Sistemi Informativi T
22 gennaio 2018
Risoluzione

Tempo a disposizione: 2:30 ore

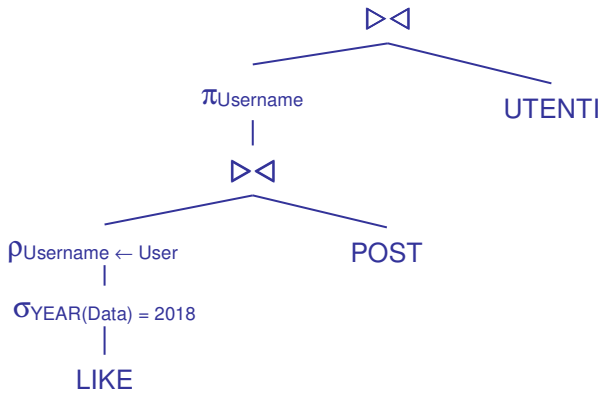
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

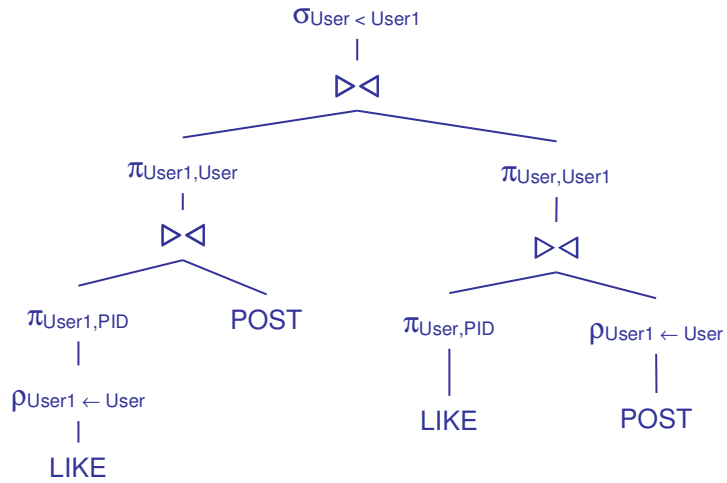
```
UTENTI(Username,Nome,Cognome,DataRegistrazione);
POST(PID,User,Data,Time,Testo),
      User REFERENCES UTENTI;
LIKE(User,PID,Data),
      User REFERENCES UTENTI,
      PID REFERENCES POST;
--
-- POST.User è l'utente che pubblica il post
-- LIKE.User è l'utente che ha messo un like a un post
-- Time è di tipo TIME (formato hh:mm:ss)
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I dati degli utenti che nel 2018 hanno messo un like a un post lo stesso giorno in cui è stato pubblicato



1.2) [2 p.] Le coppie di utenti che hanno almeno un like reciproco



Il primo operando dell'ultimo join è dato da coppie di utenti (User1,User) in cui User1 ha espresso un like su un post di User; il secondo operando è dato da coppie di utenti (User,User1) in cui User ha espresso un like su un post di User1

Sistemi Informativi T
22 gennaio 2018
Risoluzione

SQL (5 punti totali)

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

2.1) [2 p.] Per gli utenti registrati da meno di 10 giorni, il numero complessivo di like ricevuti

```
SELECT   U.Username, COUNT(*) AS TOT_LIKE
FROM     UTENTI U, POST P, LIKE L
WHERE    U.Username = P.User
AND      P.PID = L.PID
AND      DAYS(CURRENT DATE) - DAYS(U.DataRegistrazione) < 10
GROUP BY U.Username

