Sistemi Informativi T
30 gennaio 2023
Risoluzione

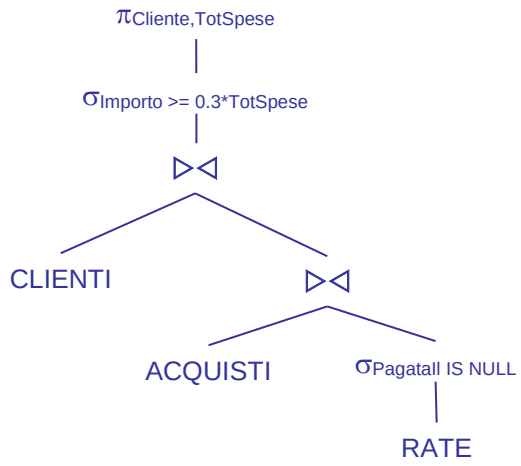
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

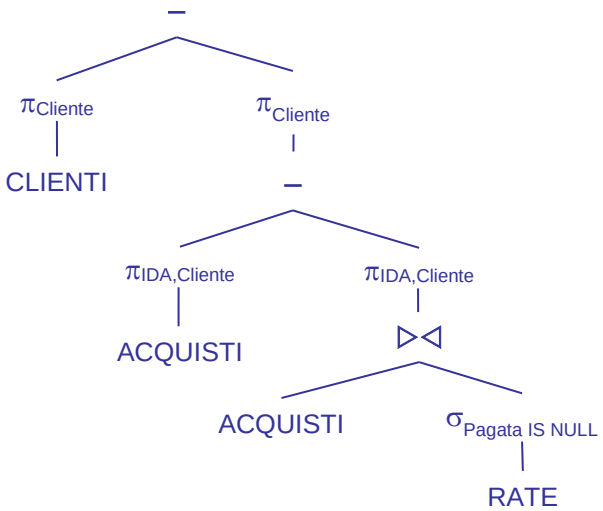
```
CLIENTI(Cliente,TotSpese);
ACQUISTI(IDA,Cliente,Importo), Cliente REFERENCES CLIENTI;
RATE(IDA,Num,ImportoRata,PagataIl*), IDA REFERENCES ACQUISTI;
-- TotSpese, Importo e ImportoRata sono di tipo DEC(8,2).
-- Num è di tipo INT, e distingue le rate di un acquisto (1,2,…).
-- PagataIl è di tipo DATE: se la rata non è stata pagata è NULL.
-- RATE contiene tutte le rate relative a un dato acquisto, sia
-- quelle già pagate che quelle ancora da pagare.
-- Importo è pari alla somma degli importi delle relative rate, e
-- TotSpese è pari alla somma degli importi degli acquisti del cliente.
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I dati dei clienti che devono ancora pagare una o più rate per un acquisto di importo almeno pari al 30% della propria spesa totale



1.2) **[2 p.]** I nomi dei clienti che non hanno ancora finito di pagare nessun acquisto



La prima differenza trova gli acquisti completamente pagati, e quindi l'operando destro della seconda differenza sono i clienti che hanno finito di pagare almeno un acquisto

Sistemi Informativi T
30 gennaio 2023
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 acquisto non ancora completamente pagato, ma per cui sono state pagate almeno 2 rate, la percentuale pagata rispetto all'importo dell'acquisto

```
SELECT   A.IDA, 100*SUM(R.IMPORTORATA)/A.IMPORTO || '%' AS PERC_PAGATA
FROM     ACQUISTI A, RATE R
WHERE    A.IDA = R.IDA
AND      R.PAGATAIL IS NOT NULL
AND      EXISTS (SELECT    *
                 FROM      RATE R
                 WHERE     R.IDA = A.IDA
                 AND       R.PAGATAIL IS NULL )
GROUP BY A.IDA, A.IMPORTO
HAVING COUNT(*) >=2  ;      -- almeno 2 rate pagate
```

2.2) [3 p.] Per ogni cliente che ha pagato almeno 2 rate in tutti gli acquisti fatti, l'identificativo dell'acquisto in cui l'importo totale delle rate pagate è massimo

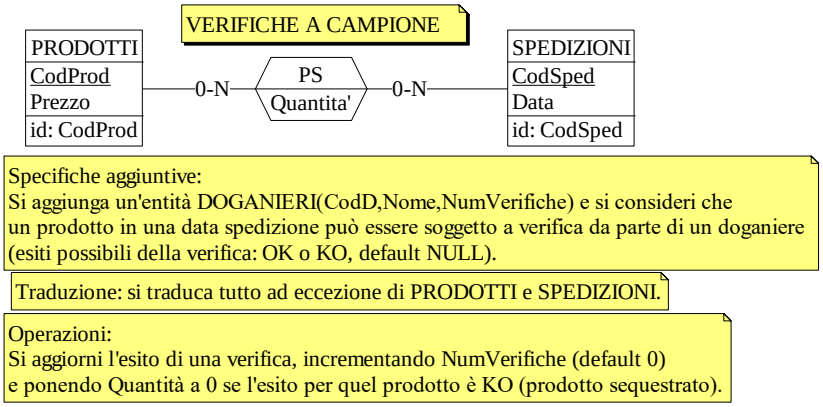
```
WITH TOTALI(CLIENTE,IDA,TOT_PAGATO) AS
(SELECT A.CLIENTE, A.IDA, SUM(R.IMPORTORATA)
 FROM    ACQUISTI A, RATE R
 WHERE   A.IDA = R.IDA
 AND     R.PAGATAIL IS NOT NULL
 AND     A.CLIENTE NOT IN
   (SELECT     DISTINCT A.CLIENTE      -- clienti da escludere
    FROM       ACQUISTI A, RATE R
    WHERE      A.IDA = R.IDA
    GROUP BY   A.IDA, A.CLIENTE
    HAVING     COUNT(R.PAGATAIL) < 2)
 GROUP BY A.CLIENTE, A.IDA
)
SELECT  *
FROM    TOTALI T1
WHERE   T1.TOT_PAGATO = ( SELECT MAX(T2.TOT_PAGATO)
                          FROM   TOTALI T2
                          WHERE  T2.CLIENTE = T1.CLIENTE)     ;

