Sistemi Informativi T
26 giugno 2023
Risoluzione

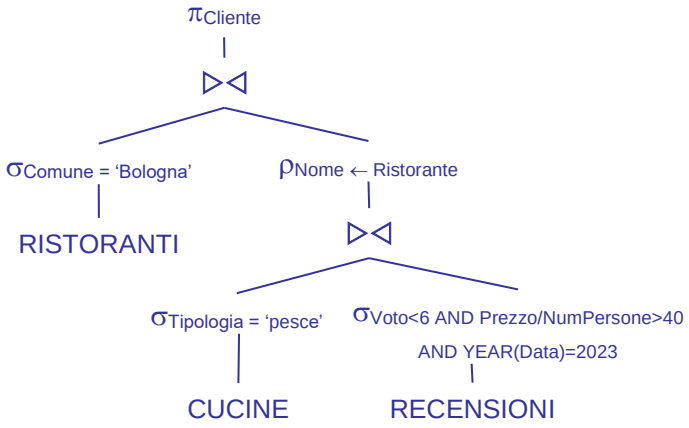
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

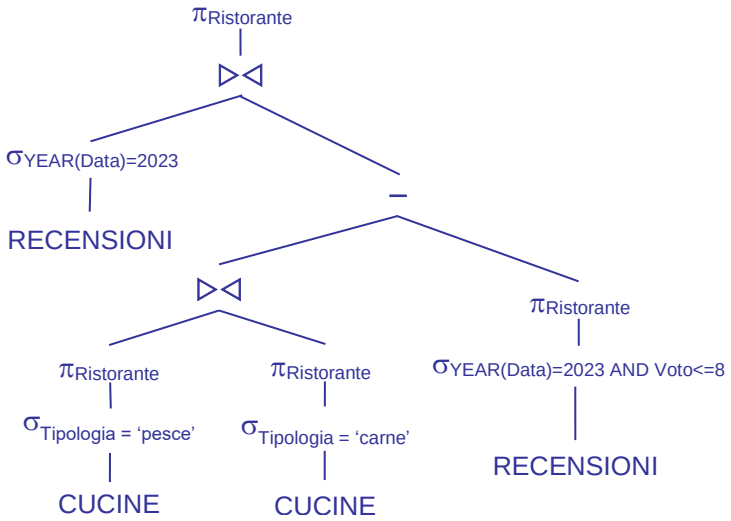
```
RISTORANTI(Nome,Via,Comune);
CUCINE(Ristorante,Tipologia),
      Ristorante REFERENCES RISTORANTI;
RECENSIONI(Ristorante,Cliente,Data,NumPersone,Prezzo,Voto),
      Ristorante REFERENCES RISTORANTI;
-- NumPersone è di tipo INT > 0.
-- Prezzo è di tipo DEC(6,2): totale pagato per NumPersone
-- Voto è di tipo INT, valori da 1 a 10.
-- Tipologia: pizza, pesce, carne, cinese, ecc.
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] Nomi dei clienti che nel 2023 hanno recensito almeno un ristorante di pesce a Bologna dando un voto minore di 6 e spendendo più di 40€ a persona



1.2) [2 p.] I ristoranti che fanno sia carne che pesce e che nel 2023 hanno avuto solo recensioni con voto maggiore di 8 (almeno una)



La differenza trova i ristoranti che fanno sia carne che pesce, e senza recensioni con voto <= 8 nel 2023. Il secondo join garantisce che il ristorante abbia almeno una recensione nel 2023

Sistemi Informativi T
26 giugno 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 cliente che ha recensito almeno 2 ristoranti diversi, il voto medio assegnato ad ogni ristorante in cui ha pranzato

```
SELECT   RC.CLIENTE, RC.RISTORANTE, DEC(AVG(RC.VOTO*1.0),4,2) AS MEDIA_VOTI
FROM     RECENSIONI RC
WHERE    RC.CLIENTE IN ( SELECT  RC1.CLIENTE  -- >= 2 ristoranti diversi
                         FROM    RECENSIONI RC1
                         GROUP BY RC1.CLIENTE
                         HAVING  COUNT(DISTINCT RC1.RISTORANTE) >= 2)
GROUP BY RC.CLIENTE, RC.RISTORANTE;

