Basi di Dati - Appunti

Cristian Di Pietratonio

1 novembre 2014

Contents

1 Introduzione

I sistemi di gestione di basi di dati (DBMS) sono strumenti software per la gestione di grandi masse di dati. Prima dell'avvento dei database, ogni programma aveva il suo file privato, organizzato *sequenzialmente* e la gestione dei dati era affidata al filesystem. Ciò causava problemi di:

- **ridondanza**: diversi files venivano replicati, dovendo essere condivisi con più applicazioni;
- **inconsistenza**: se un'informazione veniva aggiornata, tale aggiornamento poteva riguardare solo una copia del dato;
- **dipendenza dei dati**: i dati venivano strutturati dalle applicazioni, in base al loro utilizzo.

Per ovviare a questi problemi si iniziò a progettare le **basi di dati**, le quali videro una grande svolta quando alcuni ingegneri dell'IBM introdussero negli anni '70 il *modello relazionale*.

1.1 Le basi di dati

Le caratteristiche di una base di dati sono:

- **multiuso e integrazione**: la stessa base di dati può essere utilizzata da diverse applicazioni con diversi scopi;
- **indipendenza e controllo centralizzato**: i dati non sono gestiti dalle applicazioni ma da un software dedicato, il quale ne gestisce anche le regole di accesso.

I vantaggi che ne derivano sono la minima ridondanza, integrità dei dati e sicurezza.

1.1.1 Integrità

I dati devono rispettare dei vincoli che esistono nella realtà di interesse. Consideriamo ad esempio il database di InfoStud della Sapienza:

- uno studente risiede in *una sola* città (**dipendenze funzionali**);
- la matricola *identifica univocamente* uno studente (**vincoli di chiave**);
- un voto è un intero positivo tra 18 e 30 (**vicoli di dominio**).

1.1.2 Sicurezza

I dati devono essere protetti da accessi non autorizzati. Il **DBA** (*Database Administrator*) deve considerare:

- valore corrente per *accessi autorizzati*;
- valore corrente per accessi *non autorizzati*;
- *chi* può accedere *a quali* dati e in *quale modalità*;
- definire regole di accesso ed effetti relativi a una violazione.

1.2 Transazione

Definizione 1.1
Si definisce **transazione** una sequenza di operazioni che costituiscono un'unica operazione logica.

Esempio 1.1
"Trasferire €1000 da c/c1 al c/c2"

1. cerca c/c1;
2. modifica *saldo* in $saldo - 1000$;
3. cerca c/c2;

4. modifica *saldo* in $saldo + 1000$;

Una transazione deve essere eseguita completamente (*committed*) o non deve essere eseguita affatto (*rolled back*).

1.3 Ripristino

Può capitare che, a causa di un malfunzionamento del sistema (ad esempio sbalzi di tensione), la base di dati si trovi con delle informazioni corrotte. In questi casi, per ripristinare il valore corretto dei dati, si hanno due possibili modi di agire:

- sfruttare il **transaction log**, che contiene i dettagli di tutte le transazioni, tra cui valori precedenti e successivi alla modifica;
- ripristinare l'ultimo **dump** effettuato. Il dump è una copia periodica del database.

1.4 Compiti del DBA

Il Database Administrator ha vari compiti, tra i quali ricadono le scelte di progettazione della base di dati. Esse implicano la definizione di:

- schema logico
- schema fisico
- sottoschema o viste

inoltre al DBA spetta l'onere di mantenere il sistema.

2 Il modello relazionale

Il modello attualmente più diffuso e con il quale si organizzano i database è chiamato **modello relazionale**. Tale nome è dovuto al fatto che esso è basato sul concetto matematico di *relazione*.

Definizione 2.1
Dati n domini D_1, D_2, ..., D_n non necessariamente distinti, una relazione r su D_1, D_2, ..., D_n è un sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times \ldots \times D_n$.

