Sistemi Informativi T
20 febbraio 2017
Risoluzione

Tempo a disposizione: 2:30 ore

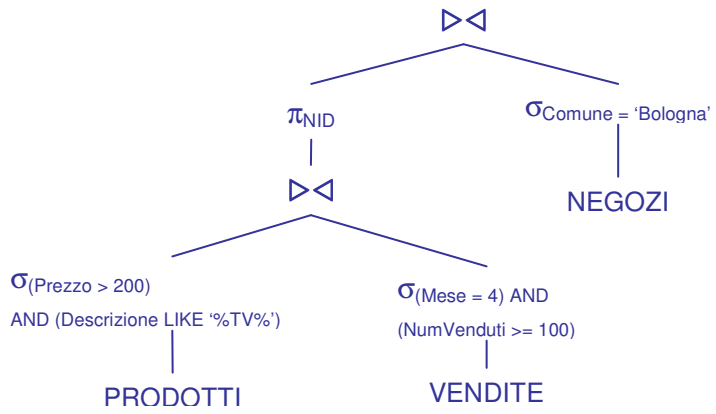
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

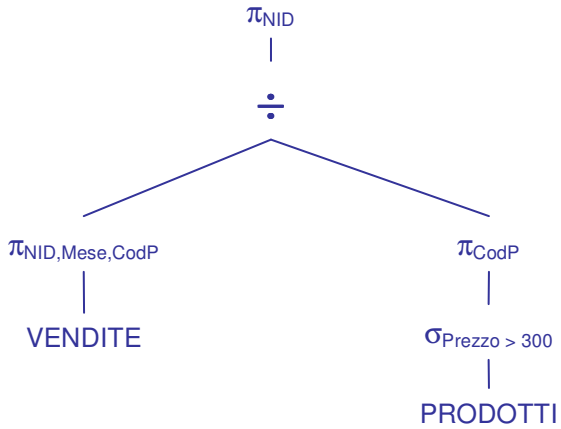
```
NEGOZI(NID,Indirizzo,Comune);
PRODOTTI(CodP,Descrizione,Prezzo);
VENDITE(NID,CodP,Mese,NumVenduti),
      NID REFERENCES NEGOZI,
      CodP REFERENCES PRODOTTI;
--
-- Mese è un intero compreso tra 1 e 12.
-- NumVenduti è un intero maggiore di zero che indica quante
-- unità (pezzi) di un prodotto un negozio ha venduto in un mese.
-- Prezzo è in formato DEC(6,2).
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) **[1 p.]** I dati dei negozi di Bologna che nel mese di Aprile (4) hanno venduto almeno 100 pezzi di una 'TV' che costa più di 200€



1.2) **[2 p.]** Gli identificatori dei negozi che in un mese hanno venduto almeno un pezzo di ogni prodotto che costa più di 300€



Come ogni volta che si usa l'operatore di divisione, è fondamentale definire correttamente gli schemi del dividendo e del divisore. In questo caso il divisore è un insieme di codici di prodotti (tutti quelli che costano più di 300€), e il dividendo include anche il Mese. In questo modo vengono selezionate le coppie (NID,Mese) accoppiate a tutti i valori del divisore, ovvero per ogni negozio tutti i mesi in cui ha venduto almeno un pezzo di ogni prodotto che costa più di 300€.

Sistemi Informativi T
20 febbraio 2017
Risoluzione

SQL (5 punti totali)

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

2.1) [2 p.] Per ogni mese, i negozi che hanno complessivamente incassato più di 500000 € (cinquecentomila €) per almeno un prodotto. Il risultato va ordinato per mese

