Sistemi Informativi T
14 settembre 2022
Risoluzione

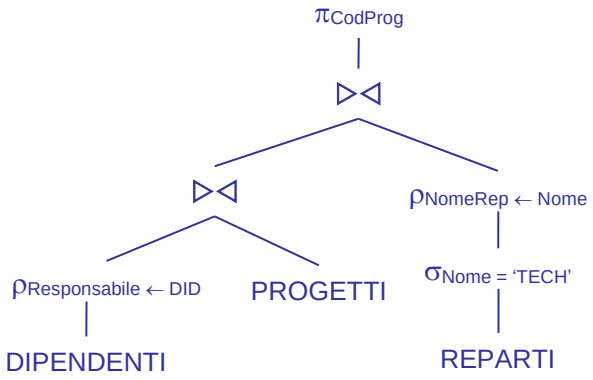
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

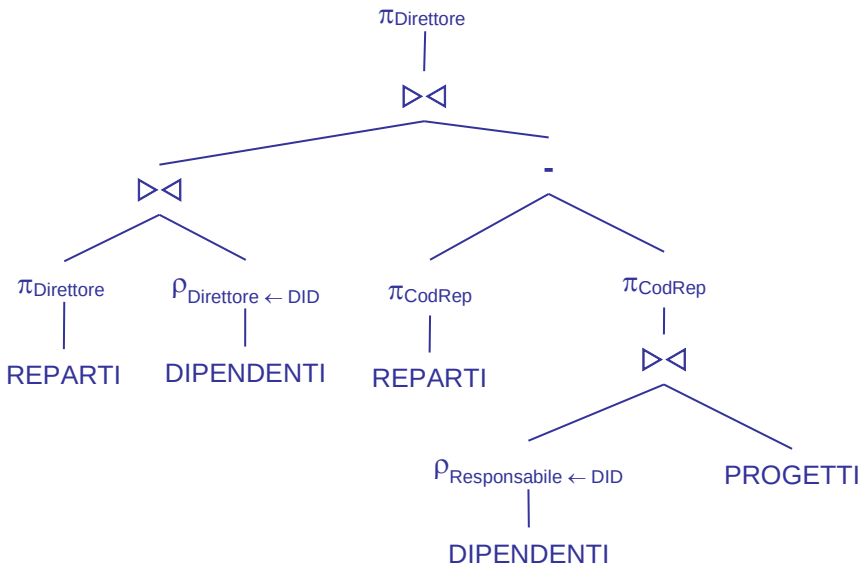
```
REPARTI(CodRep,Nome,Direttore),
      Direttore REFERENCES DIPENDENTI;
DIPENDENTI(DID,Nome,Stipendio,CodRep),
      CodRep REFERENCES REPARTI;
PROGETTI(CodProg,Titolo,Budget,Responsabile),
      Responsabile REFERENCES DIPENDENTI;
-- Stipendio e Budget sono di tipo DEC(8,2).
-- Più reparti possono avere lo stesso direttore.
-- Un progetto afferisce al reparto del responsabile del progetto.
```

si esprimano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I codici dei progetti che afferiscono al reparto di nome 'TECH'



1.2) [2 p.] I codici dei direttori che sono dipendenti di reparti senza alcun progetto



L'operando destro della differenza trova i reparti con almeno un progetto afferente, quindi l'operando destro dell'ultimo join consiste dei codici dei reparti senza progetti. L'operando sinistro di tale join fornisce, tra le altre informazioni, per ogni Direttore il codice del suo reparto.

Sistemi Informativi T
14 settembre 2022
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 reparto per il quale il budget totale dei progetti afferenti è maggiore di 500000€, tale totale e il numero di progetti

```
SELECT   R.CodRep, SUM(P.Budget) AS BudgetTotale,
         COUNT(*) AS NumProgetti
FROM     REPARTI R, DIPENDENTI D, PROGETTI P
WHERE    D.CodRep = R.CodRep
AND      P.Responsabile = D.DID
GROUP BY R.CodRep
HAVING   SUM(P.Budget) > 500000;
```

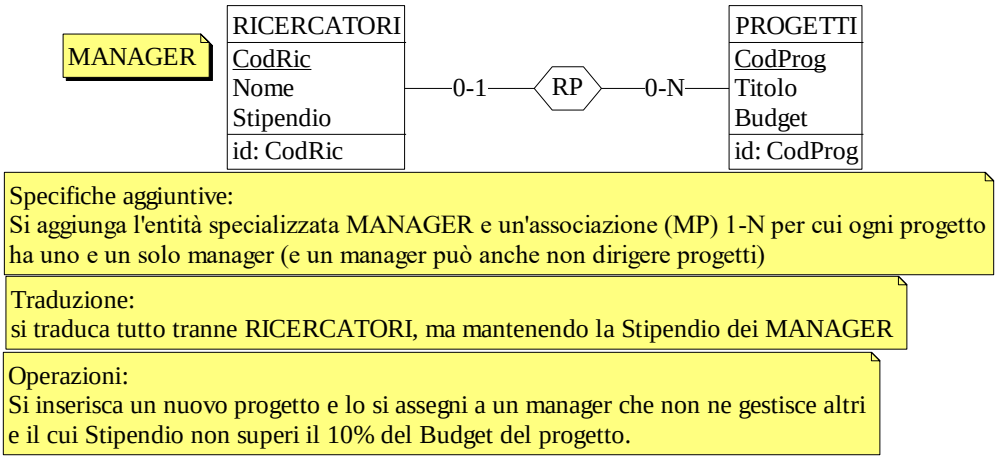
2.2) [3 p.] Il numero di reparti nei quali lo stipendio medio dei dipendenti che non sono responsabili di progetti è maggiore di 2400€

```
WITH StipMedi(StipMedio) AS (
   SELECT AVG(D.Stipendio)
   FROM   DIPENDENTI D
   WHERE  D.DID NOT IN ( SELECT  P.Responsabile
                         FROM    PROGETTI P)
   GROUP BY D.CodRep )
SELECT   COUNT(*) AS NumReparti
FROM     StipMedi
WHERE    StipMedio > 2400;
```

