Name:____

CSE 544, Winter 2009, Final Examination
11 March 2009

Rules:

- Open books and open notes.
- No laptops or other mobile devices.
- Calculators allowed.
- Please write clearly.
- Relax! You are here to learn.

Question	Max	Grade
1	15	
2	15	
3	10	
4	14	
5	14	
6	12	
7	8	
8	12	
Total	100	

Name:____

1. (**15** points) **Relational Model**

   (a) (**3** points) Briefly explain what is a “record-at-a-time” data manipulation language.
   Solution:
   A “record-at-a-time” language is a procedural language that lets the programmer navigate through individual records in a database to locate data of interest: i.e., a language statement can specify and retrieve individual records from a set. IMS and Codasyl offered “record-at-a-time” languages. In IMS, data was organized in a tree and the programmer navigated through this tree to answer queries. In Codasyl, the programmer kept track of a set of currency indicators that captured the current position in a multi-dimensional hyperspace representing the database. The programmer had to “navigate in this hyperspace” to locate records of interest. Hence, with a “record-at-a-time” language, the programmer constructs and manually optimizes a detailed algorithm for solving his or her query.

   (b) (**3** points) Briefly explain what is a “set-at-a-time” data manipulation language.
   Solution:
   In a “set-at-a-time” language, the programmer manipulates the data through high-level set-based operations: a single statement in the language can specify and retrieve many records. The relational algebra and calculus are examples of “set-at-at-time” data manipulation languages.

   (c) (**4** points) What are the benefits of a “set-at-a-time” over a “record-at-a-time” data manipulation language?
   Solution:
   A “set-at-a-time” data manipulation language faciliates physical data independence. Such a language makes it easier to hide the physical structure of the data on disk and change it under the covers without affecting programs. It also relieves the programmer from the burden of manually optimizing queries. The system performs this optimization.

   (d) (**3** points) What is an integrity constraint? Why are integrity constraints important?
   Solution:
   An integrity constraint is a condition specified on a database schema that restricts the data that can be stored in an instance of the database. The DBMS enforces integrity constraints: it only allows legal database instances (i.e., those that satisfy all constraints) to exist.

   Integrity constraints help prevent the entry of incorrect information. Having the DBMS rather than applications validate all updates ensures that all necessary checks are always performed and avoids duplicating the verification logic in each application.

   (e) (**2** points) Give examples of two types of integrity constraints.
   Solution:
   Any two from the following list are good answers: domain constraint, key constraint, foreign key constraint, and general constraint (table constraint and assertion).

Name:____

2. (**15** points) **Storage Manager**

   (a) (**4** points) Why do DBMSs implement their own buffer managers?
   Solution:
   There are two key reasons why DBMSs implement their own buffer managers.

   - A DBMS needs to control when data is written to disk in order to implement transactions. OS buffering can delay writes, causing problems when crashes occur
   - The OS optimizes buffer management for general workloads while the DBMS understands its workload and can do better. Areas of possible optimizations include the page replacement policy, read-ahead algorithms (physical vs logical), and deciding when to flush the tail of the write-ahead log to disk.

   (b) (**3** points) To store its data, a DBMS can either use OS files or it can talk directly to a disk device. Give one benefit of each approach.
   Solution:
   Benefits of using the raw device include:

   - Using the raw device gives more spatial control to the DBMS. The DBMS can ensure that important queries access data sequentially. This approach can thus lead to the highest performance.
   - There are no OS limits on file sizes (files can occupy multiple disks) or number of open file descriptors.

   Benefits of using an OS file include:

   - Increased portability.
   - Does not require devoting entire disks to the DBMS.

   (c) (**4** points) What is GiST and when is it helpful?
   Solution:
   GiST is an index structure supporting an extensible set of queries *and* data types. It allows new data types to be indexed in a manner supporting queries natural to the types. It is also a single piece of code for many disparate tree-structured indexes (it unifies disparate data structures such as R-trees and B+-trees used for common data).

   GiST is basically a template for tree indexes. When extending a DBMS with a new data type, the programmer needs to add an index to speed-up queries that involve this new data type. With GiST, the programmer only needs to implement a couple of simple methods to achieve this goal.

   (d) (**4** points) Give one limitation of the GiST approach.
   Solution:
   GiST has several limitations including:

   i. There are no guarantees that an index implemented using GiST will be efficient (See Section 5 of the paper).
   ii. The GiST does not include a model of the cost of probing the index. The integration of the GiST with a query optimizer is thus a challenge (See Section 6 of the paper).

