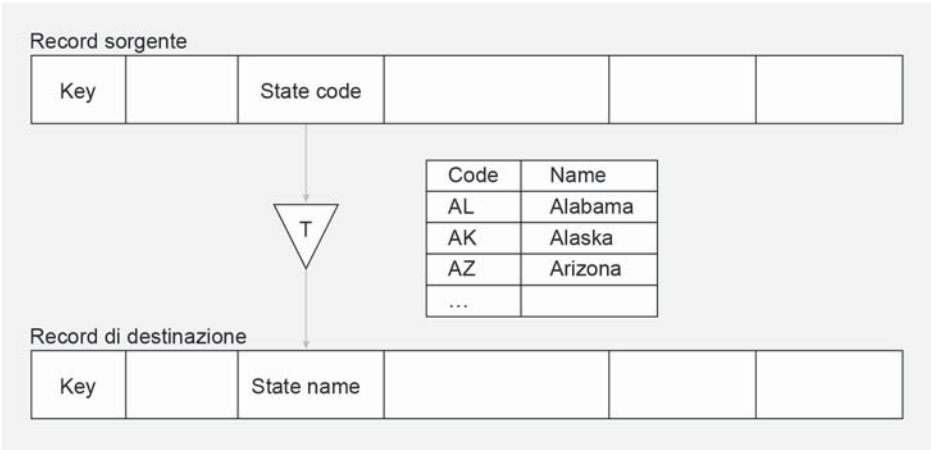
**Figura 12.11**  
Trasformazione a campo singolo.  
(a) Rappresentazione di base.



(b) Trasformazione algoritmica.



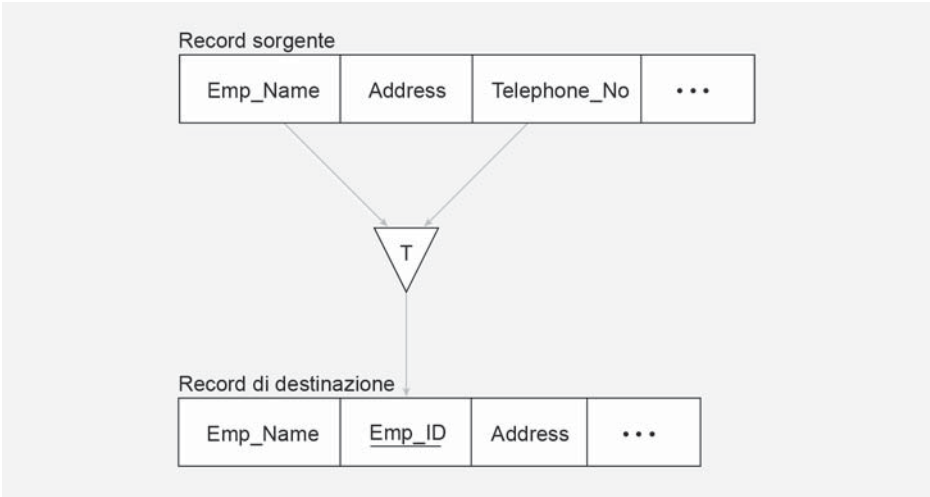
(c) Tabella di ricerca.



tamente, sono disponibili numerosi strumenti software potenti che assistono le aziende nello sviluppo di queste applicazioni. In questo paragrafo descriveremo tre categorie di tali strumenti, ovvero quelli per la qualità, la conversione e la pulizia dei dati (per una trattazione più dettagliata, si veda Williams, 1997). La Tabella 12.3 riassume gli strumenti di queste tre categorie.



**Figura 12.12**  
Trasformazioni multicampo.  
(a) Un'associazione M:1.



(b) Un'associazione 1:M.

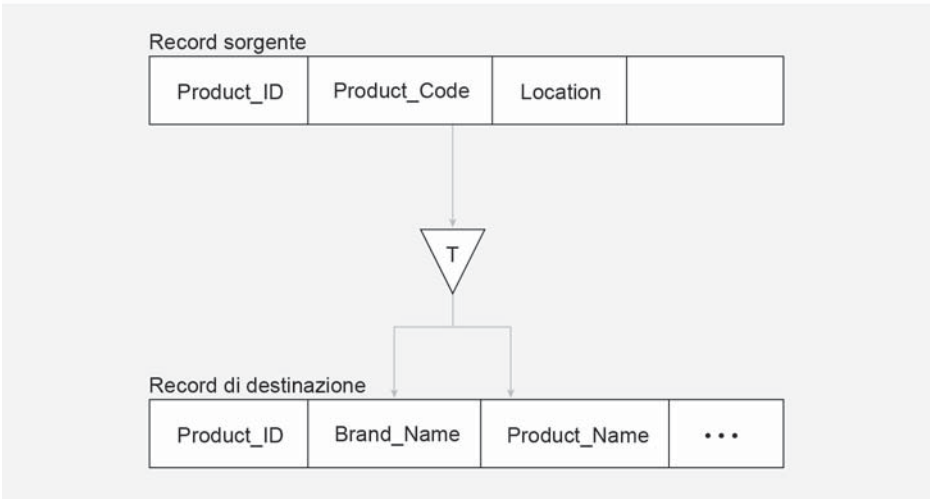


**Tabella 12.3** Strumenti di supporto alla riconciliazione dei dati.

| Nome del prodotto | Società                                                   | Descrizione                                                                 |
|-------------------|-----------------------------------------------------------|-----------------------------------------------------------------------------|
| WizRules          | WizSoft, Inc.<br>www.wizsoft.com                          | Ricerca delle regole                                                        |
| Extract           | Evolutionary Technologies International<br>www.evtech.com | Estrazione, trasformazione, caricamento e indicizzazione                    |
| InfoRefiner       | Computer Associates www.ca.com                            | Estrazione, trasformazione, caricamento e indicizzazione                    |
| DataBridge        | Taurus Software, Inc. www.taurus.com                      | Estrazione, trasformazione, caricamento e indicizzazione                    |
| Genio Suite       | Hummingbird, Ltd. www.hummingbird.com                     | Estrazione, trasformazione, arricchimento e caricamento                     |
| Power Mart        | Informatica www.informatica.com                           | Estrazione, trasformazione, caricamento e indicizzazione                    |
| Trillium          | Harte-Hanks www.harte-hanks.com                           | Analisi della qualità e pulizia dei dati                                    |
| Centric           | Firstlogic www.firstlogic.com                             | Analisi della qualità e pulizia dei dati (si veda Information Quality Demo) |
| Integrity         | Vality Technology, Inc. www.vality.com                    | Famiglia di prodotti per l'analisi della qualità e il data scrubbing        |

### **Strumenti per la qualità dei dati**

Questi strumenti sono concepiti per definire la qualità dei dati nei sistemi esistenti e per confrontarli con i data warehouse. Sono quindi utili durante la fase iniziale dello sviluppo del warehouse. Uno di questi strumenti è WizRules (WizSoft, Inc.), che cerca tra i record nelle tabelle esistenti e scopre le regole associate ai dati. Per esempio, una regola potrebbe asserire: “Se Customer\_Name è Able o Baker, allora la City è San Diego; la probabilità è dello 0,95. Questa regola si ritrova in 50 record”. Il prodotto identifica anche i record che deviano dalle regole stabilite.

### **Strumenti di conversione dei dati**

Gli strumenti di conversione dei dati effettuano generalmente le funzioni mainstream, ovvero estrazione, trasformazione, caricamento e indicizzazione. Tali strumenti, elencati nella Tabella 12.3 includono Extract (Evolutionary Technologies International), InfoRefiner (Computer Associates), DataBridge (Taurus Software, Inc.), Genio Suite (Hummingbird Ltd.) e Power Mart (Informatica). Si tratta fondamentalmente di strumenti di generazione di programmi che accettano come input lo schema (o le descrizioni dei file) dei file sorgente e di destinazione e le regole aziendali che devono essere utilizzate per la trasformazione dei dati. Regole aziendali tipiche includono le formule, gli algoritmi e le tabelle di ricerca (come quelle mostrate nella Figura 12.11). Gli strumenti generano quindi il codice di programma necessario a eseguire le funzioni di trasformazione su una base corrente. Alcuni di essi, come Genio MetaLink per SAP R/3, che è parte della Genio Suite, forniscono dei modelli per l'estrazione e la trasformazione dei dati dai pacchetti software più diffusi.

### **Strumenti pulizia dei dati**

L'ultima categoria contiene uno strumento concepito specificatamente per fornire funzioni di pulizia e correlate. Integrity (Vality Technology, Inc.), Trillium (Harte-Hanks) e Centric (Firstlogic) sono pensati per eseguire le analisi di qualità, la pulizia e la riorganizzazione dei dati (nel concreto, scoprire le regole aziendali e le associazioni tra le entità).

## **12.6 Il livello dei dati derivati**

Il livello dei dati derivati è il livello associato ai data mart logici o fisici (Figura 12.6) e quello con cui interagiscono normalmente gli utenti per le loro applicazioni di supporto alle decisioni. In linea teorica, viene prima progettato il livello dei dati riconciliati come base per quello dei dati derivati, a prescindere dal fatto che i data mart siano dipendenti, indipendenti o virtuali. In questo paragrafo si analizzeranno le caratteristiche del livello dei dati derivati e il modo in cui viene tratto dal livello dei dati riconciliati. Si vedrà quindi lo schema a stella (o modello dimensionale), che è quello attualmente più utilizzato per implementare questo livello di dati. Uno schema a stella è uno speciale modello di dati relazionali de-normalizzato. Come si vedrà, il livello dei dati derivati può utilizzare le relazioni normalizzate; tuttavia, la maggior parte delle aziende costruisce numerosi data mart, ognuno ottimizzato attorno a un oggetto aziendale di interesse per un gruppo di utenti (Moriarty, 1995).

### **12.6.1 Caratteristiche dei dati derivati**

In precedenza abbiamo definito i dati derivati come dati che sono stati selezionati, formattati e aggregati per le applicazioni di supporto alle decisioni degli utenti finali. Come mostrato nella Figura 12.6, l'origine dei dati derivati è rappresentata dai dati riconciliati. I dati derivati in un data mart vengono

solitamente ottimizzati per le necessità di determinati gruppi di utenti, come funzioni, gruppi di lavoro o anche individui. Un modo comune di agire è quello di selezionare quotidianamente i dati importanti per il data warehouse aziendale, formattare e aggregare i dati come necessario e poi caricarli e indicizzarli nei data mart di destinazione.

Gli obiettivi che si raggiungono con i dati derivati sono piuttosto diversi da quelli dei dati riconciliati.

1. Rendere le applicazioni di supporto alle decisioni facili da utilizzare.
2. Fornire una risposta rapida alle interrogazioni predefinite degli utenti o alle richieste di informazioni.
3. Personalizzare i dati per determinati gruppi di utenti di destinazione.
4. Supportare interrogazioni ad hoc e applicazioni di data mining.

Per soddisfare queste esigenze, si ricorre alle caratteristiche elencate di seguito, tipiche dei dati derivati.

1. Sono presenti sia i dati dettagliati sia quelli aggregati.
  - a. I dati dettagliati sono spesso, ma non sempre, legati a un periodo; offrono cioè una registrazione storica.
  - b. I dati aggregati vengono formattati in modo da rispondere rapidamente alle interrogazioni predefinite o comuni.
2. I dati sono distribuiti ai server di funzione.
3. Il modello dei dati più utilizzato per un data mart è lo schema a stella, che è un modello del genere relazionale. A volte si utilizzano anche i modelli proprietari.

## 12.6.2 Lo schema a stella

Secondo Devlin (1997), uno schema a stella è una rappresentazione semplice della base di dati, particolarmente adatta per interrogazioni ad hoc, in cui i dati dimensionali (che descrivono come i dati vengono aggregati comunemente) sono separati dai dati dei fatti o degli eventi (che descrivono le singole transazioni aziendali). Un altro nome spesso utilizzato è quello di *modello dimensionale* (Kimball, 1996). Sebbene lo schema a stella sia più adatto alle interrogazioni ad hoc e ad altre forme di elaborazione delle informazioni, non si rivela adatto per l'elaborazione delle transazioni on-line, ed è quindi poco utilizzato nei sistemi e data store operazionali e nei data warehouse aziendali.

### Tabelle di fatti e tabelle di dimensioni

Uno schema a stella è costituito da due tipi di tabelle: di fatti e di dimensioni. Le tabelle di fatti contengono dati reali o quantitativi di un'azienda, come le unità vendute, gli ordini effettuati e così via. Le tabelle di dimensioni contengono dati descrittivi sugli oggetti dell'azienda e sono solitamente la sorgente degli attributi utilizzati per qualificare, classificare o riepilogare i fatti in interrogazioni, report o grafici. Lo schema a stella più semplice è costituito da una tabella di fatti circondata da più tabelle di dimensioni. Le dimensioni (oggetti) dell'azienda possono essere Product, Customer e Period. Period, ovvero il tempo, è l'unica dimensione sempre presente. Questa struttura è mostrata nella Figura 12.13, che contiene quattro tabelle di dimensioni.

Ogni tabella di dimensioni ha al proprio centro un'associazione uno-a-molti, una chiave primaria semplice e numerosi attributi non di chiave. La chiave primaria è a sua volta una chiave esterna nella

**Figura 12.13**  
Componenti  
di uno schema a stella.

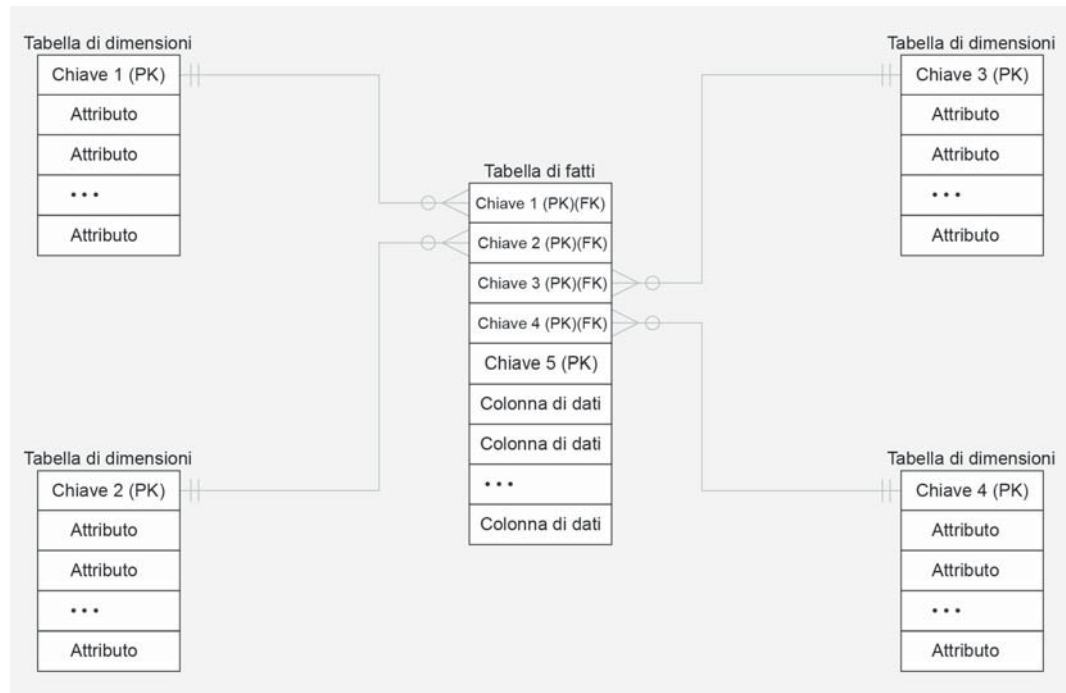


tabella di fatti (come mostrato nella Figura 12.13). La chiave primaria della tabella di fatti è una chiave composta costituita dalla concatenazione di tutte le chiavi esterne (quattro chiavi nella Figura 12.13), oltre ad eventuali altri componenti che non corrispondono alle dimensioni. L'associazione tra ogni tabella di dimensioni e la tabella di fatti fornisce un percorso di unione che consente agli utenti di interrogare facilmente la base di dati, utilizzando istruzioni SQL per ogni interrogazione predefinita o ad hoc. Gli attributi non di chiave sono solitamente chiamati *colonne di dati*, come mostrato nella Figura 12.13.

Come si sarà notato, lo schema a stella non è un nuovo modello di dati, quanto un'implementazione denormalizzata del modello di dati relazionale. La tabella di fatti gioca il ruolo di un'entità associativa *n*-aria che collega le istanze delle varie dimensioni. Per un ripasso sulle entità associative si veda il Capitolo 5, mentre per un uso dell'entità associativa si veda la Figura 5.11. Le tabelle di dimensioni sono denormalizzate. La maggior parte degli esperti considera la denormalizzazione come accettabile perché le dimensioni non vengono aggiornate e risparmiano unioni costose; la stella viene quindi ottimizzata attorno a determinati fatti e oggetti aziendali.

### Esempio di schema a stella

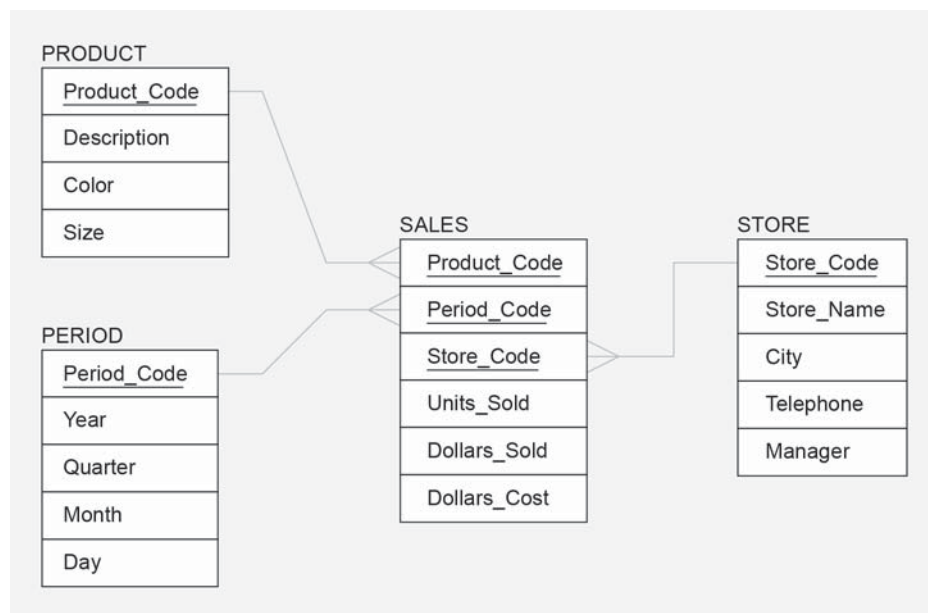
La Figura 12.14 mostra un semplice esempio di schema a stella. Lo schema ha tre tabelle di dimensioni, PRODUCT, PERIOD e STORE, e una di fatti, SALES. La tabella di fatti è utilizzata per registrare tre fatti aziendali: unità totali vendute, totali delle vendite in dollari e costo totale in dollari. Questi totali vengono registrati per ogni valore di prodotto, periodo e negozio possibile.

Alcuni dati di esempio contenuti in questo schema sono mostrati nella Figura 12.15. Dalla tabella di fatti si ricavano, per esempio, i seguenti fatti per il prodotto numero 110 durante il periodo 002.

1. Trenta unità sono state vendute nel negozio S1. Le vendite totali in dollari sono state di 1500 e il costo totale in dollari è stato di 1200.

**Figura 12.14**

Esempio di un semplice schema a stella.



2. Quaranta unità sono state vendute nel negozio S3. Le vendite totali in dollari sono state di 2000 e il costo totale in dollari è stato di 1200.

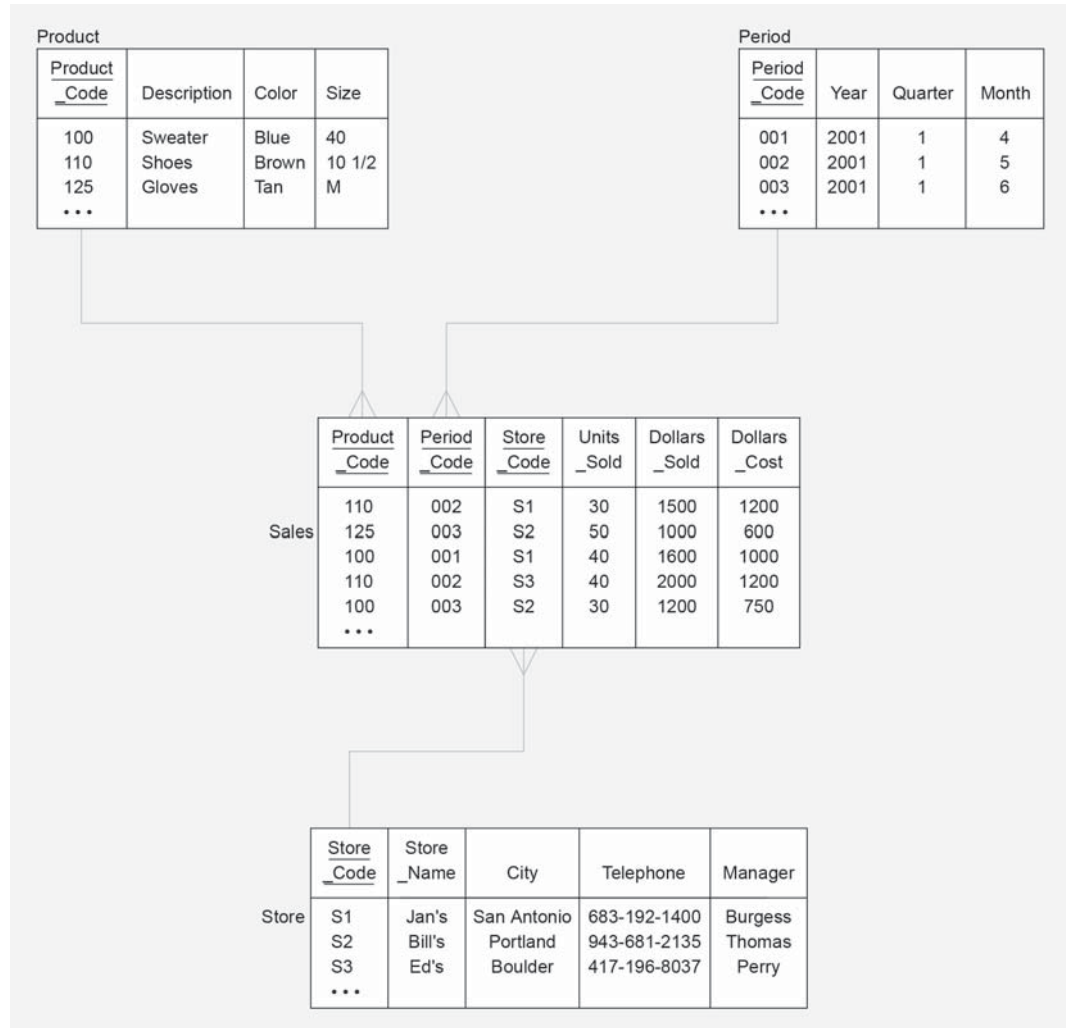
Altri dettagli sulle dimensioni possono essere ottenuti dalle tabelle di dimensioni. Per esempio, nella tabella PERIOD si vede che il periodo 002 corrisponde all'anno 2001, primo trimestre, quinto mese. Si provi a tracciare le altre dimensioni in modo analogo.

### Chiavi surrogate

Ogni chiave utilizzata per unire la tabella di fatti a una di dimensioni dovrebbe essere surrogata (non intelligente o assegnata dal sistema) e non una chiave che utilizza un valore aziendale (chiave intelligente o di produzione). Nella Figura 12.15, Product\_Code, Store\_Code e Period\_Code dovrebbero quindi essere tutte chiavi surrogate in entrambi i tipi di tabelle. Per esempio, se si vuole conoscere il numero di catalogo di un prodotto, quello di assemblaggio o quello delle scorte, questi attributi dovrebbero essere memorizzati con Description, Color e Size come attributi della tabella di dimensioni PRODUCT. Di seguito sono elencate le ragioni principali per questa regola della chiave surrogata (Kimball, 1998).

- Le chiavi aziendali cambiano, spesso lentamente, nel tempo, mentre è necessario ricordare i valori delle chiavi aziendali vecchie e nuove per lo stesso oggetto. Come si vedrà in seguito, una chiave surrogata permette di gestire facilmente le chiavi mutevoli e non note.
- Utilizzando una chiave surrogata si può tenere traccia dei diversi valori di attributo non di chiave per la stessa chiave di produzione nel tempo. Così, se la confezione di un prodotto cambia di dimensioni nel tempo, sarà possibile associare la stessa chiave di produzione del prodotto a più chiavi surrogate, una per ciascuna dimensione.
- Le chiavi surrogate sono spesso più semplici e più brevi, soprattutto quando la chiave di produzione è tipo di composto.

**Figura 12.15**  
Schema a stella con  
dati di esempio.



- Le chiavi surrogate possono avere la stessa lunghezza e lo stesso formato di tutte le chiavi, a prescindere dalle dimensioni coinvolte nella base di dati, comprese le date.

### Grana delle tabelle di fatti

I dati grezzi dello schema a stella sono conservati nella tabella di fatti. Come si potrà intuire, determinare il livello più basso di dati di fatti dettagliati è la fase più importante nella progettazione del data mart. Il livello del dettaglio di questi dati è determinato dall'intersezione di tutti i componenti della chiave primaria della tabella di fatti. L'intersezione delle chiavi primarie è detta **grana** della tabella di fatti. La definizione della grana è cruciale e dipende dalle esigenze del processo decisionale aziendale. C'è sempre il modo di riepilogare i dati di fatto aggregando gli attributi di dimensioni, ma nel data mart non c'è modo di capire l'attività aziendale a un livello di dettaglio maggiore di quello garantito dalla grana.

La grana più fine potrebbe corrispondere a ciascuna transazione aziendale, come un articolo di una singola linea o un singolo articolo su una ricevuta di vendita, un ordine di cambiamento nel personale, un articolo di una linea su una ricevuta di produzione, un reclamo su una polizza d'assicurazione o una transazione di un singolo individuo presso uno sportello bancomat. Una grana transazionale permette agli utenti di eseguire un'analisi del paniere di mercato, che è lo studio dei comportamenti di acquisto dei singoli clienti. Una grana più grossa del livello transazionale potrebbe essere tutte le vendite di un prodotto in un dato giorno, tutte le ricevute di un materiale grezzo in un dato mese presso un magazzino specifico, o l'effetto a rete di tutte le transazioni bancomat per una sessione dello sportello automatico. Più la grana della tabella di fatti è sottile, più dimensioni sono presenti, più righe di fatti esistono e, spesso, più il modello del data mart è vicino al modello del data store operativo.

Kimball (2001) e altri raccomandano l'uso della grana più fine possibile, entro i limiti della tecnologia del data mart. Anche quando le esigenze informative dell'utente del data mart implicano un certo livello di grana aggregata, gli utenti pongono domande più dettagliate (drill down) per spiegare il perché dell'esistenza di determinati pattern. Non si può eseguire il "drill-down" al di sotto del livello della grana delle tabelle di fatti (senza accedere ad altre sorgenti, come il data warehouse aziendale, il data store operazione o i sistemi sorgente originali, che possono conferire più complessità all'analisi).

### Dimensione della tabella di fatti

Come ci si può aspettare, la grana della tabella di fatti ha un impatto diretto sulla dimensione della tabella stessa. Per calcolare il numero delle righe in una tabella di fatti si procede come descritto di seguito.

1. Calcolare il numero di valori possibili per ciascuna dimensione associata alla tabella di fatti (in altre parole, il numero di valori possibili per ogni chiave esterna nella tabella di fatti).
2. Moltiplicare i valori ottenuti al punto 1 dopo aver apportato le modifiche necessarie.

Proviamo ad applicare questo approccio allo schema a stella mostrato nella Figura 12.15, presumendo i valori seguenti per le dimensioni:

numero totale dei negozi: 1000

numero totale dei prodotti: 10.000

numero totale dei periodi: 24 (due dati di anni mensili)

Sebbene ci sia un totale di 10.000 prodotti, è probabile che solo una parte di essi registri delle vendite in un dato mese. Poiché nella tabella di fatti appare il totale degli articoli solo per quegli articoli che registrano una vendita in un mese specifico, la figura andrà modificata. Si supponga che una media del 50 per cento (o 5000 articoli) registri una vendita in un dato mese; per calcolare il numero di righe nella tabella di fatti si proceda con la seguente operazione:

$$\begin{aligned}\text{righe totali} &= 1000 \text{ negozi} \times 5000 \text{ prodotti attivi} \times 24 \text{ mesi} \\ &= 120.000.000 \text{ righe (!)}\end{aligned}$$

In questo esempio, relativamente piccolo, ci si aspetta che la tabella di fatti che contiene due anni di totali mensili abbia ben oltre 100 milioni di righe. L'esempio illustra chiaramente che la dimensione della tabella di fatti è spesso più grande di quella delle tabelle di dimensioni. Per esempio, la tabella STORE ha 1000 righe, la tabella PRODUCT ha 10.000 righe e la tabella PERIOD ne ha 24.

Se si conosce la dimensione di ciascun campo nella tabella di fatti, se ne potrà stimare ulteriormente (in byte) la dimensione. La tabella di fatti SALES nella Figura 12.15 ha sei campi; se ognuno di essi è lungo in media 4 byte, si potrà valutare la dimensione totale della tabella di fatti come segue:

$$\begin{aligned}\text{dimensione totale} &= 120.000.000 \text{ righe} \times 6 \text{ campi} \times 4 \text{ byte/campo} \\ &= 2.880.000.000 \text{ byte o } 2,88 \text{ gigabyte}\end{aligned}$$

La dimensione della tabella di fatti dipende sia dal numero delle dimensioni sia dalla grana della tabella di fatti. Si immagini che dopo aver utilizzato la base di dati nella Figura 12.15 per un breve periodo di tempo, l'ufficio marketing richieda che i totali giornalieri vengano accumulati nella tabella di fatti (si tratta di un'evoluzione tipica di un data mart). In questo caso, il numero di righe è calcolato come segue:

$$\begin{aligned}\text{righe totali} &= 1000 \text{ negozi} \times 2000 \text{ prodotti attivi} \times 720 \text{ giorni (due anni)} \\ &= 1.440.000.000 \text{ righe}\end{aligned}$$

Nel calcolo precedente si è presunto che il 20 per cento di tutte vendite di prodotti si registrasse in un dato mese. È ora lecito aspettarsi che la base di dati contenga ben oltre 1 miliardo di righe. La dimensione della base di dati viene calcolata come segue:

$$\begin{aligned}\text{dimensione totale} &= 1.440.000.000 \text{ righe} \times 6 \text{ campi} \times 4 \text{ byte/campo} \\ &= 34.560.000.000 \text{ byte (o } 34,56 \text{ gigabyte)}\end{aligned}$$

Attualmente molte grandi catene di negozi (come Wal-Mart, Kmart o Sears) e aziende elettroniche (come Travelocity.com o MatchLogic.com) hanno data warehouse o data mart. La dimensione dei warehouse è nell'ordine dei multipli dei terabyte e cresce rapidamente con l'aumento della richiesta degli utenti di avere più dimensioni e una grana il più possibile sottile nella tabella di fatti.

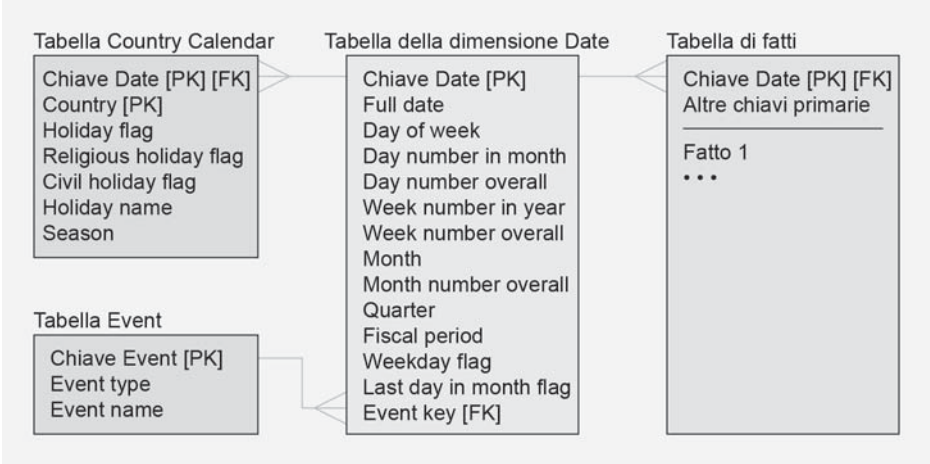
### Modellare data e ora

Poiché i data mart registrano i fatti sulle dimensioni nel tempo, la data e l'ora, d'ora in poi chiamate semplicemente "data", sono sempre una tabella di dimensioni, e la chiave surrogata di data è sempre uno dei componenti della chiave primaria di qualsiasi tabella di fatti. Poiché un utente potrebbe voler aggregare i fatti in base a più aspetti della data, una dimensione di data può avere attributi non di chiave. Inoltre, poiché alcune caratteristiche della data dipendono dal Paese o da un evento (per esempio, se la data è considerata festiva o se in quella data è sempre programmato un evento standard, come un festival o una partita di calcio), la modellazione della dimensione di data può essere molto più complessa di quanto illustrato finora.

La Figura 12.16 mostra una rappresentazione tipica della dimensione di data. Come visto in precedenza, una chiave di data surrogata appare come parte della chiave primaria della tabella di fatti ed è la chiave primaria della tabella di dimensioni. Gli attributi non di chiave della tabella di dimensione di data includono tutte le caratteristiche delle date che gli utenti utilizzano per classificare, riepilogare e raggruppare i dati che non variano in base al Paese o agli eventi. Per un'azienda che lavora in più Paesi (o per più unità geografiche in cui le date hanno caratteristiche diverse), abbiamo aggiunto una tabella Country Calendar che contiene le caratteristiche di ogni data in ciascun Paese. Nella tabella Country Calendar, la chiave Date è esterna, e ogni riga è univoca per la combinazione della chiave Date e di Country, che formano la chiave primaria composta di questa tabella. Un evento speciale può verificarsi in una data specifica (per semplicità, qui si presume che non avvenga più di un evento in una data specifica). Abbiamo normalizzato i dati Event creando una tabella di eventi, così che i dati descrittivi di ciascun evento (per esempio, "Strawberry Festival" o "Homecoming Game") siano memorizzati una volta sola.



