5 / 12 / 2024

Riepilogo / Situazioni in cui richiedo un commento

Commentate ogni:
~ Select interno.
~ Select che sarà combinato mediante un operatore insiemistico.
~ Tabella che state rinominando perché si trova in un self join.
~ Parametro di un select.
~ Vista.

NB Un select esterno non necessita di commento perché coincide con la consegna iniziale.

Riepilogo / Modalità 1

Questa modalità può essere usata soltanto se il select interno individua al massimo una riga; altrimenti, Sql darà un errore in esecuzione.

Quindi, per essere certi che l'interrogazione funzioni sempre (cioè, con qualsiasi input), dovremo progettare adeguatamente il select interno.

Riepilogo / Modalità 2

Consideriamo la query ATZE.35:

```
SELECT Nome
FROM IMPIEGATI
  INTERSECT
SELECT Cognome
FROM IMPIEGATI;
```

Possiamo sostituire l'`INTERSECT` con l'`IN`:

```
SELECT Nome
FROM IMPIEGATI
WHERE Nome IN ( /* Ogni cognome di impiegati. */
    SELECT Cognome
    FROM IMPIEGATI
);
```

Notate che otteniamo una soluzione equivalente sostituendo:
~ l'operatore `>= ALL` (dentro il select esterno)
con
~ l'operatore `MAX` (dentro il select interno).

SOLU.Aeroporti.P

Elencate le città italiane che hanno la massima "ricezione".

La **ricezione** di una città si ottiene sommando la capienza dei voli che arrivano in quella città.

L'elenco deve avere un'unica colonna.

Testate la soluzione usando questi dati:

TIPI_AEREI

TipoAereo	Capienza
A	100
B	70
C	70
D	200

VOLI

NomeCittaParte	NomeCittaArri	TipoAereo
Milano	Paris	D
Milano	Roma	B
Milano	Roma	C
Milano	Venezia	B
Milano	Venezia	B
Roma	Milano	A

I dati suddetti individuano le ricezioni seguenti:

```
Ricezione(Roma)        = 70 + 70   = 140
Ricezione(Milano)                  = 100
Ricezione(Paris)                   = 200
Ricezione(Venezia)     = 70 + 70   = 140
```

Quindi, l'output corrispondente è:

Roma
Venezia

Ricordo che è sintatticamente errato l'annidamento di due operatori di aggregazione.

a) Risolvete mediante operatori insiemistici.

   Suggerimento:

   1) Risolvete prima (erroneamente) annidando due operatori di aggregazione.
   2) Risolvete la consegna finale sostituendo adeguatamente uno dei due operatori che avete inserito nella vostra soluzione del passo 1).

Modalità 3

Questa modalità, a differenza delle due precedenti, non ha il compito di confrontare un valore del select esterno con i valori di quello interno.

Più semplicemente, deve soltanto valutare se il select interno individua una tabella vuota.

Sintassi:

```
/* Select esterno */
..
WHERE [NOT] EXISTS (
    SELECT *
    ..
);
```

Risultato:

~ In presenza di `NOT EXISTS`:

  La riga del select esterno compare nel risultato se non esiste nessuna riga nel select interno, cioè se questo individua un tabella vuota.

~ In presenza di `EXISTS`:

  La riga del select esterno compare nel risultato se il select interno individua almeno una riga.

NB Nella clausola `SELECT` del select interno ci conviene mettere sempre l'`*` (invece di un attributo o di un'espressione) perché ci interessa soltanto sapere se questo select individua almeno una riga; quindi, i valori specifici della riga non ci interessano.

Nota sull'utilità di EXISTS

Se il select interno è indipendente, l'`EXISTS` non ha nessuna utilità perché il suo esito è identico su tutte le righe del select esterno (cioè, è sempre vero oppure sempre falso) e quindi non seleziona le righe del select esterno.

Es: Il seguente select interno è indipendente (perché usa una tabella definita nel proprio `FROM`).

Provo a confrontarlo (impropriamente) con quello esterno mediante un `EXISTS`.

```
SELECT T1.K, T1.X
FROM T1
WHERE EXISTS (
    SELECT T2.*
    FROM T2
    WHERE T2.X = 3
);
```

L'interrogazione suddetta ha il seguente effetto.

Se il select interno individua una tabella vuota (cioè, se `T2.X` di ogni riga è sempre `!= 3`),

  Allora nessuna riga del select esterno compare nel risultato.

Se, invece, il select interno individua almeno una riga,

  Allora ogni riga del select esterno compare nel risultato.

Quindi, l'`EXISTS` in questo caso è inutile, perché il select interno (e indipendente) non visualizza soltanto alcune righe del select esterno, ma le visualizza tutte o nessuna.

