Capitolo 1

Basi di dati e sistemi di gestione di basi di dati

Le informazioni sono sempre state una risorsa vitale in qualsiasi organizzazione, indipendentemente dal grado di informatizzazione della stessa. Pensiamo ad esempio alle banche, che fanno della gestione delle informazioni il loro principale obiettivo ed esistono da ben prima dell'avvento dei calcolatori. Oggi, la crescente diffusione dell'informatica ed il fatto che molte organizzazioni offrano principalmente servizi (ad esempio, consulenze di vario genere, servizi di prenotazione o di assistenza remota) piuttosto che produrre beni materiali, hanno reso l'informazione una delle risorse maggiormente strategiche in ogni contesto. La diffusione di Internet e del Web ha ulteriormente accresciuto tale tendenza, in quanto ha reso più agevole lo scambio delle informazioni, abbattendo di fatto le barriere geografiche. In questo contesto, appare pertanto ovvio come un ruolo di primaria importanza sia ricoperto da tutti quegli strumenti in grado di acquisire, elaborare, trasmettere ed archiviare informazioni. Questi strumenti sono solitamente costituiti da risorse umane, da strumenti preposti all'archiviazione delle informazioni e da procedure manuali od automatizzate per il trattamento delle informazioni.

L'importanza della gestione dell'informazione ha motivato notevoli sforzi ed investimenti in ricerca e sviluppo con l'obiettivo di ottenere sistemi di gestione dell'informazione sempre più sofisticati e completi. Questi sforzi hanno portato allo sviluppo di tecniche in grado di minimizzare i tempi ed i costi di archiviazione ed elaborazione delle informazioni assicurando al contempo la correttezza e la qualità delle stesse.

L'obiettivo di questo libro è una trattazione quanto più completa possibile dei moderni sistemi per la gestione dell'informazione, discutendone sia gli aspetti legati all'uso sia gli aspetti architetturali. Questo primo capitolo, oltre a delineare brevemente l'evoluzione storica di tali sistemi, introduce i concetti fondamentali necessari alla trattazione seguente.

1.1 Concetti introduttivi

Come abbiamo ricordato nell'introduzione al capitolo, una delle principali esigenze di ogni organizzazione è quella di gestire e rendere disponibili le informazioni. Il sistema preposto a tale compito prende il nome di *sistema informativo*. Più precisamente, un sistema informativo è quella componente di un'organizzazione

che gestisce le informazioni di interesse, che si occupa cioè di produrre, acquisire, elaborare, conservare e distribuire le informazioni. Esso è costituito da strumenti, procedure e strutture, sia automatizzate sia manuali, e la sua definizione è del tutto indipendente dal grado di automazione in essere. Ad esempio, uno scaffale in cui vengono archiviate delle pratiche è una componente del sistema informativo, così come lo è un insieme di file in cui sono archiviate le informazioni anagrafiche dei dipendenti di un'azienda. Per sottolineare il fatto che un sistema informativo non presuppone necessariamente l'ausilio di un supporto informatico, denotiamo con il termine *sistema informatico* la parte del sistema informativo che gestisce l'informazione mediante l'ausilio di strumenti e tecnologie informatiche. Oggi, data la capillare diffusione dell'informatica, i termini sistema informativo e sistema informatico sono spesso utilizzati come sinonimi. Nel prosieguo della trattazione utilizzeremo anche noi tale convenzione a meno che sia necessaria un'esplicita distinzione.

Un sistema informativo deve quindi in primo luogo offrire degli strumenti per la rappresentazione dell'informazione, mediante una qualche codifica. Completeranno poi il sistema informativo una serie di programmi applicativi e di sistema che, operando su tale rappresentazione, realizzano tutte le funzioni necessarie alla gestione delle informazioni. Vediamo quindi di iniziare a chiarire meglio il concetto di informazione. Tutti noi abbiamo una percezione intuitiva del suo significato, ma se chiediamo a persone diverse di fornire una definizione di informazione è molto facile ottenere risposte differenti, proprio per la natura immateriale di tale concetto. Affidiamoci quindi al dizionario della lingua italiana, dove alla voce informazione troviamo: "*Tutto ciò che produce variazioni nel patrimonio conoscitivo di un soggetto detto percettore dell'informazione*" [BPLS91]. Notiamo come, in questa definizione, sia fondamentale il concetto di *utilità*. Se chiedo l'orario di un treno ad una persona che parla una lingua che non conosco, la risposta non è per me un'informazione in quanto non sono in grado di interpretarne il significato e quindi non produce variazioni nel mio patrimonio conoscitivo. Un sistema informativo, quindi, deve in primo luogo fornire una chiave di lettura mediante cui interpretare l'informazione che gestisce. Nei sistemi informatici, le informazioni sono rappresentate sotto forma di *dati*, dove per dato intendiamo una registrazione della descrizione di una qualsiasi caratteristica del dominio di interesse su un supporto che ne garantisca la conservazione e, mediante un insieme di simboli, ne garantisca la comprensibilità e la reperibilità. Consideriamo ad esempio un dato rappresentato dal numero 4; tale dato non fornisce in effetti alcuna informazione. Viceversa, dire che 4 è il numero di film noleggiati da Anna Rossi nell'ultimo mese fornisce un'informazione. Come appare evidente da questo semplice esempio, i dati hanno bisogno di un *contesto interpretativo* che permetta di estrarre da essi le informazioni di interesse per gli utenti. Uno degli obiettivi fondamentali di un sistema informativo è quindi quello di fornire un contesto interpretativo ai dati, in modo da consentire un accesso efficace alle informazioni da essi rappresentate. Con il termine *base di dati* denotiamo, quindi, una collezione di dati tra loro correlati, utilizzati per rappresentare le informazioni di interesse in un sistema informativo.

Un *sistema di gestione di basi di dati* (*DBMS – Data Base Management System*), spesso abbreviato nel seguito in *sistema di gestione dati* è, invece, un sistema software, centralizzato o distribuito, che fornisce gli strumenti necessari a gestire le informazioni. Una base di dati può quindi essere alternativamente definita come una collezione di dati gestita da un DBMS.

Le basi di dati ed i DBMS, a cui è dedicato il presente testo, costituiscono il "cuore" di ogni sistema informativo, in quanto sono gli strumenti mediante cui viene realizzata la gestione delle informazioni. Nei prossimi paragrafi, discuteremo le evoluzioni dei sistemi di gestione dati che hanno portato ai moderni DBMS, chiariremo meglio il ruolo del DBMS all'interno di un sistema informativo ed illustreremo i principali servizi che esso offre.

