Sistemi Informativi T
15 giugno 2010
Risoluzione

Tempo a disposizione: 2:30 ore

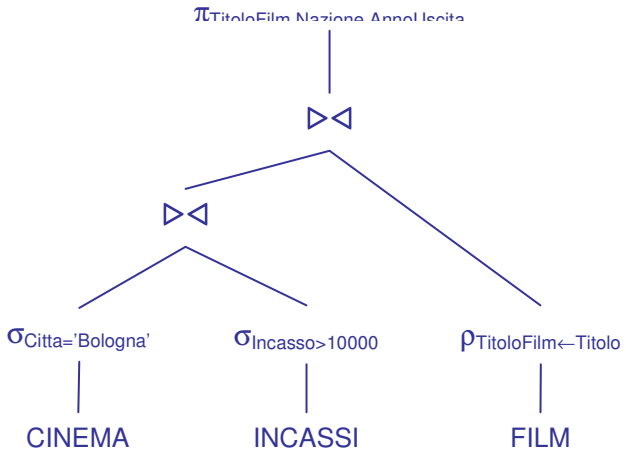
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

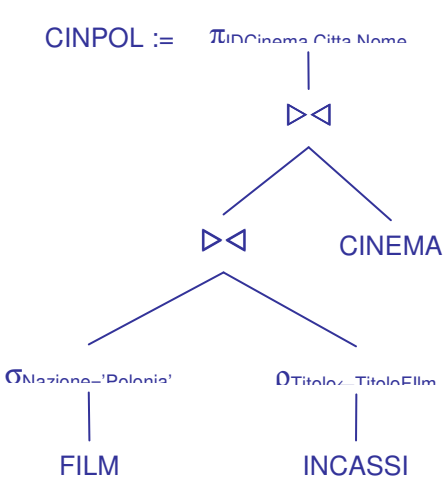
FILM(Titolo,Nazione,AnnoUscita);
CINEMA(IDCinema,Citta,Nome);
INCASSI(TitoloFilm,IDCinema,Data,Incasso),
TitoloFilm references FILM, IDCinema references CINEMA;

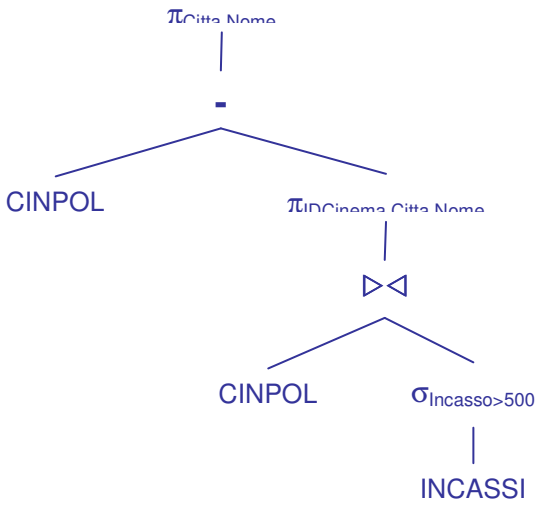
si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** Dettagli (senza duplicati) dei film che in un giorno hanno incassato più di 10000 € in un cinema di Bologna



1.2) **[2 p.]** Città e nome dei cinema che non hanno mai incassato in un giorno più di 500 € per un film di produzione polacca (e avendone proiettato almeno uno di tale nazione)





La vista CINPOL consiste dei soli cinema che hanno proiettato almeno un film polacco. A questi vengono sottratti quelli che in un giorno hanno incassato più di 500 €

Sistemi Informativi T
15 giugno 2010
Risoluzione

2) SQL (5 punti totali)

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

2.1) [2 p.] Per ogni anno e per ogni città, gli incassi totali dei film usciti nello stesso anno

```
SELECT YEAR(I.Data) AS Anno, C.Citta, SUM(I.Incasso) AS IncassoTotale
FROM CINEMA C, INCASSI I, FILM F
WHERE C.IDCinema = I.IDCinema
AND I.TitoloFilm = F.Titolo
AND YEAR(I.Data) = F.AnnoUscita
GROUP BY YEAR(I.Data), C.Citta
```

