Sistemi Informativi T
21 gennaio 2015
Risoluzione

Tempo a disposizione: 2:30 ore

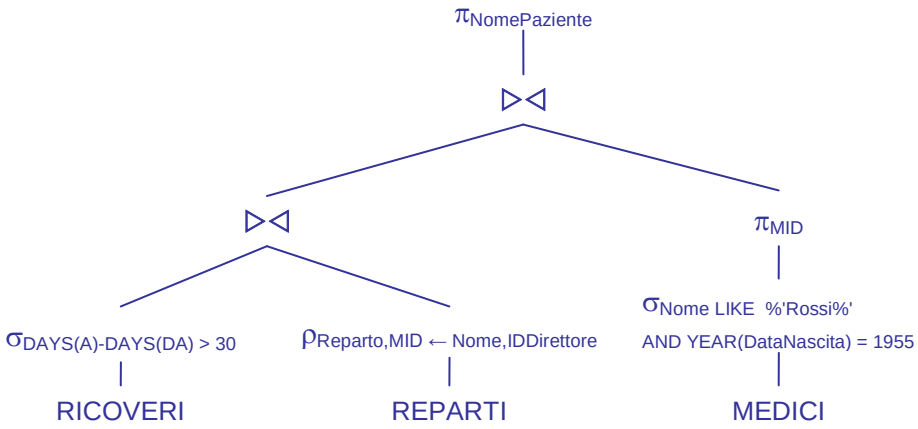
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

```
MEDICI(MID,Nome,DataNascita,DataAssunzione,Reparto),
      Reparto REFERENCES REPARTI;
REPARTI(Nome,IDDirettore*),
      IDDirettore REFERENCES MEDICI;
RICOVERI(RID,NomePaziente,Motivo,Reparto,Da,A*),
      Reparto REFERENCES REPARTI;
-- A ha valore nullo per i pazienti ancora ricoverati
-- Ogni Direttore lavora nel reparto che dirige
-- Un reparto può anche non avere un direttore
```

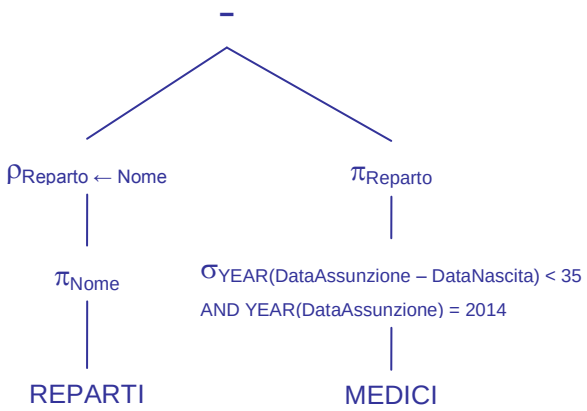
si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I pazienti che sono stati ricoverati più di 30 giorni nel reparto diretto dal dottor Rossi (quello nato nel 1955) e il cui ricovero è terminato



Se il ricovero non è terminato, la selezione su RICOVERI scarta la tupla

1.2) [2 p.] I nomi dei reparti che nel 2014 non hanno assunto nessun medico con meno di 35 anni (alla data dell'assunzione)



Sistemi Informativi T
21 gennaio 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.] I nomi dei reparti che nel 2014 non hanno assunto nessun medico con meno di 35 anni (alla data dell'assunzione)

```
SELECT   R.Nome
FROM     REPARTI R
WHERE    NOT EXISTS
         ( SELECT    *
           FROM      MEDICI M
           WHERE     M.Reparto = R.Nome
           AND       YEAR(M.DataAssunzione) = 2014
           AND       YEAR(M.DataAssunzione – M.DataNascita) < 35 )
```

