Sistemi Informativi T
11 settembre 2012
Risoluzione

Tempo a disposizione: 2:30 ore

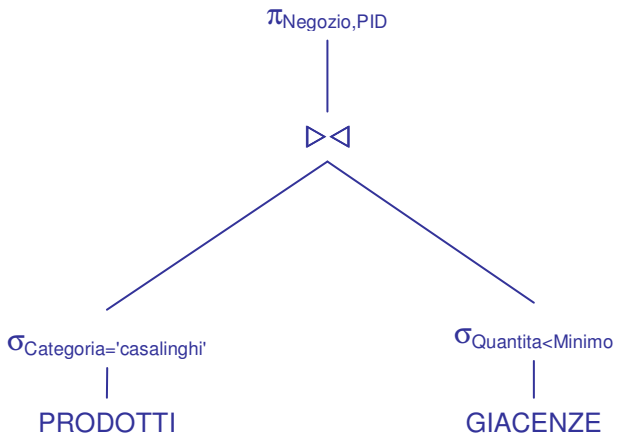
1) Algebra relazionale (3 punti totali):

Date le seguenti relazioni:

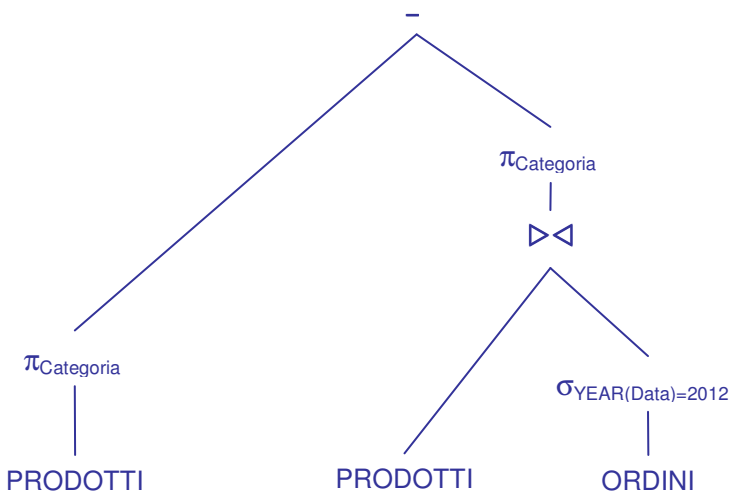
```
PRODOTTI(PID,Categoria,Prezzo);
GIACENZE(Negozio,PID,Quantita,Minimo),
PID REFERENCES PRODOTTI;
ORDINI(Negozio,PID,Data,QtaOrdinata),
Negozio,PID REFERENCES GIACENZE;
-- Minimo = valore minimo di quantita' che un dato negozio stabilisce
-- per decidere quando riordinare un dato prodotto
-- Quantita, Minimo e QtaOrdinata sono di tipo INT
```

si scrivano in algebra relazionale le seguenti interrogazioni:

1.1) [1 p.] I negozi e relativi prodotti di categoria 'casalinghi' per cui deve essere eseguito un ordine



1.2) [2 p.] Le categorie per cui nel 2012 non è stato eseguito nessun ordine



Sistemi Informativi T
11 settembre 2012
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 ordinato almeno una volta, il negozio per cui il valore di (`QtaOrdinata-Minimo`) è stato il maggiore

```
SELECT   G.PID, G.Negozio
FROM     GIACENZE G, ORDINI O
WHERE    G.Negozio = O.Negozio
AND      G.PID = O.PID
WHERE    O.QtaOrdinata - G.Minimo =
         ( SELECT    MAX(O1.QtaOrdinata - G1.Minimo)
           FROM      GIACENZE G1, ORDINI O1
           WHERE     G1.Negozio = O1.Negozio
           AND       G1.PID  = O1.PID
           AND       O1.PID = O.PID    )

-- Una soluzione alternativa consiste ovviamente nel definire una
-- Common Table Expression
```

