Sistemi Informativi T
16 gennaio 2013
Risoluzione

Tempo a disposizione: 2:30 ore

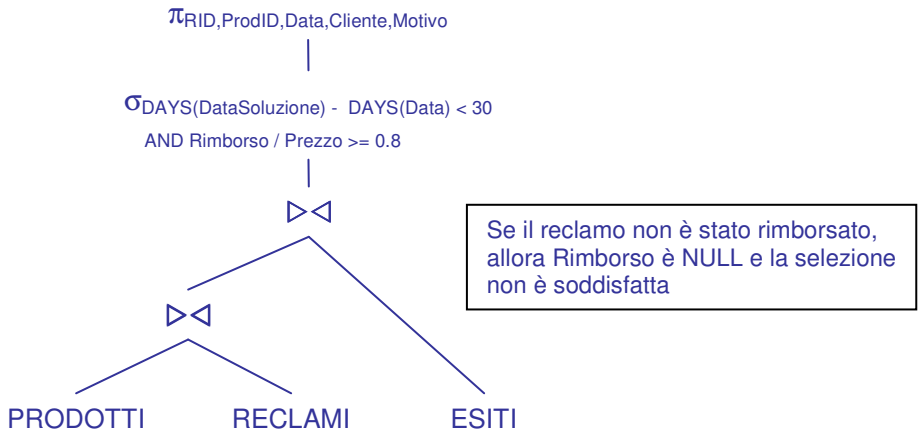
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

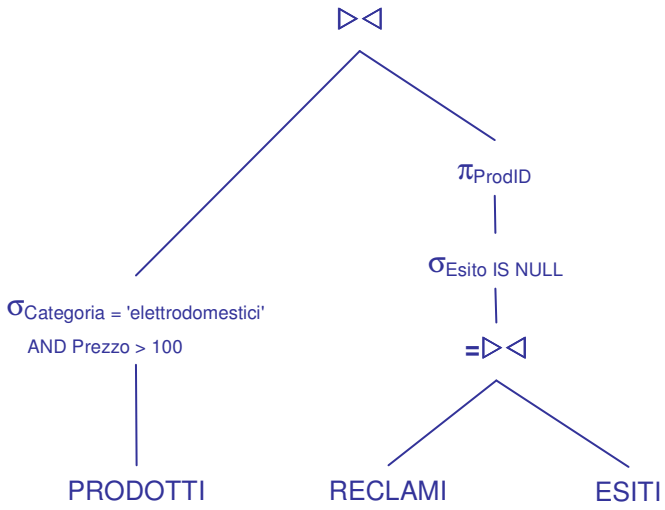
```
PRODOTTI(ProdID,Categoria,Prezzo);
RECLAMI(RID,ProdID,Data,Cliente,Motivo),
ProdID REFERENCES PRODOTTI;
ESITI(RID,DataSoluzione,Esito,Rimborso*),
RID REFERENCES RECLAMI;
-- DataSoluzione = la data in cui si e' deciso come trattare il reclamo
-- Esito = descrive se e come il reclamo è stato accolto
-- Se Esito = 'RIMBORSO', allora l'attributo Rimborso riporta l'importo
--   rimborsato al cliente (minore o uguale del prezzo del prodotto),
--   altrimenti Rimborso è NULL
-- Prezzo e Rimborso sono di tipo DEC(8,2)
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I dati dei reclami che sono stati risolti in meno di 30 giorni e che hanno dato luogo a un rimborso almeno pari all'80% del prezzo del prodotto



1.2) **[2 p.]** I dati dei prodotti di categoria 'elettrodomestici' di prezzo superiore ai 100 € per cui esiste un reclamo senza esito



Sistemi Informativi T
16 gennaio 2013
Risoluzione

SQL (5 punti totali)

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

2.1) **[2 p.]** I dati dei prodotti di categoria 'elettrodomestici' di prezzo superiore ai 100 € per cui esiste un reclamo senza esito

```
SELECT   P.*
FROM     PRODOTTI P
WHERE    P.Categoria = 'elettrodomestici'
AND      P.Prezzo > 100
AND      P.ProdID IN
         ( SELECT    R.ProdID
           FROM      RECLAMI R LEFT JOIN ESITI E ON (R.RID = E.RID)
           WHERE     E.RID IS NULL           )