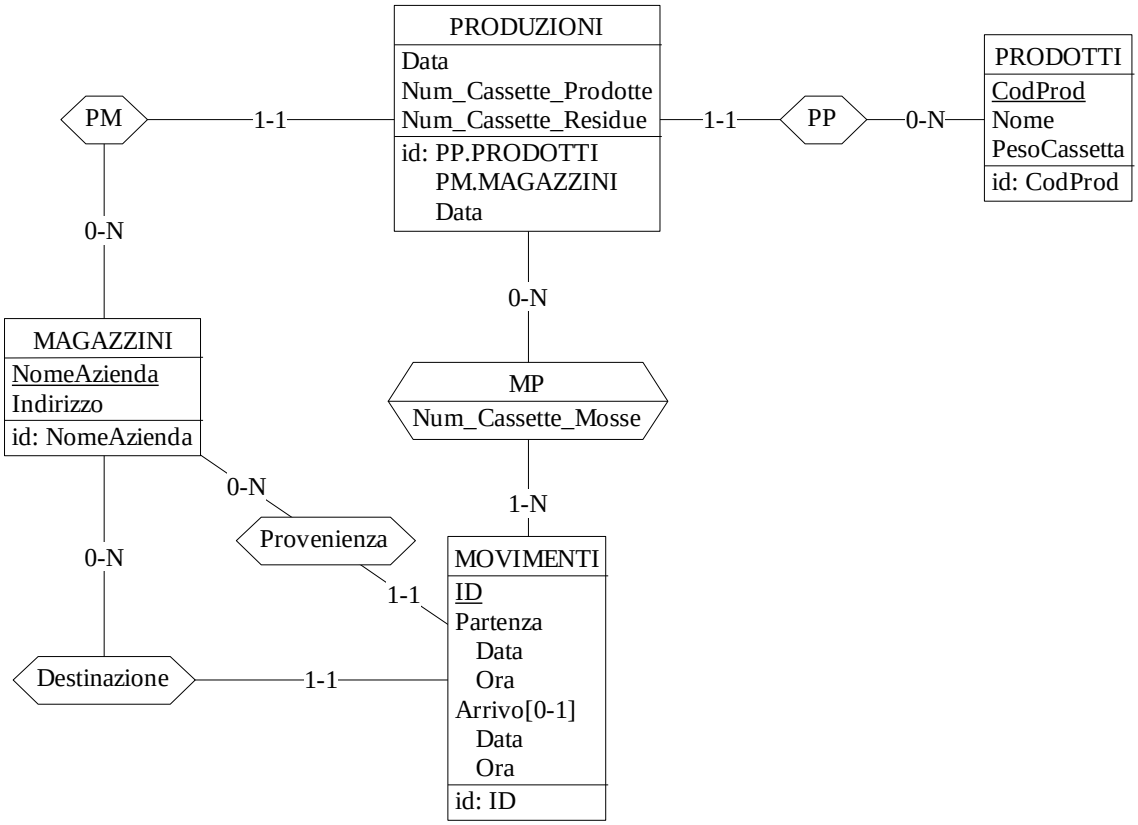
2.2) [3 p.] Per ogni fascia di età (20-29, 30-39, ecc.), il reparto con il maggior numero di medici, escludendo dal conteggio i direttori

```
WITH FasceReparti(Fascia, Reparto, NumMedici) AS (
   SELECT YEAR(CURRENT DATE - M.DataNascita)/10, M.Reparto, COUNT(*)
   FROM   REPARTI R, MEDICI M
   WHERE  M.Reparto = R.Nome
   AND    M.MID NOT IN (   SELECT R1.IDDirettore
                           FROM   REPARTI R1
                           WHERE  R1.IDDirettore IS NOT NULL )
   GROUP BY YEAR(CURRENT DATE – M.DataNascita)/10, M.reparto   )

SELECT   F1.*
FROM     FasceReparti F1
WHERE    F1.NumMedici >= ALL ( SELECT F2.NumMedici
                               FROM   FasceReparti F2
                               WHERE  F2.Fascia = F1.Fascia )


Per formattare l'output nella forma 20-29, 30-39, ecc.:

10*(YEAR(CURRENT DATE - M.DataNascita)/10) CONCAT '-' CONCAT
10*(YEAR(CURRENT DATE - M.DataNascita)/10)+9
```

Sistemi Informativi T
21 gennaio 2015
Risoluzione

3) Progettazione concettuale (6 punti)

Il consorzio di agricoltori eCampi riunisce diverse aziende agricole.
Ogni azienda gestisce un proprio magazzino, in cui mantiene la frutta e verdura disponibile. Per ogni prodotto viene registrato il numero di cassette prodotte e quante ve ne sono ancora in magazzino, distinguendo sulla base delle diverse date di produzione (ad es. l'azienda Gregoretti ha prodotto 500 cassette di bietole il 12/12/2014 e in magazzino ce ne sono ancora 250, per la produzione del 10/01/2015 i numeri sono 200 e 150, rispettivamente). La dimensione di una cassetta (peso medio) è fissa per un dato prodotto (ad es. il peso medio di una cassetta di bietole è di 30 kg).
Per ottimizzare le produzioni, la gestione dei magazzini e la rete di vendita, le aziende della eCampi ricorrono spesso a movimenti di prodotti da un magazzino all'altro. Ogni movimento riguarda di norma diversi prodotti, e per esso vengono registrati data e ora di partenza e di arrivo (le seconde solo a consegna avvenuta), e per ogni prodotto inviato il numero di cassette (con relative date di produzione).
Il sistema informativo di eCampi mantiene solo la situazione attuale delle giacenze nei magazzini, ma tiene traccia di tutte le produzioni e dei movimenti eseguiti, senza limiti di tempo.