Ogni elemento di r è una n-pla $(d_1, d_2, \ldots, d_n)$ $\forall$ $d_i \in D_i$

2.1 Relazioni e tabelle

Una relazione r può essere rappresentata mediante una tabella: le righe corrispondono agli elementi della relazione (n-ple), le colonne corrispondono ai domini $D_1 \ldots D_n$.

Esempio 2.1
La seguente tabella rappresenta una relazione con due 4-ple sui domini *String*, *String*, *Int* e *Real*.

String	String	Int	Real
Paolo	Rossi	2	26,5
Mario	Biachi	10	28,7

A questo punto è poco chiaro cosa rappresenti tale relazione. Bisogna stabilire metodo per fornire un'interpretazione standard che rispecchi la realtà di interesse a cui essa si riferisce:a questo fine diamo nomi alle colonne e alla tabella stessa. La precedente tabella diverrà quindi:

Nome	Cognome	Esami	Media
Paolo	Rossi	2	26,5
Mario	Biachi	10	28,7

Ora risulta più evidente che la realtà di interesse sono gli studenti di una data università. Daremo quindi il nome *Studenti* a tale tabella.

Il nome dato ad ogni colonna viene chiamato **attributo**.

Definizione 2.2
Uno *schema di relazione* è un insieme di attributi.

Definizione 2.3
Uno *schema di base di dati relazionale* è un insieme $\{R_1, R_2, \ldots, R_n\}$ di schemi di relazione.

Definizione 2.4
Una *base di dati relazionale* con schema $\{R_1, R_2, \ldots, R_n\}$ è un insieme $\{r_1, r_2, \ldots, r_n\}$ dove r_i è *un'istanza* di relazione con schema R_i.

3 Algebra relazionale

Data una base di dati, dobbiamo essere in grado di *interrogarla*. Interrogare una base di dati significa accedere a determinate informazioni tramite uno specifico linguaggio di interrogazione. Le specifiche essenziali di tale linguaggio sono formalizzate nell'**algebra relazionale**; essa consiste di un insieme di *operatori* che possono essere applicati a una (operatore unario) o due (operatore binario) istanze di relazione e *restituiscono* un'istanza di relazione. L'algebra relazionale è un **linguaggio procedurale**: l'interrogazione consiste in un'espressione in cui compaiono *operatori* e *istanze di relazione* della base di dati.

3.1 Proiezione

La *proiezione* è un operatore che consente di effettuare un "taglio verticale" su una relazione, cioè di selezionare dolo alcune colonne (attributi). Viene indicata con il simbolo:

$$\pi_{A_1,A_2,\ldots,A_k}(r)$$

e seleziona le colonne di r che corrispondono agli attributi A_1, A_2, ..., A_k.

Esempio 3.1
Si consideri la seguente relazione *Cliente*:

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Interroghiamo la base di dati con l'operatore di proiezione nel seguente modo: $\pi_{Nome}(Cliente)$. Verrà restituita la seguente istanza:

Nome
Rossi
Bianchi
Verdi

Da notare come siano stati eliminati i valori doppi poiché viene restituito un *insieme*.

3.2 Selezione

L'operatore di *selezione* consente di effettuare un "taglio orizzontale" su una relazione, cioè di selezionare solo le righe (tuple) che soddisfano una data condizione. Si denota con il simbolo

$$\sigma_C(r)$$

dove C è detta condizione di selezione.
La *condizione di selezione* è un'espressione booleana in cui i termini semplici sono del tipo

$$A\Theta B$$
oppure
$$A\Theta\text{`}a\text{'}$$

dove:

- Θ è un operatore di confronto $\Theta \in \{<, =, >, \leq, \geq\}$;
- A e B sono due attributi con lo stesso dominio, $dom(A) = dom(B)$;
- $a \in dom(A)$

Esempio 3.2
Si riprenda in considerazione la seguente tabella:

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Vogliamo ottenere i dati dei clienti che risiedono a Roma. La *query* si traduce in: $\sigma_{Città='Roma'}(Cliente)$. L'istanza di relazione che ci viene restituita è la seguente:

