FONDAMENTI DI BASI DI DATI

Antonio Albano, Giorgio Ghelli, Renzo Orsini

12 febbraio 2019

INDICE

Prefazione

Dopo molti anni dalla pubblicazione della prima edizione del volume *Fondamenti di Basi di Dati* presso l'Editore Zanichelli, aggiornato con una seconda versione uscita nel 2005, è tempo di un'ulteriore ammodernamento, che coincide con la sua diffusione con un canale diverso da quello della carta stampata: il web, attraverso il sito http://fondamentidibasididati.it.

Riteniamo che questo materiale possa essere utile non solo per i classici corsi di *Basi di Dati*, fondamentali per le lauree in *Informatica* o *Ingegneria Informatica*, ma, data l'attenzione cha sta oggi avendo l'informatica in sempre piuù ampi settori della formazione e dell'istruzione ad ogni livello, in molti altri corsi di laurea e momenti formativi, in forma anche parziale, come abbiamo potuto sperimentare di persona durante questi ultimi anni.

Il passaggio alla nuova modalità di distribuzione, permettendo di mantenere aggiornati i contenuti, ci ha richiesto di modificare la struttura del sito di supporto al libro, che non avrà più la parte di errata corrige e di approfondimenti, ma conterrà materiale aggiuntivo utile per lo studio, come le soluzioni agli esercizi, i collegamenti ai software gratuiti, gli appunti del corso, e gli esempi scaricabili.

Organizzazione del testo

Il libro inizia presentando le ragioni che motivano la tecnologia delle basi di dati, ed i concetti principali che caratterizzano le basi di dati ed i sistemi per la loro gestione.

In maggior dettaglio, il Capitolo 2 si sofferma sulle nozioni fondamentali di modello informatico finalizzato al trattamento delle informazioni di interesse dei sistemi informativi delle organizzazioni e sui meccanismi d'astrazione per costruire modelli informatici. Il modello di riferimento è il modello ad oggetti, motivato non solo dalla sua naturalezza per la progettazione di basi di dati, ma anche per essere il modello dei dati dell'attuale tecnologia relazionale ad oggetti per basi di dati. Il formalismo grafico adottato si ispira all'*Unified Modeling Language*, UML, ormai uno standard dell'ingegneria del software.

Il Capitolo 3 presenta una panoramica del problema della progettazione di basi di dati, si sofferma sulle fasi dell'analisi dei requisiti e della progettazione concettua-

le usando il modello ad oggetti e il formalismo grafico proposto nel Capitolo 2, e descrive un metodo di lavoro per queste fasi.

I Capitoli 4 e 5 sono dedicati alla presentazione rigorosa del modello relazionale dei dati e ad un'introduzione alla teoria della normalizzazione. La scelta di dedicare questo spazio all'argomento è giustificata dal ruolo fondamentale che svolge il modello relazionale per la comprensione della tecnologia delle basi di dati e per la formazione degli addetti.

I Capitoli 6, 7 e 8 trattano il linguaggio relazionale SQL da tre punti di vista, che corrispondono ad altrettante categorie di utenti: (1) gli utenti interessati ad usare interattivamente basi di dati relazionali, (2) i responsabili di basi di dati interessati a definire e gestire basi di dati, (3) i programmatori delle applicazioni.

Il Capitolo 9 presenta una panoramica delle principali tecniche per la realizzazione dei sistemi relazionali. Vengono presentate in particolare la gestione delle interrogazioni e dei metodi di accesso e la gestione dell'atomicità e della concorrenza in sistemi centralizzati.

Ringraziamenti

L'organizzazione del materiale di questo testo è il risultato di molti anni di insegnamento deii corsi di *Basi di Dati* per le lauree in *Scienze dell'Informazione* e in *Informatica*.

Molte persone hanno contribuito con i loro suggerimenti e critiche costruttive a migliorare la qualità del materiale. In particolare si ringraziano i numerosi studenti che nel passato hanno usato le precedenti versioni del testo e Gualtiero Leoni per la sua collaborazione.

Si ringrazia infine l'Editore Zanichelli per il supporto che ci ha dato in questi anni, e, adesso che il libro è uscito dal suo catalogo, per il permesso di diffondere la nuova versione attraverso il web.

A. A.
G. G.
R. O.

Capitolo 1

SISTEMI PER BASI DI DATI

Spesso interagiamo con *basi di dati* senza saperlo: quando facciamo un prelievo dal conto corrente con il bancomat, quando facciamo un acquisto con la carta di credito, quando acquistiamo un biglietto ferroviario o prenotiamo un viaggio presso un'agenzia, quando usiamo il sito web di un dipartimento universitario per iscriversi ad un esame ecc. In tutte queste situazioni è richiesto l'accesso a certe raccolte di dati o la loro modifica per registrare l'effetto dell'operazione compiuta. In generale queste raccolte di dati sono gestite da organizzazioni che dedicano molte attività e risorse alla raccolta, archiviazione ed elaborazione di informazioni per fornire opportuni servizi. Negli ultimi anni, per l'evoluzione della tecnologia elettronica e la conseguente riduzione dei costi, si è assistito ad una crescente diffusione degli elaboratori elettronici per agevolare e potenziare le possibilità di trattamento delle informazioni. Questo fenomeno ha interessato organizzazioni di ogni tipo e dimensioni ed ha portato ad una richiesta sempre più ampia di sistemi dedicati a questo scopo. In seguito, dopo una precisazione sul ruolo del *sistema informativo* e del *sistema informatico* all'interno di un'organizzazione, vengono definite le nozioni di *base di dati* e di *sistema per la gestione di basi di dati* e vengono presentati i concetti fondamentali che verranno sviluppati nei capitoli successivi.

1.1 Sistemi informativi e informatici

Ogni organizzazione, per il proprio funzionamento, ha bisogno di disporre di informazioni che costituiscono una risorsa gestita da un proprio *sistema informativo*, del quale si propone la seguente definizione.

Definizione 1.1
Il sistema informativo di un'organizzazione è una combinazione di risorse, umane e materiali, e di procedure organizzate per la raccolta, l'archiviazione, l'elaborazione e lo scambio delle informazioni necessarie alle attività operative (*informazioni di servizio*), alle attività di programmazione e controllo (*informazioni di gestione*), e alle attività di pianificazione strategica (*informazioni di governo*).

Con il termine *sistema* si evidenzia il fatto che esiste un insieme organizzato di elementi, di natura diversa, che interagiscono in modo coordinato, mentre con *informativo* si precisa che tutto ciò è finalizzato alla gestione delle informazioni e quindi le inte-

razioni che preme evidenziare sono quelle dovute a scambi di informazioni (*flussi informativi*).

Per esempio, un'industria manifatturiera gestisce informazioni per svolgere attività che includono:

1. gestione degli ordini e dei pagamenti dei prodotti venduti;
2. gestione degli ordini e dei pagamenti ai fornitori di materiali;
3. gestione del magazzino;
4. programmazione della produzione;
5. controllo di gestione;
6. pianificazione di nuovi prodotti.

Le informazioni di un'organizzazione, una volta ridotte a dati mediante un processo di interpretazione, quantificazione e formalizzazione, possono essere trattate automaticamente con gli elaboratori elettronici. La riduzione dei costi della tecnologia informatica ha diffuso largamente questa possibilità, rendendo più accurate e rapide le procedure e potenziando i modi di elaborazione delle informazioni.

Con il termine *sistema informativo automatizzato* si indica la parte del sistema informativo realizzata impiegando strumenti informatici e della comunicazione. In generale, raramente il sistema informativo di un'organizzazione viene completamente automatizzato sia a causa di problemi tecnici che per motivi di convenienza economica.

Spesso nella letteratura si parla di sistemi informativi pensando alla parte automatizzata. Per brevità e per evitare confusione useremo il termine *sistema informatico* per riferirci ai sistemi informativi automatizzati.

Un sistema informatico, per fornire i servizi attesi dagli utenti, prevede i seguenti componenti principali (Figura 1.1):

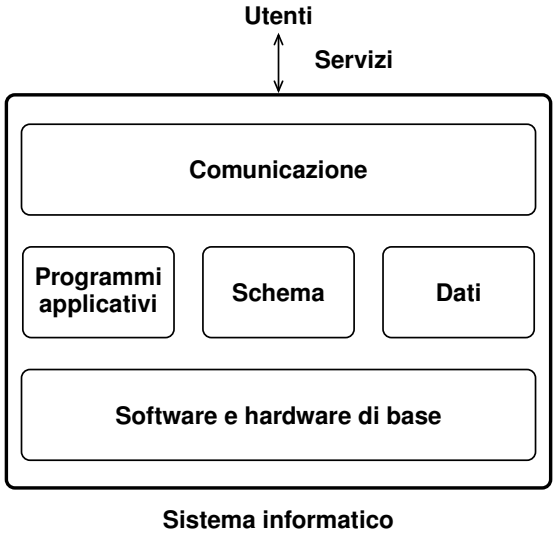

Figura 1.1: Componenti del sistema informatico

- il *software* e *l'hardware di base*;
- una *base di dati*, che contiene una rappresentazione del patrimonio informativo dell'organizzazione;
- uno *schema*, che descrive la struttura della base di dati, le operazioni per agire su di essa e le restrizioni sui valori memorizzabili nella base di dati e sui modi in cui essi possono evolvere nel tempo (*vincoli di integrità*). Lo schema viene usato dal sistema per garantire un uso corretto della base di dati;
- i *programmi applicativi*, che forniscono i servizi agli utenti eseguendo un certo insieme di operazioni sulla base di dati.
- la *comunicazione*, che permette l'accesso ai servizi del sistema informatico ad utenti e programmi.

In questo testo vengono prese in considerazione, tra le informazioni trattate da un'organizzazione, solo quelle strutturate, con un formato predeterminato, e di carattere prevalentemente globale, cioè utili a più reparti.

