Sistemi Informativi T
15 febbraio 2024
Risoluzione

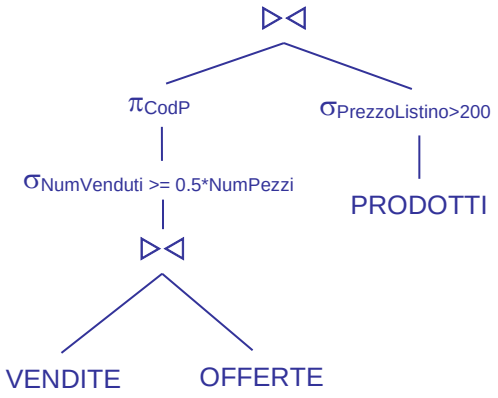
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

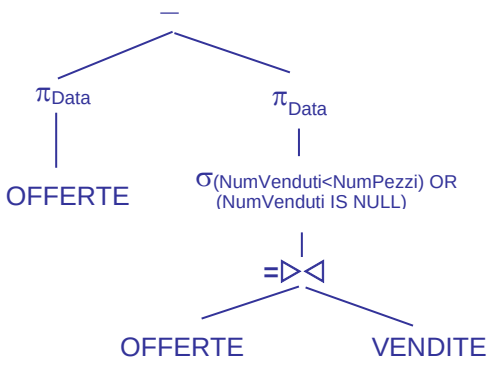
```
PRODOTTI(CodP,PrezzoListino);
OFFERTE(CodP,Data,PrezzoOfferta,NumPezzi),
      CodP REFERENCES PRODOTTI;
VENDITE(CodP,Data,NumVenduti),
      (CodP,Data) REFERENCES OFFERTE;
--
-- PrezzoListino e PrezzoOfferta (< PrezzoListino) sono di tipo DEC(6,2).
-- NumPezzi e NumVenduti (<= NumPezzi) sono interi > 0.
-- NumPezzi indica quanti sono i pezzi di un prodotto messi in offerta
-- in una certa data.
-- NB: Se per l'offerta di un prodotto in una data non ci sono state
-- vendite, la corrispondente coppia (CodP,Data) non compare in VENDITE
-- (poiché NumVenduti > 0).
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I dettagli dei prodotti con prezzo di listino maggiore di 200€ che hanno avuto un'offerta in cui sono stati venduti almeno la metà dei pezzi disponibili



1.2) **[2 p.]** Le date in cui tutti i prodotti in offerta sono stati completamente venduti



L'operando destro della differenza restituisce le date in cui almeno un prodotto non è stato completamente venduto oppure non ha nessun pezzo venduto

Sistemi Informativi T
15 febbraio 2024
Risoluzione

2) SQL (5 punti totali)

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

2.1) [2 p.] Per ogni data il rapporto tra pezzi complessivamente venduti e quelli messi in offerta, ma considerando solo i prodotti messi in offerta in almeno 2 giorni

```
SELECT   O.DATA,
         DEC(COALESCE(SUM(V.NUMVENDUTI)*1.0/SUM(O.NUMPEZZI),0),4,2)
                            AS RAPPORTO
FROM     OFFERTE O LEFT JOIN VENDITE V ON ((O.CODP,O.DATA) = (V.CODP,V.DATA))
WHERE    O.CODP IN ( SELECT O1.CODP
                     FROM   OFFERTE O1
                     GROUP BY O1.CODP
                     HAVING COUNT(*) >= 2 )
GROUP BY O.DATA;

