Appunti del corso di basi di dati I

Riccardo Cereghino

17 marzo 2020

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

DBMS - Data Base Managament System

1.1 Sistema di gestione di basi di dati

I sistemi di gestione di basi di dati si pongono di risolvere i problemi legati alla gestione e salvataggio di dati, quali:

- **ridondanza ed inconsistenza dei dati**, ovvero la duplicazione dei dati su file multipli, che comporta anche un pericolo di inconsistenza;
- **difficoltà nell'accesso ai dati**, la mancanza di una descrizione di alto livello e centralizzata dei dati ne rende estremamente difficoltoso l'utilizzo al fine di rispondere a nuove esigenze applicative, a questo proposito i *DBMS* forniscono linguaggi per facilitare l'accesso ai dati, integrazione con i linguaggi di programmazione e funzionalità reattive;
- **problemi nell'accesso concorrente ai dati**, operazioni multiple sullo stesso dato possono causare problemi di concorrenza, i *DBMS* mettono a disposizione il concetto di *transazione*, che garantisce la consistenza dei dati in presenza di transazioni concorrenti;
- **problemi di integrità dei dati**, i *DBMS* forniscono la possibilità di stabilire dei vincoli al salvataggio ed alla modifica dei dati, preservandone l'integrità.

1.1.1 Obiettivi e servizi di un DBMS

I DBMS implementano una serie di servizi per offrire l'utilizzo della base di dati e lo sviluppo di applicazioni con cui si interfacciano, alcuni sono invocabili dagli utenti, altri sono utilizzati per il funzionamento interno.

I DBMS adottano un'*architettura client-server*, per cui le funzionalità del sistema sono realizzate da due moduli distinti: il **client** che gestisce l'interazione tra utente e DBMS ed il **server** che si occupa della memorizzazione e gestione dei dati.

Principali servizi offerti da un DBMS

- **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.

1.1.2 Modelli dei dati

Una delle caratteristiche principali di un DBMS è di poter disporre di una rappresentazione logica ad alto livello dei dati, i *modelli di dati* astraggono i dati presenti nel sistema per essere utilizzati più semplicemente da utenti ed applicazioni.

Concetti di base

Un *modello di dati* è un formalismo che racchiude tre componenti fondamentali:

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

Struttura dati Ad modello di dati corrisponde una struttura dati, composta da:

- **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 alle proprietà delle associazioni;
- **istanza di associazione:** corrispondenza tra le istanze di un certo numero di entità.

1.1.3 Modello relazionale

Il modello relazionale è basato su di una singola struttura dati, la relazione. Una relazione viene solitamente rappresentata come una tabella con righe (dette tuple) e colonne contenenti dati di tipo specificato.

Una tupla tipicamente rappresenta un'istanza dell'entità modellata dalla tabella a cui la tupla appartiene, mentre la colonna di una tabella rappresenta un particolare attributo dell'entità modellata dalla tabella stessa.

Attributi con la proprietà di identificare univocamente le tuple di una relazione vengono chiamati *chiavi*, mentre i corrispondenti attributi nell'altra relazione, prendono il nome di *chiavi esterne*.

1.1.4 Schemi ed istanze

In un DBMS individuiamo lo *schema della base di dati* e l' *istanza della base di dati*.

Lo schema fornisce una descrizione dei dati per tipo, l'istanza è l'insieme delle tuple contenute nella base di dati.

1.1.5 Livelli nella rappresentazione dei dati

Uno degli scopi di un DBMS è di fornire una rappresentazione ad alto livello di un insieme di dati nascondendone l'effettiva memorizzazione, per cui la base di dati fornisce tre livelli diversi di visualizzazione/astrazione:

- **livello fisico:** il livello più basso con cui viene definito lo schema fisico della base di dati, precisando come i dati sono effettivamente memorizzati tramite strutture di memorizzazione;

- **livello logico:** il secondo livello di astrazione in cui viene decritto lo *schema logico*, ovvero quali sono i dati memorizzati nella base di dati, eventuali associazione tra di essi ed eventuali vincoli di integrità semantica e di autorizzazione, l'intera base di dati è descritta tramite un numero limitato di strutture dati che costituiscono il modello dei dati;

- **livello esterno:** 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.

La presenza di questi livelli di astrazione assicura alcune importanti proprietà ai datim l' **indipendenza fisica** e l' **indipendenza logica**.

Il concetto di indipendenza fisica esprime la possibilità di interagire con il livello logico senza apportare modifiche al livello fisico.

Il concetto di indipendenza logica consente la possibilità di interagire con il livello esterno senza apportare modifiche al livello logico.

1.1.6 Linguaggi di un DBMS

I DBMS mettono 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 i linguaggi più rilevanti sono:

DDL - Data Definition Language Il linguaggio di definizione dei dati 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.

Il DDL deve supportare la specifica del nome della base di dati, come pure di tutte le unità logiche elementari della base di dati e di eventuali vincoli di integrità semantica e di autorizzazione prevedendo comandi per l'aggiornamento delle strutture dati previste dal modello.

Per quanto concerne i linguaggio di manipolazione dei dati si distingue tra *linguaggi procedurali (operazionali)* e *linguaggi non procedurali (dichiarativi)* per esempio **SQL**.

