**Sistemi Informativi T**
**24 gennaio 2022**
**Risoluzione**

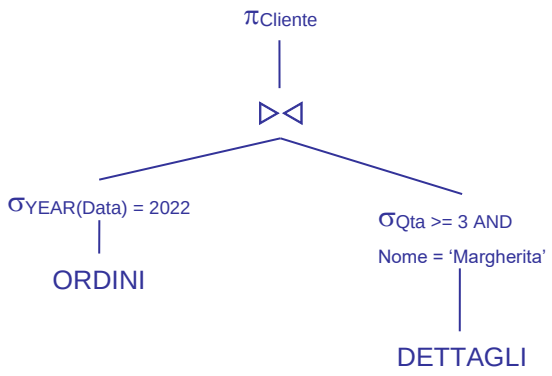
## 1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

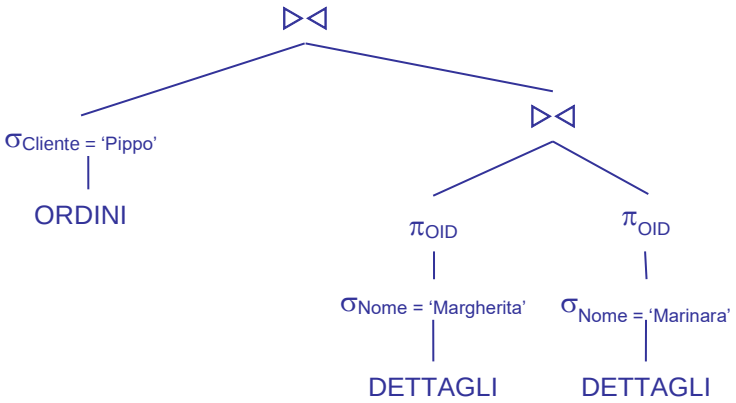
```
PIZZE(Nome,Prezzo);
ORDINI(OID,Data,Importo,Cliente);
DETTAGLI(OID,Nome,Qta),
      OID REFERENCES ORDINI,
      Nome REFERENCES PIZZE;
-- Importo può essere minore della somma Prezzo*Qta se c'è uno sconto.
-- Qta è di tipo INT, Prezzo e Importo sono di tipo DEC(6,2).
```

si esprimano in algebra relazionale le seguenti interrogazioni:

**1.1) [1 p.]** I clienti che in almeno un ordine del 2022 hanno preso almeno 3 pizze Margherita




**1.2) [2 p.]** I dati degli ordini in cui Pippo ha ordinato almeno una Margherita e una Marinara



**Sistemi Informativi T**
**24 gennaio 2022**
**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 ordine di importo superiore ai 40€ e in cui è stato applicato uno sconto, l'importo di tale sconto

```
SELECT   O.OID, (SUM(D.QTA*P.PREZZO) - O.IMPORTO) AS SCONTO
FROM     ORDINI O, DETTAGLI D, PIZZE P
WHERE    O.OID = D. OID
AND      D.NOME = P.NOME
AND      O.IMPORTO > 40
GROUP BY O.OID, O.IMPORTO
HAVING   SUM(D.QTA*P.PREZZO) > O.IMPORTO;

-- E' necessario raggruppare anche su IMPORTO per poterlo usare
-- nelle clausole SELECT e HAVING
```

**2.2) [3 p.]** Considerando solo gli ordini di almeno 3 pizze, non tutte uguali, l'ordine e il relativo cliente in cui, considerando l'importo pagato, il costo medio di una pizza è stato massimo

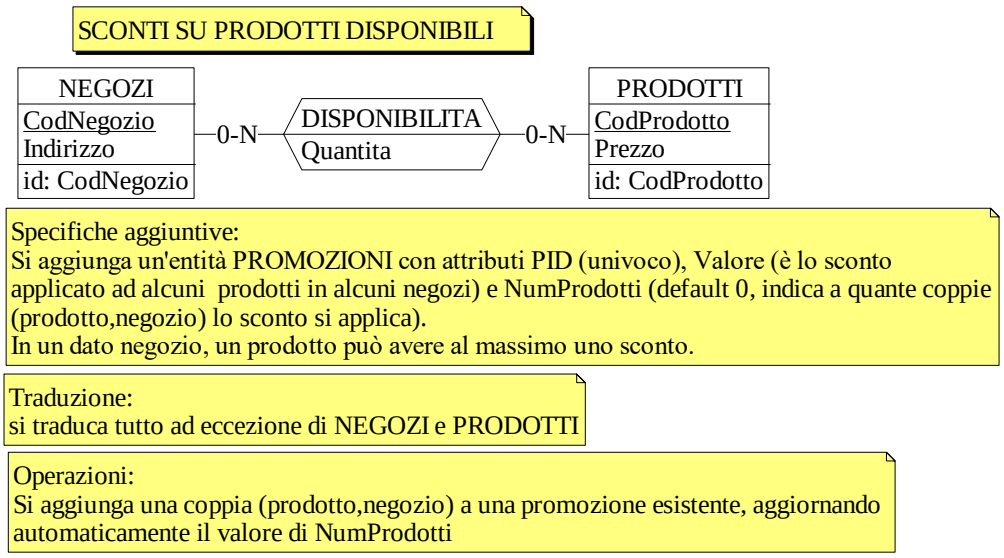
```
WITH ORDINI3 (OID,CLIENTE,COSTOMEDIO) AS (
    SELECT   O.OID,O.CLIENTE,DEC(O.IMPORTO/SUM(D.QTA),6,2)
    FROM     ORDINI O, DETTAGLI D
    WHERE    O.OID = D. OID
    GROUP BY O.OID, O.IMPORTO, O.CLIENTE
    HAVING   SUM(D.QTA) >= 3
    AND      COUNT(*) > 1
)
SELECT *
FROM   ORDINI3 O
WHERE  O.COSTOMEDIO >= ALL ( SELECT O1.COSTOMEDIO
                       FROM   ORDINI3 O1    );

   -- La c.t.e. seleziona i soli ordini che rispettano le specifiche e
   -- per ognuno di questi calcola il costo medio di una pizza.
   -- La seconda condizione dell'HAVING garantisce che l'ordine comprenda almeno
   -- 2 pizze diverse (il numero di pizze diverse è pari al numero di tuple
   -- in ogni gruppo)
```

