Sistemi Informativi T
21 febbraio 2018
Risoluzione

Tempo a disposizione: 2:30 ore

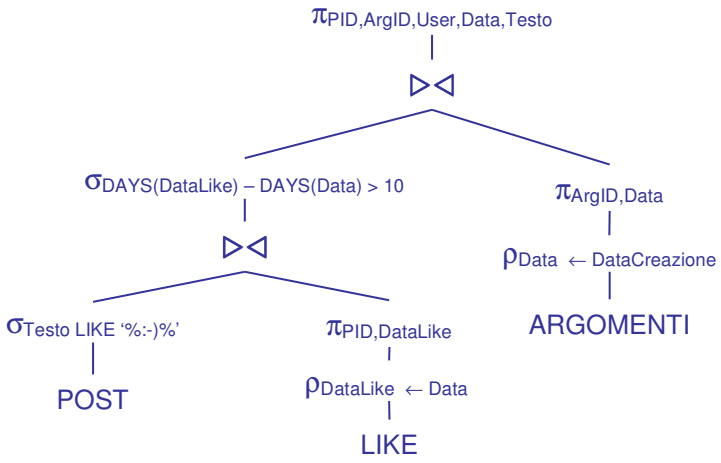
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

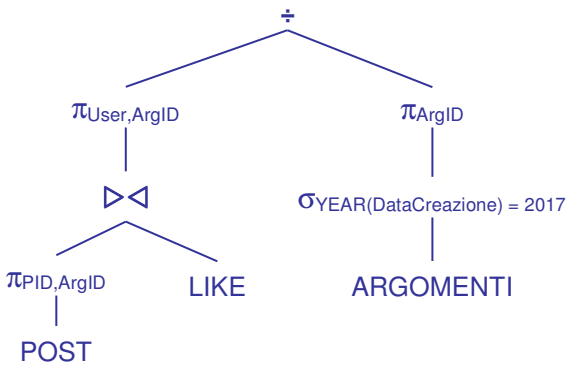
```
ARGOMENTI(ArgID,Titolo,DataCreazione,CreatoDa);
POST(PID,ArgID,User,Data,Testo),
     ArgID REFERENCES ARGOMENTI;
LIKE(User,PID,Data),
     PID REFERENCES POST;
--
-- ARGOMENTI.CreatoDa è l'utente che ha creato l'argomento
-- POST.User è l'utente che pubblica il post
-- LIKE.User è l'utente che ha messo un like a un post
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I dati dei post pubblicati il giorno stesso della creazione dell'argomento, che nel testo hanno uno smile, ':-)', e che hanno ricevuto un like anche a distanza di più di 10 giorni



1.2) [2 p.] Gli utenti che hanno messo un like ad almeno un post di ogni argomento creato nel 2017



Sistemi Informativi T
21 febbraio 2018
Risoluzione

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 che ha creato almeno un argomento, il numero di like che ha messo a post di argomenti diversi da quelli da lui creati