1.2 Dai sistemi operativi ai DBMS

La continua evoluzione dei sistemi di gestione dati, iniziata alla fine degli anni sessanta, ha ormai affermato la tecnologia dei DBMS come una componente essenziale nella realizzazione di qualsiasi sistema informativo. I primi sistemi informativi erano basati sull'uso di archivi separati, gestiti dal sistema operativo. A partire da questa tecnologia si è passati ad un approccio in cui i dati vengono organizzati in un unico insieme logicamente integrato, la base di dati appunto, gestito dal DBMS ed in grado di soddisfare il fabbisogno informativo di tutte le applicazioni.

Per meglio comprendere il notevole salto di qualità apportato dalla tecnologia dei DBMS rispetto ai precedenti approcci basati sul semplice uso del file system, consideriamo un esempio applicativo che ci servirà per discutere le problematiche tipiche di tali approcci.

Esempio 1.1 Supponiamo che una videoteca voglia mantenere informazioni relative ai propri clienti ed ai noleggi che questi hanno effettuato e supponiamo che le applicazioni usino direttamente i servizi del file system per la memorizzazione e l'accesso a tali dati. In base a tale approccio, i dati relativi ai clienti ed ai noleggi sono mantenuti in record memorizzati in vari file su memoria secondaria. Supponiamo che, in aggiunta ai file, esista un insieme di programmi applicativi tra cui:

- un programma di modifica della residenza di un dato cliente;
- un programma per l'inserimento di un nuovo noleggio;
- un programma per l'inserimento e la cancellazione di un cliente;
- un programma che stampa la lista di tutti i clienti della videoteca in ordine alfabetico. □

L'approccio discusso nell'esempio precedente presenta numerosi svantaggi, i principali dei quali sono discussi nel seguito.

Ridondanza ed inconsistenza nei dati. Poiché i file di dati ed i programmi sono creati in tempi diversi da progettisti software e programmatori diversi e non esiste una descrizione ad alto livello e centralizzata dei dati, una stessa informazione può essere replicata in file diversi; determinare se ed in che file una certa informazione è memorizzata è estremamente difficile se non impossibile. Con riferimento all'Esempio 1.1, il nome, il cognome ed il numero di tessera di un cliente possono essere memorizzati sia nel file che contiene le informazioni sui clienti sia nel file che contiene le informazioni sui noleggi. La presenza dello stesso dato in file diversi è detta *ridondanza* e può causare alti costi di memorizzazione ed inconsistenze nei dati. Ad esempio, se il numero di tessera di un cliente viene modificato, a seguito dello smarrimento della vecchia tessera, è necessario riportare tale cambiamento non solo nel file dei clienti ma anche in tutti i record relativi ai noleggi effettuati dal cliente in questione. Nei Capitoli 2, 5, 6 e 10 vedremo quali strumenti gli attuali DBMS forniscono per limitare inconsistenze e ridondanze dei dati, sia a livello di rappresentazione dei dati sia a livello di strumenti di progettazione della base di dati.

Difficoltà nell'accesso ai dati. La mancanza di una descrizione ad alto livello e centralizzata dei dati ne rende estremamente difficoltoso l'utilizzo al fine di rispondere a nuove esigenze applicative. Supponiamo, ad esempio, che la videoteca preveda un programma di fidelizzazione che qualifica i clienti come VIP a seguito dell'accumulo di un certo numero di punti. Supponiamo che sia necessario stampare periodicamente la lista dei clienti VIP ordinata in ordine alfabetico. Poiché tale richiesta non era stata prevista al momento della progettazione delle applicazioni illustrate nell'Esempio 1.1, non esiste un programma per la generazione della lista dei clienti VIP, mentre ne esiste uno che stampa la lista di tutti i clienti in ordine alfabetico. Sono quindi possibili due alternative:

1. stampare la lista di tutti i clienti ed estrarre manualmente da tale lista quella relativa ai clienti VIP;
2. richiedere che sia sviluppato un nuovo programma che estragga automaticamente la lista dei clienti VIP in ordine alfabetico.

Entrambe le soluzioni hanno delle evidenti controindicazioni. Ad esempio, supponiamo di optare per la seconda soluzione. Supponiamo che dopo qualche tempo sia necessario stampare la lista di tutti i clienti VIP ordinata per numero di punti accumulati. Ovviamente un programma che generi tale lista non è disponibile e pertanto ritorniamo alla situazione precedente. Un DBMS supera queste limitazioni, mettendo a disposizione dei linguaggi (vedi Capitoli 3 e 10) che facilitano l'accesso ai dati secondo modalità non necessariamente note a priori. Tali linguaggi possono poi essere integrati con generici linguaggi di programmazione per lo sviluppo di applicazioni complesse (vedi Capitolo 4) o con delle funzionalità reattive (vedi Capitolo 11) che consentono di eseguire delle operazioni sulla base di dati in modo automatico, al verificarsi di alcuni eventi. Inoltre, il DBMS mette a disposizione delle strutture ausiliarie di accesso (vedi Capitolo 7) che consentono di rispondere più efficientemente alle richieste di utenti ed applicazioni.

Problemi nell'accesso concorrente ai dati. Per migliorare le prestazioni, tutti i sistemi permettono di eseguire accessi concorrenti ai dati. L'interazione di modifiche concorrenti, se non opportunamente controllata, può causare inconsistenze dei dati, portando utenti ed applicazioni ad agire su dati non corretti. I sistemi operativi forniscono meccanismi per garantire la mutua esclusione nella modifica dei dati. Questi però non sono sufficienti in un ambiente di basi di dati dove devono essere messi a disposizione meccanismi più sofisticati per garantire la consistenza dei dati. A titolo di esempio, consideriamo ancora il dominio della videoteca e supponiamo che esista un'applicazione che, ad intervalli regolari, conti il numero di noleggi effettuati da ogni cliente della videoteca e generi un report con queste informazioni. Supponiamo che, mentre l'applicazione è in esecuzione, un cliente per cui il conteggio è già stato effettuato noleggi un video. In questo caso, l'esecuzione concorrente delle due applicazioni genera un conteggio non esatto. D'altro canto, non permettere l'esecuzione concorrente di applicazioni degraderebbe in modo non accettabile le prestazioni del sistema. Per risolvere questi problemi, i DBMS mettono a disposizione il concetto di *transazione*, argomento del Capitolo 8. Informalmente, una transazione è una porzione di programma applicativo a cui il DBMS assicura particolari proprietà durante la sua esecuzione che garantiscono, tra le altre cose, la consistenza dei dati in presenza di transazioni concorrenti, senza che il programmatore debba preoccuparsi della gestione di tali problematiche.

