BASI DI DATI

Enrico Martini

A.A. 2018/2019

1 Algebra relazionale

1.1 Operatori insiemistici

Applicabili solo a relazioni con lo stesso schema:

- Unione: $r_1 \bigcup r_2$
- Differenza: $r_1 - r_2$
- Intersezione: $r_1 \cap r_2$

1.2 Operatori specifici

Applicabili a singole relazioni:

- Ridenominazione: $\rho_{A_1,A_2,...,A_k \to B_1,B_2,...,B_k}(r)$
- Selezione: $\sigma_F(r)$
- Proiezione: $\Pi_y(r)$

1.3 Operatori di giunzione

Applicabili a coppie di relazioni:

- Join naturale: $r_1 \bowtie r_2$
- Theta-join: $r_1 \bowtie_F r_2$

In caso di presenza di elementi nulli, si utilizzano join esterni:

- Left join: $r_1 \bowtie_{LEFT} r_2$
- Right join: $r_1 \bowtie_{RIGHT} r_2$
- Full join: $r_1 \bowtie_{FULL} r_2$

1.4 Ottimizzazione

L'ottimizzatore esegue trasformazioni di equivalenza allo scopo di ridurre la dimensione dei risultati intermedi.

1.4.1 Trasformazioni di equivalenza

Data E espressione di schema X si definiscono le seguenti trasformazioni di equivalenza:

- Atomizzazione delle selezioni: $\sigma_{F_1 \wedge F_2} \equiv \sigma_{F_1}(\sigma_{F_2}(E))$
- Idempotenza delle proiezioni: $\Pi_Y(E) = \Pi_Y(\Pi_{YZ}(E))$
- Anticipazione delle selezioni: $\sigma_F(E_1 \bowtie E_2) \equiv E_1 \bowtie \sigma_F(E_2)$
- Anticipazione delle proiezioni: $\Pi_{X_1 Y}(E_1 \bowtie E_2) \equiv E_1 \bowtie \Pi_Y(E_2)$
- Inglobamento delle selezioni: $\sigma_F(E_1 \bowtie E_2) \equiv E_1 \bowtie_F B_2$

2 SQL

2.1 Vista

E' una relazione derivata, in cui si specifica l'espressione che genera il suo contenuto, che dipende dalle relazioni che compaiono nell'espressione. Una vista può essere funzione di altre viste purchè esista un ordinamento in grado di guidare il calcolo delle relazioni derivate.

2.1.1 Vista virtuale

Si dice virtuale se viene calcolata ogni volta che serve. Conviene quando l'aggiornamento è frequente e l'interrogazione che genera è una vista semplice. Consente di personalizzare l'interfaccia utente, memorizzare interrogazioni complesse condivise e sono utili per rendere l'interfaccia delle applicazioni indipendente dallo schema logico.

2.1.2 Vista materializzata

Si dice materializzata se viene calcolata e memorizzata esplcitamente nella base di dati. Conviene materializzare quando l'aggiornamento è raro e l'interrogazione che genera è una vista complessa. E' necessario ricalcolarne il contenuto ogni volta che le relazioni da cui dipende vengono aggiornate.

2.2 Viste SQL

Le viste SQL sono sempre virtuali e consentono di salvare interrogazioni complesse condivise o di definire interfacce esterne verso applicazioni o utenti finali:

```
CREATE VIEW <nome_vista> [lista_attributi] AS <query>
```

Le viste consentono di applicare in cascata due operatori aggregati. Non è possibile definire viste ricorsive, ma è possibile definire una vista in termini di altre viste, purchè si evitino dipendenze circolari. Le viste non sono in generale aggiornabili.

2.3 Interrogazione SQL

Struttura base:

```
SELECT <Lista attributi>
FROM <Lista tabelle>
[WHERE <Condizioni>]
```

2.3.1 Clausola SELECT

Struttura:

```
SELECT [DISTINCT] <lista_attributi>
```

La keyword `DISTINCT` se presente richiede l'eliminazione nella relazione risultato delle tuple duplicate. Si può sostituire la lista di attributi con `*` se si vuole tenere tutti gli attributi.

