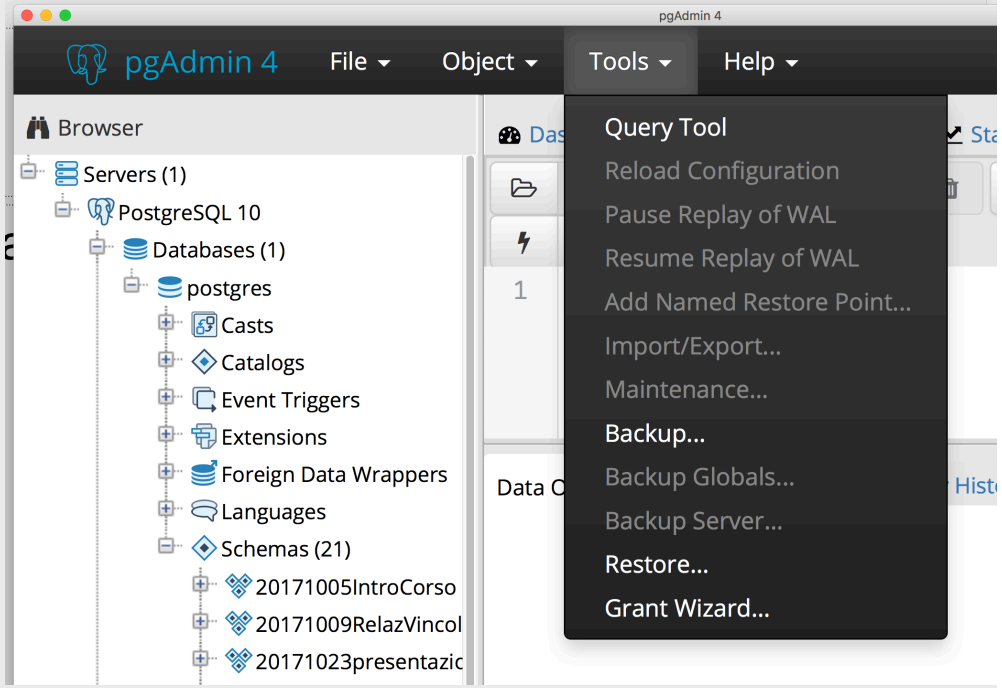
SQL

Esercitazioni pratiche

- Per SQL è possibile (e fondamentale) svolgere esercitazioni pratiche
- Verranno anche richieste come condizione per svolgere le prove parziali
- Soprattutto sono utilissime
- Si può utilizzare qualunque DBMS
  - IBM DB2, Microsoft SQL Server, Oracle, PostgresSQL o anche un servizio online (Sqliteonline)
- A lezione utilizziamo PostgresSQL e Sqliteonline

