12 / 12 / 2024

Esercizio risolto / 2 / Conclusione

Considerate il db di riferimento.

Elencate, per ogni dipartimento:

~ le matricole che hanno 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.

Possiamo notare che l'insieme `I_DIPA(d)` dipende dal dipartimento `d`. Ciò implica che:

~ Il select interno deve essere collegato.

~ Il select interno usa la tabella `IMPIEGATI` con due finalità differenti.
Quindi, dobbiamo rinominare `IMPIEGATI` nel select interno.

La query in Sql è:

```
SELECT I.NomeDipa, I.MatriImpie, I.Stipendio
FROM IMPIEGATI AS I
WHERE I.Stipendio = ( /* Stipendio massimo tra quelli degli impiegati
    che lavorano nel dipartimento I.NomeDipa */
    SELECT MAX(I_DIPA.Stipendio)
    FROM IMPIEGATI AS I_DIPA
    WHERE I.NomeDipa = I_DIPA.NomeDipa
)
ORDER BY I.NomeDipa;
```

Questo esercizio è un esempio molto importante di esercizi aventi la tipologia seguente:

Per ogni sottoinsieme individuato al variare di un attributo `A`,

Trovate le righe che hanno il valore massimo nell'attributo `B`.

In questo caso, `A` è il dipartimento, le righe cercate sono gli impiegati, e `B` è lo stipendio.

Esecuzione della traccia:

Uso questi valori:

MatriImpie	Stipendio	NomeDipa
1	50	SW
3	15	HW
5	15	SW
6	90	SW
7	90	SW

Select esterno | Select interno

I.MatriImpie, I.Stipendio, I.NomeDipa	ON I.NomeDipa = I_DIPA.NomeDipa	I_DIPA.Stipendio	WHERE I.Stipendio = MAX(I_DIPA.Stipendio)
1, 50, SW	SW	50	50 = 90 ? F
		15	
		90	
		90	
3, 15, HW	HW	15	15 = 15 ? T
5, 15, SW	SW	50	15 = 90 ? F
		15	
		90	
		90	
6, 90, SW	SW	50	90 = 90 ? T
		15	
		90	
		90	
7, 90, SW	SW	50	90 = 90 ? T
		15	
		90	
		90	

L'interrogazione visualizzerà questa tabella:

I.NomeDipa	I.MatriImpie	I.Stipendio
HW	2	15
SW	6	90
SW	7	90

SOLU.ESE.3

3. È dato il seguente db:

CITTA(NomeCitta, NomeStato, NAbita)

Elencate, per ogni stato, la città più abitata e la sua popolazione.

Ordinate l'elenco rispetto allo stato.

Testate la soluzione usando questi dati:

NomeCitta	NomeStato	NAbita
A	X	300
B	X	200
C	Y	200
D	Y	100
E	Y	200

L'output corrispondente è:

NomeStato	NomeCitta	NAbita
X	A	300
Y	C	200
Y	E	200

```
/* Risolto da: Pietro Trabuio */
SELECT C.NomeStato, C.NomeCitta, C.NAbita
FROM CITTA AS C
WHERE C.NAbita = ( /* N. di abitanti della città più popolosa tra quelle che sono
    nello stesso stato in cui si trova la città C */
    SELECT MAX(CI_STA.NAbita)
    FROM CITTA AS CI_STA
    WHERE C.NomeStato = CI_STA.NomeStato
)
ORDER BY C.NomeStato;
```

VISTE

Descrizione

Le tabelle che abbiamo visto finora sono soltanto un caso particolare (e il più importante) tra le varie categorie di "tabelle nel senso più ampio del termine" che possiamo trovare in un DB.

Possiamo classificare le "tabelle nel senso più ampio del termine" mediante questo albero:

~ Tabelle "normali" (o tabelle)
Sono le tabelle che abbiamo visto finora.

~ Viste
Le viste sono tabelle che contengono soltanto dati ridondanti, che noi inseriamo mediante istruzioni opportune.
Più precisamente, le viste contengono soltanto

  ~ alcuni dei dati contenuti nelle tabelle "normali",
  oppure

  ~ dati ricavabili da altri dati contenuti nelle tabelle.

Le viste si distinguono nelle seguenti categorie, a seconda della modalità con cui il DBMS le gestisce:

~ Viste “normali” (o viste)
Sono le viste di default.
Rimangono nel db finché non le elimineremo esplicitamente mediante il comando
`DROP NomeVista`.

~ Viste temporanee
Saranno eliminate quando chiuderemo la sessione di accesso al db.

Sintassi / ATZE.5.1.3

Considerare le definizioni delle viste contenute nell’esempio suddetto di Atze.

Dopo averle definite, potrò usarle per definire query come se le viste fossero tabelle.

Ad es. posso definire le query seguenti:

```
SELECT *
FROM IMPIEGATIAMMIN;
```

e

```
SELECT *
FROM IMPIEGATIAMMINPOVER;
```

Esempi

Osservate i file `Prove4_...sql` e leggete i commenti.
Troverete esempi di definizioni di viste sia “normali”, sia temporanee.

