Sistemi Informativi T
17 settembre 2018
Risoluzione

Tempo a disposizione: 2:30 ore

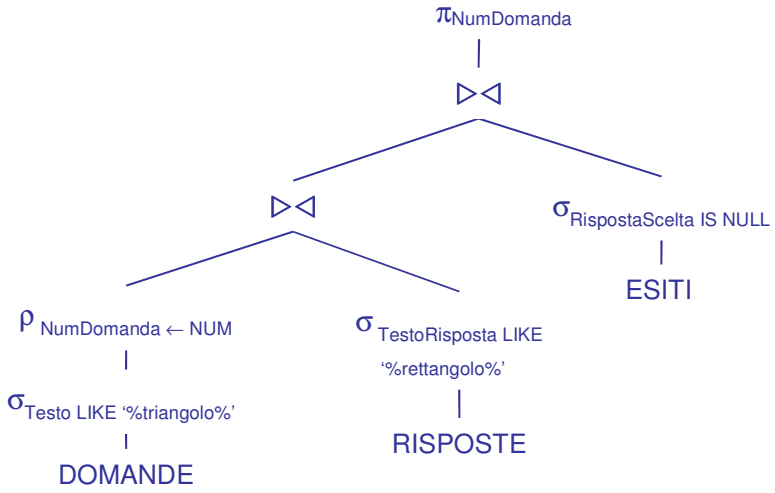
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

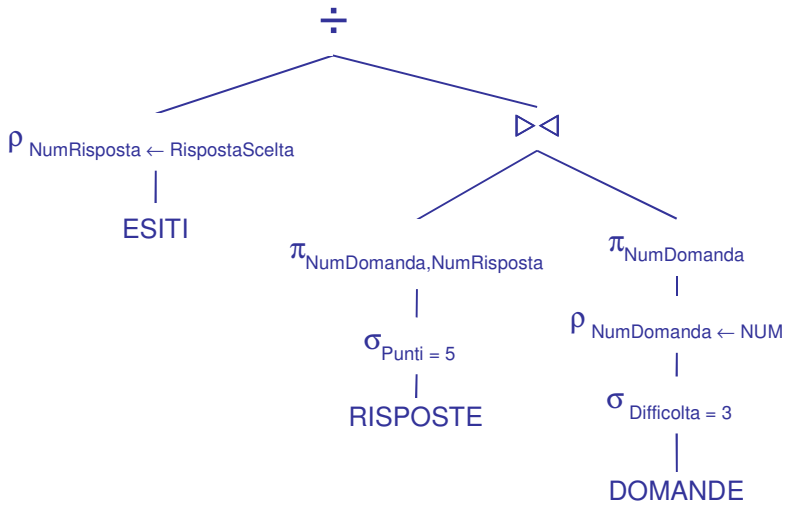
```
DOMANDE(NUM,Testo,Difficolta);
RISPOSTE(NumDomanda,NumRisposta,TestoRisposta,Punti),
      NumDomanda references DOMANDE;
ESITI(Utente,NumDomanda,RispostaScelta*),
      (NumDomanda,RispostaScelta) references RISPOSTE;
-- Difficolta ha valori 1 (facile), 2 (media), 3 (difficile)
-- Punti è di tipo INT e varia tra 0 e 5
-- RispostaScelta può essere NULL se un utente non ha risposto
-- a tutte le domande
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I numeri delle domande che nel testo contengono "triangolo", hanno una risposta che contiene "rettangolo" e almeno un utente non ha risposto alla domanda



1.2) **[2 p.]** Gli utenti che hanno risposto esattamente (5 punti) a tutte le domande difficili



Il divisore consiste di tutte le coppie (domanda difficile, risposta giusta), e quindi la divisione restituisce gli utenti associati in ESITI a tutte queste coppie

Sistemi Informativi T
17 settembre 2018
Risoluzione

2) SQL (5 punti totali)

Con riferimento al DB dell'esercizio 1, si scrivano in SQL le seguenti interrogazioni:

2.1) **[2 p.]** Il punteggio totalizzato da ogni utente, ordinando il risultato per punteggio decrescente e, a parità di punteggio, per utente (ogni risposta data viene pesata con la difficoltà della domanda, ovvero contribuisce al totale con Punti*Difficolta)

