Sistemi Informativi T
6 febbraio 2015
Risoluzione

Tempo a disposizione: 2:30 ore

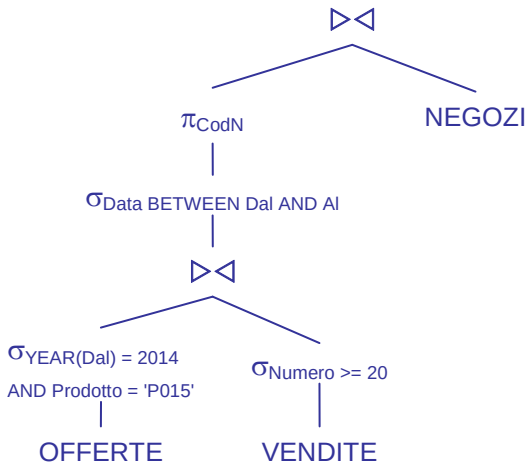
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

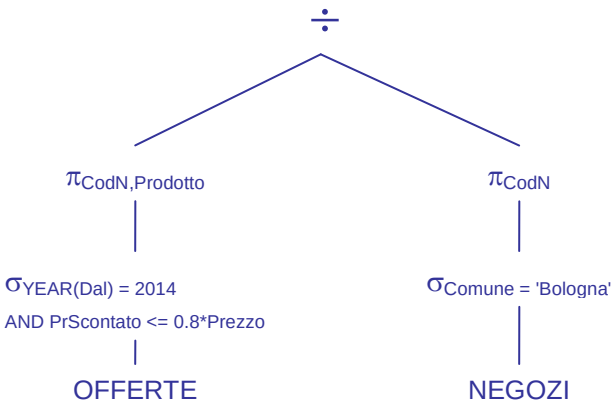
```
NEGOZI(CodN,Indirizzo,Comune);
OFFERTE(Prodotto,CodN,Dal,Al,Prezzo,PrScontato),
      CodN REFERENCES NEGOZI;
VENDITE(Prodotto,CodN,Data,Numero),
      CodN REFERENCES NEGOZI;
--
-- Prezzo e PrScontato sono di tipo DEC(6,2), PrScontato < Prezzo
-- Dal e Al sono date (di uno stesso anno) che definiscono
-- 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 dati dei negozi che nel 2014 hanno fatto almeno un'offerta per il prodotto P015 vendendone (in un giorno di quell'offerta) almeno 20 pezzi



1.2) [2 p.] I prodotti che sono stati messi in offerta nel 2014 da tutti i negozi di Bologna con almeno il 20% di sconto



Sistemi Informativi T
6 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.] I prodotti che sono stati messi in offerta nel 2014 da tutti i negozi di Bologna con almeno il 20% di sconto

```
SELECT   O.Prodotto
FROM     OFFERTE O, NEGOZI N
WHERE    N.CodN = O.CodN
AND      N.Comune = 'Bologna'
AND      YEAR(O.Dal) = 2014
AND      O.PrScontato <= 0.8*O.Prezzo
GROUP BY O.Prodotto
HAVING   COUNT(DISTINCT O.CodN) =
                  (  SELECT COUNT(*)
                     FROM   NEGOZI N1
                     WHERE  N1.Comune = 'Bologna'  )

