

# **BASI DI DATI**

Prof. Stefano Ceri  
5 CFU

**Tommaso Fontana**  
**Valentina Deda**

Lecture Notes  
Year 2017/2018



Triennale Ingegneria Informatica  
Politecnico di Milano  
Italy  
4 gennaio 2018

# Indice

<b>1</b>	<b>Le query SQL</b>	<b>2</b>
1.1	Sintassi delle query SQL	2
1.2	Attributi con valore null	2
1.3	Tabelle e join	3
1.4	Ridenominazione	3
1.5	Modificare il database in SQL	3
1.6	Query complesse	4
1.6.1	Ordinamento	4
1.6.2	Funzioni aggregate	4
1.6.3	Raggruppamento	4
1.6.4	Query binarie	4
1.6.5	Query nidificate	5

# 1

## Le query SQL

### 1.1 Sintassi delle query SQL

Ogni query SQL si compone di tre clausole:

- `select`
- `from`
- `where`

Ogni query ha sintassi:

`select *attributo*`

`from *tabella*`

`[ where *condizione* ]`

Dunque *select* indica l'attributo che ci interessa estrarre, *from* la tabella da cui estrarre l'attributo, *where* la condizione che l'attributo deve rispettare affinché sia rilevante per la nostra query.

La condizione indicata dal *where* può essere espressa tramite un'espressione i cui operatori sono:

- **predicati semplici dell'algebra**: uguaglianza, operatori booleani ...
- **between**: indica gli estremi di un intervallo.
- **distinct**: impedisce che vi siano duplicati fra i risultati della query

**like**: indica parte del nome dell'attributo cercato, ad esempio

```
1 | nome like 'Mar\%o'
```

cerca i nomi che iniziano con “Mar” e finiscono con “o”. Con `like` si usano i simboli `_` e `%`, che indicano rispettivamente un carattere e una sottostringa (di lunghezza arbitraria) di cui non conosciamo (o non ci interessa) il valore; dunque qualunque valore essi abbiano nell'attributo, tale attributo sarà ritenuto valido e restituito dalla query.

### 1.2 Attributi con valore null

Esiste un valore speciale **null**, che viene utilizzato se non si conosce un valore, se un valore non si può applicare a un determinato attributo, se non si sa se tale valore possa essere applicato a un determinato attributo. `is null` può essere usato nelle query attraverso gli operatori

```
1 | attributo is null
```

e

```
1 | attributo is null
```

che restituiscono rispettivamente le righe con valore nullo per un determinato attributo e quelle con valore non nullo per un determinato attributo.

## 1.3 Tabelle e join

Le tabelle indicate dopo la clausola *where* indicano il dominio della query. Possono essere indicate più tabelle, separate da una virgola: la query ne farà automaticamente il prodotto cartesiano e cercherà l'attributo richiesto in tale prodotto cartesiano. Si può anche effettuare la join in modo esplicito, con l'espressione

```
1 | select attributo from tabella1 join tabella on condizione_su_cui_viene_effettuata_la_join
```

La condizione della join è espressa come in algebra.

La *join* può essere di due tipi differenti:

- interna
- esterna

Per indicare la *join* interna si scrive semplicemente “join”, per indicare una *join* esterna si scrive “left/right/full join”. Una *join* esterna restituisce anche le righe per cui la condizione espressa dall’“on” della join restituisca valori nulli. Dunque con una *left join* prenderò tutte le righe della tabella dichiarata a sinistra della join, anche se in quella di destra alcune di esse avranno attributi con valori nulli. La *right join* fa lo stesso ma con le righe della tabella a destra dell’operatore join. La *full join* prende tutte le righe di entrambe le tabelle.

## 1.4 Ridenominazione

È possibile “dare un nome” al risultato della query usando l’operatore “as”, ad esempio:

```
1 | select * as informatico from STUDENTE where ...
```

As si usa anche nell’espressione della *from* nel caso si debbano estrarre più variabili dalla stessa tabella. Ad esempio

```
1 | select * from STUDENTE as stud1, STUDENTE as stud2, ...
```

## 1.5 Modificare il database in SQL

È possibile:

- Effettuare inserimenti:

```
1 | insert into nome tabella values lista valori
```

vengono messi a null o al valore di default.

Se mancano dei valori nella lista Al posto di usare “values + lista valori” si può inserire il risultato di una query, scrivendo la query, come una normale query, alla fine del comando:

```
1 | insert into nome tabella query
```

- Effettuare cancellazioni:

```
1 | delete from nome tabella where condizione che identifica gli elementi da cancellare
```

- Modificare i valori degli attributi:

```
1 | update nome_tabella set attributo = valore where condizione
```

Cancellare intere tabelle: **drop table \*nome tabella\***

```
1 | update nome_tabella set attributo = valore where condizione
```

## 1.6 Query complesse

### 1.6.1 Ordinamento

È possibile riordinare i risultati di una query attraverso il comando **order by**. Si usa la sintassi:

```
1 | order by AttributoOrdinamento [ crescente | decrescente ]
```

### 1.6.2 Funzioni aggregate

Le funzioni aggregate sono funzioni, utilizzate all'interno delle espressioni della clausola *where*, che utilizzano operatori complessi, che operano su più elementi del database. Esistono cinque operatori SQL per realizzare funzioni aggregate: - **count**: restituisce il numero di righe per un certo attributo. Se si aggiunge l'operatore *distinct*, restituisce il numero di righe per l'attributo omettendo le righe duplicate.

```
1 | count( * | [ distinct | all ] ListaAttributi )
```

- **sum**: restituisce la somma dei valori dell'attributo passato come parametro. - **max**: restituisce il massimo fra i valori dell'attributo passato come parametro. - **min**: restituisce il minimo fra i valori dell'attributo passato come parametro. - **avg**: restituisce la media fra i valori dell'attributo passato come parametro.

```
1 | sum|max|min|avg ([distinct|all] Attributo )
```

### 1.6.3 Raggruppamento

È possibile che ci sia il bisogno di applicare gli operatori appena visti a un sottoinsieme di righe di una tabella, non all'intera tabella. Allora si usa l'operatore *group by*, che seleziona le righe che ci interessano.

```
1 | group by attributo1 having operatore_aggregato(attributo2)
```

### 1.6.4 Query binarie

Sono realizzate concatenando due query attraverso gli operatori insiemistici di unione, intersezione e differenza.

```
1 | Query1 union | intersect | except [ all ] Query2
```

Se si usa l'operatore "all" vengono inclusi anche eventuali duplicati, se si omette essi sono esclusi automaticamente.

### 1.6.5 Query nidificate