-- Come detto durante la prova, si omettono per semplicità gli utenti
-- senza like
```

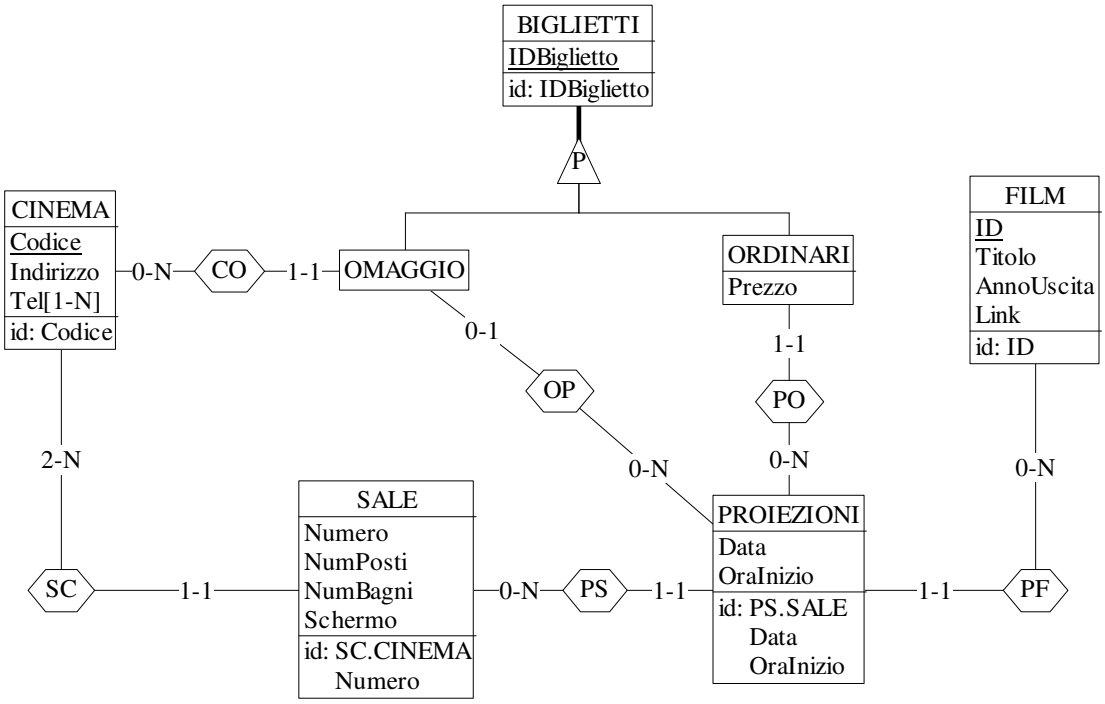
2.2) [3 p.] L'utente che ha la media dei like per post pubblicati più alta (vanno considerati anche i post senza nessun like)

```
WITH
NUMLIKE_POST (Utente, PID, NumLike) AS (
   SELECT P.User,P.PID,COUNT(L.Data)
   FROM   POST P LEFT JOIN LIKE L ON (P.PID = L.PID)
   GROUP BY P.PID,P.User         ),

MEDIE_LIKE (Utente,Media) AS (
   SELECT P.User, CAST(AVG(N.NumLike/1.0) AS DEC(6,2))
   FROM  NUMLIKE_POST N
   GROUP BY N.User   )

SELECT   M1.*
FROM     MEDIE_LIKE M1
WHERE    M1.Media = ( SELECT MAX(M2.Media)
                      FROM   MEDIE_LIKE M2 )

-- La prima c.t.e. calcola il numero di like per ogni post (si noti che è
-- necessario raggruppare anche per utente); la seconda c.t.e. calcola per
-- ogni utente la media dei like ricevuti. Nella prima c.t.e. non è corretto
-- usare COUNT(*), che darebbe 1, anziché 0, per i post senza like.
-- In alternativa all'uso di outer join si poteva definire la prima c.t.e.
-- mediante UNION:

 ( SELECT P.User,P.PID,COUNT(*)
   FROM  POST P, LIKE L
   WHERE P.PID = L.PID
   GROUP BY P.PID,P.User )
 UNION
 ( SELECT P.User,P.PID,0
   FROM  POST P
   WHERE P.PID NOT IN (SELECT PID FROM LIKE) )
```

Sistemi Informativi T
22 gennaio 2018
Risoluzione

3) Progettazione concettuale (6 punti)

MOViE ON (MO) è una catena di cinema multisala distribuiti sul territorio nazionale. Ogni cinema, descritto da indirizzo, uno o più numeri di telefono e un codice univoco, ha due o più sale. Ogni sala ha un numero (univoco per quel cinema), un numero di posti, almeno un bagno, e uno schermo di una certa dimensione.
Per ogni film in programmazione (di cui il sistema informativo di MO mantiene traccia del titolo e dell'anno di uscita, rimandando con un link a un sito esterno per tutti i dettagli) si riportano per ogni sala in cui viene proiettato le date e gli orari. Ogni biglietto venduto ha un identificativo, univoco su tutta la catena, e riporta cinema, sala, data, orario e prezzo pagato. Ogni cinema ha a disposizione una serie di biglietti omaggio (anch'essi dotati di identificativo) che, quando presentati alla cassa, vengono convertiti in biglietti ordinari per una proiezione di quel cinema (a prezzo ovviamente pari a zero).



Commenti:

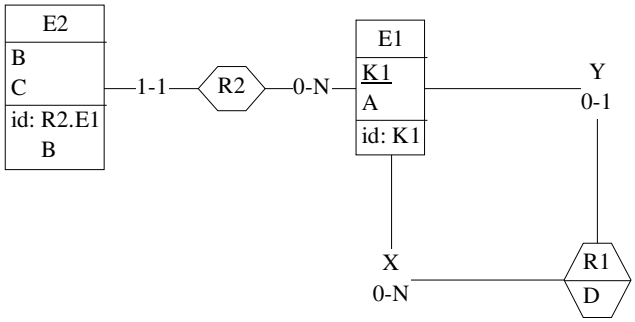
- L'entità PROIEZIONI, ottenuta per reificazione, è fondamentale per la corretta risoluzione dell'esercizio. Si noti che FILM non serve per identificare una proiezione.
- Per la modellazione della gerarchia dei BIGLIETTI si poteva anche introdurre un'altra entità (ad es. CONVERTITI) associata a PROIEZIONI, oppure lasciare al valore del Prezzo la distinzione con i biglietti ORDINARI. La soluzione adottata è di fatto la più semplice e precisa.
- E' da notare che l'unica informazione da fornire per i biglietti ORDINARI è il prezzo, in quanto tutte le altre (cinema, sala, data, orario) si ricavano dall'associazione con PROIEZIONI.
- Il vincolo che ogni sala ha almeno un bagno non è esprimibile a livello di schema E/R, così come non è esprimibile il vincolo che un biglietto omaggio può essere usato solo nelle sale del cinema che lo ha messo a disposizione.

Sistemi Informativi T
22 gennaio 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 associazioni R1 e R2 non vengono tradotte separatamente;
c) un'istanza di E1 che partecipa a R1 con ruolo Y non può partecipare a R2;



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,
R1Y     SMALLINT NOT NULL CHECK (R1Y IN (0,1)), -- non essenziale
K1X     INT REFERENCES E1,
D       INT,
CONSTRAINT R1 CHECK (  (R1Y = 0 AND K1X IS NULL AND D IS NULL) OR
                       (R1Y = 1 AND K1X IS NOT NULL AND D IS NOT NULL)  )        );

CREATE TABLE E2 (
K1R2    INT NOT NULL REFERENCES E1,
B       INT NOT NULL,
C       INT NOT NULL,
PRIMARY KEY (K1R2,B)                    );
```

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**, definiti in un file TRIGGER.txt e usando se necessario il simbolo '@' per terminare gli statement SQL (altrimenti ';')

```
-- Trigger che garantisce il rispetto del vincolo al punto c)

CREATE TRIGGER PUNTO_C
BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (     SELECT *
                    FROM    E1
                    WHERE N.K1R2 = E1.K1
                    AND     E1.R1Y = 1              )   )
SIGNAL SQLSTATE '70001' ('La tupla referenzia una tupla di E1 che partecipa a R1 con ruolo Y!');
```