Sistemi Informativi T
20 febbraio 2015
Risoluzione

Tempo a disposizione: 2:30 ore

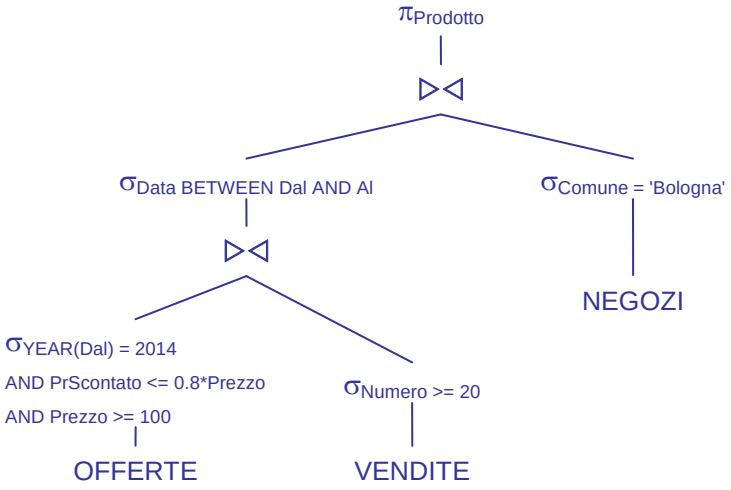
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

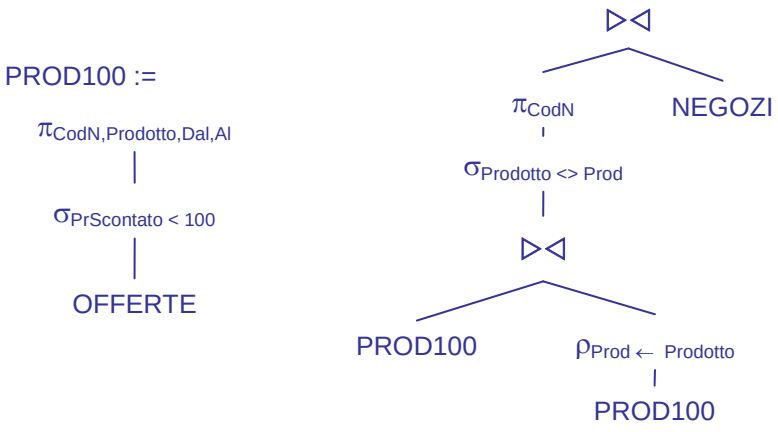
```
NEGOZI(CodN,Indirizzo,Comune);
OFFERTE(Prodotto,CodN,Dal,Al,Prezzo,PrScontato),
      CodN REFERENCES NEGOZI;
VENDITE(Prodotto,CodN,Data,Numero),
      CodN REFERENCES NEGOZI;
--
-- Prezzo e PrScontato sono di tipo DEC(6,2), PrScontato < Prezzo
-- Dal e Al sono date (di uno stesso anno) che definiscono
-- un periodo di offerte (i periodi di offerte di un negozio sono
-- tra loro disgiunti)
-- Numero è il numero di unità di Prodotto vendute in una certa
-- Data dal negozio CodN; Numero > 0
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I prodotti che nel 2014 sono stati offerti con almeno il 20% di sconto (a partire da un prezzo non scontato di almeno 100€) da un negozio di Bologna, e che per quell'offerta in un giorno sono stati venduti in almeno 20 pezzi



1.2) [2 p.] I dati dei negozi che, in uno stesso periodo, hanno messo in offerta due prodotti entrambi a prezzo scontato minore di 100€



Sistemi Informativi T
20 febbraio 2015
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 prodotto l'incasso totale

