FONDAMENTI DI BASI DI DATI

Antonio Albano, Giorgio Ghelli, Renzo Orsini

Copyright © 2019 A. Albano, G. Ghelli, R. Orsini

Si concede il diritto di riprodurre gratuitamente questo materiale con qualsiasi mezzo o formato, in parte o nella sua interezza, per uso personale o per uso didattico alle seguenti condizioni: le copie non sono fatte per profitto o a scopo commerciale; la prima pagina di ogni copia deve riportare questa nota e la citazione completa, incluso il titolo e gli autori. Altri usi di questo materiale inclusa la ripubblicazione, anche di versioni modificate o derivate, la diffusione su server o su liste di posta, richiede un permesso esplicito preventivo dai detentori del copyright.

12 febbraio 2019

INDICE

1 Sistemi per basi di dati **1**
1.1 Sistemi informativi e informatici 1
1.2 Evoluzione dei sistemi informatici 3
1.3 Tipi di sistemi informatici 6
1.3.1 Sistemi informatici operativi 6
1.3.2 Sistemi informatici direzionali 6
1.4 I sistemi per basi di dati 8
1.5 Funzionalità dei DBMS 13
1.5.1 Definizione della base di dati 13
1.5.2 Uso della base di dati 17
1.5.3 Controllo della base di dati 19
1.5.4 Distribuzione della base di dati 23
1.5.5 Amministrazione della base di dati 24
1.6 Vantaggi e problemi nell'uso dei DBMS 25
1.7 Conclusioni 26
Esercizi 26
Note bibliografiche 27

2 I modelli dei dati **29**
2.1 Progettazione e modellazione 29
2.2 Considerazioni preliminari alla modellazione 30
2.2.1 Aspetto ontologico 30
2.2.2 Aspetto linguistico astratto 38
2.2.3 Aspetto linguistico concreto 38
2.2.4 Aspetto pragmatico 38
2.3 Il modello dei dati ad oggetti 39
2.3.1 Rappresentazione della struttura della conoscenza concreta 40
2.3.2 Rappresentazione degli altri aspetti della conoscenza astratta 51
2.3.3 Rappresentazione della conoscenza procedurale 52

2.3.4 Rappresentazione della comunicazione 53
2.4 Altri modelli dei dati 54
2.4.1 Il modello entità-relazione 55
2.4.2 Il modello relazionale 56
2.5 Conclusioni 58
Esercizi 58
Note bibliografiche 60

3 La progettazione di basi di dati 61
3.1 Introduzione 61
3.2 Le metodologie di progettazione 62
3.2.1 Il ruolo delle metodologie 63
3.2.2 Le metodologie con più fasi 64
3.2.3 Le metodologie con prototipazione 66
3.3 Gli strumenti formali 67
3.3.1 I diagrammi di flusso dati 68
3.3.2 I diagrammi di stato 72
3.4 L'analisi dei requisiti 73
3.4.1 Scopo dell'analisi dei requisiti 74
3.4.2 Come procedere 75
3.4.3 Un esempio di analisi dei requisiti 76
3.5 La progettazione concettuale 82
3.5.1 Scopo della progettazione concettuale 82
3.5.2 Come procedere 83
3.5.3 I passi della progettazione concettuale 84
3.6 Riepilogo della metodologia di progettazione 92
3.7 Conclusioni 93
Esercizi 94
Note bibliografiche 99

4 Il modello relazionale 101
4.1 Il modello dei dati 101
4.1.1 La relazione 101
4.1.2 I vincoli d'integrità 103
4.1.3 Una rappresentazione grafica di schemi relazionali 105
4.1.4 Operatori 105
4.2 Progettazione logica relazionale 107
4.2.1 Rappresentazione delle associazioni binarie uno a molti e uno ad uno 108
4.2.2 Rappresentazione di associazioni molti a molti 110
4.2.3 Rappresentazione delle gerarchie fra classi 112
4.2.4 Definizione delle chiavi primarie 115
4.2.5 Rappresentazione delle proprietà multivalore 117
4.2.6 Appiattimento degli attributi composti 117
4.3 Algebra relazionale 118

4.3.1 Gli operatori primitivi . . . 118
4.3.2 Operatori derivati . . . 123
4.3.3 Proprietà algebriche degli operatori relazionali . . . 126
4.3.4 Altri operatori . . . 130
4.4 Calcolo relazionale su ennuple . . . 132
4.5 I linguaggi logici . . . 133
4.6 Conclusioni . . . 135
Esercizi . . . 135
Note bibliografiche . . . 136

5 Normalizzazione di schemi relazionali 137
5.1 Le anomalie . . . 137
5.2 Dipendenze funzionali . . . 141
5.2.1 Definizione . . . 141
5.2.2 Dipendenze derivate . . . 142
5.2.3 Chiusura di un insieme di dipendenze funzionali . . . 145
5.2.4 Chiavi . . . 147
5.2.5 Copertura di un insieme di dipendenze . . . 151
5.3 Decomposizione di schemi . . . 153
5.3.1 Decomposizioni che preservano i dati . . . 154
5.3.2 Decomposizioni che preservano le dipendenze . . . 156
5.4 Forme normali . . . 161
5.4.1 Forma Normale di Boyce-Codd . . . 161
5.4.2 Normalizzazione di schemi in BCNF . . . 163
5.4.3 Terza forma normale . . . 165
5.4.4 Normalizzazione di schemi in 3NF . . . 166
5.5 Dipendenze multivalore . . . 171
5.6 La denormalizzazione . . . 172
5.7 Uso della teoria della normalizzazione . . . 173
5.8 Conclusioni . . . 174
Esercizi . . . 174
Note bibliografiche . . . 177

6 SQL per l'uso interattivo di basi di dati 179
6.1 SQL e l'algebra relazionale su multinsiemi . . . 180
6.2 Operatori per la ricerca di dati . . . 181
6.2.1 La clausola **SELECT** . . . 183
6.2.2 La clausola **FROM** . . . 184
6.2.3 La clausola **WHERE** . . . 186
6.2.4 Operatore di ordinamento . . . 192
6.2.5 Funzioni di aggregazione . . . 192
6.2.6 Operatore di raggruppamento . . . 193
6.2.7 Operatori insiemistici . . . 195
6.2.8 Sintassi completa del **SELECT** . . . 195
6.3 Operatori per la modifica dei dati . . . 197

6.4 Il potere espressivo di SQL . . . 198
6.5 QBE: un esempio di linguaggio basato sulla grafica . . . 199
6.6 Conclusioni . . . 200
Esercizi . . . 201
Note bibliografiche . . . 202

7 SQL per definire e amministrare basi di dati 203
7.1 Definizione della struttura di una base di dati . . . 203
7.1.1 Base di dati . . . 204
7.1.2 Tabelle . . . 205
7.1.3 Tabelle virtuali . . . 206
7.2 Vincoli d'integrità . . . 209
7.3 Aspetti procedurali . . . 212
7.3.1 Procedure memorizzate . . . 212
7.3.2 Trigger . . . 213
7.4 Progettazione fisica . . . 219
7.4.1 Definizione di indici . . . 219
7.5 Evoluzione dello schema . . . 221
7.6 Utenti e Autorizzazioni . . . 221
7.7 Schemi esterni . . . 223
7.8 Cataloghi . . . 223
7.9 Strumenti per l'amministrazione di basi di dati . . . 224
7.10 Conclusioni . . . 224
Esercizi . . . 225
Note bibliografiche . . . 226

8 SQL per programmare le applicazioni 227
8.1 Linguaggi che ospitano l'SQL . . . 228
8.1.1 Connessione alla base di dati . . . 229
8.1.2 Comandi SQL . . . 230
8.1.3 I cursori . . . 230
8.1.4 Transazioni . . . 231
8.2 Linguaggi con interfaccia API . . . 234
8.2.1 L'API ODBC . . . 235
8.2.2 L'API JDBC . . . 237
8.3 Linguaggi integrati . . . 239
8.4 La programmazione di transazioni . . . 242
8.4.1 Ripetizione esplicita delle transazioni . . . 245
8.4.2 Transazioni con livelli diversi di isolamento . . . 246
8.5 Conclusioni . . . 248
Esercizi . . . 249
Note bibliografiche . . . 250

9 Realizzazione dei DBMS **251**
9.1 Architettura dei sistemi relazionali 251
9.2 Gestore della memoria permanente 252
9.3 Gestore del buffer 252
9.4 Gestore delle strutture di memorizzazione 253
9.5 Gestore dei metodi di accesso 257
9.6 Gestore delle interrogazioni 258
9.6.1 Riscrittura algebrica 258
9.6.2 Ottimizzazione fisica 261
9.6.3 Esecuzione di un piano di accesso 273
9.7 Gestore della concorrenza 273
9.8 Gestore dell'affidabilità 274
9.9 Conclusioni 276
Esercizi 277
Note bibliografiche 279

Bibliografia **281**

Indice analitico **285**

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 2

I MODELLI DEI DATI

Sfruttare la tecnologia informatica nella gestione delle informazioni significa essenzialmente costruire un modello di una realtà di interesse, che chiameremo *universo del discorso*, con l'obiettivo di lavorare poi sul modello per riprodurre o predire l'evoluzione della situazione oggetto di studio. I modelli ricorrono ampiamente nella tecnologia e in ogni campo che richiede un'attività di progettazione. Essi permettono di riprodurre le caratteristiche essenziali di fenomeni reali, omettendo quei dettagli che, ai fini dello studio che ci si prefigge, costituirebbero un'inutile complicazione.

In questo capitolo si chiariranno innanzitutto i diversi aspetti del processo di modellazione, poi si esamineranno i fatti che si modellano, per passare infine a vedere come questi si rappresentano utilizzando un modello dei dati a oggetti ed un semplice formalismo grafico, adatto per costruire modelli di analisi.

2.1 Progettazione e modellazione

Nel progettare una base di dati è conveniente immaginare che l'organizzazione committente voglia raccogliere informazioni riguardanti lo stato di una porzione della realtà, l'universo del discorso. In questa situazione la base di dati deve essere strutturata in modo tale che essa rappresenti un *modello astratto* o *simbolico* di questo universo del discorso, dove il termine modello astratto si definisce come segue.

Definizione 2.1
Un modello astratto è la rappresentazione formale di idee e conoscenze relative ad un fenomeno.

Questa definizione evidenzia tre aspetti fondamentali di un modello astratto:

1. un modello è la *rappresentazione* di alcuni fatti di un fenomeno;
2. la rappresentazione è data con un *linguaggio formale*;
3. il modello è il risultato di un processo di *interpretazione* di un fenomeno, guidato dalle *idee* e *conoscenze* già possedute dal soggetto che interpreta.

La progettazione della base di dati avviene in più fasi, come verrà descritto nel prossimo capitolo, ciascuna delle quali produce un progetto sempre più dettagliato, fino ad arrivare alla realizzazione del sistema. È utile descrivere queste fasi dicendo che ciascuna di esse costruisce un modello dell'universo del discorso raffinando il modello emerso dalla fase precedente. Nelle prime fasi si utilizzano formalismi più astratti

e più vicini a come la realtà è percepita dagli esseri umani, mentre nelle fasi finali si utilizzano formalismi in cui è possibile specificare un maggior numero di dettagli, e che sono direttamente interpretabili da un elaboratore.

2.2 Considerazioni preliminari alla modellazione

Prima di affrontare la costruzione del modello di un universo del discorso è necessario porsi le seguenti domande: "cosa si modella?", "come si modella? ovvero con quali strumenti concettuali si modella?", "con quale linguaggio formale si definisce il modello?", "come si procede nella costruzione del modello?". Le risposte a queste domande caratterizzano i seguenti aspetti della modellazione: *ontologico*, *linguistico astratto*, *linguistico concreto* e *pragmatico*.

2.2.1 Aspetto ontologico

L'aspetto ontologico (studio di ciò che esiste) riguarda ciò che si suppone esistere nell'universo del discorso e quindi sia da modellare. Molte ipotesi sono possibili a questo riguardo, e quella di solito preferita quando si utilizza la tecnologia delle basi di dati è che vadano modellati i seguenti fatti: la *conoscenza concreta*, la *conoscenza astratta*, la *conoscenza procedurale* e la *comunicazione*.

La conoscenza concreta

La *conoscenza concreta* riguarda i fatti specifici che si vogliono rappresentare. Adottando un approccio semplificato, si suppone che la realtà consista di *entità*, che hanno un *tipo* ed alcune *proprietà*. Si suppone inoltre che queste entità siano classificabili all'interno di *collezioni* di entità omogenee, e che esistano *istanze di associazioni* fra queste entità.

