Sistemi Informativi T
18 gennaio 2016
Risoluzione

Tempo a disposizione: 2:30 ore

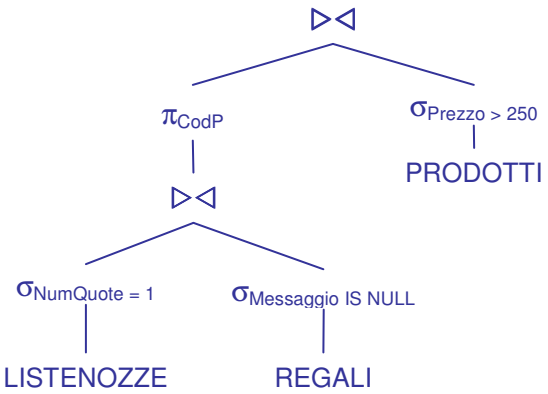
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

```
PRODOTTI(CodP,Descrizione,Prezzo,Categoria);
LISTENOZZE(LID,CodP,NumQuote,PrezzoQuota,QuoteRegalate);
      CodP REFERENCES PRODOTTI;
REGALI(LID,CodP,Invitato,NumQuote,Messaggio*),
      (LID,CodP) REFERENCES LISTENOZZE;
--
-- LISTENOZZE.NumQuote: indica in quante quote (una o più)
-- il prezzo è diviso (Prezzo = PrezzoQuota*NumQuote)
-- LISTENOZZE.QuoteRegalate (<= LISTENOZZE.NumQuote) è il numero di
-- quote sinora regalate
-- REGALI.NumQuote (<= LISTENOZZE.QuoteRegalate) è il numero di quote
-- di un dato prodotto che un invitato ha regalato
-- Tutti i prezzi sono in formato DEC(6,2)
```

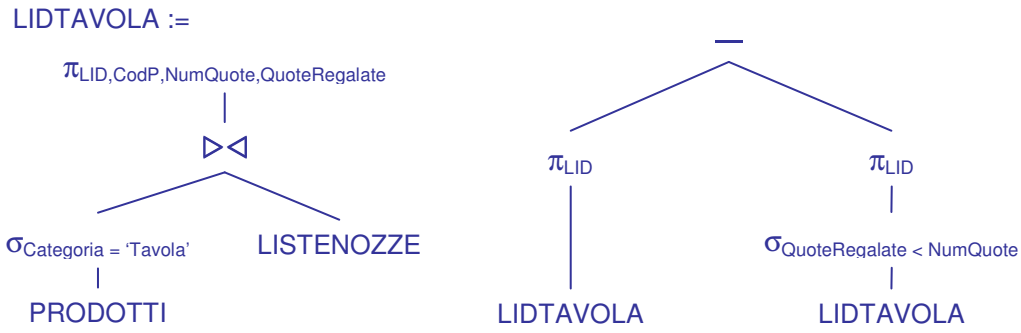
si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I dati completi dei prodotti con prezzo maggiore di 250€ per cui esiste almeno una lista di nozze in cui il prodotto ha una sola quota, già regalata e senza messaggio



Si noti che non è necessario specificare QuoteRegalate = 1 (se fosse 0 il join non troverebbe un match in REGALI), né è necessario evitare il join sull'attributo NumQuote (che deve per forza valere 1)

1.2) **[2 p.]** Le liste di nozze (LID) in cui ci sono uno o più prodotti della categoria 'Tavola' e sono stati tutti completamente regalati



Non è corretto risolvere l'esercizio usando l'operatore di divisione, in quanto il divisore (tutti i prodotti della categoria 'Tavola') non è una costante, ma varia da una lista di nozze all'altra

Sistemi Informativi T
18 gennaio 2016
Risoluzione

2) SQL (5 punti totali)

Con riferimento al DB dell'esercizio 1, si scrivano in SQL le seguenti interrogazioni:

2.1) [2 p.] Per ogni categoria, il numero di prodotti diversi presenti complessivamente nelle liste di nozze

```
SELECT   P.Categoria, COUNT(DISTINCT L.CodP) AS NUM_PRODOTTI
FROM     PRODOTTI P, LISTENOZZE L
WHERE    P.CodP = L.CodP
GROUP BY P.Categoria

-- Soluzione immediata usando COUNT(DISTINCT L.CodP)
```

