**Sistemi Informativi T**
**19 gennaio 2023**
**Risoluzione**

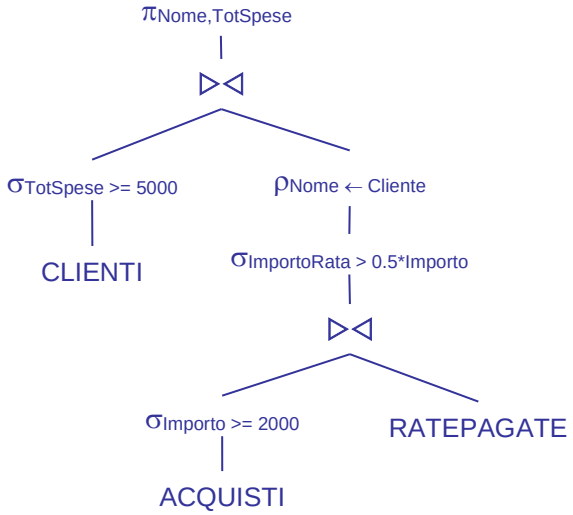
## 1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

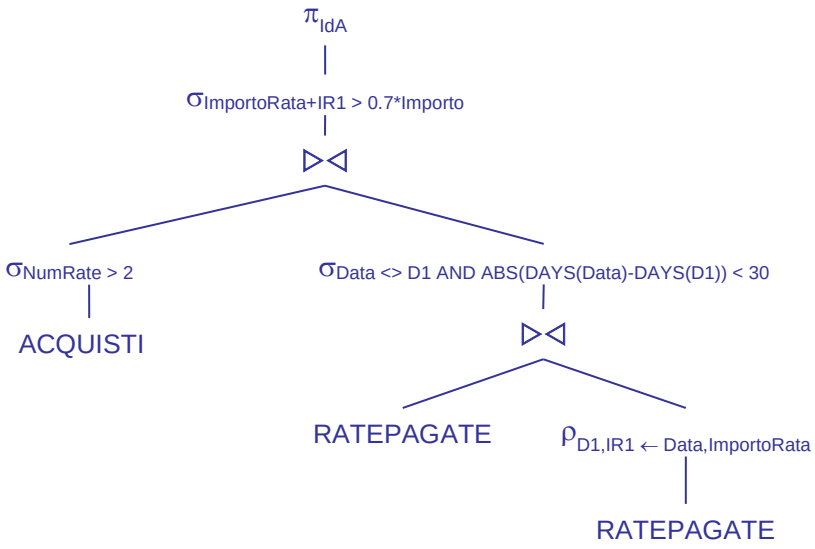
```
CLIENTI(Nome,TotSpese);
ACQUISTI(IDA,Cliente,Importo,NumRate),
      Cliente REFERENCES CLIENTI;
RATEPAGATE(IDA,Data,ImportoRata),
      IDA REFERENCES ACQUISTI;
-- TotSpese, Importo e ImportoRata sono di tipo DEC(8,2).
-- NumRate è di tipo INT (> 1).
```

si esprimano in algebra relazionale le seguenti interrogazioni:

**1.1) [1 p.]** I dati dei clienti con un totale spese di almeno 5000€ che hanno fatto un acquisto di 2000€ o più, e hanno pagato una rata di importo maggiore del 50% dell'importo di tale acquisto




**1.2) [2 p.]** L'identificativo degli acquisti con 3 o più rate in cui almeno 2 rate sono state pagate a distanza di meno di 30 giorni l'una dall'altra, per un importo complessivo maggiore del 70% dell'importo dell'acquisto