Questa visione della realtà, pur così schematica, non è lontana da alcune proposte sviluppate nell'ambito della riflessione filosofica; un esempio, tra i tanti possibili, è il seguente:

> Tutte le cose nominabili, esterne alla mente, si ritengono appartenenti o alla classe delle sostanze o a quella degli attributi.
>
> Un attributo, dicono i logici Scolastici, dev'essere attributo di qualche cosa; un colore, per esempio, dev'essere colore di qualche cosa. Se questo qualche cosa cessasse di esistere, o cessasse di essere connesso con quell'attributo, l'esistenza dell'attributo sarebbe finita.
>
> Una sostanza, invece, esiste di per sé; parlandone non occorre che poniamo "di" dopo il suo nome. Una pietra non è la pietra di qualcosa.
>
> Abbiamo detto che le qualità d'un corpo sono gli attributi, fondati sulle sensazioni che la presenza di quel particolare corpo ai nostri organi eccita nelle nostre menti. Ma quando attribuiamo ad un qualche oggetto lo speciale attributo chiamato relazione, il fondamento dell'attributo dev'essere qualche cosa in cui sono interessati altri oggetti, oltre all'oggetto stesso e al soggetto percipiente.

— John Stuart Mill, *System of Logic, Ratiocinative and Inductive*, Parker, Son, and Bourn, West Strand, London, Fifth Edition, 1862 (trad. it. di G. Facchi, *Sistema di Logica, raziocinativa e induttiva*, Ubaldini Editore, Roma, 1984).

Si dà ora una descrizione dei termini entità, proprietà e associazione, avvertendo che questa descrizione sarà in qualche misura discutibile e imprecisa, data la natura essenzialmente filosofica del problema ontologico.

Definizione 2.2

Un'*entità* è qualcosa di concreto o astratto di cui interessa rappresentare alcuni fatti.

Esempi tipici di entità sono oggetti concreti (un libro, un utente della biblioteca), oggetti astratti (la descrizione bibliografica di un libro) ed eventi (un prestito). Ciò che interessa di un'entità sono le sue *proprietà*.

Definizione 2.3

Il valore di una *proprietà* è un fatto che descrive una qualità di un'entità.

Esempi di proprietà sono il nome ed il recapito di un utente oppure l'autore, il titolo, l'editore e l'anno di pubblicazione di un libro.

La differenza che esiste tra una proprietà ed un'entità scaturisce dalla diversa interpretazione del loro ruolo nel modello: i valori delle proprietà non sono fatti che interessano di per sé, ma solo come caratterizzazione di altri concetti interpretati come entità.

Un'entità non coincide con l'insieme dei valori assunti dalle sue proprietà, poiché:

- i valori delle proprietà di un'entità cambiano in generale nel tempo, anche se l'entità resta la stessa (si pensi, ad esempio, all'età di una persona);
- due entità possono avere le stesse proprietà con gli stessi valori ma essere ugualmente due entità distinte (si pensi, ad esempio, a due persone diverse con lo stesso nome, cognome ed età).

Il concetto di proprietà è strettamente collegato al concetto di *tipo*.

Definizione 2.4

Un *tipo* è una descrizione astratta di ciò che accomuna un insieme di entità omogenee (della stessa natura), esistenti o possibili.

Ad esempio, Persona è il tipo di Giovanna, Mario ecc.

Un tipo non è una collezione di entità esistenti, ma descrive la natura di tutte le entità "possibili" o "concepibili" con tale natura. Ad esempio, il tipo Persona descrive non solo tutte le persone esistenti, ma anche quelle che esisteranno o che potrebbero esistere; quindi un tipo può essere visto come una collezione *infinita* di entità possibili. Ad un tipo sono associate le proprietà che interessano delle entità che appartengono a tale tipo, nonché le *caratteristiche* di tali proprietà. Ad esempio, il tipo Utente ha le proprietà Nome e Recapito intendendo con questo che ogni utente ha un nome e

un recapito, ma con un valore in generale diverso da quello di tutti gli altri. Per ciò che riguarda le *caratteristiche* delle proprietà, ogni proprietà ha associato un *dominio*, ovvero l'insieme dei possibili valori che tale proprietà può assumere, e può essere inoltre classificata come segue:

1. proprietà *atomica*, se il suo valore non è scomponibile (ad esempio, il nome di una persona); altrimenti è detta *strutturata* (ad esempio, la proprietà residenza è scomponibile in indirizzo, CAP, città);
2. proprietà *univoca*, se il suo valore è unico (ad esempio, il nome di un utente ha un unico valore); altrimenti è detta *multivalore* (ad esempio, la proprietà recapiti telefonici di una persona è multivalore se ammettiamo che le persone possano essere raggiungibili attraverso diversi numeri telefonici);
3. proprietà *totale* (*obbligatoria*), se ogni entità dell'universo del discorso ha per essa un valore specificato, altrimenti è detta *parziale* (*opzionale*) (ad esempio, si può considerare il nome di un utente una proprietà totale ed il suo recapito telefonico una proprietà parziale).

Si osservi che non solo tutte queste caratteristiche sono indipendenti tra di loro, ma anche i campi di una proprietà strutturata possono essere nuovamente classificati nello stesso modo; ad esempio, nella residenza, il campo indirizzo può essere a sua volta strutturato in via, o piazza, e numero civico.

Altro concetto interessante è quello di *collezione*.

Definizione 2.5

Una *collezione* è un insieme variabile nel tempo di entità omogenee interessanti dell'universo del discorso.

Ad esempio, i libri di una biblioteca sono la collezione dei libri da essa posseduta. L'insieme degli elementi di una collezione in un determinato momento è detto *estensione* della collezione.

Quindi, mentre un tipo descrive tutte le infinite possibili entità con certe qualità, una collezione è un insieme finito e variabile di entità, dello stesso tipo, che fanno effettivamente parte dell'universo del discorso. Si osservi che, nel realizzare un modello, non si rappresentano tutte le possibili collezioni di entità, così come non si rappresentano tutte le entità che esistono nell'universo del discorso; nel modello si rappresentano solo quelle entità e quelle collezioni che sono considerate interessanti rispetto agli obiettivi per cui viene costruito il modello.

Un altro importante tipo di informazione da modellare è l'eventuale esistenza di *gerarchie di specializzazione* (o *di generalizzazione*, a seconda del verso di percorrenza della gerarchia), tra diverse collezioni di entità: le collezioni in gerarchia modellano insiemi di entità ad un diverso livello di dettaglio. Se una collezione C_{sub} della gerarchia è una specializzazione della collezione C_{sup}, C_{sub} è un sottoinsieme di C_{sup} e gli elementi di C_{sub} ereditano le caratteristiche degli elementi C_{sup}, oltre ad averne altre proprie.

L'uso delle gerarchie di collezioni è molto comune, ad esempio, per classificare gli organismi animali e vegetali: quando diciamo che i marsupiali sono mammiferi

intendiamo che le femmine sono dotate di ghiandole mammarie per l'allattamento dei piccoli, proprietà di ogni mammifero, ma hanno come proprietà specifica il marsupio.

Nell'esempio della biblioteca, la collezione degli Utenti può essere pensata come una generalizzazione delle collezioni Studenti e Docenti, per rappresentare il fatto che entità classificate come elementi di Studenti e Docenti possono essere classificate anche come elementi di Utenti prescindendo dalle proprietà che le rendono semanticamente diverse, ed evidenziando invece che sono entità omogenee ad un diverso livello di astrazione, ad esempio con cognome, residenza e data di nascita come proprietà comuni. Si dice anche che Studenti e Docenti sono *specializzazioni* di Utenti.

Nel processo di modellazione è critico sia stabilire che cosa descrivere come una proprietà o come un'entità, sia stabilire una corretta gerarchia di collezioni per il problema in esame.

Due o più entità possono essere collegate da un'*istanza di associazione*.

Definizione 2.6

Un'*istanza di associazione* è un fatto che correla due o più entità, stabilendo un legame logico fra di loro.

Esempi di istanze di associazioni sono quelle descritte dai seguenti fatti: "lo studente Rossi ha in prestito il libro la Divina Commedia", "lo studente Bianchi frequenta il corso di Basi di dati". Nel primo caso le entità Rossi e la Divina Commedia sono associate dal fatto espresso dal verbo "ha in prestito", nel secondo caso le entità Bianchi e Basi di dati sono associate dal fatto espresso dal verbo "frequenta".

Un'istanza di associazione può correlare anche più di due entità (associazione n-aria), come nel fatto: "lo studente Bianchi frequenta il corso di Basi di dati in aula A1", che correla le tre entità Bianchi, Basi di dati, e aula A1.

Un'istanza di associazione può avere delle proprietà, come nel fatto: "lo studente Bianchi ha acquistato il libro la Divina Commedia con uno sconto del 5%", dove l'ammontare dello sconto è una proprietà che non caratterizza né l'acquirente, che può avere sconti diversi su diversi libri, né il testo, che può essere acquistato con sconti diversi, ma caratterizza proprio la specifica coppia acquirente-libro. Si osservi che lo stesso fatto potrebbe essere anche considerato non un'istanza di associazione con proprietà, ma un evento, ovvero un'entità di tipo acquisto, associato a Bianchi e alla Divina Commedia, con una proprietà sconto.

Così come si è interessati all'esistenza di collezioni omogenee di entità, allo stesso modo si è in genere interessati all'esistenza di insiemi di istanze di associazione.

Definizione 2.7

Un'*associazione* tra le collezioni $C_1, \ldots, C_n$ è un insieme di istanze di associazione tra elementi di $C_1, \ldots, C_n$, che varia in generale nel tempo. Il prodotto cartesiano delle estensioni di $C_1, \ldots, C_n$ è detto il *dominio dell'associazione*.[1]

1. Si osservi che, mentre il dominio di una proprietà è definito da un tipo, ed è quindi un insieme immutabile e in generale infinito, il dominio di un'associazione, essendo il prodotto delle estensioni

Per chiarire le idee, se vediamo due collezioni X ed Y come due insiemi, un'istanza di associazione tra X ed Y può essere vista come una coppia di elementi (x, y), con $x \in X$ ed $y \in Y$, e quindi un'associazione R tra X ed Y può essere vista come un sottoinsieme del prodotto $X \times Y$, ovvero come una relazione (nel senso matematico del termine) tra tali insiemi.

Un'associazione è caratterizzata, oltre che dal suo dominio e dalle caratteristiche delle eventuali proprietà, anche dalle seguenti *proprietà strutturali*: la *molteplicità* e la *totalità*, che definiamo, per semplicità, solo per le associazioni binarie.

Definizione 2.8

La *molteplicità* di un'associazione fra X ed Y riguarda il numero massimo di elementi di Y che possono trovarsi in relazione con un elemento di X e viceversa. Si dice che l'associazione è *univoca* da X ad Y se ogni elemento di X può essere in relazione con *al più* un elemento di Y. Se non esiste tale vincolo si dice che l'associazione è *multivalore* da X ad Y. Allo stesso modo si definisce il vincolo di univocità da Y ad X.

Si osservi che il vincolo di univocità da X ad Y è indipendente dal vincolo di univocità da Y ad X, dando luogo a quattro possibili combinazioni di presenza ed assenza dei due vincoli. Queste combinazioni si esprimono in modo compatto come segue.

Definizione 2.9

La *cardinalità* di un'associazione fra X ed Y descrive contemporaneamente la molteplicità dell'associazione e della sua inversa. Si dice che la cardinalità è *uno a molti* $(1 : N)$ se l'associazione è multivalore da X ad Y ed univoca da Y ad X. La cardinalità è *molti ad uno* $(N : 1)$ se l'associazione è univoca da X ad Y e multivalore da Y ad X. La cardinalità è *molti a molti* $(N : M)$ se l'associazione è multivalore in entrambe le direzioni, ed è *uno ad uno* $(1 : 1)$ se l'associazione è univoca in entrambe le direzioni.

Ad esempio, in un universo del discorso popolato da studenti, dipartimenti, corsi del piano di studi e professori, l'associazione Frequenta(Studenti, Corsi) (dove indichiamo con la notazione $A(C_1, \ldots, C_n)$ un'associazione A con dominio $C_1 \times \ldots \times C_n$) ha cardinalità $(M : N)$, l'associazione Insegna(Professori, Corsi) ha cardinalità $(1 : N)$, l'associazione Dirige(Professori, Dipartimenti) ha cardinalità $(1 : 1)$.

Definizione 2.10

La *totalità* di un'associazione fra due collezioni X ed Y riguarda il numero *minimo* di elementi di Y che sono associati ad ogni elemento di X. Se *almeno* un'entità di Y deve essere associata ad ogni entità di X, si dice che l'associazione è *totale* su

di alcune collezioni, è un insieme finito che dipende dal tempo, ovvero dallo stato dell'universo del discorso.

X, e viceversa sostituendo X con Y ed Y con X. Quando non sussiste il vincolo di totalità, si dice che l'associazione è *parziale*.

Ad esempio, l'associazione Dirige(Professori, Dipartimenti) è totale su Dipartimenti, in quanto ogni dipartimento ha un direttore, ma non su Professori.

Le definizioni precedenti possono essere generalizzate per definire molteplicità e totalità di un'associazione tra $X_1, \ldots, X_n$ rispetto ad X_1, sostituendo "X" con "X_1", ed "elemento di Y" con "ennupla di elementi di $X_2, \ldots, X_n$".

