Sistemi Informativi T
18 gennaio 2021
Risoluzione

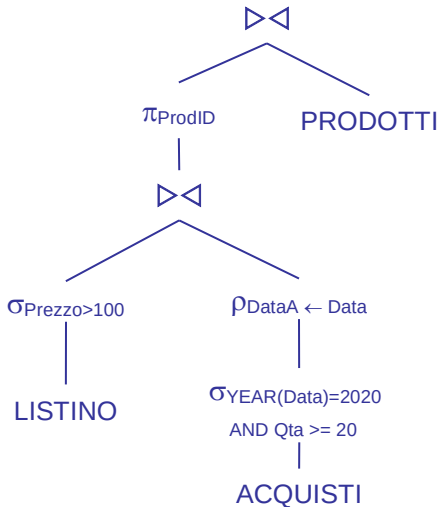
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

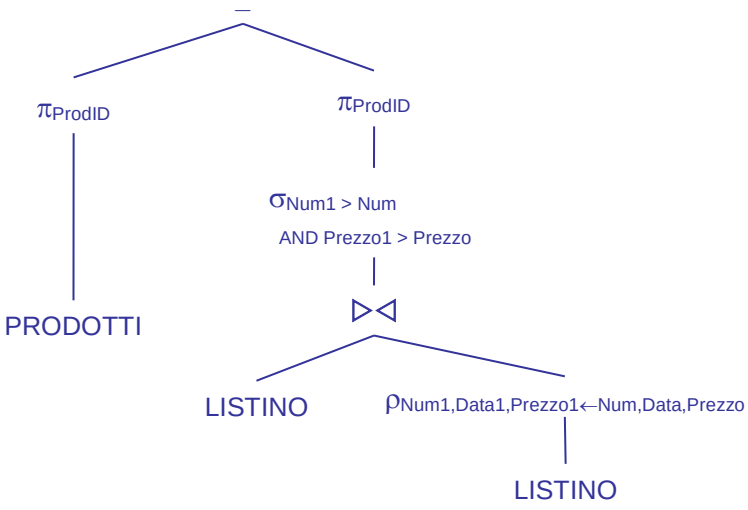
```
PRODOTTI(ProdID,Descrizione);
LISTINO(ProdID,Num,Data,Prezzo),
      ProdID REFERENCES PRODOTTI;
ACQUISTI(ProdID,Data,Qta),
      ProdID REFERENCES PRODOTTOI;
-- Per un dato prodotto, Num è un intero progressivo (1,2,…) che
-- ordina i successivi prezzi del prodotto.
-- Un Prezzo è valido dalla Data associata alla successiva, in cui cambia.
-- Per ogni acquisto il prezzo pagato è quello del listino valido
-- alla Data di acquisto.
-- Qta è di tipo INT, Prezzo è di tipo DEC(8,2).
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I dati dei prodotti che hanno avuto almeno una volta un prezzo maggiore di 100€ e per cui nel 2020 c'è stato almeno un acquisto di 20 o più unità



1.2) [2 p.] I codici dei prodotti che non hanno mai avuto un aumento di prezzo



Sistemi Informativi T
18 gennaio 2021
Risoluzione

2) SQL (5 punti totali)

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

2.1) [2 p.] Per ogni prodotto, il numero massimo di giorni in cui il prezzo non è variato (per semplicità si ignori il periodo che include la data attuale)

