
SQL

Materiale adattato dal libro Albano et al e
dal libro Atzeni-et al., Basi di dati

Interrogazione di una base dati

- L'interrogazione di una base di dati è uno degli aspetti più importanti del linguaggio SQL.
- I comandi di interrogazione, o **QUERY**, permettono di effettuare una ricerca dei dati presenti nel database che soddisfano particolari condizioni, richieste dall'utente

SQL

```
SELECT s.Nome, e.Data
FROM   Studenti s, Esami e
WHERE  e.Materia = 'BD' AND e.Voto = 30 AND e.Matricola =
       s.Matricola
```

```
SELECT s.Nome As Nome, 2018 - s.AnnoNascita As Eta,
                                0 As NumeroEsami
FROM   Studenti s
WHERE  NOT EXISTS (SELECT *
                   FROM   Esami e
                   WHERE  e.Matricola = s.Matricola )
```

SQL: Structured Query Language

- SQL è stato definito nel 1973 ed è oggi il linguaggio universale dei sistemi relazionali
- Standard: SQL-84, SQL-89, SQL-92 (o SQL2), SQL:1999 (o SQL3) (ANSI/ISO)
- SQL-92: entry, intermediate e full SQL.
- SQL:1999: a oggetti.
- SQL: DDL, DML, Query language.

Una distinzione terminologica (separazione fra dati e programmi)

Nei linguaggi di interrogazione di basi di dati
distinguiamo tra

Data Manipulation Language (DML)

per l'interrogazione e l'aggiornamento di
(istanze di) basi di dati

Select insegnamento

From Orario

Where docente="Mario Rossi"

Data Definition Language (DDL)

per la definizione di schemi (logici, esterni,
fisici) e altre operazioni generali

```
CREATE TABLE orario (  
    insegnamento CHAR(20) ,  
    docente        CHAR(20) ,  
    aula           CHAR(4)  ,  
    ora            CHAR(5)  
)
```

Istruzione SELECT per l'interrogazione

- Le query vengono effettuate mediante il comando SELECT.
- Sintassi:

```
SELECT ListaAttributi  
FROM ListaTabelle  
[ WHERE Condizione ]
```

Bisogna specificare:

- La "target list" cioè la lista degli attributi interessati.
- clausola **FROM** per stabilire in quale/quali tabella/e sono contenuti i dati che ci occorrono.
- clausola **WHERE** per esprimere le condizioni che i dati cercati devono soddisfare.

Target List (lista degli attributi)

- Specificare la target list corrisponde a scegliere alcuni degli attributi della tabella o delle tabelle considerate.
- Implementa l'operazione di **proiezione** dell'algebra relazionale

Rubrica

Matr	Cognome	Nome	Email	tel

Select Cognome, nome, tel
FROM Rubrica

Where

- La clausola **WHERE** serve a scegliere le righe della tabella che soddisfano una certa condizione.
- In questo modo la clausola where implementa la **selezione** dell'algebra relazionale

```
SELECT *  
FROM Rubrica  
WHERE nome= 'Mario'
```

Rubrica

Matr	Cogn	Nome	Email	tel
...	...	Pippo
...	...	Mario
...	...	Silvia
...	...	Mario
...	...	Marco
...	...	Mario

From

- La clausola **FROM** ha lo scopo di scegliere quali sono le tabelle del database da cui vogliamo estrarre le nostre informazioni.
- Nel caso in cui le tabelle elencate sono due, la clausola **FROM**, insieme con la clausola **WHERE**, che stabilisce quali righe delle due tabelle bisogna accoppiare, implementa il **theta join**

```
SELECT *  
FROM Studenti, Esami  
Where Studenti.Matricola=Esami.Studente
```

Studenti

Matricola	Nome	Cognome	Indirizzo

Esami

cod.Materia	Nome	Docente	studente

SELECT

```
SELECT ListaAttributi  
FROM ListaTabelle  
[ WHERE Condizione ]
```

La query considera il prodotto cartesiano tra le tabelle in *ListaTabelle* (**JOIN**).

Fra queste seleziona solo le righe che soddisfano la *Condizione* (**SELEZIONE**) e infine valuta le espressioni specificate nella target list *ListaAttributi* (**PROIEZIONE**).

La SELECT implementa quindi gli operatori di Proiezione, Selezione e Join dell'algebra Relazionale, ecc.

SQL PER INTERROGARE: SELECT FROM WHERE

- SQL è un calcolo su **multiinsiemi**.