-- Le tuple aggiunte dal left join corrispondono ai reclami senza esito
```

2.2) **[3 p.]** La categoria di prodotti con la massima percentuale di reclami rimborsati, considerando solo i reclami con esito definito

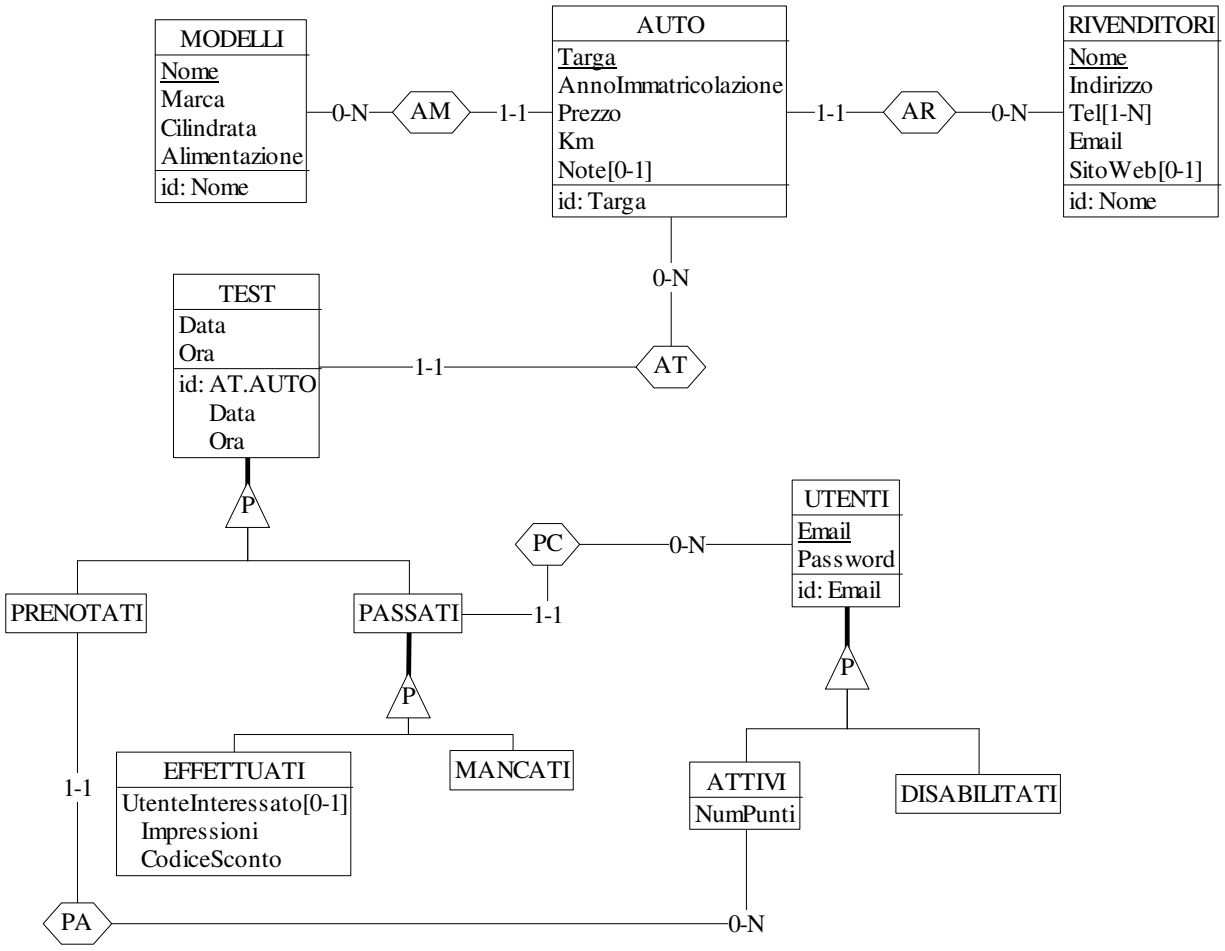
```
WITH RECLAMICAT (Categoria, TotReclami, TotRimborsati) AS (
   SELECT P.Categoria, COUNT(E.RID), COUNT(E.Rimborso)
   FROM   PRODOTTI P, RECLAMI R, ESITI E
   WHERE  P.ProdID = R.ProdID
   AND    R.RID = E.RID
   GROUP BY  P.Categoria )
SELECT RC1.Categoria
FROM   RECLAMICAT RC1
WHERE  CAST(RC1.TotRimborsati AS DEC(6,2))/ RC1.TotReclami =
      (  SELECT MAX(CAST(RC2.TotRimborsati AS DEC(6,2)) / RC2.TotReclami)
         FROM   RECLAMICAT RC2    )