Definizione 2.11

La *struttura della conoscenza concreta* riguarda la conoscenza dei seguenti fatti:

1. esistenza di alcune collezioni, nome delle collezioni, tipo degli elementi delle collezioni (e quindi, nome e caratteristiche delle proprietà di tali elementi);
2. esistenza di alcune associazioni, nome delle associazioni, collezioni correlate da ogni associazione, proprietà strutturali delle associazioni.

Definizione 2.12

La *conoscenza concreta* presuppone che sia nota la struttura di tale conoscenza, e riguarda la conoscenza dei seguenti fatti:

1. quali entità esistono, che tipo ha ciascuna di queste entità, e quali sono i valori delle sue proprietà;
2. per ogni collezione esistente, quali entità appartengono a tale collezione (estensione della collezione);
3. per ogni associazione, quali istanze appartengono a tale associazione (estensione dell'associazione).

In questa classificazione, la conoscenza concreta comprende tutto ciò che riguarda le singole entità (esistenza, valori delle proprietà, appartenenza a collezioni, partecipazione ad associazioni), mentre ciò che descrive la struttura della conoscenza concreta fa parte di una categoria diversa, la conoscenza astratta, descritta nella prossima sezione.

In generale, la conoscenza concreta, che è anche detta *stato* dell'universo del discorso, si evolve nel tempo, in quanto le collezioni, le entità, i valori delle loro proprietà e le associazioni cambiano nel tempo per effetto di processi continui (dipendenti dal trascorrere del tempo), o di processi discreti (dipendenti dal verificarsi di eventi in certi istanti), come il cambio della residenza di un utente, l'acquisizione di un nuovo libro, la concessione di un nuovo prestito ecc. Anche la struttura della conoscenza si evolve nel tempo, perché nell'universo del discorso si possono creare nuovi tipi o collezioni di entità, ma soprattutto perché l'universo del discorso è definito come ciò che interessa modellare, e l'interesse dell'osservatore cambia, in generale, nel tempo.

La conoscenza astratta

La *conoscenza astratta* riguarda i fatti generali che descrivono (*a*) la struttura della conoscenza concreta, (*b*) le restrizioni sui valori possibili della conoscenza concreta

e sui modi in cui essi possono evolvere nel tempo (*vincoli d'integrità*), (*c*) regole per derivare nuovi fatti da altri noti.

È utile classificare i vincoli d'integrità in *vincoli d'integrità statici* e *dinamici*. I *vincoli d'integrità statici* definiscono condizioni sui valori della conoscenza concreta che devono essere soddisfatte indipendentemente da come evolve l'universo del discorso. Le condizioni possono riguardare:

1. I valori di una proprietà. Ad esempio, (*a*) un utente ha le proprietà codice fiscale, nome, residenza, con valori di tipo stringa di caratteri alfanumerici e anno di nascita, con valori di tipo intero; (*b*) uno studente universitario deve avere almeno diciassette anni; (*c*) lo stipendio di un impiegato è un numero positivo.
2. I valori di proprietà diverse di una stessa entità. Ad esempio, (*a*) per ogni impiegato le trattenute sulla paga devono essere inferiori ad un quinto dello stipendio; (*b*) se X è sposato con Y allora il suo stato civile è coniugato.
3. I valori di proprietà di entità diverse di uno stesso insieme. Ad esempio, (*a*) le matricole degli studenti sono tutte diverse; (*b*) se due persone hanno la stessa data di nascita, allora hanno anche la stessa età; (*c*) se una persona X è sposata con Y, allora Y è sposata con X. Un insieme di proprietà è detto *chiave*, rispetto ad una collezione di elementi, se (*a*) i suoi valori identificano univocamente un elemento della collezione, (*b*) ogni proprietà della chiave è necessaria a questo fine. Un esempio di chiave per la collezione degli utenti è il codice fiscale.
4. I valori di proprietà di entità di insiemi diversi. Ad esempio: il presidente della commissione degli esami deve essere il titolare del corrispondente corso.
5. Caratteristiche di insiemi di entità. Ad esempio, il numero degli studenti è inferiore ad un limite massimo prestabilito oppure un laureato in Informatica deve aver accumulato almeno 180 CFU.

I *vincoli d'integrità dinamici* definiscono delle condizioni sul modo in cui la conoscenza concreta può evolvere nel tempo. Ad esempio, una persona coniugata non potrà cambiare stato civile diventando scapolo o nubile, uno studente di un anno di corso non può iscriversi ad un anno precedente dello stesso corso di studio, una data di nascita non può essere modificata. In conclusione, mentre un vincolo statico riguarda ogni singolo stato dell'universo del discorso, un vincolo dinamico riguarda le transizioni da uno stato ad un altro.

Infine, esempi di fatti derivabili da altri sono l'età di una persona, ricavabile per differenza fra l'anno attuale e il suo anno di nascita, oppure la media dei voti degli esami superati da uno studente.

La conoscenza procedurale

L'universo del discorso è una realtà in evoluzione che interagisce con un ambiente. Mentre la conoscenza astratta riguarda la *struttura* dell'universo del discorso, la *conoscenza procedurale* riguarda le operazioni a cui può essere soggetta la conoscenza concreta, ed in particolare l'effetto di tali operazioni ed il modo in cui esse si svolgono.

È utile distinguere due tipi di conoscenza procedurale:

– le operazioni con le quali avvengono le interazioni con l'ambiente esterno per fornire i servizi previsti (*conoscenza delle operazioni degli utenti*);
– le operazioni elementari che interessano le entità dell'universo del discorso per produrre determinati effetti, in particolare le operazioni per la loro creazione, modifica e cancellazione che devono soddisfare le condizioni espresse dai vincoli d'integrità (*conoscenza delle operazioni di base*).

Ad esempio, le operazioni di base che possono interessare uno studente universitario sono: superamento di un esame, passaggio ad altro corso di laurea, conseguimento della laurea, trasferimento ad altra università ecc.

Le operazioni di base riguardanti un'entità ne completano il significato e ne caratterizzano il "comportamento". Secondo un punto di vista, un'entità potrebbe essere caratterizzata completamente in termini delle operazioni che si possono compiere su di essa, senza ricorrere alla nozione separata di proprietà, potendo vedere anche quest'ultime come esempi di operazioni.

Mentre le operazioni di base riguardano una singola entità, un'operazione degli utenti è finalizzata a fornire un servizio agli utenti del sistema informativo secondo modalità prestabilite codificate nelle procedure dell'organizzazione.

Un esempio di procedura nell'ambiente universitario è quella che viene eseguita quando un docente si trasferisce ad un'altra sede (operazione trasferimento docente): (*a*) si interrompe il pagamento dello stipendio; (*b*) si esclude dagli indirizzari per le comunicazioni; (*c*) per ogni corso tenuto dal docente, si inizia la procedura per assegnare il corso ad un altro docente; (*d*) per ogni commissione a cui partecipava il docente, si inizia la procedura per nominare un sostituto ecc.

Altri esempi di procedure sono quelle previste per la gestione del servizio conti correnti di una banca che prevede operazioni del tipo: apertura di un conto, versamento, prelievo, interrogazione saldo, interrogazione movimenti.

La comunicazione

La *comunicazione* riguarda le modalità offerte agli utenti del sistema informativo per scambiare informazioni con il sistema e per accedere alle risorse informative rispettando le regole e le possibilità loro concesse. In particolare, modellare la comunicazione significa anche rappresentare le *interfacce* del sistema informativo, quali ad esempio l'aspetto dei moduli cartacei e delle schermate che vanno riempite per comunicare con tale sistema.

Si osservi che la conoscenza concreta, astratta e delle operazioni di base può essere modellata facendo riferimento essenzialmente all'universo del discorso, sia pure visto in funzione delle necessità del sistema informativo di un'organizzazione. Passando invece alla conoscenza delle operazioni degli utenti, e ancor più alla comunicazione, l'interesse si sposta gradatamente dall'universo del discorso al sistema informativo stesso. Poiché la modellazione è in generale finalizzata alla progettazione di un sistema informativo che modifichi quello preesistente, gli strumenti per modellare procedure e comunicazione possono essere usati per rappresentare tanto il sistema informativo esistente quanto quello in via di progettazione.

L'interesse verso questo aspetto della progettazione e della modellazione, in particolare verso la modellazione dei dialoghi fra gli utenti e il sistema informatico, è molto cresciuto negli ultimi anni poiché, con lo sviluppo degli strumenti hardware e software che supportano la realizzazione di applicazioni con interfaccia grafica, la realizzazione di modalità di interazione gradevoli e personalizzate ai diversi tipi di utenti coinvolti ha assunto un'importanza sempre maggiore, sia come requisito, sia per ciò che riguarda la quantità di lavoro che i programmatori dedicano alla realizzazione dell'interfaccia stessa.

2.2.2 Aspetto linguistico astratto

L'aspetto linguistico astratto riguarda gli strumenti concettuali, o *meccanismi di astrazione*, adottati per modellare l'universo del discorso.

Non sorprende che negli studi sull'argomento sia stata posta la massima attenzione su questi meccanismi, perché l'astrazione è lo strumento concettuale principale per acquisire e organizzare conoscenza.

> Sì come a voler che i calcoli tornino sopra i zuccheri, le sete e le lane, bisogna che il computista faccia le sue tare di casse, invoglie e altre bagaglie, così, quando il filosofo geometra vuol riconoscere in concreto gli effetti dimostrati in astratto, bisogna che difalchi gli impedimenti della materia.
> — Galileo Galilei, *Dialogo sopra i due massimi sistemi del mondo.*

Un meccanismo di astrazione è lo strumento fondamentale per cogliere un aspetto della situazione da descrivere, tralasciando dettagli ritenuti in quel momento poco significativi (l'astrazione come forza semplificatrice). Vedremo nelle prossime sezioni i meccanismi proposti per modellare alcuni dei fatti precedentemente elencati.

2.2.3 Aspetto linguistico concreto

L'aspetto linguistico concreto riguarda le caratteristiche, e la definizione, del linguaggio formale usato per costruire il modello. Una volta fissati i meccanismi di astrazione da usare per modellare, si possono usare linguaggi formali con caratteristiche molto diverse che supportano i meccanismi di astrazione prescelti, a seconda degli obiettivi che ci si prefigge con la costruzione del modello. Si possono usare linguaggi di specifica non eseguibili, linguaggi logici o linguaggi di programmazione. In questo libro l'attenzione sarà sui formalismi grafici.

2.2.4 Aspetto pragmatico

L'aspetto pragmatico riguarda la metodologia da seguire nel processo di modellazione. Una metodologia è un insieme di regole finalizzate alla costruzione del modello informatico. La natura delle metodologie dipende dal tipo di modello da costruire e quindi dal livello di dettaglio al quale vanno trattati i vari aspetti del modello.

Nel prossimo capitolo ci limiteremo a mostrare un esempio di metodologia per progettare uno *schema concettuale* del modello, cioè una rappresentazione ad alto livello della struttura della conoscenza concreta da trattare. Un'analoga metodologia da utilizzare per la progettazione degli altri aspetti del modello informatico è molto più complessa e la sua presentazione esula dai fini di questo libro. Si rinvia alle note bibliografiche per riferimenti a testi specifici.

Nel resto del capitolo si approfondisce l'aspetto linguistico astratto della modellazione presentando dei meccanismi di astrazione per modellare alcuni dei fatti precedentemente elencati e si mostra un formalismo grafico per rappresentare tali fatti. Sebbene il formalismo grafico non consenta di rappresentare ogni aspetto di un modello informatico, vedremo come esso sia molto utile nelle fasi di specifica dei requisiti e di progettazione dello schema di una base di dati, quando occorre ragionare sui fatti più importanti da trattare.

2.3 Il modello dei dati ad oggetti

Nella costruzione di un modello informatico si procede in due stadi:

1. si "definisce" il modello, ovvero si rappresenta la struttura della conoscenza concreta, le altre parti della conoscenza astratta, la conoscenza procedurale, i tipi di comunicazione (definizione di schema e applicazioni);
2. si "costruisce" la rappresentazione di fatti specifici conformi alle definizioni date, ovvero la rappresentazione della conoscenza concreta (immissione dati).

Esempio 2.1 Per costruire un modello informatico per la gestione di informazioni sui libri, prima si devono definire, tra le molteplici proprietà che caratterizzano un libro, quelle che interessano ai fini dell'applicazione: possono essere titolo, autore, editore ecc. come si usa in una biblioteca, oppure dettagliate informazioni qualitative su materiale e stato di conservazione come può interessare ad un laboratorio di restauro.

Una volta definite le proprietà interessanti comuni a tutti i possibili libri, si passa a costruire per ogni entità "libro" dell'universo del discorso una rappresentazione nel modello informatico, assegnando un valore per ogni proprietà definita.

