Sistemi Informativi T
3 febbraio 2016
Risoluzione

Tempo a disposizione: 2:30 ore

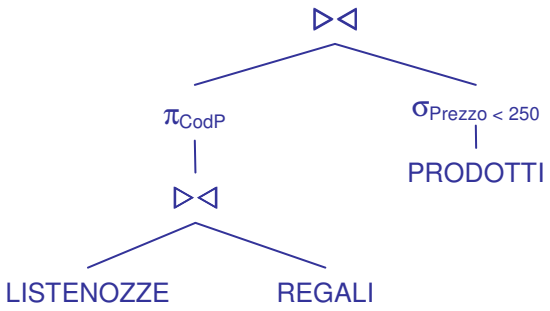
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

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

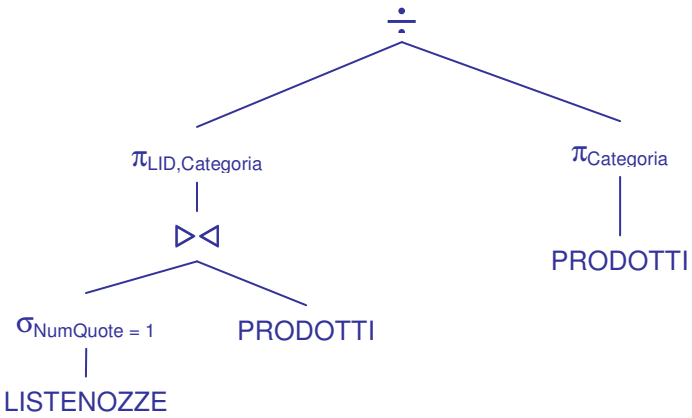
si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I dati completi dei prodotti con prezzo minore di 250€ per cui esiste almeno una lista di nozze in cui il prodotto è stato completamente regalato da un singolo invitato



Si noti che la query di fatto richiede che sia LISTENOZZE.NumQuote = REGALI.NumQuote, da cui la semplice soluzione

1.2) **[2 p.]** Le liste di nozze (LID) in cui, per ogni categoria, compare almeno un prodotto con una sola quota



Sistemi Informativi T
3 febbraio 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 lista di nozze e per ogni prodotto relativo con almeno due quote e regalato da almeno un invitato, il rapporto (con due cifre decimali) tra le quote sinora pagate e le quote totali (es. prodotto presente con 3 quote di cui 2 già regalate: 2/3 = 0,66)

```
SELECT   L.LID, L.CodP,
         CAST(CAST(SUM(R.NumQuote) AS DEC(4,2))/L.NumQuote AS DEC(4,2)) AS
           RAPPORTO_PAGATE
FROM     LISTENOZZE L, REGALI R
WHERE    L.LID = R.LID
AND      L.CodP = R.CodP
AND      L.NumQuote > 1
GROUP BY L.LID, L.CodP, L.NumQuote

-- Si noti che bisogna raggruppare anche su L.NumQuote, in quanto a tale
-- attributo non si applica nessuna funzione aggregata
```

2.2) **[3 p.]** Per ogni fascia di prezzo (0-499 €, 500-999 €, ecc.), il numero di liste di nozze in cui la somma sinora pagata dagli invitati per i regali cade in quella fascia