1.2 Evoluzione dei sistemi informatici

L'architettura dei sistemi informatici per la gestione di informazioni strutturate è cambiata molto negli ultimi quarant'anni con l'evolvere della tecnologia degli elaboratori elettronici e degli strumenti per la gestione di dati permanenti. Vediamo i passaggi più significativi.

Primo stadio: applicazioni operative

Agli inizi degli anni '60, le applicazioni riguardavano le attività delle funzioni amministrative che richiedevano l'elaborazione sistematica e ripetitiva di grandi quantità di dati, come il calcolo delle paghe e degli stipendi o l'emissione delle fatture. Successivamente, a questo primo nucleo di applicazioni, si sono aggiunte altre più complesse, come la gestione del magazzino e la contabilità dei clienti, che ancora oggi costituiscono il nucleo essenziale di ogni sistema informatico nelle aziende.

Secondo stadio: servizi informatici di funzione

Dalla fine degli anni '60, l'interesse si è spostato anche sull'elaborazione delle informazioni di supporto ai responsabili delle varie funzioni aziendali, con lo sviluppo di applicazioni per la contabilità generale, per il controllo di gestione e per la valutazione del funzionamento dell'azienda.

Questi tipi di sistemi informatici rispondevano quindi a due precise esigenze:

- rendere più efficienti le attività ripetitive dei livelli esecutivi dell'azienda;
- permettere una migliore gestione dell'azienda fornendo ai responsabili le informazioni sintetiche sull'andamente delle attività controllate.

Terzo stadio: servizi informatici per l'organizzazione

La realizzazione di servizi informatici per le funzioni aziendali non presupponeva l'integrazione dei dati in comune alle diverse funzioni e quindi comportava sia la duplicazione di dati, con il rischio di copie incoerenti, sia una limitata possibilità di correlare dati settoriali per generare informazioni di interesse globale per l'organizzazione.

A partire dagli anni '70, il progresso della tecnologia ha reso disponibili nuovi strumenti informatici, i *sistemi per la gestione di basi di dati* (*Data Base Management System, DBMS*), che, rendendo possibile una gestione integrata dei dati, interessavano ogni livello delle organizzazioni: i dati trattati automaticamente non erano suddivisi per interessi settoriali, ma venivano trattati globalmente, in modo che ciascuna informazione, benché rappresentata una sola volta, era utilizzabile per attività diverse del sistema informativo. Si è passati quindi da *sistemi informatici settoriali* a *sistemi informatici per l'organizzazione*, con notevoli riflessi sulla struttura dell'organizzazione stessa, in quanto un impiego razionale della tecnologia informatica comporta necessariamente una revisione del modo di funzionare della struttura organizzativa che deve utilizzarla.

I vantaggi più evidenti di questa soluzione sono:

1. *Integrazione dei dati*. Invece di avere per ogni applicazione una coppia (dati, programmi), con dati in generale non completamente distinti da quelli usati da un'altra applicazione, si vuole prevedere un'unica raccolta di dati comuni, che costituiscono le informazioni di base, e tanti programmi che realizzano le applicazioni operando solo sui dati di loro interesse. Questo obiettivo comporta i seguenti vantaggi:

   Disponibilità dei dati. Quando i dati non sono organizzati in funzione di una specifica applicazione, è più semplice renderli disponibili ad usi diversi.

   Limitazione delle ridondanze. L'esistenza di un'unica raccolta di dati, accessibili con modalità diverse, elimina la necessità di duplicazioni, che comportano maggiore occupazione di memoria e possibilità di inconsistenze fra i dati.

   Efficienza. La gestione integrata dei dati si presta allo sviluppo di strumenti per ottimizzare globalmente la loro organizzazione fisica e, quindi, l'efficienza complessiva del sistema.

2. *Flessibilità della realizzazione*. Le modifiche ad un sistema informatico funzionante sono inevitabili e devono poter essere fatte senza dover intervenire sulla realizzazione in modo massiccio. Infatti, il progetto e la realizzazione di un sistema informatico sono, in generale, dei compiti complessi: dal momento in cui si raccolgono le specifiche, al momento in cui si ha una prima versione funzionante della realizzazione, può trascorrere un intervallo di tempo sufficientemente lungo perché ci sia una modifica dei requisiti iniziali. Un'altra ragione, che richiede un'evoluzione della realizzazione, è che un sistema informatico di supporto a sistemi informativi si progetta in modo incrementale: si parte automatizzando un nucleo di funzioni essenziali e, successivamente, il sistema si amplia inglobando nuovi dati e funzioni. Esiste, infine, un'altra ragione, meno prevedibile, che può causare un'evoluzione

’60	Sistemi gerarchici (IMS e System 2000)
’70	Sistemi reticolari secondo la proposta CODASYL (IDMS, IDS II, DM IV, DMS 1100 ecc.) E.F. Codd propone il modello relazionale
’80	Sistemi relazionali (System R, Ingres, Oracle, SQL/DS, DB2, Informix ecc.)
’90	Sistemi relazionali distribuiti Architetture cliente/servente e parallele Sistemi ad oggetti e relazionali ad oggetti (GEMSTONE, ONTOS, Objectstore, O_2, Illustra, UniSQL ecc.)
1995	Integrazione con Internet

Tabella 1.1: Prospetto cronologico dell’evoluzione dei sistemi per basi di dati.

delle specifiche: quando il sistema funziona in modo soddisfacente, gli utenti vengono stimolati nella loro immaginazione e intravedono nuove possibilità d’impiego del sistema informatico, modificando così i requisiti iniziali.

Inizialmente i sistemi per basi di dati erano di tipo centralizzato, ovvero la base di dati era gestita da un unico elaboratore. Agli inizi degli anni ’90, invece, l’evoluzione della tecnologia ha reso possibile anche la realizzazione di sistemi per basi di dati distribuite, su rete locale o geografica, che consentono di avere una visione unica dei dati, indipendentemente da dove essi siano fisicamente memorizzati.

In Tabella 1.1 sono riportati i passaggi più significativi dell’evoluzione della tecnologia delle basi di dati.

Quarto stadio: servizi informatici per la pianificazione strategica

A partire dagli anni ’80, infine, lo sviluppo della tecnologia ha permesso una progressiva copertura delle esigenze informative per le attività di programmazione e di pianificazione strategica, allargando ulteriormente lo spettro delle possibilità di applicazione della tecnologia informatica nelle organizzazioni per l’automazione dei sistemi informativi.

Quinto stadio: servizi informatici per la cooperazione fra organizzazioni

A partire dalla fine degli anni ’90, infine, con la grande diffusione di standard di comunicazione dovuti allo sviluppo della rete Internet e del Web, sono iniziati a diffondersi protocolli di cooperazione fra sistemi informatici diversi, e strumenti che semplificano lo scambio di dati e le richieste di servizi fra di loro (*Web services*). L’obiettivo è di facilitare lo sviluppo di applicazioni di commercio elettronico che prevedano l’interazione di organizzazioni separate, con sistemi informatici eterogenei, come le aziende produttrici, quelle di intermediazione, di logistica, pubbliche amministrazioni ecc.

1.3 Tipi di sistemi informatici

Illustriamo le differenze fra le finalità di due tipici sistemi informatici delle organizzazioni, usando come esempio il caso di una catena di supermercati con negozi sul territorio nazionale.

1.3.1 Sistemi informatici operativi

Ogni negozio utilizza una base di dati operazionale (o transazionale) che raccoglie informazioni sugli effetti delle operazioni di routine necessarie quotidianamente per condurre le attività aziendali. Per esempio, per avere informazioni sui prodotti di cui dispone, per stabilire a quale prezzo vendere il prodotto quando viene fatto un acquisto, come stampare lo scontrino, come aggiornare il saldo del venduto ad ogni cassa e come aggiornare lo stato del magazzino. È facile immaginare come le cose si complichino quando sia necessario tener conto anche del fatto che i pagamenti possono essere fatti non solo in contanti, ma anche con bancomat o con carta di credito. Quando il cliente ha ricevuto lo scontrino, il sistema garantisce che la base di dati ha subito tutte le modifiche necessarie per tener conto di tutti gli effetti che ha prodotto l'evento che si è verificato alla cassa. Il termine *transazionale* si usa per riferirsi al fatto che le interazioni con il sistema avvengono mediante *transazioni*, un evento che innesca sulla base di dati una sequenza S di azioni che devono tutte andare a buon fine perché essa rimanga coerente, ovvero rifletta correttamente gli effetti dell'evento. In caso di un malfunzionamento che interrompa l'esecuzione di S, il meccanismo della transazione garantisce che la base di dati si troverà nello stato in cui si trovava prima che S iniziasse.

Le basi di dati operazionali sono gestite dalle applicazioni che fanno parte dei cosiddetti *sistemi informatici operativi* o *sistemi di elaborazione di transazioni* (*Transaction Processing Systems, TPS*). Le applicazioni assistono i dipendenti al livello operativo nello svolgimento delle attività di loro competenza (Figura 1.2).

Un esempio di questo tipo di soluzione sono i sistemi ERP, *Enterprise Resource Planning*, acronimo che non spiega la finalità di questi sistemi, che non è la pianificazione delle risorse aziendali, ma l'integrazione dei processi aziendali in un unico sistema software che possa soddisfare tutti i requisiti informativi dell'azienda usando una base di dati centralizzata.