Problemi di protezione dei dati. Non tutti gli utenti devono poter accedere a tutti i dati presenti nel sistema. Ad esempio, un commesso della videoteca potrebbe essere autorizzato a conoscere la data in cui un cliente ha effettuato un noleggio ma non, per motivi di privacy, il titolo del film che il cliente ha noleggiato. È quindi necessario disporre di meccanismi che consentano un accesso selettivo ai dati: solo a determinati record (selezionati ad esempio in base al valore di alcuni campi) o solo a determinati campi di ogni record. Poiché i file system tipicamente non hanno meccanismi di controllo dell'accesso adeguati a supportare tali requisiti di protezione, questi controlli devono essere implementati a livello di programmi applicativi. In questo caso, però, è difficile verificare che tali controlli siano effettivamente e correttamente incorporati in tutti i programmi applicativi. Vedremo nel Capitolo 9 quali meccanismi offre un DBMS per gestire efficacemente il controllo dell'accesso senza dover ricorrere allo sviluppo di programmi ad-hoc.

Problemi di integrità dei dati. Per riflettere correttamente una certa realtà applicativa, è necessario che i dati rispettino determinate condizioni, note come *vincoli di integrità semantica*. Un esempio di vincolo è che la data di restituzione di un video non sia antecedente alla data di noleggio. Un insieme di dati è semanticamente corretto se verifica tutti i vincoli di integrità semantica ad esso associati. Se il sistema di gestione dati non consente la specifica e la verifica automatica di tali vincoli, è necessario implementarli come codice applicativo. L'aggiunta di nuovi vincoli o la modifica di un qualche vincolo rende però necessarie costose modifiche ai programmi applicativi, rendendo di fatto molto difficile assicurare l'integrità se-

mantica dei dati. Vedremo nel Capitolo 3 come i DBMS offrano adeguati strumenti per la specifica di vincoli e la loro verifica automatica.

1.3 Obiettivi e servizi di un DBMS

La tecnologia dei DBMS fornisce una soluzione efficace alle problematiche delineate nel paragrafo precedente. Il meccanismo fondamentale che rende possibile questo è una definizione centralizzata ed ad alto livello dei dati detta *schema* (o *schema logico*) della base di dati, condivisa da utenti ed applicazioni, a cui il DBMS assicura requisiti di affidabilità, efficienza, consistenza e protezione. Uno schema logico descrive il contenuto della base di dati tramite un formalismo ad alto livello che esula dai dettagli dell'effettiva implementazione fisica, detto *modello dei dati*. Questo meccanismo, e la definizione di una serie di opportuni linguaggi, rende i dati agevolmente disponibili ad una vasta gamma di utenti ed ambiti applicativi, anche secondo modalità non anticipate al momento della definizione e realizzazione della base di dati. La condivisione dei dati da parte di utenti ed applicazioni evita il problema della ridondanza ed inconsistenza dei dati. Inoltre, poter disporre di una visione centralizzata ed ad alto livello dei dati consente di introdurre un controllo centralizzato sugli stessi, minimizzando quindi i problemi connessi alla protezione e correttezza semantica dei dati.

Gli obiettivi precedententemente discussi sono resi concreti da un insieme di servizi – i principali dei quali sono elencati nella Tabella 1.1 – che un DBMS tipicamente offre e che facilitano l'utilizzo della base di dati e lo sviluppo di applicazioni che ad essa si interfacciano. Alcuni di questi servizi sono direttamente invocabili dagli utenti e dalle applicazioni; altri sono servizi interni al DBMS che permettono di assicurare efficienza e qualità nella gestione dei dati. La realizzazione dei vari servizi implica lo sviluppo di specifici linguaggi, tecniche, strutture dati ed algoritmi che costituiscono nel loro insieme il DBMS inteso come sistema software. Scopo dei capitoli successivi di questo libro è illustrare in dettaglio tali servizi sia per quanto riguarda il loro uso sia, per alcuni di essi, per quanto riguarda gli aspetti realizzativi.

Da un punto di vista architetturale, gli attuali DBMS adottano un'*architettura client-server*, in cui le funzionalità del sistema sono realizzate da due moduli distinti. Il modulo client, che di solito viene eseguito sull'elaboratore dell'utente finale, gestisce principalmente l'interazione tra l'utente ed il DBMS, mettendo a disposizione opportune interfacce per l'utilizzo dei servizi del DBMS e l'accesso ai dati. Il modulo server, invece, si occupa principalmente della memorizzazione e gestione dei dati e contiene un insieme di sotto-moduli che realizzano i servizi elencati nella Tabella 1.1. Ulteriori dettagli sulla struttura interna di un DBMS sono discussi nel Capitolo 7.

Servizio	**Descrizione**
Descrizione dei dati	Per specificare i dati da memorizzare nella base di dati
Manipolazione dei dati	Per: - accedere ai dati - inserire nuovi dati - modificare dati esistenti - cancellare dati esistenti
Controllo di integrità	Per evitare di memorizzare dati non corretti
Strutture di memorizzazione	Per rappresentare in memoria secondaria i costrutti del modello dei dati
Ottimizzazione di interrogazioni	Per determinare la strategia più efficiente per accedere ai dati
Protezione dei dati	Per proteggere i dati da accessi non autorizzati
Ripristino della base di dati	Per evitare che errori e malfunzionamenti: - determinino una base di dati inconsistente - provochino perdite di dati
Controllo della concorrenza	Per evitare che accessi concorrenti alla base di dati provochino inconsistenze dei dati

Tabella 1.1: Principali servizi offerti da un DBMS

1.4 Modelli dei dati

Abbiamo visto nel paragrafo precedente come una delle caratteristiche principali di un DBMS sia il poter disporre di una rappresentazione logica ed ad alto livello dei dati, espressa tramite un opportuno modello dei dati. È a questa rappresentazione logica che utenti ed applicazioni accedono senza dovere, nella maggior parte dei casi, occuparsi di come le strutture dati messe a disposizione dal modello siano effettivamente implementate sui supporti di memorizzazione di un elaboratore. Un modello dei dati deve quindi fornire dei costrutti per rappresentare le "entità" della realtà di interesse mediante un insieme di "concetti" che il DBMS è in grado di interpretare e gestire. Nei paragrafi successivi preciseremo meglio tali concetti ed introdurremo il modello relazionale, il modello dei dati più utilizzato negli attuali DBMS.

