Sistemi Informativi T
12 febbraio 2026
Risoluzione

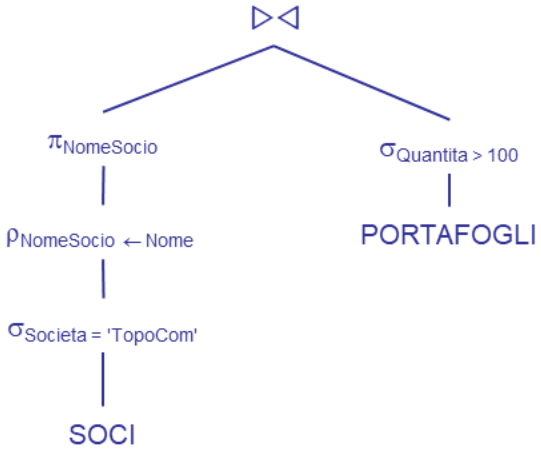
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

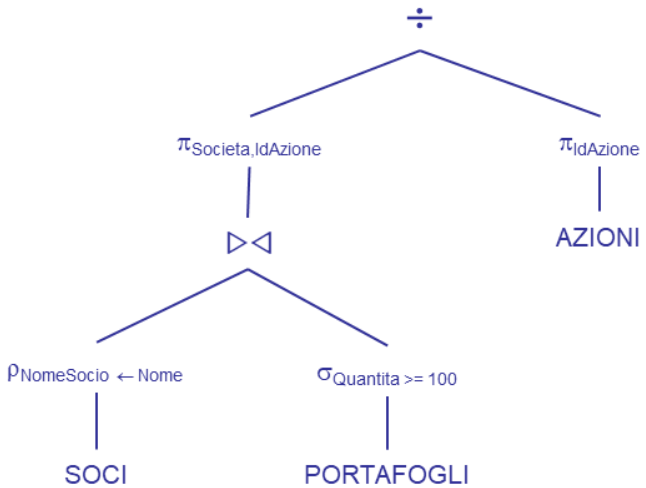
```
SOCI(Nome,Societa);
AZIONI(IdAzione,Data,Valore);
PORTAFOGLI(NomeSocio,IdAzione,Quantita),
      NomeSocio references SOCI;
-- Quantita, di tipo INT, e Valore, di tipo DEC(8,2), sono entrambi
-- maggiori di zero.
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I dati dei portafogli della società TopoCom in cui almeno un'azione è posseduta da un socio della società in quantità maggiore di 100



1.2) [2 p.] I nomi delle società che, considerando tutti i portafogli dei loro soci, detengono tutte le azioni, ognuna in quantità almeno pari a 100




```
-- Una formulazione più precisa della query è: "I nomi delle società per le quali ogni azione è
-- posseduta da almeno un socio della società in quantità almeno pari a 100". L'interpretazione per cui
-- la quantità >= 100 potrebbe risultare dalla somma delle quantità dei singoli soci non è comunque
-- compatibile con quanto esprimibile in algebra relazionale, in cui non è possibile aggregare.
```

Sistemi Informativi T
12 febbraio 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.] Per ogni azione che fa parte di almeno 2 portafogli di 2 diverse società, la variazione percentuale di valore dal primo all'ultimo giorno in cui l'azione è presente nel DB

```
SELECT      A1.IdAzione, DEC(100*(A2.Valore-A1.Valore)/A1.Valore,4,2) || '%' AS "VAR_%"
FROM        SOCI S, AZIONI A1, AZIONI A2, PORTAFOGLI P
WHERE       S.Nome = P.NomeSocio
AND         A1.IdAzione = P.IdAzione
AND         A1.IdAzione = A2.IdAzione
AND         NOT EXISTS (   SELECT  *
                           FROM    AZIONI A3
                           WHERE A3.IdAzione = A1.IdAzione
                           AND     A3.DATA < A1.DATA )
AND         NOT EXISTS (   SELECT  *
                           FROM    AZIONI A4
                           WHERE A4.IdAzione = A2.IdAzione
                           AND     A4.DATA > A2.DATA )
GROUP BY A1.IdAzione, A1.Valore, A2.Valore
HAVING      COUNT(DISTINCT S.Societa) >= 2          ;

-- La condizione nella clausola HAVING ovviamente garantisce anche che i portafogli siano almeno 2
```

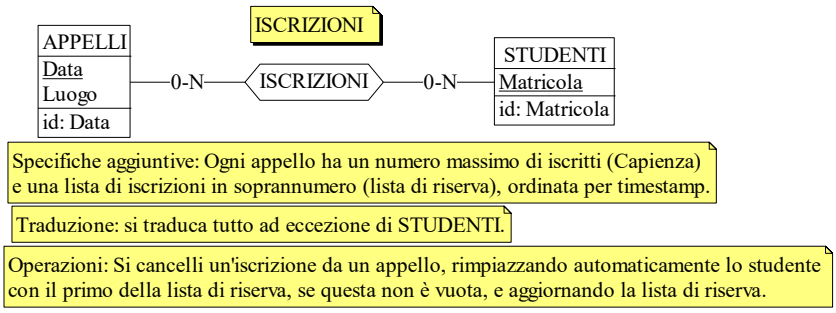
2.2) [3 p.] Per ogni società, la data e il relativo guadagno in cui il guadagno complessivo rispetto alla data precedente presente nel DB è risultato massimo