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 per gestire le istanze della base di dati e sono fondamentalmente quattro:

- **inserimento:** per l'immisione di nuovi dati;
- **ricerca:** per il ritrovamento dei dati in interesse, detta interrogazione (*query*);
- **cancellazione:** per l'eliminazione di dati obsoleti;
- **modifica:** per variare dati esistenti.

SDL - Storage Definition Language La corrispondenza tra le strutture logiche 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 può influenzare le scelte operate dal DBMS tramite i comandi del linguaggio di definizione delle strutture di memorizzazione.

Capitolo 2

Modello relazionale

2.1 Modello dei dati

Il modello relazionale è stato proposto da *E.F. CODD* nel 1970.

Le interrogazioni sulle relazioni possono essere espresse in due formalismi:

- **algebra relazionale:** in cui le interrogazioni sono espresse applicando operatori specializzati alle relazioni;
- **calcolo relazionale:** in cui le interrogazioni sono espresse per mezzo di formule logiche che devono essere verificate dalle tuple ottenute come risposta all'interrogazione.

2.1.1 Relazioni

Il concetto alla base del modello relazionale è la *relazione*, definita partendo dalla nozione di dominio, un dominio è un insieme anche infinito di valori.

Sia $\mathbb{D}$ l'insieme di tutti i domini, che assumeremo contenere i tipi:

- **int:** numeri interi;
- **real:** numeri reali;
- **string:** stringhe;
- **date:** date.

I valori dei domini costituiscono i valori atomici che popleranno la base di dati, a partire dai quali vengono costruite le *tuple* delle relazioni, effeuando un prodotto cartesiano di tali valori.

Definizione 1 (Prodotto cartesiano) *Siano $D_1, D_2, \ldots, D_k \in \mathbb{D}$* k *domini. Il prodotto cartesiano di tali domini, indicato con $D_1 \times D_2 \times \cdots \times D_k$ è definito come l'insieme:*

$$\{(v_1, v_2, \ldots, v_k) | v_1 \in \mathbb{D}_1, \ldots, v_k \in \mathbb{D}_k\}$$

Gli elementi appartenenti al prodotto cartesiano sono detti ***tuple****; il prodotto cartesiano di* k *domini ha grado* k*.*

Definizione 2 (Relazione) *Siano $D_1, D_2, \ldots, D_k \in \mathbb{D}$* k *domini. Una relazione su $D_1, D_2, \ldots, D_k \in \mathbb{D}$ è un sottoinsieme finito del prodotto cartesiano $D_1 \times D_2 \times \cdots \times D_k$.*

Nomenclatura

- **Relazione:** sottonsieme del prodotto cartesiano di k domini, ha grado k;
- **tupla:** ogni tupla di una relazione di grado k ha k componenti, uno per ogni dominio su cui è definita la relazione cui la tupla appartiene.
- **componente:** data una relazione R di grado k, una tupla $t \in \mathbb{R}$ ed un intero $i \in \{1, \ldots, k\}$, la notazione $t[i]$ denota la i-esima componente di t;
- **cardinalità:** esprime il numero di tuple appartenenti alla relazione, una relazione è sempre un insieme finito;
- **attributo:** un'alternativa alla formulazione del modello relazionale è di associare un nome, detto *nome di attributo* ad ogni componente delle tuple in una relazione.

Definizione 3 (Schema di relazione) *Siano R un nome di relazione, $\{A_1, \ldots, A_n\}$ un insieme di nomi di attributi, $dom : \{A_1, \ldots, A_n\} \to \mathbb{D}$ una funzione totale che associa ad ogni nome di attributo in $\{A_1, \ldots, A_n\}$ il corrispondente dominio.*

La coppia $(R(A_1, \ldots, A_n), dom)$ è uno schema di relazione. U_R denota l'insieme dei nomi di attributi di R cioè $\{\{A_1, \ldots, A_n\}\}$.

Definizione 4 (Schema di base di dati) *Siano $S_1, \ldots, S_n$ schemi di relazioni, con nomi di relazione diversi, $S = \{S_1, \ldots, S_n\}$ è detto schema di base di dati.*

Definizione 5 (Tupla e relazione) *Sia $R((A_1, \ldots, A_n), dom)$ uno schema di relazione.*

Una tupla t definita su R è un insieme di funzioni totali $f_1, \ldots, f_n$, dove $f_i : A_i \to dom)A_i), i = 1, \ldots, n$, associa all'attributo di nome A_i un valore del dominio di tale attributo. Una relazione definita su uno schema di relazione è un insieme finito di tuple definite su tale schema; tale relazione è anche detta istanza dello schema*.*

2.1.2 Valori nulli

Il valore nullo è un valore ammissibile per ogni dominio, rappresenta la mancanza di un valore di una tupla.

Denotiamo il valore nullo con il simbolo ?.

2.1.3 Chiavi

Una chiave di una relazione è un insieme di attributi che distingue fra loro le tuple della relazione.

