Fall, 2004 CIS 550

Database and Information Systems
Homework 2 Solutions

Problem 1 [30 points]: Consider the following schema based on the TPC-H benchmark (which you'll hear more about later on in the course):

Part(*PartID*: int, *Name*: string, *Mfgr*: string, *Brand*: string, *Type*: string, *RetailPrice*: float)
Supplier(*SuppID*: int, *Name*: string, *Address*: string, *NationID*: int, *Phone*: string, *AcctBal*: float)
PartSupp(*PartID*: int, *SuppID*: int, *AvailQty*: int, *SupplyCost*: float)
Nation(*NationID*: int, *Name*: string, *RegionID*: int)
Region(*RegionID*: int, *Name*: string)

The key fields are underlined. Foreign keys are indicated by naming. (In other words, if x is the key of relation X, then each appearance of x outside of X is a foreign key referencing X.)

Write the following queries in SQL:

1. Find the IDs of parts manufactured by Dupont.

```
SELECT p.PartID
FROM Part p
WHERE p.Mfgr = 'Dupont';
```

2. Find the total quantity of parts available from American suppliers (nation name 'USA').

```
SELECT SUM(ps.AvailQty)
FROM Part p, PartSupp ps, Supplier s, Nation n
WHERE p.PartID = ps.PartID AND ps.SuppID = s.SuppID AND s.NationID =
n.NationID AND n.Name = 'USA';
```

3. Find the total number of nations.

```
SELECT COUNT(*)
FROM Nation;
```

4. $\{\langle n, r\rangle \mid \exists m, s \text{ s.t. } \langle n, m, r\rangle \in \text{Nation} \wedge \langle r, s\rangle \in \text{Region} \wedge s = \text{“Asia”}\}$

```
SELECT n.NationID, n.RegionID
FROM Nation n, Region r
WHERE n.RegionID = r.RegionID AND r.Name = 'Asia';
```

5. Find the IDs of suppliers whose total supply cost for all parts they supply is the highest of all suppliers in their region. (Ignore available quantity when computing total supply cost.)

```
SELECT s.SuppID
FROM Supplier s, PartSupp ps, Nation n
WHERE s.SuppID = ps.SuppID AND s.NationID = n.NationID
GROUP BY s.SuppID, n.RegionID
HAVING SUM(ps.SupplyCost) >= ALL (
       SELECT SUM(ps2.SupplyCost)
       FROM Supplier s2, PartSupp ps2, Nation n2
       WHERE s2.NationID = n2.NationID and n.RegionID = n2.RegionID
       AND s2.SuppID = ps2.SuppID
       GROUP BY ps2.SuppID);
```

Problem 2 [30 points]: Consider the following relations:

Student(*sid*: int, *name*: string, *email*: string)
Takes(*sid*: int, *expgrade*: char, *cid*: int)
Course(*cid*: int, *sem*: string)
Professor(*fid*: int, *name*: string, *email*: string)
Teaches(*fid*: int, *cid*: int)
CourseOffering(*cid*: int, *cno*: string, *sub*: string)

The key fields are underlined. Foreign keys are indicated by naming. Write the SQL statements required to create these relations, including appropriate versions of all primary and foreign key integrity constraints.

```
CREATE TABLE Student(sid INTEGER,
                     name VARCHAR(32),
                     email VARCHAR(64),
                     PRIMARY KEY (sid));

CREATE TABLE Course(cid INTEGER,
                    sem VARCHAR(8),
                    PRIMARY KEY (cid));

CREATE TABLE Takes(sid INTEGER,
                   expgrade VARCHAR(1),
                   cid INTEGER,
                   FOREIGN KEY (sid) REFERENCES Student,
                   FOREIGN KEY (cid) REFERENCES Course);

CREATE TABLE Professor(fid INTEGER,
                       name VARCHAR(32),
                       email VARCHAR(64),
                       PRIMARY KEY (fid));

CREATE TABLE Teaches(fid INTEGER,
                     cid INTEGER,
                     FOREIGN KEY (fid) REFERENCES Professor,
                     FOREIGN KEY (cid) REFERENCES Course);

CREATE TABLE CourseOffering(cid INTEGER,
```

```
cno VARCHAR(16),
sub VARCHAR(32),
FOREIGN KEY (cid) REFERENCES Course);
```

