Final Exam: Database Systems

This exam has three sections, each of equal importance. **Be sure to read the instructions at the top of each section before beginning.** The exam is open book – you may consult any paper references. You may not consult with other people or with any electronic resources. You have 90 minutes to work on the exam.

**SECTION 1**

Answer all of the questions in this section.

1. **Sorting**
   You need to sort a relation of $B$ blocks. What is the minimum number of in-memory buffers required to sort the relation in two passes? Prove the correctness of your answer.

2. **Buffer Management**
   Give examples of relational query processing scenarios that result in the following reference patterns:
   a) Straight sequential
   b) Looping sequential
   c) Hierarchical with clustered sequential
   d) Independent random

3. **Recovery**
   Explain the distinctions between
   (a) ATOMIC and ¬ATOMIC,
   (b) STEAL and ¬STEAL,
   (c) FORCE and ¬FORCE,
   (d) What choices does ARIES make for these variables, and why?
SECTION 2
You may choose any one of the questions in this section to answer.

4. System R Query Optimization
   a) For a query on one table, how many access plans does the System R optimizer consider? (Hint: Your answer should involve the number of indices on the table.)
   b) Disregarding "interesting orders", what is the maximum number of partial solutions stored when optimizing a query of n relations? Explain how you derive your answer, which should be a function of n.
   c) Explain the difference, both in terms of language and processing, between a standard subquery and a "correlation" subquery.
   d) State 3 assumptions made in the System R optimizer that may not be true in modern DBMSs.

5. Access Methods
   The UCB EECS grades over all time are going to be stored on a CD-ROM for widespread distribution. The grades form one big table, with 10 columns, 6 of which (IDno, name, age, gpa, home-latitude, home-longitude), are commonly referenced in the WHERE clause of queries. You have to choose access methods to store on the CD-ROM for speeding up queries. You are space-constrained, and can choose to store only a few indices. Assume that a B+-tree takes up B blocks on the CD-ROM, whereas a d-dimensional R-tree takes up dB blocks. You have only 4B blocks of room for indices.
   a) Assume you choose to build an R-tree on the pair of columns (age, gpa). Describe how you would build this R-tree, what queries it would support well, and why.
   b) Suppose you choose instead to build a B+-tree on age, and another B+-tree on gpa. What queries will be more efficient than the R-tree solution, what queries will be less efficient, and why.
   c) Suggest a good choice of indexes for the CD-ROM, and the way you would build these indexes. You may choose from single-key B+-trees, and R-trees of any number of dimensions, but may only use 4B blocks of storage. Describe queries that are efficiently answered by your indexes.

6. Nested Top Actions in ARIES
   Explain what Nested Top Actions are, and why they are easy to implement in the context of ARIES. Describe what happens when a transaction containing a Nested Top Action aborts.
SECTION 3
You may choose any one of the questions in this section to answer.

7. Concurrency, Recovery, and Buffer Management
It was asserted in class that the triple {concurrency, recovery, buffer management} had to be implemented in a tightly-coupled manner. In this question, consider what would happen if any proper subset of these modules was implemented by the OS, and the rest were implemented in the DBMS. State the subset you chose to consider putting in the OS, and the implications of the division in terms of performance and transactional semantics.

8. Query optimization in object-relational database systems.
Object-relational database systems such as Illustra and UniSQL as well as object-oriented database systems such as O2 and Orion support the notion of set-valued attributes and nested queries. For example, the definition of a Person type might contain the attribute "children: set of person".
   a) Describe two alternative ways of implementing set-valued attributes, one of which requires variable length records even if each element of a set is of fixed-length.
   b) Discuss the impact of each strategy on the task of optimizing queries involving set-valued attributes.
   c) What kinds of statistics should an optimizer maintain for set-valued attributes? What problems do you foresee in maintaining such statistics? What solutions would you propose?
   d) What impact do set-valued attributes have on the effectiveness of parallel query processing in a system like Gamma?

9. Bitmap Indexes
Consider the following two implementations of the leaf level of a B+-tree:
   I. For each value \( V_i \) in the indexed column, there is a linked list of tuple identifiers (tids), pointing to the tuples in the relation having \( V_i \) in their corresponding column (traditional index).
   II. For each value \( V_i \), there is a bitmap with as many bits as there are tuples in the relation; the \( i \)th bit is 1 if the \( i \)th tuple in the relation has \( V_i \) in its corresponding column (bitmap index).
   a) Indicate when representation (I) is to be preferred over the more traditional representation (II), and what its benefit is in those cases.
   b) Describe an effective mapping between integers (bit positions) in the bitmap and the tuples (tids) indexed in the relation when the relation can hold (i) fixed-length tuples only, and (ii) variable-length tuples.
   c) Consider relation \( R(X, Y, Z) \) whose attributes \( X \) and \( Y \) are indexed using bitmapped indices. Describe an effective way to process the following query, where \( C_x \) and \( C_y \) are constants:
      \[
      \text{select } Z \text{ from } R \text{ where } X = C_x \text{ and } Y = C_y
      \]
   d) Repeat the previous question for the following query, where \( C_x \) and \( C_y \) are constants:
      \[
      \text{select } Z \text{ from } R \text{ where } X = C_x \text{ or } X = C_y
      \]