-- Si noti che la stessa clausula HAVING nel blocco esterno, ovvero:
-- HAVING COUNT(DISTINCT RC.RISTORANTE) >= 2
-- non avrebbe senso, perché per definizione ogni gruppo consiste di un
-- singolo ristorante (e singolo cliente)
```

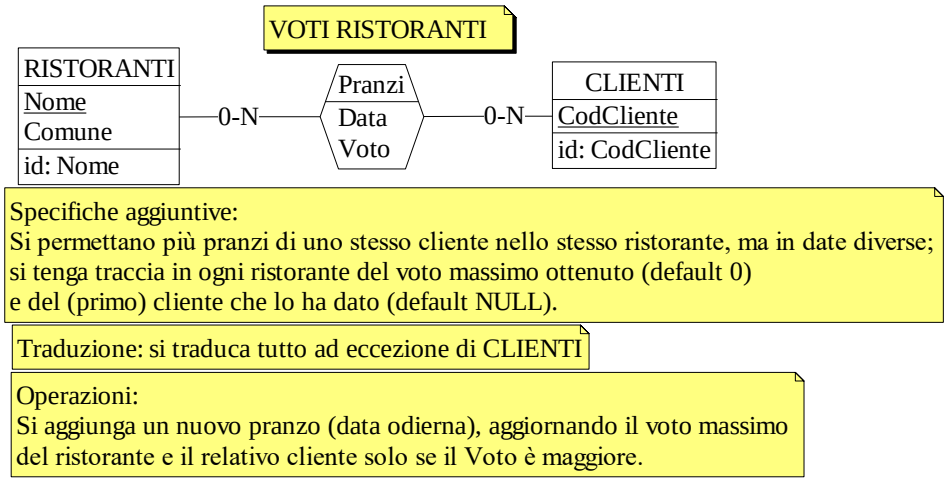
2.2) [3 p.] I dati del ristorante in cui la media del prezzo a persona è minima, fornendo anche quanti tipi di cucina propone

```
WITH
MEDIA_E_TIPI(RISTORANTE,MEDIA_PP,NUM_TIPI) AS
(SELECT RC.RISTORANTE, AVG(RC.PREZZO/RC.NUMPERSONE),
                                        COUNT(DISTINCT C.TIPOLOGIA)
 FROM    RECENSIONI RC, CUCINE C
 WHERE   C.RISTORANTE = RC.RISTORANTE
 GROUP BY RC.RISTORANTE
)
SELECT  R.*, DEC(M.MEDIA_PP,6,2) AS MEDIA_PREZZO_PERSONA, M.NUM_TIPI
FROM    MEDIA_E_TIPI M, RISTORANTI R
WHERE   R.NOME = M.RISTORANTE
AND     M.MEDIA_PP = ( SELECT MIN(M1.MEDIA_PP)
                       FROM   MEDIA_E_TIPI M1 );

