Sistemi Informativi T
30 giugno 2021
Risoluzione

1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

```
SEZIONI(Num,NElettori);
RISULTATI(NumSezione,Partito,Voti),
      NumSezione REFERENCES SEZIONI;
-- NElettori e Voti sono interi, rispettivamente >0 e >=0.
-- RISULTATI riporta per ogni sezione i Voti ottenuti da ogni Partito
-- (eventualmente 0).
-- La somma dei Voti in una sezione ("voti validi") può essere minore
-- di NElettori, in quanto non tutti votano, oppure votano scheda
-- bianca o nulla.
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I partiti che in almeno una sezione con più di 500 elettori hanno ottenuto almeno il 30% dei voti sul totale degli elettori di quella sezione

1.2) [2 p.] I partiti che in tutte le sezioni con meno di 500 elettori hanno ottenuto almeno 100 voti

L'esercizio si poteva anche risolvere con una semplice differenza tra tutti i partiti e quelli che, in almeno una sezione con meno di 500 elettori, hanno preso meno di 100 voti. Ciò è possibile perché RISULTATI contiene tutte le combinazioni (Partito,NumSezione)

Sistemi Informativi T
30 giugno 2021
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 partito, le sezioni in cui quel partito ha ottenuto un numero di voti maggiore della somma dei voti di tutti gli altri partiti nella stessa sezione

```
SELECT   R.Partito, R.NumSezione
FROM     RISULTATI R
WHERE    R.Voti > (  SELECT SUM(R1.Voti)
                     FROM   RISULTATI R1
                     WHERE  R1.NumSezione = R.NumSezione
                     AND    R1.Partito <> R.Partito ) ;
```

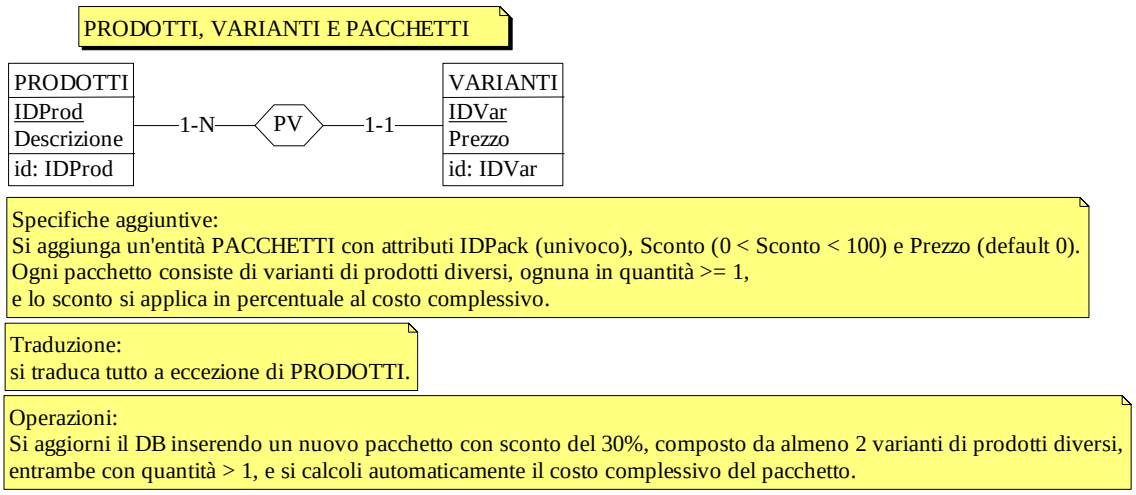
2.2) [3 p.] La sezione in cui il numero di voti validi è stato il maggiore in percentuale rispetto al numero di elettori in quella sezione

