Capitolo 3 (Algebra relazionale)

Richiami di linguaggi per basi di dati

Operazioni sullo schema:

- DDL: data definition language

Operazioni sui dati:

- DML: data manipulation language
  - Interrogazione (”query”)
  - Aggiornamento (inserimento, modifica, cancellazione)

Linguaggi di interrogazione per basi di dati

Dichiarativi:

- Specificano le proprietà del risultato (”che cosa”)

Procedurali:

- Specificano le modalità di generazione del risultato (”come”)

Linguaggi di interrogazione

- L’**algebra relazionale** è un linguaggio procedurale
- Il **calcolo relazionale** è teoricamente dichiarativo (non è mai stato implementato)
- SQL è un linguaggio **parzialmente dichiarativo (semidichiarativo)**, implementa alcune cose dell’algebra relazionale e altre del calcolo relazionale
- QBE (Query by Example) è un linguaggio **dichiarativo**, permette di interrogare la base di dati specificando un esempio del risultato

Algebra relazionale

Consiste in un linguaggio procedurale, basato su concetti di tipo algebrico . E' costituito da un insieme di operatori definiti su realazioni e che producono come risultato una relazione

Possono anche essere composti poiché il risultato di un'operazione può essere usato come operando di un'operazione successiva (veste procedurale)

In quanto relazioni, gli operatori sono anche insiemi e possono essere sottoposti agli operatori di insiemistica con qualche vincolo in più sull'applicabilità

Grado di una relazione: numero di attributi

Cardinalità: numero di ennuple

Operatori:

- Unione
- Intersezione
- Differenza
- Ridenominazione (operatore monadico che prende cioè in input un solo operatore)
- Selezione (operatore monadico che permette di raggruppare le ennuple sotto una stessa proprietà)
- Proiezione
- Join (join naturale, prodotto cartesiano, theta-join)

Operatori insiemistici

Le relazioni sono insiemi e i risultati devono essere relazioni

Quindi è possibile applicare **unione**, **intersezione** e **differenza** alle relazioni, ma solo se sono definite sugli stessi attributi (devono essere un insieme di tuple **omogenee**):

💡 **Union compatibility:**
Due relazioni $R_1(A_1A_2...A_n)$ e $R_2(B_1B_2...B_3)$ sono **union compatible** (or **type compatible**) se hanno lo stesso grado e se $dom(A_i) = dom(B_i)$ con $1 \leq i \leq n$

Quindi entrambe le relazioni hanno lo stesso numero di attributi e ogni coppia corrispondente di attributi ha lo stesso dominio

Essa è una proprietà richiesta per poter applicare gli operatori insiemistici

(dagli appunti: le relazioni devono avere lo stesso schema)

Applicazioni:

- L'**unione** di due relazioni r_1 e r_2 definite sullo stesso insieme di attributi X e indicato con $r_1 \cup r_2$ sarà una relazione definita sugli stessi attributi delle relazioni operande e conterrà le ennuple che appartengono a r_1 oppure a r_2, oppure di entrambe
- L'**intersezione** di $r_1(X)$ e $r_2(X)$ e indicata con $r_1 \cap r_2$ sarà una relazione definita sugli stessi attributi delle relazioni operande e avrà soltanto le ennuple appartenenti sia a r_1 che a r_2
- La **differenza di** $r_1(X)$ e r_2 e indicata con $r_1 - r_2$ sarà una relazione definita sugli stessi attributi delle relazioni operande e conterrà le ennuple di r_1 da cui vengono tolte quelle in comune a r_2

Ridenominazione

💡 La **ridenominazione** cambia il nome dell'attributo lasciando inalterato il contenuto delle relazioni

Modifica lo schema (non agisce sull'istanza), prende una sola relazione (essendo *monadico*) e rinomina determinati attributi

Non modifichiamo propriamente lo schema, ma creiamo come risultato una relazione avente lo stesso schema eccetto gli attributi rinominati

Spesso utilizzato per garantire union compatibility nelle operazioni insiemistiche

Sintassi:

$$\rho_{B_1 B_2 \ldots B_k \leftarrow A_1 A_2 \ldots A_k}(r)$$

Selezione

> 💡 La **selezione** è definita su un operando e produce come risultato una porzione dell'operando. La selezione produce un sottoinsieme delle tuple, su tutti gli attributi

- Anche la selezione è un operatore *monadico*

Sintassi:

$$SEL_{Condizione}(Operando) \text{ oppure } \sigma_{Condizione}(Operando)$$

La **condizione** è un'espressione booleana (come quelle dei vincoli di ennupla)

