Sistemi Informativi T
6 luglio 2015
Risoluzione

Tempo a disposizione: 2:30 ore

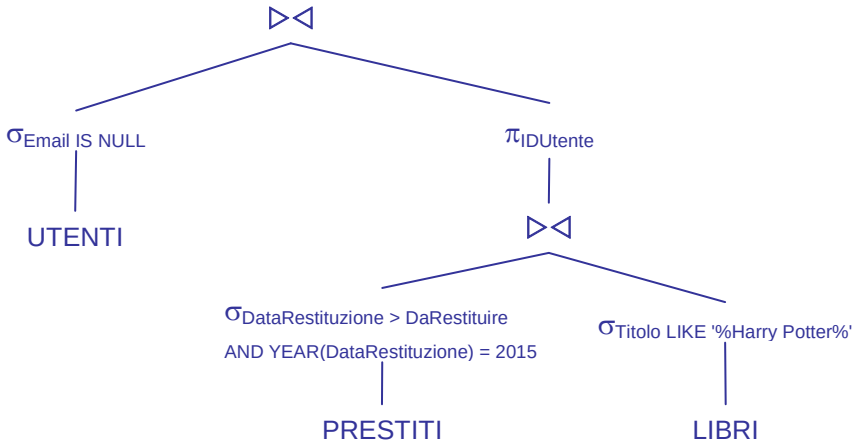
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

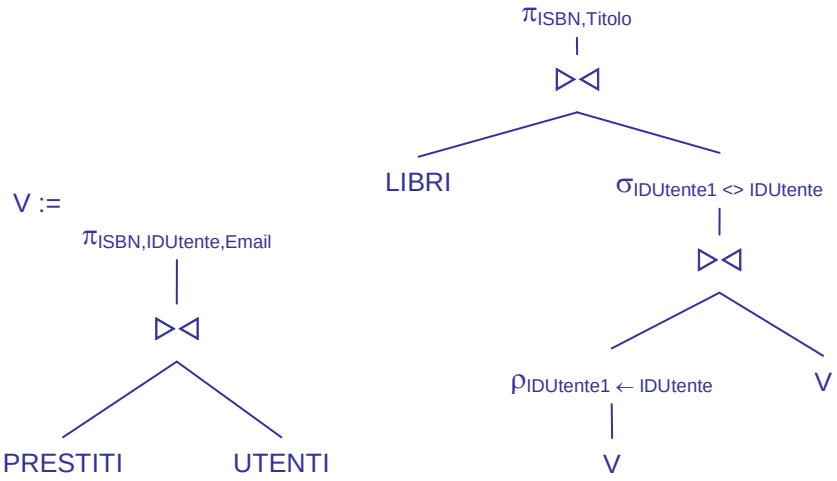
```
LIBRI(ISBN,Titolo,Autore);
UTENTI(IDUtente,Nome,Cognome,Email*);
PRESTITI(ISBN,IDUtente,Data,DaRestituire,DataRestituzione*),
ISBN references LIBRI, IDUtente references UTENTI;
-- l'asterisco indica la possibilità di valori nulli
-- DaRestituire (> Data) è la data in cui il libro va restituito
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I dati degli utenti senza email che nel 2015 hanno restituito oltre la data prevista un libro nel cui titolo compare 'Harry Potter'



1.2) **[2 p.]** I codici ISBN e i titoli dei libri presi in prestito da due (o più) utenti diversi, ma con la stessa email



Sistemi Informativi T
6 luglio 2015
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 utente, il numero di volte che ha restituito un libro in ritardo

```
SELECT   P.IdUtente, COUNT(*) AS NumRitardi
FROM     PRESTITI P
WHERE    P.DataRestituzione > P.DaRestituire
GROUP BY P.IDUtente
```

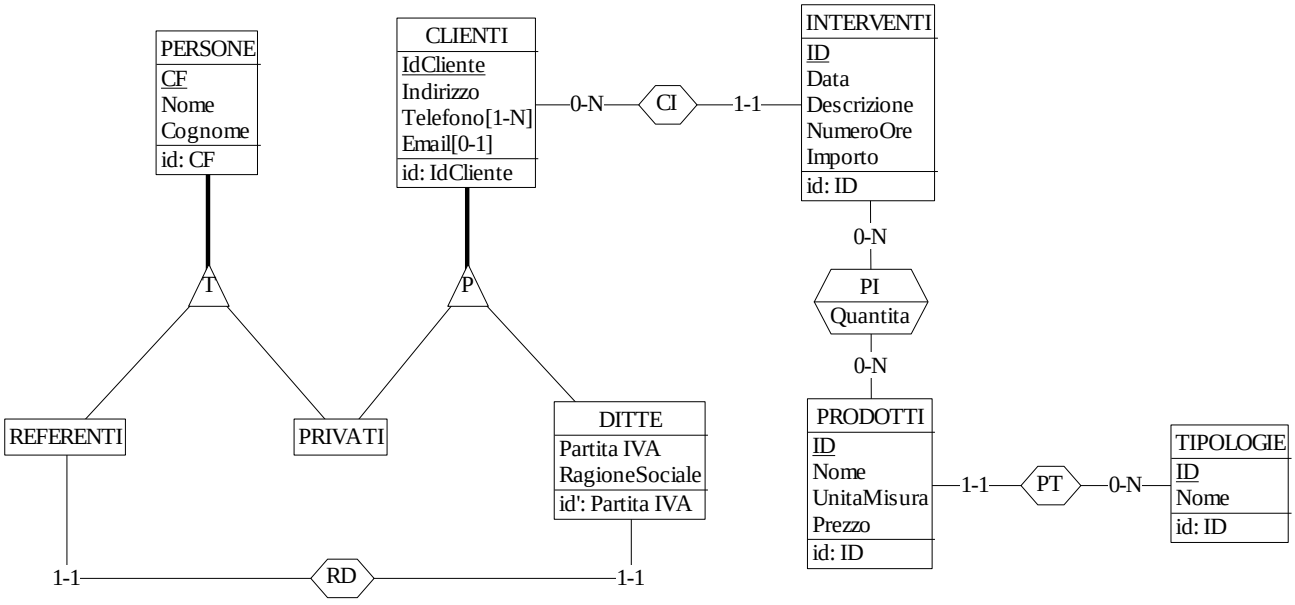
2.2) [3 p.] Per ogni anno, i dati del libro che, considerando tutti i prestiti iniziati in quell'anno e terminati lo stesso anno o anche dopo, ha totalizzato il maggior numero di giorni di ritardo nella restituzione