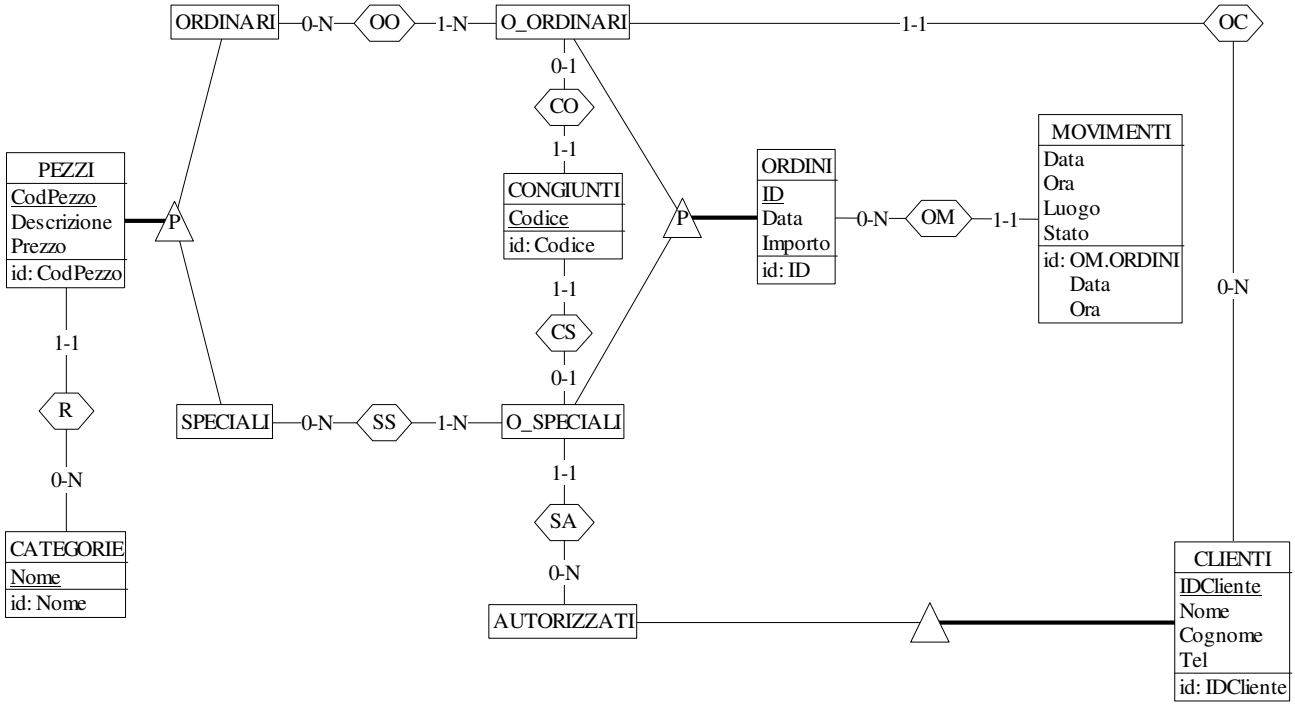
2.2) [3 p.] Per ogni categoria, il prodotto che è stato complessivamente (quindi considerando tutti gli ordini) ordinato in quantità maggiore

```
WITH TOTORDINI (Categoria, PID, TotQta) AS (
   SELECT O.Relatore,O.Argomento, SUM(O.QtaOrdinata)
   FROM   PRODOTTI P, ORDINI O
   WHERE  P.PID = O.PID
   GROUP BY  P.Categoria,P.PID )
SELECT T.*
FROM   TOTORDINI T
WHERE  T.TotQta = (  SELECT MAX(T1.TotQta)
                     FROM   TOTORDINI T1
                     WHERE  T1.Categoria = T.Categoria )
```

