SQL

Università degli Studi di Salerno

Corso di Laurea in Scienze della Comunicazione

Informatica generale (matr. Dispari)

Docente: Angela Peduto

A.A. 2005/2006

SQL

SQL (pronunciato anche come l'inglese *sequel*) è l'acronimo di Structured Query Language
(*linguaggio di interrogazione strutturato*)

E' un linguaggio completo che presenta anche le proprietà di:
DDL (Data Definition Language)
DML (Data Manipulation Language)

Con SQL è quindi possibile:

- definire schemi di basi di dati
- eseguire query
- modificare il contenuto della base di dati

SQL

1986 Prima standardizzazione

1989 SQL-89

1992 SQL-2 (SQL-92): versione attualmente diffusa (e tuttora non completamente implementata)

1998 SQL-3 (SQL-99): nuovo standard proposto con funzionalità avanzate (DB a oggetti, operazioni ricorsive ecc.)

3 implementazioni disponibili:

- Entry SQL
- Intermediate SQL
- Full SQL

Definizione dei dati

Esistono 6 domini elementari:

- Character
- Bit
- Tipi numerici esatti
- Tipi numerici approssimati
- Data e ora
- Intervalli temporali

Definizione tabelle

Una tabella SQL è costituita da una collezione ordinata di attributi e da un insieme di vincoli

```
create table NomeTabella
( NomeAttributo Dominio [ValDefault] [Vincoli]
{,NomeAttributo Dominio [ValDefault] [Vincoli]
}
 AltriVincoli
)
```

Una tabella è inizialmente vuota e chi la crea possiede tutti i diritti su di essa

Definizione Tabelle

Es.

```
 create table Dipartimento
(
  Nome           char(20) primary key,
  Indirizzo char(50),
  Città          char(20)
)
```

Vincoli intrarelazionali

I più semplici vincoli intrarelazionali predefiniti sono

- **`not null`** indica che il valore nullo non è ammesso su uno specifico attributo. Quindi richiede che sia inserito un valore, salvo che non sia già definito un valore di default.
  SQL non distingue i diversi tipi di valore nullo. Se è necessario farlo vanno definiti opportuni domini.
- **`unique (`***Attributo* {, *Attributo*}**`)`** indica che l'insieme di attributi deve essere una superchiave per la tabella.
- **`primary key (`***Attributo* {, *Attributo*}**`)`** definisce la chiave primaria. Può essere una sola. Tutti gli attributi sono **`not null.`**

Vincoli intrarelazionali

Es.

```
Nome        character(20) not null,
Cognome     character(20) not null,
unique (Nome, Cognome)
```

impone che non ci sia una riga con nome e cognome uguali

```
Nome        character(20) not null unique,
Cognome     character(20) not null unique
```

impone che sia nome che cognome siano diversi in tutte le righe

Vincoli interrelazionali

Vincoli di integrità referenziale

In SQL si usa il vincolo di foreign key (*chiave esterna*) per creare un legame fra i valori di un attributo della tabella corrente (*interna*) e un attributo di un'altra tabella (*esterna*). Si impone che per ogni riga della tabella interna il valore dell'attributo sia presente nel corrispondente attributo della tabella esterna.

L'attributo della tabella esterna deve essere **`unique.`**

Se l'attributo è unico allora si usa **`references.`**

Altrimenti si usa **`foreign key.`**

Vincoli interrelazionali

Es.

```
create table Impiegato
(
   Matricola   character(6) primary key,
   Nome character(20) not null,
   Cognome     character(20) not null,
   Dipart      character(15)
               references Dipartimento(NomeDip),
   Stipendio   numeric(9) default 0,
   unique (Cognome, Nome),
   foreign key(Nome, Cognome)
               references Anagrafica(Nome,Cognome)
)
```

Interrogazioni

- Le interrogazioni in SQL sono formulate in modo *dichiarativo* specificando cioè *cosa* si vuole ottenere e non *come* lo si vuole ottenere.
- L'interrogazione viene passata *all'ottimizzatore di interrogazioni (query optimizer)* che fa parte del DBMS. Questo la analizza e la traduce nel linguaggio di interrogazione interno al DBMS.
- Per questo chi programma in SQL deve cercare di scrivere codice *leggibile* e *facilmente modificabile*, piuttosto che efficiente.

