CSci 4707
Homework 2 Solution

Spring 2015

Chapter 3, 4 and 5

Due Thursday, 02/26/2015 14:30

B1a.
```
CREATE TABLE Customer(custid: Integer, PRIMARY KEY(custid));
CREATE TABLE Products(pid: Integer, PRIMARY KEY(pid));
CREATE TABLE Buy(custid: Integer, pid: Integer, did: Integer,
                PRIMARY KEY(custid, pid),
                FOREIGN KEY(custid) REFERENCE Customer(custid)
                FOREIGN KEY(pid) REFERENCE Products(pid));
```

B1b.
```
CREATE TABLE Customer(custid: Integer, PRIMARY KEY(custid));
CREATE TABLE Products(pid: Integer, PRIMARY KEY(pid));
CREATE TABLE Date(did: Integer, PRIMARY KEY(did));
CREATE TABLE Buy(custid: Integer, pid: Integer, did: Integer,
                PRIMARY KEY(custid, pid, did),
                FOREIGN KEY(custid) REFERENCE Customer(custid),
                FOREIGN KEY(pid) REFERENCE Products(pid),
                FOREIGN KEY(did) REFERENCE Date(did));
```

B1c. Same as B1a. We cannot keep track participation constraint using PRIMARY KEY AND FOREIGN KEY.

B1d.
```
CREATE TABLE Products(pid: Integer, PRIMARY KEY(pid));
CREATE TABLE Customer_Buy(custid: Integer, pid: Integer NOT NULL,
                did: Integer,
                PRIMARY KEY(custid, did),
                FOREIGN KEY(pid) REFERENCE Products(pid),
                FOREIGN KEY(did) REFERENCE Date(did));
```

B2.
```
CREATE TABLE Publishers(name: String, PRIMARY KEY(name));
CREATE TABLE Authors(name: String, PRIMARY KEY(name));
CREATE TABLE Categories(name: String, PRIMARY KEY(name));
CREATE TABLE Books(ISBN: String, name: String,
                pname: String NOT NULL,
                PRIMARY KEY(ISBN),
                FOREIGN KEY (pname) REFERENCE Publishers(name));
CREATE TABLE Authored_By(ISBN: String, aname: String,
                PRIMARY KEY(ISBN, sname),
                FOREIGN KEY(ISBN) REFERENCE Products(ISBN),
                FOREIGN KEY(sname) REFERENCE Authors (name));
CREATE TABLE Belongs_To(ISBN: String, cname: String,
                PRIMARY KEY(ISBN, cname),
                FOREIGN KEY(ISBN) REFERENCE Products(ISBN),
                FOREIGN KEY(cname) REFERENCE Categories(name));
CREATE TABLE Parent_Of(parent_name: String, child_name: String,
        PRIMARY KEY(child_name),
        FOREIGN KEY(parent_name) REFERENCE Categories(name),
        FOREIGN KEY(child_name) REFERENCE Categories(name));
```

B3.
```
CREATE TABLE Professor(SSN: Integer, Salary: Integer, Phone: Integer,
        PRIMARY KEY(SSN));
CREATE TABLE Department(DNO: Integer, Name: String, Budget: Integer,
        chair_SSN: String NOT NULL,
        PRIMARY KEY(DNO),
        FOREIGN KEY(chair_SSN) REFERENCE Professor(SSN));
CREATE TABLE Grad_Student(Name: String,
        advisor_SSN: Integer NOT NULL,
        PRIMARY KEY(Name),
        FOREIGN KEY(advisor_SSN) REFERENCE Professor(SSN)
                ON DELETE CASCADE);
```

C1. S = Student, O = OneStop, C = Course

C1a. $\pi_{snames}((S \infty O \infty \sigma_{cname="RDBMS"} C) \cap (S \infty O \infty \sigma_{cname="NoSQL"} C))$

C1b. $\pi_{snames}(S \infty ((\pi_{sid, cid} O) / (\pi_{cid} \sigma_{department="Computer Science"} C)))$

C1c. $\pi_{O1.cid} \sigma_{O1.cid = O2.cid \wedge O1.sid != O2.sid} (O1 X O2)$

C1d. $\pi_{S1.cid, S2.sid} (\sigma_{S1.year > S2.year} (S1 X S2))$

C1e. $\rho (R1, ((\pi_{sid, cid} O) / \pi_{cid} \sigma_{department = "Computer Science"} C)))$

$\rho (R1, ((\pi_{sid, cid} O) / \pi_{cid} \sigma_{department = "Electrical Engineering"} C)))$

