

# Una prima interrogazione con selezione, proiezione e join

- I padri di persone che guadagnano più di 20

```
PROJPadre(paternita  
  JOINFiglio = Nome  
  SELReddito > 20(persone))
```

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

# I tre passi

```
select *  
from persone, paternita
```

**Prodotto  
cartesiano**

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

**Selezione**

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

**Proiezione**

- Vediamoli in azione

# Un commento

- In algebra relazionale

```
PROJPadre(paternita  
  JOINFiglio =Nome  
  SELReddito>20(persone))
```

```
PROJPadre (  
  SELReddito>20 (  
    (paternita JOINFiglio =Nomepersone)))
```

o anche

```
PROJPadre (  
  SELFiglio =Nome AND Reddito>20 (  
    (paternita JOIN persone)))
```

# Algebra e SQL

- In algebra possiamo scrivere un'interrogazione 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"

# Versione originaria di SQL

- Si specifica
  - prodotto cartesiano
  - selezione
  - proiezione
- Si esegue
  - selezione
  - join (e ulteriore selezione)
  - proiezione

# Qualche anno dopo

- È stato introdotto il join esplicito

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

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

# Join esplicito

- Nella clausola FROM:
  - equijoin
    - `R1 JOIN R2 ON R1.A = R2.B`

# Un'osservazione

- Con nomi di attributi diversi, è implicito quale sia la relazione cui appartiene un attributo
  - Ma se non sono diversi?
    - Serve specificare



# Necessità di distinzione

- Per ogni persona per la quale entrambi i genitori sono nella base di dati, mostrarli
- Intuitivamente
  - Join di maternità e paternità
  - Ma "Figlio" compare in entrambe
  - Possiamo "premettere" il nome della relazione

```
SELECT Padre, Madre, Paternita.Figlio  
FROM Paternita JOIN Maternita  
ON Paternita.Figlio = Maternita.Figlio
```

# Join naturale

- 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 naturale

```
SELECT *  
FROM Paternita JOIN Maternita  
USING (Figlio)
```

# Necessità di distinzione, ancora

- Può servire di avere più volta la stessa relazione in una interrogazione (join di una relazione con se stessa)
- Si utilizzano variabili (chiamate "alias" in SQL)

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

```

PROJNome, Reddito, RP (SELReddito>RP
(RENNP,EP,RP ← Nome, Eta, Reddito (persone)
      JOINNP=Padre
(paternita JOINFiglio =Nome persone)))

```

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

```

PROJNome, Reddito, RP (SELReddito>RP
(RENNP,EP,RP ← Nome, Eta, Reddito (persone)
      JOINNP=Padre
(paternita JOINFiglio =Nome 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

```

# Meglio 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
```



- 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 i dettagli non sono importanti 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

<u>Nome</u>	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

<u>Nome</u>	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
```

# Operatori aggregati: COUNT

- Il numero di figli di Franco

$\gamma$  count(\*)  $\rightarrow$  NumFigliDiFranco ( $\sigma$  Padre = 'Franco' (Paternita))



# Operatori aggregati: COUNT

- Il numero di figli di Franco

$\gamma$  count(\*)  $\rightarrow$  NumFigliDiFranco ( $\sigma$  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

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

## Altri operatori aggregati

- SUM, AVG, MAX, MIN
- Media dei redditi dei figli di Franco

$\gamma$  avg(Reddito)  $\rightarrow$  RedditoMedioFigliDiFranco  
( $\sigma$  Padre = 'Franco' (Paternita)  $\bowtie$  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

$\gamma$  Padre; count(\*)  $\rightarrow$  NumFigli (Paternita)

# Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

$\gamma$  Padre; count(\*)  $\rightarrow$  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
- Proviamo con PostgreSQL

# 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"
  - troviamo il reddito medio (vista)
  - confrontiamo 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;
```