1.3.2 Sistemi informatici direzionali

Le direzioni intermedia e operativa della catena di supermercati hanno bisogno di analizzare i dati di ogni negozio per produrre rapporti di riepilogo sull'andamento delle vendite e sul funzionamento del supermercato per prendere decisioni che riguardano tutta l'azienda al livello nazionale. Assumiamo che i rapporti da produrre riguardino (a) l'andamento delle operazioni di base dell'azienda nel breve periodo (settimanale, mensile e annuale), (b) siano standard e ripetitivi, (c) siano relativamente poco complessi, (d) siano producibili a partire dalle basi di dati operazionali. Per

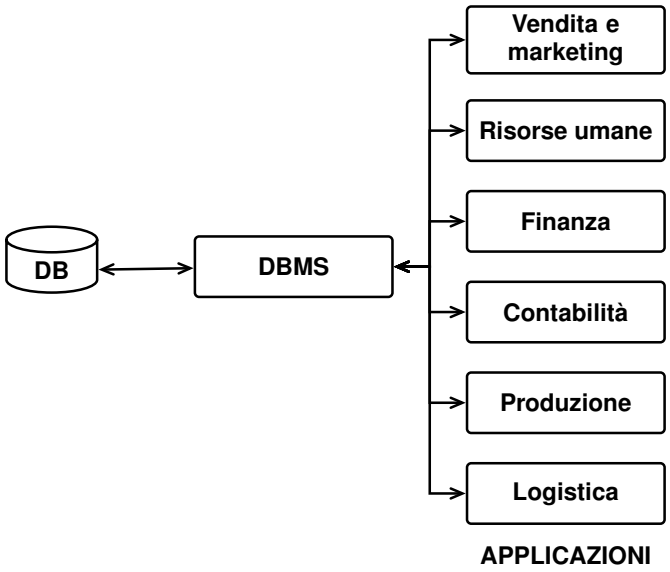

Figura 1.2: Sistemi informatici operativi

esempio, conoscere le vendite del supermercato con sede a Pisa nelle ultime quattro settimane.

Le informazioni sono ricavate dalle basi di dati operazionali usando applicazioni specializzate per assistere le direzioni intermedia e operativa dell'azienda nelle funzioni di programmazione e controllo.

Per prendere decisioni, le direzioni intermedia ed alta della catena di supermercati hanno invece bisogno di analisi storiche dell'andamento degli affari che comportano la produzione interattiva di rapporti di sintesi non programmati, più complessi e da punti di vista diversi, per scoprire situazioni anomali o tendenze interessanti, che non possono essere prodotti a partire dalle basi di dati operazionali perché esse di solito rendono disponibili solo i dati più recenti oppure perché le operazioni necessarie per produrre i rapporti sono complesse e rallenterebbero le operazioni di cassa oltre il tollerabile.

Per soddisfare queste esigenze, i dati storici vengono integrati con dati provenienti da fonti esterne e organizzati opportunamente in un altro tipo di base di dati, detta *data warehouse*, gestita separatamente con un opportuno sistema che metta a disposizione strumenti adatti per fare facilmente analisi interattive dei dati. Mentre una base di dati operazionale è aggiornata tempestivamente ad ogni verificarsi di un evento aziendale, il *data warehouse* viene aggiornato periodicamente con nuovi dati storici, o esterni, perché ai fini delle analisi di supporto alle decisioni non è necessario disporre di dati accurati al 100%.

Mentre nel caso precedente le esigenze sono soddisfacibili con richieste specifiche ("Trovare i possessori di carta di credito che hanno speso più di 50 euro in alcolici nel mese di gennaio"), un'altra esigenza dell'alta dirigenza è di scoprire automaticamente qualche aspetto interessante di un insieme di dati con tecniche di *data mining* ("Qual

è il profilo generale dei possessori di carta di credito che traggono profitto dalle promozioni?"). Il *data mining* è una fase di un processo interattivo e iterativo che cerca di estrarre modelli da un insieme di dati utili ai dirigenti per prendere decisioni. Un modello, in questo contesto, è una rappresentazione concettuale che evidenzia in una forma opportuna certe caratteristiche di informazioni implicite, sconosciute a priori e potenzialmente utili, presenti nei dati.

I dati per produrre le informazioni che aiutino i dirigenti a prendere decisioni sono gestiti dai cosiddetti *sistemi di supporto alle decisioni* (*Decision Support Systems, DSS*) (Figura 1.3).

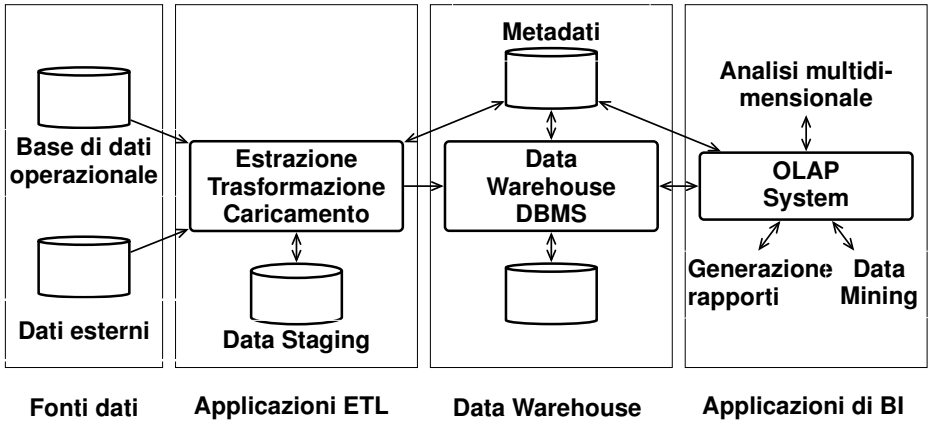

Figura 1.3: Sistemi informatici direzionali

1.4 I sistemi per basi di dati

Il termine *base di dati* viene spesso usato per riferirsi ad un qualsiasi insieme di dati permanenti trattati con un elaboratore elettronico, ma qui verrà usato con il seguente significato.

Definizione 1.2

Una base di dati è una raccolta di dati permanenti, gestiti da un elaboratore elettronico, suddivisi in due categorie:

1. i *metadati*, ovvero lo *schema* della base di dati (*database schema*), una raccolta di definizioni che descrivono la struttura dei dati, le restrizioni sui valori ammissibili dei dati (*vincoli d'integrità*), le relazioni esistenti fra gli insiemi, e a volte anche alcune operazioni eseguibili sui dati. Lo schema va definito prima di creare i dati ed è indipendente dalle applicazioni che usano la base di dati;
2. i *dati*, le rappresentazioni di certi fatti conformi alle definizioni dello schema, con le seguenti caratteristiche:

- sono organizzati in insiemi omogenei, fra i quali sono definite delle relazioni. La struttura dei dati e le relazioni sono descritte nello schema con opportuni meccanismi di astrazione dipendenti dal *modello dei dati* (*data model*) utilizzato, che prevede anche operatori per estrarre elementi da un insieme e per conoscere quelli che, in altri insiemi, sono in relazione con loro;
- sono molti, in assoluto e rispetto ai metadati, e non possono essere gestiti tutti contemporaneamente in memoria temporanea;
- sono permanenti, cioè, una volta creati, continuano ad esistere finché non sono esplicitamente rimossi; la loro vita quindi non dipende dalla durata delle applicazioni che ne fanno uso;
- sono accessibili mediante *transazioni* (*transactions*), unità di lavoro atomiche che non possono avere effetti parziali;
- sono protetti sia dall'accesso di utenti non autorizzati, sia da corruzione dovuta a malfunzionamenti hardware e software;
- sono utilizzabili contemporaneamente da utenti diversi.[1]

Esempio 1.1 Per chiarire i concetti esposti, si mostra un semplice esempio di base di dati che memorizzi informazioni relative a studenti ed esami di un'università. Ogni insieme è visto come una tabella con tante colonne quanti sono gli attributi d'interesse.

Questo tipo di base di dati è detto *relazionale* e sarà discusso in modo approfondito nel Capitolo 4.

Studenti

Nome	**Matricola**	**Provincia**	**Nascita**
Isaia	71523	Pisa	1980
Rossi	67459	Lucca	1981
Bianchi	79856	Livorno	1980
Bonini	75649	Pisa	1981

ProveEsami

Materia	**Candidato**	**Data**	**Voto**
BD	71523	12/01/01	28
BD	67459	15/09/02	30
IA	79856	25/10/03	30
BD	75649	27/06/01	25
IS	71523	10/10/02	18

1. Il termine "utente" viene usato sia con il significato di persona che accede ai dati da un terminale in modo interattivo usando un opportuno linguaggio, sia con il significato di programma applicativo che contiene istruzioni per l'accesso ai dati.

I metadati riguardano le seguenti informazioni:

1. il fatto che esistono due collezioni di interesse Studenti e ProveEsami;
2. la struttura degli elementi di queste due collezioni (intestazione delle tabelle): ogni studente ha una matricola, di tipo intero, che lo contraddistingue, un nome, di tipo stringa, un anno di nascita, una città di residenza; ogni esame ha una materia, la matricola dello studente, la data dell'esame e il voto ottenuto;
3. il fatto che ad ogni esame (inteso come l'evento in cui uno studente viene esaminato) corrisponde uno studente con la matricola specificata, e ad ogni studente corrispondono uno, nessuno, o più esami;
4. alcuni vincoli sui valori ammissibili, quali il fatto che il valore della matricola identifica una riga della tabella Studenti e i valori della matricola e della materia identificano una riga della tabella ProveEsami, oppure che il voto deve essere un numero intero compreso fra 18 e 30.

I dati invece sono le righe delle tabelle che contengono le informazioni relative ai singoli studenti ed esami.

Le caratteristiche delle basi di dati sono garantite da un *sistema per la gestione di basi di dati* (*Data Base Management System, DBMS*), che ha il controllo dei dati e li rende accessibili agli utenti autorizzati.

Definizione 1.3
Un DBMS è un sistema centralizzato o distribuito che consente (*a*) di definire schemi di basi di dati, (*b*) di scegliere le strutture dati per la memorizzazione e l'accesso ai dati, (*c*) di memorizzare, recuperare e modificare i dati, interattivamente o da programmi, ad utenti autorizzati e rispettando i vincoli definiti nello schema.