**Figura 12.16**  
Modellazione delle date.



### 12.6.3 Varianti dello schema a stella

Lo schema a stella semplice introdotto in precedenza si adatta a molte applicazioni. Tuttavia, per risolvere problemi di modellazione più complessi sono spesso necessarie diverse estensioni dello schema. In questo paragrafo ne descriveremo brevemente due: le tabelle di fatti multiple e lo schema a snowflake. Per avere informazioni su altre estensioni, si vedano Poe (1996) e [www.ralphkimball.com](http://www.ralphkimball.com).

#### Tabelle di fatti multiple

Per ragioni di prestazioni o per altri motivi, può essere utile definire più di una tabella di fatti in uno schema a stella dato. Per esempio, si supponga che i vari utenti richiedano livelli diversi di aggregazione (ossia una grana di tabella diversa). Le prestazioni possono essere migliorate definendo una tabella di fatti diversa per ciascun livello di aggregazione. Lo svantaggio più ovvio è che con l’aggiunta delle nuove tabelle di fatti, le esigenze di memorizzazione possono aumentare radicalmente. In genere le tabelle multiple di fatti servono a memorizzare i fatti per combinazioni di dimensioni diverse. La Figura 12.17 illustra un tipico esempio di tabelle di fatti multiple con due schemi a stella correlati. In questo esempio, ci sono due tabelle di fatti, una al centro di ciascuna stella.

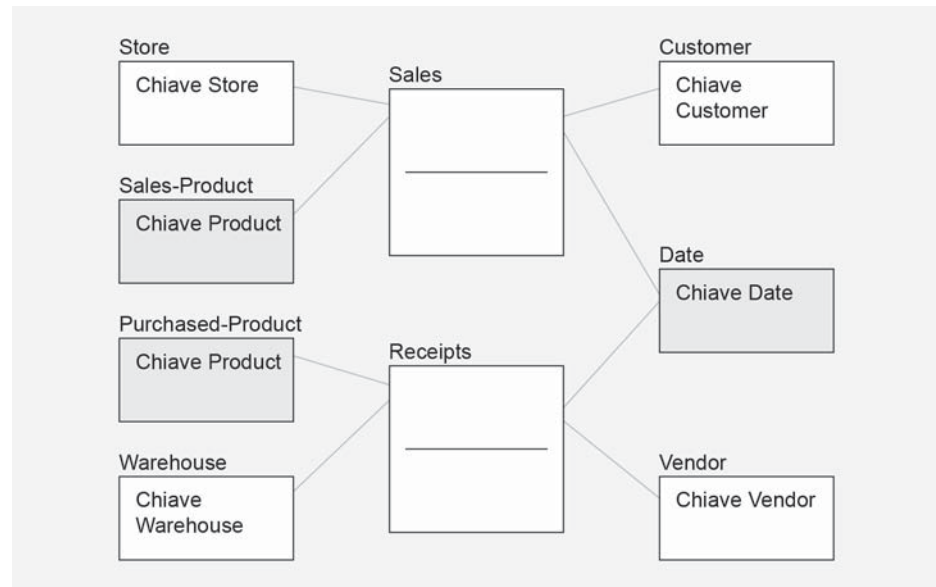
1. Sales: fatti sulla vendita di un prodotto a un cliente in un negozio in una determinata data.
2. Receipts: fatti sulla ricevuta di un prodotto da un produttore a un magazzino in una determinata data.

Com’è comune in queste situazioni, i dati su uno o più oggetti aziendali (in questo caso Product e Date) devono essere memorizzati in tabelle di dimensioni per ognuna delle tabelle di fatti. Per gestire le tabelle di dimensioni condivise, sono stati adottati due approcci. In un caso, poiché la descrizione di un prodotto è piuttosto diversa da quella delle vendite e delle ricevute, sono state create due tabelle di dimensioni per il prodotto. Dall’altro lato, poiché gli utenti desiderano le stesse descrizioni delle date, viene utilizzata una tabella di dimensioni. In ognuno dei due casi, abbiamo creato una dimensione conforme, dove la dimensione significa la stessa cosa con ogni tabella di fatti, e quindi utilizza le stesse chiavi primarie surrogate. Anche quando i due schemi a stella sono memorizzati in data mart fisici distinti, se le dimensioni sono conformi, si crea un potenziale per porre domande tra i data mart (per esem-

pio, alcuni produttori riconoscono le vendite più rapidamente e sono in grado di procurare i rifornimenti in meno tempo). In generale, le dimensioni conformi permettono all'utente quanto segue.

- Condividere dati di dimensione non di chiave.
- Eseguire interrogazioni tra tabelle di fatti mantenendo la coerenza.
- Lavorare sui fatti e gli oggetti aziendali il cui significato è condiviso da tutti gli utenti

**Figura 12.17**  
Dimensioni conformi.

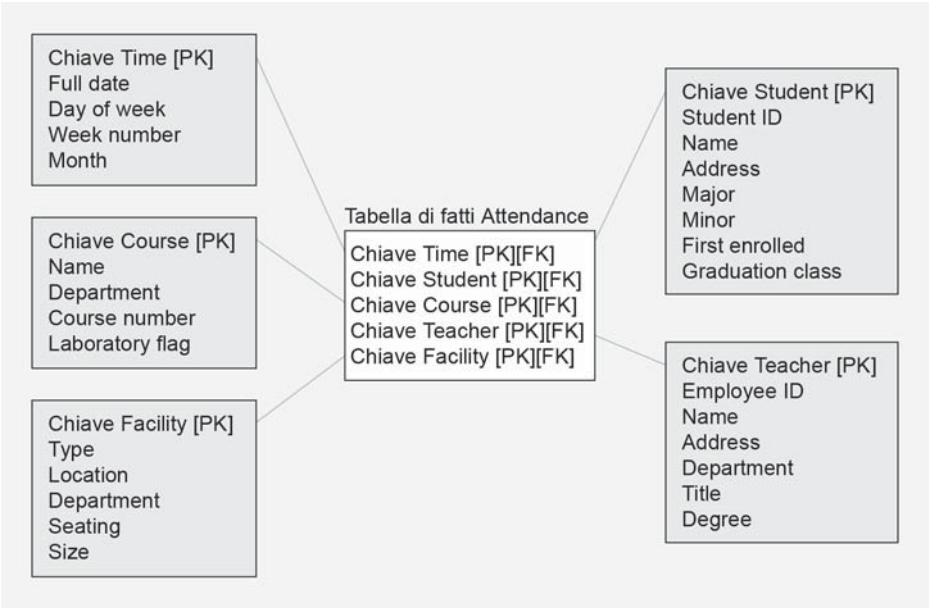


### Tabelle di fatti prive di fatti

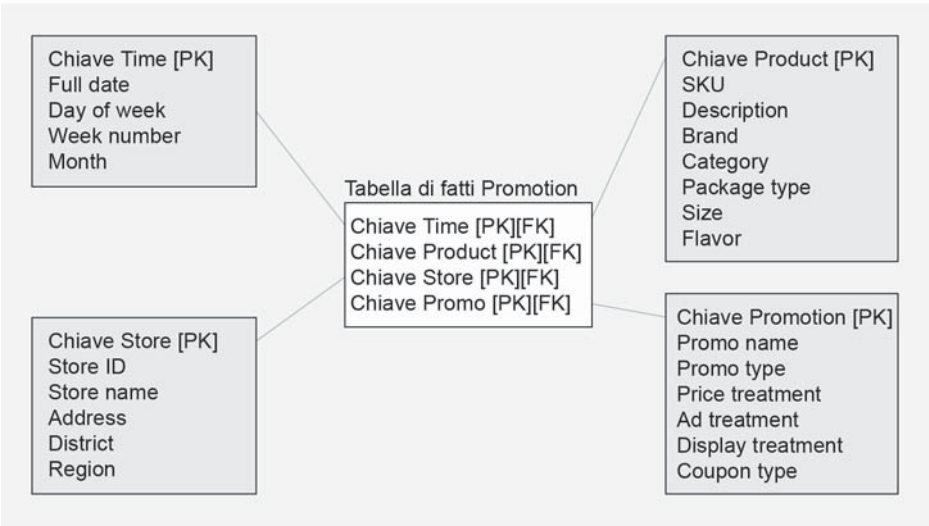
Esistono applicazioni per tabelle di fatti senza dati non di chiave, e solo con le chiavi esterne per le dimensioni associate. Le due situazioni in cui possono essere applicate tabelle di fatti prive di fatti sono quando si tiene una traccia degli eventi (Figura 12.18a) e quando si elenca l'insieme delle occorrenze possibili, o *coverage* (Figura 12.18b). Lo schema a stella nella Figura 12.18a tiene traccia di quali studenti frequentano quali corsi in quale momento in quali spazi e con quali insegnanti. Tutto quello che occorre sapere è se l'evento avviene, come rappresentato dall'intersezione delle cinque chiavi esterne. Lo schema a stella nella Figura 12.18b mostra l'insieme delle vendite possibili di un prodotto in un negozio in un momento particolare durante una promozione data. Una seconda tabella di fatti di vendita, non mostrata nella figura, potrebbe contenere le vendite in dollari e in unità (fatti) per la stessa combinazione di dimensioni (per esempio, con le stesse quattro chiavi esterne come nella tabella di fatti Promotion oltre a questi due fatti non di chiave). Con queste due tabelle di fatti e quattro dimensioni conformi, è possibile scoprire quali prodotti che erano in una data promozione in un tempo specifico in un dato negozio non hanno venduto (per esempio, hanno avuto vendite pari a zero), cosa che si può scoprire trovando una combinazione dei quattro valori di chiave nella tabella di fatti della promozione che non si trovano in quella delle vendite. Quest'ultima tabella, da sola, non è sufficiente a rispondere alla domanda, poiché le mancano le righe per una combinazione dei quattro valori di chiave, che ha vendite pari a zero.

**Figura 12.18**

Tabella di fatti priva di fatti.  
(a) Tabella di fatti priva di fatti  
che mostra il verificarsi  
di un evento.



(b) Tabella di fatti priva di fatti  
che mostra il coverage.



### 12.6.4    Normalizzare le tabelle di dimensioni

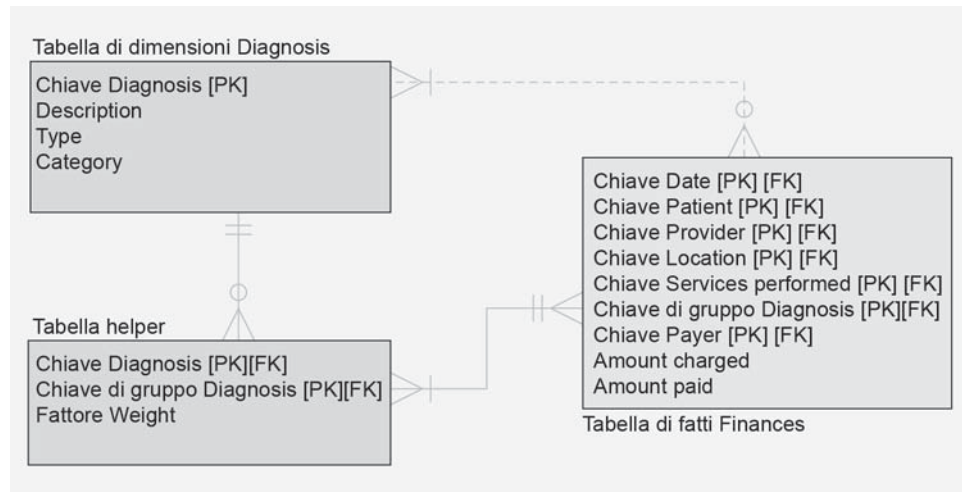
Le tabelle di fatti sono completamente normalizzate perché ogni fatto dipende dalla chiave primaria composta complessa e da nient'altro se non dalla chiave composta. D'altra parte, queste tabelle possono non essere normalizzate. Molti degli esperti di data warehousing lo ritengono accettabile nel caso di data mart ottimizzati e semplificati per un dato gruppo di utenti, così che tutti di dati di dimensione si trovano a un solo join dai fatti associati. A volte, come con qualsiasi base di dati relazionale, le anomalie delle tabelle di dimensioni normalizzate comportano problemi di aggiunta, aggiornamento ed elimina-

zione. In questo paragrafo affronteremo le varie situazioni in cui ha senso o è essenziale normalizzare ulteriormente le tabelle di dimensioni.

### Dimensioni multivalore

Può accadere che sia necessario che i fatti vengano qualificati da un insieme di valori per lo stesso oggetto aziendale. Per esempio, si consideri il caso dell'ospedale mostrato nella Figura 12.19. In questo esempio, una particolare tariffa e pagamento per un paziente in una data (per esempio per tutte le chiavi esterne nella tabella Finances) è associata a una o più diagnosi (come indicato dalla linea di associazione M:N tratteggiata tra le tabelle Diagnosis e Finances). Potremmo scegliere la diagnosi più importante come chiave componente della tabella Finances, ma questo vorrebbe dire perdere potenzialmente informazioni importanti sulle altre diagnosi associate a una riga. Oppure potremmo indicare la tabella Finances con un numero fisso di chiavi di diagnosi, più di quante sia possibile associare a una riga della tabella; questo creerebbe però componenti nulli della chiave primaria per molte righe, cosa che viola una proprietà delle basi di dati relazionali. L'approccio migliore, ovvero quello della normalizzazione, consiste nel creare una tabella per un'entità associativa tra Diagnosis e Finances. Nel mondo dei data warehouse una tabella di questo tipo è detta *tabella helper*; ne vedremo alcuni esempi nei prossimi paragrafi. Una tabella helper può avere attributi non di chiave (come qualsiasi tabella per un'entità non associativa); per esempio, il fattore del peso nella tabella del gruppo Diagnosis della Figura 12.19 indica il ruolo relativo giocato da ciascuna diagnosi in ciascun gruppo, presumibilmente normalizzato a un totale del 100 percento di tutte le diagnosi in un gruppo. Si noti inoltre che è possibile che più di una riga di Finances sia associata allo stesso gruppo Diagnosis.

**Figura 12.19**  
Dimensione multivalore.



### Gerarchie

Accade spesso che una dimensione in uno schema a stella formi una gerarchia naturale. Per esempio, ci sono gerarchie geografiche (mercati in una regione, regioni in uno Stato e Stati in un continente) e di prodotto (confezioni in un prodotto, prodotti in pacchetti e pacchetti in gruppi di prodotti). Quando una dimensione partecipa a una gerarchia, il progettista della base di dati ha due opportunità.

1. Includere tutte le informazioni per ciascun livello della gerarchia in un'unica tabella di dimensioni denormalizzata, creando così una notevole ridondanza e anomalie di aggiornamento.
2. Normalizzare la dimensione in un insieme di tabelle nidificate con un'associazione 1:M tra di esse. È ancora possibile aggregare i dati dei fatti a qualsiasi livello della gerarchia, ma ora l'utente dovrà effettuare unioni nidificate lungo la gerarchia, oppure gli si può fornire una vista pre-unità della gerarchia.

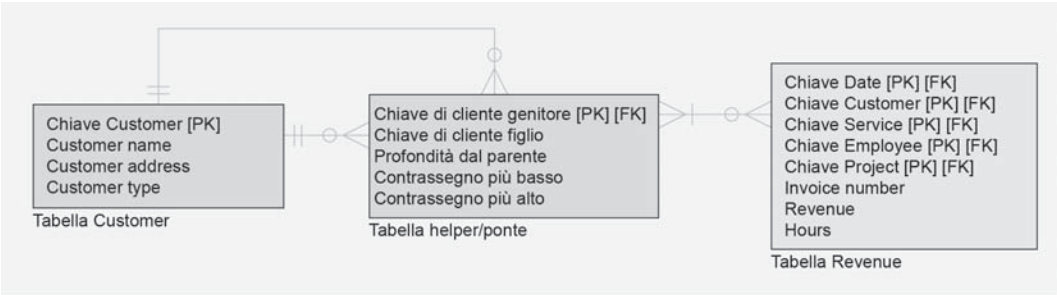
Si consideri l'esempio di una società di consulenza che fattura i progetti ai clienti in base a periodi di tempo. Una tabella di fatti dei ricavi potrebbe mostrare il ricavo fatturato per il numero di ore su ogni fattura, che riguarda un determinato periodo di tempo, cliente, servizio, impiegato e progetto. Poiché il lavoro di consulenza può essere effettuato per divisioni diverse della stessa azienda, se si vuole comprendere il ruolo complessivo della consulenza a qualsiasi livello di un'azienda cliente, occorrerà una gerarchia di cliente. Questa gerarchia è un'associazione ricorsiva tra unità dell'azienda; come mostrato nella Figura 7.17, il modo standard per rappresentare questa situazione in una base di dati normalizzata è quella di collocare nella riga della società una chiave esterna della chiave Company per la sua unità genitore.

Le associazioni ricorsive così implementate risultano difficili per l'utente finale che prende le decisioni, perché specificare come aggregare a qualsiasi livello della gerarchia richiede una programmazione SQL complessa. La Figura 12.20 mostra un'alternativa più semplice; la Figura 12.20a illustra come questa gerarchia sia solitamente modellata in un data warehouse che utilizza una tabella helper (Kimball, 1998b; Chisholm, 2000). A ogni unità dell'azienda cliente servita dalla società di consulenza è assegnata una chiave cliente surrogata diversa e una riga nella tabella di dimensioni Customer, e questa chiave è utilizzata come chiave esterna nella tabella di fatti Revenue. Il problema con l'unione in un'associazione ricorsiva con una profondità arbitraria è che l'utente deve scrivere il codice per unire un numero arbitrario di volte (una volta per ciascun livello di subordinazione) e che questi join in un data warehouse, a causa delle notevoli dimensioni di quest'ultimo, possono richiedere molto tempo (tranne che nel caso di tecnologie a prestazioni elevate che utilizzano l'elaborazione parallela). Per evitare questo problema, la tabella helper uniforma la gerarchia registrando una riga per ogni sotto-unità dell'azienda e ognuna delle sue unità superiori (compresa se stessa) fino all'unità più alta dell'azienda. Ogni riga di questa tabella helper ha tre descrittori: il numero di livelli di distanza della sotto-unità dall'unità superiore per quella riga di tabella, un contrassegno che indica se la sotto-unità è la più bassa nella gerarchia e uno che indica se è la più alta. La Figura 12.20b riproduce una gerarchia di un'azienda d'esempio e le righe che dovrebbero trovarsi nella tabella helper per rappresentare l'azienda nel suo complesso (ci dovrebbero essere altre righe per le associazioni sotto-unità-unità genitore nelle altre aziende cliente).

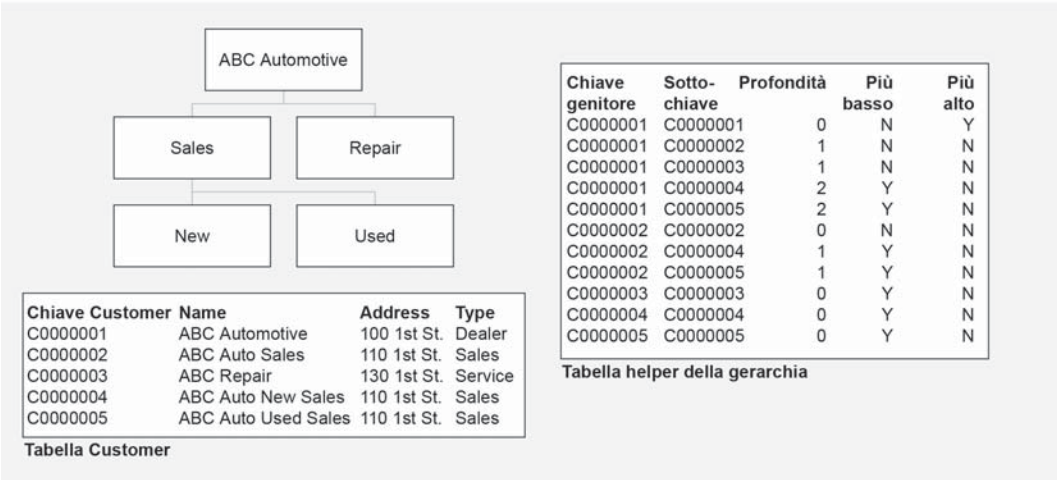
La tabella di fatti Revenue nella Figura 12.20a include un attributo non di chiave di Invoice number; questo è un esempio della cosiddetta *dimensione generativa*, che non ha attributi di dimensione interessanti (non esistono quindi tabelle di dimensione e Invoice number non fa parte della chiave primaria della tabella), ma non è nemmeno un fatto che verrà utilizzato per l'aggregazione. Questo attributo può rivelarsi utile quando occorre esplorare un data store operativo o i sistemi sorgente per trovare dettagli supplementari sulla transazione delle fatture.

Quando le tabelle di dimensione vengono ulteriormente normalizzate mediante le tabelle helper (chiamate a volte *tabelle ponte* o *tabelle di riferimento*), lo schema a stella semplice si trasforma in uno schema a snowflake (ovvero a *fiocco di neve*). Uno schema a snowflake assomiglia a un segmento di un data store operativo o di una base di dati centrati sulle tabelle di transazione riepilogate nella tabella di fatti e in tutte le tabelle direttamente o indirettamente ad esse correlate. Gli esperti di warehouse scoraggiano l'uso degli schemi a snowflake perché sono più complessi per gli utenti e perché richiedono più unioni per raggruppare i risultati in un'unica tabella. Uno schema di questo tipo può essere adatto

**Figura 12.20**  
Rappresentazione  
delle associazioni  
gerarchiche  
in una dimensione.  
(a) Uso di una tabella  
helper.



(b) Esempio  
di gerarchia con tabelle  
di cliente e helper.



qualora la normalizzazione risparmi uno spazio ridondante significativo (per esempio, quando ci sono molti attributi ridondanti con un testo lungo) o quando gli utenti possono trovare utile navigare tra le tabelle normalizzate.

### 12.6.5 Dimensioni a cambiamento lento

Come si ricorderà, i data warehouse e i data mart tengono una traccia delle attività aziendali nel tempo. L'azienda non rimane statica: i prodotti cambiano di dimensione e peso, i clienti si spostano, i negozi cambiano tipologia e il personale di vendita è assegnato a luoghi diversi. La maggior parte dei sistemi mantiene solo i valori correnti per gli oggetti aziendali (per esempio, l'indirizzo corrente dei clienti), ma in un data warehouse o data mart occorre conoscere la storia dei valori per trovare le corrispondenze con la storia dei fatti con le descrizioni dimensionali corrette nel momento in cui si sono verificati i fatti.

Per esempio, può accadere di dover associare un fatto di vendita alla descrizione del cliente relativo durante il periodo di tempo del fatto, che potrebbe non corrispondere alla descrizione attuale di quel cliente. Naturalmente gli oggetti aziendali cambiano lentamente rispetto alla maggior parte delle transazioni operazionali. I dati dimensionali cambiano, ma lentamente.

Per gestire gli attributi di dimensione che cambiano si possono adottare tre approcci (Kimball, 1996b e 1999).

1. Sovrascrivere il valore corrente con quello nuovo, procedura però inaccettabile perché elimina la descrizione del passato che invece è necessaria per interpretare i fatti storici.
2. Per ogni attributo di dimensione che cambia, creare un campo di valore corrente e tanti campi per i valori vecchi quanti se ne desiderano (per esempio, un attributo multivalore con un numero fisso di occorrenze). Questo schema potrebbe funzionare se fosse possibile prevedere il numero di cambiamenti lungo la storia conservata nel data warehouse (per esempio, se si conservassero 24 mesi di storia e un attributo cambiasse il proprio valore almeno mensilmente). Questo approccio funziona però solo con questo presupposto restrittivo e non può essere esteso in modo indiscriminato a qualsiasi attributo di dimensione che cambia lentamente.
3. Creare una nuova riga di dimensione nella tabella (con una nuova chiave) ogni volta che l'oggetto di dimensione cambia; questa nuova riga conterrà tutte le caratteristiche dimensionali. Una riga di fatti è associata alla chiave i cui attributi si applicano al momento del fatto. Si potrebbe anche voler memorizzare la data/ora in cui si è verificato il cambiamento e un codice che indichi il tipo di cambiamento. Questo approccio permette di creare tanti oggetti di dimensione quante ne occorrono, ma non è adatto se le righe sono troppo lunghe o se cambiano spesso.

Il terzo schema è quello utilizzato più di frequente. In base a questo approccio, si può memorizzare nella riga della dimensione anche il valore di chiave surrogata per l'oggetto originale, così da poter correlare tutte le modifiche alla loro origine.

Come visto in precedenza, tuttavia, questo schema può comportare la crescita nel numero delle righe delle dimensioni qualora i cambiamenti negli oggetti siano frequenti o quando le dimensioni sono molto grandi (*dimensioni mostro*). La Figura 12.21 illustra l'approccio della segmentazione delle dimensioni, che gestisce questa situazione e quella più generale dei sottoinsiemi di attributi di dimensione che cambiano con frequenze diverse. In questo esempio, la dimensione Customer è segmentata in due tabelle di dimensioni; un segmento contiene quasi ogni dimensione costante o che cambia lentamente, mentre gli altri segmenti contengono i cluster di attributi che cambiano più rapidamente e, nel caso di attributi nello stesso cluster, che cambiano spesso nello stesso momento. Questi attributi a cambiamento rapido sono spesso chiamati *caldi* (*hot*).

Un altro aspetto della segmentazione è che nel caso degli attributi caldi abbiamo modificato i singoli attributi di dimensione, come le entrate di un cliente (per esempio, 75.400 \$ l'anno) in uno per una *banda*, o intervallo, di valori di entrata (per esempio, 60.000 \$-89.999 \$ l'anno). Le bande vengono definite come richiesto dagli utenti e sono larghe o strette a seconda delle esigenze, anche se la precisione è parzialmente compromessa. Le bande rendono gli attributi caldi meno mutevoli, perché una modifica in una banda non comporta il bisogno di riscrivere una nuova riga. Questa struttura risulta più complessa agli utenti, perché questi devono unire i fatti con i segmenti di dimensione multipli, a seconda dell'analisi.

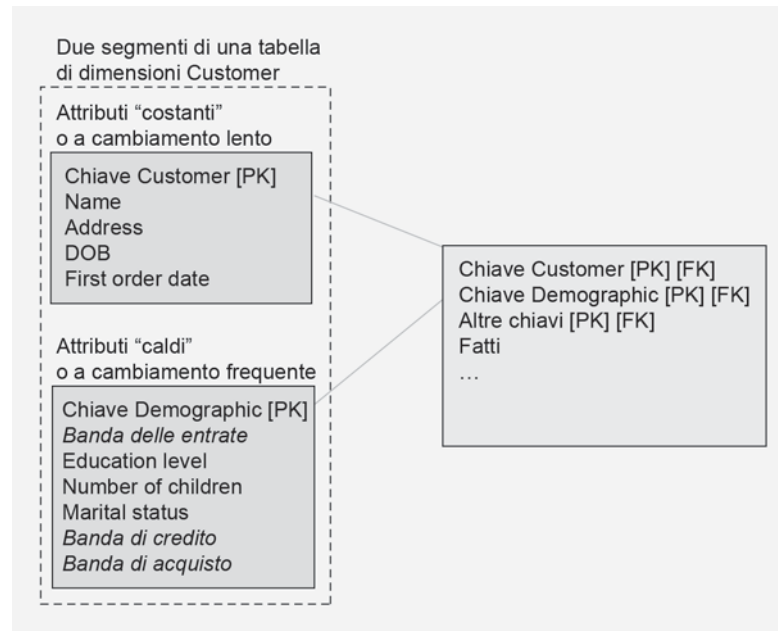
## 12.7 L'interfaccia utente

Anche un data mart ben progettato, caricato dei dati importanti, può restare inutilizzato a meno che gli utenti non vengano dotati di un'interfaccia potente e intuitiva che permetta loro di accedere facilmente ai dati e di analizzarli. In questo paragrafo forniremo una breve introduzione alle interfacce più moderne per i data warehouse.

Per interrogare e analizzare i dati memorizzati nei data warehouse e nei data mart sono disponibili numerosi strumenti, che possono essere classificati come segue.

**Figura 12.21**

Segmentazione delle dimensioni.



1. Strumenti di interrogazione e di report tradizionali.
2. Strumenti OLAP (*On-Line Analytical Processing*), MOLAP e ROLAP.
3. Strumenti di data mining.
4. Strumenti di visualizzazione dei dati.

Gli strumenti di interrogazione e di report tradizionali includono i fogli elettronici, le basi di dati su personal computer e i generatori e scrittori di report. Per ragioni di spazio non li descriveremo in questo capitolo, mentre approfondiremo le altre tre categorie di strumenti dopo aver considerato il ruolo dei metadati.

## 12.7.1 Ruolo dei metadati

Il primo requisito per costruire un'interfaccia user-friendly è quello di impostare un insieme di metadati che descrivano i dati contenuti nel mart in termini aziendali facilmente comprensibili dagli utenti.

I metadati associati ai data mart sono chiamati *catalogo dei dati*, *directory dei dati* o in altri modi analoghi. I metadati sono una sorta di "pagine gialle" dei dati contenuti nei mart e dovrebbero aiutare gli utenti a rispondere facilmente a domande come le seguenti.

1. Quali oggetti sono descritti nel data mart? (Oggetti tipici sono i clienti, i pazienti, gli studenti, i prodotti, i corsi e così via.)
2. Quali dimensioni e fatti sono inclusi nel data mart? Qual è la grana della tabella di fatti?
3. Come i dati nel mart sono derivati dai dati del warehouse aziendale? Quali regole vengono utilizzate nella derivazione?



4. Come i dati nel warehouse aziendale sono derivati dai dati operazionali? Quali regole vengono utilizzate nella derivazione?
5. Quali report e interrogazioni predefinite sono disponibili per visualizzare i dati?
6. Quale drill-down e altre tecniche di analisi dei dati sono disponibili?
7. Chi è il responsabile della qualità dei dati nei data mart e a chi si devono proporre le richieste di modifica?

## 12.7.2 Strumenti di interrogazione

Il linguaggio di interrogazione delle basi di dati più diffuso, ovvero SQL (trattato nei Capitoli 9 e 10), è sottoposto a una serie di estensioni per il supporto di alcuni tipi di calcoli e interrogazioni necessari in un ambiente di data warehouse. SQL non si può però considerare un linguaggio analitico (Mundy, 2001). SQL-99 include alcuni estensioni per i data warehouse e in particolare la clausola WINDOW per definire gli insiemi dinamici di righe, proprio perché le operazioni dei warehouse hanno a che fare con oggetti, eventualmente ordinati per data. Per esempio, una clausola WINDOW può essere utilizzata per definire tre giorni adiacenti come base per il calcolo delle medie mobili (si immagini una finestra che si sposta tra la parte superiore e inferiore del proprio bordo offrendo una visualizzazione scorrevole delle righe dei dati). PARTITION in una clausola WINDOW è simile a GROUP BY; PARTITION comunica alla clausola WINDOW la base per ciascun insieme, una clausola ORDER BY mette in sequenza gli elementi di un insieme e una clausola ROWS dice quante righe in una sequenza utilizzare in un calcolo. La funzione RANK calcola un risultato molto difficile da calcolare in SQL standard, ovvero la riga di una tabella in una posizione relativa specifica in base a un determinato criterio (per esempio, il cliente con le terze vendite più alte in un dato periodo). La clausola WINDOW è introdotta nel Capitolo 10. SQL-99 non è ancora uno strumento completo per l'analisi e l'interrogazione dei data warehouse, ma è un punto di partenza nel riconoscere che esistono esigenze di interrogazione speciali dei sistemi di supporto alle decisioni.

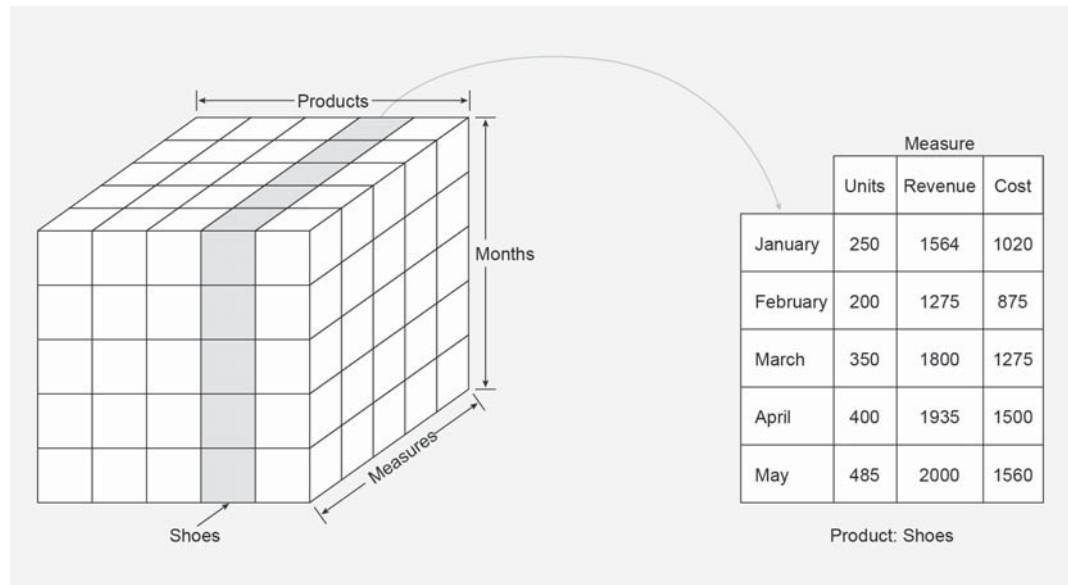
## 12.7.3 Strumenti OLAP (On-Line Analytical Processing)

Gli strumenti **OLAP** (*On-Line Analytical Processing*) forniscono agli utenti viste multidimensionali dei loro dati che facilitino le analisi grazie a un'interfaccia grafica. Nel caso più semplice, i dati vengono visualizzati come un semplice cubo a tre dimensioni. Il termine *elaborazione analitica on-line* (*on-line analytical processing*), si oppone al termine più tradizionale di *elaborazione delle transazioni on-line*. Le differenze tra i due tipi di elaborazione sono riassunte nella Tabella 12.1. Il termine *analisi multidimensionale* viene spesso utilizzato come sinonimo di OLAP.