1.4.1 Concetti di base

Un modello dei dati è un *formalismo*, che consta di tre componenti fondamentali:

- un insieme di strutture dati;
- un linguaggio per specificare le strutture dati previste dal modello, per aggiornare tali strutture e per specificare vincoli su tali strutture;
- un linguaggio per manipolare i dati.

Iniziamo ad occuparci in questo paragrafo del primo aspetto, rimandando gli aspetti legati ai linguaggi al Paragrafo 1.6. Qualsiasi sia il modello dei dati prescelto, esso deve fornire opportune strutture per rappresentare i seguenti concetti:

- **Entità**. Insieme di "oggetti" della realtà applicativa di interesse, aventi caratteristiche comuni.
- **Istanza di entità**. Singolo oggetto della realtà applicativa di interesse, modellato da una certa entità.
- **Attributo**. Proprietà significativa di un'entità, ai fini della descrizione della realtà applicativa di interesse. Ogni entità è caratterizzata da uno o più attributi. Un attributo di un'entità assume uno o più valori per ciascuna delle istanze dell'entità, detti *valori dell'attributo*, in un insieme di possibili valori, detto *dominio dell'attributo*.
- **Associazione**. Corrispondenza tra un certo numero di entità. Anche le associazioni possono avere degli attributi che corrispondono a proprietà dell'associazione.
- **Istanza di associazione**. Corrispondenza tra le istanze di un certo numero di entità.

Esempio 1.2 Consideriamo la realtà applicativa relativa alla videoteca. Sono esempi di entità: `Cliente`, `Noleggio`, `Video` e `Film`. Ad esempio, l'entità `Cliente` rappresenta l'insieme dei clienti della videoteca. `anna rossi` e `pulp fiction` sono esempi di istanze delle entità `Cliente` e `Film`, rispettivamente. Il nome di un cliente, la data di restituzione del video noleggiato, il regista di un film sono esempi di proprietà che possono essere modellate come attributi delle entità `Cliente`, `Noleggio` e `Film`, rispettivamente. Il legame tra i clienti ed i film che consigliano è un esempio di associazione, che potrebbe avere un attributo `giudizio`, che modella il giudizio espresso dai clienti della videoteca sui film da loro consigliati. Infine, la relazione che lega l'istanza dell'entità `Film` relativa al film "Pulp fiction" di Quentin Tarantino con l'istanza dell'emtità `Cliente` relativa al cliente con numero di tessera 6610 è un esempio d'istanza dell'associazione `Consiglia`. □

Qualsiasi modello dei dati deve quindi rispondere a due domande fondamentali: *(i)* come rappresentare le entità ed i loro attributi; *(ii)* come rappresentare le associazioni ed i loro attributi. I modelli dei dati differiscono principalmente per come rappresentano le associazioni; a seconda dello specifico modello dei dati, la rappresentazione può avvenire tramite strutture, valori o puntatori. Per quanto riguarda invece la rappresentazione delle entità, la maggioranza dei modelli usa strutture concettualmente simili ai record, in cui ogni componente rappresenta un attributo.

1.4.2 Modello relazionale

Un esempio molto semplice di modello dei dati è il *modello relazionale* (discusso in dettaglio nel Capitolo 2), attualmente utilizzato dalla maggioranza dei DBMS in commercio. Il modello relazionale è basato su una singola struttura dati: la

Film

titolo	regista	anno	genere	valutaz
underground	emir kusturica	1995	drammatico	3.20
edward mani di forbice	tim burton	1990	fantastico	3.60
nightmare before christmas	tim burton	1993	animazione	4.00
ed wood	tim burton	1994	drammatico	4.00
mars attacks	tim burton	1996	fantascienza	3.00
il mistero di sleepy hollow	tim burton	1999	horror	3.50
big fish	tim burton	2003	fantastico	3.10
la sposa cadavere	tim burton	2005	animazione	3.50
la fabbrica di cioccolato	tim burton	2005	fantastico	4.00
io non ho paura	gabriele salvatores	2003	drammatico	3.50
nirvana	gabriele salvatores	1997	fantascienza	3.00
mediterraneo	gabriele salvatores	1991	commedia	3.80
pulp fiction	quentin tarantino	1994	thriller	3.50
le iene	quentin tarantino	1992	thriller	4.00

Video

colloc	titolo	regista	tipo
1111	underground	emir kusturica	v
1112	underground	emir kusturica	d
1113	big fish	tim burton	v
1114	big fish	tim burton	d
1115	edward mani di forbice	tim burton	d
1116	nightmare before christmas	tim burton	v
1117	nightmare before christmas	tim burton	d
1118	ed wood	tim burton	d
1119	mars attacks	tim burton	d
1120	il mistero di sleepy hollow	tim burton	d
1121	la sposa cadavere	tim burton	d
1122	la fabbrica di cioccolato	tim burton	d
1123	la fabbrica di cioccolato	tim burton	d
1124	io non ho paura	gabriele salvatores	d
1125	nirvana	gabriele salvatores	d
1126	mediterraneo	gabriele salvatores	d
1127	pulp fiction	quentin tarantino	v
1128	pulp fiction	quentin tarantino	d
1129	le iene	quentin tarantino	d

Figura 1.1: Base di dati relazionale relativa alla videoteca

relazione. Una relazione viene spesso rappresentata come una tabella con righe (dette *tuple*) e colonne contenenti dati di tipo specificato, come ad esempio interi e stringhe. Nel seguito useremo i termini relazione e tabella come sinonimi. Una tupla tipicamente rappresenta un'istanza dell'entità modellata dalla tabella a cui la tupla appartiene; una colonna (o attributo) di una tabella rappresenta, invece, un particolare attributo dell'entità modellata dalla tabella stessa.

