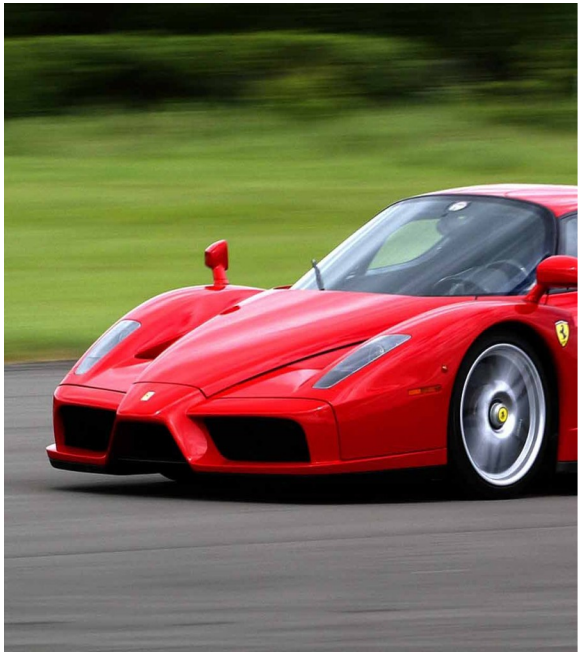
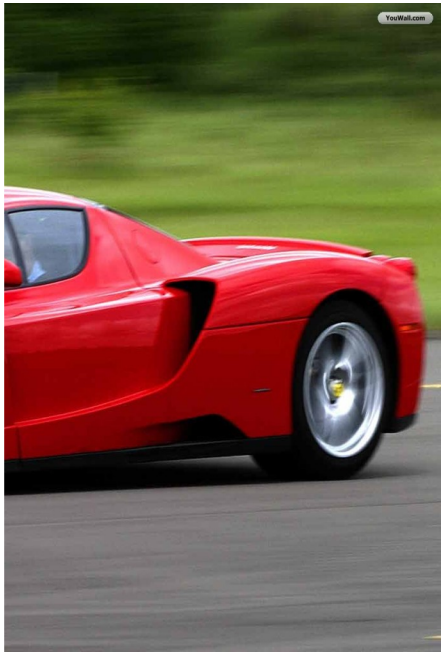
JOIN





SELECT, notazione puntata

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

In questo caso è equivalente ma in caso di omonimia può servire di specificare il nome della tabella a cui appartiene l'attributo nella select: per farlo si utilizza la cosiddetta ***"notazione puntata" :***

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

Selezione, proiezione e join

-Le istruzioni SELECT con una sola relazione (tabella) nella clausola FROM implementanole operazioni di

- **selezione**
- **proiezione**
- **ridenominazione**

- Indicando più tabelle nella clausola FROM si implementa invece l'operazione di JOIN (e di prodotto cartesiano)

Analizziamo alcuni casi in cui è necessario correlare due tabelle.

Join

AUTO

id	targa	marca	modello	colore	id_categoria
1	XY111	Fiat	Panda	bianco	1
2	AA333	Ferrari	GTO	rosso	3
5	AM777	Fiat	Punto	rosso	1
10	EE666	Fiat	Panda	rosso	1

CATEGORIA

id	descr	prezzo
1	citycar	10
3	supercar	20
4	SUV	30

Schema logico

Autonoleggio={**AUTO**(id,targa,marca,modello,colore,id_categoria), **CATEGORIA**(id,descr)}

Condizione di Join

AUTO.id_categoria= CATEGORIA.id

o equivalentemente

CATEGORIA.ID=AUTO.id_categoria

Join

AUTO

id	targa	marca	modello	colore	id_categoria
1	XY111	Fiat	Panda	bianco	1
2	AA333	Ferrari	GTO	rosso	3
5	AM777	Fiat	Punto	rosso	1
10	EE666	Fiat	Panda	rosso	1

CATEGORIA

id	descr	prezzo
1	citycar	10
3	supercar	20
4	SUV	30

OPERAZIONE di Join

Resultset

id	targa	marca	modello	colore	id_categoria	id	desc	prezzo
1	XY111	Fiat	Panda	bianco	1	1	citycar	10
2	AA333	Ferrari	GTO	rosso	3	3	supercar	20
5	AM777	Fiat	Punto	rosso	1	1	citycar	10
10	EE666	Fiat	Panda	rosso	1	1	citycar	10

Join

AUTO

id	targa	marca	modello	colore	id_categoria
1	XY111	Fiat	Panda	bianco	1
2	AA333	Ferrari	GTO	rosso	3
5	AM777	Fiat	Punto	rosso	1
10	EE666	Fiat	Panda	rosso	1

CATEGORIA

id	descr	prezzo
1	citycar	10
3	supercar	20
4	SUV	30

```
SELECT *
FROM auto JOIN categoria ON
auto.id_categoria= categoria.id;
```

id	targa	marca	modello	colore	id_categoria	id	desc	prezzo
1	XY111	Fiat	Panda	bianco	1	1	citycar	10
2	AA333	Ferrari	GTO	rosso	3	3	supercar	20
5	AM777	Fiat	Punto	rosso	1	1	citycar	10
10	EE666	Fiat	Panda	rosso	1	1	citycar	10