```
SELECT   E.Utente, SUM(D.Difficolta*R.Punti) AS Punteggio
FROM     DOMANDE D, RISPOSTE R, ESITI E
WHERE    D.NUM = R.NumDomanda
AND      (R.NumDomanda,R.NumRisposta) = (E.NumDomanda,E.RispostaScelta)
GROUP BY E.Utente
ORDER BY Punteggio DESC, E.Utente;
```

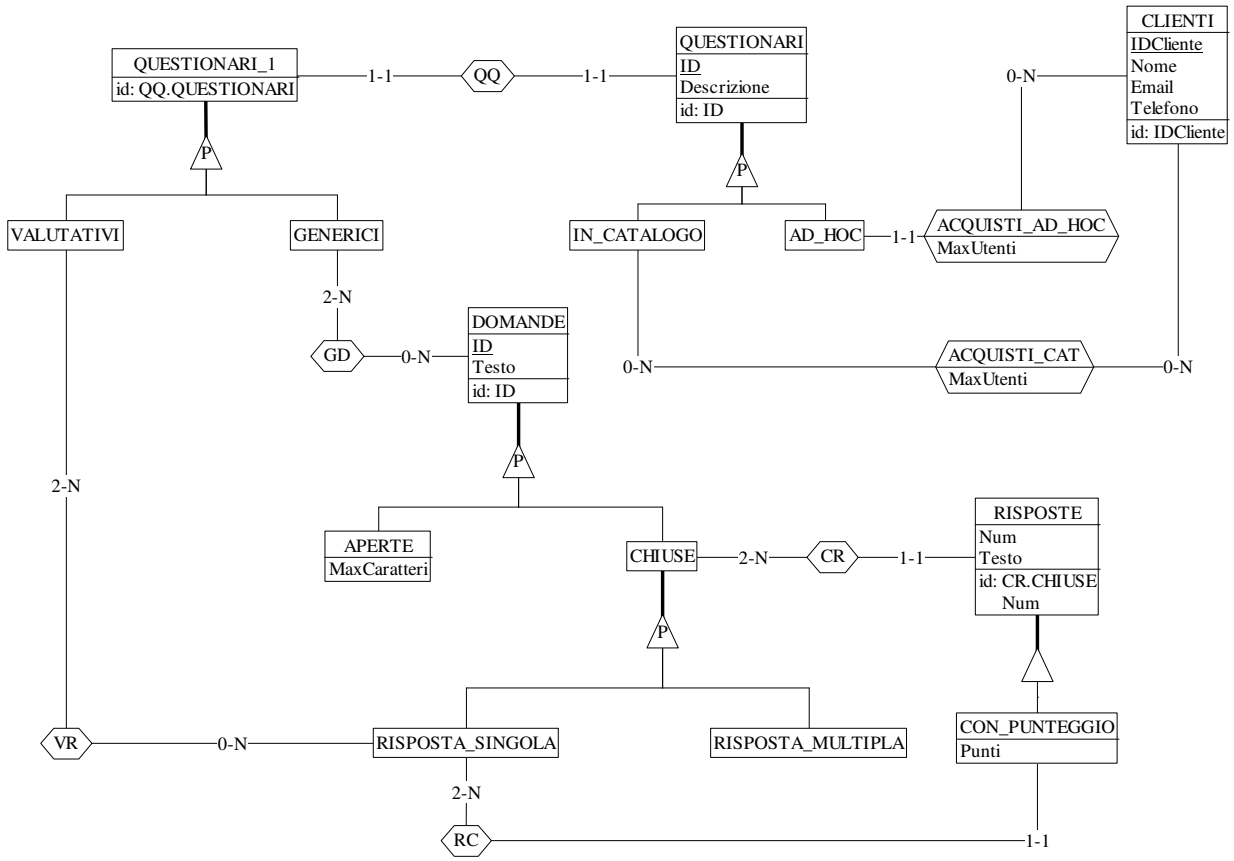
2.2) **[3 p.]** L'utente con la massima differenza di punti totalizzati tra domande difficili e facili (in questo caso i pesi non vanno considerati)

```
WITH
   USER_DIFF(Utente,TotDiff) AS (
   SELECT   E.Utente, SUM(R.Punti)
   FROM     DOMANDE D, RISPOSTE R, ESITI E
   WHERE    D.NUM = R.NumDomanda
   AND      (R.NumDomanda,R.NumRisposta) = (E.NumDomanda,E.RispostaScelta)
   AND      D.Difficolta = 3
   GROUP BY E.Utente       ),

   USER_EASY(Utente,TotEasy) AS (
   SELECT   E.Utente, SUM(R.Punti)
   FROM     DOMANDE D, RISPOSTE R, ESITI E
   WHERE    D.NUM = R.NumDomanda
   AND      (R.NumDomanda,R.NumRisposta) = (E.NumDomanda,E.RispostaScelta)
   AND      D.Difficolta = 1
   GROUP BY E.Utente       ),

   DIFFERENZE(Utente,Differenza) AS (
   SELECT   E.Utente, TotDiff - TotEasy
   FROM     USER_DIFF D, USER_EASY E
   WHERE    D.Utente = E.Utente  )

