Sistemi Informativi T
5 febbraio 2019
Risoluzione

Tempo a disposizione: 2:30 ore

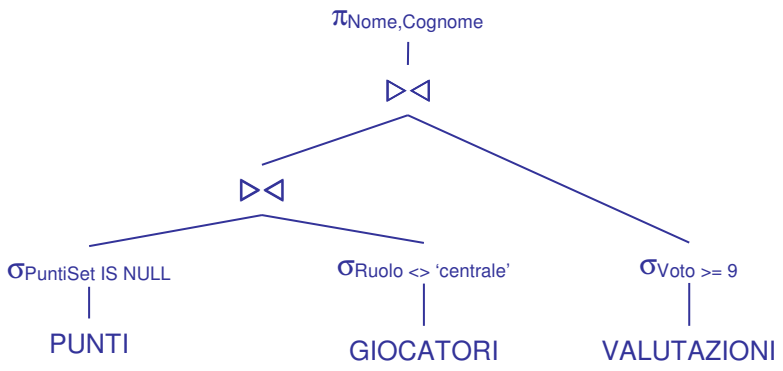
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

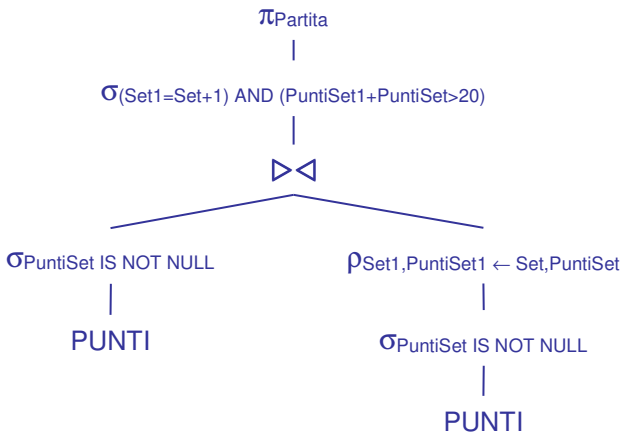
```
GIOCATORI(IDG,Nome,Cognome,DataNascita,Ruolo);
PUNTI(IDG,Partita,Set,PuntiSet*),
      IDG REFERENCES GIOCATORI;
VALUTAZIONI(IDG,Partita,Voto),
      IDG REFERENCES GIOCATORI;
--
-- Ogni Partita (di pallavolo) si compone di più Set (da 3 a 5).
-- PuntiSet è di tipo INT: è nullo se il giocatore in quel Set non ha
-- giocato, mentre vale 0 se ha giocato ma senza far punti.
-- Se un giocatore non prende parte a una partita, allora la relativa
-- coppia (IDG,Partita) non compare nel DB, altrimenti in VALUTAZIONI
-- Voto (di tipo DEC(4,2)) rappresenta la valutazione della prestazione.
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] Nome e cognome dei giocatori che non giocano nel ruolo di 'centrale' e che in almeno una partita in cui non hanno giocato tutti i set hanno ottenuto una valutazione maggiore o uguale a 9



1.2) [2 p.] Le partite in cui almeno un giocatore ha fatto complessivamente più di 20 punti in 2 set consecutivi, in entrambi dei quali ha giocato



Entrambe le condizioni PuntiSet IS NOT NULL sono essenziali per garantire che un giocatore abbia giocato in entrambi i set

Sistemi Informativi T
5 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.]** Le partite in cui almeno 2 giocatori hanno preso un voto maggiore di 7 e ognuno ha totalizzato complessivamente almeno 20 punti

```
WITH TOT_PUNTI (Partita, IDG) AS (
   SELECT   P.Partita, P.IDG
   FROM     PUNTI P
   GROUP BY P.Partita, P.IDG
   HAVING   SUM(P.PuntiSet) >= 20      )

SELECT   T.Partita
FROM     TOT_PUNTI T, VALUTAZIONI V
WHERE    (T.IDG,T.Partita) = (V.IDG,V.Partita)
AND      V.Voto > 7
GROUP BY T.Partita
HAVING   COUNT(*) >= 2

