Appunti del corso di basi di dati I

Riccardo Cereghino

10 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* $f_i : A_i \rightarrow dom\ )A_i), i = 1, \ldots, n$*, 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.

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.*