Esempio 1.3 La Figura 1.1 mostra come le informazioni relative ai film e ai video di una videoteca possano essere organizzate in due tabelle, di nome `Film` e `Video`, rispettivamente. Ad esempio, per ciascun film a disposizione nella videoteca memorizziamo il titolo, il regista, l'anno in cui è stato girato, il genere e la valutazione della critica. Tali informazioni sono rappresentate da opportuni attributi della tabella `Film`. Ogni attributo corrisponde ad una colonna della tabella, mentre ogni riga rappresenta uno specifico film offerto dalla videoteca. □

Per quanto riguarda la rappresentazione delle associazioni, il modello relazionale utilizza una rappresentazione cosiddetta *per valore*, in quanto le associazioni non sono rappresentate esplicitamente (mediante costrutti quali puntatori) ma implicitamente, replicando alcune colonne di una tabella in quella con cui vogliamo stabilire un'associazione, oppure creando una tabella ad-hoc per modellare l'associazione contenente alcune colonne delle tabelle che vogliamo mettere in relazione. Una trattazione completa di tali concetti esula dagli scopi di questo capitolo e sarà oggetto del Capitolo 2. Nel seguito introdurremo comunque un semplice esempio per chiarire meglio i concetti discussi.

Esempio 1.4 Consideriamo ancora la base di dati della Figura 1.1. L'associazione tra un video ed il film che contiene è ottenuta replicando nella tabella `Video` gli attributi `titolo` e `regista` della tabella `Film`. In questo modo, è possibile sapere, dato un video, il genere del film corrispondente o la sua valutazione. Come sarà più chiaro nel Capitolo 2, questo tipo di modellazione presuppone che non esistano nella tabella `Film` due tuple distinte con gli stessi valori per gli attributi `titolo` e `regista`. □

Attributi con la proprietà di identificare univocamente le tuple di una relazione vengono chiamati *chiavi*, mentre i corrispondenti attributi nell'altra relazione, inseriti per modellare l'associazione tra le tuple delle due relazioni, prendono il nome di *chiavi esterne* (vedi Capitolo 2).

La corrispondenza tra i concetti illustrati nel Paragrafo 1.4.1 ed i costrutti messi a disposizione dal modello relazionale è riassunta nella Tabella 1.2.

1.4.3 Schemi ed istanze

Indipendentemente dal modello dei dati prescelto, in un DBMS distinguiamo solitamente tra la descrizione dei dati ed il contenuto effettivo della base di dati. La descrizione dei dati memorizzati in una base di dati, specificata tramite il modello

Concetto	**Costrutto relazionale**
Entità	Relazione
Istanza di entità	Tupla
Attributo	Attributo di una relazione
Associazione	Relazione/attributi di una relazione
Istanza di associazione	Tupla Valori uguali per chiavi e chiavi esterne

Tabella 1.2: Corrispondenza tra concetti generali di un modello dei dati e costrutti del modello relazionale

dei dati, è detta *schema della base di dati*, mentre l'insieme dei dati presenti in un dato momento in una base di dati è detto *istanza della base di dati*; tale insieme cambia molto spesso nel tempo dato che nuovi dati sono immessi di frequente, ed i dati presenti possono essere modificati o cancellati. Le modifiche allo schema sono invece molto meno frequenti anche se possibili. Lo schema fornisce pertanto una descrizione *intensionale* del contenuto della base di dati (in un certo qual senso indica il "tipo" dei dati contenuti nella base di dati, a prescindere dal contenuto vero e proprio). Il primo passo nello sviluppo di una base di dati è pertanto rappresentato dalla definizione dello schema della base di dati; successivamente vengono immessi i dati veri e propri che devono conformarsi alla definizione data dallo schema. Come vedremo nel Paragrafo 1.6, questa distinzione si riflette anche nei linguaggi messi a disposizione da un DBMS.

In una base di dati relazionale, lo schema della base di dati è costituito dall'insieme degli schemi delle relazioni in essa presenti. Lo schema di una relazione, a sua volta, è costituito dal nome della relazione e dal nome e dai domini dei suoi attributi. Con riferimento alla Figura 1.1, lo schema della base di dati è costituito dal nome delle due tabelle e dai nomi e domini dei loro attributi, mentre l'istanza è costituita dall'insieme di tuple contenute nelle due tabelle.

1.5 Livelli nella rappresentazione dei dati

Come è già stato evidenziato, uno degli scopi di un DBMS è fornire una rappresentazione ad alto livello di un insieme di dati, nascondendo i dettagli che riguardano la loro effettiva memorizzazione. Inoltre, una base di dati è spesso usata da gruppi di applicazioni e/o utenti per cui non tutti i dati presenti nella base di dati sono di interesse; a tali applicazioni e/o utenti viene spesso messa a disposizione una visione, detta *vista*, di una parte dello schema della base di dati. Una base di dati può pertanto essere osservata a tre diversi livelli di astrazione (vedi Figura 1.2):

- **Livello fisico**. È il livello più basso in cui viene definito lo *schema fisico* della base di dati, precisando *come* i dati sono effettivamente memorizzati tramite strutture di memorizzazione (file, record, ecc.).

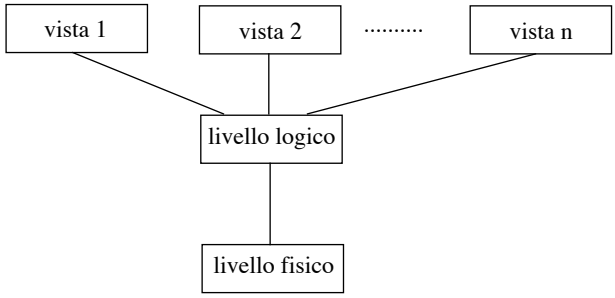

Figura 1.2: Livelli nella rappresentazione dei dati

- **Livello logico**. È il secondo livello di astrazione in cui viene decritto lo *schema logico*, cioè *quali* sono i dati memorizzati nella base di dati, eventuali associazioni tra di essi e vincoli di integrità semantica e di autorizzazione. L'intera base di dati è descritta tramite un numero limitato di strutture dati, relativamente semplici, che costituiscono il modello dei dati (relazioni nel caso del modello relazionale).

- **Livello esterno** o **livello delle viste**. È il livello di astrazione più alto; descrive una porzione dell'intero schema della base di dati; possono essere definite più viste di una stessa base di dati.

I primi due livelli sono sempre presenti, mentre il terzo è principalmente utilizzato in basi di dati il cui schema è di dimensione medio-grandi.