-- La c.t.e. restituisce, per ogni partita, i giocatori che han realizzato
-- almeno 20 punti
```

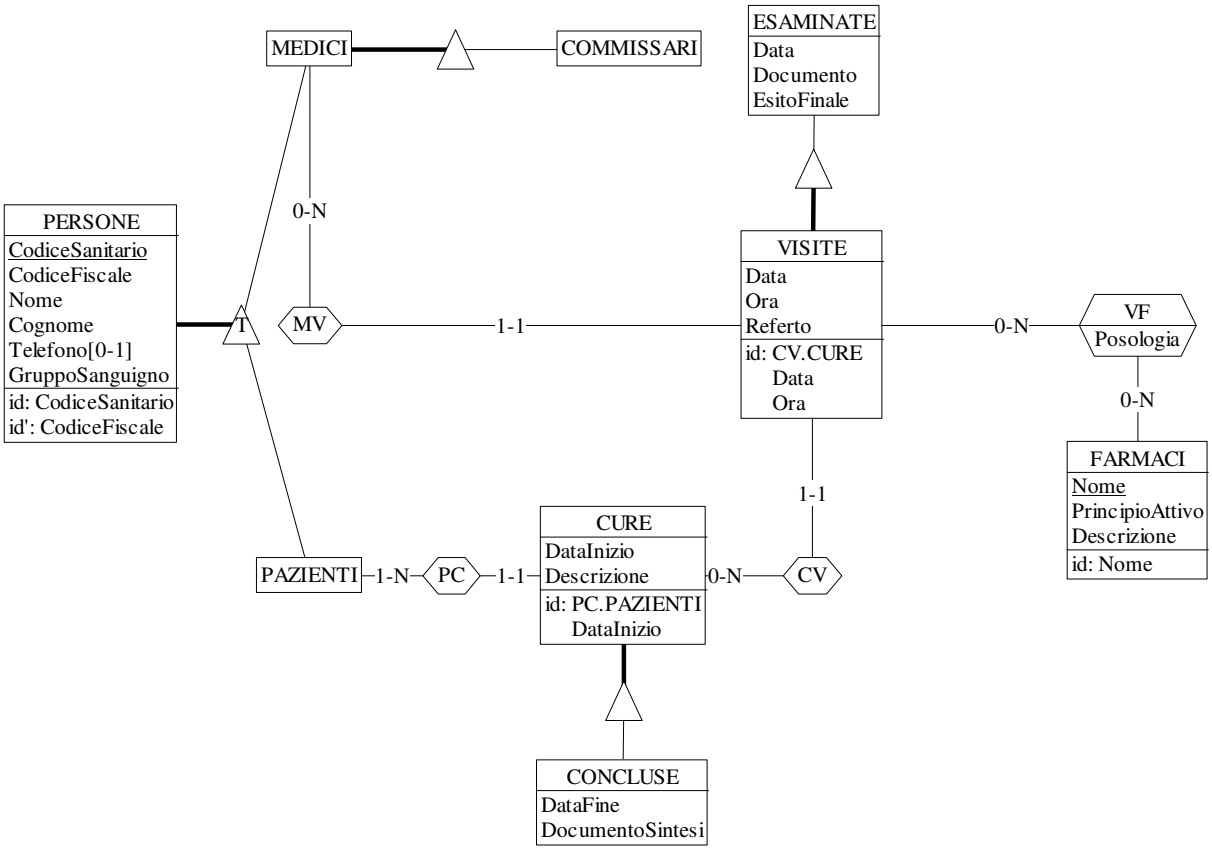
2.2) **[3 p.]** Per ogni partita, i giocatori che, escludendo gli eventuali set non giocati, hanno sempre incrementato i punti da un set all'altro

```
SELECT   P.Partita,P.IDG
FROM     PUNTI P
   EXCEPT
SELECT   P.Partita,P.IDG
FROM     PUNTI P
WHERE    EXISTS ( SELECT *
                  FROM   PUNTI P1
                  WHERE  (P1.Partita,P1.IDG) = (P.Partita,P.IDG)
                  AND    P1.Set > P.Set
                  AND    P1.PuntiSet <= P.PuntiSet       )

-- La soluzione proposta sottrae all'insieme delle possibili soluzioni quelle
-- coppie (Partita,IDG) per le quali esistono 2 set, P1.Set e P.Set, con
-- P1.Set che segue P.Set, in cui quel giocatore non ha fatto più punti nel
-- secondo rispetto al primo.