Per la definizione del modello si possono usare diversi tipi di formalismi, che si differenziano innanzitutto per il modello dei dati che supportano, cioè per i meccanismi di astrazione offerti per modellare. A loro volta, i modelli dei dati si differenziano per:

- i modi in cui si rappresenta la struttura della conoscenza concreta;
- la possibilità di rappresentare aspetti procedurali;
- gli operatori disponibili sui fatti rappresentati.

Nel seguito si esamineranno innanzitutto i meccanismi di astrazione di un *modello ad oggetti*, che sono ritenuti i più adatti per modellare in modo naturale e diretto situazioni reali. Per semplicità non si prenderanno in considerazione la conoscenza procedurale e la comunicazione. Successivamente verrà fatta una panoramica dei meccanismi di astrazione di altri modelli dei dati e in particolare del *modello relazionale*, che verrà poi approfondito nei capitoli successivi perché è ormai uno standard dei sistemi per basi di dati.

2.3.1 Rappresentazione della struttura della conoscenza concreta

I modelli dei dati sono stati pensati inizialmente per descrivere solo la struttura della conoscenza concreta ed alcuni vincoli d'integrità, e non per descrivere proprietà calcolate od operazioni di base. L'aspetto interessante del modello ad oggetti è invece la proposta di un insieme di meccanismi che permettono di modellare sia i dati che le operazioni di base per agire su di loro.

Un modello dei dati ad oggetti è caratterizzato dalle nozioni di *oggetto*, *tipo di oggetto*, *classe*, *gerarchie fra tipi*, *definizioni per ereditarietà* e *gerarchie fra classi*. Per rendere più chiara la presentazione, verranno dati esempi di utilizzo di questi meccanismi utilizzando un formalismo grafico adeguato per definire ad un primo livello di astrazione lo schema concettuale di una base di dati, ovvero la struttura della conoscenza concreta. Per un esempio di linguaggio testuale per definire lo schema nei dettagli si veda [Albano et al., 1985], [Albano et al., 2000].

Oggetto

Un *oggetto* è la rappresentazione nel modello informatico degli aspetti strutturali e comportamentali di un'entità della realtà. Una caratteristica essenziale del modello ad oggetti, che lo differenzia dagli altri modelli, è il fatto che non esistono limitazioni sulla complessità della struttura degli oggetti, così che è sempre possibile avere una corrispondenza biunivoca fra le entità e gli oggetti del modello che le rappresentano.[2]

Definizione 2.13

Un *oggetto* è un'entità software con *stato*, *comportamento* e *identità*, che modella un'entità dell'universo del discorso. Lo stato è costituito da un insieme di *campi*, o *componenti*, che possono assumere valori di qualsiasi complessità che modellano le proprietà dell'entità. Il comportamento è costituito da un insieme di procedure locali, eventualmente dotate di parametri, chiamate *metodi*, che modellano le operazioni di base che riguardano l'oggetto e le proprietà derivabili da altre. Le operazioni applicabili ad un oggetto sono dette *messaggi* a cui l'oggetto risponde, utilizzando i propri metodi e manipolando il proprio stato.

2. Il termine *entità* si usa per riferirsi ad uno specifico fatto interessante della realtà da modellare, mentre il termine *oggetto* si usa per riferirsi alla rappresentazione dell'entità nel modello informatico.

Definizione 2.14
L'*interfaccia* di un oggetto specifica l'insieme dei messaggi a cui esso può rispondere, con il tipo dei parametri e del risultato, e l'insieme dei campi dell'oggetto che sono accessibili dall'esterno dell'oggetto stesso, con il loro tipo.

Definizione 2.15
L'*identità* di un oggetto (*Object Identifier, OID*) è una caratteristica che è associata all'oggetto stesso fin dal momento in cui esso è creato, che non può essere modificata, e che non viene in particolare modificata dall'aggiornamento dello stato dell'oggetto. L'identità di due oggetti diversi è sempre diversa. L'identità modella quelle caratteristiche di un'entità del mondo reale che fanno sì che tale entità sia sempre percepita come la stessa anche quando cambiano i valori di alcune delle sue proprietà.

Di solito i componenti dello stato sono accessibili direttamente dall'esterno, ma sono modificabili solo mediante le procedure associate all'oggetto. Quando anche l'accesso ai componenti dello stato può avvenire solo mediante i metodi associati all'oggetto, si dice che l'oggetto *incapsula* lo stato.

Come è stato detto in precedenza, in questo capitolo si vuole proporre un formalismo grafico per descrivere gli aspetti essenziali di uno schema concettuale di una base di dati: le collezioni di oggetti e le associazioni fra loro che diano una visione immediata della struttura della base di dati. Per questa ragione i metodi degli oggetti non si prenderanno in considerazione.

Tipo oggetto

Un tipo definisce un insieme di possibili valori e le operazioni ad essi applicabili. Nella costruzione di un modello informatico i tipi scaturiscono dal processo di astrazione con il quale prima si raggruppano entità diverse, perché ritenute omogenee ai fini dell'applicazione in esame, e poi si prescinde dalle differenze tra le singole entità per evidenziare invece ciò che le accomuna e le rende omogenee, cioè la loro *struttura*.

Esempio 2.2 Entità diverse come "Tizio", "Caio" e "Sempronio", vengono considerate (classificate) di tipo Utente, per porre l'accento sul fatto che di esse interessano le proprietà tipiche delle persone in quanto fruitori dei servizi di una biblioteca, quali il nome, la residenza e il recapito telefonico.

Ogni oggetto è un valore di un tipo. Ad ogni tipo sono associati degli operatori per costruire oggetti di quel tipo (costruttori di oggetti) e l'interfaccia degli oggetti del tipo stesso.

Definizione 2.16
L'*interfaccia di un tipo oggetto* specifica i componenti dello stato che sono accessibili dall'esterno e il loro tipo.

I nomi dei componenti accessibili dello stato sono detti anche *attributi degli oggetti*.

Come accade per le proprietà delle entità, un attributo di un oggetto può avere valori di tipo *atomico* o *strutturato*, *univoco* o *multivalore*.

Esempio 2.3 Nell'esempio della biblioteca, alcuni possibili attributi del tipo Utente sono CodiceFiscale, Nome, Residenza, AnnoDiNascita, con i primi due associati a valori di tipo string, sequenza di caratteri alfanumerici, il terzo strutturato e il quarto associato a valori di tipo int.

Rappresentazione grafica

Un tipo oggetto si rappresenta con una notazione simile a quella usata in UML (*Unified Modeling Language*):[3] un rettangolo diviso in tre parti; la prima contiene il nome del tipo; la seconda parte contiene i componenti dello stato e il loro tipo; la terza parte, che può mancare, contiene la descrizione di vincoli d'integrità. Per esempio, in Figura 2.1 è mostrata una rappresentazione per il tipo Persona.

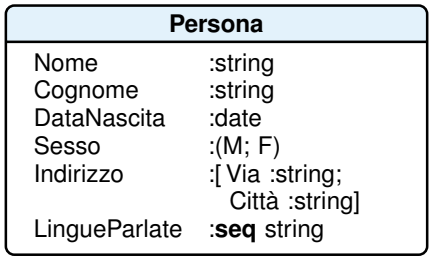

Figura 2.1: Esempio di tipo oggetto

Per definire i tipi degli attributi di un tipo oggetto si utilizzano tipi primitivi e non primitivi, ma non tipi oggetti, come invece accade in UML.[4]

I tipi primitivi comprendono int, real, bool, date e string.

Tipi non primitivi possono essere costruiti applicando i seguenti operatori ad altri tipi (non necessariamente primitivi):

– l'operatore *tipo record* costruisce un tipo i cui valori sono record, ovvero insiemi di coppie (etichetta, valore associato). Il tipo record specifica le etichette che de-

3. Un tipo oggetto è detto *classe* in UML, ma noi useremo questo termine con un altro significato, come vedremo più avanti.
4. Questa modalità di definizione di un tipo oggetto non coincide con quella prevista nell'UML e, come vedremo anche più avanti, per modellare in modo semplice una base di dati si preferirà introdurre altre notazioni per trattare alcuni aspetti importanti delle basi di dati e non ancora previste nello standard UML.

vono essere presenti nei valori di quel tipo, ed il tipo del valore associato ad ogni etichetta, con la seguente sintassi:

[A_1:T_1; . . . ; A_n:T_n]

– l'operatore *tipo enumerazione*, insieme di etichette separate da un punto e virgola e racchiuse fra parentesi tonde. Per esempio, il tipo dell'attributo Sesso è (M; F).
– l'operatore *tipo sequenza*, seq T, costruisce un tipo i cui valori sono sequenze di valori con tipo T. Una sequenza si differenzia da un insieme perché gli elementi sono ordinati e possono essere ripetuti; una sequenza è quindi un multinsieme finito e ordinato di valori dello stesso tipo, eventualmente vuoto.

Una proprietà strutturata si modella con un attributo di tipo record, mentre il tipo sequenza modella le proprietà multivalore.

Classe

Come gli oggetti modellano entità, così le classi modellano collezioni. Si osservi che nel gergo dei linguaggi ad oggetti il termine classe si usa con il significato di tipo oggetto.

Definizione 2.17

Una *classe* è un insieme di oggetti dello stesso tipo, modificabile con operatori per includere o estrarre elementi dall'insieme, al quale sono associabili alcuni vincoli di integrità.

Rappresentazione grafica

Per non complicare la rappresentazione grafica dello schema concettuale di una basi di dati, si preferisce non introdurre una nuova notazione grafica per descrivere le classi, ma fare l'ipotesi che (*a*) gli unici tipi oggetti che si modellano sono quelli che rappresentano elementi di classi e (*b*) il nome del tipo degli oggetti è anche il nome della classe. Pertanto da ora in poi, una definizione come quella di Figura 2.1 si interpreterà come la classe Persona con elementi aventi la struttura mostrata. Useremo nel seguito la convenzione di usare nomi al plurale negli esempi di classi.

Quando lo schema concettuale contiene numerose classi, ed elementi con struttura complessa, per non appesantire la rappresentazione grafica, si preferisce descriverle usando solo un rettangolo etichettato con il nome della classe e descrivere poi separatamente la struttura dei suoi elementi.[5]

5. Gli editori grafici di schemi concettuali di solito consentono di passare con un clic sul rettangolo di una classe alla sua visualizzazione estesa con gli attributi.

Rappresentazione delle associazioni

Associazioni binarie senza proprietà

Un'associazione fra due collezioni C_1 e C_2 è una relazione binaria fra C_1 e C_2, e si rappresenta nella notazione grafica con una linea che collega le classi che rappresentano le due collezioni. La linea è etichettata con il nome dell'associazione che di solito viene scelto utilizzando un predicato che dia un significato alla frase con la struttura "soggetto predicato complemento" ottenuta leggendo il diagramma da sinistra a destra (o dall'alto al basso), dove il soggetto è il generico elemento della prima collezione e il complemento il generico elemento della seconda collezione. Ad esempio, con riferimento alla Figura 2.2: *uno studente ha sostenuto un esame*. Quando però non si riesce a trovare un verbo specifico per l'associazione, oppure quando più associazioni in uno schema finirebbero per avere lo stesso nome, si può utilizzare come nome dell'associazione la concatenazione dei nomi delle classi coinvolte.

L'univocità di un'associazione, rispetto ad una classe C_1, si rappresenta disegnando una freccia singola sulla linea che esce dalla classe C_1 ed entra nella classe C_2; l'assenza di tale vincolo è indicata da una freccia doppia. Una freccia singola che esce dalla classe C_1 si può quindi leggere come "ad ogni elemento della classe C_1 corrisponde un unico elemento nella classe C_2", ovvero come "ogni elemento della classe C_1 partecipa al più ad un'istanza dell'associazione". Similmente, la parzialità è rappresentata con un taglio sulla linea vicino alla freccia, mentre il vincolo di totalità è rappresentato dall'assenza di tale taglio.

Esempio 2.4 In Figura 2.2 è rappresentata l'associazione fra studenti ed esami superati dagli studenti (esami intesi come eventi di esaminazione): ogni esame riguarda uno ed un solo studente, mentre non ci sono vincoli di totalità né di univocità sugli esami superati da uno studente.

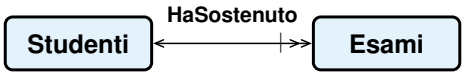

Figura 2.2: Rappresentazione di classi e associazioni

Alcuni autori propongono una specifica notazione per trattare associazioni (1:M) con inversa totale quando si modellano oggetti *composti*, ovvero oggetti con componenti che sono altri oggetti, come una bicicletta con componenti un sedile, due ruote, un telaio ecc. (associazione *parte-di* o *part-of*). Questa descrizione può essere utile per evidenziare che alcune operazioni su un oggetto composto si applicano anche a tutte le sue parti; ad esempio, spostare in un disegno una bicicletta vuol dire spostare tutte le sue componenti, oppure eliminare una bicicletta da una collezione vuol dire eliminare anche le sue componenti dalle rispettive classi.

