

# SQL

Tuesday, 9 May 2023 09:24

- Usato per manipolare dati di un database relazionale  
Ed è adottato da tutti i dbms
- A seconda delle feature implementate dai dbms, abbiamo 3 tipologie:
  - o Entry, aka solo funzioni basi dell'sql
  - o Intermediate, che soddisfa tutte le possibili esigenze del mercato
  - o Full, il pacchetto completo con tutto e tutti
- Abbiamo imparato l'algebra relazionale siccome essa può essere trasferita in SQL
  - o Adotta logica a 3 valori (T, F, ?)
  - o Linguaggio completo per il supporto di oggetti persistenti
  - o Istruzioni equivalenti differiscono in leggibilità e non in efficienza
  - o Le relazioni sono tabelle anzichè insiemi
  - o Siccome sono tabelle possiamo avere righe uguali
  - o Linguaggio dichiarativo, quindi l'ordine è prestabilito
- Varie operazioni:
  - o DDL (data definition language)  
Operazione di definizione e modifica dello schema
    - Create
      - Definisce il db, tabelle, domini, viste, vincoli ed autorizzazioni
    - Alter
      - Modifica attributi e vincoli
    - Drop
      - Elimina database e tabelle
  - o DML (data manipulation language)
    - Operazioni di interrogazione
      - Select
        - ◆ Formula query per ottenere degli elementi
        - Formula:  
**Select** attributo/i  
**From** tabella/e  
**Where** condizione
        - ◆ Select \* from R1 -> seleziona tutti gli attributi da R  
aka prendiamo tutte le istanze della tabella  
aka prendiamo la tabella
        - Select A, B from R1 -> Prende attributi A, B da R1
        - ▲ I duplicati non vengono rimossi

a



- ▼ I dupliquati non vengono rimossi  
Affinchè essi vengano rimossi dobbiamo aggiungere DISTINCT  
**SELECT DISTINCT Ruolo FROM Docente**
- ◆ Nel WHERE possiamo aggiungerci delle condizioni  
**SELECT matricola, cognome, nome FROM Docente WHERE Ruolo = ?**  
In AR:  
 $Proj_{matricola,Cognome,Nome}(o_{ruolo} = Ricercatore \text{ AND } Stipendio \geq 20000)$
- ◆ E' consigliato selezionare le stringhe con l'operatore Like  
Questo ci dà più libertà  
Es.  
**SELECT \* from Docente WHERE nome like 'A\_d%**'  
Noi qui diciamo che la parola deve iniziare con la A  
Ed avere la d come terza lettera  
E successivamente possiamo avere un numero non prestabilito  
Es. tutte le parole che hanno la parola bio: "%bio%"
- ◆ Valori null: IS NULL
- ◆ |Prodotto cartesiano  
**SELECT \* FROM R1, R2**  
Aka possiamo selezionare più tabelle  
Se volessimo fare il theta join, allora si usa la clausola  
Nota: per evitare ambiguità, trattare ora gli attributi come 'valore'  
Es.
- SELECT Docente.nome, Studenti.nome from Docente, Studenti**  
Es.  
SQL:  
**SELECT Docente.\* FROM Docente, Stipendio WHERE Docente.Ruolo = ?**  
Ar:  
 $o_{valore} > 60000 \text{ AND classe}Stipendio = classe$  (*Docente* >< *Stipendio*)  
AR:  $o_{ruolo} = Ricercatore(Docente) >< classe Stipendio = classe$   $o_{valore}$   
SQL: **SELECT \* FROM Docente, Stipendio WHERE ClasseStipendio = ClasseDocente**
- Es.  
formulare una query SQL che produca l'intersezione tra persone e docenti  
AR:  
 $p_{matr<-matr_{ad}} \left( Proj_{matricola_{ad}, cognome, nome}(Personale) \right) ><$   
 $p_{matr<-matr_{st}} \left( Proj_{matricola_{st}, cognome, nome}(studente) \right)$   
SQL:  
**SELECT Matricola\_d Personale.NomDoc nome Pers.NomDoc nome**

olo="Ricercatore" AND Stipendio>=2000

$\cup_0(Docente)$

abilito di letter

Tabella.attributo

ente

nte.classeStipendio = Stipendio.classe AND Stipendio.valore > 60000

$dio)$

$valore>6000(Stipendio)$

ipendio=Classe AND Valore>6000 AND Ruolo like "Ricercatore"

rsonale\_non docente e studente senza usare gli operatori INTERSEZIONE né di DIFFERENZA

$NoDoc)$

con nome



```
SELECT * FROM Personale_nodoc, Personale_nodoc, Personale_nodoc  
FROM Personale_nodoc, studente  
WHERE MATricola_d=Matricola_st AND pers_nodoc.nome =
```

