Sistemi Informativi T
18 febbraio 2019
Risoluzione

Tempo a disposizione: 2:30 ore

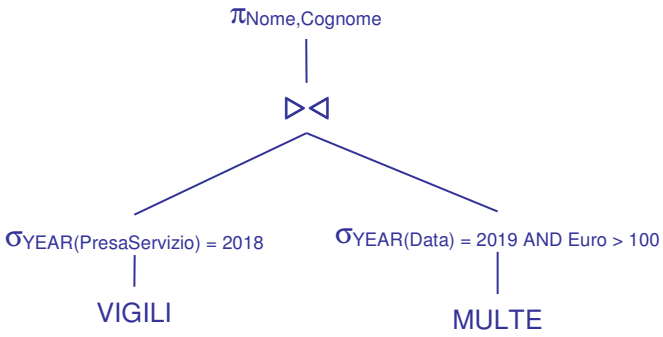
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

```
VIGILI(IDV,Nome,Cognome,PresaServizio);
MULTE(IDM,IDV,Targa,Data,Euro),
      IDV REFERENCES VIGILI;
--
-- PresaServizio è la data in cui un vigile ha preso servizio.
-- Euro è di tipo DEC(6,2) e rappresenta l'importo della multa.
```

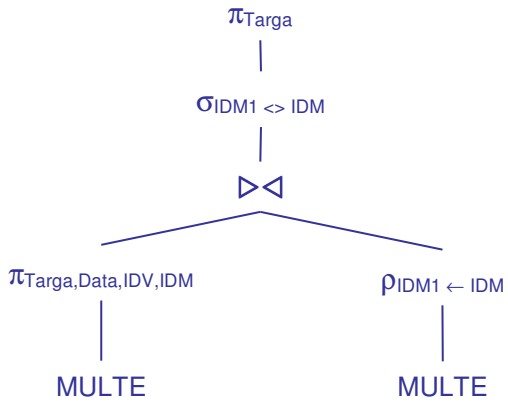
si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] Nome e cognome dei vigili che han preso servizio nel 2018 e nel 2019 hanno fatto almeno una multa di più di 100 Euro



NB: la formulazione della query non è compatibile con l'interpretazione in cui i vigili hanno preso servizio nel 2018 **o** 2019. Se così fosse si sarebbe scritto: "…nel 2018 e nel 2019 **e** hanno fatto almeno una multa di più di 100 Euro"

1.2) [2 p.] Le targhe delle auto che, nello stesso giorno, hanno preso più di una multa da uno stesso vigile



Sistemi Informativi T
18 febbraio 2019
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 vigile (fornendo IDV, nome e cognome), l'importo medio delle multe fatte nei primi 60 giorni dalla presa di servizio, ordinando il risultato per valori decrescenti della media

```
SELECT   V.IDV,V.Nome,V.Cognome, DEC(AVG(M.Euro),6,2) AS ImportoMedio
FROM     VIGILI V, MULTE M
WHERE    V.IDV = M.IDV
AND      DAYS(M.Data) - DAYS(V.PresaServizio) <= 60
GROUP BY V.IDV,V.Nome,V.Cognome
ORDER BY ImportoMedio DESC
```

2.2) **[3 p.]** Per ogni fascia di prezzo (meno di 50€, da 50€ ma meno di 100€, ecc.) il numero di giorni (date) in cui il maggior numero di multe è stato in quella fascia

