Sistemi Informativi T
23 gennaio 2014
Risoluzione

Tempo a disposizione: 2:30 ore

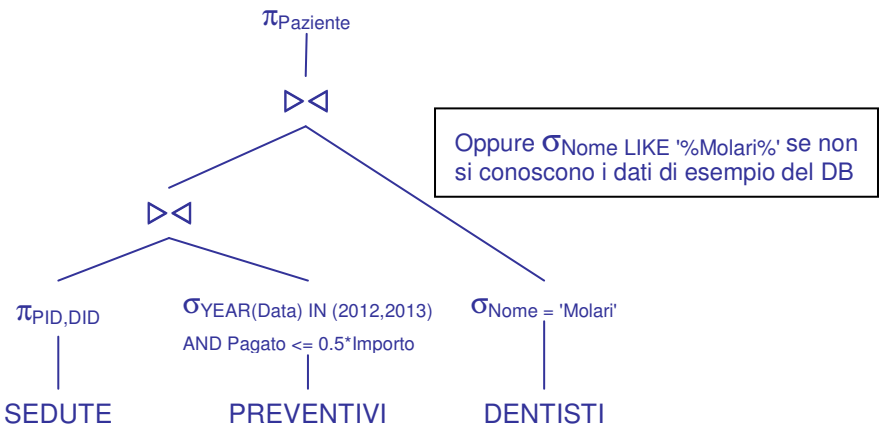
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

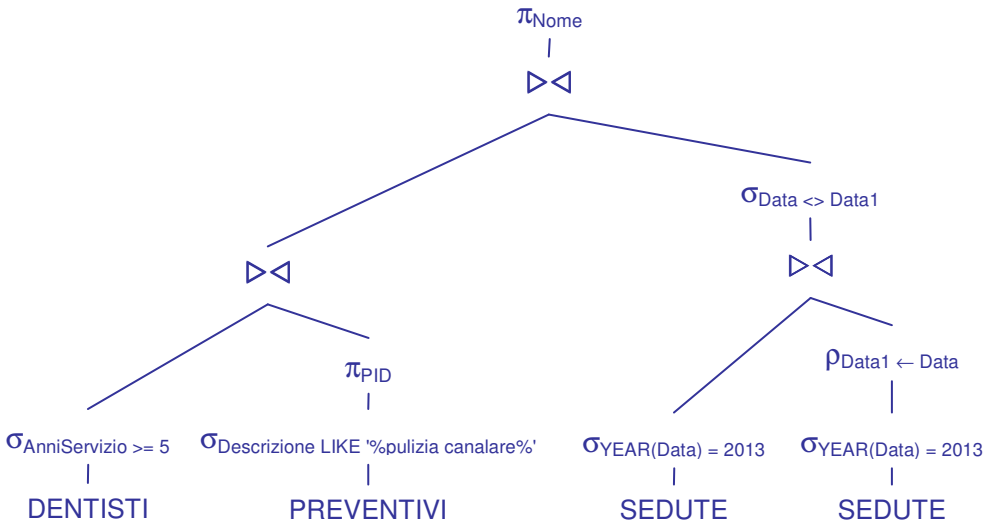
```
DENTISTI(DID,Nome,AnniServizio);
PREVENTIVI(PID,Data,Paziente,Descrizione,Importo,Pagato);
SEDUTE(PID,Data,DID),
      PID REFERENCES Preventivi,
      DID REFERENCES DENTISTI;
--
-- Importo e Pagato sono di tipo DEC(8,2)
-- Pagato ha valore di default 0 ed e' sempre <= Importo
-- PREVENTIVI.Data e' la data in cui viene eseguito il preventivo
-- di spesa
-- Descrizione riporta la cura relativa a un dato preventivo
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I pazienti che devono ancora pagare almeno il 50% dell'importo di un preventivo del 2012 o del 2013 e per cui almeno una seduta è stata con il dentista Molari



1.2) [2 p.] I nomi dei dentisti con almeno 5 anni di servizio che hanno eseguito cure di tipo 'pulizia canalare' con almeno 2 sedute nel 2013 (relative a uno stesso preventivo)



Sistemi Informativi T
23 gennaio 2014
Risoluzione

2) SQL (5 punti totali)

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

2.1) [2 p.] I nomi dei dentisti con almeno 5 anni di servizio che hanno eseguito cure di tipo 'pulizia canalare' con 3 o più sedute nel 2013, tutte relative a uno stesso preventivo