Problem 3 [40 points] Recall the schema for the PBAY system:

Sellers(*sellerID*: int, *rating*: char[2], *email*: string)
Items(*itemID*: int, *typeID*: int)
Stock(*itemID*: int, *startBid*: float, *qty*: int)
SoldBy(*itemID*: int, *sellerID*: int)
Description(*itemID*: int, *descr*: string)
Purchases(*purchaseID*: int, *itemID*: int, *custID*: int, *soldFor*: float, *quant*: int)
Customers(*custID*: int, *address*: string)

The key fields are underlined. Foreign keys are indicated by naming. ItemID is the vendor-specific code for an item, and typeID is its global identifier (e.g. ISBN).

Translate the following queries into SQL:

1. Find the descriptions of all items that are sold by some seller

```
SELECT d.descr
FROM description d, soldby b
WHERE d.itemID = b.itemID;
```

2. Find the IDs of the sellers of items stocked in quantity ≥ 3

```
SELECT b.sellerID
FROM stock s, soldby b
WHERE s.itemID = b.itemID AND s.qty >= 3;
```

3. Find the IDs of item types sold only by sellers whose email address is either `zives@cis.upenn.edu` or `tjgreen@cis.upenn.edu`

```
SELECT b.itemID
FROM soldby b, sellers s, items i
WHERE b.sellerID = s.sellerID AND b.itemID = i.itemID AND
s.email IN ('zives@cis.upenn.edu', 'tjgreen@cis.upenn.edu')
MINUS
SELECT b.itemID
FROM soldby b, sellers s, items i
WHERE b.sellerID = s.sellerID AND b.itemID = i.itemID AND
s.email NOT IN ('zives@cis.upenn.edu', 'tjgreen@cis.upenn.edu');
```

4. For every seller of an item whose description contains the substring "hits", list the ID of the most expensive item or items (determined by start bid) the seller stocks

```
SELECT b.itemID
FROM soldby b, sellers s, stock k
WHERE b.sellerID = s.sellerID AND k.itemID = b.itemID
AND b.sellerID IN (
  SELECT b2.sellerID
  FROM soldby b2, description d
  WHERE b2.itemID = d.itemID AND d.descr LIKE '%Hits%'
)
AND k.startBid >= ALL (
  SELECT k2.startBid
  FROM stock k2, soldby b2
  WHERE b2.sellerID = b.sellerID AND b2.itemID = k2.itemID
);
```

5. Find the addresses of buyers who have bought at least $10.00 worth of items. (Note that Purchases(..., 10.0, 3) means $10.00 paid for each of the three items, i.e., $30.00 total.)

```
SELECT c.address
FROM customers c, purchases p
WHERE c.custID = p.custID
GROUP BY p.custID, c.address
HAVING SUM(p.soldFor * p.quant) >= 10;
```

6. Find the IDs of sellers who are asking a higher start bid for some item than is average for that item type

```
SELECT s.sellerID
FROM sellers s, soldby b, stock k, items i
WHERE
s.sellerID = b.sellerID
AND b.itemID = k.itemID
AND i.itemID = k.itemID
AND k.startBid > ANY (
  SELECT AVG(k2.startBid)
  FROM stock k2, items i2
  WHERE k2.itemID = i2.itemID AND i.typeID = i2.typeID
);
```

7. $\sigma_{sellerID=10}(Sellers \bowtie Stock \bowtie Items)$

```
SELECT s.*, k.*, i.typeID
FROM sellers s, stock k, items i
WHERE k.itemID = i.itemID AND s.sellerID = 10;
```