In Figura 1.4 è mostrata una versione semplificata della struttura di un DBMS. È interessante ricorrere ad un'analogia con concetti dei linguaggi di programmazione per chiarire alcuni dei concetti finora introdotti: modello dei dati, linguaggio per basi di dati, base di dati e schema, sistema per la gestione di basi di dati.

Modello dei dati

Un *modello dei dati* è un insieme di meccanismi di astrazione per definire una base di dati, con associato un insieme predefinito di operatori e di vincoli d'integrità.

I modelli dei dati adottati dai moderni sistemi commerciali sono il *relazionale* e quello *ad oggetti*. Le loro caratteristiche verranno presentate nel prossimo capitolo. I meccanismi di astrazione di un modello dei dati corrispondono ai meccanismi di astrazione per rappresentare i dati in un linguaggio di programmazione e vanno tenuti distinti dal modo in cui sono trattati in un particolare linguaggio per basi di dati, nello stesso modo in cui nei linguaggi di programmazione si parla, ad esempio, del meccanismo

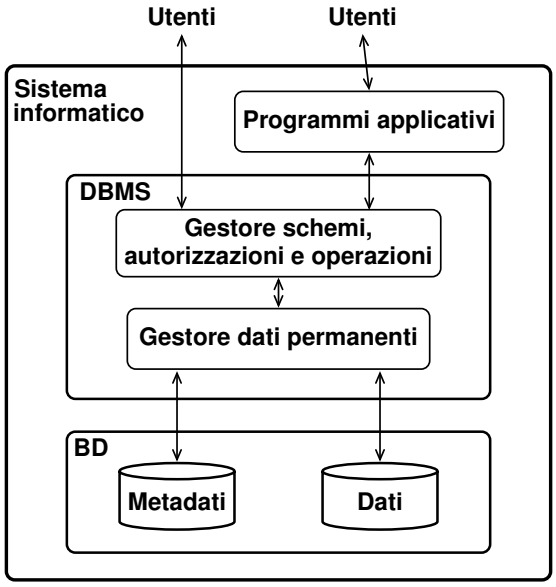

Figura 1.4: Il sistema informatico e il sistema per la gestione di basi di dati

di astrazione del *record* e del *file*, prescindendo dalla forma sintattica che prende in un particolare linguaggio.

Un buon modello dei dati dovrebbe essere caratterizzato da:

- *espressività*: il modello dovrebbe permettere di rappresentare in modo naturale e diretto il significato di ciò che si sta modellando;
- *semplicità d'uso*: il modello dovrebbe essere basato su di un numero minimo di meccanismi semplici da utilizzare e da comprendere;
- *realizzabilità*: i meccanismi del modello di astrazione, e i relativi operatori per la manipolazione dei dati, devono essere realizzabili in modo efficiente su di un elaboratore elettronico.

Linguaggio per basi di dati

Quando si pensa ad un linguaggio di programmazione si pensa ad un linguaggio che prevede un sistema di tipi con opportuni operatori e una struttura del controllo. Quando si pensa ad un linguaggio per base di dati, invece, per motivi storici, si usa distinguere fra:

1. il linguaggio per la definizione dello schema, ovvero della struttura delle collezioni dei dati (*Data definition language, DDL*);
2. il linguaggio degli operatori del modello dei dati, che permette di accedere ai dati e di modificarli, ma è in genere privo dei costrutti tipici dei linguaggi di programmazione (funzioni, struttura del controllo, variabili ecc.) (*Data manipulation language, DML*);

3. il linguaggio per la codifica delle applicazioni che richiedono lo sviluppo di programmi che usano la base di dati. Si tratta di solito di un linguaggio tradizionale, tipo C o Java, esteso aggiungendovi gli operatori del DML (*host language*);
4. il linguaggio di interrogazione (*query language*) per il recupero e la modifica di dati interattivamente, ma non per lo sviluppo di applicazioni.

Sono stati proposti sia linguaggi che offrono alcune delle funzionalità sopra elencate, come il linguaggio SQL che verrà descritto nei Capitoli 6-8, sia linguaggi di programmazione completi per basi di dati che offrono tutte le funzionalità sopra descritte: definizione dello schema, interrogazione e modifica dei dati, realizzazione di applicazioni complete. Questi linguaggi possono essere visti come dei linguaggi di programmazione arricchiti con la possibilità di definire alcuni dati come persistenti e la disponibilità di operatori e tipi di dati adatti a manipolare collezioni di dati omogenei. Un esempio di questi linguaggi verrà presentato nel Capitolo 8.

Base di dati e schema

Secondo la visione tradizionale, i dati di una base di dati ed il relativo schema corrispondono rispettivamente ad un insieme di variabili (che denotano insiemi modificabili di valori permanenti) che più applicazioni possono leggere e modificare in maniera concorrente, ed alla definizione del tipo di tali variabili, che tuttavia in questo caso è anch'essa permanente e in generale utilizzabile dalle applicazioni.

Gli orientamenti più recenti, invece, prevedono di trattare nello schema sia dati che procedure. In questa visione, uno schema è analogo ad un insieme di definizioni non solo di tipi e variabili, ma anche di procedure scritte nel linguaggio del DBMS, accessibili da tutte le applicazioni che fanno riferimento a tale schema. Esempi verranno mostrati nel Capitolo 7.

Sistema per la gestione di basi di dati

È il sistema, hardware e software, che consente la definizione e l'uso di basi di dati in un opportuno linguaggio. In altre parole, è la macchina astratta il cui linguaggio è il *linguaggio per basi di dati*. Esso, quindi, è analogo al compilatore, o l'interprete, di un certo linguaggio, più l'insieme dei moduli attivi durante l'esecuzione dei programmi (*run time system*). L'architettura e le tecniche di realizzazione dei DBMS saranno discusse nel Capitolo 9.

Si passa ora ad esaminare le funzionalità che caratterizzano un DBMS. Non tutti i sistemi offrono tutte le funzionalità che si prenderanno in considerazione, in particolare i DBMS previsti per calcolatori personali ne sacrificano alcune per ragioni di costo (tipicamente la gestione delle transazioni e l'accesso concorrente ai dati), ma l'elenco che segue include quelle funzionalità da considerarsi irrinunciabili per prodotti da usare nella gestione delle informazioni nelle organizzazioni.

1.5 Funzionalità dei DBMS

Si analizzano le principali funzionalità che un DBMS mette a disposizione per i seguenti scopi:

1. definizione della base di dati;
2. uso della base di dati;
3. controllo della base di dati;
4. distribuzione della base di dati;
5. amministrazione della base di dati.

1.5.1 Definizione della base di dati

Nei DBMS la base di dati è descritta separatamente dai programmi applicativi che ne fanno uso ed è utile distinguere tre diversi livelli di descrizione dei dati: il *livello fisico*, il *livello logico* e il *livello di vista logica*.

Al *livello fisico* viene descritto il modo in cui vanno organizzati fisicamente i dati nelle memorie permanenti e quali strutture dati ausiliarie definire per facilitarne l'uso. La descrizione di questi aspetti viene detta *schema fisico* o *interno* (*physical* o *internal schema*).

Al *livello logico* viene descritta la struttura degli insiemi di dati e delle relazioni fra loro, secondo un certo modello dei dati, senza nessun riferimento alla loro organizzazione fisica nella memoria permanente. La descrizione della struttura della base di dati viene detta *schema logico* (*logical schema*).

Al *livello di vista logica* viene definito come deve apparire la struttura della base di dati ad una certa applicazione. Questa descrizione viene anche detta *schema esterno* o *vista* (*external schema* o *user view*), per evidenziare il fatto che essa si riferisce a ciò che un utente immagina che sia la base di dati. Le differenze fra una vista e lo schema logico della base di dati riguardano sia gli insiemi di dati accessibili che la struttura dei dati. Mentre lo schema logico è unico, esistono in genere più schemi esterni, uno per ogni applicazione (o gruppo di applicazioni correlate), che permettono di vedere e modificare sottoinsiemi diversi, in generale non disgiunti, della base di dati (Figura 1.5).

Esempio 1.2 Per chiarire la differenza fra i tre livelli di descrizione dei dati, si consideri una base di dati per gestire informazioni sui docenti di un'università, di supporto alle attività dell'ufficio stipendi e della biblioteca. Al livello di vista logica, l'ufficio stipendi richiede una vista dei dati sui docenti che include i seguenti attributi: nome e cognome, codice fiscale, parametro e stipendio. La biblioteca richiede invece una vista dei dati sui docenti che include i seguenti attributi: nome e cognome, recapito telefonico. Al livello logico, i dati sui docenti sono descritti da un unico insieme di registrazioni che includeranno gli attributi

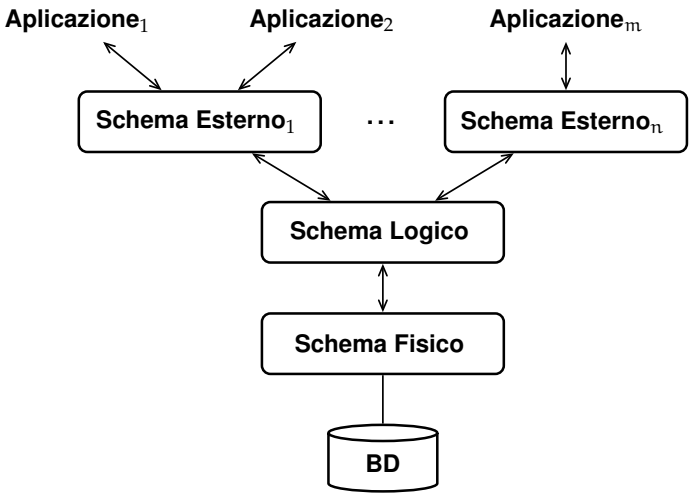

Figura 1.5: Livelli di descrizione dei dati

diversi che occorrono nelle due viste. Grazie al meccanismo degli schemi esterni, ogni applicazione vedrà poi solo i dati di sua competenza.