- ◆ AS permette di avere una scrittura più clean  
**SELECT G.nome FROM Giochi AS G**  
Aka sostituisci G con Giochi
- ◆ Sono possibili operazioni aritmetiche  
Es. **SELECT A+B\*10 FROM ...**
- ◆ Nella WHERE è possibile trasformare questo  
**WHERE (votoAr >= 22 AND votoAr <= 25)**  
In questo  
**WHERE votoAr BETWEEN 22 AND 25**
- ◆ E' possibile ordinare i risultati con la seguente sintassi:  
**ORDER BY attributo1 [asc | desc] {, attributo2 [asc | desc]}**
- ◆ Nella select abbiamo delle operazioni aggiuntive:
  - ◊ **COUNT**
    - ▶ **SELECT count(\*)** conta il numero di righe  
I valori null non contano quindi
    - ▶ **SELECT count(Classe\_stipendio)** conta le numeri di righe per classe
    - ▶ **SELECT count(distinct Classe\_stipendio)** ^ solo le classi uniche
  - ◊ **MIN**
  - ◊ **MAX**
  - ◊ **AVG**
  - ◊ **SUM**

## □ Join

Praticamente semplifichiamo ciò che abbiamo fatto prima

```
SELECT *  
FROM Personale_docente, Stipendio  
WHERE Classe_stipendio=Classe AND  
Valore>=60000 AND Ruolo='Ricercatore'
```

```
SELECT *  
FROM Personale_docente JOIN Stipendio  
Classe_stipendio=Classe  
WHERE Valore>=60000 AND Ruolo='Ricercatore'
```

E qui abbiamo i 3 join:

- ◊ Inner, quello di default
- ◊ Esterno (Outer)
- ◊ Naturale (Natural)

Ed in più possiamo anche specificare:

- ◊ Left
- ◊ Right

**SELECT \* FROM tabella1 [left/right] INNER/OUTER/NATURAL JOIN tabella2**



cognome

= studente.nome AND pers\_nodoc.cognome = studente.cognome

c] ...}

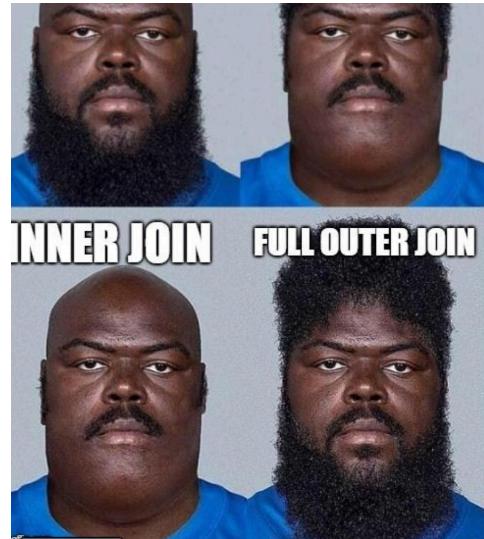
nero di righe di classe stipendio senza null  
lo cose uniche

dio **ON**

ercatore'

**URAL JOIN** tabella2 **ON** condizione





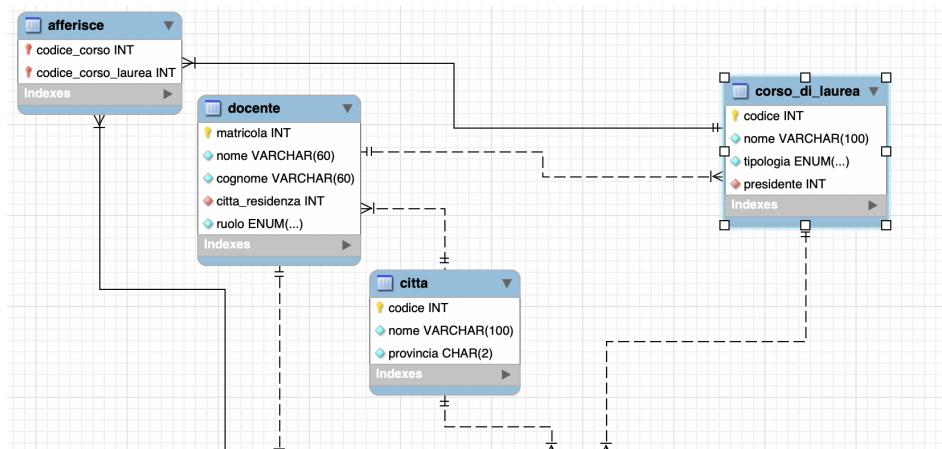
Nota: con il self join bisogna per forza usare AS  
**FROM Docente AS A JOIN Docente AS B ON A.stipendio=B.stipendio**

Quando in AR per unire più tabelle bisogna avere gli stessi  
 In SQL si può fare il join se e solo se abbiamo lo stesso numero  
 Può fare tante operazioni:

- ◆ Operazioni aritmetiche  
 $+,-,*,/$   
 Aka possiamo fare la somma di 2 colonne

- Operazioni di aggiornamento
  - Insert
    - ◆ Inserisce nuove tuple
  - Delete
    - ◆ Elimina tuple
  - Update
    - ◆ Modifica tuple

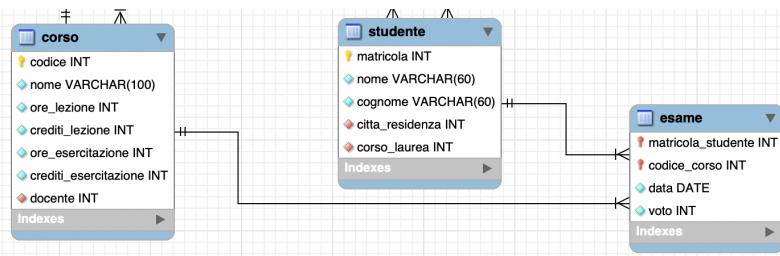
Queste sono mie domande che ho creato per dei miei amici da risolvere  
 Sono un pò stronzini, però eh è divertente!



.stipendio

attributi  
ero di attributi





- 1) Gli studenti con la media più alta superiore a 25 ordinati decrescentemente che arriva

```

select st.nome, avg(voto) as media
    from universita.studente as st, universita.corso_di_laurea as cl, universita.esame as es
where st.corso_laurea = cl.codice and cl.nome like "informatica" and es.matricola_studente = st.matricola
group by st.matricola
having media > 25
order by media

```

- 2) Le 3 provincie dove abitano più persone

```

select ct.provincia, count(*) as num from (
    select citta_residenza, st.matricola from universita.studente as st UNION (
        select citta_residenza, dc.matricola from universita.docente as dc
    )
) as test, universita.citta as ct
where ct.codice = citta_residenza
group by ct.provincia
order by num DESC
LIMIT 3

```

- 3) La provincia con gli studenti aventi la media più alta

```

select citta.provincia
from studente join citta on citta_residenza=citta.codice
join (
    select matricola_studente, avg(voto) as media
    from esame
    group by matricola_studente
    having media >= all(
        select avg(voto)
        from esame
    )
    group by matricola_studente
)
) as medie on medie.matricola_studente = studente.matricola;

```

no dal corso di laurea di informatica



```

select avg(voto) as media, ct.provincia
    from universita.studente as st, universita.esame as es , universita.citta as ct
    where st.matricola = es.matricola_studente and ct.codice = st.citta_residenza
    group by st.matricola
having avg(voto) >= (
    select max(media) from (
        select st.nome, st.cognome, avg(voto) as media, st.citta_residenza
            from universita.studente as st, universita.esame as es
            where st.matricola = es.matricola_studente
            group by st.matricola
    ) as t
)

```

```

SELECT provincia, media
FROM studente, citta, (
    SELECT matricola, avg(voto) as media
    FROM studente AS s, esame as e
    WHERE matricola_studente=matricola
    GROUP BY matricola, nome, cognome
) AS medie
WHERE medie.matricola=studente.matricola AND citta_residenza=codice AND media = (
    SELECT max(media)
    FROM medie
)
GROUP BY provincia, media

```

- 4) Il corso di laurea più numeroso di persone
- 5) Il presidente che possiede il corso di laurea con la media ponderata più bassa di tutta
- 6) La data dove ci sono stati più esami

Mia soluzione:

```

select data, count(data) from universita.esame group by data having count(data) = (
    select MAX(conto) FROM (
        select data, count(data) as conto from universita.esame group by data
    ) AS USELESSNAME
)

```

Soluzione alternativa:

---

```

select data
from esame
group by data
having count(*) >= all(
    select esami_interni from (
        select data, count(*) as esami_interni
        from esame
    )
)

```





```
        group by data  
    ) as ConteggioEsami  
)
```

- 7) La media dei corsi che hanno con "programmazione" da qualunque parte

```
select avg(voto)  
      from universita.corso as cr, universita.esame as es  
     where cr.codice = es.codice_corso  
       and cr.nome like "%programmazione%"
```

```
SELECT avg(media) as media_totale  
FROM(  
    SELECT corso.nome AS nome, avg(voto) as media  
      FROM esame, corso  
     WHERE codice_corso=corso.codice AND corso.nome LIKE "%programmazione%"  
      GROUP BY corso.nome  
) AS prog;
```

- 8) Ordinare la data prima per data e poi per voto

```
select * from universita.esame  
order by data and voto
```

- 9) Le tipologie di corsi di laurea

```
select distinct tipologia from universita.corso_di_laurea
```