Interrogazioni

L'istruzione base per le interrogazioni è **`select`**

`select` *ListaAttributi* *(target list)*
`from` *ListaTabelle* *(clausola* **`from`***)*
`[ where` *Condizione* **`]`** *(clausola* **`where`***)*

Più in dettaglio:

`select` *AttrEspr* **`[[as]`** *Alias***`]{,`** *AttrEspr* **`[[as]`** *Alias***`]}**
`from` *Tabella* **`[[as]`** *Alias***`]{,`** *Tabella* **`[[as]`** *Alias***`]}**
`[ where` *Condizione* **`]`**

Seleziona le righe che soddisfano la condizione **`where`** fra quelle appartenenti al prodotto cartesiano delle tabelle in *ListaTabelle.*
Ogni colonna (tabella) può essere ridenominata con un alias.

Esempio con assenza where

`select` *Età, Nome*
`from` *Persone*

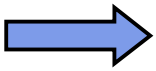
	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0

	Età	Nome
	25	Aldo
	27	Andrea
	50	Anna
	26	Filippo
	60	Franco
	50	Luigi
	75	Luisa
	55	Maria
	30	Olga
	85	Sergio
▶	0	

Esempio con assegnazione diverso (alias) ad un campo

SELECT Età, Reddito, Nome AS Dipendente
FROM Persone
WHERE Età <30

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0



	Età	Reddito	Dipendente
▶	25	15.000.000	Aldo
	27	21.000.000	Andrea
	26	0	Filippo
*	0	0	

Esempio con where

Data una base di dati che contiene le tabelle:
IMPIEGATO(Nome, Cognome, Dipart, Ufficio, Stipendio, Città)
DIPARTIMENTO(Nome, Indirizzo,Città)

```
select Stipendio as SalarioMensile
from       Impiegato
where Cognome = `Rossi`
```

Il risultato è una tabella con una colonna rinominata *SalarioMensile* e tante righe quanti sono gli impiegati che si chiamano Rossi.

Se si usa ***** dopo **`select`** si selezionano tutti gli attributi

Maternità

Madre	Figlio
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Paternità

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Persone

Nome	Età	Reddito
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	30
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Selezione e proiezione

Nome e reddito delle persone con meno di trenta anni

$$PROJ_{Nome,\ Reddito}(SEL_{Eta<30}(Persone))$$

```
select nome, reddito
from persone
where eta < 30
```

```
select p.nome as nome, p.reddito as reddito
from persone p
where p.eta < 30
```


SELECT *Età*, ***Reddito***, *Nome*
FROM ***Persone***
WHERE ***Età <30***

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

Età	Reddito	Nome
25	15.000.000	Aldo
27	21.000.000	Andrea
26	0	Filippo
0	0	

Nota: ordine delle colonne

Clausola `where`

Ammette come argomento una condizione logica.

Gli operatori ammessi per i predicati semplici (confronto attributo-costante o attributo-espressione) sono

`=, <>, <, >, <=, >=`

I predicati semplici possono essere modificati tramite gli operatori logici **`and, or, not.`**

`not` ha precedenza su **`and`** e **`or`**, ma non è definita la precedenza fra **`and`** e **`or.`** Quando si coordinano più predicati con **`and`** e **`or`** è bene esplicitare le precedenze con le parentesi.

```
select Nome
from       Impiegato
where      Cognome = 'Rossi' and
    (Dipart = 'Amministraz' or Dipart = 'Produz')
```


Es. con uso di Wild character

SELECT *******
FROM Persone
WHERE ***Età <30***

WILDCARD
Equivale ad elencare i nomi dei campi nell'ordine standard

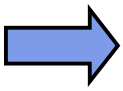
	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Filippo	26	0
▶		0	0