Nome	C#	Città
Rossi	C1	Roma
Bianchi	C3	Roma
Verdi	C4	Roma

Esempio 3.3
Facendo riferimento alla tabella dell'esempio precendente, vogliamo ottenere i dati dei clienti che si chiamano "Rossi" e risiedono a Roma. La *query* si traduce in: $\sigma_{Città='Roma' \wedge Nome='Rossi'}(Cliente)$. L'istanza di relazione che ci viene restituita è la seguente:

Nome	C#	Città
Rossi	C1	Roma

3.3 Unione

L'operatore di *unione* consente di costruire una relazione contenente tutte le tuple che appartengono ad almeno uno dei due operandi. Si denota con $r_1 \cup r_2$. L'operazione di unione può essere applicata solo ad operandi **union compatibili**, cioè tali che:

- hanno lo stesso numero di attributi;
- gli attributi corrispondenti sono definiti sullo stesso dominio.

3.4 Differenza

L'operatore di *differenza* consente di costruire una relazione contenente tutte le tuple del primo operando che non appartengono al secondo operando. L'operatore è applicabile solo a operandi *union compatibili*.

3.5 Intersezione

L'operatore di *intersezione* consente di costruire una relazione contenente tutte le tuple che appartengono ad entrambi gli operandi. Si denota con $r_1 \cap r_2 = (r_1 - (r_1 - r_2))$.

3.6 Prodotto cartesiano

L'operatore *prodotto cartesiano* consente di costruire una relazione contenente tutte le tuple che si ottengono concatenando una tupla del primo operando con una tupla del secondo operando. Si denota con $r_1 \times r_2$.

Esempio 3.4
Consideriamo le seguenti relazioni *Cliente* e *Ordine*:

Nome	C#	Città
Rossi	C1	Roma
Rossi	C2	Milano

C#	A#	Num
C1	A1	100
C2	A2	200
C1	A2	200

Il prodotto cartesiano tra queste due relazioni da come risultato la relazione che chiameremo per convenienza *Risultato*:

Nome	C#	Città	C#	A#	Num
Rossi	C1	Roma	C1	A2	100
Rossi	C1	Roma	C2	A2	200
Rossi	C1	Roma	C1	A2	200
Rossi	C2	Milano	C1	A1	100
Rossi	C2	Milano	C2	A2	200
Rossi	C2	Milano	C1	A2	200

Esempio 3.5
Riprendendo la relazione *Risultato* dall'esempio precendente, vogliamo effettuare una query che richieda i "dati dei clienti e dei loro ordini":

$$\pi_{Nome,\ Cliente.C\#,\ Città,\ A\#,\ Num}(\sigma_{Cliente.C\# \,=\, Ordine.C\#}(Cliente \times Ordine))$$

La query viene eseguita in quest'ordine:

1. Viene effettuato il prodotto cartesiano $Cliente \times Ordine$;
2. dal prodotto cartesiano vengono selezionate le tuple dove i corrispondenti attributi $C\#$ hanno lo stesso valore;
3. sulla relazione ricavata dalla selezione (punto 2) viene operata una proiezione per prendere le colonne volute, che andranno a formare la relazione che sarà il risultato finale della query.

Nome	C#	Città	A#	Num
Rossi	C1	Roma	A2	100
Rossi	C1	Roma	A2	200
Rossi	C2	Milano	A2	200

3.7 Join naturale

L'operatore *join naturale* consente di selezionare le tuple del prodotto cartesiano dei due operandi che soddisfano la condizione

$$(R_1.A_1 = R_2.A_1) \wedge (R_1.A_2 = R_2.A_2) \wedge \ldots \wedge (R_1.A_k = R_2.A_k)$$

Dove R_1 e R_2 sono i nomi delle due relazioni operando e $A_1 \ldots A_k$ sono gli attributi comuni.
Il join naturale è definito come

$$r_1 \rhd\lhd r_2 = \pi_{XY}(\sigma_C(r_1 \times r_2))$$

dove

- $C = (R_1.A_1 = R_2.A_1) \wedge (R_1.A_2 = R_2.A_2) \wedge \ldots \wedge (R_1.A_k = R_2.A_k)$;
- X sono gli attributi di r_1;
- Y sono gli attributi di r_2 che non sono in r_1;

