Sistemi Informativi T
20 settembre 2011
Risoluzione

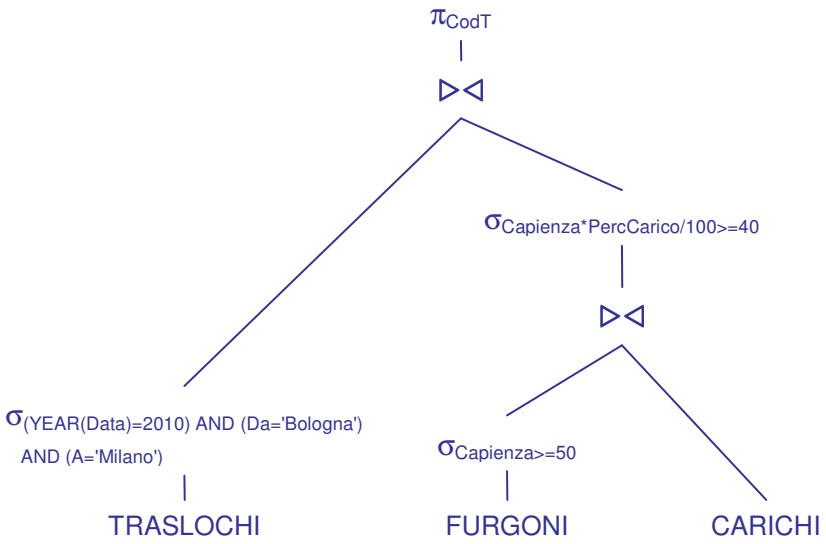
Tempo a disposizione: 2:30 ore

1) Algebra relazionale (3 punti totali):

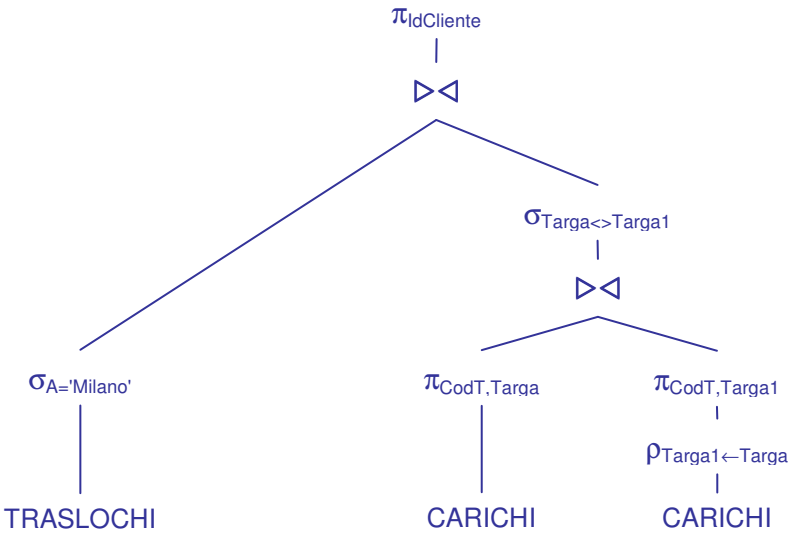
Date le seguenti relazioni:

```
FURGONI(Targa,Capienza);
TRASLOCHI(CodT,Da,A,Prezzo,Data,IdCliente);
CARICHI(Targa,CodT,PercCarico),
Targa REFERENCES FURGONI, CodT REFERENCES TRASLOCHII;
-- Capienza e' un intero che rappresenta i metri cubi di capacita'
-- del furgone
-- PercCarico e' un intero tra 1 e 100 che indica la percentuale di
-- utilizzo della capienza di un furgone in un dato trasloco:
```

1.1) **[1 p.]** I traslochi del 2010 da Bologna a Milano che hanno utilizzato un furgone di capienza almeno pari a 50 m^3 (metri cubi) ed effettivamente caricato almeno per 40 m^3



1.2) **[2 p.]** I clienti che hanno traslocato a Milano utilizzando almeno 2 furgoni



Sistemi Informativi T
20 settembre 2011
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 coppia "Da, A", il volume totale traslocato nel 2010

```
SELECT   T.Da, T.A, SUM(F.Capienza*C.PercCarico/100)
FROM     TRASLOCHI T, FURGONI F, CARICHI C
WHERE    F.Targa = C.Targa
AND      T.CodT = C.CodT
AND      YEAR(Data) = 2010
GROUP BY T.Da, T.A
```

