Fall, 2005 CIS 550

Database and Information Systems

Midterm

The exam is 80 minutes long. There are 100 points total, plus 16 points extra credit.

Problem 1 [30 points]: Suppose we are maintaining a database of articles published in our newspaper, the Philadelphia Daily Debater. We have the following schema (where keys are underlined):

Article(issueID, articleID, author, title)
Citation(articleID,issueID,citedArticleID,citedIssueID)
WordAppears(wordID,issueID,articleID,position)
WordIs(wordID,wordText)
Issue(issueID,date,howManyDistributed)

For each of the following queries, write the query 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. Assume that dates can be compared using comparison operators ($<, >, =$). Assume that *position* is an index specifying where the word appears (1 = first word, 2 = second, etc.).

1. Find the documents in which the words "politician" and "corruption" appear. [10pts]

   i.

```
SELECT DISTINCT wa1.issueID, wa1.articleID,
FROM WordAppears wa1, WordIs wi1,
     WordAppears wa2, WordIs wi2
WHERE wa1.issueID = wa2.issueID AND wa1.articleID = wa2.articleID
      AND wa1.wordID = wi1.wordID AND wa2.wordID = wi2.wordID
      AND wa1.wordText = 'politician' AND wa2.wordText = 'corruption'
```

   ii. $\pi_{issueID,articleID}(\sigma_{wordText='politician'}(WordAppears \bowtie WordIs)) \bowtie$
   $\pi_{issueID,articleID}(\sigma_{wordText='corruption'}(WordAppears \bowtie WordIs))$

   OR

   $\pi_{issueID,articleID}(\sigma_{wordText='politician'}(WordAppears \bowtie WordIs)) \cap$
   $\pi_{issueID,articleID}(\sigma_{wordText='corruption'}(WordAppears \bowtie WordIs))$

iii. $\{X | \exists wa1 \in WordAppears, wi1 \in WordIs, wa2 \in WordAppears,$
$wi2 \in WordIs(wa1.issueID = wa2.issueID \wedge wa1.articleID = wa2.articleID \wedge$
$wa1.wordID = wi1.wordID \wedge$
$wa2.wordID = wi2.wordID \wedge wa1.wordText =' politician' \wedge wa2.wordText =' corruption'$
$\wedge X.issueID = wa1.issueID \wedge X.articleID = wa1.articleID\}$

2. Find the most-cited article(s) in the newspaper's history. [bonus up to 10 pts]

   i.

```
SELECT citedIssueID, citedArticleID
FROM Citation
GROUP BY citedIssueID, citedArticleID
HAVING COUNT(*) >= ALL (
 SELECT COUNT(*)
 FROM Citation
 GROUP BY citedArticleID, citedIssueID
)
```

   (Could also do a MAX in a query *over* the COUNT query.)

3. Find articles with the most common word. [bonus up to 10 pts]

   i.

```
SELECT issueID,articleID
FROM Article
WHERE wordID IN (
  SELECT COUNT(wordID)
  FROM WordAppears
  GROUP BY wordID
  HAVING COUNT(wordID) >= ALL(
   SELECT COUNT(wordID)
   FROM WordAppears
   GROUP BY wordID
  )
)
```

   (Again, could do a MAX query over the COUNT query.)

4. Find articles in which “City Hall” are the last two words. [10pts]

   i.

```
SELECT wa1.issueID,wa1.articleID
FROM WordAppears wa1, WordIs wi1,
     WordAppears wa2, WordIs wi2
WHERE wa1.issueID = wa2.issueID AND wa1.articleID = wa2.articleID
      AND wa1.wordID = wi1.wordID AND wa2.wordID = wi2.wordID
      AND wi1.wordText = 'City' AND wi2.wordText = 'Hall'
      AND wa1.position = wa2.position - 1 AND NOT EXISTS (
SELECT position
        FROM WordAppears wa
        WHERE wa.issueID = wa1.issueID AND wa.articleID = wa1.articleID AND
              wa.position > wa2.position
      )
```