L'introduzione di questi tre livelli assicura alcune importanti proprietà ai dati, conosciute con il nome di *indipendenza fisica* e *logica*, che facilitano l'accesso ai dati e lo sviluppo di applicazioni. Indipendenza fisica significa che utenti ed applicazioni che accedono alla rappresentazione logica dei dati sono indipendenti da qualsiasi modifica a livello di rappresentazione fisica in quanto modifiche a tale livello non influenzano la rappresentazione dei dati al livello logico. Ad esempio, nel caso del modello relazionale, utenti ed applicazioni agiscono su tabelle. Tradurre ogni operazione effettuata sulle tabelle (ad esempio, la creazione di una tabella) nelle corrispondenti operazioni a livello fisico (ad esempio su file) è compito del DBMS e può avvenire in modo del tutto trasparente rispetto ad utenti ed applicazioni. Analogamente, la presenza delle viste permette di ottenere un certo grado di indipendenza logica, cioè di nascondere (entro certi limiti) modifiche alla rappresentazione dei dati al livello logico alle applicazioni/utenti che accedono alla rappresentazione esterna (tramite vista) dei dati. Ad esempio, nel contesto relazionale la modifica di una tabella non influisce su utenti ed applicazioni che operano su una vista che non contiene tale tabella.

1.6 Linguaggi di un DBMS

Oltre a fornire un modello dei dati, un DBMS mette a disposizione un insieme di linguaggi che permettono agli utenti di interagire con il DBMS per descrivere e manipolare i dati di interesse e specificare vincoli. Tra questi linguaggi i più rilevanti sono:

- **Linguaggio di definizione dei dati** (**DDL – Data Definition Language**). Il DDL consente di specificare ed aggiornare lo schema di una base di dati. In particolare, il DDL concretizza il modello dei dati fornendo la notazione che permette di specificarne le strutture dati. Esso deve supportare la specifica del nome della base di dati, come pure di tutte le unità logiche elementari della base di dati (ad esempio tabelle e colonne nel caso del modello relazionale) e di eventuali vincoli di integrità semantica e di autorizzazione. Deve inoltre prevedere comandi per l'aggiornamento delle strutture dati previste dal modello (ad esempio aggiunta/eliminazione di una nuova colonna in una tabella o modifica del dominio di un attributo nel caso del modello relazionale).
- **Linguaggio di manipolazione dei dati** (**DML – Data Manipulation Language**). Una base di dati, organizzata logicamente tramite il modello dei dati e definita tramite il DDL, è accessibile agli utenti ed alle applicazioni tramite il DML. Le operazioni fornite da questo linguaggio servono quindi per gestire le istanze della base di dati e sono fondamentalmente quattro:
  - inserimento, per l'immissione di nuovi dati;
  - ricerca, per il ritrovamento dei dati di interesse; un'operazione di ricerca è spesso detta *interrogazione* (o *query*) da cui il nome di *linguaggio di interrogazione* (o *query language*) per quella componente del DML che permette di esprimere le operazioni di ricerca;
  - cancellazione, per l'eliminazione di dati obsoleti;
  - modifica, per variare dati esistenti.
- **Linguaggio di definizione delle strutture di memorizzazione** (**SDL – Storage Definition Language**). La corrispondenza fra le strutture logiche, specificate nello schema della base di dati, e le strutture di memorizzazione deve essere opportunamente definita. Nella maggior parte dei DBMS attuali la definizione di tale corrispondenza è eseguita automaticamente dal DBMS stesso una volta che lo schema logico è definito. Tuttavia, l'utente esperto può influenzare le scelte operate dal DBMS richiedendo ad esempio l'allocazione di strutture ausiliarie di accesso (vedi Capitolo 7) per velocizzare determinati accessi ai dati. Tali richieste vengono effettuate tramite i comandi del linguaggio di definizione delle strutture di memorizzazione.

Una distinzione fondamentale riguardante i linguaggi di manipolazione dei dati è tra *linguaggi procedurali*, detti anche *operazionali*, e *linguaggi non procedurali*,

detti anche *dichiarativi*. Per grado di proceduralità intendiamo il grado di dettaglio con cui è necessario specificare i vari passi che il DBMS deve eseguire per effettuare le operazioni richieste. Nei linguaggi procedurali è responsabilità del programmatore specificare sia il risultato che vuole ottenere sia come ottenerlo. In sostanza, una volta effettuato l'accesso ad un'istanza di un'entità (ad esempio un record) della base di dati, il programmatore deve specificare a quale altra istanza accedere per effettuare l'operazione richiesta. I DBMS di prima generazione (vedi Paragrafo 1.7), come ad esempio i DBMS basati sul modello Codasyl, erano caratterizzati da linguaggi procedurali. Nei linguaggi non procedurali, invece, dobbiamo solo specificare quali caratteristiche devono avere i dati su cui vogliamo operare, tipicamente indicando condizioni sugli attributi, lasciando al DBMS la responsabilità di decidere come accedere effettivamente a tali dati. Ad esempio, se diciamo che siamo interessati a conoscere il nome del film contenuto nel video con codice di collocazione 1116, stiamo utilizzando un approccio dichiarativo in quanto non esplicitiamo come effettivamente reperire tale informazione nella base di dati. Il linguaggio SQL (vedi Capitolo 3), che è il linguaggio standard per la definizione e manipolazione di dati relazionali, è un esempio di linguaggio dichiarativo ed a questa caratteristica deve molta della sua fortuna. In effetti, l'evoluzione dei linguaggi per la manipolazione dei dati è stata caratterizzata dall'affermarsi di linguaggi dichiarativi. Il vantaggio di tali linguaggi è duplice: da una parte sono di facile utilizzo, anche per utenti poco esperti di informatica, in quanto con tali linguaggi è solo necessario dichiarare le caratteristiche che il risultato deve possedere senza dover specificare anche il procedimento operativo per ottenere tale risultato. D'altro canto, non specificare il modo operativo con cui ottenere il risultato consente al DBMS di applicare tutta una serie di strategie (alcune delle quali trattate nel Capitolo 7) per eseguire in modo ottimizzato l'operazione limitando il numero di accessi a disco necessari.

