SQL

Università degli Studi di Salerno

Corso di Laurea in Scienze della Comunicazione

Informatica generale (matr. Dispari)

Docente: Angela Peduto

A.A. 2007/2008

Interrogazioni con raggruppamento

- E' possibile **ripartire le tuple in gruppi omogenei ed applicare gli operatori aggregati ai singoli gruppi**, aggiungendo in coda alla SELECT-FROM-WHERE la clausola GROUP BY

  GROUP BY $A_1,\ldots A_k$
  HAVING Ψ

- Ψ è una clausola opzionale che si applica ai gruppi $A_1,\ldots A_k$ (per selezionare solo un sottoinsieme dei gruppi)
- In tal modo **due tuple si trovano nello stesso gruppo se e solo se esse sono uguali componente per componente sugli attributi $A_1,\ldots A_k$**
- Gli attributi $A_1,\ldots A_k$ devono comparire anche nella clausola SELECT

Interrogazioni con raggruppamento

Gli operatori aggregati vengono applicati a tutte le righe che vengono prodotte come risultato dell'operazione.

Può essere necessario applicare l'operatore solo ad un sottoinsieme delle righe.

L'operatore **group by** specifica come suddividere le tabelle in sottoinsiemi.

Es.

```
SELECT    Età, AVG(Reddito) AS
Reddito_medio_età
FROM      Persone
GROUP BY Età
```


Esempio GROUP BY

SELECT Età, AVG(Reddito) AS Reddito_medio_età
FROM Persone
GROUP BY Età

Età e stipendio medio dei dipendenti, raggruppati per età

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

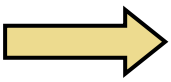
Età	Reddito_medio_età
25	15000000
26	0
27	21000000
30	25000000
50	37500000
55	42000000
60	20000000
75	87000000
85	35000000

Esempio GROUP BY

SELECT ***Età, AVG(Reddito) AS 'Reddito Medio >45'***
FROM ***Persone***
WHERE ***Età > 45 GROUP BY Età***

Età e stipendio medio dei dipendenti di età superiore a 45 anni, raggruppati per età

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



Età	'Reddito Medio >45'
50	37500000
55	42000000
60	20000000
75	87000000
85	35000000

Operatori aggregati e raggruppamenti

- I padri con il relativo numero di figli che ha ciascuno

```
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

Condizioni sui gruppi

Può essere anche necessario restringere i gruppi attraverso l'applicazione di condizioni.

Se le condizioni sono verificabili a livello delle singole righe, basta utilizzare la clausola **`where`**, altrimenti si aggiunge una condizione alla **`group by`** attraverso l'estensione **`having`**

```
SELECT    Età, AVG(Reddito) AS 'Stipendio Medio'
FROM      Persone
          GROUP BY Età  HAVING  Count(Età)>1
```

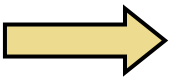
Se non si specifica **`group by`** e si usa **`having`** da solo la condizione è applicata a tutte le righe. Il problema è che se la condizione non è verificata, il risultato sarà vuoto.

Esempio Raggruppamento - HAVING

SELECT Età, AVG(Reddito) AS 'Stipendio Medio'
FROM Persone
GROUP BY Età HAVING Count(Età)>1

Età e Stipendio medio dei dipendenti raggruppati per età, con almeno 2 dipendenti per gruppo

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



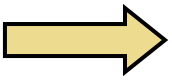
Età	Stipendio_Medio
50	37500000

Esempio Raggruppamento - HAVING

SELECT Età, AVG(Reddito) AS Reddito_Medio
FROM Persone
GROUP BY Età
HAVING AVG(Reddito)>25000000

Stipendi medi dei dipendenti che guadagnano più di 25.000.000 euro, raggruppati per età

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



Età	Stipendio_Medi
50	37500000
55	42000000
75	87000000
85	35000000

WHERE o HAVING?

- I padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 20

```
select padre, avg(f.reddito)
from persone f join paternita on
                        figlio = nome
where eta < 30
group by padre
having avg(f.reddito) > 25
```

Select

La forma di select cui siamo arrivati dopo le estensioni viste è quindi:

SelectSQL **::= select** *ListaAttributiOEspressioni*
from *ListaTabelle*
[**where** *CondizioniSemplici*]
[**group by** *ListaAttributiDiRaggruppamento*]
[**having** *CondizioniAggregate*]
[**order by** *ListaAttributiDiOrdinamento*]

SELECT Nidificate (Sub-Queries)

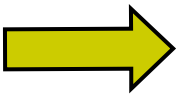
- SQL permette di **usare insiemi come operandi**.
- Gli insiemi sono definiti tramite **istruzioni complete SELECT nidificate all'interno di una clausola WHERE**.
- Le SELECT nidificate vengono dette sub-Queries.
- **Viene eseguita prima la Select più interna** e poi i suoi risultati sono utilizzati dalla Select esterna

Esempi di Sub Queries