```
SELECT   DISTINCT V.Mese,V.NID
FROM     PRODOTTI P, VENDITE V
WHERE    P.CodP = V.CodP
AND      P.PRezzo*V.NumVenduti > 500000
ORDER BY V.Mese;
```

Ogni tupla costruita mediante join nella clausola WHERE fornisce informazioni sulle vendite di un prodotto in un negozio in un certo mese, da cui si ottiene facilmente il risultato voluto.

2.2) [3 p.] Per ogni prodotto, il numero di mesi in cui c'è stato almeno un negozio che per quel prodotto ha incassato più di 100000 € (centomila €)

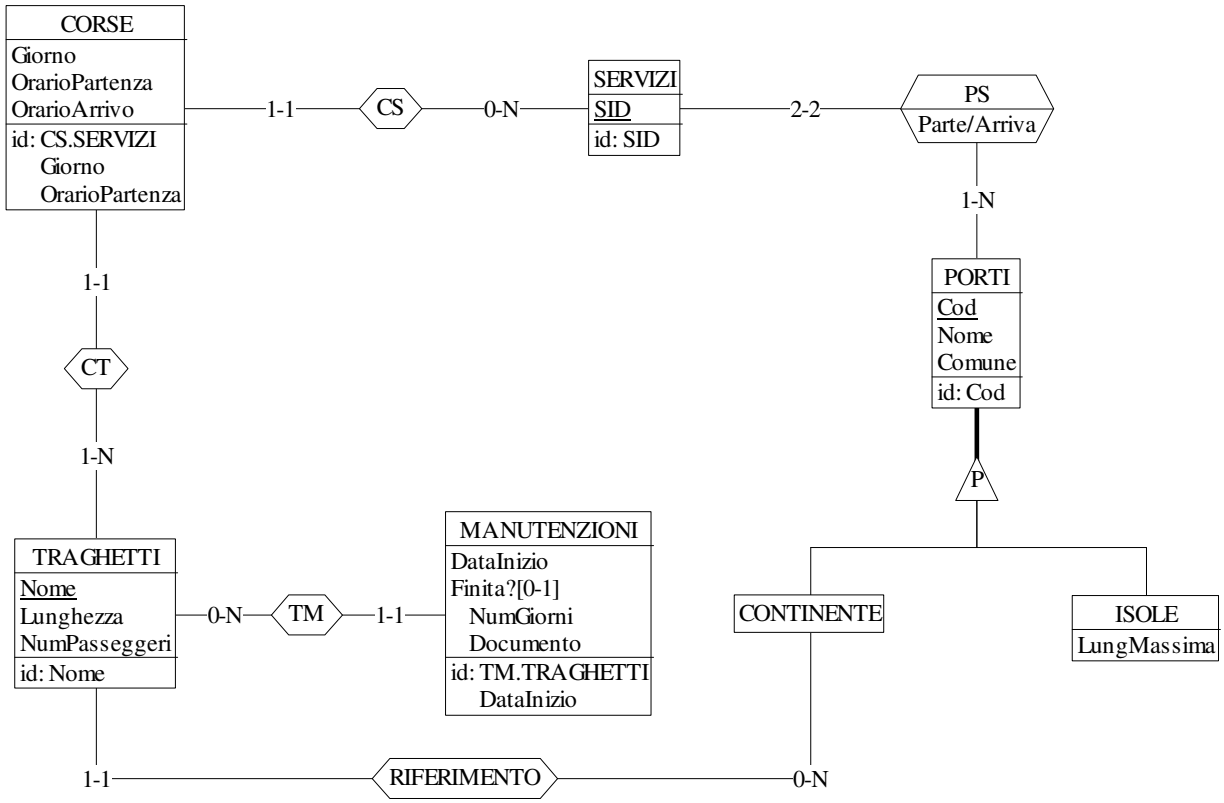
```
WITH INCASSIMESE (CodP, Mese) AS (
   SELECT DISTINCT V.CodP, V.Mese
   FROM  PRODOTTI P, VENDITE V
   WHERE P.CodP = V.CodP
   and   P.Prezzo*V.NumVenduti > 100000      )