```
WITH RITARDI (Anno,ISBN,TotRitardo) AS (
   SELECT   YEAR(P.Data), P.ISBN,
             SUM(DAYS(DataRestituzione) - DAYS(P.DaRestituire))
   FROM     PRESTITI P
   WHERE    P.DataRestituzione > P.DaRestituire
   GROUP BY YEAR(P.Data), P.ISBN       )

SELECT   R.Anno, L.*
FROM     RITARDI R, LIBRI L
WHERE    R.ISBN = L.ISBN
AND      R.TotRitardo = ( SELECT MAX(R1.TotRitardo)
                          FROM   RITARDI R1
                          WHERE  R1.Anno = R.Anno )

-- La c.t.e. calcola, per ogni anno e ogni libro, il numero totale di giorni
-- di ritardo. Si noti la condizione P.DataRestituzione > P.DaRestituire,
-- senza la quale restituzioni prima della scadenza contribuirebbero alla
-- somma con un termine negativo
```

Sistemi Informativi T
6 luglio 2015
Risoluzione

3) Progettazione concettuale (6 punti)

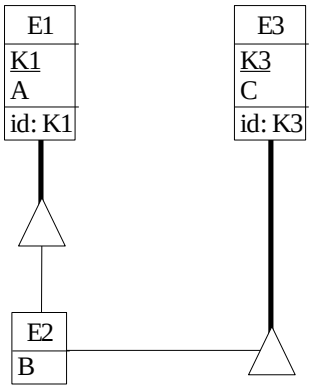
Jardin Vert (JV) è una ditta di giardinaggio che opera presso clienti privati e ditte. Per ogni cliente, JV opera con una serie di interventi, ognuno dei quali è caratterizzato da una descrizione, una data, il numero di ore impiegate e l'elenco con quantità degli eventuali prodotti utilizzati. Di ogni prodotto, oltre al codice identificativo, è dato il nome, la tipologia (concime, piante, ecc.), l'unità di misura e il prezzo unitario. Sulla base di queste informazioni viene quindi determinato un costo complessivo dell'intervento.
Per i clienti privati JV mantiene il codice fiscale (CF), nome, cognome, indirizzo, telefono (uno o più) ed eventuale email. Per le ditte JV mantiene la ragione sociale, il numero di Partita IVA (che identifica la ditta), indirizzo, telefono (uno o più), eventuale email, e i dati (CF, nome e cognome) di un dipendente di riferimento nella ditta stessa. Può capitare che la persona di riferimento di una ditta sia anche un cliente privato, nel qual caso le sue informazioni non devono essere duplicate nel sistema della JV.



Commenti:

- La sola difficoltà dell'esercizio risiede nella corretta modellazione delle relazioni gerarchiche esistenti tra le vari tipologie di clienti e persone.

Sistemi Informativi T
6 luglio 2015
Risoluzione



- **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 e separatamente da E3;
c) I valori di A e C di un'istanza di E2 sono sempre diversi tra loro;

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,
C INT NOT NULL     );
-- Volendo si può anche inserire un selettore per E2, ad es:
-- TIPO2 SMALLINT NOT NULL CHECK (TIPO2 IN (3,2))
-- In questo caso è necessario anche prevedere un trigger che ad ogni inserimento in E1 verifichi che la
-- tupla referenziata in E3 abbia TIPO2 = 2

CREATE TABLE E1 (
K1 INT NOT NULL PRIMARY KEY,
A INT NOT NULL,
TIPO2 SMALLINT NOT NULL CHECK (TIPO2 IN (1,2)),          -- 2: istanza anche di E2
B INT,
K3 INT REFERENCES E3,
CONSTRAINT E2 CHECK ((TIPO2 = 1 AND B IS NULL AND K3 IS NULL) OR
                                (TIPO2 = 2 AND B IS NOT NULL AND K3 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 ';')

```
-- Trigger che garantisce l'unicità dei valori di K3
CREATE TRIGGER UNIQUE_K3
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS  ( SELECT * FROM E1
                        WHERE    N.K3 = E1.K3)     )
SIGNAL SQLSTATE '70001' ('I valori di K3 non possono essere duplicati!');

-- Trigger che garantisce il rispetto del vincolo di cui al punto c)
CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.A = (SELECT E3.C FROM E3          -- ovviamente, se N.K3 è NULL non genera errore
                    WHERE N.K3 = E3.K3    ) )
SIGNAL SQLSTATE '70002' ('I valori di A e C di un"istamza di E3 devono essere diversi!');
```