```
SELECT   Nome,  Reddito
FROM     Persone
WHERE    Nome IN
         (SELECT   Padre FROM Paternità)
```

Individua Nome e Reddito dei padri

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



Nome	Reddito
Franco	20.000.000
Luigi	40.000.000
Sergio	35.000.000
	0

Modo Alternativo equivalente alla precedente Query:

```
SELECT    DISTINCT Nome,  Reddito
FROM      Persone, Paternità
WHERE     Persone.Nome=Paternità.padre
```


Seleziona Nome e Reddito di chi guadagna più di Luigi

```
SELECT P2.Nome, P2.Reddito
FROM Persone AS P1, Persone AS P2
WHERE P1.Nome = 'Luigi'     AND
            P2.Reddito > P1.Reddito;
```

```
SELECT Nome, Reddito
FROM Persone
WHERE Reddito > (Select Reddito
                 From Persone
                 Where Nome like 'Luigi');
```

Interrogazioni di tipo insiemistico

SQL fornisce anche gli operatori di tipo insiemistico
`union, intersect, except (minus)`
con la seguente sintassi:
SelectSQL
`{< union | intersect | except > [all]}` *SelectSQL*

NB
• **`intersect`** e **`except`** potrebbero anche essere derivati attraverso opportune query
• gli operatori insiemistici eseguono per default una eliminazione dei duplicati; **`all`** specifica di non farla
• gli schemi su cui si opera non devono essere identici ma avere uguale numero di attributi, con domini compatibili. La corrispondenza è per posizione.

SELECT Nidificate (Sub-Queries)

- È possibile utilizzare le sub queries per:
  - **Verificare l'esistenza di alcuni risultati della sub query** utilizzando le parole riservate EXISTS o NOT EXISTS.
  - **Trovare qualsiasi valore nella query principale che sia uguale, maggiore o minore dei valori restituiti dalla sub query**, utilizzando le parole riservate ANY, IN o ALL.
  - Creare sottoquery all'interno di sub queries , ossia sub queries nidificate.

Unione di Select

- 2 o più SELECT possono essere combinate tramite la clausola **UNION**
- Il risultato contiene sia i records **selezionati dalla prima SELECT che quelli selezionati dalla seconda**
- Il nome della/e colonna/e è quello determinato nella prima Select
- L'UNION è possibile solo se **le Select da unire presentano gli stessi campi (o campi compatibili) nello stesso ordine**

Interrogazioni di tipo insiemistico: UNION

SELECT Padre AS Genitore
FROM Paternità
UNION
SELECT Madre
FROM Maternità

Chi sono i genitori?

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

Figlio	Madre
Filippo	Anna
Olga	Anna
Luigi	Luisa
Maria	Luisa
Andrea	Maria

Genitore
Anna
Franco
Luigi
Luisa
Maria
Sergio

Interrogazioni di tipo insiemistico: UNION

```
SELECT  Madre AS Genitore, Figlio
FROM    Maternità
  UNION
SELECT  Padre, Figlio
FROM    Paternità
```

Qual è l'insieme dei Padri e delle Madri, con i nomi dei rispettivi figli?

Figlio	Madre
Filippo	Anna
Olga	Anna
Luigi	Luisa
Maria	Luisa
Andrea	Maria

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

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

Interrogazioni nidificate

E' possibile anche realizzare clausole **`where`** in cui il confronto non avviene fra predicati semplici o fra valori, ma fra valori e risultati di interrogazioni.

Tipicamente, il risultato dell'interrogazione con cui si fa il confronto è un attributo (il confronto deve avvenire fra entità omogenee).

Sorge il problema di confrontare un valore con un insieme di valori (il risultato della interrogazione).

SQL offre 2 possibilità per estendere i normali operatori di confronto al caso del confronto valore/risultato query:

`all` specifica che il confronto è vero se è vero per tutte le righe del risultato dell'interrogazione.

`any` specifica che il confronto è vero se è vero per una qualunque delle righe del risultato dell'interrogazione.

Interrogazioni Nidificate

```
select *
   from Impiegato
   where Dipart = any (select Nome
                       from Dipartimento
                       where Città='Firenze')
```

Il risultato è una tabella che comprende tutte le righe di IMPIEGATO per cui il valore Dipart è uguale ad almeno uno dei valori di Nome in DIPARTIMENTO, limitatamente alle tuple per cui Citta'='Firenze'.

Lo stesso risultato si poteva ottenere con un join, ma così, specialmente per interrogazioni complesse, è più leggibile.

Massimo e nidificazione

- La persona (o le persone) con il reddito massimo

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

Interrogazioni Nidificate

Visibilità delle variabili

Le variabili SQL sono utilizzabili solo nell'ambito della query in cui sono definite o nell'ambito di una query nidificata all'interno di essa.

Se due query sono allo stesso livello non possono condividere variabili.

`exists`

è un operatore logico applicabile a query nidificate. Restituisce vero se la query dà un risultato non nullo, falso se è nullo.

E' utilizzabile in modo significativo solo se esiste un passaggio di binding fra interrogazione esterna e interrogazione nidificata.

ES: 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)
```

Manipolazione dei dati

I comandi SQL che permettono di modificare il contenuto di una base di dati sono

`insert` **`delete`** **`update`**

Quando una base di dati viene creata con i comandi del DDL inizialmente è vuota.