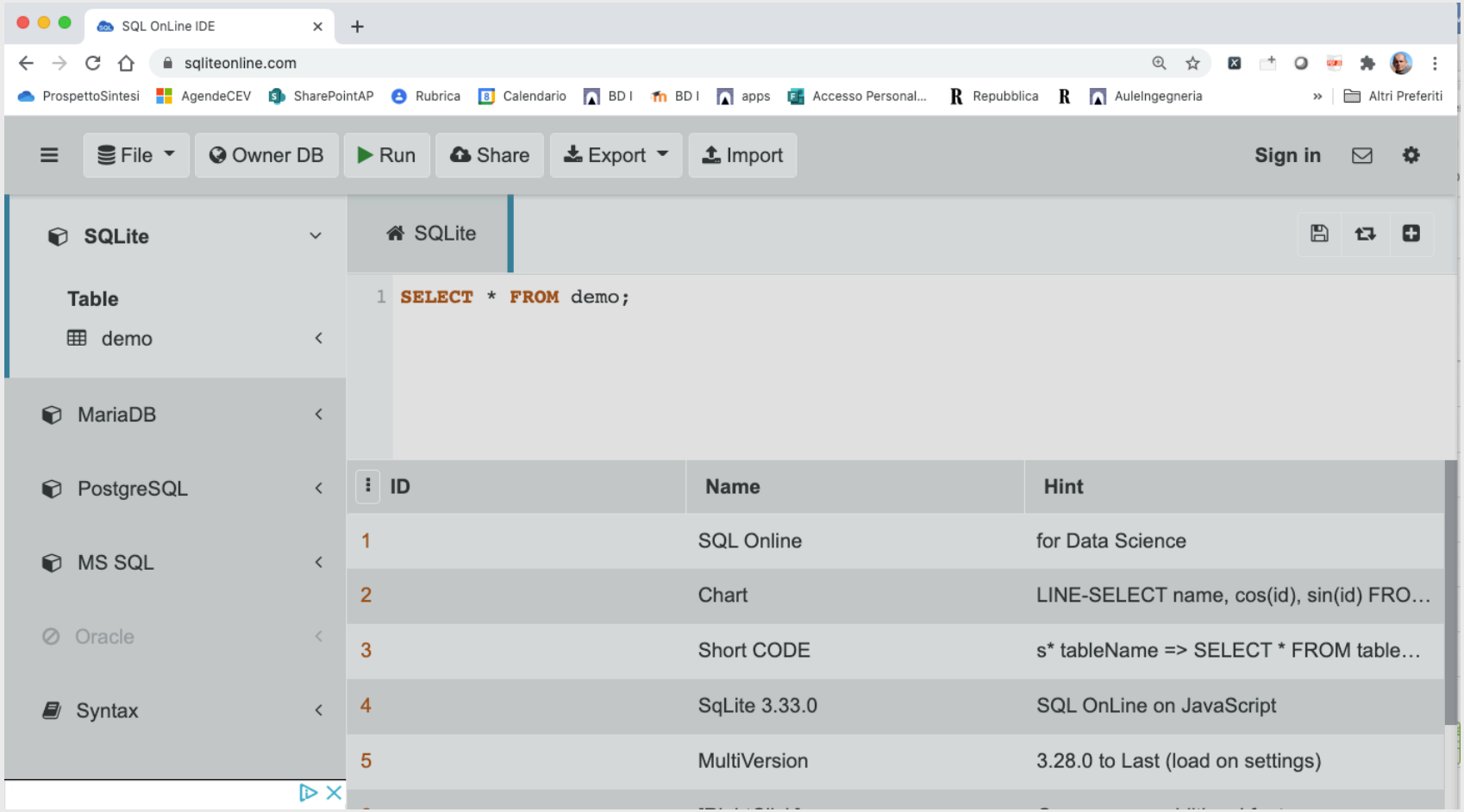
Come usare Postgres

- Scaricare
- Installare
- Lanciare pgAdmin
- Espandere l'albero a sinistra fino a "Schemas"
- Creare uno schema
- E poi
  - lavorare sugli elementi dello schema
  - oppure lanciare "Query tool" (SQL interattivo)