-- Si noti che la soluzione restitusice, per ogni partita, anche i giocatori
-- presenti alla partita che hanno giocato meno di 2 set. Per eliminarli è
-- sufficiente aggiungere al primo operando della differenza le clausole:
-- GROUP BY P.Partita,P.IDG
-- HAVING   COUNT(P.PuntiSet) > 1
```

Sistemi Informativi T
5 febbraio 2019
Risoluzione

3) Progettazione concettuale (6 punti)

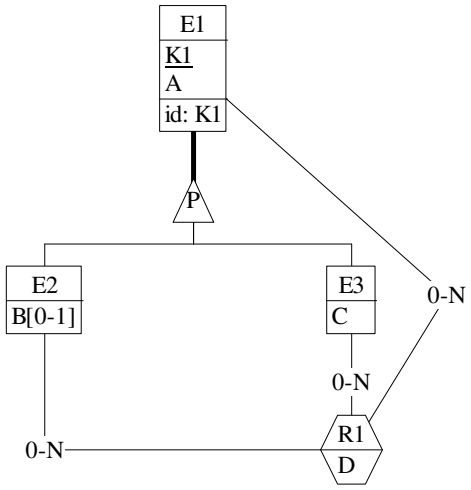
Il Servizio di qualità degli Ospedali Riuniti (OR) prevede dei controlli sulle modalità con cui vengono svolte le attività di cura dei propri pazienti. Di ogni paziente, identificato dal suo codice sanitario (e anche dal suo codice fiscale), OR mantiene nome, cognome, eventuale recapito telefonico e gruppo sanguigno. Per ogni paziente esiste una cartella clinica che riporta, per ogni cura specifica cui il paziente si sottopone, i dettagli delle visite: medico (identificato da codice sanitario e con le stesse informazioni dei pazienti), data, ora, referto ed eventuale ricetta per 1 o più farmaci. OR mantiene l'elenco completo dei farmaci prescrivibili (ogni farmaco ha un nome univoco, un principio attivo e una descrizione). In ogni ricetta, per ogni farmaco, il medico deve ovviamente indicare la posologia (es., 3 compresse al giorno dopo i pasti, per 1 settimana). Per le cure terminate viene redatto un documento di sintesi finale e registrata la data di fine cura. I medici degli OR si possono curare presso gli OR stessi.
Alcune delle visite eseguite presso gli OR vengono, casualmente, scelte ed esaminate da una commissione di controllo, composta da 5 medici degli OR. Per ogni controllo si riporta la data in cui viene eseguito e si redige un documento, oltre a un esito finale espresso come voto da 0 a 10.



Commenti:

- L'entità VISITE, ottenuta per reificazione, non ha come parte dell'identificatore il medico, in quanto un paziente in una certa data e ora non può avere più di una visita.
- Si è ipotizzato che un Paziente possa iniziare al massimo una cura in un dato giorno. In alternativa è possibile introdurre un IdCura.
- Si è anche ipotizzato che una visita sia relativa a una singola cura.
- Non è necessario introdurre entità per i concetti di cartella clinica e di ricetta.
- Nell'interpretazione data, la commissione di controllo è sempre la stessa, e questo implica che:
  - non è necessario associare COMMISSARI a ESAMINATE;
  - Il vincolo che la commissione sia formata da 5 commissari non è esprimibile.
- In un'interpretazione alternativa, parimenti compatibile con le specifiche, ogni visita controllata ha una specifica commissione. In questo caso va rimossa l'entità COMMISSARI e introdotta un'associazione, ME, tra MEDICI ed ESAMINATE con min-card(ESAMINATE,ME) = max-card(ESAMINATE,ME) = 5.

Sistemi Informativi T
5 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) le istanze di E2 con B nullo partecipano solo a istanze di R1 con D > 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,
TIPO    SMALLINT NOT NULL CHECK (TIPO IN (2,3)), -- 2 se istanza di E2, 3 se istanza di E3
B       INT,
C       INT,
CONSTRAINT E2E3 CHECK ((TIPO = 2 AND C IS NULL) OR
                       (TIPO = 3 AND B IS NULL AND C IS NOT NULL) )        );

CREATE TABLE R1 (
K1      INT NOT NULL REFERENCES E1,
K1E2    INT NOT NULL REFERENCES E1,
K1E3    INT NOT NULL REFERENCES E1,
D       INT NOT NULL,
PRIMARY KEY (K1,K1E2,K1E3)                          );
```

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 K1E2 referenzi un'istanza di E2 e K1E3 in'istanza di E3

CREATE TRIGGER R1_E2_E3
BEFORE INSERT ON R1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( NOT EXISTS ( SELECT *
                    FROM    E1 E2, E1 E3
                    WHERE N.K1E2 = E2.K1 AND E2.TIPO = 2
                    AND     N.K1E3 = E3.K1 AND E3.TIPO = 3 )    )
SIGNAL SQLSTATE '70001' ('Le tupla referenziate devono appartenere a E2 ed E3!');

CREATE TRIGGER PUNTO_C
BEFORE INSERT ON R1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( N.D <= 20 AND EXISTS ( SELECT *
                              FROM    E1
                              WHERE N.K1E2 = E1.K1
                              AND     E1.B IS NULL )    )
SIGNAL SQLSTATE '70002' ('Le tuple di E2 con B nullo possono partecipare a R1 solo se D > 20!');

-- Il trigger R1_E2_E3, che va quindi dichiarato per primo, garantisce che la tupla referenziata da N.K1E2 sia di E2.
-- In alternativa, se si dichiara prima PUNTO_C, è necessario aggiungere a tale trigger la condizione E1.TIPO = 2
```