Sistemi Informativi T
8 settembre 2023
Risoluzione

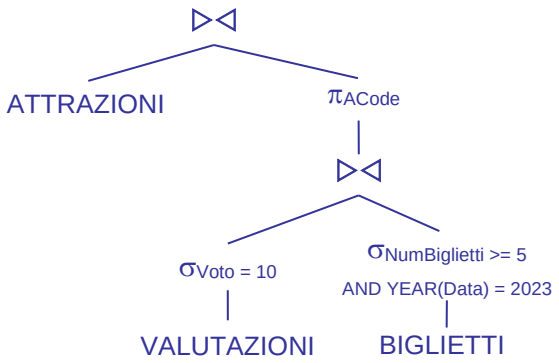
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

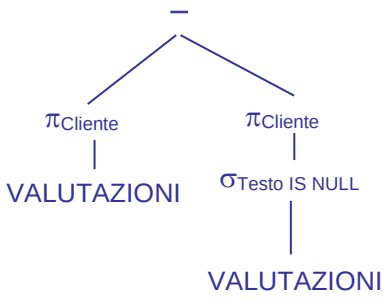
```
ATTRAZIONI(ACode,Nome,Prezzo);
BIGLIETTI(ACode,Cliente,NumBiglietti,Data),
      ACode REFERENCES ATTRAZIONI;
VALUTAZIONI(ACode,Cliente,Voto,Testo*),
      (ACode,Cliente) REFERENCES BIGLIETTI;
-- Prezzo è di tipo DEC(6,2).
-- Voto è di tipo INT, valori da 1 a 10.
-- NumBiglietti è di tipo INT (quanti biglietti un cliente ha comprato).
-- Testo ammette valori nulli.
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I dati delle attrazioni che nel 2023 hanno ottenuto almeno un 10 da un cliente che per quell'attrazione ha comprato almeno 5 biglietti



1.2) [2 p.] I clienti che hanno sempre lasciato valutazioni accompagnate da un testo



L'operando sinistro della differenza trova tutti i clienti con almeno una valutazione. Sottraendo a questi quelli che in almeno una valutazione non hanno inserito alcun testo si ottiene il risultato

Sistemi Informativi T
8 settembre 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 dati delle attrazioni che in un giorno hanno incassato complessivamente almeno 1000€, considerando solo i clienti che hanno valutato l'attrazione inserendo anche un testo

```
SELECT   A.*
FROM     ATTRAZIONI A, BIGLIETTI B
WHERE    A.ACODE = B.ACODE
AND      (B.ACODE,B.CLIENTE) IN
               (     SELECT V.ACODE, V.CLIENTE
                     FROM   VALUTAZIONI V
                     WHERE  V.TESTO IS NOT NULL  )
GROUP BY A.ACODE, A.NOME, A.PREZZO, B.DATA
HAVING   A.PREZZO*SUM(B.NUMBIGLIETTI) >= 1000 ;

-- Bisogna raggruppare su tutti gli attributi di ATTRAZIONI e anche su DATA.
-- Se una coppia (ACODE, CLIENTE) compare in VALUTAZIONI è garantito che la
-- valutazione esiste, bisogna quindi solo verificare che il testo sia
-- presente
```

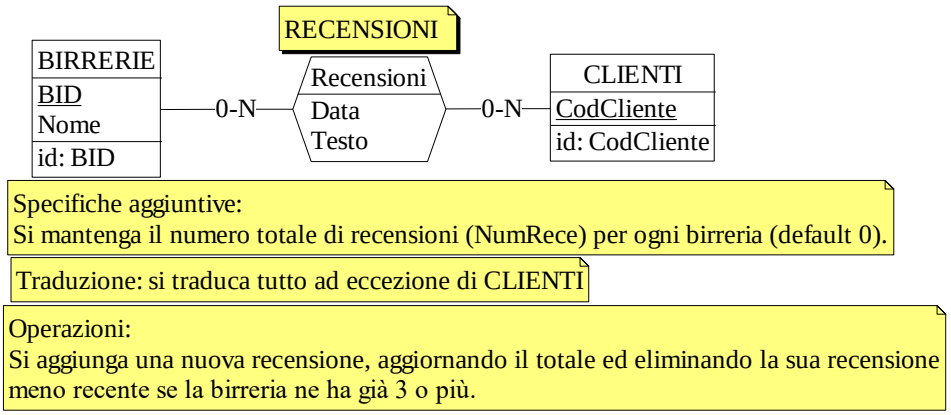
2.2) [3 p.] I clienti che in uno stesso giorno hanno acquistato biglietti per 2 o più attrazioni, e che per ognuna di queste hanno lasciato una valutazione positiva (Voto >= 6)