-- La subquery nella c.t.e. restituisce i clienti con almeno un acquisto
-- per cui sono state pagate meno di 2 rate.
```

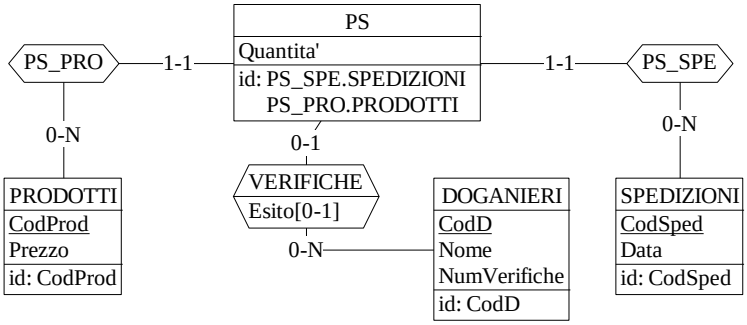
Sistemi Informativi T
30 gennaio 2023
Risoluzione

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.

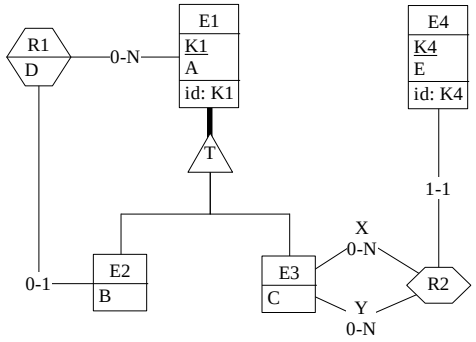
```
UPDATE VERIFICHE
SET       Esito = :esito
WHERE (CodSped, CodProd) = (:codsped,:codprod);

CREATE OR REPLACE TRIGGER AGGIORNA_NUM_VERIFICHE
AFTER UPDATE OF Esito ON VERIFICHE
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.Esito IS NOT NULL)
UPDATE     DOGANIERI
SET           NumVerifiche = NumVerifiche + 1
WHERE      CodD = N.CodD ;

CREATE OR REPLACE TRIGGER AZZERA_QUANTITA
AFTER UPDATE OF Esito ON VERIFICHE
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.Esito = 'KO')
UPDATE     PS
SET           Quantita = 0
WHERE      (CodSped, CodProd) = (N.CodSped, N.CodProd);
```

Sistemi Informativi T
30 gennaio 2023
Risoluzione

4) Progettazione logica (6 punti totali)



Dato lo schema concettuale in figura e considerando che:

a) la gerarchia viene tradotta mediante un collasso verso il basso;
b) nessuna associazione viene tradotta separatamente;
c) un'istanza di E3 non può essere referenziata, tramite R1, da un'istanza di E2 se C < 5;
d) se un'istanza di E3 già appartiene anche a E2, allora non può partecipare all'associazione R2 dal ramo X se ha C + D > 10;

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 E3 (
K1          INT NOT NULL PRIMARY KEY,
A           INT NOT NULL,
C           INT NOT NULL          );

CREATE TABLE E2 (
K1          INT NOT NULL PRIMARY KEY,
A           INT NOT NULL,
B           INT NOT NULL,
K1R1E2      INT REFERENCES E2,                  -- R1 va sdoppiata
K1R1E3      INT REFERENCES E3,
D           INT,                                -- non nullo se almeno una FK è non nulla
CONSTRAINT MAX_CARD_R1 CHECK (K1R1E2 = K1R1E3),
     -- se entrambe non nulle devono referenziare una stessa istanza, se una è nulla il CHECK non è violato
CONSTRAINT R1_D CHECK ((K1R1E2 IS NULL AND K1R1E3 IS NULL AND D IS NULL) OR
          ((K1R1E2 IS NOT NULL OR K1R1E3 IS NOT NULL) AND D IS NOT NULL))          );

CREATE TABLE E4 (
K4          INT NOT NULL PRIMARY KEY,
E           INT NOT NULL,
K1R2X       INT NOT NULL REFERENCES E3,
K1R2Y       INT NOT NULL REFERENCES E3                  );
```

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**

```
CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( 5 >  (       SELECT E3.C
                    FROM   E3
                    WHERE N.K1R1E3 = E3.K1  )  )
SIGNAL SQLSTATE '70001' ('La tupla inserita in E2 non rispetta il vincolo del punto c)! ');

-- Il vincolo al punto d) può essere violato solo inserendo una tupla in E4
-- NB: il testo originale non includeva "già" al punto d). Senza "già" è possibile violare il vincolo anche
-- con un INSERT in E2, cosa che non era voluta nella preparazione dell'esercizio
CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E4
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( 10 <  (      SELECT E2.D + E3.C FROM E2, E3
                    WHERE N.K1R2X = E2.K1
                    AND     N.K1R2X = E3.K1  )  )
SIGNAL SQLSTATE '70002' ('La tupla inserita in E4 non rispetta il vincolo del punto d)! ');
```