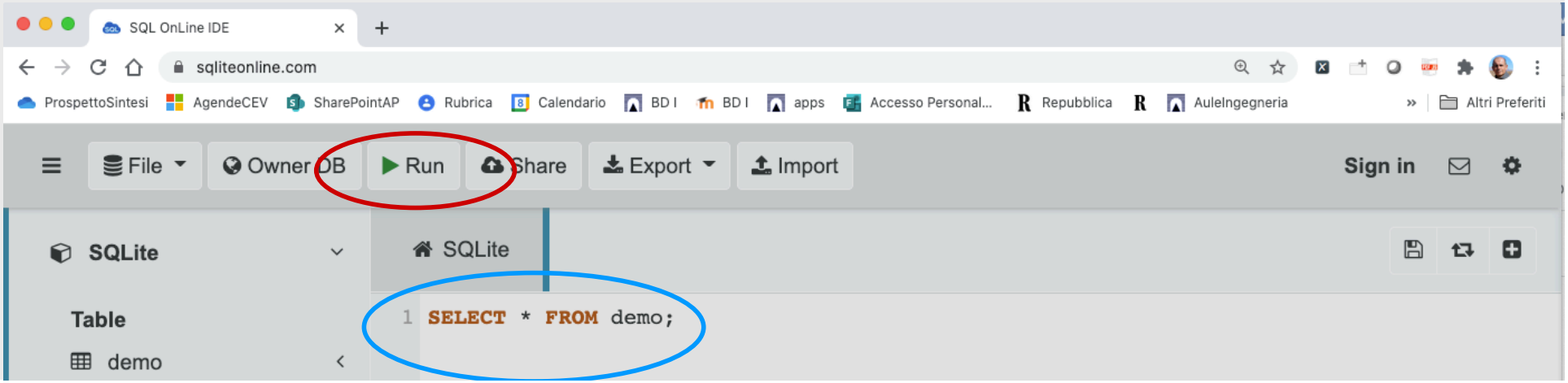
Come usare Sqliteonline

https://sqliteonline.com/



Come usare Sqliteonline

- I comandi vanno scritti nella finestra grande (al posto di
  `SELECT * FROM demo;`
- Vengono eseguiti premendo il pulsante Run
- Per inserire i dati si può caricare uno script (cioè un file di testo con una lista di comandi), come ad esempio (per I dati delle interrogazioni che vedremo fra poco):
  http://dia.uniroma3.it/~atzeni/didattica/BDN/20202021/protected/BD-04-esempiSQL2020%20persone.sql



CREATE TABLE, esempi

```
CREATE TABLE corsi(
    codice numeric NOT NULL PRIMARY KEY,
    titolo character(20) NOT NULL,
    cfu numeric NOT NULL)

CREATE TABLE esami(
    corso numeric REFERENCES corsi (codice),
    studente numeric REFERENCES studenti,
    data date NOT NULL,
   voto numeric NOT NULL,
   PRIMARY KEY (corso, studente))
```

La chiave primaria viene definita come NOT NULL anche se non lo specifichiamo (in Postgres)

DDL, in pratica

- In molti sistemi si utilizzano strumenti diversi dal codice SQL per definire lo schema della base di dati

SQL, operazioni sui dati

- interrogazione:
  - SELECT
- modifica:
  - INSERT, DELETE, UPDATE

Inserimento

(necessario per gli esercizi)

INSERT INTO Tabella [(Attributi)]
VALUES(Valori)

oppure

INSERT INTO Tabella [(Attributi)]
SELECT ...
(vedremo più avanti)

```
INSERT INTO Persone VALUES ('Mario',25,52)

INSERT INTO Persone(Nome, Reddito, Eta)
      VALUES('Pino',52,23)

INSERT INTO Persone(Nome, Reddito)
      VALUES('Lino',55)
```

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

Esercizi

- Definire la base di dati per gli esercizi
  - installare un sistema
  - creare lo schema
  - creare le relazioni (CREATE TABLE)
  - inserire i dati

  - eseguire le interrogazioni
    - suggerimento, per chi usa Postgres: usare schemi diversi

      set search_path to <*nome schema*>

```
create table persone (
  nome char (10) not null primary key,
  eta numeric not null,
  reddito numeric not null);
create table paternita (
  padre char (10) references persone,
  figlio char (10) primary key references persone);
...
insert into Persone values('Andrea',27,21);
...
insert into Paternita values('Sergio','Franco');
...
```

Vedere file:

http://dia.uniroma3.it/~atzeni/didattica/BDN/20202021/protected/BD-04-esempiSQL2020%20persone.sql

Vediamo gli esempi

- con RelaX

  http://dbis-uibk.github.io/relax/calc/gist/1362a9bb84dd2e4052561b714613b1de

- con Sqliteonline

  https://sqliteonline.com/

  con i dati

  http://dia.uniroma3.it/~atzeni/didattica/BDN/20202021/protected/BD-04-esempiSQL2020%20persone.sql