```
WITH BIG_VAL(ACODE,CLIENTE,DATA,VOTO)
AS (
SELECT   B.ACODE,B.CLIENTE,B.DATA,V.VOTO
FROM     BIGLIETTI B LEFT JOIN VALUTAZIONI V ON
         (B.ACODE,B.CLIENTE) = (V.ACODE, V.CLIENTE) AND V.VOTO >= 6
    )
SELECT   BV.CLIENTE --, BV.DATA, count(*), count(BV.VOTO)
FROM     BIG_VAL BV
GROUP BY BV.CLIENTE, BV.DATA
HAVING   COUNT(*) >= 2
AND      COUNT(*) = COUNT(BV.VOTO) ;

-- La c.t.e. sfrutta l'outer join per produrre una table in cui voti negativi
-- o mancanti (se non esiste la relativa valutazione) vengono rimpiazzati con
-- dei NULL.
-- Il predicato COUNT(*) = COUNT(BV.VOTO) è quindi soddisfatto se e solo se
-- per tutte le attrazioni visitate da un cliente in un giorno il voto
-- è definito, ovvero >= 6.
```

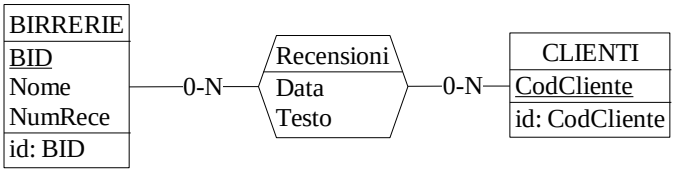
Sistemi Informativi T
8 settembre 2023
Risoluzione

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 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.

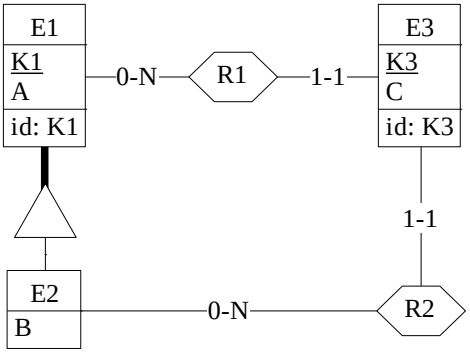
```
CREATE OR REPLACE TRIGGER AGGIORNA_NUM_RECE
AFTER INSERT ON RECENSIONI
REFERENCING NEW AS N
FOR EACH ROW
UPDATE     BIRRERIE
SET        NumRece = NumRece + 1
WHERE      BID = N.BID                                  ;

CREATE OR REPLACE TRIGGER MAX_3_RECE
AFTER INSERT ON RECENSIONI
REFERENCING NEW AS N
FOR EACH ROW
WHEN (3 < (SELECT COUNT(*) FROM RECENSIONI WHERE BID = N.BID))
DELETE FROM RECENSIONI
WHERE      BID = N.BID
AND        Data = ( SELECT MIN(Data)
                    FROM   RECENSIONI
                    WHERE  BID = N.BID )                  ;

INSERT INTO RECENSIONI VALUES (:BID,:Cliente,:Data,:Testo);
```

Sistemi Informativi T
8 settembre 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) nessuna associazione viene tradotta separatamente;
c) un'istanza di E3 è associata tramite R1 e R2 a istanze diverse di E1;
d) le due istanze di E1 a cui un'istanza di E3 è associata tramite R1 e R2 hanno A + B < 20 (A tramite R1 e B tramite R2);



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,
CONSTRAINT E2 CHECK ((TIPO2 = 1 AND B IS NULL) OR (TIPO2 = 2 AND B IS NOT NULL))  );

CREATE TABLE E3 (
K3          INT NOT NULL PRIMARY KEY,
C           INT NOT NULL,
K1R1        INT NOT NULL REFERENCES E1,
K1R2        INT NOT NULL REFERENCES E1,
CONSTRAINT PUNTO_C CHECK (K1R1 <> K1R2)     );
```

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 E2
CREATE OR REPLACE TRIGGER R2_E2
BEFORE INSERT ON E3
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS ( SELECT *
                FROM E1
                WHERE N.K1R2 = E1.K1
                AND E1.TIPO2 = 1 ) )
SIGNAL SQLSTATE '70001' ('La tupla referenzia una tupla che non appartiene a E2!');

CREATE OR REPLACE TRIGGER PUNTO_D
BEFORE INSERT ON E3
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS ( SELECT   *
                FROM     E1, E1 E2
                WHERE    N.K1R1 = E1.K1
                AND      N.K1R2 = E2.K1
                AND      E1.A + E2.B >= 20          ) )
SIGNAL SQLSTATE '70002' ('La tupla inserita referenzia tramite R2 e R1 istanze di E1 con A + B >= 20! ');
```