Corso di Basi di dati

Secondo Compitino - Fila B - 19 maggio 2004

Soluzioni

1. $EMITTENTE(\underline{CodE}, NomeE, Tipo)$
   $SPOT(\underline{CodS}, NomeS, Prodotto, Durata)$
   $TRASMETTE(\underline{CodE^*, CodS^*, DataIn}, DataF, Freq)$

   (a) i. Determinare i prodotti di cui esistono almeno due spot.

   ```
   SELECT Prodotto FROM SPOT
   GROUP BY Prodotto
   HAVING COUNT(DISTINCT NomeS) > 1
   ```

   N.B.; poiché la formulazione é ambigua, va bene anche COUNT(*) o COUNT(DISTINCT CodS)

   ii. Determinare gli spot di durata superiore alla media degli spot dello stesso prodotto.

   ```
   SELECT CodS FROM SPOT S
   WHERE Durata > (SELECT AVG(Durata)
                   FROM SPOT
                   WHERE Prodotto = S.Prodotto)
   ```

   iii. Determinare tra gli spot della pasta Barilla che sono stati trasmessi almeno 1000 volte quello piú lungo.

   ```
   BM = SELECT CodS, Durata
        FROM TRASMETTE NATURAL JOIN SPOT
        WHERE Prodotto = 'pasta~Barilla'
        GROUP BY CodS
        HAVING SUM(Freq * (DataF-DataI)) > 1000
   ```

   (spot della pasta barilla trasmessi almeno 1000 volte e loro durata)

   ```
   SELECT CodS FROM BM
   WHERE Durata = (SELECT MAX(Durata) FROM BM)
   ```

   iv. Determinare i prodotti che sono stati reclamizzati almeno 10 volte al giorno su tutte le emittenti regionali.

   ```
   SELECT Prodotto
   FROM SPOT S
   WHERE NOT EXISTS (SELECT * FROM EMITTENTE
                     WHERE Tipo = 'regionale' AND
                           CodE NOT IN (SELECT CodE
                                        FROM TRASMETTE NATURAL JOIN SPOT
                                        WHERE Prodotto = S.Prodotto AND Freq > 10))
   ```

   (b) Vista che associa ad ogni emittente il suo tipo e il numero di messaggi pubblicitari trasmessi, la durata totale di tali messaggi, il numero di prodotti diversi reclamizzati nel 2003.

   ```
   CREATE VIEW TRASM2003 AS
   (SELECT *
    FROM TRASMETTE
    WHERE DataI >= DATE '01/01/2003' AND DataF <= DATE '31/12/2003')
   UNION
   (SELECT CodE, CodS, DATE '01/01/2003', DataF, Freq
    FROM TRASMETTE
    WHERE (DataI,DataF) OVERLAPS (DATE '01/01/2003', DATE '31/12/2003')
   ```

```
       AND DataI < DATE '01/01/2003' AND DataF <= DATE '31/12/2003')
UNION
(SELECT CodE, CodS, DataI, DATE '31/12/2003', Freq
 FROM TRASMETTE
 WHERE (DataI,DataF) OVERLAPS (DATE '01/01/2003', DATE '31/12/2003')
       AND DataI >= DATE '01/01/2003' AND DataF > DATE '31/05/2003')
UNION
(SELECT CodE, CodS, DATE '01/01/2003', DATE '31/12/2003', Freq
 FROM TRASMETTE
 WHERE DataI < DATE '01/01/2003' AND DataF > DATE '31/12/2003')

CREATE VIEW STAT_EMITT AS
SELECT CodE, Tipo, SUM(Freq * (DataF - DataI)) AS NumM,
       SUM(Freq * (DataF - DataI) * Durata) AS TotM,
       COUNT(DISTINCT Prodotti) AS NumP
FROM SPOT NATURAL JOIN TRASM2003 NATURAL JOIN EMITTENTE
GROUP BY CodE, Tipo
```

i. Sulla vista si possono effettuare solo interrogazioni ma nessun aggiornamento (contiene nella sua definizione JOIN, funzioni di gruppo, GROUP BY).

