

**Matteo Galfarelli  
Stefano Rizzi**

# **Data Warehouse**

*teoria e pratica della progettazione*

**seconda edizione**

**McGraw-Hill**

# Introduzione al data warehousing

---

L'informazione è un bene a valore crescente, necessario per pianificare e controllare le attività aziendali con efficacia; essa costituisce di fatto la materia prima che viene trasformata dai sistemi informativi come i semilavorati vengono trasformati dai sistemi di produzione. Purtroppo, come ben sa chi è solito navigare il *mare magnum* di Internet, l'equazione *dati = informazione* non è sempre corretta: spesso, infatti, la disponibilità di troppi dati rende arduo, se non impossibile, estrarre le informazioni veramente importanti.

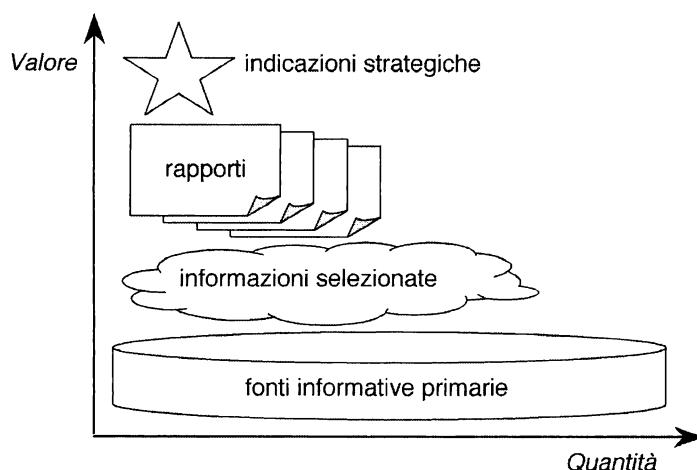
Il fenomeno del *data warehousing* nasce proprio dall'enorme accumulo di dati registrato nell'ultimo decennio, e dalla pressante richiesta di utilizzare attivamente questi dati per scopi che superino quelli, di routine, legati all'elaborazione giornaliera. Uno scenario tipico è quello di una grande azienda, con numerose filiali, i cui dirigenti desiderano quantificare e valutare il contributo dato da ciascuna di esse al rendimento commerciale globale dell'impresa. Essendo i dati elementari sulle attività svolte disponibili nel database aziendale, un approccio possibile consiste nel chiedere ai tecnici che lo amministrano di formulare una interrogazione *ad hoc* che effettui i calcoli necessari sui dati (in genere aggregazioni). Quando i tecnici saranno riusciti a formulare l'interrogazione voluta (tipicamente in SQL, dopo avere a lungo consultato i cataloghi del database), e una volta terminata la sua elaborazione (il che richiederà probabilmente alcune ore, dato l'elevato volume dei dati, la complessità dell'interrogazione e la contemporanea incidenza sui dati delle interrogazioni facenti parte del normale carico di lavoro), ai dirigenti verrà restituito un "rapporto" sotto forma di foglio elettronico su cui basare le loro decisioni future.

Già da parecchi anni si è capito che questa via è difficilmente percorribile, perché porta a un inutile consumo di tempo e risorse e al contempo non sempre produce il risultato desiderato. Tra l'altro, mescolare questo tipo di interrogazioni "analitiche" con quelle "transazionali" di routine porta a inevitabili rallentamenti che rendono insoddisfatti gli utenti di entrambe le categorie. L'idea alla base del data warehousing è allora quella di separare l'elaborazione di tipo analitico (OLAP, *On-Line Analytical Processing*) da quella legata alle transazioni (OLTP, *On-Line Transactional Processing*), costruendo un nuovo raccoglitore di informazioni che integri i dati elementari provenienti da sorgenti di varia natura, li organizzi in una forma appropriata e li renda quindi disponibili per scopi di analisi e valutazione finalizzate alla pianificazione e al processo decisionale (Lechtenbörger, 2001).

Passiamo velocemente in rassegna alcune delle aree in cui tecnologie di data warehousing vengono impiegate con successo: commercio (analisi delle vendite e dei reclami, controllo di spedizioni e inventari, cura del rapporto con i clienti), manifattura (controllo dei costi di produzione, supporto fornitori e ordini), servizi finanziari (analisi del rischio e delle carte di credito, rivelazione di frodi), trasporti (gestione parco mezzi), telecomunicazioni (analisi del flusso delle chiamate e del profilo dei clienti), sanità (analisi di ricoveri e dimissioni, contabilità per centri di costo). D'altronde, il campo di utilità dei sistemi di data warehousing non è ristretto al dominio aziendale e d'impresa: spazia ulteriormente dall'area medico-epidemiologica a quella demografica, dall'area delle scienze naturali a quella didattica. Caratteristica comune a tutti questi campi è la necessità di strumenti di archiviazione e interrogazione per ottenere facilmente e in tempi ridotti, dall'enorme quantità di dati immagazzinati nei database o resi disponibili da Internet, informazioni di sintesi che permettano la valutazione di un fenomeno, la scoperta di correlazioni significative e, in definitiva, l'acquisizione di conoscenza utile come supporto alle decisioni.

## 1.1 I sistemi di supporto alle decisioni

La funzione svolta dalle basi di dati in ambito aziendale è stata fino a qualche anno fa solo quella di memorizzare *dati operazionali*, ossia dati generati da operazioni, principalmente di carattere amministrativo, svolte all'interno dei processi gestionali (per esempio gestione acquisti, gestione vendite, fatturazione). D'altronde, per ogni azienda è fondamentale poter disporre in maniera rapida e completa delle informazioni necessarie al processo decisionale: le indicazioni strategiche sono estrapolate principalmente dalla mole dei dati operazionali contenuti nei database aziendali, attraverso un procedimento di selezione e sintesi progressiva schematizzato in Figura 1.1.



**Figura 1.1** Il valore dell'informazione in funzione della quantità.

Ben presto, l'aumento esponenziale del volume dei dati operazionali ha reso il calcolatore l'unico supporto adatto al processo decisionale svolto dai dirigenti aziendali. Il ruolo delle basi di dati ha così cominciato a cambiare, a partire dai tardi anni '80, con la nascita dei *sistemi di supporto alle decisioni* (*decision support system*), termine con cui si intende l'insieme delle tecniche e degli strumenti informatici atti a estrapolare *informazioni* da un insieme di *dati* memorizzati su supporti elettronici. I diversi ruoli dei sistemi di supporto alle decisioni in ambito aziendale sono riassunti in Tabella 1.1.

**Tabella 1.1** Ruolo dei sistemi di supporto alle decisioni.

Nel passato	Nel futuro
descrivere il passato	anticipare il futuro
descrivere i problemi	suggerire i cambiamenti da apportare
ridurre i costi	aumentare i profitti

Tra le problematiche da affrontare per la realizzazione di un sistema di supporto alle decisioni, ricordiamo la necessità di gestire grandi moli di dati; di accedere a diverse fonti di dati su piattaforme eterogenee; di garantire l'accesso a più utenti per interrogazioni, analisi in tempo reale e simulazioni; di gestire versioni storiche dei dati.

## 1.2 Il data warehousing

Tra i sistemi di supporto alle decisioni, i sistemi di data warehousing sono probabilmente quelli su cui negli ultimi anni si è maggiormente focalizzata l'attenzione sia nel mondo accademico sia in quello industriale. È possibile definire in modo informale il data warehousing come segue:

**Data warehousing.** Una collezione di metodi, tecnologie e strumenti di ausilio al cosiddetto “lavoratore della conoscenza” (*knowledge worker*: dirigente, amministratore, gestore, analista) per condurre analisi dei dati finalizzate all’attuazione di processi decisionali e al miglioramento del patrimonio informativo.

Questa definizione, volutamente molto generale, rende un’idea degli scopi del processo ma non ne esprime le specifiche caratteristiche. Per capire a fondo il ruolo e l’utilità del data warehousing occorre allora analizzare le esigenze che ne hanno decretato la nascita. Alcune lamentele ricorrenti da parte degli utenti finali dei sistemi informativi tradizionali sono efficacemente riassunte da Kimball (1996):

“Abbiamo montagne di dati ma non possiamo accedervi!”. Questa frase esprime la frustrazione da parte di chi ha il ruolo e la competenza per decidere del futuro aziendale ma non possiede gli strumenti tecnici per ottenere, nella forma desiderata, i dati necessari.

“Come è possibile che persone che svolgono lo stesso ruolo presentino risultati sostanzialmente diversi?”. In un contesto aziendale medio-grande sono tipicamente presenti più basi di dati, ciascuna relativa a una diversa area del *business*, spesso memorizzate

su piattaforme logico-fisiche differenti e non integrate dal punto di vista concettuale. I risultati prodotti all'interno delle diverse aree saranno allora molto probabilmente inconsistenti tra loro.

“*Vogliamo selezionare, raggruppare e manipolare i dati in ogni modo possibile!*”. Il processo decisionale è difficilmente pianificabile a priori. L'utente finale vorrebbe disporre di uno strumento sufficientemente amichevole e flessibile da consentirgli di condurre l'analisi in modo estemporaneo, lasciandosi guidare dalle informazioni via via ottenute per decidere sul momento quali nuove correlazioni ricercare.

“*Mostratemi solo ciò che è importante!*”. Esaminare i dati al massimo livello di dettaglio è non solo inutile per il processo decisionale, ma addirittura controproducente, perché non consente di focalizzare l'attenzione sulle informazioni veramente significative.

“*Tutti sanno che alcuni dati non sono corretti!*”. Questo è un altro punto dolente. Una percentuale non trascurabile dei dati transazionali è non corretta, o addirittura assente. Evidentemente, basare il procedimento analitico su dati errati e incompleti non permette di raggiungere risultati validi.

Da questo elenco di difficoltà e problemi possiamo facilmente estrarre un elenco di parole chiave che diventano fattori distintivi e requisiti indispensabili del *processo di data warehousing*, ossia del complesso di attività che consentono di trasformare i dati operazionali in conoscenza a supporto delle decisioni:

- *accessibilità* a utenti con conoscenze limitate di informatica e strutture dati;
- *integrazione* dei dati sulla base di un modello standard dell'impresa;
- *flessibilità di interrogazione* per trarre il massimo vantaggio dal patrimonio informativo esistente;
- *sintesi* per permettere analisi mirate ed efficaci;
- *rappresentazione multidimensionale* per offrire all'utente una visione intuitiva ed efficacemente manipolabile delle informazioni;
- *correttezza e completezza* dei dati integrati.

Al centro del processo, il *data warehouse* (letteralmente, magazzino di dati) è un contenitore (*repository*) di dati che diventa garante dei requisiti esposti. La definizione che ne diamo è tratta dal testo di Inmon (1996):

**Data warehouse.** Un *Data Warehouse* (DW) è una collezione di dati di supporto per il processo decisionale che presenta le seguenti caratteristiche:

- è orientata ai soggetti di interesse;
- è integrata e consistente;
- è rappresentativa dell'evoluzione temporale e non volatile.

Si intende che il DW è orientato ai soggetti perché si incentra sui concetti di interesse dell'azienda, quali i clienti, i prodotti, le vendite, gli ordini. Viceversa, i database operazionali sono organizzati intorno alle differenti applicazioni del dominio aziendale.

L'accento sugli aspetti di integrazione e consistenza è importante poiché il DW si appoggia a più fonti di dati eterogenee: dati estratti dall'ambiente di produzione, e quindi originariamente archiviati in basi di dati aziendali, o addirittura provenienti da sistemi informativi esterni all'azienda. Di tutti questi dati il DW si impegna a restituire una visione unificata. In linea di massima, si può dire che

la costruzione di un sistema di data warehousing non comporta l'inserimento di nuove informazioni bensì la riorganizzazione di quelle esistenti, e implica pertanto l'esistenza di un sistema informativo.

Mentre i dati operazionali coprono un arco temporale di solito piuttosto limitato, poiché la maggior parte delle transazioni coinvolge i dati più recenti, il DW deve permettere analisi che spazino sulla prospettiva di alcuni anni. Per questo motivo, il DW è aggiornato a intervalli regolari a partire dai dati operazionali ed è in crescita continua. Volendo fare un paragone possiamo supporre che, a intervalli regolari, venga scattata una fotografia istantanea dei dati operazionali. La progressione delle fotografie scattate viene immagazzinata nel DW, dove genera un film che documenta la situazione aziendale da un istante zero fino al tempo attuale.

Proprio il fatto che, in linea di principio, non vengano mai eliminati dati dal DW e che gli aggiornamenti siano tipicamente eseguiti "a freddo", ossia quando il DW è fuori linea, fa sì che un DW possa essere fondamentalmente considerato come un database a sola lettura. Questa caratteristica, insieme all'esigenza degli utenti di contenere i tempi di risposta alle interrogazioni di analisi, ha importanti conseguenze a vari livelli. Innanzitutto, incide sulle tecnologie adottate dai DBMS specializzati per il data warehousing: non sono infatti più necessarie le tecniche sofisticate di gestione delle transazioni richieste dalle applicazioni operazionali. Inoltre, il fatto di operare in sola lettura differenzia profondamente le soluzioni di progettazione logica per i DW da quelle utilizzate per i database operazionali: l'aspetto forse più evidente nelle implementazioni relazionali è che la pratica della normalizzazione delle tabelle viene abbandonata a favore di una parziale denormalizzazione mirata al miglioramento delle prestazioni.

Ulteriori e fondamentali differenze tra database operazionali e DW sono legate alle tipologie di interrogazioni. Per i primi, le interrogazioni eseguono transazioni che in genere leggono e scrivono un ridotto numero di record da diverse tabelle legate da semplici relazioni: per esempio, si ricercano i dati di un cliente per inserire un suo nuovo ordine. Questo tipo di elaborazione viene comunemente detto *On-Line Transactional Processing* (OLTP). Al contrario, il tipo di elaborazione per cui nascono i DW viene detto *On-Line Analytical Processing* (OLAP), ed è caratterizzato da un'analisi dinamica e multidimensionale che richiede la scansione di un'enorme quantità di record per calcolare un insieme di dati numerici di sintesi che quantifichino le prestazioni dell'azienda. È importante osservare che, mentre nei sistemi OLTP il nucleo sostanziale del carico di lavoro è "congelato" all'interno dei programmi applicativi e solo occasionalmente vengono lanciate interrogazioni estemporanee o di manutenzione straordinaria sui dati, in un DW l'interattività è una caratteristica irrinunciabile delle sessioni di analisi e fa sì che il carico di lavoro effettivo vari continuamente nel tempo.

Le peculiari caratteristiche delle interrogazioni OLAP fanno sì che i dati nel DW siano normalmente rappresentati in forma *multidimensionale*. L'idea di base è quella di vedere i

dati come punti in uno spazio le cui dimensioni corrispondono ad altrettante possibili dimensioni di analisi; ciascun punto, rappresentativo di un evento accaduto nell'azienda, viene descritto tramite un insieme di misure di interesse per il processo decisionale. Una descrizione approfondita del modello multidimensionale, la cui conoscenza risulta assolutamente indispensabile per la comprensione di tutti gli aspetti legati alla modellazione concettuale e logica dei DW nonché alla loro interrogazione, è oggetto del Paragrafo 1.5.

Le principali differenze tra database operazionali e DW sono riassunte in Tabella 1.2. Per una documentazione esaurente e dettagliata sul processo di data warehousing, si vedano per esempio Chaudhuri e Dayal (1997), Jarke e altri (2000), Kelly (1997), Kimball (1996), Mattison (1996) e Silverston (1997).

**Tabella 1.2** Differenze tra database operazionali e DW (Kelly, 1997).

	<i>Database operazionali</i>	<i>Data warehouse</i>
utenti	migliaia	centinaia
carico di lavoro	transazioni predefinite	interrogazioni di analisi <i>ad hoc</i>
accesso	a centinaia di record, in lettura e scrittura	a milioni di record, per lo più in lettura
scopo	dipende dall'applicazione	supporto alle decisioni
dati	elementari, sia numerici sia alfanumerici	di sintesi, prevalentemente numerici
integrazione dei dati	per applicazione	per soggetto
qualità	in termini di integrità	in termini di consistenza
copertura temporale	solo dati correnti	dati correnti e storici
aggiornamenti	continui	periodici
modello	normalizzato	denormalizzato, multidimensionale
ottimizzazione	per accessi OLTP su una frazione del database	per accessi OLAP su gran parte del database
sviluppo	a cascata	iterativo

### 1.3 Architetture per il data warehousing

In questo paragrafo vengono analizzate e discusse le architetture tipiche dei *sistemi di data warehousing*, ossia dei sistemi che implementano il processo di data warehousing. Molto spesso, il termine *data warehouse* viene colloquialmente utilizzato in modo esteso per denotare tutto il sistema di data warehousing anziché il solo contenitore dei dati di sintesi. Laddove riterremo che non si generi confusione, adotteremo anche noi questa comoda abbreviazione.

Le caratteristiche architetturali irrinunciabili per un sistema di data warehousing possono essere così enunciate (Kelly, 1997):

- *Separazione*: l'elaborazione analitica e quella transazionale devono essere mantenute il più possibile separate.

- **Scalabilità:** l'architettura hardware e software deve poter essere facilmente ridimensionata a fronte della crescita nel tempo dei volumi di dati da gestire ed elaborare e del numero di utenti da soddisfare.
- **Estendibilità:** deve essere possibile accogliere nuove applicazioni e tecnologie senza riprogettare integralmente il sistema.
- **Sicurezza:** il controllo sugli accessi è essenziale a causa della natura strategica dei dati memorizzati.
- **Amministrabilità:** la complessità dell'attività di amministrazione non deve risultare eccessiva.

### 1.3.1 Architettura a un livello

Obiettivo di questa architettura, a dire il vero poco utilizzata nella pratica, è la minimizzazione dei dati memorizzati, ottenuta eliminando le ridondanze. Come mostrato in Figura 1.2, il DW è in questo caso *virtuale*, nel senso che viene implementato come una vista multidimensionale dei dati operazionali generata da un apposito *middleware*, ossia da uno strato d'elaborazione intermedio (Devlin, 1997).

Il primo punto debole di questa architettura è che non rispetta il requisito di separazione tra l'elaborazione analitica OLAP e quella transazionale OLTP. Le interrogazioni di analisi vengono infatti ridirette sui dati operazionali dopo essere state reinterpretate dal middleware, interferendo così con il normale carico di lavoro transazionale. Inoltre, mentre con questa architettura è possibile (anche se complesso) rispondere ai requisiti di integrazione e correttezza dei dati, diventa impossibile esprimere un livello di storicitizzazione superiore a quello delle sorgenti. Per questi motivi, l'approccio virtuale al data warehousing ha avuto successo solamente in contesti in cui le esigenze di analisi sono particolarmente limitate e il volume dei dati da esaminare è molto ampio.

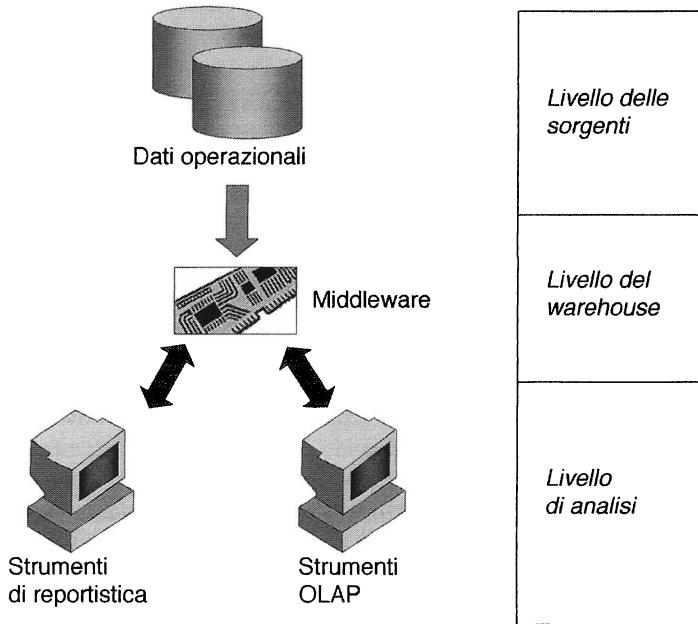
### 1.3.2 Architettura a due livelli

Il requisito di separazione gioca un ruolo fondamentale nel determinare la classica architettura di un sistema di data warehousing, rappresentata in Figura 1.3. Sebbene tradizionalmente designata come *architettura a due livelli* per evidenziare la separazione tra il livello delle sorgenti e quello del DW, essa si articola in realtà complessivamente su quattro livelli distinti, che descrivono stadi successivi del flusso di dati (Lechtenbörger, 2001):

1. *Livello delle sorgenti.* Il DW utilizza fonti di dati eterogenei: estratti dall'ambiente di produzione, e quindi originariamente archiviati in database aziendali relazionali o *legacy*<sup>1</sup>, oppure provenienti da sistemi informativi esterni all'azienda.

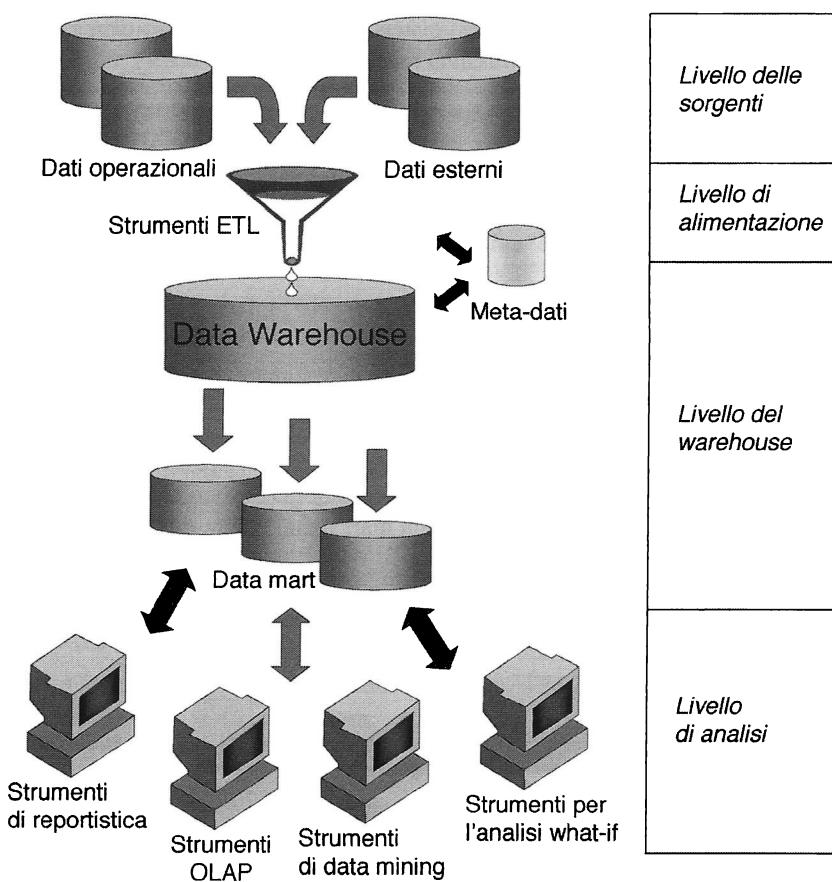
---

<sup>1</sup> Con il generico termine *sistemi legacy* (letteralmente, lasciti o retaggi) ci si riferisce alle applicazioni esistenti in azienda, tipicamente su mainframe o minicomputer, che pur essendo correntemente utilizzate per la gestione operazionale non rispondono ai requisiti architettonici moderni e agli standard attuali, risultando di conseguenza difficilmente accessibili e integrabili con sistemi di più recente realizzazione. Esempio tipico di sistema legacy è un'applicazione che poggia su una base di dati non relazionale.



**Figura 1.2** Architettura a un livello per un sistema di data warehousing.

2. *Livello dell'alimentazione.* I dati memorizzati nelle sorgenti devono essere estratti, ripuliti per eliminare le inconsistenze e completare eventuali parti mancanti, integrati per fondere sorgenti eterogenee secondo uno schema comune. I cosiddetti *strumenti ETL* (*Extraction, Transformation and Loading*) permettono di integrare schemi eterogenei, nonché di estrarre, trasformare, ripulire, validare, filtrare e caricare i dati dalle sorgenti nel DW (Jarke, 2000). Dal punto di vista tecnologico vengono trattate problematiche tipiche dei servizi informativi distribuiti, come la gestione di dati inconsistenti e di strutture dati incompatibili (Zhuge, 1996). Alcuni aspetti rilevanti per questo livello verranno discussi nel Paragrafo 1.4.
3. *Livello del warehouse.* Le informazioni vengono raccolte in un singolo “contenitore” centralizzato logicamente: il DW. Esso può essere direttamente consultato ma anche usato come sorgente per costruire *data mart*, che ne costituiscono una parziale replica, orientati verso specifiche aree dell’impresa. Accanto al DW, il contenitore dei *metadati* mantiene informazioni sulle sorgenti, sui meccanismi di accesso, sulle procedure di pulitura e alimentazione, sugli utenti, sugli schemi dei data mart ecc. (per dettagli si veda il Paragrafo 1.6).
4. *Livello di analisi.* Permette la consultazione efficiente e flessibile dei dati integrati a fini di stesura di report, di analisi, di simulazione. Dal punto di vista tecnologico sono richieste capacità di navigazione degli aggregati (Gupta, 1995), ottimizzazione di interrogazioni complesse (Chaudhuri, 1994), tecniche di indicizzazione avanzate (Lomet, 1990) e interfacce visuali amichevoli (Colliat, 1996; Fayyad, 1996). Le diverse tipologie di analisi di interesse per il supporto alle decisioni sono discusse nel Paragrafo 1.7.



**Figura 1.3** Architettura a due livelli per un sistema di data warehousing.

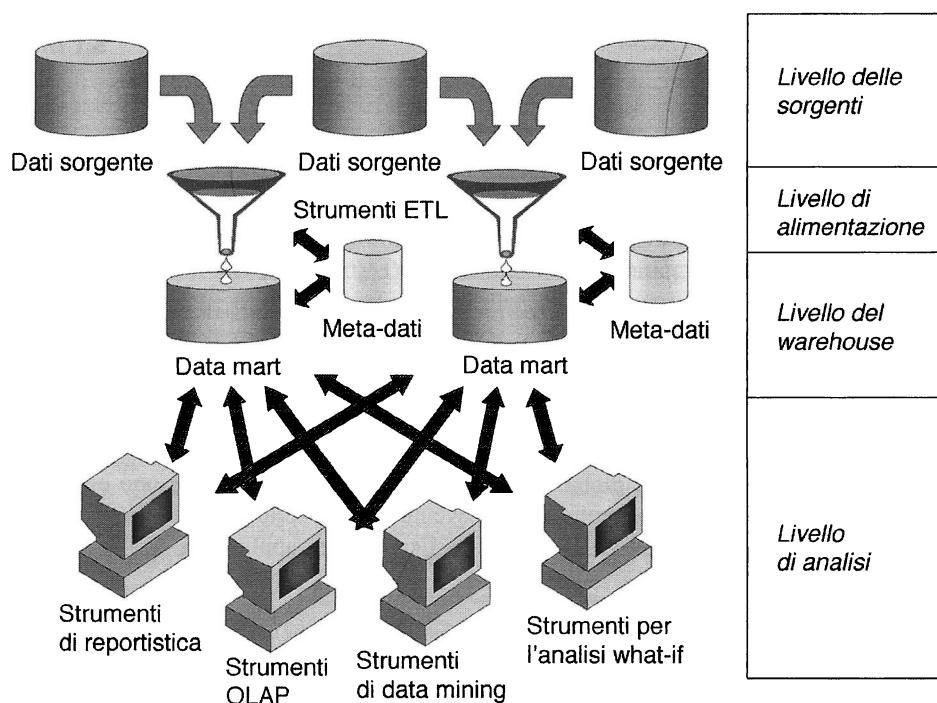
La distinzione introdotta nell'architettura tra *data warehouse* e *data mart* merita un approfondimento. Il blocco designato brevemente come *data warehouse* in Figura 1.3 viene spesso chiamato anche *data warehouse primario* o *data warehouse aziendale*, e svolge il ruolo di contenitore centrale e globale dei dati di sintesi. I *data mart* possono essere visti come piccoli DW “locali” che replicano (ed eventualmente sintetizzano ulteriormente) la porzione di DW primario di interesse per una particolare area applicativa.

**Data mart.** Con il termine *data mart* si intende un sottoinsieme o un’aggregazione dei dati presenti nel DW primario, contenente l’insieme delle informazioni rilevanti per una particolare area del business, una particolare divisione dell’azienda, una particolare categoria di soggetti.

I data mart alimentati dal DW primario sono spesso detti *dipendenti*. Sebbene in linea di principio non strettamente necessari, per i sistemi collocati all'interno di realtà aziendali medio-grandi essi costituiscono un'utilissima risorsa:

- come blocchi costruttivi durante la realizzazione incrementale del DW;
- in quanto delineano i contorni delle informazioni necessarie a un particolare tipo di utenti per le loro interrogazioni;
- poiché, essendo di dimensioni inferiori al DW primario, permettono di raggiungere prestazioni migliori.

In alcuni contesti, principalmente per motivi organizzativi e politici, si preferisce adottare l'architettura illustrata in Figura 1.4, in cui i data mart vengono alimentati direttamente dalle sorgenti e vengono pertanto detti *indipendenti*. L'assenza di un DW primario snellisce le fasi progettuali, ma determina uno schema complesso di accessi ai dati e ingenera il rischio di inconsistenze tra i data mart. A volte, pur rispettando l'indipendenza dei data mart, si preferisce allora creare comunque un DW centrale: rispetto all'architettura standard a due livelli di Figura 1.3, i ruoli dei data mart e del DW sono di fatto invertiti, infatti il DW viene in questo caso alimentato dai data mart e può essere direttamente interrogato al fine di semplificare il pattern degli accessi.



**Figura 1.4** Architettura a due livelli con data mart indipendenti.

Le principali motivazioni a sostegno dell'architettura a due livelli, in cui il livello del warehouse funge da separatore tra le sorgenti e le applicazioni di analisi, sono così riassumibili (Jarke, 2000; Lechtenbörger, 2001):

- A livello del warehouse è continuamente disponibile informazione di buona qualità anche quando, per motivi tecnici oppure organizzativi, è temporaneamente precluso l'accesso alle sorgenti.
- L'interrogazione analitica effettuata sul DW non interferisce con la gestione delle transazioni a livello operazionale, la cui affidabilità è essenziale per il funzionamento dell'azienda.
- L'organizzazione logica del DW è basata sul modello multidimensionale, mentre le sorgenti offrono in genere modelli relazionali o semi-strutturati.
- C'è una discordanza temporale e di granularità tra sistemi OLTP, che trattano dati correnti e al massimo livello di dettaglio, e sistemi OLAP che operano su dati storici e di sintesi.
- A livello del warehouse è possibile impiegare tecniche specifiche per ottimizzare le prestazioni per applicazioni di analisi e reportistica.

È utile osservare che alcuni autori, con riferimento a questa architettura, utilizzano la nostra stessa terminologia per indicare concetti differenti. In particolare, essi considerano il DW come un contenitore di dati integrati e consistenti ma ancora in forma operazionale, introducendo la rappresentazione multidimensionale dei dati solo a livello dei data mart. Nella nostra terminologia, questa visione “operazionale” del DW corrisponde sostanzialmente al livello dei dati riconciliati nelle architetture a tre livelli, trattate nel prossimo paragrafo.

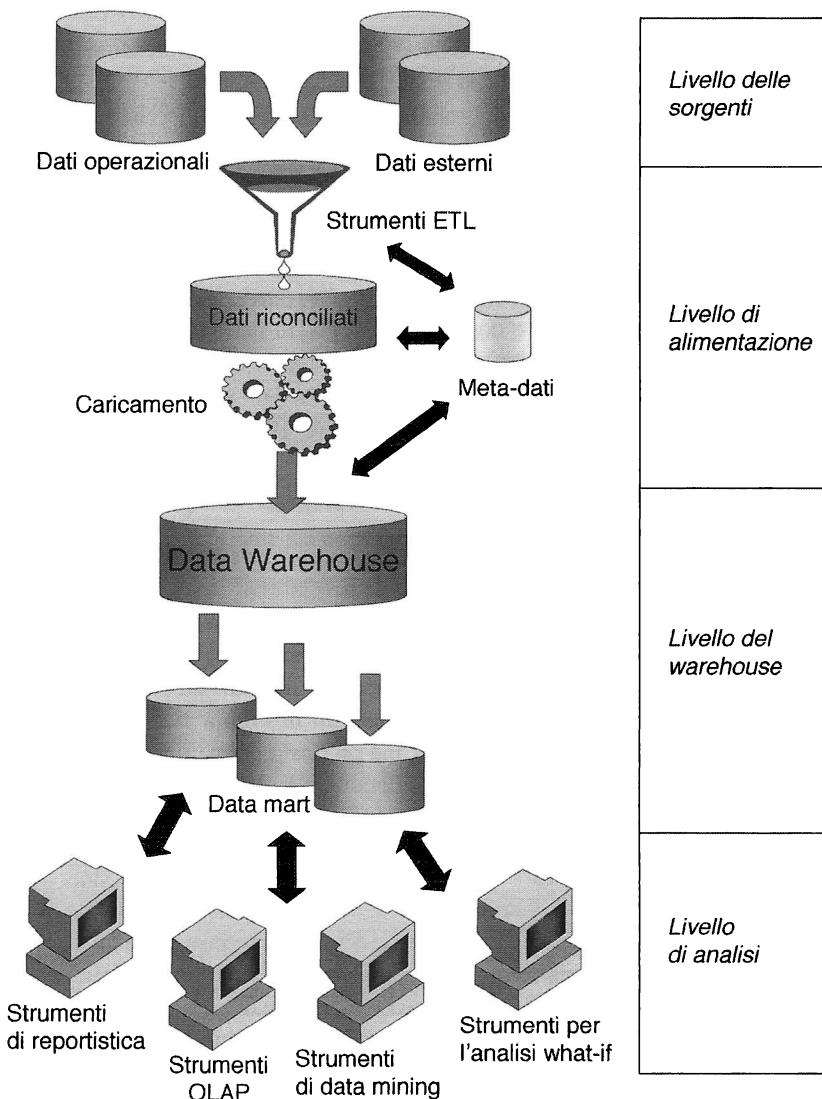
### 1.3.3 Architettura a tre livelli

Il terzo livello introdotto in questa architettura è il cosiddetto *livello dei dati riconciliati*, detto anche *operational data store*, che materializza i dati operazionali ottenuti a valle del processo di integrazione e ripulitura dei dati sorgente: quindi dati integrati, consistenti, corretti, volatili, correnti e dettagliati. Come illustrato in Figura 1.5, il DW viene allora alimentato non più direttamente dalle sorgenti, ma piuttosto dai dati riconciliati.

Il vantaggio principale del livello dei dati riconciliati è che esso crea un modello di dati comune e di riferimento per l'intera azienda, introducendo al contempo una separazione netta tra le problematiche legate all'estrazione e integrazione dei dati dalle sorgenti e quelle inerenti l'alimentazione del DW. D'altro canto, i dati riconciliati introducono un'ulteriore ridondanza rispetto ai dati operazionali sorgente. Va comunque detto che, in realtà, si può assumere che anche nelle architetture a due livelli sia presente un livello riconciliato, che non sarà in quel caso materializzato ma solamente virtuale, essendo definito come una vista integrata e consistente dei dati nelle sorgenti operazionali.

A conclusione di questo paragrafo riportiamo per completezza l'esistenza di un approccio architettonico che può essere descritto come un ibrido tra la soluzione a un livello e quella a due/tre livelli. In questo approccio, sebbene esista fisicamente un DW, si suppone che esso non sia in grado di risolvere la totalità delle interrogazioni formulate: si assume cioè che l'utente sia interessato ad accedere ai dati sorgente a partire dai dati

aggregati (*drill-through*). A tal fine, parte delle interrogazioni devono evidentemente essere riscritte sulle sorgenti (oppure sui dati riconciliati, se presenti). Questo tipo di architettura, implementata nel prototipo descritto da Cui e Widom (2000), richiede la capacità di risalire dinamicamente ai dati sorgente necessari a risolvere le interrogazioni (*lineage*); le implicazioni dal punto di vista dell'ottimizzazione delle prestazioni, in particolare della materializzazione di viste, sono trattate da Hull e Zhou (1996), Yang e altri (1997) e Gupta (1997a).



**Figura 1.5** Architettura a tre livelli per un sistema di data warehousing.

## 1.4 Gli strumenti ETL

In questo paragrafo e nei seguenti esamineremo in maggior dettaglio alcuni aspetti di fondamentale importanza ai diversi livelli dell'architettura, cominciando dall'alimentazione.

Il ruolo degli strumenti di *Extraction, Transformation and Loading* è quello di alimentare una sorgente dati singola, dettagliata, esauriente e di alta qualità che possa a sua volta alimentare il DW. In caso di architettura a tre livelli, di fatto questi strumenti alimentano il livello dei dati riconciliati; le operazioni da essi svolte vengono quindi spesso globalmente designate con il termine *riconciliazione* e sono, tra le diverse fasi del processo di warehousing, tra quelle più complesse e tecnicamente impegnative (McFadden, 1998).

Durante il processo di alimentazione del DW, la riconciliazione avviene in due occasioni: quando il DW viene popolato per la prima volta, e periodicamente quando il DW viene aggiornato. Come mostrato in Figura 1.6, essa consiste di quattro distinti processi detti rispettivamente *estrazione* (*extraction* o *capture*), *pulitura* (*cleaning* o *cleansing* o *scrubbing*), *trasformazione* (*transformation*) e *caricamento* (*loading*). Nel seguito esamineremo brevemente ciascuno di questi processi, rimandando a Jarke e altri (2000), McFadden e altri (1998) e English (1999) per dettagli.

Occorre precisare che, nella letteratura, i confini tra pulitura e trasformazione sono spesso sfumati dal punto di vista terminologico, per cui è spesso poco chiara l'attribuzione di una specifica operazione all'uno o all'altro processo. Si tratta evidentemente di un problema formale ma non sostanziale; per chiarezza di esposizione adotteremo qui il presupposto espresso da McFadden e altri (1998), secondo il quale la pulitura è essenzialmente mirata alla correzione dei *valori* dei dati, mentre la trasformazione si occupa più propriamente del loro *formato*.

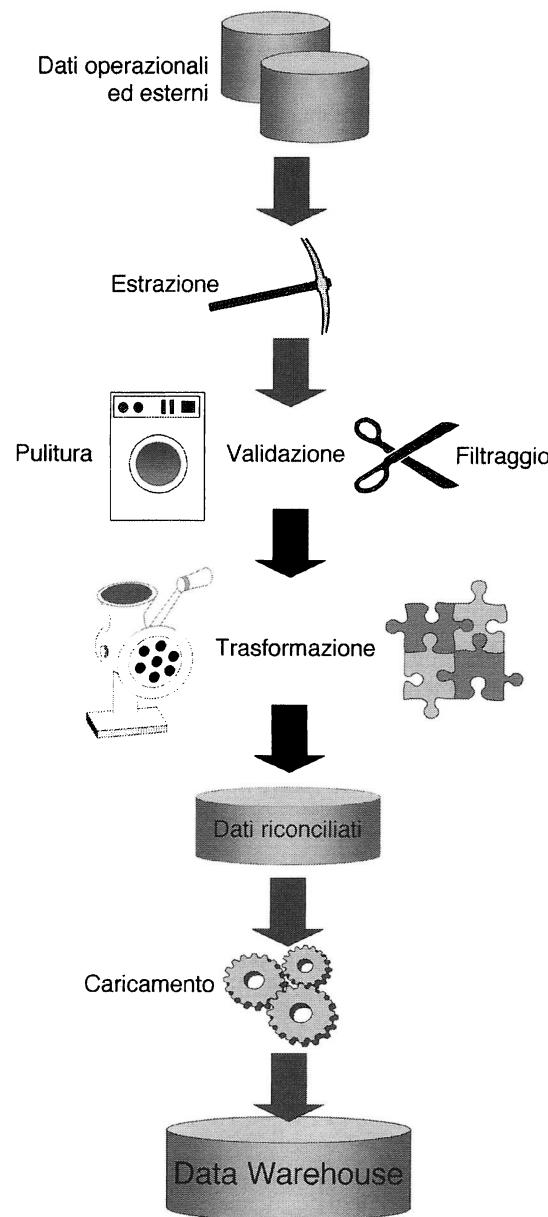
Il problema globale del progetto dell'alimentazione verrà affrontato nel Capitolo 10. Nel Capitolo 3, invece, discuteremo una fase progettuale preliminare alla progettazione del DW vero e proprio: l'*integrazione*, necessaria in presenza di sorgenti eterogenee per definire lo schema del livello dei dati riconciliati e per determinare le trasformazioni specifiche a cui i dati operazionali dovranno essere sottoposti durante l'alimentazione.

### 1.4.1 Estrazione

Durante questa fase i dati rilevanti vengono estratti dalle sorgenti. L'*estrazione statica* viene effettuata quando il DW deve essere popolato per la prima volta e consiste concettualmente in una fotografia dei dati operazionali. L'*estrazione incrementale* viene usata per l'aggiornamento periodico del DW, e cattura solamente i cambiamenti avvenuti nelle sorgenti dall'ultima estrazione. L'estrazione incrementale è in genere basata sul “giornale” (*log*) mantenuto dal DBMS operazionale; se ai dati operazionali è associata una marcatura temporale (*time-stamp*) che ne identifica l'istante di modifica o inserimento, essa è utilizzabile per semplificare l'estrazione.

L'estrazione può anche essere guidata dalle sorgenti, qualora sia possibile riscrivere le applicazioni operazionali per far sì che esse notifichino in modo asincrono le modifiche via via che avvengono, oppure qualora nel database operazionale possano essere implementati dei trigger abbinati alle transazioni di modifica dei dati rilevanti.

La scelta dei dati da estrarre avviene principalmente in base alla loro qualità (English, 1999), che dipende tra l'altro dalla completezza e accuratezza dei vincoli implementati nelle sorgenti, dall'adeguatezza del formato dei dati, dalla chiarezza degli schemi.



**Figura 1.6** Estrazione, trasformazione e caricamento (il livello dei dati riconciliati può essere assente).

### 1.4.2 Pulitura

La pulitura è una fase critica nel processo di data warehousing, poiché si incarica di migliorare la qualità dei dati – normalmente piuttosto scarsa nelle sorgenti (Galhardas, 2001). Tra gli errori e le inconsistenze tipiche che rendono *sporchi* i dati segnaliamo:

- dati duplicati (per esempio, uno stesso paziente che compare più volte in un'anagrafica ospedaliera);
- inconsistenza tra valori logicamente associati (per esempio tra l'indirizzo e il CAP);
- dati mancanti (per esempio la professione di un cliente);
- uso non previsto di un campo (per esempio, un campo codice fiscale potrebbe essere impropriamente utilizzato per memorizzare il numero di telefono d'ufficio);
- valori impossibili o errati (per esempio '30/2/1999');
- valori inconsistenti per la stessa entità dovuti a differenti convenzioni (per esempio una nazione può essere indicata dalla sua sigla internazionale o dal suo nome per esteso) e abbreviazioni (per esempio, 'V. Risorgimento' e 'Via Risorgimento');
- valori inconsistenti per la stessa entità dovuti a errori di battitura (per esempio, 'Via Risorgimento' e 'Via Risogimento').

Si fa notare in particolare che gli ultimi due tipi di errore sono particolarmente frequenti quando si ha a che fare con più sorgenti e in caso di immissione manuale dei dati.

Le principali funzionalità di pulitura dei dati riscontrabili negli strumenti ETL sono la *correzione* e *omogeneizzazione*, che utilizza dizionari appositi per correggere errori di scrittura e riconoscere sinonimi, e la *pulitura basata su regole* che applica regole proprie del dominio applicativo per stabilire le corrette corrispondenze tra valori.

### 1.4.3 Trasformazione

La trasformazione è la fase centrale del processo di riconciliazione, e converte i dati dal formato operazionale sorgente a quello del DW; in caso di architettura a tre livelli, l'output di questa fase è il livello dei dati riconciliati. In entrambi i casi, la corrispondenza con il livello sorgente è in genere complicata dalla presenza di più fonti distinte eterogenee, che richiede durante la progettazione una complessa fase di integrazione (Capitolo 3).

Alcune situazioni che devono essere corrette durante questa fase sono:

- presenza di testi liberi che nascondono informazioni importanti: per esempio, 'Grandi Affari S.p.A.' non codifica esplicitamente che si tratta di una società per azioni;
- utilizzo di formati differenti per lo stesso dato: per esempio una data può essere memorizzata come stringa o tramite tre interi.

Tra le funzionalità di trasformazione per l'alimentazione del livello dei dati riconciliati segnaliamo la *conversione* e la *normalizzazione*, che operano sia a livello di formato di memorizzazione sia a livello di unità di misura al fine di uniformare i dati, il *matching* che stabilisce corrispondenze tra campi equivalenti in sorgenti diverse, la *selezione* che eventualmente riduce il numero di campi e di record rispetto alle sorgenti. Per l'alimentazione

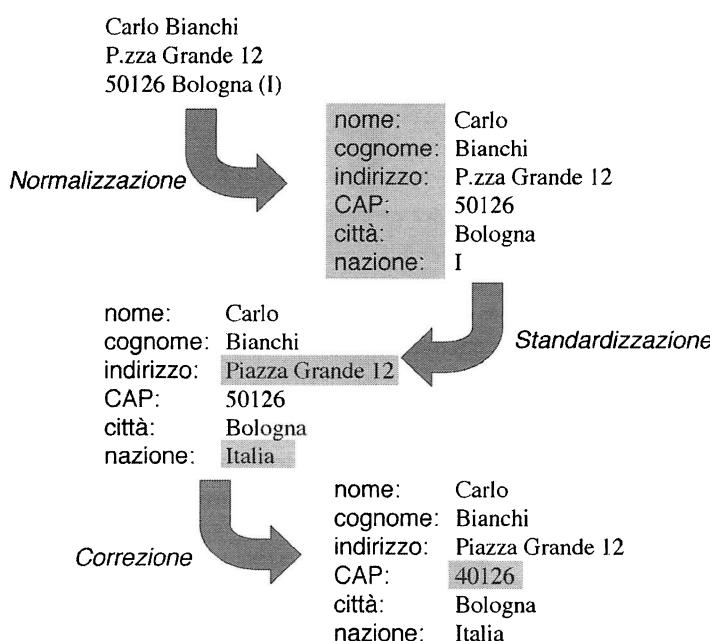
del DW si hanno sostanzialmente due differenze: la normalizzazione è sostituita dalla *denormalizzazione*, poiché i dati nel DW sono tipicamente denormalizzati, e si introduce l'*aggregazione*, che realizza le opportune sintesi dei dati.

Negli strumenti ETL le attività di pulitura e trasformazione sono spesso inscindibilmente allacciate e sovrapposte. Un esempio di pulitura e trasformazione di un dato anagrafico è mostrato in Figura 1.7: a partire da un testo libero viene dapprima estratta una struttura a campi, poi alcuni valori vengono riportati a una forma standard eliminando le abbreviazioni, infine valori logicamente associati vengono corretti.

#### 1.4.4 Caricamento

L'ultimo passo da compiere è il caricamento dei dati nel DW, che può avvenire secondo due modalità:

1. *Refresh*. I dati del DW vengono riscritti integralmente, sostituendo quelli precedenti. Questa tecnica viene normalmente utilizzata, in abbinamento all'estrazione statica, solo per popolare inizialmente il DW.
2. *Update*. I soli cambiamenti occorsi nei dati sorgente vengono aggiunti nel DW, tipicamente senza distruggere o alterare i dati esistenti. Questa tecnica viene utilizzata, in abbinamento all'estrazione incrementale, per l'aggiornamento periodico del DW.



**Figura 1.7** Esempio di pulitura e trasformazione di un dato anagrafico.

## 1.5 Il modello multidimensionale

Entriamo con questo paragrafo nel livello del warehouse, di interesse centrale per questo testo. In particolare introduciamo una delle parole chiave del data warehousing, *multidimensionalità*, e lasciamo che il lettore cominci a familiarizzare con i concetti e la terminologia che verranno ripresi e intensamente utilizzati nei capitoli sulla modellazione e la progettazione concettuale e logica.

Negli ultimi anni, le basi di dati multidimensionali hanno suscitato un vasto interesse di ricerca e di mercato essendo alla base di varie applicazioni per il supporto alle decisioni, tra cui in particolare quelle di data warehousing. Il motivo per cui il modello multidimensionale viene adottato come paradigma di rappresentazione dei dati nei DW è fondamentalmente legato alla sua semplicità e intuitività anche per utenti non esperti di informatica, in parte dovuta a sua volta alla vasta diffusione di applicazioni di tipo *spreadsheet* (foglio elettronico) come strumento di produttività individuale.

Per un approccio efficace al modello multidimensionale, forse il miglior punto di partenza è una descrizione del tipo di interrogazioni alla cui soddisfazione esso più si presta. Come si vedrà in maggiore dettaglio nel Paragrafo 1.7, alcune classiche interrogazioni orientate al processo decisionale sono le seguenti (Jarke, 2000):

*“Che incassi sono stati registrati l’anno passato per ciascuna regione e ciascuna categoria di prodotto?”*

*“Che correlazione esiste tra l’andamento dei titoli azionari dei produttori di PC e i profitti trimestrali lungo gli ultimi 5 anni?”*

*“Quali sono gli ordini che massimizzano gli incassi?”*

*“Quale di due nuove terapie risulterà in una diminuzione della durata media di un ricovero?”*

*“Che rapporto c’è tra i profitti realizzati con spedizioni di meno di 10 elementi e quelli realizzati con spedizioni di più di 10 elementi?”*

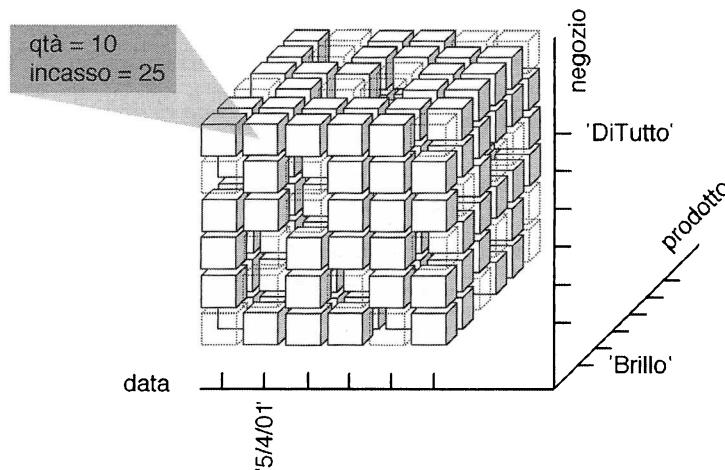
È chiaro che esprimere interrogazioni di questa natura tramite linguaggi tradizionali come SQL risulta alquanto complesso, ed è altrettanto chiaro che la loro esecuzione su database operazionali porterebbe a tempi di risposta difficilmente accettabili.

Il modello multidimensionale prende le mosse dalla constatazione che gli oggetti che influenzano il processo decisionale sono *fatti* del mondo aziendale, quali per esempio le vendite, le spedizioni, i ricoveri, gli interventi chirurgici. Le occorrenze di un fatto corrispondono a *eventi* accaduti: ogni singola vendita o spedizione effettuata è un evento. Per ciascun fatto, interessano in particolare i valori di un insieme di *misure* che descrivono quantitativamente gli eventi: l’incasso di una vendita, la quantità spedita, il costo di un ricovero, la durata di un intervento chirurgico.

Gli eventi che accadono nell’azienda sono evidentemente tantissimi, troppi per poter essere analizzati singolarmente. Per poterli agevolmente selezionare e raggruppare si immagina allora di collocarli in uno spazio *n*-dimensionale i cui assi, chiamati appunto *dimensioni* di analisi, definiscono diverse prospettive per la loro identificazione. Per esempio, le vendite in una catena di negozi possono essere rappresentate in uno spazio tridimensionale le cui dimensioni sono i prodotti, i negozi e le date. Per le spedizioni, le

dimensioni possono essere il prodotto, la data di spedizione, l'ordine, la destinazione e la modalità. I ricoveri possono essere identificati dalla terna reparto, data e paziente, mentre per gli interventi chirurgici occorre aggiungere il tipo di intervento.

È proprio il concetto di dimensione che ha dato origine alla diffusissima metafora del *cubo* per la rappresentazione dei dati multidimensionali. Secondo questa metafora, gli eventi corrispondono a celle di un cubo i cui spigoli rappresentano le dimensioni di analisi (se le dimensioni sono più di tre, si tratta più propriamente di un *iper cubo*). Ogni cella del cubo contiene un valore per ciascuna misura. La Figura 1.8 mostra una rappresentazione grafica intuitiva di un cubo in cui il fatto descritto è la vendita in una catena di negozi. Le dimensioni di analisi sono *negozi*, *prodotto* e *data*; un evento corrisponde alla vendita di un certo prodotto in un certo negozio in un certo giorno, ed è descritto da due misure: la quantità venduta e l'incasso. La figura mette in evidenza il fatto che il cubo è *sparso*, ossia che molti eventi non si sono in effetti verificati: chiaramente, non tutti i prodotti possono essere venduti tutti i giorni in tutti i negozi!



**Figura 1.8** Il cubo a tre dimensioni che modella le vendite in una catena di negozi. Nel negozio 'DiTutto', il giorno 5/4/01 sono state vendute 10 confezioni di detersivo 'Brillo' per un incasso complessivo pari a 25 euro.