3.8 Theta Join

Il $\Theta-Join$ consente di selezionare le tuple del prodotto cartesiano di due operandi che soddisfano una condizione del tipo $A\ \Theta\ B$, dove:

- Θ è un operatore di confronto $\Theta \in \{<, =, >, \leq, \geq\}$;
- A è un attributo dello schema del primo operando e B uno del secondo;
- $dom(A) = dom(B)$;

4 Progettazione di una base di dati

4.1 Anomalie

Si immagini di dover progettare una base di dati relazionale contenente i dati degli studenti e dei corsi di un'Università. La soluzione più immediata è di creare un'unica relazione

$$Università(Matr, Nome, Città, Prov, C\#, Titolo, Docente, C\_laurea, Data, Voto)$$

in cui una tupla $(m, n, c, p, C, t, D, l, d, v)$ rappresenta il fatto che uno studente con matricola m e nome n, residente nella città c che si trova in provincia di p, ha sostenuto l'esame del corso, con codice C e titolo t, tenuto dal docente D, del corso di laurea l in data d riportando il voto v.

Adottando questa soluzione si avrebbero un certo numero di inconvenienti che vanno sotto il nome di **anomalia**. Un anomalia è essenzialmente un comportamento *inaspettato e indesiderato* da parte della base di dati, generato in risposta ad un'operazione. Le anomalie più comuni (spiegate relativamente all'esempio) sono:

- **anomalie di inserimento**: non si possono inserire i dati di uno studente se non ha sostenuto almeno un esame;
- **anomalie di cancellazione**: se si cancellano i dati di un corso (perché il corso è stato disattivato) e c'è uno studente che ha sostenuto solo l'esame relativo a quel corso, perdo le informazioni sullo studente;
- **anomalie di aggiornamento**: se devo modificare il docente di un corso devo farlo per ogni tupla in cui compare il corso;
- **ridondanza** dei dati: le informazioni anagrafiche di uno studente sono ripetute per ogni esame sostenuto dallo studente.

Queste anomalie sono dovute al fatto che si sono rappresentati in un'unica relazione più *concetti*: il concetto "Studente", "Corso" ed "Esame". Rappresentando i tre concetti in tre relazioni distinte

$$Studente(Matr, Nome, Città, Prov)$$
$$Corso(C\#, Titolo, Docente, C\_laurea)$$
$$Esame(Matr, C\#, Data, Voto)$$

tali anomalie vengono eliminate. Tuttavia è possibile riscontrare il permanere di simili anomalie concernenti le città:per eliminare queste ultime anomalie che hanno la stessa origine di quelle viste precedentemente, posso utilizzare lo *schema* seguente:

$$Studente(Matr, Nome, Città)$$
$$Comune(Città, Prov)$$
$$Corso(C\#, Titolo, Docente, C\_laurea)$$
$$Esame(Matr, C\#, Data, Voto)$$

4.2 La Terza Forma Normale

Si è constatato che ci sono schemi migliori di altri. Esistono dunque *regole e proprietà formali* che ci permettono di costruire un *buono* schema?
Se si analizzano le anomalie nella relazione $Università$ si nota che sono legate al fatto che

- $Voto$ e $Data$ sono *determinati univocamente* da $Matr$ e $C\#$;
- i dati di uno studente sono *determinati univocamente* da $Matr$;
- i dati di un corso sono *determinati univocamente* da $C\#$.

Il concetto di "determina univocamente" è colto dal concetto formale di **dipendenza funzionale**.

Definizione 4.1
Dato uno schema di relazione R, una *dipendenza funzionale* su R è una coppia ordinata di sottoinsiemi non vuoti X, $Y \in R$ e viene denotata come $X \to Y$.

Proposizione 4.1
Un'istanza r di R *soddisfa* la dipendenza funzionale $X \to Y$ se per ogni coppia di tuple t_1, $t_2 \in r$ si ha che se $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$.

