Sistemi Informativi T
11 giugno 2012
Risoluzione

Tempo a disposizione: 2:30 ore

1) Algebra relazionale (3 punti totali):

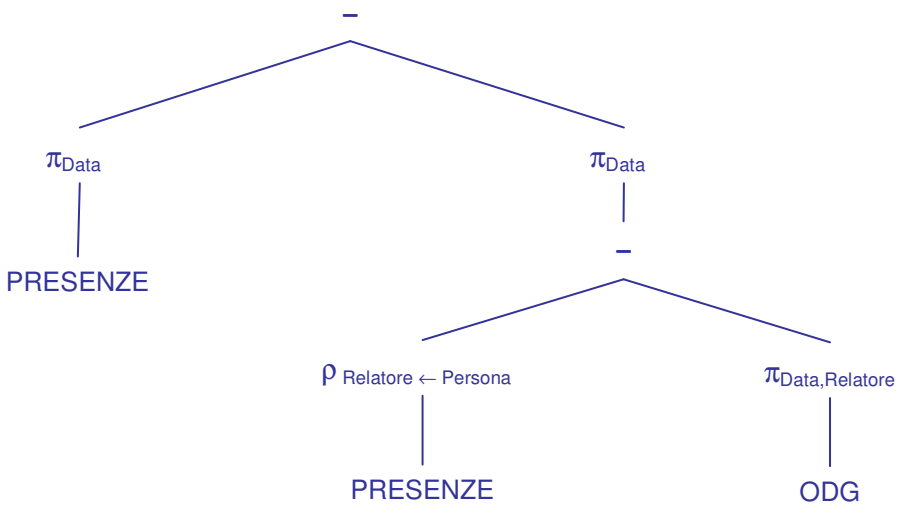
Date le seguenti relazioni:

```
RIUNIONI(Data,Presidente);
PRESENZE(Data,Persona),
Data REFERENCES RIUNIONI;
ODG(Data,Punto,Titolo,Argomento,Relatore),
Data,Relatore REFERENCES PRESENZE;
-- ODG = Ordine Del Giorno
-- Presidente, Persona e Relatore sono definiti sullo stesso dominio
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] Le persone che hanno relazionato sull'argomento 'tempo libero' nel 2012

1.2) [2 p.] Le date delle riunioni in cui tutti i presenti hanno svolto la funzione di relatore



La differenza sulla destra calcola le date in cui c'è almeno un presente che non svolge la funzione di relatore.

Sistemi Informativi T
11 giugno 2012
Risoluzione

2) SQL (5 punti totali)

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

2.1) [2 p.] Le date delle riunioni in cui tutti i presenti hanno svolto la funzione di relatore

```
SELECT   P.Data
FROM     PRESENZE P
GROUP BY P.Data
HAVING   COUNT(P.Persona) = ( SELECT   COUNT(DISTINCT O.Relatore)
                              FROM     ODG O
                              WHERE    O.Data = P.Data )
```

2.2) [3 p.] Per ogni relatore, l'argomento da lui complessivamente più trattato quando era anche presidente della riunione

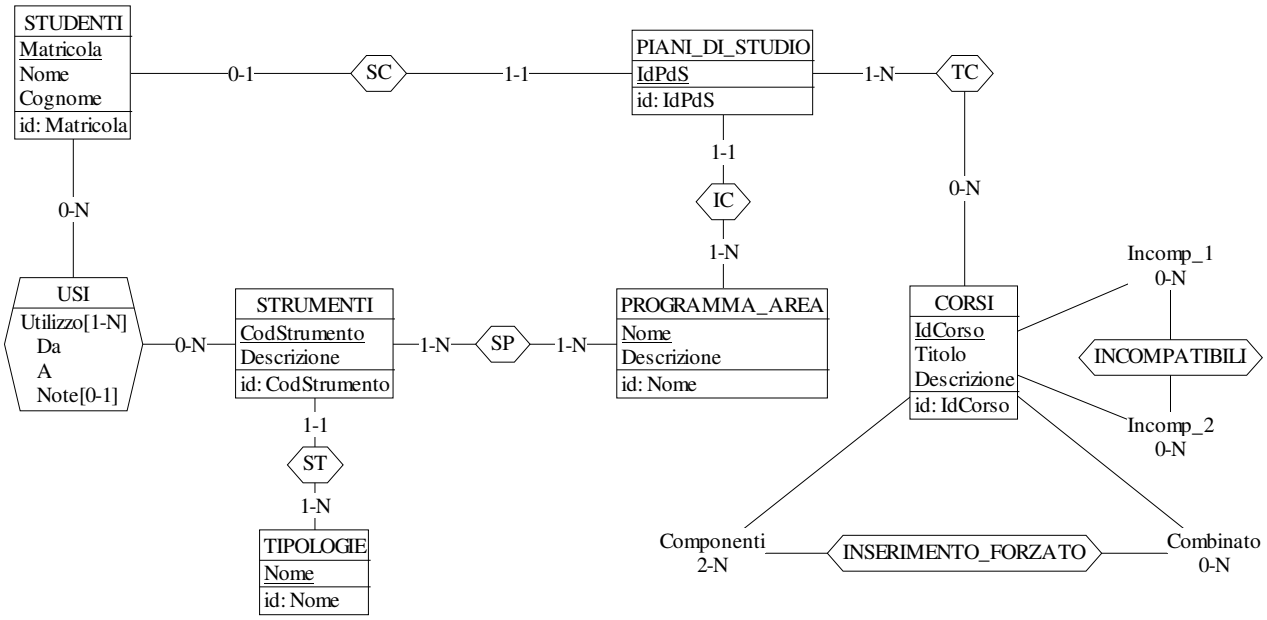
```
WITH ARGREL (Relatore, Argomento, NumVolte) AS (
   SELECT O.Relatore,O.Argomento, COUNT(*)
   FROM   ODG O, RIUNIONI R
   WHERE  O.Relatore = R.Presidente
   AND    O.Data = R.Data
   GROUP BY  O.Relatore,O.Argomento )
SELECT A.Relatore, A.Argomento
FROM   ARGREL A
WHERE  A.NumVolte >= ALL ( SELECT A1.NumVolte
                           FROM   ARGREL A1
                           WHERE  A1.Relatore = A.Relatore )