Se si volesse rappresentare il cubo delle vendite attraverso il modello relazionale, lo si potrebbe fare tramite lo schema relazionale:

**VENDITE (negozio, prodotto, data, quantità, incasso)**

in cui gli attributi sottolineati sono quelli che costituiscono la chiave primaria e gli eventi corrispondono a tuple (per esempio <'DiTutto', 'Brillo', '5/4/01', 10, 25>). Il vincolo espresso dalla chiave primaria asserisce che non possono esistere due eventi associati alla stessa terna di valori di negozio, prodotto e data, ma anche che ogni terna di valori

determina funzionalmente un unico valore di quantità e un unico valore di incasso. Esiste cioè una dipendenza funzionale<sup>2</sup> del tipo:

negozio, prodotto, data → quantità, incasso

Per evitare malintesi sul significato del termine *evento*, è importante a questo punto precisare che l'insieme delle dimensioni prescelte per rappresentare un fatto identifica univocamente un evento nel modello multidimensionale, ma non necessariamente un evento del dominio applicativo. Per chiarire questa affermazione apparentemente sibillina, si consideri ancora il caso delle vendite. Nel dominio applicativo un singolo evento di *vendita* corrisponde presumibilmente all'acquisto, da parte di un cliente, di un insieme di prodotti in un negozio in una certa data: in pratica, al concetto di *scontrino*.

Dal punto di vista del modello multidimensionale, invece, se il fatto vendite ha dimensioni prodotto, negozio e data, un evento corrisponde al venduto giornaliero complessivo di un certo prodotto in un certo negozio. Come è evidente, la differenza tra le due interpretazioni dipende da un lato dal fatto che uno scontrino contiene in generale più prodotti, dall'altro dal fatto che uno stesso prodotto viene in generale venduto più volte nello stesso giorno e negozio. Nel seguito, quando non diversamente specificato, useremo il termine *evento*, e di conseguenza il termine *fatto*, con riferimento alla granularità che esso assume nel modello multidimensionale.

Normalmente, ciascuna dimensione è associata a una *gerarchia* di livelli di aggregazione (chiamata a volte *gerarchia di roll-up*) che ne raggruppa i valori in diversi modi. Chiameremo *attributi dimensionali* i livelli che compongono una gerarchia. La Figura 1.9 propone un piccolo esempio di gerarchie sulle dimensioni prodotto e negozio: i prodotti sono raggruppati in tipi, ulteriormente suddivisi in categorie; i negozi si trovano in città che a loro volta fanno parte di regioni. In cima a ciascuna gerarchia si trova un livello fittizio che raggruppa tutti i valori relativi a una dimensione. Dal punto di vista della teoria relazionale, una gerarchia è esprimibile tramite un insieme di dipendenze funzionali tra attributi dimensionali:

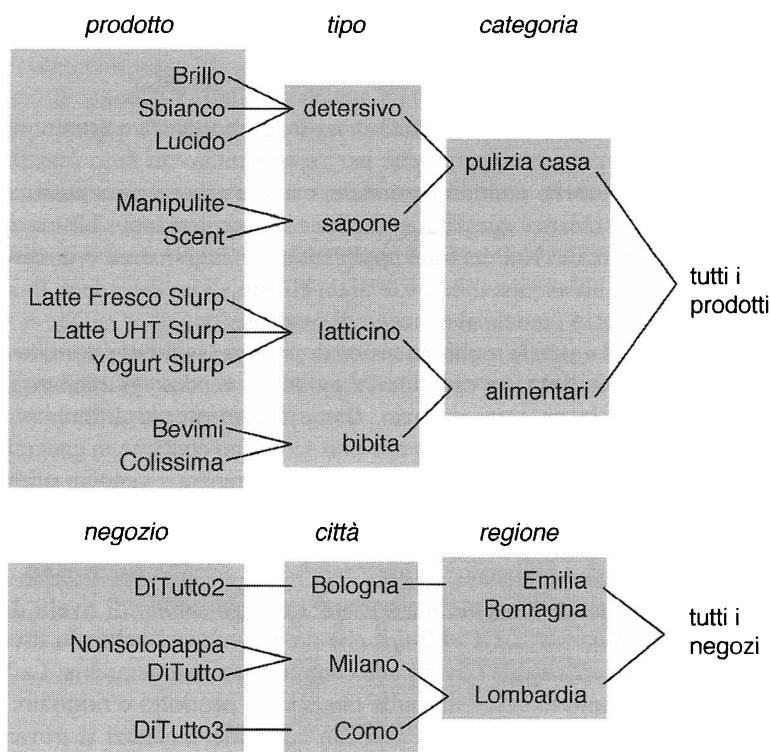
. prodotto → tipo → categoria  
negozio → città → regione

Riassumendo,

un *cubo multidimensionale* è incentrato su un *fatto* di interesse per il processo decisionale. Esso rappresenta un insieme di *eventi*, descritti quantitativamente da *misure numeriche*. Ogni asse del cubo rappresenta una possibile *dimensione* di analisi; ciascuna dimensione può essere vista a più livelli di dettaglio individuati da *attributi strutturati in gerarchie*.

---

<sup>2</sup> La definizione di dipendenza funzionale proviene dalla teoria relazionale. Dato uno schema relazionale  $R$  e due insiemi di attributi  $X = \{a_1, \dots, a_n\}$  e  $Y = \{b_1, \dots, b_m\}$ , si dice che  $X$  determina funzionalmente  $Y$  (scritto  $X \rightarrow Y$ ) se e solo se, per tutte le istanze legali  $r$  di  $R$ , per ciascuna coppia di tuple  $t_1, t_2$  di  $r$  si ha che  $t_1[X] = t_2[X]$  implica  $t_1[Y] = t_2[Y]$ , dove  $t[X/Y]$  denota i valori assunti in  $t$  dagli attributi in  $X/Y$ . Per estensione, nel seguito diremo che esiste una dipendenza funzionale tra due insiemi di attributi  $X$  e  $Y$  quando ciascun insieme di valori di  $X$  corrisponde necessariamente, nel dominio applicativo, a un unico insieme di valori di  $Y$ . Per semplicità di notazione, nell'enumerare gli elementi di ciascun insieme eliminiamo le parentesi graffe.



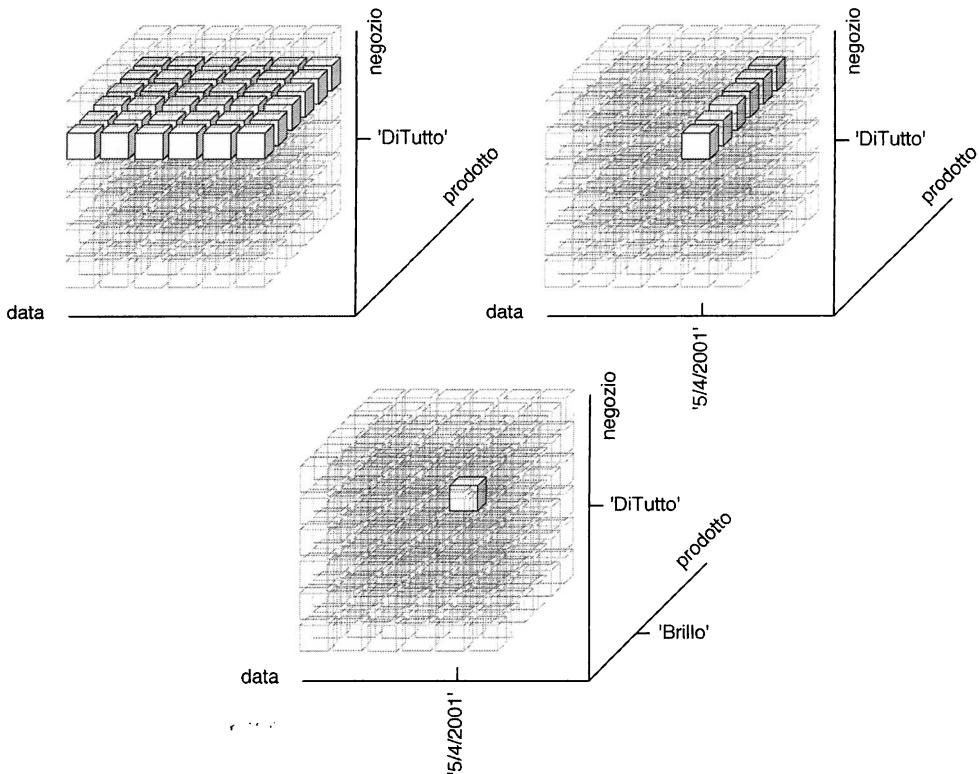
**Figura 1.9** Gerarchie di aggregazione sulle dimensioni prodotto e negozio.

In letteratura sono presenti diverse espressioni formali del modello multidimensionale, più o meno complesse ed esaustive (Agrawal, 1995; Gyssens, 1997; Datta, 1997; Vassiliadis, 1998; Cabibbo, 1998). È opportuno a tale proposito riportare un breve cenno a terminologie alternative adottate per il modello multidimensionale in letteratura e negli strumenti commerciali. I termini *fatto* e *cubo* vengono spesso usati in modo intercambiabile. C'è accordo sostanziale sull'uso del termine *dimensioni* per denotare le coordinate che permettono di classificare e identificare le occorrenze del fatto; a volte però, invece di riferirsi in particolare all'attributo che determina la granularità di aggregazione più fine, vengono chiamate *dimensioni* le intere gerarchie (per esempio, il termine *dimensione temporale* si può riferire all'intera gerarchia costruita sulla dimensione *data*). Le misure sono a volte dette *variabili*, *metriche*, *proprietà*, *attributi* o *indicatori*. Gli attributi dimensionali delle gerarchie vengono, in alcuni modelli, chiamati *livelli* o *parametri*.

Osserviamo ora che le informazioni rappresentate nel cubo multidimensionale, pur costituendo di fatto una sintesi di quelle memorizzate nella base di dati operazionale (dove per esempio vengono distinte le *singole vendite*), sono ancora difficilmente fruibili dall'utente a causa della loro quantità. Se la catena comprende 50 negozi che vendono complessivamente 1000 prodotti, e il DW copre 3 anni di transazioni (circa 1000 giorni), il numero totale di eventi possibili risulta pari a  $50 \times 1000 \times 1000 = 5 \times 10^7$ . Anche supponen-

do che, in ciascun giorno, ogni negozio riesca a vendere solo il 10% dei prodotti disponibili, il numero complessivo degli eventi risulta pari a  $5 \times 10^6$ : ancora troppi per poter essere analizzati da un utente senza far ricorso a strumenti automatici.

Le tecniche per ridurre la quantità di dati e ottenere così informazioni utili sono essenzialmente due: la *restrizione* e l'*aggregazione*; per entrambe, come vedremo nei due paragrafi seguenti, la metafora del cubo offre un'agile e intuitiva chiave di interpretazione.



**Figura 1.10** Slicing di un cubo tridimensionale.

### 1.5.1 Restrizione

Restringere i dati significa ritagliare una porzione dal cubo circoscrivendo il campo di analisi; nella terminologia dell'algebra relazionale, effettuare selezioni e/o proiezioni.

La forma più semplice di restrizione è il cosiddetto *slicing* (letteralmente, affettatura) dei dati, illustrato in Figura 1.10, in cui si riduce la dimensionalità del cubo fissando un valore per una o più dimensioni. Vediamo un esempio per il cubo delle vendite. Se si fissa un valore per una delle dimensioni, per esempio *negozio*=*'DiTutto'*, si ottiene come risultato l'insieme degli eventi associati alle vendite effettuate presso il negozio *DiTutto*: secondo la metafora si tratterà di un piano, ovvero una "fettina" di dati agevolmente

visualizzabile all'interno di un foglio elettronico (secondo le ipotesi sui volumi di dati introdotte in precedenza, si avranno circa  $10^5$  eventi). Se vengono fissate due dimensioni, per esempio `negozi='DiTutto'` e `data='5/4/2001'`, il risultato sono tutte le vendite di prodotti distinti effettuate presso DiTutto il 5 aprile (circa 100 eventi); graficamente, l'intersezione di due piani perpendicolari, ovvero una retta. Infine, se tutte le dimensioni vengono fissate, si identifica un unico evento corrispondente a un punto nello spazio tridimensionale delle vendite.

La *selezione* è una generalizzazione dello slicing in cui si riduce la grandezza del cubo esprimendo condizioni sugli attributi dimensionali. Per esempio, si possono selezionare le sole vendite di detersivo Brillo nei negozi di Bologna nei giorni di gennaio 2001. In questo modo, se i 50 negozi sono uniformemente distribuiti su 10 città, il numero di eventi da esaminare passa a  $5 \times 31 = 155$ , comodamente visionabili all'interno di una matrice bidimensionale o in un grafico. Infine, la *proiezione* è riconducibile alla scelta di mantenere, per ciascun evento, solo un sottoinsieme di misure, scartando le altre.

### 1.5.2 Aggregazione

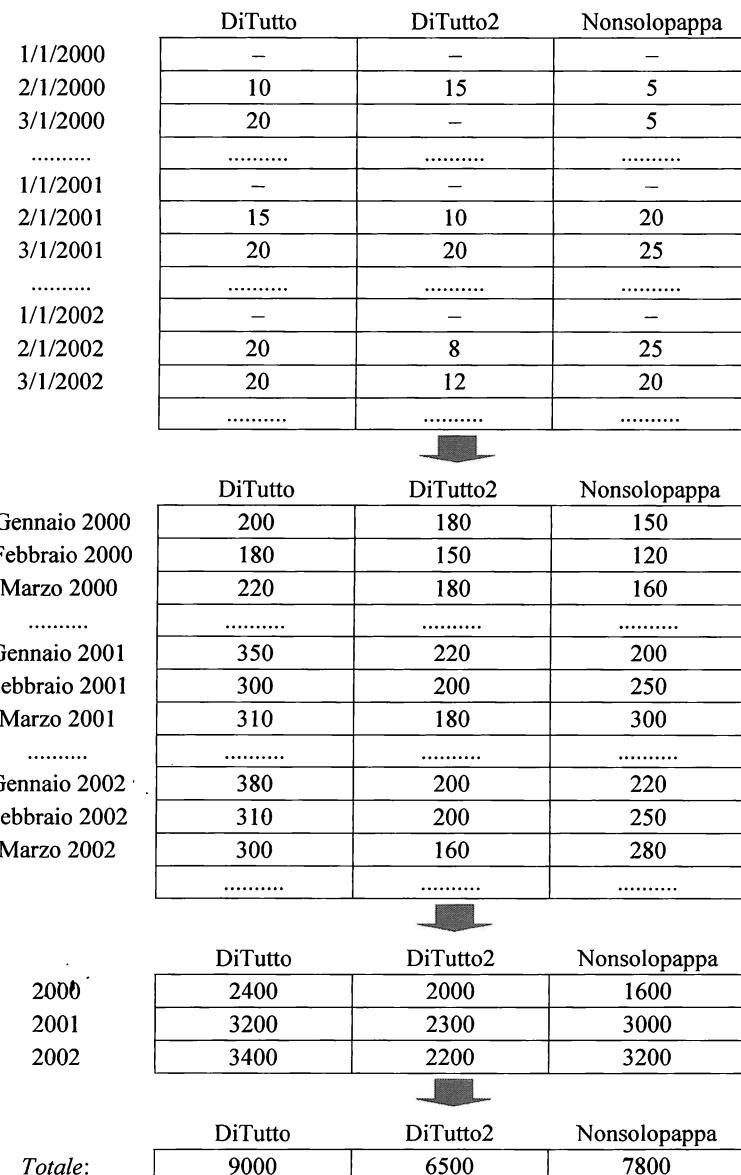
L'aggregazione è un meccanismo di importanza fondamentale nelle basi di dati multidimensionali. Si supponga di voler analizzare le vendite non nel loro dettaglio giornaliero, bensì a livello mensile; continuando la metafora del cubo, ciò significa raggruppare, per ciascun prodotto e negozio, tutte le celle relative ai giorni di uno stesso mese in un'unica macro-cellula. Nel cubo così aggregato, il numero complessivo di eventi (inteso come il numero di macro-celle risultanti) sarà pari a  $50 \times 1000 \times 36$ , poiché sulla dimensione tempo la granularità non è più a livello di giorno ma di mese, e 36 sono i mesi nel triennio. Ciascun evento conterrà una sintesi dei dati presenti negli eventi che esso aggredisce: nel caso in esame, il numero totale di esemplari venduti nel mese e l'incasso complessivo calcolati sommando i valori elementari delle corrispondenti misure (Figura 1.11). Aggregando ulteriormente sul tempo, per ogni combinazione prodotto-negozi si possono ottenere tre soli eventi: uno per ciascun anno. Al massimo livello di aggregazione sulla dimensione tempo, ciascuna combinazione corrisponde a un unico evento, che riporta il numero totale di esemplari di un prodotto venduti in un negozio nei tre anni e l'incasso complessivo.

L'aggregazione può essere operata contemporaneamente su più dimensioni. Per esempio, come mostrato in Figura 1.12, è possibile aggregare le vendite per mese, tipo di prodotto e città del negozio, nonché solo per mese e tipo di prodotto. Inoltre, selezione e aggregazione possono essere combinate per permettere un processo di analisi mirato con precisione alle esigenze dell'utente.

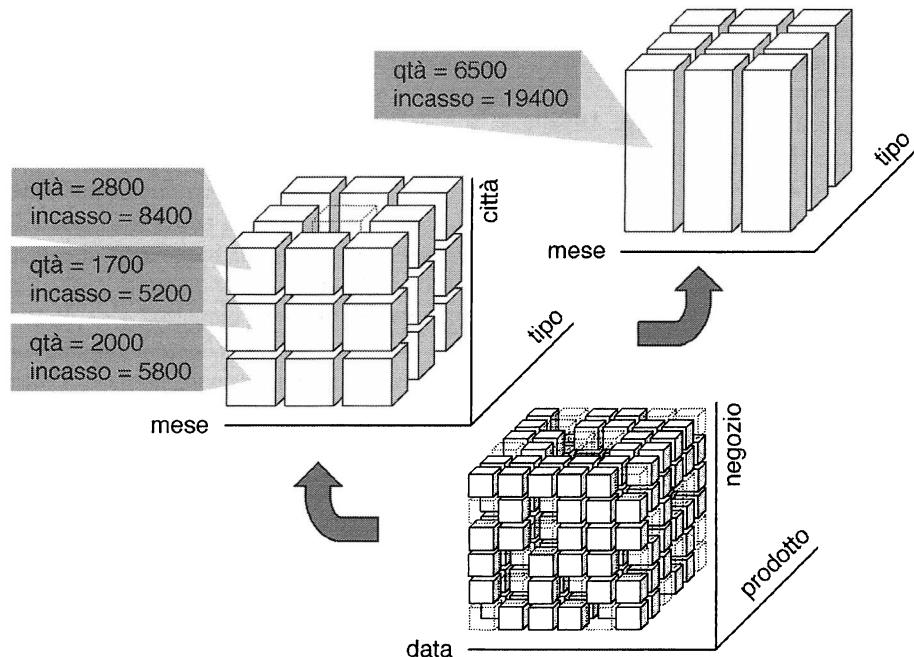
## 1.6 I meta-dati

Il termine *meta-dati* si applica ai dati usati per descrivere altri dati. Nel contesto del data warehousing, in cui giocano un ruolo sostanziale, essi indicano le sorgenti, il valore, l'uso e le funzioni dei dati memorizzati nel DW e descrivono come i dati sono alterati e trasformati durante il passaggio attraverso i diversi livelli dell'architettura. Come mostrato nelle Figure 1.3, 1.4 e 1.5, il contenitore dei meta-dati è strettamente collegato al DW vero e

proprio; le applicazioni ne fanno un intenso uso sia sul lato dell'alimentazione sia su quello dell'analisi.



**Figura 1.11** Aggregazione sulla gerarchia temporale operata sulle quantità vendute per un dato prodotto in tre negozi. Un trattino indica l'assenza di un evento dovuta al fatto che non è stato venduto alcun esemplare.



**Figura 1.12** Due livelli di aggregazione a partire dai dati elementari. I valori delle misure in ogni macro-evento sono la somma dei valori per gli eventi componenti. Nel mese in esame, sono stati venduti complessivamente 6500 esemplari di prodotti del tipo in esame, ripartiti su 3 città, per un incasso complessivo pari a 19 400 euro.

Secondo Kelly (1997) è possibile distinguere due categorie di meta-dati, parzialmente sovrapposte, in base ai differenti utilizzi che ne fanno l'amministratore del sistema e gli utenti finali. I *meta-dati interni*, di interesse per l'amministratore, descrivono tra gli altri le sorgenti, le trasformazioni, le politiche di alimentazione, gli schemi logici e fisici, i vincoli, i profili degli utenti. I *meta-dati esterni*, di interesse per gli utenti, riguardano per esempio le definizioni, la qualità, le unità di misura, le aggregazioni significative.

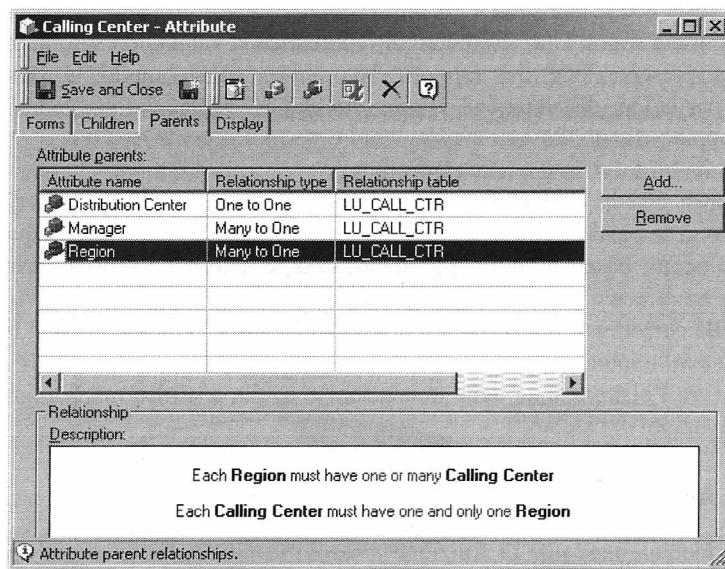
I meta-dati sono immagazzinati in un apposito contenitore, al quale tutti gli altri componenti dell'architettura possono accedere. Requisiti desiderabili per lo strumento di gestione dei meta-dati sono i seguenti (Kelly, 1997):

- permettere di svolgere funzioni di amministrazione, legate in particolare alla sicurezza;
- rendere possibile il browsing e l'interrogazione dei meta-dati da parte degli utenti finali;
- essere dotato di un'interfaccia grafica;
- permettere l'estensione dei meta-dati da parte degli utenti;
- essere aperto per importazioni/esportazioni dei meta-dati verso altri strumenti e formati standard.

Per quanto concerne il formato di rappresentazione, OMG ha recentemente proposto uno standard chiamato *Common Warehouse Metamodel* (CWM), che si poggia su tre standard affermati quali UML (*Unified Modeling Language*), XML (*eXtensible Markup Language*) e XMI (*XML Metadata Interchange*) ed è stato elaborato grazie allo sforzo congiunto di partner tra cui IBM, Unisys, NCR, Oracle. Si tratta di una specifica che descrive lo scambio di meta-dati tra tecnologie legate al data warehousing, alla *business intelligence*, alla gestione della conoscenza e ai portali web; per una descrizione completa e approfondita si veda OMG (2000).

La Figura 1.13 riporta, a titolo di esempio, un'immagine che mostra come si visualizzano i meta-dati esterni riguardanti le gerarchie in *MicroStrategy Desktop*, della suite di strumenti MicroStrategy 8 (MicroStrategy, 2005). In particolare, in figura si visualizzano gli attributi padri dell'attributo *Calling Center*; si evince che un centro di chiamata fa capo a un centro di distribuzione, appartiene a una regione ed è gestito da un manager.

Per un'approfondita trattazione degli aspetti collegati alla rappresentazione e alla gestione dei meta-dati, rimandiamo il lettore a Jarke e altri (2000), a Kelly (1997) e a Barquin e Edelstein (1996).



**Figura 1.13** Accesso ai meta-dati sulle gerarchie in MicroStrategy.

## 1.7 Accedere al DW: reportistica e OLAP

L'ultimo livello comune a tutte le architetture di data warehousing è quello dell'analisi. Infatti, una volta che i dati sono stati ripuliti, integrati e trasformati, occorre capire come trarne il massimo vantaggio informativo. Nei prossimi paragrafi presenteremo i due principali approcci all'interrogazione di un DW da parte degli utenti finali: la *reportistica* e l'*OLAP*. Le informazioni contenute nel DW vengono inoltre spesso utilizzate come base di

# Il ciclo di vita dei sistemi di data warehousing

I sistemi di data warehousing stanno acquisendo popolarità via via che aziende e imprese nei più disparati settori ne intuiscono i benefici. D'altronde, larga parte delle organizzazioni mancano della necessaria esperienza e capacità per affrontare con successo le sfide implicite nei progetti di data warehousing. In particolare, uno dei fattori che a tutt'oggi maggiormente minaccia la riuscita di questi progetti è la mancata adozione di un approccio metodologico, che ripropone l'esperienza fatta da altri e minimizza i rischi di insuccesso essendo basato su un'analisi costruttiva degli errori commessi.

Questo capitolo è dedicato alla disamina di alcune metodologie descritte in letteratura per la gestione dell'intero ciclo di vita dei sistemi di data warehousing e alla definizione di un quadro metodologico per lo sviluppo di ciascun data mart, che fungerà da riferimento per i capitoli successivi.

## 2.1 Fattori di rischio

Innanzitutto, come diceva uno stratega cinese 4000 anni fa, “conosci il tuo nemico”: prima di esaminare le metodologie, è allora opportuno passare velocemente in rassegna i principali fattori di rischio e le tipiche ragioni di fallimento dei progetti di data warehousing (Kelly, 1997). È possibile evidenziare quattro categorie di rischio:

- *Rischi legati alla gestione del progetto.* Un progetto di data warehousing coinvolge trasversalmente tutti i livelli aziendali e porta con sé importanti implicazioni politiche e organizzative. Da questo punto di vista, un fattore che spesso ostacola l'attuazione dei progetti di data warehousing è la necessità di condivisione dell'informazione tra i reparti, che viene vissuta con riluttanza dai responsabili aziendali poiché può comportare “perdita di potere” e far emergere le carenze del proprio operato (Vassiliadis, 2000). Altre frequenti cause di insuccesso sono legate alla definizione dell'ambito e delle finalità del sistema e alla sua sponsorizzazione; in particolare, parecchi progetti in aziende medio-piccole sono stati stroncati sul nascere dall'incapacità del progettista di presentare una convincente valutazione dei costi e dei benefici.

- *Rischi legati alle tecnologie.* Le tecnologie per la progettazione, l'implementazione, l'uso e il mantenimento dei sistemi di data warehousing si stanno evolvendo rapidamente, e l'architettura proposta deve essere in grado di tenere il passo. Problemi classici a questo livello sono la scarsa scalabilità dell'architettura in termini di volume dati e numero di utenti; l'assenza di estendibilità per accogliere nuove tecnologie, componenti o applicazioni; la gestione inefficiente dell'interscambio di meta-dati tra i diversi componenti.
- *Rischi legati ai dati e alla progettazione.* Questi fattori di rischio dipendono dalla qualità dei dati e del progetto realizzato. Citemo in particolare il rischio di ottenere risultati di scarso valore a causa dell'instabilità e inaffidabilità delle sorgenti, oppure a causa dell'inaccurata specificazione dei requisiti utente. Un altro frequente motivo di insuccesso dipende dall'incapacità di fornire un elevato valore aggiunto agli utenti alla consegna dei primi prototipi, che mina la fiducia nell'intero progetto.
- *Rischi legati all'organizzazione.* Assolutamente non ultime in ordine di importanza tra le cause di insuccesso, l'incapacità di impegnare l'utente finale in un reale interesse e sostegno per il progetto, la difficoltà di trasformare radicalmente la cultura aziendale per renderla conscia del valore dell'informazione, l'impossibilità degli utenti di trarre vantaggio dai risultati ottenuti causa inerzia organizzativa.

Complessivamente, il rischio di ottenere un risultato insoddisfacente nei progetti di data warehousing è particolarmente alto proprio a causa delle elevatissime aspettative degli utenti. Nella cultura aziendale contemporanea, è infatti diffusissima la credenza che attribuisce al data warehousing il ruolo di panacea in grado di sanare tutte le incongruità dell'organizzazione e le manchevolezze del sistema informativo aziendale. In realtà, come si è appena visto, una larga parte della responsabilità della riuscita del progetto ricade sulla qualità dei dati sorgente e sulla lungimiranza, disponibilità e dinamismo del personale dell'azienda.

## 2.2 Top-down vs. bottom-up

Nell'affrontare l'aspetto metodologico, il punto fondamentale da discutere riguarda la natura top-down o bottom-up dell'approccio da seguire per la costruzione del DW, che incide profondamente sulla struttura del ciclo di vita. In sintesi, adottare un approccio top-down significa analizzare i bisogni globali dell'intera azienda e pianificare lo sviluppo del DW per poi progettarlo e realizzarlo *nella sua interezza*. Questa modalità di lavoro sembra promettere ottimi risultati, poiché si basa su una visione globale dell'obiettivo e garantisce in linea di principio di produrre un DW consistente e ben integrato; d'altronde, una lunga storia di fallimenti ha insegnato che:

- il preventivo di costi onerosi a fronte di lunghi tempi di realizzazione scoraggia la direzione dall'intraprendere il progetto;
- affrontare contemporaneamente l'analisi e la riconciliazione di *tutte* le sorgenti di interesse è un compito estremamente complesso;

- riuscire a prevedere a priori nel dettaglio le esigenze delle diverse aree aziendali impegnate nel progetto è pressoché impossibile, e il processo di analisi rischia di subire una paralisi;
- il fatto di non prevedere la consegna a breve termine di un prototipo non permette agli utenti di verificare l'utilità del progetto e ne fa scemare l'interesse e la fiducia.

Una tecnica più prudente, ormai quasi universalmente accettata, prevede invece di procedere bottom-up.

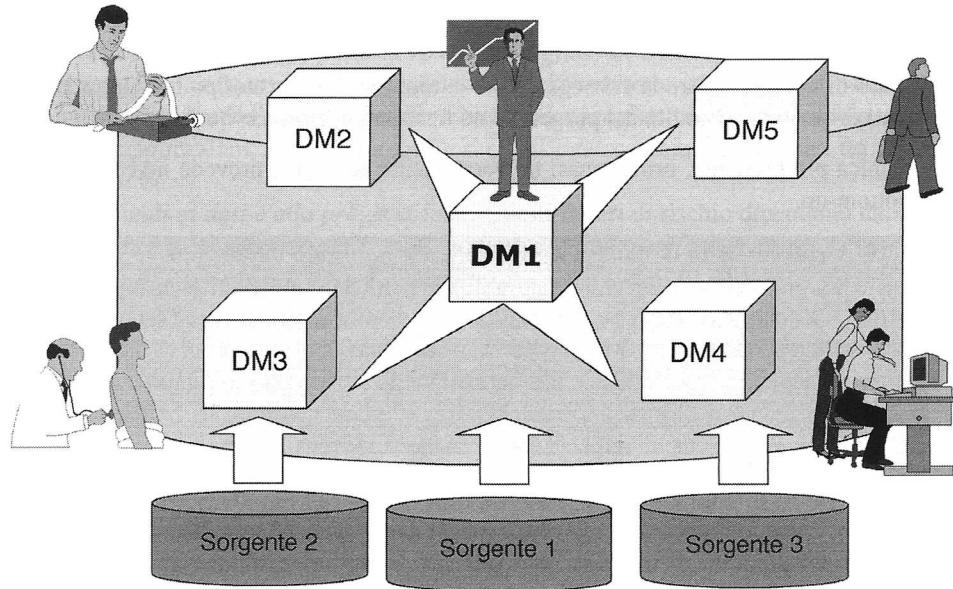
Il DW viene costruito in modo incrementale, assemblando iterativamente più data mart, ciascuno dei quali incentrato su un insieme di fatti collegati a uno specifico settore aziendale e di interesse per una certa categoria di utenti (per esempio data mart del magazzino, del marketing ecc.). Abbinando questo approccio a tecniche di prototipazione veloce si riducono notevolmente l'intervallo di tempo e il costo necessari affinché la dirigenza aziendale possa avere un riscontro sull'effettiva utilità del sistema in via di realizzazione, mantenendo così costantemente elevata l'attenzione sul progetto.

Ovviamente, anche questo approccio non è esente da rischi in quanto determina una visione parziale del dominio di interesse. Per ottenere i migliori risultati è importante porre attenzione alla selezione del primo data mart da prototipare: deve essere quello che gioca il ruolo più strategico per l'azienda e deve ricoprire un ruolo centrale e di riferimento per l'intero DW, cosicché i successivi data mart possano essere con esso agevolmente integrati. Inoltre, è altamente consigliabile che il data mart prescelto si appoggi su fonti dati già disponibili e consistenti.

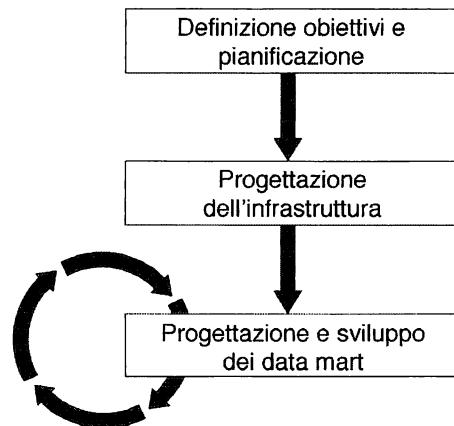
La Figura 2.1 mostra un esempio in ambito ospedaliero. Il primo data mart implementato, DM1, è quello per la rilevazione e la rendicontazione delle attività ospedaliere. Seguono progressivamente un data mart per l'area contabile (DM2), uno per l'area epidemiologica (DM3), uno per la gestione del personale (DM4) e infine uno per l'area degli acquisti (DM5).

I punti fondamentali del ciclo di vita per lo sviluppo bottom-up di un sistema di data warehousing sono schematizzati in Figura 2.2 (Mattison, 1996) e illustrati nel seguito:

1. *Definizione degli obiettivi e pianificazione.* Questa fase preliminare si basa su uno studio di fattibilità orientato alla chiara individuazione degli obiettivi e dei confini del sistema, alla stima delle sue dimensioni, alla scelta dell'approccio per la costruzione, alla valutazione dei costi e del valore aggiunto. Vengono inoltre affrontate questioni di tipo organizzativo attraverso l'analisi dei rischi e delle aspettative e lo studio delle competenze del gruppo di lavoro. Viene così prodotto un piano di attuazione del progetto di data warehousing che viene sottoposto per l'approvazione alla direzione.
2. *Progettazione dell'infrastruttura.* Durante questa fase si analizzano e si comparano le possibili soluzioni architettoniche valutando le tecnologie e gli strumenti disponibili, al fine di realizzare un progetto di massima dell'intero sistema.
3. *Progettazione e sviluppo dei data mart.* Ciascuna iterazione di questa fase comporta la creazione di un nuovo data mart e di nuove applicazioni, che vengono via via integrate nel sistema di data warehousing.



**Figura 2.1** Costruzione bottom-up di un DW in ambito ospedaliero.

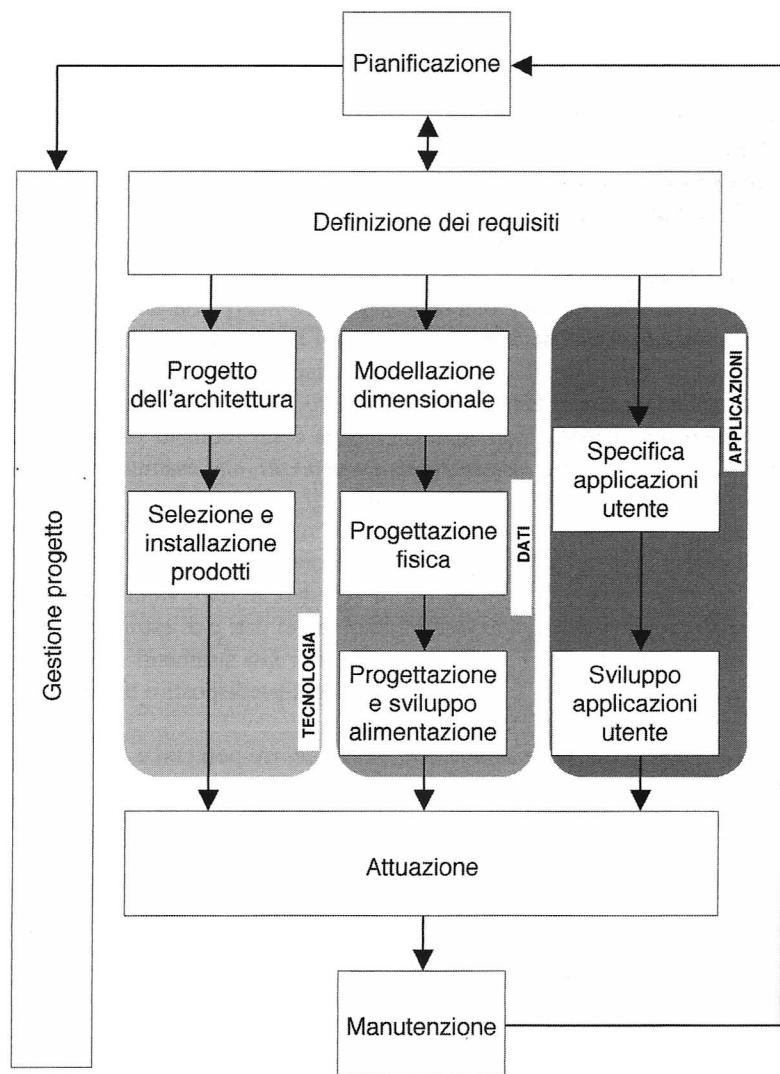


**Figura 2.2** Il ciclo di vita del DW.

Sul quadro generale appena delineato c'è ormai ampio accordo. Nei paragrafi seguenti si vedranno due esempi specifici di metodologie che a questo quadro sono conformi, nate entrambe dall'esperienza progettuale "sul campo".

## 2.2.1 Il “Business Dimensional Lifecycle”

Il *business dimensional lifecycle* è il ciclo di vita per la progettazione, lo sviluppo e l'attuazione di sistemi di data warehousing riportato da Kimball (1998). Evolutosi nel corso di quasi vent'anni di esperienza professionale, ha attualmente la struttura illustrata in Figura 2.3. Nel seguito ne vengono brevemente esaminate le diverse fasi.



**Figura 2.3** Business dimensional lifecycle (Kimball, 1998).

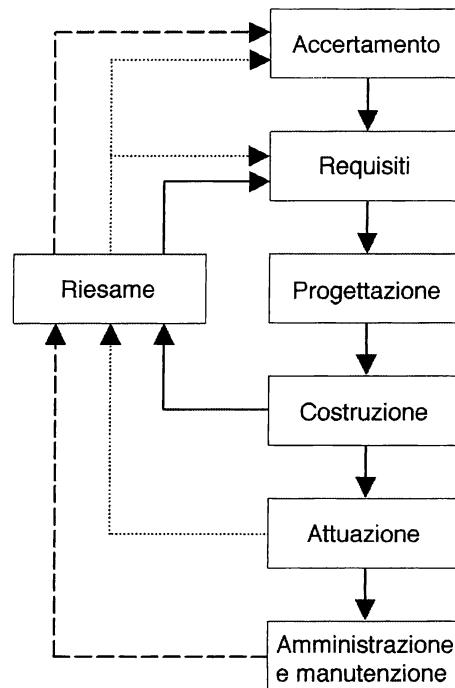
1. La *pianificazione del progetto* include la definizione degli scopi e dei confini del sistema, la valutazione degli impatti organizzativi, la stima dei costi e dei benefici, l'allocazione delle risorse necessarie, infine l'identificazione di un piano di massima per l'attuazione del progetto.
2. La *definizione dei requisiti* ha un ruolo primario nell'assicurare che le richieste degli utenti siano correttamente e interamente recepite dai progettisti, così da massimizzare l'utilità e la redditività del sistema in via di definizione. Durante questa fase, i progettisti devono catturare i fattori chiave del processo decisionale e trasformarli in specifiche che guidino la progettazione. La definizione dei requisiti dà il via a tre percorsi paralleli, ciascuno articolato in differenti fasi: quello *dei dati*, quello *della tecnologia* e quello *delle applicazioni*.
3. La prima fase del percorso dei dati è la *modellazione dimensionale*, durante la quale i requisiti espressi dagli utenti e l'analisi delle sorgenti operazionali guidano la definizione delle strutture che ospiteranno i dati all'interno del DW. In uscita si ottengono un insieme di schemi logici e l'insieme delle corrispondenze con gli schemi sorgente. Il percorso dati prosegue con la *progettazione fisica*, che si occupa principalmente delle questioni legate all'implementazione degli schemi logici sul DBMS prescelto (per esempio, indicizzazione e partizionamento), per terminare con la *progettazione e sviluppo dell'alimentazione* che affronta tutti i temi legati a estrazione, trasformazione e caricamento, non ultimo quello della qualità dei dati.
4. Il percorso della tecnologia comprende il *progetto dell'architettura*, basato sull'attuale quadro tecnico dei sistemi informatici d'azienda e sui requisiti prestazionali espressi dagli utenti, e la *selezione e installazione dei prodotti*, che esamina e valuta le piattaforme hardware utilizzabili nonché i DBMS, gli strumenti ETL e gli strumenti per l'analisi dei dati disponibili in commercio.
5. Durante il percorso delle applicazioni si raccolgono le specifiche relative alle applicazioni che permetteranno agli utenti finali di accedere ai dati, valutando le necessità di produzione di rapporti, di navigazione interattiva dei dati e di estrazione automatica di conoscenza (fase di *specifica applicazioni utente*). Gli strumenti di analisi scelti durante la fase di selezione dei prodotti verranno poi predisposti e configurati di conseguenza (fase di *sviluppo applicazioni utente*).
6. La fase di *attuazione* costituisce la convergenza dei tre percorsi e comporta l'effettivo avviamento del sistema sviluppato.
7. L'attuazione non termina il ciclo di vita del sistema, che necessita di una continua *manutenzione* per assicurare supporto e formazione agli utenti.
8. Durante tutte le fasi del ciclo di vita, un'accurata *gestione del progetto* permette di mantenere le diverse attività sincronizzate, di monitorare lo stato corrente del progetto e di assicurare che il team dei progettisti resti costantemente in stretto contatto con l'organizzazione aziendale.

### 2.2.2 La "Rapid Warehousing Methodology"

La *rapid warehousing methodology* è una metodologia iterativa ed evolutiva per la gestione di progetti di data warehousing, pensata da una delle aziende leader del settore (SAS Institute, 1998). Essa si basa sulla suddivisione di progetti vasti e potenzialmente rischiosi

in sottoprogetti, più piccoli e molto meno rischiosi, chiamati *build*. Ciascun sottoprogetto riprende l'ambiente sviluppato dal sottoprogetto precedente, estendendolo per conseguire nuove funzionalità e facendolo evolvere per far sì che continui a soddisfare i mutevoli bisogni degli utenti. Questo approccio mantiene vivo nel tempo il coinvolgimento e l'interesse degli utenti, gettando le basi per il successo a lungo termine del progetto.

La successione di fasi prevista da questa metodologia è illustrata in Figura 2.4 e brevemente discussa nel seguito.



**Figura 2.4** Rapid warehousing methodology (SAS Institute, 1998).

1. *Accertamento*. L'obiettivo di questa fase, in larga parte corrispondente alla fase di *pianificazione* di Kimball, è duplice: verificare che l'azienda sia effettivamente pronta ad affrontare il progetto da un lato, determinarne gli scopi, i rischi e i benefici dall'altro.
2. *Requisiti*. Questa fase corrisponde alla *definizione dei requisiti* e alla *specifica dalle applicazioni utente* di Kimball, e consiste nella raccolta delle specifiche di analisi, di progetto e di architettura per l'intero sistema.
3. *Progettazione*. L'attenzione è ora concentrata su un solo sottoprogetto per volta. Le specifiche di analisi vengono raffinate per generare il progetto logico e fisico dei dati e il progetto dell'alimentazione; vengono inoltre selezionati gli strumenti di implementazione.

4. *Costruzione*. Il DW viene implementato e popolato con i dati estratti dalle sorgenti, le applicazioni di front-end vengono sviluppate e collaudate.
5. *Attuazione*. Finalmente il sistema viene consegnato e avviato, dopo che gli utenti sono stati adeguatamente addestrati.
6. *Amministrazione e manutenzione*. Questa fase continua durante tutta la vita del sistema e prevede l'estensione delle sue funzionalità, il ridimensionamento dell'architettura per venire incontro ai nuovi fabbisogni, il controllo della qualità dei dati.
7. *Riesame*. Ciascun sottoprogetto si accompagna a tre processi di riesame: uno a verifica dell'implementazione, uno a seguito dell'attuazione per accertarsi che il sistema sia stato ben accettato dall'organizzazione, uno a valle dell'intero processo per misurarne i benefici effettivi.

## 2.3 Le fasi di progettazione di un data mart

Come emerso dai paragrafi precedenti, il punto focale di ciascuna iterazione prevista durante lo sviluppo bottom-up di un DW è la costruzione di un singolo data mart, processo reso complesso anche dal fatto che richiede l'adozione di tecniche di progettazione completamente diverse da quelle utilizzate per le tradizionali basi di dati operazionali.

Il quadro metodologico per la progettazione di data mart tracciato in questo testo include le sette fasi riassunte in Tabella 2.1, di cui nei paragrafi seguenti anticipiamo una succinta descrizione rimandando ai capitoli specifici per una trattazione dettagliata. Una vista funzionale di massima dei flussi informativi scambiati tra le diverse fasi nel caso più generale è schematizzata in Figura 2.5. Restano escluse le fasi di implementazione delle applicazioni utente e dell'alimentazione, che dipendono principalmente dagli specifici strumenti e linguaggi utilizzati e non sono pertanto trattate in questo testo.

Nel Paragrafo 2.4 si vedrà come le sette fasi descritte siano diversamente collocabili nel quadro metodologico di riferimento a seconda dell'approccio alla progettazione che le specifiche circostanze suggeriscono di seguire.

### 2.3.1 Analisi e riconciliazione delle fonti dati

Questa fase di progettazione richiede di definire e documentare lo schema del livello dei dati operazionali a partire dal quale verrà alimentato il data mart: quello che dal punto di vista architettonico, nella terminologia del Capitolo 1, abbiamo chiamato *livello dei dati riconciliati*.

Occorre quindi analizzare e comprendere gli schemi delle sorgenti disponibili (*ricognizione*), eventualmente trasformarli per portare alla luce correlazioni utili precedentemente inespresso (*normalizzazione*), determinare quali porzioni siano utili ai fini del processo decisionale nel settore aziendale cui il data mart è dedicato e infine valutare la qualità dei dati. Qualora le sorgenti da utilizzare siano più d'una, i loro schemi dovranno inoltre essere sottoposti a un'attività di omogeneizzazione e integrazione tesa a individuare i tratti comuni e a sanare le eventuali inconsistenze.

**Tabella 2.1** Le sette fasi della progettazione di un data mart.

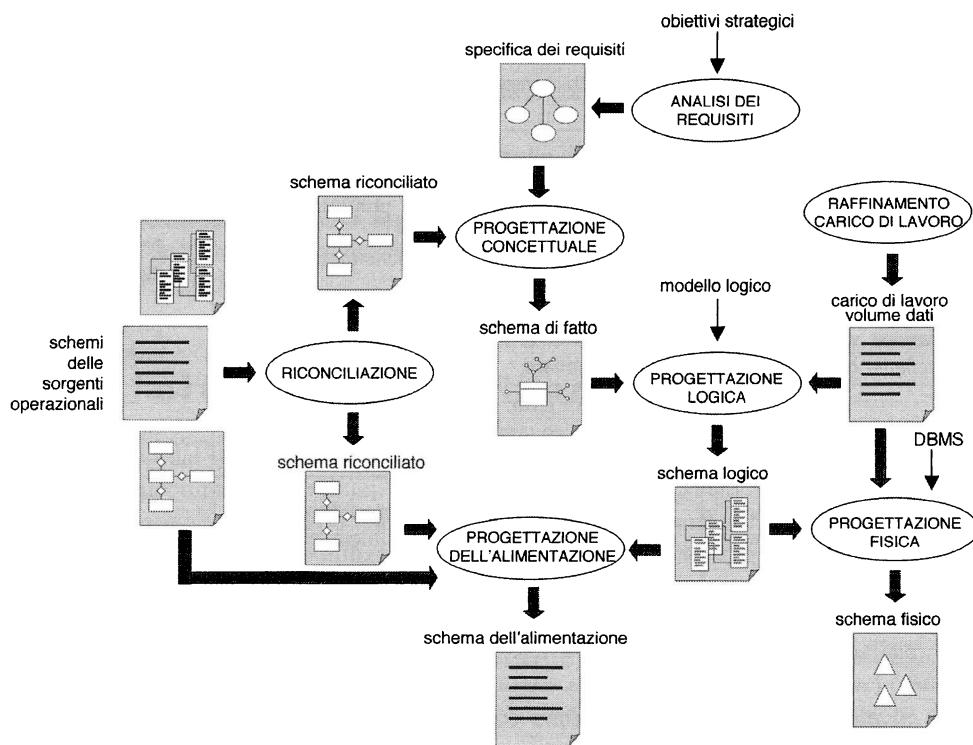
Fase	Ingresso	Uscita	Figure coinvolte
<i>Analisi e riconciliazione delle fonti dati</i>	schemi delle sorgenti	schema riconciliato	progettista; amministratori db operazionale
<i>Analisi dei requisiti</i>	obiettivi strategici	specifiche dei requisiti; carico di lavoro preliminare	progettista; utenti finali
<i>Progettazione concettuale</i>	schema riconciliato; specifiche dei requisiti	schema di fatto	progettista; utenti finali
<i>Raffinamento carico di lavoro, validazione schema concettuale</i>	schemi di fatto; carico di lavoro preliminare	carico di lavoro; schemi di fatto validati	progettista; utenti finali
<i>Progettazione logica</i>	schemi di fatto; modello logico target; carico di lavoro	schema logico del data mart	progettista
<i>Progettazione dell'alimentazione</i>	schemi delle sorgenti; schema riconciliato; schema logico del d.m.	procedure di alimentazione	progettista; amministratori db operazionale
<i>Progettazione fisica</i>	schema logico del data mart; DBMS target; carico di lavoro	schema fisico del data mart	progettista

Oltre al progettista, in questa fase sono coinvolti gli amministratori dei database operazionali, che tipicamente sono gli unici in grado di attribuire un significato a schemi e tracciati record spesso incomprensibili; inoltre, la loro conoscenza del dominio applicativo è indispensabile per normalizzare gli schemi. La tipica carenza o addirittura assenza di documentazione delle basi di dati operazionali fa sì che questa fase risulti essere la più lunga (può infatti richiedere anche il 50% dei tempi dell'intero progetto) e, in genere, la più ardua. Il Capitolo 3 è dedicato alla fase di analisi e riconciliazione delle fonti dati. Dopo aver toccato i temi della ricognizione e normalizzazione delle sorgenti, la trattazione si focalizza sul più complesso problema dell'integrazione.

### 2.3.2 Analisi dei requisiti

Nella fase di analisi dei requisiti il progettista raccoglie, filtra e documenta i requisiti degli utenti finali, con l'obiettivo di delineare quali sono le informazioni di interesse da rappresentare in sintonia con gli obiettivi strategici perseguiti. In uscita da questa fase vengono prodotte specifiche sui fatti da modellare e indicazioni preliminari sul carico di lavoro.

Come si è visto nel Paragrafo 1.5, i fatti sono concetti di interesse primario per il processo decisionale; le loro occorrenze sono eventi che accadono dinamicamente nell'azienda. La scelta dei fatti è essenzialmente responsabilità dell'utente finale, guidato in questo compito cruciale dal progettista sulla base della documentazione del livello riconciliato prodotta al passo precedente. Per ogni fatto occorre definire l'intervallo di storicizzazione, ovvero quale arco temporale dovranno abbracciare gli eventi memorizzati.



**Figura 2.5** Le sette fasi della progettazione.

Il carico di lavoro preliminare può essere espresso in linguaggio pseudo-naturale e ha l'obiettivo di permettere al progettista di identificare dimensioni e misure per la progettazione concettuale; per ciascun fatto, esso dovrebbe specificare le misure quantitative e le aggregazioni più interessanti. Naturalmente abbinata alla determinazione del carico di lavoro è la scelta del livello minimo di sintesi delle informazioni, ovvero la definizione della **granularità** nella rappresentazione dei fatti, la cui adeguatezza costituisce un fattore critico per la riuscita dell'intero progetto poiché determina la flessibilità di interrogazione del data mart.

La granularità è determinata da un compromesso tra velocità di risposta richiesta al sistema e livello di dettaglio massimo delle interrogazioni. Quando ciò non appesantisce troppo il sistema può essere conveniente utilizzare una granularità più fine di quella necessaria agli utenti, poiché ciò conferisce una maggiore flessibilità di utilizzo.

Le problematiche tipiche dell'analisi dei requisiti utente sono affrontate nel Capitolo 4, in cui verrà anche proposto un approccio specifico basato su un formalismo grafico *goal-oriented*.

### 2.3.3 Progettazione concettuale

Sebbene la successione delle tre fasi di progettazione concettuale, logica e fisica sia apparentemente simile a quella che regola il processo di costruzione di una base di dati operazionale, è importante evidenziare che la progettazione di un data mart segue principi molto diversi, poiché diverse sono le caratteristiche del carico di lavoro.

Secondo l'approccio seguito in questo testo, la progettazione concettuale comporta l'utilizzo dei requisiti utente catturati durante la fase precedente per disegnare, a partire dallo schema riconciliato, uno schema concettuale per il data mart. Il modello concettuale adottato è il *Dimensional Fact Model*, descritto ampiamente nel Capitolo 5, che prevede la creazione di uno *schema di fatto* per ciascun fatto di interesse evidenziato dall'utente. Uno schema di fatto è in grado di descrivere graficamente tutti i concetti del modello multidimensionale: fatti, misure, dimensioni e gerarchie; comprende inoltre un insieme di costrutti avanzati in grado di rappresentare accuratamente le diverse sfumature concettuali che caratterizzano gli scenari reali più complessi.

Il problema della costruzione di uno schema di fatto basato sul livello riconciliato e sulle specifiche utente viene dettagliatamente trattato nel Capitolo 6. Si mostrerà in particolare come sia possibile derivare automaticamente uno schema concettuale di massima dalle dipendenze funzionali espresse dallo schema riconciliato, ristrutturandolo e riadattandolo alla luce dei requisiti evidenziati dagli utenti.