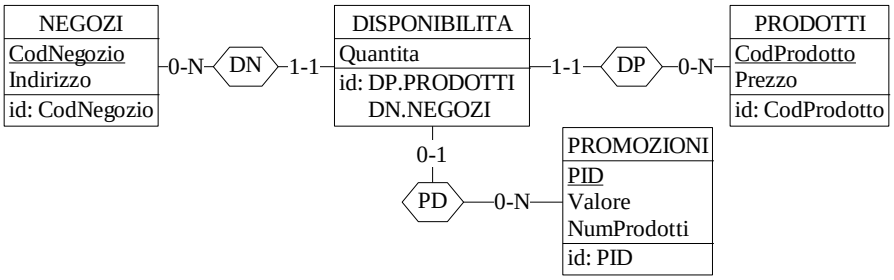
**Sistemi Informativi T**
**24 gennaio 2022**
**Risoluzione**

## 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    DISPONIBILITA
SET       PID = :idPromozione
WHERE     (CodProdotto,CodNegozio) = (:codprod,:codnegozio);
-- In alternativa si poteva anche considerare un INSERT in DISPONIBILITA

CREATE OR REPLACE TRIGGER UPDATE_NUM_PRODOTTI
AFTER UPDATE OF PID ON DISPONIBILITA
REFERENCING NEW AS N
FOR EACH ROW
UPDATE    PROMOZIONI
SET       NumProdotti = NumProdotti + 1
WHERE     PID = N.PID ;
```

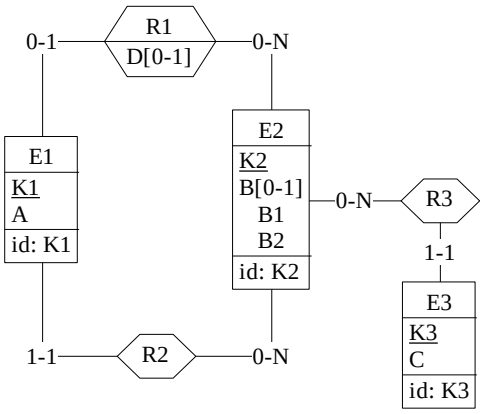
**Sistemi Informativi T**
**24 gennaio 2022**
**Risoluzione**

## 4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) nessuna associazione viene tradotta separatamente;

b) le istanze di E1 ed E3 che sono associate, tramite R2 e R3, alla stessa istanza di E2, hanno valori di A e C con A < C;




**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 E2 (
K2          INT NOT NULL PRIMARY KEY,
B1          INT,
B2          INT,
CHECK ((B1 IS NOT NULL AND B2 IS NOT NULL) OR (B1 IS NULL AND B2 IS NULL) );

CREATE TABLE E1 (
K1          INT NOT NULL PRIMARY KEY,
A           INT NOT NULL,
K2R2        INT NOT NULL REFERENCES E2,
K2R1        INT REFERENCES E2,
D           INT,
CHECK ((K2R1 IS NULL AND D IS NULL) OR K2R1 IS NOT NULL)    );
-- logicamente equivalente a (D IS NULL OR K2R1 IS NOT NULL)

CREATE TABLE E3 (
K3          INT NOT NULL PRIMARY KEY,
C           INT NOT NULL,
K2R3        INT NOT NULL REFERENCES E2                  );
```

**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 b) può essere violato inserendo in E1 o in E3

CREATE TRIGGER PUNTO_B_E1
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (    SELECT *
                   FROM E3
                   WHERE N.K2R2 = E3.K2R3
                   AND     N.A >= E3.C    ) )
SIGNAL SQLSTATE '70001' ('La tupla inserita in E1 non rispetta il vincolo del punto b)! ');

CREATE TRIGGER PUNTO_B_E3
BEFORE INSERT ON E3
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (     SELECT *
                   FROM E1
                   WHERE N.K2R3 = E1.K2R2
                   AND     N.C <= E1.A    ) )
SIGNAL SQLSTATE '70002' ('La tupla inserita in E3 non rispetta il vincolo del punto b)! ');
```