`insert` ha la seguente sintassi

`insert into` *NomeTabella* [*ListaAttributi*]
< **`values`** (*ListaDiValori*) | *SelectSQL* >

L'esito delle operazioni di inserimento è condizionato al rispetto dei vincoli di integrità. Le operazioni di inserimento che porterebbero la base di dati in uno stato che viola un vincolo non vengono portate a termine, e non hanno nessun effetto sulla base di dati.

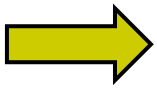
.

Manipolazione dei Dati: inserimento

INSERT INTO Persone (nome, reddito, età)
VALUES ("Ugo",20000000,35)

Inserire in Persone il record (Ugo, 20.000.000, 35)

	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
▶		0	0

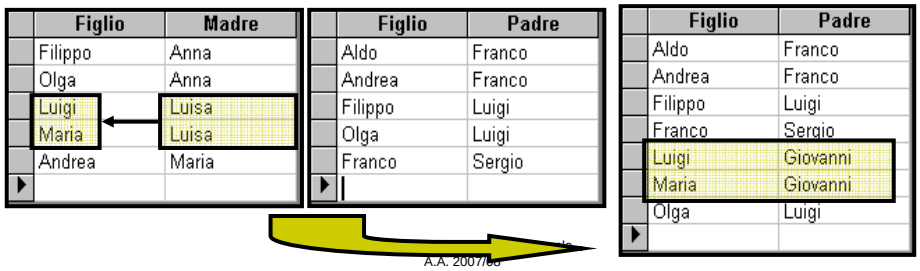


	Nome	Età	Reddito
	Aldo	25	15.000.000
	Andrea	27	21.000.000
	Anna	50	35.000.000
	Filippo	26	0
	Franco	60	20.000.000
	Luigi	50	40.000.000
	Luisa	75	87.000.000
	Maria	55	42.000.000
	Olga	30	25.000.000
	Sergio	85	35.000.000
	Ugo	35	20.000.000
▶		0	0

Esempio inserimento con sub-query

INSERT INTO paternità (padre,figlio)
SELECT 'Giovanni', figlio
FROM maternità
WHERE Madre='Luisa'

Inserire in Paternità I figli di Giovanni che sono i figli di Luisa (Aggiunge a Paternità i nomi dei figli di Giovanni e Luisa)



Cancellazione

`delete from` *NomeTabella* **`[ where`** *Condizione* **`]`**

se **`where`** non è specificato tutte le righe della tabella vengono eliminate. Se esiste un vincolo di integrità referenziale con politica di **`cascade …`** ATTENZIONE!!!!

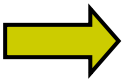
Siccome **`delete`** ha la stessa sintassi di **`select`**, è possibile utilizzare interrogazioni nidificate al suo interno.

Esempio di cancellazione

DELETE FROM Persone
WHERE Età = 50

Cancella da Persone tutte le persone di 50 anni

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Filippo	26	0
Franco	60	20.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0

Esempio di cancellazione

DELETE FROM Paternità
WHERE Padre="Giovanni"

Cancella da Paternità tutti i records dei figli di "Giovanni"

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

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

Modifica

update *NomeTabella*
set *Attributo* = < *Espressione* | *SelectSQL* | **null** | **default** >
{ , *Attributo* = < *Espressione* | *SelectSQL* | **null** | **default** > }
[**where** *Condizione*]

aggiorna uno o più attributi delle righe di *NomeTabella* che soddisfano l'eventuale *Condizione.* Se non c'è condizione la modifica avviene per tutte le righe.

Il nuovo valore può essere:

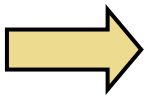
- il risultato di un'espressione valutata sugli attributi della tabella
- il risultato di una generica interrogazione SQL
- il valore nullo
- il valore di default

Esempio di Modifica

Incrementare del 10% il reddito di tutte le persone

UPDATE Persone
SET Reddito = Reddito *1.1

Nome	Età	Reddito
Aldo	25	15.000.000
Andrea	27	21.000.000
Anna	50	35.000.000
Filippo	26	0
Franco	60	20.000.000
Luigi	50	40.000.000
Luisa	75	87.000.000
Maria	55	42.000.000
Olga	30	25.000.000
Sergio	85	35.000.000
	0	0



Nome	Età	Reddito
Aldo	25	16.500.000
Andrea	27	23.100.000
Anna	50	38.500.000
Filippo	26	0
Franco	60	22.000.000
Luigi	50	44.000.000
Luisa	75	95.700.000
Maria	55	46.200.000
Olga	30	27.500.000
Sergio	85	38.500.000
	0	0