Sistemi Informativi T
19 giugno 2017
Risoluzione

Tempo a disposizione: 2:30 ore

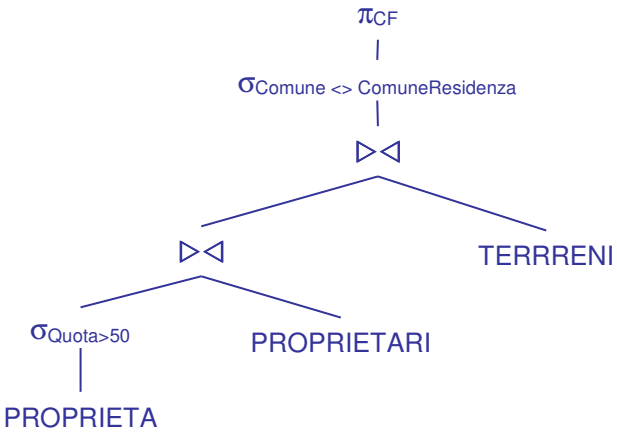
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

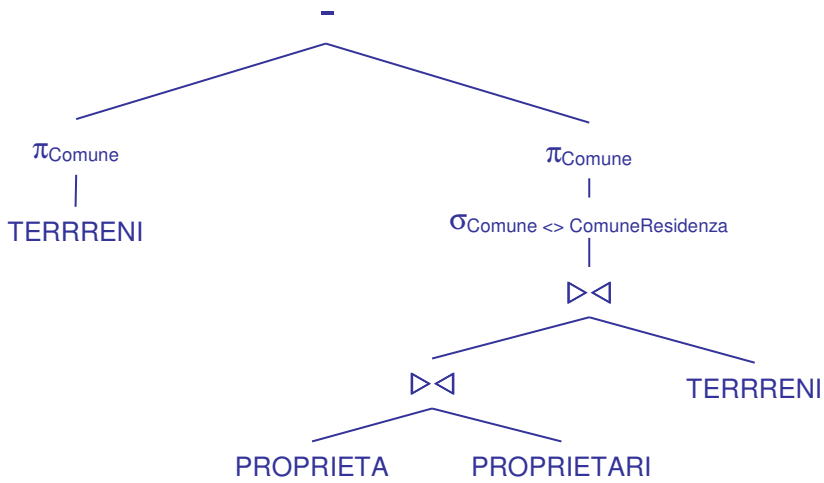
```
TERRENI(IDT,MetriQuadri,Comune);
PROPRIETARI(CF,ComuneResidenza);
PROPRIETA(CF,IDT,Quota),
      CF references PROPRIETARI,
      IDT references TERRENI;
-- MetriQuadri è di tipo INT
-- Quota è di tipo DEC(5,2) e rappresenta la percentuale di possesso
-- di un terreno (0 < Quota ≤ 100); Quota = 100 se il terreno ha un
-- solo proprietario
-- Il valore di Quota*MetriQuadri/100 rappresenta i metri quadri
-- effettivamente posseduti da un proprietario)
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I codici fiscali (CF) dei proprietari che possiedono per più del 50% un terreno in un comune diverso da quello in cui risiedono



1.2) [2 p.] I comuni in cui tutti i terreni hanno solo proprietari residenti nello stesso comune



L'operando destro della differenza calcola i comuni in cui c'è almeno un terreno con proprietario residente in altro comune

Sistemi Informativi T
19 giugno 2017
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 comune, il numero di terreni con singolo proprietario.
[+1 p. extra] Se il risultato include anche i comuni con 0 terreni con singolo proprietario

```
SELECT   T.Comune, COUNT(*)
FROM     TERRENI T, PROPRIETA P
WHERE    T.IDT = P.IDT
AND      P.Quota = 100
GROUP BY T.Comune

-- Per la parte opzionale è necessario fare uso di un outer join
-- (si veda anche la slide 19 in 04.5.SQL.viste)

SELECT   T.Comune, COUNT(P.CF)
FROM     TERRENI T LEFT OUTER JOIN PROPRIETA P ON (T.IDT = P.IDT)
         AND (P.Quota = 100)
GROUP BY T.Comune
```

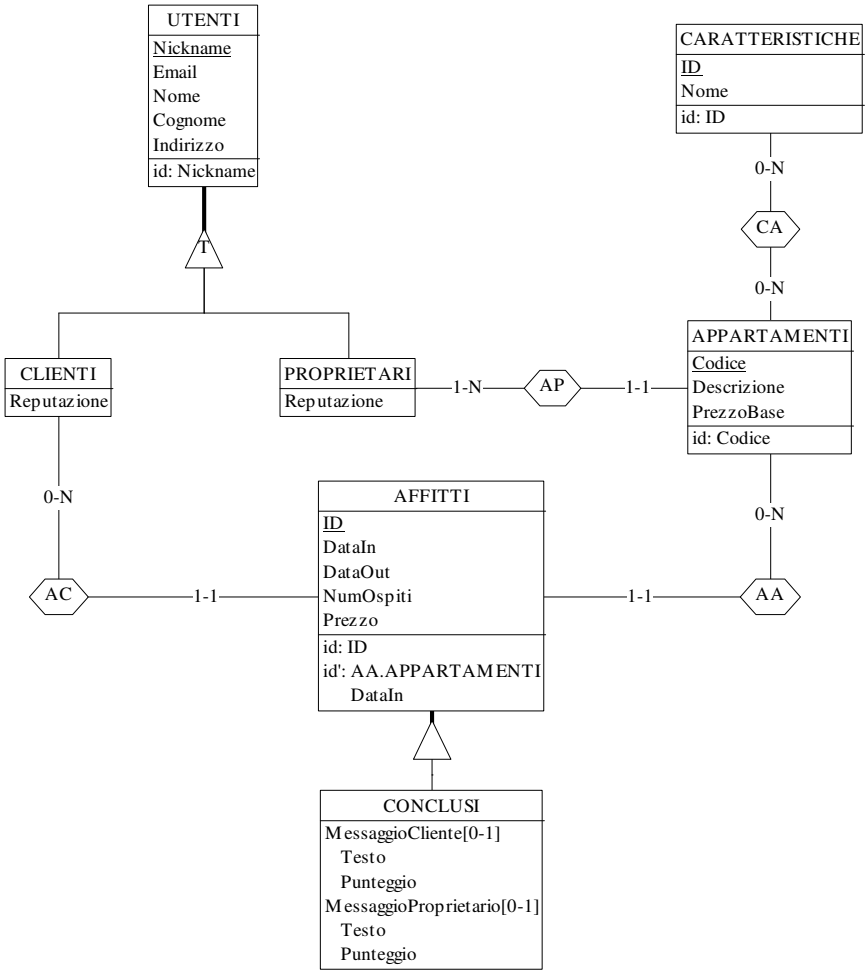
2.2) **[3 p.]** Per ogni comune, il numero di proprietari (residenti nello stesso comune o meno) che complessivamente posseggono più di 100000 metri quadri in quel comune

