Sistemi Informativi T
30 gennaio 2026
Risoluzione

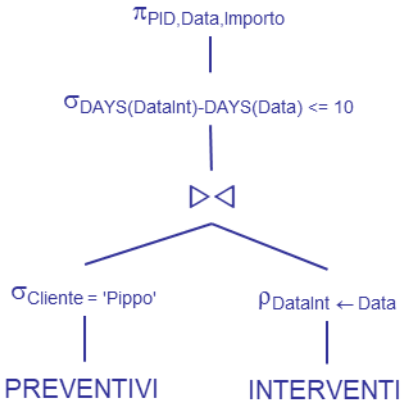
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

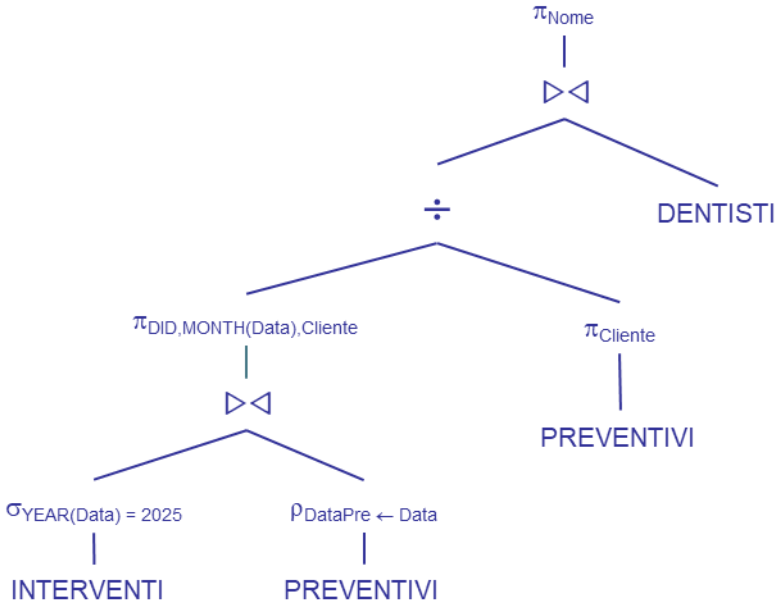
```
DENTISTI(DID,Nome,DataNascita);
PREVENTIVI(PID,Cliente,Data,Importo);
INTERVENTI(CodI,PID,DID,Data,NumDente*),
      DID REFERENCES DENTISTI,
      PID REFERENCES PREVENTIVI;
-- Importo è di tipo DEC(8,2)
-- NumDente è un intero (valori compresi tra 1 e 32) che identifica
-- uno specifico dente. Se è NULL allora l'intervento è di tipo
-- generico (ad es. ablazione tartaro)
-- Ogni intervento fa riferimento a uno specifico preventivo, il quale
-- può richiedere anche più interventi
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I dati dei preventivi del cliente Pippo in cui almeno un intervento è stato eseguito entro 10 giorni dalla data del preventivo



1.2) [2 p.] I nomi dei dentisti che hanno fatto interventi a tutti i clienti in uno stesso mese del 2025



La divisione trova le coppie (DID, mese del 2025) accoppiate a tutti i clienti.

Sistemi Informativi T
30 gennaio 2026
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 codici delle coppie di interventi consecutivi fatti su uno stesso dente per lo stesso preventivo

```
SELECT      I1.CodI, I2.CodI
FROM        INTERVENTI I1, INTERVENTI I2
WHERE       I1.PID = I2.PID
AND         I1.NumDente = I2.NumDente
AND         I1.DATA < I2.DATA
AND NOT EXISTS                                      -- interventi consecutivi
            ( SELECT *
              FROM    INTERVENTI I3
              WHERE   I3.PID = I1.PID
              AND     I1.DATA < I3.DATA
              AND     I3.DATA < I2.DATA ) ;
```

2.2) [3 p.] Per ogni dentista, fornendo identificativo e nome, il numero complessivo di interventi fatti su clienti che non hanno mai fatto interventi con altri dentisti, includendo anche i dentisti per cui tale numero è 0

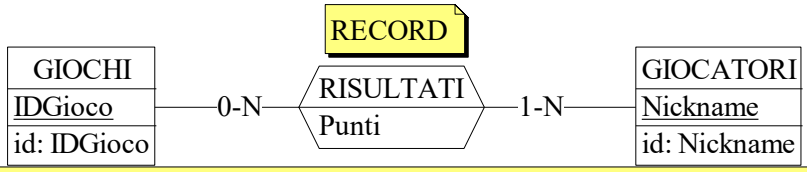
```
WITH NUM_INTERVENTI(DID,NOME,NINTERVENTI) AS (
     SELECT  D.DID, D.NOME, COUNT(*)
     FROM   (INTERVENTI I JOIN PREVENTIVI P ON (I.PID = P.PID))
                    JOIN DENTISTI D ON (I.DID = D.DID)
     WHERE NOT EXISTS
           ( SELECT *
             FROM    INTERVENTI I1 JOIN PREVENTIVI P1 ON (I1.PID = P1.PID)
             WHERE P.CLIENTE = P1.CLIENTE
             AND     I1.DID <> I.DID )
     GROUP BY D.DID, D.NOME                                          )
SELECT      N.*
FROM        NUM_INTERVENTI N
UNION
SELECT      D.DID, D.NOME, 0 AS NINTERVENTI
FROM        DENTISTI D
WHERE       D.DID NOT IN
            ( SELECT DID
              FROM   NUM_INTERVENTI ) ;