```
SELECT   O.Prodotto, SUM(O.PrScontato*V.Numero) AS IncassoTot
FROM     OFFERTE O, VENDITE V
WHERE    O.CodN = V.CodN
AND      O.Prodotto = V.Prodotto
AND      V.Data BETWEEN O.Dal AND O.Al
GROUP BY O.Prodotto
```

2.2) [3 p.] Per ogni negozio il prodotto maggiormente venduto, escludendo i periodi di offerte di durata superiore a 7 giorni

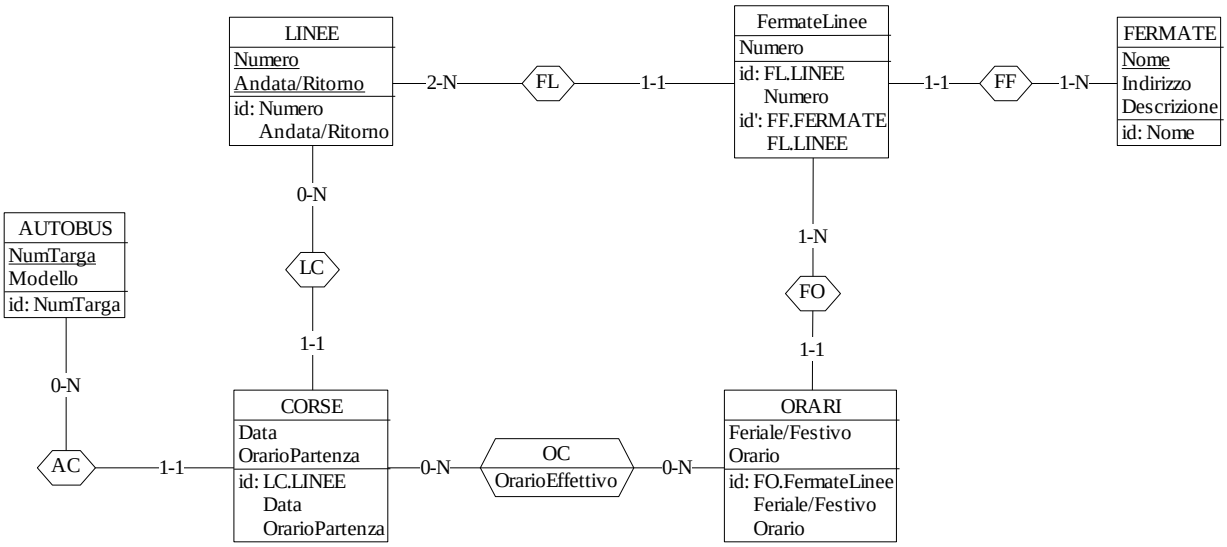
```
WITH PRODNEG(Prodotto,CodN,TotVendite) AS (
   SELECT V.Prodotto, V.CodN, SUM(V.Numero)
   FROM   OFFERTE O, VENDITE V
   WHERE  V.CodN = O.CodN
   AND    V.Prodotto = O.Prodotto
   AND    V.Data BETWEEN O.Dal AND O.Al
   AND    DAYS(O.Al) - DAYS(O.Dal) < 7
   GROUP BY V.Prodotto, V.CodN            )
SELECT   P.*
FROM     PRODNEG P
WHERE    P.TotVendite = (  SELECT MAX(P1.TotVendite)
                           FROM   PRODNEG P1
                           WHERE  P1.CodN = P.CodN)

