Fall, 2003 CIS 550

Database and Information Systems

Midterm

80 minutes, 80 pts; please look at **both sides of the paper.**
Question 5 is extra-credit and should be done last!

1. (10pts) Given the relations (where underlined attributes represent the key):

   Customer(<u>*custid*</u>: integer, *name*: string)
   Buys(<u>*custid*</u>: integer, <u>*itemno*</u>: integer, *rating*: char(2))
   Item(<u>*itemno*</u>: integer, *description*: string, *seller*: string)

   write a relational algebra expression to obtain the *custid*s and *name*s of customers who buy the item with *description* "*Memento movie*" from *seller* "*Hollywood Video*".

   $\pi_{\text{custid,name}}(\sigma_{\text{description=``Memento movie''} \wedge \text{seller=``Hollywood Video''}}(\text{Item}) \bowtie \text{Buys} \bowtie \text{Customer})$

2. (30pts) Given the relations from Problem 1, write SQL queries to compute the following:

   a. (10pts) All item *descriptions* and *sellers* for items rated "*A*" by ***any*** customer. Ensure there are no duplicate tuples in the output (hint: this can be done several ways).

   ```
   SELECT description, seller
   FROM Item I
   WHERE EXISTS (SELECT *
     FROM Buys B
     WHERE B.itemno = I.itemno AND rating = "A")
   ```

   b. (10pts) For each item, the *descriptions*, *sellers*, and the number of **Customer**s who bought that item.

   ```
   SELECT I.itemno, description, seller, COUNT(custid)
   FROM Item I, Buys B
   WHERE I.itemno = B.itemno
   GROUP BY I.itemno, description, seller
   ```

c. (10pts) Give (in English, not SQL!) an example of the *kind* of query that can be expressed in SQL but not with the relational algebra or relational calculus.

Perhaps the simplest example we've seen is aggregation. Recursion is another common example.

3. (20pts) Given the relation:

Data(*custid*: integer, *name*: string, *itemno*: integer, *count*: integer, *rating*: char(2), *cost*: real, *description*: string, *seller*: string)

and a minimal cover:

custid → *name*
itemno, *seller* → *cost*
itemno → *description*
itemno, *custid* → *rating*

a. (7pts) Is **Data** in 3NF?

No

b. (13pts) If not, decompose it into a set of relations that are in 3NF. Recall that the 3NF algorithm starts with the minimal cover, which is already provided for you.

(There was a typo in that the *count* field should not actually have been in the relation. I will ignore it in this list; some of you counted it as an additional key, which was fine.)

Customer(<u>custid</u>, name)
Cost(<u>itemno</u>, seller)
Description(<u>itemno</u>, description)
Rating(<u>itemno</u>, <u>custid</u>, rating

CustomerSaleSeller(<u>itemno</u>, <u>custid</u>, <u>seller</u>) determines Data

4. (20pts) Given the XML fragment:

```
<books>
  <story key= “123”>
    <author>Aesop</author>
    <title>The Hare and the Tortoise</title>
    <crossref>Aesops-fables</crossref>
   </story>
   <book key=“Aesops-fables”>
     <editor>Bob McBob</editor>
     <title>Aesop’s Fables</title>
     <publisher>Fabu-lous Publishers</publisher>
  </book>
…
</books>
```

a. (10pts) Write an XPath to return the editor of every book.

   /books/book/editor

b. (10pts) Write an XQuery to return a sequence of `<book-story>` elements containing the following data:

   - The `<editor>` and `<title>` element content of a book.
   - Nested within, all stories that have a `<crossref>` element whose content matches the key of the book.

```
for $bk in doc(“current-file.xml”)/books/book,
     $t in $bk/title,
     $e in $bk/editor,
     $k in $bk/@key/text()
return <book-story>
            { $e }
            { $t }
            {
             for $st in doc(“current-
file.xml”)/books/story,
                  $cref in $st/crossref/text()
             where $cref = $k
             return $st
            }
         </book-story>
```

5. (5 pts extra credit) You have seen both the XML and relational data models thus far. It has been claimed that the relational model is truly general – hence there is nothing that can be captured in XML that can't also be represented in relational data.

Suppose we have an XML document of the form:

```
<parts>
  <part key="123" name="WheelAssembly">
    <part key="124" name="Wheel">
      <part key="125" name="Bearing"></part>
      <part key="126" name="Disc"></part>
    </part>
    <part key="127" name="Tire"></part>
    <part key="128" name="AirStem"></part>
  </part>
</parts>
```

This is a so-called recursive XML schema. Such a schema can be easily encoded in a single relation; show the relational schema and how it would encode the above XML data.

R(key: integer, name: string, parent: integer)

Key	Name	Parent
123	WheelAssembly	(NULL)
124	Wheel	123
125	Bearing	124
126	Disc	124
127	Tire	123
128	AirStem	127