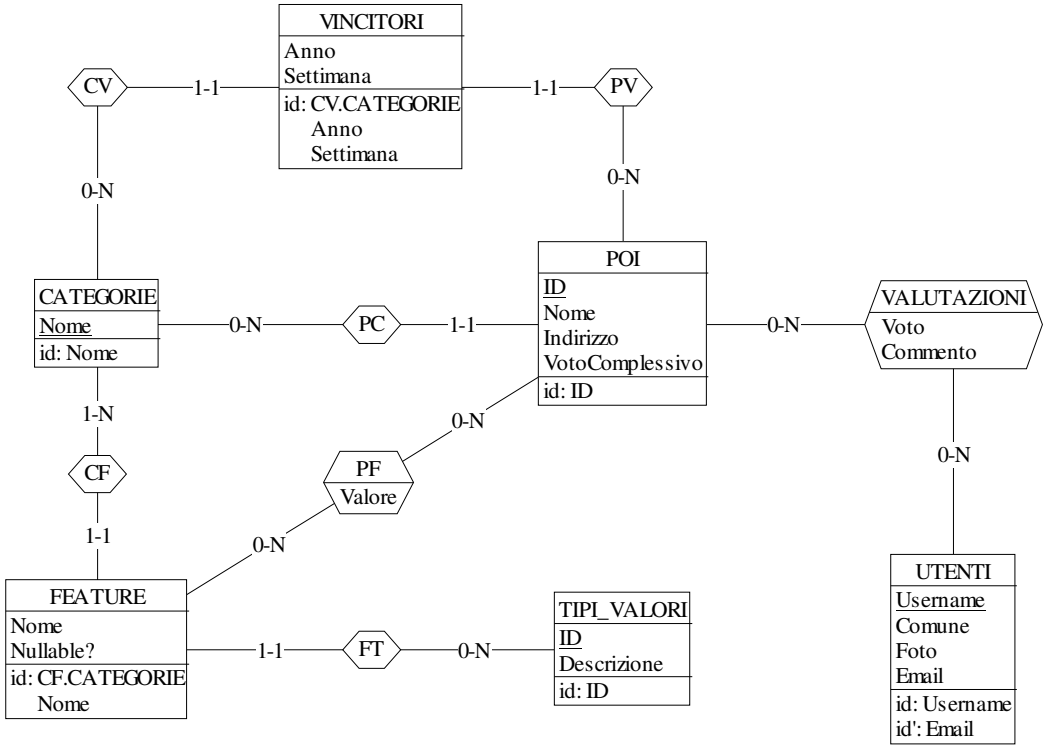
2.2) [3 p.] La lista di nozze (LID) e il relativo importo totale che gli sposi devono pagare per completare l'acquisto di tutti i prodotti previsti per la lista per cui tale importo è massimo

```
WITH DAPAGARE (LID, Totale) AS (
   SELECT   L.LID, SUM(L.PrezzoQuota*(L.NumQuote-L.QuoteRegalate))
   FROM     LISTENOZZE L
   GROUP BY L.LID          )

SELECT   D.*
FROM     DAPAGARE D
WHERE    D.Totale = ( SELECT MAX(D1.Totale)
                      FROM   DAPAGARE D1 )

-- La c.t.e. calcola, per ogni lista di nozze, l'importo totale che resta
-- da pagare
```

Sistemi Informativi T
18 gennaio 2016
Risoluzione

3) Progettazione concettuale (6 punti)

ThreeAngles (3A) è un social network che permette di condividere esperienze relative a diverse tipologie di "Point Of Interest" (POI), quali monumenti, negozi, ristoranti, ecc. Ogni POI fa parte di una categoria (le categorie sono prestabilite dal sistema) ed è caratterizzato da un nome e un indirizzo. Per ogni categoria sono poi definite delle caratteristiche (*feature*) rilevanti (ad es. per i monumenti i giorni e gli orari di apertura, ecc.), in modo che per i POI di quella categoria sia possibile specificare opportuni valori. Il tipo di valori ammessi è contenuto nella definizione della feature e deve essere uno di quelli previsti da 3A. Per ogni feature è inoltre specificato se è possibile o meno per un POI non specificare alcun valore,
Gli utenti di 3A hanno uno username (univoco), un comune di residenza, una foto e una email (non è possibile creare due o più account con la stessa email). Per ogni POI un utente può esprimere una sola valutazione (voto da 1 a 10) accompagnata da un commento (un'eventuale nuova valutazione e/o commento per lo stesso POI rimpiazza quanto già esistente). Sulla base delle valutazioni ricevute, 3A assegna a ogni POI un voto complessivo. Ogni settimana, e per ogni categoria, viene stabilito il POI vincitore, che resta perennemente visibile sul sito di 3A.