Es. clusola where con condizione composta

SELECT *******
FROM ***Persone***
WHERE ***(Età >= 20) AND (Età <=30)***

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0

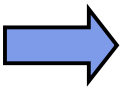


	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Filippo	26	0
	Olga	30	25.000.000
▶		0	0

Es. clusola where con condizione composta between…and

SELECT *******
FROM ***Persone***
WHERE ***Età Between 20 AND 30***

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0



	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Filippo	26	0
	Olga	30	25.000.000
▶		0	0

Operatore `like`

Per i confronti fra stringhe è definito anche l'operatore **`like.`**

Il confronto è effettuato con una stringa che può contenere i caratteri speciali **%** e **_ .**

_ rappresenta un carattere arbitrario

% rappresenta in numero arbitrario di caratteri (anche zero).

Es.

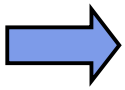
```
select *
from Impiegato
where Cognome like '_o%i'
```

La condizione è soddisfatta da Rossi Borroni Poli Pollastri ecc.

Es. ricerca parziale (like)

SELECT *******
FROM ***Persone***
WHERE ***Nome LIKE "A*"***

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0



Selezione delle persone
Che hanno il nome che
Comincia con la A

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
▶		0	0

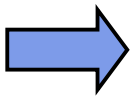
Condizioni di ricerca parziale (like)

- seleziona le tuple in cui il valore è recuperato con le wildcards. Funziona solo su attributi stringa
  - ***** sequenza qualunque lunga
    - **no*** nome, note e notare
  - **?** esattamente un carattere
    - **B?llo** ballo, bello e bollo
  - **[ ]** qualsiasi singolo carattere all'interno delle parentesi.
    - **B[ae]llo** ballo e bello, ma non bollo
  - **!** qualsiasi carattere non incluso nelle parentesi quadre.
    - **B[!ae]llo** bollo e bullo, ma non ballo e bello
  - **-** uno qualsiasi dei caratteri di un intervallo.
    - **b[a-c]d** bad, bbd e bcd
  - **#** qualsiasi singolo carattere numerico.
    - **1#3** 103, 113, 123

Nome, Età e Reddito di chi ha il nome che inizia per 'A' con una 'd' come terzo carattere

SELECT *******
FROM ***Persone***
WHERE ***Nome LIKE "A?d*"***

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0

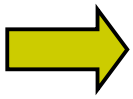


	Nome	Età	Reddito
▶	Aldo	25	15.000.000
	Andrea	27	21.000.000
*		0	0

Es. chi ha il nome che è lungo 4 caratteri

SELECT *******
FROM ***Persone***
WHERE ***Nome LIKE "????"***

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0

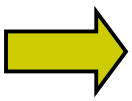


	Nome	Età	Reddito
▶	Aldo	25	15.000.000
	Anna	50	35.000.000
	Olga	30	25.000.000
*		0	0

Es. chi ha il nome che termina con una 'a'

SELECT *******
FROM ***Persone***
WHERE ***Nome LIKE "*a"***

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0



	Nome	Età	Reddito
▶	Andrea	27	21.000.000
	Anna	50	35.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
*		0	0

Ricerca in un sottoinsieme (IN)

SELECT ***Figlio***
FROM ***Paternità***
WHERE ***Padre IN ("Sergio","Luigi", "Nicola")***

Ricerca del nome dei figli di Sergio Luigi e Nicola

Figlio	Padre
Aldo	Franco
Andrea	Franco
Filippo	Luigi
Olga	Luigi
Franco	Sergio

Figlio
Filippo
Franco
Olga

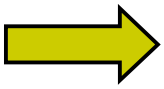
Funzioni su tuple

SELECT può presentare come argomenti anche FUNZIONI definite su attributi o su costanti

SELECT ***Nome, Reddito/1000000 AS Stipendio_milioni***
FROM ***Persone***

Nota: reddito espresso in milioni

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



Nome	Stipendio_milioni
Aldo	
Andrea	
Anna	
Filippo	
Franco	
Luigi	
Luisa	
Maria	
Olga	
Sergio	