-- Fondamentale usare COUNT(DISTINCT O.CodN), perché uno stesso negozio
-- può mettere più volte in offerta lo stesso prodotto
```

2.2) [3 p.] I giorni e i negozi in cui ogni prodotto in offerta quel giorno in quel negozio è stato venduto almeno 5 volte

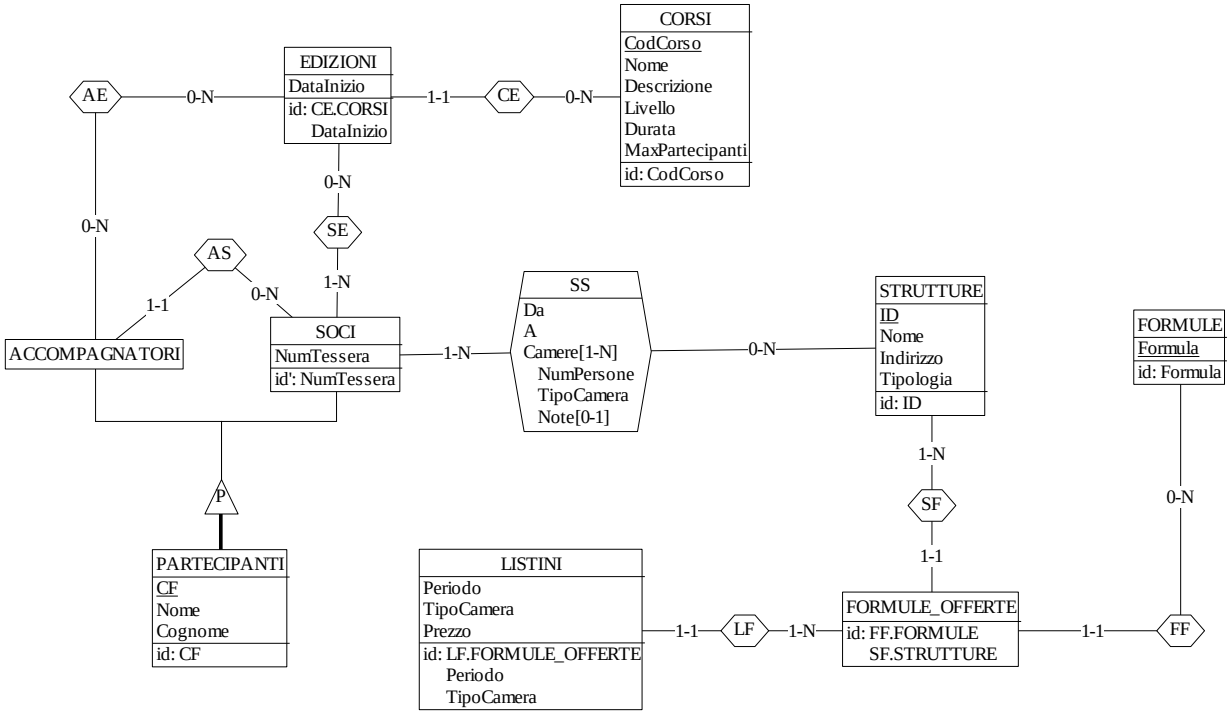
```
SELECT   V.CodN, V.Data
FROM     OFFERTE O, VENDITE V
WHERE    V.CodN = O.CodN
AND      V.Data BETWEEN O.Dal AND O.Al
AND      V.Numero >= 5
GROUP BY V.CodN, V.Data
HAVING   COUNT(DISTINCT V.Prodotto) = COUNT(DISTINCT O.Prodotto)