Istruzione SELECT (versione base)

SELECT ListaAttributi
FROM ListaTabelle
[WHERE Condizione]

- "target list"
- clausola FROM
- clausola WHERE

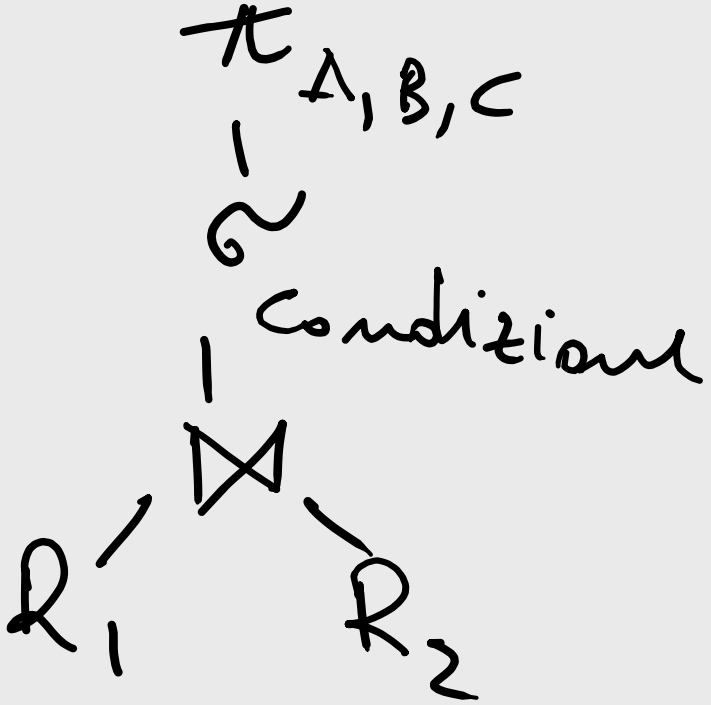
Intuitivamente

SELECT ListaAttributi
FROM ListaTabelle
[WHERE Condizione]

- Prodotto cartesiano di ListaTabelle
- Selezione su Condizione
- Proiezione su ListaAttributi

Intuitivamente

SELECT A, B, C
FROM R1, R2
[WHERE Condizione]



Una sola relazione

- Selezioni e proiezioni

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
```

Selezione, senza proiezione

- Nome, età e reddito delle persone con meno di trenta anni

$$SEL_{Eta<30}(Persone)$$

```
select *
from persone
where eta < 30
```

Proiezione, senza selezione

- Nome e reddito di tutte le persone

$$PROJ_{Nome,\ Reddito}(Persone)$$

```
select nome, reddito
from persone
```

Proiezione, con ridenominazione

- Nome e reddito di tutte le persone

$$REN_{Anni \leftarrow Eta}(PROJ_{Nome,\ Eta}(Persone))$$

```
select nome, eta as anni
from persone
```

Proiezione, attenzione

```
select
    madre
from maternita
```

```
select distinct
    madre
from maternita
```

Condizione complessa

```
select *
from persone
where reddito > 25
    and (eta < 30 or eta > 60)
```

Selezione, proiezione e join

- I padri di persone che guadagnano più di 20

$$\text{PROJ}_{\text{Padre}}(\text{paternita}\ \text{JOIN}_{\text{Figlio}=\text{Nome}}\ \text{SEL}_{\text{Reddito}>20}(\text{persone}))$$