- Il comando base dell'SQL:

SELECT [DISTINCT] Attributo {, Attributo}

FROM Tabella [Identificatore] {, Tabella [Identificatore]}

[WHERE Condizione]

- Semantica: prodotto + restrizione + proiezione.
- Un attributo A di una tabella " R x " si denota come:
 - A oppure
 - $R.A$ oppure
 - $x.A$



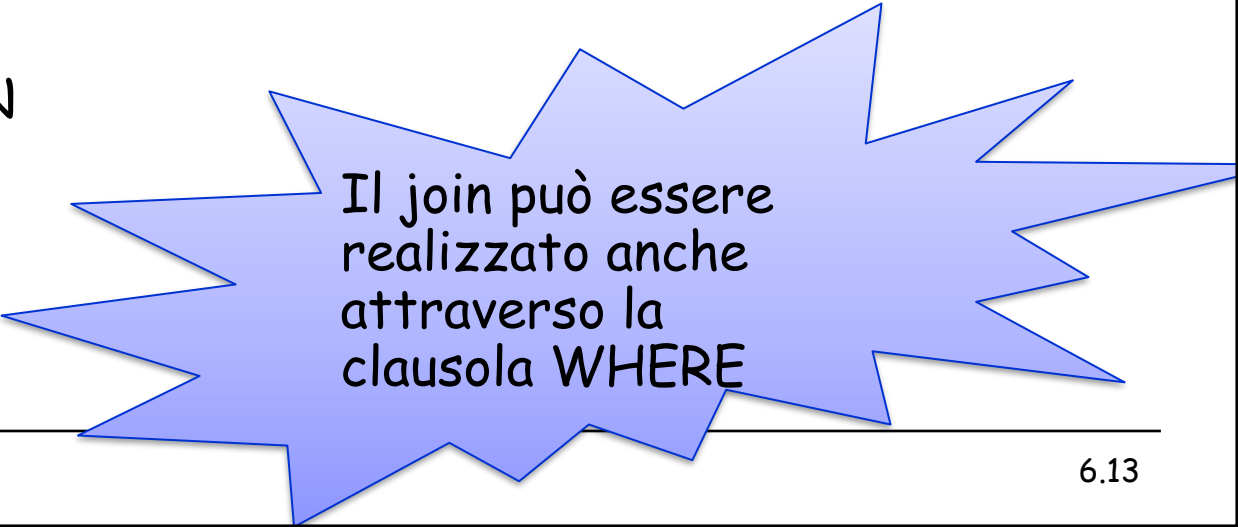
Ridenominazione
della tabella R

LA LISTA DEGLI ATTRIBUTI

- $\text{Attributi} ::= *$
| $\text{Expr} \text{ [[AS] Nuovonome] } \{, \text{Expr} \text{ [[AS] Nuovonome] } \}$
- $\text{Expr} ::= [\text{Ide.}] \text{Attributo} \mid \text{Const}$
| $(\text{Expr}) \mid [-] \text{Expr} [\text{Op Expr}]$
| $\text{COUNT}(*)$
| $\text{AggrFun} ([\text{DISTINCT}] [\text{Ide.}] \text{Attributo})$
- e AS x: dà un nome alla colonna di e
- $\text{AggrFun} ::= \text{SUM} \mid \text{COUNT} \mid \text{AVG} \mid \text{MAX} \mid \text{MIN}$
- AggrFun: o si usano tutte funzioni di aggregazione (e si ottiene un'unica riga) o non se ne usa nessuna.

LA LISTA DELLE TABELLE

- Le tabelle si possono combinare usando:
 - “,” (prodotto): FROM T1,T2
 - Giunzioni di vario genere:
 - Studenti s JOIN Esami e ON s.Matricola=e.Matricola
 - Studenti s JOIN Esami e USING Matricola
 - Studenti s NATURAL JOIN Esami e
 - Studenti s LEFT JOIN Esami e ON s.Matricola=e.Matricola
 - LEFT JOIN - USING
 - NATURAL LEFT JOIN
 - RIGHT JOIN
 - FULL JOIN



Il join può essere
realizzato anche
attraverso la
clausola WHERE

LA CONDIZIONE

- Combinazione booleana di predicati tra cui:
 - Expr Comp Expr
 - Expr Comp (Sottoselect che torna un valore)
 - [NOT] EXISTS (Sottoselect)
 - Expr Comp (ANY | ALL) (Sottoselect)
 - Expr [NOT] IN (Sottoselect) (oppure IN (v1,...,vn))
- Comp: <, =, >, <>, <=, >=

