Fall, 2009 CIS 550

Database and Information Systems

Midterm Sample Solutions

There are 90 points total, with up to 20 points of extra credit.

Problem 1 [40 points]: Given our "bare bones" PennSocial schema:

UserAccounts(*userID*: int, *name*: string, *email*: string, *age*: int, *city*: string)
UserEducation(*userID*: int, *school*: string, *startYear*: int, *finishYear*: int)
UserInterests(*userID*: int, *interest*: string)
UserPartner(*userID*: int, *partnerUserID*: int, *marriedToPartner*: boolean)
ConnectedTo(*userID*: int, *friendUserID*: int, *privilegeLevel*: int)

Write each of the following queries.

(a) (10pts) Find the users interested in "swimming" who are studying at Penn (assume finishYear is the *predicted* year of degree completion). **Write this query in SQL**.

```
SELECT userID, name
FROM UserAccounts UA, UserEducation UE, UserInterests UI
WHERE UA.userID = UE.userID AND UA.userID = UI.userID
 AND UI.interest = 'swimming' AND UE.school = 'Penn'
 AND UE.finishYear > 2009
```

(b) (10pts) Find the IDs of users who have shared interests. **Write this query in the relational algebra.**

$\Pi_{userID,userID2}(UserInterest \bowtie_{interest=interest2 \wedge userID \neq userID2} (\rho_{userID,interest \rightarrow userID2,interest2}(UserInterest))$

(c) (10pts) Find the ID(s) of the "most popular" user(s), i.e., the one(s) connected to the most friends (note there can be a tie). **Write this query in the tuple relational calculus.**

Impossible: This query is not expressible in the TRC.

Note: this question will be counted as extra credit since the instructions to use the answer "Impossible" were inadvertently omitted on the text of the exam (though announced in class)

(d) (10pts) Find the user(s) who has(have) enrolled in the most schools. **Write this query in SQL.**

```
SELECT userID
FROM UserEducation UE
GROUP BY userID
HAVING COUNT(DISTINCT school) >= ALL(
  SELECT COUNT(DISTINCT school)
  FROM UserEducation
  GROUP BY userID
)
```

Problem 2 [40 points]: Suppose we have implemented within our PennSocial environment a notion of user mailboxes: a "private" inbox for messages (analogous to email) and a public "wall" where others can post (and where messages are visible to all friends).

Each message gets a message ID and a visibility level (private or public). Each word is given an ID, and may occur multiple times in each message.

We need to keep track of the following attributes:

R(userID,msgID,visibility,wordID,wordText,wordPosition)

You are given the following FDs:

$userID, msgID \rightarrow visibility$
$wordID \rightarrow wordText$
$wordText \rightarrow wordID$
$msgID, wordPosition \rightarrow wordID$
$msgID \rightarrow userID$
$msgID, wordID \rightarrow visibility$
$msgID, wordPosition \rightarrow wordText$

(a) (10pts) Specify a *minimal cover* for the functional dependencies.

$msgID \rightarrow visibility$
$wordID \rightarrow wordText$
$wordText \rightarrow wordID$
$msgID, wordPosition \rightarrow wordID$
$msgID \rightarrow userID$

(b) (5pts) Is the FD $wordID, msgID, visibility \rightarrow userID$ in the *closure* of the functional dependencies?

Yes: $msgID \rightarrow userID$.

(c) (10pts) Provide a relational schema in 3NF for the domain, using relational schema notation as above, and indicating keys with underlines under the attributes.

$MsgVisibility(\underline{msgID}, visibility)$
$Lexicon(\underline{wordID} \rightarrow wordText)$
$Occurs(\underline{msgID}, \underline{wordPosition}, wordID)$
$Owner(\underline{msgID}, userID)$

Is your decomposition lossless? Dependency preserving?

Yes, this is given by the properties of 3NF normalization.

(d) (10pts) Show an ER diagram for your relational schema. Indicate any participation constraints that make sense.

...

(e) (5pts) Write a relational algebra expression, putting parentheses around each pair of expressions being joined, that re-joins all of your normalized relations to build the single original relation R.

$$((MsgVisibility \bowtie Owner) \bowtie Occurs) \bowtie Lexicon$$

Problem 3 [20 points]: Briefly answer the following questions:

(a) (4 pts) What does *access path independence* mean?

It means the specification of the query is agnostic as to the means of requesting and reading the data — through an index, according to a particular sort order, etc.

(b) (4 pts) What is the connection between database *integrity constraints* and *functional dependencies*?

A database integrity constraint tries to enforce certain properties over attributes within tables. A common kind of constraint is the key, which is a special case of a functional dependency (the key determines the tuple).

(c) (4 pts) What is the key challenge in accessing a DBMS from a programming language?

The so-called impedance mismatch, which refers to the fact that the underlying abstractions in databases (tables, rows) and programming languages (classes, objects, variables) are not equivalent. This typically results in interfaces built on row sets (sometimes called result sets) and cursors.

(d) (4 pts) It has been argued that the advent of XML will make the relational data model and relational algebra obsolete. You have seen XQuery during lecture. How do the operations of XQuery *replace* (if it obsoletes) or *reuse* (if it doesn't obsolete) the basic relational algebra operations?

XQuery includes XPath expressions as a way of navigating through an XML document and selecting portions (binding to them). Once XQuery variables are bound, the standard relational algebra operations (select, project, join, rename) are in fact all present. Hence we see a case of reuse and extension, rather than replacement.

(e) (4 pts) Is normalization important in XML? Why or why not?

No, XML is deliberately non-normalized, primarily because its focus is not on updates and storage, but rather data export (sometimes to a human).

4. Challenge problem (bonus — tackle last). *This problem will add up to 10 points of extra credit if you scored* > 90 *points on the rest of the exam, and up to 5 points otherwise.*

Assume you are given an XML structure describing family ancestries (following each parent separately), like the following:

```
<familyTree>
 <person>
  <name>Jane Smith</name>
  <children>
   <person>
    <name>Hannah Smith-Nebraska</name>
    <children>
     <person>
       <name>Jane N. Nebraska</name>
       <children>
        <person>
          <name>Dakota Nebraska</name>
        </person>
       </children>
     </person>
    </children>
   </person>
   <person>
    <name>William Smith</name>
    <children>
     <person>
        <name>Nancy Smith</name>
     </person>
    </children>
   </person>
  </children>
 </person>
 <person>
   <name>Omaha Nebraska</name>
   <children>
     <person>
       <name>Dakota Nebraska</name>
     </person>
   </children>
 </person>
</familyTree>
```

Define a relational schema to capture this data (and variations thereof).

One can simply define a general XML-storage schema with two relations: $Element(ID, parentID, label)$ and $Text(parentID, value)$. There are many specialized alternatives as well, e.g.:

$Person(parentID, name)$