Corso di Laurea in Ingegneria Gestionale
Sapienza Università di Roma

Corso di Basi di Dati
A.A. 2019/2020

4 – SQL : Structured Query Language

Tiziana Catarci

SQL : Structured Query Language

- SQL non è un semplice linguaggio per le interrogazioni...
- ...ma si divide in 3 sotto-linguaggi principali:
  - **DDL (D**ata **D**efinition **L**anguage**) :** linguaggio che permette di creare\eliminare\modificare gli **oggetti** in un database (tabelle e viste). I comandi DDL definiscono la **struttura** del database.
  - **DML (D**ata **M**anipulation **L**anguage**) :** linguaggio che permette di leggere\inserire\modificare\eliminare i **dati** di un database. Le **interrogazioni** in SQL appartengono a DML.
  - **DCL (D**ata **C**ontrol **L**anguage**) :** permette di fornire o revocare agli utenti i permessi necessari per poter utilizzare i comandi di DDL e DML su specifici oggetti/dati di un database.
- Altre componenti: **trigger** (azioni eseguite dal DBMS che soddisfano determinate condizioni), **SQL dinamico ed embedded**, **esecuzione client-server**, **gestione di transazioni**, **sicurezza**.

Interrogazioni in SQL

- SQL si basa sia sull'algebra che sul calcolo relazionale, perciò esprime le interrogazioni in modo misto: in parte ***dichiarativo*** e in parte ***procedurale***.
- L'interrogazione SQL viene passata all'***ottimizzatore di interrogazioni*** (***query optimizer***), un componente del DBMS che analizza l'interrogazione e ne costruisce una versione equivalente in un linguaggio procedurale basato sull'algebra relazionale interno al DBMS.
- **NOTA BENE** : **Esistono molti modi diversi per esprimere un'interrogazione in SQL.**
  - Il programmatore dovrà effettuare una scelta non basandosi sull'efficienza, bensì su caratteristiche come la leggibilità e la modificabilità dell'interrogazione.

SQL : Alcune Notazioni

- Notazione utilizzata per specificare la sintassi dei comandi:
  - Le parentesi quadre **[ ]** indicano che il termine contenuto al suo interno è **opzionale**, cioè può non comparire oppure comparire una sola volta.
  - Le parentesi graffe **{ }** indicano che il termine racchiuso può non comparire o essere ripetuto un numero arbitrario di volte.
  - Le barre verticali **|** indicano che deve essere scelto **uno** tra i termini separati dalle barre.
  - Le parentesi tonde **( )** dovranno essere intese sempre come termini del linguaggio SQL e non come simboli per la definizione della grammatica.

Sintassi di un interrogazione SQL

SELECT [**DISTINCT**] *listaAttributi*
FROM *listaTabelle*
[**WHERE** *condizione*]

- Un'interrogazione SQL può essere valutata analizzando i comandi che la compongono nel seguente ordine :

1. ***listaTabelle*** = lista delle tabelle su cui calcolare il risultato.
2. ***condizione*** = espressione booleana ottenuta combinando gli operatori di confronto (<, <=, =, <>, >=, >) e gli operatori logici AND, OR, NOT.
3. ***listaAttributi*** = lista di attributi (presi dalle tabelle contenute in *listaTabelle*) che definiscono il risultato dell'interrogazione.
   - **DISTINCT** è una parola chiave opzionale utile per specificare che il risultato dell'interrogazione **non deve contenere duplicati.**

Strategia di valutazione concettuale

SELECT [**DISTINCT**] *listaAttributi*
FROM *listaTabelle*
[**WHERE** *condizione*]

- L'interrogazione SQL seleziona, tra le righe che appartengono al **prodotto cartesiano** delle tabelle elencate nella clausola **FROM**, quelle che soddisfano le condizioni espresse nell'argomento della clausola **WHERE**.
- Il risultato di un'interrogazione SQL è **una tabella** (non necessariamente una relazione matematica!) le cui colonne si ottengono dalla valutazione delle espressioni cha appaiono nella clausola **SELECT**.

1.Clausola **FROM**

Per formulare un'interrogazione che coinvolge tuple appartenenti a più di una tabella, si pone come argomento della clausola **FROM** l'insieme di tabelle alle quali si vuole accedere.

```
SELECT DISTINCT Impiegato
FROM Impiegati, Reparti
WHERE Impiegato = 'Neri'
```

Impiegati

Impiegato	Codice
Rossi	A
Neri	B
Bianchi	B

Reparti

Capo	Codice
Mori	A
Bruni	B

Il risultato *parziale* consiste nel **prodotto cartesiano** delle tabelle elencate nella clausola **FROM**.