1.7 Evoluzione dei modelli dei dati

L'evoluzione dei DBMS è stata guidata dall'evoluzione dei modelli dei dati. Lo sviluppo di diversi modelli dei dati caratterizza infatti ogni nuova generazione di DBMS. La tendenza nello sviluppo di nuovi modelli dei dati è sempre stata di aumentare il potere espressivo del modello dei dati, rendendolo in grado di rappresentare sempre meglio ed in modo diretto la realtà di interesse alle più svariate applicazioni e, nel contempo, di rendere la rappresentazione dei dati attraverso il modello il più indipendente possibile da aspetti implementativi.

I primi DBMS, sviluppati negli anni '60, erano caratterizzati dal *modello gerarchico*, mentre successivamente sono comparsi DBMS caratterizzati dal *modello reticolare*, standardizzato dalla Conferenza sui Linguaggi per Sistemi di Dati (Codasyl). La generazione successiva, sviluppata durante gli anni '70, viene segnata dall'avvento dei DBMS basati sul modello relazionale, conosciuti come *RDBMS – Relational Database Management System*. La semplicità delle strutture di rappresentazione dati adottate dal modello relazionale ha nettamente differenziato questo

modello rispetto ai precedenti, permettendo lo sviluppo di linguaggi di definizione e manipolazione dei dati semplici da utilizzare. Il modello relazionale costituisce la base su cui è stato sviluppato il linguaggio standard SQL, oggi utilizzato nella maggior parte delle applicazioni di gestione dati. Inoltre, il modello relazionale, a differenza dei precedenti, è caratterizzato da una base matematica che ha dato notevoli impulsi a ricerche, anche teoriche, in ambito basi di dati. Oggi la tecnologia relazionale è alla base dei principali DBMS presenti sul mercato (quali Oracle, leader mondiale del mercato dei DBMS, IBM DB2, Microsoft SQL Server, Informix – recentemente acquisito da IBM, Sybase SQL Server, ed i DBMS open source MySQL e PostgreSQL).

Gli inizi degli anni '80 hanno visto inoltre l'affacciarsi di nuove applicazioni, rese possibili dalle innovazioni hardware, quali ad esempio le applicazioni CAD (*Computer-Aided Design*), CAM (*Computer-Aided Manufacturing*), CASE (*Computer-Aided Software Engineering*), GIS (*Geographical Information System*) ed applicazioni multimediali. Tali applicazioni sono caratterizzate dall'esigenza di rappresentare direttamente oggetti applicativi con strutture complesse. Una risposta a tali esigenze è costituita dai *DBMS orientati ad oggetti* (*OODBMS – Object-Oriented Database Management System*) che, come indica il nome, integrano la tecnologia dei DBMS con il paradigma di orientamento ad oggetti sviluppato nell'area dei linguaggi di programmazione e delle metodologie di progettazione del software. Questa linea di tendenza è stata caratterizzata da sviluppi industriali ed applicazioni significative, oltreché da uno sforzo di standardizzazione.

L'intensa attività di ricerca e sviluppo degli anni '80 ha evidenziato che, se da un lato determinate funzionalità come ad esempio la possibilità di modellare oggetti complessi e gerarchie di tipi, sono essenziali per un modello dei dati, dall'altro la compatibilità con il modello relazionale ed in particolare l'uso del linguaggio SQL sono requisiti altrettanto cruciali. Gli inizi degli anni '90 hanno visto pertanto la confluenza della tecnologia dei DBMS relazionali con la tecnologia dei DBMS orientati ad oggetti, il cui risultato sono i *DBMS relazionali ad oggetti* (*ORDBMS – Object-Relational Database Management System*), argomento del Capitolo 10. Questo tipo di DBMS, pur avendo un modello dei dati estremamente flessibile e ricco (che incorpora di fatto molti dei costrutti del modello ad oggetti), mantiene la stessa filosofia del modello relazionale e del linguaggio SQL con, ovviamente, le opportune estensioni.

Un'altra evoluzione particolarmente rilevante in ambito basi di dati è stata quella di dotare i DBMS della possibilità di definire *trigger*, cioè regole attive che effettuano azioni nel sistema automaticamente al verificarsi di determinati eventi (ad esempio inserimento di dati con particolari valori). I DBMS che forniscono tali funzionalità sono chiamati *DBMS attivi* e verranno trattati nel Capitolo 11.

Tra le varie tendenze evolutive ricordiamo inoltre i *sistemi di gestione dati deduttivi* ed i *sistemi di gestione dati "intelligenti"*. I primi integrano la tecnologia delle basi di dati con la programmazione logica. La principale caratteristica di tali sistemi è di fornire meccanismi di inferenza, basati su regole, che permettono di derivare informazioni aggiuntive dai dati memorizzati nella base di dati. Tali siste-

```
<elencoFilm>
   <film>
      <titolo>underground</titolo>
      <regista>emir kusturica</regista>
      <anno>1995</anno>
      <genere>drammatico</genere>
      <valutaz>3.20</valutaz>
   </film>
   <film>
      <titolo>edward mani di forbice</titolo>
      <regista>tim burton</regista>
      <anno>1990</anno>
      <genere>fantastico</genere>
      <valutaz>3.60</valutaz>
   </film>
   <film>
      <titolo>nightmare before christmas</titolo>
      <regista>tim burton</regista>
      <anno>1993</anno>
      <genere>animazione</genere>
      <valutaz>4.00</valutaz>
   </film>
...
</elencoFilm>
```

Figura 1.3: Documento XML corrispondente alla tabella `Film` della Figura 1.1

mi sono caratterizzati da fondamenti teorici abbastanza consolidati. Gli sviluppi industriali e le applicazioni sono stati però molto limitati e ristretti ad applicazioni di tipo scientifico. I secondi, invece, integrano la tecnologia delle basi di dati con vari paradigmi e tecniche sviluppate nell'area dell'intelligenza artificiale. Esempi tipici sono i sistemi basati sui modelli di rappresentazione della conoscenza.