Name:____

3. (**10** points) **Operator Algorithms**

   Relation R has 90 pages. Relation S has 80 pages. Explain how a DBMS could efficiently join these two relations given that only 11 pages can fit in main memory at a time. Your explanation should be **detailed**: specify how many pages are allocated in memory and what they are used for; specify what exactly is written to disk and when.

   (a) (**5** points) Present a solution that uses a hash-based algorithm.
   Solution:
   The algorithm proceeds as follows:

   - First, we split R into partitions:
     - Allocate one page for the input buffer.
     - Allocate 10 pages for the output buffers: one page per partition.
     - Read in R one page at the time. Hash into 10 buckets. As the pages of the different buckets fill-up, write them to disk. Once we process all of R, write remaining incomplete pages to disk. At the end of this step, we have 10 partitions of R on disk. Assuming uniform data distribution, each partition comprises 9 pages.
   - Then, we split S into partitions the same way we split R (must use same hash function).
   - For each pair of partitions that match:
     - Allocate one page for input buffer
     - Allocate one page for the output buffer.
     - Read one 9-page partition of R into memory. Create a hash table for it using a different hash function than above.
     - Read corresponding S partition into memory one page at the time. Probe the hash table and output matches.

   (b) (**5** points) Present a solution that uses a sort-based algorithm.
   Solution:
   The algorithm proceeds as follows

   - First, we sort R:
     - Read 10 pages worth of R tuples into memory, sort, and write to disk.
     - Repeat for next 10 pages until all R tuples have been processed.
     - Now we have 9 runs of 10 pages sorted on disk.
     - Allocate one page per run and one page for the output.
     - Merge runs in sorted order and output into file.
   - Sort S the same way that we sorted R above.
   - Read S and R one page at the time. Merge them and output matches.

   Note that we can be more efficient if we use a priority queue and output runs of length twice the size of memory in the first step of the algorithm. Then, we can join R and S together while merging their individual runs.

Name:____

4. (**14** points) **Query Optimization**

Consider the following SQL query that finds all applicants who want to major in CSE, live in Seattle, and go to a school ranked better than 10 (i.e., rank < 10).

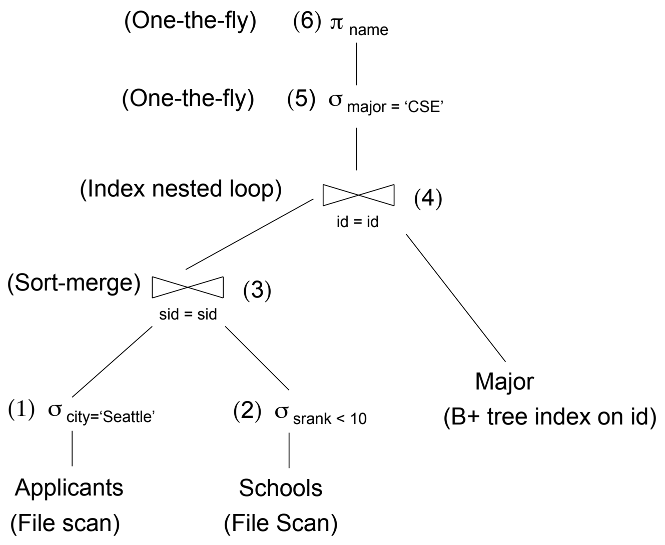
Relation	Cardinality	Number of pages	Primary key
`Applicants (id, name, city, sid)`	2,000	100	`id`
`Schools (sid, sname, srank)`	100	10	`sid`
`Major (id, major)`	3,000	200	`(id,major)`

```
SELECT A.name
FROM   Applicants A, Schools S, Major M
WHERE  A.sid = S.sid AND A.id = M.id
AND    A.city = 'Seattle' AND S.rank < 10 AND M.major = 'CSE'
```

And assuming:

- Each school has a *unique* rank number (`srank` value) between 1 and 100.
- There are 20 different cities.
- `Applicants.sid` is a foreign key that references `Schools.sid`.
- `Major.id` is a foreign key that references `Applicants.id`.
- There is an unclustered, secondary B+ tree index on `Major.id` and all index pages are in memory.

(a) (**10** points) What is the cost of the query plan below? Count only the number of page I/Os.