-- La c.t.e. calcola il prezzo medio per persona e il numero di tipi
-- di cucina (notare la forma COUNT(DISTINCT …)
.
```

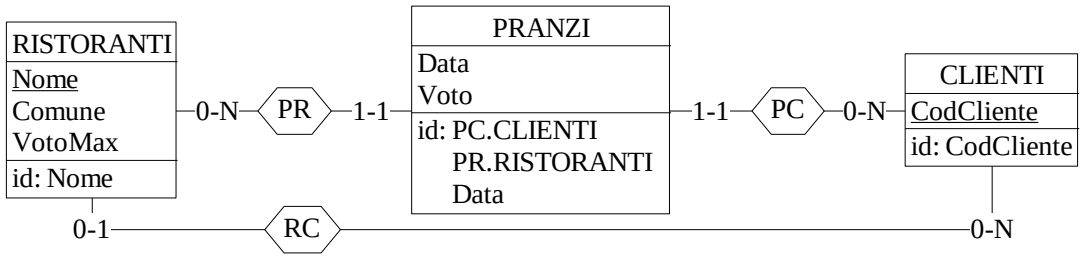
Sistemi Informativi T
26 giugno 2023
Risoluzione

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 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.

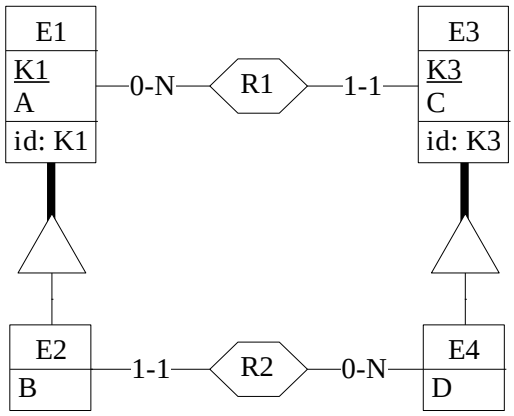
```
CREATE OR REPLACE TRIGGER AGGIORNA_VOTO_MAX_E_CLIENTE
AFTER INSERT ON PRANZI
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.Voto > (SELECT VotoMax FROM RISTORANTI WHERE Nome = N.Nome))
UPDATE     RISTORANTI
SET        VotoMax = N.Voto,
           CodCliente = N.CodCliente
WHERE      Nome = N.Nome                          ;

INSERT INTO PRANZI VALUES (:NomeRistorante,:CodCliente,CURRENT DATE,:Voto);
```

Sistemi Informativi T
26 giugno 2023
Risoluzione

Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) le entità E1 ed E2 vengono tradotte assieme;
b) le entità E3 ed E4 vengono tradotte assieme;
c) nessuna associazione viene tradotta separatamente;
d) un'istanza di E2 non è mai associata, tramite R2, a un'istanza di E4 che è associata, tramite R1, a un'istanza di E1 con A > 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 E1 (
K1          INT NOT NULL PRIMARY KEY,
A           INT NOT NULL,
TIPO2       SMALLINT NOT NULL CHECK (TIPO2 IN (1,2)), -- 2: istanza di E2
B           INT,
K3R2        INT,
CONSTRAINT E2 CHECK ((TIPO2 = 1 AND B IS NULL AND K3R2 IS NULL) OR
                     (TIPO2 = 2 AND B IS NOT NULL AND K3R2 IS NOT NULL))      );

CREATE TABLE E3 (
K3          INT NOT NULL PRIMARY KEY,
C           INT NOT NULL,
K1R1        INT NOT NULL REFERENCES E1,
TIPO4       SMALLINT NOT NULL CHECK (TIPO4 IN (3,4)), -- 4: istanza di E4
D           INT,
CONSTRAINT E4 CHECK ((TIPO4 = 3 AND D IS NULL) OR
                     (TIPO4 = 4 AND D IS NOT NULL))          );

ALTER TABLE E1 ADD CONSTRAINT FK_E4 FOREIGN KEY (K3R2) 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**

```
-- Trigger che garantisce che R2 referenzi un'istanza di E4
CREATE OR REPLACE TRIGGER R2_E4
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS ( SELECT *
                FROM E3
                WHERE N.K3R2 = E3.K3
                AND E3.TIPO4 = 3 ) )
SIGNAL SQLSTATE '70001' ('La tupla referenzia una tupla che non appartiene a E4!');

CREATE OR REPLACE TRIGGER PUNTO_D
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS ( SELECT   *
                FROM     E3, E1
                WHERE    N.K3R2 = E3.K3
                AND      E3.K1R1 = E1.K1
                AND      E1.A > 10        ) )
SIGNAL SQLSTATE '70002' ('La tupla inserita referenzia tramite R2 e R1 un"istanza di E1 con A > 10! ');
```