Un esempio di “cubo di dati” (o vista multidimensionale) tipico di OLAP è mostrata nella Figura 12.22. Questa vista corrisponde piuttosto da vicino allo schema a stella presentato nella Figura 12.14. Due delle dimensioni nella Figura 12.22 corrispondono alle tabelle di dimensioni (PRODUCT e PERIOD) nella Figura 12.14, mentre la terza dimensione (le misure con nome) corrisponde ai dati nella tabella di fatti (SALES) nella Figura 12.14.

OLAP è un termine generico che comprende in realtà numerose categorie di strumenti di accesso ai data warehouse e ai data mart (Dyché, 2000). Gli strumenti **ROLAP** (*Relational OLAP*) utilizzano varianti di SQL e visualizzano i dati come una base di dati relazionale tradizionale in uno schema a stella o in un altro insieme normalizzato o denormalizzato di tabelle. Gli strumenti ROLAP accedono

**Figura 12.22**  
Vista dei dati sotto  
forma di cubo.



direttamente ai data warehouse e ai data mart. Gli strumenti **MOLAP (Multidimensional OLAP)** caricano i dati in una struttura intermedia, solitamente un array a tre o più dimensioni. Illusteremo l'uso di MOLAP nei prossimi paragrafi, considerata la sua grande diffusione. È importante notare che con MOLAP i dati non vengono semplicemente visualizzati come array multidimensionali (il cubo per tre dimensioni), bensì viene creato un data mart OLAP estraendo i dati da un data warehouse o data mart e memorizzando quindi i dati in un data store separato e specializzato attraverso il quale i dati possono essere visti solo in una struttura multidimensionale. Altre categorie meno diffuse di strumenti OLAP sono DOLAP (*Database OLAP*), che include la funzionalità OLAP nel linguaggio di interrogazione del DBMS (esistono sistemi proprietari SQL di standard non ANSI che lo fanno), e HOLAP (*Hybrid OLAP*), che permette l'accesso sia mediante i cubi multidimensionali sia attraverso i linguaggi di interrogazione relazionali.

### Silicing di un cubo

La Figura 12.22 mostra una tipica operazione MOLAP, ovvero lo *slicing* del cubo di dati per produrre una semplice tabella o vista a due dimensioni. Nella figura, l'operazione riguarda il prodotto chiamato "Shoes". La tabella che ne risulta mostra tre misure (Units, Revenues e Cost) per questo prodotto in base al periodo (o mese). Altre viste possono essere facilmente sviluppate dall'utente per mezzo di semplici operazioni "trascina e rilascia", dette anche *slicing and dicing* di un cubo.

Un'altra operazione strettamente correlata allo *slicing and dicing* è il *pivoting* dei dati. Questo termine fa riferimento alla rotazione della vista per un particolare punto dei dati in modo da ottenere un'altra prospettiva. Per esempio, la Figura 12.22 mostra le vendite per 400 paia di scarpe per il mese di aprile. L'analista può ruotare la vista in modo da ottenere, per esempio, le vendite delle scarpe in base al negozio per lo stesso mese.

### Drill-down

Un altro tipo di operazione tipica delle analisi multidimensionali è il *drill-down*, ovvero l'analisi di un insieme di dati con un livello di dettaglio sempre più elevato. Un esempio di drill-down è mostrato nella

Figura 12.23. La Figura 12.23a mostra un report di riepilogo per le vendite totali di tre tipi di confezione diversi per una data marca di salviettine di carta (2 pacchi, 3 pacchi e 6 pacchi). All'interno di ciascun tipo di confezione, le salviettine sono però di diverso colore. Utilizzando uno strumento OLAP, si può rintracciare rapidamente questa caratteristica utilizzando un approccio “punta e fai clic” con un mouse.

Il risultato del drill-down è mostrato nella Figura 12.23b. Si noti che una presentazione di drill-down equivale all'aggiunta di un'altra colonna al report originale (in questo caso è stata aggiunta una colonna per l'attributo del colore).

L'esecuzione di un drill-down può richiedere che lo strumento OLAP acceda al data warehouse per ottenere i dati dettagliati necessari all'operazione. La cosa è possibile solo se per lo strumento è disponibile uno insieme di metadati integrato. Alcuni strumenti permettono anche che lo strumento OLAP acceda ai dati operazionali se necessario per l'interrogazione.

**Figura 12.23**  
Esempio di drill-down.  
(a) Report di riepilogo.

| Brand    | Package size | Sales |
|----------|--------------|-------|
| SofTowel | 2-pack       | \$75  |
| SofTowel | 3-pack       | \$100 |
| SofTowel | 6-pack       | \$50  |

(b) Drill-down  
con aggiunti i colori.

| Brand    | Package size | Color  | Sales |
|----------|--------------|--------|-------|
| SofTowel | 2-pack       | White  | \$30  |
| SofTowel | 2-pack       | Yellow | \$25  |
| SofTowel | 2-pack       | Pink   | \$20  |
| SofTowel | 3-pack       | White  | \$50  |
| SofTowel | 3-pack       | Green  | \$25  |
| SofTowel | 3-pack       | Yellow | \$25  |
| SofTowel | 6-pack       | White  | \$30  |
| SofTowel | 6-pack       | Yellow | \$20  |

## 12.7.4 Strumenti di data mining

Con l’elaborazione OLAP, gli utenti cercano risposte alle domande che loro stessi hanno sollevato, per esempio: “I costi delle cure sono maggiori per i coniugati o per i non coniugati?”. Con il data mining, gli utenti cercano invece pattern o tendenze in una raccolta di fatti e osservazioni. Il data mining è la scoperta della conoscenza attraverso un insieme sofisticato di tecniche provenienti dalla statistica, dall’intelligenza artificiale e dalla computer grafica (Weldon, 1996).

Il data mining deve essere come descritto di seguito.

- 1. *Esplorativo*, per spiegare alcuni eventi o condizioni osservati, per esempio il motivo per cui la vendita dei pickup è aumentata in Colorado.
- 2. *Confermativo*, per confermare un’ipotesi, per esempio se è più probabile che le famiglie con due redditi stipulino un’assicurazione sanitaria rispetto alle famiglie monoreddito.
- 3. *Esplorativo*, per analizzare i dati per relazioni nuove o non previste, per esempio come i pattern di spesa sono legati alle frodi con le carte di credito.

### Tecniche di data mining

Per il data mining sono utilizzate numerose tecniche diverse, come riepilogato nella Tabella 12.4. La scelta di una tecnica piuttosto che di un’altra dipende dalla natura dei dati da analizzare e dalle dimensioni dell’insieme di dati. Il data mining può essere eseguito tanto sui data mart quanto sui data warehouse aziendali (o su entrambi).

**Tabella 12.4** Tecniche di data mining.

| Tecnica                  | Funzione                                                                                  |
|--------------------------|-------------------------------------------------------------------------------------------|
| Ragionamento sui casi    | Trae le regole da esempi del mondo reale.                                                 |
| Scoperta delle regole    | Cerca pattern e correlazioni in insiemi di dati molto grandi.                             |
| Elaborazione dei segnali | Identifica gruppi di osservazioni con caratteristiche simili.                             |
| Reti neurali             | Sviluppa modelli di previsione in base a principi modellati su quelli del cervello umano. |
| Frattali                 | Comprime le basi di dati grandi senza perdere le informazioni.                            |

### Applicazioni di data mining

Le tecniche di data mining sono state utilizzate con successo per un’ampia gamma di applicazioni del mondo reale, come riepilogato nella Tabella 12.5. Le applicazioni di data mining stanno crescendo rapidamente per le ragioni elencate di seguito.

- 1. L’entità dei dati nei data warehouse e data mart sta crescendo in modo esponenziale; gli utenti necessitano quindi delle tecniche automatizzate fornite dagli strumenti di data mining per dedurre alcune informazioni da questi dati.
- 2. Vengono introdotti sempre più nuovi strumenti di data mining con potenzialità avanzate.
- 3. Pressioni sempre maggiori in termini di competitività stanno spingendo le società a fare un uso migliore delle informazioni e della conoscenza contenute nei dati.

**Tabella 12.5** Tecniche di data mining (adattato da Zaitz, 1997 e Diché, 2000).

| Tipo di applicazione                   | Esempio                                                                                                                                   |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Definizione dei soggetti               | Sviluppare profili di clienti di valore, dei rischi finanziari e delle frodi con carte di credito.                                        |
| Analisi delle tendenze degli affari    | Identificare i mercati con la crescita sopra (o sotto) la media.                                                                          |
| Obiettivo del marketing                | Identificare i clienti (o segmenti di clienti) per le attività promozionali.                                                              |
| Analisi d'uso                          | Identificare i pattern d'uso di prodotti e servizi.                                                                                       |
| Efficacia delle campagne               | Confrontare le strategie delle campagne a scopo di efficacia.                                                                             |
| Affinità dei prodotti                  | Identificare i prodotti che vengono acquistati in contemporanea o le caratteristiche degli acquirenti per determinati gruppi di prodotti. |
| Mantenimento e spostamento dei clienti | Esaminare il comportamento dei clienti che sono passati ai concorrenti per impedire l'abbandono da parte dei clienti rimasti.             |
| Analisi di redditività                 | Determinare quali clienti sono redditizi considerato l'insieme totale delle attività del cliente con l'azienda.                           |
| Analisi del valore dei clienti         | Determinare se i clienti di valore sono in fasi diverse della loro vita.                                                                  |
| Up-selling                             | Identificare i nuovi prodotti o servizi per vendere a un cliente in base a eventi cruciali e modifiche nel suo stile di vita.             |

### 12.7.5    Visualizzazione dei dati

L'occhio umano riesce a discernere meglio i pattern quando i dati sono rappresentati graficamente. La visualizzazione dei dati è la rappresentazione dei dati in formati grafici e multimediali adatti all'analisi umana. I vantaggi della visualizzazione dei dati includono la possibilità di osservare meglio tendenze e pattern e di identificare le correlazioni e i cluster; inoltre la visualizzazione viene spesso utilizzata in combinazione con il data mining e con altre tecniche analitiche.

## Riepilogo

Nonostante le grandi quantità di dati raccolte attualmente nelle aziende, la maggior parte dei manager ha difficoltà nell'ottenere le informazioni necessarie per prendere le decisioni. Due fattori principali contribuiscono a questo "gap informativo". Innanzitutto i dati sono spesso eterogenei e incoerenti come risultato di approcci frammentari dello sviluppo dei sistemi. In secondo luogo, i sistemi sono sviluppati (o acquistati) principalmente per soddisfare gli obiettivi operazionali, senza prendere in considerazione le esigenze informative dei manager.

Tra i sistemi operazionali e quelli informativi, e tra i dati che appaiono al loro interno, esistono alcune differenze sostanziali. I sistemi operazionali sono utilizzati per esercitare l'attività su una base corrente, e l'obiettivo principale della progettazione è quello di fornire prestazioni elevate agli utenti che elaborano le transazioni e aggiornano le basi di dati. I sistemi direzionali sono utilizzati per supportare le decisioni dei manager, e il loro obiettivo principale è quello di fornire facilità di accesso e d'uso a chi lavora con le informazioni.

Lo scopo di un data warehouse è quello di consolidare e integrare i dati da un'ampia gamma di sorgenti e di formattarli in un contesto per prendere decisioni aziendali accurate. Un data warehouse è uno store integrato e coerente di dati orientati all'oggetto tratti da varie sorgenti e formattati in un contesto significativo per supportare le decisioni in un'azienda.

La maggior parte dei data warehouse più moderni segue un'architettura a tre livelli. Il primo livello è costituito dai dati distribuiti attraverso i vari sistemi operazionali. Il secondo livello è un data warehouse aziendale, centralizzato e integrato, che funge da punto di controllo e sorgente unica di tutti i dati resi

disponibili agli utenti finali per le applicazioni di supporto alle decisioni. Il terzo livello è una serie di data mart. Un data mart è un data warehouse i cui dati hanno un ambito limitato per le esigenze decisionali di un particolare gruppo di utenti. Un data mart può essere indipendente da un data warehouse aziendale (EDW), derivato da un EDW o un sottoinsieme logico di un EDW.

Il livello dei dati nel data warehouse aziendale è chiamato livello dei dati riconciliati. Questo livello dovrebbe essere, idealmente, dettagliato, storico, normalizzato, comprensivo e controllato per quanto riguarda la qualità. I dati riconciliati sono ottenuti completando il data warehouse aziendale o il data store operativo dai vari sistemi operazionali. La riconciliazione dei dati richiede quattro passaggi: acquisire i dati dai sistemi sorgente; eseguire lo scrubbing dei dati per eliminare le incoerenze; trasformare i dati per convertirli nel formato richiesto dal data warehouse e caricare e indicizzare i dati nel warehouse. Ai dati riconciliati non è solitamente concesso l'accesso diretto da parte degli utenti. Il livello dei dati nei data mart è detto livello dei dati derivati; si tratta dei dati a cui possono accedere gli utenti finali per le loro applicazioni di supporto alle decisioni.

I dati vengono spesso memorizzati in un data mart utilizzando una variante del modello relazionale chiamato schema a stella, o modello dimensionale. Uno schema a stella è una semplice rappresentazione di una base di dati in cui i dati delle dimensioni sono separati da quelli dei fatti o degli eventi. Uno schema a stella è costituito da due tipi di tabelle: tabelle di dimensioni e tabelle di fatti. La dimensione di una tabella di fatti dipende in parte dalla sua grana (livello di dettaglio). Nelle applicazioni di data warehouse più moderne può essere presente anche più di un miliardo di righe. Lo schema a stella prevede numerose varianti, compresi i modelli con tabelle di fatti multiple e schemi a snowflake che sorgono quando una o più dimensioni hanno una struttura gerarchica.

Sono disponibili numerose interfacce utente per accedere e analizzare i dati di supporto alle decisioni. OLAP (*On-Line Analytical Processing*) è un insieme di strumenti grafici che forniscono agli utenti viste multidimensionali dei loro dati (spesso sotto forma di un cubo). OLAP facilita le operazioni di analisi dei dati, come lo slice and dice, il pivoting dei dati e il drill-down. Il data mining è una forma di scoperta della conoscenza che utilizza un insieme sofisticato di tecniche tratte dalla statistica, dall'intelligenza artificiale e dalla computer grafica.

Ripasso del capitolo

Parole chiave

|                                |                            |                               |                          |
|--------------------------------|----------------------------|-------------------------------|--------------------------|
| @ctive warehouse               | Data store operativo (ODS) | Estrazione progressiva        | (OLAP)                   |
| Aggregazione                   | Data warehouse             | Estrazione statica            | Schema a snowflake       |
| Analisi del paniere di mercato | Data warehouse aziendale   | Evento                        | Schema a stella          |
| Data mart                      | (EDW)                      | Grana                         | Selezione                |
| Data mart dipendente           | Dati derivati              | Modalità di aggiornamento     | Sistema operativo        |
| Data mart indipendente         | Dati persistenti           | Modalità di refresh           | Sistemi direzionali      |
| Data mart logico               | Dati riconciliati          | OLAP multidimensionale        | Trasformazione dei dati  |
| Data mining                    | Dati transienti            | OLAP relazionale              | Unione                   |
| Data scrubbing                 | Dimensione conforme        | On-Line Analytical Processing | Visualizzazione dei dati |

Domande di ripasso

1. Definire ciascuno dei seguenti termini chiave:

a. data warehouse

b. data mart

c. dati riconciliati

d. dati derivati

e. On-Line Analytical Processing

f. data mining

g. schema a stella

- h. schema a snowflake
  - i. grana
  - j. estrazione statica
  - k. estrazione progressiva
  - l. evento
2. Unire i seguenti termini alle rispettive definizioni:
- \_\_\_ evento; \_\_\_ dati persistenti; \_\_\_ data mart;
  - \_\_\_ schema a stella; \_\_\_ dati riconciliati;
  - \_\_\_ data mart dipendente; \_\_\_ visualizzazione dei dati;
  - \_\_\_ dati transienti; \_\_\_ evento; \_\_\_ schema a snowflake;
  - \_\_\_ data scrubbing; \_\_\_ schema a stella; \_\_\_ dati riconciliati;
  - \_\_\_ data mart dipendente; \_\_\_ visualizzazione dei dati;
  - \_\_\_ dati transienti.
- a. il contenuto precedente è andato perduto
  - b. dati cronologici dettagliati
  - c. converte i formati dei dati
  - d. corregge gli errori nei dati sorgente
  - e. i dati non vengono modificati né eliminati
  - f. un'azione sulla base di dati (per esempio, creare)
  - g. data warehouse con un ambito ristretto
  - h. tabelle di dimensioni e di fatti
  - i. forma di scoperta della conoscenza
  - j. riempiti dal data warehouse
  - k. deriva dalle dimensioni gerarchiche
  - l. dati rappresentati in forma grafica
3. Confrontare in seguenti termini:
- a. estrazione statica; estrazione progressiva
  - b. dati transienti; dati persistenti
  - c. data warehouse; data mart; data store operativa
  - d. data scrubbing; trasformazione dei dati
  - e. dati riconciliati; dati derivati
  - f. tabella di fatti; tabella di dimensioni
  - g. schema a stella; schema a snowflake
  - h. data mart indipendente; data mart dipendente; data mart logico
4. Elencare cinque tendenze principali che rendono necessario il data warehousing in molte delle aziende più moderne.
5. Descrivere brevemente i componenti principali di un'architettura di data warehouse.
6. Elencare tre tipi di metadati che appaiono in un'architettura di data warehouse a tre livelli e descrivere brevemente lo scopo di ciascuno.
7. Elencare cinque caratteristiche tipiche dei dati riconciliati.
8. Elencare e descrivere brevemente i quattro passi del processo di riconciliazione dei dati.
9. Elencare cinque errori e incoerenze facilmente riscontrabili nei dati operazionali.

10. Descrivere brevemente tre tipi di operazioni che possono essere eseguite facilmente con gli strumenti OLAP.
11. Spiegare come l'estrazione, la trasformazione e il caricamento si correlano al processo di riconciliazione dei dati.
12. Spiegare i pro e i contro/limiti dei data mart indipendenti.
13. Spiegare come la volatilità di un data warehouse è diversa da quella di una base di dati per un sistema informativo operativo.
14. Spiegare i pro e i contro dei data mart logici.
15. Elencare le operazioni comuni eseguite durante la pulizia dei dati (cleansing).
16. Descrivere le caratteristiche di una chiave surrogata così come viene utilizzata in un data warehouse o data mart.
17. Perché il tempo è quasi sempre una dimensione in un data warehouse o data mart?
18. Qual è lo scopo delle dimensioni conformi per i diversi schemi a stella nello stesso ambiente di data warehouse?
19. Può una tabella di fatti contenere attributi non di chiave? Perché sì o no?
20. In quali modi le tabelle di dimensioni sono spesso non normalizzate?
21. Cos'è una gerarchia rispetto a una tabella di dimensioni?
22. Qual è il significato della frase "dimensione a cambiamento lento" ?
23. Spiegare l'approccio più comune utilizzato per gestire le dimensioni a cambiamento lento.

**Problemi ed esercizi**

1. Esaminare le tre tabelle con i dati sugli studenti nella Figura 12.1. Progettare un unico formato di tabella che contenga tutti i dati (in modo non ridondante) che si trovano in queste tre tabelle. Scegliere i nomi di colonna che si ritengono più appropriati per questi dati.
2. La tabella riportata di seguito mostra alcuni semplici dati di studenti in data 20/06/2001.

| Key | Name | Major       |
|-----|------|-------------|
| 001 | Amy  | Music       |
| 002 | Tom  | Business    |
| 003 | Sue  | Art         |
| 004 | Joe  | Math        |
| 005 | Ann  | Engineering |

In data 21/06/2001 si verificano le seguenti transazioni.

- 1. Lo studente 004 cambia Major da 'Math' a 'Business'.
- 2. Lo studente 005 viene eliminato dal file.
- 3. Un nuovo studente 006 viene aggiunto al file. Il suo Name è 'Jim', il suo Major è 'Phys Ed'.

In data 22/06/2001 si verificano le seguenti transazioni.

- 1. Lo studente 003 cambia Major da 'Art' a 'History'.

2. Lo studente 006 cambia Major da 'Phys Ed' a 'Basket Weaving'.

L'esercitazione si suddivide in due parti.

- Costruire tabelle per le date 21/06/2001 e 22/06/2001 che riflettano le transazioni sopra elencate, presumendo che i dati siano transienti (Figura 12.8).
- Costruire tabelle per le date 21/06/2001 e 22/06/2001 che riflettano le transazioni sopra elencate, presumendo che i dati siano persistenti (Figura 12.9).

3. Il Millennium College chiede che si progetti uno schema a stella che registri i voti ottenuti ai corsi completati dagli studenti. Sono presenti quattro tabelle di dimensioni, con i seguenti attributi.

■ **Course\_Section.** Attributi: Course\_ID, Section\_Number, Course\_Name, Units, Room\_ID, Room\_Capacity. Durante un dato semestre, il college offre una media di 500 seminari di corso.

■ **Professor.** Attributi: Prof\_ID, Prof\_Name, Title, Department\_ID, Department\_Name.

■ **Student.** Attributi: Student\_ID, Student\_Name, Major. Ogni seminario di corso ha una media di 40 studenti.

■ **Period.** Attributi: Semester\_ID, Year. La base di dati conterrà i dati per 30 periodi (un totale di 10 anni).

Il solo fatto che deve essere registrato nella tabella di fatti è Course\_Grade.

- Progettare uno schema a stella per questo problema. Si veda la Figura 12.14 per il formato da seguire.
- Calcolare il numero di righe nella tabella di fatti utilizzando i presupposti sopra elencati.
- Calcolare le dimensioni totali della tabella di fatti (in byte), presumendo che ogni campo abbia una media di 5 byte.
- Le varie caratteristiche di seminari, professori e studenti cambiano nel tempo. Che tipo di schema a stella si può proporre per consentire queste modifiche? Perché?

4. In base ai principi della normalizzazione descritti nel Capitolo 7, si nota immediatamente che lo schema a stella sviluppato per il Millennium College (problema ed esercizio 3) non è nella terza forma normale. Utilizzando questi principi, convertire lo schema a stella in uno a snowflake. Quale impatto ha questo sulle dimensioni della tabella di fatti in questo problema?

5. Si immagini di costruire uno schema a stella per la Simplified Automobile Insurance Company (si veda Kimball, 1996 per un esempio più realistico). Le dimensioni importanti, gli attributi di dimensione e la loro entità sono i seguenti.

■ **Insured Party.** Attributi: Insured\_Party\_ID, Name. C'è una media di due parti assicurate per ogni polizza e voce coperta.

■ **Coverage Item.** Attributi: Coverage\_Key, Descrizione. C'è una media di 10 voci coperte per polizza.

■ **Agent.** Attributi: Agent\_ID, Agent\_Name. C'è un agente per ogni polizza e voce coperta.

■ **Policy.** Attributi: Policy\_ID, Type. Attualmente, la compagnia ha circa un milione di polizze.

■ **Period.** Attributi: Date\_Key, Fiscal\_Period.

I fatti da registrare per ogni combinazione di queste dimensioni sono Policy\_Premium, Deductible e Number\_of\_Transactions.

- Progettare uno schema a stella per questo problema. Si veda la Figura 12.14 per il formato da seguire.
- Calcolare il numero di righe nella tabella di fatti utilizzando i presupposti sopra elencati.
- Calcolare le dimensioni totali della tabella di fatti (in byte), presumendo che ogni campo abbia una media di 5 byte.

6. La Simplified Automobile Insurance Company vorrebbe aggiungere una dimensione Claims al suo schema a stella (problema ed esercizio 5). Gli attributi di Claim sono Claim\_ID, Claim\_Description e Claim\_Type. Gli attributi della tabella di fatti sono ora Policy\_Premium, Deductible e Monthly\_Claim\_Total.

- Estendere lo schema a stella del problema ed esercizio 5 per includere i nuovi dati.
- Calcolare il numero stimato di righe nella tabella di fatti presupponendo che la compagnia sperimenti una media di 200 richieste al mese.

7. Il Millennium College (problema ed esercizio 3) vuole includere dei nuovi dati sui seminari di corso: il dipartimento che offre il corso, l'unità accademica a cui il dipartimento fa riferimento e l'unità finanziaria a cui il dipartimento è assegnato. Modificare la risposta al problema ed esercizio in base a queste nuove esigenze. Spiegare perché si sono implementate quelle modifiche nello schema a stella.

8. Come citato nel capitolo, Kimball (1997), Inmon (1997 e 2000) e Armstrong (2000) hanno discusso i meriti dei data mart indipendenti e dipendenti e normalizzati rispetto ai dati mart denormalizzati. Procurarsi le copie di questi articoli in biblioteca oppure in Internet e riassumere le posizioni di ciascuna delle parti.

## Esercitazioni

- Visitare un'azienda che ha sviluppato un data warehouse e intervistare l'amministratore dei dati o altre persone con un ruolo chiave. Discutere i seguenti aspetti.
  - Qual è il livello di soddisfazione degli utenti rispetto al data warehouse? In che termini questo ha migliorato la loro capacità decisionale?
  - Il data warehouse utilizza un'architettura a due o a tre livelli?
  - L'architettura utilizza uno o più data mart? In caso affermativo, questi sono dipendenti o indipendenti?
  - Quali strumenti vengono utilizzati dagli utenti finali? Il data mining è utilizzato?
  - Quali sono stati gli ostacoli o le difficoltà principali da superare durante lo sviluppo dell'ambiente del data warehouse?



2. Visitare i seguenti siti Web per avere ulteriori informazioni sui data warehouse, compresi alcuni esempi pratici di implementazioni, descrizioni dei prodotti di data warehouse più recenti e comunicazioni su conferenze e altri eventi.
  - a. International Data Warehouse Association: <http://www.idwa.com>
  - b. The Data Warehousing Institute: <http://www.dw-institute.com>
  - c. Knowledge Discovery Mine: <http://www.kdnuggets.com>
  - d. Data Mining Institute: <http://www.datamining.org>
  - e. Data Warehousing Knowledge Center: <http://www.datawarehousing.org>
  - f. Una rivista elettronica sul data warehousing: <http://www.tdan.com>

## Bibliografia

Armstrong, R., 1997, "A Rebuttal to the Dimensional Modeling Manifesto", documento emesso da NCR Corporation.

Armstrong, R., 2000, "Avoiding Data Mart Traps", *Teradata Review* (estate), pp. 32–37.

Barquin, R., 1996, "On the First Issue of The Journal of Data Warehousing", *The Journal of Data Warehousing* 1 (luglio), pp. 2–6.

Chisholm, M., 2000, "A New Understanding of Reference Data", *DM Review* (ottobre), pp. 60, 84–85.

Devlin, B., 1997, *Data Warehouse: From Architecture to Implementation*, Reading, MA, Addison Wesley Longman.

Devlin, B. e Murphy, P., 1988, "An Architecture for a Business Information System", *IBM Systems Journal* 27 (No. 1), pp. 60–80.

Dyché, J., 2000, *e-Data: Turning Data into Information with Data Warehousing*, Reading, MA, Addison-Wesley.

English, L. P., 1999, *Improving Data Warehouse and Business Information Quality*, New York, John Wiley.

Griffin, J., 1997, "Fast Payback Projects", *DM Review* 7 (dicembre), pp. 10, 11.

Hackathorn, R., 1993, *Enterprise Database Connectivity*, New York, John Wiley & Sons.

IBM, 1993, "Information Warehouse Architecture", SC26–324–00, White Plains, NY, IBM Corp.

Imhoff, C., 1998, "The Operational Data Store: Hamering Away", *DM Review* (luglio), disponibile sul sito [www.dmreview.com](http://www.dmreview.com).

Imhoff, C., 1999, "The Corporate Information Factory", *DM Review* (dicembre), disponibile sul sito [www.dmreview.com](http://www.dmreview.com).

Inmon, B., 1997, "Iterative Development in the Data Warehouse", *DM Review* 7 (novembre), pp. 16, 17.

Inmon, W., 1992, *Building the Data Warehouse*, Wellesley, MA, QED Information Sciences.

Inmon, W., 1998, "The Operational Data Store: Designing the Operational Data Store", *DM Review* (luglio), disponibile sul sito [www.dmreview.com](http://www.dmreview.com).

Inmon, W., 1999, "What Happens When You Have Built the Data Mart First?", in *TDAN*, una rivista elettronica disponibile sul sito [www.tdan.com](http://www.tdan.com).

Inmon, W., 2000, "The Problem with Dimensional Modeling", *DM Review* (maggio), pp. 68–70.

Inmon, W. H. e Hackathorn, R. D., 1994, *Using the Data Warehouse*, New York, John Wiley & Sons.

Kimball, R., 1996a, *The Data Warehouse Toolkit*, New York, John Wiley & Sons.

Kimball, R., 1996b, "Slowly Changing Dimensions", *DBMS* (aprile), pp. 18–20.

Kimball, R., 1997, "A Dimensional Modeling Manifesto", *DBMS* (agosto), disponibile sul sito [www.dbmsmag.com](http://www.dbmsmag.com).

Kimball, R., 1998a, "Pipelining Your Surrogates" *DBMS* (giugno), pp. 18–22.

Kimball, R., 1998b, "Help for Hierarchies", *DBMS* (settembre), pp. 12–16.

Kimball, R., 1999, "When a Slowly Changing Dimension Speeds Up", *Intelligent Enterprise* (3 agosto), pp. 60–62.

Kimball, R., 2001, "Declaring the Grain", tratto da *Kimball University Design Tip #21* sul sito [www.ralphkimball.com](http://www.ralphkimball.com), verificato il 21 marzo 2001.

Martin, J., 1997a, "New Tools for Decision Making" *DM Review* 7 (settembre), p. 80.

Martin, J., 1997b, "Web Warehousing", *DM Review* 7 (novembre), p. 10.

Meyer, A., 1997, "The Case for Dependent Data Marts", *DM Review* 7 (luglio/agosto), pp. 17–24.

Moriarty, T., 1995, "Modeling Data Warehouses", *Database Programming & Design*, 8 (agosto), pp. 61–65.

Mundy, J., 2001, "Smarter Data Warehouses", *Intelligent Enterprise* (16 febbraio), pp. 24–29.

Poe, V., 1996, *Building a Data Warehouse for Decision Support*, Upper Saddle River, NJ, Prentice-Hall.

Strange, K., 1997, "Can Data Marts Grow?", *CIO* (luglio), pp. 34, 36.

Weldon, J. L., 1996, "Data Mining and Visualization", *Database Programming & Design* 9 (maggio), pp. 21–24.

Westerman, P., 2001, *Data Warehousing: Using the Wal-Mart Model*, San Francisco, CA, Morgan Kaufmann.

White, C., 2000, "First Analysis", *Intelligent Enterprise* (giugno), disponibile sul sito [www.intelligententerprise.com](http://www.intelligententerprise.com).

Williams, J., 1997, "Tools for Traveling Data", *DBMS* 10 (giugno), pp. 69–76.

Zaitz, J., 1997, "Data Mining Neural Clusters", *DM Review* 7 (luglio/agosto), pp. 91, 92.

## Per approfondire

Bischoff, J. e Alexander, T., 1997, *Data Warehouse: Practical Advice from the Experts*, Upper Saddle River, NJ, Prentice-Hall.

Goodhue, D., Mybo, M. e Kirsch, L., 1992, "The Impact of Data Integration on the Costs and Benefits of Information Systems", *MIS Quarterly* 16 (settembre), pp. 293–311.

Jenks, B., 1997, "Tiered Data Warehouse", *DM Review* 7 (ottobre), pp. 54–57.

## Project Case



# MOUNTAIN VIEW COMMUNITY HOSPITAL

### Descrizione del progetto

Sotto molti aspetti, Mountain View Community Hospital ha seguito un approccio attentamente pianificato per progettare, selezionare e installare i propri sistemi direzionali. L'azienda ha sviluppato un modello dei dati aziendali che funge da linea guida per lo sviluppo della base di dati (si vedano i Capitoli 5 e 6). L'ospedale ha installato sistemi di computer che supportano la maggior parte delle operazioni di routine dell'azienda. Per esempio, ci sono sistemi per la contabilità pazienti, per i servizi amministrativi e per la gestione finanziaria. La maggior parte di questi sistemi è stata acquistata da produttori esterni dopo un attento processo di selezione. Nonostante questa pianificazione attenta, il management è consapevole che nei sistemi informativi dell'ospedale siano presenti lacune e limiti. In particolare, sono stati rilevati due problemi.

1. I dati vengono spesso duplicati in file e basi di dati diversi, in formati diversi e anche in media diversi. Per esempio, un insieme di dati sui pazienti (utilizzato per scopi di fatturazione) risiede in un sistema di contabilità pazienti basato su una base di dati relazionale. Per contro, la maggior parte dei record medici sui pazienti è mantenuta in un sistema manuale (una cartella per paziente).
2. I sistemi sono concepiti principalmente per supportare l'elaborazione operativa (o di transazione), ma non sono molto adatti a fornire informazioni di gestione né a supportare gli studi analitici sempre più necessari per la gestione di un ospedale moderno.

L'industria sanitaria in generale (compresi gli ospedali come Mountain View Community) è guidata dalla tendenza a ottenere un accesso migliore e più centralizzato alle proprie informazioni cliniche, operazionali e finanziarie (Griffin, 1997). Un data warehouse tipico di un ospedale può contenere quattro tipi di dati: record sui pazienti; record su medici, cliniche e ospedali; record sulle aziende farmaceutiche; record sulle compagnie assicurative (Martin, 1997b).

Per affrontare questi problemi, i manager del Mountain View Community Hospital vorrebbero sapere se le tecniche di data warehousing possano essere applicate con successo all'azienda. Poiché l'ospedale non è molto grande, i manager si rendono subito con-

to che è necessario sviluppare un piccolo prototipo per verificare l'applicazione. Dopo un po' di studio, decidono di sviluppare due piccoli data mart prototipo:

1. un data mart che registri le informazioni di riepilogo sui test e le procedure eseguite dai medici dell'ospedale;
2. un data mart più dettagliato che registri le informazioni di riepilogo sugli esami e le procedure eseguite dai medici per i singoli pazienti.

### Domande sul progetto

1. Quali sono alcuni dei vantaggi che un ospedale come Mountain View Community può trarre da un data warehouse e/o data mart?
2. Si supponga che un ospedale decida di sviluppare un data mart utilizzando uno schema a stella. Elencare le numerose dimensioni che si presume debbano essere rappresentate in questo schema.
3. Mountain View Community Hospital dovrebbe considerare l'ipotesi di sviluppare un data mart come quello proposto anche senza avere un data warehouse solido? Quali sono i rischi associati a questo approccio? Si ritiene che un'azienda possa sviluppare un data mart prototipo per indagare senza avere un warehouse solido? Discutere alcuni dei probabili vantaggi e svantaggi di questo approccio.