- 10) La provincia dove è stato fatto di più l'esame di basi di dati

```
select count(*) as cd, ct.provincia  
      from universita.esame as es, universita.studente as sd, universita.corso as cr, universita.citta as ct  
     where es.matricola_studente = sd.matricola  
       and cr.nome = "BASI DI DATI"  
       and ct.codice = sd.citta_residenza  
       and es.codice_corso = cr.codice  
      group by ct.provincia
```





```

    ⊖ having cd = (
    ⊖ select max(cd) from (
        select count(*) as cd
            from universita.esame as es, universita.studente as sd, universita.corso as cr, universita.citta as ct
        where es.matricola_studente = sd.matricola
            and cr.nome = "BASI DI DATI"
            and ct.codice = sd.citta_residenza
            and es.codice_corso = cr.codice
        group by ct.provincia
    ) AS unnamed
)

```

- 11) La città dove sono stati effettuati il numero maggiore di esami  
 /  
 12) Il cognome più comune tra tutti i ricercatori nella città più comune  
 13) Il cognome più comune

```

SELECT cognome, count(*) AS count
FROM (SELECT cognome, matricola
      FROM studente
      UNION
      SELECT cognome, matricola
      FROM docente) AS persone
GROUP BY cognome
HAVING count(*) >= ALL (
    SELECT count(*)
    FROM (SELECT cognome, matricola
          FROM studente
          UNION
          SELECT cognome, matricola
          FROM docente) AS persone
    GROUP BY cognome
)

```

```

select cognome, conto from (select dc.cognome, count(*) as conto from universita.docente as dc group by dc.cognome union (
    select sd.cognome, count(*) as conto from universita.studente as sd group by sd.cognome
)) AS useless2 HAVING conto >= (
select max(conto) from (
select dc.cognome, count(*) as conto from universita.docente as dc group by dc.cognome union (
    select sd.cognome, count(*) as conto from universita.studente as sd group by sd.cognome
)) as useless)

```

- **select cognome**
- from(**
- select cognome**





```
        from studente
    UNION ALL
    select cognome
    from docente
) as cognomi
group by cognome
order by count(*) desc
limit 1;
```

```
create view Cognomi as
select cognome from studente
union all
select cognome from docente;
```

```
select *
from Cognomi
group by cognome
having count(*) >= all(
    select count(*)
    from Cognomi
    group by cognome
)
```

- 14) La lista della media di tutti gli studenti

```
select st.nome, st.cognome, avg(voto)
      from universita.studente as st, universita.esame as es
     where st.matricola = es.matricola_studente
   group by st.matricola
```

- 15) Il corso più comune effettuato dagli studenti nella provincia di TO

1  
2 • SELECT cl.nome as corso, count(\*) as numero
3 FROM studente, corso\_di\_laurea AS cl, citta





```

4      WHERE citta_residenza=citta.codice AND corso_laurea=cl.codice AND citta.provincia="TO"
5      GROUP BY cl.nome
6      HAVING numero >= ALL (
7          SELECT count(*)
8          FROM studente, corso_di_laurea AS cl, citta
9          WHERE citta_residenza=citta.codice AND corso_laurea=cl.codice AND citta.provincia="TO"
0          GROUP BY cl.nome
1      );

```

---

*Proj<sub>nome,cognome,matricola</sub>(SEL<sub>voto>18 and data>10 10 2015</sub>(STUDENTE JOIN ESAME))*

*Proj<sub>nome,cognome,matricola</sub>(Studente) – Proj<sub>nome,cognome,matricola</sub>(roba sopra)*

*SEL<sub>matricolaPrimo≠matricolaSecondo</sub> (REN<sub>matricolaPrimo<-matr,nomePrimo<-Nome,cognome</sub>)*

Select Nome, count(\*) as numero from Viaggio, Porto

    Where portoPartenza = Nome

Group by Nome

UNION

Select Nome, count(\*) as numero from Viaggio, Porto

    Where portoPartenza = Nome

Group by Nome

Having numero = (

Select max(numero) from (

    Select Nome, count(\*) as numero from Viaggio, Porto

        Where portoPartenza = Nome

    Group by Nome

    UNION

    Select Nome, count(\*) as numero from Viaggio, Porto

        Where portoPartenza = Nome

    Group by Nome

)

)

Select nomeBarba from flotta

MINUS (

    Select distinct nomeBarca from VIAGGIO

)

Select portoPartenza, avg(cost) from VIAGGIO, CLIENTE, Porto

WHERE     Viaggio.idCliente = Cliente.idCliente

          Viaggio.portoPartenza = Porto.Nome

          Porto.nazione = Cliente.nazione

Group by Porto.nome

$Primo \leftarrow Cognome, data$   $Prima \leftarrow -data$  (*Studente*)  $JOINREN\dots$  (*Studente*) $\right)$



ԳՐԱՆՔ ԽԵՎՈՐԴԻ ԽՈՎՀԱՆՆԵԼ