SINTASSI DELLA SELECT

- Sottoselect:

```
SELECT [DISTINCT] Attributi  
FROM Tabelle  
[WHERE Condizione]  
[GROUP BY A1,...,An [HAVING Condizione]]
```

- Select:

```
Sottoselect  
{ (UNION | INTERSECT | EXCEPT)  
  Sottoselect }  
[ ORDER BY Attributo [DESC] {, Attributo [DESC]} ]
```

Esempio: proiezione

- Visualizzare il nome e l'età dei dati presenti nella tabella Persone

`SELECT nome, eta`

`FROM Persone`

Nome	Eta
ANDREA	27
ALDO	25
MARIA	55
ANNA	50
FILIPPO	26
LUIGI	50
FRANCO	60
OLGA	30
SERGIO	85
LUISA	75

Persone

NOME	ETA	REDDITO
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	29
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

ESEMPI: Proiezione

- Trovare il nome, la matricola e la provincia degli studenti:

```
SELECT Nome, Matricola, Provincia  
FROM   Studenti
```

Nome	Matricola	Provincia
Isaia	171523	PI
Rossi	167459	LU
Bianchi	179856	LI
Bonini	175649	PI

Selezione (Restrizione), la clausola where

- Con SQL è anche possibile effettuare la selezione dell'algebra relazionale mediante la clausola **WHERE**.
- La clausola **WHERE** permette infatti di specificare le condizioni che devono soddisfare le righe cercate.
- Ovviamente la clausola WHERE è opzionale, e se si omette, si selezionano tutte le righe della tabella specificata
- **Esempio:** Nome, età e reddito delle persone con meno di trent'anni:

SELECT nome, eta, reddito

FROM persone

WHERE eta < 30

Esempio query

- Nome e reddito delle persone con meno di trenta anni

$\pi_{\text{Nome, Reddito}}(\sigma_{\text{Eta} < 30}(\text{Persone}))$

SELECT nome, reddito

FROM persone

WHERE eta < 30

Nome	Reddito
Andrea	21
Aldo	15
Filippo	29

Persone

NOME	ETA	REDDITO
Andrea	27	21
Aldo	25	15
Maria	55	42
Anna	50	35
Filippo	26	29
Luigi	50	40
Franco	60	20
Olga	30	41
Sergio	85	35
Luisa	75	87

Esempio

- Visualizzare tutte le colonne della tabella Maternita.

`SELECT *`

`From Maternita`

Se si desidera visualizzare **tutti gli attributi** della tabella, si può semplificare la target list indicando la lista con un semplice **asterisco ***

Maternita

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

ESEMPI: Restrizione

- Trovare tutti i dati degli studenti di Pisa:

```
SELECT  *  
FROM    Studenti  
WHERE   Provincia = 'PI'
```

Nome	Matricola	Provincia	AnnoNascita
Isaia	171523	PI	1996
Bonini	175649	PI	1996

- Trovare la matricola, l'anno di nascita e il nome degli studenti di Pisa
(*Proiezione+Restrizione*):

```
SELECT  Nome, Matricola, AnnoNascita  
FROM    Studenti  
WHERE   Provincia = 'PI'
```

Nome	Matricola	AnnoNascita
Isaia	171523	1996
Bonini	175649	1996

ESEMPI: Prodotto e giunzione

Vedremo il join più in dettaglio più avanti

- Trovare tutte le possibili coppie
Studente-Esami:

```
SELECT  
FROM
```

```
*  
Studenti, Esami
```

- Trovare tutte le possibili coppie
Studente - Esame sostenuto dallo
studente:

```
SELECT  
FROM  
WHERE
```

```
*  
Studenti s, Esami e  
s.Matricola = e.Matricola
```

- Trovare il nome e la data degli
esami per gli studenti che hanno
superato l'esame di BD con 30:

```
SELECT  
FROM  
WHERE
```

```
Nome, Data  
Studenti s, Esami e  
e.Materia = 'BD' AND e.Voto = 30  
AND e.Matricola = s.Matricola
```

Alias delle colonne

- L'alias serve a dare a una colonna un nome diverso rispetto a quello che è utilizzato nella definizione della tabella.
- Implementa l'operatore ρ (Ridenominazione) dell'algebra relazionale
- Può essere usata opzionalmente la parola chiave **AS** tra il nome della colonna e l'alias richiede necessariamente le virgolette se l'alias ha degli spazi.

```
Select Figlio as Figlio_M  
From Maternita
```

```
Select Figlio 'Figlio madre'  
From Maternita
```

