Fall, 2007 CIS 550

Database and Information Systems

Midterm

The exam is 80 minutes long. There are 100 points total.

Problem 1 [60 points]: Suppose we are building a Web crawler to index pages. We need to keep track of the following attributes:

R(docID,docURL,docTitle,docDate,linkedDocID,linkText,wordID,wordText)

where each document has an (integer) ID, URL, title, and date; documents link to other documents with a particular text label on their link; and documents contain words (each of which has a word ID) and text (which is potentially different for each word ID).

You are given the following FDs:

$docID \rightarrow docURL, docTitle, docDate$
$docID, linkedDocID \rightarrow linkText$
$wordID \rightarrow wordText$
$wordID, wordText \rightarrow wordID$
$docID, docDate \rightarrow docTitle$

1. (7pts) Specify the (or a) *minimal cover* for the functional dependencies.

   Answer:
   $docID \rightarrow docURL$
   $docID \rightarrow docTitle$
   $docID \rightarrow docDate$
   $docID, linkedDocID \rightarrow linkText$
   $wordID \rightarrow wordText$
   $docID, docDate \rightarrow docTitle$

2. (7pts) Is the FD $docID, wordID \rightarrow docDate, wordText$ in the *closure* of the functional dependencies?

   Answer:
   Yes.

3. (7pts) Provide a relational schema in 3NF for the domain, using the $R(A, B, C)$ notation, indicating keys with underlines under the attributes. Is the decomposition lossless? Dependency preserving?

   Answer:
   $docs(\underline{docID}, docURL, docDate)$
   $titles(\underline{docID}, \underline{docDate}, docTitle)$
   $words(\underline{wordID}, wordText)$
   $links(\underline{docID}, \underline{linkedDocID}, linkText)$
   $occurs(wordID, docID, linkedDocID)$. $occurs(wordID, docID)$ also works although it will not be the result of the normal 3NF Synthesis algorithm.
   The result is lossless and dependency preserving.

4. (8pts) Show an ER diagram for your relational schema. Indicate any participation constraints that make sense.

5. (7pts) Write a relational calculus expression that re-joins all of your normalized relations to build the single original relation R. You may use either the tuple or domain relational calculus.

   $\{Q | \exists d \in docs, t \in titles, w \in words, o \in occurs, l \in links \wedge d.docID = t.docID \wedge d.docDate = t.docDate \wedge o.docID = d.docID \wedge o.wordID = w.wordID \wedge d.docID = l.docID \wedge Q.docID = o.docID \wedge Q.docURL = d.docURL \wedge Q.docDate = d.docDate \wedge Q.docTitle = t.docTitle \wedge Q.wordID = o.wordID \wedge Q.wordText = w.wordText \wedge Q.linkedDocID = l.linkedDocID \wedge Q.linkText = l.linkText\}$

6. (24 pts) Write each of the following queries in (i) SQL, (ii) the relational algebra, if it can be expressed that way, and (iii) the tuple relational calculus, if it can be expressed that way.

   (a) (6pts) Find the documents in which the words "apartment" or "house" appear.
   SQL:

```
SELECT DISTINCT O.docID
FROM   words W, occurs O
WHERE  W.wordID = O.wordID
  AND  W.wordText = 'apartment'
UNION
SELECT DISTINCT O.docID
FROM   words W, occurs O
WHERE  W.wordID = O.wordID
  AND  W.wordText = 'house'
```

   Relational Algebra:
   $\Pi_{docID}(occurs \bowtie (\sigma_{wordText=\text{"apartment"} \vee wordText=\text{"house"}} words))$.

Tuple Relational Calculus:
$\{Q \mid (\exists W \in words, O \in occurs(W.wordID = O.wordID \wedge Q.docID = O.docID \wedge (W.wordText = \text{“apartment”} \vee W.wordText = \text{“house”})))\}$

(b) (6pts) Find the ID of the most-common word indexed in the database.

```
SELECT wordID
FROM   occurs
GROUP BY wordID
HAVING COUNT(*) >= ALL (
                          SELECT count(*)
                          FROM   occurs
                          GROUP BY wordID
                        )
```

It should be count(*) or count(docID). Many people wrote count(wordID) which is meaningless when you group by wordID.