ii. A. basta che il creatore della vista STAT_EMITT esegua

```
CREATE VIEW NONUMPROD AS
SELECT CodE, Tipo, NumM, TotM
FROM STAT_EMITT;

GRANT SELECT ON NONUMPROD TO Bob;
```

B. basta che il creatore della vista STAT_EMITT esegua

```
CREATE VIEW SOLOREG AS
SELECT *
FROM STAT_EMITT
WHERE Tipo = 'regionale';

GRANT SELECT ON SOLOREG TO Bob;
```

C. non é possibile inserire informazioni nella vista (contiene nella sua definizione JOIN, funzioni di gruppo, GROUP BY).

2. $SPOT(\underline{CodS}, NomeS, Prodotto, Durata)$

   (a) $CodS \rightarrow NomeS\ Durata$
   $NomeS\ Durata \rightarrow CodS$
   $NomeS \rightarrow Prodotto$

   (b) Chiavi: $CodS$ e $NomeS\ Durata$

   (c) La relazione non é in 3NF né in BCNF: la dipendenza $NomeS \rightarrow Prodotto$ non ha a sinistra una chiave e l'attributo a destra non é primo.

3. $R(X, Y, Z, W, S, T)$ con le seguenti dipendenze funzionali
   $XYZ \rightarrow SW, XZW \rightarrow T, XW \rightarrow Y, Y \rightarrow Z, Z \rightarrow W$
   in forma minimale:
   $XY \rightarrow S, XZ \rightarrow T, XW \rightarrow Y, Y \rightarrow Z, Z \rightarrow W$

(a) che verifica:

$a_1\ b_1\ c_1\ d_1\ e_1\ f_1$
$a_2\ b_2\ c_2\ d_2\ e_2\ f_2$

che viola:

$a_1\ b_1\ c_1\ d_1\ e_1\ f_1$
$a_1\ b_1\ c_1\ d_2\ e_2\ f_2$

(b) Chiavi; XY, XW, XZ

(c) Lo schema é in 3NF ma non in BCNF (le ultime due dipendenze non hanno a sinistra una chiave ma hanno a destra un attributo primo.

(d) Scomposizione lossless join in BCNF: (YZ), (ZW), $(XYST)$ non preserva ad esempio la dipendenza $AC \rightarrow F$

(e) Scomposizione non lossless join: (XYZ) e (WST), perché gli schemi sono disgiunti.

(f) Scomposizione lossless join che preserva le dipendenze: quella che ottengo con l'algoritmo per 3NF, cioé (XYS), (XZT), (XYW), (YZ), (ZW)

4. (a)
- su A, indice denso non clusterizzato, per verificare l'unicitá della chiave e per eventuali join con relazioni in cui A sia chiave esterna
- su C, indice clusterizzato (se sparso, con un'entrata per blocco, 50 entrate – se denso, comunque 50 entrate) se si accede alla relazione attraverso il valore di C in selezioni o join
- su D, indice denso non clusterizzato, se utile nelle operazioni
- non é molto utile definire un indice su B, visto che assume due soli valori

(b)
i. dannoso di qualsiasi tipo (lo devo aggiornare)
ii. utile di qualsiasi tipo (mi aiuta a trovare direttamente le tuple da aggiornare)
iii. dannoso di qualsiasi tipo (lo devo aggiornare)
iv. utile di qualsiasi tipo
v. utile indice ad albero
vi. inutile di qualsiasi tipo, ma non dannoso
vii. inutile di qualsiasi tipo (devo comunque guardare tutte le tuple di R), ma non dannoso
viii. utile di qualsiasi tipo
ix. inutile, la relazione é giá clusterizzata (ordinata) su C *** controlla uso di indici clusterizzati per ordinamento
x. utile indice ad albero se denso si puó eseguire la proiezione accedendo solo all'indice e non al file dei dati