Sistemi Informativi T
11 settembre 2012
Risoluzione

3) Progettazione concettuale (6 punti)

La ditta di ricambi TuttoDunPezzo (TDP) mette a disposizioni una vasta gamma di pezzi per veicoli spaziali. Alcune categorie di pezzi "speciali", tuttavia, sono riservate solo a clienti autorizzati. Se si vogliono ordinare assieme pezzi speciali e altri (ordinari), è necessario eseguire due ordini separati (che però hanno un codice che li mette in relazione). Per ogni ordine viene mantenuta traccia dettagliata dei movimenti seguiti dalla merce in fase di consegna. Ogni movimento è caratterizzato da data, ora e luogo, oltre che da un'informazione di "stato" che descrive il movimento stesso (ad es. partenza, arrivo, consegna, ecc.).



Commenti:

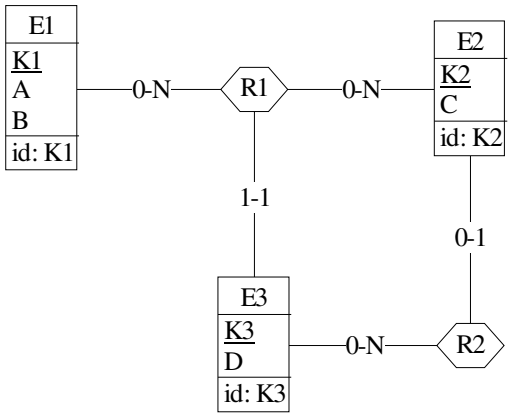
- Non è rappresentato il vincolo che un ordine di pezzi ordinari che è collegato a un ordine di pezzi speciali è necessariamente eseguito da un cliente autorizzato.
- Si noti che associare AUTORIZZATI a ORDINI anziché O_SPECIALI sarebbe errato, in quanto si verrebbe a creare un doppio collegamento tra i clienti autorizzati e gli ordini di prodotti ordinati, con conseguente ridondanza.
- L'entità CONGIUNTI è ottenuta per reificazione, allo scopo di rappresentare l'unicità dei valori di Codice.

Sistemi Informativi T
11 settembre 2012
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 associazione R1 e R2 non vengono tradotte separatamente;
c) per le solo istanze di E2 che partecipano a R2, esiste una dipendenza funzionale da C ad A (tramite R2 e R1);



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,
B INT NOT NULL    );

CREATE TABLE E2 (
K2 INT NOT NULL PRIMARY KEY,
C INT NOT NULL,
K3R2 INT    );

CREATE TABLE E3 /
K3 INT NOT NULL PRIMARY KEY,
D INT NOT NULL,
K1R1 INT NOT NULL REFERENCES E1,
K2R1 INT NOT NULL REFERENCES E2    );

ALTER TABLE E2
ADD CONSTRAINT FKR2 FOREIGN KEY (K3R2) REFERENCES E3;
```

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

```
-- Per garantire il rispetto del vincolo di cui al punto c) è necessario impostare il seguente trigger:
CREATE TRIGGER FD_C_A
NO CASCADE BEFORE INSERT ON E2
REFERENCING NEW AS N
FOR EACH ROW
WHEN (EXISTS (    SELECT *
                  FROM E2, E3 E3X, E1 E1X, E3 E3Y, E1 E1Y
                  WHERE E2.C = N.C
                  AND E2.K3 = E3X.K3
                  AND E3X.K1R1 = E1X.K1
                  AND N.K3 = E3Y.K3
                  AND E3Y.K1R1 = E1Y.K1
                  AND E1X.A <> E1Y.A) )
SIGNAL SQLSTATE '70001' ('La tupla non rispetta la FD da C ad A!')@
-- Si noti che E3 ed E1 vengono usate 2 volte, una per trovare il valore di A (E1X.A) associato a un dato valore di C,
-- l'altra per trovare il valore di A (E1Y.A) associato alla nuova tupla inserita in E2
```