Ad esempio, lo schema logico di una base di dati relazionale è dichiarato come segue:

```
CREATE  TABLE Personale (
        Nome            CHAR(30),
        CodiceFiscale   CHAR(15),
        Stipendio       INTEGER,
        Parametro       CHAR(6),
        Recapito        CHAR(8) );
```

Al livello di vista logica, con un opportuno meccanismo di autorizzazioni, all'ufficio stipendi e alla biblioteca non viene consentito di accedere alla tabella Personale, ma di accedere solo ai dati di loro competenza definiti come

```
CREATE  VIEW    PersonalePerUfficioStipendi
    AS  SELECT  Nome, CodiceFiscale, Stipendio, Parametro
        FROM    Personale;

CREATE  VIEW    PersonalePerLaBiblioteca
    AS  SELECT  Nome, Recapito
        FROM    Personale;
```

Una *view* è una tabella calcolata da altre con opportuni operatori, che vedremo più avanti. Nell'esempio è stato usato solo l'operatore che proietta una tabella su alcune colonne rendendo così inaccessibili le altre; pertanto se agli addetti della

biblioteca viene concessa l'autorizzazione ad usare solo i dati della tabella PersonalePerLaBiblioteca, essi potranno conoscere solo gli attributi Nome e Recapito del personale.

Infine, al livello fisico, il progettista della base di dati fisserà un'organizzazione fisica per l'insieme dei dati dei docenti descritto al livello logico, scegliendone una fra quelle previste dal DBMS: ad esempio, deciderà di memorizzare l'insieme in modo sequenziale oppure con una tecnica *hash* usando come chiave il nome:

MODIFY Personale **TO HASH ON** Nome

Naturalmente fra i tre livelli di descrizione dei dati devono esistere delle precise e dichiarate corrispondenze. Esse vengono utilizzate dal DBMS per convertire le operazioni sui dati "virtuali" accessibili da uno schema esterno, in quelle sui dati dello schema logico e, quindi, sui dati realmente presenti nel sistema, memorizzati secondo lo schema fisico.

I tre schemi dei dati sono gestiti dal sistema e non fanno parte dei programmi delle applicazioni. Un programma, prima di accedere alla base di dati, deve comunicare al sistema, con opportune modalità, a quale schema esterno fa riferimento. È come se, per programmare in un linguaggio ad alto livello, prima si definisse un modulo contenente le definizioni dei dati e poi si scrivessero le procedure specificando le dichiarazioni a cui fanno riferimento. Questo approccio ai DBMS è stato proposto nel '75 dal comitato ANSI/X3/SPARC (*Standards Planning and Requirements*), dell'*American National Standards Institute*, costituito con lo scopo di proporre una standardizzazione dell'architettura dei DBMS che aggiornasse la precedente del Codasyl-DBTG fatta nel 1969, ma i DBMS esistenti in commercio hanno finito per adottare approcci diversi.

L'approccio con tre livelli di descrizione dei dati è stato proposto come un modo per garantire le proprietà di *indipendenza logica* e *fisica* dei DBMS, che sono un obiettivo importante di questi sistemi.

Per *indipendenza fisica*, da leggere "indipendenza delle applicazioni dall'organizzazione fisica dei dati", si intende il fatto che non è necessario modificare i programmi applicativi quando si modifica l'organizzazione fisica dei dati. Il caso più frequente di modifica dell'organizzazione fisica dei dati si presenta quando occorre intervenire sulle strutture dati ausiliarie che agevolano il reperimento dei dati per migliorare le prestazioni di certe applicazioni, oppure, nel caso di sistemi distribuiti, quando occorre cambiare il nodo della rete dove certi dati sono memorizzati per ridurre i costi di trasferimento.

Esempio 1.3 Si supponga che l'ufficio stipendi esegua con frequenza l'operazione di recupero dei dati riguardanti i docenti che abbiano un certo parametro. Se il numero dei docenti è basso, l'operazione potrebbe essere eseguita con ritardi accettabili visitando serialmente tutti i dati presenti, ma se il numero dei docenti cresce nel tempo, ad un certo punto questo modo di procedere compor-

terebbe tempi di risposta intollerabili. Occorre allora modificare lo schema fisico aggiungendo un indice sull'attributo Parametro:

```
CREATE INDEX IndiceParametro ON Personale(Parametro)
```

Per garantire l'indipendenza fisica non è necessario che il DBMS abbia un'architettura con tre livelli di descrizione dei dati, ma è sufficiente che gli operatori sulla base di dati disponibili agli utenti non dipendano dall'organizzazione fisica dei dati. L'indipendenza fisica corrisponde alla proprietà di indipendenza dei programmi dalla rappresentazione di un tipo di dato. Ad esempio, la rappresentazione di un tipo insieme può essere modificata da una struttura *hash* ad una ad albero senza effetti sui programmi che ne fanno uso, a patto di lasciare inalterata l'interfaccia del tipo di dato.

Per *indipendenza logica*, da leggere "indipendenza delle applicazioni dall'organizzazione logica dei dati", si intende il fatto che non è necessario modificare i programmi applicativi quando si modifica lo schema logico. Le modifiche possono essere l'aggiunta di nuove definizioni, la modifica o l'eliminazione di alcune di quelle esistenti.

Quindi, mentre l'indipendenza fisica rende le applicazioni indipendenti da modifiche dell'organizzazione fisica dei dati, l'indipendenza logica rende le applicazioni indipendenti da modifiche delle esigenze informative. Quanto sia ampio quest'ultimo tipo di indipendenza dipende dai meccanismi che offre il DBMS per definire la corrispondenza fra uno schema esterno, al quale fanno riferimento i programmi applicativi, e lo schema logico. Infatti, o la modifica non interessa un certo schema esterno, oppure, perché non vengano modificati i programmi, occorre poter ridefinire la relazione tra lo schema logico e lo schema esterno in modo da lasciare inalterata la visione della base di dati.

Esempio 1.4 Supponiamo che si decida di cambiare l'organizzazione logica dei dati sul personale memorizzandoli in due tabelle, Docenti e TecniciEAmministrativi. Per rendere le applicazioni che usano la tabella Personale indipendenti da questa modifica, la base di dati si ridefinisce come segue:

```
CREATE TABLE Docenti (
       Nome           CHAR(30),
       CodiceFiscale  CHAR(15),
       Stipendio      INTEGER,
       Parametro      CHAR(6),
       Recapito       CHAR(8) );

CREATE TABLE TecniciEAmministrativi (
```

```
            Nome            CHAR(30),
            CodiceFiscale   CHAR(15),
            Stipendio       INTEGER,
            Parametro       CHAR(6),
            Recapito        CHAR(8) );

CREATE      VIEW        Personale
      AS    SELECT      *
            FROM        Docenti
              UNION
            SELECT      *
            FROM        TecniciEAmministrativi;
```

Nei sistemi commerciali relazionali l'indipendenza fisica è di solito garantita e quella logica è garantita in una certa misura.

1.5.2 Uso della base di dati

Come conseguenza dell'integrazione dei dati, un DBMS deve prevedere più modalità d'uso per soddisfare le esigenze delle diverse categorie di utenti che possono accedere alla base di dati. Il valore della base di dati, infatti, dipende dalla facilità con cui può essere utilizzata e quindi dalle possibilità di accesso offerte dal sistema. Prendiamo in considerazione le esigenze di tre categorie di utenti: i *programmatori delle applicazioni*, gli *utenti non programmatori* e gli *utenti delle applicazioni*.

Programmatori delle applicazioni

Questa categoria comprende coloro che sviluppano le applicazioni che verranno poi utilizzate dagli "utenti delle applicazioni" per accedere o modificare i dati. Per lo sviluppo di queste applicazioni, un DBMS mette normalmente a disposizione alcune delle seguenti modalità:

1. utilizzo di "generatori di applicazioni", ovvero di strumenti i quali generano automaticamente il codice a partire da alcune definizioni date dal programmatore in modo dichiarativo o grafico. Ad esempio, questi strumenti permettono di indicare graficamente l'aspetto di menu e maschere nonché il codice da associare a determinate azioni dell'utente, e generano poi automaticamente il codice che realizza la modalità di interazione così specificata. Questi strumenti sono molto indicati per gestire gli aspetti di interazione delle applicazioni, ma il codice generato può avere bisogno di essere integrato manualmente quando si realizzano applicazioni complesse;
2. programmazione in un linguaggio tradizionale esteso con gli operatori del DML del sistema; normalmente il programmatore può scegliere tra alcuni linguaggi di-

versi, quali, ad esempio, C e Java. Gli operatori predefiniti del modello dei dati possono essere codificati nel linguaggio ospite secondo diverse modalità:

- come procedure predefinite;
- come costrutti da preelaborare, per convertire i nuovi costrutti in chiamate a procedure predefinite, prima di sottoporre il programma alla traduzione con il compilatore tradizionale del linguaggio ospite;
- come nuovi costrutti, e il compilatore del linguaggio ospite viene esteso per trattare anche questi, traducendoli in chiamate a procedure predefinite.

Questo approccio presenta alcuni problemi, dovuti alla necessità di effettuare conversioni improduttive tra il formato con cui i dati vengono rappresentati nel linguaggio ospite ed il formato con cui essi sono gestiti dal DBMS. Ad esempio, il DBMS gestisce collezioni di dati, che non hanno un corrispettivo diretto nei linguaggi di programmazione tradizionali, i quali a loro volta offrono tipi di dati, quali i puntatori o le liste, che non possono essere trasferiti direttamente nella base di dati. Per quanto riguarda invece la *comunicazione* fra il programma applicativo e il DBMS (scambio di dati e messaggi sull'esito delle operazioni), in generale si ricorre ad aree di lavoro comuni (*user working area, UWA*), viste dai programmi come un insieme di variabili predefinite. Il trasferimento dei dati dalla base di dati al programma, e viceversa, è causato dall'attivazione degli operatori sulla base di dati (Figura 1.6);

3. programmazione con un "linguaggio per basi di dati", ovvero con un linguaggio completo che integri le caratteristiche di un linguaggio di programmazione con i meccanismi di definizione ed uso della base di dati, quali cosiddetti *linguaggi della quarta generazione* (*fourth generation languages, 4GL*) definiti per i sistemi relazionali. Questi linguaggi risolvono i problemi di cattiva integrazione tra linguaggio ospite e DML discussi al punto precedente.