In simboli:

$$(\ \forall (t_1, t_2) \in r \ t.c.\ t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]) \Rightarrow (X \to Y \text{ soddisfa } r).$$

Sia F un insieme di dipendenze funzionali su R ed r un'istanza di R. Se r soddisfa tutte le dipendenze in F, diciamo che r è un'istanza *legale* di R.

Definizione 4.2
La *chiusura di F*, denotata con F^+, è l'insieme di dipendenze funzionali che sono soddisfatte da ogni istanza legale di R.

Banalmente si ha che $F \subseteq F^+$.

Definizione 4.3
Dato uno schema di relazione R, un insieme di dipendenze funzionali F su R e un sottoinsieme K di R, diciamo che K è una **chiave** per R se:

- $K \to R \in F^+$
- $\forall K' \subset K,\ K' \to R \notin F^+$.

In pratica una chiave è il minimo sottoinsieme di R che determina univocamente il valore dei restanti attributi di R (banalmente la chiave determina se stessa e i suoi sottoinsiemi) tale che, se prendessimo un suo sottoinsieme, esso non sarebbe chiave.
In futuro denoteremo con $A_1, A_2, \ldots, A_n$ un insieme di attributi, con X e Y sottoinsiemi di R, con XY l'insieme $X \cup Y$.

Esempio 4.1
Considerando il precendente schema $Università$, possiamo osservare che un'istanza di *Università* per rispecchiare la realtà di interesse deve soddisfare le seguenti dipendenze funzionali:

- $Matr,\ C\# \to Nome,\ Città,\ Prov,\ Titolo,\ Docente,\ C\_laurea,\ Data,\ Voto$;
- $C\# \to Titolo,\ Docente,\ C\_laurea$
- $Matr \to Nome\, Città,\ Prov$
- $Città \to Prov$

Quindi $\{Matr, C\#\}$ costituisce una *chiave* per Università.

Ci sono attributi che **dipendono parzialmente** dalla chiave. Nell'esempio [4.1] $\{Titolo$, $Docente$, $C\_laurea\}$ dipendono funzionalmente da $C\#$; altri attributi invece **dipendono transitivamente** dalla chiave, ad esempio $Prov$ dipende funzionalmente da $Matr$ in quanto $Matr \to Città \to Prov$.
Prendendo lo schema finale, vediamo che in nessuno schema di relazione ci sono attributi che dipendono parzialmente né transitivamente dalla chiave.

Proposizione 4.2
Uno schema di relazione in cui non ci sono sono attributi che dipendono parzialmente né transitivamente dalla chiave è detto in **Terza Forma Normale** (3NF).

Formalizziamo i concetti appena introdotti:

Definizione 4.4
Dati uno schema di relazione R e un insieme di dipendenze funzionali F su R diciamo che:

- un attributo $A \in R$ è **primo** se appartiene ad una chiave di R;
- un sottoinsieme $X \subset R$ è una **superchiave** se contiene una chiave di R.

Nell'esempio [4.1], considerando la relazione *Università*, $Matr$ è primo mentre $\{Matr, C\#, Nome\}$ è una superchiave.

Definizione 4.5
Siano R uno schema di relazione ed F un insieme di dipendenze funzionali su R.

- $X \to A \in F^+$ è una dipendenza parziale su R se A non è primo e X è contenuto propriamente in una chiave di R.
- $X \to A \in F^+$ è una dipendenza transitiva su R se A non è primo e $\forall K \subset R$ si ha che X non è contenuto propriamente in K e $K - X \neq \emptyset$.

Proposizione 4.3
Siano R uno schema di relazione ed F un insieme di dipendenze funzionali su R. R è in 3NF se, $\forall (X \to A) \in F^+$ t.c. $A \notin X$, si ha che A è primo oppure X è una superchiave.

Teorema 4.1
Siano R uno schema di relazione e F un insieme di dipendenze funzionali su R. Uno schema R è in 3NF se e solo se non esistono né dipendenze parziali né dipendenze transitive in R.

