

# SQL - Dispensa Basi Di Dati 2019

Fabio Chiarani

January 16, 2019

## Contents

<b>1</b>	<b>Introduzione</b>	<b>2</b>
1.1	Schema di appoggio . . . . .	2
1.2	Visite . . . . .	2
<b>2</b>	<b>Clausola SELECT</b>	<b>2</b>
<b>3</b>	<b>Clausola FROM</b>	<b>3</b>
<b>4</b>	<b>Clausola FROM</b>	<b>3</b>
<b>5</b>	<b>Variabili di tupla</b>	<b>3</b>
<b>6</b>	<b>Clausola ORDER BY</b>	<b>3</b>

# 1 Introduzione

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

La forma 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 seguente schema:

## 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.

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

# 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 FROM

WHERE<ListaCondizioni> , dove <ListaCondizioni> è un espressione 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> [<ASCII|DESC>] {, <Attributo> [<ASCII|DESC>]}
```