Corso di Basi di dati

Secondo Compitino - Fila A- 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 gli spot trasmessi su almeno due emittenti.

   ```
   SELECT CodS FROM TRASMETTE
   GROUP BY CodS
   HAVING COUNT(DISTINCT CodE) > 1
   ```

   ii. Determinare gli spot trasmessi su un'emittente con frequenza superiore alla media per quell'emittente.

   ```
   SELECT CodS FROM TRASMETTE T
   WHERE Freq > (SELECT AVG(Freq)
                 FROM TRASMETTE
                 WHERE CodE = T.CodE)
   ```

   iii. Determinare tra gli spot di durata superiore a 30 secondi che sono stati trasmessi almeno 1000 volte quello che é stato trasmesso su piú emittenti.

   ```
   NE = SELECT CodS, COUNT(DISTINCT CodE) AS NumE
        FROM TRASMETTE NATURAL JOIN SPOT
        WHERE Durata > 30
        GROUP BY CodS
        HAVING SUM(Freq * (DataF-DataI)) > 1000
   ```

   (per ogni spot di durata superiore a 30 secondi che é stato trasmesso almeno 1000 volte, su quante emittenti é stato trasmesso)

   ```
   SELECT CodS FROM NE
   WHERE NumE = (SELECT MAX(NumE) FROM NE)
   ```

   iv. Determinare le emittenti regionali che hanno trasmesso tutti gli spot di durata superiore a 30 secondi almeno 10 volte al giorno.

   ```
   SELECT CodE
   FROM EMITTENTE E
   WHERE Tipo = 'regionale' AND
         NOT EXISTS (SELECT * FROM SPOT
                     WHERE CodS NOT IN (SELECT CodS
                                        FROM TRASMETTE
                                        WHERE CodE = E.CodE 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 periodo gennaio-maggio 2004.

   ```
   CREATE VIEW TRASM_GEN_MAG_04 AS
   (SELECT *
    FROM TRASMETTE
    WHERE DataI >= DATE '01/01/2004' AND DataF <= DATE '31/05/2004')
   UNION
   (SELECT CodE, CodS, DATE '01/01/2004', DataF, Freq
    FROM TRASMETTE
   ```

```
 WHERE (DataI,DataF) OVERLAPS (DATE '01/01/2004', DATE '31/05/2004')
       AND DataI < DATE '01/01/2004' AND DataF <= DATE '31/05/2004')
UNION
(SELECT CodE, CodS, DataI, DATE '31/05/2004', Freq
 FROM TRASMETTE
 WHERE (DataI,DataF) OVERLAPS (DATE '01/01/2004', DATE '31/05/2004')
       AND DataI >= DATE '01/01/2004' AND DataF > DATE '31/05/2004')
UNION
(SELECT CodE, CodS, DATE '01/01/2004', DATE '31/05/2004', Freq
 FROM TRASMETTE
 WHERE DataI < DATE '01/01/2004' AND DataF > DATE '31/05/2004')
```

```
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 TRASM_GEN_MAG_04 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. $R(A, B, C, D, E, F)$ con le seguenti dipendenze funzionali

   $ABC \rightarrow ED, ACD \rightarrow F, AD \rightarrow B, B \rightarrow C, C \rightarrow D$

   in forma minimale:

   $AB \rightarrow E, AC \rightarrow F, AD \rightarrow B, B \rightarrow C, C \rightarrow D$

(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; AB, AD, AC

(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: (BC), (BD), $(ABEF)$ non preserva ad esempio la dipendenza $AC \rightarrow F$

(e) Scomposizione non lossless join: (ABC) e (DEF), perché gli schemi sono disgiunti.

(f) Scomposizione lossless join che preserva le dipendenze: quella che ottengo con l'algoritmo per 3NF, cioé (ABE), (ACF), (ABD), (BC), (CD)

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 B, indice sparso clusterizzato (un'entrata per blocco, 50 entrate) se si accede alla relazione attraverso il valore di B in selezioni o join
   - su C, indice denso non clusterizzato, se utile nelle operazioni
   - non é molto utile definire un indice su D, visto che assume due soli valori

   (b)
   i. utile indice ad albero perché essendo sicuramente denso (é non clusterizzato) si puó eseguire la proiezione accedendo solo all'indice e non al file dei dati
   ii. potrebbe essere utile indice ad albero per effettuare l'ordinamento (anche se non essendo clusterizzato é probabilmente piú conveniente la scansione sequenziale + external sorting)
   iii. utile di qualsiasi tipo
   iv. inutile di qualsiasi tipo, ma non dannoso
   v. utile indice ad albero
   vi. utile di qualsiasi tipo
   vii. inutile di qualsiasi tipo (devo comunque guardare tutte le tuple di R), ma non dannoso
   viii. dannoso di qualsiasi tipo (lo devo aggiornare)
   ix. dannoso di qualsiasi tipo (lo devo aggiornare)
   x. utile di qualsiasi tipo (mi aiuta a trovare direttamente le tuple da aggiornare)