```
SELECT   D.Nome
FROM     DENTISTI D
WHERE    D.AnniServizio >= 5
AND      D.DID IN
         ( SELECT    S.DID
           FROM      PREVENTIVI P, SEDUTE S
           WHERE     P.PID = S.PID
           AND       P.Descrizione LIKE '%pulizia canalare%'
           AND       YEAR(S.Data) = 2013
           GROUP BY  S.DID,S.PID
           HAVING    COUNT(*) >= 3)
```

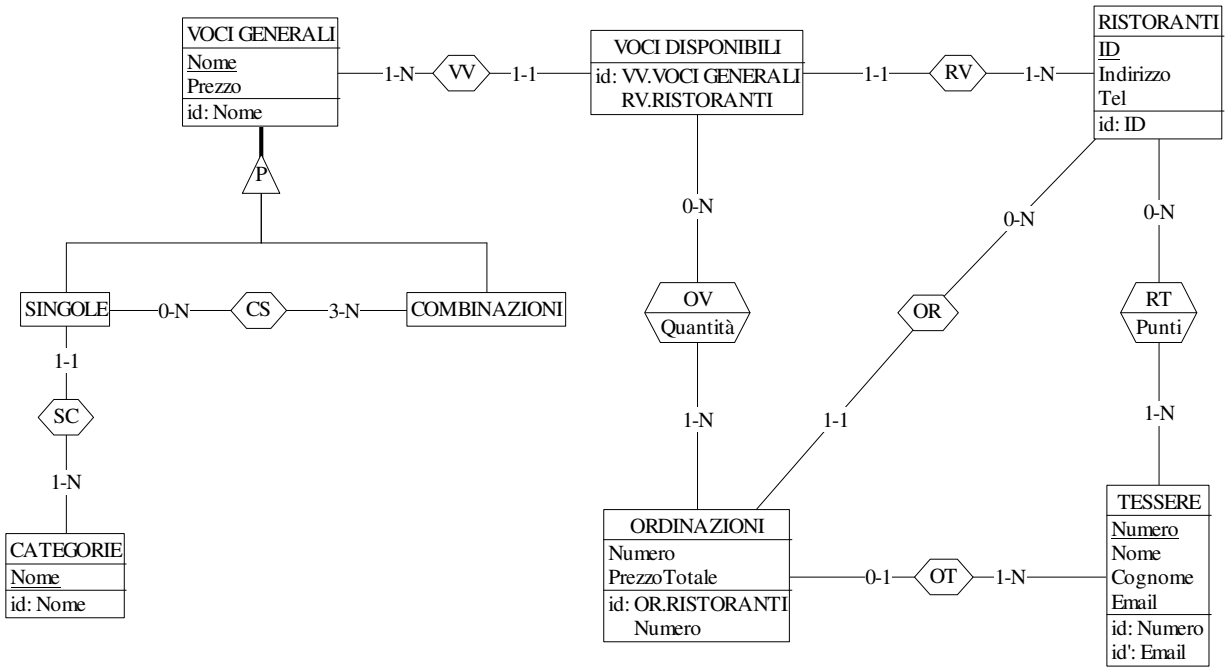
2.2) [3 p.] Per ogni preventivo completamente pagato e in cui tutte le sedute (almeno 2) sono state con uno stesso dentista, il numero di giorni trascorsi tra la prima e l'ultima seduta