Impiegati Reparti

Impiegato	Codice	Capo	Codice
Rossi	A	Mori	A
Rossi	A	Bruni	B
Neri	B	Mori	A
Neri	B	Bruni	B
Bianchi	B	Mori	A
Bianchi	B	Bruni	B

2.Clausola **WHERE**

SELECT DISTINCT *Impiegato, Codice*
FROM *Impiegati, Reparti*
WHERE *Impiegato = 'Neri'*

Impiegati **Reparti**

Impiegato	Codice	Capo	Codice
Rossi	A	Mori	A
Rossi	A	Bruni	B
Neri	**B**	**Mori**	**A**
Neri	**B**	**Bruni**	**B**
Bianchi	B	Mori	A
Bianchi	B	Bruni	B

Sul **prodotto cartesiano** delle tabelle elencate nella clausola **FROM** verranno applicate le condizioni contenute nella clausola **WHERE**.

3.Clausola **SELECT**

La clausola **SELECT** specifica quali attributi faranno parte della tabella risultato.

```
SELECT DISTINCT Impiegato
FROM Impiegati, Reparti
WHERE Impiegato = 'Neri'
```

Impiegati **Reparti**

Impiegato	Codice	Capo	Codice
Rossi	A	Mori	A
Rossi	A	Bruni	B
Neri	**B**	**Mori**	**A**
Neri	**B**	**Bruni**	**B**
Bianchi	B	Mori	A
Bianchi	B	Bruni	B

Il risultato di un'interrogazione SQL è un **multi-insieme**. Se si desidera che la tabella calcolata **non contenga duplicati**, si deve includere la parola chiave **DISTINCT**.

Impiegato
Neri
~~Neri~~

Impiegato
Neri

Confronto con Algebra Relazionale: Proiezione e Select

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Rossi	Amministrazione	40
Franco	Neri	Distribuzione	45

Il risultato di un'interrogazione in Algebra Relazionale è una **relazione**, perciò i duplicati saranno automaticamente scartati nel risultato. Viceversa, il risultato di un'interrogazione SQL è un **multi-insieme.** Affinché la tabella calcolata **non contenga duplicati**, si deve includere la parola chiave **DISTINCT**.

ESERCIZIO **:Estrarre cognome e dipartimento di tutti gli impiegati. Scrivere l'interrogazione sia in Algebra Relazione che in SQL.**

$\Pi_{Cognome,Dipart}$ **(Impiegato)**

Cognome	Dipart
Rossi	Amministrazione
Bianchi	Produzione
Neri	Distribuzione

SELECT Cognome, Dipart
FROM Impiegati

Cognome	Dipart
Rossi	Amministrazione
Bianchi	Produzione
Rossi	Amministrazione
Neri	Distribuzione

SELECT DISTINCT Cognome, Dipart
FROM Impiegati

Cognome	Dipart
Rossi	Amministrazione
Bianchi	Produzione
Neri	Distribuzione

Select con asterisco (*)

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Come argomento della clausola SELECT può anche comparire il carattere speciale * (**asterisco**), che rappresenta la selezione di tutti gli attributi delle tabelle elencate nella clausola **FROM**.

ESERCIZIO **:Estrarre tutte le informazioni degli impiegati di cognome "Rossi"**

```
SELECT *
FROM Impiegato
WHERE Cognome='Rossi'
```

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Rossi	Direzione	80

Ridenominazione

Ogni attributo del risultato può essere rinominato con un ***Alias.***

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre lo Stipendio (e ri-nominarlo come "Salario") degli impiegati di cognome "Rossi"**

```
SELECT StipAnn AS Salario
FROM Impiegato
WHERE Cognome='Rossi'
```

Salario
45
80

NOTA BENE: Se non vi fossero impiegati di cognome "Rossi", l'interrogazione restituirebbe un insieme vuoto.

AS : operatore di **ridenominazione**. Consente di rinominare gli attributi del risultato.

Ridenominazione

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

Nella clausola ***SELECT*** possono comparire generiche espressioni sul valore degli attributi di ciascuna riga selezionata.

ESERCIZIO **:Estrarre lo stipendio mensile dell'impiegato di cognome "Bianchi"**

```
SELECT StipAnn/12 AS StipendioMensile
FROM Impiegato
WHERE Cognome='Bianchi'
```

StipendioMensile
3

Convenzioni sui nomi

- Per evitare ambiguità, ogni nome di attributo è composto da
  NomeTabella.***NomeAttributo***
- Quando l'ambiguità non sussiste, si può omettere la parte
  NomeTabella

```
SELECT persone.nome, persone.reddito
FROM persone
WHERE persone.età<30
```

si può scrivere come:

```
SELECT nome, reddito
FROM persone
WHERE età<30
```

Variabili di range

Per evitare ambiguità tra attributi aventi lo stesso nome in tabelle diverse, si possono definire ***variabili di range*** nella clausola FROM.

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre i nomi e i cognomi degli Impiegati e le Città in cui lavorano.**

```
SELECT I.Nome, Cognome, Città
FROM Impiegato AS I, Dipartimento AS D
WHERE Dipart=D.Nome
```

Nome	Cognome	Città
Mario	Rossi	Milano
Carlo	Bianchi	Torino
Giuseppe	Verdi	Milano
Franco	Neri	Roma
Carlo	Rossi	Milano
Lorenzo	Gialli	Milano
Paola	Rosati	Milano
Marco	Franco	Torino

Variabili di range

Per specificare variabili di range, **non è necessario** utilizzare ***AS***.

```
SELECT I.Nome, Cognome, Città
FROM Impiegato as I, Dipartimento as D
WHERE Dipart=D.Nome
```

equivale a

```
SELECT I.Nome, Cognome, Città
FROM Impiegato I, Dipartimento D
WHERE Dipart=D.Nome
```

- Le variabili di range possono essere anche utilizzate per disporre di un "duplicato" di una tabella, utile ai fini di un'interrogazione.