Definizione 6 (Chiave e super-chiave) *Sia $R(A_1, \ldots, A_n)$ uno schema di relazione.*

Un insieme $X \subseteq U_R$ di attributu di R è chiave di R se verifica entrambe le seguenti proprietà:

1. *qualsiasi sia lo stato di R, non esistono due tuple distinte di R che abbiano lo stesso valore per tutti gli attributi in X;*

2. *nessun sottoinsieme proprio di X verifica la prima proprietà.*

Un insieme di attributi che verifica la prima proprietà, ma non la seconda è detto super-chiave *di R.*

Una relazione puà avere più di un insieme S di attributi che verificano le due proprietà, in tale caso si usa il termine *chiavi candidate* per indicare tutte queste chiavi.

Nel caso una relazione abbia più chiavi candidate, è possibile selezionare tra queste una *chiave primaria*, le chiavi restanti si dicono *chiavi alternative*.

Per fare riferimento ad una tuple è possibile usare qualunque chiave, ma è preferibile utilizzare la primaria dato che i DBMS ottimizza le operazioni.

Chiavi esterne

Nel modello relazionale è possibile specificare relazioni utilizzando chiavi esterne.

Le chiavi esterne permettono di collegare tra loro tuple di relazioni diverse e costituiscono un meccanismo per realizzare tali associazioni. L'approccio alla modellazione delle associazioni basato su chiavi esterne è detto *per valore*.

Definizione 7 (Chiave esterna) *Siano R ed R' due relazioni, sia $Y \subseteq U'_R$ una chiave per R' e sia $X \subseteq U_R$ un insieme di attributi di R tale che Y ed X contengano lo stesso numero di attributi e di dominio compatibile.*

X è una chiave esterna di R su R' sem qualsiasi siano gli stati di R ed R', per ogni tupla t di R esiste una tupla t' di R' tale che $t[X] = t'[Y]$. R viene detta relazione referente ed R' viene detta relazione riferita.

Integrità referenziale Il vincolo di integrità semantica che assicura il riferimento della referente su una referita si dice *vincolo di integrità referenziale.*

Può essere violata da inserimenti e modifiche nella relazione referente e da cancellazioni e modifiche nella relazione riferita.

2.2 Algebra relazionale

L'algebra relazionale è costituita da 5 operazioni per la manipolazione delle relazioni:

- proiezione;
- selezione;
- prodotto cartesiano;
- unione;
- differenza.

Ogni operazione ha come argomento una o più relazioni. Esistono operazioni addizionali che possono essere espresse in termini delle operazioni di base.

2.2.1 Operazioni di base

Sicchè facciamo riferimento alla notazione con nome, introduciamo l'operazione di ridenominazione.

Ridenominazione La ridenominazione di una relazione R rispetto ad una lista di coppie di nomi di attributi $(A_1, B_1), \ldots, (A_m, B_m)$, tale che $A_i \in U_R$ è un nome di attributo di R indicata con $\rho_{A_1,\ldots,A_m \leftarrow B_1,\ldots B_m}(R)$, ridenomina l'attributo di nome A_i con il nome B_i, $i = 1, \ldots, m$.

La ridenominazione è corretta se il nuovo schema di relazione per R ha attributi con nomi tutti distinti. La relazione ottenuta ha lo stesso grado della relazione R ed ha lo stesso contenuto. Gli attributi della relazione risultato sono $U_R \backslash \{A_1, \ldots, A_m\} \cup \{B_1, \ldots, B_m\}$.

Proiezione La proiezione di una relazione R su un insieme $A = \{A_1, \ldots, A_m\} \subseteq U_R$ di nomi di attributi di R, indicata con $\Pi_{A_1,\ldots,A_m} \subseteq (R)$, è una relazione di grado m le cui tuple hanno come attributi solo gli attributi specificati in A. Pertanto la proiezione genera un insieme T di tuple con m attributi. Sia $t = [A_1 : v_1, \ldots A_m : v_m]$ una tupla in T; t è tale che esiste una tupla t' in R tale che $t[A] = t'[A]$. Nella relazione risultato gli attributi compaiono secondo l'ordine specificato in A; è pertanto possibile specificare operazioni di proiezione che permutano gli attributi di una relazione. La proiezione permette di estrarre da una relazione solo alcune delle informazioni in essa contenute, eliminando gli attributi al cui valore non siano interessati. La relazione risultato, oltre ad avere

un grado inferiore a quello della relazione argomento, può anche avere cardinalità inferiore a quella della relazione argomento, poichè eliminando alcunti attributi possono venire generate delle tuple duplicate che compariranno una sola volta nella relazione risultato.

Selezione La selezione su una relazione R, dato un predicato F su R, indicata con $\sigma_F(R)$, genera una relazione che contiene tutte le tuple di R che verificano F. Un predicato F su R è un predicato semplice su R, oppure una combinazione booleana, ottenuta mediante gli operatori booleani $\wedge$, $\vee$, $\neg$, di predicati semplici su R. Un predicato semplice su R i cui domini devono essere compatibili; *op* è un operatore relazionale di confronto ed appartiene all'insieme $\{<, =, >, \geq, \leq\}$; v è un valore costante compatibile con il dominio A.

Se il grado della relazione operando R è k, la selezione $\sigma_F(R)$ genera un insieme T di tuple di grado k (sottoinsieme di R). Quindi lo scema (ed il grado) della relazione risultato sono uguali a quelli della relaione operando.

