Student Exam ID #_____

Spring 2012

CISE Ph.D. Qualifying Examination

Databases

Instructions:

1. You must have a picture ID to receive an exam.
2. **Answer 4 questions.**
3. All of your answers must be on the ***answer* sheets** provided. Clearly indicate which questions you are answering on the answer sheets.
4. The exam is exactly 2 hours long.
5. Keep all your work and answers within the lines as your answers will be copied.
6. Use a pen so the copies will be legible.
7. Raise your hand if you have a question. ***Do not get out of your seat*** for any reason unless you have permission.
8. At the end of 2 hours, the proctor will announce that the exam is over. At that time, ***stop writing***, place all of your papers and the exam in the envelope, seal the envelope, and sign the envelope across the edge of the seal. Wait for the proctor to come and pick up your exam. You must show the signed envelope and your picture ID to the proctor when you turn it in. Do not leave until everyone has turned in their exam.
9. You must give your exam to the proctor immediately when you are asked. Otherwise, it will not be accepted.
10. You may not leave the exam room until the end of the exam unless it is a medical emergency or you need to use the restroom.
11. Proctors will not answer technical questions. Please do the best you can with the information provided on the question sheet.
12. No calculators or other electronic devices are permitted.

Good luck!

Question 1 (Functional Dependencies and Normal Forms) [25 points]

(a) [18 points]

The attributes listed below represent data about a movie copy at a video rental store. Each movie is identified by a movie number and has a title and information about the director and the studio that produced the movie. Each movie has one or several characters, and there is exactly one actor playing the role of each of the characters (but one actor can play multiple roles in each of the movies). A video store has multiple copies of the same movie, and the store differentiates copies with a movie copy number, which is unique within a single movie but not unique between different movies. Each movie copy has a rental status and return date; in addition, each copy has a type (VHS, DVD, or Blu-ray). The rental price depends on the movie and the copy type, but the price is the same for all copies of the same type. The attributes are as follows:

> MovieNbr, Title, DirectorID, DirectorName, StudioID, StudioName, StudioLocation, StudioCEO, Character, ActorID, ActorName, MovieCopyNbr, MovieCopyType, MovieRentalPrice, CopyRentalStatus, CopyReturnDate

A sample data set regarding a movie would be as follows (the data in the curly brackets are character/actor data, in this case for four different characters):

> 567, "It's a Wonderful Life", 25, "Frank Capra", 234, "Liberty Films", "Hollywood, CA", "Orson Wells", {"George Bailey", 245, "James Stewart" | "Mary Bailey", 236, "Donna Reed" | "Clarence Oddbody", 765, "Henry Travers" | "Henry F. Potter", 325, "Lionel Barrymore" }, 5434, "DVD", 2.95, "Rented", "12/15/2010"

1. [7 points] Identify the functional dependencies between the attributes.
2. [4 points] Give the reasons why this set of data items is not in 3NF, and tell what normal form it is in.
3. [7 points] By using the 3NF synthesis algorithm, present the attributes organized into 3NF tables, which should be named appropriately. Underline primary key attributes. Describe the individual steps of this algorithm and the effect of their application to the scenario here.

(b) [7 points]

The pseudo-transitive rule is one of Armstrong axioms that can be derived from three other, fundamental Armstrong axioms. List the pseudo-transitive rule and prove its correctness. In each step, mention the fundamental Armstrong axiom that you apply.

Question 2 (Relational Algebra and SQL) [25 points]

(a) [7 points] Let $R(A_1, \ldots, A_n)$ be a relation schema. Let further $\{A_1\}$ be the primary key of R and $B \in \{A_2, \ldots, A_n\}$, $n \geq 2$ be an attribute of a numerical data type. The (standard) Relational Algebra does not provide aggregate functions like *min*, *max*, *sum*, *avg*, and *count*. Design a strategy in terms of a Relational Algebra expression that enables the user to query for the *complete* tuples in R with the *maximum* B value *without* the use of aggregate functions. Explain your strategy in sufficient detail.

(b) [18 points] Consider the following, self-explanatory database schema about employees:

employee(employee_name, street, city)
works(employee_name, company_name, salary)
company(company_name, city)
manages(employee_name, manager_name)

Primary key attributes are underlined. Foreign key attributes have the same name as the primary keys they reference. Express the following queries in SQL:

1. [3 points] Find the lowest salary of all maximum salaries payed in each company.
2. [3 points] Companies may be located in several cities. Find the names of all companies located in every city in which Small Bank Corporation is located.
3. [3 points] Find the names of the companies that have the most employees.
4. [3 points] Give all managers of First Bank Corporation a 10-percent raise unless the salary becomes greater than \$100,000; in such cases, give only a 3-percent raise.
5. [3 points] Find the names of all employees who earn more than the average salary of all employees of their company.
6. [3 points] Find the names of companies that have the smallest payroll.