Formula proposizionale F: Data una relazione $r(X)$, una formula proposizionale F su X è un a formula ottenuta combinando, con i connettivi $\neg, \vee, \wedge$ condizioni atomiche del tipo $A\theta B$ oppure $A\theta c$ dove:

- θ è un operatore di confronto ($=, \neq, >, <, \geq, \leq$)
- A e B sono attributi in X su cui valori il confronto θ abbia senso
- c è una costante "compatibile" con il dominio di A

Completiamo la definizione di selezione come segue:

- $\sigma_F(r)$

dove r è una relazione e F è una formula proposizionale

Semantica:

- Produce una relazione contente le ennuple dell’operando che soddisfano la condizione

La selezione non ha effetti collaterali, ma produce una copia

Proiezione

> 💡 La **proiezione** proietta le ennuple su un insieme di attributi, se queste ennuple sono uguali verranno compattate nella stessa ennupla

E’ simile alla proiezione sugli assi cartesiani

Lo schema della relazione risultante può differire da quello dell’operatore

Anche la proiezione è un operatore *monadico*

- Produce un risultato che ha parte degli attributi dell’operando
- Contiene ennuple a cui contribuiscono tutte le ennuple dell’operando
- In pratica vengono eliminati eventuali duplicati delle ennuple, indotti dalla eliminazione di alcuni attributi

Sintassi:

$$PROJ_{ListaAttributi}(Operando) \text{ oppure } \pi_{ListaAttributi}(Operando)$$

Semantica:

- Il risultato contiene le ennuple dell’operando ristrette agli attributi nella lista

Cardinalità delle proiezioni

Una proiezione contiene al più tante ennuple quante ne contiene l’operando

Se X è una superchiave di R, allora $PROJ_X(R)$ contiene esattamente tante ennuple quante ne contiene R (facendo la proiezione su una

superchiave viene prodotta una relazione avente tutte le ennuple dell’operando)

Selezione e proiezione

Combinando selezione e proiezione possiamo estrarre interessanti informazioni da una relazione

Sono operatori “ortogonali”:

- Selezione:
  - Decomposizione orizzontale
- Proiezione
  - Decomposizione verticale

Combinare informazioni da più relazioni

Sebbene la combinazione di selezione e proiezione consenta di estrarre informazioni interessanti da una relazione, essa non consente di correlare informazioni presenti in relazioni diverse

Occorre un operatore specifico

Join

L’operatore Join permette di correlare dati in relazioni diverse

L’operatore Join è pesante dal punto di vista computazionale (per questo è opportuno evitare frammentazioni di dati durante lo sviluppo di una base di dati)

Esistono due versioni dell’operatore:

- Join naturale
- Theta-join

Join naturale

💡 I **join naturale** è un operatore che correla dati in relazioni diverse, sulla base di valori uguali in attributi con lo stesso nome (attributi comuni)

Produce un risultato:

- Sull'unione degli attributi degli operandi
- Con ciascuna ennupla costruita a partire da una ennupla di ognuno degli operandi:
  - Le sue ennuple sono il risultato della combinazione di ennuple degli operandi con valori coincidenti su attributi comuni

La prima tupla del join deriva dalla combinazione della prima tupla della relazione r_1 e della seconda tupla della relazione r_2

Il **join naturale** $r_1 \bowtie r_2$ di $r_1(X_1)$ e $r_2(X_2)$ è una relazione definita su X_1X_2 (cioè sull'unione degli insiemi X_1 e X_2) come segue:

💡 $\mathbf{r_1 \bowtie r_2 = \{t \text{ su } X_1X_2 \mid \text{esistono } t_1 \in r_1 \text{ e } t_2 \in r_2 \text{ con } t[X_1] = t_1 \text{ e } t[X_2] = t_2\}}$

più sinteticamente:

$$r_1 \bowtie r_2 = \{t \text{ su } X_1X_2 \mid t[X_1] \in r_1 \text{ e } t[X_2] \in r_2\}$$

Importante:

Il join naturale legge ogni riga (tupla o ennupla) della prima tabella e verifica se esiste lo stesso valore dell'attributo anche nella seconda tabella. E' quindi la combinazione di ogni riga di una tabella con le righe dell'altra

Il grado della relazione ottenuta come risultato del join è minore o uguale della somma dei gradi dei due operandi

E' solito eseguire join sulla base di valori della chiave di una delle relazioni coinvolte. In molti di questi casi, fra gli attributi coinvolti è definito un vincolo di integrità referenziale

Join completi e incompleti

💡 **Join completo:** Ciascuna tupla di ciascuno degli operandi contribuisce ad almeno una tupla del risultato