Prodotto cartesiano Il prodotto cartesiano di due relazioni R ed S, di grado rispettivamente k_1 e k_2, indicato con $R \times S$, è una relazione di grado $k_1 + k_2$ le cui tuple sono tutte le possibili tuple ch hanno:

- come prime k_1 componenti tuple di R;
- come ultime k_2 componenti tuple di S.

Il prodotto cartesiano può essere applicato solo se le due relazioni R ed S hanno schemi disgiunti. Nella relazione risultato i primi $k_1$attributi sono gli attributi della relazione R, gli ultimi k_2 attributi sono gli attributi della relazione S. Il prodotto cartesiano permette di costruire nuove tuple combinando le informazioni presenti nelle tuple delle relazioni argomento. Poichè ogni tupla di R viene combinata con ogni tupla di S, la cardinalità del risultato è il ptodotto delle cardinalità degli argomenti.

Unione L'unione delle relazioni R ed S, indicata con $R \cup S$, è l'insieme delle tuple che sono in R ed in S. L'unione delle relazioni può essere eseguita solo se le relazioni hanno lo stesso schema. La relazione risultato ha lo stesso schema delle relazioni argomento. Essendo l'unione un'operazione tra insiemi, le tuple duplicate vengono eliminate dal risultato.

Differenza La differenza delle relazioni R ed S, indicata con $R - S$, è l'insieme delle tuple che sono in R ma non in S. La differenza, come l'unione, di due relazioni può essere eseguita solo se le relazioni hanno lo stesso schema e produce una relazione con lo stesso schema. Se le relazioni hanno attributi con nomi diversi, si applica quanto già detto per l'unione.

2.2.2 Operazioni derivate

Le operazioni derivate sono definite attraverso le operazioni di base.

Intersezione L'intersezione di due relazioni R ed S è denotata con $R \cap S$ ed è espressa come $R - (R - S)$. L'intersezione di R ed S restituisce le tuple che sono sia in R che in S. L'intersezione può essere eseguita solo se le relazioni hanno lo stesso schema e produuce una relazione con lo stesso schema.

Join Il join (detto anche *theta-join*) di due relazioni R ed S sugli attributi A di R ed A' di S, indicato con $R \overset{\bowtie}{_{A\Theta A'}} S$, dove Θ è un operatore relazionale di confronto, è definito dall'espressione algebrica $\sigma_{A\Theta A'}(R \times S)$. Il join è perdanto un prodotto cartesiano seguito da una selezione; il predicato $A\Theta A'$ è detto predicato di join. Come per il prodotto cartesiano, gli schemi delle due relazioni argomento devono essere disgiunti e lo schema della relazione risultato è dato dalla loro unione.

Il join permette di collegare tuple di relazioni diverse e di attraversare le associazioni rappresentate nelle relazioni della base di dati mediante il meccanismo delle chiavi esterne.

Join naturale Rappresenta una semplificazione del join. Il join naturale di dure relazioni R ed S, denotato come $R \bowtie S$, è definito come:

sia $\{A_1, \ldots, A_k\} = U_R \cap U_S$ l'insieme dei nomi di attributi presenti sia nello schema di R sia in quello di S. Sia inoltre $\{I_1, \ldots, I_k\} = U_R \cap U_S$ l'insieme dei nomi di attributo unione dell'insieme dei nomi degli attributi di R e dell'insieme dei nomi degli attributi di S. Siano infine $\{B_1, \ldots, B_k\}$ nomi di attributo non appartenenti nè ad R nè ad S.

L'espressione che definisce il join naturale è:

$$R \bowtie S = \Pi_{I_1,\ldots,I_m}(\sigma_C(R \times (\rho_{A_1,\ldots,A_k \leftarrow B_1,\ldots,B_k}(S))))$$

dove C è un predicato della forma $A_1 = B_1 \wedge A_2 = B_2 \wedge \cdots \wedge A_k = B_k$.

Pertanto il join naturale esegue un join eguagliando gli attributi con lo stesso nome delle due relazioni argomento dell'operazione e poi elimina gli attributi duplicati dalla relazione risultato. Si noti che, affinchè l'operazione di join sia ben definita, gli attributi con nomi uguali nelle relazioni argomento dell'operazione devono avere domini compatibili.

Capitolo 3

Linguaggio SQL

Il linguaggio SQL è basato sui modell relazioneli, è di tipo dichiarativo.

3.1 Linguaggio di definizione dei dati

Le relazioni possono essere definite tramite il comando $CREATE$, modificate tramite il comando $ALTER$ e cancellate tramite il comando $DROP$. Questi comandi sono i principali del DDL.

3.1.1 Tipi di dato

Vi sono quattro diverse categorie di tipi di dato, divisi a loro volta per sottocategorie:

- tipi numerici;
- tipi carattere;
- tipi temporali;
- tipi definiti dall'utente.

Tipi numerici

Sono classificati in *tipi numerici esatti* e *tipi numerici approssimati*.

Tipi numerici esatti

- **INTEGER:** rappresenta i valori interi, la precisione varia a seconda della specifica implementazione di SQL;
- **SMALLINT:** rappresenta i valori interi, deve essere meno preciso di INTEGER;
- **BIGINT:** rappresenta i valori interi, deve essere più preciso di INTEGER;

- **NUMERIC:** tipo di dato caratterizzato da una precisione (numero totale di cifre) e da una scala (numero di cifre dopo la virgola decimale), la specifica ha forma $NUMERIC[(p[, s])]$, dove i predefiniti sono $p = 1$ e $s = 0$;
- **DECIMAL:** analogo a $NUMERIC$, differisce nella possibilità di inserire numeri meno precisi rispetto a quanto definito.