```
WITH
   MQ_TOT (Comune,CF) AS (
   SELECT   T.Comune, P.CF
   FROM     TERRENI T, PROPRIETA P
   WHERE    T.IDT = P.IDT
   GROUP BY T.Comune,P.CF
   HAVING   SUM(P.Quota,T.MetriQuadri)/100 > 100000 )

SELECT   M.Comune, COUNT(*)
FROM     MQ_TOT M
GROUP BY M.Comune

-- La c.t.e. restituisce, per ogni comune, i proprietari per cui il
-- totale dei m.q. posseduti in quel comune soddisfa il vincolo
```

Sistemi Informativi T
19 giugno 2017
Risoluzione

3) Progettazione concettuale (6 punti)

Il sistema EarthBnB tratta l'affitto di appartamenti per le vacanze per conto dei rispettivi proprietari. Sia proprietari che clienti (un cliente può anche essere proprietario) sono identificati da un nickname e per ognuno sono noti i dati anagrafici.
Ogni appartamento ha un codice univoco, una descrizione e una serie di caratteristiche booleane, prese da un elenco predefinito (ad es. aria condizionata, lavatrice, ascensore, ecc.). In funzione del periodo scelto per il soggiorno, del numero di giorni e del numero di ospiti, EarthBnB calcola automaticamente il prezzo finale a partire da un prezzo base di riferimento per quell'appartamento.
Per ogni contratto di affitto, EarthBnB registra i dati relativi (periodo, numero persone e prezzo). Al termine del soggiorno, sia il cliente che il proprietario sono tenuti a lasciarsi un messaggio di valutazione (testo e punteggio), che contribuisce a stabilire la relativa reputazione. Un utente ha, in generale, reputazioni diverse come cliente e come proprietario.
N.B. Si assume per semplicità che un appartamento non cambi proprietario nel corso del tempo.

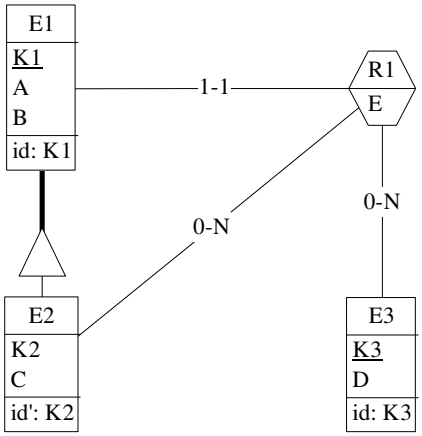


Commenti:

- L'esercizio non presenta difficoltà degne di nota.

Sistemi Informativi T
19 giugno 2017
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 ed E2 vengono tradotte insieme;
c) l'associazione R1 non viene tradotta separatamente;

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 E3 (
K3 INT NOT NULL PRIMARY KEY,
D INT NOT NULL                                  );

CREATE TABLE E12 (
K1 INT NOT NULL PRIMARY KEY,
A INT NOT NULL,
B INT NOT NULL,
K1E2 INT NOT NULL REFERENCES E12,
K3 INT NOT NULL REFERENCES E3,
E INT NOT NULL,
TIPO SMALLINT NOT NULL CHECK (TIPO IN (1,2)),          -- 2: istanza anche di E2
K2 INT,
C INT,
CONSTRAINT E2 CHECK ((TIPO2 = 1 AND C IS NULL AND K2 IS NULL) OR
                     (TIPO2 = 2 AND C IS NOT NULL AND K2 IS NOT NULL) )          );
```

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 ';')

```
CREATE TRIGGER UNIQUE_K2       -- Garantisce l'unicità dei valori di K2
BEFORE INSERT ON E12
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS  ( SELECT * FROM E12
                WHERE    N.K2 = E12.K2)  )
SIGNAL SQLSTATE '70001' ('I valori di K2 non possono essere duplicati!');

-- Trigger che garantisce che una tupla di R1 referenzi, tramite K1E2, una tupla di  E2
CREATE TRIGGER R1_E2
BEFORE INSERT ON E12
REFERENCING NEW AS N
FOR EACH ROW
WHEN (NOT EXISTS  ( SELECT * FROM E12
                    WHERE    N.K1E2 = E12.K1
                    AND      E12.TIPO = 2 )      )
SIGNAL SQLSTATE '70002' ('La tupla inserita deve referenziare una tupla di E2!');
```