SQL - Dispensa Basi Di Dati 2019

Fabio Chiarani

January 18, 2019

Contents

1 Introduzione

SQL è un linguaggio iterattivo nato negli anni '70-'90. Noi vedremmo SQL2 e non SQL3. SQL è quindi un linguaggio di interrogazione dichiarativo (*indico le propietà che deve avere il risultato. Non dico come, ma indico le proprietà*) e fa riferimento al calcolo relazionale (logica del primo ordine).

La fomra base di una *query* è :

```
SELECT <ListaAttributi>
FROM  <ListaTabelle>
[WHERE <condizione>]
```

Indicando con [...] opzionalità (può mancare).

Lo *schema* risultante di una *query* è costituito dagli attributi indicati in `<ListaAttributi>` della clausola `SELECT`. Il contenuto sono tuple proiettato su `<ListaAttributi>` dove le tuple `t'`di `<ListaTabelle>` soddisfano la condizione `WHERE` nella nostra *query*.

1.1 Schema di appoggio

Per gli esempi in questa prima parte utilizzerò il sequente schema:

```
TRENO(NUM, CAT, PART, ARRIVO, DEST)
FERMATA(TRENO, STAZIONE, ORARIO)
```

1.2 Visite

La visita è una relazione derivata. Si specifica l'espressione che genera il suo contenuto. Esso dipende quindi dalle relazioni che compaiono nell'espressione.

Una visita si dice *virtuale* se viene calcolata ogni volta che serve. Una visita si dice *materializzata* se viene calcolata e memorizzata esplicitamente nella base di dati.

2 Clausola SELECT

`SELECT`<ListaAttributi> , dove <*ListaAttributi*> è una lista di espressioni con la seguente sintassi:

```
< [DISTINCT] <espr> [[AS] <alias>] {, <espr> [[AS] <alias> ]} | * >
```

Indicando con:

- `[WORD]` Una parola chiave
- `|` oppure (*or*)
- `<alias>` un nuovo nome che assegno all'attributo (*aliasing*)
- `*` '*star*', ossia prendo tutti gli attributi
- <espr> è una espressione che coinvolge gli attributi della tabella
- `DISTINCT` serve per eliminare i duplicati nella relazione risultato. Non produce quindi risultati (se ho una superchiave non serve)

3 Clausola FROM

`FROM`<ListaTabelle> , dove *<ListaTabelle>* è una lista di espressioni con la seguente sintassi:

```
<tabella> [[AS] <alias>] {, <tabella> [[AS] <alias> ]}
```

- Più tabelle sono seprate da virgole
- Se ci sono più tabelle, la semantica prevede che si genera il prodotto cartesiano tra le tabelle e poi si applica il prodotto (clausola) WHERE. Non viene eseguito alcun JOIN naturale.
- Non c'è dipendenza dallo schema: gli attributi vengono denotati con: `<NomeTabella>.<NomeAttributo>` se ci sono due attributi con nomi uguali in due tabelle distinte.

4 Clausola WHERE

`WHERE`<ListaCondizioni> , dove *<ListaCondizioni>* è un espressionne booleana ottenuta combinando condizioni semplici AND-OR-NOT.

5 Variabili di tupla

Le variabili di tupla (*alias di tabella*) vengono usate per risolvere ambiguità sui nome degli attributi e per gestire il riferimento a più volte della stessa tabella. Esempio:

```
SELECT C.NOME, C.COGNOME
FROM CONTO as C
```

6 Clausola ORDER BY

Questa clausola consente l'ordinamento, la sua sintassi è la seguente:

```
SELECT [...] FROM [...] WHERE [...]
ORDER BY <Attributo> [<ASCI|DESC>] {, <Attributo> [<ASCI|DESC>]}
```

- L'ordine degli attributi determina l'ordine dell'ordinamento.

Esempio di una *query*:

```
SELECT Numero, Partenza
FROM TRENO
WHERE CAT='FrecciaArgento'
ORDER BY Part
```

La sequente query ordinerà per Partenza la tabella risultato.

7 Interrogazioni Nidificate

Un'interrogazione nidificata è ottenuta quando nella clausola `WHERE` compare un **predicato complesso**, vale a dire un predicato che contiene un'altra interrogazione SQL.

```
SELECT ...
FROM ...
WHERE <espr> [theta] (SELECT ... FROM ... WHERE ...)
```

Un **predicato complesso** è un predicato che confronta il valore di un attributo (o espressione) con il risultato di un altra interrogazione SQL.

Attenzione: nel caso tipico l'interrogazione nidificata è mono-attributo, ossia nella clausola SELECT è presente un solo attributo.

I predicati complessi sono:

- `A` op ANY : questo predicato è soddisfatto dalla tupla t se esiste almeno un valore v contenuto nel risultato della query nidificata verifica la condizione: *t[A] op v*
- `A` op ALL : questo predicato è soddisfatto dalla tupla t se per ogni valore v contenuto nel risultato della query nidificata verifica la condizione: *t[A] op v*
- `EXISTS(q)` / NOT EXISTS(q) : ritorna true se q produce almeno una tupla, false altrimenti. q è una interrogazione.

Tips:

- `=ANY` si può scrivere `IN` .
- `<>ANY` si può scrivere `NOTIN` .

Categorizzazione delle Interrogazioni Nidificate Le interrogazioni nidificate si possono categorizzare nel seguente modo: **indipendenti e dipendenti**.

Si dicono interrogazioni **indipendenti** rispetto all'interrogazione esterna che le contiene: in questo caso l'interrogazione interna viene valutata una sola volta in quanto non dipende dalla tupla esterna. L'indipendenza consiste nel fatto che non ci sono variabili di tupla condivise tra l'interrogazione interna ed esterna.

Si dicono interrogazioni **dipendenti** rispetto all'interrogazione esterna che le contiene: in questo caso l'interrogazione interna viene valutata ogni volta e condivide una variabile di tupla che realizza il cosidetto *"passaggio di binding"*.