Condizioni

- La condizione che segue il costrutto where è una condizione Booleana. Fa quindi uso di
 - Operatori di confronto =, <, >, <> (oppure !=), <=, >=
 - Connettori logici AND, OR, NOT
 - Operatori BETWEEN, IN, LIKE, IS NULL

Operatori di confronto, esempi

Selezionare le persone che si chiamano Mario

```
SELECT *  
FROM Persone  
Where nome='Mario'
```

Persone

NOME	ETA	REDDITO
------	-----	---------

Selezionare gli impiegati che guadagnano più di 1200 euro

```
SELECT *  
FROM Impiegato  
WHERE Stip>1200
```

Stipendio

NOME	Cognome	Stip
------	---------	------

Operatori di confronto, esempi

Selezionare il nome e cognome delle persone che non hanno età superiore a 30 anni

```
SELECT Nome, Cognome  
FROM Persone  
WHERE eta <= 30
```

Persone

NOME	ETA	REDDITO
------	-----	---------

Selezionare tutti gli impiegati tranne quelli che lavorano nel dipartimento di Produzione.

```
SELECT *  
FROM Impiegato  
WHERE dipartimento <> 'Produzione'
```

Impiegati

Matricola	Cognome	Dipart	Stip
-----------	---------	--------	------

Uso dell'operatore BETWEEN

- **BETWEEN** consente la selezione di righe con attributi in un particolare range.
- Esempio: cercare gli impiegati (nome e dipartimenti) che guadagnano tra 1000 e 1500 euro e visualizzare nome e dipartimento:

Impiegati

Matricola	nome	Dipart	Stip
-----------	------	--------	------

SELECT nome, Dipart

FROM Impiegato

Where Stip **between** 1000 and 1500

Uso dell'operatore IN

- E' usato per selezionare righe che hanno un attributo che assume valori contenuti in una lista.

ESEMPIO

Selezionare i nomi dei professori che tengono i corsi di BD1, BD2, Algoritmi e Sistemi

```
SELECT *  
FROM Insegnanti  
WHERE corso IN ( 'BD1' , 'BD2' , 'Algoritmi' , 'Sistemi' )
```

Selezionare le persone che hanno più di 30 anni e che non abitano a Pisa

Persone

```
SELECT *  
FROM Persone  
WHERE (eta>30) AND NOT(citta= 'Pisa' )
```

NOME	ETA	DIPART	CITTA
------	-----	--------	-------

Selezionare gli impiegati che lavorano nel dipartimento di produzione e quelli che lavorano in segreteria

```
SELECT *  
FROM Persone  
WHERE Dipart= 'Produzione' OR Dipart= 'Segreteria'
```

Uso dell'operatore LIKE

LIKE è usato per effettuare ricerche "wildcard" (ossia con un simbolo jolly) di una stringa di valori.

Le condizioni di ricerca possono contenere letterali, caratteri o numeri.

Esistono due tipi di wildcard:

% denota zero o più caratteri.

_ denota un carattere.

Uso dell'operatore LIKE

Esempio: Fra le persone elencate nella tabella
Persone, determinare quelle il cui nome termina per 'a'

```
SELECT *  
FROM Persone  
Where nome LIKE '%a'
```

NOME	ETA	REDDITO
Andrea	27	21
Maria	55	42
Anna	50	35
Olga	30	41
Luisa	75	87

Esempio: Selezionare le sequenze di DNA in cui ci sono almeno
due simboli 'G' che distano 3 caratteri

```
SELECT *  
FROM DNA  
WHERE Sequenza LIKE '%G___G%'
```

Simbolo escape

- A volte può succedere che uno dei simboli coinvolti nel pattern matching sia proprio _ oppure %.
- In questo caso, si sceglie un simbolo che non è ammesso fra i simboli della stringa (supponiamo '#') detto **simbolo escape**, nell'espressione per la ricerca si fa precedere il simbolo _ o % cercato dal simbolo escape, e poi si specifica che '#' è il simbolo di escape.
- **Esempio:** Modelli il cui nome inizia per C_F

SELECT *

FROM Modelli

WHERE nome_modello LIKE 'C#_F%' ESCAPE '#'