L'insieme degli schemi di fatto così determinati costituisce lo schema concettuale per il data mart. Nel Capitolo 13, dedicato alla gestione della documentazione di progetto, si vedrà come in realtà sia conveniente associare agli schemi di fatto un insieme di glossari che ne completano la definizione, nonché uno *schema di data mart* in grado di riassumere efficacemente le correlazioni esistenti tra i diversi schemi di fatto.

### 2.3.4 Raffinamento del carico di lavoro e validazione dello schema concettuale

Al termine della fase di progettazione concettuale occorre raffinare il carico di lavoro, già espresso in forma preliminare, formulando ciascuna interrogazione direttamente sullo schema concettuale. Ciò permette di verificare che tutte le interrogazioni previste siano effettivamente esprimibili, e quindi in ultima analisi di validare lo schema concettuale prodotto al passo precedente.

Il Capitolo 7 introduce un semplice linguaggio per denotare interrogazioni multidimensionali sugli schemi di fatto. Esso discute inoltre il problema della determinazione del volume dati, che riveste fondamentale importanza per le successive fasi di progettazione logica e fisica.

### 2.3.5 Progettazione logica

Evidentemente, requisito essenziale per affrontare la progettazione logica è la scelta del modello logico di riferimento: si tratta cioè di scegliere tra un'implementazione ROLAP e una MOLAP. Il resto della trattazione sarà incentrato sulla prima, di gran lunga più diffusa sul mercato, e sarà di conseguenza riferito al modello relazionale. Una volta definito

lo schema logico di base secondo il cosiddetto *schema a stella*, la tecnica che maggiormente impatta sulle prestazioni è la *materializzazione delle viste*; può inoltre risultare vantaggioso applicare tecniche di *frammentazione verticale e orizzontale*. Entrambe le tecniche sono guidate dal carico di lavoro presunto per il data mart e dal volume dati stimato. Mentre i modelli logici utilizzati nei sistemi di data warehousing sono descritti nel Capitolo 8, i temi relativi alla progettazione logica da schemi di fatto sono trattati nel Capitolo 9.

### 2.3.6 Progettazione fisica

Il problema di maggior rilievo durante la progettazione fisica è la scelta degli indici da costruire per ottimizzare le prestazioni. In questa fase non è più sufficiente avere scelto il modello relazionale come piattaforma a livello logico, occorre anche riferirsi a uno specifico DBMS. Anche per la progettazione fisica, il carico di lavoro e il volume dati svolgono un ruolo sostanziale. Nel Capitolo 11 vengono descritte le principali tipologie di indici per le applicazioni di data warehousing, mentre nel Capitolo 12 si discutono alcune tecniche per la progettazione fisica in DBMS relazionali.

### 2.3.7 Progettazione dell'alimentazione

Durante questa fase vengono prese, di concerto con gli utenti e gli amministratori del sistema informatico, tutte le decisioni che riguardano il processo di alimentazione del livello riconciliato, se presente, e del data mart. Tra queste, particolarmente importante è la scelta dell'intervallo di aggiornamento periodico del data mart a partire dalle sue sorgenti. Gli aspetti principali della progettazione dell'alimentazione sono discussi nel Capitolo 10.

## 2.4 Il quadro metodologico

Essenzialmente, gli approcci alla progettazione di data mart presentati nella letteratura scientifica sono classificabili in due categorie (Winter, 2003):

- Gli approcci *guidati dai dati* (detti anche *supply-driven*) progettano il data mart a partire da una dettagliata analisi delle sorgenti operazionali. In questo caso, i requisiti utente impattano sul progetto guidando il progettista nella selezione delle porzioni di dati considerate rilevanti per il processo decisionale, e determinando la loro strutturazione secondo il modello multidimensionale.
- Gli approcci *guidati dai requisiti* (o *demand-driven*) iniziano determinando i requisiti informativi degli utenti del data mart. Il problema di come creare una mappatura tra questi requisiti e le sorgenti dati disponibili viene affrontato solo in seguito, attraverso l'implementazione di procedure ETL adatte.

Il prototipo di approccio del primo tipo risale al 1992, quando Inmon dichiarò per la prima volta che lo sviluppo di un DW deve essere guidato dai dati, in contrasto con lo sviluppo guidato dai requisiti adottato per i sistemi operazionali (Inmon, 1996). Altri approcci guidati dai dati sono stati proposti da Hüsemann e altri (2000), Golfarelli e altri

(1998) e Moody e Kortink (2000); la progettazione concettuale viene qui radicata nello schema delle sorgenti operazionali ed è effettuata a partire, rispettivamente, dall'identificazione delle misure, dalla selezione dei fatti e da una classificazione delle entità operazionali.

Il grande vantaggio degli approcci guidati dai dati è che la progettazione dell'ETL risulta notevolmente semplificata, poiché ciascuna informazione nel data mart è direttamente associata a uno o più attributi delle sorgenti. Infatti, come si vedrà nel Capitolo 6, uno schema concettuale di massima per il data mart può essere derivato algoritmicamente a partire dal livello dei dati riconciliati, ossia in funzione della struttura delle sorgenti. D'altro canto, questi approcci assegnano ai requisiti utente un ruolo secondario nel determinare i contenuti informativi per l'analisi, e danno al progettista un supporto limitato per l'identificazione di fatti, dimensioni e misure. Inoltre, essi risultano scarsamente applicabili nei casi in cui, durante la progettazione concettuale, non sia disponibile una conoscenza approfondita delle sorgenti operazionali.

Tra gli approcci guidati dai requisiti, ricordiamo innanzitutto quello proposto da Prakash e Gosain (2003), di tipo *goal-oriented* e basato sul modello goal-decisione-informazione. Un approccio *process-oriented* è invece proposto da Bruckner e altri (2001): vengono previste tre diverse prospettive a livelli crescenti di dettaglio, ciascuna abbinata a uno specifico *template* per i requisiti, e si raccomanda che la raccolta dei requisiti avvenga iterativamente e incrementalmente per mezzo di casi d'uso. Un terzo approccio si basa su una tecnica a più passi per l'analisi dei requisiti (Winter, 2003); vengono interposte due differenti fasi: analisi "as-is", mirata a descrivere i dati presenti sulle sorgenti, e analisi "to-be", che studia i fabbisogni informativi dei decisori incrociandoli con i dati operazionali.

Gli approcci guidati dai requisiti portano i *desiderata* degli utenti in primo piano, ma richiedono al progettista uno sforzo consistente durante il disegno dell'alimentazione. In questo caso i fatti, le misure e le gerarchie vengono desunte direttamente dalle specifiche dettate dagli utenti, e solo a posteriori si verifica che le informazioni richieste siano effettivamente disponibili nei database operazionali. Questa soluzione rischia di minare la fiducia del cliente verso il progettista e, più in generale, verso l'utilità del data mart. Infatti, durante la prima parte della progettazione l'utente vedrà prospettarsi una soluzione che risponde in pieno alle sue esigenze mentre, probabilmente, alcune delle analisi da lui richieste non saranno attuabili a causa della mancanza dei dati operazionali necessari. Questo problema è in realtà meno grave in presenza di un precedente sistema di reportistica, in quanto gli utenti saranno già abituati a richiedere solo le analisi per le quali sono disponibili tutti i dati necessari.

Un interessante confronto, basato su un caso di studio, tra i due tipi di approcci è stato presentato da List e altri (2002). Si giunge alla conclusione che le tecniche basate sui dati e sui requisiti sono in realtà complementari, e possono essere usate in parallelo per ottenerne una progettazione ottimale. Alla luce di quanto detto, non sorprende che si siano fatti strada in letteratura approcci di tipo misto, guidati dai requisiti e dai dati, in cui le sorgenti operazionali vengono comunque esplorate per plasmare le gerarchie, e al contempo i requisiti utente giocano un ruolo fondamentale nel restringere l'area di interesse per l'analisi e nella scelta di fatti, dimensioni e misure (Bonifati, 2001; Luján-Mora, 2003).

Il quadro metodologico per la progettazione di data mart che qui delineiamo, intorno al quale si incentrano tutti i prossimi capitoli, può essere adattato, a seconda delle circostanze e delle preferenze del progettista, al supporto di un approccio di tipo guidato dai dati, guidato dai requisiti, oppure misto. Nei paragrafi seguenti ipotizziamo tre realistici scenari di progettazione di data mart, abbinando ciascuno a un quadro metodologico di riferimento che specifica la collocazione delle diverse fasi di progettazione delineate nel Paragrafo 2.3.

### 2.4.1 Scenario 1: approccio guidato dai dati

Nell'approccio guidato dai dati si dà forte peso, nella modellazione del data mart, alle sorgenti dati. La fase di analisi e riconciliazione delle fonti dati viene allora svolta per prima: diventa così possibile condurre l'analisi dei requisiti e disegnare lo schema concettuale del data mart senza perdere di vista la natura dei dati effettivamente disponibili, in modo da produrre specifiche con essi compatibili.

Inoltre, in questo approccio la fase di analisi dei requisiti utente risulta strettamente legata alla fase di progettazione concettuale. Da un lato ciò riflette il fatto che certe attività condotte durante l'analisi dei requisiti, come per esempio la scelta delle dimensioni, trovano il loro completamento durante la progettazione concettuale, quando si concretizzano in operazioni che determinano la struttura dello schema concettuale. Soprattutto, però, il legame tra le due fasi rispecchia il fatto che, avendo operato la scelta metodologica di basare la progettazione concettuale sul livello dei dati operazionali, diventa in pratica possibile e conveniente effettuare l'analisi dei requisiti *contemporaneamente* alla progettazione concettuale. Infatti, nel Capitolo 6 dedicato alla progettazione concettuale si vedrà come uno schema concettuale di massima per il data mart sia derivabile dallo schema riconciliato applicando un semplice algoritmo; alcune decisioni di progetto (per esempio quelle riguardanti la strutturazione delle gerarchie) devono sì esprimere i desiderata dell'utente, ma al contempo devono essere improniate su questo schema di massima. Come conseguenza di ciò, il quadro metodologico proposto in questo testo prevede che, nel caso di adozione di un approccio guidato dai dati, la fase di analisi dei requisiti utente venga svolta in maniera informale, appoggiandosi sulla definizione di *glossari* dei requisiti piuttosto che su schemi formali (si veda a questo proposito il Paragrafo 4.2).

A valle della progettazione concettuale si collocherà il raffinamento del carico di lavoro, cui seguirà la progettazione logica. La progettazione dell'alimentazione e quella fisica potranno essere svolte in parallelo. Prerequisiti ragionevoli per adottare con successo l'approccio guidato dai dati sono i seguenti:

1. deve essere disponibile preliminarmente, oppure ottenibile con costi e tempi contenuti, una *conoscenza approfondita* delle sorgenti da cui il data mart si alimenterà;
2. gli schemi delle sorgenti devono mostrare un buon grado di *normalizzazione*;
3. la *complessità* degli schemi delle sorgenti non deve essere eccessiva.

In pratica, quando l'architettura prescelta prevede l'adozione di un livello riconciliato questi requisiti sono largamente soddisfatti: infatti la normalizzazione e la conoscenza approfondita sono garantite a valle del processo di riconciliazione. Lo stesso vale, grazie allo svolgimento di un'attenta attività di ricognizione, nel caso non infrequente in cui la sorgente si riduca a un singolo database, ben progettato e di dimensioni limitate.

L'esperienza di progettazione ha mostrato che, qualora applicabile, l'approccio guidato dai dati risulta preferibile agli altri poiché permette di raggiungere i risultati prefissati in tempi estremamente contenuti: infatti, da un lato la progettazione concettuale risulta, come già detto, parzialmente automatizzabile; dall'altro, viene semplificata la progettazione dell'alimentazione in quanto la mappatura tra le sorgenti e il data mart è ottenuta "a costo zero" dalla progettazione concettuale.

### 2.4.2 Scenario 2: approccio guidato dai requisiti

Secondo questo approccio, il fattore trainante della progettazione sono i requisiti espressi dall'utente finale. Si suggerisce allora di adottare, per la specifica dei requisiti, un formalismo adatto; nel Paragrafo 4.3 ne viene proposto uno particolarmente espressivo. I requisiti catturati verranno poi trasformati in uno schema concettuale come mostrato nel Paragrafo 6.5; in questo caso la fase di progettazione concettuale risulta particolarmente delicata, poiché non è possibile avvalersi della conoscenza dettagliata dei dati operazionali per desumere le relazioni strutturali tra le diverse informazioni rappresentate nel data mart.

Dopo la progettazione concettuale, le altre fasi si susseguono come nello scenario precedente. L'aver trascurato l'analisi delle sorgenti porterà però un forte peso a gravare sul progetto dell'alimentazione, in quanto si dovranno ricavare manualmente tutti i legami tra i dati nel data mart e le loro fonti operazionali.

Questo approccio è, a nostro modo di vedere, il più difficilmente perseguitibile dei tre. Ciononostante, esso costituisce l'unica alternativa nei casi in cui non sia fattibile a priori un'analisi approfondita delle sorgenti (per esempio quando il data mart viene alimentato da un sistema ERP, il cui schema logico è estremamente vasto e di difficile interpretazione), oppure qualora le sorgenti siano rappresentate da sistemi legacy di tale complessità da sconsigliarne la ricognizione e la normalizzazione, e di conseguenza non si ritenga opportuno creare il livello riconciliato.

### 2.4.3 Scenario 3: approccio misto

*In medio stat virtus:* è fin troppo facile affermare che un approccio misto costituisce il più ragionevole compromesso di progettazione. L'analisi dei requisiti e l'analisi delle sorgenti vengono condotte in parallelo: la prima porta a definire in modo formale requisiti di progetto, mentre la seconda conduce al disegno del livello riconciliato. La progettazione concettuale viene ancora condotta in modo semiautomatico, come nello scenario 1, ma usando i requisiti per limitarne a priori la complessità. Nel paragrafo 6.4 si vedrà nel dettaglio come ciò sia possibile.

Anche in questo caso, la successione delle rimanenti fasi è invariata. Come nello scenario 1, la complessità della progettazione dell'alimentazione è contenuta.

L'approccio misto, che unisce le facilitazioni di quello guidato dai dati alle garanzie di quello guidato dai requisiti, è raccomandabile quando l'estensione e la complessità del livello riconciliato sono notevoli. Infatti, il costo di una più attenta e formale analisi dei requisiti viene compensato dallo svelimento della progettazione concettuale.



## Analisi e riconciliazione delle fonti dati

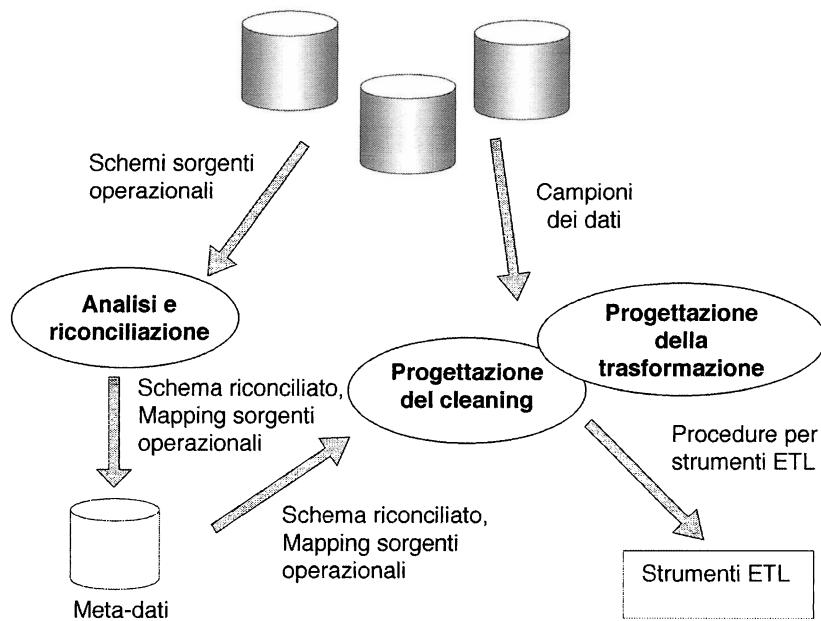
---

Come già più volte ripetuto, i dati contenuti nel DW sono ricavati da un insieme di sorgenti che si possono differenziare sia per la tecnologia che le gestisce (DBMS, fogli elettronici, file system ecc.), sia per il modello attraverso il quale rappresentano la realtà aziendale (modello relazionale, modello a oggetti, *flat file* ecc.), sia infine per la semantica che esprimono; nel seguito, con il termine *schema locale di una sorgente* indicheremo lo schema concettuale che descrive la semantica dei dati immagazzinati nella sorgente prescindendo dal modello e dalla tecnologia utilizzata. L'attenzione è posta cioè sulla rappresentazione del dominio applicativo, che è ugualmente possibile (anche se con capacità espressive ben differenti) utilizzando una base di dati relazionale, un foglio elettronico o un semplice file: nel primo caso lo schema è codificato dall'insieme delle relazioni e dei vincoli che le collegano, nel secondo dai legami tra le diverse celle del foglio, infine nel terzo lo schema è ricavabile analizzando il tracciato dei record del file.

Le varie sorgenti operazionali possono essere fortemente correlate o completamente indipendenti e i domini che esse descrivono possono essere disgiunti oppure sovrapposti. Compito primario del progettista durante la fase di analisi delle fonti dati è acquisire una conoscenza quanto più possibile approfondita che gli consenta di individuare i concetti rilevanti ai fini del DW e le sorgenti dati più adatte ad alimentarlo.

L'analisi delle fonti operazionali non è di per sé sufficiente a creare i presupposti per le fasi successive della progettazione in quanto, nella stragrande maggioranza dei casi, essa evidenzierà un insieme di inconsistenze e di errori che devono forzatamente essere risolti prima di procedere oltre. Infatti, uno dei principi fondanti del data warehousing è il concetto di *dato integrato* che permette di derivare informazioni consistenti e prive di errori. Il raggiungimento di questo ambizioso risultato necessita di un processo di *riconciliazione* che comporta *integrazione, pulizia e trasformazione* dei dati e può richiedere un elevato dispendio di tempo e di risorse.

Si noti che la fase di integrazione è incentrata sulla componente intensionale delle sorgenti operazionali, ossia riguarda la consistenza degli schemi che le descrivono. Al contrario, pulizia e trasformazione dei dati operano a livello estensionale, ossia coinvolgono direttamente i dati veri e propri.



**Figura 3.1** Le tre fasi per la progettazione del livello riconciliato a partire dal livello operazionale.

In Figura 3.1 sono rappresentate (mediante degli ovali) le fasi che permettono di progettare il livello riconciliato: come si può notare il processo di analisi e riconciliazione riceve in ingresso gli schemi delle sorgenti e produce un insieme di meta-dati che modellano lo schema riconciliato e le corrispondenze (*mapping*) tra gli elementi di quest'ultimo e quelli del sistema operazionale. Utilizzando le informazioni contenute nei meta-dati e ricorrendo alle sorgenti operazionali per valutare la qualità delle sorgenti informative, attraverso i due rimanenti processi vengono progettate le procedure ETL che saranno utilizzate per alimentare il DW. Sebbene le tre fasi siano tra loro strettamente legate, in questo capitolo ci concentreremo sull'analisi e riconciliazione che sono indispensabili per definire il livello riconciliato, punto di partenza della progettazione concettuale. Le rimanenti due fasi verranno descritte nel Capitolo 10, nel contesto più generale del progetto dell'alimentazione.

Come modello architettoniale di riferimento utilizzeremo quello a tre livelli proposto nel Paragrafo 1.3.3 in cui si suppone che il livello di dati riconciliato, ossia una versione integrata del sistema operazionale, esista e sia materializzato. Questo strato rappresenta il punto di arrivo della fase di integrazione e di pulizia dei dati; a questo livello sia lo schema sia i dati sono consistenti e privi di errori. È importante sottolineare che lo schema riconciliato deve essere creato, almeno virtualmente, anche nelle soluzioni a due livelli poiché esso dimostra che il progettista ha definitivamente acquisito una precisa conoscenza sul sistema operazionale, nozione indispensabile per realizzare una corretta alimentazione del DW che, in questo caso, verrà effettuata direttamente a partire dalle sorgenti dati.

Si ritiene preferibile la soluzione a tre livelli poiché l'alimentazione diretta del DW è un compito troppo complesso per essere eseguito in modo atomico: la presenza di uno stadio intermedio facilita il compito del team di progettazione, semplifica le procedure di alimentazione e pulizia dei dati ed evita che lo schema concettuale integrato delle sorgenti rimanga criptato all'interno delle logiche delle procedure di ETL.

Esaminiamo ora più da vicino la fase di analisi e riconciliazione individuando le attività che la compongono. È bene precisare che il quadro generale è abbastanza complesso, a causa dell'elevato numero di combinazioni che si possono presentare in funzione delle variabili in gioco.

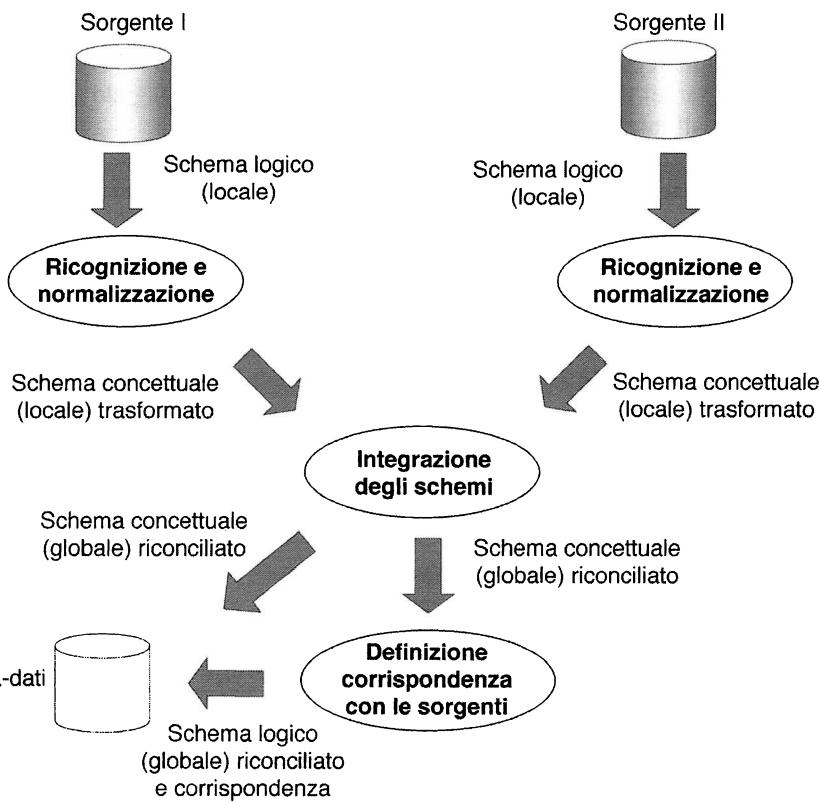
Per ogni sorgente può essere noto lo schema concettuale oppure solo quello logico, entrambi espressi nei formalismi più disparati; vi può essere una sola oppure più sorgenti dati e i relativi schemi possono essere tra loro consistenti oppure fortemente eterogenei e inconsistenti. Per uscire indenni da una situazione di tale complessità è quindi necessario operare con diligenza e in base a una scaletta di operazioni ben definita.

La Figura 3.2 dettaglia la fase di analisi e riconciliazione di Figura 3.1 delineando i passi necessari nel caso più complesso che prevede più sorgenti inconsistenti di cui è noto il solo schema logico (magari codificato con formalismi diversi). In questo caso si procederà dapprima alla *ricognizione e normalizzazione* dei diversi schemi locali producendo come risultato un insieme di schemi concettuali, localmente consistenti e completi; quindi si procederà alla fase di *integrazione* producendo uno schema concettuale globalmente consistente. Infine, a partire da quest'ultimo si effettuerà la progettazione logica dello schema riconciliato per poi definirne la corrispondenza con gli schemi logici delle sorgenti.

Sebbene il collegamento tra i dati sia realizzato a livello logico, l'utilizzo di formalismi di livello concettuale è fortemente consigliato poiché garantisce una maggiore chiarezza nell'interpretazione dei concetti e perché permette di produrre una migliore documentazione.

Nel caso in cui si presentino situazioni più semplici sarà sufficiente eliminare le fasi non richieste. Per esempio, in presenza di una sola sorgente dati sarà sufficiente effettuare la ricognizione e la normalizzazione dello schema locale. Quest'ultimo coinciderà in questo caso con lo schema globale riconciliato e potrà quindi essere utilizzato come ingresso per la successiva fase di definizione della corrispondenza con le sorgenti.

Gli schemi concettuali delle sorgenti rappresentano senz'altro il risultato principale dell'analisi e devono essere espressi mediante lo stesso formalismo per tutte le sorgenti. Ove assenti, essi devono essere ricavati per *reverse engineering* possibilmente con l'ausilio di appositi strumenti quali, per esempio, ERWIN ERX di LogicWorks e Groundworks di Cayenne per il formalismo Entity/Relationship, Rational Rose per UML. Il formalismo maggiormente utilizzato nei casi reali è senz'altro quello Entity/Relationship, anche se gli studi accademici propugnano l'utilizzo di modelli dei dati più ricchi e in grado di rappresentare informazioni relative alle dipendenze tra i dati, valori nulli e altre proprietà semantiche che aumentano il livello di conoscenza sulle sorgenti dati.



**Figura 3.2** Passi progettuali che compongono la fase di analisi e riconciliazione di due sorgenti operazionali.

### 3.1 Ricognizione e normalizzazione degli schemi

Indipendentemente da eventuali operazioni di integrazione, prima di procedere alla fase di progettazione concettuale il progettista deve acquisire un'approfondita conoscenza delle sorgenti operazionali attraverso attività di:

- *ricognizione*, che consiste in un esame approfondito degli schemi locali mirato alla piena comprensione del dominio applicativo;
- *normalizzazione*, il cui obiettivo è correggere gli schemi locali al fine di modellare in modo più accurato il dominio applicativo<sup>1</sup>.

<sup>1</sup> Usiamo il termine *normalizzazione* in questo contesto, anche se non necessariamente riferendolo a schemi relazionali, poiché questa attività porta in genere ad arricchire lo schema locale di dipendenze funzionali precedentemente non rappresentate.

La fase di ricognizione e normalizzazione deve essere svolta anche qualora sia presente una sola sorgente dati e per questo motivo non può essere considerata come parte dell'integrazione. Nel caso in cui esistano più sorgenti, l'operazione dovrà essere ripetuta per ogni singolo schema locale.

Durante questa fase di analisi il progettista, confrontandosi con gli esperti del dominio applicativo (amministratori del sistema, manager ecc.), deve verificare la completezza degli schemi locali sforzandosi di individuare eventuali correlazioni involontariamente omesse. Esempi tipici sono l'esplicitazione di dipendenze funzionali precedentemente tralasciate e l'individuazione di nuove associazioni tra le entità. Queste operazioni possono portare a una trasformazione dello schema locale al fine di conformarlo alle peculiari esigenze del data warehousing.

Le trasformazioni apportate allo schema non devono introdurre nuovi concetti, bensì rendere esplicativi tutti quelli ricavabili dai dati memorizzati nelle sorgenti operazionali. Si consideri l'esempio di Figura 3.3, in cui sono rappresentate due possibili trasformazioni dello schema Entity/Relationship che modella i prodotti trattati da una ipotetica azienda. La prima trasformazione è lecita in questo ambito poiché la dipendenza funzionale codiceProd → codiceCategoria è vera, anche se non rappresentata nello schema, e la cardinalità dell'associazione appartiene a è verificabile analizzando i dati della sorgente. Al contrario la seconda trasformazione non è lecita poiché, sebbene possa esistere una suddivisione in sottocategorie, la sorgente operazionale non contiene le informazioni necessarie per estrarla. Volendo comunque evidenziare questa classificazione nel data mart, si dovrà individuare la sorgente dati contenente le informazioni necessarie e procedere conseguentemente all'integrazione.

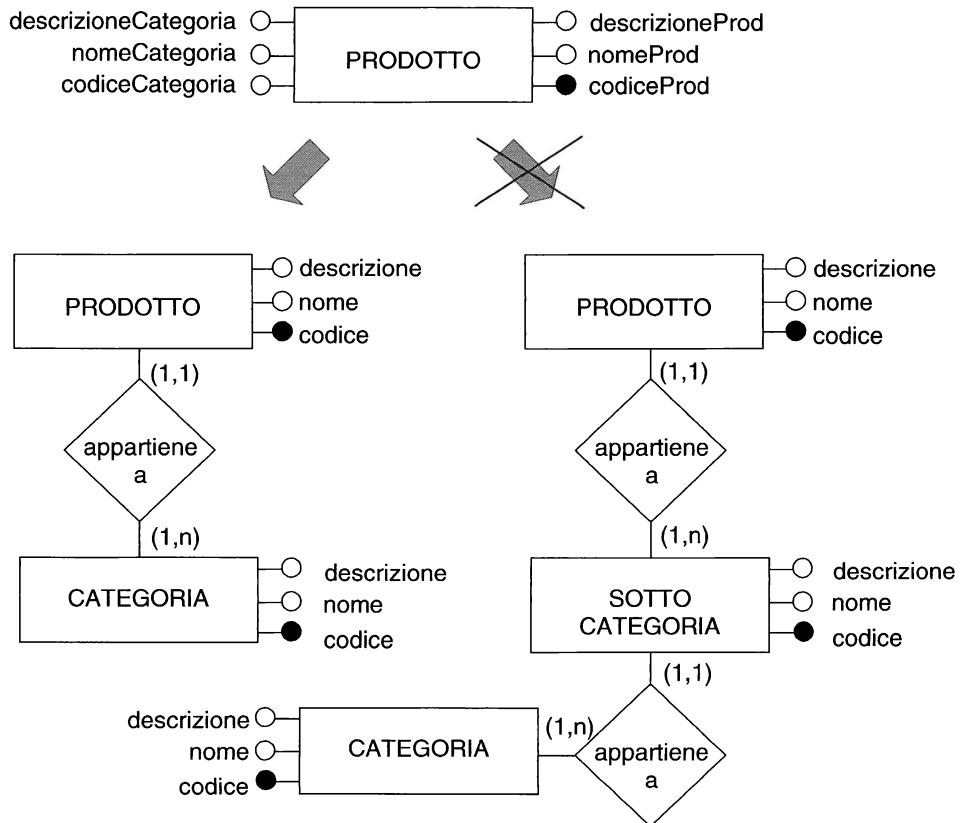
Oltre all'applicazione delle necessarie trasformazioni, il progettista deve anche individuare eventuali porzioni degli schemi locali non utili al data mart. Per esempio, nell'ambito del data mart delle vendite non sono interessanti le informazioni di carattere amministrativo relative al personale commerciale, oppure le informazioni riguardanti i costi di gestione dei negozi appartenenti alla catena di vendita.

## 3.2 Il problema dell'integrazione

Il problema dell'integrazione è oggetto di studio da ormai vent'anni, nel corso dei quali sono stati evidenziati i problemi legati all'integrazione di diverse classi di sorgenti dati quali per esempio le basi di dati relazionali, a oggetti e le sorgenti semi-strutturate (Jarke, 2000), sono state proposte metodologie per meglio affrontare il problema (Batini, 1986) e sono stati definiti formalismi in grado di codificare le informazioni rilevanti (Blanco, 1994; Calvanese, 1998; Garcia-Solaco, 1995).

Attualmente l'attività di ricerca è concentrata sull'automazione del processo di integrazione. Nonostante i notevoli passi in avanti compiuti negli ultimi anni, i sistemi realizzati (TSIMMIS (Chawathe, 1994), Garlic (Carey, 1995), Squirrel (Hull, 1996) e MOMIS (Beneventano, 2000)) devono ancora considerarsi dei prototipi di ricerca piuttosto che vere e proprie applicazioni di tipo commerciale. D'altronde, le tecniche concettuali che essi impiegano sono difficilmente utilizzabili "manualmente", ossia senza

il supporto di strumenti automatici. Per questo motivo la trattazione che presentiamo sarà ispirata ai più maturi canoni dall'ormai ventennale esperienza nell'ambito delle basi di dati eterogenee, che meglio si prestano a un'applicazione in ambito aziendale.



**Figura 3.3** Trasformazioni lecite e illecite durante la fase di ricognizione e normalizzazione di una sorgente.

L'integrazione di un insieme di sorgenti dati eterogenee (basi di dati relazionali, file dati, sorgenti *legacy*) consiste nell'individuazione delle corrispondenze tra i concetti rappresentati negli schemi locali e nella risoluzione dei conflitti evidenziati finalizzate alla creazione di un unico schema globale i cui elementi possano essere correlati con i corrispondenti elementi degli schemi locali (*mapping*).

Se le diverse sorgenti dati modellassero porzioni indipendenti e distinte del mondo reale, il problema dell'integrazione non sussisterebbe e il compito del progettista sarebbe notevolmente semplificato: l'analisi dei singoli schemi locali sarebbe sufficiente al fine di comprendere quali porzioni del mondo aziendale sono modellate dal sistema informativo.

Sfortunatamente ciò non può avvenire in pratica poiché ogni azienda è un universo coeso in cui le diverse unità organizzative partecipano a processi comuni che condividono tutti o parte degli attori. Per esempio il concetto di prodotto esiste sia nei processi legati alla logistica di magazzino sia in quelli legati alle attività amministrative; questi ultimi completeranno anche il concetto di dipendente, probabilmente in modo diverso rispetto alla modalità con cui esso deve essere percepito nei processi legati alla gestione di progetti.

A questa diversità percettiva si aggiunge il fatto che il processo di informatizzazione di un sistema informativo<sup>2</sup> è di tipo incrementale ed evolutivo e quindi il suo risultato è un mosaico composto da più frammenti, ognuno dei quali utilizza la tecnologia in voga al momento della sua informatizzazione e modella la realtà esistente in quel momento cercando di essere, dove necessario e per quanto possibile, compatibile con i frammenti già esistenti. Se a ciò si aggiungono gli errori di modellazione e di implementazione da cui nessun sistema complesso è esente, risulta chiaro il livello di difficoltà del problema.

La fase di integrazione non si deve limitare a evidenziare le differenze di rappresentazione dei concetti comuni a più schemi locali, ma deve anche identificare l'insieme di concetti distinti e memorizzati in schemi differenti che sono correlati attraverso proprietà semantiche (*proprietà inter-schema*). Queste proprietà, non esplicitamente modellate negli schemi locali, possono essere identificate solo a fronte di un'analisi congiunta.

Per poter ragionare sui concetti espressi negli schemi delle diverse sorgenti dati è necessario utilizzare un unico formalismo in modo da fissare i costrutti utilizzabili e la potenza espressiva. A tal fine, dove siano presenti schemi espressi in formalismi diversi (Entity/Relationship, UML, relazionale, a oggetti, DTD ecc.) è necessario fare una scelta riscrivendo poi nel formalismo prescelto gli schemi coinvolti nel processo di integrazione.

Il formalismo prescelto dovrebbe essere quello che garantisce il maggior potere espressivo, in questo modo non potranno verificarsi perdite di informazioni durante il processo di traduzione. A torto molti progettisti preferiscono adottare il formalismo comune alla maggior parte degli schemi, oppure quello in cui si sentono più esperti.

Nel seguito di questo paragrafo elencheremo, utilizzando per maggior chiarezza come formalismo di riferimento l'Entity/Relationship, le principali cause dei problemi che devono essere risolti durante l'integrazione; ovviamente tutte le considerazioni fatte sono indipendenti dal particolare formalismo adottato.

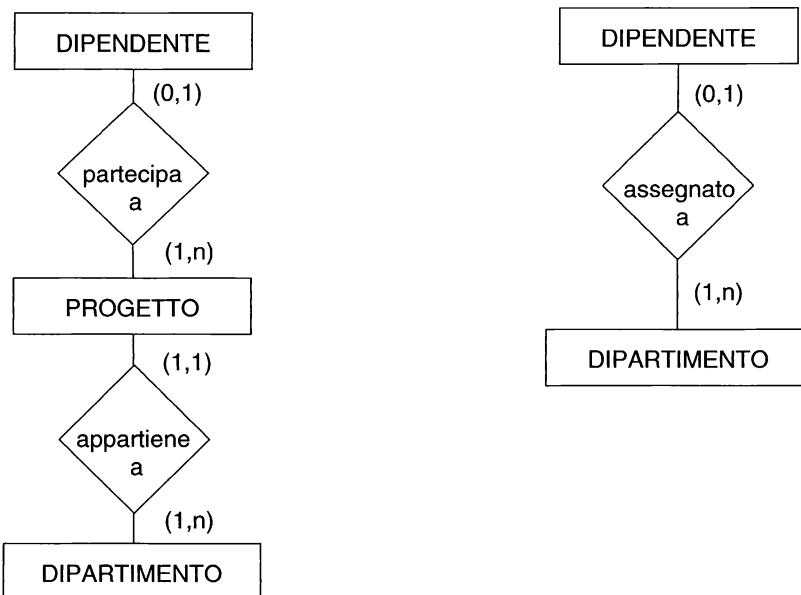
### **3.2.1 Diversità di prospettiva**

Il punto di vista rispetto al quale diversi gruppi di utenti vedono uno stesso oggetto del dominio applicativo può differenziarsi notevolmente in base agli aspetti rilevanti per la funzione a cui essi sono preposti. Consideriamo per esempio il caso di Figura 3.4, in cui viene modellata l'appartenenza di un dipendente a un dipartimento. Nella modellazione di

---

<sup>2</sup> Si sottolinea che il concetto di *sistema informativo* è indipendente da quello di *sistema informatico*: col primo termine si indica l'insieme delle informazioni gestite ai fini del controllo dell'attività aziendale, con il secondo si intende la porzione di sistema informativo gestita tramite supporto informatico.

sinistra, che potrebbe essere adatta, per esempio, per il database utilizzato nella gestione dell'organigramma aziendale, compare il concetto di progetto in cui il dipendente è inserito, mentre nella modellazione di destra tale concetto scompare in quanto non rilevante.



**Figura 3.4** Due diverse prospettive rispetto a cui modellare l'assegnamento dei dipendenti ai dipartimenti (Batini, 1986).

### 3.2.2 Equivalenza dei costrutti del modello

Tipicamente, i formalismi di modellazione permettono di rappresentare uno stesso concetto utilizzando combinazioni diverse dei costrutti a disposizione. Si considerino per esempio i due schemi riportati in Figura 3.5, che modellano il rapporto tra libro e casa editrice: nello schema di destra si è utilizzato un semplice attributo, mentre in quello a sinistra si è voluta dare maggiore enfasi al concetto creando l'entità CASA EDITRICE.

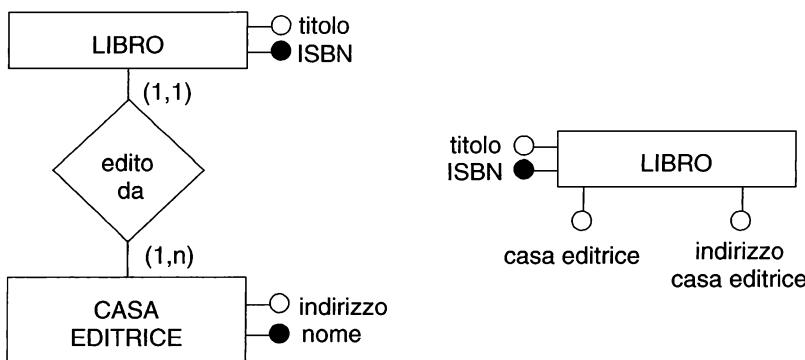
Si noti che in questo caso la diversità è puramente di tipo sintattico: le due modellazioni sono perfettamente equivalenti. Diversamente, nel caso presentato nel paragrafo precedente veniva introdotto un nuovo concetto, il progetto, che faceva variare anche il potere espressivo dello schema.

### 3.2.3 Incompatibilità delle specifiche

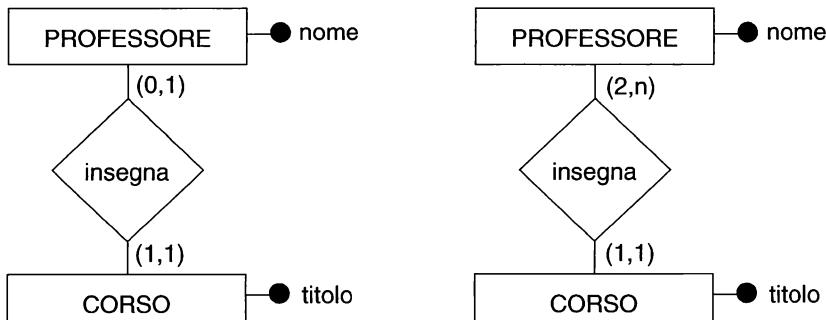
L'incompatibilità delle specifiche indica che schemi differenti che modellano una stessa porzione del dominio applicativo racchiudono concetti diversi, in contrasto tra loro. Tale

diversità deriva normalmente da errate scelte progettuali che possono coinvolgere per esempio la scelta dei nomi, dei tipi di dati e dei vincoli di integrità. Per esempio le due modellazioni presentate in Figura 3.6 potrebbero essere frutto di un errore, poiché è poco realistico sia che un professore non possa tenere più corsi contemporaneamente (a sinistra in figura), sia che ne debba tenere forzatamente più di uno (a destra).

Oltre a ciò si deve considerare che, come già detto, la realtà aziendale evolve nel tempo e quindi le assunzioni fatte nel passato potrebbero non essere più vere oggi. Per esempio, un programma per la gestione degli stipendi scritto prima del passaggio all'euro poteva utilizzare attributi di tipo intero per memorizzare gli importi, mentre oggi si rende necessario l'utilizzo di attributi di tipo *floating point*.



**Figura 3.5** Due rappresentazioni equivalenti per modellare il legame tra i libri e le case editrici.



**Figura 3.6** Due improbabili modellazioni dell'associazione tra professori universitari e corsi.

### 3.2.4 Concetti comuni

A causa dei problemi sin qui evidenziati è necessario definire il tipo di relazione semantica esistente tra concetti comuni modellati diversamente in schemi distinti. Quattro sono le possibili relazioni esistenti tra due distinte rappresentazioni  $R_1$  e  $R_2$  di uno stesso concetto:

- *Identità*: questa situazione si verifica quando vengono utilizzati gli stessi costrutti, il concetto è modellato dallo stesso punto di vista e non vengono commessi errori di specifica; in altre parole quando  $R_1$  e  $R_2$  coincidono.
- *Equivalenza*: questa situazione si verifica quando  $R_1$  e  $R_2$  non sono esattamente le stesse poiché sono stati utilizzati costrutti diversi (ma equivalenti) e non sussistono errori di specifica o diversità di percezione. Tra le varie definizioni di equivalenza riportiamo quella proposta da Rissanen (1977):

**Equivalenza tra schemi.** Due schemi  $R_1$  e  $R_2$  sono equivalenti se le loro istanze possono essere messe in corrispondenza uno-a-uno.

Si noti come l'equivalenza tra schemi implichi che a livello estensionale esistano sempre due insiemi di dati, diversi ma equivalenti, che memorizzano le stesse informazioni. Per esempio, gli schemi presentati in Figura 3.5 sono tra loro equivalenti; due possibili istanze degli schemi che soddisfano la definizione precedente sono riportate nella Figura 3.7.

- *Comparabilità*: questa situazione si verifica quando  $R_1$  e  $R_2$  non sono né identici né equivalenti, ma i costrutti utilizzati e i punti di vista dei progettisti non sono in contrasto tra loro. Per esempio gli schemi di Figura 3.4 sono tra loro compatibili: infatti, pur non essendo equivalenti dato che quello di sinistra è più espressivo (modella anche la partecipazione dei dipendenti ai progetti), non sono in contrasto poiché, tramite la navigazione dell'entità PROGETTO, si può evincere immediatamente che un dipendente è assegnato a 0 o 1 dipartimento e che viceversa a un dipartimento sono assegnati uno o più dipendenti.
- *Incompatibilità*: questa situazione si verifica quando  $R_1$  e  $R_2$  sono in contrasto a causa dell'incoerenza nelle specifiche, in altre parole quando la realtà modellata da  $R_1$  nega la realtà modellata da  $R_2$ . Gli schemi di Figura 3.6 sono tra loro incompatibili poiché sono contraddittori gli scenari prospettati per i professori: in un caso un professore non può tenere più di un corso, nell'altro deve tenerne almeno due.

A esclusione della situazione di identità, i casi precedenti determinano dei conflitti la cui soluzione rappresenta la componente principale nella fase di integrazione.

**Conflitto.** Si verifica un *conflitto* tra due rappresentazioni  $R_1$  e  $R_2$  di uno stesso concetto ogniqualvolta le due rappresentazioni non sono identiche.

**LIBRO**

ISBN	titolo	casa editrice
123445	Il DFM	McGraw-Hill
4354543	Mi Sembra Logico	LettiTutti
4566454	La Giusta Misura	NonSoloLibri
.....	.....	.....

**CASA EDITRICE**

nome	indirizzo
McGraw-Hill	Via Ripamonti, 89
LettiTutti	Via dei Brutti, 21
NonSoloLibri	Via Tumedei, 57
.....	.....

**LIBRO**

ISBN	titolo	nome c.e.	indirizzo c.e.
123445	Il DFM	McGraw-Hill	Via Ripamonti, 89
4354543	Mi Sembra Logico	LettiTutti	Via dei Brutti, 21
4566454	La Giusta Misura	NonSoloLibri	Via Tumedei, 57
.....	.....	.....	.....

**Figura 3.7** Due istanze per gli schemi equivalenti in Figura 3.5.

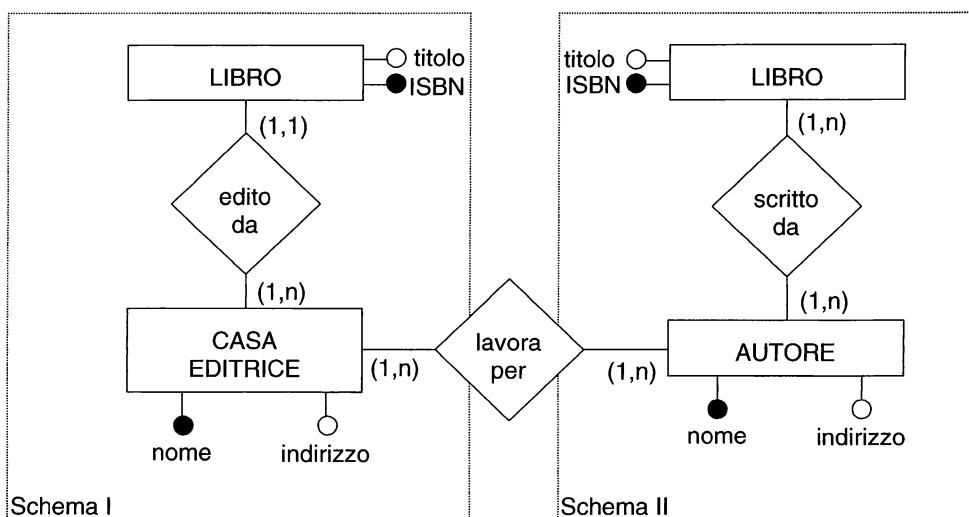
### 3.2.5 Concetti correlati

A seguito dell'integrazione, molti concetti diversi ma correlati verranno a trovarsi nello stesso schema dando vita a nuove relazioni che non erano percepibili in precedenza. Tali relazioni sono dette *proprietà inter-schema* e devono essere identificate e rappresentate esplicitamente. Per esempio, in Figura 3.8 è rappresentata l'associazione tra CASA EDITRICE e AUTORE che non era evidenziabile nei due schemi sorgente.

## 3.3 Le fasi dell'integrazione

Risolvere i problemi fin qui elencati ed evidenziare le proprietà che emergono a seguito dell'integrazione degli schemi locali richiede un complesso insieme di operazioni, per la cui corretta gestione è necessario adottare una metodologia. Le molte metodologie proposte in letteratura concordano sulla sequenza di passi che devono essere svolti e che possono essere così sintetizzati (Batini, 1986):

- Preintegrazione
- Comparazione degli schemi
- Allineamento degli schemi
- Fusione e ristrutturazione degli schemi



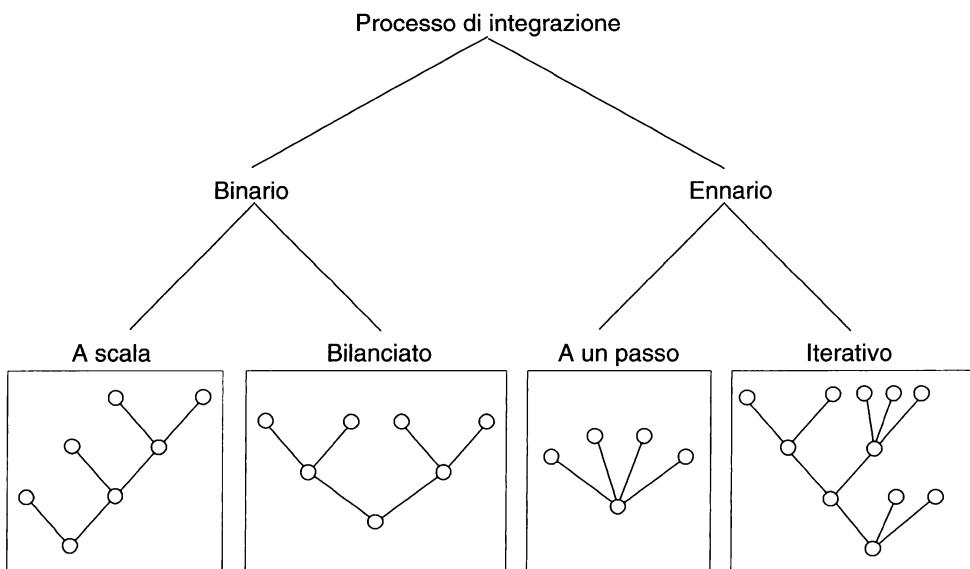
**Figura 3.8** Una proprietà inter-schema tra le entità CASA EDITRICE e AUTORE.

### 3.3.1 Preintegrazione

Durante questa fase viene svolta l'analisi vera e propria delle sorgenti dati, che porta a definire la politica generale dell'integrazione. Le principali decisioni da prendere riguardano:

- *Le porzioni degli schemi che dovranno essere integrate:* non tutti i dati operazionali sono utili ai fini decisionali e quindi alcuni di essi potranno essere scartati a priori.
- *La strategia di integrazione:* è necessario decidere in che ordine si procederà all'integrazione degli schemi. In Figura 3.9 sono rappresentate alcune possibili soluzioni: la principale dicotomia è tra le *tecniche ennarie*, nelle quali il processo di integrazione considera più di due schemi contemporaneamente, e le *tecniche binarie*, in cui il processo di integrazione considera sempre coppie di schemi. Queste ultime sono dette tecniche *a scala* quando i nuovi schemi sono integrati allo schema temporaneo determinato fino a quel momento.

La maggior parte delle metodologie proposte in letteratura predilige l'approccio binario, che rende ogni passo di integrazione più semplice grazie al ridotto numero di concetti coinvolti contemporaneamente. Al contrario i sostenitori delle tecniche ennarie asseriscono che, integrando contemporaneamente più schemi, ogni concetto verrà analizzato avendo a disposizione contemporaneamente tutte le informazioni che lo caratterizzano; inoltre con le tecniche ennarie si diminuisce il numero totale di comparazioni tra concetti poiché ognuno di essi viene analizzato una sola volta.



**Figura 3.9** Possibili strategie per l'integrazione degli schemi.

Utilizzando una tecnica binaria a scala è possibile definire l'ordine con cui i diversi schemi sorgenti saranno esaminati e aggiunti allo schema riconciliato: normalmente si preferisce iniziare dalle sorgenti che costituiscono il “cuore” del sistema informativo e la cui integrazione definirà lo scheletro dello schema riconciliato, partendo dal quale si passerà all’integrazione delle porzioni di schemi “secondari” che verranno aggiunte all’impalcatura già definita. Utilizzando questa soluzione, durante la soluzione dei conflitti si dovrà privilegiare la versione dello schema parzialmente riconciliato che già si conforma agli schemi sorgenti più importanti.

La scelta della strategia da utilizzare dipende fortemente dalla complessità e dal numero degli schemi: le soluzioni ennarie possono essere vantaggiose quando la complessità dell’intero processo è limitata. In generale, si consiglia l’utilizzo di tecniche binarie a scala poiché semplificano l’integrazione e il loro principio di base è in linea con quello, proposto in questo testo, secondo il quale la progettazione di un DW debba avvenire con modalità bottom-up costruendo iterativamente nuovi data mart. Alla creazione di ogni nuovo data mart corrisponde una nuova fase di integrazione che porta all’analisi di nuove sorgenti operazionali.

Sia nel decidere la porzione delle sorgenti da integrare, sia nello scegliere la strategia di integrazione è necessario considerare da quale data mart si inizierà la progettazione. Ricordiamo che, sebbene l’analisi dettagliata dei requisiti utente sia successiva all’analisi del livello operazionale, progettista e committente devono avere già sommariamente concordato quali siano le priorità.

Durante l'analisi delle sorgenti operazionali il progettista si curerà anche di verificare la qualità dei dati contenuti all'interno delle sorgenti (percentuale di errori nei dati, percentuale di dati mancanti, presenza di meccanismi di verifica della consistenza nei database ecc.), poiché anch'essa può incidere sulle scelte discusse in precedenza.