### Esercizi sul progetto

1. Data mart di riepilogo.
  - a. Progettare uno schema a stella.  
Dimensione Physician:  
Physician\_ID (pk): 5 byte  
Physician\_Name: 10 byte  
Specialty: 5 byte  
Physician\_Address: 10 byte  
Physician\_Telephone: 5 byte  
Dimensione Treatment:  
Treatment\_ID (pk): 3 byte

Treatment\_Description: 6 byte

Dimensione Period:

Period\_ID (pk): 2 byte

Month: 1 byte

Year: 2 byte

Tabella di fatti Treatment:

Grana: riepilogo mensile dei trattamenti e costi medi degli stessi in base al medico e al trattamento.

Monthly\_Total: 3 byte

Average\_Cost: 5 byte

- b. Calcolare il numero di righe previsto nella tabella di fatti e la dimensione stimata (byte).

Presupposti.

1. Trattamenti: circa 500 trattamenti diversi vengono effettuati presso l'ospedale. In un mese tipico, ne viene eseguito circa il 30 per cento (o 150 trattamenti diversi).
2. Medici: ogni trattamento è effettuato da un medico.
3. Periodi: se viene sviluppato un data mart su vasta scala, si anticipa che verranno accumulati 36 periodi (o 3 anni) di dati.

2. Data mart dettagliato.

- a. Progettare uno schema a stella.

Dimensione Physician: come sopra

Dimensione Treatment: come sopra

Dimensione Period:

Period\_ID (pk): 3 byte

Date: 5 byte

Dimensione Patient:

Patient\_ID (pk): 5 byte

Patient\_Name: 10 byte

Patient\_Address: 10 byte

Patient\_Telephone: 5 byte

Tabella di fatti Treatment:

Grana: dettaglio per ogni occorrenza di trattamento effettuato su un paziente da un medico.

Treatment\_Cost: 4 byte

Treatment\_Result: 20 byte

Doctors\_Order: 10 byte

- b. Calcolare il numero di righe previsto nella tabella di fatti e la dimensione stimata (byte).

Presupposti.

1. Trattamenti: una media di 200 trattamenti totali vengono effettuati sui pazienti in un dato giorno (in base a un censimento su una media di 200 pazienti e a una media di due trattamenti per paziente al giorno).
  2. Pazienti: ogni trattamento è effettuato su un paziente. Per semplificare il tutto, si presuma che un dato trattamento possa essere effettuato solo una volta da un dato medico su un determinato paziente in un dato giorno.
  3. Medici: ogni trattamento è effettuato da un medico.
  4. Period: se viene sviluppato un data mart su vasta scala, si anticipa che verranno raccolti 1000 giorni (quasi tre anni) di dati.
- c. Perché è necessario presumere che un dato trattamento possa essere effettuato solo una volta da un dato medico su un determinato paziente in un dato giorno? Si proponga un metodo per superare questo limite.

## APPENDICE

# A

## Strumenti e notazione della modellazione E-R

**Il Capitolo 5 contiene una notazione per la rappresentazione dei modelli di dati che viene estesa nel Capitolo 6 e seguita per tutto il libro. A seconda dello strumento software che si ha a disposizione per la rappresentazione di un modello di dati, la capacità di riprodurre questa notazione può variare. Considerato che le regole e le politiche aziendali non sono universali, lo stesso si può dire dei simboli e delle notazioni utilizzati nei diversi strumenti di modellazione dei dati. Ognuno di questi si basa su costrutti grafici e metodologie differenti che potrebbero riuscire o meno a trasmettere il significato di una determinata regola aziendale.**

Questa appendice è concepita come un aiuto per confrontare la notazione del libro con quella del proprio strumento di modellazione. In particolare, vengono trattati quattro degli strumenti più comunemente utilizzati: Visible Analyst 7.4, Platinum ERwin 3.5.2, Microsoft Access 2000 e Oracle Designer 6.0. Le Figure A.1a e A.1b riepilogano la notazione utilizzata in ogni singolo strumento per indicare le entità, le associazioni, gli attributi, le regole, i vincoli e così via. Le Figure dalla A.4 alla A.7 rappresentano alcune schermate tratte da ciascuno strumento per mostrarne l'interfaccia.

La Figura 5.22, un diagramma E-R per Pine Valley Furniture Company (PVFC), è la base per gli esempi riportati in questa appendice. La figura mostra il modello di dati tratto dalle regole aziendali di PVFC descritte nel Capitolo 5 utilizzando il sistema di notazione adottato nel libro. La Figura A.1 consente un confronto di questa notazione con quella disponibile nei quattro strumenti software.

### A.1 Confrontare le convenzioni della modellazione E-R

Come si può vedere dalla Figura A.1, gli strumenti di modellazione differiscono notevolmente nella notazione disponibile per creare un diagramma entità-associazioni (ERD) che rappresenti il modello di dati logico. Sebbene quello che segue non sia un confronto approfondito tra i vari strumenti, è comunque un mezzo per analizzare le diversità fra di essi utilizzando l'ERD di PVFC rappresentato nella Figura 5.22. Si presti particolare attenzione alle differenze nella rappresentazione delle associazioni molti-a-molti, alle cardinalità, alle facoltatività, alle chiavi esterne e alle associazioni supertipo/sottotipo.

**Figura A.1**  
Un confronto  
della notazione  
di modellazione  
di Hoffer, Prescott  
e McFadden  
in quattro strumenti  
software diversi.  
(a) Notazioni  
degli strumenti  
di modellazione  
più diffusi.

|                       | <i>Notazione di Hoffer,<br/>Prescott e McFadden</i> | <i>Visible Analyst<br/>7.4</i> | <i>Platinum ERwin<br/>3.5.2</i> | <i>Microsoft Access<br/>2000</i>                                 | <i>Oracle Designer<br/>6.0</i>                                   |
|-----------------------|-----------------------------------------------------|--------------------------------|---------------------------------|------------------------------------------------------------------|------------------------------------------------------------------|
| Entità<br>di base     |                                                     |                                |                                 |                                                                  |                                                                  |
| Entità<br>associativa |                                                     |                                |                                 | (Nessun simbolo speciale. Utilizza il normale simbolo di entità) | (Nessun simbolo speciale. Utilizza il normale simbolo di entità) |
| Sottotipi             |                                                     |                                |                                 | Nessuna notazione specifica                                      |                                                                  |
| Entità<br>ricorsiva   |                                                     |                                |                                 |                                                                  |                                                                  |
| Attributi             |                                                     |                                |                                 |                                                                  |                                                                  |

(b) Notazioni di  
cardinalità/facoltatività  
degli strumenti  
di modellazione  
più diffusi.

|                     | <i>Notazione di Hoffer,<br/>Prescott e McFadden</i> | <i>Visible Analyst<br/>7.4</i>      | <i>Platinum ERwin<br/>3.5.2</i>     | <i>Microsoft Access<br/>2000</i>  | <i>Oracle Designer<br/>6.0</i> |
|---------------------|-----------------------------------------------------|-------------------------------------|-------------------------------------|-----------------------------------|--------------------------------|
| 1:1                 |                                                     | (Non disponibile senza cardinalità) | (Non disponibile senza cardinalità) |                                   |                                |
| 1:M                 |                                                     | (Non disponibile senza cardinalità) | (Non disponibile senza cardinalità) |                                   |                                |
| M:N                 |                                                     | (Non disponibile senza cardinalità) |                                     | (Non consentita)                  |                                |
| Obbligatoria<br>1:1 |                                                     |                                     |                                     | (Nessun simbolo di facoltatività) |                                |
| Obbligatoria<br>1:M |                                                     |                                     |                                     | (Nessun simbolo di facoltatività) |                                |
| Facoltativa<br>1:M  |                                                     |                                     |                                     | (Nessun simbolo di facoltatività) |                                |

## A.1.1 Notazione di Visible Analyst

Questo strumento flessibile fornisce all'utente numerose opzioni per la rappresentazione grafica dei modelli di dati. Offre cinque possibilità di notazione tra le quali scegliere, ovvero IDEF1X, Bachman, Crow's Foot, Arrow e UML. L'opzione Crow's Foot (zampa di gallina) è stata utilizzata nella Figura A.1 insieme alle convenzioni Gane & Sarson, e nella descrizione dello strumento. La scelta di una qualsiasi delle altre opzioni fornisce caratteristiche di rappresentazione diverse.

### Entità

Sono disponibili tre simboli di entità: di base, associativo e di attributo. Anche il livello del dettaglio del diagramma è flessibile; per esempio, si può visualizzare solo il nome dell'entità, o il nome dell'entità e la chiave primaria o includere anche tutti gli attributi. Il simbolo per un'entità associativa, per esempio Order\_Line, come mostrato nella Figura A.1a, si trasforma automaticamente in un simbolo diverso, con un indicatore [AS] accanto al nome dell'entità, quando viene collocato un attributo su di esso, per esempio Ordered\_Quantity. La Figura A.4 illustra questa situazione.

### Associazioni

Le linee di unione possono essere etichettate in una o entrambe le direzioni e possono avere più segmenti, tra le altre opzioni, che consentono che i punti di partenza e di arrivo siano distanti l'uno dall'altro. A seconda della notazione scelta, le linee delle associazioni possono mostrare la cardinalità, e possono memorizzare il numero specifico di istanze che mettono in relazione un'entità con un'altra. Le linee delle associazioni possono indicare 0:1, 1:1, 0:molti, 1:molti o molti:molti. Una finestra di dialogo consente di modificare la cardinalità per una relazione esistente. Un'associazione ricorsiva visualizza un genitore e un figlio della stessa entità. L'esempio nella Figura A.1 mostra che un Employee può essere supervisore di molti impiegati, ma che non tutti gli impiegati sono supervisori; la notazione indica che sono consentiti i valori nulli. L'entità genitore può avere un certo numero di entità figlio, ma un'entità figlio può avere solo un genitore. La regola in base alla quale un impiegato ha esattamente un supervisore non può essere mostrata; la notazione di facoltatività mostra che l'impiegato può non avere alcun supervisore.

## A.1.2 Notazione di Platinum ERwin (strumento CASE)

Platinum ERwin permette di scegliere tra le notazioni IDEF1X, IE (*Information Engineering*) o DM (*Dimensional Modeling*). Negli esempi è stata utilizzata la notazione IE.

### Entità

Se in un'associazione identificativa un'entità è di tipo figlio (debole), questa appare come entità dipendente, un rettangolo con gli angoli arrotondati. Anche i simboli delle entità associative sono rappresentati secondo questa convenzione. L'utente può decidere che le chiavi si spostino nell'area delle chiavi primarie o non di chiave di un rettangolo di entità.

### Associazioni

Le opzioni di cardinalità sono flessibili e possono essere specificate senza ambiguità. Un'entità genitore può essere connessa a "Zero, Una o Più," indicata da uno spazio vuoto, "Una o Più," indicata da una P; "Zero o Una", indicata da una Z, o "Esattamente" un determinato numero di istanze, che appariranno

nell'ERD. Le associazioni multi-a-molti vengono risolte automaticamente; un'entità associativa viene visualizzata nel modello logico, e l'associazione multi-a-molti viene eliminata. L'associazione ricorsiva non identificativa, in cui l'entità genitore e quella figlio sono rappresentate come la stessa entità, mostra che un Employee (un Supervisor) può essere supervisore di molti impiegati, ma che non tutti gli impiegati sono supervisori. La notazione indica che i valori nulli sono permessi. L'entità genitore può avere qualsiasi numero di figli, ma un'entità figlio può avere solo un genitore. Le chiavi migrano automaticamente quando le associazioni sono stabilite e le chiavi esterne sono indicate come (FK).

Il grafico catturato dalla guida in linea di ERwin mostrato nella Figura A.2 illustra la gamma dei simboli di cardinalità che possono essere utilizzati in questo prodotto.

**A.1.3      Notazione di Microsoft Access 2000**

Access non può agire come strumento per la creazione di grafici e diagrammi per costruire un modello logico precedente alla progettazione della base di dati. Le tabelle con attributi devono essere definite in una base di dati perché sia possibile rappresentarle in un modello e stabilire le associazioni tra le entità. Una volta che l'utente è consapevole di questo vincolo, la definizione della tabella consente comunque di modellarla nella finestra delle associazioni, e gli utenti di Microsoft Access 2000 devono cercare di approfittare di questa funzionalità per poter stabilire le associazioni.

**Figura A.2**  
Simboli di cardinalità/  
facoltatività di ERwin.

| Cardinality<br>Description   | IDEF1X Notation |                          |          | IE Notation |                          |          | DM Notation |                          |          |
|------------------------------|-----------------|--------------------------|----------|-------------|--------------------------|----------|-------------|--------------------------|----------|
|                              | Identifying     | Non-identifying<br>Nulls | No Nulls | Identifying | Non-identifying<br>Nulls | No Nulls | Identifying | Non-identifying<br>Nulls | No Nulls |
| One to zero,<br>one, or more |                 |                          |          |             |                          |          |             |                          |          |
| One to one<br>or more (P)    |                 |                          |          |             |                          |          |             |                          |          |
| One to zero<br>or one (Z)    |                 |                          |          |             |                          |          |             |                          |          |
| One to<br>exactly (N)        |                 |                          |          |             |                          |          |             |                          |          |

## Entità

Poiché Access è concepito più per la progettazione che per la modellazione delle basi di dati, non esistono simboli specifici per i diversi tipi di entità, comprese quelle associative o di supertipo e sottotipo.

## Associazioni

Rappresentate sotto forma di linee, le associazioni vengono disegnate collegando gli attributi esistenti tra tabelle già presenti. Le possibilità di manipolare la disposizione delle entità in modo che le associazioni possono essere mostrate con chiarezza sono piuttosto limitate. Le linee delle associazioni possono essere modificate per cambiare la cardinalità, ma non è possibile effettuare alcuna distinzione per stabilire la facoltatività. Quando le associazioni vengono stabilite, non si verifica alcuna migrazione automatica delle chiavi, come accade in ERwin, e gli attributi devono essere definiti per poter essere utilizzati come collegamenti. Le associazioni ricorsive uno-a-uno o uno-a-molti vengono rappresentate collocando due copie di una tabella sull'area di lavoro e stabilendo il collegamento tra di esse. Nella Figura 1.Aa sono mostrate due copie di EMPLOYEE, etichettate automaticamente da Access come EMPLOYEE\_t e EMPLOYEE\_t\_1. L'associazione ricorsiva è mostrata come un collegamento tra due tabelle che unisce la chiave primaria nella tabella EMPLOYEE\_t\_1 all'attributo Employee-supervisor nella tabella EMPLOYEE\_t. La natura uno-a-molti dell'associazione ricorsiva è mostrata nel collegamento tra EMPLOYEE\_t\_1 ed EMPLOYEE\_t. Le entità associative, come Order\_Line, vengono rappresentate utilizzando la stessa notazione delle altre entità. Come nella Figura 5.22, Order\_Line viene denominata con una capitalizzazione mista per distinguerla dalle altre entità, denominate con le lettere tutte maiuscole. L'associazione uno-a-molti tra PRODUCT e ORDER viene risolta creando l'entità associativa Order\_line che ha una chiave composta, Order\_ID e Product\_ID.

## A.1.4 Notazione di Oracle Designer

I diagrammi disegnati utilizzando lo strumento Entity Relationship Diagrammer possono essere impostati in modo da mostrare solo i nomi delle entità, i nomi delle entità e la chiave primaria o i nomi delle entità e tutte le etichette degli attributi.

## Entità

Non esistono simboli specifici per i diversi tipi di entità, comprese quelle associative o di supertipo e sottotipo. Tutte le entità vengono rappresentate come rettangoli arrotondati e gli attributi possono essere visualizzati all'interno di questi. Gli identificatori univoci sono preceduti da un segno # e devono essere obbligatori; gli attributi obbligatori sono contrassegnati con il simbolo \*, mentre quelli facoltativi sono contrassegnati con il simbolo °.

## Associazioni

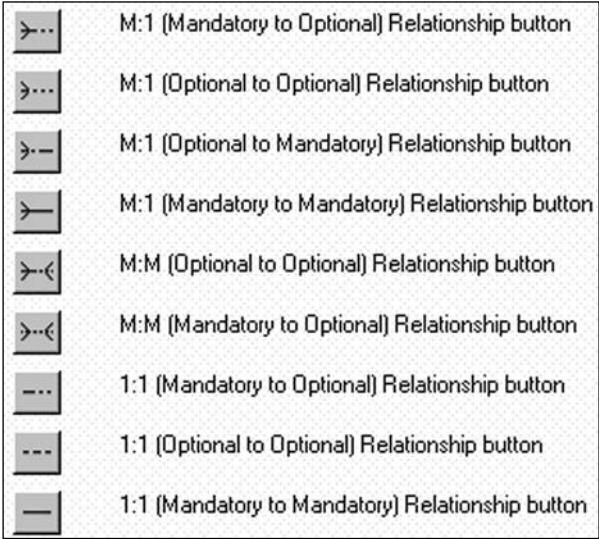
Le linee devono essere etichettate in entrambe le direzioni, non solo in una, non hanno segmenti multipli e sono piuttosto difficili da manipolare e allineare. Le linee delle associazioni rappresentano tanto la cardinalità quanto la facoltatività e vengono disegnate selezionando innanzitutto la cardinalità e la facoltatività desiderate; queste possono essere modificate in una finestra di dialogo. La facoltatività viene rappresentata utilizzando una linea continua per le associazioni obbligatorie e una tratteggiata per quelle facoltative. In un'associazione binaria in cui la facoltatività differisce a seconda della direzione dell'associazione, la linea tra le due entità viene suddivisa tra una linea continua e una tratteggiata. Per leggere l'associazione, si parte da un'entità e si segue la linea che è associata ad essa. Per esempio, nella Figura A.7 il prodotto deve essere fabbricato in un Work Center, ma non è richiesto che un Work Center



fabbrichi un prodotto per poter esistere. La cardinalità viene letta considerando il segno di cardinalità associato all'altra entità. Quindi, per esempio, un Customer può effettuare o meno un ordine, ma quando un ordine è effettuato, deve essere correlato a un determinato cliente. Osservando l'entità EMPLOYEE, l'associazione ricorsiva di supervisione viene rappresentata con l'ovale associato all'entità e mostra che un Employee può essere supervisore di uno o più impiegati e che un impiegato può essere supervisionato da un impiegato, o supervisore. Se poi la cardinalità multipla sia zero, uno o molti è ambiguo. La Figura A.3 mostra i pulsanti della barra degli strumenti disponibili in Designer per stabilire un'associazione; c'è un pulsante per ogni combinazione possibile.

Quando si lavora con Oracle Designer, è importante creare una bozza del modello dei dati in modo attento e completo prima di provare a utilizzare lo strumento. La modifica del modello può risultare complessa; si ricordi che eliminare un oggetto dal diagramma non significa eliminarlo automaticamente dal repository.

**Figura A.3**  
La barra degli strumenti delle associazioni di Oracle Designer 6.0.

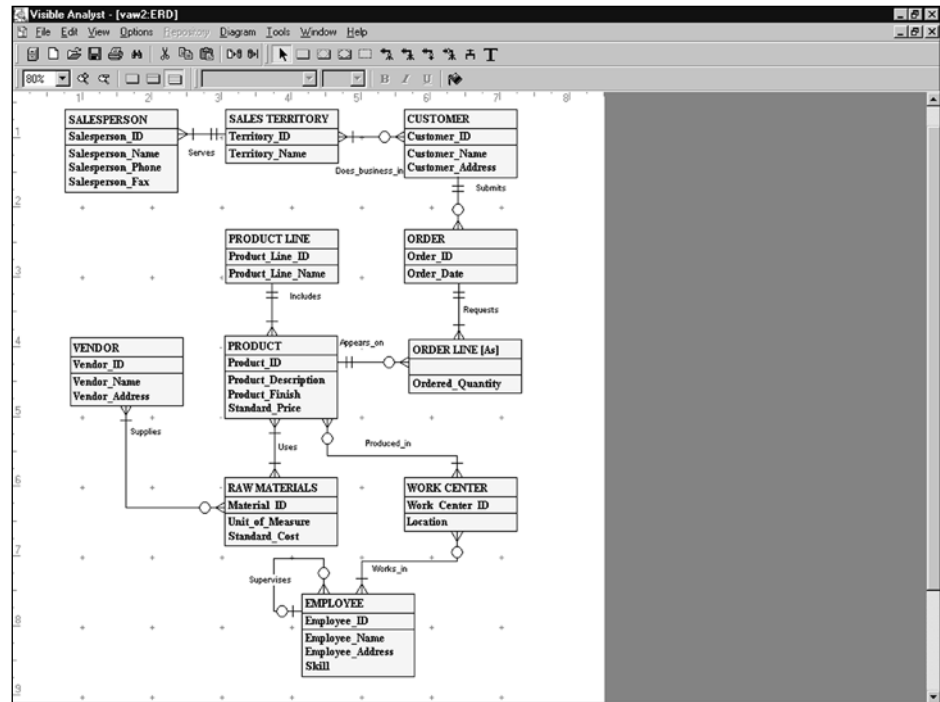


## A.2      Confronto tra interfacce degli strumenti e diagrammi E-R

Per ognuno degli strumenti software di modellazione elencati nella Figura A.1, viene inclusa una schermata di parte del modello di dati intero riportato nella Figura 5.22. Le figure dovrebbero fornire un'idea più chiara dell'aspetto della notazione con i simboli in uso. La Figura A.4 è stata creata utilizzando Visible Analyst 7.4 con le opzioni Crow's Foot e Gane & Sarson selezionate. La Figura A.5 è stata creata utilizzando Platinum ERwin 3.5.2 e l'opzione Information Engineering (IE). Se si desidera, è possibile visualizzare i simboli dell'integrità referenziale (non visibili in questo diagramma). La Figura A.6 è una cattura della schermata delle associazioni di Microsoft Access 2000, mentre la Figura A.6b mostra lo stampato che viene generato dalla schermata utilizzando l'opzione Print Relationships del menu File. La Figura A.7 è stata creata utilizzando Oracle Designer 6.0 con l'opzione Information Engineering.

**Figura A.4**

L'interfaccia di Visible Analyst 7.4.

**Figura A.5**

L'interfaccia di Platinum ERwin 3.5.2.

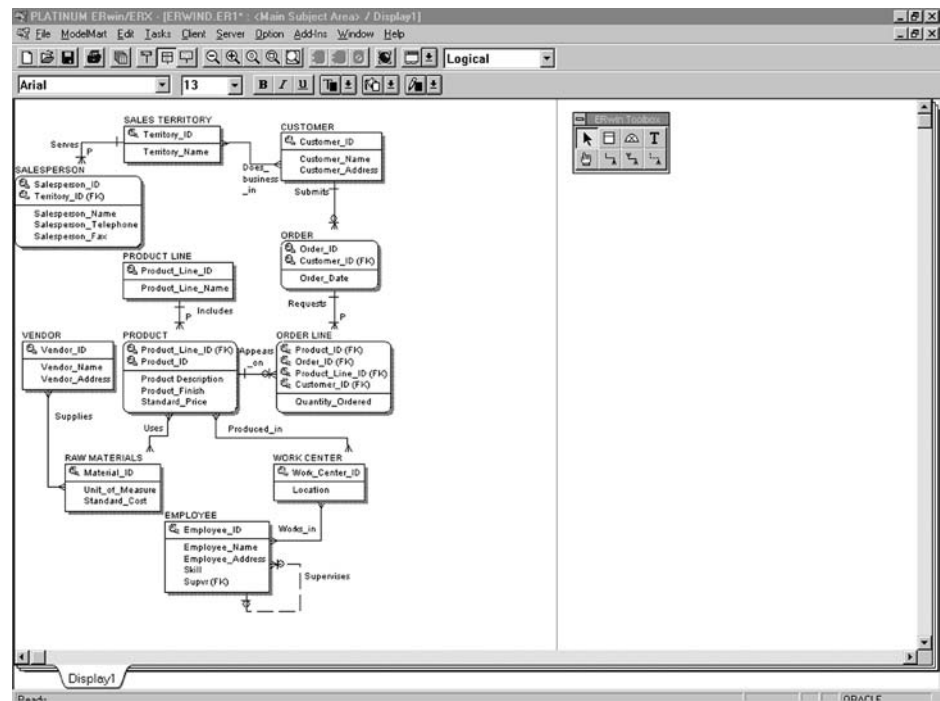
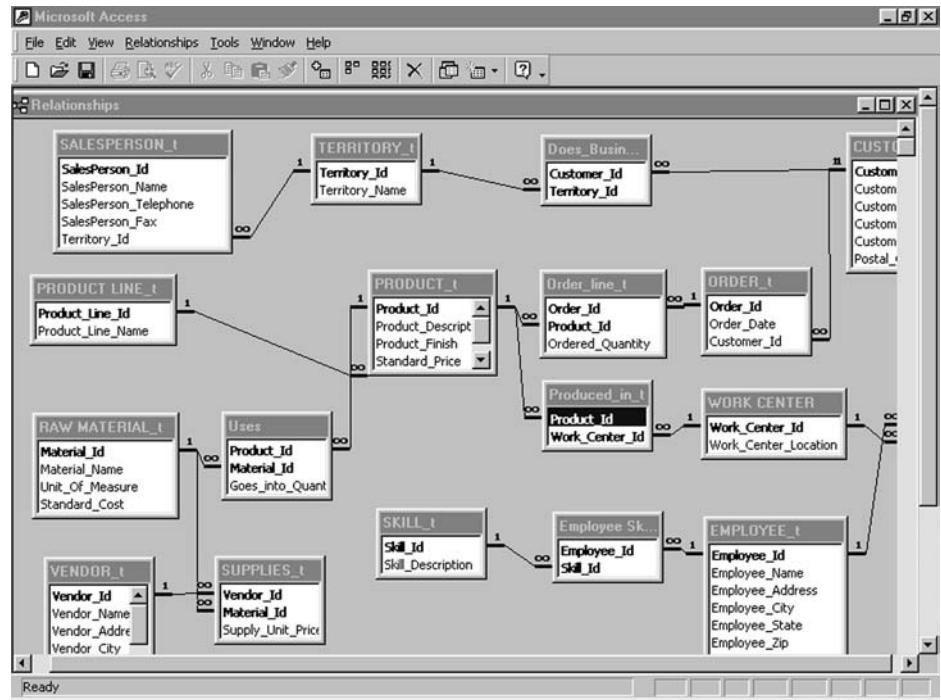


**Figura A.6**  
Microsoft Access 2000.  
(a) L'interfaccia  
di Microsoft Access  
2000: schermata delle  
associazioni.



(b) L'interfaccia  
di Microsoft Access  
2000: report  
delle associazioni.

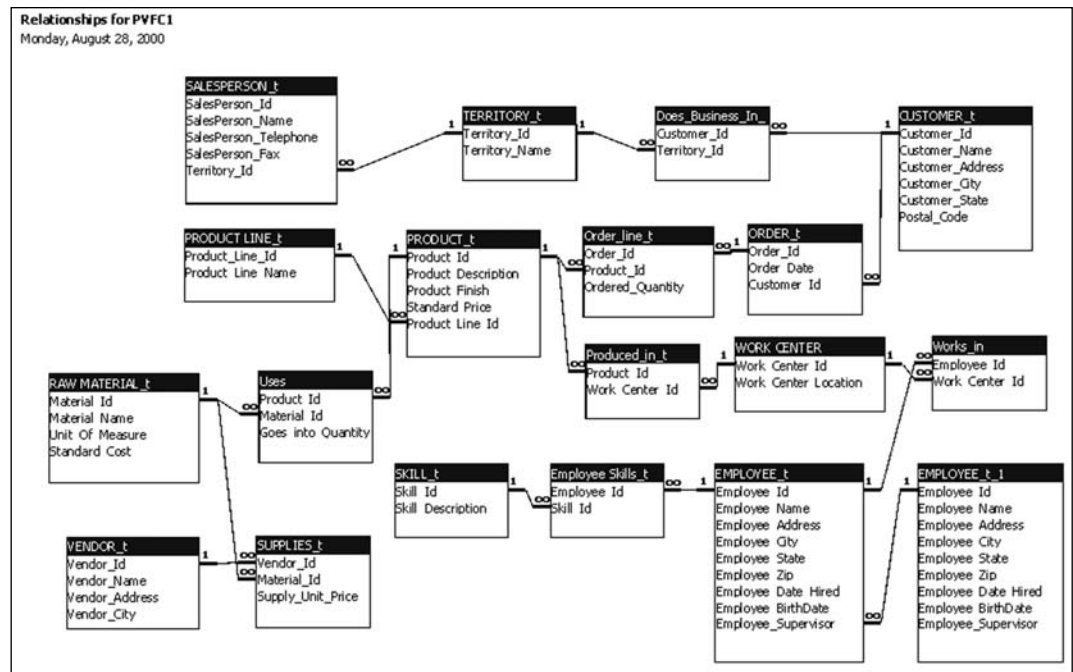
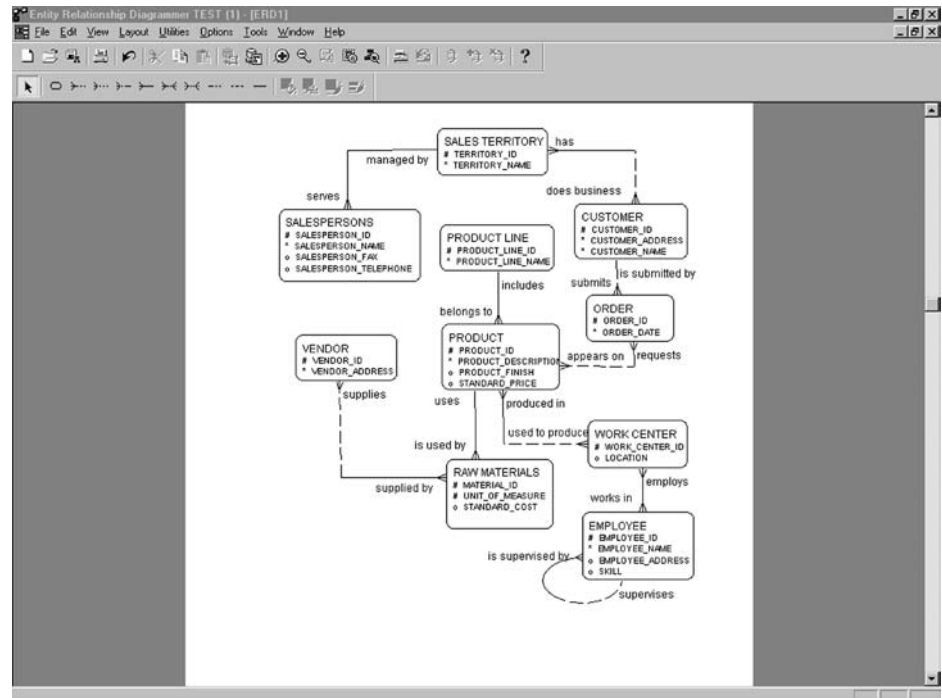


**Figura A.7**  
L'interfaccia di Oracle Designer 6.0.





## APPENDICE

# B

## Forme normali avanzate

Nel Capitolo 7 è stato introdotto il concetto di normalizzazione, che è stato poi descritto nel dettaglio a partire dalle terze forme normali. Le relazioni nella terza forma normale (3NF) sono sufficienti per la maggior parte delle applicazioni di basi di dati, ma non garantiscono che tutte le anomalie siano state rimosse. Come indicato nel Capitolo 7, per rimuovere tali anomalie sono state concepite altre forme normali supplementari: quella di Boyce-Codd, la quarta e la quinta (Figura 7.22). In questa appendice descriveremo la forma normale di Boyce-Codd e la quarta forma normale.

### B.1 Forma normale di Boyce-Codd

Quando una relazione ha più di una chiave candidata, possono risultare delle anomalie anche se la relazione è nella terza forma normale. Per esempio, si consideri la relazione `STUDENT_ADVISOR` mostrata nella Figura B.1. Questa relazione ha gli attributi `SID` (Student ID), `Major`, `Advisor` e `Maj_GPA`. I dati di esempio sono mostrati nella Figura B.1a, mentre le dipendenze funzionali sono illustrate nella Figura B.1b.

Come mostrato nella Figura B.1b, la chiave primaria per questa relazione è quella composta costituita da `SID` e `Major`. I due attributi `Advisor` e `Maj_GPA` sono quindi funzionalmente dipendenti da tale chiave. Questo riflette il vincolo in base al quale, sebbene possa avere più di una specializzazione, uno studente ha esattamente un tutor e una media di voti per ciascuna specializzazione.

La relazione contiene una seconda dipendenza funzionale: `Major` è funzionalmente dipendente da `Advisor`. Ogni tutor esegue il proprio lavoro esattamente all'interno di una specializzazione. Si noti che questa non è una dipendenza transitiva, cioè una dipendenza funzionale tra due attributi non di chiave (si veda il Capitolo 5). Per contro, in questo esempio un attributo di chiave (`Major`) è funzionalmente dipendente da uno non di chiave (`Advisor`).

#### B.1.1 Anomalie nella relazione `STUDENT_ADVISOR`

La relazione `STUDENT_ADVISOR` è chiaramente nella terza forma normale, poiché non esistono dipendenze funzionali parziali né transitive. Nondimeno, a causa della dipendenza funzionale tra `Major` e `Advisor`, la relazione presenta delle anomalie. Si considerino i seguenti esempi.

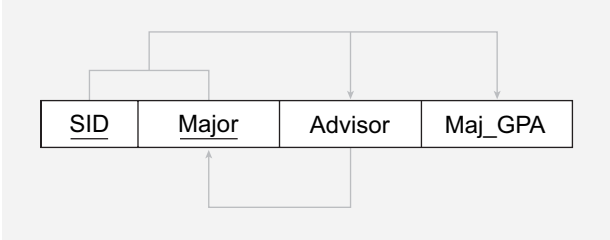
1. Si supponga che in Physics il tutor Hawking venga sostituito da Einstein. Questo cambiamento deve essere riportato in due o più righe della tabella (anomalia di aggiornamento).

**Figura B.1**

Relazione nella terza forma normale, ma non nella forma normale di Boyce-Codd.  
(a) Relazione con dati di esempio.

| STUDENT_ADVISOR |              |          |         |
|-----------------|--------------|----------|---------|
| <u>SID</u>      | <u>Major</u> | Advisor  | Maj_GPA |
| 123             | Physics      | Hawking  | 4.0     |
| 123             | Music        | Mahler   | 3.3     |
| 456             | Literature   | Michener | 3.2     |
| 789             | Music        | Bach     | 3.7     |
| 678             | Physics      | Hawking  | 3.5     |

(b) Dipendenze funzionali nella relazione STUDENT\_ADVISOR.