Esercizio, join

Dato un DB contenente le seguenti tabelle:

Impiegati

Id	Cognome	Id_Reparto
1	Rossi	1
5	Bianchi	5
6	Verdi	1

Reparti

Id	Capo
1	Neri
5	Rosati
8	Gialli

1) Quali sono gli attributi che mi permetteranno di correlare le due tabelle?
2) Scrivere la condizione di Join in notazione puntata
3) Scrivere l'operazione di Join
4) Scrivere il ResultSet dell'operazione di Join
5) Scrivere l'operazione di Join invertendo le tabelle
6) Scrivere il ResultSet dell'operazione di Join a tabelle invertite

Esercizio, join

Impiegati

Id	Cognome	Id_Reparto
1	Rossi	1
5	Bianchi	5
6	Verdi	1

Reparti

Id	Capo
1	Neri
5	Rosati
6	Gialli

1) *gli attributi di correlazione sono Impiegato.id_reparto e reparti.id*
2) *Scrivere la condizione di Join*
    1) *Impiegato.id_reparto = reparti.id*
3) *Scrivere operazione di Join*
    1) ***SELECT** * **FROM** Impiegati **JOIN** Reparti **ON** Impiegato.id_reparto = reparti.id*
4) *ResultSet*

Id	Cognome	Id_reparto	Id	Capo
1	Rossi	1	1	Neri
5	Bianchi	5	5	Rosati
6	Verdi	1	1	Neri

Esercizio, join

Impiegati

Id	Cognome	Id_Reparto
1	Rossi	1
5	Bianchi	5
6	Verdi	1

Reparti

Id	Capo
1	Neri
5	Rosati
6	Gialli

1) *gli attributi di correlazione sono Impiegato.id_reparto e reparti.id*
2) *Scrivere la condizione di Join*
    1) *Impiegato.id_reparto = reparti.id*
3) *Scrivere operazione di Join*
    1) *Impiegati J* $_{Impiegato.id\_reparto = reparti.id}$ *Reparti*
4) *ResultSet*

Id	Cognome	Id_reparto	Id	Capo
1	Rossi	1	1	Neri
5	Bianchi	5	5	Rosati
6	Verdi	1	1	Neri

Ambiguità nei nomi dei campi

Impiegati

Id	Cognome	Id_Reparto
1	Rossi	1
5	Bianchi	5
6	Verdi	1

Reparti

Id	Capo
1	Neri
5	Rosati
8	Gialli

```
SELECT id,cognome
FROM impiegati JOIN reparti ON id_reparto=id
WHERE capo="Neri";
```

Join

- *Permette di correlare dati contenuti in 2 (o più) tabelle diverse.*
- *Operatore di join:* JOIN
- Condizione di Join: uguaglianza tra chiave primaria e foreign key che ad essa si riferisce
- *Il ResultSet avrà*
  - Come colonne, **TUTTE** le colonne delle tabelle di partenza
  - Come righe, tutte le righe delle tabelle di partenza che soddisfano la condizione di join
- *Schema e istanza delle tabelle di partenza non cambiano*

Focus: INNER JOIN

1) Mi serve una JOIN quando nel mio resultset desidero visualizzare o porre condizioni su dati che provengono da più tabelle

2) La regola di confronto va SEMPRE scritta ed è sempre data dall'**uguaglianza** tra la chiave primaria e la foreign key che ad essa si riferisce

3) Il resultset sarà composto

- da **TUTTE** le colonne delle tabelle di partenza
- SOLO dallle righe delle tabelle di partenza che soddisfano la condizione di join

Join tra tre tabelle

Studenti

Matricola	Nome	Cognome
150	Mario	Rossi
151	Nino	Bianchi
153	Rino	Rano

Corsi

CODICE	NOME
3	FISICA1
5	CANTO
8	DISEGNO

Esami

Matricola_studente	Codice_corso	Voto
150	5	18
151	5	30
151	3	20

JOIN:
ESERCIZI E PROGETTI

Join

Dato lo schema logico

`ABITANTI(id,nominativo, Indirizzo,id_città)`

`CITTA'(id,nome,kmq)`

Scrivere:

1)Condizione di Join (su carta)
2)Operazione di Join (su carta)
3)Colonne del resultset (su carta)
4)Implementare lo schema logico in un db chiamato "anagrafe"
5)Inserire voi stessi e 4 compagni di classe nel db
6)Visualizzare solo nominativo e città di residenza
7)Visualizzare I kmq totali

Join

Schema logico
ABITANTI(nominativo, Indirizzo,id_città)
CITTA'(id,nome,kmq)

1) Condizione di Join: ABITANTI.id_città=CITTA'.id
2) Operazione di Join: ABITANTI $J_{\text{ABITANTI.id\_città=CITTA'.id}}$ CITTA'
3) Resultset: (nominativo,indirizzo,id_città,id,nome,kmq)

OPZIONALE: modificare lo schema precedente e aggiungere id_citta_lavoro e individuare I pendolari