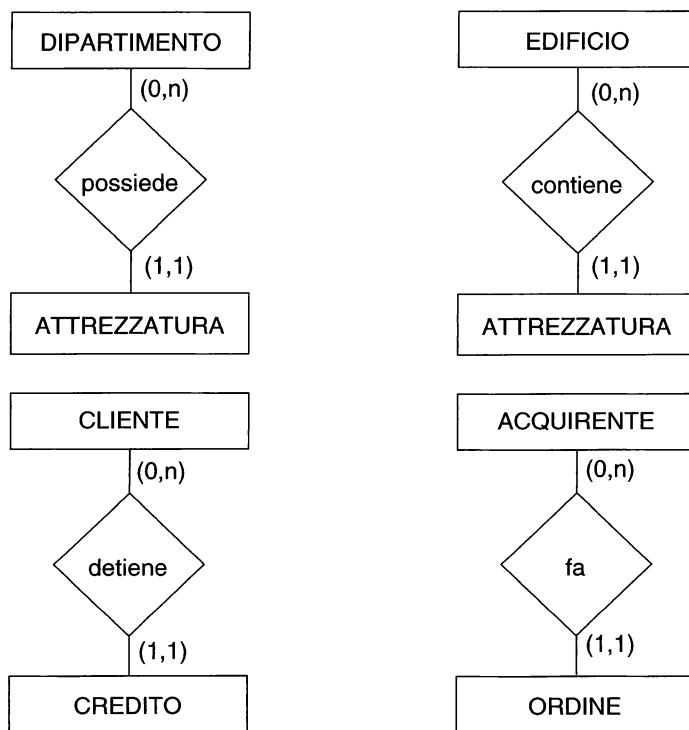
### 3.3.2 Comparazione degli schemi

Questa fase consiste in un'analisi comparativa dei diversi schemi che mira a identificare le correlazioni e i conflitti tra i concetti in essi espressi. La sua efficacia dipende dalle conoscenze acquisite dal progettista rispetto al dominio applicativo e alla struttura delle sorgenti informative; in ogni caso, per quanto approfondita possa essere stata la fase di analisi, il progettista non può prescindere dal collaborare con gli amministratori e gli utenti del sistema informativo operazionale al fine di dissipare i propri dubbi sui concetti modellati. I tipi di conflitti che possono essere evidenziati, facilmente riconoscibili ai problemi presentati nel Paragrafo 3.2, ricadono nelle seguenti categorie (Batini, 1986; Reddy, 1994):

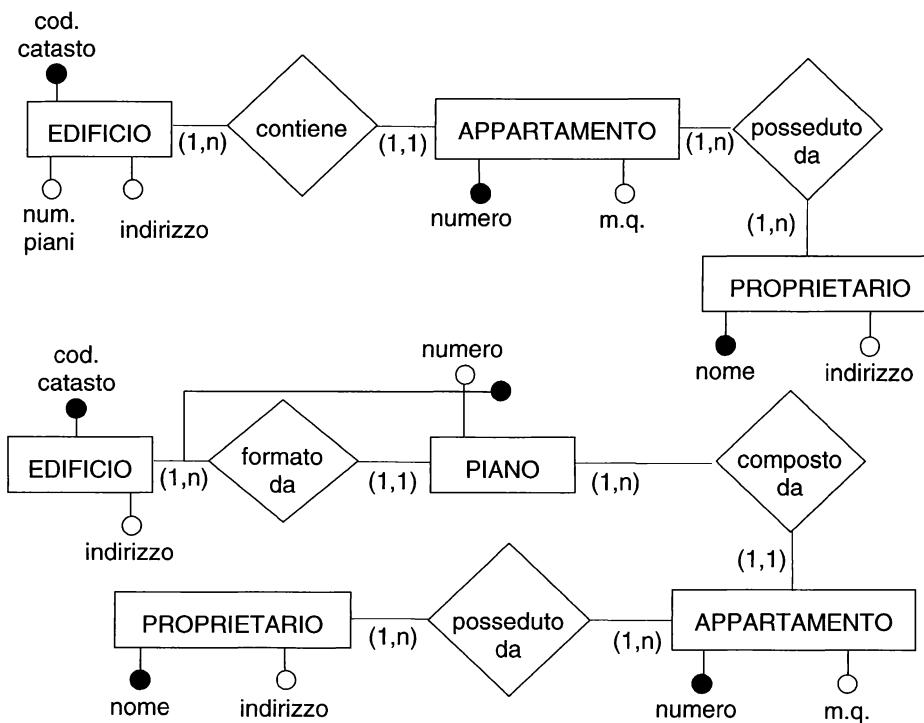
- *Conflitti di eterogeneità*: indicano le discrepanze dovute all'utilizzo di formalismi con diverso potere espressivo negli schemi sorgenti.
- *Conflitti sui nomi*: si verificano a causa delle differenze nelle terminologie utilizzate nei diversi schemi sorgenti; i due principali conflitti di questo tipo sono le *omonimie* e le *sinonimie*. Nelle omonimie lo stesso termine viene utilizzato per denotare due concetti diversi; nelle sinonimie due nomi diversi denotano uno stesso concetto. Mentre le omonimie possono essere individuate attraverso una semplice comparazione dei concetti che presentano gli stessi nomi in schemi diversi, per riconoscere eventuali sinonimi è necessaria una conoscenza approfondita del dominio applicativo. In Figura 3.10 sono riportati alcuni esempi relativi a questi tipi di conflitto: i due schemi della parte alta della figura rappresentano entrambi l'entità ATTREZZATURA, ma mentre nello schema di sinistra questo nome sottintende attrezzatura scientifica (computer, stampanti, plotter ecc.), l'entità nella parte destra fa riferimento al mobilio degli edifici; ovviamente nello schema riconciliato i due concetti dovranno essere opportunamente differenziati. Nella parte bassa della figura risultano essere sinonimi i nomi delle entità CLIENTE e ACQUIRENTE, che fanno riferimento allo stesso concetto del mondo reale. Le due entità dovranno essere accorpate nello schema riconciliato al fine di evitare un'inutile duplicazione. Una corretta documentazione di questo tipo di conflitto può essere ottenuta mediante dizionari dati in cui annotare le corrispondenze (Navathe, 1986).
- *Conflitti semantici*: si verificano quando due schemi sorgenti modellano la stessa porzione di mondo reale a un diverso livello di astrazione e dettaglio. Per esempio, sono in conflitto semantico le coppie di schemi presentate nelle Figure 3.4 e 3.11: in entrambi i casi il livello di dettaglio adottato negli schemi locali è diverso e di conseguenza anche l'insieme di informazioni rappresentabili cambia.
- *Conflitti strutturali*: sono causati da scelte diverse nella modellazione di uno stesso concetto, oppure dall'applicazione di differenti vincoli di integrità. Più in dettaglio i conflitti strutturali possono essere classificati in quattro categorie. I *conflitti di tipo* si verificano quando uno stesso concetto è modellato utilizzando due costrutti diversi (si

veda per esempio la modellazione del concetto di casa editrice in Figura 3.5). I *conflitti di dipendenza* si verificano quando due o più concetti sono correlati con dipendenze diverse in schemi diversi (per esempio nella parte destra di Figura 3.12 il matrimonio tra uomo e donna è codificato mediante un'associazione uno-a-uno, mentre nella parte sinistra l'associazione è di tipo molti-a-molti per permettere la storicitizzazione dell'informazione). I *conflitti di chiave* si verificano quando per uno stesso concetto vengono utilizzati identificatori diversi in schemi diversi (per esempio le persone potrebbero essere identificate mediante il proprio codice fiscale oppure mediante un numero progressivo). I *conflitti di comportamento* si verificano quando diverse politiche di cancellazione/modifica dei dati vengono adottate per uno stesso concetto in schemi diversi (per esempio, in uno schema le informazioni relative a un cliente vengono cancellate se questo non ha ordini attivi, mentre in un secondo schema vengono comunque mantenute per evitare di reinserirne i dati in futuro).

L'individuazione delle correlazioni e dei conflitti richiede una approfondita conoscenza della semantica degli elementi coinvolti, per questo motivo la documentazione delle caratteristiche degli schemi operazionali deve essere svolta in modo accurato e utilizzando formalismi potenti.

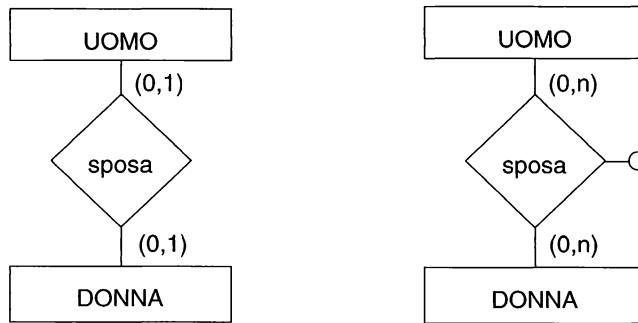


**Figura 3.10** Esempi di sinonimie e omonimie (Batini, 1986).



**Figura 3.11** Due schemi in conflitto semantico: modellano a un diverso livello di astrazione una porzione comune del dominio applicativo.

---



**Figura 3.12** Due diversi schemi per modellare il matrimonio: lo schema di destra permette la storicizzazione delle informazioni<sup>3</sup>.

---

<sup>3</sup> Al fine di semplificare l'esempio si è volutamente omesso di modellare il caso in cui un uomo possa sposare più volte la stessa donna.

### 3.3.3 Allineamento degli schemi

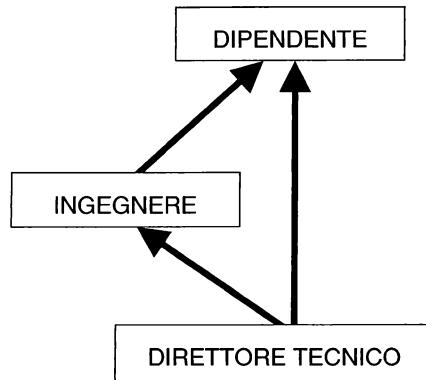
Scopo di questa fase è la risoluzione dei conflitti evidenziatisi al passo precedente, ciò si ottiene applicando primitive di trasformazione agli schemi sorgenti o allo schema riconciliato temporaneamente definito. Tipiche primitive di trasformazione riguardano il cambio dei nomi e dei tipi degli attributi, la modifica delle dipendenze funzionali e dei vincoli esistenti sugli schemi. Non sempre i conflitti possono essere risolti poiché derivano da inconsistenze di base del sistema informativo (si veda la Figura 3.6); in questo caso la soluzione deve essere discussa con gli utenti che dovranno fornire indicazioni su qual è la più fedele interpretazione del mondo reale.

Come già evidenziato in precedenza, in caso di incertezza si preferiscono le trasformazioni che avvantaggiano gli schemi ritenuti centrali nella struttura del futuro data mart. In questo senso la strategia migliore è quella binaria a scala, in cui è possibile iniziare l'integrazione a partire dagli schemi ritenuti più importanti che costituiranno il nucleo dello schema riconciliato. A cominciare da questa fase il progettista deve definire il *mapping* tra gli elementi degli schemi sorgenti e quelli dello schema riconciliato, annotando opportunamente quali siano le corrispondenze tra gli elementi degli schemi sorgenti e quelli dello schema riconciliato.

### 3.3.4 Fusione e ristrutturazione degli schemi

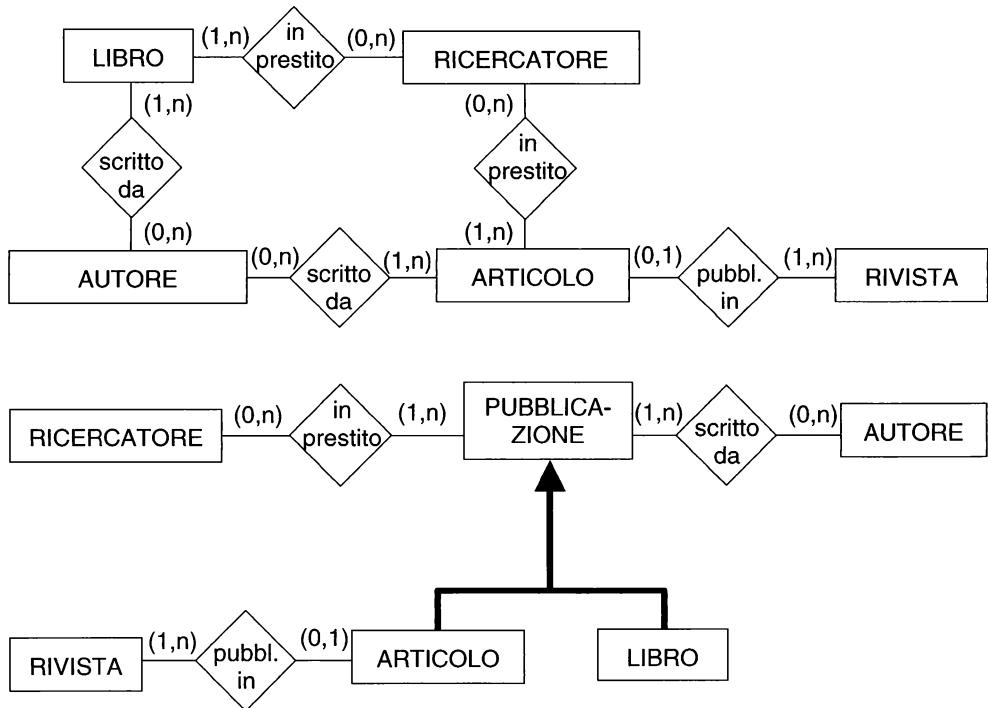
In quest'ultima fase gli schemi allineati vengono fusi a formare un unico schema riconciliato; l'approccio più diffuso è quello di sovrapporre i concetti comuni a cui saranno collegati tutti i rimanenti concetti provenienti dagli schemi locali. Dopo questa operazione si renderanno necessarie ulteriori trasformazioni mirate a migliorare la struttura dello schema riconciliato rispetto a una delle seguenti proprietà:

- *Completezza*: dopo la sovrapposizione degli schemi sorgenti risulteranno visibili ulteriori proprietà inter-schema che non si erano evidenziate in precedenza. Il progettista deve quindi esaminare lo schema fin qui costruito alla ricerca di nuove proprietà che potranno essere esplicitate inserendo nuove associazioni e gerarchie di generalizzazione.
- *Minimalità*: la sovrapposizione di più schemi può generare una forte ridondanza dei concetti che, sebbene espressi in modo univoco negli schemi sorgenti, risultano duplicati o comunque derivabili l'un l'altro in quello riconciliato. Per esempio, in Figura 3.13 la generalizzazione tra DIRETTORE TECNICO e DIPENDENTE è ridondante e può essere eliminata; altre comuni fonti di ridondanza sono le relazioni cicliche tra i concetti e gli attributi derivati.
- *Leggibilità*: il miglioramento della leggibilità dello schema facilita e svelcisce le successive fasi di progettazione. Sebbene sia difficile misurare la differenza di leggibilità tra i due schemi, la nozione qualitativa del termine è relativamente semplice. In Figura 3.14 riportiamo un esempio tratto da Batini (1986) in cui sono mostrati due schemi equivalenti; è facile verificare che lo schema nella parte bassa della figura risulta di più facile interpretazione poiché le relazioni tra i concetti espressi sono meglio organizzate.



**Figura 3.13** Schema con relazione di inclusione ridondante.

---



**Figura 3.14** Due schemi equivalenti con un diverso livello di leggibilità (Batini, 1986).

### 3.4 Definizione delle corrispondenze

Il risultato dell’analisi delle sorgenti operazionali finalizzata all’integrazione è composto da due elementi: lo schema riconciliato in cui sono stati risolti i conflitti presenti tra gli schemi locali, e l’insieme di corrispondenze tra gli elementi presenti negli schemi sorgenti e quelli dello schema destinazione. Come vedremo nel Capitolo 10, queste corrispondenze sono necessarie durante la fase di progettazione dell’alimentazione per definire le procedure utilizzate dagli strumenti ETL per eseguire la migrazione dei dati dalle sorgenti al livello riconciliato.

Nella maggioranza dei casi, l’approccio seguito per stabilire la corrispondenza tra i due livelli dell’architettura prevede che lo schema globale sia espresso in termini degli schemi sorgente ed è perciò detto GAV (*Global-As-View*); più in dettaglio, a ogni concetto dello schema globale deve essere associata una vista il cui significato è definito in base ai concetti che risiedono sugli schemi sorgenti (Levy, 2000). La modalità con cui vengono definite le corrispondenze incide sulla formulazione delle interrogazioni utilizzate negli strumenti ETL per il caricamento dei dati. Con la soluzione GAV sarà sufficiente sostituire a ogni concetto dello schema globale la vista che lo definisce in termini di concetti degli schemi locali; questa operazione, detta *unfolding*, è descritta per esempio da Halevy (2000). La semplicità di formulazione delle interrogazioni viene pagata in termini di estensibilità dello schema riconciliato, infatti l’aggiunta di una nuova sorgente richiederà la modifica di tutti i concetti dello schema globale che la utilizzano.

In Figura 3.15 sono riportati due mapping di tipo GAV nell’ambito di un sistema di ordini. Il formalismo utilizzato è quello relazionale e utilizza il linguaggio SQL per definire le corrispondenze. Ovviamente, nel caso in cui la fase di integrazione sia stata svolta utilizzando il formalismo Entity/Relationship sarà necessario eseguire il progetto logico dello schema riconciliato ricavato prima di definire il mapping con le sorgenti operazionali<sup>4</sup>. Nella parte alta della figura sono riportate due porzioni degli schemi locali del database del magazzino e dell’amministrazione di un’ipotetica azienda, mentre nella parte bassa sono definiti i concetti di cliente e ordine utilizzati nello schema riconciliato. Il primo condensa in un’unica relazione le informazioni presenti nei due schemi relativamente ai clienti, il secondo riunisce le informazioni relative agli ordini, memorizzate per l’anno in corso nel database del magazzino, per gli anni precedenti nel database dell’amministrazione.

Si ricorda che, oltre all’approccio GAV, alcuni autori hanno proposto una soluzione diametralmente opposta che prevede che lo schema globale sia espresso indipendentemente dalle sorgenti, i cui concetti saranno invece definiti come viste sullo schema globale (Calvanese, 1999). Questa soluzione, denominata LAV (*Local-As-View*), non prevede la definizione della vista che descrive direttamente il concetto dello schema riconciliato e richiede pertanto trasformazioni più complesse, indicate in generale con il termine di *query rewriting* (Abiteboul, 1995; Qian, 1996; Ullman, 1997), per capire quali elementi degli schemi sorgenti occorre prendere in considerazione al fine di ricreare il concetto espresso nello schema globale. D’altro canto, l’approccio LAV favorisce l’estensibilità

---

<sup>4</sup> Gli strumenti prototipali per l’automazione della fase di integrazione utilizzano formalismi di natura concettuale che prevedono appositi linguaggi per la definizione diretta delle corrispondenze.

dello schema riconciliato e la sua manutenzione: per esempio, l'aggiunta di una nuova sorgente al sistema richiederebbe solo la definizione della vista sullo schema globale, che non verrebbe quindi necessariamente modificato. Per un confronto approfondito tra le due tecniche si rimanda per esempio a Ullman (1997).

---

```
// DB1 Magazzino
ORDINI2001(chiaveO, chiaveC, data ordine, impiegato)
CLIENTE(chiaveC, nome, indirizzo, città, regione, stato)
.....
// DB2 Amministrazione
CLIENTE(chiaveC, partitalva, nome, telefono, fatturato)
FATTURE(chiaveF, data, chiaveC, importo, iva)
STORICO_ORDINI2000(chiaveO, chiaveC, data ordine, impiegato)
.....
CREATE VIEW CLIENTE AS
SELECT CL1.chiaveC, CL1.nome, CL1.indirizzo, CL1.città, CL1.regione,
       CL1.stato, CL2.partitalva, CL2.telefono, CL2.fatturato
  FROM DB1.CLIENTE AS CL1, DB2.CLIENTE AS CL2
 WHERE CL1.chiaveC = CL2.chiaveC;

CREATE VIEW ORDINI AS
SELECT * FROM DB1.ORDINI2001
UNION
SELECT * FROM DB2.STORICO_ORDINI2000;
```

---

**Figura 3.15** Mapping di tipo GAV per un database di ordini; le chiavi primarie sono sottolineate.

## Modellazione concettuale

---

Mentre è ormai universalmente riconosciuto che un data mart poggia su una visione multidimensionale dei dati, non c'è ancora accordo su come portare a termine la progettazione concettuale a partire dai requisiti utente. D'altronde, è noto che un accurato progetto concettuale è requisito fondamentale per la costruzione di una base di dati ben documentata e pienamente rispondente alle specifiche. Il modello Entity/Relationship è molto diffuso nelle imprese come strumento concettuale per la documentazione standard delle basi di dati relazionali e per la loro progettazione, e molti sforzi sono stati fatti per utilizzarlo anche nella progettazione di sistemi non relazionali (Fahrner, 1995); essendo però orientato a interrogazioni che percorrono le associazioni tra i dati piuttosto che sintetizzarli, mal si presta a essere adottato nel caso dei data mart. Secondo Kimball (1996),

“I modelli Entity/Relationship [...] non possono essere compresi dagli utenti né essere “navigati” efficacemente dal software dei DBMS. I modelli Entity/Relationship non possono essere adottati come fondamento per i data warehouse.”

In effetti, un modello Entity/Relationship ha un'espressività sufficiente per rappresentare la maggior parte dei concetti necessari alla modellazione di data mart; d'altronde, nella sua forma base non è in grado di mettere correttamente in luce gli aspetti peculiari del modello multidimensionale e il suo uso per il DW risulta pertanto poco intuitivo. Inoltre, come apparirà chiaramente dagli esempi illustrati nel seguito, l'uso dell'Entity/Relationship in questo contesto risulterebbe poco economico dal punto di vista grafico-notazionale.

In molti casi i progettisti fondano la costruzione di un data mart sul livello logico, ovvero definiscono direttamente gli *schemi a stella* che, come verrà mostrato nel Capitolo 8, costituiscono l'implementazione standard del modello multidimensionale nei sistemi relazionali. In effetti uno schema a stella non è altro che uno schema relazionale, e racchiude pertanto solo la definizione di un insieme di relazioni e di vincoli di integrità. Utilizzare lo schema a stella come supporto alla progettazione concettuale equivale dunque a progettare un database relazionale senza disegnare prima uno schema Entity/Relationship o, peggio ancora, a iniziare la realizzazione di un sistema informatico complesso dalla fase di scrittura del codice, senza basarsi su alcuno schema di natura statica, dinamica o funzionale: pratica che conduce in genere a risultati piuttosto scoraggianti dal punto di vista della rispondenza ai requisiti, della manutenibilità e del riuso. Nel caso dei DW, la situazione è ulteriormente aggravata dal fatto che, essendo gli schemi a stella quasi completamente

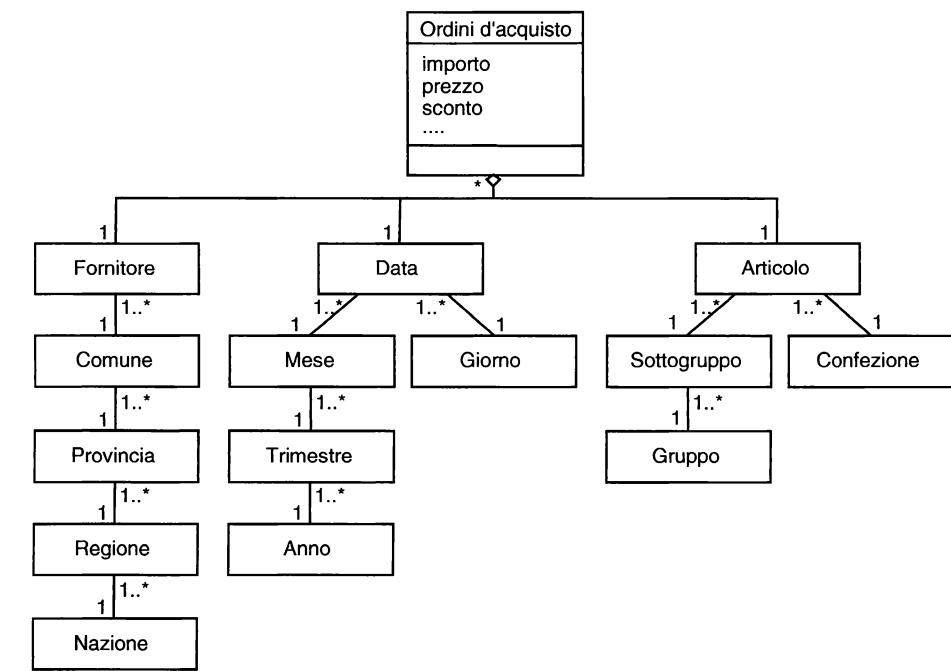
denormalizzati, essi non codificano proprio quelle dipendenze funzionali su cui si basa la definizione delle gerarchie di analisi. Per tutti questi motivi, negli ultimi anni la letteratura ha proposto diversi approcci originali alla modellazione di data mart, alcuni dei quali basati su estensioni del modello Entity/Relationship, altri su estensioni di UML. La Tabella 5.1 elenca i principali modelli presentati, indicando per ciascuno se è definito come modello concettuale oppure logico e se è associato a una metodologia di progettazione; per i modelli concettuali viene anche specificato se sono basati sul modello Entity/Relationship, sul modello a oggetti o se sono modelli ad hoc. Nel resto del paragrafo vengono brevemente delineati i modelli concettuali ritenuti più rappresentativi, con l'obiettivo di evidenziarne le similarità di potere espressivo.

La Figura 5.1 mostra un diagramma delle classi per l'analisi degli ordini d'acquisto (Luján-Mora, 2002). Si tratta di un formalismo a oggetti, e più precisamente di un'estensione al linguaggio UML. L'ordine d'acquisto svolge il ruolo di fatto, ed è modellato tramite una classe UML che ha come attributi le misure del fatto; le dimensioni sono il fornitore, la data e l'articolo ordinato. Sia le dimensioni sia i diversi livelli di aggregazione che le descrivono sono rappresentati come classi. Il fatto e le dimensioni sono legati da aggregazioni (rappresentate in UML da piccoli rombi), i diversi livelli di aggregazione sono collegati da associazioni molti-a-uno.

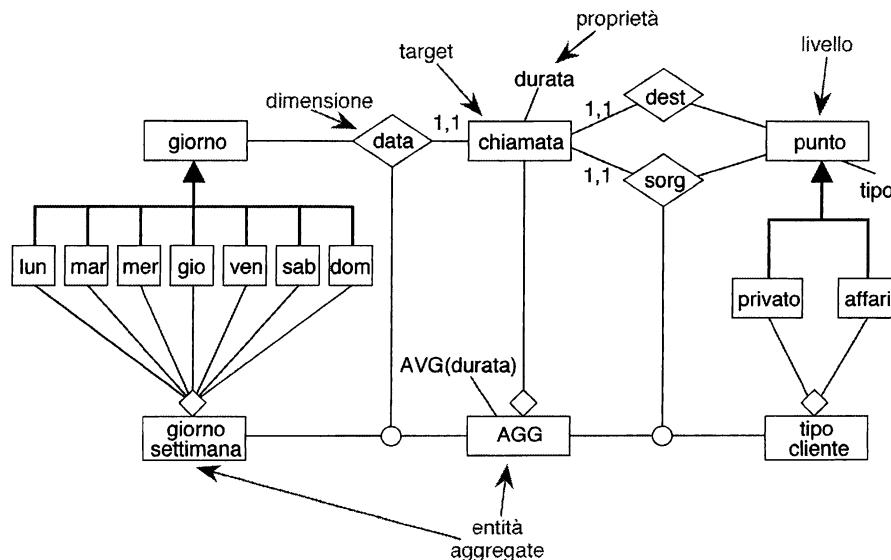
Il modello su cui si basa lo schema concettuale di data warehouse mostrato in Figura 5.2 è invece un'estensione dell'Entity/Relationship (Franconi, 1999). Lo schema nell'esempio analizza le chiamate effettuate tramite una compagnia telefonica; il fatto viene qui chiamato *target*, e le misure *proprietà*. È possibile rappresentare aggregazioni interessanti (*entità aggregate*), e utilizzare le gerarchie di specializzazione dell'Entity/Relationship per enumerare i valori assunti da un livello di aggregazione.

**Tabella 5.1** Approcci alla modellazione di data mart.

Modello	Livello	Metodologia
Agrawal e altri (1995)	Logico	no
Cabibbo e Torlone (1998)	Logico	sì
Datta e Thomas (1997)	concettuale (ad hoc)	no
Franconi e Sattler (1999)	concettuale (E/R)	no
Golfarelli e altri (1998)	concettuale (ad hoc)	sì
Gyssens e Lakshmanan (1997)	logico	no
Hüsemann e altri (2000)	concettuale (ad hoc)	sì
Li e Wang (1996)	logico	no
Luján-Mora e altri (2002)	concettuale (OO)	sì
Nguyen e altri (2000)	concettuale (OO)	no
Pedersen e Jensen (1999)	concettuale (ad hoc)	no
Sapia e altri (1998)	concettuale (E/R)	no
Tryfona e altri (1999)	concettuale (E/R)	no
Tsois e altri (2001)	concettuale (ad hoc)	no
Vassiliadis (1998)	concettuale (ad hoc)	no

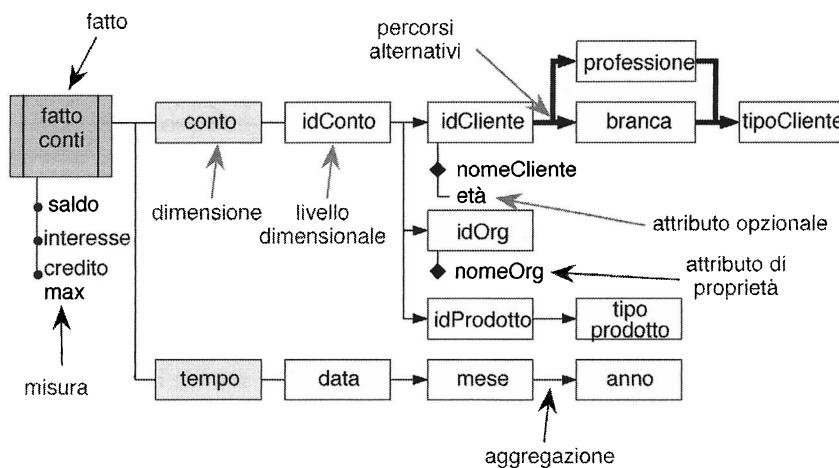


**Figura 5.1** Un diagramma delle classi UML per l'analisi degli ordini d'acquisto (Luján-Mora, 2002).



**Figura 5.2** Uno schema concettuale di data warehouse per le chiamate telefoniche (Franconi, 1999).

Infine, la Figura 5.3 riporta un *fact schema* per l'analisi di conti correnti bancari (Hüsemann, 2000). Oltre a fatti, dimensioni e misure vengono rappresentati attributi non utilizzabili per l'aggregazione (*attributi di proprietà*), attributi opzionali e percorsi alternativi di aggregazione (a un valore di idCliente si abbina o un valore di professione o un valore di branca).



**Figura 5.3** Un fact schema per l'analisi di conti correnti bancari (Hüsemann, 2000).

Nel resto del capitolo viene descritto approfonditamente il Dimensional Fact Model proposto da Golfarelli, Maio e Rizzi (1998). In particolare, nei Paragrafi 5.1 e 5.2 vengono presentati, rispettivamente, i concetti di base e i costrutti più avanzati. Il Paragrafo 5.3 propone una formalizzazione degli aspetti intensionali del modello necessaria per affrontare sia il Paragrafo 5.4, relativo alla sovrapposizione tra schemi, sia i Paragrafi 5.5 e 5.6 che trattano gli aspetti estensionali definendo la semantica dell'aggregazione.

## 5.1 Il Dimensional Fact Model: concetti di base

Il *Dimensional Fact Model* (DFM) è un modello concettuale specificamente concepito per fungere da supporto alla progettazione di data mart; è essenzialmente di tipo grafico, e può essere considerato come una specializzazione del modello multidimensionale per applicazioni di data warehousing (Golfarelli, 1998). Il DFM si pone i seguenti obiettivi:

- dare supporto efficace alla progettazione concettuale;
- creare un ambiente in cui le interrogazioni utente possano essere formulate in modo intuitivo;
- rendere possibile la comunicazione tra il progettista e l'utente finale con l'obiettivo di raffinare le specifiche dei requisiti;

- costituire una piattaforma stabile per la progettazione logica;
- fornire una documentazione di progetto espressiva e non ambigua.

Queste caratteristiche fanno del DFM un ottimo candidato per l'utilizzo in contesti applicativi reali. In particolare, la scelta di questo modello rispetto agli altri presentati nel paragrafo precedente è dovuta al fatto che riteniamo il formalismo grafico particolarmente semplice ed espressivo e quindi adatto alla comunicazione tra progettista e utente finale; inoltre, l'esistenza di una tecnica semi-automatica per la progettazione concettuale a partire dagli schemi delle sorgenti operazionali facilita l'opera del progettista, soprattutto qualora essa venga implementata all'interno di un tool di progettazione. Nel Capitolo 6 viene discussa la tecnica di progettazione concettuale; sul sito internet associato al libro è disponibile una versione dimostrativa di WAND, lo strumento CASE sviluppato dagli autori.

La rappresentazione concettuale generata dal DFM consiste in un insieme di *schemi di fatto*. Gli elementi di base modellati dagli schemi di fatto sono i fatti, le misure, le dimensioni e le gerarchie.

**Fatto.** Un *fatto* è un concetto di interesse per il processo decisionale; tipicamente modella un insieme di eventi che accadono nell'impresa.

Esempi di fatti nel dominio del commercio sono le vendite, le spedizioni, gli acquisti, i reclami. In ambito sanitario esempi interessanti sono i ricoveri, le dimissioni, i trasferimenti, gli interventi chirurgici, gli accessi al Pronto Soccorso. In ambito finanziario si considerano le transazioni di borsa, le situazioni di conti correnti e carte di credito, la creazione di polizze, l'erogazione di prestiti ecc.; in ambito turistico i voli, i noleggi auto, i soggiorni ecc.

È essenziale che un fatto abbia aspetti dinamici, ovvero evolva in qualche modo nel tempo. In realtà, pochi tra i concetti rappresentati in una base di dati sono completamente statici; perfino l'associazione tra città e regioni può cambiare, se i confini vengono modificati. Perciò, la distinzione tra fatti e altri concetti deve essere basata o sulla periodicità media dei cambiamenti, o sugli specifici interessi dell'impresa. Per esempio, l'assegnazione di un nuovo direttore vendite a un reparto avviene meno frequentemente dell'abbinamento di una promozione a un prodotto; quindi, mentre l'associazione tra promozioni e prodotti è una buona candidata a essere modellata come fatto, quella tra direttori di vendita e reparti in genere non lo è, a meno che gli utenti siano interessati a monitorare i trasferimenti dei direttori di vendita per scoprire le correlazioni tra chi è a capo di un reparto e quanto il reparto vende. Per una discussione più approfondita sugli aspetti dinamici negli schemi di fatto si veda il Paragrafo 5.2.9.

**Misura.** Una *misura* è una proprietà numerica di un fatto e ne descrive un aspetto quantitativo di interesse per l'analisi.

Per esempio, ogni vendita è misurata dal numero di unità vendute, dal prezzo unitario, dall'incasso ecc. Il motivo per cui le misure devono essere preferibilmente numeriche è che esse vengono in genere usate per effettuare calcoli. Un fatto può anche non avere mi-

sure, nel caso in cui sia interessante registrare solo il *verificarsi* di un evento; in questo caso lo schema di fatto viene detto *vuoto*. Alcune specifiche problematiche che sorgono per gli schemi vuoti sono trattate nel Paragrafo 5.6.5.

**Dimensione.** Una *dimensione* è una proprietà con dominio finito di un fatto e ne descrive una coordinata di analisi.

Un fatto ha in genere più dimensioni, che ne determinano la granularità minima di rappresentazione. Dimensioni tipiche per il fatto delle vendite sono il prodotto, il negozio e la data. In questo caso l'informazione elementare che può essere rappresentata riguarda le vendite di un prodotto effettuate in un negozio in un giorno; non è possibile distinguere tra vendite effettuate da impiegati diversi o in differenti ore del giorno. In generale, poiché i fatti hanno natura dinamica, uno schema di fatto avrà quasi certamente una dimensione temporale la cui granularità potrà variare tra il minuto e il mese (più probabilmente, giorno o settimana).

Il legame tra misure e dimensioni è espresso, a livello estensionale (ossia a livello di dati piuttosto che di schemi) dal concetto di evento, che qui definiamo informalmente rimandando al Paragrafo 5.5 per gli aspetti formali.

**Evento primario.** Un *evento primario* è una particolare occorrenza di un fatto, individuata da una ennupla costituita da un valore per ciascuna dimensione. A ciascun evento primario è associato un valore per ciascuna misura.

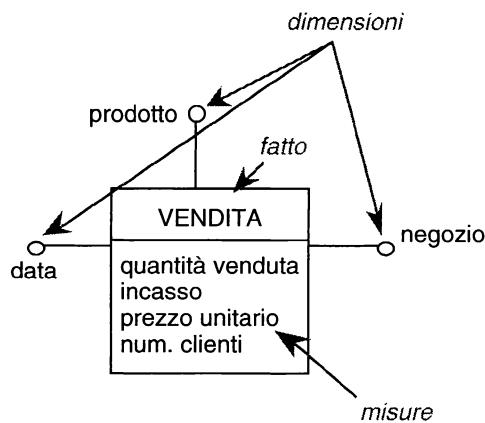
Con riferimento alle vendite, un possibile evento primario regista per esempio che, il 10/10/2001, nel negozio NonSoloPappa sono state vendute 10 confezioni di detersivo Brillo per un incasso complessivo pari a 25 €. Come emerge da questo esempio, le dimensioni vengono normalmente impiegate per l'identificazione e la selezione degli eventi primari.

Usando i concetti introdotti finora è possibile disegnare un semplice schema di fatto per le vendite nella catena di negozi. Come mostrato in Figura 5.4, un fatto è raffigurato da un rettangolo che ne riporta il nome insieme ai nomi delle eventuali misure; le dimensioni sono rappresentate da circoletti collegati al fatto tramite linee.

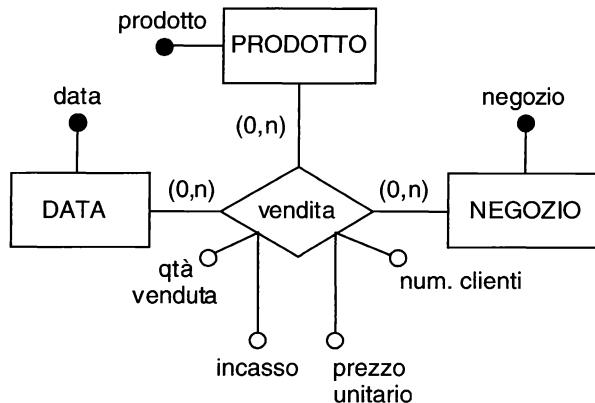
Un fatto esprime un'associazione multi-a-molti tra le dimensioni.

Per questo motivo lo schema Entity/Relationship corrispondente a uno schema di fatto consiste principalmente in una associazione *n*-aria, che modella il fatto, tra entità che modellano le dimensioni. Le misure sono attributi dall'associazione. Lo schema Entity/Relationship corrispondente allo schema di fatto in Figura 5.4 è rappresentato in Figura 5.5.

Occorre notare che alcuni modelli multidimensionali nella letteratura si focalizzano sul trattamento simmetrico di dimensioni e misure (Agrawal, 1995; Gyssens, 1997); ciò costituisce un importante risultato dal punto di vista dell'uniformità del modello logico e da quello della flessibilità degli operatori OLAP. Ciononostante riteniamo che, a livello concettuale, distinguere tra misure e dimensioni sia necessario poiché fa sì che l'attività di progettazione logica possa essere maggiormente mirata a raggiungere l'efficienza richiesta dalle applicazioni di DW.



**Figura 5.4** Un semplice schema di fatto per le vendite.



**Figura 5.5** Schema Entity/Relationship corrispondente allo schema di fatto in Figura 5.4.

Prima di definire cosa si intende per gerarchia, occorre introdurre il concetto di attributo dimensionale.

**Attributo dimensionale.** Con il termine generale *attributi dimensionali* si intendono le dimensioni e gli eventuali altri attributi, sempre a valori discreti, che le descrivono.

Per esempio, un prodotto è descritto dal suo tipo, dalla categoria cui appartiene, dalla sua marca, dal reparto in cui è venduto; prodotto, tipo, categoria, marca, reparto saranno pertanto tutti attributi dimensionali. Le relazioni tra gli attributi dimensionali sono espresse dalle gerarchie.

**Gerarchia.** Una *gerarchia* è un albero direzionato<sup>1</sup> i cui nodi sono attributi dimensionali e i cui archi modellano associazioni molti-a-uno tra coppie di attributi dimensionali. Essa racchiude una dimensione, posta alla radice dell'albero, e tutti gli attributi dimensionali che la descrivono.

Non si confonda il termine *gerarchia* usato in questo contesto con l'identico termine usato nella modellazione Entity/Relationship, dove esso si riferisce ai legami di specializzazione tra entità (gerarchie IS-A). Nel contesto della modellazione multidimensionale la gerarchia si riferisce invece a legami associativi di varia natura, in modo non dissimile dalle gerarchie di aggregazione dei modelli a oggetti. Per esempio, nella gerarchia sulla dimensione prodotto, si avrà un arco da prodotto a tipo per esprimere il tipo di ciascun prodotto, un arco da prodotto a marca per esprimere la marca, un arco da tipo a categoria per esprimere il fatto che tutti i prodotti di tipo bibita appartengono alla categoria alimentari, un arco da categoria a reparto per esprimere il fatto che tutti i prodotti di categoria alimentari sono venduti al reparto alimentare e così via. Nella terminologia relazionale, ogni arco di una gerarchia modella una *dipendenza funzionale* tra due attributi:

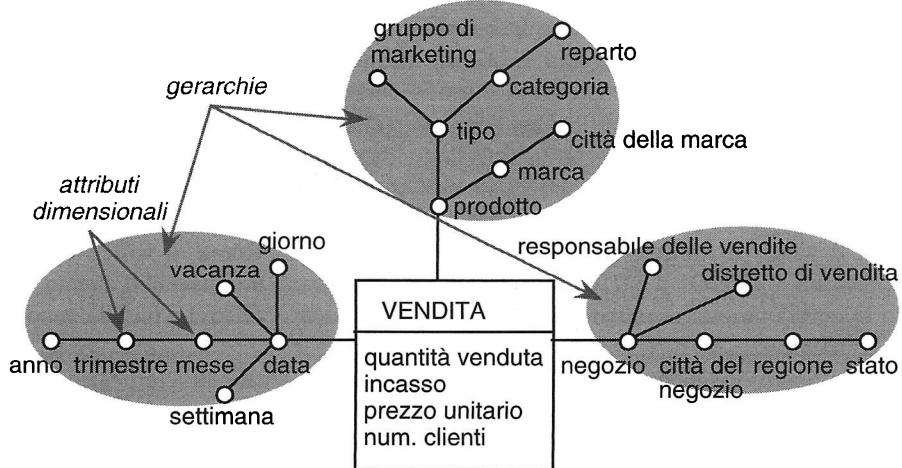
prodotto → tipo, prodotto → marca, tipo → categoria, categoria → reparto ...

Poiché per le dipendenze funzionali vale la proprietà transitiva, ogni cammino direzionato all'interno di una gerarchia rappresenta a sua volta una dipendenza funzionale tra l'attributo di partenza e quello di arrivo: per esempio, prodotto → tipo e tipo → categoria implicano prodotto → categoria.

La Figura 5.6 mostra come lo schema di fatto di Figura 5.4 può essere arricchito aggiungendo gerarchie sulle dimensioni. Gli attributi dimensionali sono rappresentati da circoletti, e sono collegati da linee che denotano gli archi delle gerarchie esprimendo le dipendenze funzionali: per esempio, la città dove è ubicato un negozio ne determina la regione di appartenenza. Le gerarchie sono strutturate come alberi con radice nelle dimensioni; pertanto, non è necessario mostrare esplicitamente le direzioni degli archi: ciascuno di essi è implicitamente orientato nella direzione che si allontana dalla radice.

La Figura 5.6 include una tipica gerarchia temporale che spazia da data ad anno. Più gerarchie temporali che modellano differenti aspetti dinamici possono essere definite sullo stesso fatto: per esempio, uno schema di fatto delle spedizioni può includere una gerarchia sulla data di spedizione e una sulla data dell'ordine. Altre gerarchie spesso presenti sono quella geografica, quella relativa all'organigramma aziendale, quella di tipo parte-componente (*part-of*). Un esempio di gerarchia geografica è quella costruita sulla dimensione negozio in Figura 5.6.

<sup>1</sup> Ricordiamo dalla teoria dei grafi che un *albero* è un grafo aciclico connesso (Berge, 1985). Un albero direzionato è un albero con una *radice*, ovvero un nodo  $v_0$  da cui è possibile raggiungere tutti gli altri nodi attraverso percorsi direzionati. All'interno di un albero direzionato, uno e un solo percorso direzionato connette la radice  $v_0$  a ciascun altro nodo  $v_i$ . Dato un nodo  $v$  in cui entra un arco  $a$  e da cui escono gli archi  $b, c, d\dots$ , chiameremo *padre* di  $v$  il nodo da cui esce  $a$ , e *figli* di  $v$  i nodi in cui entrano  $b, c, d\dots$  I *predecessori* di  $v$  sono, oltre a suo padre, il padre di suo padre, e così via; i *discendenti* di  $v$  sono, oltre ai suoi figli, i figli dei suoi figli, e così via.



**Figura 5.6** Schema di fatto arricchito per le vendite.

Tutti gli attributi dimensionali all'interno di uno schema di fatto devono avere nomi diversi tra loro. Nomi uguali possono essere differenziati qualificandoli con il nome di un attributo dimensionale che li precede nella gerarchia (per esempio, città del negozio e città della marca).

La convenzione proposta da Kimball (1998) prevede che ciascun nome possa essere costituito da tre componenti: un *oggetto* (cliente, prodotto, città ...), una *classe* (medio, totale, data, descrizione ...) e un *qualificatore* (iniziale, primario ...). Per esempio, si potrà così avere data inizio conto corrente, oppure descrizione primaria prodotto.

La presenza delle dipendenze funzionali stabilisce un'associazione molti-a-uno tra i valori assunti da una dimensione e quelli assunti da ogni attributo dimensionale che compone la corrispondente gerarchia. Per esempio, esiste un'associazione molti-a-uno tra i prodotti e i loro tipi (un prodotto è di un solo tipo, un tipo identifica un insieme di prodotti). Di conseguenza, ciascuna ennupla di valori di un insieme qualsiasi di attributi dimensionali è associata a un insieme di ennuple di valori delle dimensioni, ossia a un insieme di eventi primari. Ciò rende possibile utilizzare le gerarchie per definire il modo in cui gli eventi primari possono essere aggregati e selezionati significativamente per il processo decisionale; mentre la dimensione in cui una gerarchia ha radice ne definisce la granularità più fine di aggregazione, agli altri attributi dimensionali corrispondono granularità via via crescenti. Questo concetto viene catturato dalla definizione di evento secondario, che proponiamo qui informalmente.

**Evento secondario.** Dato un insieme di attributi dimensionali (in genere appartenenti a gerarchie distinte), ciascuna ennupla di loro valori individua un *evento secondario* che aggrega tutti gli eventi primari corrispondenti. A ciascun evento secondario è associato un valore per ciascuna misura, che riassume in sé tutti i valori della stessa misura negli eventi primari corrispondenti.

Per esempio, le vendite possono essere raggruppate a seconda della categoria dei prodotti venduti, oppure a seconda del mese in cui sono effettuate le vendite, oppure a seconda della città in cui si trova il negozio, o ancora secondo una qualsiasi loro combinazione. Sceglieremo città del negozio, prodotto e mese come attributi dimensionali per l'aggregazione e consideriamo in particolare l'ennupla (città del negozio: 'Roma', prodotto: 'Brillo', mese: '10/2001'). Questa ennupla individua un evento secondario che aggrega tutte le vendite di prodotto Brillo effettuate nell'ottobre 2001 nei negozi di Roma; in altri termini, tutti gli eventi primari corrispondenti a ennupla in cui il valore di prodotto è 'Brillo', il valore di negozio è uno qualsiasi dei negozi di Roma, il valore di data è compreso tra 1/10/2001 e 31/10/2001. Il valore della misura incasso nell'evento secondario in questione sarà espresso come la somma degli incassi relativi a tutte le vendite da esso aggregate.

Per un approfondimento sulle complesse problematiche legate all'aggregazione, si veda il Paragrafo 5.6.

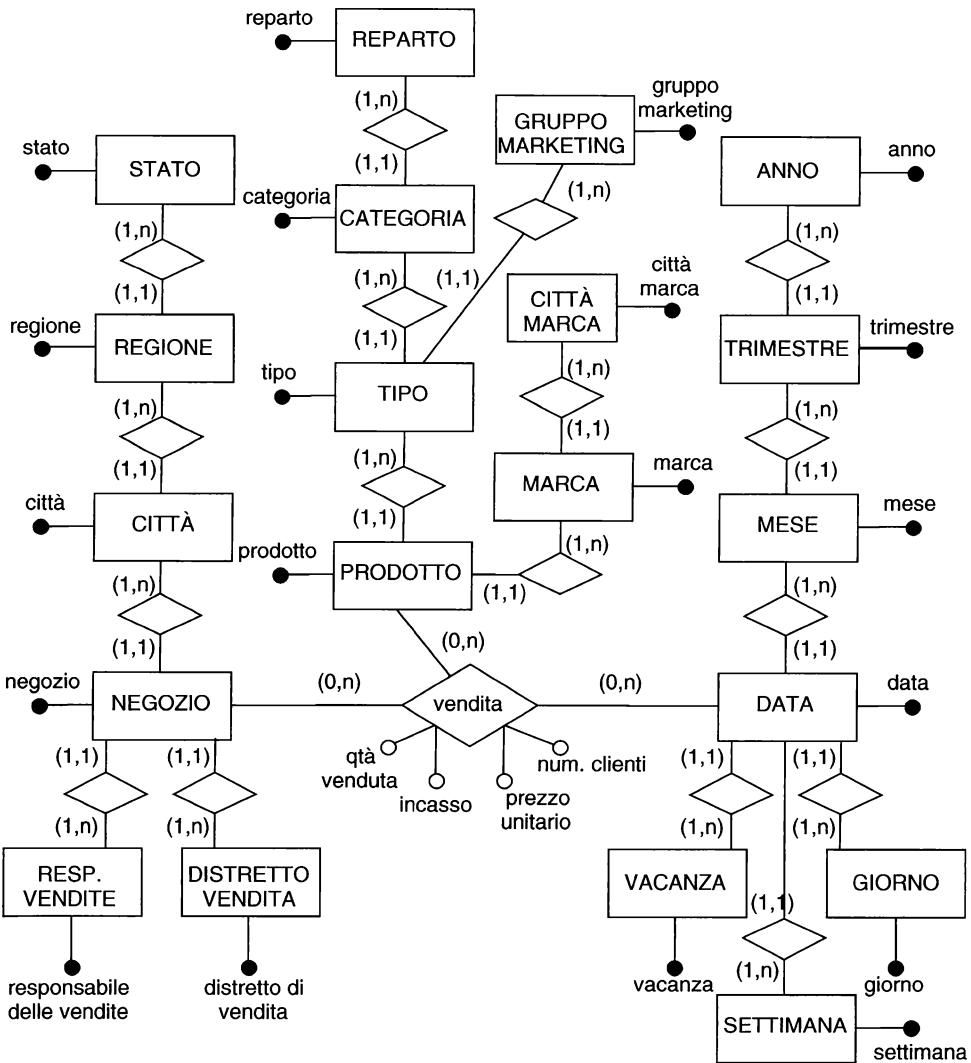
È opportuno a questo punto un rapido confronto con lo schema Entity/Relationship corrispondente allo schema di fatto di Figura 5.6, mostrato in Figura 5.7. Si noti come a ciascun attributo dimensionale (incluse le dimensioni) sia fatta corrispondere un'entità che ha quell'attributo per identificatore, e come ogni arco delle gerarchie sia rappresentato da un'associazione molti-a-uno. Da questo punto di vista, la notazione delle gerarchie adottata negli schemi di fatto può essere interpretata come una "abbreviazione" di quella Entity/Relationship in cui la rappresentazione delle associazioni viene semplificata (non interessa né il loro nome né la loro molteplicità, che è sempre molti-a-uno) e in cui per ciascuna entità viene mostrato il solo identificatore.

## 5.2 Modellazione avanzata

I concetti che verranno introdotti in questo paragrafo, insieme ai costrutti grafici corrispondenti, sono gli attributi descrittivi e cross-dimensional; le convergenze; le gerarchie condivise, incomplete e ricorsive; gli archi multipli e opzionali; la dinamicità; l'additività. Essi, sebbene non necessari nelle più semplici e comuni situazioni di modellazione, risultano altresì utilissimi per esprimere al meglio la moltitudine di sfumature concettuali che caratterizza gli scenari reali. Si vedrà in particolare come l'introduzione di alcuni costrutti faccia sì che le gerarchie cessino di essere semplicemente alberi per diventare, nel caso generale, strutture a grafo più complesse.

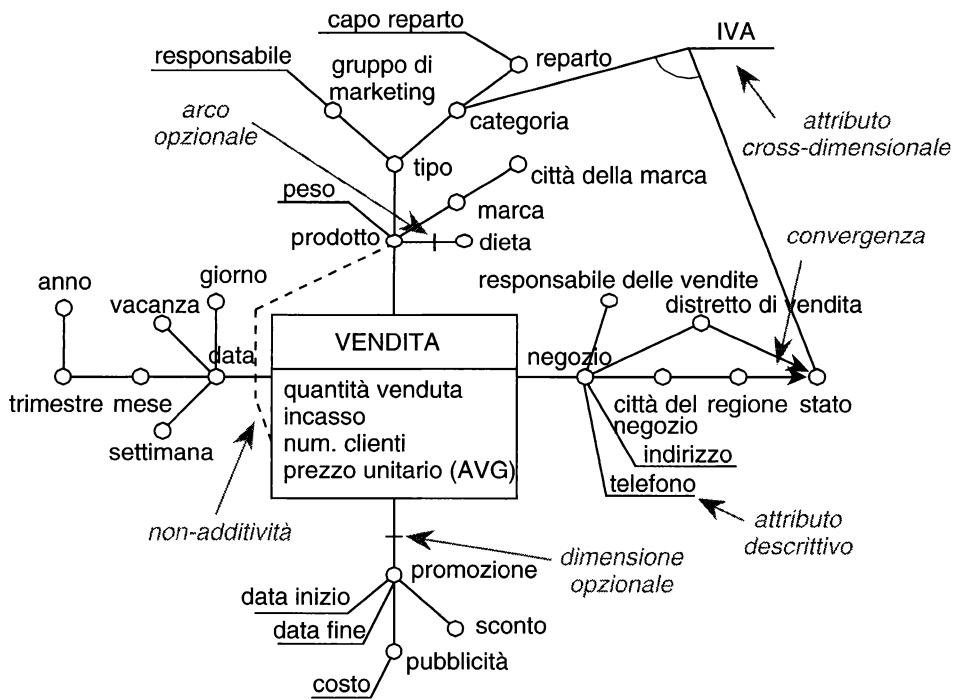
### 5.2.1 Attributi descrittivi

In molti casi è importante rappresentare informazioni aggiuntive su un attributo dimensionale presente in una gerarchia, sebbene sia poco interessante utilizzare queste informazioni come criterio di aggregazione. Per esempio, per l'utente potrà essere utile conoscere l'indirizzo di ciascun negozio, ma difficilmente egli vorrà aggregare le vendite a seconda dell'indirizzo del negozio in cui sono state effettuate. Informazioni di questo tipo sono rappresentate negli schemi di fatto tramite attributi descrittivi.



