Sistemi Informativi T
4 febbraio 2022
Risoluzione

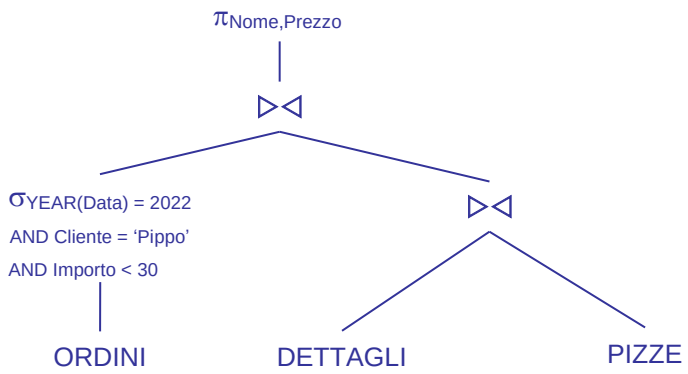
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

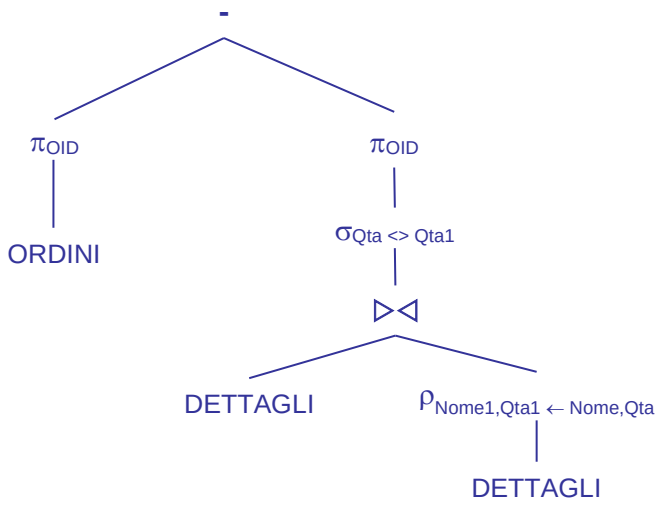
```
PIZZE(Nome,Prezzo);
ORDINI(OID,Data,Importo,Cliente);
DETTAGLI(OID,Nome,Qta),
      OID REFERENCES ORDINI,
      Nome REFERENCES PIZZE;
-- Importo può essere minore della somma Prezzo*Qta se c'è uno sconto.
-- Qta è di tipo INT, Prezzo e Importo sono di tipo DEC(6,2).
-- Un cliente non fa mai più di un ordine al giorno (ossia vale il vincolo
-- UNIQUE(Cliente,Data))
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] Le pizze, e il relativo prezzo, ordinate nel 2022 da Pippo in ordini di importo minore di 30€



1.2) [2 p.] Gli identificativi degli ordini in cui le pizze sono state ordinate tutte nella stessa quantità



L'operando destro della differenza calcola gli identificativi degli ordini in cui ci sono almeno 2 pizze ordinate in quantità diversa. Il risultato include anche gli ordini con un solo tipo di pizza, possibilità non esclusa dalla specifica della query

Sistemi Informativi T
4 febbraio 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 cliente, le pizze che ha ordinato almeno 2 volte di seguito (ossia in ordini consecutivi)

```
SELECT   DISTINCT O1.CLIENTE, D1.NOME AS PIZZA
FROM     ORDINI O1, ORDINI O2, DETTAGLI D1, DETTAGLI D2
WHERE    O1.CLIENTE = O2.CLIENTE -- stesso cliente
AND      O1.DATA < O2.DATA       -- ordini diversi, O1 fatto prima
AND      O1.OID = D1. OID
AND      O2.OID = D2.OID
AND      D1.NOME = D2.NOME       -- stessa pizza
AND      NOT EXISTS (SELECT *    -- ordini consecutivi
                     FROM   ORDINI O
                     WHERE  O.CLIENTE = O1.CLIENTE
                     AND    O1.DATA < O.DATA
                     AND    O.DATA < O2.DATA);
```

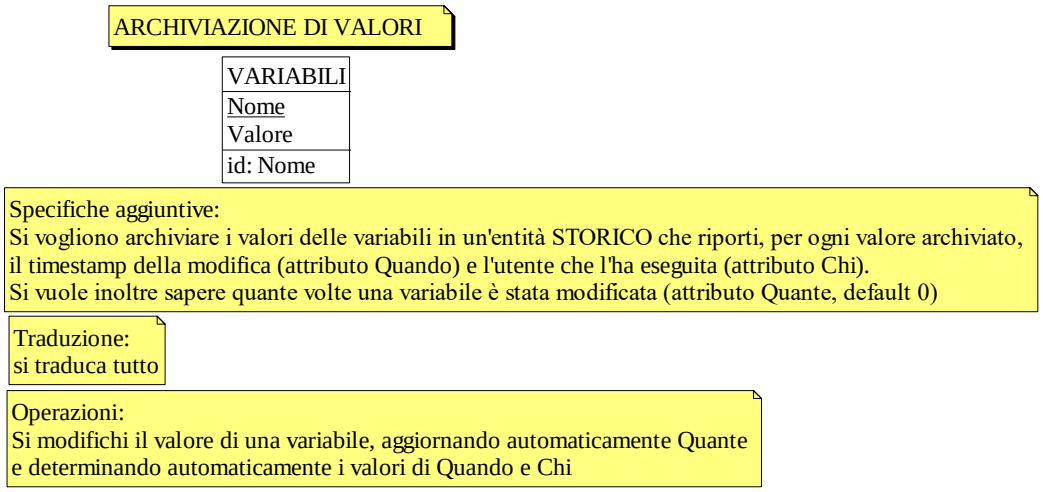
2.2) [3 p.] Le coppie di clienti che, ogni volta che hanno fatto un ordine lo stesso giorno, hanno sempre preso lo stesso numero totale di pizze