SELECT   *
FROM     DIFFERENZE D
WHERE    D.Differenza = (  SELECT MAX(Differenza)
                           FROM   DIFFERENZE            );
```

Sistemi Informativi T
17 settembre 2018
Risoluzione

3) Progettazione concettuale (6 punti)

La società Surveys for the World (S4W) si occupa di fornire e gestire questionari online a uso dei propri clienti. La S4W dispone di un catalogo di questionari già predisposti, e utilizzabili da ogni cliente, ma sviluppa anche questionari ad hoc per i clienti che ne fanno richiesta (nel qual caso solo il cliente coinvolto può utilizzarlo).
Ogni questionario si compone di diverse domande e ogni domanda può essere a risposta aperta (nel qual caso per la risposta è previsto un numero massimo di caratteri, specifico per quella domanda) o a risposta chiusa (nel qual caso sono previste 2 o più risposte tra cui scegliere e la domanda specifica se si può scegliere una sola risposta o anche più di una).
Per i questionari di tipo valutativo le domande sono tutte a risposta chiusa e ad ogni risposta prevista è associato un punteggio. Per questo tipo di questionari per ogni domanda si può scegliere una sola risposta.
La S4W tiene traccia dei propri clienti e di quali questionari hanno acquistato. Per ogni acquisto la S4W impone un limite massimo di utenti che possono compilare il questionario.



Commenti:

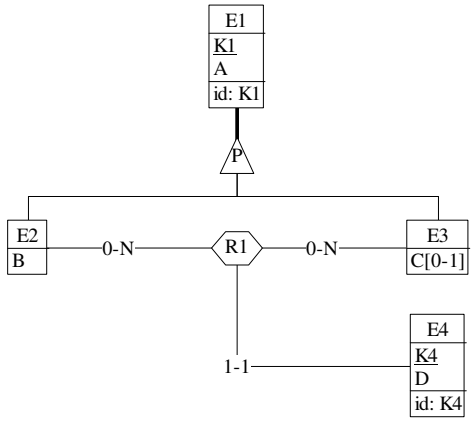
- L'esercizio si caratterizza per la difficoltà concettuale di associare tra loro le principali entità in gioco, ovvero CLIENTI, QUESTIONARI, DOMANDE e RISPOSTE. Considerando le specifiche, appare evidente la necessità di specializzare le entità, il che porta a uno schema con molte entità e gerarchie.
- Il limite di DB-Main, che non permette di avere un'entità come radice di più gerarchie distinte, viene superato introducendo l'entità fittizia QUESTIONARI_1, in relazione 1-1 con QUESTIONARI e da questa identificata

Sistemi Informativi T
17 settembre 2018
Risoluzione

4) Progettazione logica (6 punti totali)

Dato lo schema concettuale in figura e considerando che:

a) tutti gli attributi sono di tipo INT;
b) le entità E1, E2 ed E3 vengono tradotte insieme;
c) l'associazione R1 non viene tradotta separatamente;
d) le istanze di E3 che partecipano a R1 hanno sempre il valore di C non nullo;



4.1) [3 p.] Si progettino gli opportuni schemi relazionali e si definiscano tali schemi in DB2 (sul database SIT_STUD) mediante un file di script denominato SCHEMI.txt

```
CREATE TABLE E1 (
K1 INT NOT NULL PRIMARY KEY,
A INT NOT NULL,
TIPO SMALLINT NOT NULL CHECK (TIPO IN (2,3)),          -- 2: istanza di E2, 3: istanza di E3
B INT,
C INT,
CONSTRAINT E2E3 CHECK ((TIPO = 2 AND B IS NOT NULL AND C IS NULL) OR
                       (TIPO = 3 AND B IS NULL ))                    );

CREATE TABLE E4 (
K4 INT NOT NULL PRIMARY KEY,
D INT NOT NULL,
K1E2 INT NOT NULL REFERENCES E1,
K1E3 INT NOT NULL REFERENCES E1                                      );
```

4.2) [3 p.] Per i vincoli non esprimibili a livello di schema si predispongano opportuni **trigger che evitino inserimenti di tuple non corrette**, definiti in un file TRIGGER.txt e usando se necessario il simbolo '@' per terminare gli statement SQL (altrimenti ';')

```
-- Trigger che garantisce che una tupla di E4 referenzi un'istanza di E2 tramite K1E2
-- e un'istanza di E3 tramite K1E3
CREATE TRIGGER R1_E2_E3
BEFORE INSERT ON E4
REFERENCING NEW AS N
FOR EACH ROW
WHEN (NOT EXISTS  ( SELECT * FROM E1
                    WHERE   N.K1E2 = E1.K1
                    AND     E1.TIPO = 2 )
      OR
      NOT EXISTS  ( SELECT * FROM E1
                    WHERE   N.K1E3 = E1.K1
                    AND     E1.TIPO = 3 )       )
SIGNAL SQLSTATE '70001' ('La tupla inserita deve referenziare una istanza di E2 e una istanza di E3!');

-- Trigger che garantisce il rispetto del vincolo al punto d)
CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E4
REFERENCING NEW AS N
FOR EACH ROW
WHEN (NOT EXISTS  ( SELECT * FROM E1
                    WHERE   N.K1E3 = E1.K1
                    AND     E1.C IS NOT NULL ) )
SIGNAL SQLSTATE '70002' ('La tupla referenziata da K1E3 deve avere C non nullo!');

-- Si noti che, dato il secondo trigger e il CHECK definito per E1, la seconda condizione del primo trigger
-- non è strettamente necessaria, in quanto se il vincolo al punto d) è soddisfatto allora l'istanza referenziata
-- appartiene a E3. Inoltre i due trigger si possono anche fondere assieme.
```