2.3.2 Clausola FROM

Struttura:

```
FROM <tabella> AS <alias>, ...
```

Se sono presenti due o più tabelle si effettua il prodotto cartesiano tra tutte le tabelle. Non è richiesto che le tabelle abbiano schemi disgiunti.

2.3.3 Clausola WHERE

Struttura:

```
WHERE <condizioni>
```

il campo `<condizioni>` è un espressione booleana ottenuta combinando `AND, OR, NOT` con espressioni contenenti riferimenti agli attributi delle tabelle nella clausola `FROM` oppure costanti, unite con `=,<,>,<>,<=,>=`.

2.4 Interrogazioni nidificate

Si ottiene quando nella clausola `WHERE` compare un predicato che contiene un'altra interrogazione SQL:

```
SELECT ...
FROM ...
WHERE <expr> OP (SELECT... FROM ... WHERE ...)
```

Confronta il risultato di un attributo con il risultato di un'altra interrogazione complessa, con degli operatori specifici:

- `A op ANY (query)`: true solo se esiste almeno un valore v contenuto nel risultato tale che soddisfi `op`;
- `A op ALL(query)`: true solo se per ogni valore contenuto nel risultato tale che soddisfi `op`;
- `A IN (query)`: true se A è contenuto nel risultato della query;
- `A NOT IN (query)`: true se A non è contenuto nel risultato della query;
- `EXISTS(query)`: true se la tupla risultato non è vuota;

Le interrogazioni nidificate si dicono **indipendenti** se non ci sono variabili tupla condivise tra l'interrogazione interna e quella esterna. Si dicono **dipendenti** se l'interrogazione nidificata condivide con l'interrogazione esterna almeno una variabile tupla.

2.5 Operatori aggregati

Costituiscono una estensione delle normali interrogazioni SQL che non hanno corrispondenza in algebra lineare. Si specificano nella clausola `SELECT`, vengono applicati all'insieme di tuple risultato di un interrogazione e agiscono su uno o più attributi delle tuple risultato.

1. `COUNT(*/ALL/DISTINCT <lista_attributi>)`: conta le ricorrenze della lista attributi;
2. `SUM(<lista_attributi>)`: torna la somma dei valori della lista attributi;
3. `MAX(<lista_attributi>)`: torna il valore massimo della lista attributi;
4. `MIN(<lista_attributi>)`: torna il valore minimo della lista attributi;
5. `AVG(<lista_attributi>)`: torna la media dei valori della lista attributi;

Se la clausola `SELECT` contiene operatori aggregati non può contenere attributi singoli o espressioni su attributi.

2.6 Clausola GROUP BY

Permettono di applicare un operatore aggregato distintamente a sottoinsiemi di tuple della relazione risultato di una interrogazione. La suddivisione viene eseguita raggruppando insieme tutte le tuple che presentano gli stessi valori per un insieme di attributi. La sintassi è:

```
GROUP BY <lista_attributi>
```

La lista attributi di `SELECT` può solo contenere solo operatori aggregati applicati a espressioni oppure attributi indicati nella clausola `GROUP BY`.
Con il `GROUP BY`:

1. Viene eseguita l'interrogazione base
2. Vengono generati i gruppi di tuple in base agli attributi di `GROUP BY`
3. Si applicano gli operatori aggregati ad ogni gruppo producendo una tupla della relazione risultato per ogni gruppo

2.7 Clausola HAVING

Clausola che permette di selezionare i gruppi in base ad un espressione booleana atomica:

```
HAVING <condizione_sel_gruppi>
```

2.8 Interrogazioni di tipo insiemistico

Consentono di eseguire operazioni insiemistiche sul risultato di due o più interrogazioni SQL:

```
<query1> ins_op <query2>
```

Operatori di tipo insiemistico sono:

- `UNION`
- `INTERSECTION`
- `EXCEPT`

Se viene passata la parola `all` allora non vengono eliminati i duplicati. Le rewlazioni coinvolte in un operatore insiemistico devono avere lo stesso numero di attributi e attributi di tipo compatibile.