Associazioni binarie con proprietà

Se l'associazione binaria ha delle proprietà, il suo nome si scrive in un rettangolo collegato alla linea che la rappresenta e contenente la definizione delle proprietà. In Figura 2.3 è mostrato un esempio di un'associazione tra i libri di una biblioteca e gli utenti, che modella i prestiti, e che ha una proprietà Data.

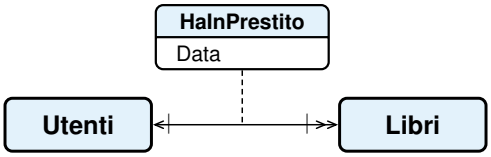

Figura 2.3: Rappresentazione di associazioni con proprietà

Esempio 2.5 Un'associazione con proprietà, come quella tra i libri di una biblioteca e gli utenti di Figura 2.3, può essere modellata interpretando un'istanza di associazione come un'entità e definendo così una classe Prestiti, associata in modo $(1:1)$ ai Libri e in modo $(N:1)$ agli Utenti, e aggiungendo un attributo Data alla classe Prestiti stessa (si veda la Figura 2.4).

Il fatto che un utente u di una biblioteca ha preso in prestito il libro l in una data d è modellato inserendo nella classe Prestiti un oggetto con attributo Data uguale a d, associato all'utente u e al libro l.

Figura 2.4: Trasformazione di associazioni con proprietà

Associazioni ricorsive

Le associazioni ricorsive sono relazioni binarie fra gli elementi di una stessa collezione. Un classico esempio è la relazione ÈMadreDi sulla collezione delle Persone: una persona può essere madre di più persone della collezione e può avere una madre nella collezione. Nella rappresentazione di questo tipo di associazione, per una corretta interpretazione del fatto rappresentato occorre etichettare la linea non solo con il nome dell'associazione, ma anche con dei nomi per specificare il ruolo che hanno i due componenti in un'istanza di associazione (Figura 2.5).

Associazioni n-arie

In generale le associazioni possono essere di grado maggiore di due. Associazioni di grado tre qualche volta si usano, quelle di grado quattro si usano raramente e quelle di

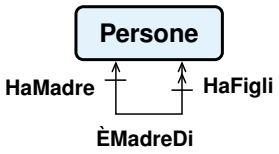

Figura 2.5: Associazione ricorsiva

grado superiore sono una curiosità difficile da comprendere e da usare. Per semplicità non daremo una notazione grafica per rappresentare associazioni non binarie, e si mostra nell'esempio che segue come trattarle con un'opportuna trasformazione della rappresentazione, come si è fatto nel caso delle associazioni con proprietà.

Esempio 2.6 Si supponga di voler rappresentare l'associazione fra Voli, Passeggeri e Posti. Per ogni volo (ad esempio, il volo Pisa-Roma del 1-1-2005), al momento dell'imbarco, viene assegnato un posto a ciascun passeggero. L'associazione è multivalore rispetto ad ogni collezione, poiché ogni singolo volo, passeggero, e posto, partecipa in generale a diverse istanze di associazione, ma con i vincoli che, in un singolo volo, ogni posto è associato al più ad un passeggero ed ogni passeggero può occupare al più un posto.

Un'associazione ternaria può essere modellata interpretando un'istanza di associazione come un'entità e definendo così una classe Imbarchi, associata in modo $(1 : N)$ ai Voli, ai Passeggeri e ai Posti. Se l'associazione avesse delle proprietà, verrebbero aggiunti degli attributi alla classe Imbarchi (si veda la Figura 2.6).

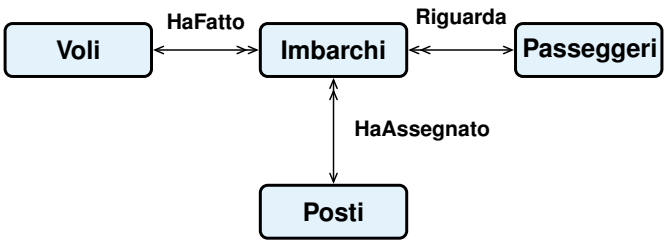

Figura 2.6: Associazione ternaria trasformata in classe

Esempio 2.7 In Figura 2.7 è mostrata la rappresentazione con il formalismo grafico di fatti riguardanti una biblioteca universitaria. Si lascia al lettore l'interpretazione dei fatti rappresentati, limitandosi ad indicare che più copie di uno stesso documento sono modellate da più documenti con la stessa descrizione bibliografica, e che la parte dello schema riguardante i termini è una rappresentazione di

un *thesaurus*, cioè di una collezione di termini su cui sono definite alcune relazioni semantiche quali sinonimia ("Base di dati" è il sinonimo standard di "Database"), specializzazione ("Base di dati relazionale" è più specifico di "Base di dati") ecc.

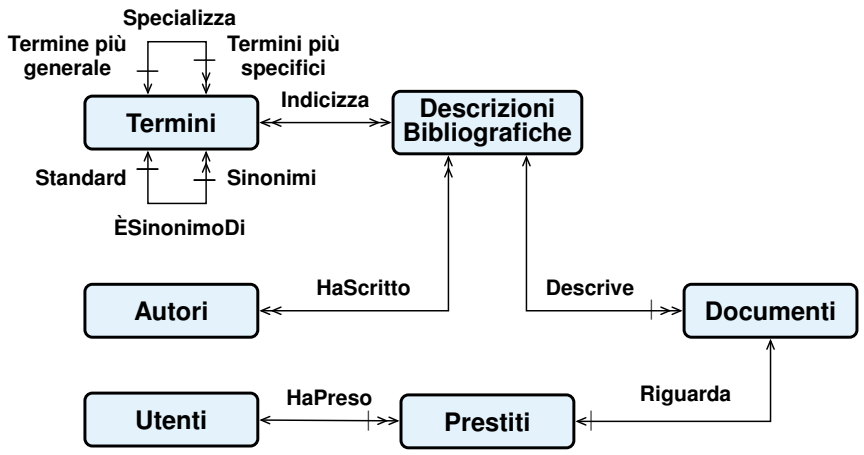

Figura 2.7: Un esempio di schema di base di dati

Gerarchie di tipi

La relazione di sottotipo è una relazione di ordinamento parziale sull'insieme dei tipi tale che un valore che appartiene al sottotipo può essere utilizzato in tutti i contesti in cui è previsto un valore del supertipo.

Ad esempio, il tipo Utente può essere definito come un supertipo dei tipi Studente e Docente, per rappresentare il fatto che un'entità classificata come Studente e Docente possa essere classificata anche come Utente, e apparire quindi in tutti i contesti in cui può apparire un Utente. Si prescinde così dalle proprietà che rendono queste entità semanticamente diverse, e si evidenzia invece che sono entità omogenee ad un particolare livello di astrazione, ad esempio con Nome, Residenza e DataDiNascita come proprietà comuni.

Qualche relazione di sottotipo è presente in molti linguaggi (tipicamente, gli interi sono un sottotipo dei numeri in virgola mobile), ma nei linguaggi ad oggetti assume un ruolo particolarmente importante poiché è collegata strettamente alla nozione di "ereditarietà".

Definizione per ereditarietà

Si dice che la definizione dell'interfaccia di un tipo oggetto è data per ereditarietà a partire da un tipo T quando tale definizione è fornita specificando cosa va aggiunto o modificato per ottenere il nuovo tipo a partire da T. In particolare, un'interfaccia è definita per ereditarietà da un'interfaccia $\mathcal{I}$ specificando quali attributi aggiungere

ad $\mathcal{I}$ e come, eventualmente, modificare (*overriding*) il tipo degli attributi "ereditati", ovvero degli attributi già presenti in $\mathcal{I}$.

In genere si impone che, nel definire un'interfaccia per ereditarietà, se si modifica il tipo di un attributo, il nuovo tipo sia un sottotipo del tipo precedente (vincolo di *ereditarietà stretta*) [Albano et al., 1985], [Albano et al., 2000]. Grazie a questo vincolo, un valore di un tipo definito per ereditarietà a partire da T ha tutti gli attributi di T e, se il tipo di un attributo è cambiato, il nuovo tipo è compatibile (un sottotipo) con il tipo precedente. Ad esempio, un oggetto di tipo Studente, definito per ereditarietà stretta dal tipo Utente, ha un Residenza, come ogni altro utente. Se il tipo di Residenza è ridefinito nel tipo Studente, sarà comunque un tipo utilizzabile in tutti i contesti in cui sono utilizzabili valori del tipo che aveva Residenza nel tipo Utente. Ne consegue che, in generale, un tipo definito per ereditarietà stretta costituisce anche un sottotipo del tipo originario.

Infine, un tipo può essere definito per ereditarietà a partire da un unico supertipo (*ereditarietà singola*) o da più supertipi (*ereditarietà multipla*). Questa seconda possibilità è molto utile ma può creare alcuni problemi quando lo stesso attributo viene ereditato, con tipi diversi, da più tipi antenato.

La possibilità di definire tipi oggetto per ereditarietà è di grande interesse dal punto di vista dell'ingegneria del software, sia perché rende il linguaggio più adatto per modellare in modo naturale situazioni complesse, sia perché consente di sviluppare le applicazioni incrementalmente per raffinamento di definizioni di tipi. Ciò che rende questo meccanismo interessante è, in particolare, la combinazione di sovraccarico dei messaggi (*overloading*) e risoluzione dei nomi dei messaggi a tempo di esecuzione (*late binding*).

Gerarchia di inclusione fra classi

La gerarchia di inclusione è una relazione di ordinamento parziale sull'insieme delle classi tale che se C_1 è una sottoclasse di C_2 (è inclusa in C_2), allora gli elementi di C_1 sono un sottoinsieme degli elementi di C_2 (*vincolo estensionale*). Come conseguenza, quando nel modello si aggiunge un elemento ad una sottoclasse, esso viene automaticamente a far parte anche delle superclassi. Affinché questo non provochi problemi di tipo, si impone anche il seguente *vincolo strutturale*: se C_1 è una sottoclasse di C_2, allora il tipo degli elementi di C_1 è un sottotipo del tipo degli elementi di C_2.

Quando si definiscono più sottoclassi di una stessa classe, su questo insieme di sottoclassi possono essere definiti i seguenti vincoli:

- un insieme di sottoclassi soddisfa il *vincolo di disgiunzione* se ogni coppia di sottoclassi in questo insieme è disgiunta, ovvero è priva di elementi comuni (*sottoclassi disgiunte*);
- un insieme di sottoclassi soddisfa il *vincolo di copertura* se l'unione degli elementi delle sottoclassi coincide con l'insieme degli elementi della superclasse (*sottoclassi copertura*).

I due vincoli sono indipendenti fra loro; quando sono entrambi soddisfatti, l'insieme di sottoclassi costituisce una *partizione* della superclasse.

Una sottoclasse può essere definita anche a partire da un'altra sottoclasse, modellando così gerarchie a più livelli. Ad esempio, la classe Studenti potrebbe essere a sua volta specializzata introducendo le sottoclassi partizione StudentiInCorso e StudentiFuoriCorso. In generale, una sottoclasse può essere popolata creando in essa nuovi elementi, che diventano anche elementi delle superclassi, oppure spostando in essa elementi già presenti nella superclasse.

Infine, la gerarchia non è necessariamente ad albero (*gerarchie singole*), perché una sottoclasse può essere definita a partire da più classi (*gerarchie multiple*). Ad esempio, gli StudentiLavoratori potrebbero essere una sottoclasse sia degli Studenti che dei Dipendenti.

In generale, al meccanismo delle sottoclassi sono associati operatori per:

- creare un elemento in una classe, che diventa anche un elemento delle sue superclassi per il vincolo estensionale;
- rimuovere un elemento da una classe, con la conseguente rimozione da tutte le sue sottoclassi per il vincolo estensionale;
- controllare se un elemento di una classe è anche in una sua sottoclasse.

Osservazione

Di solito si assume che un oggetto una volta creato con un certo tipo (ad esempio, Persona) non possa acquisire nuovi tipi (ad esempio, Studente), e che non sia quindi possibile "spostare" un elemento di una classe in una o più sottoclassi. Nel seguito non si terrà conto di questa limitazione, perché sono stati proposti linguaggi ad oggetti per basi di dati con operatori che permettono questo tipo di spostamenti, come il linguaggio Galileo [Albano et al., 1985], [Albano et al., 2000].

Rappresentazione grafica

I quattro tipi di sottoclassi si descrivono come mostrato in Figura 2.8: il vincolo di disgiunzione viene rappresentato con il pallino nero, mentre il vincolo di copertura viene rappresentato con una freccia doppia verso la superclasse.