💡 **Join incompleto:** Almeno una tupla degli operandi non contribuisce al risultato

💡 **Join vuoto:** Nessuna tupla degli operandi contribuisce al risultato finale

In caso non tutte le tuple contribuiscano al risultato del join, tali tuple vengono chiamate **dangling**

Il caso limite è quando nessuna delle tuple degli operandi risulta combinabile: il risultato del join è una *relazione vuota*

All'estremo opposto, è possibile che ciascuna delle tuple di ciascun operando sia combinabile con tutte le tuple dell'altro. In tal caso, il risultato contiene un numero di tuple pari al prodotto delle cardinalità degli operandi e cioè $|r_1| \times |r_2|$

Complessità: prodotto delle cardinalità delle relazioni R_1 e R_2

Cardinalità del join

Il join di r_1 e r_2 contiene un numero di ennuple compreso fra zero (join vuoto) e il prodotto di $|r_1|$ e $|r_2|$ (join completo)

Se il join coinvolge una chiave di r_2, allora il numero di ennuple è compreso fra zero e $|r_1|$ (può essere generato al massimo un match per ennupla)

Se il join coinvolge una chiave di r_2 e un vincolo di integrità referenziale (chiave esterna in r_1), allora il numero di ennuple è pari a $|r_1|$ (viene generato necessariamente uno e un solo match per ennupla)

Ricapitolando:

- $r_1(A, B)$, $r_2(B, C)$
  - In generale:
    - $0 <= |r_1 \; JOIN \; r_2| <= |r_1| \times |r_2|$
  - Se il join è completo:
    - Massima cardinalità: $|r_1| \times |r_2|$ → non si hanno attributi in comune
    - Minima cardinalità: $max(|r_1|, |r_2|)$
  - Se B è chiave in r_2:
    - $0 <= |r_1 \; JOIN \; r_2| <= |r_1|$
  - Se B è chiave in r_2 ed esiste vincolo di integrità referenziale fra B (in r_1) e r_2:
    - $|r_1 \; JOIN \; r_2| = |r_1|$

Join esterno

> 💡 **Join esterno:** tutte le tuple danno un contributo al risultato, eventualmente estese con valori nulli in caso non ci siano controparti opportune

Esiste in tre versioni:

- **Sinistro**:
  - Mantiene tutte le ennuple del primo operando, estendendole con valori nulli, se necessario
- **Destro**:
  - Mantiene tutte le ennuple del secondo operando, estendendole con valori nulli, se necessario
- **Completo**:
  - Mantiene tutte le ennuple di entrambi gli operandi, estendendole con valori nulli, se necessario

Proprietà del join naturale

- Commutativo: $r_1 \bowtie r_2 = r2 \bowtie r_1$
- Associativo: $(r_1 \bowtie r_2) \bowtie r_3 = r_1 \bowtie (r_2 \bowtie r_3)$

Facciamo adesso ipotesi sugli attributi:

- In caso $X_1 = X_2$
  - Avremo che: $r_1(X_1) \bowtie r_2(X_2) = r_1(X_1) \cap r_2(X_2)$
  - Per la definizione avremo che il join dovrà essere definito sull'unione dei due attributi, ma , essendo che l'insieme degli attributi su cui sono definite le relazioni, sono uguali, allora l'unione corrisponderà proprio a X_1
  - t risulta definita su X_1: avremo che $t \in r_1$ e $t \in r_2$ che è la definizione di **intersezione**
- In caso $X_1 \cap X_2 = \emptyset$ (insiemi disgiunti)
  - Il join verrà sempre definito sull'unione di X_1 e X_2, ma, visto che le relazioni non avranno nessun attributo in comune, questa degenererà in una condizione sempre vera
  - Il risultato del join saràa la combinazione in tutti i modi possibili delle tuple degli operandi, ovvero il **prodotto cartesiano**

Join e proiezioni

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Impiegato	Reparto
Neri	B
Bianchi	B

Reparto	Capo
B	Mori

Generazione di ennuple spurie (nel rifondere le tabelle ho generato tabelle che non hanno senso nel dominio delle tabelle)

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Bruni
Verdi	A	Bini

Impiegato	Reparto
Neri	B
Bianchi	B
Verdi	A

Reparto	Capo
B	Mori
B	Bruni
A	Bini

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Bruni
Neri	B	Bruni
Bianchi	B	Mori
Verdi	A	Bini

- $R_1(X_1), R_2(X_2)$

- $PROJ_{X_1}(R_1 \ JOIN \ R_2) \subseteq R_1$

- $R(X), X = X_1 \cup X_2$
  - $(PROJ_{X_1}(R)) \ JOIN \ (PROJ_{X_2}(R)) \supseteq R$

