Fall, 2005 CIS 550

Database and Information Systems

Homework 2

Due on 10/4/2005

For this assignment, you will want to test your queries by running them on a real database. So please begin by signing up for an Oracle account *(http://www.seas.upenn.edu/ora)*, accessible from eniac. (For those who do not have eniac accounts, please email the instructor.) Then read over the Oracle setup instructions from the course web page *(http://www.seas.upenn.edu/ zives/cis550/oracle-faq.html)* and modify your eniac .cshrc file as directed. Also read over the Oracle guide referenced from the course web pages *(http://www.cs.wisc.edu/ dbbook/openAccess/thirdEdition/Oracle/user guide/oracle guide.html)*. Finally, download **hw2.sql** to your eniac account, launch Oracle (using the command **sql**), and then **start hw2** to create some sample tables for Problems 1 and 3. These will only be sparsely populated to test your solutions, you may need to INSERT more VALUES into the tables.

Note 1: The SQL string datatype is **VARCHAR(*length*)**, and you'll need to choose an appropriate length.

Note 2: The key fields are underlined in schema. 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).

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):

Parts(*partID*: int, *name*: string, *mfgr*: string, *brand*: string, *type*: string, *retailprice*: float)
Suppliers(*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)

Write the following queries in SQL:

1. Find the IDs of suppliers with account balance < $1000.

```
Select id
  from Partsupp
```

```
 where acctbal < 1000
```

2. Find total number of suppliers in nation 'USA'.

```
Select count(s.*)
  from Suppliers s, Nation n
 where s.nationID = n.nationID
   and n.name = 'USA'
```

3. Find the brands of parts that are supplied by suppliers in region 'North America'.

```
Select p.brand
  from Parts p, PartSupp A, Suppliers s, Nation n, Region R
 where p.partID = A.partID
   and a.suppID = s.suppID
   and s.nationID = n.nationID
   and n.regionID = r.regionID
   and r.name = 'North America'
```

4. Find the IDs of parts which are supplied by 3 different suppliers in nation 'USA'.

```
Select a.partID
  from Partsupp A, Suppliers s, Nation n
 where A.suppID = s.suppID
   and s.nationID = n.nationID
   and n.name = 'USA'
  group by a.partID
  having count(distinct s.suppID) >=3
```

5. $\{\langle n\rangle | \exists s, a, t, h, b, p, e, m, r, y, i, v, c(\langle s, n, a, t, h, b\rangle \in supplier \wedge \langle p, s, v, c\rangle \in partsupp \wedge \langle p, e, m, r, y, i\rangle \in parts \wedge e =' widget' \wedge v > 50)\}$

```
Select s.name
  from Suppliers s, Partsupp a, Parts p
 where s.suppID = a.suppID
   and a.partID = p.partID
   and p.name = 'widget'
   and a.availqty > 50
```

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

Employee(*eid*: int, *ename*: string, *age*: int, *salary*: float)
Department(*did*: int, *budget*: float)
Manager(*mid*: int, *mname*: string)
Works(*eid*: int, *did*: int, *workinghours*:int)
Admins(*mid*: int, *did*: int)

Write the SQL DDL statements to create these relations, including all primary and foreign key integrity constraints.

```
create table Employee (
       eid     integer,
       ename   varchar(32),
       age     integer,
       salary  float,
       primary key (eid));

create table Department (
       did   integer,
       budget float,
       primary key (did));

create table Manager(
       mid   integer,
       mname varchar(32),
       primary key (mid));

create table Works (
       eid   integer,
       did   integer,
       workinghours integer,
       foreign key (eid) references Employee,
       foreign key (did) references Department);

create table Admins (
       mid   integer,
       did   integer,
       foreign key (mid) references Manager,
       foreign key (did) references Department);
```

Problem 3[40 points]: Use the schema from Homework 1's PBAY system:

Sellers(*sellerID*: int, *rating*: char, *email*: string)
Items(*itemID*: int, *type: string*)
Buyers(*buyerID*: int, *email*: string, *address*: string)
Stock(*itemID*: int, *sellerID*: int, *startBid*: float, *quantity*: int, *endingTime*: int)
Purchases(*itemID*: int, *buyerID*: int, *sellerID*: int, *price*: float, *purchaseQuantity*: int, *bidTime*: int)

Write the following queries in SQL:

1. Find the types of items that are in stock.

```
select i.type
  from Items i, Stock s
 where i.itemID = s.itemID
```

2. Find the IDs of sellers who either have sold some items or still have items in stock.

```
 select distinct p.sellerID
   from Purchases p
union
 select distinct s.sellerID
   from Stock s
```

3. Find the total price each buyer has paid for each item type. (Total price = price × purchase quantity. Also note that you don't need to return 0 for purchases that aren't made.)

```
Select p.buyerID, i.type, sum(p.price * p.purchaseQuantity)
  from Purchase p, Items i
 where p.itemID = i.itemID
 group by p.buyerID, i.type
```

4. Find the types of items stocked by ≥ 2 sellers but not bought by any buyer.

```
Select i.type
  from Items i, Stock s
 where i.itemID = s.itemID
 group by i.type
 Having count(s.sellerID)>=2
minus
 Select i.type
```

```
 from Items i, Purchase p
where i.itemID = p.itemID
```

5. Find the IDs of buyers who have purchased some item with a price lower than the average price for that item type.

```
Select distinct p.buyerID
  from Purchase p, Items i
 where p.itemID = i.itemID
   and p.price < ANY (select AVG(p2.price)
                        from Purchase p2, Items i2
                       where p2.itemID = i2.itemID
                         and i.type = i2.type );
```

   (Note that the ANY keyword is optional here, since there should only be one answer.)

6. Find the email of buyers who bought 3 same items from sellers who have a 'book' in stock.

```
Select b.email
  from Buyers b, Purchase p, Stock s, Items i
 where b.buyerID = p.buyerID
   and p.sellerID = s.sellerID
   and s.itemID = i.itemID
   and i.type = 'book'
   group by p.sellerID, p.buyerID, s.itemID, b.email
   having sum(p.purchaseQuantity)=3
```

7. $\pi_{rating}(\pi_{sid}(\sigma_{i1 \neq i2 \wedge s1=s2}(\rho_{itemID \to i1, sellerID \to s1}(Stock)$
   $\bowtie \rho_{itemID \to i2, sellerID \to s2}(\sigma_{quantity \geq 3}(Stock)))) \bowtie Sellers)$

```
Select l.rating
  from Sellers l, Stock s
 where l.sellerID = s.sellerID
   and s.quantity >=3
   and exists (select s2.sellerID
                 from Stock s2
                where s2.sellerID = s.sellerID
                  and s2.itemID <> s.itemID
               )
```