SQL

Università degli Studi di Salerno

Corso di Laurea in Scienze della Comunicazione

Informatica generale (matr. Dispari)

Docente: Angela Peduto

A.A. 2007/2008

DDL e DML

La definizione di una base di dati comporta due aspetti:

- *Definizione dei dati*
- *Definizione delle interrogazioni sui dati stessi*

I DBMS usano quindi due tipi di linguaggi diversi per definire e manipolare i dati

- Data Definition Languages **(DDL):** *linguaggi di definizione di dati, servono per definire le tabelle e le autorizzazioni per l'accesso (riferendosi ad un modello relazionale)*
- Data Manipulation Languages **(DML):** *linguaggi per la manipolazione di dati, servono per esprimere le interrogazioni che permettono di accedere ai dati per ottenere delle informazioni*

Spesso DDL e DML coesistono nello stesso linguaggio (ad esempio in SQL).

Introduzione a SQL

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

E' un linguaggio completo contenendo al suo interno funzioni 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

Un po' di storia…

- SQL (Structured Query Language) era originariamente il linguaggio di interrogazione di *System R*, Database Management System (DBMS) relazionale sviluppato da IBM negli anni Settanta.
- Successivamente è stato adottato da molti altri sistemi, è stato standardizzato ed è diventato il linguaggio di riferimento per le basi di dati relazionali.
- La diffusione di SQL è dovuta in buona parte alla intensa opera di standardizzazione dedicata a questo linguaggio, svolta principalmente dall'American National Standard Institute (ANSI), l'organismo nazionale statunitense degli standard, e dall'International Organization for Standardization (ISO), l'organismo internazionale che raggruppa i vari organismi di standard nazionali.

...

- Gran parte dei produttori del settore hanno avuto modo di partecipare al processo decisionale, senza che un particolare produttore assumesse posizione dominante.
- Il processo di standardizzazione ha avuto inizio nella prima metà degli anni Ottanta e continua tuttora. Sono state così prodotte nel tempo diverse versioni dello standard del linguaggio, caratterizzate da una crescente completezza e sofisticazione.

Standardizzazione del SQL

Il processo di standardizzazione ha avuto inizio nella prima metà degli anni Ottanta e continua tuttora

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 (simile ad SQL-2)
- Intermediate SQL (contiene caratteristiche importanti per rispondere al mercato)
- Full SQL (lo standard nella sua interezza)

I domini elementari

- In una base di dati è necessario definire il tipo dei dati che vi vengono memorizzati, ad esempio se si vuole memorizzare il nome di una persona sarà opportuno usare un tipo di dato testuale, mentre se si vuole memorizzare l'altezza di una montagna espressa in metri sarà opportuno usare un dato di tipo numerico.
- Ad ogni attributo di ogni tabella viene quindi associato un dominio, che specifica qual è l'insieme dei valori ammissibili per quell'attributo.
- Il DDL mette a disposizione alcuni domini elementari.

Definizione dei dati

Esistono 6 domini elementari, a partire dai quali si possono definire i domini da associare agli attributi dello schema:

- Caratteri (per rappresentare singoli caratteri o stringhe)
- Bit (per rappresentare singoli caratteri binari o stringhe di bit)
- Tipi numerici esatti (per rappresentare interi o numeri frazionari con parte decimale di lunghezza prefissata)
- Tipi numerici approssimati (per rappresentare valori reali)
- Data e ora
- Intervalli temporali (per rappresentare intervalli di tempo come ad esempio la durata di un evento)

Definizione tabelle

Una convenzione che adotteremo da questo momento è quella di usare la posto di “relazione” il termine ***tabella*** ed al posto di “tupla” il termine ***riga***

Una tabella SQL è costituita da una collezione ordinata di attributi e da un insieme (eventualmente vuoto) di vincoli

La sintassi SQL è:

`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

- Sia nella definizione dei domini che nella definizione delle tabelle è possibile definire dei vincoli, ovvero delle proprietà che devono essere verificate da ogni istanza della base di dati
- Come abbiamo visto nelle scorse lezioni i vincoli di distinguono in:
  - **Intrarelazionali** (che coinvolgono una sola relazione)
  - **Interrelazionali** (che coinvolgono diverse relazioni)

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*}**`)`** impone che i valori di tale attributo siano tutti diversi l'uno dall'altro. Indica che l'attributo o l'insieme di attributi deve essere una superchiave per la tabella, cioè righe differenti della tabella non possono avere gli stessi valori.