L'`EXISTS` è utile soltanto nei select collegati, come vedremo tra poco.

Select collegati

L'effetto di un select collegato è il seguente:

Per ogni riga `r` del select esterno:

Il select interno individua l'insieme `I(r)` di righe del select interno. (Quindi, `I(r)` dipende dalla riga `r`).

Se il confronto (indicato nel `WHERE` o nell'`HAVING`) tra `r` (del select esterno) e `I(r)` è soddisfatto,

La riga `r` compare nel risultato.

Esempio

Traccio l'interrogazione:

```
SELECT *
FROM T1
WHERE T1.A = (
    SELECT COUNT(*)
    FROM T2
    WHERE T1.B = T2.B
);
```

sui seguenti dati:

T1

A	B
3	12
2	14

T2

B	C
12	8
14	7
13	5
14	6

Esecuzione della traccia:

Select esterno			Select interno		
T1.A	T1.B		WHERE T1.B ..	T2.COUNT(*)	WHERE T1.A = (SELECT COUNT(*) ..
3	12		12	1	3 = 1 ? F
2	14		14	2	2 = 2 ? T

Risultato:

A	B
2	14

Useremo un select annidato collegato quando l'interrogazione deve visualizzare le righe che soddisfano un confronto tra:

~ un valore del select esterno, e
~ un insieme di righe che varia al variare della riga del select esterno.

Esercizio risolto / 1

Considerate il db di riferimento.

Elencate ogni matricola che ha lo stesso cognome di un'altra matricola.

Usate un select annidato e l'operatore `EXISTS`.

Soluzione

Abbiamo già risolto un problema simile mediante un self join.

Adesso lo risolviamo mediante un algoritmo del tipo:

Per ogni impiegato `i` del select esterno

Il select interno individua l'insieme `I_O(i)` composto da quegli impiegati omonimi (nel cognome) di `i` (del select esterno) e che hanno inoltre una matricola differente.

Se l'insieme `I_O(i)` contiene almeno un impiegato,

La matricola di `i` compare nel risultato.

Possiamo notare che l'insieme `I_O(i)` dipende dall'impiegato `i`. Ciò implica che:

~ Il select interno deve essere collegato.

~ Il select interno confronta il cognome della sua tabella `IMPIEGATI` con il cognome di `IMPIEGATI` del select esterno. Quindi, usa la tabella `IMPIEGATI` con due finalità differenti; per questo, dobbiamo rinominare `IMPIEGATI` nel select interno.

La query in Sql è:

```
SELECT I.MatriImpie
FROM IMPIEGATI AS I
WHERE EXISTS ( /* Ogni impiegato omonimo della matricola I.MatriImpie */
    SELECT *
    FROM IMPIEGATI AS I_O
    WHERE I.Cognome = I_O.Cognome
      AND I.MatriImpie <> I_O.MatriImpie
);
```

Esecuzione della traccia:

Uso questi valori:

MatriImpie	Cognome
1	A
2	A
3	B
4	A

Select esterno		Select interno	
I.MatriImpie	I.Cognome	WHERE I.Cognome = I_O.Cognome AND I.MatriImpie <> I_O.MatriImpie	WHERE EXISTS
1	A	'A' = 'A' AND 1 <> 1 ? F	EXISTS({2, 4}) ? T
		'A' = 'A' AND 1 <> 2 ? T	
		'A' = 'B' AND 1 <> 3 ? F	
		'A' = 'A' AND 1 <> 4 ? T	
2	A	È analogo a I.MatriImpie = 1	EXISTS({1, 4}) ? T
3	B	'B' = 'A' AND 3 <> 1 ? F	EXISTS({ }) ? F
		'B' = 'A' AND 3 <> 2 ? F	
		'B' = 'B' AND 31 <> 3 ? F	
		'B' = 'A' AND 4 <> 4 ? F	
4	A	È analogo a I.MatriImpie = 1	EXISTS({1, 2}) ? T

L'interrogazione visualizzerà questa tabella:

I.MatriImpie
1
2
4

Esercizio risolto / 2

Considerate il db di riferimento.

Elencate, per ogni dipartimento:

~ la matricola con lo stipendio massimo nel dipartimento;

~ il valore di questo stipendio.

Ordinate l'elenco rispetto ai dipartimenti.

Soluzione

Per ogni impiegato (`m, d, s`) del select esterno

/* `m, d, s` sono rispettivamente la matricola, il dipartimento e lo stipendio dell'impiegato. */

Il select interno individua l'insieme `I_DIPA(d)` composto dagli impiegati che lavorano nel dipartimento `d`.

Se lo stipendio `s` (del select esterno) è uguale al massimo degli stipendi di `I_DIPA(d)`,

La matricola `m` (del select esterno) compare nel risultato.