Università

Studenti

Matricola	Nome	Cognome
150	Mario	Rossi
151	Nino	Bianchi
153	Rino	Rano

Corsi

CODICE	NOME
3	FISICA1
5	CANTO
8	DISEGNO

Esami

Matricola_studente	Codice_corso	Voto
150	5	18
151	5	30
151	3	20

Esercizio Libretto universitario

1) Vogliamo ottenere un resultset che riporti I nomi dei corsi e i voti dello studente Bianchi
2) Implementare fisicamente lo schema logico
3) Popolare il database
4) Progettare e testare la query per ottenere il libretto
5) OPZIONALE: Scrivere codice Java dove
   1) INPUT Leggo con Scanner il Cognome di uno studente
   2) OUTPUT stampo a schermo la query per ottenere il suo libretto

Join tra tre tabelle: soluzione

Operazione

studenti **J** esami **J** corsi

Condizioni di Join

STUDENTI.MATRICOLA = ESAMI.MATRICOLA_STUDENTE

ESAMI.CODICE_CORSO = CORSI.CODICE

```
SELECT *
FROM studenti JOIN esami ON studenti.matricola = esami.matr_studente
JOIN corsi ON esami.codice_corso = corsi.codice
```

MATRICOLA	NOME	COGNOME	MATR_STUDENTE	CODICE_CORSO	VOTO	CODICE	NOME
150	MARIO	ROSSI	150	5	18	5	CANTO
151	NINO	BIANCHI	151	5	30	5	CANTO
151	NINO	BIANCHI	151	3	20	3	FISICA

Libretto universitario

Libretto: voglio estrarre I nomi dei corsi e i voti conseguiti da nino

Cond1 : STUDENTI.MATRICOLA= ESAMI.MATRICOLA_STUDENTE

Cond2: CORSI.CODICE=ESAMI.CODICE_CORSO

Operazione:

Result set della JOIN

MATRICOLA	NOME	COGNOME	MATR_STUDENTE	CODICE_CORSO	VOTO	CODICE	NOME
150	MARIO	ROSSI	150	5	18	5	CANTO
151	NINO	BIANCHI	151	5	30	5	CANTO
151	NINO	BIANCHI	151	3	20	3	FISICA

MATRICOLA	NOME	COGNOME	MATR_STUDENTE	CODICE_CORSO	VOTO	CODICE	NOME
151	NINO	BIANCHI	151	5	30	5	CANTO
151	NINO	BIANCHI	151	3	20	3	FISICA

NOME	VOTO
CANTO	30
FISICA	20

Progetto RESIDENZA

Creazione dello schema

Creare il database anagrafe
Creare le tabelle seguenti

```
CREATE TABLE città (
  id int unsigned NOT NULL AUTO_INCREMENT,
  nome varchar(20) NOT NULL,
  kmq smallint unsigned DEFAULT NULL,
  PRIMARY KEY (id),
  UNIQUE KEY nome (nome)
);

CREATE TABLE abitanti (
  id int unsigned NOT NULL AUTO_INCREMENT,
  nome varchar(20) NOT NULL,
  id_città_res int unsigned NOT NULL,
  id_città_lav int unsigned DEFAULT NULL,
  PRIMARY KEY (id),
  UNIQUE KEY nome (nome),
  KEY id_città_res (id_città_res),
  KEY id_città_lav (id_città_lav),
  CONSTRAINT abitanti_ibfk_1 FOREIGN KEY (id_città_res) REFERENCES città (id),
  CONSTRAINT abitanti_ibfk_2 FOREIGN KEY (id_città_lav) REFERENCES città (id)
);
```

Inserimento BULK di dati

```
INSERT INTO città VALUES
  (1,'Roma',1200),(2,'Milano',400),
  (3,'Ladispoli',22);

INSERT INTO abitanti VALUES
  (1,'Rino',1,NULL),(2,'Pino',1,1),
  (3,'Gino',2,1),(4,'Dino',2,NULL),
  (5,'Lino',2,2);
```

Altri tipi di Join

```
select * from abitanti;
+----+------+--------------+--------------+
| id | nome | id_città_res | id_città_lav |
+----+------+--------------+--------------+
|  1 | Rino |            1 |         NULL |
|  2 | Pino |            1 |            1 |
|  3 | Gino |            2 |            1 |
|  4 | Dino |            2 |         NULL |
|  5 | Lino |            2 |            2 |
+----+------+--------------+--------------+


select * from città;
+----+-----------+------+
| id | nome      | kmq  |
+----+-----------+------+
|  1 | Roma      | 1200 |
|  2 | Milano    |  400 |
|  3 | Ladispoli |   22 |
+----+-----------+------+
```

Rino e Dino sono disoccupati

Ladispoli non ha lavoratori

Inner Join

Con la Inner Join Rino,Dino e Ladispoli non fanno parte del Resultset perché non partecipano alla relazione di lavoro

```
select abitanti.nome,città.nome as lavoro
from abitanti join città on id_città_lav = città.id;

+------+--------+
| nome | lavoro |
+------+--------+
| Lino | Milano |
| Pino | Roma   |
| Gino | Roma   |
+------+--------+
```