```
WITH NUMPIZZE (CLIENTE,DATA,TOTPIZZE) AS (
   SELECT O.CLIENTE,O.DATA,SUM(D.QTA)
   FROM   ORDINI O, DETTAGLI D
   WHERE  O.OID = D. OID
   GROUP BY O.CLIENTE, O.DATA                )
SELECT   N1.CLIENTE AS CLIENTE1, N2.CLIENTE AS CLIENTE2
FROM     NUMPIZZE N1, NUMPIZZE N2
WHERE    N1.CLIENTE < N2.CLIENTE
AND      N1.DATA = N2.DATA
   EXCEPT
SELECT   N1.CLIENTE AS CLIENTE1, N2.CLIENTE AS CLIENTE2
FROM     NUMPIZZE N1, NUMPIZZE N2
WHERE    N1.CLIENTE < N2.CLIENTE
AND      N1.DATA = N2.DATA
AND      N1.TOTPIZZE <> N2.TOTPIZZE;

-- La c.t.e. calcola il numero di pizze di ogni ordine.
-- La soluzione consiste nel sottrarre dall'insieme delle coppie di utenti
-- con almeno un ordine in una stessa data quelle coppie che in almeno
-- una data hanno ordinato un numero diverso di pizze
```

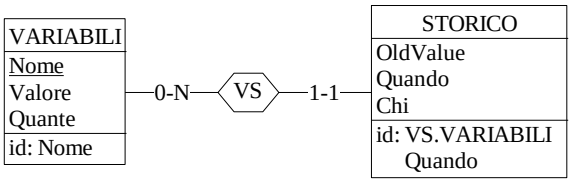
Sistemi Informativi T
4 febbraio 2022
Risoluzione

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.

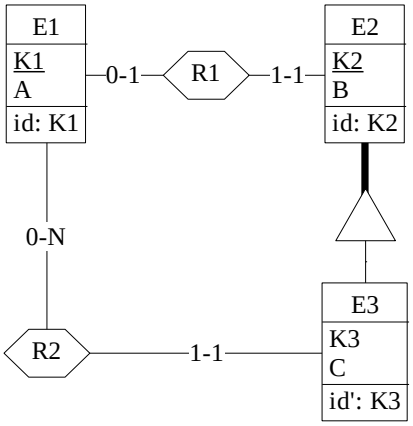
```
CREATE OR REPLACE TRIGGER ARCHIVIA_E_AGGIORNA
AFTER UPDATE OF Valore ON VARIABILI
REFERENCING OLD AS O
FOR EACH ROW
BEGIN ATOMIC
INSERT INTO STORICO(Nome,OldValue,Quando,Chi)
VALUES (O.Nome,O.Valore,CURRENT TIMESTAMP,CURRENT USER);
UPDATE    VARIABILI
SET       Quante = Quante + 1
WHERE     Nome = O.Nome ;
END
-- possibili anche 2 trigger separati

UPDATE VARIABILI
SET       Valore = :newValue
WHERE     Nome = :nomeVar ;
```

Sistemi Informativi T
4 febbraio 2022
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) le entità E2 ed E3 vengono tradotte insieme;
b) nessuna associazione viene tradotta separatamente;
c) le istanze di E1 che partecipano a R1 non possono essere associate, tramite R2, a istanze di E3;



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                                  );

CREATE TABLE E2 (
K2          INT NOT NULL PRIMARY KEY,
B           INT NOT NULL,
K1R1        INT NOT NULL REFERENCES E1 UNIQUE,        -- E1 partecipa a R1 con card-max = 1
TIPO3       SMALLINT NOT NULL CHECK (TIPO3 IN (2,3)), -- TIPO3 = 3 se appartiene anche a E3
K3          INT,
C           INT,
K1R2        INT REFERENCES E1,
CONSTRAINT E3 CHECK ((TIPO3 = 2 AND K3 IS NULL AND C IS NULL AND K1R2 IS NULL)
            OR (TIPO3 = 3 AND K3 IS NOT NULL AND C IS NOT NULL AND K1R2 IS NOT NULL)),
CONSTRAINT PUNTO_C CHECK (K1R1 <> K1R2)     );          -- punto c) da completare con trigger
```

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 K3_UNIQUE
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (     SELECT *
                    FROM    E2
                    WHERE N.K3 = E2.K3  )  )
SIGNAL SQLSTATE '70001' ('Il valore di K3 non può essere duplicato! ');

-- Il vincolo al punto c) può essere violato inserendo in E2 o una tupla che referenzia tramite R1 un'istanza di
-- E1 che già partecipa a R2 (N.K1R1 = E2.K1R2), o una tupla che referenzia tramite R2 un'istanza di E1
-- che già partecipa a R1 (N.K1R2 = E2.K1R1).
-- Il caso N.K1R1 = N.K1R2 è gestito mediante CHECK.

CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (      SELECT *
                    FROM E2
                    WHERE N.K1R1 = E2.K1R2
                    OR        N.K1R2 = E2.K1R1    )  )
SIGNAL SQLSTATE '70002' ('La tupla inserita in E2 non rispetta il vincolo del punto c)! ');
```