   ii. $\pi_{i1,i2}((\rho_{issueID\rightarrow i1,articleID\rightarrow a1,position\rightarrow p1}(\sigma_{wordText='City'}(WordAppears \bowtie WordIs))) \bowtie_{i1=i2\wedge a1=a2\wedge p1=p2-1} (\pi_{i2,a2,p2}(\rho_{issueID\rightarrow i2,articleID\rightarrow a2,position\rightarrow p2}(\sigma_{wordText='Hall'}(WordAppears \bowtie WordIs)) \bowtie_{issueID=i2\wedge articleID=a2\wedge p2>=position} WordAppears)))$

   iii. $\{X|\exists wa1 \in WordAppears, wi1 \in WordIs, wa2 \in WordAppears, wi2 \in WordIs(wa1.issueID = wa2.issueID \wedge wa1.articleID = wa2.articleID \wedge wa1.wordID = wi1.wordID \wedge wa2.wordID = wi2.wordID \wedge wa1.position = (wa2.position - 1) \wedge wa1.wordText =' City' \wedge wa2.wordText =' Hall' \wedge \nexists wa \in WordAppears(wa.issueID = wa1.issueID \wedge wa.articleID \wedge wa.position > wa2.position) \wedge X.issueID = wa1.issueID \wedge X.articleID = wa1.articleID\}$

5. Find the number of citations per author for “senior” authors (i.e., an author who has at least one article that was published 10 or more years ago). [10pts]

```
SELECT a.author, COUNT(*)
FROM Article a, Citation c
WHERE a.issueID = c.issueID AND a.articleID = c.articleID
      AND EXISTS (
         SELECT *
         FROM Article a2, Issue i
         WHERE a2.issueID = i.issueID AND a.author = a2.author
               AND i.date <= today() - years(10)
         )
GROUP BY a.author
```

Problem 2 [25 points]: Suppose we are commissioned to design a schema for a new map and direction-finding site, Mondo Maps. Like MapQuest and Google Maps, our site needs to display route and map information. Underneath it lies a database of cities, states, roads, and landmarks, in a two-dimensional plane (with longitude and latitude specifying a coordinate).

- *States* have abbreviations (unique) and names, and each state has a unique *boundary*.
- *Cities* are unique within states and have names, and each city has a single *boundary*.
- A *boundary* corresponds to a city or state, and it has an ID and a polygon. (Assume there is a special polygon data type.)
- *Roads* have IDs and names, and are made up of multiple *segments*. Assume that a road is associated with a single city.
- A road *segment* has a start and an end coordinate, as well as a directionality (one-way or two-way).
- A *landmark* has a single coordinate, a name, and a type. Assume landmarks are associated with cities, and that landmark names are globally unique.

1. Draw an ER diagram for this domain. Include participation constraints. You may need to add relationship sets that weren't explicitly specified.

2. Write down the 3NF-normalized schema for this ER diagram, using the simplified schema notation of the form $R(A, B, C)$. Underline keys.

Problem 3 [25 points]: Given a relation **T**(A,B,C,D,E,F) and a set F of functional dependencies, $F = \{A \rightarrow BC, AC \rightarrow DE, D \rightarrow F, E \rightarrow AB\}$:

1. What are the candidate keys? [8 pts] *A,E*
2. What attributes are *not* in the attribute closure of A? [7 pts] (Hint: think before you write!) *All attributes are in the closure, since A is key.*
3. Find a minimal cover for attribute set ABC. [5 pts] $A \rightarrow B$, $A \rightarrow C$*; we also accepted* $A \rightarrow BC$*, although it isn't really in completed form.*
4. Is T in 3NF? BCNF? 1NF (i.e., neither 3NF nor BCNF)? [5 pts] *1NF*

Problem 4 [20 points]: Briefly answer the following questions:

1. What would selection and projection mean in a *tree*-structured (hierarchical) data model, as we saw in the case of XML? [5 pts]

   Selection: return all subtrees whose nodes satisfy some condition (e.g., have a value. Projection: return subtrees with certain labels, or perhaps paths.

2. Why might we *not* want to normalize a schema? [10 pts]

   Possible reasons include: (1) performance, where we may not want to use joins, (2) presentation: for reports, sometimes non-normalized data is preferable.

3. What is the difference between an integrity constraint and a property of a database instance? [5 pts]

   An integrity constraint is a property that must hold over all *possible instances of a database.*