Sistemi Informativi T
25 giugno 2019
Risoluzione

Tempo a disposizione: 2:30 ore

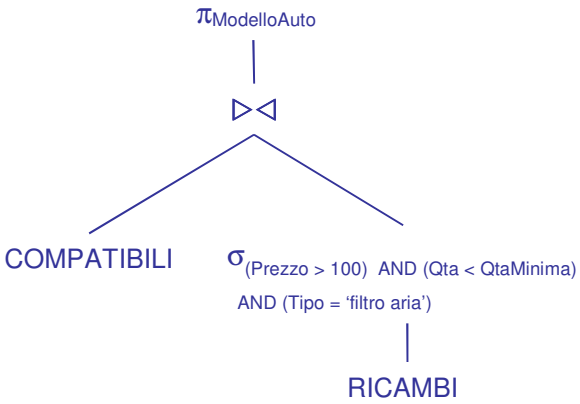
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

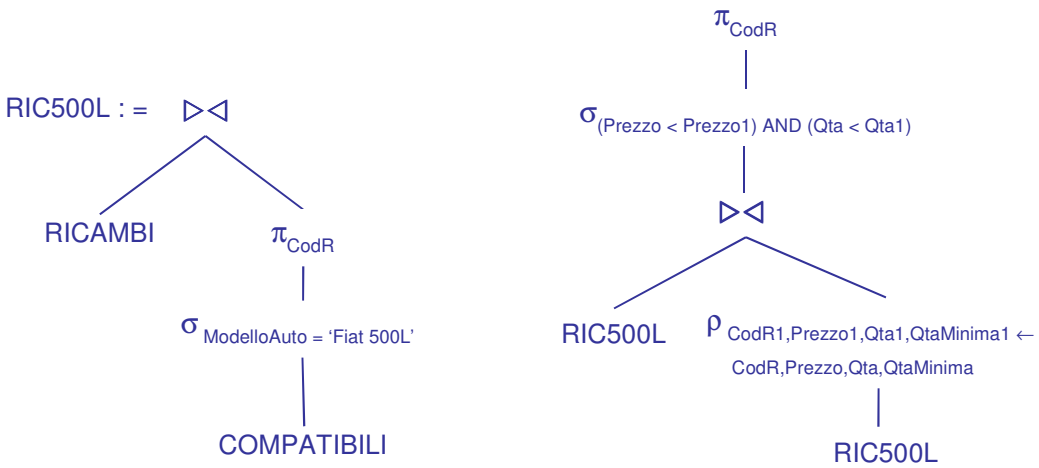
```
RICAMBI(CodR,Tipo,Prezzo,Qta,QtaMinima);
COMPATIBILI(CodR,ModelloAuto),
CodR references RICAMBI;
-- Qta è la quantità disponibile, QtaMinima quella sotto la quale
-- il ricambio va riordinato (entrambi interi)
-- Prezzo è di tipo DEC(6,2)
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I modelli di auto che montano un 'filtro aria' che costa più di 100€ e che deve essere riordinato



1.2) [2 p.] I codici dei ricambi per una 'Fiat 500L' per cui ne esiste un altro dello stesso tipo che però costa di più ma ha maggior disponibilità



Sistemi Informativi T
25 giugno 2019
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 tipo di ricambio, considerando solo i ricambi compatibili con modelli della Fiat, la quantità complessiva disponibile

```
SELECT   R.Tipo, SUM(Qta) AS QtaTotale
FROM     RICAMBI R
WHERE    R.CodR IN ( SELECT C.CodR
                     FROM   COMPATIBILI C
                     WHERE  C.ModelloAuto LIKE '%Fiat%' )
GROUP BY R.Tipo;