Operandi costanti (inserimento di campi a valore costante)

SELECT ***Nome,*** ***"guadagna" AS Guadagna,***
Reddito/1000000 AS Stipendio, "milioni" AS Milioni
FROM ***Persone***

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

Nome	Guadagna	Stipendio	Milioni
Aldo	guadagna	15	milioni
Andrea	guadagna	21	milioni
Anna	guadagna	35	milioni
Filippo	guadagna	0	milioni
Franco	guadagna	20	milioni
Luigi	guadagna	40	milioni
Luisa	guadagna	87	milioni
Maria	guadagna	42	milioni
Olga	guadagna	25	milioni
Sergio	guadagna	35	milioni

Operatori aggregati

In algebra relazionale le espressioni vengono valutate sulle singole tuple in successione. Talvolta però possono essere necessarie informazioni derivabili dall'esame di tutte le tuple o di più tuple contemporaneamente.

SQL prevede una serie di operatori aggregati:

`count, sum, max, min, avg`

con sintassi

`count` (< **`* | [distinct | all]`** *ListaAttributi* >)
< **`sum|max|min|avg`** >(**`[distinct | all]`** *AttrEspr*)

Es. Determinare il numero degli impiegati che si chiamano Rossi

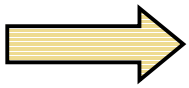
```
select count(*)
from Impiegato
where nome= 'Rossi'
```

Funzioni aggregative (COUNT)

SELECT ***COUNT (*) AS Totale***
FROM ***Persone***
Where ***Età Between 20 AND 30***

Quante persone hanno tra i 20 ed i 30 anni?

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0



	Totale
▶	4

Duplicati

L'algebra relazionale non ammette duplicati, SQL li ammette.
Quindi

```
select Città
from Persona
where Cognome= 'Rossi'
```

estrae una lista di città in cui una città può comparire più volte.
Per evitare i duplicati SQL prevede la parola chiave **`distinct`** da inserire subito dopo **`select.`**

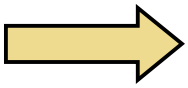
```
select distinct Città
from Persona
where Cognome= 'Rossi'
```

Funzioni aggregative (COUNT)

SELECT COUNT (Padre) AS Padri
FROM Paternità

Quanti sono i padri?

Figlio	Padre
Aldo	Franco
Andrea	Franco
Filippo	Luigi
Olga	Luigi
Franco	Sergio



Padri
5

Per conoscere il numero di padri senza ripetizione va utilizzata la clausola **DISTINCT**

SELECT COUNT (DISTINCT Padre) AS Padri
FROM Paternità

In Access tale istruzione non funziona

Funzioni aggregative (AVG – Media)

SELECT AVG (Reddito) AS Reddito_medio
FROM Persone
Where Età Between 20 AND 30

Quale è il reddito medio delle Persone di età compresa Tra i 20 e 30 anni

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

Reddito_medio
15250000

Funzioni aggregative (MIN)

SELECT ***MIN (Reddito) AS Reddito_minimo***
FROM ***Persone***
Where ***Età Between 20 AND 30***

Qual è reddito Minimo delle persone di età compresa fra 20 e 30 anni?

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

Reddito_minimo
0

Ordinamento

E' possibile anche ordinare le righe del risultato di una interrogazione attraverso la clausola **`order by`**, a chiusura di una interrogazione.

`order by` *AttrdiOrdinamento* [**`asc`** | **`desc`**]
`{,` *AttrdiOrdinamento* [**`asc`** | **`desc`**]`}`
`asc` (default) indica ordinamento ascendente, **`desc`** discendente.

Il primo attributo ha priorità, a parità di valore si usa il secondo ecc.