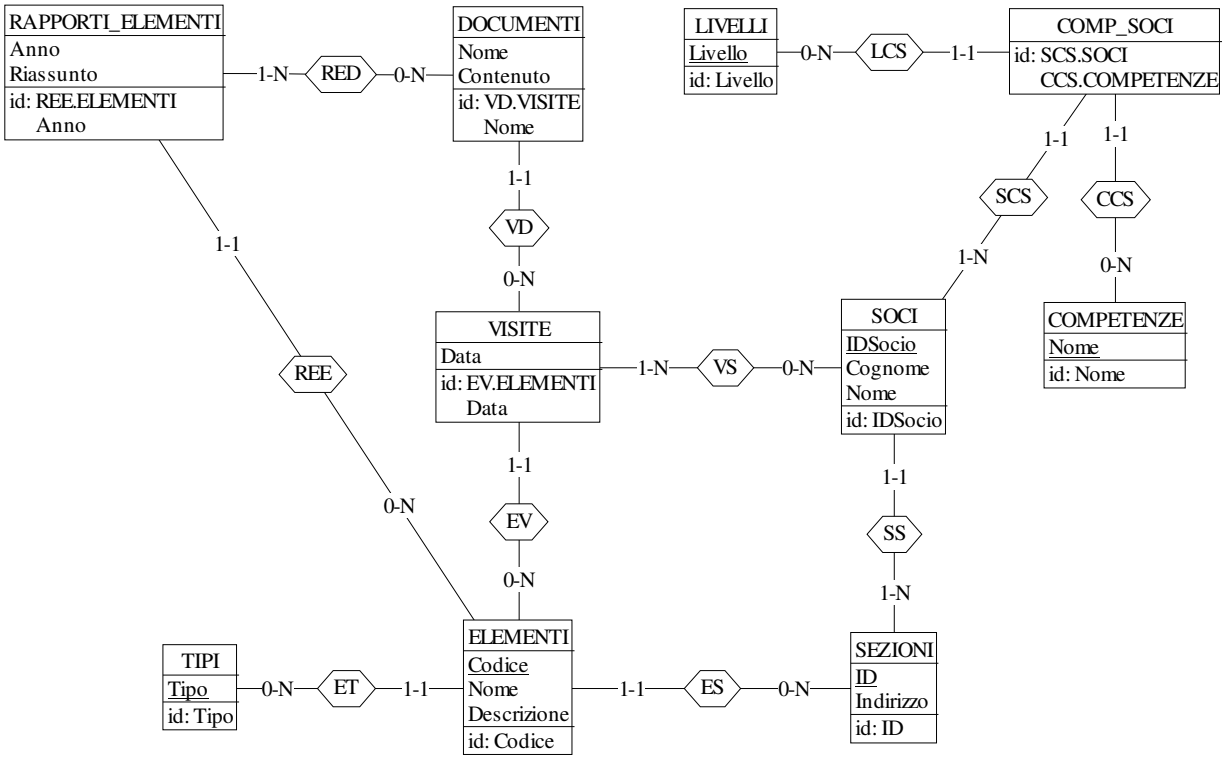
```
WITH LID_PAGATO (LID,Totale) AS (
   SELECT   L.LID, CAST(SUM(L.PrezzoQuota*R.NumQuote) AS INT)
   FROM     LISTENOZZE L, REGALI R
   WHERE    L.CodP = R.CodP
   AND      L.LID  = R.LID
   GROUP BY L.LID          )
SELECT (Totale/500)*500 CONCAT '-' CONCAT ((Totale/500 + 1)*500 - 1)
         AS FASCIA, COUNT(*) AS NUM_LISTE
FROM     LID_PAGATO
GROUP BY Totale/500

-- La c.t.e. calcola, per ogni lista di nozze, l'importo totale pagato.
-- Fondamentale eseguire il CAST della somma totale pagata
```

Sistemi Informativi T
3 febbraio 2016
Risoluzione

3) Progettazione concettuale (6 punti)