-- Senza fare uso di una subquery, ovvero scrivendo:
-- SELECT      R.Tipo, SUM(Qta) AS QtaTotale
-- FROM        RICAMBI R, COMPATIBILI C
-- WHERE       R.CodR = C.CodR
-- AND         C.ModelloAuto LIKE '%Fiat%'
-- GROUP BY    R.Tipo;
-- uno stesso ricambio viene contato più volte (tante quanti sono i
-- modelli della Fiat con cui è compatibile), e quindi il risultato
-- è errato
```

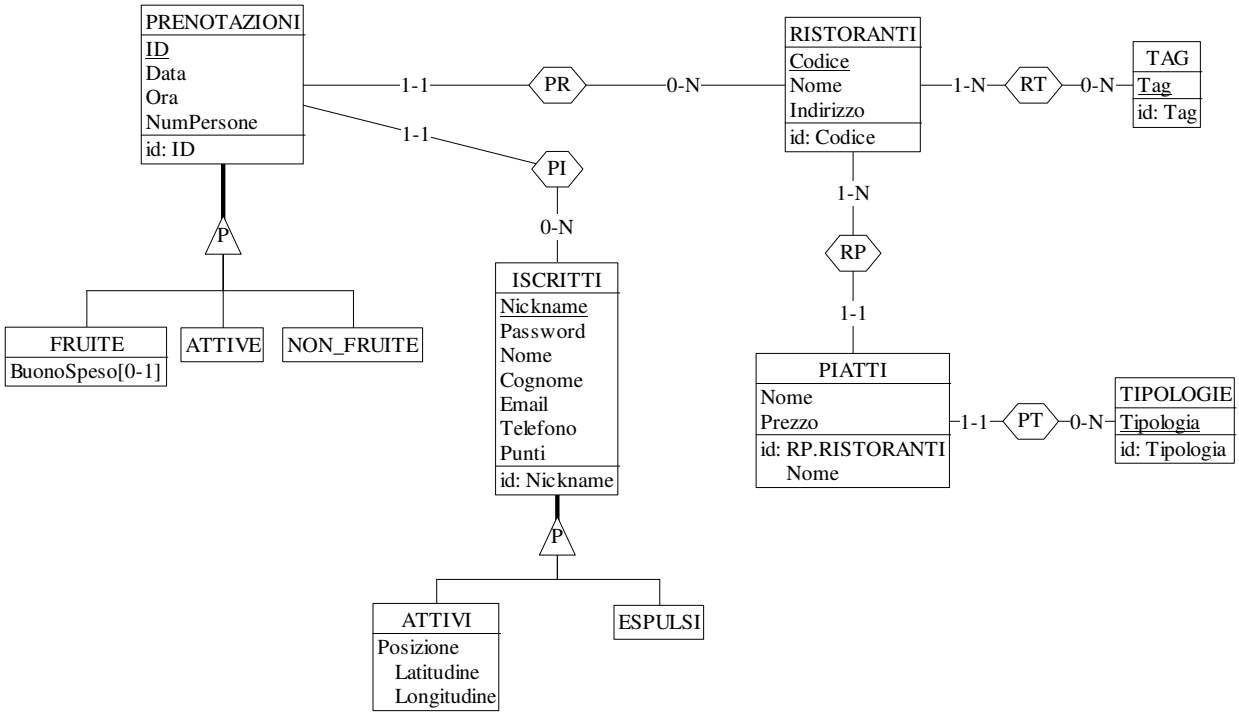
2.2) [3 p.] Per ogni tipo di ricambio, i dati del ricambio per cui è necessario spendere di più per riportare la quantità disponibile al minimo previsto

```
WITH
   SPESA_RIC (CodR,Tipo,Spesa) AS (
   SELECT   R.CodR, R.Tipo, R.Prezzo*(R.QtaMinima - R.Qta)
   FROM     RICAMBI R
   WHERE    R.Qta < R.QtaMinima   )

SELECT   R.*
FROM     RICAMBI R
WHERE    R.CodR IN ( SELECT S.CodR
                     FROM   SPESA_RIC S
                     WHERE  S.Spesa = ( SELECT MAX(S1.Spesa)
                                        FROM   SPESA_RIC S1
                                        WHERE  S1.Tipo = S.Tipo ) );