IL VALORE NULL

- Il valore di un campo di un'ennupla può mancare per varie ragioni; SQL fornisce il valore speciale NULL per tali situazioni.
- La presenza del NULL introduce dei problemi:
 - occorrono dei predicati per controllare se un valore è/non è NULL.
 - la condizione "reddito>8" è vera o falsa quando il reddito è uguale a NULL? Cosa succede degli operatori AND, OR e NOT?
 - Occorre una logica a 3 valori (vero, falso e unknown).
 - Va definita opportunamente la semantica dei costrutti. Ad es. il WHERE elimina le ennuple che non rendono vera la condizione.
- Nuovi operatori sono utili (es. giunzioni esterne)

Is Null

- L'operatore **IS NULL** verifica se il contenuto di un operando è null.
 - A IS NULL è true se A vale NULL, false altrimenti
 - A IS NOT NULL è true se A ha un valore noto, false altrimenti
- Esempio:

Veicoli

Targa	Cod_mod	Cod_cat	Cilindrata	Cod_comb.	cav.Fisc	Velocita	Posti	Imm
-------	---------	---------	------------	-----------	----------	----------	-------	-----

```
SELECT *  
FROM Veicoli  
WHERE Cilindrata IS NULL
```

Impiegati

Matricola	Cognome	Filiale	Età
5998	Neri	Milano	45
9553	Bruni	Milano	NULL

$\sigma_{\text{Età} > 40 \text{ OR Et\`a IS NULL}} (\text{Impiegati})$

Select *
FROM Impiegati
WHERE Eta>40 or Eta=null

E' equivalente?

Logica predicati a tre valori: True, false, sconosciuto (NULL)

- I valori NULL rappresentano l'assenza di informazione, cioè:
 - Valore sconosciuto
 - Valore non disponibile
 - Attributo non applicabile
- Ciascun valore NULL è quindi considerato diverso dagli altri valori NULL
- Il risultato del confronto logico con un valore che è NULL da come output: sconosciuto (UNKNOWN), indicato con **S**

NOT	
V	F
F	V
S	S

OR	V	F	S
V	V	V	V
F	V	F	S
S	V	S	S

AND	V	F	S
V	V	F	S
F	F	F	F
S	S	F	S

Esempio

OR	V	F	S	AND	V	F	S
V	V	V	V	V	V	F	S
F	V	F	S	F	F	F	F
S	V	S	S	S	S	F	S

- `SELECT * FROM Persone WHERE Nome=Luigi OR AnnoNascita=2022;`

Persone

Nome	Cognome	AnnoNascita	LuogoDiNascita
Luigi	Bianchi	NULL	Roma
Mario	Verdi	2022	Pisa
Marco	Rossi	NULL	NULL

$T \text{ OR } S \rightarrow T$

$F \text{ OR } T \rightarrow T$

$F \text{ OR } S \rightarrow S$

- `SELECT * FROM Persone WHERE Nome=Luigi AND AnnoNascita=2022;`

Persone

Nome	Cognome	AnnoNascita	LuogoDiNascita
Luigi	Bianchi	NULL	Roma
Mario	Verdi	2022	Pisa
Marco	Rossi	NULL	NULL

$T \text{ AND } S \rightarrow S$

$F \text{ AND } T \rightarrow F$

$F \text{ AND } S \rightarrow F$

Esempio

OR	V	F	S	AND	V	F	S
V	V	V	V	V	V	F	S
F	V	F	S	F	F	F	F
S	V	S	S	S	S	F	S

- `SELECT * FROM Persone WHERE AnnoNascita IS NULL`

Nome	Cognome	AnnoNascita	LuogoDiNascita	Persone
Luigi	Bianchi	NULL	Roma	ok
Mario	Verdi	2022	Pisa	
Marco	Rossi	NULL	NULL	ok

- `SELECT * FROM Persone WHERE AnnoNascita IS NULL AND/OR Luogo IS NULL`

Nome	Cognome	AnnoNascita	Luogo	Persone
Luigi	Bianchi	NULL	Roma	Restituita dall'OR
Mario	Verdi	2022	Pisa	
Marco	Rossi	NULL	NULL	Restituita dall'AND e dall'OR

Valore NULL e operatori logici

OR	V	F	S
V	V	V	V
F	V	F	S
S	V	S	S

MATRIC NOME	COGNOME	DATAISCRIZIONE
282320 Gianna	Neri	04-MAG-20
369871 Mario	Rossi	04-MAG-20
515140 Mario	Verdi	
090456 Mario	Bianchi	04-MAG-20
579555 Luigi	Rossi	
018701 Luca	Bianchi	04-MAG-20

Select * from studenti
where cognome='Rossi'