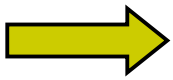
```
select *
      from Persona
      order by Cognome, Nome
```

Ordinamento della tabella risposta (ORDER BY)

SELECT *******
FROM ***Persone ORDER BY Reddito***

Ordina le persone in base al reddito

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



Nome	Età	
Filippo	26	
Aldo	25	
Franco	60	
Andrea	27	
Olga	30	
Sergio	85	
Anna	50	
Luigi	50	
Maria	55	
Luisa	75	
	0	

Ordinamento della tabella risposta in ordine decrescente (ORDER BY DESC)

Ordina le persone in base al reddito in ordine decrescente

SELECT *******
FROM ***Persone ORDER BY Reddito DESC***

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

Nome	Età	
Luisa	75	
Maria	55	
Luigi	50	
Sergio	85	
Anna	50	
Olga	30	
Andrea	27	
Franco	60	
Aldo	25	
Filippo	26	
	0	

Interrogazioni su più tabelle

Se si vogliono estrarre informazioni da più tabelle, si pone come argomento della clausola **`from`** una lista delle tabelle.

Se si deve formulare un join, è possibile farlo esplicitando il collegamento fra le due tabelle nella clausola **`where`**.

Es.

Estrarre i nomi degli impiegati e le città dove lavorano.

```
select    Padre
from      Paternità, Persone
where     Paternità.Figlio = Persone.Nome
```

Nota: Per indicare gli attributi di ogni relazione si usa la notazione
<Nome Relazione>.<attributo>

Persone (**Nome**, Età, Reddito)

Paternità (**Figlio**, Padre)

Maternità (**Figlio**, Madre)

Persone

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

Paternità

Figlio	Padre
Aldo	Franco
Andrea	Franco
Filippo	Luigi
Olga	Luigi
Franco	Sergio

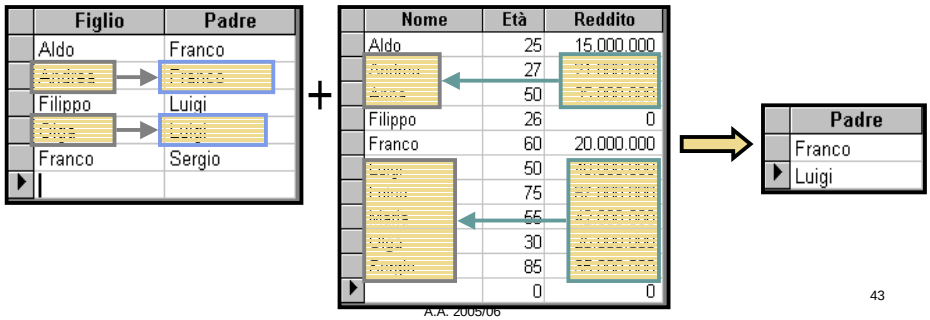
Maternità

Figlio	Madre
Filippo	Anna
Olga	Anna
Luigi	Luisa
Maria	Luisa
Andrea	Maria

Esempio

SELECT ***Padre***
FROM ***Paternità, Persone***
WHERE ***Persone.Reddito>20000000 AND***
Paternità.Figlio=Persone.Nome

Individua i padri delle persone che guadagnano più di 20 Milioni



Interrogazioni su più tabelle

E' possibile omettere il nome della tabella per quegli attributi che non presentano ambiguità.

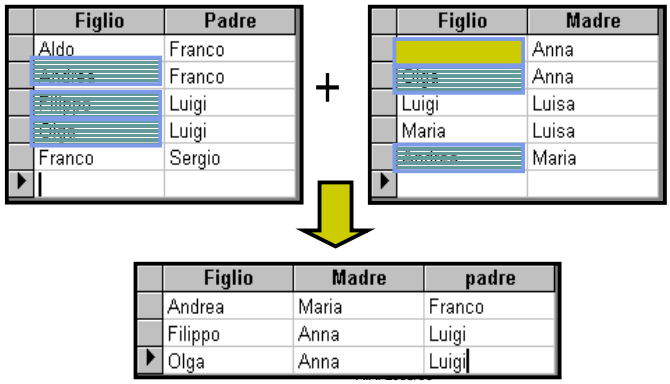
```
select    Maternità.Figlio, Madre,
          Padre
from      Paternità, Maternità
where     Paternità.Figlio = Maternità.Figlio
```

e abbreviare ulteriormente il codice utilizzando gli alias

```
select    M2.Figlio, Madre, Padre
from      Paternità as M1, Maternità as M2
where      M1.Figlio = M2.Figlio
```