```
SELECT   L1.ProdID, MAX(DAYS(L2.Data)-DAYS(L1.Data)) AS NumMaxGiorni
FROM     LISTINO L1, LISTINO L2
WHERE    L1.ProdID = L2.ProdID
AND      L2.Num = L1.Num + 1
GROUP BY L1.ProdID;
```

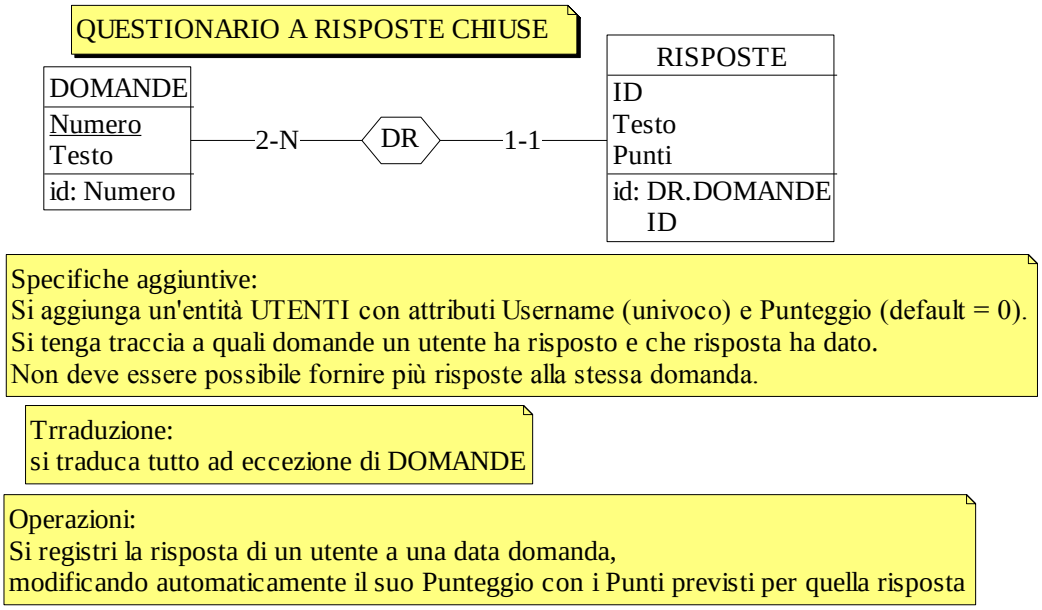
2.2) [3 p.] Il prodotto che ha complessivamente incassato di più

```
WITH PRODTOTINC (ProdID,TotIncassi) AS (
   SELECT L.ProdID, SUM(L.Prezzo*I.Qta)
   FROM   LISTINO L, ACQUISTI I
   WHERE  L.ProdID = I.ProdID
   AND    I.Data >= L.Data
   AND NOT EXISTS (  SELECT *
                     FROM   LISTINO L1
                     WHERE  L1.Data > L.Data
                     AND    L1.Data <= I.Data  )
   GROUP BY L.ProdID       )
SELECT *
FROM   PRODTOTINC P
WHERE  P.TotIncassi >= ALL (  SELECT  P1.TotIncassi
                              FROM   PRODTOTINC P1    );