ESERCIZIO **: Estrarre il cognome degli Impiegati con lo stesso Nome che lavorano in reparti differenti.**

```
SELECT I1.Cognome,
FROM Impiegato I1, Impiegato I2
WHERE I1.Nome=I2.Nome AND
      I1.Reparto <> I2.Reparto
```

I1

Nome	Cognome	Reparto
Mario	Rossi	A
Mario	Bianchi	B
Gianni	Verdi	A

I2

Nome	Cognome	Reparto
Mario	Rossi	A
Mario	Bianchi	B
Gianni	Verdi	A

Cognome
Rossi
Bianchi

NOT, AND, OR

La clausola **WHERE** consente di costruire un'espressione booleana combinando predicati semplici con gli operatori **AND**, **OR** e **NOT**.

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre il nome ed il cognome degli Impiegati che lavorano nel dipartimento Amministrazione ed hanno stipendio maggiore di 70**

```
SELECT Nome,Cognome
FROM Impiegato
WHERE Dipart = 'Amministrazione'
AND StipAnn > 70
```

Nome	Cognome

NOT, AND, OR

ATTENZIONE : in SQL l'**AND** e l'**OR** **hanno la stessa priorità** (mentre il **NOT** ha **priorità maggiore**). Conviene esplicitare l'ordine di valutazione degli operatori **mediante parentesi .**

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre i nomi degli Impiegati di cognome "Rossi" che lavorano nei dipartimenti Amministrazione o Produzione**

```
SELECT Nome
FROM Impiegato
WHERE Cognome='Rossi' AND
(Dipart = 'Amministrazione' OR
Dipart = 'Produzione')
```

Nome
Mario

Esercizio

- Calcolare la tabella ottenuta dalla tabella **Persone** selezionando solo quelle persone con reddito tra 20 e 30, e aggiungendo al risultato un attributo che ha, in ogni tupla, lo stesso valore dell'attributo **Reddito**.

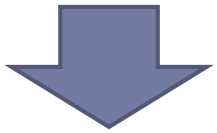
- Mostrare il risultato dell'interrogazione.

Persone

Nome	Età	Reddito

Soluzione Esercizio

```
SELECT nome, età, reddito, reddito AS ancoraReddito
FROM Persone
WHERE Reddito >= 20   AND   Reddito <= 30
```



Nome	Età	Reddito	ancoraReddito

Operatore LIKE

SQL mette a disposizione un operatore **LIKE** per il confronto fra stringe.

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	**Franco**	**Produzione**	**46**

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre gli Impiegati con un nome che comincia una "m" e che ha la coppia di caratteri "rc" in penultima posizione**

```
SELECT *
FROM Impiegato
WHERE Nome LIKE 'm%rc_'
```

Il carattere _ rappresenta un confronto con un carattere arbitrario, mentre **%** rappresenta un confronto con una stringa di lunghezza arbitraria (eventualmente nulla).

Gestione dei valori nulli

Impiegato

Nome	Cognome	Dipart	Età
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	**Rossi**	**Direzione**	**80**
Lorenzo	**Gialli**	**Direzione**	**73**
Paola	Rosati	Amministrazione	40
Marco	**Franco**	**Produzione**	**NULL**

Per selezionare o meno i termini con i valori NULL, SQL fornisce il predicato **IS [NOT] NULL.**

Di default, la condizione espressa nella **WHERE** è vera solo per valori **NON NULLI.**

ESERCIZIO **:Estrarre gli Impiegati la cui età potrebbe essere maggiore di 70**

SELECT *
FROM Impiegato
WHERE Età>70 or Età **IS NULL**

Stessa interrogazione espressa in algebra relazionale.

$\sigma_{Età>70\ OR\ Età\ IS\ NULL}$ **(Impiegato)**

Esercizio

- Calcolare la tabella ottenuta dalla tabella **Impiegato** selezionando solo quelli delle filiali di Roma e Milano, proiettando i dati sull'attributo **Stipendio,** ed aggiungendo un attributo che ha, in ogni tupla, il valore doppio dell'attributo **Stipendio**.

- Mostrare il risultato dell'interrogazione.