2.2) [3 p.] Per ogni cinema e ogni film che è stato proiettato in quel cinema almeno 30 giorni, l'incasso massimo registrato e le date in cui si è ottenuto

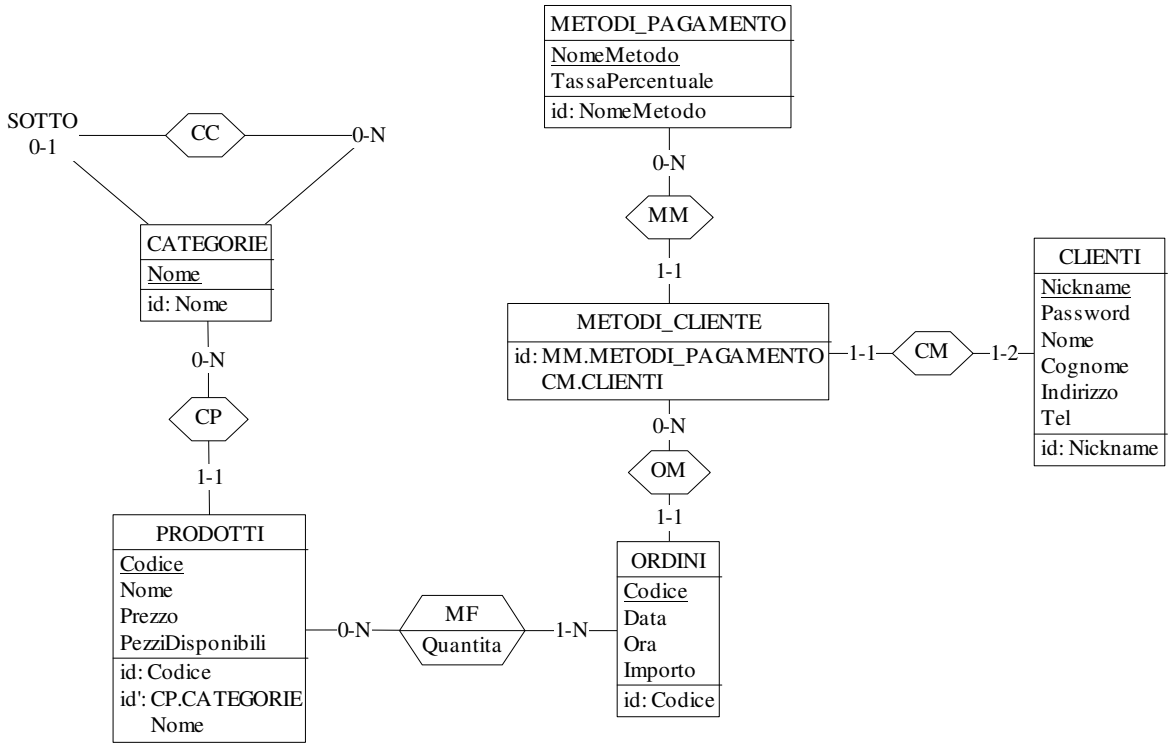
```
WITH INCASSIMASSIMI(IDCinema,TitoloFilm,IncassoMax)
AS SELECT I.IDCinema, I.TitoloFilm, MAX(I.Incasso)
   FROM INCASSI I
   GROUP BY I.IDCinema, I.TitoloFilm
   HAVING COUNT(*) >= 30
SELECT IM.*,I.Data
FROM INCASSIMASSIMI IM, INCASSI I
WHERE I.IDCInema = IM.IDCinema
AND I.TitoloFilm = IM.TitoloFilm
AND I.Incasso = IM.IncassoMax

-- si noti che la common table expression INCASSIMASSIMI serve (solo)
-- in quanto in output sono anche richieste le date in cui si e'
-- registrato l'incasso massimo
```

Sistemi Informativi T
15 giugno 2010
Risoluzione

3) Progettazione concettuale (6 punti)