2.2) [3 p.] Per ogni valore di capienza il furgone che, in media, è stato maggiormente caricato

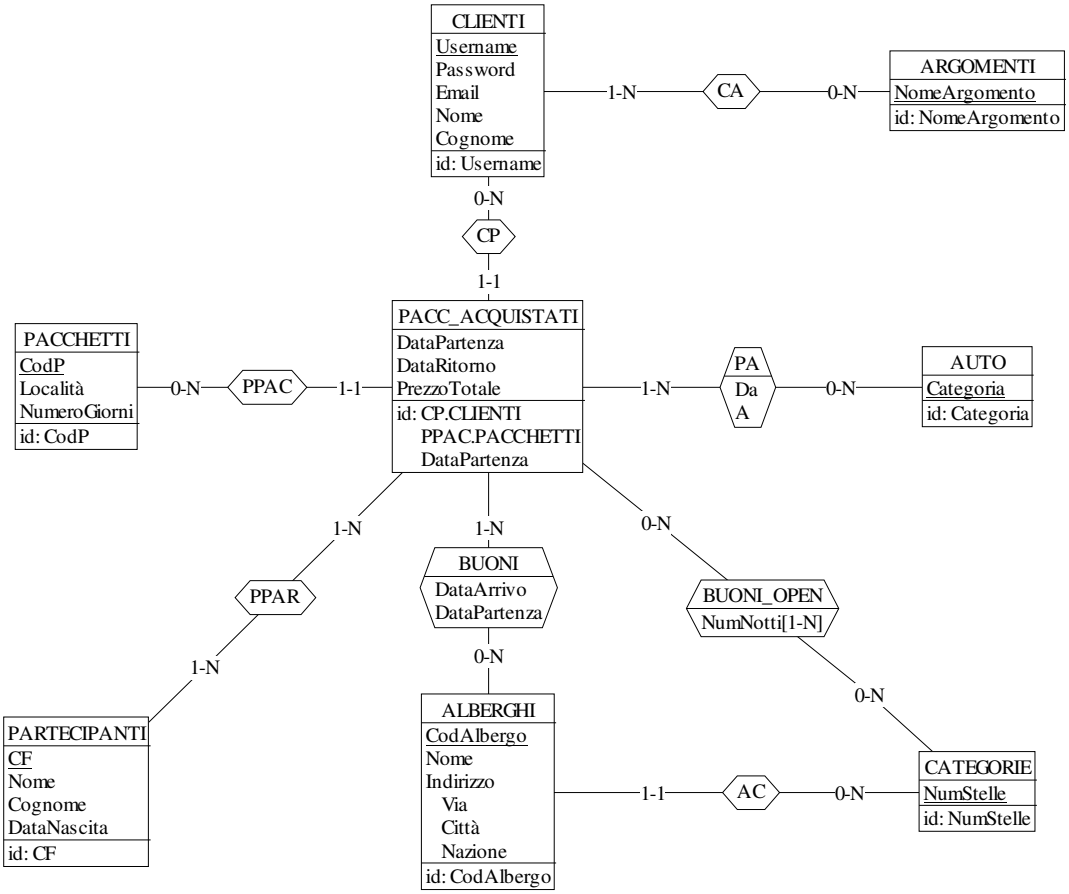
```
WITH
CARICHIMEDI(Targa, Capienza, CaricoMedio) AS (
   SELECT F.Targa, F.Capienza, AVG(C.PercCarico)
   FROM   FURGONI F, CARICHI C
   WHERE  F.Targa = C.Targa
   GROUP BY  F.Targa, F.Capienza )
SELECT CM.Targa, CM.Capienza, CM.CaricoMedio
FROM   CARICHIMEDI CM
WHERE  CM.CaricoMedio = ( SELECT MAX(CM1.CaricoMedio)
                          FROM CARICHIMEDI CM1
                          WHERE CM1.Capienza = CM.Capienza )

-- La Common Table Expression calcola il carico medio di ogni furgone.
-- Il blocco SELECT confronta quindi tali carichi a parita' di capienza
-- e restituisce il furgone con carico medio massimo.
-- Si noti che non e' strettamente necessario nella CTE raggruppare anche
-- su Capienza. In tal caso e' pero' necessario riutilizzare Furgoni
-- nel blocco SELECT.
```

Sistemi Informativi T
20 settembre 2011
Risoluzione

3) Progettazione concettuale (6 punti)

