

Alias e variabili

L'uso degli alias consente di:

- compattare il codice
- fare riferimento a più esemplari della stessa tabella
- creare interrogazioni nidificate con riferimenti fra la query esterna e quella interna

Se una tabella compare una sola volta nell'interrogazione non c'è differenza fra variabile ed alias.

Se compare più volte si parla più propriamente di variabile.

Alias e variabili

Esempio

(impiegati il cui nome è anche il cognome di un altro impiegato del dipartimento Produzione)

```
select I1.Cognome, I1.Nome  
from Impiegato I1, Impiegato I2  
where I1.Nome = I2.Cognome  
and I1.Nome <> I2.Nome  
and I2.Dipart = 'Produzione'
```

Ordinamento

E' possibile ordinare le righe del risultato di una interrogazione attraverso la clausola **order by**, a chiusura di una interrogazione.

```
order by AttrdiOrdinamento [asc | desc]  
          { , AttrdiOrdinamento [asc | desc] }
```

asc (default) indica ordinamento ascendente,
desc discendente.

Il primo attributo ha priorità, a parità si ordina sulla base del secondo, ecc.

```
select *
```

```
from Persona
```

```
order by Cognome, Nome
```

Operatori aggregati

In algebra relazionale le espressioni vengono valutate sulle singole tuple in successione. Talvolta però possono essere necessarie informazioni derivabili dall'esame di tutte le tuple o di sottoinsiemi di tuple.

SQL prevede una serie di operatori aggregati

count, sum, max, min, avg

Operatori aggregati

count, sum, max, min, avg

Sintassi

count (< * | [**distinct**|**all**] *ListaAttributi* >)

<**sum**|**max**|**min**|**avg**> ([**distinct**|**all**] *AttrEspr*)

Determinare il numero degli impiegati che si chiamano Rossi

```
select count(*)  
from Impiegato  
where nome= 'Rossi'
```

Interrogazioni con raggruppamento

Gli operatori aggregati vengono applicati a **tutte** le righe che vengono prodotte come risultato dell'interrogazione. In questo caso, SQL non ammette che nella stessa target list compaiano funzioni aggregate ed espressioni a livello di riga, come il nome di un attributo. Non è infatti possibile associare un singolo risultato, calcolato su tutte le righe, a un attributo che, se ad es. è chiave, può anche assumere valori tutti diversi in ogni riga.

In alcuni casi è utile/necessario applicare l'operatore **separatamente ad uno o più sottoinsiemi** delle righe.

Interrogazioni con raggruppamento

SQL non ammette che nella stessa target list compaiano funzioni aggregate e nomi di attributi, poiché l'operatore agisce su tutte le righe della tabella

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

Di chi è il nome? Nome dovrebbe restituire tutti i nomi, max() un singolo valore.

La target list **deve essere omogenea**.

Interrogazioni con raggruppamento

In tale situazione l'interrogazione è corretta solo se la target list contiene solo operatori di aggregazione

Es.

```
select    min(eta), avg(reddito)
from      persone
where     reddito > 100
```

Restituisce due valori singoli, entrambi calcolati su tutte le tuple della tabella persone: l'età minima e il reddito medio di tutte le persone con reddito > 100.

Interrogazioni con raggruppamento

L'operatore **group by** *Attr {, Attr}*

specifica come suddividere le tabelle in sottoinsiemi omogenei **per valore** su cui valutare separatamente l'operatore aggregato.

```
select Dipart, sum(Stipendio)  
from Impiegato  
group by Dipart
```

(per ogni valore *x* assunto da **Dipart** calcola la somma dei valori di *Stipendio* di tutte le tuple con *Dipart=x*)

Interrogazioni con raggruppamento: **group by**

Se è presente un raggruppamento, nella target list possono comparire anche attributi, ma devono costituire un sottoinsieme di quelli usati nel **group by**

Es. (scorretto)

```
select Ufficio  
from Impiegato  
group by Dipart
```

Poiché deve venire prodotta una sola riga per ogni valore di Dipart, e, a parità di tale valore, possono aversi diversi valori di Ufficio, il risultato dell'interrogazione è indeterminato.

Interrogazioni con raggruppamento: **group by**

Es. (scorretto)

```
select Ufficio  
from Impiegato  
group by Dipart
```

Se per l'attributo che è nella target list senza comparire nella clausola **group by** esiste una *dipendenza funzionale* dagli attributi su cui si raggruppa (cioè se, a parità di valore di questi, il valore di tale attributo è uno solo) la query potrebbe fornire un risultato corretto, ma per semplicità SQL lo vieta comunque.

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

Raggruppamenti e target list

scorretta

```
select padre, avg(f.reddito), p.reddito  
from persone f join paternita on figlio = nome  
                join persone p on padre = p.nome  
group by padre
```

corretta

```
select padre, avg(f.reddito), p.reddito  
from persone f join paternita on figlio = nome  
                join persone p on padre = p.nome  
group by padre, p.reddito
```

Condizioni sui gruppi

Può essere anche necessario restringere la visualizzazione dei gruppi attraverso l'applicazione di condizioni logiche.

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**, che introduce una condizione valutata su ogni gruppo.

```
select Dipart, sum(Stipendio) as SommaStipendi  
from Impiegati  
group by Dipart  
having sum(Stipendio) > 100
```

Se non si specifica **group by** e si usa **having** da solo la condizione è applicata all'insieme di tutte le righe (come **where**). Se la condizione non è verificata, il risultato è vuoto.

Condizioni sui gruppi

- I padri i cui figli hanno un reddito medio maggiore di 25

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

Where o having ?

- I padri i cui figli sotto i 30 anni hanno un reddito medio maggiore di 25

```
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 sintassi 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*]