2. Si supponga di voler inserire una riga con l'informazione che dice che Babbage è tutor in Computer Science. L'inserimento non può avvenire fino a che ad almeno uno studente che si specializza in Computer Science non viene assegnato Babbage come tutor (anomalia d'inserimento).
3. Se lo studente 789 si ritira dall'università, si perde l'informazione in base alla quale Bach è tutor in Music (anomalia di eliminazione).

### B.1.2      Definizione della forma normale di Boyce-Codd (BCNF)

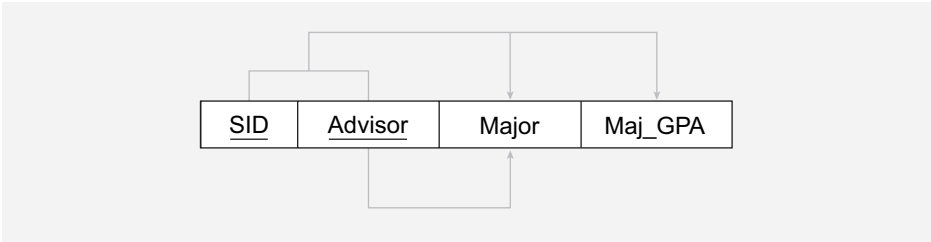
Le anomalie nella relazione STUDENT\_ADVISOR dipendono dal fatto che è presente una determinante (Advisor) che non è una chiave candidata nella relazione. R. F. Boyce ed E. F. Codd hanno identificato questo limite e hanno proposto una definizione più articolata di 3NF che risolve il problema. Si dice che una relazione è nella forma normale di Boyce-Codd (BCNF) se, e solo se, ogni determinante nella relazione è una chiave candidata. STUDENT\_ADVISOR non è in BCNF perché, sebbene l'attributo Advisor sia una determinante, non è una chiave candidata (solo Major è funzionalmente dipendente da Advisor).

### B.1.3      Convertire una relazione in BCNF

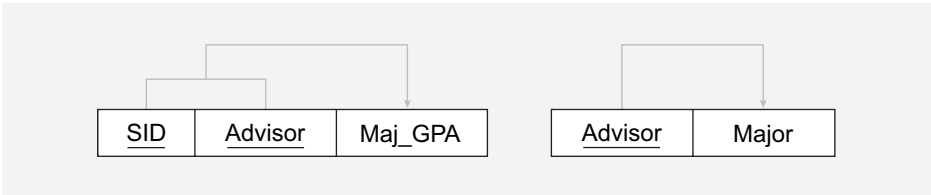
Una relazione in 3NF (ma non in BCNF) può essere convertita in relazioni in BCNF mediante un semplice processo in due passaggi, illustrato nella Figura B.2.

Nel primo passo, la relazione viene modificata in modo che la determinante nella relazione che non è una chiave candidata diventi un componente della chiave primaria della relazione rivisitata. L'attributo

**Figura B.2**  
Convertire una relazione in relazioni  
in BCNF.  
(a) Relazione STUDENT\_  
ADVISOR rivisitata (2NF).



(b) Due relazioni in BCNF.



(c) Relazioni con dati di esempio.

| STUDENT    |                |         | ADVISOR        |            |
|------------|----------------|---------|----------------|------------|
| <u>SID</u> | <u>Advisor</u> | Maj_GPA | <u>Advisor</u> | Major      |
| 123        | Hawking        | 4.0     | Hawking        | Physics    |
| 123        | Mahler         | 3.3     | Mahler         | Music      |
| 456        | Michener       | 3.2     | Michener       | Literature |
| 789        | Bach           | 3.7     | Bach           | Music      |
| 678        | Hawking        | 3.5     |                |            |

che dipende funzionalmente da quella determinante diventa un attributo non di chiave. Si tratta di una ristrutturazione valida della relazione originale dovuta alla dipendenza funzionale.

Il risultato dell'applicazione di questa regola alla relazione STUDENT\_ADVISOR è mostrato nella Figura B.2a. La determinante Advisor diventa parte della chiave primaria composta. L'attributo Major, che è funzionalmente dipendente da Advisor, diventa un attributo non di chiave.

Se si esamina la Figura B.2a, si scopre che la nuova relazione ha una dipendenza funzionale parziale (Major è funzionalmente dipendente da Advisor, che è solo un componente della chiave primaria). La nuova relazione è quindi nella prima (ma non nella seconda) forma normale.

Il secondo passaggio nel processo di conversione consiste nello scomporre la relazione per eliminare la dipendenza funzionale parziale, come illustrato nel Capitolo 7. Il risultato sarà costituito da due relazioni (Figura B.2b), entrambe nella terza forma normale. In realtà le relazioni sono anche in BCMF, poiché c'è solo una chiave candidata (quella primaria) in ogni relazione. Se una relazione ha solo una chiave candidata, che diventa quella primaria, 3NF e BCNF sono quindi equivalenti.

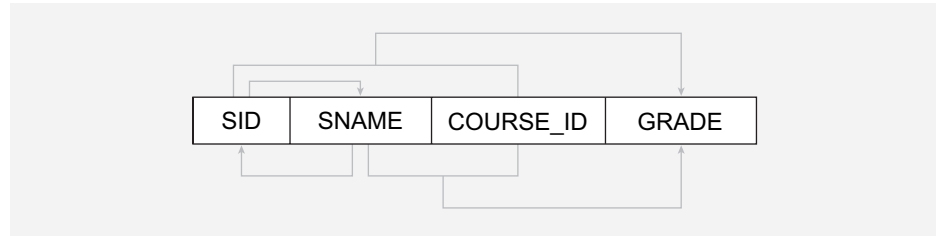
Le due relazioni (chiamate ora STUDENT e ADVISOR) con i dati di esempio sono mostrate nella Figura B.2c. È necessario assicurarsi che queste relazioni non contengano le anomalie descritte per STUDENT\_ADVISOR, oltre a verificare di poter ricreare la relazione STUDENT\_ADVISOR unendo le due relazioni STUDENT e ADVISOR.



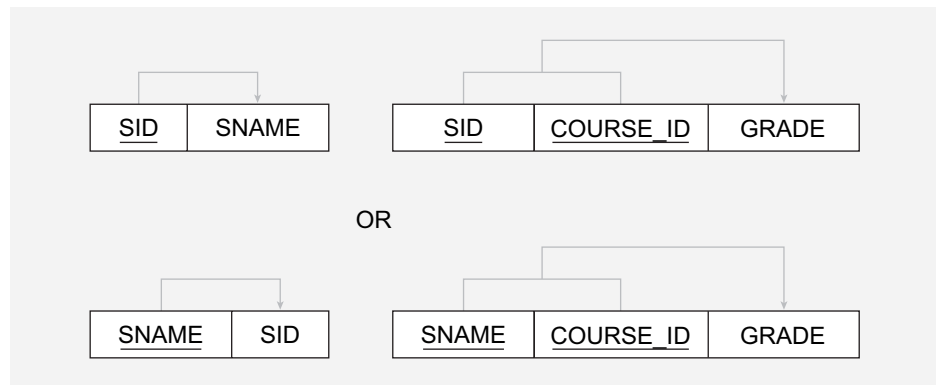
Un'altra situazione comune in cui la BCNF viene violata è quando nella relazione ci sono due o più chiavi candidate che si sovrappongono. Si consideri la relazione nella Figura B.3a; in questo esempio, ci sono due chiavi candidate (SID,COURSE\_ID) e (SNAME,COURSE\_ID), e COURSE\_ID appare in entrambe le chiavi candidate. Il problema con questa associazione è che non è possibile registrare i dati dei record dello studente (SID e SNAME) fino a quando questi non frequenta un corso. La Figura B.3b mostra due soluzioni possibili, ognuna delle quali crea due relazioni in BCNF.

**Figura B.3**

Convertire in BCNF una relazione con chiavi candidate che si sovrappongono.  
(a) Relazione con chiavi candidate che si sovrappongono.



(b) Due coppie alternative di relazioni in BCNF.



## B.2 Quarta forma normale

Quando una relazione è in BCNF, non ci sono più anomalie che risultano dalle dipendenze funzionali. Tuttavia possono esserci ancora delle anomalie che derivano dalle dipendenze multivalore (definite di seguito). Per esempio, si consideri la tabella mostrata nella Figura B.4a. Questa vista utente mostra per ogni corso il docente che lo tiene e i libri di testo adottati (che appaiono nella tabella come gruppi ripetuti). Nella tabella sono considerati i seguenti presupposti.

1. Ogni corso ha un insieme ben definito di docenti (per esempio, Management ha tre docenti).
2. Ogni corso ha un insieme ben definito di libri di testo adottati (per esempio, Finance ha due libri di testo).
3. I libri di testo adottati per un dato corso sono indipendenti dal docente di quel corso. Per esempio, gli stessi due libri di testo vengono utilizzati per Management a prescindere da quale dei tre docenti tiene il corso.

Nella Figura B.4b questa tabella è stata convertita in una relazione riempiendo tutte le celle vuote. La relazione, chiamata OFFERING, è nella prima forma normale. Di conseguenza, nella relazione OFFERING appaiono, per ogni corso, tutte le possibili combinazioni docente-libro di testo. Si noti che la chiave primaria della relazione è costituita da tre attributi, ovvero Course, Instructor e Textbook. Poiché non ci sono determinanti diverse dalla chiave primaria, la relazione è effettivamente in BCNF. Ciononostante, contiene diversi dati ridondanti che possono facilmente condurre a delle anomalie. Per esempio, si supponga di voler aggiungere un terzo libro di testo (autore: Middleton) al corso Management. Questa modifica richiederà l'aggiunta di tre nuove righe alla relazione nella Figura B.4b, una per ogni Instructor (altrimenti il libro si applicherebbe solo a determinati docenti).

**Figura B.4**  
Dati con dipendenze multivalore.

OFFERING

| Course     | Instructor              | Textbook          |
|------------|-------------------------|-------------------|
| Management | White<br>Green<br>Black | Drucker<br>Peters |
| Finance    | Gray                    | Jones<br>Chang    |

(a) Tabella dei corsi, docenti e libri di testo

OFFERING

| Course     | Instructor | Textbook |
|------------|------------|----------|
| Management | White      | Drucker  |
| Management | White      | Peters   |
| Management | Green      | Drucker  |
| Management | Green      | Peters   |
| Management | Black      | Drucker  |
| Management | Black      | Peters   |
| Finance    | Gray       | Jones    |
| Finance    | Gray       | Chang    |

(b) Relazione in BCNF

B.2.1 Dipendenze multivalore

Il tipo di dipendenza mostrata in questo esempio è chiamata **dipendenza multivalore**, e si crea quando ci sono almeno tre attributi (per esempio, A, B e C) in una relazione, e per ogni valore di A c'è un insieme di valori ben definito di B e uno di C. L'insieme di valori di B è tuttavia indipendente dall'insieme di C e viceversa.

Per rimuovere la dipendenza multivalore da una relazione, si deve dividere la relazione in due nuove relazioni. Ognuna di queste tabelle contiene due attributi che hanno un'associazione multivalore nella relazione originale. La Figura B.5 mostra il risultato della scomposizione per la relazione OFFERING della Figura B.4b. Si noti che la relazione chiamata TEACHER contiene gli attributi Course e Instructor, poiché ogni corso c'è un insieme di docenti ben definito. Per la stessa ragione, TEXT contiene gli attributi Course e Textbook. Non ci sono tuttavia relazioni che contengono tutti gli attributi Instructor e Course, poiché questi sono indipendenti.

Una relazione è nella **quarta forma normale (4NF)** se è nella forma normale di Boyce-Codd e non contiene dipendenze multivalore. Nella Figura B.5 si può vedere chiaramente che le due relazioni sono in 4NF e che non contengono le anomalie descritte in precedenza. Si può inoltre verificare che è possibile ricostruire la relazione originale OFFERING unendo queste due relazioni. Si noti che ci sono meno dati nella Figura B.5 che nella Figura B.4b. Per semplicità, si presuma che Course, Instructor e Textbook

siano tutti della stessa lunghezza. Poiché ci sono 24 celle di dati nella Figura B.4b e 16 nella Figura B.5, c'è un risparmio di spazio del 25 percento nelle tabelle nella quarta forma normale.

**Figura B.5**  
Relazioni in 4NF.

| TEACHER       |                   | TEXT          |                 |
|---------------|-------------------|---------------|-----------------|
| <u>Course</u> | <u>Instructor</u> | <u>Course</u> | <u>Textbook</u> |
| Management    | White             | Management    | Drucker         |
| Management    | Green             | Management    | Peters          |
| Management    | Black             | Finance       | Jones           |
| Finance       | Gray              | Finance       | Chang           |

### B.3    Forme normali di grado più elevato

Sono state definite almeno altre due forme normali di grado più elevato: la quinta forma normale (5NF) e la forma normale di chiave di dominio (DKNF). La 5NF ha a che fare con una proprietà chiamata “join senza perdita”. Secondo Elmasri e Navathe (2000), “questi casi si verificano molto raramente e sono difficili da individuare nella pratica”. Per questa ragione, e anche perché la sua definizione è complessa, in questo libro non descriveremo la quinta forma normale.

La DKNF è un tentativo di definire “una forma normale definitiva” che prenda in considerazione tutti i tipi di dipendenze e vincoli possibili (Elmasri e Navathe, 2000). Sebbene la definizione di DKNF sia semplice, secondo questi autori, “la sua utilità pratica è piuttosto limitata”. Per questa ragione non la descriveremo.

Per ulteriori informazioni su queste due forme normali di grado più elevato, si vedano Elmasri e Navathe (2000) e Dutka e Hanson (1989).

### Bibliografia

Dutka, A. e Hanson, H., 1989, *Fundamentals of Data Normalization*, Reading, MA, Addison-Wesley.  
Elmasri, R. e Navathe, S., 2000, *Fundamentals of Database System*, terza edizione, Reading, MA, Addison-Wesley, pp. 440, 443.

Le strutture di dati sono le fondamenta di qualsiasi architettura di base di dati. A prescindere dall'organizzazione di file o dal DBMS adottati, sono utilizzate per connettere segmenti di dati correlati. Sebbene molti DBMS moderni nascondano le strutture di dati, la modifica della base di dati richiede la comprensione delle scelte compiute dal progettista riguardo a esse. Questa appendice affronta gli elementi fondamentali di tutte le strutture di dati e offre una panoramica sugli schemi più diffusi per la memorizzazione e l'individuazione degli elementi fisici dei dati.

## C.1 Puntatori

Nel Capitolo 8 è stato introdotto l'elemento puntatore, che viene utilizzato solitamente come riferimento per l'indirizzo di un altro segmento di dati. Nella pratica, esistono tre tipi di puntatori, come illustrato nella Figura C.1.

1. *Puntatore di indirizzo fisico.* Contiene l'indirizzo su disco effettivo e completo (dispositivo, volume, traccia e numero di blocco) dei dati referenziati. Un puntatore fisico è il modo più veloce per trovare un altro segmento di dati, ma è anche il più limitato: se l'indirizzo dei dati referenziati cambia, dovranno essere modificati anche tutti i puntatori. I puntatori fisici sono solitamente utilizzati nelle applicazioni di basi di dati legacy con le architetture di rete e gerarchiche.
2. *Puntatore di indirizzo relativo.* Contiene la posizione relativa (*offset*, o *distanza*) dei dati associati rispetto a una base, o punto di partenza. L'indirizzo relativo può essere una posizione in byte, un record o un numero di riga. Un puntatore relativo ha il vantaggio che quando l'intera struttura dei dati cambia di posizione, tutti i riferimenti relativi ai dati vengono mantenuti. I puntatori relativi vengono utilizzati in un'ampia gamma di DBMS; un uso comune è quello negli indici, dove le chiavi di indice vengono associate agli identificatori di riga (un tipo di puntatore relativo) per i record con quel valore di chiave.
3. *Puntatore di chiave logico.* Contiene dati significativi sull'elemento di dati associato. Un puntatore logico deve essere trasformato in uno fisico o relativo da una tabella di ricerca, una ricerca per indici o un calcolo matematico per riuscire a trovare effettivamente i dati referenziati. Le chiavi esterne in una base di dati relazionale sono spesso puntatori di chiave logici.

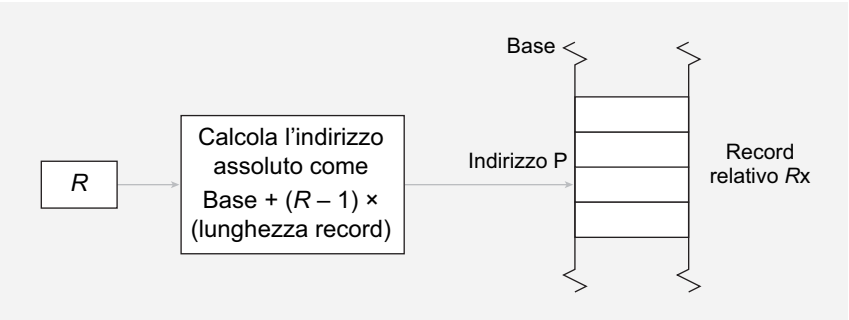
**Figura C.1**

Tipi di puntatore.

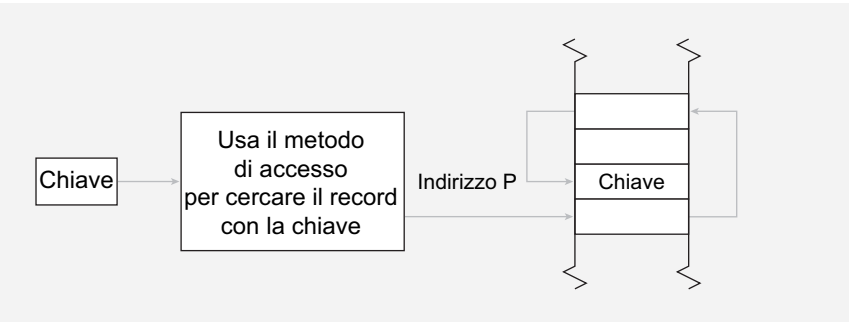
(a) Puntatore di indirizzo fisico.



(b) Puntatore di indirizzo relativo per il record  $R_x$  nel file.



(c) Puntatore di chiave logico per il record con la chiave.



La Tabella C.1 riepiloga le caratteristiche salienti di ciascuno di questi tre tipi di puntatori. Un progettista deve essere in grado di decidere quale puntatore utilizzare nelle diverse situazioni presentate dalla base di dati. Per esempio, una chiave esterna in una relazione può essere implementata utilizzando uno qualsiasi dei tre tipi di puntatori. Inoltre, quando una base di dati viene danneggiata, l'amministratore che riesce a individuare il tipo di puntatore utilizzato può ricostruire i collegamenti interrotti tra i contenuti della stessa.

**Tabella C.1** Confronto tra i tipi di puntatori.

| Caratteristica                        | Tipo di puntatore                                |                                                     |                                     |
|---------------------------------------|--------------------------------------------------|-----------------------------------------------------|-------------------------------------|
|                                       | Fisico                                           | Relativo                                            | Logico                              |
| Forma                                 | Indirizzo reale della memoria secondaria (disco) | Distanza dal punto di riferimento (inizio del file) | Dati aziendali significativi        |
| Velocità di accesso                   | Il più veloce                                    | Velocità media                                      | Il più lento                        |
| Sensibilità allo spostamento dei dati | Il più sensibile                                 | Sensibile solo alle modifiche delle posizioni       | Il meno sensibile                   |
| Vulnerabilità alla distruzione        | Alta                                             | Alta                                                | Può essere ricostruito facilmente   |
| Requisiti di spazio                   | Fissi, solitamente brevi                         | Variabili, solitamente i più brevi                  | Variabili, solitamente i più lunghi |

## C.2 Le fondamenta delle strutture di dati

Tutte le strutture di dati vengono costruite combinando numerosi elementi per la connessione e il posizionamento dei dati. I metodi di connessione permettono lo spostamento tra gli elementi di dati correlati; la loro individuazione consente che i dati all'interno di una struttura vengano prima posizionati o memorizzati e poi trovati. Per connettere gli elementi di dati sono disponibili due soli metodi.

1. *Connessione sequenziale di indirizzo.* Un elemento successore (o correlato) viene posizionato e individuato nello spazio fisico di memoria immediatamente dopo l'elemento corrente (Figure C.2a e c). Questo tipo di connessione offre le prestazioni migliori nella lettura dell'intero insieme dei dati o del record successivo nella sequenza memorizzata. Per contro, è però poco efficiente nel recupero dei record arbitrari e nelle operazioni di aggiornamento dei dati (aggiunta, eliminazione e modifica). Anche le operazioni di aggiornamento sono poco efficienti, perché l'ordine fisico deve essere costantemente mantenuto, cosa che richiede solitamente una riorganizzazione immediata dell'intero insieme di dati.
2. *Connessione sequenziale di puntatore.* Un puntatore (o più puntatori) viene memorizzato con un elemento di dati per identificare la posizione dell'elemento di dati successore (o correlato), come mostrato nelle Figure C.2b e d. La connessione sequenziale di puntatore è più efficiente nelle operazioni di aggiornamento, perché i dati possono essere collocati ovunque fino a quando i collegamenti tra i dati correlati sono mantenuti. Un'altra funzione principale degli schemi sequenziali di puntatore è la capacità di mantenere più collegamenti sequenziali diversi nello stesso insieme di dati utilizzando più puntatori. Vedremo alcune forme comuni di questi schemi (strutture di dati lineari) tra breve.

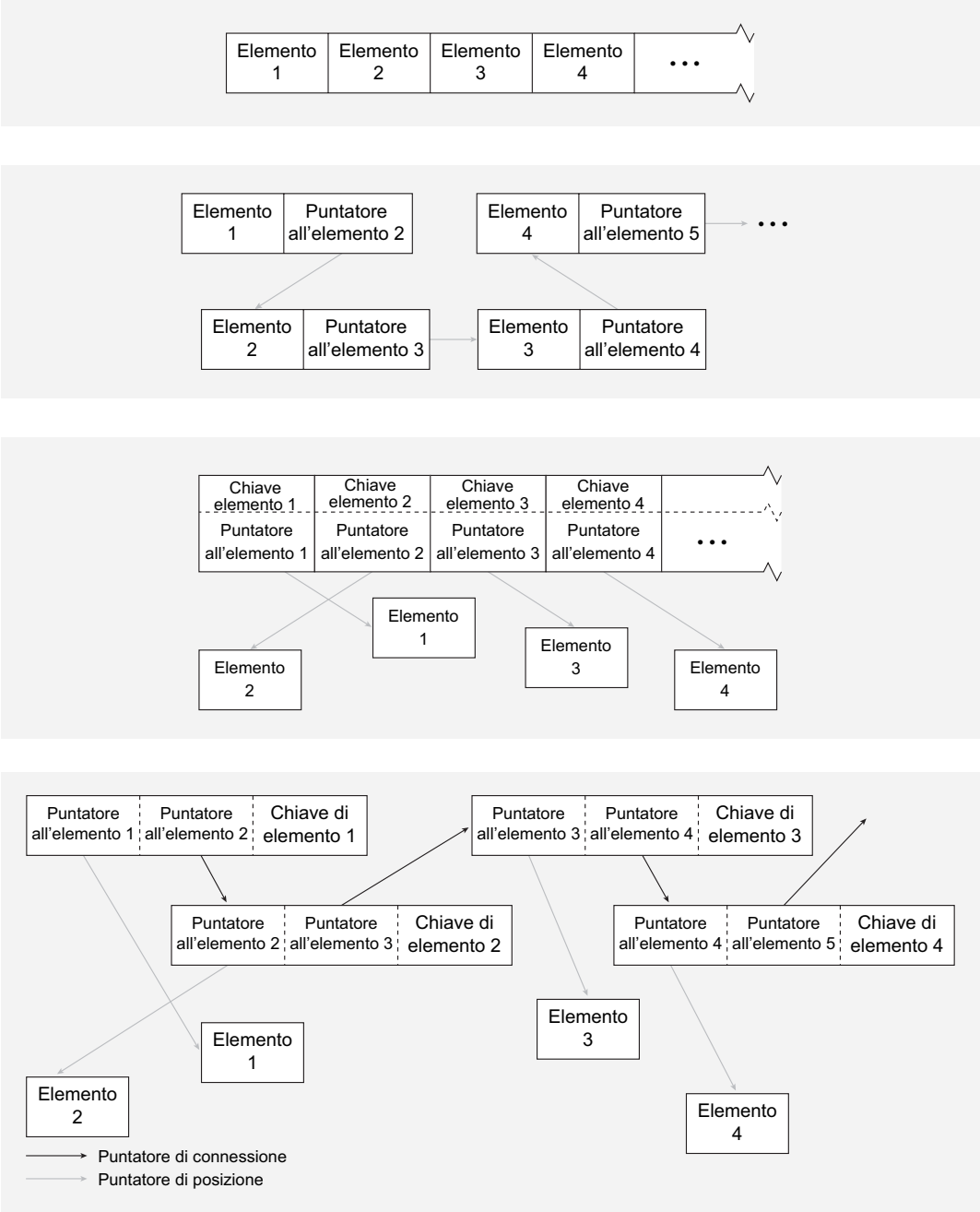
Esistono poi due metodi fondamentali per posizionare i dati in base al meccanismo di connessione.

1. *Posizionamento diretto dei dati.* Il meccanismo di connessione collega un elemento di dati direttamente al suo successore (o correlato), come mostrato nelle Figure C.2a e b. Il posizionamento diretto ha il vantaggio di permettere il recupero immediato dei dati una volta che una connessione viene attraversata. Lo svantaggio è che i dati reali vengono distribuiti su più aree estese del disco perché lo spazio per i dati reali deve essere allocato tra gli elementi della connessione.
2. *Posizionamento indiretto dei dati.* Il meccanismo di connessione collega i puntatori ai dati, ma non ai dati reali (Figure C.2c e d). Il vantaggio del posizionamento indiretto è che l'analisi di una struttura di dati alla ricerca di dati con caratteristiche specifiche è solitamente più efficiente, poiché può essere effettuata attraverso l'immissione globale di caratteristiche delle chiavi e dei puntatori ai dati associati. La connessione e il posizionamento dei dati vengono inoltre spaiati, così che l'organizzazione dei record dei dati possa seguire lo schema più adatto (per esempio, la connessione sequenziale fisica per un determinato ordinamento). Gli svantaggi sono il tempo di accesso supplementare necessario per poter recuperare entrambi i riferimenti ai dati e i dati, e lo spazio in più richiesto dai puntatori.

Qualsiasi struttura di dati, organizzazione di file o architettura di base di dati utilizza una combinazione di questi quattro metodi di base.

**Figura C.2**

Metodi di posizionamento di base.  
(a) Connessione sequenziale di indirizzo (sequenziale).  
(b) Connessione sequenziale di puntatore (catena semplice o elenco).  
(c) Connessione dei dati indiretta e sequenziale di indirizzo (indice di chiave).  
(d) Connessione dei dati indiretta e sequenziale di puntatore (indice di chiave a catena).

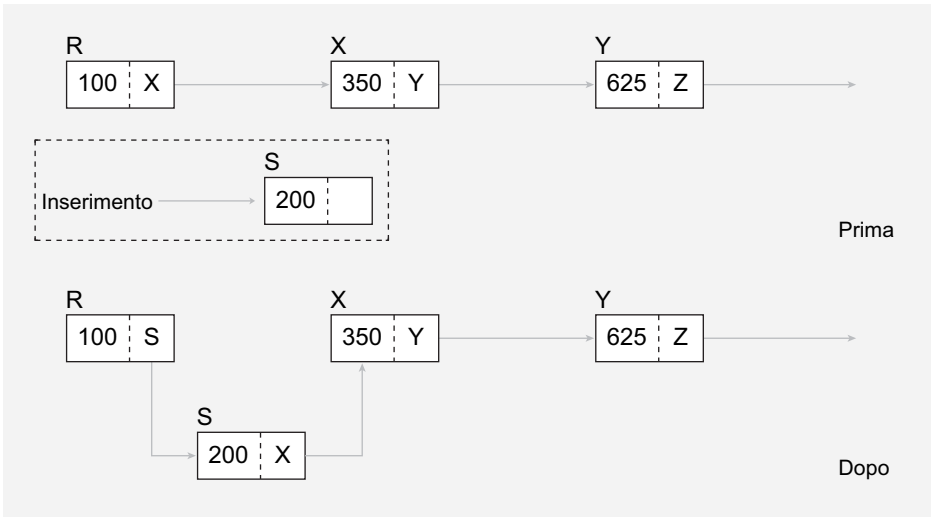


### C.3 Strutture di dati lineari

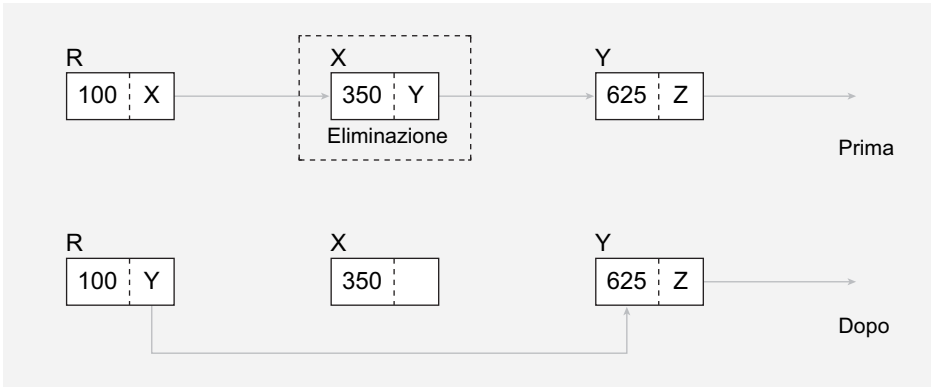
Le strutture di dati sequenziali di puntatore sono diventate popolari per la memorizzazione dei dati particolarmente volatili, come sono solitamente quelli delle basi di dati. I dati delle transazioni (come gli ordini dei clienti o le richieste di cambio del personale) e i dati storici (come i prezzi dei prodotti e le iscrizioni degli studenti ai corsi) costituiscono una parte consistente delle basi di dati operazionali. Poiché i loro utenti in genere vogliono visualizzare i dati in base a più sequenze diverse (per esempio, gli ordini in ordine di data, di numero di prodotto o di numero di cliente), la capacità di mantenere più catene di puntatori rivolti verso gli stessi dati può soddisfare diverse esigenze degli utenti rispetto a un insieme di dati.

La capacità delle strutture di dati lineari di gestire gli aggiornamenti dei dati è illustrata nella Figura C.3. La Figura C.3a mostra come sia facile inserire un nuovo record in una struttura lineare (o catena). La figura mostra un file di record di prodotti. Per semplicità, ogni record di prodotto è rappresentato dal solo numero di prodotto e da un puntatore al record di prodotto successivo in sequenza in base al

**Figura C.3**  
Gestire una struttura di dati di puntatore sequenziale.  
(a) Inserimento.



(b) Eliminazione.





numero del prodotto. Un nuovo record è memorizzato in una posizione libera (S) e inserito nella catena modificando i puntatori associati ai record nelle posizioni E e S. Nella Figura C.3b si dimostra come l'eliminazione di un record sia altrettanto facile, poiché solo il record nella posizione R viene modificato. Sebbene ci sia uno spazio supplementare per memorizzare i puntatori, questo è ridotto al minimo se confrontato con le centinaia di byte necessari per produrre tutti i dati (numero di prodotto, descrizione, quantità disponibile, prezzo standard e così via). Trovare i record nell'ordine in base al numero di prodotto fornito nella struttura è semplice, ma il tempo effettivo per recuperare i record in una sequenza può essere lungo se i record sequenziali logici sono memorizzati distanti fra loro sul disco.

In questa breve introduzione alle strutture di dati lineari, prenderemo ora in considerazione quattro versioni specifiche di tali strutture: stack, code, elenchi ordinati ed elenchi multipli. Concluderemo il paragrafo con alcuni avvertimenti sulle strutture di dati a catena lineari.

### C.3.1      **Stack**

La caratteristica principale di uno stack (*pila*) è che tutti gli inserimenti e le eliminazioni di record vengono eseguite alla stessa estremità della struttura di dati. Gli stack possiedono una proprietà LIFO (*last-in-first-out*). Un esempio di stack del mondo reale è una colonna verticale di vassoi in una caffetteria. Nei sistemi informativi aziendali, gli stack sono utilizzati per mantenere i record a cui non sono assegnate priorità particolari o che non sono ordinati (per esempio, gli articoli delle linee di prodotti associati allo stesso ordine effettuato da un cliente).

### C.3.2      **Code**

La caratteristica principale di una coda è che tutti gli inserimenti si verificano a un'estremità della struttura e tutte le eliminazioni all'altra. Le code possiedono una proprietà FIFO (*first-in-first-out*). Un esempio di coda del mondo reale è una corsia di cassa in una drogheria. Nei sistemi informativi aziendali, le code sono utilizzate per mantenere elenchi di record in ordine cronologico di inserimento. Per esempio, la Figura C.4 illustra una coda a catena dei record di Order\_Line in ordine di arrivo per un record di Product comune in Pine Valley Furniture.