Tipi numerici

- **REAL:** rappresenta valori a singola precisione in virgola mobile, la precisione varia a seconda della specifica implementazione di SQL;
- **DOUBLE PRECISION:** rappresenta valori a singola precisione in virgola mobile (solitamente a doppia precisione), più preciso rispetto a $REAL$;
- **FLOAT:** permette di richiedere la precisione desiderata, ha forma $FLOAT[(p)]$, la precisione minima è 1, la precisione di default e la massima variano a seconda dell'implementazione di SQL.

Tipi di carattere

- **CHARACTER:** permette definire stringhe di caratteri di lunghezza predefinita nella forma $CHAR(n)$, la stringa viene completata con spazi vuoti, il valore di default di n è 1;
- **CHARACTER VARYING:** permette di definire stringhe di caratteri di una lunghezza massima predefinita, di forma $VARCHAR(n)$, differisce da $CHAR$ nella possibilità di non dover occupare tutto lo spazio allocato.

Tipi temporali

- **DATE:** rappresenta le date espresse come anno (4 cifre), mese (2 cifre) e giorno (2 cifre), sono disponibili diversi formati di rappresentazione;
- **TIME:** rappresenta i tempi espressi come ora (2 cifre), minuto (2 cifre) e secondo (2 cifre), la specifica ha forma $TIME[(p)]$ dove p è l'eventuale numero di cifre frazionarie cui si è interessati (6 cifre, microsecondo);
- **TIMESTAMP:** rappresenta una concatenazione dei tipi di dato $DATE$ e $TIME$, la specifica è $TIMESTAMP[(p)]$;
- **INTERVAL:** rappresenta una durata temporale in riferimento ad uno o più qualificatori tra $YEAR$, $MONTH$, DAY, $HOUR$, $MINUTE$ e $SECOND$, i valori di questo tipo sono rappresentati dalla parola chiave $INTERVAL$ seguita da una stringa che caratterizza la durata in termini di uno o più qualificatori.

Altri tipi definiti

- **BOOLEAN:** i cui valori sono $TRUE$, $FALSE$, $UNKOWN$;
- **BLOB:** *Binary Large OBject*;
- **CLOB:** *Character Large Object.*

3.1.2 Creazione di relazioni

In SQL la crezione di relazioni avviene in SQL tramite l'uso del comando `CREATE TABLE`

Specifica dello schema di una relazione

Listing 3.1: Schema di creazione tabelle

```
CREATE TABLE <nome relazione>
  (<specifica colonna> [,<specifica colonna>]*);
```

Dove:

- <**nome relazione**>**:** è il nome della relazione che viene creata;
- <**specifica colonna**>**:** è una specifica di colonna, il cui formato è:

  ```
  <nome colonna> <dominio> [DEFAULT] <valore di default>
  ```

  - `<nome colonna>` è il nome della colonna;
  - `<dominio>` è il dominio della colonna e deve essere uno dei tipi di dato di SQL;
  - la clausola `DEFAULT` specifica un valore di default per la colonna.

Vincoli di obbligatorietà, chiavi e chiavi esterne

SQL offre la possibilità di stabilire dei vincoli di obbligatorietà su colonne, chiavi e chiavi esterne; oltre a vincoli aggiuntivi specificabili sulle tuple della relazioni o colonne di queste tuple.

Obbligatorietà di colonne Per la specifica dell'obbligatorietà di una colonna è sufficiente aggiungere `NOT NULL` alla specifica della colonna.

Chiavi La specifica delle chiavi si effettua in SQL mediante le parole chiave `UNIQUE` e `PRIMARY KEY`.

La parola chiave `UNIQUE` garantisce che non esistano due tuple che condividano lo stesso valore.

La parola chiave `PRIMARY KEY` impone sia che i valori non siano nulli e che non esistano due tuple che condividano lo stesso valore.

Chiavi esterne La specifica di chiavi esterne avviene mediante la clausola `FOREIGN_KEY` ed il comando `CREATE TABLE`, la clausola è opzionale e ripetibile.

Listing 3.2: Sintassi per le chiavi esterne

```
FOREIGN_KEY (<lista nomi colonne>)
  REFERENCES <nome relazione>
  [ON DELETE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
  [ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
```

Definisce:

- una lista di una o più colonne che costituiscono una chiave esterna della relazione R;
- la relazione riferita R' di cui le suddette colonne sono chiave primaria;
- l'azione da eseguire se una tupla della relazione riferita è cancellata ed esistono tuple in R che fanno riferimento a tale tupla, l'opzione di default è `NO ACTION`, le opzioni sono:
  - `NO ACTION` la cancellazione di una tupla dalla relazione riferita è eseguita solo se non esiste alcuna tupla in R che fa riferimento a tale tupla;
  - `CASCADE` la cancellazione della dupla della relazione riferita implica la cancellazione di tutte le tuple di R che fanno riferimento a tale tupla;
  - `SET NULL` la cancellazione della tupla della relazione riferita implica che, in tutte le tuple di R che fanno riferimento a tale tupla, il valore della chiave esterna viene posto uguale al valore nullo;
  - `SET DEFAULT` la cancellazione della tupla della relazione riferita implica che, in tutte le tuple di R che fanno riferimento a tale tupla, il valore della chiave esterna viene posto uguale al valore di default specificato per le colonne che costituiscono la chiave esterna.
- l'azione da eseguire specificata nella clausola `ON UPDATE` se la chiave primaria di una tupla della relazione riferita viene modificata ed esistono tuple in R che fanno riferimento a tale tupla, le azioni possibili sono le stesse precedenti, eccezione per l'opzione `CASCADE`, la quale ha l'effetto di assegnare il nuovo valore della chiave primaria di quest'ultima.

3.1.3 Cancellazione e modifica di relazioni

La cancellazione di una relazione è eseguita con la sintassi:

Listing 3.3: Cancellazione di relazioni

```
DROP TABLE <nome relazione> {RESTRICT | CASCADE}
```

Si rende necessario specificare aun opzione:

- `RESTRICT` la relazione viene cancellata solo se non è riferita da altri elementi dello schema della base di dati;
- `CASCADE` elimina la relazione e tutti gli elementi dello schema che la riferiscono.

Listing 3.4: Modifica di relazioni

```
ALTER TABLE <nome relazione> <modifica>;

-- aggiunta di una colonna
ADD [COLUMN] <specifica colonna>

-- modifica di una colonna
ALTER [COLUMN] {SET DEFAULT <valore default> | DROP DEFAULT}

-- eliminazione di una colonna
DROP [COLUMN] <nome colonna> {RESTRICT | CASCADE}
```

3.2 Interrogazioni

3.2.1 SELECT

I comandi di interrogazione in SQL sono espressi tramite il comando `SELECT`, ha forma:

Listing 3.5: Forma del comando SELECT

```
SELECT {DISTINCT R_{i1}.C_1, ..., R_{in}.C_n*}
FROM R_1, ..., R_kWHEREF;
```

Dove $R_i(i = 1, \ldots, k)$ è un nome di relazione. La clausola `FROM` specifica pertanto le relazioni oggetto dell'interrogazione.

L'ordine in cui le colonne sono elencate nella clausola di priezione determina l'ordine da sinistra a destra secondo cui le colonne appaiono nella relazione risultato.

La clausola di qualificazione può contenere i connettivi booleani `AND,OR,NOT`.

3.2.2 Operatori e funzioni

SQL fornisce operatori aggiuntivi, oltre ai classici operatori relazionali di confronto.

Operatori di confronto

- `BETWEEN`, condizione su intervalli di valori, permette di trovare le tuple che contengono valori di una colonna in un intervallo specificato;

Listing 3.6: Esempio di BETWEEN

```
SELECT * FROM Film WHERE anno BETWEEN 1995 AND 2000;
```

- `IN`, ricerca di valori in un insieme, permette di determinare le tuple che contengono uno tra i valori di un insieme specificato;

Listing 3.7: Esempio di IN

```
SELECT * FROM Film WHERE genere IN ('horror','fantascienza')
```

- `LIKE`, condizioni di confronto per stringhe di caratteri, permette di eseguire alcune semplici operazioni di *pattern matching* su colonne di tipo stringa, il carattera speciale % denota una sequenza di caratteri arbitrari di lunghezza qualsiasi, il simbolo _ indica esattamente un carattere.

Listing 3.8: Esempio di LIKE, match di stringhe il cui terzo carattere è d

```
SELECT * FROM Film WHERE titolo LIKE '__d%';
```

Espressioni e funzioni

Le espressioni possono essere formulate applicanto funzioni alle colonne delle tuple, usate nei predicati o comparire nella clausola di proiezione delle interrogazioni nelle espressioni di `UPDATE`; sono applicate su colonne *virtuali*.

Espressioni e funzioni aritmetiche Oltre agli operatori aritmetici di base sono previste le funzioni:

- `ABS(n)` che ha come argomento un valore numerico n e ne calcola il valore assoluto;
- `MOD(n,b)` che ha come argomenti due valori interi n, b, e calcola il resto dell'intero della divisione di n per b.

Espressioni e funzioni per stringhe Il principale operatore tra stringhe è l'operatore di concatenazione ||, sono inoltre previste le funzioni:

- `LENGTH(str)`;
- `UPPER(str)`, `LOWER(str)`;

- `SUBSTR(str,m[,n])`;
- `TRIM [str1] FROM str2`

Espressioni e funzioni per date e tempi

- `CURRENT_DATE`;
- `CURRENT_TIME`;
- `CURRENT_TIMESTAMP`;
- `EXTRACT q FROM e`, estrae il campo corrispondente al qualificatore demprale q dall'espressione e.

3.2.3 Ordinamento del risultato di un'interrogazione

Il comando `SELECT` prevede la clausola addizionale `ORDER BY` per determinare le condizioni di ordinamento.

L'ordinamento non è limitato nè ad una sola colonna nè ad un ordine crescente.