Esempio

SELECT ***Maternità.Figlio, Madre, Padre***
FROM ***Paternità, Maternità***
WHERE ***Paternità.Figlio=Maternità.Figlio;***



Figlio	Padre
Aldo	Franco
Andrea	Franco
Filippo	Luigi
Olga	Luigi
Franco	Sergio

+

Figlio	Madre
	Anna
Olga	Anna
Luigi	Luisa
Maria	Luisa
Andrea	Maria

Figlio	Madre	padre
Andrea	Maria	Franco
Filippo	Anna	Luigi
Olga	Anna	Luigi

Variabili tupla (Query su unica tabella)

L'uso degli alias consente di:

- compattare il codice
- fare riferimento a più esemplari della stessa tabella
- creare interrogazioni nidificate

Se una tabella compare una sola volta non c'è differenza fra variabile ed alias.

Se compare più volte si parla più propriamente di variabile.

- In questo caso la query corrisponde ad un **confronto da effettuare all'interno della stessa tabella**.
- Dobbiamo assegnare **due nomi (ALIAS) differenti alla stessa tabella**, in modo da poter confrontare elementi appartenenti alla stessa tabella come se fossero due tabelle.

Es. variabili tupla (Query su unica tabella)

Seleziona Nome e Reddito di chi guadagna più di Luigi

```
SELECT   P2.Nome,      P2.Reddito
FROM     Persone  As P1,    Persone As P2
WHERE    P1.Nome = 'Luigi'     AND
          P2.Reddito > P1.Reddito
```

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

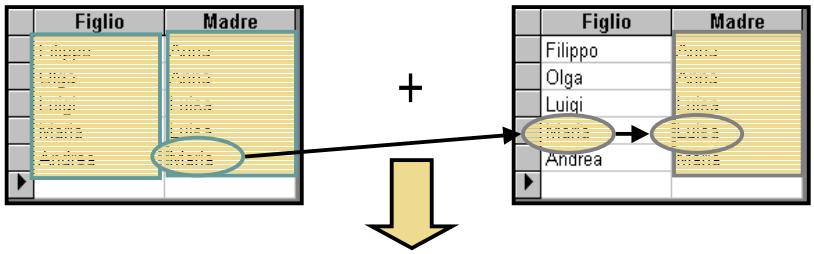
Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

NOME	REDDITO
Luisa	87.000.000
Maria	42.000.000

Esempio di query con variabile di tupla

```
SELECT   m1.figlio,   m1.madre,   m2.madre AS Nonna
FROM     Maternità  M1,    Maternità M2
WHERE    m1.Madre=m2.Figlio
```

Nome, Nome della madre e Nome della nonna di ogni persona presente nel DB



figlio	madre	Nonna
Andrea	Maria	Luisa

Esempio

```
SELECT   figli.Nome AS Figli,   figli.reddito,
         padri.nome AS Padri, padri.reddito
FROM     Persone figli, Persone padri, Paternità
WHERE    paternità.figlio = figli.nome     AND
         paternità.padre = padri.nome          AND
         figli.Reddito < padri.reddito
```

Seleziona tutti coloro che guadagnano meno dei padri

Figlio	Padre
Aldo	Franco
Andrea	Franco
Filippo	Luigi
Olga	Luigi
Franco	Sergio

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

Figli	Figli.reddito	Padri	Padri.reddito
Aldo	15.000.000	Franco	20.000.000
Filippo	0	Luigi	40.000.000
Franco	20.000.000	Sergio	35.000.000
Olga	25.000.000	Luigi	40.000.000