AMAZZONE è una società specializzata nella vendita di prodotti on-line.
Ogni prodotto, di cui si memorizzano nome, prezzo e numero di pezzi disponibili, appartiene a una specifica categoria ed è identificato da un codice, o, alternativamente, dal suo nome e da quello della categoria di appartenenza. Le categorie sono organizzate in maniera gerarchica, con una categoria che può contenere una o più sottocategorie.
Per effettuare acquisti, i potenziali clienti devono prima registrarsi, fornendo alcuni dati personali (nome, cognome, indirizzo, telefono). I clienti sono identificati da un nickname da loro scelto, a cui è associata una password. Al momento della registrazione, il cliente può esprimere fino a due preferenze riguardo al metodo di pagamento privilegiato, scegliendo tra quelli gestiti dal sistema (carta di credito, bonifico bancario, ecc.). Per ogni metodo di pagamento è anche definita una tassa percentuale, uguale per tutti i clienti.
Un cliente può effettuare degli ordini, dove un ordine è costituito da uno o più prodotti (ognuno in quantità variabile) ed è identificato da un codice univoco generato automaticamente dal sistema. Il sistema registra la data e l'ora in cui l'ordine è stato effettuato e il metodo di pagamento utilizzato (tra quelli scelti dal cliente in fase di registrazione).



Commenti

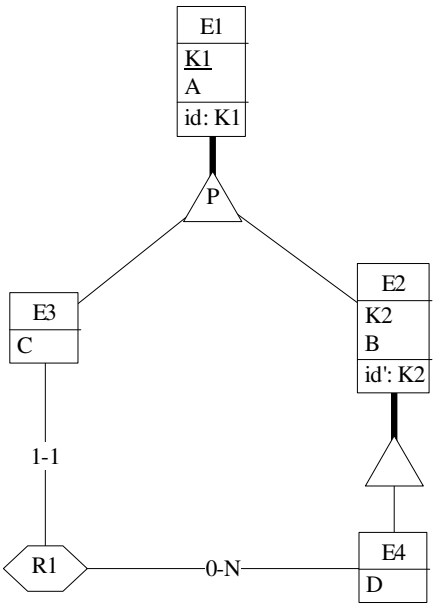
- L'entità METODI_CLIENTE è ottenuta mediante reificazione, e serve a vincolare il metodo di pagamento di un ordine.

Sistemi Informativi T
15 giugno 2010
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) l'associazione R1 non viene tradotta separatamente;
c) le entità E1, E2 ed E3 vengono tradotte assieme;
d) l'entità E4 viene tradotta separatamente;



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 E4(
K1 INT NOT NULL PRIMARY KEY,
D INT NOT NULL    );

CREATE TABLE E1(
K1 INT NOT NULL PRIMARY KEY,
A INT NOT NULL,
SEL23 SMALLINT NOT NULL CHECK (SEL2 IN (2,3)),
-- SEL23 = 2 (3) : la tupla e' un'istanza di E2 (E3)
K2 INT,
B INT,
C INT,
K1E4 INT REFERENCES E4
CONSTRAINT E2_E3 CHECK (
(SEL23 = 2 AND K2 IS NOT NULL AND B IS NOT NULL AND C IS NULL AND K1E4 IS NULL) OR
(SEL23 = 3 AND K2 IS NULL AND B IS NULL AND C IS NOT NULL AND K1E4 IS NOT NULL))    );

ALTER TABLE E4
ADD CONSTRAINT FK FOREIGN KEY (K1) REFERENCES E1;
-- va dichiarato dopo la definizione di 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 il simbolo '@' per terminare gli statement SQL

```
-- vincoli da esprimere: E4 e' subset di E2 e K2 e' chiave
CREATE TRIGGER INS_E4
NO CASCADE BEFORE INSERT ON E4
REFERENCING NEW AS N
FOR EACH ROW MODE DB2SQL
WHEN ( NOT EXISTS (SELECT *
                   FROM E1
                   WHERE E1.K1 = N.K1
                   AND E1.SEL23 = 2)
SIGNAL SQLSTATE '70001' ('Un'istanza di E4 deve essere anche istanza di E2 !')@

CREATE TRIGGER INS_E1
NO CASCADE BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW MODE DB2SQL
WHEN (EXISTS (SELECT *
              FROM E1
              WHERE E1.K2 = N.K2)
SIGNAL SQLSTATE '70002' ('I valori di K2 non possono essere duplicati!')@
-- si noti che, poiche' NULL <> NULL, il trigger non da errore se si inserisce un'istanza di E3 per cui
-- K2 non e' definito
```