Listing 3.9: Esempio di ORDER BY

```
SELECT colloc, dataNo1, dataRest FROM Noleggio
WHERE codCli = 6635
ORDER BY dataNo1 DESC, colloc;
```

3.2.4 Operazione di join

Tradizionalmente il join è espresso in SQL tramite un prodotto cartesiano a cui sono applicati uno o più predicatu du join, tra gli operatori troviamo:

- `CROSS JOIN`, corrisponde al prodotto cartesiano;
- `JOIN ON`, corrisponde al theta-join;
- `JOIN USING`, in cui viene richiesta l'uguaglianza dei valori delle colonne specificate nella clausola `USING`;
- `NATURAL JOIN`, corrisponde al join naturale, in cui viene richiesta l'uguaglianza delle colonne che hanno lo stesso nome nelle due relazioni.

Listing 3.10: Esempi di JOIN

```
SELECT titolo FROM Video, Noleggio
WHERE codCli = 6635 AND dataNo1 = DATE '15-Mar-2006'
  AND Video.colloc = Noleggio.colloc;

SELECT titolo FROM Video CROSS JOIN Noleggio
WHERE codCli = 6635 AND dataNo1 = DATE '15-Mar-2006'
  AND Video.colloc = Noleggio.colloc;

SELECT titolo
  FROM Video JOIN Noleggio
    ON Video.colloc = Noleggio.colloc;
WHERE codCli = 6635 AND dataNo1 = DATE '15-Mar-2006';

SELECT titolo FROM Video JOIN Noleggio USING (colloc)
WHERE codCli = 6635 AND dataNo1 = DATE '15-Mar-2006';

SELECT titolo FROM Video NATURAL JOIN Noleggio
WHERE codCli = 6635 AND dataNo1 = DATE '15-Mar-2006';
```

Outer join In `R JOIN S` non si ha traccia delle tuple di R che non corrispondono ad alcuna tupla di S. Per questo motivo SQL prevede un operatore di `OUTER JOIN` che aggiunge al risultato le tuple di R e/o s che non hanno partecipato al join, completandole con $NULL$.

L'operatore di join originario, per contrasto, viene anche detto `INNER JOIN`.

Esistono diverse varianti dell'outer join, che vengono specificate permettendo il corrispondente qualificatore all'operatore `OUTER JOIN`. Consideriamo `R OUTER JOIN S`:

- `FULL`, sia le tuple di R sia quelle di S che non partecipano al join vengono completate ed inserite nel risultato;
- `LEFT`, le tuple di R che non partecipano al join vengono completate ed inserite nel risultato;
- `RIGHT`, le tuple di S che non partecipano al join vengono completate ed inserite nel risultato.

Listing 3.11: Esempio di OTER JOIN

```
SELECT colloc, titolo, codCli
FROM Film NATURAL JOIN Video NATURAL LEFT OUTER JOIN Noleggio
WHERE regista = 'tim␣burton' AND genere = 'fantastico';
```

Capitolo 4

Memorizzazione dei dati ed elaborazione delle interrogazioni

Le prestazioni di un DBMS dipendono dall'efficienza delle strutture dati e dall'efficienza del sistema nell'operare su tali strutture. A livelloi fisico le tuple sono memorizzate in record di file su memoria secondaria ed una manipolazione efficiente verrà garantita dall'uso di opportune tecniche di elaborazione delle interrogazioni.

Per velocizzare la ricerca dei dati vengono in genere utilizzate particolari strutture di accesso, dette *indici*, che consentono di accedere direttamente ai record corrispondenti alle tuple con un certo valore per un attributo, senza scandire l'intero contenuto del file.

La scelta delle strutture di memorizzazione e di indicizzazione più efficienti dipende dal tipo di accessi che si eseguono sui dati; ogni DBMS ha le proprie strategie di implementazione di un modello di dati e le scelte dell'utente a questo proposito costituiscono la *progettazione fisica della base di dati*.

4.1 Architettura di un DBMS

Le basi di dati memorizzano in modo persistente grosse quantità di dati, memorizzate in memoria secondaria; per cui non possono essere elaborate direttamente dalla CPU, ma copiati in memoria principali in un *buffer*.

Quindi un DBMS è costituito da diverse componenti funzionali che includono:

- **gestore dei file:** gestisce l'allocazione dello spazio su disco e le strutture dati usate per rappresentare le informazioni memorizzate su disco;
- **gestore del buffer:** responsabile del trasferimento delle informazioni tra disco e memoria principale;

- **esecutore di interrogazioni:** responsabile dell'esecuzione delle richieste utente e costituito da:
  - **parser:** traduce i comandi del DDL e del DML in un formato interno (*parse tree*);
  - **selezionatore del piano:** stabilisce il modo più efficiente di processare una richiesta utente;
  - **esecutore del piano:** processa le richieste utente in accordo al piano di esecuzione selezionato;
- **gestore delle autorizzazioni:** controlla che gli utenti abbiano gli opportuni diritti di accesso ai dati;
- **gestore del ripristino:** assicura che la base di dati rimanga in uno stato consistente a fronte di cadute o malfunzionamenti del sistema;
- **gestore della concorrenza:** assicura che le esecuzioni concorrenti di processi procedano senza conflitti.

Quindi il DBMS memorizza, oltre ai dati utente, anche dati di sistema, strutture ausiliarie di accesso a tali dati e dati statistici, utilizzati dal selezionatore del piano per determinare la migliore strategia di esecuzione.