L'agenzia di viaggi TravelWorld (TW) propone pacchetti di viaggio "fai da te" particolarmente vantaggiosi, riservati ai propri clienti registrati sul sito www.TW.net. Oltre a username, password e dati anagrafici, ogni cliente esprime sul sito uno o più argomenti di viaggio di suo interesse, scelti tra quelli proposti da TW.
Un pacchetto comprende volo andata a ritorno per la località prescelta, dei buoni per il pernottamento negli alberghi scelti (tra quelli convenzionati con TW), e l'affitto di una o più auto. Ogni cliente, all'atto dell'acquisto di un pacchetto, deve specificare le date di partenza e ritorno e il numero di partecipanti al viaggio, completi di generalità (nome, cognome e data di nascita). Deve inoltre scegliere gli alberghi in cui soggiornare, con le relative date di arrivo e partenza, e le auto desiderate, ognuna di una certa categoria (economy, 4x4, SUV, ecc.) e per un determinato periodo.
Opzionalmente, un cliente può acquistare uno o più buoni "open", ognuno da utilizzare per soggiornare in un albergo convenzionato ma non ancora scelto dal cliente. Un buono open riporta pertanto solo il numero di notti per cui è valido e la categoria (numero di stelle) dell'albergo. Un pacchetto non può consistere solo di buoni open.

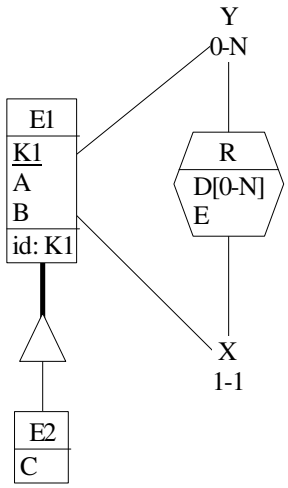


Commenti:

- La soluzione proposta è centrata sull'entità PACC_ACQUISTATI, che permette di rappresentare ciò che un cliente ha effettivamente acquistato. Data la vaghezza delle specifiche, e considerando che un "pacchetto" è qualcosa di fortemente personalizzabile, appare opportuno separare tale entità da quella dei PACCHETTI veri e propri, che qui consistono di un insieme minimale di attributi (più realisticamente, un pacchetto potrebbe anche comprendere vincoli sulle categorie di alberghi prenotabili, sulle tipologie di auto noleggiabili, ecc.)
- L'entità CATEGORIE permette di introdurre una sola volta nello schema tale concetto, che altrimenti andrebbe rappresentato, perdendo in precisione, come attributo di ALBERGHI e di BUONI_OPEN (quest'ultima, mancando l'entità CATEGORIE, diventerebbe un attributo composto e ripetuto di PACC_ACQUISTATI)
- Nell'associazione BUONI_OPEN l'attributo NumNotti è ripetuto, in quanto è possibile avere più buoni open relativi a una stessa categoria di alberghi

Sistemi Informativi T
20 settembre 2011
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 R non viene tradotta separatamente;
c) le entità E1 ed E2 non vengono tradotte assieme;
d) un'istanza di E1 non è mai associata, tramite il ruolo X di R, a istanze di E2 con C < 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,
B INT NOT NULL,
TIPO SMALLINT NOT NULL CHECK (TIPO IN (1,2)),          -- 2: istanza anche di E2
K1R INT NOT NULL REFERENCES E1,
E INT NOT NULL                );

CREATE TABLE E2 (
K1 INT NOT NULL PRIMARY KEY REFERENCES E1,
C INT NOT NULL     );

CREATE TABLE RD
K11 INT NOT NULL REFERENCES E1,
D INT NOT NULL,
PRIMARY KEY (K1,D)         );
```

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 il simbolo '@' per terminare gli statement SQL

```
-- Quando si inserisce una tupla in E1 di TIPO = 2 (ovvero di E2), bisogna anche eseguire, nella stessa
-- transazione, un inserimento in E2
-- Per garantire il rispetto del vincolo di cui al punto d) è necessario impostare il seguente trigger:

CREATE TRIGGER PUNTO_D
NO CASCADE BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW MODE DB2SQL
WHEN (EXISTS (SELECT *
                    FROM E2
                    WHERE E2.K1 = N.K1R
                    AND E2.C < 20 ))
SIGNAL SQLSTATE '70001' ('La tupla inserita referenzia una tupla con C<20!')@
```