Vantaggi delle viste

Ci conviene usare una vista in questi casi:

1 Vogliamo definire soltanto una volta un'espressione che dovremo riusare in più interrogazioni.

In questo caso, definiremo una vista che calcola questa espressione; poi la riuseremo nelle varie interrogazioni.

2a Vogliamo scomporre un’interrogazione complicata.

2b Vogliamo scomporre “un’interrogazione non risolubile direttamente” in due o più sottointerrogazioni più semplici.

In questi due sottocasi (2a e 2b), definiremo una vista che risolve una delle sottointerrogazioni più semplici; poi la useremo, insieme ad altre interrogazioni, per risolvere il problema iniziale.

3 Vogliamo nascondere a un utente alcuni dati che non gli sono necessari.

In questo caso, definiremo una vista che contiene soltanto i dati che vogliamo rendere visibili.
Poi, consentiremo all’utente di accedere a questa vista (e non alla tabella originale).

4 Abbiamo ristrutturato il DB (e quindi stiamo usando tabelle che sono differenti da quelle del DB precedente). Inoltre vogliamo continuare a usare le interrogazioni che esistevano prima della ristrutturazione.
Per rendere compatibili le interrogazioni alla versione originaria, dovremmo disporre di ogni tabella presente nella versione originaria; ma alcune di queste tabelle non sono più disponibili a causa della ri-

strutturazione.

In questo caso, per ogni tabella non più disponibile, definiremo una vista “corrispondente” a questa tabella. Così, le interrogazioni precedentemente definite potranno funzionare accedendo a queste viste, anziché alle tabelle che esistevano prima della ristrutturazione (ma che adesso non esistono più).

Svantaggi delle viste

Il DBMS può gestire le viste in due modi differenti:

a) Può memorizzare i dati contenuti nella vista.

oppure

b) Invece di memorizzare i dati, può inserirli nella vista immediatamente prima di ogni volta che un’interrogazione deve usare questa vista.

In entrambi i casi c'è uno svantaggio.

a) Se il DBMS memorizza la vista,
Il DBMS deve aggiornare la vista ogni volta che l'utente modifica qualche dato da cui la vista dipende.

Quindi, questa situazione richiede un’elaborazione aggiuntiva ogni volta che modifichiamo qualche tabella dalla quale la vista dipende.

b) Se il DBMS inserisce i dati della vista quando deve usarla,
Il DBMS elaborare la vista (e anche che per elaborare l'interrogazione).

Quindi, questa situazione richiede un’elaborazione aggiuntiva ogni volta che usiamo la vista.

Esempi di applicazioni delle viste

Definizione di un'espressione che dovremo usare in più interrogazioni

Nel problema degli aeroporti, consideriamo le consegne E, F:

E: “Per il volo con identificativo 3, elencate il giorno della settimana, la nazione di partenza e quella di arrivo.”

F: “Elencate le coppie di città collegate da voli internazionali.”

Notate che entrambe le consegne richiedono di conoscere, per ogni volo, la nazione di partenza e quella di arrivo; quindi, le consegne suddette usano il medesimo FROM.

Allora. ci conviene definire una vista contenente ogni volo (con ogni suo attributo), al quale aggiungiamo i suoi stati di partenza e di arrivo.

Dopo, risolveremo le consegne E e F mediante la vista suddetta:

Consegna E

Abbiamo visto precedentemente questa soluzione:

```
SELECT V.GiornoSetti, C_P.Stato, C_A.Stato
FROM (VOLI AS V JOIN CITTA AS C_P ON V.NomeCittaParte = C_P.NomeCitta)
  JOIN CITTA AS C_A ON V.NomeCittaArri = C_A.NomeCitta
WHERE V.IdeVolo = 3;
```

Possiamo definire una seconda soluzione, che scomponiamo in due passi:

1 Definiamo una vista contenente ogni volo, al quale aggiungiamo la nazione di partenza e quella di arrivo.

```
CREATE VIEW VOLI_STATI (IdeVolo, GiornoSetti, NomeCittaParte, NomeCittaArri,
  TipoAereo, StatoParte, StatoArri) AS (
    SELECT V.*, C_P.Stato, C_A.Stato
    FROM (VOLI AS V JOIN CITTA AS C_P ON V.NomeCittaParte = C_P.NomeCitta)
      JOIN CITTA AS C_A ON V.NomeCittaArri = C_A.NomeCitta
);
```

2 Adesso, risolviamo la consegna mediante la vista suddetta:

```
SELECT GiornoSetti, NazioneParte, NazioneArri
FROM VOLI_STATI
WHERE IdeVolo = 3;
```

Consegna F

Abbiamo visto precedentemente questa soluzione:

```
SELECT DISTINCT V.NomeCittaParte, V.NomeCittaArri
FROM (VOLI AS V JOIN CITTA AS C_P ON V.NomeCittaParte = C_P.NomeCitta)
  JOIN CITTA AS C_A ON V.NomeCittaArri = C_A.NomeCitta
  AND C_P.Stato <> C_A.Stato;
```

Possiamo definire una seconda soluzione, che scomponiamo in due passi.