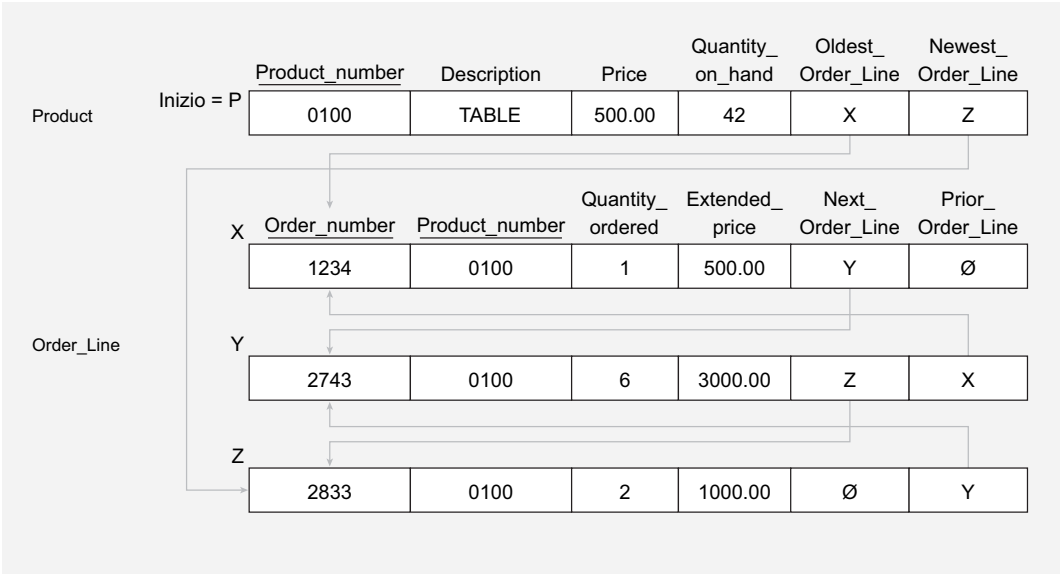
In questo esempio, il record Product agisce come nodo di inizio della catena nella struttura dei dati. Il valore del campo Oldest\_order\_line è un puntatore al record Order\_Line vecchio (il primo immesso) per il prodotto 0100. Il campo Next\_order\_line nel record Order\_Line contiene i puntatori al record successivo nella sequenza in ordine cronologico. Il valore  $\emptyset$  in un puntatore è chiamato *puntatore nullo* e indica la fine della catena.

Questo esempio introduce anche il concetto di catena bidirezionale, che contiene puntatori rivolti sia in avanti sia all'indietro. Il vantaggio dei puntatori precedente e successivo è che i dati nei record possono essere recuperati e presentati in un ordine normale o contrario e che il codice per gestire la catena è più facile da implementare che non con le catene a direzione unica.

### C.3.3      **Elenchi ordinati**

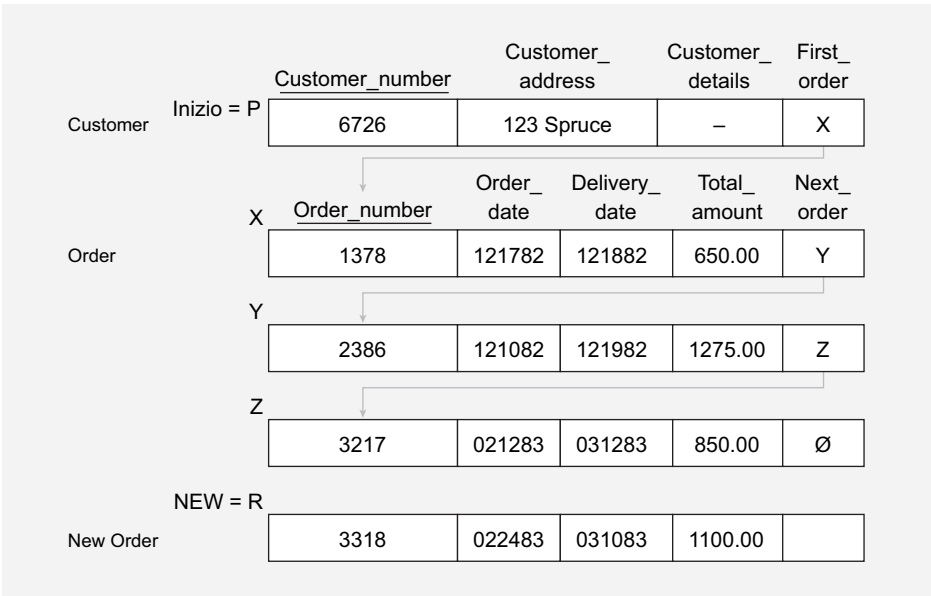
La caratteristica principale di un elenco ordinato è che gli inserimenti e le eliminazioni possono verificarsi ovunque nell'elenco; i record sono mantenuti in un ordine logico basato su un valore di campo di chiave. Un esempio di elenco ordinato del mondo reale è un elenco telefonico. Nei sistemi informativi aziendali gli elenchi ordinati sono molto frequenti. La Figura C.5a illustra un elenco ordinato sequenziale di puntatori a direzione unica dei record Order correlati a un record Customer, nel quale i record sono ordinati in base alla Delivery\_date.

**Figura C.4**  
Esempio di coda con  
puntatori bidirezionali.

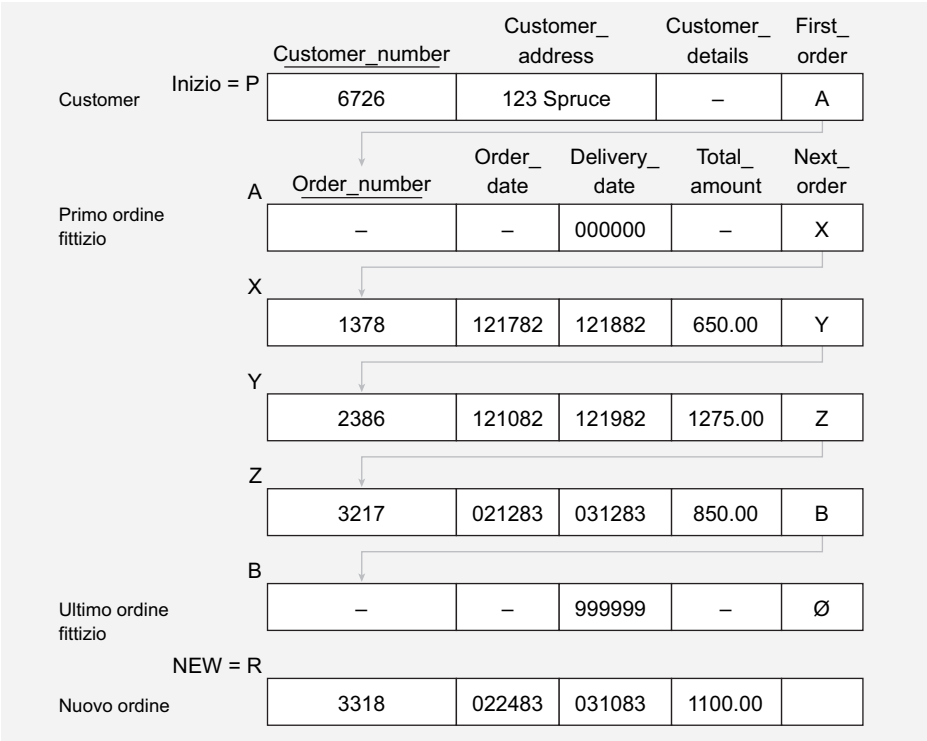


Mantenere un elenco ordinato è molto più complesso che non mantenere uno stack o una coda, perché gli inserimenti e le eliminazioni possono verificarsi in un punto qualsiasi della catena, che può avere zero o molti record. Per assicurare che queste operazioni avvengano sempre all'interno della catena, spesso vengono inclusi un primo e un ultimo record "fittizi" (Figura C.5b). La Figura C.5c mostra il risultato dell'inserimento di un nuovo Order nell'elenco ordinato della Figura C.5b. Per eseguire l'inserimento, l'elenco viene analizzato partendo dall'indirizzo nel puntatore First\_order.

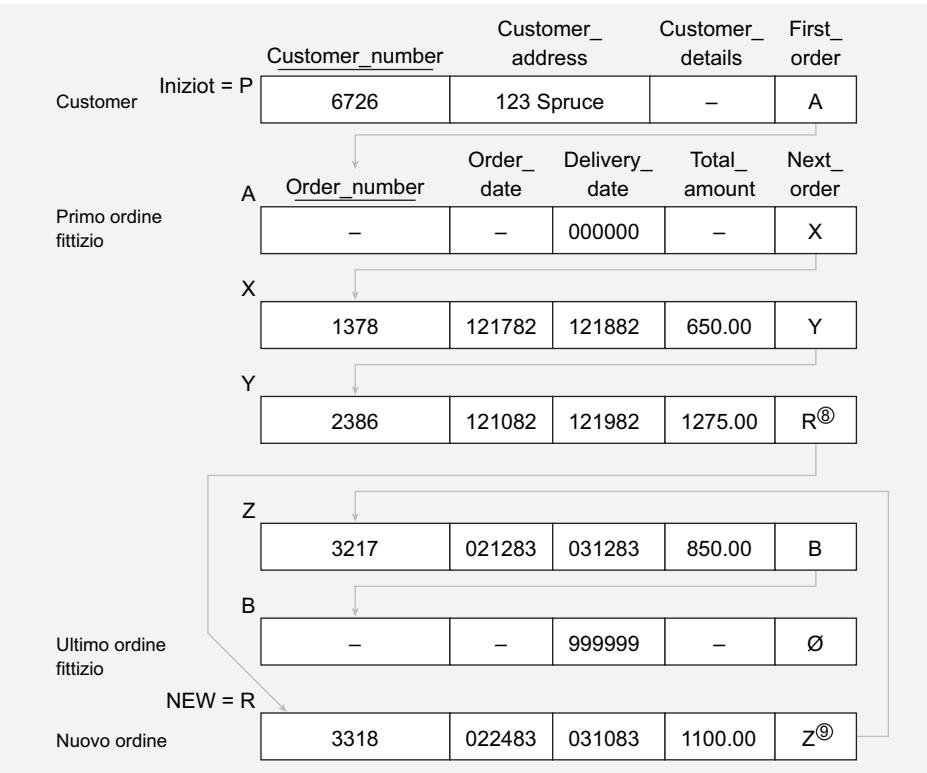
**Figura C.5**  
Esempio di elenco ordinato.  
(a) Prima dell'inserimento  
di un nuovo Order e senza il primo  
e l'ultimo ordine fittizi.



(b) Prima dell'inserimento di un nuovo Order e con il primo e l'ultimo ordine fittizi.



c) Dopo l'inserimento di un nuovo Order (i numeri cerchiati accanto ai puntatori indicano il numero del passaggio nella procedura di gestione associata che modifica il valore del puntatore).



Una volta trovata la posizione corretta nella catena, occorre stabilire una regola per decidere dove memorizzare un record con un valore di chiave duplicato, se i duplicati sono consentiti, come in questo esempio. Tale posizione sarà solitamente tra i duplicati, così che l'analisi venga ridotta al minimo.

Se si utilizza un'organizzazione di file o un DBMS che supportano la catena, e nello specifico gli elenchi ordinati, non sarà necessario scrivere il codice per mantenere gli elenchi. Questo farà infatti già parte della tecnologia utilizzata. Il programma eseguirà semplicemente un comando di inserimento, eliminazione e aggiornamento, e il software di supporto penserà alla gestione della catena. La Figura C.6 delinea il codice necessario a inserire un nuovo record nell'elenco ordinato nella Figura C.5b. Le variabili di posizione PRE e AFT sono utilizzate per contenere rispettivamente i valori del predecessore e del successore del nuovo record Order. Il passaggio 7 è racchiuso tra parentesi quadre per indicare dove appare l'eventuale controllo delle chiavi duplicate. Il simbolo  $\leftarrow$  indica la sostituzione del valore della variabile a sinistra con il valore della variabile a destra. Nella Figura C.5, i passaggi 8 e 9, che modificano i valori dei puntatori, mostrano esattamente quali puntatori cambieranno nell'esempio. Si può anche decidere di controllare questa routine manualmente eseguendola per vedere come i valori delle variabili sono impostati e modificati.

**Figura C.6**

Passaggi del codice di inserimento di un nuovo ordine.

```

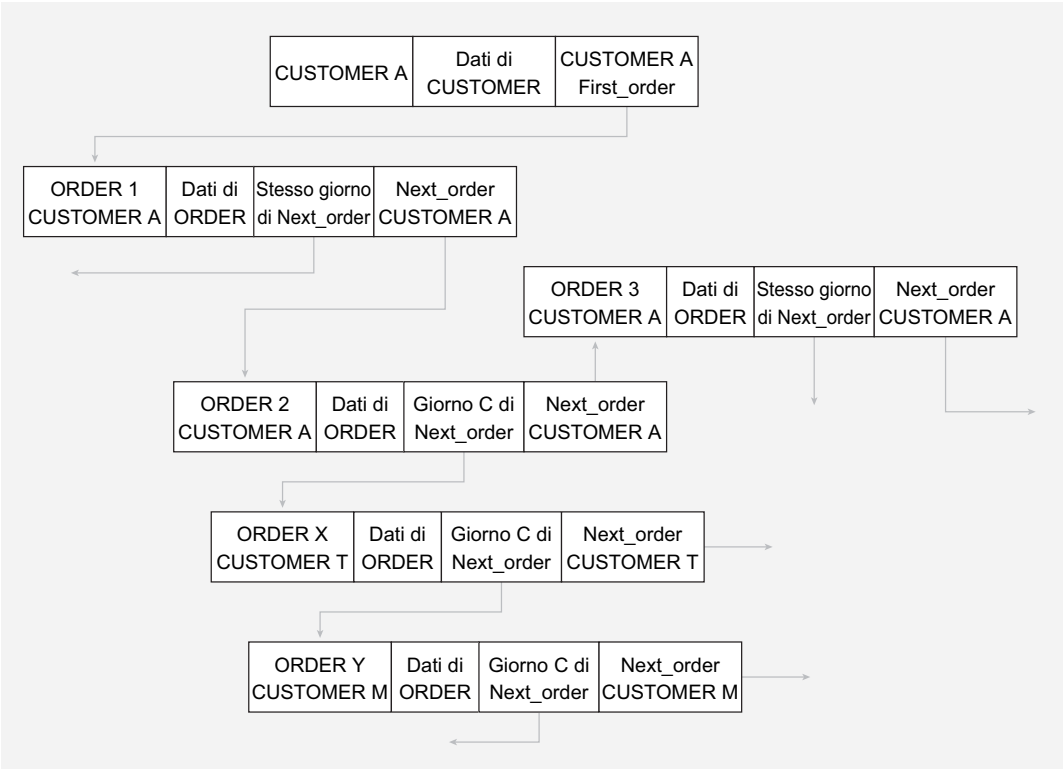
/* Stabilisce la posizione dei valori di inizio delle variabili */
1 PRE $\leftarrow$ First_order(START)
2 AFT $\leftarrow$ Next_order(PRE)
/* Analizza/salta nella catena fino a trovare la posizione corretta */
3 DO WHILE Delivery_date(AFT) < Delivery_date(NEW)
4 PRE $\leftarrow$ AFT
5 AFT $\leftarrow$ Next_order(AFT)
6 ENDO
7 [Se Delivery_date(AFT) = Delivery_date(NEW), allora segnala
 un errore di duplicazione e termina la procedura]
/* Inserisce un nuovo elemento nella catena */
8 Next_order(PRE) $\leftarrow$ NEW
9 Next_order(NEW) $\leftarrow$ AFT

```

### C.3.4 Elenchi multipli

In una struttura di dati multielenco, viene mantenuta più di una sequenza tra gli stessi record. Le catene multiple sono quindi costruite in thread tra gli stessi record, e i record possono essere analizzati in una qualsiasi delle sequenze mantenute senza duplicare i record dei dati. Lo svantaggio di questo accesso così flessibile è la necessità di uno spazio di memoria supplementare e della gestione di ogni singola catena. Con un elenco multiplo è possibile seguire un'associazione e decidere a metà strada di seguirne un'altra. Per esempio, quando si accede ai record di Order per un dato Customer (un elenco), si può scoprire che tutti gli articoli ordinati devono essere consegnati nello stesso giorno di consegna di un dato record di Order. Un elenco multiplo di questo tipo è illustrato nella Figura C.7.

**Figura C.7**  
Esempi di strutture  
multielenco.



Un elenco multiplo fornisce alcuni dei vantaggi degli indici multipli (si veda il Capitolo 8 per una trattazione degli indici di chiave primari e secondari). Gli svantaggi principali degli elenchi multipli, e la ragione prima per cui non vengono utilizzati nei DBMS relazionali, è che il costo dell'analisi di un elenco è molto alto se confrontato con l'accesso a un indice, e non c'è un modo rapido per rispondere alle qualifiche delle chiavi multiple con gli elenchi multipli (per esempio, trovare tutti gli ordini dei clienti nella zona Nord-est e i prodotti nella linea Paper). Questi sono alcuni dei motivi per cui gli indici hanno generalmente sostituito le strutture di dati lineari nelle tecnologie di basi di dati moderne. Le applicazioni legacy possono tuttavia continuare a utilizzare le tecnologie che impiegano strutture a elenchi singoli e multipli.

## C.4 Rischi delle strutture a catena

Oltre al limite per il quale non possono essere utilizzate per rispondere rapidamente alle qualifiche delle chiavi multiple, le catene presentano altri rischi e restrizioni.

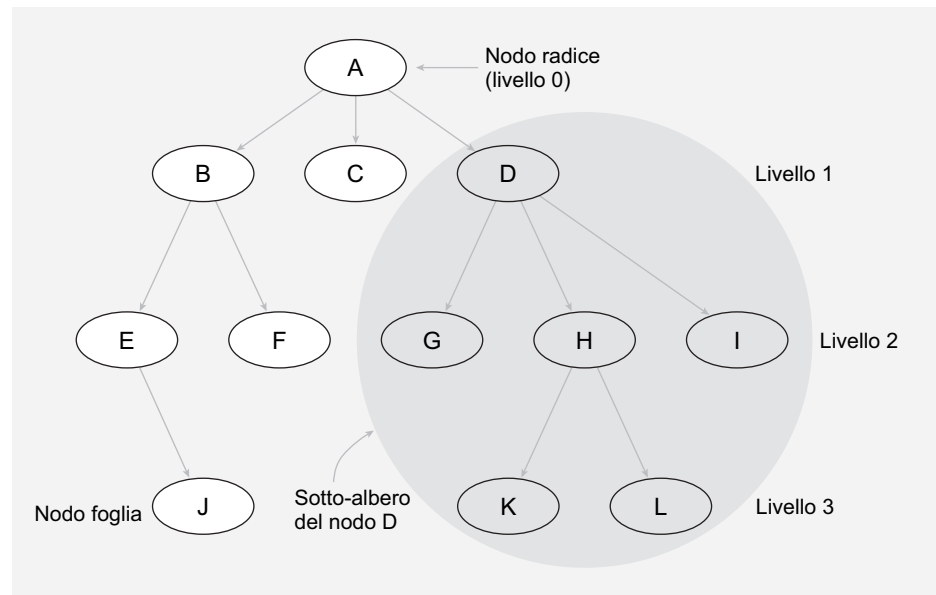
1. Può essere necessario molto tempo per analizzare le catene lunghe, poiché non è detto che i record in sequenza siano memorizzati gli uni vicini agli altri.
2. Le catene sono vulnerabili alle interruzioni. Se si verifica un evento insolito a metà della routine di manutenzione della catena, questa può risultare aggiornata solo parzialmente e diventa incompleta.

e imprecisa. Per risolvere questi errori, si possono adottare alcune misure di protezione, che però richiedono costi superiori in termini di spazio di memoria e di elaborazione.

## C.5 Alberi

Il problema della lunghezza della struttura di dati lineare e del tempo necessario alla sua analisi è particolarmente rilevante. Fortunatamente sono state sviluppate strutture non lineari, che implementano una strategia *dividi et impera*. Un tipo di struttura non lineare molto comune è l'albero. Un *albero* (Figura C.8) è una struttura di dati costituita da un insieme di nodi che dipartono da un nodo in cima all'albero (che è quindi capovolto!), detto nodo radice. Ogni nodo dell'albero, a eccezione di quello radice, ha esattamente un genitore e può avere zero, uno o più nodi figlio. I nodi vengono definiti in termini di livelli; la radice è il livello zero, i figli di questo nodo sono al livello uno e così via. Un nodo foglia è un nodo che non ha figli (per esempio, i nodi J, F, C, G, K, L e I nella Figura C.8). Un sotto-albero di un nodo è costituito da quel nodo e da tutti i suoi discendenti.

**Figura C.8**  
Esempio di una struttura di dati ad albero.



### C.5.1 Alberi bilanciati

L'uso più diffuso degli alberi in un sistema di gestione di basi di dati moderno è quello che organizza le entità in un indice di chiave. Come nelle strutture di dati lineari, il programmatore della base di dati non deve mantenere la struttura ad albero, perché sarà il DBMS a farlo. Tuttavia, può avere la possibilità di controllare la struttura di un albero di indice per perfezionare le prestazioni dell'elaborazione dell'indice.

La forma di albero più comune utilizzata per costruire gli indici di chiave è l'*albero bilanciato*, o *albero B* (*B-tree*). In un albero B, tutte le foglie hanno la stessa distanza dalla radice; per questo motivo,

tali alberi hanno un'efficienza prevedibile. Gli alberi B supportano il recupero sia casuale sia sequenziale dei record. La forma più diffusa di albero B è l'*albero B+*. Un albero B+ di grado  $m$  possiede la seguente proprietà speciale di albero bilanciato.

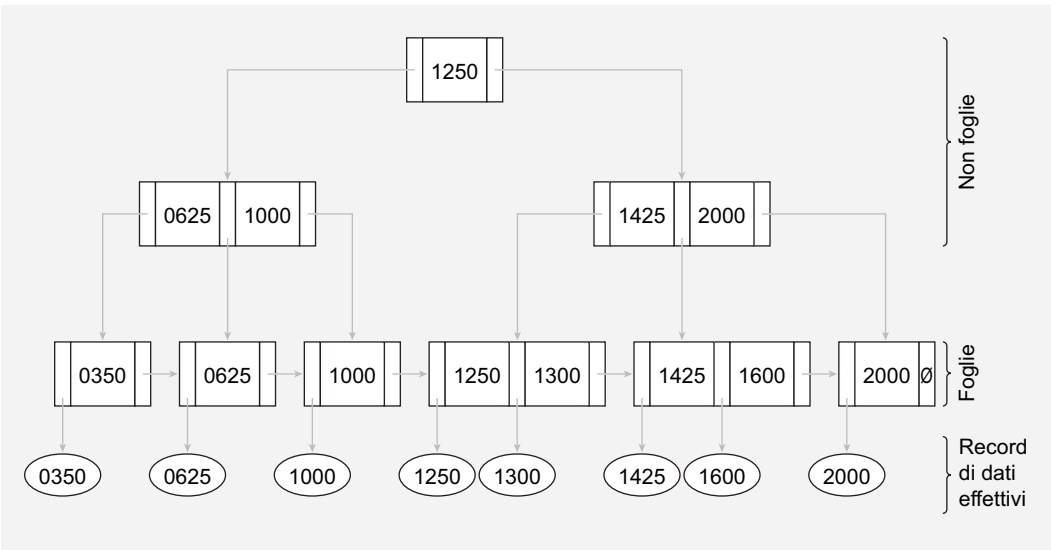
Ogni nodo ha tra  $m$  e  $m/2$  figli ( $m$  è un intero maggiore di o uguale a 3, solitamente dispari), a eccezione del nodo radice (che non rispetta questo limite inferiore).

È questa proprietà che conduce alla riorganizzazione dinamica dei nodi, che vedremo successivamente nel paragrafo.

Il metodo VSAM (*Virtual Sequential Access Method*), un metodo di accesso ai dati supportato da numerosi sistemi operativi, si basa sulla struttura di dati dell'albero B+. Tra ISAM e VSAM ci sono due differenze principali. La prima è che la posizione delle voci degli indici in ISAM è limitata dai confini fisici di un'unità disco, mentre in VSAM le voci degli indici possono superare i confini fisici; la seconda differenza è che un file ISAM deve essere ricostruito quando la sua struttura diventa inefficace dopo molti inserimenti ed eliminazioni, mentre in VSAM l'indice viene riorganizzato dinamicamente in modo progressivo quando i segmenti dell'indice non sono più produttivi.

Un esempio di albero B+ di grado 3 è mostrato nella Figura C.9, e riguarda il file Product di Pine Valley Furniture. In questo diagramma, ogni freccia verticale rappresenta il percorso seguito per i valori che sono uguali al numero a sinistra della freccia, ma inferiori rispetto al numero a destra. Per esempio, nel nodo non foglia che contiene i valori 625 e 1000, la freccia centrale che parte dal basso di questo nodo è il percorso seguito per i valori uguali a 625 ma minori di 1000. Le frecce orizzontali vengono utilizzate per connettere i nodi foglia di modo che l'elaborazione sequenziale possa avvenire senza che sia necessario spostarsi su e giù tra i livelli dell'albero.

**Figura C.9**  
Esempio di albero B+.

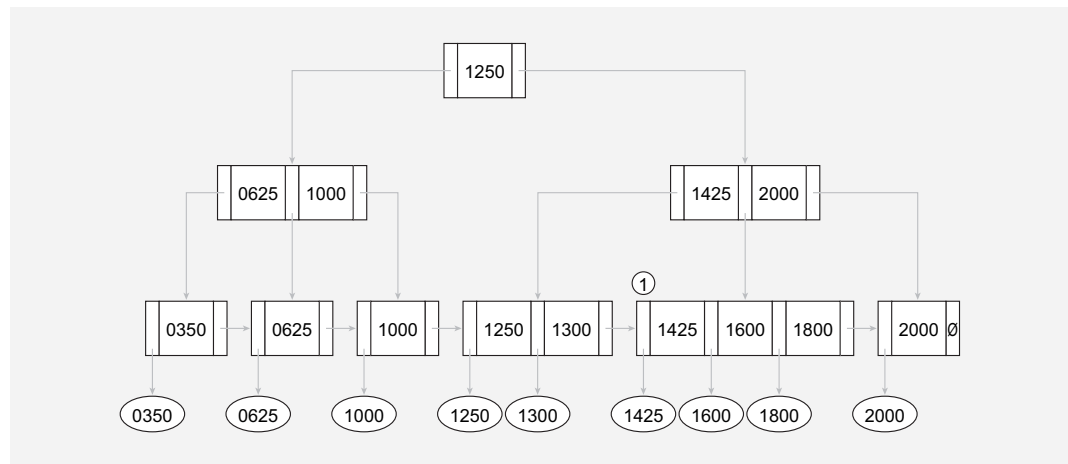


Si immagini di voler recuperare i record di dati per il numero di prodotto 1425. Si noti che il valore nel nodo radice è 1250. Poiché 1425 è maggiore di 1250, si dovrà seguire la freccia a destra di questo valore verso il basso fino al livello successivo. In questo nodo si trova il valore ricercato (1425), quindi si seguirà la freccia centrale verso il basso fino al nodo foglia che contiene il valore 1425. Questo nodo

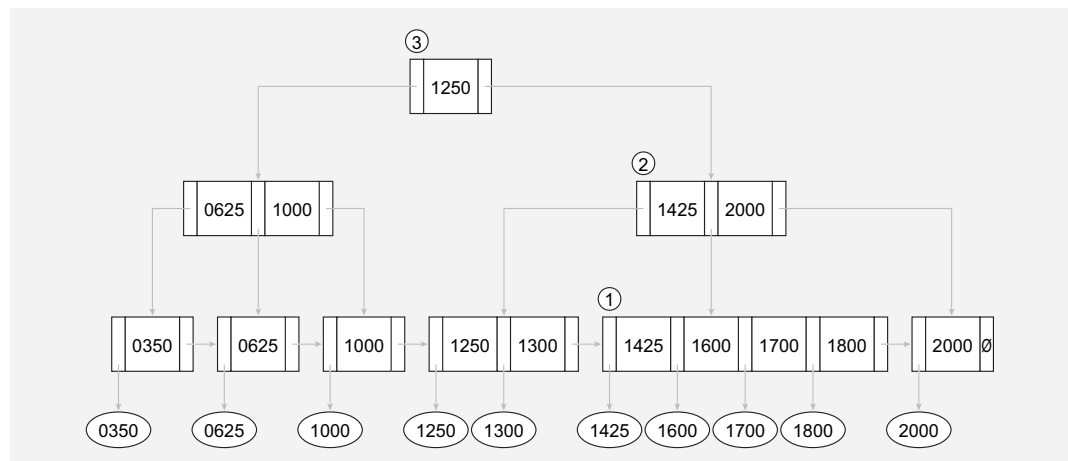
contiene un puntatore al record di dati per il numero di prodotto 1425, quindi il record può essere recuperato. Si può seguire un percorso analogo per trovare il record per il numero di prodotto 1000. Poiché i record di dati vengono memorizzati all'esterno dell'indice, possono essere mantenuti indici multipli di alberi B+ sugli stessi dati.

Un albero B+ può anche supportare facilmente l'aggiunta o l'eliminazione dei record. Qualsiasi modifica apportata alla struttura di un albero B+ è dinamica e non altera le proprietà dell'albero. Si supponga di aggiungere un record con la chiave 1800 all'albero B+ nella Figura C.9. Il risultato di questo inserimento è mostrato nella Figura C.10a. Poiché il nodo 1 ha solo tre figli (il puntatore orizzontale non conta come puntatore figlio), l'albero B+ nella Figura C.10a soddisfa tutte le proprietà dell'albero B+. Si consideri ora l'effetto dell'aggiunta di un altro record all'albero, questa volta con la chiave 1700. Un primo risultato di questo inserimento appare nella Figura C.10b. In questo caso, il nodo 1 viola il limite del grado, e deve quindi essere suddiviso in due nodi. La suddivisione del nodo 1 comporta una nuova immissione nel nodo 2, che farà sì che questo nodo abbia quattro figli, uno di troppo. Anche il nodo 2 dovrà quindi essere suddiviso, cosa che comporta una nuova immissione nel nodo 3. Il risultato finale è mostrato nella Figura C.10c.

**Figura C.10**  
Inserire record  
in un albero B+.  
(a) Inserimento  
del record 1800.

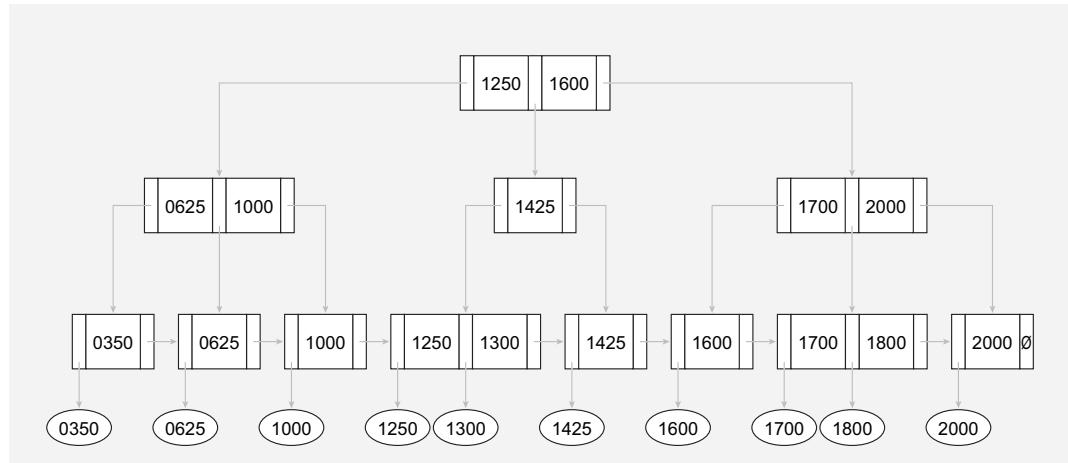


(b) Primo tentativo  
di inserimento del  
record 1700.





(c) L'albero B+ finale  
dopo l'inserimento  
del record 1700.



Se si desidera approfondire l'argomento degli alberi B, si veda Comer (1979), un articolo fondamentale sulla loro progettazione e le loro proprietà.

Una situazione interessante si verifica quando il nodo radice diventa troppo grande (ha più di  $m$  figli). In questo caso, è il nodo radice a essere suddiviso, cosa che comporta l'aggiunta di un nuovo livello all'albero. L'eliminazione di un record causa l'eliminazione di una voce nel nodo. Se questa eliminazione fa sì che una foglia abbia meno di  $m/2$  figli, la foglia verrà unita a una foglia adiacente; se la foglia unita è troppo grande (ha più di  $m$  figli), viene suddivisa, cosa che comporta semplicemente una ridistribuzione più equilibrata delle chiavi tra i nodi. Il risultato è che un albero B+ viene riorganizzato dinamicamente perché sia possibile mantenerlo bilanciato (stessa profondità lungo qualsiasi percorso dalla radice) e con un numero limitato di voci per ogni nodo (così da controllare lo "spessore" dell'albero).

## Bibliografia

Comer, D., 1979, "The Ubiquitous B-tree", *ACM Computing Surveys*, 11 (giugno), pp. 121–137.

## Basi di dati relazionali a oggetti

In questo libro di testo abbiamo descritto i due modelli più ampiamente utilizzati per la progettazione e l'implementazione delle basi di dati: quello relazionale e quello orientato agli oggetti. Il modello relazionale (con i sistemi di gestione relativi) è attualmente quello impiegato più di frequente per implementare le applicazioni aziendali. Alcune delle sue potenzialità sono elencate di seguito.

- Indipendenza dei dati, descritta nel Capitolo 3.
- Linguaggio di interrogazione potente (SQL), descritto nel Capitolo 9.
- Tecnologia matura di gestione delle basi di dati relazionali, comprese funzioni come il backup e il ripristino on-line e il controllo flessibile delle operazioni simultanee.

Nonostante queste potenzialità, è stato notato che il modello di dati relazionale non è particolarmente adatto per gestire tipi di dati complessi, come immagini, audio, video e dati spaziali o geografici. Il modello di dati orientato agli oggetti è stato introdotto principalmente per gestire questi tipi di dati complessi.

Il mercato dei sistemi di gestione delle basi di dati orientate agli oggetti è cresciuto lentamente, ma la tecnologia è estremamente promettente per le applicazioni come i server Web che incorporano tipi di dati multimediali. Le debolezze di questi sistemi, almeno fino a questo momento, sono l'assenza di funzioni importanti che si trovano nei sistemi relazionali, come la potenzialità di interrogazione, il backup e il ripristino on-line e il controllo flessibile delle operazioni simultanee.

Per riassumere, i DBMS orientati agli oggetti hanno alcune forze, così come alcune debolezze; in generale, la debolezza di un tipo di sistema tende a essere la forza dell'altro (e viceversa). Questo ha portato allo sviluppo di una nuova generazione di sistemi di basi di dati ibridi, i sistemi relazionali a oggetti, che cercano di unire le funzioni di entrambi i modelli (Stonebraker et al., 1990). La maggior parte dei produttori di DBMS ha rilasciato una versione di un sistema di questo tipo. Tali sistemi gestiscono gli oggetti e le regole, l'incapsulazione, il polimorfismo e l'ereditarietà e sono compatibili con i RDBMS. In questa appendice presenteremo una breve descrizione di questo tipo di sistema, compresi vantaggi e svantaggi.

