Sistemi Informativi T
25 gennaio 2024
Risoluzione

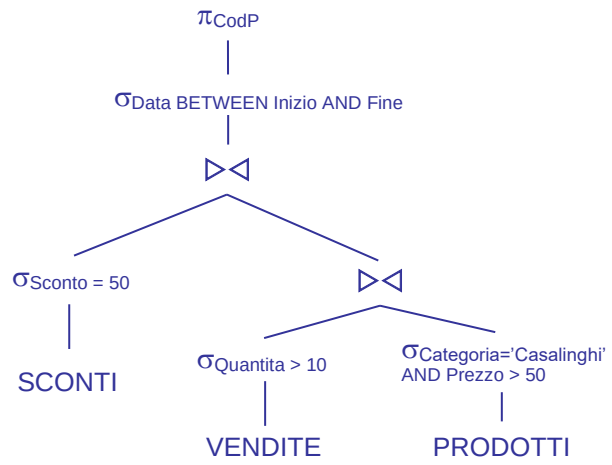
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

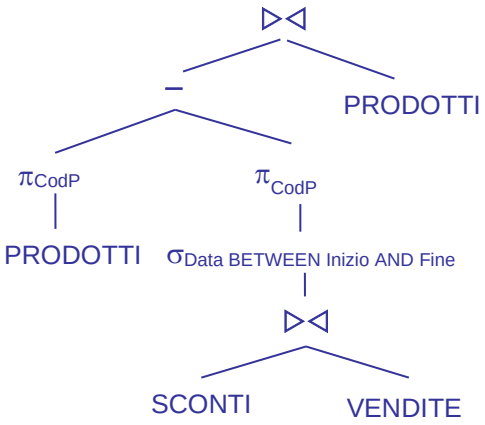
```
PRODOTTI(CodP,Categoria,Prezzo);
SCONTI(Inizio,Fine,Sconto);
VENDITE(CodP,Data,Quantita),
      CodP REFERENCES PRODOTTI;
--
-- Prezzo è di tipo DEC(6,2).
-- Inizio e Fine sono date che definiscono il periodo
-- in cui viene praticato un certo Sconto su tutti i prodotti
-- (i periodi di sconto sono tra loro disgiunti).
-- Sconto è un intero, 0 < Sconto < 100, che indica la percentuale
-- di sconto (ad es. 35 è il 35% di sconto).
-- Quantita è un intero > 0.
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I codici dei prodotti di categoria Casalinghi e prezzo maggiore di 50€ che in un giorno sono stati venduti in quantità maggiore di 10 con uno sconto del 50%



1.2) **[2 p.]** I dettagli dei prodotti che non sono mai stati venduti a prezzo scontato



L'operando destro della differenza trova i prodotti venduti almeno una volta a prezzo scontato

Sistemi Informativi T
25 gennaio 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.] I dettagli di ogni prodotto per il quale la quantità complessivamente venduta a prezzo non scontato è maggiore di 5, e il valore di tale quantità

```
SELECT   P.CODP, P.CATEGORIA, P.PREZZO, SUM(V.QUANTITA) AS QTATOTALE
FROM     PRODOTTI P, VENDITE V
WHERE    P.CODP = V.CODP
AND      NOT EXISTS ( SELECT *  -- solo vendite non scontate
                      FROM   SCONTI S
                      WHERE  V.DATA BETWEEN S.INIZIO AND S.FINE )
GROUP BY P.CODP, P.CATEGORIA, P.PREZZO
HAVING SUM(V.QUANTITA) > 5;
```

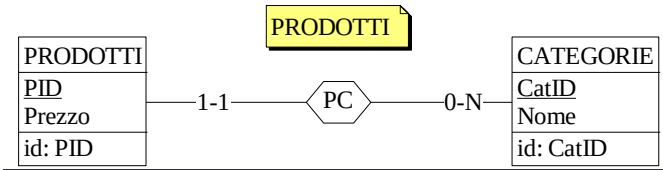
2.2) [3 p.] Per ogni categoria, il prodotto che ha complessivamente incassato di più nei periodi di sconto, con il relativo incasso

```
WITH INCASSISCONTATI(CODP,CATEGORIA,TOTINCASSO) AS
( SELECT   P.CODP, P.CATEGORIA,
            DEC(SUM(V.QUANTITA*P.PREZZO*(1 - S.SCONTO/100.0)),8,2)
   FROM    PRODOTTI P, VENDITE V, SCONTI S
   WHERE   P.CODP = V.CODP
   AND     V.DATA BETWEEN S.INIZIO AND S.FINE )
   GROUP BY P.CODP, P.CATEGORIA )
SELECT   I.*
FROM     INCASSISCONTATI I
WHERE    I.TOTINCASSO >= ALL ( SELECT  I1.TOTINCASSO
                               FROM    INCASSISCONTATI I1
                               WHERE   I1.CATEGORIA = I.CATEGORIA );