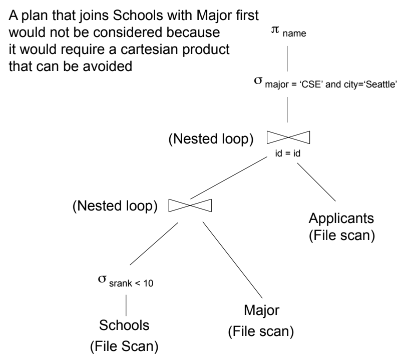
Name:____

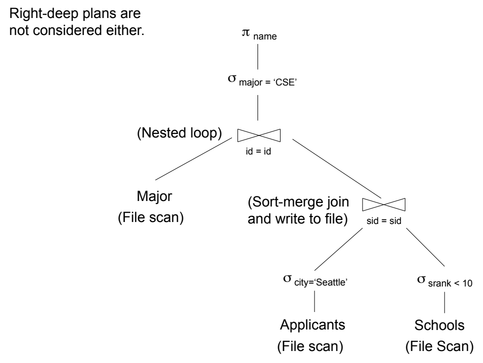
Solution:
The total cost of this query plan is 119 I/Os computed as follows:

- (1) The cost of scanning Applicants is 100 I/Os. The output of the selection operator is $\frac{100}{20} = 5$ pages or $\frac{2000}{20} = 100$ tuples.
- (2) The cost of scanning Schools is 10 I/Os. The selectivity of the predicate on rank is $\frac{10-1}{100} = 0.09$. The output is thus $0.09 * 10 \approx 1$ page or $0.09 * 100 \approx 9$ tuples.
- (3) Given that the input to this operator is only six pages, we can do an in-memory sort-merge join. The cardinality of the output will be 9 tuples. There are two ways to compute this: (a) $\frac{100*9}{(\max(100,9))} = 9$ (see book Section 15.2.1 on page 484) or (b) consider that this is a key-foreign key join and each applicant can match with at most one school but keep in mind that the predicates on city and rank were independent, hence only 0.9 of the applicants end-up with a matching school.
- (4) The index-nested loop join must perform one look-up for each input tuple in the outer relation. We assume that each student only declares a handful of majors, so all the matches fit in one page. The cost of this operator is thus 9 I/Os.
- (5) and (6) are done on-the-fly, so there are no I/Os associated with these operators.

(b) (**4** points) The Selinger optimizer uses a dynamic programming algorithm coupled with a set of heuristics to enumerate query plans and limit its search space. Draw two query plans for the above query that the Selinger optimizer (as described in the paper) would NOT consider. For each query plan, indicate why it would not be considered.

Solution:
Many solutions were possible including:


A plan that joins Schools with Major first would not be considered because it would require a cartesian product that can be avoided
π name
σ major = 'CSE' and city='Seattle'
(Nested loop)
id = id
(Nested loop)
Applicants (File scan)
σ srank < 10
Schools (File Scan)
Major (File scan)



Name:____

5. (**14** points) **Transactions**

   For each statement below, indicate if it is true or false.

   (a) (**2** points) Serializability is the property that a (possibly interleaved) execution of a group of transactions has the same effect on the database and produces the same output as some serial execution of those transactions.

   TRUE FALSE

   (b) (**2** points) The following schedule is serializable:
   $r_0[A] \rightarrow w_0[A] \rightarrow r_1[B] \rightarrow w_1[B] \rightarrow r_1[A] \rightarrow w_1[A] \rightarrow r_0[C] \rightarrow w_0[C] \rightarrow c_0 \rightarrow c_1$

   TRUE FALSE

   (c) (**2** points) A NO-STEAL buffer manager policy means that all pages modified by a transaction are forced to disk before the transaction commits.

   TRUE FALSE

   (d) (**2** points) Strict two-phase locking (2PL) ensures that transactions never deadlock.

   TRUE FALSE

   (e) (**2** points) Strict two-phase locking (2PL) ensures serializability.
   Note: This question was confusing so we did not count it. To get serializability, one needs both strict 2PL and predicate locking. Strict 2PL alone guarantees only conflict serializability. Please see book and paper for more details.

   TRUE FALSE

   (f) (**2** points) In the ARIES protocol, at the end of the analysis phase, the Dirty Page Table contains the exact list of all pages dirty at the moment of the crash.

   TRUE FALSE

   (g) (**2** points) The ARIES protocol uses the "repeating history" paradigm, which means that updates for all transactions (committed or otherwise) are redone during the REDO phase.