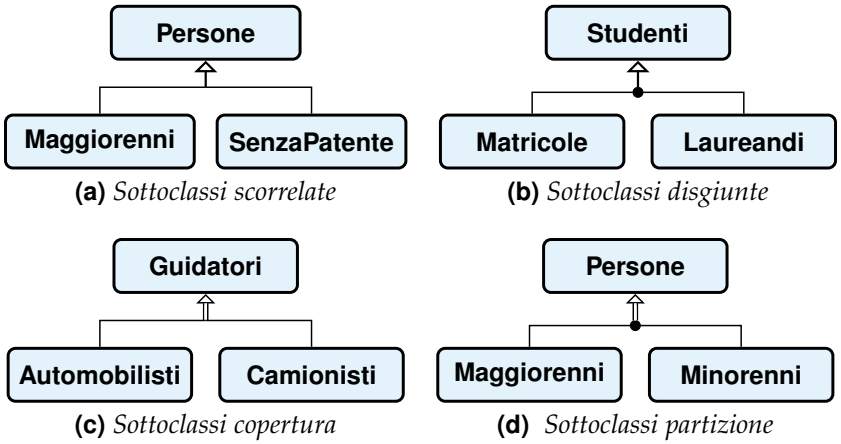

Figura 2.8: Rappresentazione grafica dei tipi di sottoclassi

Sottoclassi scorrelate, non richiedendo né il vincolo di copertura né quello di disgiunzione, si possono rappresentare anche con la notazione usata in Figura 2.9. Le gerarchie multiple si rappresentano come in Figura 2.10.

Figura 2.9: Rappresentazione grafica di sottoclassi scorrelate

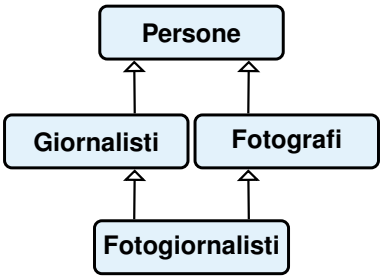

Figura 2.10: Rappresentazione grafica di gerarchie multiple

Esempio 2.8 Passando ad un successivo livello di dettaglio si può rendere più fedele lo schema per la base di dati della biblioteca di Figura 2.7, distinguendo ad esempio le sottoclassi degli studenti e docenti della classe utenti, nonché la sottoclasse dei documenti per sola consultazione, che possono essere presi in prestito solo dagli utenti che sono docenti (Figura 2.11). Infine in Figura 2.12 lo schema viene completato con la struttura degli elementi delle classi.

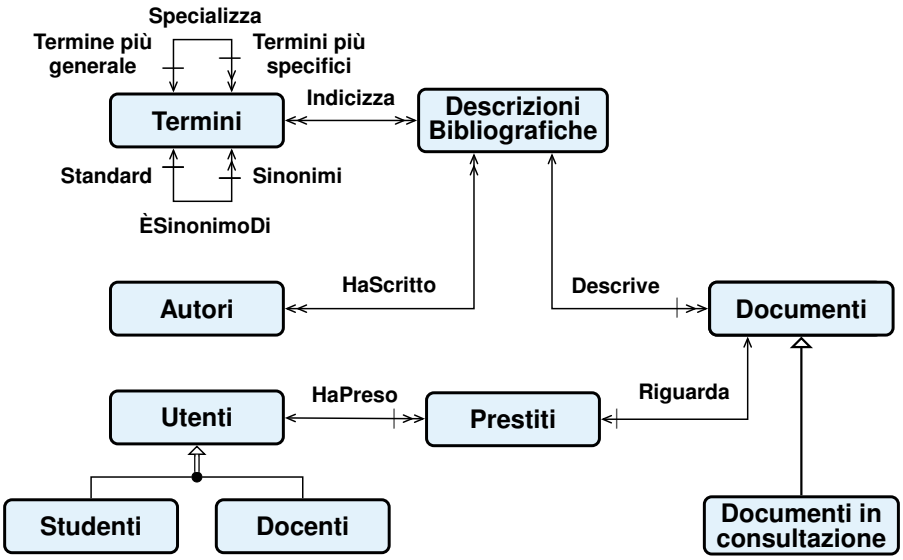

Figura 2.11: Raffinamento delle schema di Figura 2.7

2.3.2 Rappresentazione degli altri aspetti della conoscenza astratta

Nel modellare una situazione reale, in generale non è sufficiente limitarsi alla descrizione della struttura della conoscenza concreta con i meccanismi di astrazione del modello dei dati prescelto, come è stato visto finora, ma è necessario descrivere anche vincoli d'integrità che impongono ulteriori restrizioni sui possibili valori della conoscenza concreta.

I vincoli possono essere descritti in modo dichiarativo, con formule del calcolo dei predicati, oppure mediante controlli da eseguire nelle operazioni (di base o degli utenti). La preferenza è per l'approccio dichiarativo per due ragioni principali. Innanzitutto è più semplice stabilire la coerenza dei vincoli con la definizione dei requisiti ed apportare modifiche per adeguarli a nuove esigenze. In secondo luogo, una descrizione dichiarativa dei vincoli si presta allo sviluppo di strumenti da usare nella fase di progettazione per stabilire eventuali inconsistenze o ridondanze, cioè

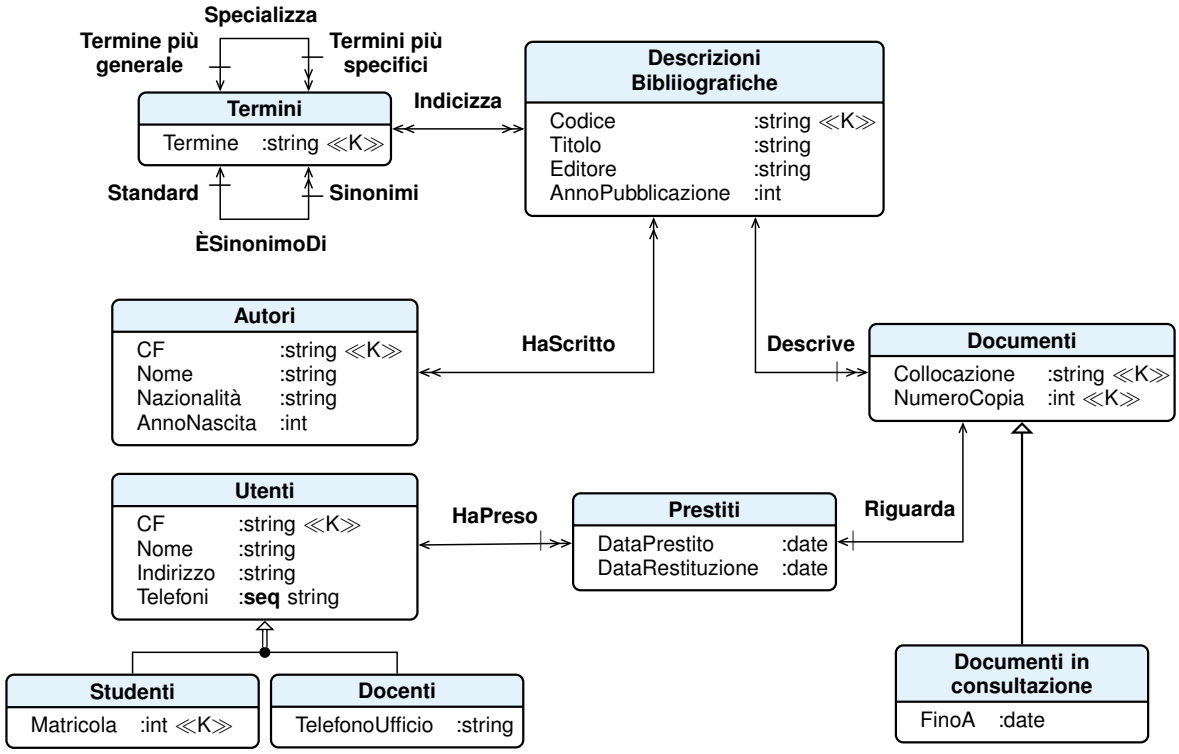

Figura 2.12: Schema di Figura 2.11 con gli attributi degli elementi delle classi

vincoli logicamente implicati da altri. Infine si evita di ripetere alcuni controlli in più operazioni.

Rappresentazione grafica

Si è già mostrato come rappresentare graficamente i vincoli sulle proprietà strutturali delle associazioni e sul tipo di sottoclassi.

In Figura 2.13 è mostrato come la definizione di una classe si arricchisce di una sezione dedicata alla descrizione dei vincoli. Gli attributi marcati con la stringa ≪K≫ sono una chiave. Se vi sono altre n chiavi si usano le stringhe ≪K1≫ ... ≪Kn≫.

Vincoli generali andrebbero espressi usando un opportuno formalismo, come l'OCL (*Object Constraint Language*) proposto per l'UML, e in figura sono mostrati alcuni semplici esempi, dove si usa la notazione self.nomeAttributo per fare riferimento all'attributo nomeAttributo dell'oggetto. In uno stadio iniziale del progetto può essere sufficiente descrivere i vincoli usando il linguaggio naturale.

2.3.3 Rappresentazione della conoscenza procedurale

Mentre la conoscenza delle operazioni di base viene modellata con il meccanismo dei metodi degli oggetti, le operazioni degli utenti vengono modellate con il mecca-

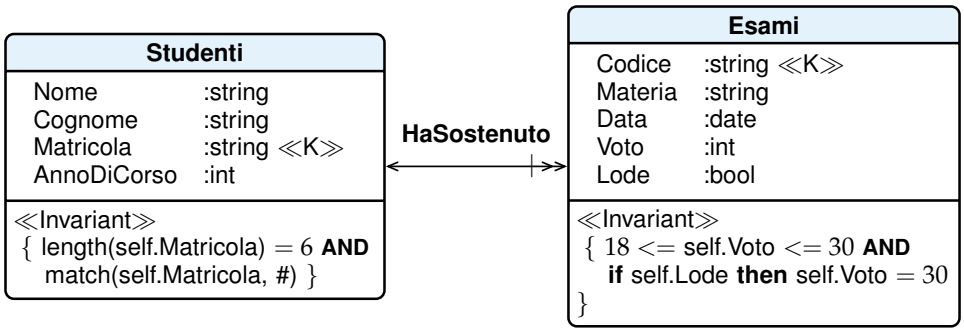

Figura 2.13: Esempio di classi con vincoli

nismo delle transazioni, programmi sequenziali che consentono di modellare l'evoluzione dell'universo del discorso astraendo da malfunzionamenti e da interferenze indesiderate con altre transazioni che accedono concorrentemente ai dati.

Un'operazione (di base o degli utenti) si specifica, a questo livello di astrazione, dando le seguenti informazioni:

- il *nome*, un identificatore non già utilizzato per altri elementi dello schema;
- lo *scopo* dell'operazione, descritto con un testo in linguaggio naturale;
- gli *argomenti* dell'operazione, un elenco di coppie "identificatore :tipo";
- il tipo del *risultato*;
- il testo di eventuali messaggi di errore;
- le classi e le associazioni che *usa* ma non modifica;
- le classi e le associazioni che *modifica*, per aggiunta di elementi o modifica di attributi degli elementi;
- due condizioni, la *precondizione* e la *postcondizione*, che devono essere vere nello stato di partenza e nello stato di arrivo della base di dati. In una condizione si possono usare i nomi dei parametri o le classi che l'operazione usa o modifica.

Queste informazioni si danno nel formato (*descrittore operazioni*) mostrato in Figura 2.14.

2.3.4 Rappresentazione della comunicazione

Nel modellare la comunicazione con il sistema informativo, ed in particolare con il sistema informatico, si suppone che questa comunicazione avvenga come un dialogo, nel quale il sistema presenta all'utente una *forma* che è caratterizzata da un certo aspetto, dalla presenza di alcuni *campi modificabili* e dalla presenza di alcuni *elementi attivi* (bottoni, menu, barre di scorrimento ecc.). L'utente modifica (o riempie) i campi modificabili oppure agisce sugli elementi attivi, ed il sistema reagisce a queste azioni dell'utente modificando la forma, o ritirandola, o presentando una nuova forma, fino a che l'interazione non termina. Questo modello può essere esteso, supponendo ad

Operazione	NuovoEsame
Scopo	Immissione dei dati di un nuovo esame
Argomenti	CodiceMateria :string, UnaMatricola :string, DataEsame :date, UnVoto :int, ConLode :bool;
Risultato	(OperazioneEseguita; Errore);
Errori	Materia sconosciuta; Candidato sconosciuto; Voto errato: deve essere fra 18 e 30; Lode solo con voto 30;
Usa	Esami, Studenti, HaSostenuto;
Modifica	Esami, HaSostenuto;

Prima	18 <= UnVoto <= 30;
	if ConLode **then** UnVoto = 30;
	some x **In** Studenti **with** Matricola = UnaMatricola;
	Not some e **In** HaSostenuto
	  with (e.Esami.Materia = CodiceMateria
	  And e.Studenti.Matricola = UnaMatricola);

Poi	**some** e **In** HaSostenuto
	  with (e.Esami.Materia = CodiceMateria
	  And e.Studenti.Matricola = UnaMatricola);

Figura 2.14: Esempio di descrittore di operazione.

esempio che il sistema sia disponibile a presentare all'utente più forme in uno stesso momento, e a permettere all'utente di agire su queste forme in qualunque ordine. Un'ulteriore estensione riguarda la possibilità per l'utente di sottomettere al sistema non solo forme riempite ma anche interrogazioni scritte in un linguaggio formale opportuno, che il sistema esegue. Un'ultima estensione riguarda la possibilità per l'utente di effettuare interazioni meno strutturate, sottoponendo al sistema richieste in forma testuale o con dialoghi vocali in linguaggio naturale. Queste forme di comunicazione non strutturata sono ovviamente essenziali in un sistema informativo in cui siano presenti anche componenti umane.