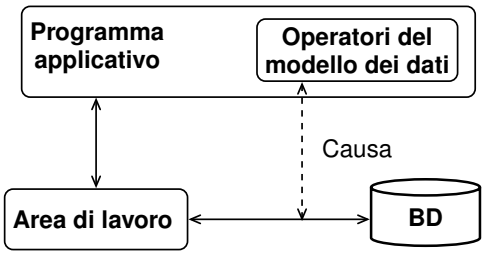

Figura 1.6: Scambio dei dati fra programmi e base di dati

Un'ultima caratteristica che distingue un linguaggio per l'uso dei dati, in particolare per la parte riguardante la ricerca, consiste nel fatto che è *dichiarativo* (*non procedurale*). Con questo termine si intende che gli operatori del linguaggio agiscono su insiemi di dati, mentre l'utente, per individuare i dati che interessano, specifica la condizione che essi devono soddisfare senza dettagliare i passi della ricerca, che vengono invece stabiliti automaticamente dal sistema.

Utenti non programmatori

In questa categoria rientrano coloro che richiedono un linguaggio interattivo a sé stante, di facile uso, per fare principalmente ricerche di dati. Sono quindi utili anche strumenti per (*a*) formulare le richieste, (*b*) definire in modo dichiarativo il formato in cui vanno stampati i risultati delle ricerche (*report generators*), (*c*) visualizzare il contenuto della base di dati e i risultati di ricerche in forma grafica (istogrammi, diagrammi, torte ecc.). I linguaggi più efficaci, in particolare quelli che prevedono l'uso della grafica e interfacce amichevoli, sono stati sviluppati grazie alla diffusione dei sistemi relazionali sui calcolatori personali. Sono inoltre disponibili per questi utenti semplici strumenti interattivi per l'importazione dei dati di una base di dati all'interno di fogli di calcolo o di altri programmi per la generazione di grafici e tabelle.

Utenti delle applicazioni

Sono coloro che richiedono delle modalità molto semplici per attivare un numero predefinito di operazioni, senza avere nessuna competenza informatica. Esempi sono gli impiegati addetti agli sportelli di una banca o alle prenotazioni di una compagnia aerea. In questi casi un'operazione è invocata interattivamente, con uso di un terminale video: l'utente agisce selezionando una delle possibili scelte proposte (menu), e fornisce i valori degli argomenti riempendo campi di opportune "maschere".

1.5.3 Controllo della base di dati

Una caratteristica molto importante dei DBMS è il tipo di meccanismi offerti per garantire le seguenti proprietà di una base di dati: *integrità*, *affidabilità* e *sicurezza*.

Integrità

I DBMS prevedono dei meccanismi per controllare che i dati inseriti, o modificati, siano conformi alle definizioni date nello schema, in modo da garantire che la base di dati si trovi sempre in uno stato *consistente*. Il linguaggio per la definizione dello schema logico consente di definire non solo la struttura dei dati, ma anche le condizioni a cui essi devono sottostare per essere significativi (*vincoli d'integrità*), quando queste condizioni vanno verificate e cosa fare in caso di violazioni. Vedremo degli esempi nel Capitolo 7.

Affidabilità

I DBMS devono disporre di meccanismi per proteggere i dati da malfunzionamenti hardware o software e da interferenze indesiderate dovute ad accessi concorrenti ai dati da parte di più utenti. A tal fine, un DBMS prevede che le interazioni con la base di dati avvengano per mezzo di *transazioni* (Figura 1.7), cioè con un meccanismo che:

- esclude effetti parziali dovuti all'interruzione delle applicazioni per una qualsiasi ragione;
- garantisce l'assenza di interferenze nel caso di accessi concorrenti ai dati.

Più precisamente vale la seguente definizione:

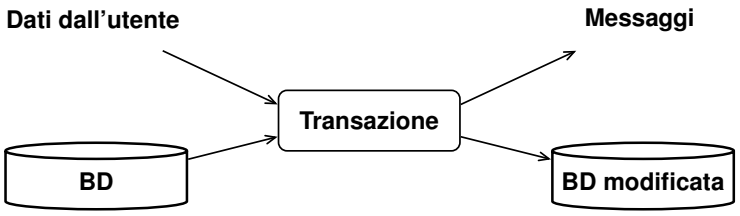

Figura 1.7: Dati e risultati di una transazione

Definizione 1.4

Una transazione è una sequenza di azioni di lettura e scrittura della base di dati e di elaborazioni di dati in memoria temporanea, che il DBMS esegue garantendo le seguenti proprietà:[2]

Atomicità. Solo le transazioni che terminano normalmente (*committed transactions*) fanno transitare la base di dati in un nuovo stato. Le transazioni che terminano prematuramente (*aborted transactions*) sono trattate dal sistema come se non fossero mai iniziate; pertanto eventuali loro effetti parziali sulla base di dati sono annullati.

Serializzabilità. L'effetto sulla base di dati dell'esecuzione concorrente di più transazioni è equivalente ad un'esecuzione seriale delle transazioni, cioè ad una esecuzione in cui le transazioni vengono eseguite una dopo l'altra in un qualche ordine.

Persistenza. Le modifiche sulla base di dati di una transazione terminata normalmente sono permanenti, cioè non sono alterabili da eventuali malfunzionamenti successivi alla terminazione.

Una transazione può essere espressa come un insieme di espressioni in un linguaggio di interrogazione, oppure come un programma sequenziale che opera sulla base di dati; in entrambi i casi esiste qualche meccanismo per segnalare al sistema il punto di inizio ed il punto finale della transazione. L'esecuzione di una transazione comporta il trasferimento di dati fra le memorie temporanea (buffer) e permanente.

Un *malfunzionamento* (*failure*) è un evento a causa del quale la base di dati può trovarsi in uno stato scorretto. Verrà fatta l'ipotesi che la manifestazione di un'anomalia sia sempre rilevata e ciò comporti:

2. Si possono scegliere altre proprietà per caratterizzare le transazioni. Nella letteratura in lingua inglese di solito si preferiscono le seguenti: *Atomicity*, *Consistency*, *Isolation*, e *Durability*, che producono l'acronimo ACID.

1. l'interruzione istantanea di una transazione o dell'intero sistema, a seconda del tipo di malfunzionamento verificatosi;
2. l'attivazione di opportune procedure che permettano di riportare la base di dati allo stato corretto precedente alla manifestazione del malfunzionamento (*procedure di ripristino della base di dati, "recovery"*).

Si distinguono tre tipi di malfunzionamenti in base al tipo di memoria interessata: *fallimenti di transazioni* (*transaction failure*), *fallimento di sistema* (*system failure*) e *disastri* (*media* (*disk*) *failure*).

Definizione 1.5
I *fallimenti di transazioni* sono interruzioni di transazioni che non comportano perdite di dati né in memoria temporanea (*buffer*) né in memoria permanente.

I fallimenti di transazioni sono dovuti a situazioni già previste nei programmi, il cui verificarsi comporta la terminazione prematura della transazione; oppure a situazioni non previste nei programmi, il cui verificarsi causa la terminazione prematura della transazione da parte del sistema. Esempi sono violazione di vincoli di integrità, tentativo di accesso a dati protetti, condizioni di stallo.

Definizione 1.6
I *fallimenti di sistema* sono interruzioni del suo funzionamento dovuti ad un'anomalia hardware o software dell'unità centrale o di una periferica, con conseguente interruzione di tutte le transazioni attive. Si assume che il contenuto della memoria permanente sopravviva, mentre si considera perso il contenuto della memoria temporanea.

Esempi di questo genere di malfunzionamento sono l'interruzione dell'alimentazione elettrica del sistema ed errori del software di base (DBMS e sistema operativo).

Definizione 1.7
I *disastri* sono malfunzionamenti che danneggiano la memoria permanente contenente la base di dati.

Possibili cause possono essere il danneggiamento delle periferiche o un errore umano che rende irrecuperabile il contenuto della base di dati. I disastri influenzano tutte le transazioni che stanno usando la porzione di base di dati danneggiata.

Compito delle procedure di ripristino è di garantire che la base di dati contenga tutte e sole le modifiche apportate dalle transazioni terminate con successo prima dell'occorrenza del malfunzionamento.

Per poter eseguire queste procedure un DBMS mantiene una copia di sicurezza della base di dati e tiene traccia di tutte le modifiche fatte sulla base di dati dal momento in cui è stata eseguita l'ultima copia di sicurezza. Grazie a questi dati ausiliari, quando si verifica un malfunzionamento il DBMS può ricostruire una versione corretta dei dati utilizzando l'ultima copia e rieseguendo tutte le operazioni che hanno modificato i dati e di cui ha mantenuto traccia.

Altra funzionalità di un DBMS è di consentire l'esecuzione *concorrente* di più transazioni, risolvendo il problema di farle funzionare correttamente, senza interferenze indesiderate quando esse operano sugli stessi dati (*controllo della concorrenza*). Il classico esempio di interferenza è quello che porta alla *perdita di modifiche*.