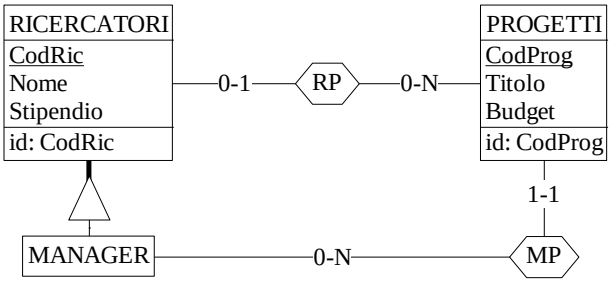
Sistemi Informativi T
14 settembre 2022
Risoluzione

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 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.

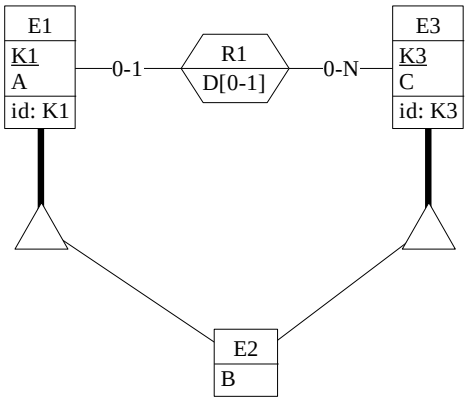
```
CREATE OR REPLACE TRIGGER MANAGER_VALIDO
BEFORE INSERT ON PROGETTI
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS
            (SELECT P.CodRic
             FROM    PROGETTI P
             WHERE  P.CodRic = N.CodRic )
       OR  ( 0.1*N.Budget < (  SELECT M.Stipendio
                               FROM    MANAGER M
                               WHERE  M.CodRic = N.CodRic) )
        )
SIGNAL SQLSTATE '70001' ('Manager gia" responsabile o con stipendio troppo elevato!')

INSERT INTO PROGETTI VALUES
    (:codProg, :titolo, :budget, :codRic);
```

Sistemi Informativi T
14 settembre 2022
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) le entità E1 ed E2 vengono tradotte insieme;
b) l'associazione R1 non viene tradotta separatamente;
c) un'istanza di E1 che partecipa a R1 non può essere associata a istanze di E3 che appartengono anche a 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 E3 (
K3          INT NOT NULL PRIMARY KEY,
C           INT NOT NULL                    );

CREATE TABLE E1 (
K1          INT NOT NULL PRIMARY KEY,
A           INT NOT NULL,
K3R1        INT REFERENCES E3,
D           INT,
TIPO2       SMALLINT NOT NULL CHECK (TIPO2 IN (1,2)), -- TIPO2 = 2 se appartiene anche a E2
B           INT,
K3SUB       INT REFERENCES E3,                        -- relazione di subset tra E2 ed E3
CONSTRAINT E2 CHECK ((TIPO2 = 1 AND B IS NULL AND K3SUB IS NULL) OR
                     (TIPO2 = 2 AND B IS NOT NULL AND K3SUB IS NOT NULL)),
CONSTRAINT R1 CHECK (K3R1 IS NOT NULL OR D IS NULL)        );
```

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**

```
CREATE TRIGGER K3SUB_UNIQUE
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (    SELECT *
                   FROM   E1
                   WHERE N.K3SUB = E1.K3SUB  ) )
SIGNAL SQLSTATE '70001' ('La foreign key K3SUB non ammette duplicati!');

-- Il vincolo al punto c) può essere violato inserendo in E1 una tupla in cui il valore di K3R1 che si sta inserendo
-- coincide con quello di K3SUB per una tupla già inserita (primo caso) o per la tupla stessa (secondo caso).
-- In alternativa si poteva usare un AFTER TRIGGER.
-- Il trigger funziona correttamente anche se N.K3R1 è NULL

CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (     SELECT *
                   FROM   E1
                   WHERE N.K3R1 = E1.K3SUB  )
          OR (N.K3R1 = N.K3SUB) )
SIGNAL SQLSTATE '70002' ('La tupla di E1 referenzia una tupla di E3 che appartiene anche a E2! ');
```