- **`primary key (`***Attributo* {, *Attributo*}**`)`** definisce la chiave primaria. Può essere specificato una sola volta per ogni singola tabella. Tutti gli attributi sono che sono definiti come primary key 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`**, tipicamente l'attributo della tabella esterna è la chiave primaria**`.`**

Se si vuol far riferimento ad un unico attributo della tabella esterna 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)
)
```

- Il vincolo impone che l'attributo `Dipart` della tabella `Impiegato` possa assumere solo uno dei valori che le righe della tabella `Dipartimento` possiedono per l'attributo `NomeDip`

SQL come DML: le interrogazioni

- Il Data Manipulation Language (DML) è il linguaggio che consente di manipolare il database offrendo i costrutti necessari a inserire, cancellare, modificare le righe delle tabelle e ad effettuare interrogazioni sul database. Le interrogazioni sono le operazioni che consentono di ricercare i dati all'interno del database, di effettuarci dei calcoli e di restituire il risultato in output.
- 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 ed algebra relazionale

- Dal punto di vista pratico risulta utile combinare gli aspetti dichiarativi del calcolo e quelli procedurali dell'algebra relazionale.
- L'algebra relazionale è un linguaggio procedurale basato su concetti di tipo algebrico, è costituito d un insieme di operatori definiti su relazioni e che producono ancora relazioni come risultati
- Gli operatori che permettono effettivamente di manipolare le relazioni sono:
  1. Selezione
  2. Proiezione
  3. join

Selezione

- Il risultato di una selezione contiene le tuple che soddisfano le condizioni di selezione

Info_città

Città	Regione	Popolazione
Roma	Lazio	3.000.000
Milano	Lombardia	1.500.000
Genova	Liguria	800.000
Pisa	Toscana	150.000

SEL $_{\text{Popolazione}>1.000.000}$(Info_città)

Città	Regione	Popolazione
Roma	Lazio	3.000.000
Milano	Lombardia	1.500.000

Proiezione

- Dati una relazione *r(X)* ed un sottoinsieme Y di X, la proiezione di r su Y è l'insieme di tuple su Y ottenute dalle tuple di *r* considerando solo i valori su Y

Info_città

Città	Regione	Popolazione
Roma	Lazio	3.000.000
Milano	Lombardia	1.500.000
Genova	Liguria	800.000
Pisa	Toscana	150.000

Proj $_{\text{Città, Regione}}$(Info_città)

Città	Regione
Roma	Lazio
Milano	Lombardia
Genova	Liguria
Pisa	Toscana

Join

- E' l'operatore che permette di correlare dati contenuti in relazioni diverse, confrontando i valori contenute in esse e utilizzando quindi la caratteristica fondamentale del modello, quello di essere basato su valori

r1

Impiegato	Reparto
Rossi	Vendite
Neri	Produzione
Bianchi	Produzione

r2

Reparto	Capo
Produzione	Mori
Vendite	Bruni

r1 join *r2*

Impiegato	Reparto	Capo
Rossi	vendite	Bruni
Neri	Produzione	Mori
Bianchi	Produzione	Mori

Interrogazioni SQL

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*

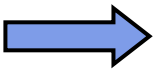
	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0

	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

L'uso del carattere jolly "*"
Equivale ad elencare i nomi dei campi nell'ordine standard

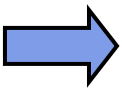
	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0

	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

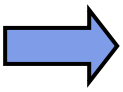


	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Filippo	26	0
	Olga	30	25.000.000
▶		0	0

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'
```

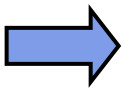
La condizione è soddisfatta da Rossi Borroni Poli Pollastri ecc.

Es. ricerca parziale (like)

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

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
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
	0	0

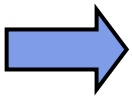
Condizioni di ricerca parziale (like)

- seleziona le righe in cui il valore è recuperato con i caratteri jolly (o wildecards). 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

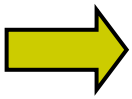


	Nome	Età	Reddito
▶	Aldo	25	15.000.000
	Andrea	27	21.000.000
✳		0	0

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

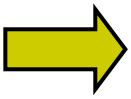


	Nome	Età	Reddito
▶	Aldo	25	15.000.000
	Anna	50	35.000.000
	Olga	30	25.000.000
✳		0	0

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)

Ricerca del nome dei figli di Sergio Luigi e Nicola

SELECT ***Figlio***
FROM ***Paternità***
WHERE ***Padre IN ("Sergio","Luigi", "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	15
Andrea	21
Anna	35
Filippo	0
Franco	20
Luigi	40
Luisa	87
Maria	42
Olga	25
Sergio	35

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

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'
```


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*)

Supponiamo di voler calcolare il numero di studenti i cui dati sono memorizzati nella omonima tabella. Il seguente comando realizza quanto voluto:

```
select count(*)
from Studenti
```

La funzione "**sum**" calcola la somma dell'attributo specificato
"**max**" restituisce il più alto tra tutti i valori dell'attributo specificato
"**min**" il minimo
"**avg**" ne effettua la media.

Vediamo qualche altro esempio:

```
select min(Voto)
from Esami
```

Restituisce il voto più basso fra tutti quelli presenti nella tabella esami, cioè tra tutti gli studenti.