Il modello ad oggetti è particolarmente adatto a modellare la comunicazione non solo per la possibilità di catturare la natura "attiva" dei componenti delle interfacce grafiche, ma anche perché esso permette di trarre vantaggio dal fatto che tra i tipi degli oggetti che costituiscono un'interfaccia grafica (detti *widget*) sono naturalmente definite delle ricche gerarchie di sottotipo e di eredità.

2.4 Altri modelli dei dati

Vengono presentate le caratteristiche principali di altri modelli dei dati, molto noti nell'area base di dati, per modellare la struttura della conoscenza concreta: il *modello entità-relazione* e il *modello relazionale*.

2.4.1 Il modello entità-relazione

Il modello entità-relazione, proposto da Chen nel 1976, è il modello più popolare per il disegno concettuale di basi di dati. Verrà presentato nella sua forma originaria, estesa successivamente per trattare altri aspetti, in particolare la generalizzazione. Rispetto al modello ad oggetti, questo modello non tratta aspetti procedurali (metodi) né gerarchie di inclusione tra collezioni, non distingue collezioni e tipi, non supporta alcun meccanismo di ereditarietà. Definisce però un meccanismo per modellare direttamente le associazioni, e in particolare le associazioni non binarie o con proprietà.

Più in dettaglio, il modello prevede due meccanismi di astrazione distinti: uno per modellare insiemi di entità, con le relative proprietà, ed un altro per modellare le associazioni (chiamate "relazioni"). Le collezioni sono qui chiamate *tipi di entità*, e gli attributi dei loro elementi possono assumere solo valori di tipo primitivo.

Il formalismo grafico usato per rappresentare i meccanismi di astrazione del modello entità-relazione è chiamato *diagrammi entità-relazione* (*entity-relationship diagrams*), o *diagrammi ER*, e, come nel caso del modello ad oggetti, i rettangoli rappresentano tipi di entità e gli archi interrotti da un rombo rappresentano associazioni. Anche qui gli archi sono etichettati con le proprietà strutturali delle associazioni, codificate però da una coppia di interi (min, max) che rappresentano, come nei descrittori testuali presentati in precedenza per le associazioni nel formalismo ad oggetti, il numero minimo e massimo di volte che un elemento di un tipo di entità può partecipare alla relazione.

In Figura 2.15 è rappresentata una relazione N ad M tra studenti e corsi, parziale per gli studenti, e totale sui corsi, con in più il vincolo che uno studente può frequentare al più cinque corsi, ed ogni corso deve avere almeno quattro studenti e può averne al più trecento. La relazione tra corsi e corsi integrativi è univoca e totale per i corsi integrativi, mentre è multivalore e parziale per i corsi, con in più il vincolo che ad un corso possono essere collegati al più due corsi integrativi.

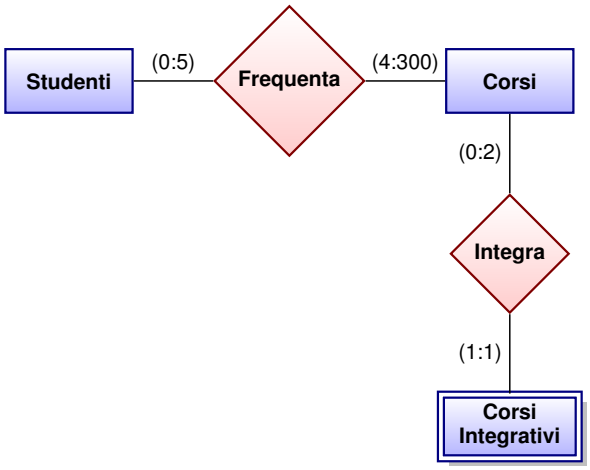

Figura 2.15: Diagramma entità-relazione

Il rettangolo doppio indica un *tipo di entità debole* (*weak entity types*), i cui elementi hanno un'esistenza che dipende da quelli del tipo entità con cui sono in relazione, similmente a quanto accade per gli oggetti composti.

Le relazioni possono essere anche non binarie e avere proprietà proprie.

Questo modello è stato successivamente esteso per trattare attributi con valori non primitivi e le gerarchie fra tipi di entità, rendendo la notazione grafica molto simile a quella di un modello ad oggetti. In Figura 2.16 è mostrato lo schema della biblioteca con un diagramma entità-relazione.

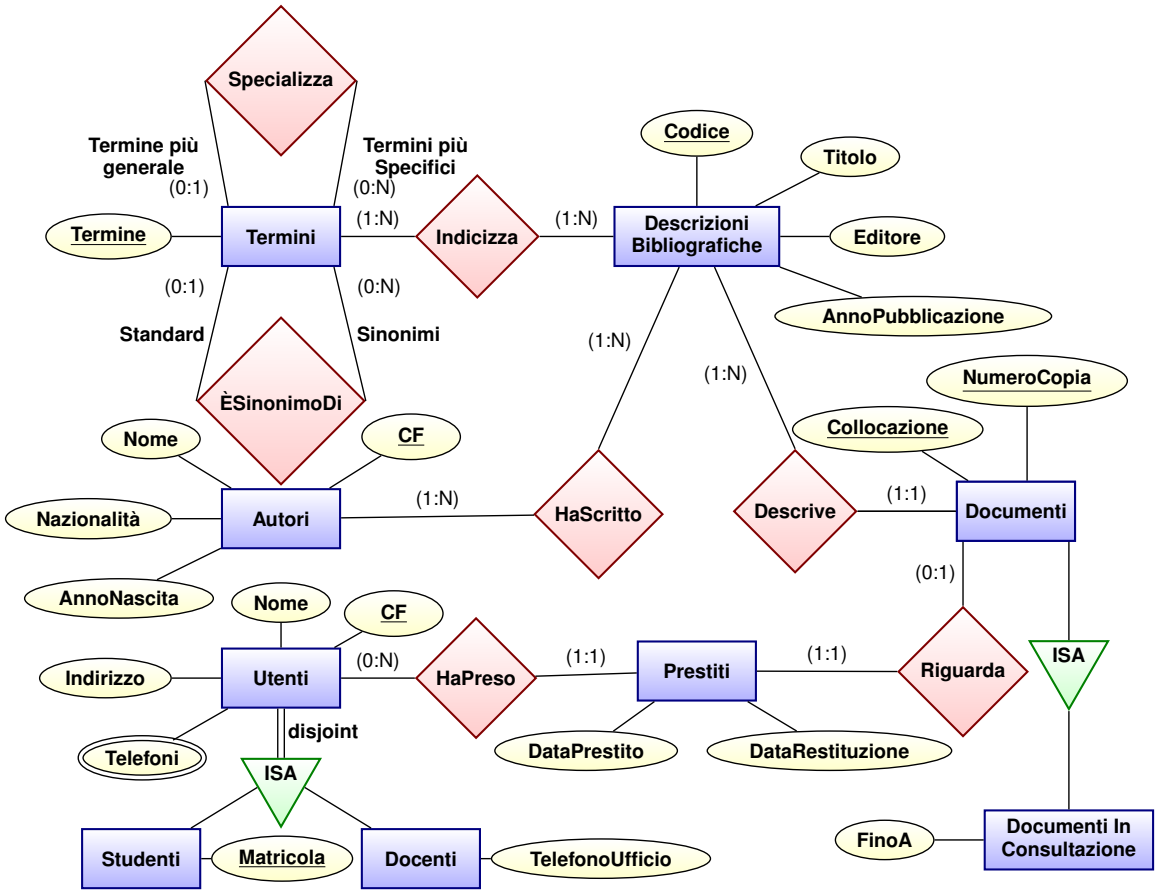

Figura 2.16: Lo schema della biblioteca con un diagramma entità-relazione

2.4.2 Il modello relazionale

Il modello relazionale dei dati è stato proposto nel 1970 ed adottato nei sistemi commerciali a partire dal 1978. È il modello dei dati usato dagli attuali sistemi commerciali.

I meccanismi per definire una base di dati con questo modello sono l'*ennupla* e la *relazione*. Un *tipo ennupla* è un insieme di coppie (attributo, tipo primitivo) ed un valore di tipo ennupla è un insieme di coppie (attributo, valore), dette anche *campi*, con gli stessi attributi del tipo e in cui il valore di ogni attributo appartiene al corrispondente tipo primitivo. Una *relazione* è un insieme di ennuple con lo stesso tipo.

Un insieme di attributi i cui valori determinano univocamente un'ennupla di una relazione R è una *superchiave* per R; una superchiave tale che togliendo un qualunque attributo essa non sia più una superchiave è una *chiave* per R. Tra le chiavi di R ne viene scelta una come *chiave primaria*. Le associazioni tra i dati sono rappresentate attraverso opportuni attributi, chiamati *chiavi esterne*, che assumono come valori quelli della chiave primaria di un'altra relazione.

Esempio 2.9 Ad esempio, per descrivere il fatto che un esame è associato ad uno studente, nello schema della relazione Esami viene previsto un attributo Candidato che assume come valore la chiave primaria degli Studenti, cioè la Matricola. In Figura 2.17 è data una rappresentazione grafica in cui si evidenzia questo tipo di informazione e in Figura 2.18 si riporta la descrizione completa della struttura delle due relazioni.

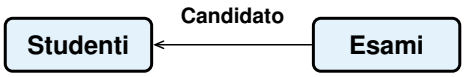

Figura 2.17: Diagramma relazionale

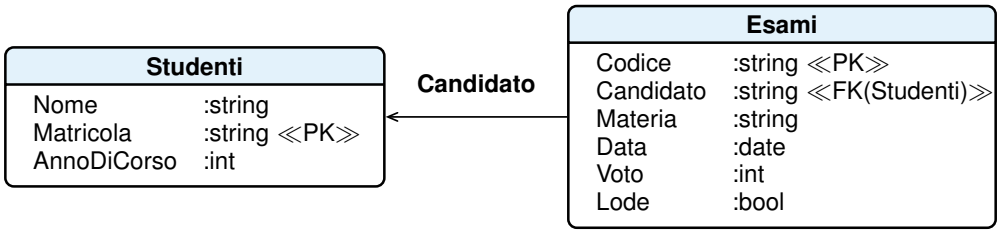

Figura 2.18: Diagramma relazionale con struttura delle relazioni

Il modello relazionale è molto intuitivo, ma l'aspetto più innovativo rispetto ai modelli preesistenti, gerarchico e reticolare, consiste nelle operazioni previste sulle relazioni: queste sono operazioni insiemistiche che restituiscono sempre altre relazioni, inoltre, fra le operazioni primitive ne è prevista una che consente di creare una nuova relazione come giunzione di due relazioni, per combinare ogni ennupla di una relazione con quelle che gli corrispondono nell'altra.

Questo modello si differenzia dal modello ad oggetti in particolare per l'assenza di metodi, di gerarchie di inclusione e di ereditarietà, per il fatto che gli attributi di

un'ennupla devono avere un tipo primitivo, e infine perché non esiste un concetto simile all'identità del modello ad oggetti, per cui due ennuple con gli stessi valori degli attributi coincidono e non possono quindi modellare due entità diverse. D'altra parte, questo modello ha una semantica insiemistica estremamente semplice, che lo rende più semplice da capire e particolarmente adatto ad un trattamento formale che ha consentito lo sviluppo di una teoria della progettazione di basi di dati. I Capitoli 4-5 sono dedicati a questo modello dei dati.

2.5 Conclusioni

Sono stati presentati i vari aspetti da trattare nella costruzione di un modello informatico di un universo del discorso e alcuni meccanismi di astrazione che permettono di modellare alcuni di tali aspetti. In particolare, per la rappresentazione della struttura della conoscenza concreta sono stati presentati i meccanismi di astrazione dei modelli a oggetti, utili per modellare in modo naturale e diretto i fatti interessanti di situazioni complesse.

I meccanismi di astrazione visti si prestano a una semplice rappresentazione grafica, per visualizzare nelle sue linee essenziali lo schema concettuale di una base di dati, in una forma facilmente comprensibile anche da non esperti. Ciò agevola le interazioni dell'analista delle applicazioni con i committenti, specialmente nei primi stadi della progettazione, quando le interazioni sono molto frequenti per chiarire gli obiettivi del sistema informatico che va realizzato. Le rappresentazioni grafiche per un modello ad oggetti o entità-relazioni esteso non sono standard, ma ne esistono diverse ed alcuni editori grafici di ambienti CASE (*Computer Aided Software Engineering*) offrono la possibilità di passare in modo automatico da una notazione all'altra.