$R1 \cup R2$

C2. S = SuppInfo, P = Purchases

C2a. ρ (S1, SuppInfo)

ρ (S2, SuppInfo)

$\rho (AllPairs, \pi_{s1, s2}(\rho (1\text{->}s1, 3\text{->}s2), S1 \infty_{S1.suppid < S2.suppid} S2))$

$\rho (Temp1, \pi_{s1, s2, prodid}(\sigma_{All Pairs.s1 = S1.suppid}(All Pairs X S1))$

$\rho (Temp2, \pi_{s1, s2, prodid}(\sigma_{All Pairs.s2 = S1.suppid}(All Pairs X S1))$

$\rho (BadPairs, \pi_{s1, s2}((Temp1 U Temp2) - (Temp1 \cap Temp2)))$

AllPairs - BadParis

Explanation:

What is going on here is that we compute Temp1 to contain all triples (a, b, x) where a and b are suppliers and a supplies product x; in Temp2 we get all triples (a, b, y) where a and b are suppliers and b supplies product y. If a and b both supply some product, say z, there will be (a, b, z) in both Temp1 and Temp2. Line 6 will contain all tuples (a, b, w) where either a supplies w but b doesn't, or vice versa; in this case, the pair (a, b) should be excluded from the final result because they do not supply the same products.

C2b. ρ (P1, Purchases)

ρ (P2, Purchases)

ρ (P3, Purchases)

ρ (P4, $\pi_{\text{custid}}(\sigma_{\text{P1.custid = P2.custid} \wedge \text{P1.pm != P2.pm}}$(P1 × P2))

ρ (P5, $\pi_{\text{custid}}(\sigma_{\text{P1.custid = P2.custid = P3.custid} \wedge \text{P1.pm != P2.pm != P3.pm}}$(P1 × P2 × P3))

P4 – P5

C3. ρ (OKBars, $\pi_{\text{did, BarId}}$(Likes ∞ Serves))

ρ (BadDid, π_{did}(Frequent - OKBars))

π_{did} (Drinkers - BadDid)

D1a.
```
SELECT suppid1, suppid2
FROM (SELECT S1.suppid as suppid1,
             S2.suppid as suppid2,
             S1.prodid AS prodid1,
             S2.prodid AS prodid2
      FROM SuppInfo S1, SuppInfo S2
      WHERE S1.suppid < S2.suppid AND S1.prodid = S2.prodid) AS TEMP
GROUP BY suppid1 , suppid2
HAVING (COUNT(*) = (SELECT COUNT(*)
                    FROM SuppInfo S3
                    WHERE S3.suppid = suppid1))
      AND (COUNT(*) = (SELECT COUNT(*)
                    FROM SuppInfo S4
                    WHERE S4.suppid = suppid2));
```

The idea here is to find suppliers that supply some prodcut(s) in common. For each pair, compute how many products they supply in common. Select the pairs (s1,s2) such that the total number of products served by s1 equals the number of products s1 and s2 supplied in common and equals the total number of products supplied by s2.

D1b.
```
SELECT custid
FROM purchases
GROUP BY custid
HAVING COUNT(DISTINCT(purchases.purchasemethod)) = 2;
```

D2.
```
SELECT S1.barId, S2.barId
FROM Serves S1, Serves S2
WHERE S1.beerId = S2.beerId AND S1.barId != S2.barId
GROUP BY S1.barId, S2.barId
HAVING COUNT(*) = (SELECT Count(*)
                   FROM Serves S WHERE S.barId = S1.barId)
     AND COUNT(*) = (SELECT Count(*)
                   FROM Serves S WHERE S.barId = S2.barId)
```

The idea here is to find bars that serve some beer(s) in common. For each pair, compute how many beers they serve in common. Select the pairs (b1,b2) such that the total number of beers served by b1 equals the number of beers b1 and b2 serve in common and equals the total number of beers served by b2.

D3.
```
SELECT C.cname
FROM Customer C
WHERE C.cid NOT IN (SELECT Buys.cid
                    FROM Buys
                    WHERE (SELECT COUNT(*) FROM Customer) =!
                    (SELECT COUNT(DISTINCT Buys2.cid)
                    FROM Buys2
                    WHERE Buys2.pid = Buys.pid));.
```

D4.
```
SELECT A.name
FROM Actors A
WHERE A.aid NOT IN (SELECT A1.aid
                    FROM Actors A1, Cast C1, Movies M1, Directors D1
                    WHERE A1.aid = C1.aid AND
                          M1.mid = C1.mid AND
                          M1.did = D1.did AND
                          D1.name != “Spielberg” );
```