```
select distinct padre
from persone, paternita
where figlio = nome and reddito > 20
```

Un commento

- In algebra relazionale

$$PROJ_{Padre}(paternita \; JOIN_{Figlio = Nome} \; SEL_{Reddito>20}(persone))$$

$$PROJ_{Padre}(SEL_{Reddito>20}((paternita \; JOIN_{Figlio = Nome} persone)))$$

Algebra e SQL

- In algebra possiamo scrivere un'intrerrogazione in più modi e ci sono differenze nell'efficienza
  - L'algebra è procedurale
- In SQL, possiamo dire che è il sistema che si preoccupa dell'efficienza
  - SQL è, almeno in parte, "dichiarativo"

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

$$\text{PROJ}_{\text{Nome, Reddito, RP}}\,(\text{SEL}_{\text{Reddito}>\text{RP}}\,(\text{REN}_{\text{NP,EP,RP}\leftarrow\text{Nome,Eta,Reddito}}(\text{persone})\ \text{JOIN}_{\text{NP=Padre}}\ (\text{paternita JOIN}_{\text{Figlio =Nome}}\ \text{persone})))$$

```
select f.nome, f.reddito, p.reddito
from persone p, paternita, persone f
where p.nome = padre and
   figlio = f.nome and
   f.reddito > p.reddito
```

SELECT, con ridenominazione del risultato

```
select figlio, f.reddito as reddito,
    p.reddito as redditoPadre
from persone p, paternita, persone f
where p.nome = padre and figlio = f.nome
      and f.reddito > p.reddito
```

Join esplicito

- Nella clausola FROM:
  - equijoin
    - R1 JOIN R2 ON R1.A = R2.B
  - Equijoin su attributi con lo stesso nome
    - R1 JOIN R2 USING (A)

Join esplicito

- Padre e madre di ogni persona

```
select madre, maternita.figlio, padre
from maternita, paternita
where maternita.figlio  = paternita.figlio

select madre, maternita.figlio, padre
from maternita join paternita on
     maternita.figlio = paternita.figlio

select madre, figlio, padre
from maternita join paternita using(figlio)
```

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

```
select f.nome, f.reddito, p.reddito as RedditoPadre
from persone p, paternita, persone f
where p.nome = padre and
   figlio = f.nome and
   f.reddito > p.reddito

select f.nome, f.reddito, p.reddito as RedditoPadre
from persone p join paternita on p.nome = padre
                    join persone f on figlio = f.nome
where  f.reddito > p.reddito
```

- Le persone che guadagnano più dei rispettivi padri; mostrare nome, reddito e reddito del padre

```
select f.nome, f.reddito, p.reddito as RedditoPadre
from persone p, paternita, persone f
where p.nome = padre and
   figlio = f.nome and
   f.reddito > p.reddito

select f.nome, f.reddito, p.reddito as RedditoPadre
from persone p join paternita on p.nome = padre
                  join persone f on figlio = f.nome
where f.reddito > p.reddito
```

Join esterno: "outer join"

- Padre e, se nota, madre di ogni persona

```
select paternita.figlio, padre, madre
from paternita left join maternita
    on paternita.figlio = maternita.figlio

select figlio, padre, madre
from paternita left join maternita using(figlio)

select paternita.figlio, padre, madre
from paternita full join maternita using(figlio)
```

Note:

- left outer, full outer, right outer equivalenti a left, full, right
- sqliteonline non supporta full e right;

Ordinamento del risultato

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

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

Espressioni nella target list

```
select Nome, Reddito/12 as redditoMensile
from Persone
```

Attenzione al tipo – guardatelo da soli (ma non è importante ai fini dell'esame)

Condizione “LIKE”

- Le persone che hanno un nome che inizia per 'A' e ha una 'd' come terza lettera

```
select *
from persone
where nome like 'A_d%'
```

Gestione dei valori nulli

Persone

Nome	Età	Reddito
Andrea	27	21
…	…	…
Luisa	75	87
Nicola	43	NULL

- Le persone il cui reddito è o potrebbe essere maggiore di 40

SEL (Reddito > 40) OR (Reddito IS NULL) (Impiegati)

Gestione dei valori nulli

Persone

Nome	Età	Reddito
Andrea	27	21
…	…	…
Luisa	75	87
Nicola	43	NULL

- Le persone il cui reddito è o potrebbe essere maggiore di 40

```
SELECT * FROM Persone
WHERE Reddito > 40 OR Reddito IS null
```

Unione

```
select A,  B
from R
union
select A , B
from S
```

```
select A,  B
from R
union all
select A , B
from S
```

Operazione non commutativa (in molti sistemi)

```
select padre, figlio
from paternita
union
select madre, figlio
from maternita
```

```
select madre, figlio
from maternita
union
select padre, figlio
from paternita
```

	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo
Luisa	Maria
Luisa	Luigi
Anna	Olga
Anna	Filippo
Maria	Andrea
Maria	Aldo

Notazione posizionale!