-- Il left join serve per considerare anche i prodotti che in una data
-- erano in offerta ma non sono stati venduti, e la funzione COALESCE
-- gestisce il caso in cui in una data nessun prodotto in offerta è stato
-- venduto (e quindi RAPPORTO vale 0).
```

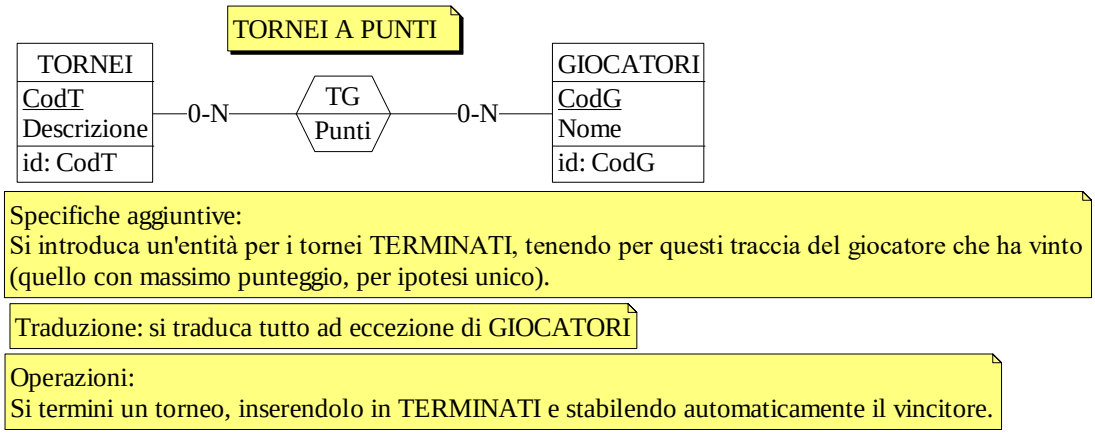
2.2) [3 p.] Il prodotto che in due sue offerte consecutive, entrambe con dei pezzi venduti, ha incassato di più

```
WITH
INCASSI2GIORNI(CODP,TOTALE) AS
( SELECT  O1.CODP,O1.PREZZOOFFERTA*V1.NUMVENDUTI +
                  O2.PREZZOOFFERTA*V2.NUMVENDUTI
  FROM  OFFERTE O1, OFFERTE O2, VENDITE V1, VENDITE V2
  WHERE O1.CODP = O2.CODP                          -- stesso prodotto
  AND   O1.DATA < O2.DATA                          -- offerte diverse
  AND   (V1.CODP,V1.DATA) = (O1.CODP,O1.DATA)      -- con vendite
  AND   (V2.CODP,V2.DATA) = (O2.CODP,O2.DATA)
  AND   NOT EXISTS ( SELECT *                  -- offerte consecutive
                     FROM   OFFERTE O3
                     WHERE  O3.CODP = O1.CODP
                     AND    O3.DATA > O1.DATA
                     AND    O3.DATA < O2.DATA )  )
SELECT   *
FROM     INCASSI2GIORNI I
WHERE    I.TOTALE = ( SELECT MAX(I2.TOTALE)
                      FROM   INCASSI2GIORNI I2 )   ;