Esercizi

1. Discutere le differenze tra i modi in cui si rappresentano le associazioni fra insiemi di dati nei modelli dei dati ad oggetti, entità-relazioni e relazionale.
2. Discutere le differenze tra le nozioni di tipo oggetto e di classe.
3. Discutere le differenze tra le nozioni di tipo oggetto definito per ereditarietà e di sottoclasse.
4. Discutere i vincoli che si possono descrivere graficamente con il modello ad oggetti.
5. Si definisca uno schema grafico per rappresentare con il modello a oggetti tre insiemi di entità: le persone e i sottoinsiemi delle persone viventi e delle persone decedute. Dare delle proprietà interessanti per gli elementi di questi insiemi. Dire quali problemi si presenterebbero nel modellare questi insiemi se il modello non offrisse né il meccanismo dei tipi oggetti definiti per ereditarietà, né il meccanismo delle sottoclassi.
6. Si vuole automatizzare il sistema di gestione degli animali in uno zoo. Ogni esemplare di animale ospitato è identificato dal suo genere (ad esempio, zebra) e da un

codice unico all'interno del genere di appartenenza. Per ogni esemplare si memorizzano la data di arrivo nello zoo, il nome proprio, il sesso, il paese di provenienza e la data di nascita. Lo zoo è diviso in aree; in ogni area c'è un insieme di case, ognuna destinata ad un determinato genere di animali. Ogni casa contiene un insieme di gabbie, ognuna contenente un solo esemplare. Ogni casa ha un addetto che pulisce ciascuna gabbia in un determinato giorno della settimana. Gli animali sono sottoposti periodicamente a controllo veterinario; in un controllo, un veterinario rileva il peso degli esemplari, diagnostica un'eventuale malattia e prescrive il tipo di dieta da seguire.

Dare uno schema grafico della base di dati.

7. Una banca gestisce informazioni sui mutui dei propri clienti e le rate del piano di ammortamento. Un cliente può avere più di un mutuo. Ai clienti viene inviato periodicamente un resoconto sulle rate pagate del tipo mostrato in figura. Per le rate pagate dopo la data di scadenza è prevista una mora. Si progetti la base di dati e successivamente si modifichi lo schema per trattare anche il fatto che i clienti fanno prima una richiesta di mutuo che poi può essere concesso con un rimborso a rate secondo un certo piano di ammortamento.

RESOCONTO MUTUO

Codice mutuo: 250 **Data**: 07/01/2005
Scadenza: 01/01/2008
Ammontare: 70 000,00
Codice cliente: 2000
Nome cliente: Mario Rossi
Indirizzo: Via Roma, 1 Pisa

Numero rata	Scadenza	Ammontare	Data Versamento
1	1/07/05	6 000,00	29/06/05
2	1/01/06	6 000,00	30/12/05
3	1/07/06	6 100,00	29/06/06
4	01/01/07	6 100,00	30/12/06

8. Si vogliono trattare informazioni su attori e registi di film. Di un attore o un regista interessano il nome, che lo identifica, l'anno di nascita e la nazionalità. Un attore può essere anche un regista. Di un film interessano il titolo, l'anno di produzione, gli attori, il regista e il produttore. Due film prodotti lo stesso anno hanno titolo diverso.

   Dare uno schema grafico della base di dati.

9. Un'azienda vuole gestire le informazioni riguardanti gli impiegati, i dipartimenti e i progetti in corso.

   Di un impiegato interessano il codice, assegnato dall'azienda, che l'identifica, il nome e cognome, l'anno di nascita, il sesso e i familiari a carico, dei quali interessano il nome, il sesso, la relazione di parentela e l'anno di nascita.

   Di un dipartimento interessano il numero, che lo identifica, il nome, la città dove si trova.

   Di un progetto interessano il numero, che lo identifica, e il nome. Un progetto è gestito da un solo dipartimento.

Gli impiegati afferiscono ad un dipartimento, che gestisce più progetti ed è diretto da un impiegato. Gli impiegati partecipano a più progetti, che si svolgono presso dipartimenti di città diverse, ad ognuno dei quali dedicano una percentuale del proprio tempo. Gli impiegati sono coordinati da un responsabile, che è un impiegato. Dei direttori e dei responsabili interessa l'anno di nomina.

Dare uno schema grafico della base di dati.

Note bibliografiche

I modelli dei dati sono trattati in tutti i libri sui sistemi per basi di dati citati nel capitolo precedente. Sulla rete si possono trovare editori grafici per schemi concettuali offerti gratuitamente.

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

2PL (Two Phase Lock), 273
3NF, 165
4GL (Fourth Generation Languages), 18, 239
4NF, 172

affidabilità, 19
algebra relazionale
differenza, 119
divisione, 123
espressione, 120
funzioni di aggregazione, 130
giunzione, 124
giunzione esterna, 125
giunzione naturale, 124
intersezione, 123
prodotto, 119
proiezione, 119
proiezione generalizzata, 130
raggruppamento, 130
restrizione, 119
ridenominazione, 118
semi-giunzione, 125
unione, 119
analisi
dei dati, 75
dei requisiti, 64
funzionale, 69, 75
anomalia, 137
API, 234
applicazione, 61
assiomi di Armstrong, 143
associazione, 33, 44
cardinalità, 34
molteplicità, 34
proprietà strutturali, 34
rappresentazione grafica, 44
attributo
di un oggetto, 42
estraneo, 151
primo, 104, 148

base di dati, 8
BCNF, 162
bloccaggio dei dati, 242
blocco a due fasi, 273, 274
Boyce-Codd
forma normale di, 162

calcolo relazionale di ennuple, 132
CASE, 75, 93
catalogo, 24, 223
checkpoint, 275
chiave, 36, 57, 104, 148
esterna, 57
calcolo, 148
esterna, 104
primaria, 57, 104
chiusura
di dipendenze funzionali, 145
di un insieme di attributi, 144
classe, 43
rappresentazione grafica, 43
collezione, 32
comunicazione, 37
conoscenza
astratta, 35
concreta, 30

rappresentazione, 40
struttura, 35
della comunicazione, 37
procedurale, 36
operazioni degli utenti, 37
operazioni di base, 37
controllo
della concorrenza, 22, 273
copertura, 151
copertura canonica, 151

data flow diagram, 68
data store, 68
Datalog, 133
DBA (Data Base Administrator), 24
strumenti per il, 224
DBMS, 10
architettura dei sistemi relazionali, 251
catalogo, 223
funzionalità, 13
DDL (Data Definition Language), 11
deadlock, 274
decomposizione, 154
che preserva i dati, 154
che preserva le dipendenze, 156
con perdita di informazione, 139
definizione per ereditarietà, 47
denormalizzazione, 172
deposito dati, 68
derivazione di una dipendenza, 143
diagramma
di contesto, 69
di flusso dati, 68
di stato, 68, 72
per la descrizione dei dati, 68
dipendenze
anomale, 162
banali, 142
copertura canonica, 151
derivate, 142
derivazione di, 143
elementari, 152
funzionali, 85, 141
implicazione logica, 142
multivalore, 171
proiezione, 157
ridondanti, 151
distribuzione della base di dati, 23
DML (Data Manipulation Language), 11

ennupla, 57
entità, 31
debole, 56
proprietà, 31
tipo, 31
ereditarietà
singola, 48
multipla, 48
stretta, 48

forma normale
3FN, 165
4NF, 172
BCNF, 162
Boyce-Codd, 162

generatore
di applicazioni, 17
di rapporti, 19
gerarchia
di inclusione, 48
di inclusione multipla, 49
di tipi, 47
gestione delle interrogazioni
albero fisico, *vedi* piano di accesso
operatori fisici, 261
ottimizzazione fisica, 261
piano di accesso, 261
riscrittura algebrica, 258
gestore
dei metodi di accesso, 257
del buffer, 252
dell'affidabilità, 274
della concorrenza, 273
della memoria permanente, 252
delle interrogazioni, 258
delle strutture di memorizzazione, 253
giornale, 275
giunzione
metodo *index nested loop*, 267
metodo *merge join*, 267

metodo *nested loop*, 267

identità degli oggetti, 41
indice, 255
indipendenza
fisica, 15
logica, 15
integrazione di schemi di settore, 89
integrità dei dati, 19
interfaccia, 68
di un oggetto, 41
di un tipo oggetto, 41
istanza
di associazione, 33
valida, 103
valida di una relazione, 141

JDBC (Java Data Base Connectivity), 237

linguaggio
di interrogazione, 12
di quarta generazione, 18
per basi di dati, 11, 18
lock, 273
log, *vedi* giornale

malfunzionamento, 20
disastro, 21
fallimento di sistema, 21
fallimento di transazione, 21
metadati, 8
metodologia di modellazione, 38
modellazione, 29
aspetto linguistico astratto, 38
aspetto linguistico concreto, 38
aspetto ontologico, 30
aspetto pragmatico, 38
modello dei dati, 10
ad oggetti, 39
entità-relazione, 55
relazionale, 56, 101, 118

normalizzazione, 139, 161
in 3NF, 166
in BCNF, 163
algoritmo di sintesi, 167
algoritmo di analisi, 164

ODBC (Open Data Base Connectivity), 235
oggetto, 40
oggetto composto, 44
OID (Object Identifier), 41
operatore fisico
per eliminare duplicati, 262
per il raggruppamento, 270
per l'intersezione, 272
per l'ordinamento, 263
per l'unione, 272
per la differenza, 272
per la giunzione, 267
per la proiezione, 262
per la restrizione, 263
per la scansione, 262
organizzazione dei dati, 253
ad albero, 254
indice, 255
procedurale (hash), 254
scelta, 256
seriale (heap) e sequenziale, 254
statica o dinamica, 255
ottimizzazione fisica, 261

piano di accesso
esecuzione, 273
PL/SQL, 239
procedure memorizzate, 212
processo, 68
progettazione
concettuale, 82
fisica relazionale, 219
logica relazionale, 107
progettazione di basi di dati, 61
analisi dei requisiti, 62, 73
CASE, 93
concettuale, 62, 64, 82
fisica, 62, 65
logica, 62, 65
metodologia, 62
strumenti formali, 67
proiezione di dipendenze, 157

QBE, 199
quarta forma normale, 172
query language, 12

relazione, 57
universale, 140
report generator, 19
RID (Row Identifier), 253
ripristino dopo fallimento, 275
riscrittura algebrica, 126

schema
concettuale, 82
della base di dati, 8
di relazione, 101
di relazione universale, 140
esterno, 13, 223
fisico, 13
logico, 13
relazionale, 102
scheletro, 79
sicurezza dei dati, 22
sistema
informatico, 2
  di supporto alle decisioni, 8
  direzionale, 6
  operativo, 6
informativo, 1
per basi di dati, *vedi* DBMS
sottoclassi, 48
copertura, 48
disgiunte, 48
partizione, 48
rappresentazione grafica, 49
sottotipo, 47
SQL, 179
ALTER TABLE, 221
CHECK, 209
COMMIT WORK, 243
CREATE SCHEMA, 204
CREATE INDEX, 219
CREATE TABLE, 205
CREATE TRIGGER, 213
CREATE VIEW, 206
CROSS JOIN, 185
DELETE, 197
DIRTY READ, 247
DROP SCHEMA, 204
DROP TABLE, 206, 207
EXCEPT, 195
FOREIGN KEY, 210
GROUP BY, 193
INSERT, 197
INTERSECT, 195
JOIN, 185
LOCK TABLE, 248
NATURAL JOIN, 185
ORDER BY, 192
PRIMARY KEY, 210
READ COMMITTED, 247
READ UNCOMMITTED, 247
REPEATABLE READ, 247
SELECT, 181, 195
SERIALIZABLE, 248
UNION, 195
UNIQUE, 210
UPDATE, 197
4GL, 239
API, 234
cursor stability, 247
dynamic, 234
embedded, 228
giunzione esterna, 185
nei linguaggi di programmazione, 228
potere espressivo, 198
procedure memorizzate, 212
tipi di giunzione, 185
transazione, 242
vincoli
  interrelazionali, 210
  intrarelazionali, 209
  su attributi, 209
SQL-89, 179
SQL-92, 179
SQL2, 179
SQL:2003, 179
stallo, 274
state diagram, 68, 72
superchiave, 57, 104, 148

terza forma normale, 165

TID (Tuple Identifier), 253
tipo
ennupla, 57
enumerazione, 43
oggetto, 41
record, 42
sequenza, 43
transazione, 20, 242
livelli di isolamento, 246
realizzazione, 274
trigger, 213

UML (Unified Modeling Language), 99
universo del discorso, 29

valore nullo, 103
vincolo
d'integrità, 19, 36
d'integrità dinamico, 36
d'integrità statico, 36
di copertura, 48
di disgiunzione, 48
estensionale, 48
strutturale, 48