```
select padre, figlio
from paternita
union
select madre, figlio
from maternita
```

Notazione posizionale, 2

```
select padre, figlio
from paternita
union
select figlio, madre
from maternita
```

NO!
Funziona, ma produce un risultato indesiderabile

```
select padre, figlio
from paternita
union
select madre, figlio
from maternita
```

OK

Notazione posizionale, 3

- Anche con le ridenominazioni non cambia niente:

```
select padre as genitore, figlio
from paternita
union
select figlio, madre as genitore
from maternita
```

- Corretta:

```
select padre as genitore, figlio
from paternita
union
select madre as genitore, figlio
from maternita
```

Differenza

```
select Nome
from Impiegato
except
select Cognome as Nome
from Impiegato
```

Intersezione

```
select Nome
from Impiegato
intersect
select Cognome as Nome
from Impiegato
```

Un'altra anomalia degli operatori inseimistici

```
select padre
from paternita
union
select madre
from maternita
```

```
select padre
from paternita
union
select padre
from paternita
```

Un'altra anomalia degli operatori inseimistici

```
select padre
from paternita
union all
select madre
from maternita
```

Operatori aggregati: COUNT

- Il numero di figli di Franco

γ count(*) $\rightarrow$ NumFigliDiFranco (σ Padre = 'Franco' (Paternita))

```
select count(*) as NumFigliDiFranco
from Paternita
where Padre = 'Franco'
```

COUNT DISTINCT

```
select count(*) from persone

select count(reddito) from persone

select count(distinct reddito) from persone
```

Altri operatori aggregati

- SUM, AVG, MAX, MIN

- Media dei redditi dei figli di Franco

$$\gamma\ \text{avg(Reddito)} \rightarrow \text{RedditoMedioFigliDiFranco}$$
$$(\sigma\ \text{Padre = 'Franco'}\ (\text{Paternita}) \bowtie \text{Figlio=Nome Persone})$$

```
select avg(reddito) redditoMedioFigliDiFranco
from persone join paternita on nome=figlio
where padre='Franco'
```

Operatori aggregati e valori nulli

```
select avg(reddito) as redditomedio
from persone
```

Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

  γ Padre; count(*) → NumFigli (Paternita)

  ```
  select Padre, count(*)  AS NumFigli
  from paternita
  group by Padre
  ```

- Gli attributi nella target list (Padre) debbono comparire nella GROUP BY
- Purtroppo in alcuni sistemi (come sqliteonlite) questo non accade

Condizioni sui gruppi

- I padri i cui figli hanno un reddito medio maggiore di 25; mostrare padre e reddito medio dei figli

```
select padre, avg(f.reddito) as redditomedio
from persone f join paternita on figlio = nome
group by padre
having avg(f.reddito) > 25
```

Un errore "classico"

- La persona con il reddito massimo

```
select nome, max(reddito)
from persone
```

- NO!! Cerchiamo di mettere insieme una ennupla con una aggregazione

Purtroppo

- In alcuni sistemi (es. Sqliteonline) funziona

```
select nome, max(reddito)
from persone
```

- Ma concettualmente è scorretto: cerca di mettere insieme una ennupla con una aggregazione
- Vediamo una cosa simile:
  - "Le persone con reddito superiore alla media"

Operatori aggregati e target list

- un'interrogazione scorretta:

```
select nome, max(reddito)
from persone
```

- di chi sarebbe il nome? La target list deve essere omogenea