### D.1 Concetti di base e definizioni

Un sistema di base di dati relazionale a oggetti (o ORDBMS) è un motore di base di dati che supporta le funzioni relazionali e orientate agli oggetti in modo integrato (Frank, 1995). L'utente

può, almeno idealmente, definire, manipolare e interrogare tanto i dati relazionali quanto gli oggetti utilizzando interfacce comuni (come SQL). Il modello di dati che soggiace al DBMS è relazionale. Laddove la base di dati può apparire a un programmatore come orientata agli oggetti, anche la memorizzazione e l’elaborazione saranno relazionali; di conseguenza, creare mappature tra i dati relazionali e gli oggetti può diventare molto costoso.

Ai server relazionali e universali sono stati estesi due termini utilizzati per le basi di dati relazionali. Il vecchio termine *relazionale esteso* deriva dal fatto che le prime versioni di questi tipi di sistemi venivano sviluppate aggiungendo funzioni orientate agli oggetti nei DBMS relazionali esistenti. Il termine *server universale*, più moderno, si basa sull’obiettivo dichiarato di gestire qualsiasi tipo di dati, compresi quelli definiti dagli utenti, con la stessa tecnologia server.

D.1.1      **Caratteristiche di un ORDBMS**

Come già citato, lo scopo di un ORDBMS è quello di unire le funzioni migliori dei modelli di base di dati relazionali e a oggetti. Quelle che seguono sono alcune delle funzioni principali solitamente incluse.

- 1. Una versione migliorata di SQL (simile a SQL3) che può essere utilizzata per creare e manipolare tabelle relazionali e tipi di oggetti o classi.
- 2. Supporto delle tradizionali funzioni orientate agli oggetti, come l’ereditarietà, il polimorfismo, i tipi di dati definiti dagli utenti (o astratti) e l’accesso mediante navigazione.

D.1.2      **Tipi di dati complessi**

In SLQ ci sono tre categorie principali di tipi di dati, ovvero NUMERIC, CHARACTER e TEMPORAL. I dati NUMERIC includono DECIMAL e INTEGER, mentre i dati TEMPORAL includono DATE e TIME (Celko, 1995).

Le applicazioni aziendali moderne richiedono spesso la memorizzazione e la manipolazione dei tipi di dati complessi che non vengono gestiti facilmente con i sistemi relazionali; alcuni di questi sono elencati nella Tabella D.1. Per ognuno di essi, viene fornita una breve descrizione e una o più applicazioni tipiche. Per esempio, il tipo di dati Disegni è costituito da oggetti geometrici come punti, linee e cerchi; applicazioni tipiche sono il CAD (*Computer-Assisted Design*), il CAM (*Computer-Assisted Manufacturing*), gli strumenti CASE (*Computer-assisted Software Engineering*) e le immagini grafiche di presentazione (come diagrammi e grafici).

**Tabella D.1**    Tipi di dati complessi.

| Tipo di dati      | Descrizione                        | Esempi di applicazione                                          |
|-------------------|------------------------------------|-----------------------------------------------------------------|
| Immagini          | Rappresentazioni bitmap            | Documenti, fotografie, illustrazioni mediche, impronte digitali |
| Disegni           | Oggetti geometrici                 | Disegni CAD, CAM, CASE e presentazioni                          |
| Spaziali          | Oggetti geografici                 | Mappe                                                           |
| Ricorsivi         | Oggetti nidificati                 | Bill of materials                                               |
| Audio             | Clip audio                         | Musica, animazione, giochi                                      |
| Video             | Segmenti video, filmati            | Video formativi, dimostrazioni di prodotti, film                |
| Array             | Variabili sottoscritte             | Serie temporali, dati multidimensionali, attributi multivalore  |
| A calcolo intenso | Dati che richiedono calcoli estesi | Data mining, strumenti finanziari (per esempio, derivativi)     |

## D.2 SQL avanzato

La caratteristica più potente di un ORDBMS consiste nel suo uso di una versione estesa di un linguaggio di interrogazione relazionale (come SQL3) per definire, acquisire e manipolare i dati (Cattell, 1994; Chaudhri e Zicari, 2001).

### D.2.1 Un esempio

Si supponga che un'azienda voglia creare una relazione chiamata EMPLOYEE per registrare i dati degli impiegati (Figura D.1). Gli attributi tradizionali in questa tabella sono Emp\_ID, Name, Address e Date\_of\_Birth. L'azienda vuole anche conservare una fotografia di ciascun impiegato che verrà utilizzata a scopi identificativi. Questo attributo, il cui tipo è definito da un oggetto, viene chiamato Emp\_Photo nella Figura D.1. Quando i dati di un determinato impiegato vengono recuperati, la foto di questi viene visualizzata con gli altri dati.

**Figura D.1**  
Relazione con un oggetto complesso.

| Emp_ID | Name    | Address  | Date_of_Birth | Emp_Photo                                                                            |
|--------|---------|----------|---------------|--------------------------------------------------------------------------------------|
| 12345  | Charles | 112 Main | 04/27/1973    |                                                                                      |
| 34567  | Angela  | 840 Oak  | 08/12/1967    |  |
| 56789  | Thomas  | 520 Elm  | 10/13/1975    |                                                                                      |

SQL3 fornisce le istruzioni per creare sia le tabelle sia i tipi di dati (o classi di oggetto). Per esempio, l'istruzione CREATE TABLE SQL per EMPLOYEE potrebbe apparire come segue:

```
CREATE TABLE EMPLOYEE
 Emp_ID INTEGER NOT NULL,
 Name CHAR(20) NOT NULL,
 Address CHAR(20),
 Date_of_Birth DATE NOT NULL,
 Emp_Photo IMAGE);
```

In questo esempio, IMAGE è un tipo di dati (o classe) ed Emp\_Photo è un oggetto di quella classe. IMAGE può essere tanto una classe predefinita quanto una definita dall'utente. In quest'ultimo caso, un'istruzione CREATE CLASS SQL viene utilizzata per definire la classe. L'utente può anche definire i metodi che operano sui dati definiti in una classe. Per esempio, un metodo di IMAGE è Scale(), che può essere utilizzato per espandere o ridurre le dimensioni di una fotografia. In questo caso il metodo verrebbe codificato in Java, C++ o Smalltalk e verrebbe incapsulato con la classe IMAGE. I metodi possono essere utilizzati nell'elenco di selezione delle interrogazioni, proprio come gli attributi.

## D.2.2 Indirizzamento del contenuto

Una delle funzioni più potenti del linguaggio SQL relazionale è la sua capacità di selezionare un sottoinsieme di record di una tabella per soddisfare le qualifiche stabilite. Per esempio, nella relazione EMPLOYEE, una semplice istruzione SQL seleziona i record per tutti gli impiegati la cui data di nascita è il 31/12/1940 o è precedente (si veda il Capitolo 9 per alcuni esempi). Per contro, i sistemi di basi di dati a oggetti puri non hanno questa funzionalità di interrogazione; l'accesso agli oggetti avviene tramite navigazione, dove ogni oggetto ha un identificatore univoco.

I prodotti ORDBMS estendono l'indirizzamento del contenuto agli oggetti complessi (Norman e Bloor, 1996). L'indirizzamento del contenuto è quella funzione che permette a un utente di interrogare una base di dati (o una sua tabella) per selezionare tutti i record e/o gli oggetti che soddisfano un determinato insieme di qualifiche.

SQL3 include estensioni per l'indirizzamento del contenuto con i tipi di dati complessi. Vediamo come può essere applicato alla tabella EMPLOYEE. Come detto in precedenza, ogni record di impiegato contiene (o perlomeno è collegato a) una fotografia. Un utente potrebbe voler porre la seguente domanda: "Data la fotografia di una persona, esaminare la tabella EMPLOYEE per determinare se esiste una corrispondenza ravvicinata tra un impiegato e quella foto, quindi visualizzare i dati (foto compresa) dell'impiegato o degli impiegati eventualmente selezionati". Si supponga che l'immagine elettronica di una fotografia venga memorizzata in un punto chiamato My\_Photo. Una semplice interrogazione per questa situazione potrebbe essere la seguente:

```
SELECT *
FROM EMPLOYEE
WHERE My_Photo LIKE Emp_Photo;
```

L'indirizzamento del contenuto in un ORDBMS è una funzione potente che permette agli utenti di cercare le corrispondenze con gli oggetti multimediali, come immagini, segmenti audio o video e documenti. Un'applicazione naturale di questa funzione è la ricerca in una base di dati di impronte digitali o vocali.

## D.3 Vantaggio dell'approccio relazionale a oggetti

Nel caso di applicazioni che richiedono un'elaborazione complessa o una significativa attività di interrogazione su grandi insiemi di dati, l'approccio relazionale a oggetti offre numerosi vantaggi rispetto a quello dei sistemi di basi di dati relazionali (Moxon, 1997).

1. *Traffico di rete ridotto.* Le interrogazioni che impiegano i metodi per analizzare i dati possono essere eseguite per intero sul server, senza che occorra rinviare grandi quantità di dati ai client.

- 2. *Prestazioni delle applicazioni e delle interrogazioni.* I metodi che elaborano grandi quantità di dati possono sfruttare grossi server paralleli per migliorare significativamente le prestazioni.
- 3. *Manutenzione del software.* Il fatto che i dati e i metodi siano memorizzati insieme sul server, semplifica enormemente l'attività di manutenzione del software.
- 4. *Dati integrati e gestione delle transazioni.* L'integrità delle transazioni e la simultaneità, il backup e il ripristino dei dati sono gestiti all'interno del motore della base di dati.

## D.4 Produttori e prodotti di ORDBMS

Un elenco dei principali produttori e prodotti di ORDBMS è fornito nella Tabella D.2. La tabella mostra anche la tecnologia extender (o framework) fornita da ciascun produttore che permette agli utenti di definire i propri tipi di dati, così come i metodi per manipolarli.

**Tabella D.2** Produttori e prodotti di ORDBMS.

| Produttore          | Prodotti ORDBMS        | Tecnologia extender | Sito Web                                                                                                                                                                 |
|---------------------|------------------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IBM                 | DB2 Universal Database | DB2 Extenders       | <a href="http://www_4.ibm.com/software/data/db2/udb/">www_4.ibm.com/software/data/db2/udb/</a>                                                                           |
| Informix            | Dynamic Server         | Data Blades         | <a href="http://www.informix.com/informix/products/integration/datablade/datablade_ds.htm">www.informix.com/informix/products/integration/datablade/datablade_ds.htm</a> |
| Oracle              | Oracle 8               | Data Cartridges     | <a href="http://www.oracle.com/oramag/webcolumns/ora8804.html">www.oracle.com/oramag/webcolumns/ora8804.html</a>                                                         |
| Computer Associates | ODB-II (Jasmine)       | Nessuna             | <a href="http://www.cai.com/products/jasmine.htm">www.cai.com/products/jasmine.htm</a>                                                                                   |

## Bibliografia

Cattell, R., 1994, *Object Data Management: Object-Oriented and Extended Relational Database Systems*, Reading, MA, Addison-Wesley, pp. 303–304.

Celko, J., 1995, *Instant SQL Programming*, Chicago, Wrox Press, pp. 54–55.

Chaudhri, A. B. e Zicari, R., 2001, *Succeeding with Object Databases*, New York, Wiley.

Frank, M., 1995, “Object-Relational Hybrids”, *DBMS* 8 (luglio), pp. 46–56.

Melton, J., 1995, “A Case for SQL Conformance Testing”, *Database Programming & Design* 10 (7), pp. 66–69.

Moxon, B., 1997, “Database out of This Universe”, *DB2 Magazine* 2 (autunno), pp. 9–16.

Norman, M. e Bloor, R., 1996, “To Universally Serve”, *Database Programming & Design* 9 (luglio), pp. 26–35.

Stonebraker, M. et al., 1990, “Third-Generation Database System Manifesto—The Committee for Advanced DBMS Function”, *SIGMOD Record* 19, pp. 3, 31–44.



# Indice analitico

## A

- @ctive data warehouse, 465
- accesso ai dati orientato a SQL, 422
- accorpamento
  - entità, 216–219
  - file fisici, 309–310
  - record, 309
- acquisizioni, 474–475
- Adaptec, prestazioni RAID e, 319
- aggregati
  - scalari, 369
  - vettoriali, 369
- aggregati scalari, 369
- aggregati vettoriali, 369
- algoritmo di hashing, 308
- alias, 273
- ALL, operatore (SQL), 367, 386
- ALTER TABLE (SQL), 346, 354
- ambienti delle basi di dati, 87–88
- ambienti di elaborazione distribuiti, 416–417
- amministrazione dei dati, 88, 111
  - backup e ripristino, 86
  - qualità dei dati e, 84
  - standard e, 84
- amministrazione delle basi di dati, 84, 111, 128, 286, 394
- analisi clickstream, 91
- analisi dei pattern d'uso, 287–288
- analisi della spesa in IT, 56–57
  - parte non discrezionale, 56, 57
- analisti delle basi di dati, 111, 143
- analisti di sistema, 111
- ancora, oggetto, 223
- AND (operatore booleano), 364
- anomalie, 247, 269
  - eliminazione, 248, 269
  - inserimento, 248, 269
  - modifica, 248, 269
- anomalie di eliminazione, 248
- anomalie di inserimento, 248
- anomalie di modifica, 248
- Antony, schema di, 41
- ANY, operatore (SQL), 386
- API di Livello 1, 439
- application server, 116, 413–414
- applicativi, programmi, 88
- applicazioni
  - buy-oriented, 48
  - sell-oriented, 48
- applicazioni delle basi di dati, 75–77
- applicazioni, partizionamento delle, 414–415
- applicazioni, sviluppo delle, 69, 84
- Aranow, E. B., sulle definizioni dei dati, 147
- architettura di non condivisione, 419–420, 420–421
- architettura informativa, 48–50
  - livello applicativo, 48, 50
  - livello database, 48, 49
  - livello dati, 48, 49
  - livello di presentazione, 48, 50
- architetture
  - computer paralleli, 417–421
  - data warehouse, 459–472
  - hardware multiprocessore, 417–420, 420–421
  - Information Systems Architecture (ISA), 99, 119–121
- architetture a processori paralleli/di non condivisione (MMP), 419–420, 420–421
- architetture a tre livelli, 412–414
  - distribuzioni della logica, 426–427
- architetture delle basi di dati



- basi di dati a tre livelli, posizione nell'architettura, 115–116
  - scelta, 287, 320–321
  - schema a tre livelli, 115–118
  - SQL nelle, 339–341
- architetture di computer parallele, 417–421
- architetture di dati a tre livelli, 466–468
- architetture a due livelli, 409–410
  - distribuzioni della logica, 426–427
- architetture hardware multiprocessore, 417–420, 420–421
- architetture multiprocessore non strettamente accoppiate, 419–420, 420–421
- Armstrong, R.
  - sui data mart indipendenti, 462
  - sulle CIF, 464
- asserzione strutturale, 221
  - fatti derivati, 222
  - fatti di base, 222
- asserzioni di azione, 221, 223–224, 246
- asset, 29
- associative, entità (gerundi), 165, 255–258
- associazioni, 163–181
  - attributi sulle, 165
  - collegare gli attributi correlati attraverso, 171
  - concetti di base e definizioni, 164–166
  - dati dipendenti dal tempo e, 174–177
  - definizione, 179–181
  - denominare, 179–181
  - entità associative e, 165–166
  - grado delle, 166–171
  - identificare, 155–157
  - multi-a-molti (M:M)
    - con attributi non di chiave, 293–294
  - multiple, 177–179
  - uno-a-uno, 293
  - relazioni in QBE, 429
  - vincoli di cardinalità e, 171–174
- associazioni binarie, 168, 252–255
  - multi-a-molti, 253
  - uno-a-molti, 252
  - uno-a-uno, 255
- associazioni multiple, 177–179
- associazioni unarie (ricorsive), 168
  - multi-a-molti, 260–261
  - uno-a-molti, 259
- asterisco (\*), carattere jolly, 363
- Atre, S., sui sistemi client/server, 426
- attributi, 73, 157–163
  - a valore singolo, 158
  - associazioni supertipo/sottotipo e, 200
  - collegare tramite le associazioni, 171
  - composti, 158, 160, 249
  - denominare, 160–163
  - derivati, 159–160
  - identificatori, 160
  - memorizzati, 159–160
  - multivalore, 158, 240, 249
  - semplici (atomici), 158
  - sulle associazioni, 165
- attributi composti, 158, 249
- attributi derivati, 159–160
- attributi memorizzati, 159–160
- attributi multivalore, 158, 240, 249
- attributi semplici (atomici), 158
- AUTOCOMMIT (ON/OFF) (SQL), 393
- AVG, funzione (SQL), 361, 362
- aziende
  - ambiente client/server e, 407–408
  - uso delle architetture SMP e MPP, 420–421
- azioni, 223–224

## B

- Babad, Y. M., sui dati mancanti, 292
- back-end, funzioni di, 411
- back-office, funzioni di, 30
- backup e ripristino, 86
- Ballinger, C., sugli indici di join, 307
- Ballou, D. P., sulla qualità dei dati, 84
- Barquin, R., sui data warehouse, 455
- basi di dati, 64, 65, 88
  - analizzare le caratteristiche, 121–122
  - approccio a, 70–75, 82, 85–87
  - aziendali, 79
  - centralizzazione, 91
  - di funzione, 78
  - di gruppo di lavoro, 77–78
  - extranet, 82
  - Internet, 81–82
  - intranet, 82
  - orientate agli oggetti, 90
  - personali, 75–77
  - relazionali, 72–73
  - relazionali/a oggetti (O/R), 421
  - usi, 63–64

basi di dati abilitate per il Web, 81  
 basi di dati aziendali, 79–81  
 basi di dati di funzione, 78  
 basi di dati di gruppi di lavoro, 77–78  
 basi di dati distribuite, 79  
 basi di dati logiche, progettazione, 107, 109, 237–276, 284  
   diagrammi E-R ed EER trasformati in relazioni,  
     248–264  
   associazioni binarie, 252–255  
   associazioni supertipi/sottotipo, 261–264  
   associazioni ternarie (e n-arie), 261  
   associazioni unarie, 258–261  
   entità associative, 255–258  
   entità deboli, 251  
   entità regolari, 249  
   forme normali, 265, 268–271  
   normalizzazione, 264–268  
   unire le relazioni, 271–275  
 basi di dati personali, 75–77  
 basi di dati relazionali, 72–73, 95–96, 241–243  
 BEGIN TRANSACTION (SQL), 392  
 Benbasat, sul modello di Nolan, 7  
 BETWEEN, clausola (SQL), 365  
 bill-of-materials, struttura, 168, 169  
 blocchi di dati, 292  
   scegliere la dimensione dei, 325  
 Bluetooth, standard wireless, 91  
 Bontempo, C. J., sugli indici di join, 307  
 booleani, (logici) operatori, 364–365  
 Boyce-Codd, forma normale di (BCNF), 265  
 Brobst, S., sul partizionamento degli intervalli di chiave,  
   296  
 browser Web, 81–82  
 BSC (Balanced Scorecard), 45

## C

CAD (Computer Aided Design), 46  
 Call-Level Interface (CLI), 423  
 campi, 289–292  
   integrità dei dati, 290–292  
   tipi di dati, 289–290  
 candidate, chiavi, 267–268  
 capabilities, 29  
 caratteri jolly, in SQL, 363  
 cardinalità, vincoli di, 171–174  
   associazioni ternarie e, 174  
   cardinalità obbligatoria uno, 173

  massima, 173  
   minima, 173  
   partecipanti facoltativi e, 173  
 cardinalità massima, 173  
 cardinalità minima, 173  
 cardinalità obbligatoria uno, 173  
 caricare i dati riconciliati, 477–478  
 CASE, sintassi condizionale, in SQL, 392  
 catalogo, 341  
 catena del valore dell'IT, 54–56  
 CHECK (SQL), 347  
 Checkland, su IT/SI, 5  
 Chen, P. P. S., sul modello entità-relazioni, 142  
 chiave aziendale, 275–276  
 chiave esterna ricorsiva, 259  
 chiavi  
   aziendali, 275–276  
   candidate, 267–268  
   composte, 240  
   esterne, 240  
   primarie, 239  
   ricorsive, 259  
   surrogate, 256, 486–487  
 chiavi composte, 240  
 chiavi di indice secondarie, 303, 311  
 chiavi esterne, 240  
   ricorsive, 259  
   vincoli di integrità referenziale e, 244–246  
 chiavi primarie, 123, 239  
 chiavi relazionali, 239–240, 275–276  
 Chisholm, M., sui data warehouse, 494  
 ciclo di vita dello sviluppo dei sistemi (SDLC), 105–108  
 CIM (Computer Integrated Manufacturing), 46  
 clash di hashing, 308  
 client  
   sottili, 413  
   spessi, 409  
 client sottili, 413  
 client spessi, 409  
 client, server, 117  
 client/server, architetture, 117, 408–412  
   a tre livelli, 412–414  
   architetture di computer parallele e, 417–421  
   componenti della logica delle applicazioni, 408–409  
   database server, 411–412  
   file server, 409–411  
 client/server, sistemi, 90, 407–445

distribuzioni della logica, 426–427  
JDBC e, 437–439  
mainframe e, 416–417  
middleware, 421–423  
ODBC e, 437–439  
partizionamento delle applicazioni, 414–415  
problemi nei, 425–426  
project case, 450–452  
Query-by-Example e, 427–437  
sicurezza nei, 423–425  
Visual Basic for Applications e, 444–445  
cluster di entità, 216–219  
Codd, E. F.  
    su modello di dati relazionale, 89, 238  
    tecnologia delle basi di dati relazionali e, 338  
codifica, 424  
    a chiave pubblica, 424  
    della chiave segreta, 424  
commercio elettronico, 408  
COMMIT WORK (SQL), 393  
Common Object Request Broker Architecture (CORBA), 422, 423  
Computer-Aided Software Engineering (CASE), 110–111, 248  
    integrated-CASE (I-CASE), 111  
    lower CASE, 111  
    repository (deposito delle informazioni) e, 111  
    strumenti, 87  
    sviluppo delle basi di dati e, 111  
    upper CASE, 111  
condizione di eguaglianza nei join, 380  
conferma progressiva, 112  
conflitti organizzativi, 87  
conflitti dei blocchi, 325  
controller del disco, operazioni di I/O e, 325–326  
controlli dei valori nulli, 291  
controlli sull'integrità dei dati, 353–354  
controlli, per i file, 310  
controllo degli accessi, per i file, 310  
controllo degli intervalli, 291  
Core API, 439  
Core SQL-99, 338  
Corporate Information Factory (CIF), 464  
costi di conversione, 86  
costi, dimensione dei blocchi di dati, 325  
COUNT, funzione (SQL), 362  
COUNT (\*), funzione (SQL), 362

CREATE ASSERTION (SQL), 346  
CREATE CHARACTER SET (SQL), 346  
CREATE COLLATION (SQL), 346  
CREATE DOMAIN (SQL), 346  
CREATE INDEX (SQL), 348, 358  
CREATE SCHEMA (SQL), 346  
CREATE TABLE (SQL), 246, 346, 346–348, 350  
CREATE TRANSLATION (SQL), 346  
CREATE VIEW (SQL), 346, 351–352  
Cronin, sui rapporti con la clientela, 46  
Croson, sul portafoglio applicativo, 42  
cubo, slicing dei dati, 499  
Customer Relationship Management (CRM), 46, 81

## D

DAM (Database Management), 50  
Darwen, H., su SQL, 340  
data mart logici e architettura di @ctive warehouse, 464–466  
data mart, 73, 461–462  
    data warehouse e, 464–466  
    dipendenti, 462  
    indipendenti, 461–462  
    logici e architettura @ctive warehouse, 464–466  
    logici, 464  
    store operazionali e, 462–464  
data mining, 91, 501–502  
    elaborazioni parallele e, 420  
data scrubbing (data cleansing), 475–476, 478  
data store operazionali, 462–464  
data warehouse, 64, 73, 80, 271, 454–502  
    @ctive, 465  
    architetture  
        a due livelli generica, 460–461  
        a tre livelli, 459, 466–468  
        a tre livelli espansa, 459  
        data mart dipendenti/data store operazionali, 462–464  
        data mart logici e @ctive warehouse, 464–466  
        data warehousing dei data mart indipendenti, 461–462  
azienda, 80, 462  
data mart e, 461–462  
dati memorizzati nei, 468–472  
elaborazione parallela e, 420  
estrarre, trasformare e caricare (ETL) i dati nei, 461  
gap informativo e, 454

- interfacce utente per, 496–502
- linee guida per l'aggiornamento, 477
- livello dei dati derivati, 483–439
- livello dei dati riconciliati, 472–478
- necessità di, vista a livello aziendale, 456–458
- normalizzazione delle tabelle di dimensioni, 492–495
- qualità dei dati e, 84
- schema a stella, 484–490
- sistemi legacy e, 86
- storia dei, 455–456
- trasformazione dei dati, 478–483
- DataBridge, 482, 483
- Date, C. J., su SQL, 340
- dati, 64–65
  - accessibilità, 85
  - coerenza, 83
  - condivisione dei, 69, 83
  - dal Web, 91
  - derivati, 466
  - dopo le operazioni ETL, 472–473
  - uplicazione dei, 69
  - eventi, 468
  - informazioni e, 65
  - inserire/aggiornare/eliminare in SQL, 355–357
  - legacy, 81
  - livello di relazione, 84
  - manca, 291
  - multidimensionali, 90
  - multimediali, 90
  - persistenti, 469–470
  - qualità dei, 84
  - ridondanza, 83
  - standard, 84
  - stato dei, 468–469
  - transienti, 469–470
- dati derivati, 466, 483–439
  - caratteristiche dei, 483–484
- dati di evento, 468
- dati di riferimento, 294
- dati di stato, 468
- dati dipendenti dal tempo, modellare, 174–177
- dati legacy, 81
- dati manca, 291
- dati multidimensionali, 90
- dati multimediali, 90
- dati persistenti, 469–470
- dati riconciliati, 466
- dati transienti, 469–470
- Davenport, sui rapporti con la clientela, 46
- DBA, vista, 394
- DBMS relazionali (RDBMS), 339
  - definizioni degli schemi interni nei, 357–358
  - indici nei, 357–358
- DEFAULT, operatore (SQL), 347
- definizioni, 146–149
  - alternative, 149
  - associazioni, 179–181
  - attributi, 160–163
  - linee guida delle, 146–147
  - tipo di entità, 155
- DELETE, comando (SQL), 353, 356, 393
- DeLoach, A., sull'ottimizzazione dell'elaborazione delle interrogazioni, 326
- denormalizzazione, 292–299
  - uplicazione dei dati, 298–299
  - partizionamento dei record, 298
  - partizionamento orizzontale, 294–298
  - partizionamento verticale, 298–299
- derivazione, 221
- determinanti, 267
- Devlin, B.
  - sull'accesso alle informazioni, 453
  - sulla trasformazione dei dati multicampo, 480
- DeWitt, D., sugli ambienti monoprocesso, 417
- DG/SQL, 338
- diagramma entità-associazione (E-R), 70, 114, 149
  - esempio, 149–151, 194–195
  - notazione, 151–152
- dimensione delle righe, dimensione dei blocchi di dati, 325
- dimensione multivalore, 493
- dimensioni conformi, 490
- dipendenti, data mart, 462
- dipendenza dei dati dai programmi, 69
- dipendenza funzionale parziale, 269
- dipendenza transitiva, 269, 270
  - nell'integrazione delle viste, 274
- dipendenze
  - transitive, 269, 270, 274
- dipendenze funzionali, 265–268
  - chiavi candidate e, 267–268
  - determinanti, 267
  - parziali, 269
- discriminante di sottotipo, 209–210
- DISTINCT (SQL), 386

Distributed Component Object Model (DCOM), 423  
 distribuzioni logiche, negli ambienti client/server, 426–427  
 dizionario dei dati, 394–395  
 DOLAP (Database OLAP), 499  
 domini  
     attributi e, 243–244  
 drill-down, 499–500  
 DROP, comando (SQL), 346  
 DROP INDEX (SQL), 358  
 DROP SCHEMA (SQL), 346  
 DROP TABLE (SQL), 346, 354  
 DROP VIEW (SQL), 346, 355  
 Drury, sul modello di Nolan, 7  
 DSS (Decision Support System), 43, 44  
 duplicazione, dati, 298–299  
 Dutka, A. F.  
     sulla dipendenza funzionale, 265  
     sulle chiavi candidate, 267  
 Dvorak, sull'IT, 22  
 dynaset, QBE, 430, 431, 432

## E

Earl, su SI/IT, 4, 15  
 EIS (Executive Information System), 44  
 ESS (Executive Support System), 43  
 EIS/ESS, strumenti applicativi, 44  
 elaborazione delle transazioni, monitor TP, 415  
 elaborazione di base di dati parallela, RAID e, 313–319  
 elaborazione operativa, 458  
 Electronic Data Interchange (EDI), sistemi, 82  
 Elmasri, R.  
     sui vincoli di completezza e separazione, 206  
     sulle gerarchie supertipo/sottotipo, 211  
     sulle istanze di associazione, 164  
 END TRANSACTION (SQL), 392  
 English, L. P.  
     su TQM e la gestione dei dati, 476  
     sull'estrazione dei dati, 474–475  
 entità, 70, 71, 153–157  
     nei diagrammi E–R, 153–157  
     regolari, 249  
     deboli, 249  
 entità deboli, 249  
 entità regolari, 249  
 equi-join, 380–381  
 ereditarietà degli attributi, 198  
 ERP (Enterprise Resource Planning), 13, 30, 46

espressioni, in SQL, 361, 392  
 estensibilità, 408  
 estensioni, 299  
 estrazione dei dati, 474–475  
 estrazione progressiva, 474  
 estrazione statica, 474  
 ETL (estrarre, trasformazione e caricare), 461  
     i dati dopo, 472–473  
     processo, 473–473  
 Evans, M., sul modello dei dati entità-associazione, 142  
 EXISTS, operatore (SQL), 387–388  
 Extract (strumento di conversione), 482, 483  
 extranet, 82

## F

fatti, 147  
     derivati, 222  
 fatti derivati, 222  
 fatti di base, 222  
 fattore di blocco, 292  
 fattori di pianificazione strategici, nella pianificazione dei sistemi informativi, 100  
 Ferguson, M.  
     sugli ambienti paralleli, 417, 418  
     sulla elaborazione simmetrica, 418, 419  
 file del sistema operativo, 299  
 file fisici, 299–310  
     accorpamento, 309–310  
     controlli per, 310  
     estensione e, 299  
     memorizzazione sequenziale, 301  
     organizzazione dei, 300–309  
     puntatori, 301  
     tablespace e, 299  
 file server, 409–411  
 Finkelstein, R., sulla denormalizzazione, 293  
 firewall, 82  
 firme digitali, 425  
 Fleming, C. C., sul modello di dati relazionale, 238  
 flessibilità, 408, 413  
 FM (facility management), 36, 38, 39  
 FOREIGN KEY REFERENCES (SQL), 246  
 forme normali, 264–268  
     di Boyce-Codd, 265  
     prima, 265, 268  
     seconda, 265, 268–269  
     terza, 265, 269–271

Frazer, W. D., sulle basi di dati a oggetti/relazionali (O/R), 421  
 frequenza degli accessi, 287  
 FROM, clausola (SQL), 351, 370, 382, 390  
 front-end, funzioni, 411  
 front-office, funzioni, 30  
 Fry, J. P., sul modello di dati entità-associazione, 142  
 funzioni, in SQL, 361–362, 400  
 funzioni analitiche, proposte per SQL-99, 395–396  
 funzioni aziendali, 102–103

## G

Gantt, diagramma di, 113  
 gap informativo, 454  
 generalizzazione, 202–205  
 Genio Suite, 482, 483  
 George, J.F.  
   su SDLC, 105  
   sull'architettura client/server, 117  
   sulla modellazione dei dati, 142  
   sulla pianificazione delle matrici, 104  
 gerarchie  
   elle dimensioni, 493–495  
   supertipo/sottotipo, 211–213  
 gestione, attività di, 6–8  
   controllo di, 6  
   controllo operativo, 6  
   pianificazione strategica, 6  
   problemi di, 9  
 Gibson, modelli di information technology, 6–11  
 Gottesdiener, E., sulle regole aziendali, 144, 145  
 Gould, J. D., su QBE, 428  
 grado, delle associazioni, 166–171  
 grana, delle tabelle di fatti, 487–488  
 GRANT, istruzione (SQL), 350  
 Gray, J., sugli ambienti monoprocesso, 417  
 Greenblatt, D., su QBE, 428  
 Grimes, S., sulle basi di dati orientate agli oggetti, 90  
 GROUP BY, clausola (SQL), 368, 369–370  
 guidato degli eventi, sistema, 445

## H

Hackathorn, R. D., sul data warehousing, 455  
 Hanson, H. H.  
   sulla dipendenza funzionale, 265  
   sulle chiavi candidate, 267

HAVING, clausola (SQL), 368, 370, 384  
 Hirschheim, sul management dei sistemi informativi, 9  
 Hoffer, J. A.  
   su SDLC, 105  
   sui dati mancanti, 291  
   sull'architettura client/server, 117  
   sulla gestione dei team di sviluppo, 112  
   sulla modellazione dei dati, 142  
   sulla pianificazione delle matrici, 104  
 HOLAP (Hybrid OLAP), 499  
 Holmes, J., sull'ottimizzazione delle interrogazioni, 326  
 Holwell, su IT/SI, 5  
 Hurwitz, J.  
   sui sistemi legacy sui mainframe, 416, 417  
   sul middleware, 421

## I

identificare il possessore, 155  
 identificare le associazioni, 153–157  
 identificatori, 155, 160  
   assegnati, 256  
   composti, 160  
   intelligenti, 160  
   non assegnati, 255  
   oggetti, 275  
   parziali, 155  
   surrogati, 256  
 identificatori composti, 160  
 identificatori intelligenti, 160  
 identificatori surrogati (o chiavi surrogate), 256, 486–487  
 IDM, 338  
 IF-THEN-ELSE, elaborazione logica, 392  
 Imhoff, C.  
   sui data store operazionali, 463  
   sulle CIF, 464  
 implementazione delle basi di dati, 108, 109  
 IN e NOT IN, liste (SQL), 367–368, 386  
 indice di join, 305–308  
 indice gerarchico, 304  
 indice primario non univoco (NUPI), 304  
 indice secondario non univoco (NUSI), 304  
 indici, 287, 303–309  
   bitmap, 304, 312, 477  
   chiavi primarie, 303  
   chiavi secondarie, 303, 311  
   di join, 305–308  
   gerarchici, 304