-- I periodi di offerte di 7 o meno giorni si individuano con la condizione
-- DAYS(O.Al) - DAYS(O.Dal) < 7, e non ...<= 7.
-- Ad esempio: DAYS('27/03/2014') - DAYS('20/03/2014') = 7, ma l'intervallo
-- è di 8 giorni, non 7
```

Sistemi Informativi T
20 febbraio 2015
Risoluzione

3) Progettazione concettuale (6 punti)

Il sistema informativo della CTU (Consorzio Trasporti Urbani) tiene traccia delle linee degli autobus, delle fermate su tali linee, degli orari e dei mezzi (autobus) utilizzati.
Per ogni linea, identificata da un numero, sono dati i 2 capolinea e tutte le fermate intermedie (usualmente diverse tra il percorso di andata e quello di ritorno). Ogni fermata ha un nome univoco, ad es. Lodi3, un indirizzo e una descrizione (ad es. "con pensilina"). Una fermata può ovviamente essere condivisa da più linee, e per ogni linea e percorso di andata o ritorno ha un numero progressivo specifico (ad es. Lodi3 è la fermata n. 5 di andata per la linea 14 e la n.7 di ritorno per la linea 27). Per ogni fermata e ogni linea interessata sono riportati tutti gli orari di passaggio feriali e festivi.
Il sistema della CTU tiene traccia di tutte le corse svolte dagli autobus (corsa: percorso completo da un capolinea all'altro svolto in un dato giorno a una data ora), mantenendo per ogni corsa gli effettivi orari di passaggio alle fermate.



Commenti:

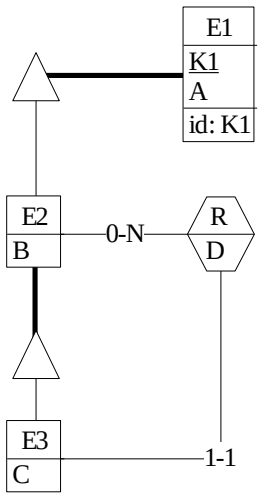
- La soluzione evidenzia la chiara distinzione tra una parte "statica" (entità LINEE, FERMATE, FermateLinee, ORARI, AUTOBUS e relative associazioni) e una parte "dinamica" (entità CORSE e associazioni LC e OC).
- Un'istanza dell'entità LINEE è una specifica linea in andata o ritorno. La scelta di introdurre l'attributo a 2 valori Andata/Ritorno in LINEE è motivata dalla necessità di identificare correttamente CORSE.
- L'entità FermateLinee è ottenuta per reificazione, sia per permettere i vincoli di identificazione sia per consentire il collegamento con ORARI.
- Le cardinalità minime poste a 0 servono a scopo di inizializzazione (nuovo autobus, nuova linea, ecc.).
- Si è optato per non introdurre esplicitamente il concetto di capolinea, in quanto coincidente con la prima fermata di una linea (in andata e in ritorno).
- Il vincolo che gli orari effettivi di una corsa devono referenziare gli orari corrispondenti della stessa linea cui la corsa si riferisce non è esprimibile.

Sistemi Informativi T
20 febbraio 2015
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 R non viene tradotta separatamente;
c) le entità E1, E2 ed E3 vengono tradotte insieme;
d) un'istanza di E2 può partecipare a R solo se B < D;



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 (1,2,3)),
     -- 1: istanza solo di E1; 2: istanza di E2 ma non di E3; 3: istanza di E3
B INT,
C INT,
K1R INT REFERENCES E1,
D INT,
CONSTRAINT E2 CHECK ((TIPO = 1  AND B IS NULL)  OR (TIPO <> 1 AND B IS NOT NULL)  ),
CONSTRAINT E3 CHECK (
     (TIPO <> 3 AND C IS NULL AND K1R IS NULL AND D IS NULL) OR
     (TIPO = 3 AND C IS NOT NULL AND K1R IS NOT NULL AND D IS NOT 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**, definiti in un file TRIGGER.txt e usando se necessario il simbolo '@' per terminare gli statement SQL (altrimenti ';')

```
-- Trigger che garantisce che K1R referenzi un'istanza di E2
CREATE TRIGGER R
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN (N.TIPO = 3 AND NOT EXISTS ( SELECT * FROM E1
                                    WHERE N.K1R  = E1.K1
                                    AND (E1.TIPO = 2 OR E1.TIPO =3)  ))  -- oppure TIPO <> 1
SIGNAL SQLSTATE '70001' ('La tupla inserita deve referenziare una tupla di E2!');

-- Trigger che garantisce il rispetto del vincolo di cui al punto d). Il vincolo può essere violato solo inserendo una
-- tupla in E1 che appartiene anche a E3
CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ((N.TIPO = 3 AND (EXISTS (  SELECT * FROM E1
                                   WHERE N.K1R  = E1.K1
                                   AND N.D <= E1.B            )
                              OR N.D <= N.B )
SIGNAL SQLSTATE '70002' ('La tupla inserita referenzia una tupla di E2 con B >= D!');
-- La seconda condizione gestisce il caso particolare in cui l'associazione R pone in relazione una tupla
-- con se stessa. Ovviamente la condizione N.TIPO = 3 può essere omessa, perché negli altri casi N.D è
-- NULL e la condizione di errore non è verificata
```