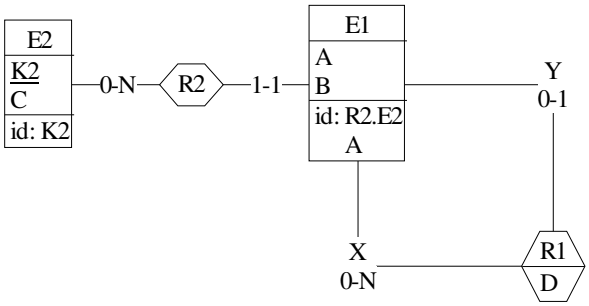


Commenti:

- La particolarità dell'esercizio sta nella necessità di gestire proprietà (*feature*) e valori per le stesse. Il tutto si ottiene mediante l'entità FEATURE, che introduce le caratteristiche di ogni categoria, e l'associazione PF, che permette di esprimere i valori specifici dei POI per tali feature.
- Il vincolo che un POI può avere definiti dei valori per le sole feature della propria categoria non è tuttavia esprimibile, così come non sono esprimibili né il fatto che siano permessi valori nulli per le sole feature in cui ciò è previsto (attributo Nullable? dell'entità FEATURE), né che i valori di un POI siano del tipo definito per la rispettiva feature.
- Un ulteriore vincolo non rappresentabile riguarda i VINCITORI, e in particolare che un POI può risultare vincitore solo per la categoria cui appartiene.

Sistemi Informativi T
18 gennaio 2016
Risoluzione

4) Progettazione logica (6 punti totali)



Dato lo schema concettuale in figura e considerando che:

a) tutti gli attributi sono di tipo INT;
b) l'associazione R1 non viene tradotta separatamente;
c) un'istanza di E1 che partecipa a R1 con il ruolo X non è mai associata a un'istanza di E1 identificata esternamente da un'istanza di E2 con valore C = 10;

4.1) [3 p.] Si progettino gli opportuni schemi relazionali e si definiscano tali schemi in DB2 (sul database SIT_STUD) mediante un file di script denominato SCHEMI.txt

```
CREATE TABLE E2 (
K2 INT NOT NULL PRIMARY KEY,
C INT NOT NULL                          );

CREATE TABLE E1 (
K2 INT NOT NULL REFERENCES E2,
A INT NOT NULL,
B INT NOT NULL,
Y_DEFINED SMALLINT NOT NULL CHECK (Y_DEFINED IN (0,1)), -- 1 se partecipa a R1 con ruolo Y
K2X INT,
AX   INT,
D INT,
CONSTRAINT R1 CHECK (
     (Y_DEFINED = 0 AND D IS NULL AND K2X IS NULL AND AX IS NULL) OR
     (Y_DEFINED = 1 AND D IS NOT NULL AND K2X IS NOT NULL AND AX IS NOT NULL) ),
PRIMARY KEY (K2,A),
CONSTRAINT FK_X FOREIGN KEY (K2X,AX) REFERENCES E1                  );
```

4.2) [3 p.] Per i vincoli non esprimibili a livello di schema si predispongano opportuni **trigger che evitino inserimenti di singole tuple non corrette**, definiti in un file TRIGGER.txt e usando se necessario il simbolo '@' per terminare gli statement SQL (altrimenti ';')

```
-- Trigger che garantisce il rispetto del vincolo di cui al punto c). Il vincolo può essere violato solo inserendo una
-- tupla in E1 che partecipa a R1 con ruolo Y.
-- Si sfrutta il valore del selettore Y_DEFINED, altrimenti un attributo che è non nullo quando la tupla inserita
-- partecipa a R1 con ruolo Y

CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (Y_DEFINED = 1 AND EXISTS (        SELECT * FROM E2
                                        WHERE N.K2 = E2.K2
                                        AND E2.C = 10              ) )
SIGNAL SQLSTATE '70001' ('La tupla inserita e'' identificata da una tupla di E2 con C=10!');
```