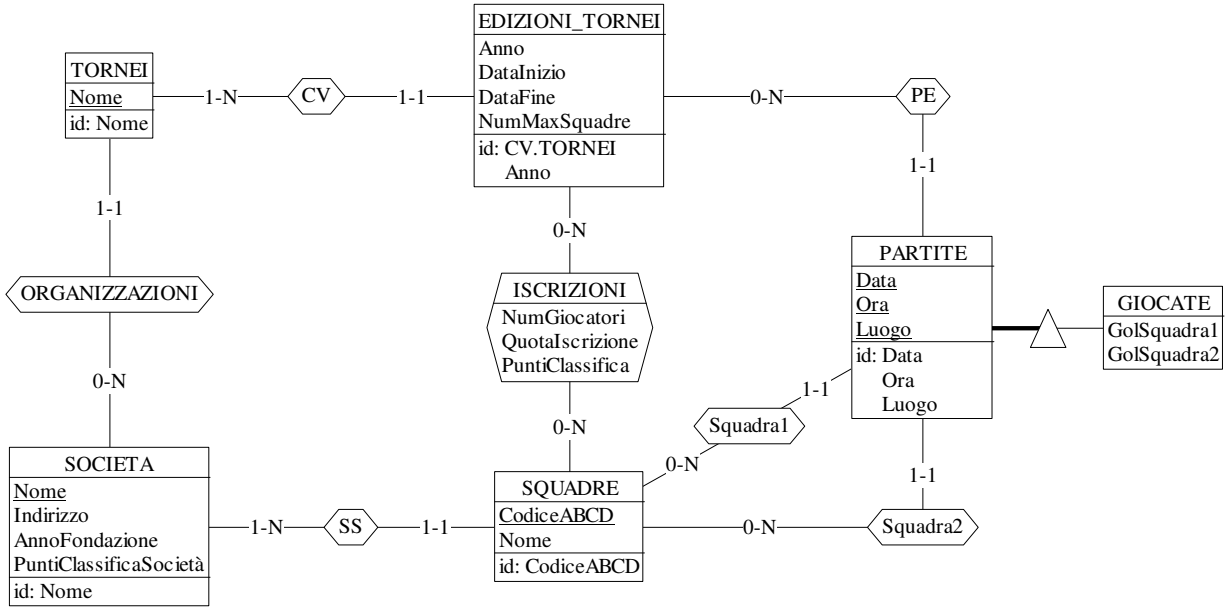
```
WITH
NumMulte(Data,Fascia,Num) AS
(  SELECT Data, INT(Euro)/50, COUNT(*)
   FROM   MULTE
   GROUP BY Data, INT(Euro)/50 ),
FasciaPiuFrequente(Data,Fascia) AS
(  SELECT M.Data,M.Fascia
   FROM   NumMulte M
   WHERE  M.Num >= ALL
       (  SELECT M1.Num
          FROM   NumMulte M1
          WHERE  M1.Data = M.Data ) )
SELECT    (50*Fascia) CONCAT '-' CONCAT (50*(Fascia+1)-1) AS Fascia,
          COUNT(*) AS NumGiorni
FROM      FasciaPiuFrequente
GROUP BY  Fascia

-- La soluzione ricalca quella della query Q5 dell'esercitazione n.4 proposta
-- in laboratorio: la prima c.t.e. determina per ogni fascia e giorno il
-- numero di multe, mentre la seconda c.t.e. determina per ogni giorno la
-- fascia più frequente. Ovviamente è fondamentale eseguire il cast di Euro.
```

Sistemi Informativi T
18 febbraio 2019
Risoluzione

3) Progettazione concettuale (6 punti)

L'Associazione Bolognese Calcio Dilettantistico (ABCD) riunisce diverse società sportive che organizzano annualmente dei tornei, cui possono prendere parte le squadre delle società. Ogni torneo ha un nome (univoco) ed è organizzato tutti gli anni sempre dalla stessa società (una società ha un nome, un indirizzo e un anno di fondazione). Il numero massimo di squadre che si possono iscrivere a un dato torneo varia di anno in anno, così come le date di inizio e fine. Ogni società ha una o più squadre, caratterizzate da un codice ABCD e dal nome. Ogni squadra può iscriversi a tutti i tornei che vuole, pagando una quota che dipende dal numero di giocatori partecipanti in quell'anno. Il database dell'ABCD mantiene informazioni su tutte le partite disputate (data, ora, luogo e risultato finale) e su quelle ancora da disputare (in questo caso ovviamente manca il risultato). Per ogni torneo viene mantenuta la classifica (per ogni squadra partecipante il relativo punteggio) e c'è anche una classifica generale delle società, che tiene conto dei risultati ottenuti dalle sue squadre in tutti i tornei.