**Sistemi Informativi T**
**19 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.]** I nomi dei clienti che hanno un totale spese maggiore di 5000€ e che hanno fatto almeno un acquisto di 3 o più rate in cui l'importo delle rate pagate (almeno 2) è sempre aumentato, riportando in output anche l'identificativo dell'acquisto

```
SELECT   C.NOME, A.IDA
FROM     CLIENTI C, ACQUISTI A
WHERE    C.NOME = A.CLIENTE
AND      C.TOTSPESE > 5000
AND      A.NUMRATE >= 3
AND      NOT EXISTS ( SELECT *          -- rata uguale o diminuita
                      FROM   RATEPAGATE R1, RATEPAGATE R2
                      WHERE  R1.IDA = R2.IDA
                      AND    A.IDA = R1.IDA
                      AND    R1.DATA < R2.DATA
                      AND    R1.IMPORTORATA >= R2.IMPORTORATA)
AND      2 <= ( SELECT COUNT(*)         -- almeno 2 rate pagate
                FROM   RATEPAGATE R
                WHERE  R.IDA = A.IDA ) ;

-- La seconda subquery è necessaria per non restituire chi ha acquisti
-- senza rate pagate o con una sola rata pagata, per i quali la prima
-- subquery non restituisce mai nulla
```

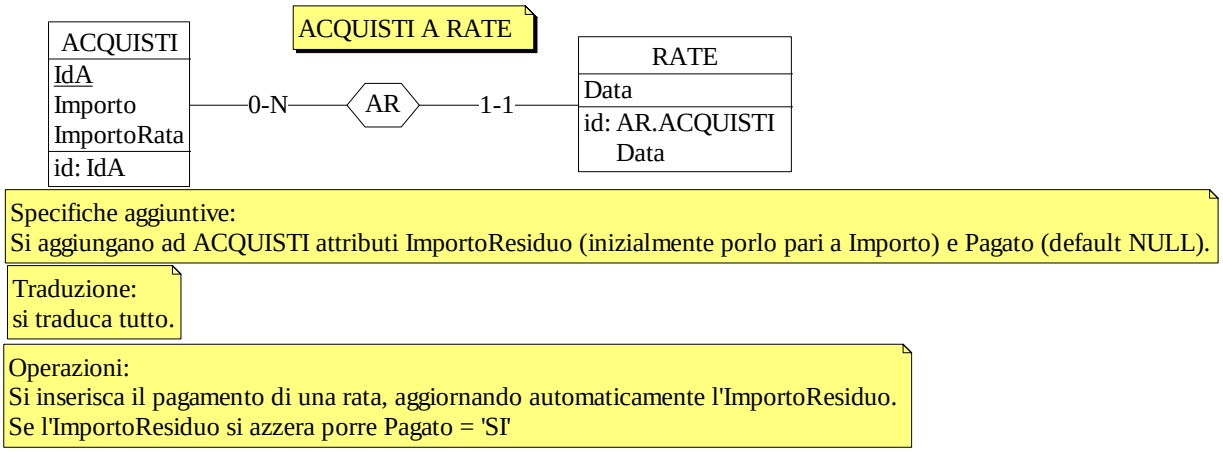
**2.2) [3 p.]** Considerando solo gli acquisti per cui sono state pagate almeno 2 rate, l'identificativo dell'acquisto e il relativo cliente per cui il tempo trascorso tra una rata e la successiva è stato massimo

```
WITH MAXTEMPO(IDA, NUMGIORNI) AS
(SELECT  R1.IDA, MAX(DAYS(R2.DATA)-DAYS(R1.DATA))
 FROM    RATEPAGATE R1, RATEPAGATE R2
 WHERE   R1.IDA = R2.IDA          -- stesso acquisto
 AND     R1.DATA < R2.DATA
 AND     NOT EXISTS ( SELECT *    -- rate consecutive
                      FROM   RATEPAGATE R
                      WHERE  R.IDA = R1.IDA
                      AND    R1.DATA < R.DATA
                      AND    R.DATA < R2.DATA      )
 GROUP BY R1.IDA     )
SELECT   A.IDA, A.CLIENTE, M1.NUMGIORNI
FROM     MAXTEMPO M1, ACQUISTI A
WHERE    M1.IDA = A.IDA
AND      M1.NUMGIORNI = ( SELECT MAX(M2.NUMGIORNI)
                          FROM   MAXTEMPO M2 )     ;

-- La c.t.e. seleziona gli acquisti con almeno 2 rate pagate (grazie alla
-- condizione R1.DATA < R2.DATA) e per questi calcola la massima differenza
-- di giorni tra rate consecutive
```