```
select min(eta), avg(reddito)
from persone
```

Proviamo …

- Si può fare con i costrutti di SQL che conosciamo
- con l'aiuto di una vista (concetto che non abbiamo ancora discusso – lo facciamo subito)

- "Le persone con reddito superiore alla media"
  - trovare il reddito medio
  - confrontare ciascun reddito con il reddito medio (prodotto cartesiano + selezione)

Viste

```
CREATE VIEW V AS
   SELECT …
```

anche (non in tutti i sistemi)

```
CREATE VIEW V AS
  SELECT …
  UNION
  SELECT …
```

```
CREATE OR REPLACE VIEW V AS
   SELECT …
```

I vari passi

```
create view  maxReddito
as select max(reddito) redditomax
   from persone;

select *
from persone, maxReddito;

select nome, reddito
from persone, maxReddito
where reddito = redditomax
```

Analogamente, i redditi superiori alla media

```
create view  mediaReddito
as select avg(reddito) as redditoMedio
   from persone;

select *
from persone, mediaReddito;

select nome, reddito, redditomedio
from persone, mediaReddito
where reddito > redditoMedio;
```

Interrogazioni nidificate (nested query o subquery)

- Varie forme di nidificazione
  - nella WHERE
  - nella FROM
  - nella SELECT

- Coerente con i tipi
  - anche Booleano (EXISTS)

Nella WHERE

- La persona che guadagna più di tutte le altre

```
select *
from persone
where reddito = (    select max(reddito)
                     from persone)
```

Nella WHERE

- Le persone che guadagnano più della media

```
select *
from persone
where reddito >= (  select avg(reddito)
                    from persone)
```

Correlated subquery

- Per ogni padre, il figlio che guadagna di più

```
select  padre, figlio, reddito
from persone  join paternita p on nome = figlio
where reddito = (       select max(reddito)
                        from persone join paternita
                                on nome = figlio
                        where p.padre = padre )
```

- L'interrogazione interna viene eseguita una volta per ciascuna ennupla della FROM esterna

Per semplificare, usiamo una vista

```
CREATE VIEW PersoneConPadre
      AS    select *
            from persone join paternita
                  on nome = figlio
```

Correlated subquery

- Per ogni padre, il figlio che guadagna di più

```
select *
from personeconPadre p
where reddito = (select max(reddito)
                 from personeconPadre
                 where p.padre = padre )
```

- L'interrogazione interna viene eseguita una volta per ciascuna ennupla della FROM esterna
- Per ogni ennupla p di personeconPadre viene eseguita

```
select max(reddito)
from personeconPadre
where p.padre = padre
```

in cui p.padre è una costante e il valore m risultante viene utilizzato in

```
select *
from personeconPadre p
where reddito = m
```

Altre nidificazioni nella WHERE

- La motivazione originaria per la nidificazione

- nome e reddito del padre di Franco

```
select Nome, Reddito
from Persone join Paternita on Nome = Padre
where Figlio = 'Franco'

select Nome, Reddito
from Persone
where Nome =     (   select Padre
                     from Paternita
                     where Figlio = 'Franco')
```

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
      and F.Reddito > 20

select Nome, Reddito
from Persone
where Nome in (select Padre
               from Paternita
               where Figlio = any (select Nome
                                   from Persone
                                   where Reddito > 20))
```

notare la distinct

- In questo caso la nidificazione non aiuta molto

- Nome e reddito dei padri di persone che guadagnano più di 20

```
select distinct P.Nome, P.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
      and F.Reddito > 20
```

```
select Nome, Reddito
from Persone
where Nome in (select Padre
               from Paternita, Persone
               where Figlio = Nome
               and Reddito > 20)
```

- Nome e reddito dei padri di persone che guadagnano più di 20, con indicazione del reddito del figlio

```
select distinct P.Nome, P.Reddito, F.Reddito
from Persone P, Paternita, Persone F
where P.Nome = Padre and Figlio = F.Nome
      and F.Reddito > 20

select Nome, Reddito, ????
from Persone
where Nome in (select Padre
               from Paternita
               where Figlio = any (select Nome
                                   from Persone
                                   where Reddito > 20))
```