-- La c.t.e. calcola l'incasso per ogni prodotto considerando due offerte
-- consecutive in entrambe delle quali si sono avute delle vendite.
```

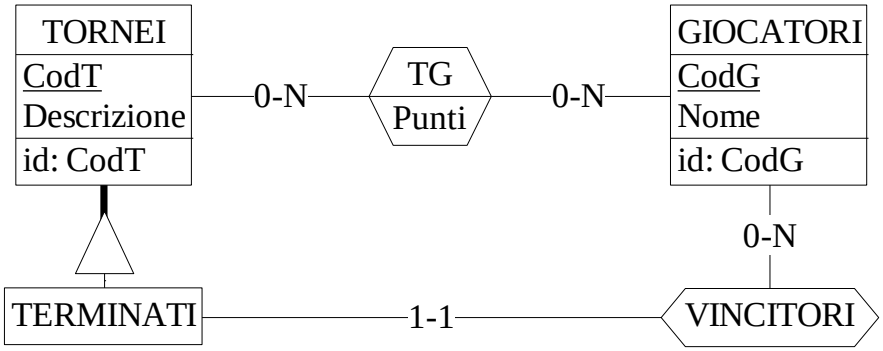
Sistemi Informativi T
15 febbraio 2024
Risoluzione

3) Modifica di schema E/R e del DB (6 punti totali)

Dato il file ESE3.lun fornito, in cui è presente lo schema ESE3-input in figura:



3.1) [2 p.] Si modifichi ESE3-input secondo le Specifiche aggiuntive;



3.2) [1 p.] Si copi lo schema modificato in uno schema ESE3-tradotto. Mediante il comando Transform/Quick SQL, si traduca la parte di schema specificata, modificando lo script SQL in modo da essere compatibile con DB2 e permettere l'esecuzione del punto successivo, ed eventualmente aggiungendo quanto richiesto dalle Specifiche aggiuntive;
Si veda il relativo file .sql

3.3) [3 p.] Si scriva l'istruzione SQL che modifica il DB come da specifiche (usare valori a scelta) e si definiscano i trigger necessari.

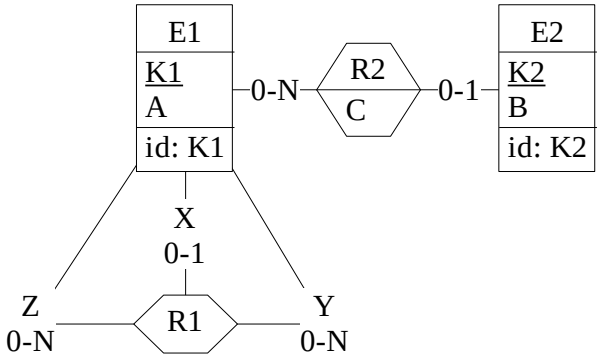
```
CREATE OR REPLACE TRIGGER DETERMINA_VINCITORE
BEFORE INSERT ON TERMINATI
REFERENCING NEW AS N
FOR EACH ROW
SET N.CodG_Vincitore =
    (    SELECT TG1.CodG
         FROM   TG TG1
         WHERE  TG1.CodT = N.CodT
         AND    TG1.Punti = ( SELECT MAX(TG2.Punti)
                              FROM   TG TG2
                              WHERE  TG2.CodT = TG1.CodT ) );


INSERT INTO TERMINATI(CodT)
VALUES (:CodT);
```

Sistemi Informativi T
15 febbraio 2024
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) nessuna associazione viene tradotta separatamente;
b) ogni terna di istanze di E1 che partecipa a R1 è costituita da istanze diverse tra loro e tali per cui la somma dei loro valori di A è minore di 50;
c) un'istanza di E2 non è mai associata tramite R2 a un'istanza di E1 che partecipa a R1 con il ruolo Z;



4.1) [3 p.] Si progettino gli opportuni schemi relazionali e si definiscano tali schemi mediante uno script SQL compatibile con DB2

```
-- il tipo degli attributi non è necessariamente INT
CREATE TABLE E1 (
K1          INT NOT NULL PRIMARY KEY,
A           INT NOT NULL,
K1Y         INT REFERENCES E1,
K1Z         INT REFERENCES E1,
CONSTRAINT R1 CHECK ((K1Y IS NOT NULL AND K1Z IS NOT NULL) OR
                     (K1Y IS NULL AND K1Z IS NULL)),
CONSTRAINT PUNTO_B CHECK ((K1 <> K1Y) AND (K1 <> K1Z) AND (K1Y <> K1Z))   );

CREATE TABLE E2 (
K2          INT NOT NULL PRIMARY KEY,
B           INT NOT NULL,
K1R2        INT REFERENCES E1,
C           INT,
CONSTRAINT R2 CHECK ((K1R2 IS NOT NULL AND C IS NOT NULL) OR
                     (K1R2 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 singole tuple non corrette**

```
-- Trigger che completa il requisito del punto b)
CREATE TRIGGER PUNTO_B
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.A +  ( SELECT E1Y.A + E1Z.A        -- funziona anche se le 2 foreign key sono nulle
               FROM E1 E1Y, E1 E1Z
               WHERE N.K1Y = E1Y.K1
               AND      N.K1Z = E1Z.K1 ) >= 50 )
SIGNAL SQLSTATE '70001' ('La tre tuple associate via R1 hanno somma di A >= 50! ');


-- Trigger che garantisce il rispetto del vincolo al punto c)
CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS  ( SELECT   *
                 FROM     E1
                 WHERE  N.K1R2 = E1.K1Z )  )
SIGNAL SQLSTATE '70003' ('La tupla inserita non rispetta il vincolo del punto c)! ');
```