**Figura 5.7** Schema Entity/Relationship corrispondente allo schema di fatto di Figura 5.6.

Un *attributo descrittivo* specifica una proprietà di un attributo dimensionale di una gerarchia, ed è pertanto da quest'ultimo determinato tramite una dipendenza funzionale. Gli attributi descrittivi hanno spesso domini con valori continui; per questo motivo, a differenza degli attributi dimensionali, non possono essere usati per l'aggregazione. In altri casi sono legati ad attributi dimensionali da associazioni uno-a-uno e pertanto non aggiungono di fatto livelli utili di aggregazione. Esempi sono l'indirizzo e il numero di telefono di un negozio e il peso di un prodotto.



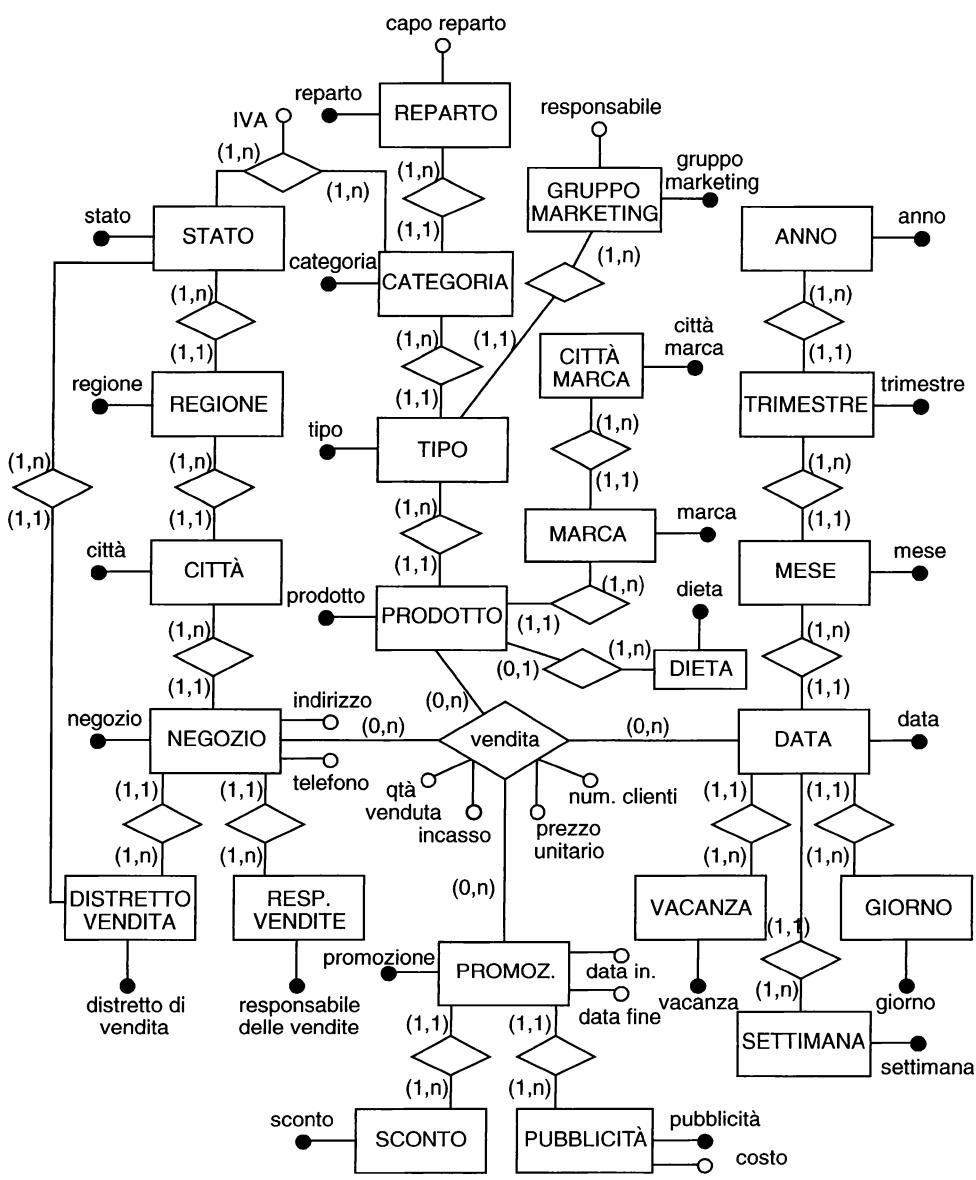
**Figura 5.8** Schema di fatto completo per le vendite.

Gli attributi descrittivi sono sempre foglie delle gerarchie<sup>2</sup> e, come mostrato in Figura 5.8, sono rappresentati graficamente nel DFM da linee orizzontali. La Figura 5.9 illustra la rappresentazione dello schema di fatto di Figura 5.8 secondo il modello Entity/Relationship; si noti come gli attributi descrittivi siano stati modellati come attributi delle entità corrispondenti agli attributi dimensionali.

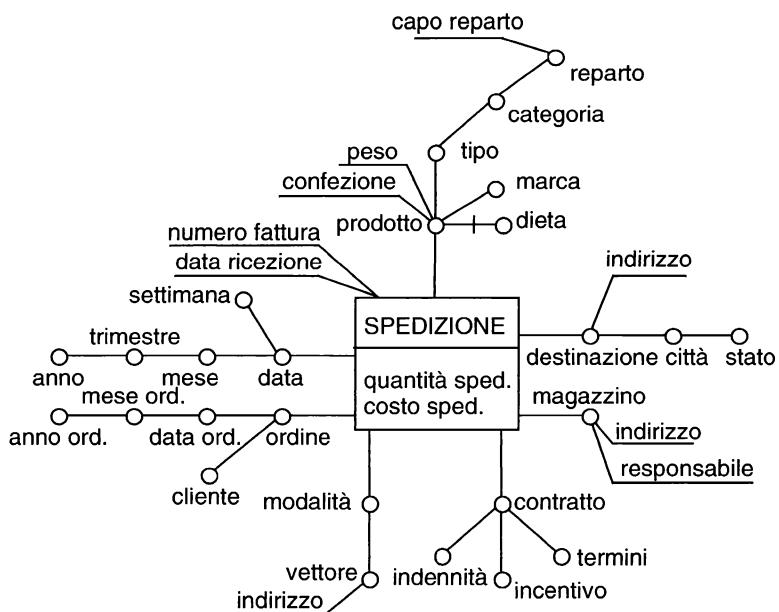
Un attributo descrittivo può anche essere connesso direttamente al fatto, se descrive un evento primario ma non è possibile o interessante utilizzarlo né per identificare i singoli eventi né per effettuare calcoli (altrimenti sarebbe, rispettivamente, una dimensione o una misura). Si consideri per esempio lo schema di fatto delle spedizioni, mostrato in Figura 5.10, in cui devono essere rappresentate la data dell'ordine, la data di spedizione e la data di ricezione. Mentre è utile definire le date di spedizione e di ordine come dimensioni e costruire su di esse due distinte gerarchie temporali, non sarà probabilmente necessario rappresentare anche la data di ricezione come dimensione. D'altronde, rappresentare una data come misura crea problemi poiché gli unici operatori di aggregazione applicabili sarebbero MAX e MIN. Quindi, la data di ricezione può correttamente essere rappresentata come attributo descrittivo del fatto spedizione. Si tenga presente che, affinché sia corretto

<sup>2</sup> Una *foglia* di un albero è un nodo privo di figli.

collegare un attributo descrittivo direttamente al fatto, occorre necessariamente che esso assuma, in corrispondenza di ciascun evento primario, un valore univoco. Nel caso delle spedizioni la rappresentazione adottata per data ricezione non sarebbe corretta qualora, per uno stesso ordine e uno stesso prodotto ordinato, si potessero avere più spedizioni differenziate ricevute in date diverse.



**Figura 5.9** Schema Entity/Relationship corrispondente allo schema di fatto di Figura 5.8.



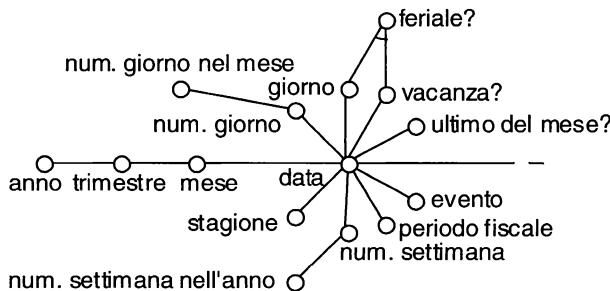
**Figura 5.10** Schema di fatto per le spedizioni.

### 5.2.2 Attributi cross-dimensionaliali

Un *attributo cross-dimensionaliale* è un attributo, dimensionale o descrittivo, il cui valore è determinato dalla combinazione di due o più attributi dimensionali, eventualmente appartenenti a gerarchie distinte. Per esempio, se l'IVA su un prodotto dipende sia dalla sua categoria sia dallo stato in cui il prodotto viene venduto, essa può essere rappresentata tramite un attributo cross-dimensionaliale come mostrato in Figura 5.8, unendo tramite un arco circolare gli archi che lo determinano. La rappresentazione Entity/Relationship equivalente, riportata in Figura 5.9, coinvolge un'associazione IVA tra le entità STATO e CATEGORIA. Un altro esempio è illustrato in Figura 5.11, che illustra la modellazione DFM per la gerarchia temporale completa proposta da Kimball (1998), in cui l'attributo booleano feriale? è determinato congiuntamente da giorno e dal booleano vacanza? (infatti, per sapere se una data corrisponde a un giorno feriale occorre verificare che non cada di domenica e non coincida con nessuna festività).

### 5.2.3 Convergenza

Il concetto di *convergenza* riguarda la struttura delle gerarchie. Di fatto, le gerarchie possono non essere “veri alberi”: infatti, due attributi dimensionali possono essere connessi da due o più cammini direzionali distinti, a patto che ciascuno di essi rappresenti ancora una dipendenza funzionale. Si consideri per esempio la gerarchia geografica su negozio: i negozi sono raggruppati in città, che a loro volta sono raggruppate in regioni appartenenti a stati.

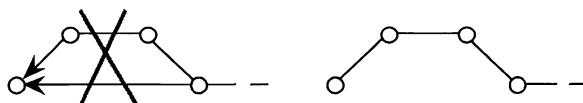


**Figura 5.11** Gerarchia temporale completa.

Si supponga che i negozi siano anche raggruppabili in distretti di vendita, e che non esista nessuna relazione di inclusione tra distretti e regioni; si supponga anche che ogni distretto faccia comunque parte di un solo stato. In questo caso, ciascun negozio appartiene a un unico stato indipendentemente dal cammino seguito ( $\text{negozio} \rightarrow \text{città del negozio} \rightarrow \text{regione} \rightarrow \text{stato}$  o  $\text{negozio} \rightarrow \text{distretto di vendita} \rightarrow \text{stato}$ ). Sullo schema di fatto, le convergenze sono denotate da due o più archi, in genere appartenenti alla stessa gerarchia, che terminano nello stesso attributo dimensionale. In presenza di una convergenza, le direzioni degli archi non possono più essere univocamente determinate dalla struttura dell'albero; vengono pertanto disegnate frecce sugli archi convergenti, come mostrato in Figura 5.8. Nella rappresentazione Entity/Relationship (Figura 5.9) la convergenza è rappresentata dal ciclo ridondante di associazioni tra le entità **NEGOZIO**, **REGIONE**, **CITTÀ**, **STATO** e **DISTRETTO VENDITA**.

Non sempre l'esistenza di attributi apparentemente uguali determina una convergenza. Si consideri per esempio l'attributo **città** della **marca** sulla dimensione **prodotto**, che rappresenta la città in cui i prodotti di una marca vengono fabbricati, e l'attributo **città** del **negozio** sulla dimensione **negozio**. In questo caso i due attributi **città** hanno ovviamente diversi significati e devono essere rappresentati separatamente; infatti, un prodotto fabbricato in una città può essere venduto anche in altre città.

Si noti infine che una gerarchia come quella illustrata in Figura 5.12, in cui uno dei percorsi alternativi non comprende attributi intermedi, non ha ragione di esistere: la convergenza è infatti del tutto ovvia in virtù della transitività delle dipendenze funzionali.

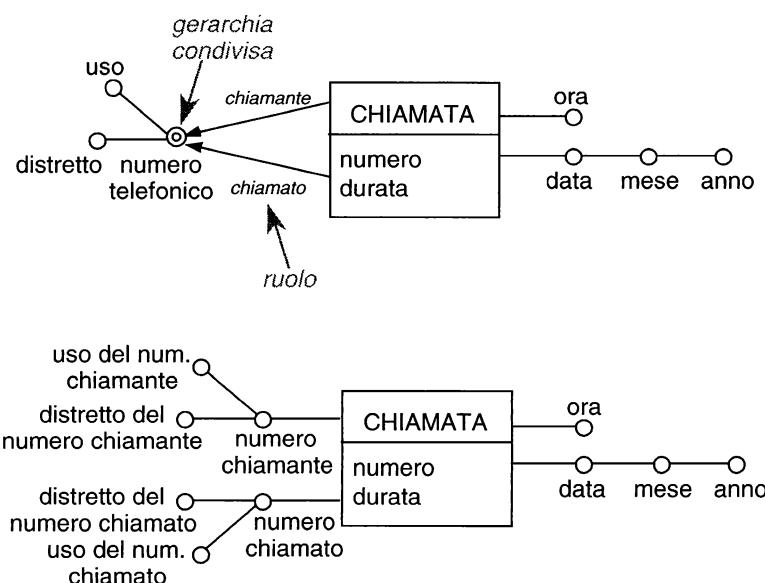


**Figura 5.12** Convergenza ridondante (a sinistra) e sua rappresentazione corretta (a destra).

### 5.2.4 Gerarchie condivise

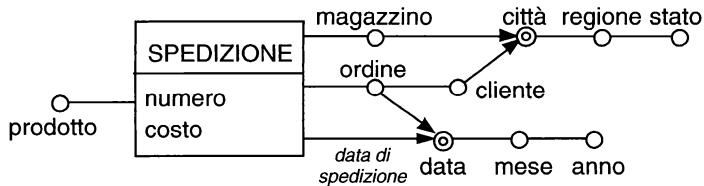
Accade frequentemente negli schemi di fatto che intere porzioni di gerarchie vengano replicate due o più volte. Un esempio classico è costituito dalla gerarchia temporale: un fatto può facilmente avere come dimensioni più attributi di tipo data con significati differenti, e può essere necessario costruire su ciascuno di essi una gerarchia mese-anno; un altro esempio sono le gerarchie geografiche, che possono essere costruite a partire da tutti gli attributi città presenti nello schema (in Figura 5.8 ce ne sono due: città del negozio e città della marca). Poiché tutti gli attributi in uno schema devono avere nomi distinti per evitare ambiguità, ciò obbliga a qualificare i nomi degli attributi appesantendo inutilmente la notazione (si hanno così regione della marca, stato della marca, regione del negozio e stato del negozio).

Si ritiene pertanto consigliabile introdurre una notazione grafica abbreviata che enfatizzi la condivisione delle gerarchie, anche nell'ottica di permettere l'adozione di soluzioni *ad hoc* di progettazione logica (si veda il Paragrafo 9.1.3). La Figura 5.13 mostra un esempio con riferimento a uno schema in cui il fatto di interesse è costituito dalle chiamate telefoniche effettuate, che ha per dimensioni il numero chiamante, il numero chiamato, la data e l'ora della chiamata. L'attributo di partenza per la condivisione (in questo caso numero telefonico) viene evidenziato raddoppiando il circolo che lo rappresenta. Si intende sempre, implicitamente, che tutti i discendenti dell'attributo di partenza siano anch'essi condivisi; qualora uno o più discendenti del tratto condiviso non fossero a loro volta condivisi, occorrerà pertanto rappresentare le gerarchie separatamente.



**Figura 5.13** Gerarchia condivisa e ruoli nello schema di fatto delle chiamate telefoniche e schema equivalente senza condivisione.

Quando la condivisione ha inizio a livello di dimensione, è indispensabile aggiungere a ciascun arco entrante un *ruolo* che ne specifichi il significato. Nel caso di città in Figura 5.14, si potrà invece tralasciare il ruolo in quanto implicitamente definito dai suoi padri (la città del magazzino, la città del cliente).

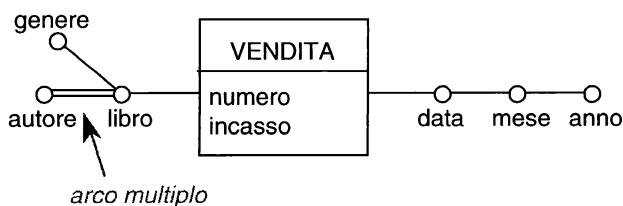


**Figura 5.14** Gerarchie condivise e ruoli nello schema di fatto delle spedizioni.

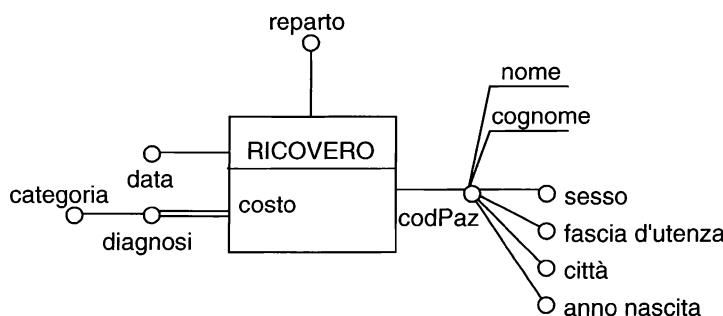
### 5.2.5 Archi multipli

Nella maggior parte dei casi, interessa inserire all'interno delle gerarchie solo attributi collegati ai loro padri da associazioni a-uno. Esistono però situazioni in cui è utile, o addirittura necessario, includere anche attributi che, in corrispondenza di un singolo valore assunto dall'attributo padre, possano assumere valori multipli.

Si consideri lo schema di fatto che modella le vendite di libri in una librerie, le cui dimensioni di interesse sono data e libro. Sarà sicuramente interessante aggregare e selezionare le vendite sulla base degli autori dei libri: d'altronde, poiché molti libri sono scritti da più autori, non sarebbe corretto modellare autore come un attributo dimensionale figlio di libro. Si utilizza allora la notazione illustrata in Figura 5.15, che dal punto di vista grafico comporta il raddoppio dell'arco entrante in autore. In generale, il significato di un arco multiplo che va da un attributo *a* a un attributo *b* è che tra *a* e *b* esiste un'associazione molti-a-molti, in altre parole che a un singolo valore assunto da *a* possono corrispondere più valori di *b*. Nel Paragrafo 5.6.4 mostreremo come, affinché l'aggregazione risulti definibile in modo consistente anche per gerarchie che comprendono archi multipli, sia necessario abbinare a ogni arco multiplo un coefficiente che stabilisca un *peso* per l'associazione tra un valore di *a* e ciascun corrispondente valore di *b*.



**Figura 5.15** Schema di fatto per le vendite di libri.



**Figura 5.16** Schema di fatto per i ricoveri.

Più complesso è il caso in cui l'arco multiplo entra in una dimensione anziché in un attributo qualsiasi. Si supponga, in ambito sanitario, di voler modellare il fatto relativo ai ricoveri; le dimensioni di interesse sono la data di ricovero, l'identificativo del paziente ricoverato, il reparto in cui è avvenuto il ricovero. In realtà sarebbe utile poter aggregare e selezionare i ricoveri anche sulla base della diagnosi di uscita: d'altro canto, a un singolo ricovero corrispondono spesso più diagnosi di uscita, ciascuna appartenente a una categoria. È possibile modellare questa situazione sullo schema di fatto, come mostrato in Figura 5.16, rendendo multiplo l'arco entrante nella dimensione diagnosi.

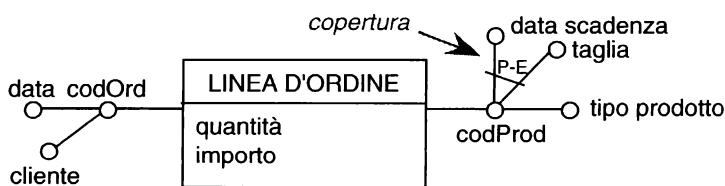
### 5.2.6 Archi opzionali

L'*opzionalità* viene impiegata per modellare situazioni in cui un'associazione rappresentata nello schema di fatto non è definita per un sottoinsieme di eventi. Gli archi opzionali sono denotati contrassegnandoli con un trattino. Se  $r$  è l'arco opzionale, vale la pena distinguere due casi notevoli: (1)  $r$  determina un attributo dimensionale  $a$ ; (2)  $r$  determina una dimensione  $d$ . Nel caso (1), se  $b$  è l'attributo dimensionale che determina  $a$  attraverso  $r$ , interpretiamo l'*opzionalità* assumendo che  $a$  (e tutti i suoi eventuali discendenti nella gerarchia) siano indefiniti in corrispondenza di uno o più valori di  $b$ . Per esempio, l'attributo dieta nello schema delle vendite di Figura 5.8 assume un valore solo per i prodotti alimentari; per gli altri prodotti presenta valore nullo (si noti, nella rappresentazione Entity/Relationship di Figura 5.9, che la molteplicità minima dell'associazione da PRODOTTO a DIETA è 0 anziché 1). Nel caso (2) diciamo che la dimensione  $d$  è opzionale, ovvero che esistono alcuni eventi primari identificati solo dalle altre dimensioni. Per esempio, la dimensione promozione nello schema delle vendite è opzionale: ciò significa che, per alcune combinazioni prodotto-negozi-data, il valore corrispondente per promozione sarà 'nessuna promozione'.

In presenza di un insieme di più archi opzionali uscenti da uno stesso attributo dimensionale è possibile denotarne la *copertura*, ovvero stabilire una relazione tra le diverse opzionalità coinvolte. Analogamente alla copertura delle gerarchie di specializzazione nel modello Entity/Relationship, la copertura di un insieme di archi opzionali è caratterizzabile attraverso due parametri indipendenti tra loro. Sia quindi  $a$  un attributo dimensionale, da cui escono archi opzionali verso gli attributi figli  $b_1, \dots, b_m$ :

- La copertura è *totale* se a ciascun valore di  $a$  è sempre abbinato un valore di almeno uno dei figli; se invece esistono valori di  $a$  per i quali tutti i figli sono indefiniti, la copertura è detta *parziale*.
- La copertura è *esclusiva* se in corrispondenza di ciascun valore di  $a$  si ha al massimo un valore per uno dei figli; se invece esistono valori di  $a$  abbinati a valori di due o più figli, la copertura è detta *sovraposta*.

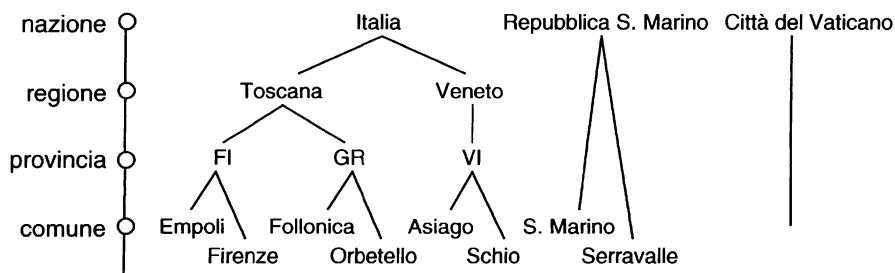
Si hanno pertanto complessivamente quattro tipi di copertura, denotati con T-E, T-S, P-E e P-S. La Figura 5.17 mostra un esempio di opzionalità annotata con la sua copertura. Si suppone che i prodotti possano essere di tre tipi: alimentari, abbigliamento e casalinghi; poiché data scadenza e taglia sono definiti solo, rispettivamente, per alimentari e abbigliamento, la copertura risulta essere parziale ed esclusiva.



**Figura 5.17** Esempio di copertura per un insieme di archi opzionali.

### 5.2.7 Gerarchie incomplete

Sia  $a_1, \dots, a_n$  una sequenza di attributi che forma un percorso all'interno di una gerarchia in uno schema di fatto (per esempio comune, provincia, regione, nazione). Abbiamo finora dato implicitamente per scontato che, per ogni valore di  $a_1$ , esista esattamente un valore per ciascuno degli altri attributi del percorso. Nel caso dell'esempio, ciò risulta effettivamente vero per ogni comune italiano. Ci sono però situazioni, non infrequenti, in cui questa assunzione cade. Basti vedere il caso di due nazioni a noi molto vicine, la Repubblica di San Marino e la Città del Vaticano, per le quali non è definita una scomposizione in regioni e province – nel caso del Vaticano, non è addirittura definita la scomposizione in comuni (si veda la Figura 5.18).



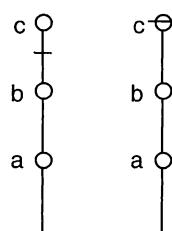
**Figura 5.18** Scomposizione irregolare in regioni e province.

Chiamiamo *gerarchia incompleta* una gerarchia in cui, per alcune istanze, risultano assenti (in quanto non noti oppure non definiti) uno o più livelli di aggregazione. Dal punto di vista grafico, come mostrato in Figura 5.19, contraddistinguiamo una gerarchia incompleta marcando con un trattino tutti gli attributi i cui valori possono mancare. In letteratura, per designare questo tipo di gerarchia si usa spesso il termine *ragged hierarchy* – letteralmente, gerarchia “sfilacciata” (Niemi, 2001). Ciascuno dei livelli di aggregazione ha una semantica precisa e consistente, ma le diverse istanze possono avere lunghezze differenti poiché uno o più livelli mancano, rendendo non uniformi le relazioni padre-figlio tra i livelli (il padre di 'Empoli' appartiene al livello provincia, il padre di 'S. Marino' al livello nazione).



**Figura 5.19** Gerarchia geografica incompleta.

È importante comprendere la differenza tra una gerarchia incompleta e un arco opzionale. Nel primo caso rappresentiamo il fatto che, in corrispondenza di certe istanze della gerarchia, mancano i valori di uno o più attributi *in una posizione qualsiasi della gerarchia* (inclusa quella iniziale e quella terminale). Attraverso un arco opzionale, invece, modelliamo il fatto che mancano il valore di un attributo *e quelli di tutti gli eventuali suoi discendenti*. Evidentemente, qualora solo l'attributo terminale di una gerarchia possa mancare, i due approcci di modellazione risultano del tutto equivalenti (si veda la Figura 5.20).



**Figura 5.20** Due modellazioni equivalenti per la stessa gerarchia.

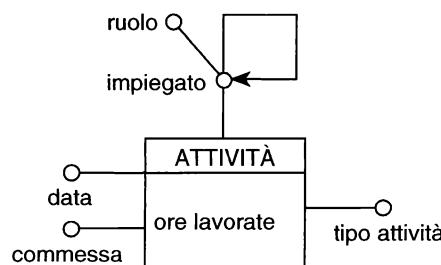
### 5.2.8 Gerarchie ricorsive

A differenza delle gerarchie incomplete, nelle gerarchie ricorsive (*unbalanced hierarchies*) le relazioni padre-figlio tra i livelli sono consistenti, ma le istanze possono avere lunghezze differenti. Un esempio tipico si riferisce a un organigramma aziendale, che mappa le relazioni di subordinazione tra impiegati. La Figura 5.21 mostra la gerarchia dei ruoli degli impiegati in un caso in cui non tutti i rami dell'organigramma hanno la stessa lunghezza; sebbene le relazioni padre-figlio siano consistenti per i due rami, i livelli di aggregazione non sono dal punto di vista concettuale equivalenti (la segretaria del direttore capo e il direttore esecutivo hanno responsabilità ben differenti!).



**Figura 5.21** Gerarchia dei ruoli in un organigramma aziendale non bilanciato.

Una gerarchia ricorsiva viene rappresentata nel DFM come mostrato in Figura 5.22. Nello schema di fatto proposto si misurano le ore lavorate da ciascun impiegato, giorno per giorno, sulle varie commesse distinguendo per tipo di attività. Ogni impiegato ha un ruolo, un ruolo può essere ricoperto da più impiegati. È di interesse modellare le relazioni di subordinazione tra impiegati indotte dall'organigramma aziendale; l'autoanello su impiegato evidenzia il fatto che i vari livelli di aggregazione non sono semanticamente distinguibili a livello dello schema di fatto.



**Figura 5.22** Gerarchia ricorsiva per l'organigramma aziendale.

### 5.2.9 Dinamicità

Il tempo è una fattore chiave nei sistemi di data warehousing, poiché il processo decisionale si basa spesso sulla valutazione di serie storiche e sul confronto tra “fotografie” dell’azienda scattate a istanti diversi.

Fino a questo punto si è ipotizzato che l’unica componente dinamica descritta in uno schema di fatto siano il fatto stesso e gli eventi che lo istanziano; alle gerarchie si è invece attribuita una natura esclusivamente statica. Ciò evidentemente non è del tutto vero: i direttori delle vendite ruotano, sebbene lentamente, sui vari reparti; nuovi prodotti si aggiungono ogni mese a quelli già in vendita, le categorie dei prodotti cambiano, e cambia la loro attribuzione ai prodotti; i distretti di vendita possono essere modificati, oppure un negozio può essere spostato da un distretto all’altro. Occorre precisare che in questo paragrafo sarà considerata solo la dinamicità a livello estensionale, ovvero dei valori: non ci si occuperà quindi delle possibili modifiche che alterano la struttura delle gerarchie (per esempio, l’aggiunta di un nuovo attributo a una gerarchia o l’aggiunta di una nuova dimensione), che non sono da considerarsi fenomeni di routine bensì eventi straordinari legati alla manutenzione del data mart.

La rappresentazione a livello concettuale della dinamicità delle gerarchie è strettamente legata al suo impatto sulle interrogazioni. In presenza di una gerarchia dinamica è infatti possibile ipotizzare quattro differenti scenari temporali di analisi degli eventi. Discuteremo questi scenari con riferimento al seguente esempio: nel dominio delle vendite si assume che, in data 1/1/2000, sia cambiato da Rossi a Bianchi il responsabile delle vendite per il negozio DiTutto, e che sia nato un nuovo negozio DiTutto2 con responsabile Rossi.

*Oggi per ieri.* Tutti gli eventi vengono riportati all’attuale configurazione delle gerarchie.

Nell’esempio, le vendite effettuate dal negozio DiTutto prima del 2000 vengono ascritte a Bianchi insieme a quelle più recenti, mentre a Rossi vengono attribuite le vendite di DiTutto2. Questo approccio risulta di interesse qualora Bianchi diventi il referente anche per le vendite passate.

*Ieri per oggi.* Tutti gli eventi vengono riportati alla configurazione precedentemente assunta dalle gerarchie. Nell’esempio, tutte le vendite effettuate da DiTutto vengono ascritte a Rossi, mentre le vendite di DiTutto2 non vengono considerate.

*Oggi o ieri.* Ciascun evento viene riportato alla configurazione assunta dalle gerarchie nell’istante di tempo in cui l’evento si è verificato. Allora, a Rossi vengono attribuite le vendite di DiTutto precedenti al 2000 e tutte quelle di DiTutto2, mentre a Bianchi vengono attribuite le vendite di DiTutto dal 2000 in poi.

*Oggi e ieri.* Vengono considerati solo gli eventi riferiti alla parte di gerarchia rimasta immutata. Non vengono quindi considerate le vendite di nessuno dei due negozi.

Nell’esempio considerato la dinamicità riguarda un arco di una gerarchia (quello che esprime l’associazione tra negozio e responsabile): variano infatti le istanze dell’associazione corrispondente. In molti casi si ha però a che fare anche con attributi dinamici: per esempio, il nome di una categoria di prodotti o di un negozio può cambiare nel tempo. Supponiamo allora che il negozio DiTutto, in data 1/1/2001, cambi nome diventando DiPiù. I quattro scenari sono così descrivibili:

*Oggi per ieri.* Tutti gli eventi vengono riportati all'attuale configurazione delle gerarchie.

Nell'esempio, tutte le vendite effettuate dal negozio (anche quelle precedenti al 2001) vengono attribuite a DiPiù.

*Ieri per oggi.* Tutti gli eventi vengono riportati alla configurazione precedentemente assunta dalle gerarchie. Nell'esempio, tutte le vendite effettuate dal negozio (anche quelle dal 2001 in poi) vengono attribuite a DiTutto.

*Oggi o ieri.* Ciascun evento viene riportato alla configurazione assunta dalle gerarchie nell'istante di tempo in cui l'evento si è verificato. Allora, a DiTutto vengono attribuite le vendite precedenti al 2001, a DiPiù quelle dal 2001 in poi.

*Oggi e ieri.* Vengono considerati solo gli eventi riferiti alla parte di gerarchia rimasta immutata. Non vengono quindi considerate le vendite del negozio modificato.

È importante osservare che, dal punto di vista della modellazione concettuale, non occorre considerare come dinamicità anche la creazione di un nuovo valore del dominio di un attributo (per esempio, un nuovo prodotto messo in vendita o un nuovo negozio aperto) e la conseguente creazione di una nuova istanza per tutte le associazioni che lo riguardano (il nuovo prodotto deve essere associato a un tipo e a una marca, il nuovo negozio a una città e a un distretto di vendita). Infatti, come si vedrà nel Paragrafo 8.4, le diverse soluzioni di progetto logico proposte si comportano uniformemente con riferimento all'aggiunta di nuovi valori.

Fatte queste premesse, ciò che è utile denotare sullo schema di fatto sono gli scenari di analisi a cui l'utente è interessato per ciascun arco e attributo. In assenza di specificazioni, si assumerà per default che il solo scenario di interesse sia *oggi per ieri*. Se alcuni attributi o archi richiedono scenari differenti si potrà predisporre una tabella in cui indicarli, come mostrato in Tabella 5.2 con riferimento allo schema di fatto delle vendite di Figura 5.8.

**Tabella 5.2** Tabella delle dinamicità per lo schema delle vendite.

Attributi/archi	oggi p ieri	ieri p oggi	oggi o ieri	oggi e ieri
negozio			x	
negozio - responsabile delle vendite			x	x
negozio - distretto di vendita			x	
tipo - gruppo marketing	x	x	x	x
categoria - reparto			x	

## 5.2.10 Additività

L'aggregazione richiede di definire un operatore adatto per comporre i valori delle misure che caratterizzano gli eventi primari in valori da abbinare a ciascun evento secondario. Da questo punto di vista è possibile distinguere tre categorie di misure (Lenz, 1997):

*Misure di flusso.* Si riferiscono a un periodo, al cui termine vengono valutate in modo cumulativo. Esempi sono il numero di prodotti venduti in un giorno, l'incasso mensile, il numero di nati in un anno.

**Misure di livello.** Vengono valutate in particolari istanti di tempo. Esempi sono il numero di prodotti in inventario e il numero di abitanti di una città.

**Misure unitarie.** Vengono valutate in particolari istanti di tempo, ma sono espresse in termini relativi. Esempi sono il prezzo unitario di un prodotto, la percentuale di sconto, il cambio di una valuta.

Gli operatori di aggregazione utilizzabili sui diversi tipi di misure, durante l'aggregazione su gerarchie temporali e non, sono riassunti nella Tabella 5.3.

**Tabella 5.3** Operatori di aggregazione validi per i tre tipi di misure (Lenz, 1997).

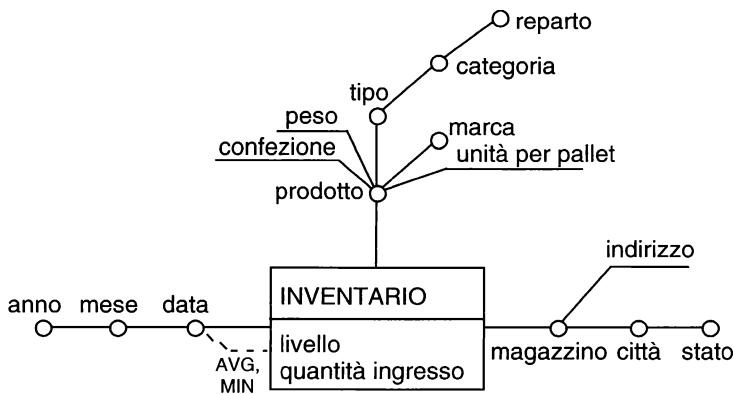
	Gerarchie temporali	Gerarchie non temporali
Misure di flusso	SUM, AVG, MIN, MAX	SUM, AVG, MIN, MAX
Misure di livello	Avg, MIN, MAX	SUM, AVG, MIN, MAX
Misure unitarie	Avg, MIN, MAX	Avg, MIN, MAX

**Additività.** Una misura è detta *additiva* su una dimensione se i suoi valori possono essere aggregati lungo la corrispondente gerarchia tramite l'operatore di somma, altrimenti è detta *non-additiva*. Una misura non-additiva è *non-aggregabile* se nessun operatore di aggregazione può essere usato su di essa.

Dalla Tabella 5.3 appare evidente che, come regola di massima, le misure di flusso sono additive su tutte le dimensioni, quelle di livello sono non-additive sulle dimensioni temporali, quelle unitarie sono non-additive su tutte le dimensioni.

Un esempio di misura additiva sullo schema delle vendite è **quantità venduta**, di tipo flusso: la quantità venduta in un mese è la somma delle quantità vendute nei singoli giorni del mese. Un esempio di misura non-additiva di tipo livello è mostrato in Figura 5.23: il livello di inventario non è additivo sul tempo, ma lo è sulle altre dimensioni. Un esempio di misura non-additiva di tipo unitario è **prezzo unitario** nello schema delle vendite. Comunque, queste misure non-additive possono essere aggregate utilizzando altri operatori quali la media, il massimo, il minimo: per esempio, ha senso calcolare la media di prezzo unitario su più prodotti, negozi e date.

Per le misure non-aggregabili, invece, l'aggregazione è intrinsecamente impossibile per motivi concettuali. Si consideri per esempio la misura **numero di clienti**, stimata per un certo prodotto, negozio e giorno contando il numero degli scontrini, emessi in quel giorno e negozio, che si riferiscono a quel prodotto. Poiché lo stesso scontrino può includere anche altri prodotti, sommare o mediare i numeri di clienti per due o più prodotti porterebbe a un risultato inconsistente; perciò **numero di clienti** è non-aggregabile sulla dimensione prodotto (mentre è additivo su data e negozio). In questo caso, la ragione della non-aggregabilità è che l'associazione tra scontrini e prodotti è di tipo molti-a-molti invece che molti-a-uno: la misura **numero di clienti** non può essere aggregata in modo consistente sulla dimensione prodotto, qualunque sia l'operatore di aggregazione adottato, a meno che la granularità degli eventi venga resa più fine.



**Figura 5.23** Schema di fatto per l'inventario.

Essendo l'additività il caso più frequente, al fine di semplificare la notazione grafica nel DFM si rappresentano esplicitamente solo le eccezioni. In particolare, come mostrato in Figura 5.8, una misura viene collegata alle dimensioni su cui è non-additiva tramite una linea tratteggiata etichettata con gli eventuali operatori di aggregazione utilizzabili. Se una misura presenta lo stesso tipo di additività su tutte le dimensioni, l'operatore di aggregazione può esserne riportato a lato. Qualora il numero complessivo di non-additività fosse tale da ridurre la leggibilità dello schema, è consigliabile adottare una rappresentazione tabellare. La tabella delle additività equivalente per lo schema in Figura 5.8 è rappresentata in Tabella 5.4.

**Tabella 5.4** Tabella delle additività per lo schema delle vendite.

	prodotto	data	negozi	promozione
quantità venduta	SUM	SUM	SUM	SUM
incasso	SUM	SUM	SUM	SUM
numero clienti	-	SUM	SUM	SUM
prezzo unitario	Avg	Avg	Avg	Avg

### 5.3 Aspetti intensionali: descrizione formale

Questo paragrafo si occupa della formalizzazione del DFM con riferimento agli aspetti intensionali. Per i lettori familiari con il linguaggio UML, nel Paragrafo 5.3.1 viene presentato il meta-modello di uno schema di fatto. Nel Paragrafo 5.3.2 vengono invece formalizzati i concetti di base del DFM con lo specifico obiettivo di preparare il terreno per la trattazione della sovrapposizione tra schemi di fatto, affrontata nel Paragrafo 5.4, degli aspetti estensionali, descritti nel Paragrafo 5.5, e in particolare per la definizione della semantica dell'aggregazione nel Paragrafo 5.6. Per semplicità, restano intenzionalmente esclusi dalla formalizzazione i costrutti avanzati del DFM.

# Modellazione logica

---

Mentre la modellazione concettuale è indipendente dal modello logico prescelto per l'implementazione, evidentemente lo stesso non si può dire per i temi legati alla modellazione logica. La struttura multidimensionale dei dati può essere rappresentata utilizzando due distinti modelli logici: quello relazionale, che dà luogo ai cosiddetti sistemi ROLAP, e quello multidimensionale, per il quale si parla invece di MOLAP. Nel seguito presenteremo dapprima brevemente le principali caratteristiche dei sistemi MOLAP, concentrandoci poi su quelli ROLAP su cui si è ormai orientata la maggior parte del mercato e che, presumibilmente, continueranno a risultare vincenti nei prossimi anni. Questa convinzione è avvallata, oltre che dalle analisi di mercato, da considerazioni pratiche: progettisti e implementatori utilizzano da anni i sistemi relazionali nell'ambito delle basi di dati operazionali e difficilmente rinunceranno alle conoscenze acquisite a favore di una tecnologia che non conoscono e verso la quale risultano conseguentemente diffidenti.

## 8.1 I sistemi MOLAP

I sistemi MOLAP (*Multidimensional On-Line Analytical Processing*) memorizzano i dati utilizzando strutture intrinsecamente multidimensionali quali, per esempio, i vettori multidimensionali in cui ogni elemento del vettore è associato a un insieme di coordinate nello spazio dei valori. Questo tipo di struttura dati costituisce la rappresentazione più naturale per i dati di un DW, e può fornire ottime prestazioni poiché ben si presta all'esecuzione delle operazioni OLAP che sono esprimibili direttamente sulla struttura dati e non devono essere “simulate” mediante complesse interrogazioni SQL. Il principale problema a cui è soggetta la soluzione MOLAP è la sparsità dei dati che, al contrario, non crea inconvenienti nell'implementazione relazionale. Mediamente in un cubo di dati meno del 20% delle celle contiene effettivamente delle informazioni (Colliat, 1996), mentre le restanti celle corrispondono a eventi non accaduti. In un DBMS multidimensionale devono essere rappresentate tutte le celle e di conseguenza parte dello spazio su disco viene sprecato per memorizzare celle non informative. Come vedremo nel Paragrafo 8.1.1 sono state sviluppate molte tecniche per evitare la memorizzazione delle celle non informative, ma ovviamente la loro adozione si riflette negativamente sulle prestazioni del sistema.

Un ulteriore freno alla diffusione dei sistemi MOLAP è la mancanza di standard: i diversi sistemi hanno in comune solo i principi di base (strutture dati multidimensionali, gestione della sparsità ecc.), mentre non si conoscono i dettagli dell'implementazione, spesso basata su strutture dati proprietarie (Oracle, 2005; Hyperion, 2005) che li rendono difficilmente sostituibili e accessibili mediante strumenti di terze parti. Non esiste infatti per questi strumenti uno standard di interrogazione che svolga il ruolo del linguaggio SQL nei sistemi relazionali. Questo insieme di elementi alimenta la diffidenza dei progettisti e dei systemisti nei confronti della tecnologia MOLAP, e li spinge a restare fedeli ai ben noti sistemi relazionali.

### 8.1.1 Il problema della sparsità

Con il termine *sparsità dei dati* si fa riferimento al fatto che solo una piccola porzione delle celle di un cubo multidimensionale contengono effettivamente informazioni, mentre le rimanenti corrispondono a eventi non accaduti. Come vedremo nel Paragrafo 8.2, i sistemi ROLAP non sono affetti dal problema della sparsità poiché consentono di memorizzare solo le celle di interesse. Al contrario, nella loro versione di base i sistemi MOLAP rappresentano tutte le celle del cubo introducendo così un forte spreco sia in termini di spazio di disco richiesto, sia in termini di tempo di reperimento delle informazioni.

Sebbene le tecniche per la gestione della sparsità in sistemi MOLAP siano eterogenee dal punto di vista implementativo esse si basano, nella maggior parte dei casi, sui medesimi principi (Zhao, 1997; Colliat, 1996):

- Suddivisione delle dimensioni: consiste nel partizionare un cubo  $n$ -dimensionale in più sottocubi  $n$ -dimensionali (*chunk*). Questa strategia permette di operare su blocchi di dati di dimensione inferiore che possono essere caricati agevolmente in memoria, permette inoltre di differenziare la gestione di chunk densi e chunk sparsi. Un chunk è detto *denso* se la maggior parte delle sue celle contengono dati, al contrario sarà detto *sparsa*.
- Compressione dei chunk: la memorizzazione diretta dei chunk sparsi comporta un notevole spreco di spazio dovuto alla rappresentazione delle celle che non contengono informazioni. Per questo motivo i chunk sparsi vengono rappresentati in forma compressa utilizzando un indice che riporta l'offset delle sole celle del chunk che contengono informazioni.

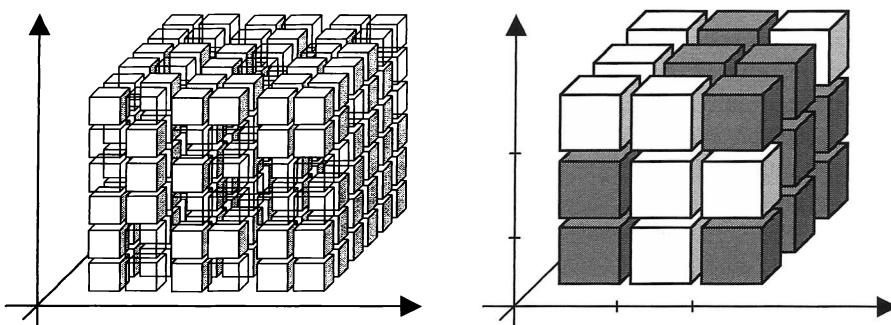
Nella parte sinistra di Figura 8.1 è rappresentato un cubo di dati in cui alcune delle celle non contengono alcun dato (celle trasparenti); in base alla percentuale di celle non utilizzate i diversi chunk di dati evidenziati nella parte destra della figura saranno gestiti come chunk sparsi (in bianco) o chunk densi (in grigio).

## 8.2 I sistemi ROLAP

La soluzione ROLAP (*Relational On-Line Analytical Processing*) utilizza il ben noto modello relazionale per la rappresentazione dei dati multidimensionali. L'adozione di un modello basato su un elemento bidimensionale (righe e colonne di una relazione) per modell-

lare concetti multidimensionali può sembrare forzato, ma in realtà questa scelta è giustificata da un insieme di motivazioni di varia natura, prima fra tutte la constatazione che il modello relazionale è lo standard *de facto* nel settore dei database e in quanto tale è conosciuto da tutti i professionisti del settore. Inoltre, l'evoluzione subita dai DBMS relazionali nell'arco dei trent'anni della loro presenza sul mercato ne fa degli strumenti estremamente raffinati e ottimizzati, al contrario dei sistemi MOLAP comparsi non più di cinque anni fa. Infine, l'assenza del problema della sparsità nella soluzione relazionale garantisce una maggiore scalabilità rispetto ai sistemi MOLAP, fattore fondamentale per database in continua crescita i DW.

Per una trattazione specifica dei temi legati al modello relazionale, una cui conoscenza è propedeutica alla lettura dei successivi paragrafi, rimandiamo ai molti testi disponibili sull'argomento (per esempio Elmasri, 2001; Atzeni, 1999).



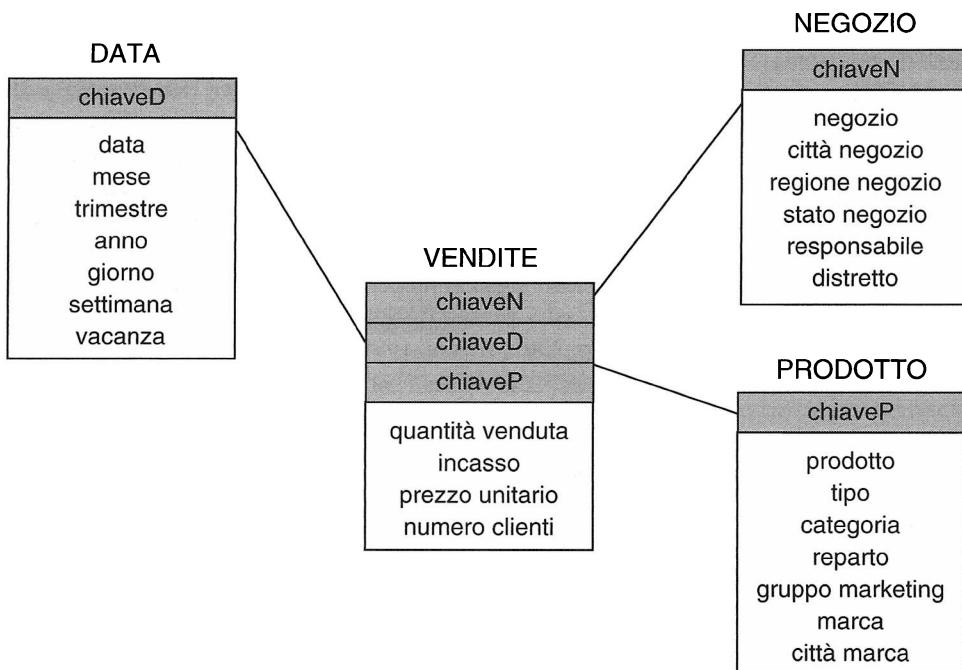
**Figura 8.1** Nella parte sinistra le celle di un cubo di dati: in bianco quelle relative a eventi effettivamente accaduti. Nella parte destra la suddivisione del cubo in chunk: in grigio i chunk densi.

### 8.2.1 Lo schema a stella

La modellazione multidimensionale su sistemi relazionali è basata sul cosiddetto *schema a stella* (*star schema*) e sulle sue varianti.

**Schema a stella.** Uno schema a stella è composto da:

- Un insieme di relazioni  $DT_1, \dots, DT_n$ , chiamate *dimension table*, ciascuna corrispondente a una dimensione. Ogni  $DT_i$  è caratterizzata da una chiave primaria (tipicamente surrogata)  $d_i$  e da un insieme di attributi che descrivono le dimensioni di analisi a diversi livelli di aggregazione.
- Una relazione  $FT$ , chiamata *fact table*, che importa le chiavi di tutte le dimension table. La chiave primaria di  $FT$  è data dall'insieme delle chiavi esterne dalle dimension table,  $d_1, \dots, d_n$ ;  $FT$  contiene inoltre un attributo per ogni misura.



**Figura 8.2** Schema a stella per le vendite; in grigio sono rappresentate le chiavi delle relazioni.

L'esempio di Figura 8.2 mostra lo schema a stella per il fatto delle vendite, il cui schema di fatto è stato presentato in Figura 5.6. La chiave della fact table VENDITE è costituita dalla combinazione delle chiavi esterne dalle tre dimension table. È interessante notare che:

- La visione multidimensionale dei dati si ottiene eseguendo il join tra la fact table e le diverse dimension table. Per il fatto delle vendite di Figura 8.2, l'interrogazione SQL che ricostruisce le celle associando i valori delle misure ai corrispondenti valori degli attributi presenti nelle gerarchie è la seguente:

```
SELECT *
FROM VENDITE AS FT, PRODOTTO AS DT1, NEGOZIO AS DT2,
      DATA AS DT3
WHERE FT.chiaveP = DT1.chiaveP AND FT.chiaveN = DT2.chiaveN AND
      FT.chiaveD = DT3.chiaveD
```

- Nel caso in cui le gerarchie siano conformi (si veda il Paragrafo 5.4), alla chiave di una dimension table si potranno riferire più fact table.

- Le dimension table non sono in terza forma normale<sup>1</sup>, poiché la presenza contemporanea di tutti gli attributi di una gerarchia dà luogo a dipendenze funzionali transitive. Ciò introduce una ridondanza (per esempio la categoria di un tipo di prodotto viene ripetuta per ogni prodotto di quel tipo) e quindi richiede più spazio su disco per la memorizzazione dei dati, ma al contempo riduce il numero dei join necessari al reperimento delle informazioni. Si tenga presente che i tradizionali problemi legati alla normalizzazione (le cosiddette anomalie di inserimento, cancellazione e modifica) non destano preoccupazione alcuna, in quanto le gerarchie sono prevalentemente statiche.
- La sparsità non rappresenta un problema dato che nella fact table vengono memorizzate solo le combinazioni di chiavi per le quali esiste effettivamente l'informazione.

Nell'esempio di Figura 8.3 è rappresentata una possibile istanza del precedente schema a stella; gli attributi in chiave sono sottolineati. La prima tupla della fact table è relativa alla vendita di latte Gnam effettuata il 2/9/2001 nel negozio COOP1 di Bologna, mentre la terza tupla è relativa alla vendita di yogurt Slurp effettuata il 3/10/2001 presso la sede COOP3. Si noti come la denormalizzazione delle dimension table comporti la duplicazione di molti valori: per esempio, per ogni prodotto di tipo 'latticino' si ripete che la categoria corrispondente è 'alimentari'. La caratteristica più evidente di questa soluzione è senza dubbio la denormalizzazione delle dimension table, giustificata dalla riduzione del costo di reperimento dei dati. Si consideri d'altronde che la cardinalità delle dimension table è normalmente molto inferiore rispetto a quella delle fact table e conseguentemente l'aumento della dimensione totale del database risulta trascurabile.