Dimostrazione. La parte *solo se* deriva banalmente dalla definizione [4.5].
Parte *se.* Supponiamo per assurdo che R non sia in 3NF nonostante non ci siano dipendenze parziali o transitive; in tal caso esiste una dipendenza funzionale $X \to A \in F^+$ t.c. A non è primo e X non è una superchiave. Poiché X non è una superchiave due casi (mutuamente esclusivi) sono possibili:

- o per ogni chiave K di R, X non è contenuto propriamente in K e $K - X \neq \emptyset$; in tal caso $X \to A$ è una dipendenza transitiva su R (contraddizione)
- oppure esiste una chiave K di R t.c. $X \subset K$; in tal caso $X \to A$ è una dipendenza parziale su R (contraddizione). □

Un obbiettivo da tenere presente quando si progetta una base di dati è quello di produrre uno schema in cui ogni relazione sia in 3NF. Nella fase di progettazione concettuale si individuano i concetti che devono essere rappresentati nella base di dati. Se questo lavoro di individuazione è fatto accuratamente lo schema relazionale che può essere derivato in modo automatico con opportune regole, è in 3NF. Se tuttavia dopo tale processo ci trovassimo a produrre uno schema che non è in 3NF dovremmo procedere ad una fase di decomposizione.

4.3 Decomposizione di schemi di relazione

Uno schema che non è in 3NF può essere decomposto in più modi in un insieme di schemi in 3NF. Sia $R = ABC$ con l'insieme di dipendenze funzionali $F = \{A \to B, B \to C\}$. R non è in 3NF per la presenza in F^+ della dipendenza transitiva $B \to C$, ma può essere decomposto in:

$$\begin{gathered} R_1 = AB \text{ con } F_1 = \{A \to B\} \\ R_2 = BC \text{ con } F_2 = \{B \to C\} \\ \text{oppure} \\ R_1 = AB \text{ con } F_1 = \{A \to B\} \\ R_2 = AC \text{ con } F_2 = \{A \to C\} \end{gathered}$$

Entrambi gli schemi sono in 3NF, tuttavia la seconda soluzione non è soddisfacente. Infatti, si consideri l'istanza della *base di dati* costituita dalle due istanze legali di R_1 e R_2:

A	B
a_1	b_1
a_2	b_1

A	C
a_1	c_1
a_2	c_2

L'istanza di R che si può ricostruire da questa tramite *join naturale* è

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

non è un'istanza legale di R, in quanto non soddisfa la dipendenza funzionale $B \to C$.

Esempio 4.2
Si consideri la relazione $Studente = \{Matr, Com, Prov\}$, con $F = \{Matr \to Com,\ Matr \to Prov,\ Com \to Prov\}$; essa è in 3FN. Una decomposizione possibile è la seguente:

$$R_1 = \{Matr, Com\} \text{ con } F_1 = \{Matr \to Com\}$$
$$R_2 = \{Matr, Prov\} \text{ con } F_2 = \{Matr \to Prov\}$$

entrabe sono in 3FN; si cosiderino le seguenti istanze di R_1 e R_2

Matr	Com
O_1	Marino
O_2	Marino

Matr	Prov
O_1	Parma
O_2	Latina

e si provi a fare il join naturale per riottenere lo schema di partenza. Ecco il risultato:

Matr	Com	Prov
O_1	Marino	Parma
O_2	Marino	Latina

chiaramente si perde la dipendenza funzionale $Com \to Prov$.

Una istanza di una relazione contiene i dati di una certa realtà che lo schema della base di dati intende rappresentare e si assumono come riferimenti veritieri. Pertanto quando si decompone uno schema si vuole che ogni sua istanza sia ricostruibile da un'istanza dello schema ottenuto dalla decomposizione.

Proposizione 4.4
Una decomposizione di uno schema di relazione deve avere i seguenti requisiti:

- deve preservare le dipendenze funzionali che valgono su ogni istanza legale dello schema originario;
- deve permettere di ricostruire mediante join naturale ogni istanza legale dello schema originario.