- indici di chiave univoca, 311
  - multipli, 304–305
  - nei DBMS relazionali, 357–358
  - primari, 303–304
  - regole per la scelta, 312–313
  - secondari, 303–304
- indici bitmap, 304, 313, 477
- indici di chiave primaria, 303–304
- indici di chiave univoca, 311
- indici primari, 303–304
- indici primari univoci (UPI), 303
- indici secondari, 303–304
- indici secondari univoci (USI), 304
- indipendenti, data mart, 461–462
- indipendenza dei dati, 83, 85
- indipendenza dei dati dai programmi, 83
- InfoRefiner, 482, 483
- Information Systems Architecture (ISA), 99
  - soddisfazione delle esigenze degli utenti, 119–121
- Information technology, 3–4, 6, 11, 16–17, 18
- informazioni, 65
  - e dati, 65
- ingegneria delle informazioni, 99–100
  - fase della pianificazione, 100–104
- Ingres, 238, 338
- Inmon, W. H.
  - sui data mart indipendenti, 462
  - sul warehousing, 455
  - sulle basi di dati normalizzate, 293
  - sulle CIF, 464
- INPUT, comando (SQL), 356
- input/output (I/O), operazioni di
  - dimensione dei blocchi di dati e, 325
- inquinamento dei dati, 475–476
- INSERT, comando (SQL), 355, 393
- installazione
  - costo e complessità della, 86
- Integrated-CASE (I-CASE), 110
- integrità
  - dati, 78, 290–292, 353–354
  - entità, 243–244
  - referenziale, 291, 353
- integrità (strumenti di pulizia), 483
- integrità dei dati, 78, 290–292
- integrità delle transazioni, 392–394
- integrità referenziale, 291, 353
- intelligenza artificiale, 91

- interfacce grafiche utente (GUI), 427
- interfacce utente, 88, 496–502
  - metadati e, 496–497
  - strumenti di data mining, 501–502
  - strumenti di interrogazione, 498
  - strumenti OLAP, 498–500
  - visualizzazione dei dati e, 502
- Internet, basi di dati, 81–82, 41
- interrogazioni, 126
  - pass-through, 433–435
  - unire, 391
- interrogazioni di join ricorsivi, 433
- interrogazioni parallele, 418
- interrogazioni pass-through, 433–435
- intervalli
  - in SQL, 365
- intranet, 81–82
  - protezione e, 424
- istanza, 73
- istanze delle associazioni, 164
- istanze di entità, 153
- IT, 4–22, 26–58
- IT Dynamic Capabilities, 33, 35
- IT Governance, 53, 55
- ITC (IT Capabilities), 35

## J

- Java 2 Enterprise Edition (J2EE), 116
- Java Database Connectivity (JDBC), 402
  - classi, 423
  - collegare le tabelle memorizzate nella base di dati con, 437–439
- Johnston, T., sulla specifica delle chiavi primarie, 275
- join, 380–392
  - cartesiani, 381
  - equi-join, 380–381
  - esterni, 382–383
  - interni, 381
  - interrogazioni di self-join (join ricorsivi), 433
  - multipli, 384
  - naturali, 381–382
  - UNION JOIN, 382, 384
- join esterni, 382–383
- join naturali, 381–382
- Jordan, A., sul data warehousing, 84

**K**

Kaplan, sulle BSC, 45  
 Keen, sull'IT, 22, 33  
 Kenney, sull'infrastruttura IT, 33  
 Kerberos, server di protezione, 425  
 Keuffel, W., sul middleware, 423  
 Kimball, R.  
   sui data mart indipendenti, 462  
   sui data warehouse, 494  
   sulle tabelle di fatti, 486  
 King, sul modello di Nolan, 7  
 Klahr, sui rapporti con la clientela, 46  
 KOWS, 47  
 KPI (Key Performance Indicators), 45, 51  
 Kraemer, sul modello di Nolan, 7  
 Kurka, A., sull'ottimizzazione delle interrogazioni, 326

**L**

Lambert, B., sui data warehouse, 63  
 LaRue, M., su ODBC, 422  
 librerie di componenti, 416  
 linguaggi di programmazione visuali, 427–428  
 linguaggio di controllo dei dati (DCL), 342  
 linguaggio di definizione dei dati (DDL), 341  
 linguaggio di manipolazione dei dati (DML), 341  
 linguaggio delle specifiche delle regole aziendali, 224  
 Linthicum, D.S.  
   su ODBC, 423  
   sulle applicazioni client/server, 425  
 livello dei dati  
   derivati, 483–439  
   riconciliati, 472–478  
 Livello 2, 439  
 Local Area Network (LAN), 409–410, 41  
   basi di dati di gruppo di lavoro e, 77  
   database server, 411–412  
   file server e, 409–410  
 logica dell'elaborazione, 408, 409, 414  
 logica della memorizzazione, 409, 414  
 logica della presentazione, 408, 409  
 logica delle applicazioni, 408–409  
 lower CASE, 111

**M**

macro, in Microsoft Access 2000, 444  
 mainframe, sistemi client/server e, 416–417

management del portafoglio IT, 53–54  
 Manufacturing Resources Planning (MRP-II), sistemi, 79  
 manutenzione  
   basi di dati, 108, 109  
   dati, 392–394  
   programmi, 70, 85  
 manutenzione dei programmi, 70, 85  
 marcatura temporale, 176  
 Martin, J., sui data warehouse, 454  
 maschere, 125  
 Materials Requirements Planning (MRP), sistemi, 79  
 matrici, pianificazione, 103–104  
 MAX, funzione (SQL), 362  
 memorizzazione sequenziale, 301  
 Message-Oriented Middleware (MOM), 422  
 metadati, 65–67, 497–498  
   architettura a tre livelli e, 467  
 Meyer, A., sull'architettura dei data mart indipendenti, 462  
 Microsoft Access, 123  
   interfacce, 338  
   QBE in, 429–431  
 Microsoft Access 2000  
   dynaset in, 430, 431, 432  
   interrogazioni a tabelle multiple in, 429–430  
   interrogazioni basate su altre interrogazioni, 433–435  
   interrogazioni pass-through SQL in, 435–437  
   interrogazioni singole, 429  
   macro in, 444  
   QBE in, 429–431  
   qualifiche dei record, 432–433  
   riquadro di progettazione dell'interrogazione, 429  
   self-join in, 433  
   VBA e, 444–445  
 middleware, 421–423  
 middleware pubblica/iscritti, 421  
 MIN, funzione (SQL), 361, 362  
 MIS (Management Information System), 44  
 misure di protezione specifiche dei produttori, 425  
 modellazione aziendale, 101–103, 106  
 modellazione dei dati aziendali, 98, 99  
 modelli di information technology, 6–11  
   di Nolan e Gibson, 6–7  
   di quarta generazione, 21–22  
   di terza generazione, 10–11  
   hardware, 9, 38  
   metodologia, 10  
   software, 9



modelli di rappresentazione delle basi di dati, 26–57  
     business, 25–27  
     organizzazione, 25–27  
     tecnologia, 25–27  
 modelli delle basi di dati  
     basi di dati di gruppo di lavoro e, 78  
     di rete, 320  
     gerarchico, 320  
     multidimensionale, 321  
     orientato agli oggetti, 320  
     relazionale, 320  
 modello di dati aziendale, 70–72, 103, 467  
 modello di base di dati di rete, 320  
 modello di base di dati gerarchico, 320  
 modello di base di dati multidimensionale, 321  
 modello di base di dati relazionale, 320  
 modello di dati, 115  
     concettuale, 107  
     importanza del, 142  
 modello di dati concettuale, 107, 110, 142  
 modello di dati relazionale (modello relazionale),  
     89, 90, 142, 238–242  
     chiavi, 239–240  
     componenti del, 238  
     definizioni, 238–241  
     struttura dei dati, 239  
     vincoli di integrità nel, 244–246  
 modello entità-associazione (modello E-R),  
     142, 149–152, 181–183  
     costrutti, 152–163  
         associazioni, 163–181  
         attributi, 157–163  
         entità, 153–157  
     dipendente dal tempo, 174–177  
     notazione, 142  
 modello entità-associazione esteso (EER),  
     198–227, 232–233  
     accorpamento delle entità e, 216–219  
     associazioni supertipo/sottotipo, 198–205  
     concetti di base e notazione, 199–202  
 modularità, 36  
 Moriarty, T.  
     sui repository, 111  
     sui dati derivati, 483  
     sulle regole aziendali, 144, 227  
 motori delle basi di dati, 117

Mountain View Community Hospital (project case)  
     basi di dati relazionali per, 95–96  
     data warehouse per, 507–508  
     diagramma EER per, 232–233  
     diagramma E-R per, 194–195  
     processo di sviluppo della base di dati, 132–138  
     progettazione fisica della base di dati, 333–334  
     progettazione logica della base di dati, 284  
     progettazione SQL della base di dati, 406  
     sistemi client/server e, 450–452  
     SQL e basi di dati per, 377  
 Multidimensional OLAP (MOLAP), 499

## N

n-arie, associazioni, 261  
 Navathe, S. B.  
     sui vincoli di completezza e separazione, 206  
     sulle gerarchie sottotipo/supertipo, 211  
     sulle istanze delle associazioni, 164  
 Nelson, sugli ERP, 46  
 Network Attached Storage (NAS), 319  
 n-livelli, architetture a, 412–414  
     distribuzioni logiche, 426–427  
 Nolan, modelli di information technology, 6–11, 42  
 nome  
     associazioni, 179–181  
     attributi, 160–163  
     dati, 146–147  
     tipi di entità, 155–157  
 nomi dei dati, 146–147  
 nomi delle associazioni, 155  
 normalizzazione, 107, 142, 238, 264–268, 479  
     chiavi, 265–268  
     delle tabelle di dimensioni, 492–495  
     dipendenza funzionale, 265–268  
     passaggi della, 265  
     sui dati di riepilogo, 271  
 Norton, sulle BSC, 45  
 NOT, operatore (SQL), 246, 364, 386  
 notazione  
     modello E-R, 142, 151–152  
     per le associazioni sottotipo/supertipo, 199–202  
 Notowidigdo, sui sistemi informativi strategici, 12  
 Novell, 40  
 NULL (SQL), 246, 353, 369, 383  
 nulli, valori, 244

**O**

Object Request Broker (ORB), 422  
 oggetti aziendali, 101  
 oggetti corrispondenti, 223  
 oggetti di dati  
   denominare e definire, 142, 146–147  
   vincoli sui, 143  
 OLA (Operational Level Agreements), 31, 36, 52, 53  
 OLAP, funzioni proposte per SQL-99, 44, 395–398  
 OLE-DB (Microsoft), 423  
 omonimi, nell'integrazione delle viste, 274  
 On-Line Analytical Processing (OLAP), 73, 498–500  
 Open Database Connectivity Standard (ODBC),  
   402, 422, 437–439  
 operations, 39, 40  
 operatori di confronto, in SQL, 363  
 operazione di aggiornamento, 74  
 OR (operatore booleano), 364  
 Oracle, 338  
   accorpate i file 309–310  
   comandi di definizione della base di dati, 310  
   Partition View, 298  
   SQL e, 338  
   SQL\*Loader, 356  
   strumento di spiegazione, 357  
   tablespace, 299  
   terminologia dei file fisici, 299  
 Oracle8i  
   metodi di partizionamento orizzontale, 297–298  
   tipi di dati in, 289  
   viste DBA, 394  
 ORDER BY, clausola (SQL), 368–369  
 organizzazione dei file sequenziale, 301  
 organizzazione di file di hashing, 308–309  
 organizzazione di file indicizzata, 303–309  
 organizzazioni di file, 287, 300–309  
   confronto tra le, 304, 309  
   di hashing, 308–309  
   fattori nelle, 300–301  
   indicizzate, 303–309  
   sequenziali, 301  
 orientata agli oggetti, base di dati, 90, 275  
 orientato agli oggetti, modello di base di dati, 320  
 ottimizzazione delle interrogazioni, 287, 321–328  
   aggirare l'ottimizzazione automatica delle  
     interrogazioni, 324  
   bilanciamento di I/O tra i controller dei dischi, 325–326

dimensione dei blocchi di dati e, 325  
 elaborazione parallela delle interrogazioni e, 323–324  
 linee guida per la progettazione delle interrogazioni,  
   326–328  
 ottimizzazione, prestazioni delle interrogazioni,  
   287, 321–328

**P**

pagine, 292  
 pagine Web, 408  
   protezione client/server e, 423–425  
 paniere di mercato, analisi del, 487  
 Park, E. K., sul modello di dati entità-associazioni, 142  
 partecipanti facoltativi, nella cardinalità minima, 173  
 partizionamento  
   applicazioni, 414–415  
   metodi di Oracle 8i, 296–298  
     record, 298  
     verticale, 298–299  
     orizzontale, 294–298  
   vantaggi e svantaggi, 297  
 partizionamento composto, 297  
 partizionamento dei record, 298  
 partizionamento di hash, 297, 298  
 partizionamento orizzontale, 294–298  
   metodo composto, 297  
   metodo dell'intervallo di chiave, 297  
   metodo di hash, 297, 298  
 partizionamento verticale, 298–299  
 Persistent Stored Modules (SQL/PSM), 396–397  
 personal computer (PC), 75  
 Personal Data Assistant (PDA), 75, 91  
 personale, specializzato, 86  
 PERT, diagramma di, 113  
 petabyte, 64  
 pianificazione  
   top-down, 99  
 pianificazione dei sistemi informativi, 100–104  
   fattori di pianificazione strategici, 100  
   matrici, 103–104  
   modello aziendale, 101–103  
   oggetti aziendali, 101  
 pianificazione delle matrici, 103–104  
 pianificazione top-down, 99  
 Poe, V., sul vantaggio di competitività, 454  
 portafoglio applicativo, 41–48  
   portafoglio direzionale 43

- portafoglio gestionale, 43, 45
- portafoglio operativo, 45
- portafoglio strategico, 43
- portafoglio transazionale, 45
- portafoglio IT, 29–35
  - componenti del sistema informativo, 29–35
  - management del, 53–54
  - punto di vista del ricercatore, 29–35
  - punto di vista interno, 35–54
    - application management, 36
    - modularità, 36
    - portafoglio applicativo, 41–48
    - portafoglio strategico, 43
    - portafoglio tecnico, 38–41
    - portafoglio tecnologico, 36, 38
    - scomponibilità, 36
- portafoglio strategico, 43
- portafoglio tecnico, 38–41
- portafoglio tecnologico, 36, 38
- posizione della base di dati, architettura a tre livelli per, 115–116
- prima forma normale (1NF), 265, 268
- PRIMARY KEY, operatore (SQL), 246, 347
- procedure, in SQL, 400–401
- processo di sviluppo delle basi di dati, 97–128, 132–138
  - architetture a tre schemi per, 114–115
  - basi di dati a tre livelli, posizione nell'architettura, 115–116
  - CASE e, 110–111
  - ciclo di vita dello sviluppo dei sistemi, 105–108
  - creazione di un prototipo, 109
  - esempio, 132–138
  - gestione del personale, 111–112
  - nello sviluppo dei sistemi informativi, 98–99
    - approcci alternativi, 108–110
  - repository e, 110–111
- progettazione delle basi di dati, 122–125
- progettazione fisica della base di dati, 108, 109, 285–328, 333–334
  - campi, 289–292
    - integrità dei dati, 290–292
    - tipi di dati, 289–290
  - denormalizzazione, 292–299
  - file fisici, 286, 299–310
    - organizzazione di file, 287, 300–309
  - indici, 287, 311–313
  - ottimizzazione delle prestazioni delle interrogazioni, 287, 321–328

- processo di, 286–287
  - volume dei dati e analisi d'uso, 287–288
- RAID e, 313–319
- record fisici, 287, 292
  - scelta dell'architettura, 287, 320–321
- progetti, 111–112
- programmatori, 111
- protezione
  - dati, 78
  - nei sistemi client/server, 423–425
  - viste e, 350
- protezione basata sull'autenticazione degli utenti, 423
- prototipi, 109
- pulizia dei dati, strumenti di, 483
- puntatori, 299
- punti di revisione, 112

## Q

- quarta forma normale (4NF), 265
- Query-by-Example (QBE), 126, 427, 427–437
  - definizione dei dati, 429
  - dynaset in, 430, 431, 432
  - gerarchia di usabilità di Microsoft Access, 429–431
  - in Access 2000, 429–431
  - interrogazioni basate su altre interrogazioni, 433–435
  - interrogazioni pass-through SQL, 433–435
  - interrogazioni singole, 429
  - interrogazioni su tabelle multiple, 430–431
  - relazioni, 429
  - selezione dei record, 432–433
  - storia e importanza di, 428
- Quinlan, T., sulle stored procedure, 412
- quinta forma normale (5NF), 265

## R

- RAID, 313–319
- Rapid Application Development (RAD), 109
- record dinamici, in QBE di Access 2000, 430
- record fisici, 287, 292, 309
- Redundant Array of Inexpensive Disks (RAID), 313–319
  - livelli di, 313–319
  - prestazioni di, 319
  - rischi di, 314
- REFERENCES, clausola (SQL), 347, 353
- refresh, modalità, 477
- regola della sovrapposizione, 208
- regola della specializzazione parziale, 206

regola della specializzazione totale, 206  
 regola di integrità delle entità, 244  
 regola di separazione, 209  
 regole aziendali, 102, 142, 143, 219–227  
   ambito delle, 145–146  
   caratteristiche delle buone, 145  
   classificazione delle, 220–221  
     asserzioni di azione, 221, 223–224  
     asserzioni strutturali, 221  
     derivazione, 221  
   definizioni e, 147  
   identificare e verificare, 227  
   nomi di dati e, 146–147  
   paradigma, 144  
   raccolta, 146  
   rappresentare e applicare, 224–227  
 Relational OLAP (ROLAP), 498  
 relazionali/a oggetti (O/R), basi di dati, 421  
 relazioni, 239  
   associative, 255–258  
   ben strutturate, 247–248  
   diagrammi E-R ed EER trasformati in, 248–264  
     associazioni binarie, 252–255  
     associazioni supertipo/sottotipo, 261–264  
     associazioni ternarie (e n-arie), 261  
     associazioni unarie, 258–261  
     entità associative, 255–258  
     entità deboli, 251  
     entità regolari, 249  
   proprietà delle, 240  
   unire (integrazione delle viste), 271–275  
 relazioni associative, 255–258  
 relazioni ben strutturate, 247–248  
 Remote Method Invocation (RMI), 421  
 Remote Procedure Call (RPC) asincrono, 421  
 Remote Procedure Call (RPC) sincrono, 421  
 report, 126  
 repository, 83, 84, 88, 111  
 Research Laboratory di IBM, SQL e, 338  
 reti, 408  
 REVOKE, istruzione (SQL), 350  
 riconciliazione, dati, 472–478  
   acquire/estrarre, 474–475  
   caricamento e indicizzazione, 477–478  
   pulizia e scrubbing, 475–476  
   strumenti di supporto, 480–483  
   trasformazione, 478–483

Riggs, S., sulla raccolta dei dati Web, 91  
 riservatezza, viste e, 351  
 Rockart, sull'information technology, 3  
 Rogers, U., sulla denormalizzazione, 293  
 Ross, R.G., sulle regole aziendali, 224, 227  
 Rotating Parity Array (RAID-5), 318  
 routine, 398  
   funzioni, 400–401  
   procedure, 400–401  
 RPC sincrono, 421  
 Rudin, K., sulle architetture hardware multiprocessore, 420  
 Rugiadini, sulla definizione di sistema informativo, 27

## S

Salin, T., sui nomi di dati, 146, 147  
 Saracco, C. M., sull'indicizzazione dei join, 307  
 scalabilità, 408, 413  
 schema a stella, 484–490  
   componenti, 484–490  
   normalizzazione delle tabelle di dimensioni e, 492–495  
   varianti dello, 490–492  
 schema concettuale, 107, 113  
 schema fisico, 114  
 schemi, 341  
   a snowflake, 494  
   a stella, 484–490  
   concettuali, 107, 113  
   esterni (viste utente), 83, 107, 115  
   fisici (interni), 115  
 schemi interni, 115  
 Schumacher, R.  
   sugli indici bitmap, 304  
   sulla tecnologia server, 323  
 scomponibilità, 36  
 scomposizione funzionale, 101  
 scrub, dei dati, 84  
 seconda forma normale (2NF), 265, 268–269  
 Secure Sockets Layer (SSL), 424  
 segmentazione delle dimensioni, 496  
 SELECT, istruzione (SQL),  
   349, 350, 358–362, 367, 369–370, 380, 384  
 selezione (subsetting), 479  
 self-join (join ricorsivo), interrogazioni, 433  
 server  
   application, 117, 413–414  
   aziendali, 117  
   basi di dati (motori dei), 77, 117, 411–412

- di certificati, 424
  - elaborazione parallela delle interrogazioni e, 323
  - file, 409–410
  - universali, 90
- server aziendali, 117
- server universali, 90
- server Web, 117
- server di base di dati, 77, 117, 411–412
  - JDBC e tabelle esterne memorizzate in, 440–444
- service delivery, 50–51
- service management, 51–53
- Shah, R., sui dispositivi di memorizzazione, 319
- SI/IT, terminologia, 4–6
  - classificazione, 11–12
  - strategia, 16–21
- sicurezza delle sessioni, pagine Web dinamiche e, 424
- Simon, sulle esigenze informative, 41
- Sinha, Atish P., sulla modellazione dei dati orientati all'oggetto, 523
- sinonimi, nell'integrazione delle viste, 273
- sistema di gestione delle basi di dati, 86
  - evoluzione, 88–91
- sistema operativo, 458–459
- sistemi delle informazioni, 459
- sistemi di elaborazione dei file, 67–70, 89
  - svantaggi, 68–70
- sistemi distribuiti, 408
- sistemi informativi, 6–22
  - definizione, 27–29
  - direzionali, 13
  - evoluzione
    - anni '50-'80, 6–9
    - fasi della, 7
    - modello di terza generazione, 10–11
  - fattori di successo, 13–15
  - implicazioni manageriali, 15–16
- sistemi legacy, 86
- sistemi monoprocesso, 417
- sistemi multiprocesso strettamente accoppiati, 417–419, 420–421
- SLA (Service Level Agreement), 31, 36, 52, 53
- Smith, R., su VBA, 444
- snowflake, schemi a, 494
- Song, L.-Y., sul modello di dati entità-associazioni, 142
- sotto-interrogazioni, 384–390
  - correlate, 389–390
  - non correlate, 388
- sotto-interrogazioni correlate, 389–390
- sottotipo, 198
  - attributi e, 198
  - separazione, 209
  - sovrapposizione, 209, 210
- Sowa, J. F., su ISA, 99
- specializzazione, 202–205
- SQL, 85, 90, 337–371, 379–402
  - ALL, 367, 386
  - ALTER TABLE, 346, 354
  - ambiente, 341–345
  - AND, operatore, 364
  - ANY, 386
  - AUTOCOMMIT (ON/OFF), 393
  - AVG, funzione, 361, 362
  - BEGIN TRANSACTION, 392
  - BETWEEN/NOT BETWEEN, clausola, 365
  - CASE, sintassi condizionale, 392
  - CHECK, 347
  - COMMIT WORK, 393
  - conformità e, 337–338
  - controlli dell'integrità dei dati, creazione, 353–354
  - COUNT, funzione, 362
  - COUNT (\*), funzione, 362
  - CREATE ASSERTION, 346
  - CREATE CHARACTER SET, 346
  - CREATE COLLATION, 346
  - CREATE DOMAIN, 346
  - CREATE INDEX, 348, 358
  - CREATE SCHEMA, 346
  - CREATE TABLE, 246, 346–348, 350
  - CREATE TRANSLATION, 346
  - CREATE VIEW, 346, 351
  - data warehouse e, 498
  - DEFAULT, 347
  - definizione degli schemi interni in RDBMS, 357–358
  - definizione delle basi di dati, 346
    - CREATE comandi, 346
    - creazione della tabella, 346–348
    - utilizzare e definire le viste, 349–353
  - definizioni delle tabelle, 246
  - definizioni delle viste, 349–353
  - DELETE, comando, 353, 356, 393
  - dinamico, 401–402, 415
  - DISTINCT, 386
  - dizionario dei dati, 394–395
  - DROP, comando, 346

- DROP INDEX, 358
- DROP SCHEMA, 346
- DROP TABLE, 346, 354
- DROP VIEW, 346, 355
- elaborare le tabelle multiple, 380–392
  - equi-join, 380–381
  - espressioni condizionali, 392
  - join esterni, 382–383
  - join naturali, 381–382
  - sotto-interrogazioni, 384–390
  - tabelle derivate, 390–391
  - UNION JOIN, 384
  - unire le interrogazioni, 391
- elaborare le tabelle singole, 358–371
  - caratteri jolly, 363
  - espressioni, 361
  - funzioni, 361–362
  - intervalli, 365
  - operatori booleani, 364–365
  - operatori di confronto, 363–364
- END TRANSACTION, 392
- estensioni a, 395–398
- EXISTS, operatore, 387–388
- FOREIGN KEY REFERENCES, 246
- FROM, clausola, 351, 370, 382, 390
- funzioni analitiche (proposte), 395–398
- GRANT, istruzione, 350
- GROUP BY, clausola, 368, 369–370
- HAVING, clausola, 368, 370, 384
- IF-THEN-ELSE, elaborazione logica, 392
- IN, 386
- IN e NOT IN, liste, 367–368
- incorporato, 401–402
- INPUT, comando, 356
- input in batch, 356
- inserire, aggiornare ed eliminare i dati, 355–357
- INSERT, comando, 355–356, 393
- integrità delle transazioni, 392–394
- istruzioni di controllo, 397
- LIKE, parola chiave, 363
- MAX, funzione, 361
- MIN, funzione, 361
- nell'architettura delle basi di dati, 339–341
- NOT, operatore, 246, 364, 386
- NOT EXISTS, 382
- notazione nei comandi, 343
- NULL, 246, 353, 369, 383
- OR, operatore, 364
- ORDER BY, clausola, 368–369
- Persistent Stored Modules, 396–397
- PRIMARY KEY, operatore, 246, 347
- REFERENCES, clausola, 347, 353
- REVOKE, istruzione, 350
- routine, 400–401
- SELECT, istruzione,
  - 351, 358–362, 367, 369–370, 380, 384
- SQL EXEC, 401
- SQLSTATE, dichiarazione, 397
- statico, 415
- storia di, 338–339
- SUM, funzione, 361, 362
- tipi di dati definiti dall'utente e, 395
- trigger, 379, 398–400
- UNION, clausola, 391
- UNIQUE, 347
- UPDATE, istruzione, 353, 357, 393
- vantaggi di, 340
- viste DBA, 394
- WHERE, clausola,
  - 350, 363, 368, 370, 380, 381, 384, 390
- WINDOW, clausola, 396
- SQL Call Level Interface (SQL/CLI), 402
- SQL dinamico 401–402, 415
- SQL Server, 438
- SQL statico, 415
- SQL\*Plus, 387
- SQL/86, 338
- SQL/89, 338
- SQL-92 di livello Entry, 338
- SQL-92, 337, 338, 340, 346
- SQL-99, 337, 338
  - Amendment 1 proposto (funzioni analitiche), 395
  - estensioni di programmazione, 396–398
  - miglioramenti, 395–398
  - procedure e sintassi della creazione di funzioni, 400–401
- SQLSTATE, dichiarazione, 397
- Storage Area Network (SAN), 319
- stored procedure, 412
- Storey, V.C., sul modello di dati entità-associazione, 142
- Strange, K., sui data warehouse rispetto ai data mart, 466
- strategia SI/IT, 16–21
  - contesto esterno, 20–21
  - contesto interno, 19–20

stripe, 313  
 strumenti di conversione dei dati, 483  
 strumenti di interrogazione, 498  
 strumenti di programmazione visiva, 110  
 strumenti di qualità dei dati, 480–483  
 strutture dei dati  
     alberi, 304  
     puntatori, 299  
     relazionali, 239  
 SUM, funzione (SQL), 361, 362  
 supertipo, 198  
     attributi e, 198  
 supertipo/sottotipo, associazioni, 198–205, 261–264  
     concetti di base e notazione, 199–202  
     generalizzazione, 202–205  
     nell'integrazione delle viste, 274  
     specializzazione, 202–205  
     vincoli, 206–209  
         completezza, 206–209  
         discriminanti, 209–210  
         gerarchie, 211–213  
         separazione, 209  
 supertipo/sottotipo, gerarchie, 211–213  
 Sussman, D., su VBA, 444  
 sviluppatori del sistema, 88  
 sviluppo dei sistemi informativi  
     approcci alternativi, 108–110  
     evoluzione dei, 3–22  
     sviluppo delle basi di dati entro, 98–99  
 Sybase, 338  
 Symmetric Multiprocessing (SMP), 323, 417–420, 420–421  
 System R, 238, 338

## T

tabella di indice hash, 308

tabelle

codice, 290  
 di base, 349, 351  
 di dimensioni, 484–485, 492–495  
 di fatti, 485, 487, 488  
 in SQL  
     creazione di, 346–348  
     definizioni, 246  
     derivate, 390–391  
     eliminare, 355  
     modificare le definizioni, 354  
     multiple, 380–392

nel modello di dati relazionale, 239, 240, 247–248  
 relazionali, 246  
 singole, 358–361  
     conversione, 290  
 tabelle di base, 349, 351  
 tabelle di dimensioni, 484–485  
     chiavi surrogate e, 486–487  
     dimensioni a cambiamento lento e, 495–496  
     gerarchie, 493–495  
     multivalore, 493  
 tabelle di fatti multiple, 490–491  
 tabelle di fatti prive di fatti, 491  
 tabelle di fatti, 485, 487, 488  
     chiavi surrogate e, 486–487  
     dimensione delle, 488–489  
     grana delle, 487–488  
     modellare le date, 489  
     multiple, 490–491  
 tabelle di indice, hash, 308  
 tabelle multiple, in SQL, 380–392  
 tabelle relazionali, 246  
 tablespace, 299  
 Tayi, G. K., sulla qualità dei dati, 84  
 team, sviluppo dei sistemi o delle basi di dati, 111  
 tecniche di codifica, 289  
 tecniche di compressione, 289  
 Teorey, T. J.  
     sui cluster di entità, 216  
     sul modello di dati entità-associazione, 142  
 terabyte, 64  
 termine, 147  
 ternarie, associazioni, 169–171, 174, 261  
 terza forma normale (3NF), 265, 269–271  
 Thomas, J. C., su Query-by-Example (QBE), 428  
 Thompson, C.  
     sull'architettura a tre livelli, 414  
     sull'architettura client/server, 117  
 tipi di dati, 287, 289–290  
     in Oracle *8i*, 289  
 tipi di dati definiti dagli utenti (UDT), 395  
 tipi di entità, 153–157  
     deboli, 155  
     definire, 155  
     denominare, 155–157  
     forti, 155  
 tipi di entità forti, 155  
 tipo di associazioni, 164–166

tipo di entità debole, 155  
 tolleranza degli errori, RAID e, 313–319  
 Total Quality Management (TQM), gestione dei dati e, 476  
 transazioni parallele, 417  
 trasformazione dei dati, 478–483  
   campo singolo, 480  
   funzioni a livello di campo, 480  
   funzioni a livello di record, 479–480  
   multicampo, 480  
 trasformazione dei dati a campo singolo, 480  
 trasformazione dei dati multicampo, 480  
 trigger, 379, 398–400

## U

UNION, clausola, 391  
 UNION JOIN (SQL), 382, 384  
 UNIQUE, operatore (SQL), 347  
 unire i dati, 479  
 UNIX/NT, 40  
 UPDATE (SQL), 353, 357, 393  
 UPDATE, modalità, 477  
 upgrade con patch, 40  
 Upper CASE, 111  
 utenti, 112  
 utenti finali, 88  
 utility, caricamento, 477

## V

Valacich, J. S.  
   su SDLC, 105  
   sulla gestione del team di sviluppo del progetto, 112  
   sulla modellazione dei dati, 142  
   sulla pianificazione delle matrici, 104  
   sulle architetture client/server, 117  
 valore predefinito, 289  
 velocità di accesso casuale alle righe, dimensione  
   dei blocchi di dati e, 325  
 Venkatraman, sui sistemi informativi strategici, 12  
 vincoli, 84  
   cardinalità, 171–174  
   di completezza, 206–209  
   di integrità, 243–248  
   di separazione, 209  
   nelle associazioni supertipo/sottotipo, 205–209  
   sugli oggetti di dati, 143  
 vincoli di completezza, 206–209

vincoli di dominio, 243  
 vincoli di integrità  
   nel modello di dati relazionale, 244–246  
   referenziale, 244–246  
 vincoli di integrità referenziale, 244–246  
 vincoli di separazione, 209  
 viste  
   in SQL, 349–353  
 viste dinamiche, 349, 350  
 viste materializzate, 349, 352–353  
 viste utente, 83, 107  
 Visual Basic for Applications (VBA), 429, 444–445  
 visualizzazione dei dati, 502  
 volume dei dati, 287–288  
 Von Berthalanffy, sui sistemi informativi, 34  
 Von Halle, B.  
   sul modello di dati relazionale, 238  
   sulle regole aziendali, 144

## W

WAN, 41  
 Waxman, J., su Query-by-Example (QBE), 428  
 Westerman, P., sull'aggiornamento dei data warehouse, 477  
 WHERE, clausola (SQL), 350, 363, 368, 370, 380, 381,  
   384, 390  
 White, C.  
   riepilogo delle applicazioni di basi di dati, 82  
   sull'estrazione dei dati, 474–475  
 Wieseman, sui sistemi informativi strategici, 11  
 Williams, J., sugli strumenti di supporto alla riconciliazione  
   dei dati, 481  
 WINDOW, clausola (SQL), 396  
   estensioni, 396–398  
 Winter, R.  
   sugli ERP, 46  
   sulle basi di dati grandi, 64  
   sulle funzioni analitiche proposte per SQL-99, 396  
 WizRules, 482, 483  
 World Wide Web (WWW)  
   raccogliere i dati da, 91  
   SQL e, 339

## X

X/Open SQL Call Level Interface, 440



**Y**

Yang, D., sul modello di dati entità-associazione, 142  
Yuhanna, N., sulla dimensione dei blocchi di dati, 325

**Z**

Zachman, J.A., su ISA, 99  
Zemke, F., sulle funzioni analitiche proposte per SQL-100,  
396  
Zloof, M. M., su Query-by-Example (QBE), 427