Chiudiamo il paragrafo con alcune considerazioni sull'impiego delle chiavi surrogate, tema che suscita spesso diverse perplessità nei neofiti del settore. Di fatto, sebbene il loro utilizzo sia uno standard nell'ambito del DW, l'adozione delle chiavi surrogate porta con sé dei pro e dei contro che è utile avere ben chiari. Tra i pro ricordiamo che:

- Riducono lo spazio richiesto per l'importazione nella fact table.
- Permettono un accesso più rapido ai dati poiché i piani di accesso possono avvalersi di indici più semplici, costruiti su un singolo attributo numerico.
- Garantiscono l'isolamento rispetto a eventuali variazioni di codifica apportate nelle sorgenti operazionali.
- Come si evincerà dal Paragrafo 8.4, in presenza di gerarchie dinamiche le chiavi surrogate permettono di rappresentare più versioni di un'unica istanza di gerarchia.

Tra i contro si ricorda invece che:

- Le chiavi surrogate determinano un aumento delle dimensioni delle fact table nel caso in cui le chiavi naturali siano comunque riportate nel DW.
- Obbligano, per la verifica di assenza di duplicati nella dimension table, a utilizzare meccanismi differenti dal controllo di integrità primaria (per esempio, a creare un ulteriore indice UNIQUE sulla chiave naturale).

---

<sup>1</sup> Una relazione si dice in *terza forma normale* quando nessuno dei suoi *attributi non primi* (quelli che non fanno parte di chiavi candidate) dipende transitivamente da una chiave. Una *dipendenza transitiva* ha la forma  $a \rightarrow b, b \rightarrow c$ .

VENDITE	<u>chiaveN</u>	<u>chiaveD</u>	<u>chiaveP</u>	qtà venduta	incasso	.....
	1	1	1	170	85	.....
	2	1	2	320	160	.....
	3	2	3	412	412	.....
	.....	.....	.....	.....	.....	.....
NEGOZIO	<u>chiaveN</u>	negozi	città	regione	.....	.....
	1	COOP1	Bologna	Em. Romagna	.....	.....
	2	COOP2	Roma	Lazio	.....	.....
	3	COOP3	Roma	Lazio	.....	.....
	.....	.....	.....	.....	.....	.....
PRODOTTO	<u>chiaveP</u>	prodotto	tipo	categoria	marca	.....
	1	Latte Slurp	latticino	alimentari	Slurp	.....
	2	Latte Gnam	latticino	alimentari	Gnam	.....
	3	Yogurt Slurp	latticino	alimentari	Slurp	.....
	.....	.....	.....	.....	.....	.....
DATA	<u>chiaveD</u>	data	mese	anno	.....	.....
	1	2/9/2001	9/2001	2001	.....	.....
	2	3/10/2001	10/2001	2001	.....	.....
	3	5/10/2001	10/2001	2001	.....	.....
	.....	.....	.....	.....	.....	.....

**Figura 8.3** Una possibile istanza dello schema a stella di Figura 8.2.

- Come si evincerà dal Capitolo 10, in fase di alimentazione rendono necessaria un’operazione di transcodifica rispetto alle corrispondenti chiavi naturali presenti sullo schema sorgente o riconciliato.

### 8.2.2 Lo schema snowflake

Una delle principali caratteristiche dello schema a stella è la presenza di dipendenze funzionali transitive nelle dimension table, che fanno sì che queste ultime non siano in terza forma normale (per esempio la dimension table NEGOZIO di Figura 8.2 contiene la dipendenza funzionale transitiva negozio→città negozio, città negozio→regione negozio). Sebbene la presenza di questo tipo di dipendenze permetta di eseguire più rapidamente le interrogazioni, può essere utile ridurre il livello di denormalizzazione al fine di ottenere uno schema logico più vicino ai dettami della teoria relazionale. Su questa considerazione si basa lo *schema a fiocco di neve* (*snowflake schema*) caratterizzato dalla normalizzazione, in genere parziale, delle dimension table.

**Schema snowflake.** Uno schema snowflake è ottenibile da uno schema a stella decomponendo una o più dimension table  $DT_i$  in più tabelle  $DT_{i,1} \dots DT_{i,m}$  in modo da eliminare alcune delle dipendenze funzionali transitive in esse presenti. Ogni dimension table sarà caratterizzata da:

- una chiave primaria (tipicamente surrogata)  $d_{i,j}$ ;
- un sottoinsieme degli attributi di  $DT_i$  che dipendono funzionalmente da  $d_{i,j}$ ;
- zero o più chiavi esterne riferite ad altre  $DT_{i,k}$  necessarie a garantire la ricostruibilità del contenuto informativo di  $DT_i$ .

Denominiamo *primarie* le dimension table le cui chiavi sono importate nella fact table, *secondarie* le rimanenti.

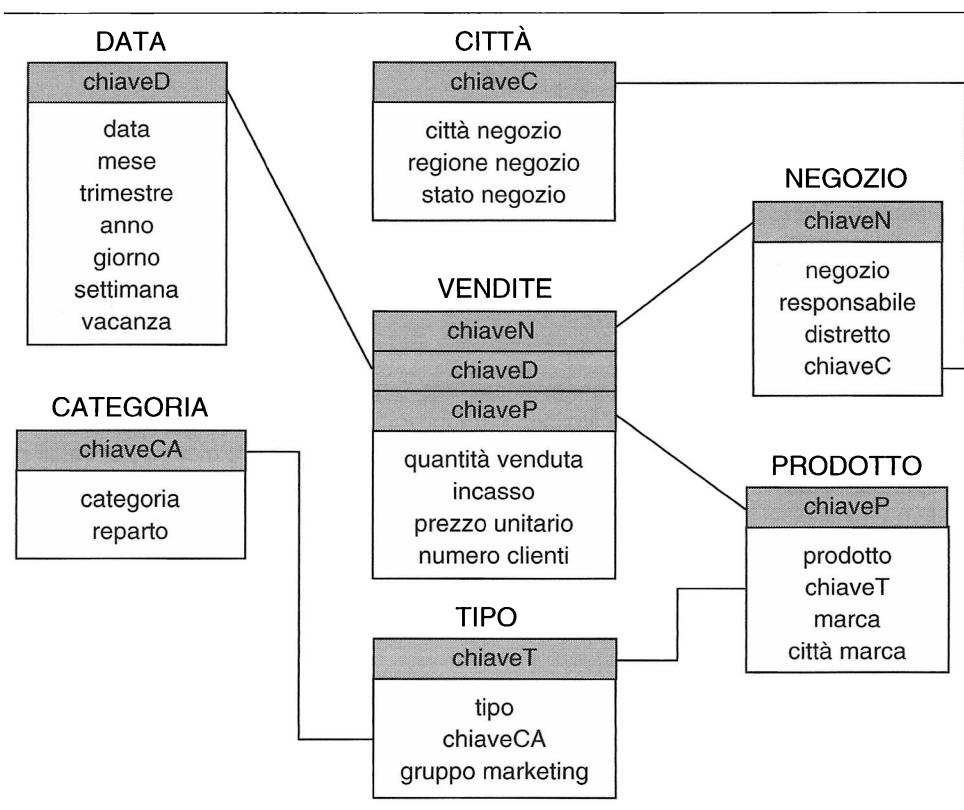
Uno schema snowflake è ottenuto eliminando progressivamente alcune delle dipendenze funzionali transitive presenti nelle dimension table. Ogni passo di normalizzazione insiste su un arco dello schema di fatto e individua una sotto-gerarchia che sarà memorizzata separatamente. Un esempio di schema snowflake è riportato in Figura 8.4. Tramite l'introduzione delle tabelle CITTÀ, TIPO e CATEGORIA si ottiene una parziale normalizzazione dei dati contenuti nelle dimension table; vengono infatti spezzate le dipendenze transitive tra negozio e regione, tra prodotto e categoria e tra tipo e reparto. Come conseguenza si ha che:

- Lo spazio richiesto per la memorizzazione dei dati si riduce poiché, per esempio, le corrispondenze tra valori degli attributi città e regione vengono memorizzate una sola volta. Se il numero di negozi per ogni città è elevato, un ulteriore motivo per cui si risparmia spazio è che a ogni negozio si abbina il surrogato chiaveC (tipicamente 4 byte) invece che il campo città (almeno 20 byte).
- È necessario inserire nuove chiavi surrogate che permettano di determinare le corrispondenze tra dimension table primarie e secondarie. Per esempio, l'importazione di chiaveT nella tabella PRODOTTO permette di associare a ogni prodotto il relativo tipo.
- L'esecuzione di interrogazioni che coinvolgono solo gli attributi contenuti nella fact table e nelle dimension table primarie è avvantaggiata poiché i join coinvolgono tabelle di dimensioni inferiori.
- Il tempo di esecuzione delle interrogazioni che coinvolgono attributi delle dimension table secondarie aumenta poiché è necessario un maggior numero di join.

Gli schemi a stella presentano alcune caratteristiche che richiedono particolare cura al fine di ottenere una corretta decomposizione. Oltre a fare attenzione che la decomposizione non riduca il potere informativo della relazione e non elimini le dipendenze funzionali esistenti nello schema della relazione di partenza (Atzeni, 1999), è necessario verificare che venga spostato nella nuova relazione il corretto insieme di attributi. Infatti, gli schemi a stella contengono normalmente più dipendenze funzionali transitive in cascata (per esempio chiaveN → negozio, negozio → città negozio, città negozio → regione negozio, regione negozio → stato negozio), quindi, affinché la decomposizione sia efficace, è necessario che tutti gli attributi che dipendono, transitivamente e non, dall'attributo che ha

determinato lo snowflaking siano posti nella nuova relazione. In caso contrario rimarrebbero in essere altre dipendenze transitive che invaliderebbero la normalizzazione. Per esempio in Figura 8.5 è mostrata una scorretta normalizzazione della relazione NEGOZIO di Figura 8.2; sebbene la dipendenza transitiva  $\text{chiaveN} \rightarrow \text{negozio} \rightarrow \text{città}$  negozio sia stata risolta, la presenza dell'attributo *stato negozio* nella relazione NEGOZIO induce una forte ridondanza.

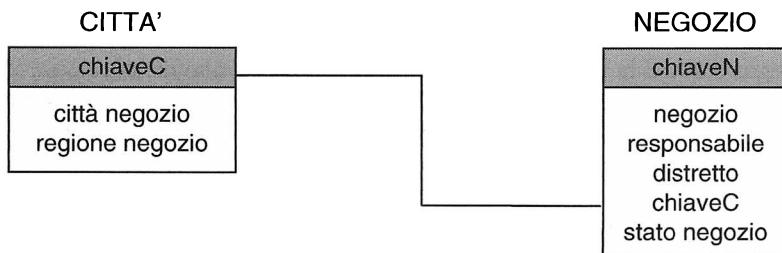
Affinché lo snowflaking sia efficace, tutti gli attributi del sottoalbero dell'attributo da cui ha origine la normalizzazione devono essere spostati nella nuova relazione.



**Figura 8.4** Un possibile snowflake per lo schema a stella presentato in Figura 8.2.

### 8.3 Le viste

L'enorme quantità di dati memorizzati nei DW rende difficili le analisi da parte degli utenti, che di conseguenza tendono a ridurne la porzione da esaminare direttamente tramite operazioni di selezione e aggregazione. Le prime permettono di restringere la porzione di dati di interesse individuando quelli effettivamente interessanti per la specifica analisi,

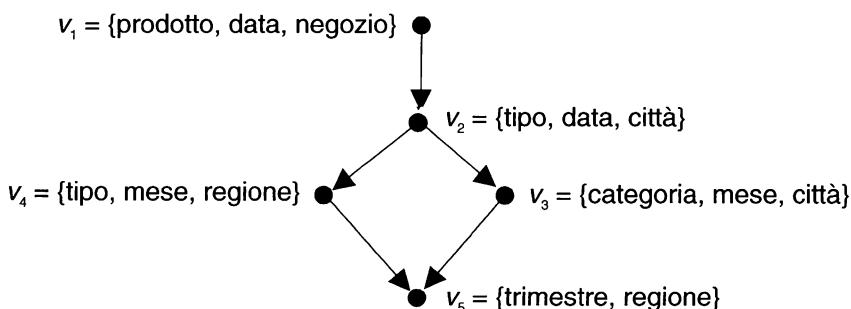


**Figura 8.5** Uno snowflake scorretto per lo schema a stella presentato in Figura 8.2.

mentre con le seconde la riduzione è ottenuta collassando più elementi non aggregati in un unico elemento aggregato (per esempio, calcolando la somma delle quantità vendute in un certo giorno si analizza un solo dato invece di tanti dati quante sono le vendite giornaliere). Inoltre, tramite l'aggregazione è possibile astrarre da casi specifici ed evidenziare trend generali.

Un significativo aumento delle prestazioni può essere ottenuto precalcolando i dati aggregati di uso più comune.

Le fact table contenenti dati aggregati sono comunemente dette *viste* e, dato uno schema di fatto, possono essere individuate dal loro pattern di aggregazione. Nel seguito denoteremo tutte le fact table con il termine *viste* distinguendo quelle *primarie*, corrispondenti al pattern primario (il più fine, definito dall'insieme delle dimensioni), da quelle *secondarie*, corrispondenti a pattern secondari (aggregati). Dove ciò non crea ambiguità utilizzeremo il pattern per denotare la vista. Un modo alternativo per individuare le viste secondarie consiste nel verificare se queste possono essere alimentate a partire da altre viste presenti nel DW, o se è invece necessario ricorrere ai dati operazionali. In Figura 8.6 sono rappresentate alcune viste ottenibili a partire dallo star schema di Figura 8.2. L'unica vista primaria è  $v_1$ . Una freccia da  $v_i$  a  $v_j$  indica che  $P_j \leq P_i$  essendo  $P_i$  e  $P_j$  rispettivamente i pattern di  $v_i$  e  $v_j$ . Conseguentemente, i dati contenuti in  $v_j$  possono essere calcolati aggregando quelli di  $v_i$ .



**Figura 8.6** Alcune viste materializzabili per lo schema delle vendite.

Supponendo di adottare lo schema di materializzazione di Figura 8.6, una interrogazione relativa alle vendite che richieda i dati aggregati per tipo del prodotto, data di vendita e città del negozio in cui la vendita è stata effettuata risulterà molto meno costosa se eseguita sulla vista  $v_2$  piuttosto che sulla vista  $v_1$  poiché essa insisterà su una fact table con un numero ridotto di tuple e non richiederà ulteriori operazioni di aggregazione. Viceversa un'interrogazione che aggreghi le vendite in base a prodotto, data di vendita e città del negozio dovrà essere forzatamente eseguita sulla vista  $v_1$ , che è l'unica con un pattern di aggregazione sufficientemente fine.

Quando si opera con dati aggregati è necessario porre molta attenzione al corretto utilizzo degli operatori di aggregazione. Negli esempi seguenti sono mostrate due potenziali situazioni di errore. In Figura 8.7 è riportata una tabella che mostra le vendite (quantità e prezzo unitario) dei diversi prodotti. Nella tabella successiva i dati sono stati aggregati in base ai tipi dei prodotti; la quantità è stata aggregata utilizzando l'operatore di somma, mentre per il prezzo è stata utilizzata la media. Si noti come, volendo calcolare l'incasso totale delle vendite a partire dalla tabella aggregata, si ottenga un dato diverso da quello ottenuto a partire dalla tabella non aggregata. Come già evidenziato nel Paragrafo 5.6.2 a proposito delle misure derivate, il motivo di questa differenza è che applicando l'operatore di media si perde l'informazione relativa ai singoli prezzi praticati. L'unico modo per rispondere all'interrogazione a partire dalla tabella aggregata consiste pertanto nel memorizzarvi anche i valori degli incassi.

The diagram illustrates the aggregation process. It starts with a detailed sales table:

tipo	prodotto	quantità	prezzo	incasso
latticino	Latte Slurp	5	1,0	5,0
latticino	Latte Gnam	7	1,5	10,5
bibita	Colissima	9	0,8	7,2

Below this table, two arrows point downwards, labeled "SUM" and "AVG", indicating the aggregation operations:

- The "SUM" arrow points to the aggregated table below, where the "quantità" column is summed up to 12.
- The "AVG" arrow points to the same table, where the "prezzo" column is averaged to 1,25.

The resulting aggregated table is:

tipo	quantità	prezzo	quantità × prezzo
latticino	12	1,25	15,0
bibita	9	0,8	7,2

At the bottom right of this table, the total value "totale: 22,2" is shown.

**Figura 8.7** Il calcolo di valori aggregati può determinare degli errori se non si considerano con attenzione le caratteristiche degli operatori utilizzati.

La seconda situazione è illustrata in Figura 8.8, in cui la tabella di partenza contiene il livello di inventario di un certo prodotto nei diversi giorni dell'anno mentre la tabella aggregata contiene il livello di inventario medio per ciascun quadri mestrale. Anche in questo caso non è possibile calcolare il corretto valore medio sull'intero periodo a partire dai dati aggregati. Per risolvere questo problema è necessario memorizzare nella tabella aggregata anche il conteggio delle tuple della tabella di partenza (eventi primari) che partecipano a determinare il valore di ogni tupla della tabella aggregata (eventi secondari).

quadrimestre	data	livello inventario
I '02	1/1/2002	100
I '02	10/2/2002	200
I '02	31/4/2002	60
II '02	5/6/2002	85
II '02	18/7/2002	125
III '02	31/12/2002	110

media: 113,33

AVG

↓

quadrimestre	livello inventario	count	liv. inv. × count
I '02	120	3	360
II '02	105	2	210
III '02	110	1	110

media: 111,66      media pesata: 113,33

**Figura 8.8** Il calcolo di valori aggregati può determinare degli errori se non si considerano con attenzione le caratteristiche degli operatori utilizzati.

Le ragioni dei problemi esemplificati nasce dalla natura degli operatori di aggregazione utilizzati. Come già illustrato nel Paragrafo 5.6.2, secondo Gray e altri (1995) questi possono essere classificati in:

- *Distributivi*: permettono di calcolare dati aggregati a partire da dati parzialmente aggregati. Appartengono per esempio a questa categoria gli operatori di somma, minimo e massimo.
- *Algebrici*: richiedono un numero finito di informazioni aggiuntive (*misure di supporto*) per calcolare dati aggregati a partire da dati parzialmente aggregati. Appartiene per esempio a questa categoria l'operatore di media, che richiede di memorizzare anche il conteggio per calcolare correttamente i dati aggregati.
- *Olistici*: non permettono di calcolare dati aggregati a partire da dati parzialmente aggregati utilizzando un numero finito di informazioni aggiuntive. Appartengono per esempio a questa categoria gli operatori di mediana e moda.

Da questa classificazione si evince chiaramente che, affinché dati pre-aggregati possano essere impiegati per il calcolo di dati ulteriormente aggregati (per esempio se si vuole calcolare l'aggregazione per anno a partire da dati aggregati per quadrimestre), può essere necessario memorizzare ulteriori informazioni. Queste possono essere le misure di supporto o direttamente le misure derivate che si intende calcolare. In presenza di operatori olistici non sarà possibile calcolare il valore aggregato da dati parzialmente aggregati.

Durante la fase di progettazione logica è quindi necessario tenere in considerazione quali operatori dovranno essere utilizzati con le diverse misure, aggiungendo dove necessario le corrispondenti misure di supporto e derivate.

Attualmente gli *aggregate navigator* dei sistemi commerciali, ossia i moduli preposti a riformulare le interrogazioni OLAP sulla “migliore” vista a disposizione, gestiscono soltanto l’aggregazione tramite operatori distributivi. Si riduce così l’utilità delle misure di supporto, che potranno essere coinvolte solo nelle interrogazioni scritte direttamente dagli utenti. Ciò si verifica verosimilmente solo nelle interrogazioni di alimentazione e in quelle che producono la reportistica di base che, modificandosi molto lentamente nel tempo, possono essere ottimizzate da personale specializzato.

### 8.3.1 Schemi relazionali in presenza di dati aggregati

In presenza di viste materializzate è possibile adottare diverse varianti del classico schema a stella. La soluzione più semplice consiste nel memorizzare nella stessa fact table sia i dati della vista primaria sia quelli delle viste secondarie. Con questa soluzione il livello di aggregazione delle singole tuple della fact table potrà essere identificato solo mediante le corrispondenti tuple nelle dimension table: i record delle dimension table corrispondenti a dati aggregati presenteranno infatti dei valori NULL in tutti i campi il cui livello di aggregazione è più fine di quello su cui si sta operando. In Figura 8.9 è rappresentato parte dello schema delle vendite per questa soluzione. Mentre la prima tupla della fact table è relativa a una singola vendita, la seconda memorizza i dati aggregati per tutte le vendite effettuate a Roma, infine la terza tupla riassume tutte le vendite effettuate nel Lazio.

Utilizzando questa soluzione tutte le interrogazioni possono essere risolte sulla stessa fact table. Ciò va a scapito delle prestazioni, che si riducono a causa dell’enorme dimensione che quest’unica tabella verrà ad avere. A essere penalizzate sono soprattutto le interrogazioni che operano su dati aggregati poiché, percentualmente, la porzione di dati per esse rilevanti è minima.

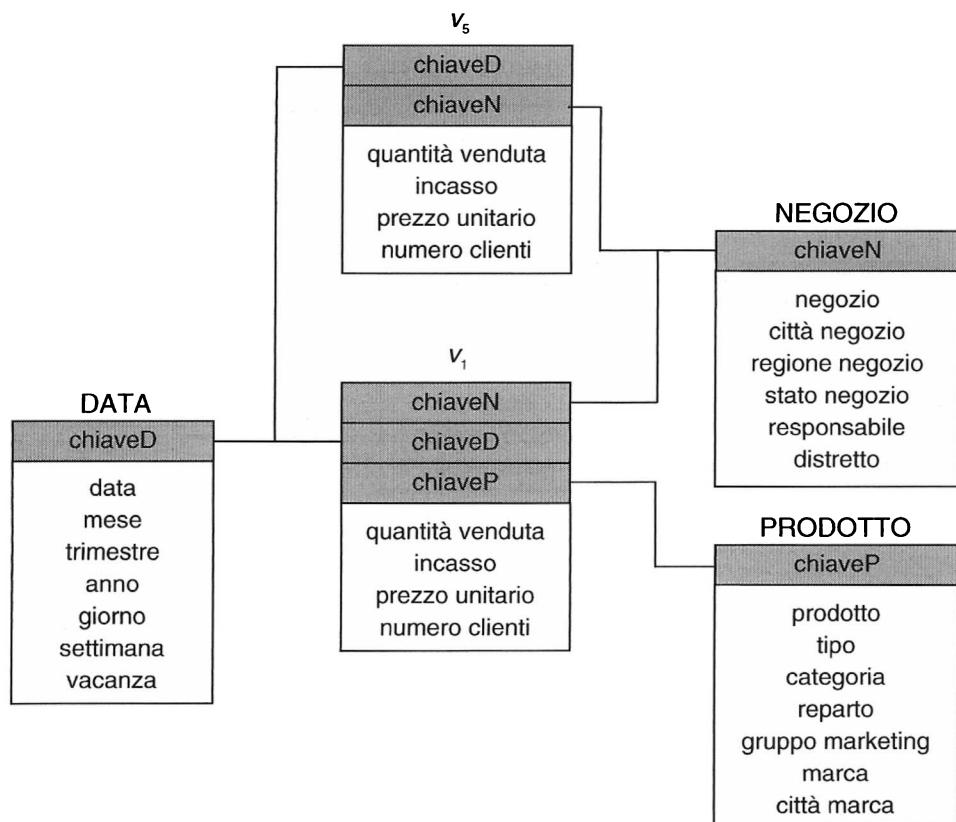
VENDITE	<u>chiaveN</u>	<u>chiaveD</u>	<u>chiaveP</u>	<u>qtà</u>	<u>incasso</u>	.....
	1	1	1	170	85	.....
	2	1	1	300	150	.....
	3	1	1	1700	850	.....
	.....	.....	.....	.....	.....	.....

NEGOZIO	<u>chiaveN</u>	<u>negozi</u>	<u>città</u>	<u>regione</u>	.....
	1	COOP1	Bologna	Em. Romagna	.....
	2	—	Roma	Lazio	.....
	3	—	—	Lazio	.....
	.....	.....	.....	.....	.....

**Figura 8.9** Memorizzazione di dati aggregati utilizzando una sola fact table.

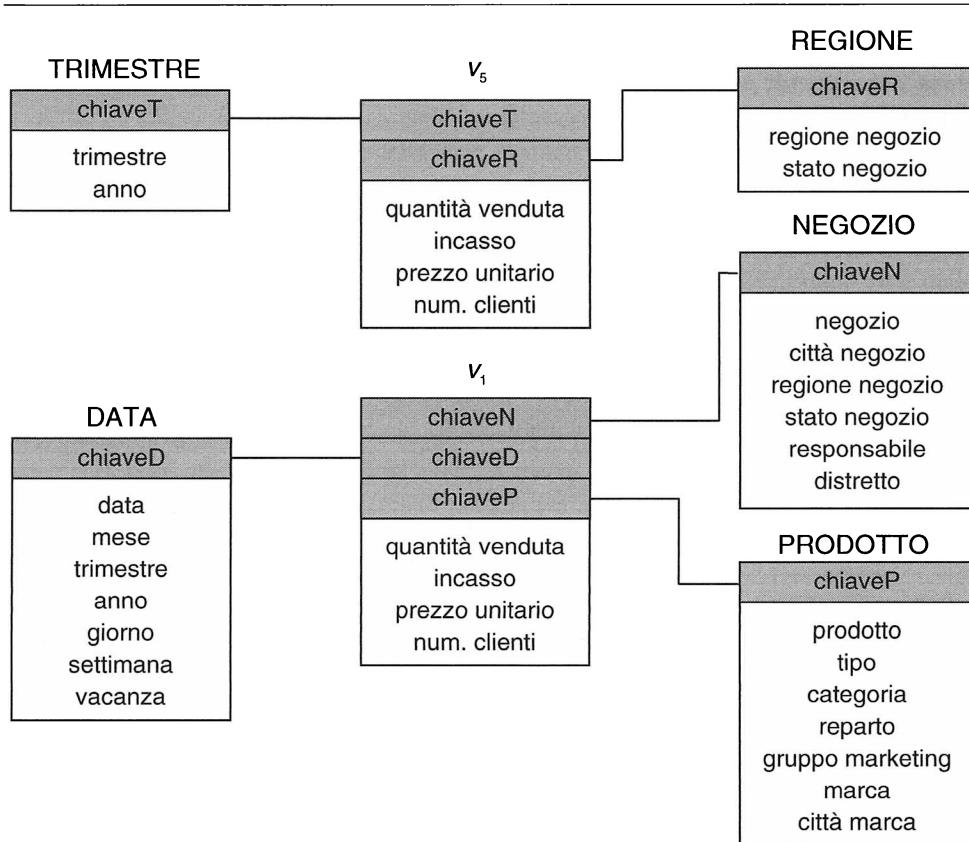
Una seconda soluzione prevede di memorizzare in fact table separate dati relativi a pattern di aggregazione diversi. Le fact table corrispondenti a pattern di aggregazione in cui una o più dimensioni sono completamente aggregate non presenteranno le relative chiavi esterne. La coesistenza di più fact table impone un'ulteriore scelta riguardante le dimension table, che possono essere mantenute unificate come nella soluzione precedente (ottenendo il cosiddetto *constellation schema*, illustrato in Figura 8.10), oppure possono essere replicate per le diverse viste aggregate. Nel primo caso, a essere ottimizzato è l'accesso alle fact table, ognuna delle quali conterrà solo dati a un particolare livello di aggregazione. Al contrario le dimension table continuano a contenere attributi a livelli diversi delle gerarchie, la loro taglia rimane elevata e continua a essere necessaria l'introduzione dei valori NULL nei campi non validi per un particolare livello di aggregazione. La giustificazione per questa soluzione progettuale deriva dalla considerazione che la dimensione delle fact table è di molto superiore a quella delle dimension table e conseguentemente la riduzione del costo di esecuzione delle interrogazioni dipende in larga misura dalla loro ottimizzazione.



**Figura 8.10** Memorizzazione dei dati aggregati tramite constellation schema per lo schema delle vendite.

Nel caso in cui anche le dimension table vengano replicate, ognuna di esse conterrà solo l'insieme di attributi validi per il livello di aggregazione a cui viene utilizzata. In Figura 8.11 sono rappresentate due diverse viste con le relative dimension table. Si noti come la chiave della fact table secondaria non presenti il campo relativo alla dimensione negozio che è stata completamente aggregata.

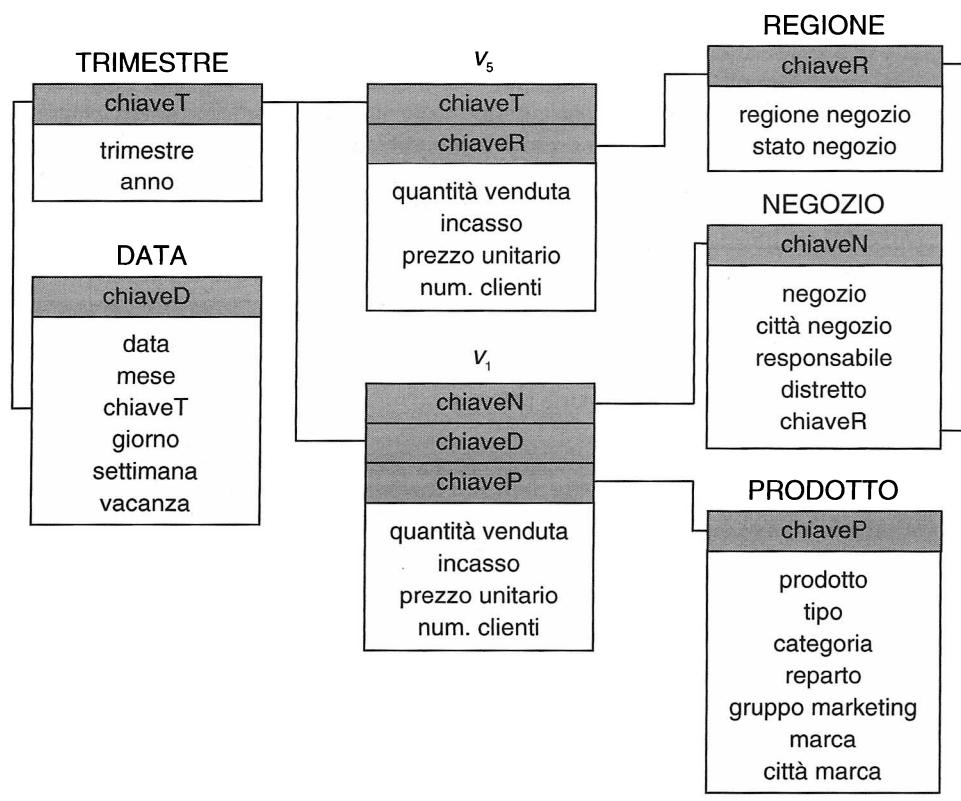
La soluzione mostrata in Figura 8.11 è quella che promette migliori prestazioni. Infatti, sia l'accesso alla fact table sia l'accesso alle dimension table sono ottimizzati. La massimizzazione delle prestazioni va a scapito dello spazio richiesto per la memorizzazione, che cresce non solo a causa del consolidamento dei dati aggregati ma anche a causa delle dimension table ridondanti.



**Figura 8.11** Memorizzazione dei dati aggregati utilizzando più schemi a stella per lo schema delle vendite.

Una soluzione intermedia rispetto alle due presentate consiste nell'applicare lo snowflaking delle dimensioni in corrispondenza dei livelli di aggregazione a cui sono presenti viste aggregate. Questa soluzione consente di usufruire della forte ottimizzazione dovuta alla separazione dei dati aggregati in base al livello di aggregazione senza dover replicare le

dimension table, già fortemente ridondanti. In Figura 8.12 il fatto delle vendite è modellato mediante uno schema snowflake i cui punti di normalizzazione coincidono con gli attributi del pattern della vista aggregata.



**Figura 8.12** Memorizzazione dei dati aggregati tramite snowflake schema per lo schema delle vendite.

## 8.4 Scenari temporali

Il modello multidimensionale assume che gli eventi che istanziano un fatto siano dinamici, e che i valori degli attributi che popolano le gerarchie siano statici. Come già evidenziato nel Paragrafo 5.2.9 questa visione, sebbene rappresentativa del modello multidimensionale, è in genere non attuabile poiché nei casi reali anche le gerarchie subiscono delle modifiche nel tempo: Kimball (1996) parla allora di *slowly-changing dimension*; in questa trattazione, consistentemente con la terminologia adottata, useremo più propriamente il termine *gerarchia dinamica*. La promozione di un dipendente, il passaggio di un prodotto da una categoria merceologica a un'altra, la modifica dell'indirizzo di un negozio sono eventi possibili che devono essere modellati nello schema logico del data mart. La possibilità di tracciare scenari temporali diversi in base ai cambiamenti intervenuti nei valori de-

gli attributi dimensionali rappresenta una caratteristica molto importante, che però richiede alcuni accorgimenti di modellazione per poter essere realizzata appieno.

Come descritto nel Paragrafo 5.2.9, durante la progettazione concettuale vengono evidenziati gli scenari temporali di interesse per i diversi nodi e archi (*ieri per oggi, oggi per ieri, oggi o ieri, oggi e ieri*); durante la progettazione logica si dovrà scegliere la soluzione in grado di modellare lo scenario richiesto. Nel seguito presenteremo tre soluzioni progettuali che permettono di coprire i suddetti scenari; esse si ispirano a quelle proposte nel libro di Kimball (1996) ma ampliano ulteriormente il ventaglio degli scenari temporali realizzabili. Riteniamo infatti che la crescente complessità delle applicazioni di data warehousing richieda una maggiore attenzione verso gli aspetti dinamici delle gerarchie poiché questi permettono di ottenere risultati di grande interesse soprattutto nell'ambito dell'analisi *what-if*. Il limite alla complessità degli scenari temporali tracciabili è sempre determinato dal sovraccosto che essi comportano sia in termini di grandezza delle dimension table, sia in termini di tempo di risposta alle interrogazioni. Esistono sistemi commerciali quali Business Warehouse di SAP (SAP, 1998) che offrono soluzioni più sofisticate di quelle presentate nel seguito, ma riteniamo che esse siano utili solo in un numero limitato di casi e preferiamo quindi non proporle al lettore.

L'adozione di gerarchie dinamiche implica un sovraccosto in termini di spazio e può comportare una forte riduzione delle prestazioni. È quindi indispensabile valutare con attenzione i casi in cui impiegarle.

In Figura 8.13 sono presentati alcuni eventi relativi al fatto delle vendite di Figura 8.2 che saranno utilizzati come esempio nei seguenti sottoparagrafi. Si noti che rispetto alla situazione al 1/1/2001 la gerarchia relativa ai negozi al 1/1/2002 prevede il nuovo negozio DiTuttoDiPiù e l'assegnazione della responsabilità del negozio DiTutto al signor Bianchi; nella parte destra della figura sono mostrate le vendite effettuate durante l'intero periodo.

Situazione al 1/1/2001

negozio	responsabile
DiTutto	Rossi
NonSoloPile	Bianchi
NonSoloPane	Bianchi

Situazione al 1/1/2002

negozio	responsabile
DiTutto	Bianchi
DiTuttoDiPiù	Rossi
NonSoloPile	Bianchi
NonSoloPane	Bianchi

Eventi di vendita

negozio	data	quantità
DiTutto	8/2/2001	100
NonSoloPile	18/10/2001	100
NonSoloPane	25/12/2001	100
DiTutto	8/2/2002	100
DiTuttoDiPiù	5/7/2002	100
NonSoloPile	18/10/2002	100
NonSoloPane	25/12/2002	100

**Figura 8.13** Evoluzione della gerarchia dei negozi ed eventi di vendita nel periodo 2001-2002.

### 8.4.1 Gerarchie dinamiche: tipo 1

Le gerarchie dinamiche di tipo 1 supportano il solo scenario *oggi per ieri*. Pertanto tutti gli eventi, anche quelli passati, vengono sempre interpretati alla luce dell'attuale configurazione delle gerarchie senza tenere traccia del passato.

Per gestire questo tipo di scenario è sufficiente utilizzare il classico schema a stella: nel momento in cui interverrà una modifica a un valore di una tupla della dimension table sarà sufficiente sovrascrivere il vecchio valore con il nuovo. Tutti i dati della fact table vengono associati al nuovo valore della dimension table. Se si utilizza questa soluzione, la dimension table NEGOZIO agli istanti 1/1/2001 e 1/1/2002 conterrà rispettivamente i valori rappresentati nella parte in alto e in basso di Figura 8.14.

La Tabella 8.1 mostra le quantità complessive vendute annualmente da Bianchi e da Rossi utilizzando lo scenario di analisi *oggi per ieri*; per maggior chiarezza, vengono confrontati i risultati ottenuti effettuando l'analisi in due istanti diversi. Si nota immediatamente che, eseguendo l'analisi il 31/12/2002, la vendita effettuata l'8/2/2001 da Rossi viene invece imputata a Bianchi.

NEGOZIO al 1/1/2001	<u>chiaveN</u>	negozi	responsabile	.....
	1	DiTutto	Rossi	.....
	2	NonSoloPile	Bianchi	.....
	3	NonSoloPane	Bianchi	.....
	.....	.....	.....	.....

NEGOZIO al 1/1/2002	<u>chiaveN</u>	negozi	responsabile	.....
	1	DiTutto	Bianchi	.....
	2	DiTuttoDiPiù	Rossi	.....
	3	NonSoloPile	Bianchi	.....
	4	NonSoloPane	Bianchi	.....
	.....	.....	.....	.....

**Figura 8.14** Evoluzione della dimension table NEGOZIO utilizzando gerarchie dinamiche di tipo 1.

**Tabella 8.1** Quantità vendute imputabili ai vari responsabili in base allo scenario *oggi per ieri*: a sinistra situazione al 31/12/2001, a destra la situazione al 31/12/2002.

responsabile	anno	2001	anno	2001	2002
Bianchi		200	Bianchi	300	300
Rossi		100	Rossi	-	100

### 8.4.2 Gerarchie dinamiche: tipo 2

Questa soluzione progettuale supporta lo scenario *oggi o ieri* e consente di registrare la verità storica. Gli eventi memorizzati nella fact table devono essere associati ai dati dimensionali che erano validi quando si è verificato l'evento. Per gestire questo tipo di scenario è sufficiente utilizzare il classico schema a stella: ogni modifica a una gerarchia comporta l'inserimento di un nuovo record che codifichi le nuove caratteristiche nella dimension table corrispondente. Al “vecchio” record non potrà più essere associato alcun nuovo evento.

Le gerarchie dinamiche di tipo 2 permettono di partizionare gli eventi rispetto al tempo senza che sia aggiunto nessun vincolo o marca temporale.

Se si utilizza questa soluzione la dimension table NEGOZIO agli istanti 1/1/2001 e 1/1/2002 conterrà rispettivamente i valori rappresentati nella parte in alto e in basso di Figura 8.15. Le quantità vendute annualmente da Bianchi e da Rossi, calcolate sulla base dello scenario *oggi o ieri*, sono mostrate in Tabella 8.2. Si noti che, diversamente da quanto visto in Tabella 8.1, la vendita effettuata l’8/2/2001 continua a essere imputata a Rossi anche quando l’analisi viene eseguita il 31/12/2002.

Se si opta per la soluzione di tipo 2 per un particolare attributo non sarà poi possibile ricreare lo scenario *oggi per ieri*. È invece possibile adottare strategie diverse su attributi appartenenti alla stessa gerarchia.

NEGOZIO al 1/1/2001	<u>chiaveN</u>	negozio	responsabile	.....
	1	DiTutto	Rossi	.....
	2	NonSoloPile	Bianchi	.....
	3	NonSoloPane	Bianchi	.....
	.....	.....	.....	.....

NEGOZIO al 1/1/2002	<u>chiaveN</u>	negozio	responsabile	.....
	1	DiTutto	Rossi	.....
	2	NonSoloPile	Bianchi	.....
	3	NonSoloPane	Bianchi	.....
	4	DiTuttoDiPiù	Rossi	.....
	5	DiTutto	Bianchi	.....
	.....	.....	.....	.....

**Figura 8.15** Evoluzione di NEGOZIO utilizzando gerarchie dinamiche di tipo 2.

**Tabella 8.2** Quantità vendute imputabili ai vari responsabili in base allo scenario *oggi o ieri*: a sinistra situazione al 31/12/2001, a destra situazione al 31/12/2002.

responsabile	anno	2001		anno	2002	
		Bianchi	Rossi		Bianchi	Rossi
Bianchi		200			200	300
Rossi		100			100	100

Si noti che a ogni evento nella fact table corrisponde un record della dimension table che codifica con esattezza la realtà che era vera al momento del verificarsi dell'evento. Nel caso in cui una interrogazione esprima una selezione sull'attributo che ha subito la modifica, questa soluzione permetterà di individuare con precisione le tuple corrispondenti allo specifico valore. Al contrario, se l'interrogazione esprime dei vincoli su attributi i cui valori non sono stati modificati, saranno individuate tutte le tuple indistintamente.

Questa caratteristica è ovviamente positiva perché fornisce un notevole livello di dettaglio sui dati senza introdurre una eccessiva complessità nella soluzione. Per esempio, si consideri la gerarchia relativa ai prodotti contenente tra gli altri un attributo che ne codifica la versione, la quale, pur cambiando nel tempo, non altera sostanzialmente le caratteristiche del prodotto. Tramite le gerarchie di tipo 2 è quindi possibile calcolare con precisione le vendite relative a ogni singola versione del prodotto, ma è altresì semplice calcolare il totale delle vendite per tutte le versioni dello stesso prodotto.

Nel caso in cui i valori delle dimension table cambino con notevole velocità questa soluzione può determinare un forte aumento delle dimensioni delle tabelle. In questo caso può essere utile eseguirne lo snowflaking in modo da ridurre la proliferazione dei record.

#### 8.4.3 Gerarchie dinamiche: tipo 3

Questa soluzione progettuale è la più sofisticata e dà supporto per tutti gli scenari temporali. La sua implementazione richiede la storizziazione completa della gerarchia e non può pertanto essere basata sul classico schema a stella. Gli elementi necessari per la gestione di una gerarchia di tipo 3 sono:

- una coppia di *time-stamp* (marche temporali) che indichino l'intervallo di validità di una tupla;
- un meccanismo per individuare tutte le tuple coinvolte in una sequenza di modifiche.

Tra le varie possibilità per la realizzazione di una soluzione con queste caratteristiche ne proponiamo una che fornisce, in fase di interrogazione, prestazioni sufficientemente rapide. Per ogni dimension table che dovrà essere gestita come tipo 3 è necessario aggiungere allo schema della dimension table:

1. una coppia di attributi da utilizzare come time-stamp;
2. un attributo master che riporta, per ogni record modificato, il valore di chiave della tupla da cui il record ha avuto origine. Qualora una tupla abbia subito più modifiche, la tupla a cui fare riferimento è quella primigenia e non quella risultante dall'ultima mo-

difica. Ovviamente i record primigeni sono identificabili poiché importano il proprio valore di chiave nel campo **master**.

In uno schema a stella così modificato la dinamicità viene gestita aggiungendo, in corrispondenza di ogni modifica di un attributo, un nuovo record nella dimension table (come per le gerarchie di tipo 2) e aggiornando di conseguenza i valori dei time-stamp e dell'attributo **master**. Considerando gli eventi di Figura 8.13 e supponendo in più che il nome del negozio NonSoloPane sia cambiato due volte nel corso del 2001, la Figura 8.16 presenta i valori della dimension table dopo l'1/1/2002.

---

NEGOZIO al 1/1/2001	chiaveN	negozio	responsabile	.....	da	a	master
	1	DiTutto	Rossi	.....	1/1/2001	—	1
	2	NonSoloPile	Bianchi	.....	1/1/2001	—	2
	3	NonSoloPane	Bianchi	.....	1/1/2001	—	3
	.....	.....	.....	.....	.....	.....	.....

NEGOZIO al 1/1/2002	chiaveN	negozio	responsabile	.....	da	a	master
	1	DiTutto	Rossi	.....	1/1/2001	31/12/2001	1
	2	NonSoloPile	Bianchi	.....	1/1/2001	—	2
	3	NonSoloPane	Bianchi	.....	1/1/2001	30/6/2001	3
	4	SoloPane	Bianchi	.....	1/7/2001	31/10/2001	3
	5	PaneEPizza	Bianchi	.....	1/11/2001	—	3
	6	DiTuttoDiPiù	Rossi	.....	1/1/2002	—	6
	7	DiTutto	Bianchi	.....	1/1/2002	—	1
	.....	.....	.....	.....	.....	.....	.....

---

**Figura 8.16** Gestione della dimension table NEGOZIO come gerarchia dinamica di tipo 3.

Si noti che per le tuple modificate è indicato anche l'istante in cui termina il periodo di validità (attributo **da**). Per ogni tupla è inoltre possibile individuare la tupla che l'ha originata: per esempio le tuple con chiave 4 e 5 derivano entrambe da quella con chiave 3, la tupla con chiave 7 deriva da quella con chiave 1 e infine le tuple con chiave 2 e 6 non sono state modificate.

La soluzione di tipo 3 porta con sé una gestione completa della dinamicità poiché, raggruppando sull'attributo **master**, si individuano tutte le tuple ottenute come modifica di un particolare record *r* e quelle da cui *r* è stato ottenuto. Per esempio, volendo conoscere la quantità venduta nei vari negozi indipendentemente dal cambio di nome o di responsabile, potrà essere usata la seguente interrogazione SQL:

```

SELECT DT.master, SUM(FT.quantità)
FROM VENDITE AS FT, NEGOZIO AS DT,
WHERE FT.chiaveN = DT.chiaveN
GROUP BY DT.master

```

mentre per individuare i nomi originari dei negozi da associare alle quantità calcolate si potrà utilizzare la seguente interrogazione:

```

SELECT negozio
FROM NEGOZIO
WHERE chiaveN = master

```

Avendo a disposizione lo schema descritto in precedenza è facile realizzare i differenti scenari temporali:

- *Oggi per ieri*: si individuano le tuple della dimension table attualmente valide, ossia quelle che presentano valore nullo nel campo a; per ciascuna si identificano le eventuali altre tuple da cui essa ha avuto origine e per tutte si accede alla fact table.
- *Ieri per oggi*: fissata un particolare data si individuano le tuple della dimension table che erano valide in quel momento, ossia quelle nel cui intervallo di validità ricade la data prescelta. Quindi si procede come nel caso precedente.
- *Oggi o ieri*: questo scenario può essere realizzato senza utilizzare i time-stamp poiché l'aggiornamento delle tuple nella dimension table avviene come per le gerarchie dinamiche di tipo 2.
- *Oggi e ieri*: tramite i time-stamp si individuano le tuple della dimension table che non hanno subito modifiche durante l'intervallo di tempo di interesse e, limitandosi a queste, si procede come per lo scenario *oggi o ieri*.

Si noti che nello scenario *ieri per oggi* alcuni degli eventi della fact table potrebbero rimanere non assegnati a nessuna delle tuple della dimension table nel caso in cui, all'istante prescelto per l'analisi, non esistesse ancora l'elemento della gerarchia a cui l'evento fa riferimento. Questo caso si verifica per la vendita del negozio DiTuttoDiPiù utilizzando come data di riferimento 1/11/2001: volendo conoscere le vendite imputabili ai due responsabili nel periodo 2001-2002 in base alla gerarchia valida in quella data, si otterrà infatti il risultato di Tabella 8.3.

Analogamente nel caso di scenario *oggi e ieri*, per il quale i risultati dell'analisi sono riportati in Tabella 8.4.

**Tabella 8.3** Quantità vendute imputabili ai vari responsabili in base allo scenario *ieri per oggi*: a sinistra situazione al 31/12/2001, a destra situazione al 31/12/2002. Come riferimento si utilizzano le gerarchie valide all'1/1/2001.

responsabile	anno	2001	responsabile	anno	2001	2002
Bianchi		200	Bianchi		200	200
Rossi		100	Rossi		100	100

**Tabella 8.4** Quantità vendute imputabili ai vari responsabili in base allo scenario *oggi e ieri*: a sinistra situazione al 31/12/2001, a destra situazione al 31/12/2002.

	anno	2001		anno	2001	2002
responsabile			responsabile			
Bianchi		200	Bianchi		200	200
Rossi		100	Rossi		—	—

#### 8.4.4 Cancellazione di tuple

La cancellazione è un'operazione molto rara nei DW poiché, dovendo mantenere la storia completa degli eventi di interesse, le tuple vengono normalmente aggiunte e non cancellate. Tuttavia, per non far crescere all'infinito la dimensione del database si definisce un intervallo temporale (che varia normalmente tra i tre e sette anni) considerato di interesse per l'azienda; quando una tupla della fact table fa riferimento a un evento accaduto al di fuori di questo intervallo può essere cancellata.

Per quanto riguarda le dimension table, la cancellazione di una tupla ha lo scopo di evitare che valori non più utilizzati di un attributo continuino a occupare spazio nel data mart. Prima di effettuare la cancellazione è necessario verificare che nella fact table non esistano record correlati, per evitare di violare il vincolo di integrità referenziale. Si supponga per esempio che lo star schema di Figura 8.2 memorizzi le vendite degli ultimi cinque anni e che oggi si smetta di vendere un particolare prodotto: il record della dimension table corrispondente al prodotto potrà essere eliminato solo tra cinque anni. Le operazioni di cancellazione vengono eseguite periodicamente e non sono interessanti ai fini della dinamicità delle gerarchie.

## Progettazione logica

---

La fase di progettazione logica include l'insieme di passi che, a partire dallo schema concettuale, permettono di determinare lo schema logico del data mart. Mentre durante la fase di progettazione concettuale viene ritagliata la porzione del dominio applicativo che deve essere inclusa in ciascun data mart e vengono prese decisioni su come esso verrà percepito e fruito dagli utenti, con la fase di progettazione logica si determinano le strutture dati che lo rappresentano con riferimento al modello logico prescelto e si attuano forme di ottimizzazione basate sulla taratura di tali strutture.

È importante sottolineare che la progettazione logica di data mart richiede tecniche profondamente diverse da quelle utilizzate nei sistemi operazionali, e spesso in contrasto con esse, poiché diversi sono gli obiettivi che si devono raggiungere. Mentre nei sistemi operazionali l'attenzione è incentrata sulla minimizzazione della quantità di informazioni da memorizzare e un forte vincolo è costituito dai problemi causati dalle transazioni, nei DW l'obiettivo primario è la massimizzazione della velocità di reperimento dei dati. Come si è visto nel capitolo precedente, il raggiungimento di questo obiettivo autorizza per esempio al ripetuto utilizzo di dati ridondanti e denormalizzati, tradizionalmente considerati indici di cattiva progettazione. Per realizzare un buon progetto logico di data mart è quindi fondamentale che il progettista si liberi dei retaggi derivanti dalla propria esperienza in ambito operazionale.

Sebbene le statistiche derivanti dall'analisi del mercato abbiano dimostrato che una delle principali cause di fallimento di progetti di DW sia la mancanza di un'adeguata fase di progettazione concettuale (Vassiliadis, 2000), la maggior parte delle ricerche in ambito data warehousing si è concentrata sulle problematiche legate ai livelli logico e fisico. Questo fenomeno può essere spiegato considerando che la corretta progettazione dei livelli logico e fisico determina in larga misura le prestazioni del sistema. I numerosi risultati ottenuti da un tale fervore accademico, assieme all'aumento della potenza dei calcolatori, hanno portato a una riduzione dei tempi di risposta dei sistemi impensabile fino a pochi anni fa.

Nel seguito presenteremo le fasi che, a partire dallo schema concettuale di un data mart, consentono di ottenere uno schema logico direttamente implementabile su un DBMS. I principali passi che fanno parte di questo processo sono:

- Traduzione degli schemi di fatto in schemi logici: schemi a stella, a fiocco di neve (*snowflake*), schema *constellation*.
- Materializzazione delle viste.
- Frammentazione verticale e orizzontale delle fact table.

## 9.1 Dagli schemi di fatto agli schemi a stella

Gli schemi di fatto si prestano a una traduzione diretta in schemi a stella del modello relazionale. Il passaggio dal livello concettuale al livello logico comporta una riduzione del potere espressivo e non è pertanto del tutto automatico, richiedendo l'intervento del progettista in diversi momenti.

In generale si può dire che:

uno schema di fatto può essere modellato in ambito relazionale mediante uno schema a stella in cui la fact table contiene tutte le misure e gli attributi descrittivi direttamente collegati al fatto, e per ogni gerarchia viene creata una dimension table che ne contiene tutti gli attributi.

In aggiunta a questa semplice regola, la corretta traduzione di uno schema di fatto richiede una trattazione più approfondita per i costrutti avanzati del DFM.

### 9.1.1 Attributi descrittivi

Come si è visto nel Paragrafo 5.2.1 un attributo descrittivo contiene informazioni non utilizzabili per effettuare aggregazione che si ritiene comunque utile mantenere. Un attributo descrittivo collegato a un attributo dimensionale da cui dipende funzionalmente deve essere incluso nella dimension table relativa alla gerarchia che lo contiene alla stregua di qualsiasi altro attributo dimensionale. Al contrario, nel caso in cui esso sia collegato direttamente al fatto (descrive un evento primario ma non è possibile o interessante utilizzarlo né come misura né come dimensione) deve essere incluso nella fact table assieme alle misure.

Un attributo descrittivo ha senso solo quando il livello di granularità dell'informazione che esprime è compatibile con il livello di granularità dell'evento descritto dalla fact table, quindi:

- Un attributo descrittivo collegato a un attributo dimensionale può essere inserito solo in dimension table che contengono anche l'attributo dimensionale.
- Un attributo descrittivo collegato direttamente al fatto non dovrà essere riportato nelle viste materializzate secondarie ottenute per aggregazione dei dati memorizzati nella fact table primaria.

