Sistemi Informativi T
5 febbraio 2024
Risoluzione

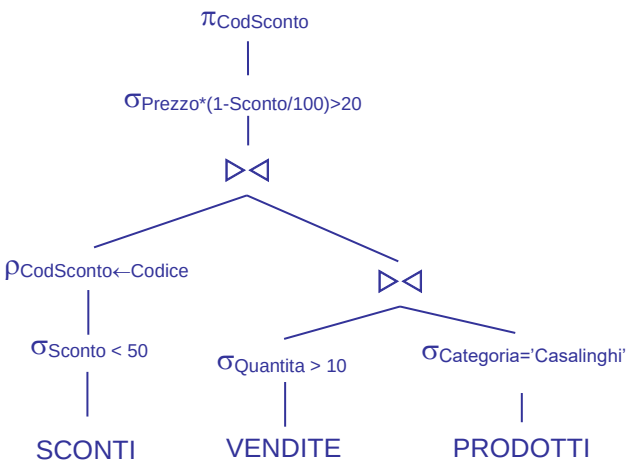
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

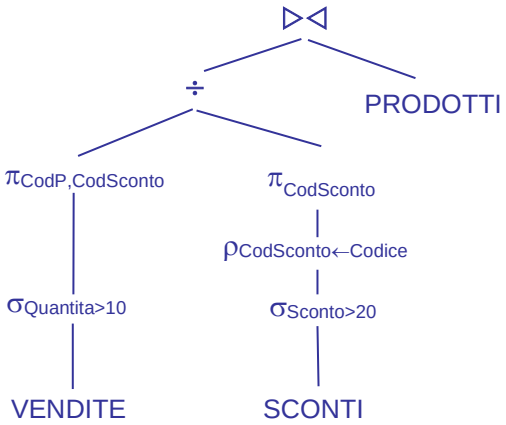
```
PRODOTTI(CodP,Categoria,Prezzo);
SCONTI(Codice,Sconto);
VENDITE(CodP,Data,Quantita,CodSconto*),
      CodP REFERENCES PRODOTTI,
      CodSconto REFERENCES SCONTI;
--
-- Prezzo è di tipo DEC(6,2).
-- Sconto è un intero, 0 < Sconto < 100, che indica la percentuale
-- di sconto (ad es. 35 è il 35% di sconto).
-- Quantita è un intero > 0.
-- CodSconto è null se la vendita è avvenuta a prezzo intero (senza sconto)
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I codici degli sconti minori del 50% per i quali in un giorno è stato venduto un articolo di categoria Casalinghi in quantità maggiore di 10 e prezzo scontato maggiore di 20€



1.2) **[2 p.]** I dettagli dei prodotti che, per ogni sconto maggiore del 20%, hanno avuto almeno un giorno in cui sono stati venduti in quantità maggiore di 10



Il divisore è dato da tutti i codici con sconto maggiore del 20%, mentre nel dividendo si considerano solo le vendite in quantità maggiore di 10

Sistemi Informativi T
5 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 codice sconto e ogni categoria, il prezzo medio a cui sono stati venduti i relativi prodotti (prezzo medio: incasso totale diviso quantità totale)

```
SELECT   V.CODSCONTO, P.CATEGORIA,
         DEC(SUM(V.QUANTITA*P.PREZZO*(1-S.SCONTO/100.0))/
                           SUM(V.QUANTITA),6,2) AS PREZZO_MEDIO
FROM     PRODOTTI P, VENDITE V, SCONTI S
WHERE    P.CODP = V.CODP
AND      V.CODSCONTO = S.CODICE
GROUP BY V.CODSCONTO, P.CATEGORIA;
```

2.2) **[3 p.]** L'incasso totale per ogni prodotto, ordinando il risultato per codice prodotto

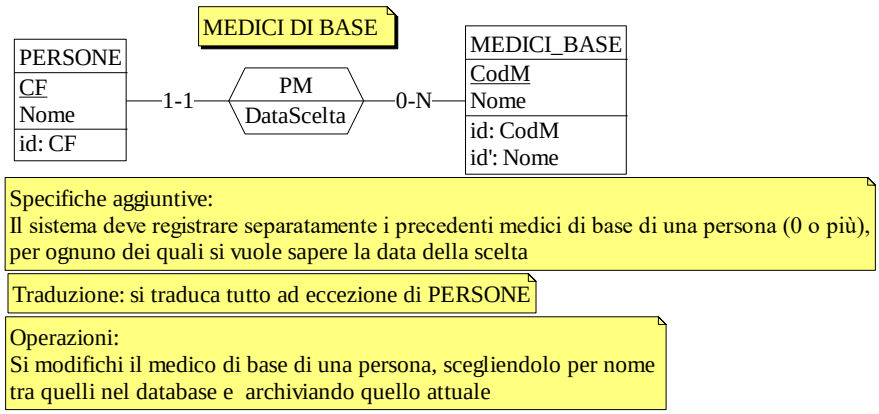
```
WITH
SCONTATI(CODP,TOTS) AS
( SELECT P.CODP, SUM(V.QUANTITA*P.PREZZO*(1-S.SCONTO/100.0))
  FROM   PRODOTTI P, VENDITE V, SCONTI S
  WHERE  P.CODP = V.CODP
  AND    V.CODSCONTO = S.CODICE
  GROUP BY P.CODP ),
NONSCONTATI(CODP,TOTNS) AS
( SELECT P.CODP, SUM(V.QUANTITA*P.PREZZO)
  FROM   PRODOTTI P, VENDITE V
  WHERE  P.CODP = V.CODP
  AND    V.CODSCONTO IS NULL
  GROUP BY P.CODP )
SELECT   COALESCE(S.CODP,NS.CODP) AS CODP,
         DEC(COALESCE(S.TOTS,0)+COALESCE(NS.TOTNS,0),8,2)
              AS INCASSO
FROM     SCONTATI S FULL JOIN NONSCONTATI NS ON (S.CODP = NS.CODP)
ORDER BY CODP;