```
WITH PERC_VALIDI(NumSezione,Perc) AS (
   SELECT   R.NumSezione, DEC(SUM(R.Voti)*1.0/S.NElettori,4,3)
   FROM     RISULTATI R, SEZIONI S
   WHERE    R.NumSezione = S.Num
   GROUP BY R.NumSezione,S.NElettori )
SELECT   *
FROM     PERC_VALIDI P
WHERE    P.Perc >= ( SELECT MAX(P1.Perc)
                     FROM   PERC_VALIDI P1 );
-- La c.t.e. calcola le percentuali di voti validi in tutte le sezioni.
-- E' fondamentale operare un cast, implicito nella soluzione proposta,
-- (moltiplicazione per 1.0) prima di eseguire la divisione
```

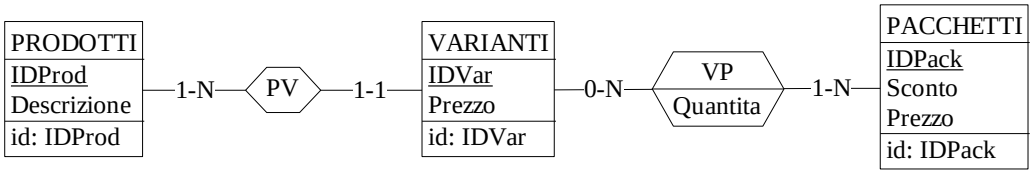
Sistemi Informativi T
30 giugno 2021
Risoluzione

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 produca uno schema ESE3-modificato 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.

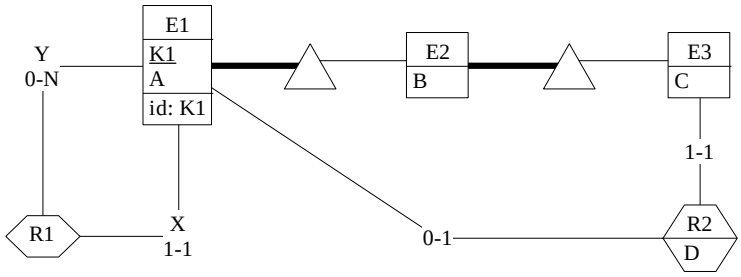
```
INSERT INTO PACCHETTI VALUES ('PK001',30,DEFAULT);

CREATE OR REPLACE TRIGGER VARIANTE_INVALIDA
BEFORE INSERT ON VP
REFERENCING NEW AS NewV
FOR EACH ROW
WHEN (EXISTS (     SELECT * FROM VP, VARIANTI V1, VARIANTI V2
                   WHERE NewV.IDPack = VP.IDPack
                   AND      NewV.IDVar = V1.IDVar
                   AND      VP.IDVar = V2.IDVar
                   AND      V1.IDProd = V2.IDProd ) )
SIGNAL SQLSTATE '70001' ('Uno stesso pacchetto deve avere varianti di prodotti diversi!');

CREATE OR REPLACE TRIGGER CALCOLA_PREZZO_PACCHETTO
AFTER INSERT ON VP
REFERENCING NEW AS NewV
FOR EACH ROW
UPDATE PACCHETTI
SET Prezzo = Prezzo +  (100-Sconto)/100.0*NewV.Quantita*       -- 100.0 per eseguire il cast implicito
                              ( SELECT V.Prezzo FROM VARIANTI V
                                WHERE V.IDVar = NewV.IDVar )
WHERE IDPack = NewV.IDPack;

INSERT INTO VP VALUES ('PK001','V0001',3); -- OK
INSERT INTO VP VALUES ('PK001','V0002',2); -- NO! (stesso prodotto)
INSERT INTO VP VALUES ('PK001','V0003',2); -- OK);
```

Sistemi Informativi T
30 giugno 2021
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) le entità E1 ed E2 vengono tradotte assieme, e separatamente da E3;
b) nessuna associazione viene tradotta separatamente; in particolare, l'associazione R2 viene tradotta assieme a E3;
c) un'istanza di E1 che partecipa a R1 con il ruolo Y è associata a istanze di E1 con valori di A tutti diversi tra loro.



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,
K1Y         INT NOT NULL REFERENCES E1,
TIPO2       SMALLINT NOT NULL CHECK (TIPO2 in (1,2)),    -- se 2 l'istanza appartiene anche a E2
B           INT,
CONSTRAINT E2 CHECK (  (TIPO2 = 1 AND B IS NULL) OR
                       (TIPO2 = 2 AND B IS NOT NULL))          );

CREATE TABLE E3 (
K1          INT NOT NULL PRIMARY KEY REFERENCES E1,
C           INT NOT NULL,
D           INT NOT NULL,
K1R2        INT NOT NULL UNIQUE REFERENCES E1                   );
```

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 la tupla inserita in E3 sia un'istanza di E2
CREATE OR REPLACE TRIGGER K1_E2
BEFORE INSERT ON E3
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( NOT EXISTS ( SELECT *
                    FROM    E1
                    WHERE N.K1 = E1.K1
                    AND     E1.TIPO2 = 2   ) )
SIGNAL SQLSTATE '70001' ('La tupla inserita in E3 deve appartenere anche a E2!');

-- Trigger che garantisce il rispetto del vincolo al punto c)
CREATE OR REPLACE TRIGGER PUNTO_C
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS ( SELECT *
                FROM    E1
                WHERE N.K1Y = E1.K1Y
                AND     N.A = E1.A ) )
SIGNAL SQLSTATE '70002' ('La tupla inserita non rispetta il vincolo del punto c! ');
```