Impiegato

Matricola	Cognome	Filiale	Stipendio

Soluzione Esercizio

```
SELECT Stipendio, Stipendio*2 AS StipendioBIS
FROM Impiegati
WHERE Filiale = 'Milano'   OR   Filiale = 'Roma'
```

Stipendio	StipendioBIS

Interpretazione formale delle interrogazioni SQL

- E' possibile costruire una corrispondenza tra le interrogazioni SQL ed equivalenti interrogazioni espresse in algebra relazionale.
- Date le tabelle: **R1(A1,A2)** e **R2(A3,A4)**
  la semantica della query :

```
SELECT R1.A1, R2.A4
FROM R1, R2
WHERE R1.A2 = R2.A3
```

si può descrivere in termini di :

- prodotto cartesiano (**from**)
- selezione (**where**)
- proiezione (**select**)

Sarebbe possibile mostrare una tecnica per tradurre ogni interrogazione SQL in un'equivalente interrogazione in algebra relazionale.

$$\Pi_{A1,A4}\left(\sigma_{A2=A3}(R1 \bowtie R2)\right)$$

Esercizio

- Si supponga di disporre di una tabella **Persone** e di una tabella **Paternità**. Sia l'attributo **Padre** che l'attributo **Figlio** sono legati da un vincolo di *foreign key* verso **Persone.nome**.

- Mostrare in SQL e in Algebra Relazionale i padri di persone che guadagnano più di 20 milioni.

Persone

Nome	Reddito

Paternità

Padre	Figlio

Soluzione Esercizio

Persone

Nome	Reddito

Paternità

Padre	Figlio

```
SELECT DISTINCT Paternità.Padre
FROM Persone, Paternità
WHERE Paternità.Figlio = Persone.Nome AND
        Persone.Reddito > 20
```

$$\Pi_{Padre}\left(paternità \bowtie_{figlio=nome} \left(\sigma_{reddito>20}(persone)\right)\right)$$

Esercizio

- Si supponga di disporre di una tabella **Persone,** di una tabella **Paternità** e di una tabella **Maternità**. L'attributo **padre** e l'attributo **figlio** della tabella **Paternità**, l'attributo **madre** e l'attributo **figlio** della tabella **Maternità** sono legati da un vincolo di *foreign key* verso **Persone.nome**.

- Mostrare in SQL e in Algebra Relazionale i padri e le madri di ogni persona.

Persone

nome	reddito

Paternità

padre	figlio

Maternità

madre	figlio

Soluzione Esercizio

Persone

nome	reddito

Paternità

padre	figlio

Maternità

madre	figlio

```
SELECT DISTINCT paternità.figlio, padre, madre
FROM Maternità, Paternità
WHERE Paternità.figlio = Maternità.figlio
```

I **JOIN** (e i prodotti cartesiani) si realizzano indicando due o più tabelle nella clausola **FROM**.

$$(paternità \bowtie maternità)$$

Esercizio

- Si supponga di disporre di una tabella **Persone** e di una tabella **Paternità**. Sia l'attributo **Padre** che l'attributo **Figlio** sono legati da un vincolo di *foreign key* verso **Persone.Nome**.

- Mostrare in SQL le persone che guadagnano più dei rispettivi padri, mostrando nome e reddito della persone e reddito del padre.

Persone

Nome	Reddito

Paternità

Padre	Figlio

Soluzione Esercizio

Persone

Nome	Reddito

Paternità

Padre	Figlio

```
SELECT f.Nome, f.Reddito, p.Reddito
FROM Persone p, Paternità t, Persone f
WHERE p.Nome = t.Padre AND
      t.Figlio = f.Nome AND
      f.Reddito>p.Reddito
```

JOIN Esplicito

- Una sintassi alternativa per la specifica dei JOIN permette di distinguere, tra le condizioni che compaiono nell'interrogazione, quelle che rappresentano condizioni di JOIN e quelle che rappresentano condizioni di selezioni fra le righe.

SELECT [DISTINCT] *listaAttributi*
FROM *Tabella* {**JOIN** *AltraTabella* **ON** *CondDiJoin*}
[**WHERE** *altraCondizione*]

- Mediante questa sintassi la condizione di **JOIN** **non compare** come argomento della clausola **WHERE**, ma viene spostata invece nell'ambito della clausola **FROM,** associata alle tabelle che vengono coinvolte nel **JOIN**.

JOIN Esplicito - Esempio

- Mostrare in SQL i padri e le madri di ogni persona.

Persone

nome	reddito

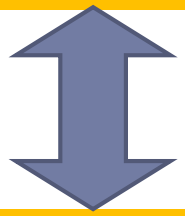
Paternità

padre	figlio

Maternità

madre	figlio

```
SELECT DISTINCT paternità.figlio, padre, madre
FROM Maternità, Paternità
WHERE Paternità.figlio = Maternità.figlio
```



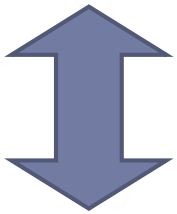
A differenza dell'Algebra Relazionale, in questo caso **vengono mantenuti nel risultato tutti gli attributi su cui viene valutato il JOIN.**

```
SELECT DISTINCT paternità.figlio, padre, madre
FROM Maternità JOIN Paternità ON Paternità.figlio = Maternità.figlio
```

JOIN Esplicito - Esempio

- Mostrare in SQL le persone che guadagnano più dei rispettivi padri, evidenziando nome e reddito della persone e reddito del padre.


```
SELECT f.nome, f.reddito, p.reddito
FROM Persone p, Paternità t, Persone f
WHERE p.nome = t.padre  AND t.figlio = f.nome  AND  f.reddito>p.reddito
```



```
SELECT DISTINCT paternità.figlio, padre, madre
FROM Persone p JOIN Paternità t ON p.nome = t.padre
               JOIN Persone f ON t.figlio=f.nome
WHERE f.reddito>p.reddito
```

JOIN Naturale

- In SQL è possibile esprimere anche la condizione di **JOIN naturale**, anche se è un comando poco diffuso.