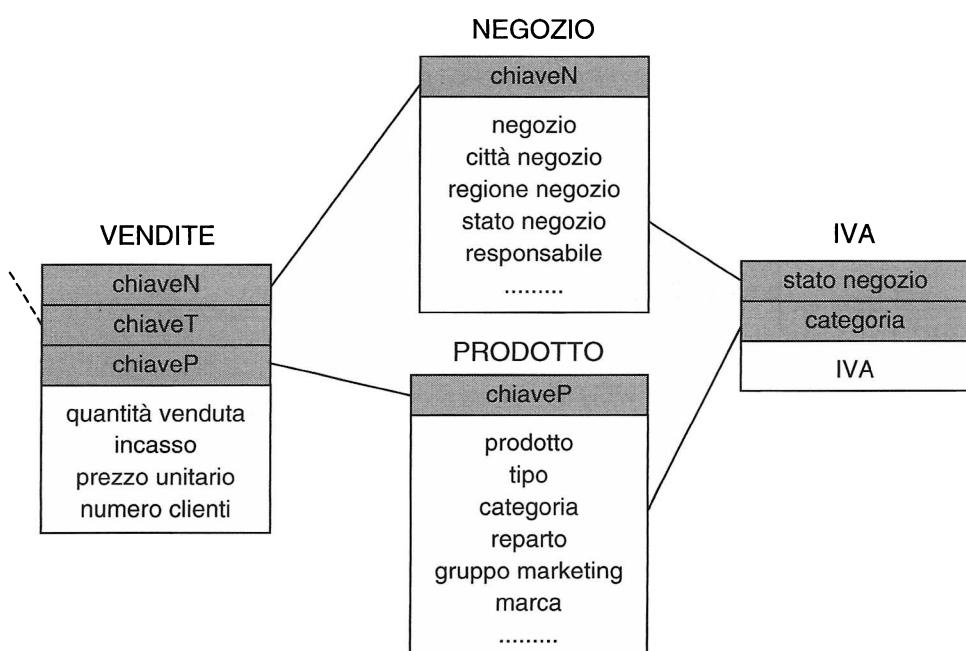
### 9.1.2 Attributi cross-dimensional

Dal punto di vista concettuale, un attributo cross-dimensionale  $b$  definisce un'associazione molti-a-molti tra due o più attributi dimensionali  $a_1, \dots, a_m$ . La sua traduzione a livello logico richiede l'inserimento di una nuova tabella che includa  $b$  e abbia come chiave gli attributi  $a_1, \dots, a_m$ . Consideriamo l'esempio di Figura 5.8, relativo alle vendite, che contiene l'attributo cross-dimensionale IVA; la Figura 9.1 riporta la porzione rilevante dello schema logico necessario a modellare questo elemento.

Si noti che nella tabella IVA sono stati importati direttamente i campi stato negozio e categoria senza introdurre chiavi surrogate. L'utilizzo di chiavi surrogate può essere una valida soluzione quando la cardinalità e la lunghezza degli attributi coinvolti nella definizione del dato cross-dimensionale è tale da compensare la quantità extra di spazio richiesta per memorizzare le chiavi surrogate nelle dimension table. Per esempio nel caso dell'IVA, supponendo di utilizzare chiavi surrogate di 4 byte e di gestire 5000 prodotti, 1000 negozi, 20 stati (lunghezza del campo 20 byte) e 30 categorie (lunghezza del campo 20 byte), lo spazio richiesto per modellare l'attributo cross-dimensionale utilizzando chiavi surrogate è pari al massimo a:

$$\begin{aligned} &\text{spazio chiavi surrogate DT} + \text{spazio chiavi surrogate CDT} \\ &\quad = 5000 \times 4 + 1000 \times 4 + 20 \times 30 \times (4+4) = 28\,800 \text{ byte} \end{aligned}$$

mentre importando direttamente i valori dei campi coinvolti si occupano  $20 \times 30 \times (20+20) = 24\,000$  byte.

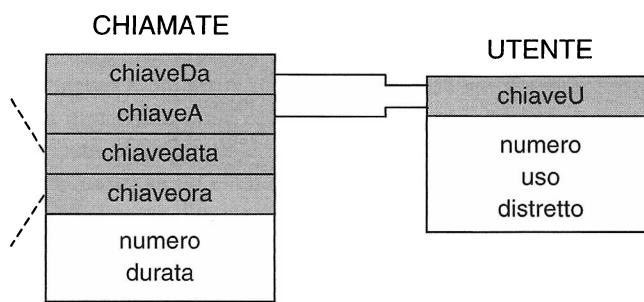


**Figura 9.1** Modellazione dell'attributo cross-dimensionale IVA.

### 9.1.3 Gerarchie condivise

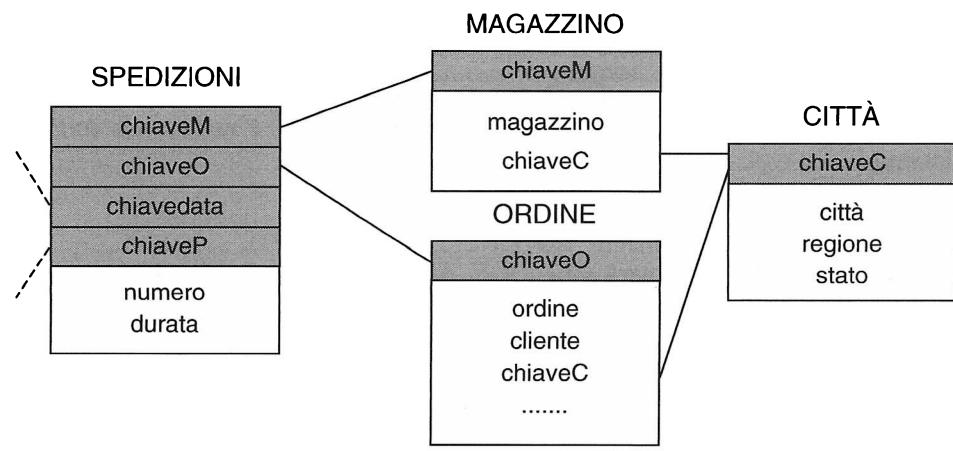
Come evidenziato nel Paragrafo 5.2.4 si verifica spesso che uno schema di fatto presenta gerarchie, o porzioni di gerarchie, ripetute. In questo caso a livello logico non è ovviamente consigliabile introdurre più dimension table che contengano del tutto o in parte gli stessi dati. Dal punto di vista progettuale esistono due soluzioni per due situazioni distinte.

- Primo caso: due gerarchie contengono esattamente gli stessi attributi utilizzati con significati diversi. Nell'esempio di Figura 5.13 questo caso si verifica per il fatto **CHIAMATA** per il quale la gerarchia del chiamante è identica a quella del chiamato. È allora sufficiente importare due volte la chiave dell'unica dimension table che modella gli utenti come mostrato in Figura 9.2; inserendo nella tupla della fact table due valori distinti della chiave della dimension table si modellerà una chiamata tra due utenti.
- Secondo caso: due gerarchie condividono solo una parte degli attributi. Nell'esempio di Figura 5.14 questo caso si verifica per il fatto **SPEDIZIONE** in cui le gerarchie **magazzino** e **ordine** condividono solo la sequenza di attributi che codifica le informazioni geografiche relative alla sede del magazzino e alla residenza del cliente. In questo caso è necessario decidere se introdurre ulteriore ridondanza nello schema logico inserendo i campi comuni in entrambe le dimension table che modellano le due gerarchie, oppure se eseguire uno snowflake sul primo attributo condiviso introducendo una terza tabella comune a entrambe le dimension table come mostrato in Figura 9.3. L'applicazione del modello snowflake va valutata considerando da un lato il risparmio in termini di spazio occupato e dall'altro il maggior costo di interrogazione dovuto alla necessità di eseguire un ulteriore join.



**Figura 9.2** Modellazione delle gerarchie condivise chiamante-chiamato nell'esempio delle chiamate telefoniche.

La soluzione proposta si applica anche al caso di gerarchie conformi, ossia quando la stessa gerarchia è presente in più fatti. Anche in questo caso, se parte dei dati delle due gerarchie è comune, non è opportuno a livello logico realizzare due dimension table disgiunte: tutti i dati saranno così inclusi nella stessa tabella la cui chiave potrà essere importata indistintamente dalle diverse fact table.



**Figura 9.3** Modellazione della gerarchia geografica parzialmente condivisa nell'esempio delle spedizioni.

#### 9.1.4 Archi multipli

Come si è visto nei Paragrafi 5.2.5 e 5.6.4, sebbene piuttosto inusuale, le gerarchie possono codificare anche associazioni del tipo molti-a-molti. A livello logico esistono diverse soluzioni progettuali per modellarle.

La soluzione più ovvia è probabilmente quella proposta da Kimball e altri (1998) che prevede l'inserimento di una tabella aggiuntiva, detta *bridge table*, per modellare gli archi multipli. La chiave della bridge table è composta dalla combinazione degli attributi collegati dall'arco multiplo. Si consideri l'esempio di Figura 9.4 che fornisce la modellazione a livello logico dello schema presentato in Figura 5.15. Come si può notare la tabella BRIDGE\_AUTORE consente di identificare correttamente tutti gli autori di ogni libro, inoltre grazie all'attributo peso anche le aggregazioni sui singoli autori potranno essere eseguite correttamente. Si ricorda che i pesi dei singoli elementi coinvolti nell'arco multiplo devono essere normalizzati, ossia la loro somma deve dare 1. In Figura 9.5 è riportata la forma estensionale dello schema proposto utilizzando i dati presentati nelle Tabelle 5.17 e 5.19.

La bridge table permette di codificare per esempio il fatto che il libro con chiave 1 è stato scritto dall'autore con chiave 1 e da quello con chiave 2, e che quest'ultimo ha partecipato anche alla stesura dei libri con chiave 3 e 4. L'interrogazione OLAP che calcola gli incassi complessivi distribuiti ai diversi autori è riportata di seguito e, come si può notare, richiede tre join per recuperare i nominativi degli scrittori, mentre in uno schema a stella standard è sufficiente un solo join per recuperare tutti i dati di una gerarchia.

```

SELECT      AUTORE.autore, sum(VENDITE.incasso * BRIDGE_AUTORE.peso)
FROM        AUTORE, BRIDGE_AUTORE, LIBRO, VENDITE
WHERE       AUTORE.chiaveA = BRIDGE_AUTORE.chiaveA
AND         BRIDGE_AUTORE.chiaveL = LIBRO.chiaveL
  
```

```
AND      LIBRO.chiaveL = VENDITE.chiaveL
GROUP BY AUTORE.autore
```

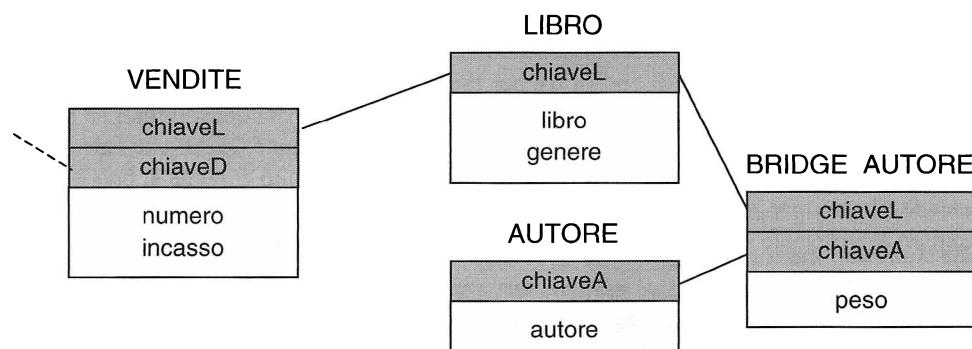
Non necessariamente l'aggregazione delle vendite deve essere pesata. Se per esempio si vuole calcolare il numero di libri venduti per ciascun autore, è ragionevole formulare l'interrogazione così:

```
SELECT  AUTORE.autore, sum(VENDITE.numero)
FROM    AUTORE, BRIDGE_AUTORE, LIBRO, VENDITE
WHERE   AUTORE.chiaveA = BRIDGE_AUTORE.chiaveA
AND     BRIDGE_AUTORE.chiaveL = LIBRO.chiaveL
AND     LIBRO.chiaveL = VENDITE.chiaveL
GROUP BY AUTORE.autore
```

Con la soluzione tramite bridge table sono quindi possibili due tipi di interrogazioni:

- 1) *Interrogazioni pesate*: viene considerato il peso dell'arco multiplo, e si ottiene pertanto l'effettivo totale.
- 2) *Interrogazioni di impatto*: il peso non viene considerato e conseguentemente le quantità risultanti sono più alte.

Nel caso in cui si voglia evitare l'utilizzo di schemi diversi da quello a stella, per esempio perché il DBMS utilizzato non supporta correttamente le bridge table, è necessario rendere più fine la granularità del fatto al fine di modellare direttamente nella fact table l'associazione molti-a-molti corrispondente all'arco. L'adozione di questa soluzione richiede l'aggiunta alla fact table di una nuova dimensione corrispondente all'attributo terminale *a* dell'arco multiplo, il che dal punto di vista concettuale corrisponde a effettuare il push-down di *a* (si veda il Paragrafo 5.6.4). Eventuali attributi figli di *a* formeranno la sua gerarchia e saranno pertanto memorizzati nella nuova dimension table. Lo schema logico ottenuto effettuando il push-down di autore è illustrato in Figura 9.6 attraverso la sua istanza corrispondente ai valori in Tabella 5.17. Si noti come, aggregando su chiaveL e chiaveD, sia possibile ricostruire gli eventi primari di Figura 9.5.



**Figura 9.4** Modellazione dell'arco multiplo libro-autore tramite bridge table.

---

LIBRO	<u>chiaveL</u>	libro	genere	
	1	Il DFM	tecnico	
	2	Mi Sembra Logico	tecnico	
	3	La Giusta Misura	attualità	
	4	Un Fatto Come e Perché	attualità	
	5	La Quarta Dimensione	fantascienza	
AUTORE	<u>chiaveA</u>	autore		
	1	Matteo Golfarelli		
	2	Stefano Rizzi		
BRIDGE_AUTORE	<u>chiavel</u>	<u>chiaveA</u>	peso	
	1	1	0,5	
	1	2	0,5	
	2	1	1,0	
	3	2	1,0	
	4	1	0,5	
	4	2	0,5	
VENDITE	<u>chiavel</u>	<u>chiaveD</u>	numero	incasso
	1	1	3	150
	2	1	5	250
	3	1	10	300
	4	1	4	80
	5	1	8	400

---

**Figura 9.5** Una possibile istanza per lo schema di Figura 9.4.

---

VENDITES	<u>chiavel</u>	<u>chiaveA</u>	<u>chiaveD</u>	numero	incasso
	1	1	1	1,5	75
	1	2	1	1,5	75
	2	1	1	5	250
	3	2	1	10	300
	4	1	1	2	40
	4	2	1	2	40
	5	1	1	8	400

---

**Figura 9.6** Un'istanza della fact table delle vendite di libri avendo effettuato il push-down di autore.

Esaminiamo ora il caso in cui l'arco multiplo entri in una dimensione. In Figura 9.7 è presentata un'istanza della fact table per il fatto dei ricoveri di Figura 5.33, con riferimento ai dati nelle Tabelle 5.21 e 5.22 e utilizzando la bridge table BRIDGE\_DIAGNOSI per modellare l'associazione molti-a-molti tra gruppo di diagnosi e diagnosi.

---

DIAGNOSI	
	<u>kDiagnosi</u>
1	cardiopatia
2	ipertensione
3	asma
4	frattura

RICOVERI	
	<u>kGruppo</u>
1	1
2	1

BRIDGE_DIAGNOSI		
	<u>kGruppo</u>	<u>kDiagnosi</u>
1	1	1
1	1	2
1	1	3
2	2	3
2	2	4

---

**Figura 9.7** Modellazione tramite bridge table degli eventi in Tabella 5.21.

La Figura 9.8 illustra la corrispondente soluzione basata sul push-down di diagnosi. L'attributo kGruppo non è ora più necessario, poiché una terna data, reparto, paziente identifica univocamente un ricovero e quindi un gruppo di diagnosi permettendo la ricostruzione degli eventi primari originari.

---

RICOVERI	
	<u>kDiagnosi</u>
1	1
2	1
3	1
3	1
4	1

RICOVERI				
	<u>kData</u>	<u>kReparto</u>	<u>kPaziente</u>	costo
1	1	1	1	500
2	1	1	1	300
3	1	1	1	200
3	1	1	2	600
4	1	1	2	1400

---

**Figura 9.8** Un'istanza della fact table dei ricoveri avendo effettuato il push-down di diagnosi.

È a questo punto possibile una valutazione della soluzione basata su push-down effettuandone il raffronto con quella basata su bridge table:

- Il potere informativo delle due soluzioni è identico: per passare dagli eventi primari di una soluzione a quelli dell'altra è sufficiente aggregarli in un caso, pesarli nell'altro.
- La soluzione push-down introduce una forte ridondanza nella fact table, le cui righe devono essere replicate tante volte quante sono le corrispondenze dell'arco multiplo. Per esempio, nella fact table dei ricoveri si passa dalle due righe di Figura 9.7 alle 5 righe di Figura 9.8.
- Nella soluzione push-down, il peso con cui ogni singola tupla contribuisce a un gruppo è codificato permanentemente all'interno della fact table e non può essere modificato a meno di innescare costose operazioni di aggiornamento. Al contrario, la soluzione tramite bridge table memorizza esplicitamente e in modo non ridondante i pesi, che possono quindi essere modificati più agevolmente o addirittura non presi in considerazione nel caso in cui esigenze estemporanee lo richiedano.
- Mentre la soluzione con bridge table supporta indifferentemente interrogazioni pesate e d'impatto, nella soluzione con push-down la formulazione di interrogazioni d'impatto risulta più complessa.
- Il calcolo degli eventi primari pesati avviene durante l'alimentazione nella soluzione con push-down, durante l'interrogazione nella soluzione con bridge table.

Di conseguenza,

le due soluzioni proposte sono tecnicamente sempre applicabili in presenza di un arco multiplo. Rispetto alla soluzione con bridge table, nella soluzione push-down il costo di esecuzione delle interrogazioni da un lato si riduce poiché vengono eliminati dei join, dall'altro aumenta a causa dell'aumentata cardinalità della fact table. Perciò la scelta tra le due soluzioni va valutata di volta in volta.

In particolare, la soluzione con bridge table è praticamente obbligatoria quando l'arco multiplo non è connesso direttamente al fatto; è fortemente consigliata anche quando, sebbene l'arco sia collegato direttamente al fatto, il numero di corrispondenze tra i due attributi coinvolti è molto elevato (ossia quando l'arco modella un'associazione molti-a-molti in cui la maggior parte degli elementi di un attributo sono collegati con la maggior parte degli elementi dell'altro). In questo caso, infatti, la soluzione con push-down potrebbe rendere eccessiva la cardinalità della fact table.

In letteratura sono state proposte altre soluzioni che trovano giustificazione solo in casistiche molto limitate (per esempio se la cardinalità massima dell'attributo terminale dell'arco multiplo è nota e limitata): per questo motivo non riteniamo utile presentarle in questo testo rimandando il lettore all'articolo di Song e altri (2001) per ulteriori dettagli.

### **9.1.5 Archi opzionali**

Si è visto nel Paragrafo 5.2.6 come alcune porzioni delle gerarchie possano essere definite come opzionali. Per modellare questa situazione non si incide sulla struttura della corrispondente dimension table, in cui l'attributo continua a comparire dato che per una o più

delle istanze esso sarà valorizzato, bensì sui valori da assegnare alle istanze anomale. Nello schema relazionale, le istanze per cui non è definito alcun valore devono essere denotate introducendo un valore fittizio. La soluzione più semplice è quella di utilizzare il valore NULL; però, dato che il valore NULL può assumere anche significati diversi (per esempio valore sconosciuto), è consigliato introdurre un valore fittizio supplementare.

Si noti come, a causa dei vincoli di integrità primaria e referenziale<sup>1</sup>, l'opzionalità di una intera gerarchia non possa essere gestita direttamente nella fact table introducendo un valore nullo o fittizio nella chiave esterna corrispondente alla gerarchia opzionale, ma richieda di aggiungere un'intera tupla fittizia nella dimension table. Questa considerazione si applica anche alle dimension table secondarie degli schemi snowflake.

### 9.1.6 Gerarchie incomplete

Nelle gerarchie incomplete, descritte nel Paragrafo 5.2.7, per alcune istanze risultano assenti uno o più livelli di aggregazione. Come nel caso degli archi opzionali, la loro modellazione logica si ottiene operando a livello estensionale mediante l'inserimento di valori fittizi. Questo caso però risulta più complicato poiché la mancanza di un valore a un certo livello di aggregazione non implica la mancanza di valori ai livelli più aggregati e di conseguenza può essere necessario mantenere la consistenza dell'operazione di roll-up. Riprendendo l'esempio di Figura 5.18, le misure relative al comune di Serravalle, ad esempio il numero di abitanti, non possono essere aggregate rispetto agli attributi provincia e regione poiché la Repubblica di San Marino non prevede tale suddivisione. Tuttavia, volendo contare gli abitanti delle diverse regioni per tutta l'Europa, è necessario fare comparire anche quelli di San Marino poiché altrimenti si perderebbe la consistenza dei valori rispetto a una statistica effettuata a livello di nazioni. In linea con quanto detto nel Paragrafo 5.6.7 sono possibili tre diverse soluzioni, che si differenziano per il valore inserito come segnaposto nell'attributo non definito:

- **Bilanciamento per esclusione:** in tutte le tuple delle dimension table i valori mancanti vengono rimpiazzati con un unico segnaposto generico, per esempio con valore “altro”. Questa soluzione è preferibile nel caso in cui molte istanze presentino valori non specificati poiché l'interpretazione del risultato sarà facilitata, tuttavia, come già evidenziato nel Paragrafo 5.6.7, va sottolineato che si sta violando la semantica del roll-up poiché *un'ulteriore aggregazione* dei dati porterà ad avere un *maggior* livello di dettaglio dei risultati; riprendendo l'esempio precedente, se si passa dal livello di aggregazione regione a quello di nazione l'unica riga che riporta il numero degli abitanti di comuni (anche di nazioni diverse) che non prevedono una classificazione in regione verrà sostituita da più righe relative alle singole nazioni.
- **Bilanciamento verso il basso:** in tutte le tuple delle dimension table i valori mancanti vengono rimpiazzati con il valore del corrispondente attributo che lo precede nella ge-

---

<sup>1</sup> Il vincolo di integrità primaria del modello relazionale impedisce che un attributo facente parte di una chiave primaria assuma valori nulli. Il vincolo di integrità referenziale asserisce che, data una chiave *a* esterna che riferisce una chiave primaria *k*, i valori ammissibili per *a* sono un sottoinsieme del dominio di *k*.

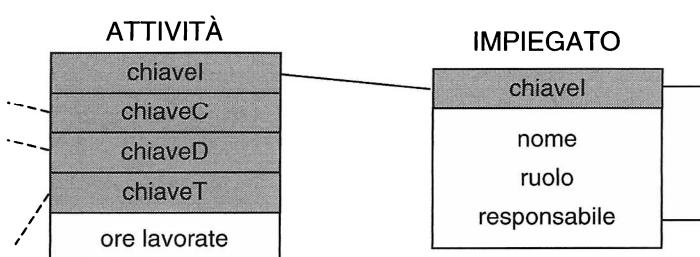
rarchia (per esempio per l'attributo provincia il valore dell'attributo comune). Se da un lato questa soluzione complica l'interpretazione dei risultati, dato che a un certo livello di aggregazione compariranno anche etichette di livelli inferiori, dall'altro fornisce informazioni più dettagliate e non viola la semantica del roll-up. Con riferimento all'esempio precedente, aggregando il numero di abitanti per province verranno visualizzate anche le etichette di quei comuni per i quali non è definita alcuna provincia. Questa soluzione è preferibile quando la quantità di dati mancanti è molto limitata.

- **Bilanciamento verso l'alto:** si opera come nel caso precedente inserendo però come segnaposto il valore dell'attributo che segue nella gerarchia (per esempio per l'attributo regione il valore dell'attributo nazione). In questo caso, utilizzando segnaposto più aggregati, il numero di valori restituiti sarà minore rispetto alla soluzione precedente portando a un output più leggibile in presenza di molti dati mancanti.

Diversi tool OLAP forniscono oggi un supporto esplicito alle gerarchie incomplete consentendo inoltre di scegliere quale delle diverse soluzioni proposte in precedenza mostrare all'utente. Per ottenere questo risultato sarà necessario associare a ogni attributo per il quale potrebbero presentarsi dei dati mancanti un attributo booleano che identifichi le istanze in cui è presente un valore segnaposto. In questo modo, in fase di interrogazione, il sistema OLAP potrà collassare assieme tutti i valori segnaposto, oppure aggregarli sulla base del valore del segnaposto stesso.

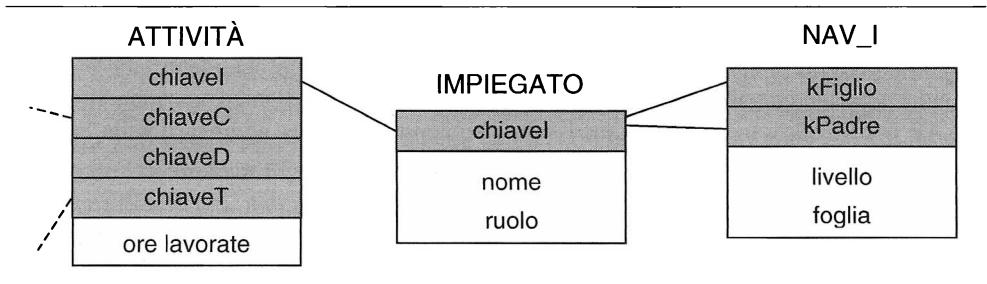
### 9.1.7 Gerarchie ricorsive

Come descritto nel Paragrafo 5.2.8, nelle gerarchie ricorsive il numero dei livelli di aggregazione non è codificabile nello schema e inoltre può variare da istanza a istanza. Questa caratteristica impedisce l'utilizzo della soluzione di base del modello a stella, che prevedrebbe di denormalizzare la gerarchia in un'unica dimension table in cui a ogni livello della prima corrisponde un attributo nella seconda. Una possibile implementazione relazionale è quella proposta in Figura 9.9, in cui la ricorsione è modellata tramite un autoanello che consente di rappresentare un numero variabile, e potenzialmente illimitato, di livelli.

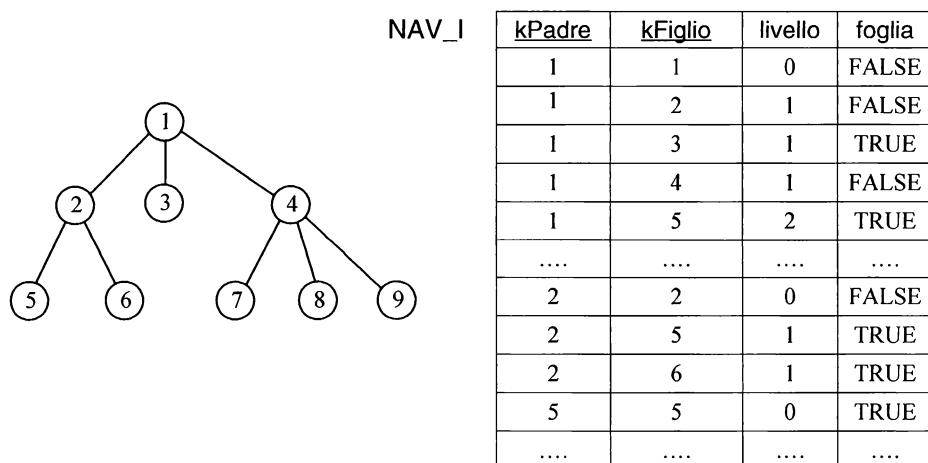


**Figura 9.9** Modellazione della gerarchia ricorsiva nell'esempio delle attività di commessa di Figura 5.22.

La soluzione prospettata nasconde diversi problemi a causa dei limiti del linguaggio SQL nel gestire le interrogazioni ricorsive. Molti dei DBMS commerciali infatti non le supportano, e anche dove ciò è possibile (Oracle, 2003) la loro gestione pone diverse limitazioni. Una soluzione alternativa, e di fatto più potente, è quella proposta da Kimball e altri (1998) ed esemplificata in Figura 9.10. L'idea è quella di "appiattire" la gerarchia esplicitando tutti i legami da essa indotti in una *tabella di navigazione* che modella un'associazione molti-a-molti tra la fact table e la dimension table. In Figura 9.11 è mostrata una possibile istanza di gerarchia e un frammento del corrispondente contenuto di NAV\_I: come si nota, per il nodo 1 sono riportati discendenti sia diretti che indiretti e per ognuno è specificato il relativo livello di profondità rispetto all'attributo kPadre, inoltre il flag foglia indica se il nodo identificato da kFiglio ha ulteriori discendenti; per poter accedere anche agli eventi che sono in relazione diretta con il relativo elemento dimensionale è necessario inserire un riferimento di livello 0 in cui padre e figlio coincidono.



**Figura 9.10** Modellazione della gerarchia ricorsiva tramite tabella di navigazione nell'esempio delle attività di commessa.



**Figura 9.11** Istanza di gerarchia e sua rappresentazione tramite tabella di navigazione.

Ovviamente la dimensione della tabella di navigazione ha una crescita esponenziale rispetto alla profondità della gerarchia (per modellare la gerarchia di Figura 9.11 sono necessarie 22 tuple); tuttavia, se la dimensione è trattabile la soluzione permette di navigarne efficacemente i diversi livelli. In particolare, dato uno specifico impiegato è possibile identificare tutti gli eventi relativi ai suoi discendenti tramite il predicato:

**ATTIVITÀ.Chiavel=NAV\_I.kFiglio AND NAV\_I.kPadre=IMPIEGATO.Chiavel**

Possono inoltre essere identificati quelli dei discendenti entro uno specifico livello di discendenza ( $livello < val$ ), o ancora quelli relativi ai discendenti al livello più basso (foglia = TRUE). Invertendo la catena di join tramite l'applicazione del predicato:

**ATTIVITÀ.Chiavel=NAV\_I.kPadre AND NAV\_I.kFiglio=IMPIEGATO.Chiavel**

è possibile risalire la gerarchia individuando di conseguenza tutti i superiori di un certo impiegato. Infine, visto che tutti gli eventi della fact table sono comunque associati a un elemento della dimension table, è sempre possibile mettere in join direttamente le due tabelle escludendo così la tabella di navigazione nel caso in cui le informazioni sulla gerarchia non siano di interesse.

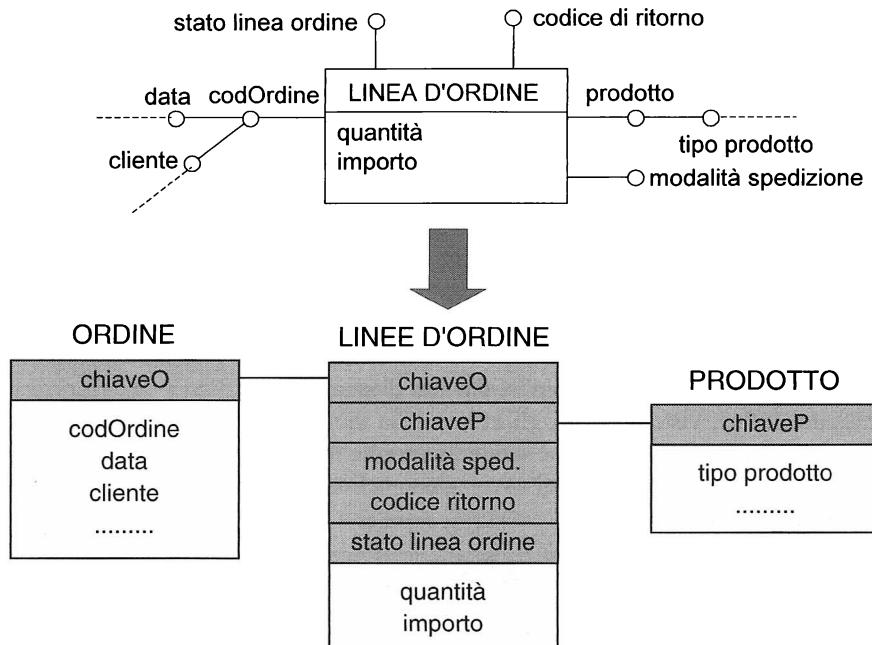
Rispetto alla modellazione con autoanello, la soluzione con tabella di navigazione garantisce un maggiore potere espressivo in fase di interrogazione e inoltre è supportata da qualsiasi strumento OLAP poiché le operazioni di roll-up richiedono il solo utilizzo della clausola di GROUP BY. Tuttavia, la crescita esponenziale del numero delle tuple ne impedisce l'utilizzo per gerarchie di grandi dimensioni. Il limite di trattabilità di una gerarchia dipende dalla sua struttura e dal DBMS utilizzato ed è quindi oggetto di studio dello specifico progetto.

### 9.1.8 Dimensioni degeneri

Il termine *dimensione degenera* indica una dimensione la cui gerarchia contiene un solo attributo (Kimball, 1996). Per le dimensioni degeneri può convenire derogare alla regola generale di traduzione al fine di evitare la riduzione delle prestazioni dovuta alla proliferazione di chiavi esterne nella fact table e al conseguente aumento del numero di join necessari a rispondere alle interrogazioni. In pratica,

risulta conveniente creare una dimension table per una dimensione degenera solo se la cardinalità del suo dominio è modesta e la lunghezza in byte dell'unico attributo che contiene è invece elevata rispetto a quella di una chiave surrogata: in questo caso, infatti, ciascun valore verrà memorizzato una sola volta nella dimension table mentre la chiave surrogata verrà duplicata nella fact table. In tutti gli altri casi, è preferibile memorizzare le dimensioni degeneri direttamente all'interno della fact table includendole anche nella chiave composta.

La Figura 9.12 riporta uno schema di fatto per i dettagli d'ordine e il corrispondente schema a stella contenente tre dimensioni degeneri incluse direttamente nella fact table: a fronte di un probabile aumento delle dimensioni della fact table, non sarà necessario alcun join per reperire le informazioni relative alle dimensioni.

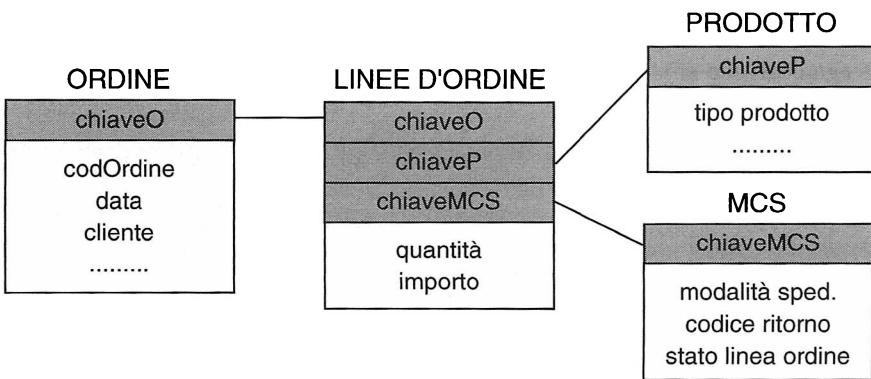


**Figura 9.12** Modellazione delle dimensioni degeneri *modalità spedizione*, *stato linea ordine*, *codice di ritorno* per il fatto **LINEA D'ORDINE**.

Una soluzione alternativa al problema delle dimensioni degeneri contempla l'uso delle cosiddette *junk dimension* (Kimball, 1996). Una junk dimension è un'unica dimension table che include un insieme di dimensioni degeneri riducendo così il numero di chiavi da importare nella fact table. Al contrario delle normali dimension table, in una junk dimension non esiste alcuna dipendenza funzionale tra gli attributi per cui risultano valide tutte le possibili combinazioni dei valori: si ha cioè un elevato numero di tuple. Questa soluzione risulta pertanto attuabile solo qualora il numero delle combinazioni lecrite per gli attributi partecipanti non sia eccessivo, o comunque quando la cardinalità dei domini sia tale da non rendere proibitiva la cardinalità del loro prodotto cartesiano.

I principali candidati a essere raggruppati in una junk dimension sono i *flag* che indicano i possibili stati di un evento primario.

La Figura 9.13 illustra la soluzione che utilizza la junk dimension MCS per modellare le dimensioni degeneri del fatto di Figura 9.12. Non essendo definita alcuna dipendenza funzionale tra gli attributi coinvolti, tutte le combinazioni di valori sono possibili. Una possibile estensione della dimension table MCS per l'esempio di Figura 9.13 è riportata in Figura 9.14; supponendo che l'attributo *modalità sped.* possa assumere tre valori ('Posta', 'Aereo', 'Corriere'), l'attributo *codice ritorno* due ('0', '1') e l'attributo *stato linea d'ordine* tre ('Eseguito', 'In corso', 'Annulloato'), la tabella conterrà in totale  $3 \times 2 \times 3 = 18$  tuple.



**Figura 9.13** Modellazione della junk dimension che ingloba le dimensioni degeneri modalità spedizione, stato linea ordine, codice di ritorno.

MCS	<u>chiaveMCS</u>	modalità sped.	codice ritorno	stato linea ordine
	1	Posta	0	Eseguito
	2	Aereo	0	Eseguito
	3	Corriere	0	Eseguito
	4	Posta	1	Eseguito
	5	Aereo	1	Eseguito
	6	Corriere	1	Eseguito
	7	Posta	0	In corso
	8	Aereo	0	In corso
	9	Corriere	0	In corso
	10	Posta	1	In corso
	11	Aereo	1	In corso
	12	Corriere	1	In corso
	13	Posta	0	Annulloato
	.....	.....	.....	.....

**Figura 9.14** Un'istanza della junk dimension MCS.

### 9.1.9 Problemi connessi all'additività

Oltre alle considerazioni sin qui fatte, nel definire la struttura di una vista materializzata a partire da uno schema di fatto è necessario valutare i problemi indotti dagli operatori di aggregazione. Come si è già detto nel Paragrafo 8.3, per poter calcolare dati aggregati a partire da dati parzialmente aggregati è necessario valutare la natura dell'operatore da im-

## Progettazione dell'alimentazione

---

Durante la fase di progettazione dell'alimentazione vengono definite le procedure necessarie a caricare all'interno del data mart i dati provenienti dalle sorgenti operazionali. In presenza del livello riconciliato il processo di alimentazione risulta suddiviso in due fasi:

- *Dalle sorgenti operazionali al livello riconciliato:* è la fase più complessa poiché prevede la definizione delle procedure che, agendo sui dati, realizzano a livello estensionale le trasformazioni definite durante l'integrazione (Capitolo 3). Inoltre, vengono progettate le procedure per la pulizia dei dati (*data cleaning*).
- *Dal livello riconciliato al livello del data mart:* in questa fase vengono definite le procedure che permettono di conformare i dati memorizzati nel livello riconciliato agli schemi a stella utilizzati in ambito multidimensionale. Oltre alle procedure per la de-normalizzazione dei dati e per l'inserimento delle chiavi surrogate (Capitolo 8), si definiscono quelle per il calcolo di dati derivati non presenti a livello operazionale. Infatti, gli aggiornamenti ai dati devono essere propagati non solo agli schemi a stella relativi a viste primarie (dati elementari) ma anche a quelli relativi a viste secondarie (dati aggregati).

Come già sottolineato nel Capitolo 3, si ritiene preferibile la soluzione architetturale a tre livelli poiché l'alimentazione diretta dei data mart è un compito troppo complesso per essere eseguito in modo atomico: la presenza di uno stadio intermedio facilita il compito del gruppo di progettazione, semplifica le procedure di alimentazione e pulizia dei dati ed evita che lo schema integrato delle sorgenti rimanga criptato all'interno delle logiche delle procedure di ETL.

I dati riconciliati sono intesi come una versione integrata e normalizzata del database operazionale e ne conservano le stesse caratteristiche in fatto di granularità delle informazioni. Il livello di storicizzazione dei dati è uguale o superiore a quello del database operazionale.

Il livello riconciliato rappresenta il punto di arrivo della fase di integrazione e di pulizia dei dati; sia lo schema riconciliato sia i dati in esso contenuti dovrebbero essere consistenti e privi di errori. È importante sottolineare che lo schema riconciliato deve essere creato, almeno virtualmente, anche nelle soluzioni a due livelli poiché esso dimostra che il proget-

tista ha definitivamente acquisito una precisa conoscenza sul sistema operazionale, indispensabile per realizzare una corretta alimentazione del DW che, in questo caso, verrà effettuata direttamente a partire dalle sorgenti.

## 10.1 Alimentazione dello schema riconciliato

La fase di analisi e riconciliazione dei dati ha reso disponibili gli schemi (concettuali e/o logici) del livello riconciliato e delle sorgenti e ha definito la corrispondenza tra i due livelli. La fase di progettazione dell’alimentazione dello schema riconciliato, di cui ci occuperemo in questo paragrafo, permette di rendere operativa sui dati questa corrispondenza definendo, per ogni concetto presente nello schema riconciliato, le procedure necessarie a svolgere le seguenti funzioni:

- *Estrazione*: indica le operazioni che permettono di acquisire i dati dalle sorgenti.
- *Trasformazione*: indica le operazioni che conformano i dati delle sorgenti allo schema riconciliato.
- *Caricamento*: indica le operazioni necessarie a inserire i dati trasformati nel database riconciliato aggiornando eventualmente quelli già presenti.

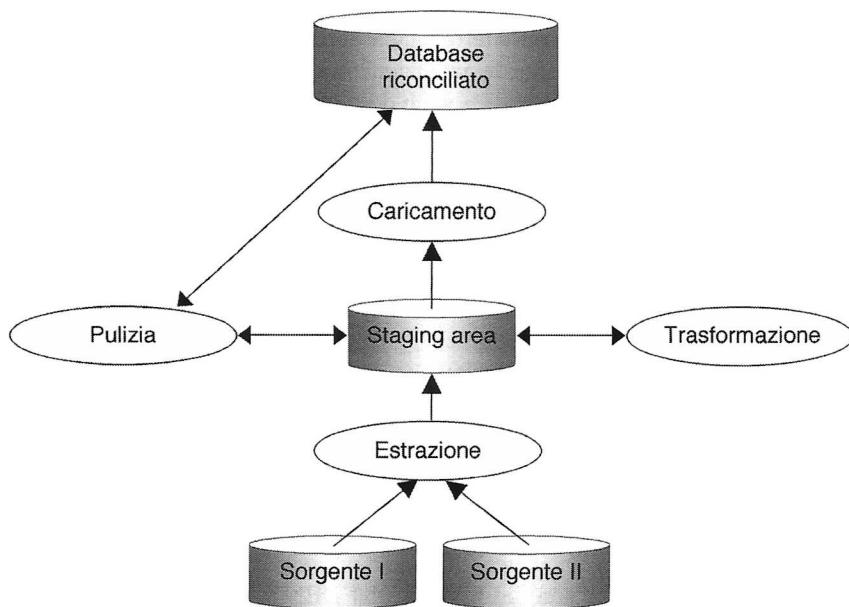
A queste tre operazioni è da affiancare la definizione delle procedure di *pulizia dei dati* che servono a eliminare le incongruenze dovute a errori e omissioni. Come si può notare in Figura 10.1, l’alimentazione del livello riconciliato utilizza in genere un’area di memorizzazione temporanea (*staging area*) contenente i dati “grezzi” a cui applicare le procedure di trasformazione e pulizia. La staging area può richiedere molto spazio e il suo dimensionamento non deve essere sottovalutato dal progettista.

Sebbene nell’approccio GAV (Paragrafo 3.4) ogni singolo concetto dello schema riconciliato sia definito come una vista sugli schemi sorgenti e possa quindi essere alimentato singolarmente, le operazioni di alimentazione vengono in realtà svolte contemporaneamente per più concetti sfruttando la staging area come buffer temporaneo in modo da minimizzare il costo degli accessi alle sorgenti.

Si noti infine che le tecniche di pulizia dei dati vengono talvolta applicate anche dopo il caricamento nel livello riconciliato al fine di estendere i confronti alla più ampia quantità di dati possibile. Questa possibilità, disponibile solo se il database riconciliato è stato effettivamente materializzato, permette di aumentare considerevolmente il livello di qualità dei dati.

### 10.1.1 L’estrazione dei dati

Le operazioni di estrazione vengono eseguite all’atto dell’inizializzazione del livello riconciliato per essere poi ripetute periodicamente, in base all’intervallo di aggiornamento stabilito dal progettista, al fine di acquisire le informazioni relative agli eventi verificatisi durante la vita del sistema.



**Figura 10.1** Il processo di alimentazione del livello riconciliato.

La definizione delle procedure di estrazione inizia individuando la natura dei dati presenti nelle sorgenti, che viene detta *transitoria* se il sistema operazionale ne mantiene solo l'immagine corrente sovrascrivendo i dati che non sono più validi, *semi-storicizzata* se il sistema mantiene un limitato numero degli stati precedenti e non è possibile determinare per quanto tempo ciascun dato verrà conservato nel sistema. Infine, le sorgenti operazionali sono dette *storicizzate* quando tengono traccia, per un intervallo di tempo ben definito, di tutte le modifiche intervenute sui dati. Per esempio, sono normalmente transitori i dati di inventario quali i livelli delle scorte di magazzino, che vengono sovrascritti all'arrivo della nuova merce; sono invece semi-storicizzati i dati riguardanti gli stati di un sistema di cui vengono mantenute le ultime  $n$  versioni. Il momento in cui lo stato meno recente viene sovrascritto dipende dal verificarsi dell' $(n+1)$ -esimo evento che varia lo stato del sistema, e non può essere previsto a priori. Infine, sono normalmente storicizzati i dati in ambito assicurativo e bancario, in cui le informazioni relative ai conti correnti e alle polizze devono essere mantenute per un lasso di tempo determinato da precisi obblighi di legge. Ovviamente i problemi nascono in presenza di sorgenti transitorie o semi-storicizzate poiché per esse dovranno essere definiti dei meccanismi in grado estrarre i dati indipendentemente dalla brevità del loro intervallo di vita.

L'approccio più semplice all'estrazione consiste nella cosiddetta *estrazione statica* che prevede la scansione completa di tutti i dati presenti nelle sorgenti operazionali. Questa soluzione, obbligatoria all'atto dell'inizializzazione, può essere impiegata ogniqualvolta la ridotta dimensione dei dati permetta di eseguirla in tempi accettabili. Il volume dei dati coinvolto ne limita tuttavia l'impiego nella maggior parte dei casi reali in cui gli aggiornamenti devono essere eseguiti in un ridotto lasso di tempo (una notte, un week-end).

Essendo la frazione dei dati modificati in uno specifico intervallo di tempo di diversi ordini di grandezza inferiore rispetto all'intero database, risultano molto più efficaci le tecniche che permettono di determinarne le variazioni limitando così la porzione di dati da leggere. Gli approcci basati su questa idea sono detti di *estrazione incrementale* e possono essere sommariamente classificati in *immediati* e *ritardati*. I primi registrano la modifica nei dati nel momento in cui essa viene registrata dal sistema operazionale, i secondi invece posticipano tale operazione. Devlin (1997) ha definito cinque diverse tecniche:

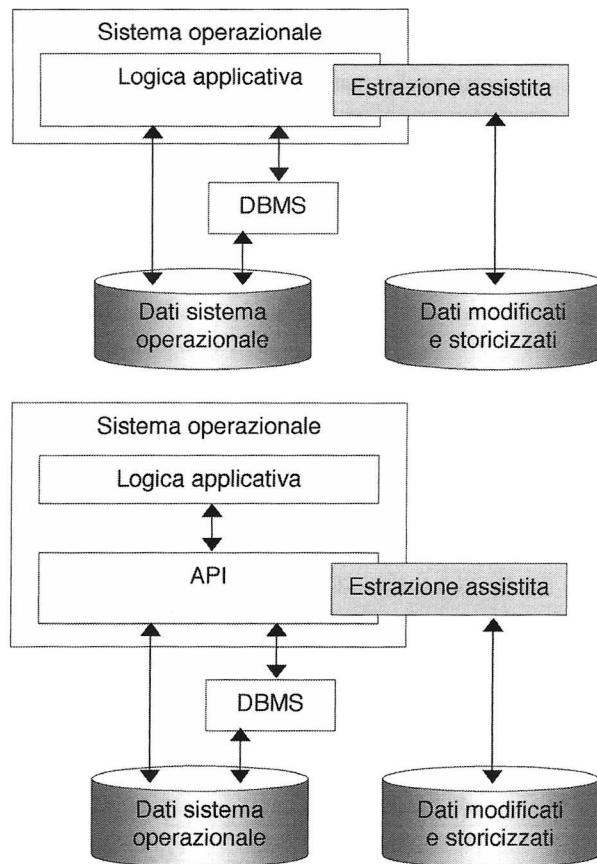
- *Estrazione assistita dall'applicazione.* È una tecnica di estrazione immediata e consiste nel realizzare all'interno delle applicazioni operazionali, senza peraltro modificarne il comportamento esterno, un insieme di funzioni che consentano di memorizzare stabilmente nella staging area le modifiche intervenute sui dati. L'estrazione assistita dall'applicazione fornisce una soluzione molto potente ma è difficile da realizzare poiché richiede di modificare applicazioni esistenti, spesso mal progettate e poco documentate; d'altro canto, soprattutto nel caso dei sistemi legacy che non supportano trigger, giornali (*log*) e marche temporali, può essere l'unica tecnica incrementale utilizzabile. L'estrazione assistita dall'applicazione trova giustificazione anche nei sistemi operazionali di nuova generazione in cui sia presente un substrato di primitive (API – *Application Programming Interface*) utilizzato uniformemente da tutte le applicazioni per accedere ai dati e rendersi indipendenti dal DBMS. La presenza di un livello di API fa sì che l'estrazione dei dati avvenga in modo centralizzato (rispetto alle applicazioni) e trasparente sia ai progettisti delle applicazioni, che possono essere modificate, sia ai gestori del DBMS, che può essere sostituito. La Figura 10.2 presenta entrambe le soluzioni architetturali.
- *Estrazione basata su trigger.* È una tecnica di estrazione immediata, in cui la responsabilità dell'estrazione dei dati è spostata dalle applicazioni al DBMS. Un trigger è una procedura attivata direttamente dal DBMS al verificarsi di particolari eventi; ai fini dell'estrazione, un trigger viene associato a ogni evento che causa cambiamenti nei dati che si vogliono monitorare. Il trigger salverà il dato modificato in un apposito file o tabella della staging area, da cui poi esso sarà prelevato per le opportune trasformazioni. La Figura 10.3 riporta la soluzione architettonica indotta da questa tecnica.

Per motivi di prestazioni non è possibile adottare in modo estensivo questa tecnica, che richiede al DBMS di monitorare continuamente le transazioni che potrebbero innescare i trigger.

- *Estrazione basata su log.* È una tecnica di estrazione immediata in cui le operazioni di estrazione dei cambiamenti sono basate sui file di log prodotti dal DBMS. I file di log sono generati da tutti i DBMS moderni e rappresentano il principale strumento per le operazioni di back-up e ripristino dei dati; il limite principale dell'estrazione basata su log riguarda la corretta interpretazione dei file, il cui formato è normalmente proprietario dello specifico DBMS.

Vista la complessità dell'analisi richiesta, si consiglia di utilizzare l'estrazione basata su log solo quando il modulo di estrazione è sviluppato direttamente dal produttore del DBMS.

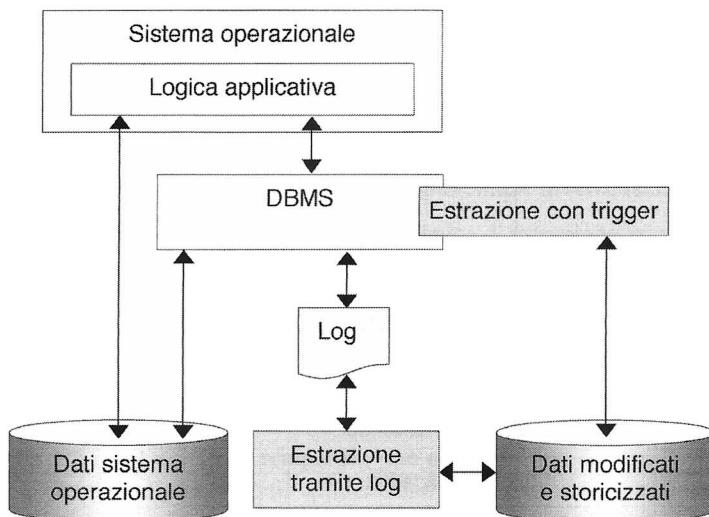
In Figura 10.3 è rappresentata la soluzione architettonale richiesta da questa tecnica.



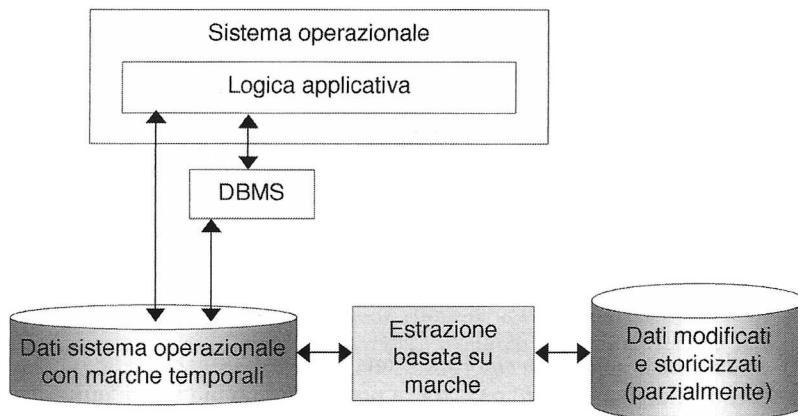
**Figura 10.2** Moduli necessari a realizzare l'estrazione assistita dall'applicazione: nella parte alta la soluzione standard, in quella bassa la soluzione con livello API.