(c) (6pts) Find the titles of the oldest documents in the database.
SQL:

```
SELECT T.docTitle
FROM   docs D, titles T
WHERE  D.docID = T.docID
  AND  D.docDate <= ALL (
                          SELECT docID
                          FROM   docs
                        )
```

Relational Algebra:
$\Pi_{docTitle}((\Pi_{docID}(docs) - \Pi_{d}ocID((\rho_{docID \rightarrow odID, docDate \rightarrow odDate}((\Pi_{docID, docDate}(docs))) \bowtie_{odDate < docDa}$
$docs)) \bowtie titles)$.

Tuple Relational Calculus:
$\{Q \mid \exists D1 \in docs, T \in titles((\forall D2 \in docs(D2.docDate \geq D1.docDate)) \wedge D1.docID = T.docID \wedge Q.docTitle = T.docTitle)\}$

(d) (6pts) Find the document IDs of documents containing the *second-most-common word* indexed in the database.

```
SELECT docID
FROM   occurs
WHERE  wordID NOT IN (
                        SELECT wordID
                        FROM   occurs
                        GROUP BY wordID
```

```
                        HAVING COUNT(*) >= ALL (
                                                  SELECT count(*)
                                                  FROM   occurs
                                                  GROUP BY wordID
                                                 )
                      )
GROUP BY wordID
HAVING COUNT(*) >= ALL (
                         SELECT count(*)
                         FROM   occurs
                         GROUP BY wordID
                         WHERE wordID NOT IN (
                                SELECT wordID
                                FROM   occurs
                                GROUP BY wordID
                                HAVING COUNT(*) >= ALL (
                                                  SELECT count(*)
                                                  FROM   occurs
                                                  GROUP BY wordID
                                                          )
                                               )
                          )
```

Problem 2 [20 points]: Given a relation **T**(A,B,C,D,E,F) and a set F of functional dependencies, $F = \{BC \rightarrow A, AC \rightarrow DE, F \rightarrow E\}$:

1. (6 pts) What are the candidate keys? *Answer:*
   BCF

2. (7 pts) What attributes are *not* in the attribute closure of BC? *Answer:*
   F

3. (7pts) Is T in 3NF? BCNF? 1NF (i.e., neither 3NF nor BCNF)? *Answer:*
   1NF

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

1. (4 pts) Provide at least two arguments for why XML is a useful model in lieu of, or in addition to, the relational model.

   Answer:
   XML can capture hierarchical data in a way that mirrors the way many people would like to think of their data. It also provides a way of importing and exporting data — part of a solution to interoperability.

2. (4 pts) What are two *physical* or *access path* properties that the relational database exploit, which are not visible at the logical (data model) level?

   Answer:
   Some examples would be indices, sort order, mirroring on a RAID system, how densely pages are packed with tuples.

3. (4 pts) Can one look at a database instance and determine a set of functional dependencies? Why or why not?

   Answer:
   No: FDs should depend on *any possible instance*, not just the one given to us. A user might modify some of the data, making some of the FDs we "inferred" by looking at the data invalid.

4. (4 pts) Between dependency preservation and losslessness, which is more important and why?

   Answer:
   Losslessness, as this ensures that the instance can be properly re-assembled without losing data. Dependency preservation is useful for validating new user input — but this is much less important than being able to keep the data that was given to us!

5. (4 pts) Why is there a difference between Third Normal Form and Boyce-Codd Normal Form?

   Answer:
   Because the requirements of BCNF are at odds with dependency preservation in certain cases where there exists an FD from a non-key attribute back to *part* of a compound key. 3NF relaxes the definition of normalization to allow a relation to have this type of FD.