Prodotto cartesiano, Theta–join, Equi-join

Un join naturale su relazioni senza attributi in comune

Contiene sempre un numero di ennuple pari al prodotto delle cardinalità degli operandi (le ennuple sono tutte combinabili)

Il prodotto cartesiano, in pratica, ha senso solo se seguito da selezione:

- $\sigma_{\mathbf{Condizione}}(\mathbf{r_1} \bowtie \mathbf{r_2})$

Si va a definire quindi un operatore derivato:

> 💡 **Theta-join**: E' definito come un prodotto cartesiano seguito da una selezione
>
> - $\mathbf{r_1} \bowtie_{\mathbf{Condizione}} \mathbf{r_2} = \sigma_{\mathbf{Condizione}}(\mathbf{r_1} \bowtie \mathbf{r_2})$
> - $\mathbf{r_1} \ \mathbf{JOIN}_{\mathbf{Condizione}} \ \mathbf{r_2}$

Ovviamente la $Condizione$ è una formula proposizionale F

Si definisce adesso un altro operatore derivato:

> 💡 **Equi-join**: E' un theta-join in cui la condizione di selezione è un'uguaglianza

Se l'operatore è sempre l'uguaglianza ($=$) allora si parla di **equi-join:**

Interrogazioni

Un’interrogazione può essere definita come una funzione che applicata a istanze di basi di dati produce relazioni

Dato uno schema R di base di dati, un’interrogazione è una funzione che, per ogni istanza r di R, produce una relazione su un dato insieme di attributi X

Indichiamo con $E(r)$ il **risultato** dell’applicazione dell’espressione E alla base di dati r

Equivalenza di espressioni

Due espressioni sono equivalenti se producono lo stesso risultato qualunque sia l’istanza attuale della base di dati

L’equivalenza è importante in pratica perché i DBMS cercano di eseguire espressioni equivalenti a quelle date, ma meno “costose”

Un’equivalenza particolarmente importante è la **push selections**: anticipazione della selezione rispetto al join(se A è attributo di R_2)

$$SEL_{A=10}(R_1 \ JOIN \ R_2) = R_1 \ JOIN \ SEL_{A=10}(R_2)$$

Selezione con valori nulli

Logica a tre valori:

In presenza di una tupla con valori nulli, il predicato può assumere 3 valori diversi rispetto alla condizioen di una selezione:

- **Vero**
- **Falso**
- **Unknown** → Rappresentato con il simbolo U
  - Unknown rappresenta un valore di verità intermedio tra vero e falso

Si fanno nuove tabelle di verità per i connettivi

not	
F	V
U	U
V	F

and	V	U	F
V	V	U	F
U	U	U	F
F	F	F	F

or	V	U	F
V	V	V	V
U	V	U	U
F	V	U	F

Una selezione su relazioni con valori nulli produce come risultato tuple per cui il predicato risulta vero

Introduciamo due forme di condizioni atomiche di selezione che valutano se un valore è specificato oppure nullo:

- A **IS NULL** assume valore vero su una tuple $t$se il valore di t su A è nullo e falso se esso è sepicificato
- A **IS NOT NULL** assume valore vero su una tupla t se il valore di t su A è specificatp e faslo se il valore è nullo

$SEL_{Età>40}(Impiegati)$

La condizione atomica è vera solo per valori non nulli (Si assuma un impiegato con valore NULL su Età

Esempio di risultato non desiderabile:

$SEL_{Età>30}(Persone) \cup SEL_{Età<=30}(Persone) != Persone$

- Le selezioni non vengono valutate separatamente

Soluzioni corrette (volendo includere i valori nulli):

- $SEL_{Età>30}(Persone) \cup SEL_{Età<=30}(Persone) \cup SEL_{Età\ IS\ NULL}(Persone)$
- $SEL_{Età>40}\ OR_{\ Età\ IS\ NULL}(Impiegati)$

Viste (relazioni derivate)

Rappresentazioni diverse per gli stessi dati (schema esterno)