-- La c.t.e. restituisce, per ogni ricambio, la spesa da sostenere per
-- riportare la quantità disponibile al minimo previsto.
```

Sistemi Informativi T
25 giugno 2019
Risoluzione

3) Progettazione concettuale (6 punti)

MyKnife è un'applicazione che permette ai suoi iscritti di prenotare tavoli presso i ristoranti convenzionati. Per ogni ristorante, MyKnife mantiene un codice (univoco), nome, indirizzo, un elenco di piatti offerti (per ogni piatto è dato il nome, il prezzo e la tipologia: primo, secondo, ecc.) e uno o più tag per definire il tipo di cucina (pesce, pizza, mediterranea, ecc.). Sia le tipologie dei piatti che i tag sono predefiniti dal sistema.
Ogni utente ha un nickname (univoco), una password, un indirizzo email e un numero di telefono. Per favorire le operazioni di ricerca, MyKnife mantiene per ogni utente l'ultima posizione rilevata (latitudine, longitudine).
Una prenotazione viene fatta da un utente specificando il ristorante, la data, l'ora e il numero di persone. Per le prenotazioni effettivamente fruite, MyKnife eroga all'utente 100 punti, al raggiungimento di 1000 punti è possibile convertire i punti in un buono da 10€ da spendere in un ristorante (cosa che quando avviene viene registrata da MyKnife). Nel caso di prenotazioni non fruite, ovvero l'utente né ha cancellato la prenotazione (cosa che semplicemente elimina la prenotazione dal sistema) né si è presentato al ristorante, MyKnife sottrae 50 punti all'utente. Al raggiungimento di 5 prenotazioni non fruite l'utente viene espulso dal sistema (ma i suoi dati vengono mantenuti).

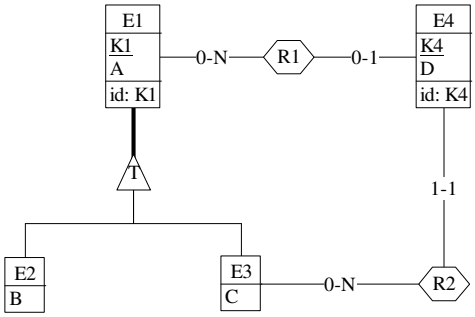


Commenti:

- La modellazione relativa alla parte di specifiche che riguarda i punti guadagnati e sottratti è implicitamente catturata dalle PRENOTAZIONI FRUITE e NON_FRUITE associate a un iscritto (e quindi anche il numero di PRENOTAZIONI NON_FRUITE, che al limite può mantenersi come attributo derivato).
- ISCRITTI.Punti è un attributo derivato, in quanto il totale punti può ricavarsi dalle prenotazioni fruite, da quelle non fruite, e dai buoni spesi.

Sistemi Informativi T
25 giugno 2019
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) le entità E1, E2 ed E3 vengono tradotte insieme;
c) le associazioni R1 e R2 non vengono tradotte separatamente;
d) per ogni istanza di E4 che partecipa all'associazione R1, i valori dell'istanza associata di E1 e dell'istanza di E3 associata tramite R2 sono tali che C > A;

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,
SEL12 SMALLINT NOT NULL CHECK (SEL12 IN (1,2)),          -- 2: istanza anche di E2
B    INT,
SEL13 SMALLINT NOT NULL CHECK (SEL13 IN (1,3)),          -- 3: istanza anche di E3
C    INT,
CONSTRAINT HIER_T CHECK (SEL12 = 2 OR SEL13 = 3),
CONSTRAINT E2 CHECK ((SEL12 = 1 AND B IS NULL) OR (SEL12 = 2 AND B IS NOT NULL) ),
CONSTRAINT E3 CHECK ((SEL13 = 1 AND C IS NULL) OR (SEL13 = 3 AND C IS NOT NULL) )  );

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

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 ';')

```
-- Trigger che garantisce che una tupla di E4 referenzi, tramite K1R2, una tupla di E3
CREATE TRIGGER R2_E3
BEFORE INSERT ON E4
REFERENCING NEW AS N
FOR EACH ROW
WHEN (NOT EXISTS  ( SELECT * FROM E1
                    WHERE    N.K1R2 = E1.K1
                    AND      E1.SEL13 = 3 )      )
SIGNAL SQLSTATE '70001' ('La tupla inserita deve referenziare una tupla di E3!');

-- Trigger che garantisce il rispetto del vincolo al punto d)
CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E4
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( (    SELECT E1.A
            FROM    E1
            WHERE N.K1R1 = E1.K1  )   >=
       (    SELECT E1.C
            FROM    E1
            WHERE N.K1R2 = E1.K1  ) )
SIGNAL SQLSTATE '70002' ('La tupla è associata a tuple di E1 ed E3 con A >= C!');
```