-- L'esercizio è solo apparentemente complesso, come dimostra la soluzione
-- riportata che sfrutta ancora la forma COUNT(DISTINCT ...).
-- Il join tra OFFERTE e VENDITE non viene eseguito su Prodotto e quindi, per
-- ogni coppia (V.CodN, V.Data), genera un numero di tuple pari al prodotto
-- tra il numero di prodotti venduti (in quantità >= 5) e il numero di
-- prodotti in offerta in quel giorno e in quel negozio
--
-- Una soluzione alternativa avrebbe potuto fare uso di common table
-- expression per il calcolo di tali quantità, complicando però inutilmente
-- la soluzione
```

Sistemi Informativi T
6 febbraio 2015
Risoluzione

3) Progettazione concettuale (6 punti)

Lo sci club K2 organizza settimane bianche per i suoi soci. All'atto dell'iscrizione un socio deve scegliere dove soggiornare, per quanto tempo, che corsi seguire, e le eventuali persone (non soci) con cui intende viaggiare. Ogni corso ha un numero massimo di partecipanti, un livello di difficoltà e un numero di edizioni, tutte della stessa durata ma svolte in periodi differenti (ad es. il corso base dura 7 giorni e ammette non più di 10 persone). Un socio che si iscrive a un corso può iscrivere, per la stessa edizione, anche una o più delle persone che lo accompagnano. Per ospitare soci e accompagnatori, K2 è convenzionato con una serie di strutture ricettive (alberghi, pensioni, B&B, ecc.). Ogni struttura ha un suo listino prezzi, che dipende dal periodo, dal tipo di camera scelta e dalla formula di soggiorno (solo colazione, mezza pensione, pensione completa). Non tutte le formule sono disponibili in tutte le strutture. Il sistema informativo di K2 non si occupa di gestire le disponibilità di camere delle diverse strutture, ma registra solamente le prenotazioni effettuate. Inoltre non mantiene informazioni di tipo storico (prenotazioni passate, corsi già seguiti, ecc.).



Commenti:

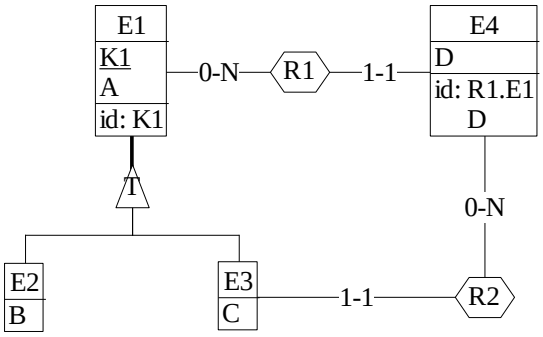
- Nonostante i molti elementi presenti, l'esercizio non presenta difficoltà di rilievo, anche perché alcuni vincoli presenti nelle specifiche non sono modellabili in uno schema E/R. Ad esempio:
  - *Un socio che si iscrive a un corso può iscrivere, per la stessa edizione, anche una o più delle persone che lo accompagnano.*
- L'entità FORMULE_OFFERTE, che mantiene le formule offerte da ciascuna struttura, è ottenuta per reificazione, al fine di poter identificare gli elementi di LISTINI usando solo combinazioni (struttura,formula) effettivamente disponibili.
- Non essendo disponibile un dettaglio delle camere delle diverse strutture, il sistema di K2 mantiene (in SS) informazioni minimali sulle prenotazioni. Poiché le specifiche su questo punto lasciano ampia libertà di interpretazione, si è proposta una soluzione in cui, per ogni camera prenotata, viene riportato quante persone la occuperanno, il tipo della camera, ed eventuali note (ad es. culla bambino).
- Si è ipotizzato che un socio possa gestire solo una prenotazione (intesa come 1 o più corsi, 1 o più strutture). Se tale ipotesi viene abbandonata allora bisogna considerare che gli accompagnatori di un socio possono variare da una prenotazione all'altra. Analogamente, si è assunto che un accompagnatore sia associato a un solo socio.
- La cardinalità minima di SOCI in SS, pari a 1, assume che il sistema che si sta modellando riguardi solo la gestione delle settimane bianche, e non anche altre cose di interesse dello sci club. Con questa ipotesi è lecito gestire solo i soci che hanno fatto una prenotazione.

Sistemi Informativi T
6 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) le associazioni R1 e R2 non vengono tradotte separatamente;
c) le entità E1, E2 ed E3 vengono tradotte insieme;
d) un'istanza di E4 che partecipa a R2 non può essere identificata esternamente da un'istanza di E1 che appartiene sia a E2 che a E3;



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,
TIPO2 SMALLINT NOT NULL CHECK (TIPO2 IN (0,2)),        -- 2: istanza di E2
TIPO3 SMALLINT NOT NULL CHECK (TIPO3 IN (0,3)),        -- 3: istanza di E3
B INT,
C INT,
K1R2 INT,
D INT,
CONSTRAINT TOTALE CHECK (TIPO2 = 2 OR TIPO3 = 3),
CONSTRAINT E2 CHECK ((TIPO2 = 0 AND B IS NULL) OR    (TIPO2 = 2 AND B IS NOT NULL)  ),
CONSTRAINT E3 CHECK (
     (TIPO3 = 0 AND C IS NULL AND K1R2 IS NULL AND D IS NULL) OR
     (TIPO3 = 3 AND C IS NOT NULL AND K1R2 IS NOT NULL AND D IS NOT NULL) )                    );

CREATE TABLE E4 (
K1 INT NOT NULL REFERENCES E1,
D INT NOT NULL,
PRIMARY KEY (K1,D)                    );

ALTER TABLE E1
     ADD CONSTRAINT FK_R2 FOREIGN KEY (K1R2,D) REFERENCES E4 ;
```

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 garantire 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.TIPO3 = 3 AND EXISTS (   SELECT * FROM E4, E1
                                 WHERE (E4.K1, E4.D) = (N.K1R2,N.D)
                                 AND E4.K1 = E1.K1
                                 AND E1.TIPO2 = 2
                                 AND E1.TIPO3 = 3  ) )
SIGNAL SQLSTATE '70001' ('La tupla inserita referenzia una tupla di E4 identificata da una tupla di E1che
appartiene sia a E2 che a E3!');
```