   TRUE FALSE

   Solution:
   T, T, F, F, F, F, T

Name:____

6. (**12** points) **Parallel Data Processing**

   (a) (**4** points) In a parallel DBMS, why is it difficult to achieve linear speedup and linear scaleup?
   Solution:
   There are three key reasons why linear speedup and linear scaleup are difficult to achieve:

   i. Startup cost: The latency involved in starting an operation on many nodes may dominate the computation time.
   ii. Interference: Each new process competes for shared resources with the other processes (e.g., network bandwidth). This resource contention can limit the performance gains of adding more processes.
   iii. Skew: The time to complete a job is the time that the slowest partition takes to complete its job. When the variance dominates the mean, increased parallelism improves elapsed time only slightly.

   (b) (**4** points) List two features common to a traditional DBMS and MapReduce.
   Clarification: Here, "traditional DBMS" means a traditional *parallel DBMS*.
   Solution:
   Many answers are possible including:

   i. Horizontal data and operator partitioning.
   ii. Distribution independence: applications need not know that the data is distributed.

   (c) (**4** points) List two features that are different between the two types of systems: i.e., features that are present in one system but not in the other. For example, you can give one feature present in MapReduce but absent in a parallel DBMS and one feature present in a parallel DBMS but missing from MapReduce (or any other combination).
   Solution:
   Again, different answers were possible including:

   i. A DBMS offers updates, transactions, indexing, and pipelined parallelism.
   ii. MapReduce has intra-query fault-tolerance and handles stragglers better.

Name:____

7. (**8** points) **Database as a Service**

   Frank and Betty own a small Internet based company that sells collector pens over the web. Recently, they decided to get rid of all their infrastructure and move to using Amazon Web Services. As part of this transformation, they plan to get rid of their DBMS and run their application on top of Amazon SimpleDB.

   (a) (**4** points) List three potential benefits of this move.
   Solution:
   Many answers were possible for this question including:

   i. Frank and Betty will no longer need to administer their DBMSs themselves.
   ii. Frank and Betty will not have to worry about equipment upgrades or purchases. SimpleDB will provide them elastic scalability and a "pay-as-you-go" pricing model.
   iii. Amazon will worry about operations problems such maintaining high-availability.

   (b) (**4** points) List three potential challenges that they will face.
   Solution:
   Again, many answers were possible include:

   i. SimpleDB does not have all the features of a DBMS: there are no transactions, the query model is limited, etc.
   ii. Frank and Betty will need to rewrite their applications to use SimpleDB.
   iii. They may worry about data privacy now that the data will be stored on Amazon's servers.

Name:____

8. (**12** points) **Column-Stores**

   (a) (**4** points) Explain why column-oriented DBMSs are advantageous for OLAP workloads compared with row-oriented DBMSs?
   Solution:
   Column-oriented DBMSs outperform their row-oriented counterparts on analytical workloads for three main reasons:

   i. First, column-stores only read from disk and process in memory those attributes access by a query. In OLAP workloads, facts and dimensions tables often have many attributes. When queries touch only a small subset of these columns, ignoring unnecessary ones significantly improves performance.
   ii. Second, once data is organized into columns, it is easier to compress, which also leads to significant performance gains on read-only workloads.
   iii. Third, column-based data organization enables important optimizations inside the query execution engine including the late materialization of tuples, block iteration, and invisible joins.

   (b) (**4** points) Describe one approach to simulate a column-oriented DBMS in a row-oriented DBMS.
   Solution:
   Several approaches are possible to simulate a column-oriented DBMS in a row-oriented one including the following:

   i. Vertically partition all tables in the system into a collection of two-column tables consisting of (table key, attribute) pairs. Rewrite queries to explicitly join these tables when retrieving multiple columns from the same relation.
   ii. Leave the base data unchanged but add unclustered B+Tree indexes on every column of every table and use index-only plans to answer queries.
   iii. Create a set of materialized views, where each view includes only those columns needed to answer a query.

   (c) (**4** points) Explain why such simulation yields worse performance compared with using a column-store directly?
   Solution:
   The above techniques to emulate a column-store DBMS in a row-store one have the following limitations:

   i. Vertical partitioning suffers from three problems. First, tuple headers required in a row-store DBMS add a significant overhead. Second, vertical partitioning causes the system to store redundant copies of the table key attribute, also adding overhead. Finally, this approach necessitates expensive joins during execution to combine the columns touched by a query back together.
   ii. The high-performance of index-only plans depends on the order in which tables are joined back together and, as the paper discussed, a row-oriented DBMS may not always be able to perform these joins in the most efficient order.
   iii. Using materialized views works well to avoid reading unnecessary columns. However, the column-store DBMS can perform additional important optimizations including compression, late materialization, block-iteration, and invisible joins that are not available in a row-oriented DBMS.