Esempio 1.5 Si supponga che Antonio e Giovanna condividano un conto corrente e che contemporaneamente facciano un prelievo e un versamento da sportelli diversi. Sia 350 il saldo, 400 la somma che Giovanna versa e 50 la somma che Antonio preleva. Supponiamo che sulla base di dati si verifichino i seguenti eventi, nell'ordine mostrato:

1. Il cassiere di Giovanna legge il saldo 350.
2. Il cassiere di Antonio legge il saldo 350.
3. Il cassiere di Giovanna modifica il saldo in 750.
4. Il cassiere di Antonio modifica il saldo in 300.

L'effetto dell'operazione di Giovanna è annullato da quello dell'operazione di Antonio e il saldo finale è di 300.

Per evitare interferenze indesiderate il DBMS coordina opportunamente l'esecuzione concorrente di un insieme di transazioni $\{T_1, \ldots, T_n\}$, intercalando opportunamente nel tempo le azioni sulla base di dati di ogni transazione, in modo che l'effetto dell'esecuzione sia quello ottenibile eseguendo le transazioni isolatamente, in un qualche ordine.

Un modo molto semplice di risolvere il problema sarebbe quello di eseguire le transazioni isolatamente, cioè in modo tale che, per ogni coppia di transazioni T_i e T_j, tutte le azioni di T_i precedono quelle di T_j, o viceversa (*esecuzione seriale*). Questa soluzione impedirebbe però ogni forma di concorrenza riducendo il DBMS ad un elaboratore seriale di transazioni. Vedremo nel Capitolo 9 le tecniche solitamente usate dai DBMS sia per gestire le transazioni che la concorrenza.

Sicurezza

I DBMS prevedono meccanismi sia per controllare che ai dati accedano solo persone autorizzate, sia per restringere i dati accessibili e le operazioni che si possono fare su di essi. Il problema presenta diversi aspetti, alcuni dei quali simili a quelli affrontati nell'ambito dei sistemi operativi, altri tipici di questa classe di applicazioni.

Aspetti del primo tipo sono ad esempio l'identificazione degli utenti autorizzati, con parole di riconoscimento, oppure la possibilità di proteggere i dati da furti mediante crittografia. Per mostrare aspetti specifici dell'area base di dati, immaginiamo di disporre di dati riguardanti cittadini, fra cui il codice fiscale, dati anagrafici e il reddito. Eventuali restrizioni che si potrebbero imporre a categorie diverse di utenti sono:

1. certi utenti non possono accedere a questi dati, ma solo ad altri presenti nella base di dati;

2. certi utenti possono accedere ai dati, ma non possono modificarli;
3. certi utenti possono accedere solo ai dati che li riguardano, senza modificarli;
4. come nel caso precedente, ma si possono modificare solo i dati anagrafici;
5. certi utenti possono accedere solo ai dati anagrafici, da certi uffici e in certe ore del giorno;
6. certi utenti possono applicare solo operazioni statistiche sul reddito, ma non possono accedere a dati singoli né fare modifiche.

Questa lista potrebbe continuare e diventare ancora più significativa se si considerassero più insiemi di dati in relazione logica, ma è sufficiente per dare un'idea della complessità e flessibilità di un meccanismo per garantire la sicurezza dei dati. Una soluzione ad alcuni dei problemi è data dallo schema esterno ed altre verranno viste quando presenteremo i sistemi commerciali.

Questo problema diventa ancora più complesso nel caso di basi di dati per uso statistico: si vuol concedere la possibilità di conoscere, ad esempio, la media dei redditi delle persone, ma non il reddito di una particolare persona. Notare che non è sufficiente imporre che si possano soddisfare solo richieste riguardanti insiemi perché certe informazioni riservate potrebbero essere ricavate lo stesso per deduzione. Ad esempio, per conoscere il reddito di Albano, Orsini potrebbe chiedere il reddito medio dell'insieme {Albano, Orsini} e, noto il proprio reddito, dedurre ciò che cercava di sapere.

Si noti, infine, che la legislazione negli ultimi anni è diventata sempre più attenta al trattamento dei dati dei cittadini (leggi sulla *privacy*), in particolare dei cosiddetti dati *sensibili* (ad esempio i dati sanitari), richiedendo particolari accorgimenti per l'accesso e la protezione.

1.5.4 Distribuzione della base di dati

Un DBMS moderno deve garantire la possibilità di distribuire i dati gestiti dal sistema su più elaboratori collegati tra di loro da una rete locale o geografica, eventualmente replicando alcuni dati. La distribuzione dei dati su reti geografiche è utile ad aziende con più sedi per poter gestire in ognuna di queste i dati di interesse locale, mantenendo la possibilità di accedere a tutti i dati dell'organizzazione. Infine, entrambi i tipi di distribuzione, se combinati con la replicazione di alcuni dati, permettono di continuare ad operare sui dati anche quando un elaboratore sia fuori servizio. Il sistema mette a disposizione in questo caso:

1. strumenti per la progettazione e definizione dello schema che permettono di definire la locazione dei dati e di valutare gli effetti di una diversa distribuzione degli stessi;
2. linguaggi per l'uso dei dati che permettono di scrivere applicazioni prescindendo dalla locazione dei dati stessi;
3. strumenti per il controllo dei dati che permettono di gestire la concorrenza ed i fallimenti di transazioni che vengono eseguite su più elaboratori, garantendo in particolare la coerenza dei dati replicati;

4. strumenti per l'amministrazione di un sistema distribuito, in particolare per verificare gli effetti della distribuzione dei dati sulle prestazioni delle applicazioni.

1.5.5 Amministrazione della base di dati

L'amministratore della base di dati (*Data Base Administrator, DBA*) è una persona (o un gruppo di persone) che, dopo aver partecipato allo studio di fattibilità per decidere l'impiego di un DBMS, ne seleziona uno, lo mette in funzione, lo mantiene in esercizio e segue ogni base di dati dalla progettazione all'impiego. In particolare, quindi, ha bisogno di strumenti per svolgere le seguenti attività:

1. analisi dei requisiti di nuove applicazioni e progettazione, sviluppo e manutenzione di basi di dati e delle applicazioni che ne fanno uso;
2. definizione degli schemi di basi di dati (logici, fisici ed esterni), delle autorizzazioni e modalità di accesso ai dati per ogni classe di utenti e delle politiche per la sicurezza dei dati;
3. manutenzione della struttura logica e fisica dei dati per correggere errori di progettazione o per adeguarle a nuove esigenze;
4. definizione delle procedure per il caricamento dei dati, la creazione di copie di sicurezza, il ripristino dei dati dopo malfunzionamenti di sistema o disastri;
5. controllo del funzionamento del sistema per decidere eventuali riorganizzazioni della struttura logica e fisica dei dati al fine di migliorare le prestazioni delle applicazioni;
6. pianificazione della formazione del personale e definizione di standard per la programmazione e prova delle applicazioni.

Un importante strumento di ausilio per l'amministrazione di basi di dati è il cosiddetto *catalogo* del sistema, che contiene informazioni su ciò che è definito nella base di dati, su quali definizioni operano le applicazioni e su come sono memorizzati i dati. In altre parole, mentre con gli schemi si descrivono i dati presenti nel sistema e le relazioni fra loro, con il catalogo si raccolgono dati riguardanti gli oggetti descritti. Un catalogo contiene pertanto sia informazioni d'interesse degli utenti e dell'amministratore della base di dati, sia informazioni sui dati memorizzati utilizzabili dal DBMS per il suo funzionamento. Un catalogo è organizzato come una base di dati aggiornata automaticamente dal sistema. Altri utili strumenti per assistere l'amministratore della base di dati nello svolgimento di tutte le attività elencate sono forniti dai produttori di DBMS, o da ditte indipendenti, e saranno discussi nel Capitolo 7.

La progettazione di basi di dati viene discussa nel Capitolo 3 con un approccio indipendente dal tipo di DBMS da usare (progettazione concettuale). La realizzazione di basi di dati relazionali viene discussa nel Capitolo 4 con un approccio che trasforma il progetto concettuale dei dati in uno schema relazionale e nel Capitolo 5 con un approccio formale detto di *normalizzazione*, proposto inizialmente per essere usato in alternativa all'approccio per trasformazione, ma che di solito si usa in modo complementare. La definizione e amministrazione di basi di dati relazionali sono discusse nel Capitolo 7 .

1.6 Vantaggi e problemi nell'uso dei DBMS

La tecnologia delle basi di dati, oltre ai vantaggi visti (integrazione dei dati e flessibilità della base di dati), ne offre anche altri, in particolare quello di stabilire degli standard riguardo alla strutturazione e nomenclatura dell'informazione, e di ridurre notevolmente tempi di sviluppo delle applicazioni rispetto a quelli richiesti usando la tecnologia degli archivi. Il problema degli standard è molto sentito in grandi organizzazioni dove la molteplicità delle esigenze richiede una terminologia e un modo comune di definire i dati per facilitare le comunicazioni e la cooperazione fra gli utenti.

L'impiego dei DBMS comporta anche dei problemi che vanno tenuti presenti nel valutare l'opportunità della loro adozione. Questi problemi si avvertono in particolare quando si tratta di realizzare sistemi informatici complessi, ma non vanno sottovalutati nemmeno nei casi più semplici, sebbene la continua riduzione dei costi della tecnologia informatica renda sempre più conveniente l'impiego di questi prodotti.

Problemi gestionali e organizzativi

1. I DBMS più sofisticati richiedono un notevole impegno di risorse hardware e software per la loro messa a punto ed il loro funzionamento.
2. Il costo di questi sistemi, ed i costi di esercizio, possono essere facilmente sottostimati.
3. È importante acquisire e mantenere personale qualificato e con competenze specifiche sul sistema impiegato.
4. L'introduzione del sistema ha un impatto sulla struttura organizzativa.