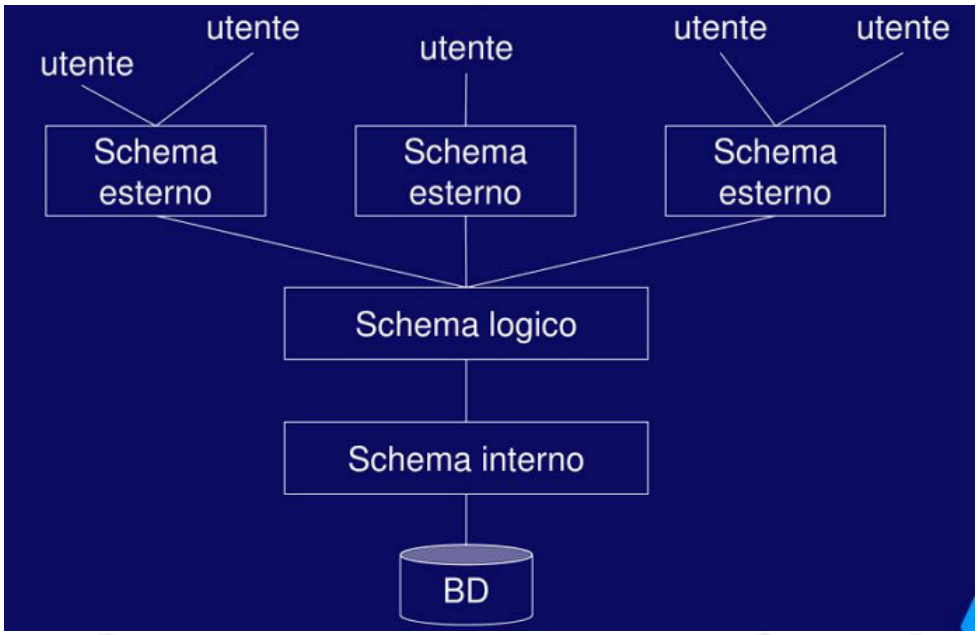
Relazioni derivate (viste):

- Relazioni il cui contento è funzione del contenuto di altre relazioni (definito per mezzo di interrogazioni)

Relazioni di base:

- Contenuto autonomo

Architettura standard (ANSI/SPARC) a tre livelli per DBMS:



Viste virtuali e materializzate

Due tipi di relazioni derivate:

- **Viste materializzate:** relazioni derivate effettivamente memorizzate nella base di dati
- **Relazioni virtuali (o viste):** relazioni definite per mezzo di funzioni (espressioni del linguaggio di interrogazione), non memorizzate nella base di dati, ma utilizzabili nelle interrogazioni come se lo fossero

Viste materializzate

Le viste materializzate vengono effettivamente memorizzate sul disco

Vantaggi:

- Immediatamente disponibili per le interrogazioni (non è necessario ricostruire la vista ad ogni utilizzo)

Svantaggi:

- Ridondanti
- Appesantiscono gli aggiornamenti
- Non sono supportate da molti DBMS

Viste virtuali

Relazioni virtuali (viste):

- Sono supportare dalla maggior parte dei DBMS
- Una interrogazione su una vista viene eseguita “ricalcolando” la vista (o quasi)

Viste, esempio

Afferenza

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B
Bianchi	B

Direzione

Reparto	Capo
A	Mori
B	Bruni
B	Bruni

Una vista:

- $Supervisione = PROJ_{Impiegato,Capo}(Afferenza\ JOIN\ Direzione)$

Interrogazioni sulle viste

Sono eseguite sostituendo alla vista la sua definizione:

$SEL_{Capo='Leoni'}(Supervisione)$

Viene eseguita come:

$SEL_{Capo='Leoni'}(PROJ_{Impiegato,Capo}(Afferenza\ JOIN\ Direzione))$

La complessità computazionale della selezione non cambia

Viste, motivazioni

Utili per…

Schema esterno:

- Ogni utente vede solo ciò che gli interessa e nel modo in cui gli interessa, senza essere distratto dal resto
- Ogni utente vede ciò che è autorizzato a vedere (autorizzazioni)

Strumento di programmazione:

- Si può semplificare la scrittura di interrogazioni: espressioni complesse e sottoespressioni ripetute

Utilizzo di programmi esistenti su schemi ristrutturanti

L'utilizzo di viste non influisce sull'efficienza delle interrogazioni

Viste, aggiornamenti

Aggiornamenti delle viste:

- Si può aggiornare una vista materializzata solo se i cambiamenti sono univocamente riconducibili alle tabelle di base
- Modificare le relazioni di base in modo che la vista, "ricalcolata" rispecchi l'aggiornamento

L'aggiornamento sulle relazioni di base corrispondente a quello specificato sulla vista deve essere univoco

In generale però non è univoco, di conseguenza, ben pochi aggiornamenti sono ammissibili sulle viste

Operatori minimali dell'algebra relazionale

5 operatori con la composizione dei quali si ottengono tutti gli altri (l'insieme è minimo, non ne posso togliere nessuno):

$SEL, PROJ, X, U, -$ (Selezione, Proiezione, Prodotto cartesiano, Unione, Differenza)