-- La c.t.e. calcola l'incasso per ogni prodotto. Si noti il CAST implicito
-- S.SCONTO/100.0, necessario poiché SCONTO è intero
```

Sistemi Informativi T
25 gennaio 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:

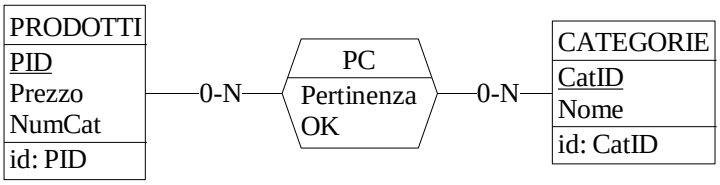


Specifiche aggiuntive:
Un prodotto può far parte di 0 o più categorie, per ognuna delle quali si specifica un grado di pertinenza da 1 a 10 e un flag OK che vale 1 se la pertinenza è maggiore di 5, altrimenti 0.
L'attributo NumCat (default 0) riassume il numero di categorie di un prodotto.

Traduzione: si traduca tutto ad eccezione di CATEGORIE

Operazioni:
Si aggiunga una categoria a un prodotto esistente, settando automaticamente OK e aggiornando NumCat.

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.

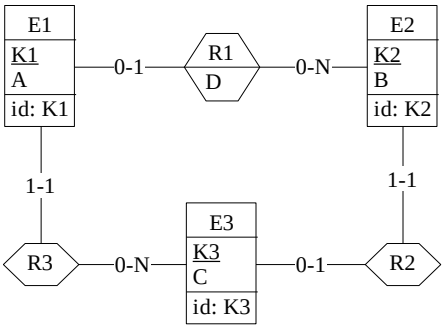
```
CREATE OR REPLACE TRIGGER NUOVA_CATEGORIA
BEFORE INSERT ON PC
REFERENCING NEW AS N
FOR EACH ROW
IF N.Pertinenza > 5 THEN SET N.OK = 1; ELSE SET N.OK = 0;
END IF

CREATE OR REPLACE TRIGGER UPDATE_NUMCAT
AFTER INSERT ON PC
REFERENCING NEW AS N
FOR EACH ROW
UPDATE PRODOTTI
SET NumCat = NumCat +1
WHERE PID = N.PID;

INSERT INTO PC(PID,CatID,Pertinenza)
VALUES (:pid,:catid,:pertinenza);
```

Sistemi Informativi T
25 gennaio 2024
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) le entità E2 ed E3 vengono tradotte assieme;
b) nessuna associazione viene tradotta separatamente;
c) un'istanza di E1 non è mai associata, tramite R1 e R2, alla stessa istanza di E3 cui è associata tramite R3;



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 E2E3 (
K3          INT NOT NULL PRIMARY KEY,
                -- la sola chiave senza valori nulli è quella di E3, che partecipa con cardinalità minima 0
C           INT NOT NULL,
TIPO2       SMALLINT NOT NULL CHECK (TIPO2 IN (2,3)), -- se 2 esiste anche l'istanza di E2
K2          INT,    -- chiave con valori nulli
B           INT,
CONSTRAINT E2 CHECK ((TIPO2 = 2 AND K2 IS NOT NULL AND B IS NOT NULL)
                OR (TIPO2 = 3 AND K2 IS NULL AND B IS NULL))            );

CREATE TABLE E1 (
K1          INT NOT NULL PRIMARY KEY,
A           INT NOT NULL,
K3R3        INT NOT NULL REFERENCES E2E3,
K2R1        INT,    -- non può essere una foreign key perché K2 non è dichiarabile come chiave
D           INT,
CONSTRAINT R1 CHECK ((K2R1 IS NULL AND D IS NULL) OR
                (K2R1 IS NOT NULL AND D IS NOT 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 garantisce l'unicità dei valori di K2
CREATE TRIGGER K2_UNIQUE
BEFORE INSERT ON E2E3
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (   SELECT * FROM   E2E3
                  WHERE N.K2 = E2E3.K2  ) )
SIGNAL SQLSTATE '70001' ('I valori di K2 non possono essere duplicati! ');

-- Trigger che garantisce che R1 referenzi un'istanza di E2
CREATE TRIGGER R1_E2
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( N.K2R1 IS NOT NULL AND NOT EXISTS (  SELECT * FROM E2E3
                                            WHERE N.K2R1 = E2E3.K2  ) )
SIGNAL SQLSTATE '70002' ('La tupla referenzia una tupla che non appartiene a E2! ');
-- La condizione E2E3.TIPO2 = 2 è superflua; se TIPO2 = 3 allora K2 è NULL e la subquery non ritorna nulla

-- Trigger che garantisce il rispetto del vincolo al punto c)
CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( N.K3R3 =  ( SELECT  E2E3.K3 FROM E2E3
                  WHERE  N.K2R1 = E2E3.K2  ) )
SIGNAL SQLSTATE '70003' ('La tupla inserita non rispetta il vincolo del punto c)! ');
```