Paternità

padre	figlio

Maternità

madre	figlio

SELECT DISTINCT paternità.figlio, padre, madre
FROM Maternità, Paternità
WHERE Paternità.figlio = Maternità.figlio

$(paternità \bowtie maternità)$

SELECT DISTINCT paternità.figlio, padre, madre
FROM Maternità **NATURAL JOIN** Paternità

Come nel JOIN naturale dell'Algebra Relazionale, **viene mantenuto nel risultato solo uno degli attributi su cui viene valutato il JOIN.**

JOIN Esterni

- La caratteristica dell'operatore di JOIN è di "tralasciare" le tuple di una tabella che non hanno controparte nell'altra.
- In alcuni casi ciò può portare ad omettere informazioni rilevanti.

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparto	Capo
B	Mori
C	Bruni

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori

Alcune tuple vengono tagliate fuori dal JOIN.

- Il join esterno estende, con valori nulli, le tuple che verrebbero tagliate fuori da un join (interno). Esiste in tre versioni:
  - **sinistro**: mantiene tutte le tuple del primo operando, estendendole con valori nulli, se necessario
  - **destro**: ... del secondo operando ...
  - **completo**: … di entrambi gli operandi ...

JOIN Esterno Sinistro

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Reparto	Capo
B	Mori
C	Bruni

Impiegati $JOIN_{LEFT}$ Reparti

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL

Mantiene tutte le tuple del primo operando, estendendole con valori nulli, se necessario.

```
SELECT Impiegato, Reparto, Capo
FROM Impiegati I  LEFT JOIN  Reparti R
                                ON I.Reparto = R.Reparto
```

JOIN Esterno Destro

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Reparto	Capo
B	Mori
C	Bruni

Impiegati $JOIN_{RIGHT}$ Reparti

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
NULL	C	Bruni

Mantiene tutte le tuple del secondo operando, estendendole con valori nulli, se necessario.

```
SELECT Impiegato, Reparto, Capo
FROM Impiegati I  RIGHT JOIN  Reparti R
                                    ON I.Reparto = R.Reparto
```

JOIN Esterno Completo

Impiegati

Impiegato	Reparto
Rossi	A
Neri	B
Bianchi	B

Reparti

Reparto	Capo
B	Mori
C	Bruni

Impiegati $\text{JOIN}_{\text{FULL}}$ Reparti

Impiegato	Reparto	Capo
Neri	B	Mori
Bianchi	B	Mori
Rossi	A	NULL
NULL	C	Bruni

Mantiene tutte le tuple di entrambi gli operandi, estendendole con valori nulli, se necessario.

```
SELECT Impiegato, Reparto, Capo
FROM Impiegati I  FULL JOIN  Reparti R
                                ON I.Reparto = R.Reparto
```

Il comando ORDER BY

- Una tabella è costituita da un insieme non ordinato di tuple. Nell'uso reale delle basi di dati sorge spesso il bisogno di **costruire un ordine** sulle righe delle tabelle.
- SQL permette di specificare un ordinamento sulle righe del risultato di un'interrogazione tramite la clausola **ORDER BY**

ORDER BY *AttrDiOrdinamento* [ASC|DESC]
{, *AttrDiOrdinamento* [ASC|DESC] }

  - ***le righe vengono ordinate in base al primo attributo nell'elenco;***
  - ***per righe che hanno lo stesso valore dell'attributo, si considerano i valori degli attributi successivi in sequenza;***
  - ***l'ordine degli attributi può essere ascendente o discendente, a seconda che si usi il qualificatore* ASC *o* DESC *(se il qualificatore è omesso, si asSUMe un ordinamento* ASC*).***

ORDER BY

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

Ordinamento ascendente per ***Nome*** e ***Cognome***. Prima vengono ordinati i valori contenuti in ***Nome***.

ESERCIZIO **:Estrarre nome e cognome degli impiegati che lavorano in Amministrazione, in ordine alfabetico di nome e cognome**

```
SELECT Nome,Cognome
FROM Impiegato
WHERE Dipart='Amministrazione'
ORDER BY Nome, Cognome
```

Nome	Cognome
Giuseppe	Verdi
Mario	Rossi
Paola	Rosati

Successivamente, per tuple che hanno lo stesso valore in ***Nome***, si ordinano i valori di ***Cognome***.