```
WITH GUADAGNI(Societa,Data,Guadagno) AS (
     SELECT  S.Societa, A2.Data, SUM(A2.Valore*P.Quantita)-SUM(A1.Valore*P.Quantita)
     FROM    SOCI S, AZIONI A1, AZIONI A2, PORTAFOGLI P
     WHERE  S.Nome = P.NomeSocio
     AND     A1.IdAzione = P.IdAzione
     AND     A1.IdAzione = A2.IdAzione
     AND     A1.Data < A2.Data
     AND     NOT EXISTS (  SELECT   *
                           FROM     AZIONI A3
                           WHERE   A3.IdAzione = A1.IdAzione
                           AND      A3.Data > A1.Data
                           AND      A3.Data < A2.Data )
     GROUP BY S.Societa, A2.DATA )
SELECT      *
FROM        GUADAGNI G
WHERE       G.Guadagno = (  SELECT          MAX(G1.Gaudagno)
                            FROM          GUADAGNI G1
                            WHERE         G1.Societa = G.Societa)    ;

-- La c.t.e. calcola per ogni società e data il guadagno rispetto al giorno precedente presente nel DB
```

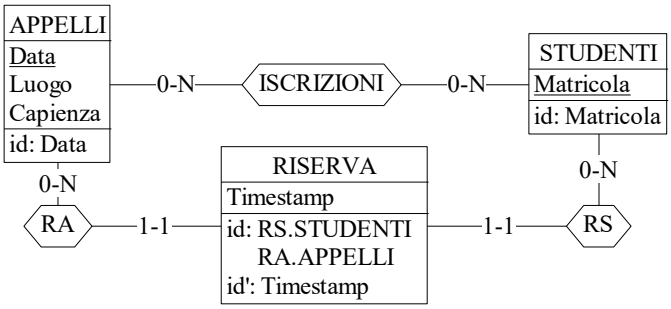
Sistemi Informativi T
12 febbraio 2026
Risoluzione

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:



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.

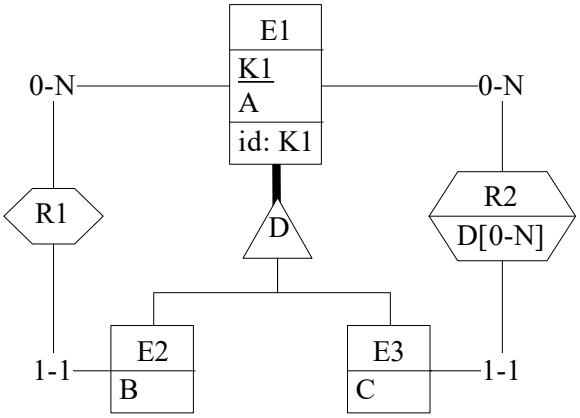
```
-- possibili anche due trigger distinti
CREATE OR REPLACE TRIGGER RIMPIAZZA_ISCRIZIONE
AFTER DELETE ON ISCRIZIONI
REFERENCING OLD AS O
FOR EACH ROW
WHEN (EXISTS (SELECT * FROM   RISERVA
                WHERE  Data = O.Data))
BEGIN ATOMIC
INSERT INTO ISCRIZIONI
SELECT O.DATA, R.Matricola
FROM    RISERVA R
WHERE R.Timestamp = (  SELECT MIN(R1.Timestamp)
                       FROM    RISERVA R1
                       WHERE R1.Data = O.Data );
--
DELETE FROM RISERVA R
WHERE  R.Timestamp = ( SELECT MIN(R1.Timestamp)
                       FROM    RISERVA R1
                       WHERE R1.Data = O.Data );
END

DELETE FROM ISCRIZIONI WHERE  Data = :data AND Matricola = :matricola;
```

Sistemi Informativi T
12 febbraio 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) nessuna associazione viene tradotta separatamente;
c) un'istanza di E3 che partecipa a R2 dal ramo 1-1 e ha almeno un valore di D non è mai associata a un'istanza di E2;



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,
TIPO    SMALLINT NOT NULL CHECK (TIPO IN (1,2,3)), -- 2 se istanza di E2, 3 se istanza di E3, 1 altrimenti
B       INT,
K1R1    INT REFERENCES E1,
C       INT,
K1R2    INT REFERENCES E1,
CONSTRAINT E2 CHECK ((TIPO = 2 AND B IS NOT NULL AND K1R1 IS NOT NULL) OR
                     (TIPO <> 2 AND B IS NULL AND K1R1 IS NULL)),
CONSTRAINT E3 CHECK ((TIPO = 3 AND C IS NOT NULL AND K1R2 IS NOT NULL) OR
                     (TIPO <> 3 AND C IS NULL AND K1R1 IS NULL))          );

CREATE TABLE R2D (
K1E3    INT NOT NULL REFERENCES E1,
D       INT NOT NULL,
PRIMARY KEY (K1E3,D)                  );
```

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 K1E3 referenzi un'istanza di E3
CREATE TRIGGER R2_E3
BEFORE INSERT ON R2D
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( NOT EXISTS ( SELECT * FROM E1 WHERE N.K1E3 = K1 AND TIPO = 3 )    )
SIGNAL SQLSTATE '70001' ('La tupla referenziata deve appartenere a E3!');

CREATE TRIGGER PUNTO_C
BEFORE INSERT ON R2D                    -- unico caso in cui il vincolo può essere violato
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS ( SELECT *
                FROM    E1, E1 E3
                WHERE N.K1E3 = E3.K1
                AND E3.K1R2 = E1.K1
                AND E1.TIPO = 2 )  )
SIGNAL SQLSTATE '70002' ('La tupla inserita viola il vincolo al punto c)!');
```