```
SELECT   L.User AS Utente, COUNT(*) AS NumLike
FROM     ARGOMENTI A, POST P, LIKE L
WHERE    L.PID = P.PID
AND      P.ArgID = A.ArgID
AND      A.CreatoDa <> L.user
AND      L.User IN (SELECT CreatoDa FROM Argomenti)
GROUP BY L.User
```

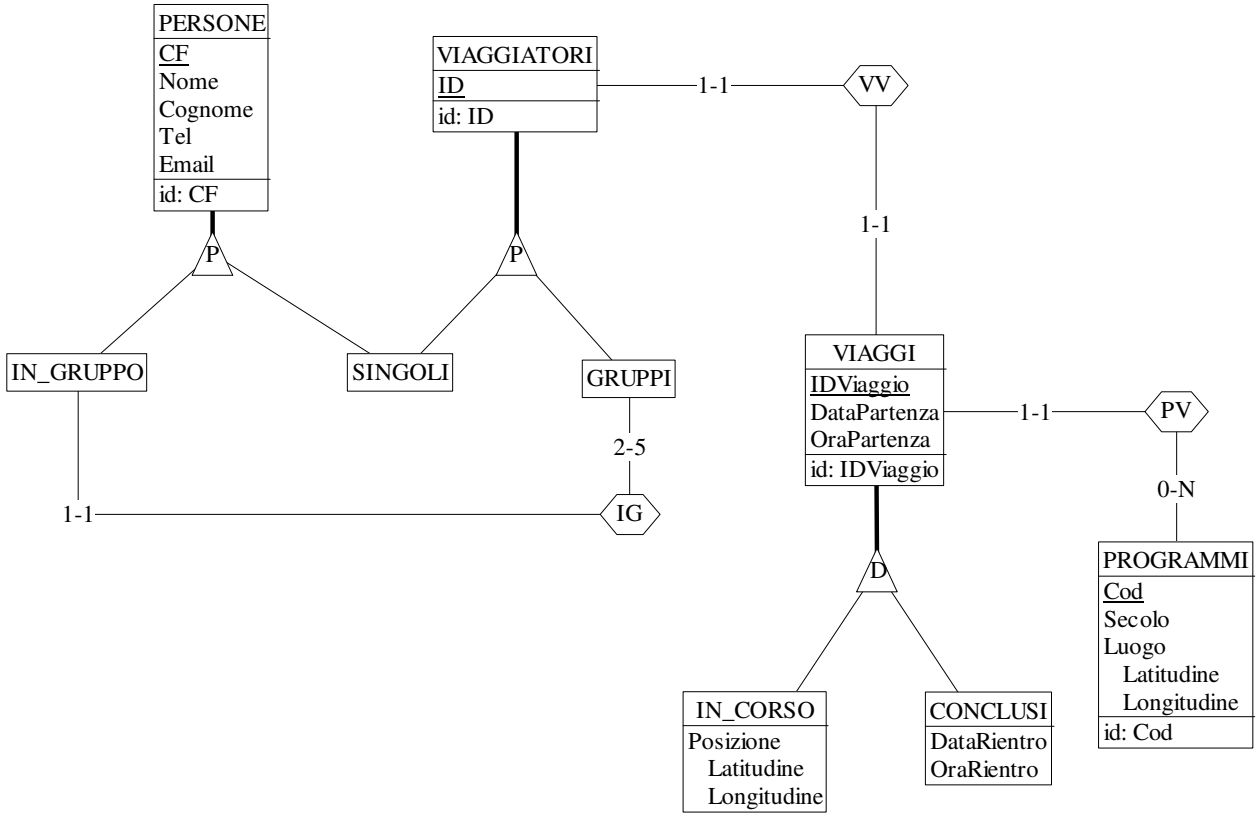
2.2) **[3 p.]** I dati dell'argomento con il numero massimo di utenti distinti che hanno pubblicato uno o più post solo su quell'argomento

```
WITH
NUM_USERS (Argomento, NumUtenti) AS (
   SELECT P.ArgID, COUNT(DISTINCT P.User)
   FROM   POST P
   WHERE  P.User NOT IN (  SELECT P1.User
                           FROM   POST P1
                           WHERE  P1.ArgID <> P.ArgID )
   GROUP BY P.ArgID                                      )

SELECT   A.*,NU.NumUtenti
FROM     NUM_USERS NU, ARGOMENTI A
WHERE    A.ArgID = NU.Argomento
AND      NU.NumUtenti = (  SELECT MAX(NU1.NumUtenti)
                           FROM   NUM_USERS NU1     )