```
select max(Voto)
from Esami
where Studente='3333'
```

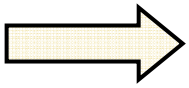
Restituisce il voto più alto ottenuto dallo studente avente matricola uguale a 3333.

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

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

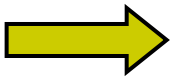
Nome	Età	Reddito
Filippo	26	0
Aldo	25	15.000.000
Franco	60	20.000.000
Andrea	27	21.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
Anna	50	35.000.000
Luigi	50	40.000.000
Maria	55	42.000.000
Luisa	75	87.000.000
	0	0

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

Interrogazioni SQL su più tabelle

- Le interrogazioni viste finora consentono di interrogare solo una tabella alla volta. Per fare delle interrogazioni più significative è necessario però estrarre dati da più tabelle contemporaneamente.
- Ad esempio, nella nostra base di dati è interessante mettere in relazione i dati contenuti nella tabella studenti con quelli della tabella esami, allo scopo di produrre in output tutti i voti conseguiti da ogni studente.
- La seguente interrogazione restituisce tre colonne, nella prime due compaiono i nomi e cognomi di ogni studente tante volte quanti esami ha sostenuto, nella terza il voto conseguito.

```
select NomeStudente, CognomeStudente, Voto
from Studenti, Esami
where Studenti.Matricola=Esami.Studente
```


Interrogazioni SQL su più tabelle (2)

- La clausola "select" dell'interrogazione elenca come di consueto gli attributi che vogliamo produrre in output, con la differenza in questo caso che sono tratti non più da una sola tabella ma da due tabelle diverse (Studenti e Esami), specificate nella clausola "from".
- Osserviamo la clausola "where" della tabella. Affinché i dati delle due tabelle siano posti correttamente in relazione è necessario imporre che il valore dell'attributo Matricola della tabella Studenti sia uguale al corrispondente attributo nella tabella Esami: cioè l'attributo Studente. In questo modo i dati di ogni studente saranno associati solo ai dati dei propri voti conseguiti.
- L'operazione eseguita nella interrogazione precedente prende il nome di "join". Una sintassi alternativa che produce lo stesso risultato è:

```
select NomeStudente, CognomeStudente, Voto
from Studenti join Esami
      on Studenti.Matricola=Esami.Studente
```

Interrogazioni SQL su più tabelle (2)

- Seguendo la stessa tecnica è possibile eseguire interrogazioni che coinvolgono contemporaneamente anche più di due tabelle.
- Ad esempio è interessante estendere l'interrogazione precedente riportando in output non solo i nomi degli studenti e i voti conseguiti, ma anche i titoli dei corsi superati da ogni studente. Per ottenere questo risultato abbiamo bisogno, oltre che delle tabelle Studenti e Esami, anche della tabella Corsi, poiché è in essa che compare il titolo del corso.
- Il codice SQL risultante è:

```
select NomeStudente, CognomeStudente, Titolo, Voto
from Studenti, Esami, Corsi
where Studenti.Matricola=Esami.Studente
      and Corsi.IDcorso=Esami.Corso
```


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 diverse 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

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

Padre
Franco
Luigi

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
Filippo	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

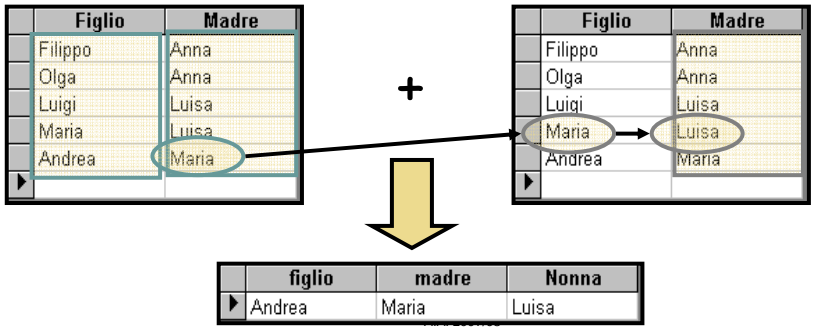
Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

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

61

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

62