Operatori Aggregati

- Gli operatori aggregati costituiscono una delle più importanti estensioni di SQL rispetto all'Algebra Relazionale.
- In Algebra Relazionale tutte le condizioni vengono valutate su una tupla alla volta, indipendentemente da tutte le altre.
- Spesso nei contesti reali viene però richiesto di valutare proprietà che dipendono da insiemi di tuple.
- SQL permette di inserire nelle espressioni della clausola **SELECT** espressioni che calcolano valori a partire da insiemi di tuple.
  - **conteggio, minimo, massimo, media, totale**

Operatori aggregati

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre il numero di Impiegati del dipartimento Produzione**

```
SELECT count(*)
FROM Impiegato
WHERE Dipart = 'Produzione'
```

count(*)
2

count : operatore aggregato di conteggio. Conta quante tuple soddisfano le condizioni inserite nella clausola WHERE.

Operatori aggregati – come gestirli

- Gli operatori aggregati vengono gestiti come **un'estensione alle normali interrogazioni**.
  - Prima di tutto viene normalmente eseguita l'interrogazione, considerando solo le parti **FROM** e **WHERE**.
  - L'operatore aggregato viene poi applicato alla tabella contenente il risultato dell'interrogazione.

ESEMPIO **: Estrarre il numero di Impiegati del dipartimento Produzione**

- Prima si costruisce la tabella che contiene tutte le righe di Impiegato che hanno "Produzione" come valore dell'attributo *Dipart*.

```
SELECT count(*)
FROM Impiegato
WHERE Dipart = 'Produzione'
```

Nome	Cognome	Dipart	StipAnn
Carlo	Bianchi	Produzione	36
Marco	Franco	Produzione	46

- Successivamente si conta il numero di righe che compongono la tabella (in questo caso 2).

Operatore Count - Sintassi

count (*) | ([DISTINCT|ALL] *ListaAttributi* **)**

- L'opzione * restituisce il numero di righe.
- L'opzione **DISTINCT** restituisce il numero di diversi valori contenuti in *ListaAttributi.*
- L'opzione **ALL** (considerata di default) restituisce invece il numero di tuple di *ListaAttributi* che possiedono valori diversi da NULL.

Operatori aggregati

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **: Estrarre il numero di diversi valori dell'attributo Stipendio fra tutte le righe di Impiegato**

```
SELECT count(DISTINCT StipAnn)
FROM Impiegato
```

count(distinct StipAnn)
6

In questo caso si conta il numero di **diversi valori** dell'attributo StipAnn.

Operatori aggregati

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **: Trovare il numero della persone che hanno sia il nome che il cognome specificato**

```
SELECT count(ALL Nome, Cognome) AS Tutti
FROM Impiegato
```

Tutti
8

Count e valori NULL

```
select count(*)
from   persone
```

Risultato = numero di ennuple
= 4

```
select count(reddito)
from   persone
```

Risultato = numero di valori diversi da NULL
= 3

```
select count(distinct reddito)
from   persone
```

Risultato = numero di valori distinti (escluso **NULL**
= 2

persone

nome	eta	reddito
Andrea	27	21
Aldo	25	NULL
Maria	55	21
Anna	50	35

Altri operatori aggregati

SUM|AVG|MAX|MIN ([DISTINCT|ALL] *AttrEspr* **)**

- ammettono come argomento un attributo o un'espressione **(ma non l'asterisco "*")**
- **ignorano i valori NULL**
- **SUM** : restituisce la somma dei valori posseduti dall'espressione.
  - accetta argomenti numerici o tempo.
- **AVG**: restituisce la media dei valori
  - accetta argomenti numerici o tempo.
- **MAX** e **MIN**: restituiscono rispettivamente il valore massimo e minimo di attributi su cui è definito un ordinamento.

L'operatore SUM

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre la somma degli Stipendi del dipartimento Amministrazione**

```
SELECT SUM(StipAnn)
FROM Impiegato
WHERE Dipart = 'Amministrazione'
```

SUM(StipAnn)
125

L'operatore MAX

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre il massimo stipendio tra quelli degli impiegati che lavorano in un dipartimento con sede a Milano**

```
SELECT MAX(StipAnn)
FROM Impiegato, Dipartimento D
WHERE Dipart = D.Nome and Città='Milano'
```

MAX(StipAnn)
80

Combinare operatori aggregati

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre gli stipendi minimo, massimo e medio fra quelli di tutti gli impiegati**

```
SELECT MAX(StipAnn),
       MIN(StipAnn),
       AVG(StipAnn)
FROM Impiegato
```

MAX(StipAnn)	MIN(StipAnn)	AVG(StipAnn)
80	36	51

ATTENZIONE

ATTENZIONE = Questa interrogazione è corretta?

```
SELECT Nome,Cognome, MAX(StipAnn)
FROM Impiegato
WHERE Dipart='Amministrazione'
ORDER BY Nome, Cognome
```

NO → gli operatori aggregati **non rappresentano un meccanismo di selezione**, ma solo funzioni che restituiscono un valore quando sono applicate ad un insieme di righe.

***Potrebbe sorgere l'esigenza di applicare l'operatore aggregato a sotto-insiemi di righe...in questi casi si utilizza la clausola* GROUP BY,** che permette di specificare come dividere la tabella in sotto-insiemi aventi caratteristiche comuni.