Problemi di produzione del software

1. Il progetto di base di dati e la messa a punto delle applicazioni richiedono personale qualificato e strumenti opportuni, che non sono messi a disposizione da tutti i sistemi.
2. L'impiego di una base di dati richiede una ristrutturazione dei dati in archivi già esistenti e la riscrittura dei programmi applicativi.
3. L'impiego di un DBMS può aumentare la dipendenza da ditte esterne all'impresa per lo sviluppo delle applicazioni.

Problemi di funzionamento

1. Una conseguenza della centralizzazione è l'aumento del rischio di interruzione dei servizi. Infatti, se il sistema non è disponibile, non lo è per nessuna applicazione.
2. Per non degradare i tempi di risposta, i dati vanno organizzati con la massima attenzione.

Problemi di pianificazione
I costi di acquisizione della tecnologia delle basi di dati e di sviluppo delle applicazioni sono tali da rendere non praticabile la sostituzione di un sistema con un nuovo prodotto non compatibile. Pertanto la scelta del tipo di DBMS va fatta pianificando l'intervento accuratamente.

1.7 Conclusioni

È stata introdotta la nozione di base di dati e sono state discusse le funzionalità che caratterizzano un sistema per la gestione di basi di dati. Questo tipo di sistema è la tecnologia più adatta per sviluppare applicazioni che usano in modo concorrente dati persistenti, in contesti in cui sia necessario accedere alle stesse informazioni da parte di più applicazioni e di più settori di un'organizzazione, e dove è prevista un'evoluzione nel tempo delle esigenze di archiviazione e gestione di informazioni. La complessità di questo tipo di sistema è dovuta alla varietà di funzionalità che deve offrire per consentire di organizzare, e proteggere da malfunzionamenti, dati persistenti da rendere facilmente accessibili a utenti specialistici e non, senza sacrificare l'efficienza delle operazioni.

I concetti introdotti in questa rapida panoramica sono molti e riguardano aspetti fondamentali delle basi di dati che saranno ripresi e trattati diffusamente nei prossimi capitoli.

Esercizi

1. Discutere le differenze fra un sistema per la gestione di basi di dati e un sistema di archiviazione.
2. Elencare alcune domande (fra cinque e dieci) da fare ad un produttore di sistemi per la gestione di dati al fine di stabilire se il sistema che propone può essere classificato come un sistema per la gestione basi di dati centralizzate.
3. Discutere vantaggi e svantaggi di un sistema per la gestione di basi di dati.
4. Spiegare la differenza tra i seguenti termini: base di dati e sistema per la gestione di basi di dati.
5. Discutere i concetti di indipendenza logica e fisica, confrontandoli con i concetti di modulo e tipo di dato astratto dei linguaggi di programmazione.
6. Discutere le differenze tra il modello dei dati di un sistema di archiviazione e di un sistema per basi di dati.
7. Discutere i compiti del programmatore delle applicazioni e dell'amministratore della base di dati.
8. Quali delle seguenti affermazioni è vera?
   (a) Sistema informatico è un sinonimo di sistema informativo.
   (b) Un linguaggio di interrogazione richiede la conoscenza di un linguaggio di programmazione.

(c) Per usare correttamente una base di dati l'utente (persona o programma) deve conoscere l'organizzazione fisica dei dati.
(d) L'organizzazione fisica di una base di dati va programmata dall'amministratore della base di dati.
(e) Schema logico, schema fisico e schema esterno sono sinonimi.
(f) Per soddisfare le esigenze degli utenti delle applicazioni non occorre un linguaggio di programmazione.
(g) Le transazioni nei sistemi per basi di dati hanno le stesse proprietà dei programmi nei linguaggi con archivi.
(h) Per realizzare un sistema informatico il personale tecnico realizza innanzitutto un sistema per basi di dati.
(i) Il programmatore delle applicazioni decide quali sono i dati accessibili agli utenti.

Note bibliografiche

Sono numerosi i libri che trattano i sistemi per la gestione di basi di dati. Fra i più recenti in italiano si segnalano [Atzeni et al., 2002], [Ramakrishnan and Gehrke, 2003], e [Elmasri and Navathe, 2001]. Per approfondimenti, riferimenti interessanti in lingua inglese sono [Abiteboul et al., 1995], [Silberschatz et al., 2002], [Kifer et al., 2005], [Ullman and Widom, 2001].

BIBLIOGRAFIA

Abiteboul, S. and Bidoit, N. (1984). An algebra for non normalized relations. In *Proceedings of the ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS)*. 135

Abiteboul, S., Hull, R., and Vianu, V. (1995). *Database Foundations*. Addison-Wesley, Reading, Massachusetts. 27, 136, 178

Albano, A. (2001). *Costruire sistemi per basi di dati*. Addison-Wesley, Milano. 220, 252, 279

Albano, A., Antognoni, G., and Ghelli, G. (2000). View operations on objects with roles for a statically typed database language. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):548–567. 40, 48, 49

Albano, A., Cardelli, L., and Orsini, R. (1985). Galileo: A strongly typed, interactive conceptual language. *ACM Transactions on Database Systems*, 10(2):230–260. Also in S.B. Zdonik and D. Maier, editors, *Readings in Object-Oriented Database Systems*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990. 40, 48, 49

Armstrong, W. (1974). Dependency structures of database relationships. In *Proceedings of the IFIP Congress*, pages 580–583. 143

Atzeni, P. and Antonellis, V. D. (1993). *Relational Database Theory*. Morgan Kaufmann Publishers, San Mateo, California. 136, 156, 178

Atzeni, P., Ceri, S., Paraboschi, S., and Torlone, R. (2002). *Basi di dati. Modelli e linguaggi di interrogazione*. McGraw-Hill, Milano. 27, 99

Batini, C., Ceri, S., and Navathe, S. (1992). *Conceptual Database Design. An Entity-Relationship Approach*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, California. 72, 99

Batini, C., Lenzerini, M., and Navathe, S. (1987). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):2–18. 90

Beeri, C. and Bernstein, P. (1979). Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems*, 4(1):30–59. 147

Bernstein, P. (1976). Synthesizing third normal form relations from functional

dependencies. *ACM Transactions on Database Systems*, 1(4):277–298. 169

Ceri, S., editor (1983). *Methodology and Tools for Database Design*. North-Holland, Amsterdam. 99

Ceri, S., Gottlob, G., and Tanca, L. (1990). *Logic Programming and Data Bases*. Springer-Verlag, Berlin. 136

Codd, E. (1970). A relational model for large shared data banks. *Communications of the ACM*, 13(6):377–387. 136, 141

Connolly, T. and Begg, C. (2000). *Database Solutions. A step-by-step guide to building databases*. Addison-Wesley, Reading, Massachusetts. 99

Diederich, J. and Milton, J. (1988). New methods and fast algorithms for database normalization. *ACM Transactions on Database Systems*, 13(3):339–365. 152

Elmasri, R. and Navathe, S. (2001). *Sistemi di basi di dati. Fondamenti. Prima edizione italiana*. Addison-Wesley, Milano. 27

Fraternali, P. and Tanca, L. (1995). A structured approach for the definition of the semantics of active databases. *ACM Transactions on Database Systems*, 20(4):414–471. 216

Harel, D. (1987). Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274. 73

Kifer, M., Bernstein, A., and Lewis, P. M. (2005). *Database Systems*. Addison-Wesley, Reading, Massachusetts, second edition. 27, 202, 250

Lucchesi, C. and Osborn, S. (1978). Candidate keys for relations. *Journal of Computer and System Sciences*, 17(2):270–280. 148

Maciaszek, L. A. (2002). *Sviluppo di sistemi informativi con UML*. Addison-Wesley, Milano. 99

Maier, D. (1983). *The Theory of Relational Databases*. Computer Science Press, Rockville, Maryland. 136, 152, 153, 178

Mannila, H. and Räihä, K. (1992). *The Design of Relational Databases*. Addison-Wesley, Reading, Massachusetts. 178

Marco, T. D. (1979). *Structured Analysis and System Specification*. Prentice Hall, Inc., Englewood Cliffs, New Jersey. 72

Ramakrishnan, R. and Gehrke, J. (2003). *Sistemi di basi di dati*. McGraw-Hill, Milano. 27

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1991). *Object-Oriented Modeling and Design*. Prentice Hall International, Inc., London. 72, 73

Schmidt, J. (1977). Some high level language constructs for data of type relation. *ACM Transactions on Database Systems*, 2(3):247–261. 239

Shasha, D. and Bonnet, P. (2002). *Database Tuning: principles, experiments, and troubleshooting techniques*. Morgan Kaufmann Publishers, San Mateo, California. 226

Silberschatz, H. F., Korth, H. F., and Sudarshan, S. (2002). *Database System Concepts*. McGraw-Hill, New York, 4th edition. 27, 136

Teorey, T. (1999). *Database Modeling and Design. The E-R Approach*. Morgan Kaufmann Publishers, San Mateo, California, third edition. 99

Tsou, D. and Fischer, P. (1982). Decomposition of a relation scheme into Boyce-Codd

Normal Form. *ACM SIGACT News*, 14(3):23–29. 165

Ullman, J. (1983). *Principles of Database Systems*. Computer Science Press, Rockville, Maryland, second edition. 158

Ullman, J. D. and Widom, J. (2001). *A First Course in Database System*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, second edition. 27, 136, 152

van der Lans, R. (2001). *Introduzione a SQL. Seconda edizione italiana*. Addison-Wesley, Milano. 202, 250

Widom, J. and Ceri, S., editors (1996). *Active Database Systems: Trigger and Rules for Advanced Database Programming*. Morgan Kaufmann Publishers, San Mateo, California. 226

Yourdon, E. (1989). *Modern Structured Analysis*. Yourdon Press, Englewood Cliffs, New Jersey. 72

Zaniolo, C., Ceri, S., Faloutsos, C., Snodgrass, R., Subrahmanian, V., and Zicari, R., editors (1997). *Introduction to Advanced Database Systems*. Morgan Kaufmann Publishers, San Mateo, California. 226

INDICE ANALITICO