EXISTS

- Quantificatore esistenziale
- Correlazione fra la sottointerrogazione e le variabili nel resto

- Le persone che hanno almeno un figlio

```
select *
from Persone
where exists (    select *
                  from Paternita
                  where Padre = Nome) or
      exists (    select *
                  from Maternita
                  where Madre = Nome)
```

Disgiunzione (OR) e unione

```
select distinct Persone.*
from Persone join Paternita on padre= nome
union
select distinct Persone.*
from Persone join Maternita on madre= nome
```

- I padri i cui figli guadagnano tutti più di 20

```
select distinct Padre
from Paternita Z
where not exists (
            select *
            from Paternita W, Persone
            where W.Padre = Z.Padre
                and W.Figlio = Nome
                and Reddito <= 20)
```

- I padri i cui figli guadagnano tutti più di 20

```
select distinct padre
from paternita
except
select distinct padre
from paternita join persone on figlio = nome
where reddito <= 20
```

- I padri i cui figli guadagnano tutti più di 20

```
select distinct Padre
from Paternita
where not exists ( select *
                   from Persone
                   where Figlio = Nome
                      and Reddito <= 20)
```

NO!!! provare ad eseguire per vedere la differenza

- I padri i cui figli guadagnano tutti più di 20

```
select distinct Padre
from Paternita
where not exists (
                    select *
                    from Persone
                    where Reddito <= 20)
```

NO!!! provare anche questa

Nidificazione nella FROM

- Per ogni padre, il figlio che guadagna di più

```
select p.*
from personeconPadre p join
          (  select padre, max(reddito) as maxreddito
             from personeconPadre
             group by padre
           ) as m on p.padre = m.padre
where reddito = maxreddito
```

Ancora nella FROM

- Tutte le persone, con il reddito massimo dei figli dello stesso padre

```
select p.*, maxreddito
from personeconPadre p ,
            (  select padre, max(reddito) as maxreddito
               from personeconPadre
               group by padre
            ) as m
where p.padre = m.padre
```

Nidificazione nella SELECT

- Calcolo di valori con la nidifcazione
- Per ogni padre, tutti i dati e il reddito massimo dei suoi figli (correlazione)

```
select distinct padre, (select max(reddito)
                        from paternita join persone
                             on figlio = nome
                        where padre = p.padre)
from paternita p join persone on padre =nome
```

Operazioni di aggiornamento

```
INSERT INTO Persone VALUES ('Mario',25,52)

INSERT INTO Persone(Nome, Reddito, Eta)
       VALUES('Pino',52,23)

INSERT INTO Persone(Nome, Reddito)
       VALUES('Lino',55)

INSERT INTO Persone ( Nome )
SELECT Padre
FROM Paternita
WHERE Padre NOT IN (SELECT Nome
                    FROM Persone)
```

Eliminazione di ennuple

```
DELETE FROM Tabella
  [ WHERE Condizione ]
```

```
DELETE FROM Persone
WHERE Eta < 35

DELETE FROM Paternita
WHERE Figlio NOT in (     SELECT Nome
                          FROM Persone)

DELETE FROM Paternita
```

Modifica di ennuple

```
UPDATE NomeTabella
SET  Attributo = < Espressione |
                   SELECT … |
                   NULL |
                   DEFAULT >
        [ WHERE Condizione ]
```

```
UPDATE Persone SET Reddito = 45
  WHERE Nome = 'Piero'

UPDATE Persone
  SET Reddito = Reddito * 1.1
  WHERE Eta < 30
```

Altre operazioni DDL

- Aggiungere vincoli, con verifica (provate)

```
alter table persone
    add constraint redditononnegativo
        check (reddito >=0)

alter table persone
    add constraint redditoesagerato
        check (reddito >=100)
```