Interrogazioni con raggruppamento

GROUP BY *listaAttributi*

- Abbiamo caratterizzato gli operatori aggregati come operatori che vengono applicati su tutte le righe che vengono prodotte come risultato dell'interrogazione.
- Spesso sorge l'esigenza di applicare l'operatore aggregato separatamente a sottoinsiemi di righe.
- SQL mette a disposizione la clausola **GROUP BY**, che permette di specificare come dividere la tabella in sottoinsiemi.

GROUP BY

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre la somma degli stipendi degli impiegati che lavorano nello stesso dipartimento**

```
SELECT Dipart, SUM(StipAnn)
FROM Impiegato
GROUP BY Dipart
```

Dipart	SUM(StipAnn)
Amministrazione	125
Produzione	82
Distribuzione	45
Direzione	153

GROUP BY – come gestirla 1\2

- La clausola **GROUP BY** ammette come argomento un insieme di attributi, e **raggruppa le righe** che possiedono gli stessi valori per questo insieme di attributi.
  - Se nella **SELECT** è presente un operatore aggregato, solo gli attributi contenuti nella GROUP BY potranno eventualmente comparire nella **SELECT** stessa (insieme all'operatore aggregato)

ESEMPIO **: Estrarre la somma degli stipendi degli impiegati che lavorano nello stesso dipartimento**

```
SELECT Dipart, SUM(StipAnn)
FROM Impiegato
GROUP BY Dipart
```

1) L'interrogazione viene eseguita come se la clausola GROUP BY e l'operatore aggregato non esistessero. Nel caso dell'esempio in esame la prima interrogazione asSUMerebbe la forma seguente :

```
SELECT Dipart, StipAnn
FROM Impiegato
```

Dipart	StipAnn
Amministrazione	45
Produzione	36
Amministrazione	40
Distribuzione	45
Direzione	80
Direzione	73
Amministrazione	40
Produzione	46

GROUP BY – come gestirla 2\2

2) La tabella ottenuta viene poi analizzata, dividendo le righe in insiemi caratterizzati dallo stesso valore degli attributi che compaiono come argomento nella clausola GROUP BY. Nell'esempio le righe vengono raggruppate in base allo stesso valore dell'attributo *Dipart*.

```
SELECT Dipart, StipAnn
FROM Impiegato
GROUP BY Dipart
```

Dipart	StipAnn
Amministrazione	45
Amministrazione	40
Amministrazione	40
Produzione	46
Produzione	36
Direzione	80
Direzione	73
Distribuzione	45

3) Dopo che le righe sono state raggruppate in sotto-insiemi, l'operatore aggregato viene applicato separatamente su ogni sotto-insieme. Il risultato dell'interrogazione è costituito da una tabella con righe che contengono l'esito della valutazione dell'operatore aggregato affiancato al valore dell'attributo che è stato usato per l'aggregazione.

```
SELECT Dipart, SUM(StipAnn)
FROM Impiegato
GROUP BY Dipart
```

Dipart	SUM(StipAnn)
Amministrazione	125
Produzione	82
Distribuzione	45
Direzione	153

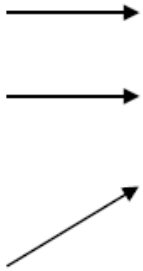
GROUP BY e operatori aggregati

Il numero di figli di ciascun padre

```
select padre, count(*) as NumFigli
from   paternita
group by padre
```

paternita

padre	figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo



padre	NumFigli
Sergio	1
Luigi	2
Franco	2

GROUP BY e operatori aggregati

In una interrogazione che fa uso di **`group by`**, possono comparire nella target list (oltre a funzioni di aggregazione) **solamente** attributi che compaiono nella **`group by`**.

Esempio:

Scorretta: redditi delle persone, raggruppati per età.

```
select eta, reddito
from persone
group by eta
```

Potrebbero esistere più valori dell'attributo per lo stesso gruppo.

Corretta: media dei redditi delle persone, raggruppati per età.

```
select eta, avg(reddito)
from persone
group by eta
```

GROUP BY con più attributi

```
SELECT Dipart, StipAnn, count(StipAnn)
FROM Impiegato
GROUP BY Dipart, StipAnn
```

Dipart	StipAnn	count(StipAnn)
Amministrazione	45	1
Amministrazione	40	2
Produzione	46	1
Produzione	36	1
Direzione	80	1
Direzione	73	1
Distribuzione	45	1

Dipart	StipAnn
Amministrazione	45
Amministrazione	40
Amministrazione	40
Produzione	46
Produzione	36
Direzione	80
Direzione	73
Distribuzione	45

HAVING – condizioni sui gruppi

Si possono anche imporre le condizioni di **selezione sui gruppi**. La selezione sui gruppi è **ovviamente diversa** dalla condizione che seleziona le tuple che devono formare i gruppi (clausola **`where`**). Per effettuare la selezione sui gruppi si usa la clausola **`having`**, che deve apparire dopo la "**`group by`**"