MATRIC NOME	COGNOME	DATAISCRIZIONE
369871 Mario	Rossi	04-MAG-20
579555 Luigi	Rossi	

Select * from studenti
where cognome='Rossi' or
dataiscrizione=NULL

MATRICOLA	NOME	COGNOME	DATAISCRIZIONE
369871	Mario	Rossi	04-MAG-20
579555	Luigi	Rossi	(null)

Select * from studenti
where cognome='Rossi' or
dataiscrizione IS NULL

MATRIC NOME	COGNOME	DATAISCRIZIONE
369871 Mario	Rossi	04-MAG-20
515140 Mario	Verdi	
579555 Luigi	Rossi	

AND	V	F	S
V	V	F	S
F	F	F	F
S	S	F	S

Valore NULL e operatori logici

MATRIC NOME	COGNOME	DATAISCRIZIONE
282320 Gianna	Neri	04-MAG-20
369871 Mario	Rossi	04-MAG-20
515140 Mario	Verdi	
090456 Mario	Bianchi	04-MAG-20
579555 Luigi	Rossi	
018701 Luca	Bianchi	04-MAG-20

Select * from studenti
where cognome='Rossi'

MATRIC NOME	COGNOME	DATAISCRIZIONE
369871 Mario	Rossi	04-MAG-20
579555 Luigi	Rossi	

Select * from studenti
where cognome='Rossi' and
dataiscrizione IS NULL

MATRIC NOME	COGNOME	DATAISCRIZIONE
579555 Luigi	Rossi	

Select * from studenti
where cognome='Rossi' and
dataiscrizione=NULL

nessuna riga selezionata

Forma negativa

Tutti gli operatori descritti sono presenti anche in forma Negativa, con ovvio significato.

- NOT BETWEEN
- NOT IN
- NOT LIKE
- NOT NULL

Espressione nella target list

- Esempio:
- Evidenziare i cognomi degli impiegati e il loro stipendio aumentato del 20%

IMPIEGATO

Nome	Cognome	Stip
Mario	Rossi	1200
Luigi	Verdi	1130
Maria	Bianchi	1450
Luisa	Gialli	1300

Select cognome, stip *120/100
FROM Impiegato

Cognome	Stip * 120/100
Rossi	1440
Verdi	1356
Bianchi	1740
Gialli	1560

Espressione nella target list con ridenominazione

- **Esempio:**
- Evidenziare i cognomi degli impiegati e il loro stipendio aumentato del 20%.
- Rinominare la colonna corrispondente al salario aumentato con la denominazione "Aumento".

```
Select cognome, stip *120/100 Aumento  
FROM Impiegato
```

IMPIEGATO

Nome	Cognome	Stip
Mario	Rossi	1200
Luigi	Verdi	1130
Maria	Bianchi	1450
Luisa	Gialli	1300

Cognome	Aumento
Rossi	1440
Verdi	1356
Bianchi	1740
Gialli	1560

ORDINAMENTO OPERATORI AGGREGATI E RAGGRUPPAMENTO

Ordinamento del risultato

E' possibile dare un ordinamento del risultato di una select. L'ordinamento si può effettuare in base a un attributo, e può essere crescente o decrescente. La sintassi è la seguente:

```
SELECT lista_attributi  
FROM nome_tabella  
WHERE condizioni  
ORDER BY Attributo [ASC/DESC]
```

Le righe vengono ordinate in base al campo Attributo in maniera crescente o decrescente secondo se è data la specifica **ASC** o **DESC**. **ASC** è il default. Secondo il tipo dell'attributo, l'ordinamento è quello più naturale su quel dominio.

Esempio

Nome e reddito delle persone con meno di trenta anni in ordine alfabetico

```
SELECT nome, reddito  
FROM persone  
WHERE reddito < 30  
ORDER BY nome
```

Persone

Nome	Reddito
Andrea	21
Aldo	15
Filippo	30

Persone

Nome	Reddito
Aldo	15
Andrea	21

Doppio ordinamento

Si può voler ordinare i dati in base a una certa chiave (attributo) e poi ordinare i dati che coincidono su quella chiave in base a un'altra chiave (attributo).

Ordinare gli studenti in base al loro cognome, in modo tale che due persone con lo stesso cognome siano ordinate in base al nome, e persone con lo stesso nome e cognome siano ordinate in base all'ordine inverso della data di nascita

```
Select *  
From Studenti  
Order by cognome [asc], nome [asc] , nascita desc
```