Question 3 (Storage) [25 points]

Modern disk drives store more sectors on the outer tracks than the inner tracks. Since the rotation speed is constant, the sequential data transfer rate is also higher on the outer tracks. The seek time and rotational delay are unchanged. Considering this information, explain good strategies for placing files with the following kinds of access patterns:

(a) [5 points] Frequent, random accesses to a small file (e.g., catalog relations). Explain why.

(b) [5 points] Sequential scans of a large file (e.g., selection from a relation with no index). Explain why.

(c) [5 points] Random accesses to a large file via an index (e.g., selection from a relation via the index). Explain why.

(d) [5 points] Sequential scans of a small file. Explain why.

(e) [5 points] Give three reason why disk-based relational database systems have always preferred to use fixed-length tuples.

Question 4 (Indexing) [25 points]

Consider a relational R(a,b,c,d) containing $1,000,000$ records, where each page of the relation holds 10 records. R is organized as a heap file with dense secondary indexes, an the records in R are randomly ordered. Assume that attribute a is a candidate key for R, with values lying in the range 0 to $999,999$. For each of the following queries, name the approach that would most likely require the fewest I/Os for processing the query. The approaches to consider are:

- Scanning through the whole heap file for R.
- Using a B+ tree index on attribute R.a.
- Using a hash index on attribute R.a.

The queries are:

(a) [5 points] Find all R tuples.

(b) [5 points] Find all R tuples such that $a < 50$.

(c) [5 points] Find all R tuples such that $a = 50$.

(d) [5 points] Find all R tuples such that $a > 50$ and $a < 100$.

(e) [5 points] How would your answers to (a) (b) (c) (d) change if attribute a is not a candidate key for R?

Question 5 (Query Execution) [25 points]

The new trend in database design is *column store*, i.e. instead of storing relations as a set of tuples packed into disk pages, the columns/attributes of a relation are stored separately. This allows savings when it comes to I/O access since only the columns required by the computation need to be brought from the disk. Consider two relations form the TPC-H schema:

```
CREATE TABLE lineitem {
  l_orderkey IDENTIFIER,
  ...
  l_extendedprice REAL,
  l_discount REAL
  ...
}
```

```
CREATE TABLE orders {
  o_orderkey IDENTIFIER,
  ...
  o_orderdate DATE,
  ...
  ...
}
```

Assume you have an storage layout that is essentially an array, storing all the values in a column as elements of this array. For example, `l_orderkey` is an array of `IDENTIFIER` values. Assume the sizes in bytes of each datatype is `|DENTIFIER|=8`, `|REAL|=|DATE|=4`. The size of a tuple in a *raw store* layout is `|lineitem|=150` and `|orders|=80`. Assume `lineitem` has 6,000,000,000 tuples and that `orders` has 1,500,000,000 tuples. The I/O in the system works at 2GB/second. Answer the following questions:

1. [5 points] Given the query `SELECT l_extendedprice*(1.0-l_discount) FROM lineitem;` and assuming that the I/O is the bottleneck, how much time a traditional execution engine (that uses row store) will take.
2. [5 points] Same question as above but for the columns store. How much faster is the column store?
3. For query

   ```
   SELECT l_extendedprice*(1.0-l_discount)
   FROM lineitem, orders
   WHERE o_orderdate < '1992-03/16';
   ```

   The query plan the execution engine will execute is to first scan `orders`, filter on `o_orderdate` then place `orders` tuples in a hash. Once the entire `orders` is hashed, it will start streaming `lineitem` and look up each tuple in the hash. Assume the layout is columns-store. Answer the following questions:

   a. [3 points] How fast should the hash insertion be in order for the I/O to be the bottleneck when `orders` is hashed. Express in million tuples/second.

   b. [7 points] Assume a random memory lookup takes 200ns (this is the cost of one hash lookup). Consider the scan of `lineitem`. If we have 48 processors, can we keep up with the I/O stream?

   c. [5 points] For the raw-store execution engine and the same setup as part 3.b, is the I/O or the number of random lookups/second the bottleneck? Now much slower than the column-store will it be?

Question 6 (Transactions and Concurrency Control) [25 points]

H-Store is a new breed of OLTP database design. At its core, is the idea that all traditional concurrency control mechanisms are slow and that better performance is obtained by executing queries one at a time without congruency control on a single processor without parallelism (H-Store allows parallelism across partitions as a last resort). To make the idea work, H-Store runs only pre-compiled transactions and uses only RAM-disks (runs completely in memory).

For such an engine re-design, explain your own approach to solving the following problems – you need no knowledge of H-Store beyond the above ideas:

1. [10 points] Explain what simplifications to the execution engine can be used. Give a small example of a transaction and how it can be executed.
2. [15 points] The database still needs to be *persistent* and to have the ACID properties. Provide a solution for dealing with this problem in a system like H-Store. You can assume that outages are rare, thus you can afford to recover for a significant amount of time.