Commenti:

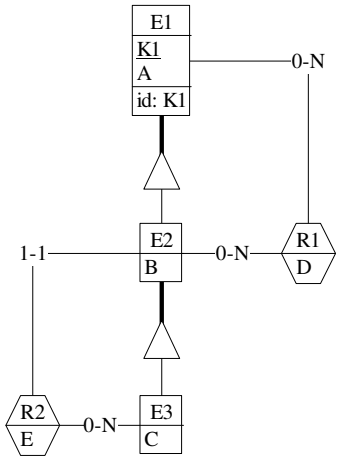
- Dalle specifiche segue chiaramente la distinzione concettuale tra TORNEI ed EDIZIONI_TORNEI. La modellazione corretta di questo pattern semplifica la rappresentazione di tutti gli altri concetti.
- Per PARTITE sono possibili anche altri criteri di identificazione (ad es. codice, identificazione esterna usando l'entità SQUADRE).

Sistemi Informativi T
18 febbraio 2019
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, E2 ed E3 vengono tradotte insieme;
c) l'associazione R2 non viene tradotta separatamente;
d) ogni istanza di E2 che partecipa all'associazione R1 (dal ramo collegato a E2) non è mai associata, tramite R2, a un'istanza di E3 tale che C + E > 20;



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 (o SCHEMI.sql)

```
CREATE TABLE E1 (
K1      INT NOT NULL PRIMARY KEY,
A       INT NOT NULL,
TIPO2   SMALLINT NOT NULL CHECK (TIPO2 IN (1,2)), -- 2 se istanza di E2, 1 altrimenti
B       INT,
K1R2    INT REFERENCES E1,
E       INT,
TIPO3   SMALLINT NOT NULL CHECK (TIPO3 IN (1,3)), -- 3 se istanza di E3, 1 altrimenti
C       INT,
CONSTRAINT E2 CHECK ((TIPO2 = 1 AND B IS NULL AND K1R2 IS NULL AND E IS NULL) OR
        (TIPO2 = 2 AND B IS NOT NULL AND K1R2 IS NOT NULL AND E IS NOT NULL)),
CONSTRAINT E3 CHECK ((TIPO3 = 1 AND C IS NULL) OR  (TIPO3 = 3 AND C IS NOT NULL)),
CONSTRAINT E2_E3 CHECK (TIPO3 = 1 OR TIPO2 = 2) ); -- esclude il caso TIPO3 = 3 AND TIPO2 = 1

CREATE TABLE R1 (
K1      INT NOT NULL REFERENCES E1,
K1E2    INT NOT NULL REFERENCES E1,
D       INT NOT NULL,
PRIMARY KEY (K1,K1E2)                   );
```

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 (o TRIGGER.sql) e usando se necessario il simbolo '@' per terminare gli statement SQL (altrimenti ';')

```
-- Trigger che garantisce che K1R2 referenzi un'istanza di E3
CREATE TRIGGER R2_E3
BEFORE INSERT ON E1
REFERENCING NEW AS N FOR EACH ROW
WHEN N.TIPO2 = 2 AND NOT EXISTS ( SELECT * FROM E1 WHERE N.K1R2 = K1 AND TIPO3 = 3 )
SIGNAL SQLSTATE '70001' ('La tupla referenziata deve appartenere a E3!');

-- Trigger che garantisce che K1E2 referenzi un'istanza di E2
CREATE TRIGGER R1_E2
BEFORE INSERT ON R1
REFERENCING NEW AS N FOR EACH ROW
WHEN ( NOT EXISTS ( SELECT * FROM E1 WHERE N.K1E2 = K1 AND TIPO2 = 2 )    )
SIGNAL SQLSTATE '70002' ('La tupla referenziata deve appartenere a E2!');

CREATE TRIGGER PUNTO_C
BEFORE INSERT ON R1                     -- unico caso in cui il vincolo può essere violato
REFERENCING NEW AS N FOR EACH ROW
WHEN ( 20 < ( SELECT E2.E +E3.C FROM E1 E2, E1 E3
                WHERE N.K1E2 = E2.K1 AND E2.K1R2 = E3.K1 )  )
SIGNAL SQLSTATE '70003' ('La tupla referenzia un'istanza di E2 non corretta!');
-- Il check E2.TIPO2 = 2 è ridondante, in quanto già garantito dal trigger R1_E2
```