49

Interrogazioni con raggruppamento

- E' possibile **ripartire le tuple in gruppi omogenei ed applicare gli operatori aggregati ai singoli gruppi**, aggiungendo in coda alla SELECT-FROM-WHERE la clausola GROUP BY

  GROUP BY $A_1,\ldots A_k$

  HAVING Ψ
- Ψ è una clausola opzionale che si applica ai gruppi $A_1,\ldots A_k$ (per selezionare solo un sottoinsieme dei gruppi)
- In tal modo **due tuple si trovano nello stesso gruppo se e solo se esse sono uguali componente per componente sugli attributi $A_1,\ldots A_k$**
- Gli attributi $A_1,\ldots A_k$ devono comparire anche nella clausola SELECT

Interrogazioni con raggruppamento

Gli operatori aggregati vengono applicati a tutte le righe che vengono prodotte come risultato dell'operazione.

Può essere necessario applicare l'operatore solo ad un sottoinsieme delle righe.

SQL non ammette che nella stessa target list compaiano funzioni aggregate ed espressioni a livello di riga, come il nome di un attributo.

L'operatore **`group by`** specifica come suddividere le tabelle in sottoinsiemi.

Es.

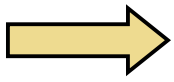
```
SELECT    Età, AVG(Reddito) AS Reddito_medio_età
FROM      Persone
GROUP BY Età
```


Esempio GROUP BY

SELECT Età, AVG(Reddito) AS Reddito_medio_età
FROM Persone
GROUP BY Età

Stipendio medio dei dipendenti, raggruppati per età

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	...	...
Filippo	26	0
Franco	60	20.000.000
Luigi	...	...
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



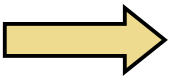
Età	Reddito_medio_età
25	15000000
26	0
27	21000000
30	25000000
...	...
55	42000000
60	20000000
75	87000000
85	35000000

Esempio GROUP BY

SELECT ***Età, AVG(Reddito) AS 'Reddito Medio >45'***
FROM ***Persone***
WHERE ***Età > 45*** ***GROUP BY Età***

Stipendio medio dei dipendenti di età superiore a 45 anni, raggruppati per età

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



Età	'Reddito Medio >45'
50	37500000
55	42000000
60	20000000
75	87000000
85	35000000

Operatori aggregati e raggruppamenti

- 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

Condizioni sui gruppi

Può essere anche necessario restringere i gruppi attraverso l'applicazione di condizioni.

Se le condizioni sono verificabili a livello delle singole righe, basta utilizzare la clausola **`where`**, altrimenti si aggiunge una condizione alla **`group by`** attraverso l'estensione **`having`**

```
SELECT    Età, AVG(Reddito) AS 'Stipendio Medio'
FROM      Persone
          GROUP BY Età  HAVING  Count(Età)>1
```

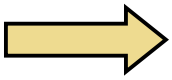
Se non si specifica **`group by`** e si usa **`having`** da solo la condizione è applicata a tutte le righe. Il problema è che se la condizione non è verificata, il risultato sarà vuoto.

Esempio Raggruppamento - HAVING

SELECT Età, AVG(Reddito) AS 'Stipendio Medio'
FROM Persone
GROUP BY Età HAVING Count(Età)>1

Età e Stipendio medio dei dipendenti raggruppati per età, con almeno 2 dipendenti per gruppo

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0



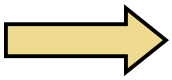
	Età	Stipendio_Medio
▶	50	37500000

Esempio Raggruppamento - HAVING

SELECT Età, AVG(Reddito) AS Reddito_Medio
FROM Persone
GROUP BY Età
HAVING AVG(Reddito)>25000000

Stipendi medi dei dipendenti che guadagnano più di 25.000.000 euro, raggruppati per età

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



Età	Stipendio_Medi
50	37500000
55	42000000
75	87000000
85	35000000

WHERE o HAVING?

- I padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 20

```
select padre, avg(f.reddito)
from persone f join paternita on
                        figlio = nome
where eta < 30
group by padre
having avg(f.reddito) > 25
```