-- La c.t.e. calcola il numero di utenti distinti che hanno pubblicato post
-- su un dato argomento, non considerando, grazie alla subquery, quelli
-- che hanno pubblicato anche su altri argomenti
.
```

Sistemi Informativi T
21 febbraio 2018
Risoluzione

3) Progettazione concettuale (6 punti)

La Touristic Time Travels (T3) è una società che organizza viaggi nel tempo (passato) per singoli o gruppi (di non più di 5 persone). I programmi di viaggio tra cui è possibile scegliere specificano il secolo prescelto e il luogo, quest'ultimo espresso in coordinate spaziali (latitudine e longitudine). Di ogni viaggio la T3 mantiene informazioni anagrafiche sui viaggiatori e sulla data e ora di partenza. Per i viaggi conclusi si mantiene la data e ora di rientro, mentre per quelli in corso si tiene aggiornata la posizione attuale dei viaggiatori. Ogni viaggio è svolto o da una singola persona o da un gruppo.
Per motivi di sicurezza dovuti a possibili effetti collaterali, ogni persona può fare al più un viaggio con la T3.



Commenti:

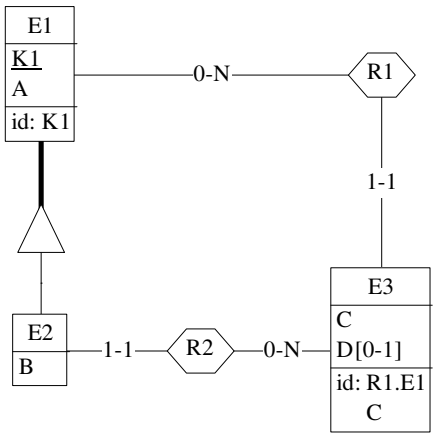
- La principale difficoltà dell'esercizio risiede nella corretta modellazione dei soggetti che viaggiano, ovvero dei VIAGGIATORI. Tale entità va tenuta distinta da quella che rappresenta le PERSONE fisiche, in quanto un viaggio può essere compiuto anche da un gruppo (che non è una persona). Senza l'entità VIAGGIATORI il vincolo che un viaggio è fatto da una persona singola o da un gruppo non è esprimibile.
- Si è posto min-card(VIAGGIATORI,VV) = 1, ipotizzando che la registrazione di gruppi e utenti nel sistema avvenga quando viene scelto il viaggio
- Per gestire i VIAGGI non ancora iniziati la relativa gerarchia non è totale.

Sistemi Informativi T
21 febbraio 2018
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 assieme;
c) le associazioni R1 e R2 non vengono tradotte separatamente;
d) un'istanza di E2 non è mai associata a un'istanza di E3 identificata da un'altra istanza di E2 con valore B > 10;



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,
SEL12   SMALLINT NOT NULL CHECK (SEL12 IN (1,2)),      -- 2: istanza anche di E2
B       INT,
K1R2    INT,
CR2     INT,
CONSTRAINT E2 CHECK (   (SEL12 = 1 AND B IS NULL AND K1R2 IS NULL AND CR2 IS NULL) OR
                (SEL12 = 2 AND B IS NOT NULL AND K1R2 IS NOT NULL AND CR2 IS NOT NULL)  )    );

CREATE TABLE E3 (
K1      INT NOT NULL REFERENCES E1,
C       INT NOT NULL,
D       INT,
PRIMARY KEY (K1,C)                          );

ALTER TABLE E1
ADD CONSTRAINT FK_E3_R2 FOREIGN KEY (K1R2,CR2) REFERENCES E3 ;
```

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 al punto d)

CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (     SELECT *
                    FROM    E1
                    WHERE N.K1R2 = E1.K1
                    AND     E1.B > 10                   )   )
SIGNAL SQLSTATE '70001' ('La tupla è associata a una tupla di E3 identificata da una tupla di E2
con valore B>10!');

-- La subquery esistenziale si potrebbe anche scrivere
    SELECT    *
    FROM      E1, E3
    WHERE     N.K1R2 = E3.K1
    AND       N.C = E3.C
    AND       E3.K1 = E1.K1
    AND       E1.B > 10
-- Osservando che il valore di N.C non ha alcuna influenza nell'eventuale violazione del vincolo e
-- rimuovendo la seconda condizione, per transitività si ottiene la soluzione proposta
```