SELECT   I.CodP, COUNT(*) AS NumMesi
FROM     INCASSIMESE I
GROUP BY I.CodP;
```

La c.t.e. prima mantiene le sole terne (CodP,NID,Mese) che soddisfano la specifica, poi proietta su (CodP,Mese) eliminando i duplicati.
In alternativa si sarebbe dovuto scrivere COUNT(DISTINCT I.Mese) anziché COUNT(*).
E' importante eliminare i duplicati (prima o dopo) perché una coppia (CodP,Mese) può presentarsi più volte, se in quel mese il prodotto ha superato la soglia in più negozi.

Sistemi Informativi T
20 febbraio 2017
Risoluzione

3) Progettazione concettuale (6 punti)

La compagnia di navigazione MariNar (MN) dispone di servizi di traghetti per tutte le isole italiane. Ogni servizio ha un porto di partenza e uno di arrivo. I porti hanno un codice univoco, un nome, e un comune in cui si trovano. I porti si dividono tra quelli sul continente e quelli sulle isole. Per questi ultimi è nota anche la lunghezza massima dei traghetti che possono attraccare nel porto.
Per ogni servizio sono dati i giorni della settimana in cui è attivo, con i relativi orari di partenza e arrivo della singola corsa (in un giorno ci possono essere anche più corse). Per ogni corsa è specificato il traghetto utilizzato. Ogni nave traghetto, oltra ad avere un nome (univoco), una lunghezza e una capacità massima di passeggeri, ha anche un porto (sul continente) cui fare riferimento per le manutenzioni periodiche (per ogni manutenzione si registrano il giorno di inizio, il numero di giorni impiegati e un documento che riassume quanto fatto).



Commenti:

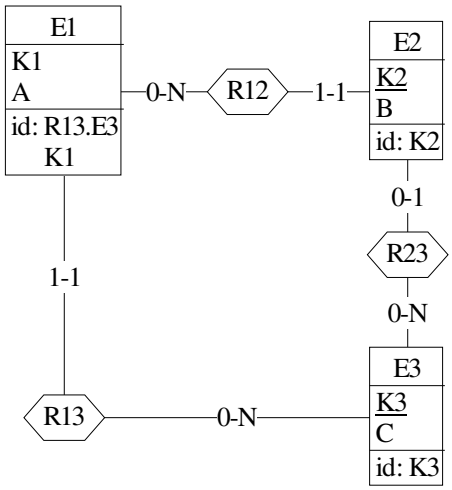
- L'attributo Parte/Arriva ha 2 valori possibili, che servono a diversificare il ruolo di un porto in quel servizio. Rispetto alla soluzione "standard" con 2 associazioni (entrambe 1-1 per SERVIZI), questa soluzione ha il vantaggio di garantire che i 2 porti sono diversi, ma, come detto, richiede di garantire la corretta specifica dei valori di Parte/Arriva
- Anziché modellare MANUTENZIONI come entità a sé si sarebbe potuto inserire un attributo composto e ripetuto con cardinalità (0,N) in RIFERIMENTO. Tuttavia, così facendo, non si sarebbe potuto specificare che le manutenzioni di uno stesso traghetto iniziano in date diverse.

Sistemi Informativi T
20 febbraio 2017
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) nessuna associazione viene tradotta separatamente;
c) un'istanza di E2 associata, tramite R23, a un'istanza di E3 non può essere associata a un'istanza di E1, tramite R12, identificata esternamente dalla stessa istanza di E3;
d) un'istanza di E1 può essere identificata esternamente solo da istanze di E3 con C > A;



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 E3 (
K3  INT NOT NULL PRIMARY KEY,
C    INT NOT NULL                          );

CREATE TABLE E1 (
K1  INT NOT NULL,
A    INT NOT NULL,
K3  INT NOT NULL REFERENCES E3,
PRIMARY KEY (K1,K3)                        );

CREATE TABLE E2 (
K2  INT NOT NULL PRIMARY KEY,
B    INT NOT NULL,
K1  INT NOT NULL,
K3R12 INT NOT NULL,
K3R23  INT REFERENCES E3,
FOREIGN KEY  (K1,K3R12) REFERENCES E1,
CONSTRAINT PUNTO_C CHECK (K3R12 <> K3R23)                    );
```

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 ';')

```
CREATE TRIGGER PUNTO_D
BEFORE INSERT ON E1
REFERENCING NEW AS N
FOR EACH ROW
WHEN ( EXISTS (     SELECT * FROM E3 WHERE E3.K3 = N.K3 AND E3.C <= N.A        ) )
SIGNAL SQLSTATE '70001' ('Il valore di C deve essere maggiore di A!');
```