

Aggregazione

- Il contenuto delle basi di dati viene spesso aggregato:
 - Il numero di figli di ciascun padre
 - Lo stipendio massimo
 - Lo stipendio medio dei figli di ciascun padre

Operatori aggregati

- Nelle espressioni della target list possiamo avere anche espressioni che calcolano valori a partire da insiemi di ennuple:
 - conteggio, minimo, massimo, media, totale
 - sintassi base (semplificata):

Funzione ([DISTINCT] *)

Funzione ([DISTINCT] Attributo)

--- questo e' un commento

drop table if exists paternita;
drop table if exists maternita;
drop table if exists persone;

```
create table persone(  
    nome            character(20) not null primary key,  
    eta             integer,  
    reddito         integer);
```

```
insert into persone values('Andrea',27,21);  
insert into persone values('Aldo',25,15);  
insert into persone values('Maria',55,42);  
insert into persone values('Anna',50,35);  
insert into persone values('Filippo',26,30);---  
insert into persone values('Luigi',50,40);  
insert into persone values('Franco',60,20);  
insert into persone values('Olga',30,41);  
insert into persone values('Sergio',85,35);  
insert into persone values('Luisa',75,87);
```

```
create table maternita(  
    madre           character(20) not null references persone(nome),  
    figlio          character(20) not null primary key references persone(nome)  
);
```

```
insert into maternita values('Luisa','Maria');  
insert into maternita values('Luisa','Luigi');  
insert into maternita values('Anna','Olga');  
insert into maternita values('Anna','Filippo');  
insert into maternita values('Maria','Andrea');  
insert into maternita values('Maria','Aldo');
```

```
create table paternita(  
    padre           character(20) not null references persone(nome),  
    figlio          character(20) not null primary key references persone(nome)  
);
```

```
insert into paternita values('Sergio','Franco');  
insert into paternita values('Luigi','Olga');  
insert into paternita values('Luigi','Filippo');  
insert into paternita values('Franco','Andrea');  
insert into paternita values('Franco','Aldo');
```

Operatori aggregati: COUNT

- Il numero di figli di Franco

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

Operatori aggregati: COUNT

- Il numero di figli di Franco

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

- Come viene eseguita?
 - prima l'interrogazione come se la target list fosse
`select *`
 - poi si applica la funzione aggregativa

Con valori nulli

```
insert into persone(nome, eta) values('Michele',75);
```

```
select count(*) from persone
```

```
select count(reddito) from persone
```

COUNT DISTINCT

```
select count(*) from persone
```

```
select count(reddito) from persone
```

```
select count(distinct reddito) from persone
```

Altri operatori aggregati

- SUM, AVG, MAX, MIN
- Stessa idea, applicare l'aggregazione al risultato della `select *`
- Media dei redditi dei figli di Franco

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

Di nuovo, i valori nulli

```
select avg(reddito) as redditomedio  
from persone  
where nome like 'M%'
```

- Si vede perché è utile il valore "speciale"
- Che cosa avremmo dovuto faer se avessimo usato 0 o -1 per un reddito non noto?

Valore "-1"

```
delete from persone where nome ='Michele';  
insert into persone values('Michele',75,-1);
```

```
select avg(reddito) as redditomedio  
from persone  
where nome like 'M%'
```

```
select avg(reddito) as redditomedio  
from persone  
where nome like 'M%' and reddito <> -1
```

Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

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

Semantica di interrogazioni con operatori aggregati e raggruppamenti

1. interrogazione senza **group by** e senza operatori aggregati

select *

from paternita

2. si raggruppa e si applica l'operatore aggregato a ciascun gruppo

Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

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

paternita

Padre	Figlio
Sergio	Franco
Luigi	Olga
Luigi	Filippo
Franco	Andrea
Franco	Aldo

Padre	NumFigli
Sergio	1
Luigi	2
Franco	2

Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

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

- Attenzione, è opportuno che gli attributi di raggruppamento siano scritti sia nella group by sia nella target list (altrimenti il risultato non è comprensibile)
- vediamo

Operatori aggregati e raggruppamenti

- Il numero di figli di ciascun padre

```
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, con risultati senza senso**

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
- Ma in effetti, non funziona se ci sono due persone con lo stesso valore massimo
`insert into persone values('Mauro',60,87);`
- Un problema 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
```

```
select nome, 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)
 - oppure con le "interrogazioni nidificate"

Proviamo ...

- Si può fare
 - **con i costrutti di SQL che conosciamo**
 - **con l'aiuto di una vista, vediamo la sintassi**
 - oppure con le "interrogazioni nidificate"

Una soluzione

- "Le persone con reddito superiore alla media"
 - troviamo il reddito medio (vista)
 - Una tabella con un attributo (la media) ed una ennupla (il valore della media)
 - 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 (persone con reddito superiore alla media)

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

Mostrare su [sqliteonline](#)

```
select *  
from persone, mediaReddito;
```

Mostrare su [sqliteonline](#)

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

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)
 - **oppure con le "interrogazioni nidificate"**

Interrogazioni nidificate (nested query o subquery)

- Varie forme di nidificazione
 - nella WHERE
 - nella FROM
 - nella SELECT
- Coerente con i tipi
 - anche Booleano (EXISTS)
 - non lo vediamo quest'anno
 - attenzione al rapporto fra singoli valori e insiemi

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)
```

Attenzione alla visibilità degli attributi

- Posso inserire la media (o il massimo nella target list)?

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 padre = p.padre )
```

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

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 )
```

**Mostrare su
sqliteonline**

- 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
```

dove **p.padre** è una costante e il valore ottenuto (chiamiamolo **m**) viene utilizzato nell'interrogazione "esterna"

```
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**

- 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)
```

Altre forme di nidificazione

- Le vediamo più per curiosità che per altro

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 nidificazione
- 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

- Vediamo solo aspetti base, per sapere operare, i dettagli non interessano e ci sono ben pochi concetti importanti

```
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)
```