-- Poiché Rimborso è NULL se il reclamo non ha dato luogo a rimborso,
-- l'espressione COUNT(E.Rimborso) conta quanti sono i reclami rimborsati
-- Si noti l'uso del CAST, che è sufficiente sia applicato a uno solo degli
-- operandi della divisione (che altrimenti restituirebbe un intero)
```

Sistemi Informativi T
16 gennaio 2013
Risoluzione

2) Progettazione concettuale (6 punti)

Il sito FourWheels (4W) dà la possibilità di accedere a tutte le informazioni sul mercato delle auto usate. Oltre alle caratteristiche di base (nome, cilindrata, alimentazione) dei modelli, per ogni specifica auto vengono forniti i dati di utilizzo (anno immatricolazione, km percorsi, prezzo richiesto ed eventuali note) e quelli del rivenditore (nome, indirizzo, uno o più telefoni, email ed eventuale sito web). 4W consente a chi si registra (basta un indirizzo email, che funge da username, e una password) di prenotare test di prova delle auto di interesse, concordando per ogni test data e ora. Al termine di un test, il venditore lo comunica a 4W e l'utente, se interessato, può così scrivere sul sito le impressioni avute dal test e quindi ricevere un codice utile per ottenere uno sconto sull'acquisto dell'auto.
Per i clienti che prenotano un test senza né disdirlo (che comporta semplicemente la cancellazione della relativa prenotazione) né effettuarlo, 4W conteggia 1 punto negativo. Totalizzati 3 punti l'utenza viene disabilitata e non si possono più creare registrazioni con la stessa email (pertanto solo gli utenti attivi, ovvero non disabilitati, possono avere dei test ancora da eseguire).



Commenti:

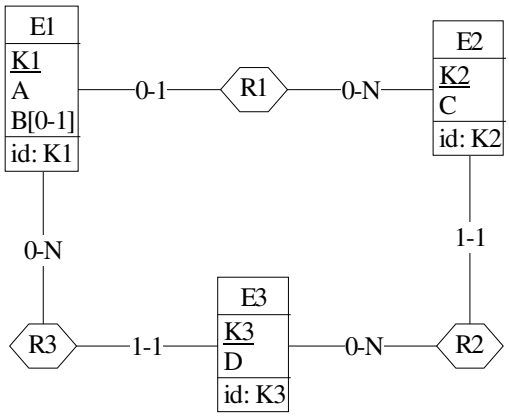
- La sola difficoltà dell'esercizio è data dalla modellazione precisa delle due gerarchie, la cui soluzione viene suggerita dall'ultima frase delle specifiche (*solo gli utenti attivi, ovvero non disabilitati, possono avere dei test ancora da eseguire*). L'associazione PC tiene traccia di quali utenti hanno prenotato test passati, effettuati o meno, e tra questi vi sono anche gli utenti DISABILITATI.
- Lo schema non modella il vincolo che ATTIVI.NumPunti deve essere uguale al numero di legami che un utente attivo ha con MANCATI, e che tale numero di legami è necessariamente pari a 3 per gli utenti DISABILITATI.
- Una soluzione alternativa (non esplicitamente esclusa dalle specifiche, ma allo stesso tempo basata su un'assunzione restrittiva) consiste nell'ipotizzare che il DB non tenga traccia dei test MANCATI. In questo caso PASSATI equivarrebbe a EFFETTUATI, e quindi la gerarchia si semplificherebbe.

Sistemi Informativi T
16 gennaio 2013
Risoluzione

Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) tutti gli attributi sono di tipo INT;
b) nessuna associazione viene tradotta separatamente;
c) un'istanza di E1 non è mai associata, tramite le associazioni R1, R2 e R3, a un'istanza di E1 con B nullo;



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,
B INT,
K2 INT        );

CREATE TABLE E3 (
K3 INT NOT NULL PRIMARY KEY,
D INT NOT NULL,
K1 INT NOT NULL REFERENCES E1            );

CREATE TABLE E2 (
K2 INT NOT NULL PRIMARY KEY,
C INT NOT NULL,
K3 INT NOT NULL REFERENCES E3    );

ALTER TABLE E1
ADD CONSTRAINT FKR1 FOREIGN KEY (K2) REFERENCES E2;
```

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

```
-- Per garantire il rispetto del vincolo di cui al punto c) va innanzitutto notato che esso può essere violato solo da un
-- inserimento in E1 (inserendo in E2 una tupla (k2,c,k3), ovviamente non può già esistere in E1 una tupla con valore
-- K2 = k2, e quindi non si crea alcun ciclo; analogamente per inserimenti in E3).
-- Pertanto è necessario definire solo il seguente trigger:

CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (     SELECT *
                   FROM E1, E2, E3
                   WHERE N.K2 = E2.K2
                   AND E2.K3 = E3.K3
                   AND E3.K1 = E1.K1
                   AND E1.B IS NULL )    )
SIGNAL SQLSTATE '70001' ('La tupla inserita referenzia una tupla di E1 con valore nullo per B!');
```