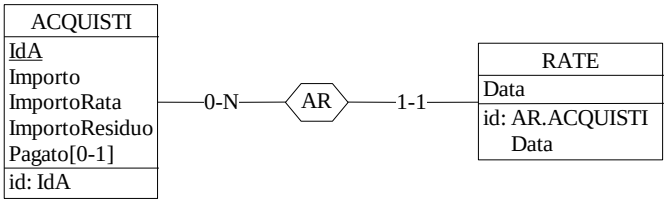
**Sistemi Informativi T**
**19 gennaio 2023**
**Risoluzione**

## 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) [1 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) [4 p.]** Si scriva l'istruzione SQL che modifica il DB come da specifiche (usare valori a scelta) e si definiscano i trigger necessari.

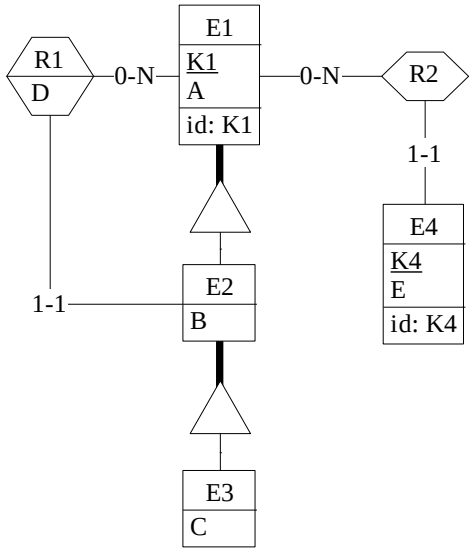
```
INSERT INTO RATE VALUES
(:idAcquisto,:data);

CREATE OR REPLACE TRIGGER UPDATE_RESIDUO
AFTER INSERT ON RATE
REFERENCING NEW AS N
FOR EACH ROW
UPDATE     ACQUISTI
SET        ImportoResiduo = ImportoResiduo - ImportoRata
WHERE      IdA = N.IdA ;

CREATE OR REPLACE TRIGGER PAGATO
AFTER UPDATE OF ImportoResiduo ON ACQUISTI
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.ImportoResiduo = 0)
UPDATE     ACQUISTI
SET        Pagato = 'SI'
WHERE      IdA = N.IdA ;
```

**Sistemi Informativi T**
**19 gennaio 2023**
**Risoluzione**

## 4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) le entità E1, E2 ed E3 vengono tradotte assieme;
b) nessuna associazione viene tradotta separatamente;
c) un'istanze di E4 non è mai associata a un'istanza di E1 che partecipa, dal ramo 0-N, a R1;




**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,
TIPO        SMALLINT NOT NULL CHECK (TIPO IN (1,2,3)),
            -- 2: istanza di E2 ma non di E3; 3: istanza di E3
B           INT,
K1R1        INT REFERENCES E1,
D           INT,
C           INT,
CHECK ((TIPO = 1 AND B IS NULL AND K1R1 IS NULL AND D IS NULL AND C IS NULL) OR
  (TIPO = 2 AND B IS NOT NULL AND K1R1 IS NOT NULL AND D IS NOT NULL AND C IS NULL) OR
  (TIPO = 3 AND B IS NOT NULL AND K1R1 IS NOT NULL AND D IS NOT NULL AND C IS NOT NULL)) );

CREATE TABLE E4 (
K4          INT NOT NULL PRIMARY KEY,
K1R2        INT NOT NULL REFERENCES E1,
E           INT 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**

```
-- Il vincolo al punto c) impedisce che E1.K1R1 e E4.K1R2 abbiano valori in comune, e può quindi
-- essere violato inserendo in E1 o in E4

CREATE TRIGGER PUNTO_C_E1
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (    SELECT *
                   FROM E4
                   WHERE N.K1R1 = E4.K1R2  )  )
SIGNAL SQLSTATE '70001' ('La tupla inserita in E1 non rispetta il vincolo del punto c)! ');

CREATE TRIGGER PUNTO_C_E4
BEFORE INSERT ON E4
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (     SELECT *
                   FROM E1
                   WHERE N.K1R2 = E1.K1R1  )  )
SIGNAL SQLSTATE '70002' ('La tupla inserita in E4 non rispetta il vincolo del punto c)! ');
```