```
SELECT   P.PID, DAYS(MAX(S.DATA)) - DAYS(MIN(S.DATA)) AS NGiorni
FROM     PREVENTIVI P, SEDUTE S
WHERE    P.PID = S.PID
AND      P.Importo = P.Pagato
GROUP BY P.PID
HAVING   COUNT(*) >= 2 AND COUNT(DISTINCT S.DID) = 1
```

Sistemi Informativi T
23 gennaio 2014
Risoluzione

3) Progettazione concettuale (6 punti)

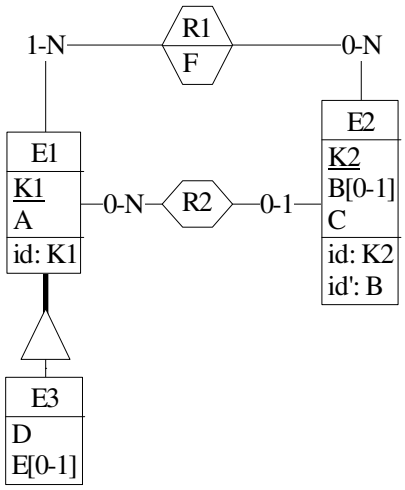
La catena BadRonalds (BR) dispone di numerosi ristoranti in cui offrire le proprie specialità culinarie. I prezzi di quanto offerto non variano da un ristorante all'altro, ma non tutti i piatti e bevande sono disponibili in tutti i ristoranti. Oltre alle singole voci di listino, ognuna relativa a una determinata categoria (antipasti, primi, bevande, ecc.), vi sono anche dei "menù combinazione", che si caratterizzano per il nome (che li identifica) e le single voci offerte (almeno 3): ad esempio, il menù "Giardino" prevede spaghetti al pesto di basilico, insalata mista, frutta e 1/2 litro di acqua. Ovviamente, nemmeno tutti i menù combinazione sono disponibili in tutti i ristoranti.
Oltre alle ordinazioni dei clienti occasionali (di tali clienti ovviamente non è nota l'identità) il DB della BR gestisce anche gli ordini dei clienti con tessera BR. Le tessere sono identificate da un numero e sono nominative, con nome, cognome ed email del cliente (un indirizzo di email è utilizzabile per una sola tessera). In base a quanto ordinato in un ristorante, una tessera viene caricata di punti, validi solo per quel ristorante (i punti servono a ricevere offerte speciali non gestite dal DB).



Commenti:

- Nella soluzione proposta si è optato per una materializzazione di VOCI DISPONIBILI, in modo che ogni ordinazione faccia riferimento solo a voci effettivamente presenti in un ristorante. Non è tuttavia esprimibile il vincolo che le istanze di VOCI DISPONIBILI presenti in un'ordinazione siano tutte del ristorante referenziato da OR (si noti che, a rigore, tale associazione è ridondante, ma è parso opportuno inserirla per maggior chiarezza).

Sistemi Informativi T
23 gennaio 2014
Risoluzione



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 R2 non viene tradotta separatamente;
c) le entità E1 ed E3 vengono tradotte insieme;
d) il valore di C non può essere superiore a quello di A dell'istanza di E1 cui l'istanza di E2 è eventualmente associata;

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,
TIPO SMALLINT NOT NULL CHECK (TIPO IN (1,3)),          -- 3: istanza anche di E3
D INT,
E INT,
CONSTRAINT E3 CHECK ((TIPO = 1 AND D IS NULL AND E IS NULL) OR
                     (TIPO = 3 AND D IS NOT NULL))                    );

CREATE TABLE E2 (
K2 INT NOT NULL PRIMARY KEY,
B INT,
C INT NOT NULL,
K1R2 INT REFERENCES E1            );

CREATE TABLE R1 (
K1 INT NOT NULL REFERENCES E1,
K2 INT NOT NULL REFERENCES E2,
F INT NOT NULL,
PRIMARY KEY (K1,K2)               );
```

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'unicita' dei valori di B
CREATE TRIGGER UNIQUE_B
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (     SELECT * FROM E2
                   WHERE N.B = E2.B  ) )
SIGNAL SQLSTATE '70001' ('L''attributo B non ammette valori duplicati!');

-- Trigger che garantire il rispetto del vincolo di cui al punto d)
CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (     SELECT * FROM E1
                   WHERE E1.K1 = N.K1R2
                   AND E1.A < N.C  ) )
SIGNAL SQLSTATE '70002' ('La tupla inserita referenzia una tupla di E1 con valore A < C!');

-- Poiche' min-card(E1,R1) = 1, ogni volta che si inserisce una tupla in E1 bisogna anche eseguire, nella stessa
-- transazione, almeno un inserimento in R1
```