-- La prima c.t.e. calcola gli incassi scontati, la seconda quelli non
-- scontati. Il full join è necessario perché un prodotto potrebbe avere
-- incassi solo di un tipo.
```

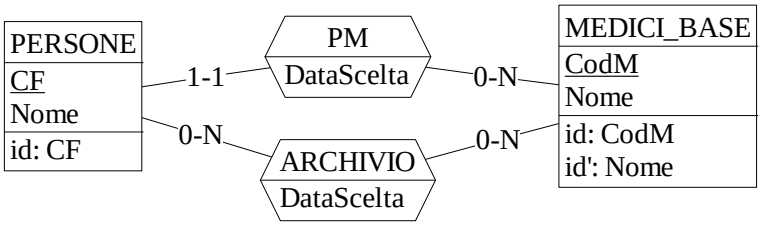
Sistemi Informativi T
5 febbraio 2024
Risoluzione

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

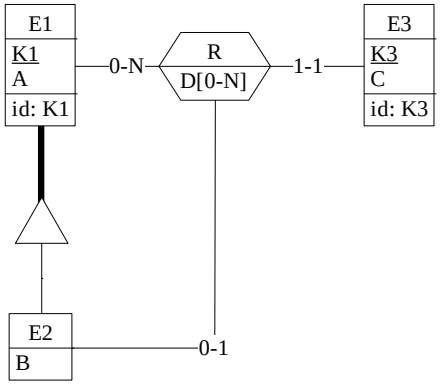
3.3) [3 p.] Si scriva l'istruzione SQL che modifica il DB come da specifiche (usare valori a scelta) e si definiscano i trigger necessari.

```
CREATE OR REPLACE TRIGGER CAMBIO_MEDICO
AFTER UPDATE ON PM
REFERENCING OLD AS O
FOR EACH ROW
INSERT INTO ARCHIVIO
VALUES (O.CF,O.CodM,O.DataScelta)      ;


UPDATE   PM
SET      CodM = (SELECT    M.CodM
                   FROM  MEDICI_BASE M
                   WHERE M.Nome = :nome),
         DataScelta = CURRENT DATE
WHERE    CF = :cf     ;
```

Sistemi Informativi T
5 febbraio 2024
Risoluzione

4) Progettazione logica (6 punti totali)



Dato lo schema concettuale in figura e considerando che:

a) le entità E1 ed E2 vengono tradotte assieme;
b) l'associazione R viene tradotta con E3;
c) un'istanza di E3 non è mai associata a istanze di E1 ed E2 tali che C < A+B;

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,
TIPO2       SMALLINT NOT NULL CHECK (TIPO2 IN (1,2)), -- se 2 è istanza anche di E2
B           INT,
CONSTRAINT E2 CHECK ((TIPO2 = 1 AND B IS NULL) OR (TIPO2 =  2 AND B IS NOT NULL))  );

CREATE TABLE E3 (
K3          INT NOT NULL PRIMARY KEY,
C           INT NOT NULL,
K1E1        INT NOT NULL REFERENCES E1,
K1E2        INT NOT NULL UNIQUE REFERENCES E1                    );

CREATE TABLE RD (
K3          INT NOT NULL REFERENCES E3,
D           INT NOT NULL,
PRIMARY KEY (K3,D)                                               );
```

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 che K1E2 referenzi un'istanza di E2
CREATE TRIGGER R_E2
BEFORE INSERT ON E3
REFERENCING NEW AS N
FOR EACH ROW
WHEN (NOT EXISTS  ( SELECT * FROM E1
                    WHERE N.K1E2 = E1.K1
                    AND      E1.TIPO2 = 2  )  )
SIGNAL SQLSTATE '70001' ('La tupla referenzia con K1E2 una tupla che non appartiene a E2! ');


-- Trigger che garantisce il rispetto del vincolo al punto c)
CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E3
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( N.C <   ( SELECT     E1.A + E2.B
                 FROM       E1, E1 E2
                 WHERE      N.K1E1 = E1.K1
                 AND        N.K1E2 = E2.K1  )  )
SIGNAL SQLSTATE '70003' ('La tupla inserita non rispetta il vincolo del punto c)! ');
```