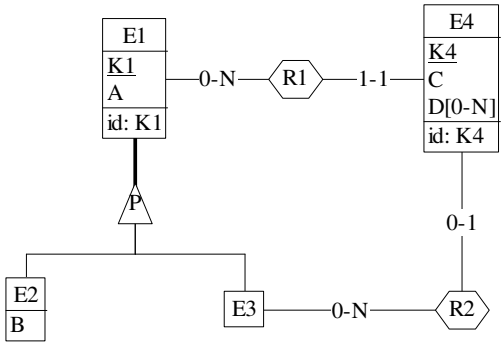
"1000 Patrimoni da Salvare" (1kPdS) è un'associazione che si occupa della tutela e salvaguardia del patrimonio artistico e naturalistico nazionale. 1kPdS è organizzata in sezioni, ognuna con uno o più soci. Ogni socio mette a disposizione le proprie competenze, descritte secondo il sistema di classificazione della 1kPdS, che prevede sia una tipologia di classificazione (ad es. fotografia) che un livello (ad es. esperto).
Periodicamente ogni sezione organizza delle visite di ispezione presso un elemento tutelato da quella sezione (ogni elemento è a carico di una sola sezione). I vari elementi hanno una tipologia (monumento, sito naturalistico, ecc.), un nome, un codice univoco e una descrizione. Ogni visita viene effettuata da uno o più soci della sezione e produce della documentazione specifica, che viene archiviata specificandone nome (univoco per quella visita) e contenuto.
Annualmente la 1kPdS rende disponibile un "rapporto di salute" di tutti gli elementi tutelati, in cui per ogni elemento viene fornito un quadro riassuntivo della situazione, corredato da alcuni elementi della documentazione prodotta durante le relative visite..



Commenti:

- L'entità COMP_SOCI è ottenuta per reificazione di un'associazione ternaria tra SOCI, COMPETENZE e LIVELLI, al fine di modellare il fatto che, per una data competenza, un socio ha un unico livello.
- Non viene introdotta nello schema un'entità RAPPORTI, che, date le specifiche, avrebbe come unico attributo significativo l'anno.
- Non c'è bisogno di introdurre un'associazione tra VISITE e SEZIONI, perché la sezione è univocamente determinata dalle associazioni EV ed ES.
- Il vincolo che i soci coinvolti in una visita facciano parte della sezione che ha in carico l'elemento visitato non è esprimibile. Non è nemmeno esprimibile il vincolo che i documenti relativi a un'istanza di RAPPORTI_ELEMENTI facciano (tutti) riferimento all'elemento che identifica l'istanza stessa.

Sistemi Informativi T
3 febbraio 2016
Risoluzione



- **Progettazione logica (6 punti totali)**

Dato lo schema concettuale in figura e considerando che:

a) tutti gli attributi sono di tipo INT;
b) le entità E1, E2 ed E3 vengono tradotte assieme;
c) le associazioni R1 e R2 non vengono tradotte separatamente;
d) un'istanza di di E4 che partecipa a R2 non può aver nessun valore di D maggiore di 20;

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 E1 (
K1 INT NOT NULL PRIMARY KEY,
A INT NOT NULL,
TIPO23 SMALLINT NOT NULL CHECK (TIPO23 IN (2,3)),      -- 2: istanza di E2; 3: istanza di E3
B INT,
CONSTRAINT E2E3 CHECK ((TIPO23 = 2 AND B IS NOT NULL) OR (TIPO23 = 3 AND B IS NULL))   );

CREATE TABLE E4 (
K4 INT NOT NULL PRIMARY KEY,
C INT NOT NULL,
K1R1 INT NOT NULL REFERENCES E1,
K1R2 INT REFERENCES E1                          );

CREATE TABLE E4D (
K4 INT NOT NULL REFERENCES E4 ,
D INT NOT NULL,
PRIMARY KEY (K4,D)          );
```

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 che R2 referenzi solo tuple di E3

CREATE TRIGGER R2_E3
BEFORE INSERT ON E4
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.K1R2 IS NOT NULL AND NOT EXISTS (    SELECT * FROM E1
                                             WHERE N.K1R2 = E1.K1
                                             AND      E1.TIPO23 = 3     ) )
SIGNAL SQLSTATE '70001' ('La tupla inserita deve referenziare una  tupla di E3!');

-- Trigger che garantisce il rispetto del vincolo di cui al punto d). Il vincolo può essere violato solo inserendo
-- un valore di D > 20 per una tupla in E4 che già partecipa a R2

CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E4D
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.D > 20 AND EXISTS ( SELECT * FROM E4
                            WHERE N.K4 = E4.K4
                            AND      E4.K1R2 IS NOT NULL                ) )
SIGNAL SQLSTATE '70002' ('La tupla inserita ha D>20 e referenzia una  tupla di E4 che partecipa a R2!');
```