-- Nella Common Table Expression si contano le volte in cui un dato relatore
-- ha relazionato su un dato argomento quando era presidente
```

Sistemi Informativi T
11 giugno 2012
Risoluzione

3) Progettazione concettuale (6 punti)

La e-MU (electronic Modern University) è un'università all'avanguardia, in cui non esistono Corsi di Laurea, ma ogni studente può scegliere liberamente i corsi del suo piano di studi, rispettando però due vincoli: 1) un piano di studi non può contenere due corsi etichettati come "incompatibili"; 2) se il piano di studi include un certo insieme di corsi, ciò comporta l'inserimento anche di un corso "combinato" ad essi associato.
In funzione del piano di studi prescelto, ogni studente viene inserito in un "programma d'area", che gli/le fornisce gli strumenti necessari allo svolgimento degli studi. Gli strumenti sono di varie tipologie (di calcolo, di scavo, chirurgici, ecc.). Per ogni strumento fornito, si tiene traccia dei periodi in cui ogni studente lo ha utilizzato e di eventuali note relative all'utilizzo (es. il PC 23145 è andato in crash).



Commenti:

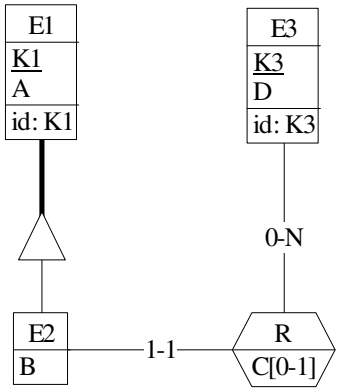
- L'esercizio non presenta difficoltà di rilievo, una volta che si nota che i due vincoli legati alla formazione dei piani di studio richiedono l'introduzione di due associazioni (INCOMPATIBILI e INSERIMENTO_FORZATO, rispettivamente). La garanzia che un piano di studi rispetti effettivamente tali vincoli non è comunque esprimibile mediante il modello E/R
- Vale inoltre il vincolo (non rappresentabile nello schema) che uno studente può utlizzare solo strumenti della propria area
- E' possibile definire il subset degli STUDENTI che hanno già presentato il piano di studi

Sistemi Informativi T
11 giugno 2012
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 vengono tradotte assieme;
d) per le solo istanze di E2, esiste una dipendenza funzionale da A a B;



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 E1 (
K1 INT NOT NULL PRIMARY KEY,
A INT NOT NULL,
TIPO2 SMALLINT NOT NULL CHECK (TIPO2 IN (0,1)),        -- 1: istanza anche di E2
B INT,
K3 INT REFERENCES E3,
C INT,
CONSTRAINT E2 CHECK
     ( (TIPO2 = 1 AND B IS NOT NULL AND K3 IS NOT NULL) OR
      (TIPO2 = 0 AND B IS NULL AND K3 IS NULL AND C IS 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 il simbolo '@' per terminare gli statement SQL

```
-- Per garantire il rispetto del vincolo di cui al punto d) è necessario impostare il seguente trigger:
CREATE TRIGGER FD_A_B
NO CASCADE BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.TIPO2 = 1 AND EXISTS (   SELECT *
                                 FROM E1
                                 WHERE E1.A = N.A
                                 AND      E1.B <> N.B) )
SIGNAL SQLSTATE '70001' ('La tupla non rispetta la FD da A a B!')@
-- La condizione sul tipo e' inserita solo per chiarezza, in quanto la seconda condizione EXISTS (...) non puo'
-- mai essere vera se TIPO = 0, in quanto N.B è NULL e l'argomento dell' EXISTS non contiene nessuna tupla
```