Infine, l'inizio del nuovo millenio ha definitivamente visto l'affermarsi di Internet e del Web come strumenti principali di comunicazione e condivisione dell'informazione. In tale ambito, un ruolo di primaria importanza è giocato dal linguaggio *XML* (*eXtensible Markup Language*), un linguaggio sviluppato dal *World Wide Web Consortium – W3C*, che ormai costituisce uno standard per la rappresentazione e lo scambio di informazioni su Web. XML è un meta-linguaggio per la definizione di linguaggi di markup atti a modellare dati strutturati di differenti domini applicativi. XML fornisce un modo per definire tag e relazioni strutturali tra di essi, senza porre alcuna restrizione sui tag che possono essere definiti. Questa estensibilità consente di personalizzare l'insieme di tag in base al dominio applicativo ed alla semantica dei dati che vogliamo modellare. A titolo di esempio, la Figura 1.3 riporta un documento XML che modella le informazioni contenute nella relazione `Film` della Figura 1.1. Per ragioni di spazio, sono riportate solo le prime tre tuple contenute nella relazione. Come vediamo dalla Figura 1.3, un documento XML ha una strutturazione gerarchica ed i tag sono utilizzati per

rappresentare la semantica dei dati (nel nostro caso entità ed attributi).

XML, come è ovvio che sia, ha notevolmente influenzato gli ultimi sviluppi in ambito DBMS; gli approcci proposti per la gestione di dati XML possono essere classificati in due categorie principali: lo sviluppo di DBMS che hanno XML come modello nativo (*XML-Native DBMS*), come ad esempio il DBMS Tamino; estensioni ai DBMS relazionali o relazionali ad oggetti con funzionalità per la gestione di dati XML (*XML-Enabled DBMS*), quali ad esempio Oracle 10g. In tale contesto, una parte del nuovo standard SQL, SQL/XML, è dedicata alla gestione di dati XML nel contesto di un DBMS relazionale.

1.8 Utenti di un DBMS

L'uso di un DBMS nella gestione dati introduce diverse professionalità che, a vario titolo, si occupano di compiti connessi alla gestione della base di dati e del DBMS. Tra le principali professionalità, identifichiamo le seguenti figure:

- **Amministratore della base di dati** (**DBA – Database Administrator**). L'introduzione di questa tipologia professionale è conseguenza del fatto che oggi i DBMS sono strumenti sempre più complessi ed utilizzati da una vasta categoria di utenti. I principali compiti del DBA sono l'amministrazione ed il controllo della base di dati. In particolare, il DBA deve stabilire, in accordo alle politiche dell'organizzazione, le regole per l'utilizzo dei dati e le procedure che assicurino la protezione (vedi Capitolo 9) e l'integrità dei dati. Deve inoltre occuparsi delle opportune procedure di ripristino (vedi Capitolo 8) della base di dati, migliorare, ove possibile, l'efficienza (vedi Capitolo 7) del sistema e mantenere i contatti con gli utenti per determinare eventuali nuove esigenze.

- **Progettista di basi di dati**. Tale figura professionale deve provvedere, in base ai requisiti dell'organizzazione e delle varie applicazioni, alla progettazione della base di dati (molto spesso questa attività è affidata a consulenti specializzati). In particolare, è compito del progettista definire lo schema logico e fisico della base di dati, ed eventuali viste sugli stessi, in base ad un'attenta analisi dei requisiti informativi del dominio di interesse. Alla progettazione dello schema logico di una base di dati sono dedicati i Capitoli 5 e 6, mentre alcuni dettagli sulla progettazione fisica verranno forniti nel Capitolo 7.

- **Programmatore applicativo**. Il programmatore applicativo è preposto allo sviluppo dei programmi applicativi che operano sulla base di dati ed effettua il mantenimento di tali programmi. È da notare che, a seconda delle dimensioni dei programmi applicativi, di solito sono presenti più programmatori applicativi e che inoltre i programmatori che hanno inizialmente sviluppato i programmi non sono necessariamente gli stessi che eseguono la manutenzione. Allo sviluppo di applicazioni che si interfacciano ad una base di dati (dette anche applicazioni per basi di dati) è dedicato il Capitolo 4.

- **Progettista, sviluppatore di DBMS**. Un DBMS, essendo un sistema software, necessita di essere progettato, sviluppato e mantenuto così come una qualsiasi applicazione software. Naturalmente, la complessità del sistema necessita di adeguate professionalità per svolgere tale compito.

Per quanto riguarda i semplici utenti possiamo classificarli in due categorie: *utenti parametrici* ed *utenti occasionali*. I primi sono utenti che usano sistematicamente la base di dati, come parte integrante delle loro mansioni; l'uso che tali utenti fanno dei dati può essere anche complesso ma è fondamentalmente noto a priori. È quindi possibile sviluppare appositi programmi applicativi che supportino le funzioni necessarie a tali utenti. I secondi sono utenti le cui richieste non sono predicibili a priori e che tuttavia nella maggior parte dei casi fanno un uso molto semplice della base di dati. Per tali utenti sono spesso predisposte interfacce interattive molto semplici da utilizzare.

Note bibliografiche

Dato che una buona conoscenza dei DBMS è ormai indispensabile per qualsiasi professionista dell'informatica, esistono numerosi testi che trattano l'argomento. Tra i testi classicamente usati a livello universitario se ne possono citare molti. In particolare, il testo di Silberschatz et al. [SKS01] ed il testo di Elmasri e Navathe [EN03], recentemente tradotto in italiano [EN04, EN05], trattano l'argomento in modo approfondito, dando molto spazio agli aspetti più tecnologici e non eccedendo in quelli teorici. Un altro testo molto esauriente è quello di Ramakrishnan e Gehrke [RG02], di cui è disponibile anche la versione italiana [RG04]. Molto completo, anche relativamente agli aspetti architetturali, è il testo di Garcia-Molina, Ullman e Widom [GMUW02]. Per una trattazione più teorica segnaliamo il testo di Ullman [Ull90], che dà ampio spazio all'uso della logica nelle basi di dati ed alle tecniche di ottimizzazione per linguaggi logici per basi di dati ma nel contempo copre approfonditamente tutti gli argomenti classicamente trattati nei corsi di basi di dati, ed il testo di Abiteboul, Hull e Vianu [AHV95], che presenta una trattazione completa degli aspetti di base, trattando inoltre questioni di espressività e complessità relative ai linguaggi per basi di dati. Sono inoltre da segnalare i testi di Atzeni et al. [ACPT02, ACF$^+$03] che trattano in modo esauriente gli argomenti di base, offrendo inoltre una panoramica su argomenti avanzati quali l'accesso a basi di dati da Web e le basi di dati distribuite. Per una presentazione concisa ed essenziale, benché rigorosa, dei concetti fondamentali relativi ai sistemi di gestione dati relazionali suggeriamo invece il testo di Bressan e Catania [BC05]. Infine, per dettagli relativi al linguaggio XML rimandiamo il lettore a [XML].