- *Estrazione basata su marche temporali.* È una tecnica di estrazione ritardata e richiede che lo schema relativo ai record da estrarre preveda uno o più campi utilizzati per contrassegnare i record modificati rispetto all'ultima esecuzione del processo di estrazione dei dati. Il sistema operazionale stesso è preposto all'aggiornamento delle marche temporali (*time-stamp*) mentre, come si può vedere in Figura 10.4, il modulo per l'estrazione dei dati opera in un secondo tempo direttamente sui dati operazionali producendo l'insieme, eventualmente incompleto, delle modifiche avvenute nel sistema.

Rispetto alle tecniche precedentemente proposte, l'efficacia di quella basata su marche dipende dalla struttura stessa del sistema operazionale: se i dati del sistema operazionale sono transitori o semi-storicizzati l'estrazione basata su marche non può identificare gli stati intermedi di quei record che vengono modificati più di una volta durante l'intervallo di aggiornamento.



**Figura 10.3** Moduli necessari a realizzare l'estrazione basata su trigger e su file di log.



**Figura 10.4** Moduli necessari a realizzare l'estrazione basata su marche temporali.

Si consideri l'esempio di Figura 10.5 che riporta, in diversi momenti, lo stato dei record relativi a un ordine di merce in un database operazionale non storizzato<sup>1</sup>. L'ordine del signor Maio (100 unità) viene inoltrato e quindi estratto nella notte dello

<sup>1</sup> Si noti che la presenza del campo **data** non implica la storizzazione della relazione, che dipende dal modo in cui vengono gestite le modifiche.

stesso giorno per il successivo inserimento nel data mart. Il giorno seguente l'acquirente aumenta l'ordine portandolo a 200 unità, e il sistema operazionale registra tale fatto sovrascrivendo il precedente valore. Infine, durante il giorno successivo, a causa della mancanza del prodotto, l'ordine viene nuovamente ritoccato al valore definitivo di 150 unità. Quando, la notte del 3/4/2002, il modulo di estrazione entra in funzione, identifica il record modificato (in base al campo data che rappresenta la marca temporale) perdendo però lo stato del 2/4/2002 sovrascritto dall'ultima modifica. Ovviamente si è verificata una perdita di informazione la cui importanza va valutata assieme al committente. Se si decide che essa incide sulla significatività dell'informazione nel data mart, la tecnica basata su marche temporali dovrà essere scartata. Un ulteriore problema legato all'estrazione basata su marche si verifica in caso di cancellazione dei record. Riprendendo l'esempio di Figura 10.5, si supponga che l'ordine venga evaso in data 4/4/2002 e conseguentemente, come in genere avviene in un database non storizzato, il record venga rimosso. Ovviamente, non essendo più presente il record, il sistema basato su marche temporali non potrà individuare la cancellazione. In questo caso, per poter ottenere la corretta sequenza temporale degli eventi, è necessario che il sistema operazionale non cancelli il record ma lo marchi come non valido sino alla successiva esecuzione della procedura di estrazione.

Situazione al 1/4/2002



cod	prodotto	cliente	qtà	data
1	Greco di tufo	Malavasi	50	15/3/2002
2	Barolo	Maio	100	1/4/2002
.....	.....	.....	.....	.....


 Estratto 1/4/2002

Situazione al 2/4/2002



cod	prodotto	cliente	qtà	data
1	Greco di tufo	Malavasi	50	15/3/2002
2	Barolo	Maio	200	2/4/2002
.....	.....	.....	.....	.....


 Estratto 1/4/2002

Situazione al 3/4/2002



cod	prodotto	cliente	qtà	data
1	Greco di tufo	Malavasi	50	15/3/2002
2	Barolo	Maio	150	3/4/2002
.....	.....	.....	.....	.....


 Estratto 3/4/2002

**Figura 10.5** L'effetto di modifiche multiple per la tabella ORDINI in un sistema operazionale non storizzato in cui l'estrazione dei dati è basata su marche temporali (Devlin, 1997).

- Comparazione di file.* È una tecnica di estrazione ritardata che prevede di confrontare due versioni successive dei dati al fine di evidenziarne le differenze e richiede pertanto di mantenere la versione dei dati antecedente all'inizio delle modifiche. Come nel caso dell'estrazione basata su marche, l'efficacia dipende dalla natura dei dati: nel caso di dati temporanei o semi-storicizzati alcune delle modifiche avvenute tra due catture successive possono essere perse. Questa tecnica deve essere considerata solo quando nessuna delle precedenti tecniche è applicabile poiché richiede di mantenere due versioni complete dei dati e comporta un elevato costo di esecuzione a causa delle operazioni di comparazione. Il suo impiego viene valutato normalmente quando la memorizzazione dei dati operazionali è basata su file o quando il DBMS utilizzato non supporta trigger o meccanismi di log. Poiché inoltre essa non richiede nessuna modifica alle applicazioni del sistema operazionale, risulta particolarmente adatta ai sistemi legacy.

Le tecniche di estrazione ritardata sono utili in presenza di sistemi operazionali non storicizzati o parzialmente storicizzati solo se gli utenti non richiedono di valutare ogni modifica dello stato delle informazioni.

La Tabella 10.1 riporta la comparazione tra le diverse tecniche proposte in base alle caratteristiche discusse in precedenza.

**Tabella 10.1** Comparazione delle caratteristiche delle diverse tecniche di estrazione incrementale (Devlin, 1997).

	<i>Statica</i>	<i>Marche temporali</i>	<i>Comparazione di file</i>	<i>Assistita da applicazione</i>	<i>Trigger</i>	<i>Log</i>
<i>Gestione di dati transitori o semi-storicizzati</i>	No	Incompleta	Incompleta	Completa	Completa	Completa
<i>Supporto per sistemi basati su file</i>	Sì	Sì	Sì	Sì	No	Raro
<i>Tecnica di realizzazione</i>	Prodotti	Prodotti o sviluppo interno	Prodotti	Sviluppo interno	Prodotti	Prodotti
<i>Costi di sviluppo interno</i>	Nessuno	Medi	Nessuno	Alti	Nessuno	Nessuno
<i>Utilizzo in sistemi legacy</i>	Sì	Difficile	Sì	Difficile	Difficile	Sì
<i>Modifiche ad applicazioni</i>	Nessuna	Probabile	Nessuna	Probabile	Nessuna	Nessuna
<i>Dipendenza delle procedure dal DBMS</i>	Limitata	Limitata	Limitata	Variabile	Alta	Limitata
<i>Impatto sulle prestazioni del sistema operaz.</i>	Nessuna	Nessuna	Nessuna	Medio	Medio	Nessuna
<i>Complessità delle procedure di estrazione</i>	Bassa	Bassa	Bassa	Alta	Media	Bassa

Qualunque tecnica incrementale si utilizzi, il risultato della fase di estrazione consiste nell'insieme di record della sorgente modificati, aggiunti o cancellati rispetto alla precedente esecuzione della procedura di estrazione, temporaneamente memorizzati nella staging area. Al fine di semplificare la successiva fase di caricamento dei dati nel data mart conviene associare a ogni record estratto il tipo di operazione (Inserimento, Modifica, Cancellazione) che ne ha determinato la variazione: in questo modo infatti il processo preposto al caricamento potrà stabilire a priori come trattare ciascun record.

In Figura 10.6 è riportato un esempio relativo a una possibile istanza della relazione ORDINI in cui si suppone di utilizzare una tecnica, per esempio quella basata sul log, che consenta di individuare tutte le variazioni dello stato della tabella. Si noti come il campo oper permetta di identificare il tipo di operazione che ha causato l'estrazione.

Situazione al 4/4/2002

<u>cod</u>	prodotto	cliente	qtà
1	Greco di tufo	Malavasi	50
2	Barolo	Maio	150
3	Barbera	Lumini	75
4	Sangiovese	Cappelli	45

Situazione al 6/4/2002

<u>cod</u>	prodotto	cliente	qtà
1	Greco di tufo	Malavasi	50
2	Barolo	Maio	150
4	Sangiovese	Cappelli	145
5	Vermentino	Maltoni	25
6	Trebbiano	Maltoni	150



Differenza incrementale

<u>cod</u>	prodotto	cliente	qtà	oper
3	Barbera	Lumini	75	C
4	Sangiovese	Cappelli	145	M
5	Vermentino	Maltoni	25	I
6	Trebbiano	Maltoni	150	I

**Figura 10.6** Risultato della procedura di estrazione incrementale per la tabella ORDINI di un database operazionale.

### 10.1.2 La trasformazione dei dati

Durante questa fase vengono eseguite le trasformazioni necessarie a conformare i dati delle sorgenti alla struttura dello schema riconciliato. Non ci soffermeremo a lungo su di essa ricordando solo le categorie di operazioni più comuni:

- *Conversione*: è applicata ai singoli campi che negli schemi sorgenti hanno un formato diverso da quello dello schema riconciliato. Tipiche conversioni sono quelle di tipo (per esempio da intero a decimale, da EBCDIC a ASCII), quelle legate al sistema di misura (per esempio da britannico a metrico decimale) o infine quelle legate al formato (per esempio da minuscolo a maiuscolo).
- *Arricchimento*: permette di combinare le informazioni presenti in uno o più campi per creare nuove informazioni o meglio rendere più facilmente fruibili quelle esistenti. Esempio classico di questo tipo di trasformazione è il calcolo di dati derivati.
- *Separazione/Concatenamento*: permette di ricombinare i campi letti dagli schemi sorgenti separando informazioni che erano memorizzate assieme in questi ultimi e che invece sono mantenute separate nello schema riconciliato e viceversa. Il caso più comune di questo tipo di trasformazione sono le operazioni di normalizzazione e denormalizzazione dei dati discusse nel Capitolo 3.

L'operazione più complessa tra quelle descritte risulta senz'altro quella di concatenamento dei dati. Infatti essa richiede normalmente di estrarre i dati da molteplici sorgenti ricombinandoli successivamente in base allo schema riconciliato. Ciò implica l'utilizzo di tabelle temporanee che possono avere anche dimensioni raguardevoli. Il progettista, durante la definizione del processo di alimentazione, deve quindi curarsi di riservare a tal fine sufficienti risorse nella staging area.

### 10.1.3 Il caricamento dei dati

In quest'ultima fase i dati estratti dalle sorgenti operazionali e opportunamente trasformati vengono caricati dalla staging area nel database riconciliato; la modalità di caricamento dipende ovviamente dalla tecnica utilizzata in fase di estrazione e può variare a seconda che la porzione del database riconciliato che si sta alimentando sia storicizzata oppure no.

Se si è utilizzata una tecnica di estrazione statica i dati presenti nel livello riconciliato saranno completamente eliminati e sostituiti con i nuovi; al contrario se si è utilizzata una tecnica incrementale la politica di aggiornamento dipenderà dal livello di storicizzazione del livello riconciliato: mentre l'inserimento di nuovi record non pone particolari problemi, la modifica di quelli già presenti va valutata con attenzione.

Se i concetti considerati non sono storicizzati è necessario individuare i record che hanno subito una modifica ed effettuarne la sostituzione. Questa operazione non è in genere particolarmente complessa, dato che nella maggior parte dei casi è facile individuare nel database riconciliato i record che corrispondono a quelli nelle sorgenti operazionali, ma può diventarlo qualora le operazioni di estrazione e trasformazione abbiano inciso profondamente sulla struttura dei record (per esempio modificandone la chiave). Per poter eseguire in modalità incrementale anche l'alimentazione dei data mart è necessario determi-

nare quali tra i record del database riconciliato sono stati modificati in seguito al caricamento: a tal fine, a ogni relazione dovrà essere aggiunto un *flag* che memorizzi quale operazione ha causato la modifica.

In Figura 10.7 è riportato un esempio che riprende quello di Figura 10.6. Il campo *oper* classifica i record in base al tipo di variazione subita rispetto alla precedente estrazione; i record con valori di chiave 1 e 2, non avendo subito nessuna modifica, riportano il valore NULL e non saranno perciò coinvolti nella successiva fase di alimentazione dei data mart. Come si può notare i valori del campo *oper* possono variare, per uno stesso record, a ogni esecuzione della procedura di estrazione. Per una migliore comprensione dell'esempio si tenga presente che in effetti, dopo l'estrazione del 6/4/2002, il record con valore di chiave 3 verrà eliminato dal livello riconciliato: infatti la sua utilità termina non appena l'informazione relativa alla cancellazione del record è stata propagata al data mart.

Se si è deciso di storicizzarre i dati del livello riconciliato è necessario gestire le marcate temporali che definiscono il periodo di validità dei record. In Figura 10.8 è riportata la versione storicizzata dell'esempio di Figura 10.7. La storicizzazione è implementata tramite la coppia di attributi *inizio* e *fine* che determinano l'intervallo di validità di ciascun record; la chiave della relazione include anche il campo *inizio* al fine di rendere possibile la memorizzazione di più versioni dello stesso record del database operazionale. In questa soluzione il record 3 non verrà cancellato, ma presentando un valore nel campo *fine* risulta non più valido. I record coinvolti nella successiva fase di alimentazione degli schemi a stella sono quelli che riportano nel campo *inizio* una data posteriore rispetto a quella in cui è stato eseguito l'ultimo aggiornamento del data mart.

Si ricorda che l'architettura a tre livelli con schema riconciliato presuppone che quest'ultimo sia caratterizzato da una storicizzazione almeno uguale a quella dello schema operazionale.

<u>cod</u>	prodotto	cliente	qtà	oper	Situazione al 4/4/2002
1	Greco di tufo	Malavasi	50	—	
2	Barolo	Maio	150	—	
3	Barbera	Lumini	75	I	
4	Sangiovese	Cappelli	45	I	

Situazione al 6/4/2002		<u>cod</u>	prodotto	cliente	qtà	oper
		1	Greco di tufo	Malavasi	50	—
		2	Barolo	Maio	150	—
		3	Barbera	Lumini	75	C
		4	Sangiovese	Cappelli	145	M
		5	Vermentino	Maltoni	25	I
		6	Trebbiano	Maltoni	150	I

**Figura 10.7** Due diversi stati della relazione ORDINI, non storicizzata, nel database riconciliato.

<u>cod</u>	prodotto	cliente	qtà	<u>inizio</u>	fine	Situazione al 4/4/2002
1	Greco di tufo	Malavasi	50	15/3/2002	—	
2	Barolo	Maio	150	1/4/2002	—	
3	Barbera	Lumini	75	4/4/2002	—	
4	Sangiovese	Cappelli	45	4/4/2002	—	

<u>cod</u>	prodotto	cliente	qtà	<u>inizio</u>	fine	Situazione al 6/4/2002
1	Greco di tufo	Malavasi	50	15/3/2002	—	
2	Barolo	Maio	150	1/4/2002	—	
3	Barbera	Lumini	75	4/4/2002	5/4/2002	
4	Sangiovese	Cappelli	45	4/4/2002	5/4/2002	
4	Sangiovese	Cappelli	145	5/4/2002	—	
5	Vermentino	Maltoni	25	5/4/2002	—	
6	Trebbiano	Maltoni	150	6/4/2002	—	

**Figura 10.8** Due diversi stati della relazione ORDINI, storicizzata, nel database riconciliato.

## 10.2 Pulizia dei dati

Con il termine pulizia dei dati (*data cleaning*, *data cleansing* o *data scrubbing*) si intende l'insieme delle operazioni atte a garantire la consistenza e la correttezza dei dati presenti nel livello riconciliato. Sebbene questa fase possa essere vista come parte integrante della trasformazione dei dati, vista la sua importanza preferiamo trattarla separatamente.

Le principali cause di inconsistenza nei dati sono le seguenti:

- *Errori di battitura.* Sono sempre possibili se al momento dell'inserimento non sono previste procedure di controllo dei valori inseriti (per esempio, il controllo di correttezza del codice fiscale).
- *Differenza di formato dei dati nello stesso campo.* È una delle situazioni più comuni e può verificarsi ogniqualvolta l'informazione memorizzata in un campo non sia rigidamente strutturata. Per esempio, uno stato all'interno di un indirizzo può essere indicato in modo abbreviato e non ('I', 'IT', 'Italia'), oppure il nome di un'azienda può apparire in forme diverse ('The Coca-Cola Company', 'Coca Cola', 'Coca-Cola Co.' ecc.).
- *Inconsistenza tra valori e descrizione dei campi.* Il problema può nascere a causa dell'evoluzione del modo di operare in azienda o dal variare delle abitudini di vita che rendono indispensabile una informazione di cui non era prevista la memorizzazione. Per ovviare a tale deficienza può accadere che gli utenti inseriscano un dato in un campo correlato (per esempio il numero di cellulare assieme al numero del telefono fisso)

oppure di minore utilità (per esempio l'indirizzo di posta elettronica in un generico campo di note). Una seconda causa di questo problema può essere la negligenza di un settore dell'azienda che, per propria comodità o consuetudine, adotta delle convenzioni diverse da quelle previste dal sistema informativo (per esempio una codifica interna differenziata per i codici dei prodotti).

- *Inconsistenza tra valori di campi correlati.* Il problema si verifica quando i valori presenti in due o più campi sono singolarmente corretti ma tra loro inconsistenti. Per esempio, in una tabella contenente dati anagrafici, un record contenente i valori città='Bologna', regione='Lazio' è sicuramente inconsistente. Queste situazioni si verificano normalmente a fronte di errori nella fase di inserimento dati. Più grave è il caso in cui informazioni provenienti da due sorgenti diverse riportino informazioni contrastanti relativamente a uno stesso soggetto: un caso tipico è quello delle anagrafiche gestite, in assenza di un sistema informativo integrato, separatamente da ogni divisione.
- *Informazioni mancanti.* Quando le applicazioni del sistema operazionale consentono di non riempire alcuni dei campi, la fretta e lo scarso interesse verso un dato spingono spesso l'operatore a tralasciarne l'inserimento. Si consideri per esempio la scheda compilata a uno sportello bancario all'atto dell'apertura di un conto corrente: la data di nascita del cliente è una informazione di scarso interesse per l'impiegato che redige il contratto e verrà probabilmente tralasciata, mentre risulterebbe molto utile all'ufficio marketing della banca per classificare i clienti.
- *Informazioni duplicate.* Tipicamente questo problema si verifica quando due sorgenti forniscono la stessa informazione ma i record utilizzano chiavi diverse; identificare corrispondenze tra i record durante la fusione delle informazioni diventa così una operazione molto complessa.

La maggior parte delle inconsistenze può essere prevenuta rendendo più rigorose le regole di inserimento dati nelle applicazioni del sistema operazionale. Compito del progettista è segnalare i problemi alla direzione e ai responsabili dei sistemi operazionali, tuttavia i tempi di aggiornamento delle applicazioni sono normalmente incompatibili con i tempi di realizzazione del DW che dovrà quindi basare la consistenza dei propri dati sulla qualità delle procedure di pulizia dei dati. Per quanto accurato sia il lavoro del progettista, difficilmente si potrà raggiungere una qualità dei dati del 100%. Alcuni errori rimangono presenti anche nel DW e saranno presentati agli utenti: questa è una realtà che il progettista deve accettare.

Ogni tipo di problema richiede tecniche diverse per la sua soluzione e già molti sono i sistemi commerciali che propongono moduli specifici per la pulizia dei dati; ciononostante, la specificità del problema rende spesso indispensabile l'implementazione di procedure *ad hoc*. Le principali funzionalità utilizzate sono descritte nei paragrafi seguenti.

### 10.2.1 Tecniche basate su dizionari

Con questo termine si indicano le tecniche di verifica della correttezza dei valori di un campo basate su tabelle di riferimento (*look-up table*) e dizionari per la ricerca di abbreviazioni e sinonimi; esse sono applicabili solo quando il dominio del campo è conosciuto e

limitato, per esempio quello dei nomi di farmaci o dei comuni di una certa nazione. L'idea di base è scandire i valori che uno specifico campo presenta verificando se appartengono al dominio specificato oppure se sono sinonimi o abbreviazioni di un valore del dominio. Nel primo caso non viene intrapresa alcuna operazione, nel secondo il valore viene rimpiazzato dal corrispondente valore del dominio; infine, se non viene trovata alcuna corrispondenza, il dato viene scartato.

Le tecniche basate su dizionari si prestano a risolvere sia gli errori di battitura sia le discrepanze di formato e, operando su domini applicativi ristretti, forniscono di norma buoni risultati. Esse possono essere applicate anche a più campi contemporaneamente al fine di verificare eventuali inconsistenze tra due campi correlati: per esempio, avendo a disposizione una tabella di riferimento che riporta le regioni italiane e, per ognuna di esse, l'elenco delle città che vi appartengono, è possibile verificare se i campi **città** e **regione** di un indirizzo sono tra loro consistenti. Tra gli strumenti che offrono capacità di questo tipo ricordiamo Data Cleanse (ETI, 2005).

### 10.2.2 Tecniche di fusione approssimata

Ogniqualvolta si debbano combinare in un'unica destinazione dati provenienti da sorgenti diverse senza una chiave comune, nasce la necessità di identificare i record che si corrispondono. Le due tipiche situazioni in cui sorge questo problema sono presentate di seguito. Nello schema di Figura 10.9, a seguito della fase di integrazione è stata scoperta un'associazione implicita tra l'entità **CLIENTE**, modellata nel database della divisione marketing, e l'entità **ORDINE** nel database dell'amministrazione. Per rendere esplicita l'associazione è necessario individuare quale cliente ha inoltrato un particolare ordine. Sfortunatamente, i due database identificano diversamente i clienti e quindi il join dovrà essere eseguito sulla base dei campi comuni (**indirizzo cliente**, **cognome cliente**) che non rappresentano un identificatore per il cliente e che, essendo di natura descrittiva, non sono sottoposti a procedure di controllo che assicurino il rispetto dei vincoli di integrità referenziale e l'assenza di errori di inserimento. In questo caso si parla di *join approssimato* poiché non c'è certezza nella definizione dei record corrispondenti.

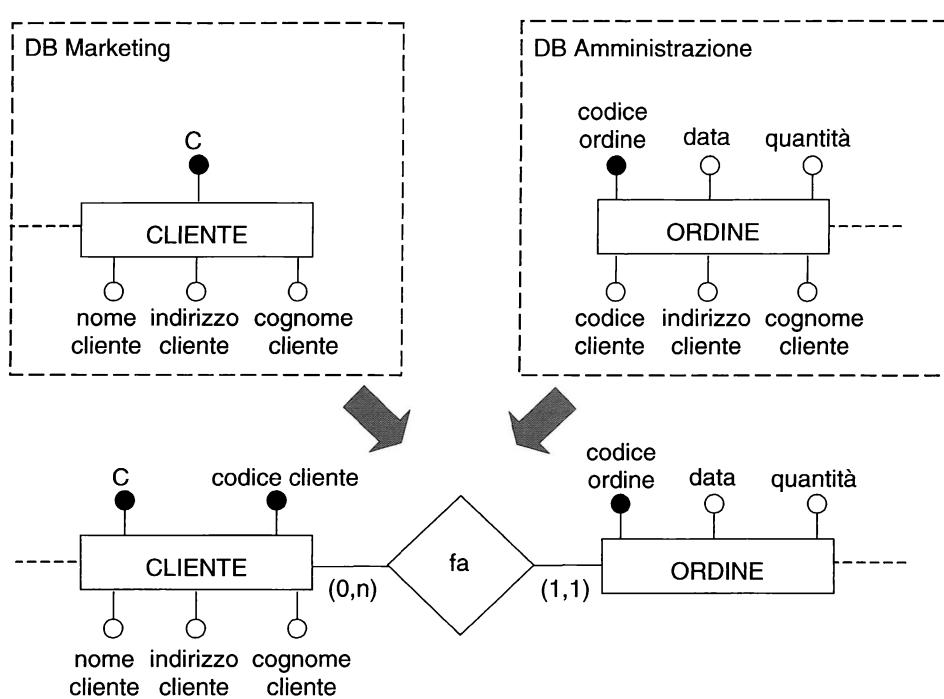
Leggermente diversa è la situazione di Figura 10.10, in cui istanze diverse di uno stesso schema devono essere fuse assieme (*purge/merge problem*). Le due istanze non sono disgiunte (un cliente può acquistare sia a Roma che a Milano), inoltre anche nei singoli database i clienti potrebbero essere inseriti più volte a causa di errori di battitura che non hanno permesso di individuare un precedente inserimento.

Il problema della similarità tra record è stato studiato da Monge (1996; 1997), che ha proposto alcuni algoritmi per il calcolo del livello di similarità tra due campi alfanumerici  $A$  e  $B$ . La funzione di affinità compara le coppie di parole  $A_i \in A$  e  $B_j \in B$  e calcola la similarità in base alla formula:

$$\text{affinità}(A, B) = \frac{1}{|A|} \sum_{i=1}^{|A|} \max_{j=1}^{|B|} \text{affinità}(A_i, B_j)$$

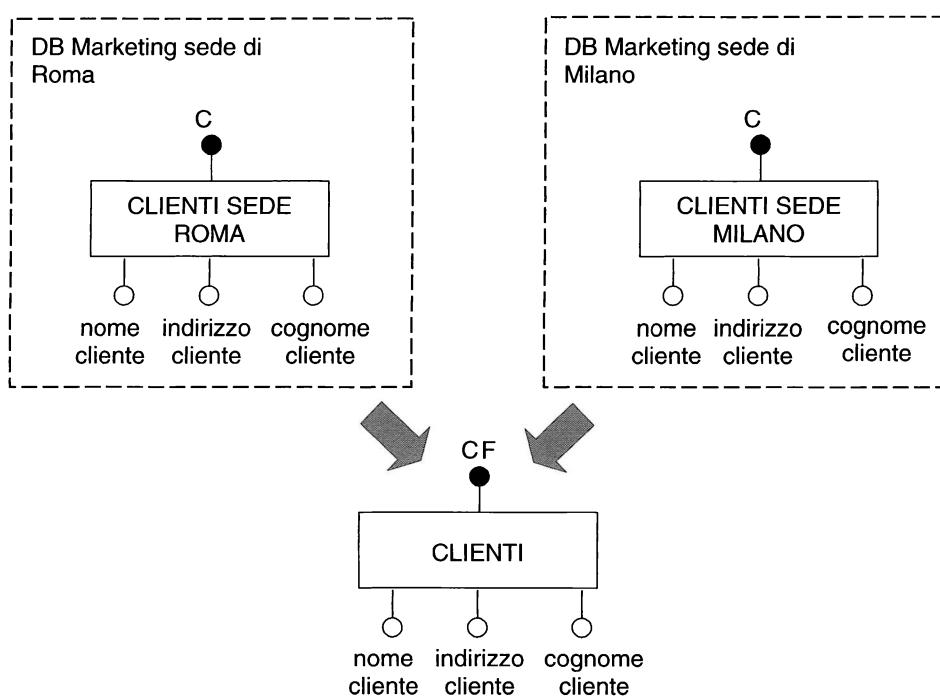
dove l'affinità tra le parole è calcolata in base al confronto tra stringhe e verificando la presenza di abbreviazioni ricercate in base a schemi ben definiti:

- Un'abbreviazione può essere un prefisso di una sottostringa, per esempio 'Univ.' è un prefisso per 'Università'.
- Un'abbreviazione può combinare un prefisso e un suffisso di una sottostringa, un esempio è 'Dott.ssa' che abbrevia "Dottoressa".
- Un'abbreviazione può essere un acronimo di una stringa, per esempio 'DEIS' è un acronimo per 'Dipartimento di Elettronica Informatica e Sistemistica'.
- Un'abbreviazione può essere la concatenazione di più prefissi contenuti nella stringa, un esempio 'UNIBO' abbrevia 'Università di Bologna'.



**Figura 10.9** La fusione dei dati tra i due schemi locali richiede un join approssimato a causa della mancanza di un identificatore comune per il concetto di cliente.

Un secondo insieme di tecniche utilizzato per identificare la corrispondenza tra record si basa su gruppi di regole ognuna delle quali valuta la similarità tra i campi corrispondenti di due record (Hernandez, 1995); normalmente vengono valutate caratteristiche quali la *edit-distance* (Bunke, 1993) tra due stringhe o la differenza di valore tra due campi numerici. A titolo di esempio in Figura 10.11 riportiamo sotto forma di pseudo-codice alcune delle regole proposte da Hernandez (1998) per la soluzione del problema di purge/merge applicato a record contenenti informazioni su persone. L'idea di base è calcolare la similarità tra coppie di record; tutti quelli la cui similarità supera una certa soglia vengono poi fusi in un unico record. Tra gli strumenti che offrono capacità di questo tipo ricordiamo EDD Data Cleanser (EDD, 2005).



**Figura 10.10** La fusione dei dati tra i due schemi locali richiede di verificare la presenza di record duplicati.

```

// in input: due record P1, P2 che includono i campi codice, nome, indirizzo, città,
// CAP, stato
// in output: VERO se P1 e P2 sono lo stesso record, FALSO se P1 e P2 non
// sono lo stesso record
codiciSimili=comparaCodici(P1,P2);
nomiSimili=comparaNomi(P1,P2);
indirizziSimili=comparaIndirizzi(P1,P2);
cittàSimili=comparaCittà(P1,P2);
CAPSimili=comparaCAP(P1,P2);
statiSimili=comparaStati(P1,P2);
se (codiciSimili e nomiSimili)
    restituisci VERO;
indirizziMoltoSimili=(indirizziSimili e cittàSimili e (CAPsimili o statiSimili));
se ((codiciSimili o nomiSimili) e indirizziMoltoSimili)
    restituisci TRUE;
.....

```

**Figura 10.11** Pseudo-codice per un algoritmo di pulizia di dati anagrafici di persone basato su regole.

### 10.2.3 Tecniche ad hoc

Ogni dominio applicativo ha caratteristiche proprie e regole di business troppo specifiche per essere verificate tramite strumenti standard. In questi casi il progettista dovrà curare la definizione di interrogazioni o piccoli programmi in grado di controllare che i dati rispettino tali regole. Per esempio le equazioni:

$$\begin{aligned} \text{profitto} &= \text{guadagno} - \text{spese} \\ \text{capitale} &= \text{disponibilità} + \text{crediti} - \text{debiti} \end{aligned}$$

devono essere sempre verificate in un database contenente dati di carattere finanziario.

In altri casi, possibili errori possono essere identificati ricercando valori fuori dalla norma (*outlier*); per esempio una variazione di un prezzo di vendita superiore al 20% è probabilmente causata da un errore.

Infine, un ultimo controllo molto comune riguarda il rispetto delle dipendenze funzionali tanto comuni nelle dimension table. Il controllo può essere eseguito nella staging area utilizzando una tabella relazionale o un semplice file. Supponendo di voler verificare un'associazione di tipo molti-a-uno tra due attributi a e b è sufficiente ordinare i record in base ad a e verificare quindi che ognuno dei suoi valori sia associato a un solo valore di b.

## 10.3 Alimentazione delle dimension table

Le dimension table sono le prime tavole da alimentare nei data mart. Questo obbligo deriva dalla struttura degli schemi a stella, che presuppone un vincolo di integrità referenziale tra la chiave primaria di ogni dimension table e la corrispondente chiave esterna nella fact table.

La tecnica da utilizzare per il caricamento dipende da come sono gestite le informazioni nei due livelli coinvolti: quello riconciliato e quello dei data mart. Se a livello riconciliato gli aggiornamenti sono stati eseguiti in modo incrementale anche il caricamento nei data mart potrà sfruttare questa tecnica, altrimenti sarà probabilmente più efficiente ricreare l'intera dimension table.

Questa seconda soluzione, ovviamente più semplice dal punto di vista progettuale, può però comportare tempi elevati. Una procedura di caricamento statico è comunque da prevedere, poiché necessaria all'atto della creazione dello schema di fatto. Nel seguito ci concentreremo sull'alimentazione incrementale che, oltre a essere il caso più complesso, può essere vista come una generalizzazione dell'alimentazione statica.

Le due principali operazioni da svolgere in questa fase, descritte nei paragrafi seguenti, sono:

- Identificare l'insieme dei dati da caricare.
- Sostituire le chiavi presenti nelle tavole del sistema riconciliato con quelle, surrogate, utilizzate negli schemi a stella.

### 10.3.1 Identificazione dei dati da caricare

La corrispondenza tra attributi dello schema riconciliato e attributi delle dimension table è già stata definita durante la progettazione degli schemi a stella; per identificare quali relazioni e quali attributi sono coinvolti nell'alimentazione di una data dimension table è quindi sufficiente consultare la documentazione redatta durante la progettazione logica (si vedano i Capitoli 9 e 13).

Più complesso è stabilire se le tuple di una dimension table hanno effettivamente subito una modifica. Infatti queste ultime, a causa della denormalizzazione che le caratterizza, contengono attributi provenienti da più relazioni dello schema riconciliato; d'altro canto, non tutti gli attributi di queste relazioni saranno inclusi nella dimension table. Purtroppo, le procedure ETL che alimentano lo schema riconciliato non rappresentano le modifiche nel dettaglio dei singoli attributi, ma piuttosto a livello degli interi record. Ciò implica che talvolta si considererà modificata una tupla i cui valori, in realtà, non hanno subito aggiornamenti. Questa situazione può creare problemi nel caso di dimensioni dinamiche poiché porta alla proliferazione di record della dimension table contenenti gli stessi valori.

Non esiste una soluzione a questo problema, se non quella di comparare il contenuto dei record che si suppongono modificati con i corrispondenti record della dimension table. Nella maggior parte dei casi il modo più efficiente consiste nel duplicare temporaneamente la dimension table nella staging area ed eseguire lì la comparazione. In alternativa, se lo schema riconciliato è materializzato la comparazione può essere fatta sullo schema riconciliato a partire dai record estratti dal database operazionale.

### 10.3.2 Sostituzione delle chiavi

La seconda importante operazione da svolgere consiste nell'introduzione delle chiavi surrogate, ed è di per sé semplice per i nuovi record poiché si concretizza nella scelta di un valore numerico che non sia già utilizzato nella dimension table. Invece, per rendere possibili le modifiche a record già presenti è necessario mantenere un collegamento tra le tuple della dimension table e quelle nello schema riconciliato; ciò viene normalmente fatto memorizzando permanentemente nella staging area una relazione che include la chiave della relazione dello schema riconciliato da cui ha origine la dimension table e la chiave surrogata della dimension table. In questo modo, per identificare la tupla della dimension table avendo a disposizione il record del database riconciliato, sarà sufficiente una operazione di ricerca. Dopo aver determinato il contenuto dei record coinvolti nell'alimentazione si procede al caricamento vero e proprio.

Mentre l'inserimento di una nuova coordinata per una dimensione (per esempio un nuovo prodotto) determina semplicemente l'aggiunta di una tupla alla dimension table, la modifica del valore di un attributo in corrispondenza di una coordinata preesistente (per esempio la sostituzione del responsabile di un negozio) si realizza in modo diverso a seconda delle caratteristiche di dinamicità dichiarate (Paragrafo 8.4):

- *Gerarchie statiche*: non essendo possibile nessuna modifica dei record già presenti, l'unica operazione consentita è l'aggiunta di nuovi record. Non è allora strettamente necessario mantenere la tabella di corrispondenza tra chiavi descritta in precedenza.
- *Gerarchie di tipo 1*: prevedono la sostituzione del dato modificato senza tener traccia del valore precedente. In questo caso è sufficiente sovrascrivere la tupla della dimension table corrispondente alla tupla modificata nel sistema relazionale, identificabile tramite la tabella di corrispondenza tra chiavi.
- *Gerarchie di tipo 2*: prevedono il mantenimento dei record modificati e l'inserimento di nuovi record per le versioni aggiornate. In questo caso una modifica dà luogo all'inserimento di una nuova tupla con un nuovo valore di chiave surrogata e all'aggiornamento della tabella di corrispondenza tra chiavi.
- *Gerarchie di tipo 3*: prevedono la completa storizziazione della dimension table. Se si utilizza la soluzione proposta nel Paragrafo 8.4.3, a fronte di una modifica sarà necessario modificare la vecchia tupla della dimension table inserendo la marca di fine validità del record, aggiungere un nuovo record contenente i dati aggiornati e aggiornare la tabella di corrispondenza tra chiavi.

Le marche temporali da inserire nelle gerarchie di tipo 3 sono quelle estratte dal livello riconciliato se questo è storizzato, in caso contrario si utilizza, per esempio, la data di alimentazione.

In Figura 10.12 è rappresentata graficamente la sequenza delle operazioni fin qui descritte; si suppone che la gerarchia modellata dalla dimension table includa gli attributi attr1, attr3, attr5 e attr6 e risultati dal join tra tre tabelle dello schema riconciliato.

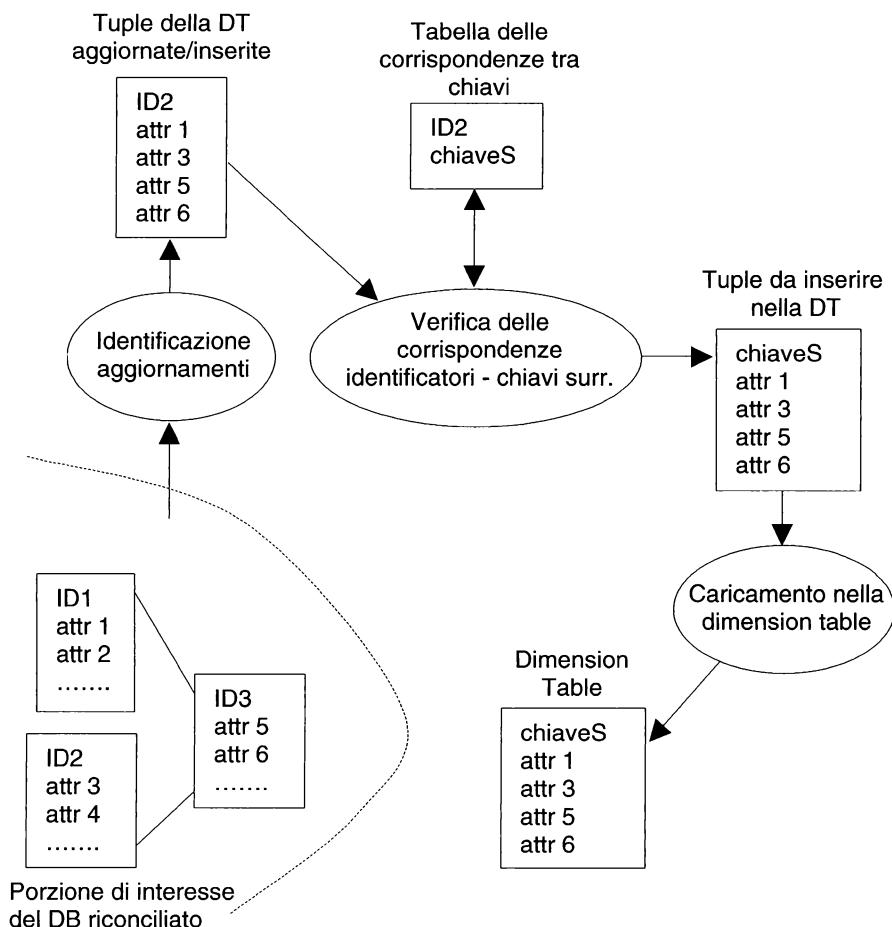
Nelle dimension table compaiono a volte alcuni attributi non presenti nello schema riconciliato, i cui valori devono essere comunque caricati manualmente. Per questi attributi si consiglia di inserire nelle dimension table, all'atto dell'alimentazione, dei valori fittizi (per esempio 'campo da completare') e di predisporre delle piccole applicazioni dedicate che consentano al personale preposto di compilarli direttamente.

Un'ultima considerazione concerne gli indici sulle dimension table, che dovranno essere aggiornati a seguito delle modifiche intervenute. La tecnica più efficace dipende dal tipo di indice e dalla percentuale di dati modificati; euristicamente, si consiglia di ricostruire un indice *ex-novo* quando la percentuale dei dati modificati è superiore al 15% del totale (Kimball, 1998).

## 10.4 Alimentazione delle fact table

La procedura di alimentazione delle fact table è molto simile a quella delle dimension table; a parte la sua prima esecuzione, all'atto dell'inizializzazione degli schemi a stella, essa ha sempre carattere incrementale.

L'aggiornamento della fact table segue sempre l'aggiornamento delle dimension table correlate al fine di rispettare i vincoli di integrità referenziale tra le chiavi di queste ultime e le corrispondenti chiavi esterne della fact table.

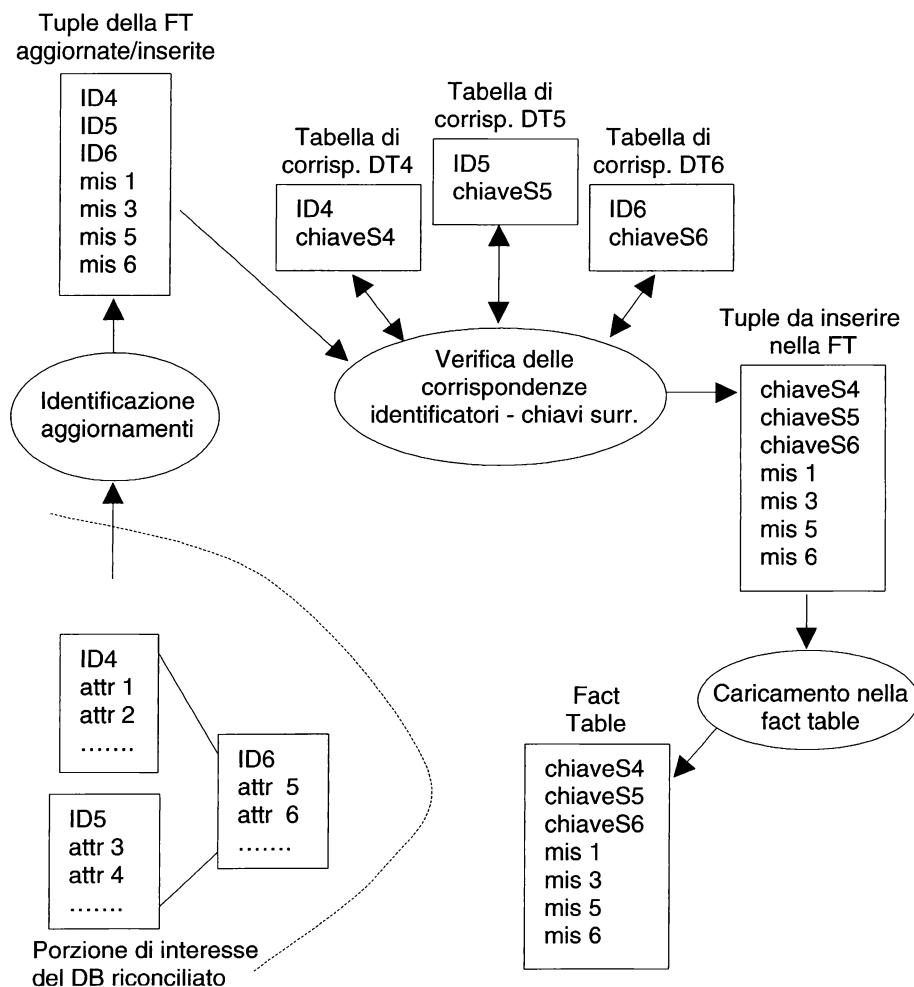


**Figura 10.12** Il processo di alimentazione delle dimension table.

Come per le dimension table, per identificare la corrispondenza tra gli attributi dello schema riconciliato e quelli della fact table è necessario fare riferimento alla documentazione di progetto logico. I valori delle chiavi surrogate da inserire nelle tuple della fact table possono invece essere determinati utilizzando le tabelle di corrispondenza predisposte durante l'alimentazione delle dimension table. Il processo di alimentazione è descritto graficamente in Figura 10.13 per una fact table le cui misure, mis1, mis3, mis5 e mis6, sono ricavate dagli attributi attr1, attr3, attr5 e attr6 dello schema riconciliato.

Se si è operato come descritto nel Paragrafo 10.1, le tuple memorizzate nel database riconciliato saranno dotate di un flag che indichi quale operazione ha generato il dato (inserimento, modifica, cancellazione) e di conseguenza come questo dovrà essere trattato nel DW. Sebbene molto più rare degli inserimenti, le modifiche e le cancellazioni di tuple già caricate nel DW sono sempre possibili a causa di errori rilevati e corretti nei dati opera-

zionali o più frequentemente a causa degli aggiornamenti retrospettivi discussi nel Paragrafo 6.1.6. A tal proposito è necessario distinguere diversi scenari: se si è deciso di modellare il solo tempo di validità, più eventi registrati in istanti diversi ma verificatisi nello stesso istante devono essere associati alla stessa tupla dando così vita a una modifica. Al contrario, se si è deciso di modellare anche (o solo) il tempo di transazione non potranno verificarsi aggiornamenti retrospettivi poiché le modifiche agli eventi già verificatisi vengono trattati come nuovi sulla base della coordinata temporale di registrazione. Con riferimento all'esempio delle iscrizioni universitarie (vedi Figura 6.24), le iscrizioni fatte a Bologna il 5 settembre 2005, per il corso di laurea di laurea in Ingegneria elettronica, nell'anno accademico 05-06 portano a eventi diversi a seconda della data in cui vengono registrati.



**Figura 10.13** Il processo di alimentazione delle fact table.

Poniamo infine l'attenzione sulla modalità di gestione delle modifiche, per le quali sono possibili due scenari: *sostituzione* e *accumulazione*. Nel primo, se un evento già registrato viene successivamente modificato, i nuovi valori che lo caratterizzano vengono sostituiti ai precedenti; nel secondo, i nuovi valori vengono invece accumulati (per esempio somma delle iscrizioni relative a una certa data). Mentre la sostituzione sembra ovvia nel caso in cui la modifica si sia generata a fronte della correzione di un errore presente nei dati operazionali, l'accumulazione si presta all'utilizzo nel caso in cui la modifica sia stata generata dalla presenza di eventi registrati in momenti diversi ma relativi allo stesso tempo di validità.

La scelta di una delle due soluzioni, o di entrambe, dipende innanzitutto dalla possibilità di discriminare le cause che hanno determinato la modifica dell'evento, ma è influenzata anche da altri fattori che coinvolgono l'intera struttura del processo di alimentazione e richiede un'analisi dettagliata basata sulle caratteristiche dello specifico progetto. A titolo di esempio si fa notare che se c'è una differenza tra la grana temporale del livello riconciliato e del fatto, ossia se un evento del fatto è determinato da più eventi del livello riconciliato, la modifica non potrà concretizzarsi con una semplice sostituzione delle misure dell'evento.

## 10.5 Alimentazione delle viste materializzate

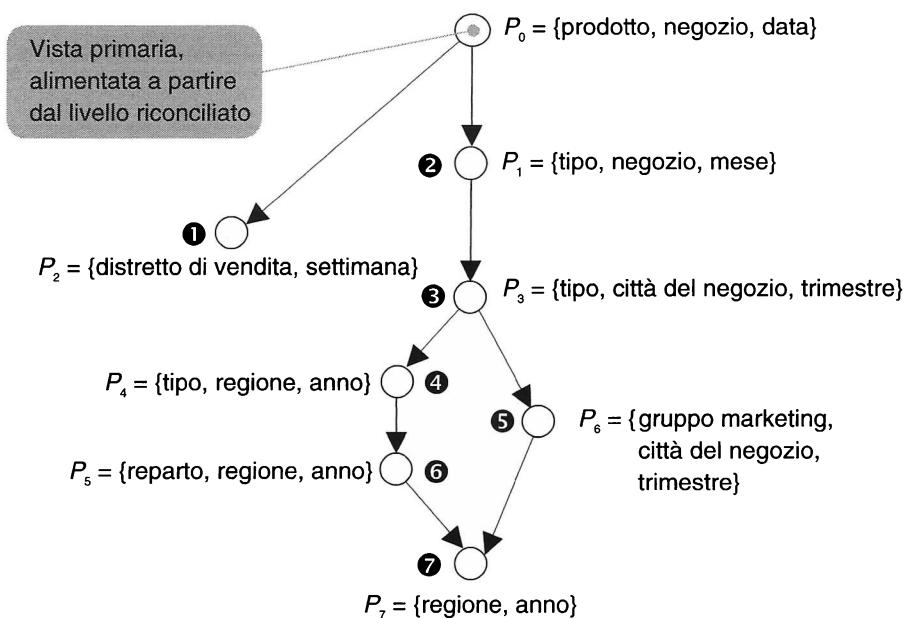
Gli aggiornamenti della fact table primaria devono essere propagati anche alle relative viste materializzate. Tale operazione è, dal punto di vista logico, molto semplice dato che una vista materializzata può essere in teoria alimentata tramite una interrogazione di aggregazione eseguita sulla fact table primaria. Questa soluzione all'aggiornamento, ossia l'eliminazione e la ricostruzione *ex-novo* della vista, viene effettivamente impiegata quando la quantità dei dati su cui operare lo consente; tuttavia, in molti casi occorre utilizzare tecniche più sofisticate onde contenere i tempi di aggiornamento.

Un primo metodo per abbattere i costi di aggiornamento delle viste consiste nel considerare l'ordinamento di roll-up tra pattern di aggregazione (Paragrafo 5.3.2), che rende possibile alimentare una vista secondaria a partire da altre viste secondarie. Riprendendo in maniera informale quanto già detto nel Paragrafo 9.2.1, se il pattern della vista  $v$  che si vuole alimentare è più grossolano di quello di un'altra vista  $v'$  già aggiornata e le misure di  $v$  sono ricavabili da quelle di  $v'$ , allora  $v$  può essere alimentata mediante un'interrogazione su  $v'$ . Ovviamente alimentare una vista a partire da una tabella più aggregata (e quindi con cardinalità più bassa) rispetto alla fact table comporta una riduzione dei costi.

In Figura 10.14 è riportata una porzione del reticolo multidimensionale per lo schema delle vendite già proposto in Figura 9.20. Assumendo che a ogni pattern corrisponda una vista materializzata e che le misure delle viste siano sempre ricavabili dalle misure delle viste a pattern più fini, la figura riporta un corretto ordine di aggiornamento delle viste: l'interrogazione di aggiornamento per una vista  $v$  viene eseguita sulla vista  $v'$  con cardinalità minima tra quelle collegate direttamente a  $v$ . Esistono più soluzioni equivalenti poiché l'aggiornamento di alcune viste è indipendente da quello di altre.

In molti casi, tuttavia, l'esecuzione di un'interrogazione di raggruppamento che coinvolga un'intera fact table ha costi troppo elevati e si rende quindi necessario adottare anche per le viste una tecnica di aggiornamento incrementale. In questo caso le stesse infor-

mazioni utilizzate per l'aggiornamento incrementale della fact table primaria possono essere utilizzate anche per le viste: esaminando le informazioni codificate nelle gerarchie è possibile individuare quali record di una vista verranno modificati in conseguenza del verificarsi di un evento primario. Per esempio, in Figura 10.15 è riportata una possibile istanza della fact table primaria relativa al fatto delle vendite con pattern primario {prodotto, negozio, data} e della relativa vista materializzata sul pattern {tipo, regione, data}; il verificarsi di un nuovo evento di vendita comporta anche la modifica del corrispondente record della vista materializzata.



**Figura 10.14** Una possibile sequenza di aggiornamento delle viste materializzate costruite sulla vista primaria.

Per viste aggregate sulla gerarchia temporale, risulta inevitabile apportare modifiche alle tuple che modellano gli eventi secondari relativi al periodo aggregato corrente. Se per esempio consideriamo nuovamente lo schema delle vendite (alimentato quotidianamente), la vista sul pattern {tipo, regione, mese} dovrà essere aggiornata in media 30 volte ogni mese. In questo caso due sono gli approcci possibili:

- Non includere i dati del mese corrente nella vista aggregata. In questo caso si evitano i problemi legati alle modifiche ed è sufficiente predisporre un processo mensile che effettui l'aggiornamento della vista.
- Includere una tupla contenente i dati relativi al mese corrente nella tabella aggregata. In questo caso è necessario predisporre un processo giornaliero che aggiorni i dati relativi al periodo in corso. Se la durata del periodo non è eccessiva è normalmente preferibile eliminare i record a esso relativi ricalcolandoli ogni notte.

Il problema dell'aggiornamento delle viste aggregate diventa particolarmente oneroso in presenza di eventi particolari che comportino la modifica di una larga parte dei dati (per esempio, una ristrutturazione delle zone di vendita che implichi la ridistribuzione dei fatturati). In questo caso è consigliabile eliminare completamente le viste coinvolte e ricalcolarle *ex-novo*.

VENDITE Nuova tupla	<b>chiaveN</b>	<b>chiaveD</b>	<b>chiaveP</b>	qtà venduta	incasso	.....
	1	1	1	170	85	.....
	2	2	2	320	160	.....
	<b>3</b>	<b>1</b>	<b>3</b>	<b>412</b>	<b>200</b>	<b>.....</b>
NEGOZIO	<b>chiaveN</b>	negozi	città	regione	.....	
	1	COOP1	Bologna	Em. Romagna	.....	
	2	COOP2	Roma	Lazio	.....	
	3	COOP3	Forlì	Em. Romagna	.....	
	.....	.....	.....	.....	.....	
PRODOTTO	<b>chiaveP</b>	prodotto	<b>Tipo</b>	categoria	marca	.....
	1	Latte Slurp	Latticini	Alimentari	Slurp	.....
	2	Ciocolocco	Dolciumi	Alimentari	Mutella	.....
	3	Yogurt Gnam	Latticini	Alimentari	Gnam	
DATA	<b>chiaveD</b>	Data	meße	anno	.....	
	1	2/9/2001	9/2001	2001	.....	
	2	3/10/2001	10/2001	2001	.....	
VENDITE1 Tupla modificata	<b>chiaveR</b>	<b>chiaveD</b>	<b>ChiaveT</b>	qtà venduta	incasso	.....
	1	1	1	582	285	.....
	2	2	2	320	160	.....
REGIONE	<b>chiaveR</b>	regione	.....			
	1	Em. Romagna	.....			
	2	Lazio	.....			
TIPO	<b>chiaveT</b>	tipo	categoria	.....		
	1	Latticini	Alimentari	.....		
	2	Dolciumi	Alimentari	.....		

**Figura 10.15** La modifica della tupla della fact table VENDITE primaria implica una modifica alla tupla della vista VENDITE1 identificabile utilizzando le informazioni codificate nelle gerarchie.