4.1.1 Cataloghi di sistema

Un DBMS sdescrive i dati che gestisce, incluse le informazioni sullo schema della base di dati e gli indici, tramite *meta-dati* che, sono memorizzate in relazioni speciali dette *cataloghi di sistema*, utilizzati dalle diverse componenti del DBMS.

Un aspetto elegante di un DBMS relazionale è che i cataloghi di sistema sono essi stessi relazioni. Anche i cataloghi di sistema permettono l'interrogazione e le techiche per l'implementazione e la gestione di relazioni.

Capitolo 5

Memorizzazione dei dati ed elaborazione delle interrogazioni

Le prestazioni di un DBMS dipendono dall'efficienza delle strutture dati e dall'efficienza del sistema nell'operare su tali strutture. A livelloi fisico le tuple sono memorizzate in record di file su memoria secondaria ed una manipolazione efficiente verrà garantita dall'uso di opportune tecniche di elaborazione delle interrogazioni.

Per velocizzare la ricerca dei dati vengono in genere utilizzate particolari strutture di accesso, dette *indici*, che consentono di accedere direttamente ai record corrispondenti alle tuple con un certo valore per un attributo, senza scandire l'intero contenuto del file.

La scelta delle strutture di memorizzazione e di indicizzazione più efficienti dipende dal tipo di accessi che si eseguono sui dati; ogni DBMS ha le proprie strategie di implementazione di un modello di dati e le scelte dell'utente a questo proposito costituiscono la *progettazione fisica della base di dati*.

5.1 Architettura di un DBMS

Le basi di dati memorizzano in modo persistente grosse quantità di dati, memorizzate in memoria secondaria; per cui non possono essere elaborate direttamente dalla CPU, ma copiati in memoria principali in un *buffer*.

Quindi un DBMS è costituito da diverse componenti funzionali che includono:

- **gestore dei file:** gestisce l'allocazione dello spazio su disco e le strutture dati usate per rappresentare le informazioni memorizzate su disco;
- **gestore del buffer:** responsabile del trasferimento delle informazioni tra disco e memoria principale;

- **esecutore di interrogazioni:** responsabile dell'esecuzione delle richieste utente e costituito da:
  - **parser:** traduce i comandi del DDL e del DML in un formato interno (*parse tree*);
  - **selezionatore del piano:** stabilisce il modo più efficiente di processare una richiesta utente;
  - **esecutore del piano:** processa le richieste utente in accordo al piano di esecuzione selezionato;
- **gestore delle autorizzazioni:** controlla che gli utenti abbiano gli opportuni diritti di accesso ai dati;
- **gestore del ripristino:** assicura che la base di dati rimanga in uno stato consistente a fronte di cadute o malfunzionamenti del sistema;
- **gestore della concorrenza:** assicura che le esecuzioni concorrenti di processi procedano senza conflitti.

Quindi il DBMS memorizza, oltre ai dati utente, anche dati di sistema, strutture ausiliarie di accesso a tali dati e dati statistici, utilizzati dal selezionatore del piano per determinare la migliore strategia di esecuzione.

5.1.1 Cataloghi di sistema

Un DBMS sdescrive i dati che gestisce, incluse le informazioni sullo schema della base di dati e gli indici, tramite *meta-dati* che, sono memorizzate in relazioni speciali dette *cataloghi di sistema*, utilizzati dalle diverse componenti del DBMS.

Un aspetto elegante di un DBMS relazionale è che i cataloghi di sistema sono essi stessi relazioni. Anche i cataloghi di sistema permettono l'interrogazione e le techiche per l'implementazione e la gestione di relazioni.

Capitolo 6

Protezione dei dati

Dato l'utilizzo, il problema della protezione dei dati si è reso estremamente critico. Pertanto i DBMS forniscono tecniche, strumenti e procedure di sicurezza. Gli obiettivi della protezione di dato sono tre: riservatezza, integrità e disponibilità; obiettivi che coinvolgono anche il sistema operativo.

6.1 Controllo dell'accesso

Il controllo dell'accesso regola le operazioni che è possibile compiere sui dati e le risorse in una base di dati; lo scome è di limitare e controllare le operazioni che gli utenti, già riconosciuto dal meccanismo di autenticazione, effettuano, prevenendo azioni accidentali o deliberate che potrebbero compromettere la correttezza e la sicurezza dei dati.

Nel controllo dell'accesso distinguiamo tre entità fondamentali: gli oggetti, i soggetti ed i privilegi.

I soggetti possono essere ulteriormente classificati in: utenti, gruppi e ruoli.

La concessione o meno di un determinato accesso deve rispecchiare le *politiche di sicurezza* dell'organizzazione.

Le autorizzazioni possono essere rappresentate mediante una tupla (s, o, p) dove s p il soggetto, o l'oggetto e p il privilegio.

Il controllo dell'accesso è effettuato mediante il meccanismo di *constrollo dell'accesso*, basato su un *modello di controllo dell'accesso*, (*reference monitor*) il cui compito è di intercettare ogni comando inviato al DBMS e stabilire, tramite l'analisi delle autorizzazioni, se il soggetto richiedente puà essere autorizzato o meno a compiere l'azione richiesta.