1 Questo passo è identico al passo 1 precedente.

2 Adesso, risolviamo la consegna mediante la vista suddetta:

```
SELECT DISTINCT NomeCittaParte, NomeCittaArri
FROM VOLI_STATI
WHERE StatoParte <> StatoArri;
```

Definizione di un'espressione che dovremo usare due volte nella stessa interrogazione

Nel problema degli aeroporti, la consegna P richiede: “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.”

Abbiamo precedentemente visto questa soluzione:

```
SELECT C_A.NomeCitta AS Nome_Citta_Con_Ricezione_Max
FROM (VOLI AS V JOIN CITTA AS C_A ON V.NomeCittaArri = C_A.NomeCitta)
  JOIN TIPI_AEREI AS TA ON V.TipoAereo = TA.TipoAereo
WHERE C_A.Stato = 'ITA'
GROUP BY C_A.NomeCitta
HAVING SUM(TA.Capienza) >= ALL ( /* Ricezioni di ogni citta` italiana */
    SELECT SUM(TA.Capienza)
    FROM (VOLI AS V JOIN CITTA AS C_A ON V.NomeCittaArri = C_A.NomeCitta)
      JOIN TIPI_AEREI AS TA ON V.TipoAereo = TA.TipoAereo
    WHERE C_A.Stato = 'ITA'
    GROUP BY C_A.NomeCitta
);
```

Notate che la soluzione suddetta usa, nel select interno, istruzioni che sono molto simili a quelle che usa nel select esterno.

Possiamo definire una seconda soluzione, che scomponiamo in due passi:

1 Definiamo una vista contenente ogni città di arrivo e la sua ricezione.

```
CREATE VIEW CI_ITA_CAPIE(NomeCittaArriIta, Ricezione) AS (
    /* Per ogni citta` italiana di arrivo, la sua ricezione. */
    SELECT C_A.NomeCitta, SUM(TA.Capienza)
    FROM (VOLI AS V JOIN CITTA AS C_A ON V.NomeCittaArri = C_A.NomeCitta)
      JOIN TIPI_AEREI AS TA ON V.TipoAereo = TA.TipoAereo
    WHERE C_A.Stato = 'ITA'
    GROUP BY C_A.NomeCitta
);
```

2 Adesso, risolviamo la consegna mediante la vista suddetta:

```
SELECT NomeCittaArriIta AS Nome_Citta_Con_Ricezione_Max
FROM CI_ITA_CAPIE
WHERE Ricezione = ( /* Ricezione massima tra quelle delle citta` italiane */
    SELECT MAX(Ricezione)
    FROM CI_ITA_CAPIE
);
```

Esercizi

2. È dato il DB che gestisce la partecipazione delle squadre ai campionati della massima serie di un dato sport. Il DB è composto da:

SQUADRE(NomeSquadra, AnnoFonda)

PARTECIPAZIONI(NomeSquadra, AnnoCampio, NPunti)

Nessun campionato ha avuto due squadre arrivate prime con lo stesso numero di punti. (Cioè, nessun campionato si è concluso con uno spareggio.)

Trovate la/e squadra/e che ha vinto più campionati, mediante una vista (che dovete definire opportunamente).

Es: Con i seguenti valori, dovete individuare la squadra A, perché ha vinto il campionato due volte (nel 2000 e nel 2010); invece, la squadra B lo ha vinto soltanto una volta (nel 2020).

PARTECIPAZIONI

NomeSquadra	AnnoCampio	NPunti
A	2000	100
B	2000	80
A	2010	200
B	2010	100
A	2020	60
B	2020	80

Prima, definisco una vista che, per ogni anno, determina la squadra vincitrice.

```
/* Risolto da: Nicolò Fioranzato */
CREATE VIEW VINCITRICI(AnnoCampio, NomeSquadra) AS (
    SELECT P1.AnnoCampio, P1.NomeSquadra
    FROM PARTECIPAZIONI AS P1
    WHERE P1.NPunti = (
        /* Punti della squadra vincitrice del campionato P1.AnnoCampio
        SELECT MAX(P2.NPunt)
        FROM PARTECIPAZIONI AS P2
        WHERE P1.AnnoCampio = P2.AnnoCampio
    )
);
```

Dopo, trovo la squadra che ha vinto più campionati.

```
/* Risolto da: Nicolò Fioranzato */
SELECT V.NomeSquadra, COUNT(*)
    FROM VINCITRICI AS V
    GROUP BY V.NomeSquadra
    HAVING COUNT(*) >= ALL (
        # Per ogni squadra, elenca il numero di campionati vinti.
        SELECT COUNT(*)
        FROM VINCITRICI AS V
        GROUP BY V.NomeSquadra
    )
);
```