-- La c.t.e. calcola l'incasso totale per ogni prodotto.
-- La subquery verifica che il prezzo di listino a cui un incasso è
-- associato sia effettivamente quello valido alla data dell'acquisto
```

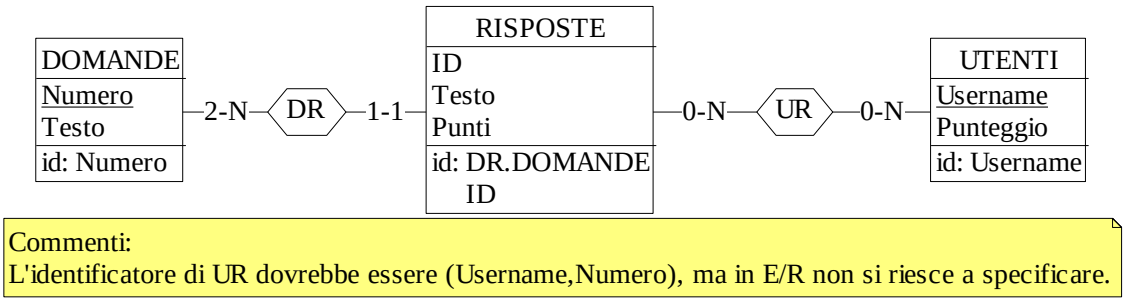
Sistemi Informativi T
18 gennaio 2021
Risoluzione

3) Modifica di schema E/R e del DB (6 punti totali)

Dato il file ESE3.lun fornito, in cui è presente lo schema ESE3-input in figura:



3.1) [2 p.] Si modifichi ESE3-input secondo le Specifiche aggiuntive;



3.2) [1 p.] Si copi lo schema modificato in uno schema ESE3-tradotto. Mediante il comando Transform/Quick SQL, si traduca la parte di schema specificata, modificando lo script SQL in modo da essere compatibile con DB2 e permettere l'esecuzione del punto successivo, ed eventualmente aggiungendo quanto richiesto dalle Specifiche aggiuntive;
Si veda il relativo file .sql

3.3) [3 p.] Si scriva l'istruzione SQL che modifica il DB come da specifiche (usare valori a scelta) e si definiscano i trigger necessari.

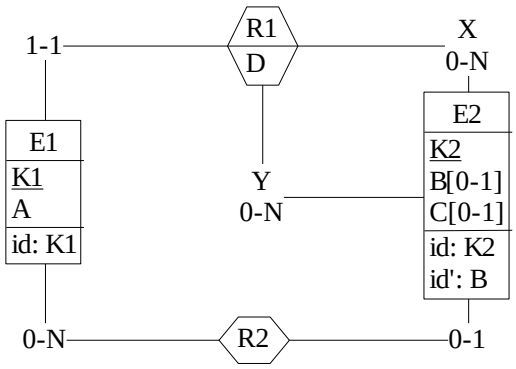
```
INSERT INTO UR
VALUES (:numdomanda,:idrisposta,:username);

CREATE OR REPLACE TRIGGER UPDATE_PUNTEGGIO
AFTER INSERT ON UR
REFERENCING NEW AS N
FOR EACH ROW
UPDATE     UTENTI
SET        PUNTEGGIO = PUNTEGGIO + (
                  SELECT R.PUNTI
                  FROM    RISPOSTE R
                  WHERE (R.NUMDOMANDA,R.ID) = (N.NUMDOMANDA,N.ID) )
WHERE      USERNAME = N.USERNAME ;

-- NUMDOMANDA è il nome usato nello script SQL al posto di Numero
```

Sistemi Informativi T
18 gennaio 2021
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) nessuna associazione viene tradotta separatamente;
b) per ogni istanza di E1, i ruoli X e Y dell'associazione R1 riguardano istanze di E2 con valori di C non nulli, diversi tra loro e la cui somma è minore del relativo D;



4.1) [3 p.] Si progettino gli opportuni schemi relazionali e si definiscano tali schemi mediante uno script SQL compatibile con DB2

```
CREATE TABLE E2 (
K2          INT NOT NULL PRIMARY KEY,        -- il tipo degli attributi non è necessariamente INT
B           INT,
C           INT,
K1R2        INT );

CREATE TABLE E1 (
K1          INT NOT NULL PRIMARY KEY,
A           INT NOT NULL,
D           INT NOT NULL,
K2X         INT NOT NULL REFERENCES E2,
K2Y         INT NOT NULL REFERENCES E2       );

ALTER TABLE E2
ADD CONSTRAINT FKR2 FOREIGN KEY (K1R2) 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**

```
-- Trigger che garantisce l'unicità dei valori di B
CREATE TRIGGER E2_KEY
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (    SELECT *
                   FROM E2
                   WHERE N.B = E2.B    ) )
SIGNAL SQLSTATE '70001' ('I valori di B non possono essere duplicati!');

-- Trigger che garantisce il rispetto del vincolo al punto b)
CREATE TRIGGER PUNTO_B
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (NOT EXISTS ( SELECT * FROM E2 X, E2 Y
                    WHERE N.K2X = X.K2
                   AND      N.K2Y = Y.K2
                   AND      X.C <> Y.C          -- se vero implica che entrambi sono non nulli
                   AND      X.C + Y.C < N.D ) )
SIGNAL SQLSTATE '70002' ('La tupla inserita non rispetta il vincolo del punto b)! ');
```