-- La c.t.e. calcola quanto richiesto per i dentisti con almeno un cliente trattato in esclusiva
```

Sistemi Informativi T
30 gennaio 2026
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:

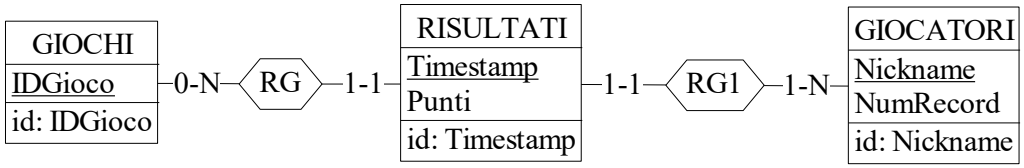


Specifiche aggiuntive: Si tenga traccia di tutti i punteggi dei giocatori, con un timestamp che identifica univocamente ogni risultato, e di quante volte (NumRecord, default 0) un giocatore è diventato primo (anche a pari merito) in qualche gioco.
NB: se un giocatore era già primo e migliora il suo punteggio NumRecord non va incrementato.

Traduzione: si traduca tutto ad eccezione di GIOCHI.

Operazioni: Si inserisca un nuovo risultato (usando il timestamp attuale) ed eventualmente si aggiorni il numero di record del giocatore.

3.1) [2 p.] Si copi lo schema ESE3-input in uno schema ESE3-modificato e si modifichi quest'ultimo 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.

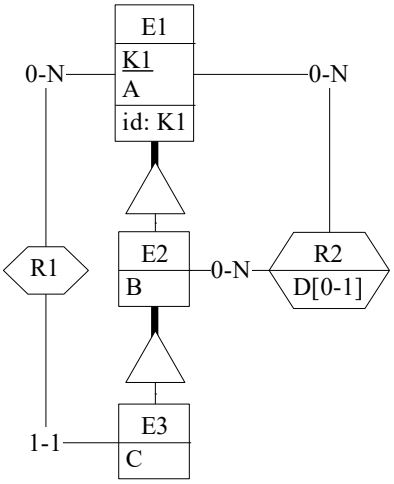
```
CREATE OR REPLACE TRIGGER NUOVO_NUM_RECORD
AFTER INSERT ON RISULTATI
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.Punti >= ( SELECT COALESCE(MAX(Punti),0)        -- record
                   FROM   RISULTATI
                   WHERE  IDGioco = N.IDGioco )
      AND N.Nickname NOT IN
                ( SELECT NickName                       -- chi deteneva il record
                  FROM   RISULTATI
                  WHERE IDGioco = N.IDGioco
                  AND     Timestamp < N.Timestamp       -- esclude il nuovo risultato
                  AND     Punti = ( SELECT MAX(Punti)
                                    FROM   RISULTATI
                                   WHERE IDGioco = N.IDGioco
                                   AND     Timestamp < N.Timestamp ) ) )
UPDATE GIOCATORI
SET       NumRecord = NumRecord + 1
WHERE  Nickname = N.Nickname ;

INSERT INTO RISULTATI VALUES
(CURRENT TIMESTAMP,:idGioco,:nickname,:punti) ;
```

Sistemi Informativi T
30 gennaio 2026
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:



a) le entità E1, E2 ed E3 vengono tradotte assieme;
b) l'associazione R1 non viene tradotta separatamente;
c) un'istanza di E3 che partecipa a R2 (dal ramo collegato a E2) non è mai associata, tramite il ramo 1-1 di R1, a un'istanza di E1 tale che A + C > 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 se istanza di E2, 1 altrimenti
B       INT,
TIPO3   SMALLINT NOT NULL CHECK (TIPO3 IN (1,3)), -- 3 se istanza di E3, 1 altrimenti
C       INT,
K1R1    INT REFERENCES E1,
CONSTRAINT E2 CHECK ((TIPO2 = 1 AND B IS NULL) OR (TIPO2 = 2 AND B IS NOT NULL)),
CONSTRAINT E3 CHECK ((TIPO3 = 1 AND C IS NULL AND K1R1 IS NULL) OR
                     (TIPO3 = 3 AND C IS NOT NULL AND K1R1 IS NOT NULL)),
CONSTRAINT E2_E3 CHECK (TIPO3 = 1 OR TIPO2 = 2) ); -- esclude il caso TIPO3 = 3 AND TIPO2 = 1

CREATE TABLE R2 (
K1      INT NOT NULL REFERENCES E1,
K1E2    INT NOT NULL REFERENCES E1,
D       INT,
PRIMARY KEY (K1,K1E2)                    );
```

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 K1E2 referenzi un'istanza di E2
CREATE TRIGGER R2_E2
BEFORE INSERT ON R2
REFERENCING NEW AS N FOR EACH ROW
WHEN ( NOT EXISTS ( SELECT * FROM E1 WHERE N.K1E2 = K1 AND TIPO2 = 2 )    )
SIGNAL SQLSTATE '70001' ('La tupla referenziata deve appartenere a E2!');


CREATE TRIGGER PUNTO_C
BEFORE INSERT ON R2                          -- unico caso in cui il vincolo può essere violato
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( 10 < ( SELECT E1.A + E3.C
              FROM    E1 E1, E1 E3
              WHERE N.K1E2 = E3.K1 AND E3.K1R1 = E1.K1 )  )
SIGNAL SQLSTATE '70002' ('La tupla referenzia un'istanza di E3 non corretta!');

-- Il check E3.TIPO3 = 3 è ridondante perché se E3.TIPO3 = 1 allora E3.K1R1 è NULL
-- e la subquery non restituisce nulla
```