CS 632: Advanced DBMS End-semester Exam, Spring 2003

Duration: 3 hours Total Marks: 100

NOTE: Answer all parts of a question together, and start each answer on a new page

1. Short answers

   (a) (Freebie) What were your most favourite and least favourite (a) topics and (b) papers. ...2

   (b) Explain the key differences between the Truman World Model (or equivalently, Oracle's Virtual Private Database model) and the Non-Truman World Model. ...2

   (c) Explain how a hierarchy of trust is established with digital certificates. Also explain how to handle the case when a private key (of either an end user or a certification authority) is compromised. ...2 + 3

   (d) A web-based information system requires caching at different levels. List all levels of caching that you can think of (from the lowest level in the hardware/DB till the highest level at the user end). ...3

   (e) The OLAP dimension constraints paper studies the problem of heterogeneous dimensions coupled with dimension constraints, and from the constraints it deduces "frozen dimensions".

      i. Explain what is a heterogeneous dimension hierarchy. ...2

      ii. Explain how frozen dimensions can be used to create a set of homogeneous dimension hierarchies. ...3

   (f) Most XML implementations on top of relational databases have used structing within the database engine coupled with late tagging outside the database engine. What would considerations could have motivated this decision? ...2

2. Suppose you wish to implement a B^+-tree index in a shared-nothing parallel database system. Consider two ways in which the index can be implemented: (a) where the data on each processor has its own index, and (b) the index is parallelized independent of the data, with the top few nodes of the index replicated at all nodes and the lower subtrees partitioned.

   (a) Explain how index lookups would be done in each case (don't worry about updates). ...3

   (b) Compare the two alternatives in terms of point lookups with a single result and range lookups with a large result set. ...4

3. MQO and Pipelining

   (a) An exhaustive algorithm for MQO can be obtained by computing the benefit of materializing each subset of equivalence nodes, rather than choose one greedily. What would be the cost of exhaustive MQO using this approach, assuming that the DAG has 2^n nodes, and assuming that each benefit computation has cost linear in the size of the DAG. ...2

   (b) Is it really necessary to examine every subset of nodes? Explain what subsets need to be examined, to reduce the cost of the above step greatly. ...3

   (c) The initial description of the shared read optimization creates a single extra node. Why is this not general enough? ...2

   (d) Explain how the greedy integrated algorithm is able to generate more general plans in the above case. ...3

4. Consider the following DTD. Draw the DTD graph and give relational mappings of the DTD under the basic and shared inlining technique. ...2 + 2 + 2

```
<ELEMENT professor(name, title, publications)>
<ATTLIST professor id IDREF IMPLIED)>
<ELEMENT name (PCDATA)>
<ELEMENT title (PCDATA)>
<ELEMENT publications( publication*)>
<ELEMENT publication (article | inproceedings)>
<ELEMENT article(title, author+, journal)>
<ELEMENT inproceedings(title, author+, conference)>
<ELEMENT journal (PCDATA)>
<ELEMENT conference (PCDATA)>
<ATTLIST author (authID IDREF REQUIRED)>
```

5. Databases can cache tuples in one of two ways: (a) at a physical level (all tuples that live in a particular page) or (b) at a logical level (all tuples that satisfy some condition).

   (a) Explain why the second form is preferable in a query result cache, where the original data comes from a backend database. ...2

   (b) Assume all queries are on a single relation, and of the form $r.A = value$. Suggest how to manage the cache in this case, first assuming only one tuple would satisfy the condition. ...2

   (c) Now extend your answer to the case where many tuples may satisfy the condition. In particular, address the issue of what data to evict when the cache is out of space. ...4

   (d) Now if there are queries of the forms $r.A = value$, as well as of the form $r.B = value$. The above schemes can still be used, but what would one potential drawback be? ...2

6. (a) The "Automated Ranking ..." paper talks about an index based top-k implementation. Explain the stopping condition, using an example (with actual query and relation instances). ...6

   (b) Equation 4 in Section 3.3 of the paper uses max, and mentions a problem with using avg. Give an example where avg would lead to unintuitive results. ...3

7. (Mediators)

   (a) Consider a mediator that has to perform a dependent join for $E_1 \bowtie_{E_1.A=E_2.A} E_2$, where access to E_2 requires the A value to be specified.

   Explain how to estimate the cost of a single access to E_2 in the dependent join, as well as the cost of the overall dependent join. (You can make use of standard cost estimation techniques as subroutines). ...2+ 2

   (b) Now suppose E_2 is stored (or can be computed) in the order of ascending A values (but there is no index on A).

      i. Explain how to exploit this feature to reduce the cost of the overall dependent join ...4

      ii. What would be wrong with the cost estimate described earlier in this case? ...1

      iii. Suggest a possible way of extending the cost estimation technique to handle this algorithm. ...4

8. A system desires to achieve serializability and failure atomicity. Still, it allows a transaction to the read the dirty data of another transaction. Answer the following questions assuming that a $force, no-steal$ protocol is used for transaction execution. (Give a short explanation for each answer.)

   (a) t_1 writes to x and t_2 reads x before t_1 commits. If t_1 were to abort, what should happen to t_2? ...2

   (b) t_1 reads x and t_2 writes to x before t_1 commits. If t_1 aborts, what should happen to t_2? ...2

   (c) t_1 writes to x and t_2 writes to x before t_1 commits. What should the system do if t_2 wants to commit and t_1 has not yet committed? ...2

9. In the *nested transaction model*, a transaction T consists of a set S of subtransactions. A subtransaction s_1 in S can abort without causing the abortion of T – in this case, T might retry s_1 or try an alternative subtransaction. When a subtransaction s_1 commits, its changes are not made permanent, instead, T takes over those changes. T can abort even if s_1 commits.

   (a) The commit of a subtransaction is usually called *conditional* commitment. Why? ...2

   (b) Suppose concurrency control is done using locks. Indicate what the locking protocol would do to effect the above semantics of subtransaction commitment. ...4

   (c) Give an example of an application where it is preferable to code it as a nested transaction as opposed to a traditional atomic transaction. ...2

   (d) Give one disadvantage of transaction models beyond atomic transactions. ...1

10. Consider a database that includes *atomic* `Increment` and `Decrement` operations in addition to `Read` and `Write`.

    `Increment` (x, val) : add val to data item x.

    `Decrement` (x, val) : subtract val from data item x.

    Neither operation returns any value. The lock compatibility matrix is as given below.

	`Read`	`Write`	`Increment`	`Decrement`
`Read`	True	False	False	False
`Write`	False	False	False	False
`Increment`	False	False	True	True
`Decrement`	False	False	True	True

(a) Give a real life example where use of `Increment`/`Decrement` operation instead of `Write` operation may greatly increase concurrency. ...1

(b) If all the transactions lock data items adhering to above matrix, does 2PL ensure serializability? Explain your answer. ...2

(c) Explain how the `Increment` and `Decrement` operations should be implemented. ...2

11. What is the basic criteria used for summarizing changes in the Diff operator? What are the difficulties in directly implementing this criteria. What are the solutions proposed in the paper for handling these. ...6

Total Marks = 100