Esempio: i padri i cui figli hanno un reddito medio maggiore di 25.

```
select padre, avg(f.reddito)
from   persone f join paternita
       on figlio = nome
group by padre
having avg(f.reddito) > 25
```

Clausola HAVING

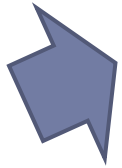
Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre i dipartimenti che spendono più di 100mila euro in stipendi**



Dipart	Sommastipendi
Amministrazione	125
Direzione	153

```
SELECT Dipart,SUM(StipAnn) AS Sommastipendi
FROM Impiegato
GROUP BY Dipart
HAVING SUM(StipAnn)>100
```

HAVING = condizione di selezione sui gruppi. Ogni sotto-insieme di righe costruito dalla **GROUP BY** fa parte del risultato dell'interrogazione solo se l'argomento – della **HAVING** risulta soddisfatto.

Clausola HAVING

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre i dipartimenti per cui la media degli stipendi degli impiegati che si chiamano "Rossi" è superiore a 25 mila euro**

```
SELECT Dipart
FROM Impiegato
WHERE Cognome='Rossi'
GROUP BY Dipart
HAVING AVG(StipAnn)>25
```

Dipart
Amministrazione
Direzione

E' preferibile che **solo gli operatori aggregati** siano usati come argomento della clausola ***HAVING***.
Le condizioni sugli attributi dovrebbero essere posti nella clausola ***WHERE.***

Sintassi Completa di un'interrogazione SQL

SELECT [**DISTINCT**] *lista-select*
FROM *lista-from*
[**WHERE** *condizione*]
[**GROUP BY** *lista gruppo*]
[**HAVING** *qualificazione gruppo*]
[**ORDER BY** *AttrDiOrdinamento*]

Esercizio

- Si supponga di disporre di una tabella **Persone** e di una tabella **Paternità**. Sia l'attributo **padre** che l'attributo **figlio** sono legati da un vincolo di *foreign key* verso **Persone.nome**.

- Mostrare in SQL i padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 20.

Persone

nome	età	reddito

Paternità

padre	figlio

Soluzione Esercizio

Persone

nome	età	reddito

Paternità

padre	figlio

```
SELECT padre
FROM Persone f JOIN paternità ON figlio=nome
WHERE f.età < 30
GROUP BY padre
HAVING AVG(f.reddito) > 20
```

Unione

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre i nomi ed i cognomi degli impiegati in una tabella con un solo attributo**

```
SELECT Nome
FROM Impiegato
UNION
SELECT Cognome
FROM Impiegato
```

Se si vogliono mantenere i duplicati si utilizza **UNION ALL**.

Nome
Mario
Carlo
...
Rossi
Bianchi
...

Contiene la lista di tutti i nomi più tutti i cognomi (senza i duplicati). Infatti, di default, **gli operatori insiemistici eliminano i duplicati**.

Intersezione

Impiegato

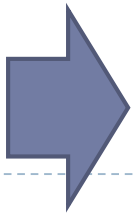
Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

ESERCIZIO **:Estrarre i nomi degli impiegati che sono anche cognomi**

```
SELECT Nome
FROM Impiegato
INTERSECT
SELECT Cognome
FROM Impiegato
```



Nome
Franco

L'**Intersezione** insiemistica **non è supportata nativamente da *molti DBMS***...ma è facilmente ottenibile tramite interrogazioni nidificate...*dettagli in futuro!*

Differenza

Impiegato

Nome	Cognome	Dipart	StipAnn
Mario	Rossi	Amministrazione	45
Carlo	Bianchi	Produzione	36
Giuseppe	Verdi	Amministrazione	40
Franco	Neri	Distribuzione	45
Carlo	Rossi	Direzione	80
Lorenzo	Gialli	Direzione	73
Paola	Rosati	Amministrazione	40
Marco	Franco	Produzione	46

Dipartimento

Nome	Città
Amministrazione	Milano
Produzione	Torino
Distribuzione	Roma
Direzione	Milano
Ricerca	Milano

La **Differenza** insiemistica **non è supportata nativamente da molti DBMS**...ma è facilmente ottenibile tramite interrogazioni nidificate...*dettagli in futuro!*

ESERCIZIO **:Estrarre i nomi degli Impiegati che non sono Cognomi per qualche impiegato**

```
SELECT Nome
FROM Impiegato
EXCEPT
SELECT Cognome
FROM Impiegato
```

Nome
Mario
Carlo
Giuseppe
Lorenzo
Paola
Marco

Conclusioni

- La nascita di SQL ha rappresentato un fattore importante nel rapido sviluppo del modello relazionale.
- Relazionalmente completo; di fatto, possiede un potere espressivo significativamente superiore all'algebra relazionale.
- Esistono molti modi alternativi di scrivere un'interrogazione; l'ottimizzatore dovrebbe cercare il piano di valutazione più efficiente.
- Nella pratica, gli utenti devono essere consci di come le interrogazioni sono ottimizzate e valutate per ottenere risultati migliori.