Commenti:

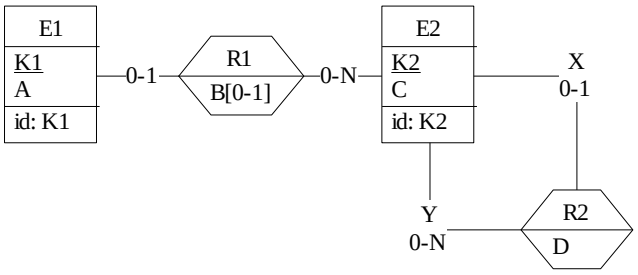
- L'esercizio si risolve agevolmente mediante l'introduzione dell'entità PRODUZIONI, ottenuta reificando l'associazione naturale tra MAGAZZINI e PRODOTTI.
- Se la giacenza di una produzione si azzera, la corrispondente istanza rimane in PRODUZIONI, per tener traccia, come richiesto, delle informazioni storiche sulle produzioni e i movimenti.
- Tutte le produzioni di un dato movimento devono essere dello stesso magazzino (vincolo non esprimibile in E/R), e i magazzini di Provenienza e Destinazione devono essere diversi (altro vincolo non esprimibile).
- Si noti che l'associazione Provenienza non è strettamente necessaria, ma viene introdotta per maggior chiarezza concettuale.
- Le cardinalità minime poste a 0 per MAGAZZINI in PM e PRODOTTI in PP servono a soli scopi di inizializzazione (nuovo magazzino e nuovo prodotto, rispettivamente).

Sistemi Informativi T
21 gennaio 2015
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 associazioni R1 e R2 non vengono tradotte separatamente;
c) il valore di B è sempre definito se la relativa istanza di E1 è associata a un'istanza di E2 che partecipa all'associazione R2 con ruolo X;
d) Per ogni istanza di E2, il valore di C è almeno pari alla somma dei valori di A delle istanze di E1 a essa associate;



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 E2 (
K2 INT NOT NULL PRIMARY KEY,
C INT NOT NULL,
D INT,
K2Y INT REFERENCES E2,              -- referenzia un'istanza che partecipa con ruolo Y
CONSTRAINT R2 CHECK ((D IS NULL AND K2Y IS NULL) OR
                     (D IS NOT NULL AND K2Y IS NOT NULL))                    );

CREATE TABLE E1 (
K1 INT NOT NULL PRIMARY KEY,
A INT NOT NULL,
B INT,
K2R1 INT REFERENCES E2,
CONSTRAINT R1 CHECK (K2R1 IS NOT NULL OR B IS NULL) -- se K2R1 è NULL lo è anche B  );
```

4.2) [3 p.] Per i vincoli non esprimibili a livello di schema si predispongano opportuni **trigger che evitino inserimenti di singole tuple non corrette**, definiti in un file TRIGGER.txt e usando se necessario il simbolo '@' per terminare gli statement SQL (altrimenti ';')

```
-- 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 (B IS NULL AND EXISTS (    SELECT * FROM E2
                                WHERE E2.K2 = N.K2R1
                                AND E2.K2Y IS NOT NULL  ) )
SIGNAL SQLSTATE '70001' ('La tupla inserita deve avere il valore di B definito!');

-- Il vincolo di cui al punto d) può essere violato inserendo una tupla in E1 che partecipa a R1 e il cui valore di A
-- è troppo grande (sommato a quelli già presenti è maggiore di E2.C):
CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (    SELECT * FROM E2 -- se ritorna 1 tupla allora N.A è troppo grande
                  WHERE E2.K2 = N.K2R1
                  AND (   E2.C < N.A + (   SELECT SUM(E1.A)
                                           FROM E1
                                           WHERE E1.K2R1 = E2.K2 )
                       OR E2.C < N.A )  )                                  )
SIGNAL SQLSTATE '70002' ('Il valore di A e" troppo grande!');
-- La condizione OR E2.C < N.A gestisce il caso in cui la subquery ( SELECT SUM(E1.A) … ) non
-- trova nessuna tupla in E1, ovvero la tupla che si sta inserendo è la prima che referenzia E2.K2.
-- In questo caso SUM(E1.A) è NULL, e cosi N.A + SUM(E1.A), e pertanto il test E2.C < N.A + SUM(E1.A)
-- restituirebbe UNKNOWN, non evidenziando la violazione del vincolo.
```