University of California, Berkeley

CS 186 Introduction to Databases, Spring 2014, Prof. Dan Olteanu

FINAL EXAM

- This is a closed book examination but you are allowed one 8.5" x 11" sheet of notes (double sided).
- No cell phones/calculators/electronic devices of any sort are allowed.
- Please display your student ID card.
- You should answer as many questions as possible. You should read all of the questions before starting the exam, as some of the questions are substantially more time-consuming than others.
- There are 100 points in total distributed over six questions.
- Write all of your answers in the designated space provided with each question. We will be grading only answers written in the designated space.
- You must put your CS 186 class account at the top of each page before you start the exam.

GOOD LUCK!

Question 1.[Potpourri] Minimum points: 0. Maximum points: 20.

For each of the following statements below, each correct tick is worth 1 point, each wrong tick is worth -1 point, and no tick is worth 0 points.

Tick the correct TRUE/FALSE boxes for each of the following statements.	**TRUE**	**FALSE**
1. An IX lock is compatible with an S lock.		
2. Shared and Exclusive locks are features found in a Pessimistic Concurrency Control scheme.		
3. Strict 2-Phase Locking avoids deadlocks but not cascading aborts.		
4. Page-oriented and block nested-loops joins have a lower IO cost in case the larger relation is the inner relation.		
5. Data entries in R-Trees are themselves regions.		
6. Given B buffer page frames, External Merge Sort can sort a file of size $B(B-1)^5$ in 6 passes.		
7. It is always more IO efficient to push a selection operator through a join and perform the selection first on one of the join relations.		
8. For a given query and cardinality estimates, the space of query plans inspected by the greedy approach is a *strict* subset of the space of query plans inspected by the dynamic programming approach (both approaches were introduced in class).		
9. Suppose a transaction T_1 writes a page A, then A is flushed to disk and T_1 aborts without undoing its updates. This is a violation of the Consistency desideratum.		
10. If a schedule is serializable, then it is also conflict serializable.		

11. Upon recovering in a FORCE/STEAL system, uncommitted transactions at the time of the crash may need to be undone. ☐ ☐

12. Upon recovering in a NO FORCE/NO STEAL system, uncommitted transactions at the time of the crash may need to be undone. ☐ ☐

13. BCNF decompositions are always lossless-join and dependency-preserving, while 3NF decompositions are not necessarily dependency-preserving. ☐ ☐

14. There are cases where aggregations can be performed by index-only plans. ☐ ☐

15. If we do not know the query workload and plenty of disk space is available, then it is always a good idea to create more indexes. ☐ ☐

16. A clustered B+-tree index is preferable over a hash index if the query workload frequently contains range selections. ☐ ☐

17. Conflict serializability is always guaranteed if the conflict graph is acyclic. ☐ ☐

18. A correct implementation of concurrent transaction support has to rely on a form of locking such as read and write locks. ☐ ☐

19. A join algorithm X is worst-case optimal if there is no join algorithm Y that performs asymptotically better than X for any given class of instances. ☐ ☐

20. The Leapfrog Triejoin algorithm is an improvement of the multi-way standard sort-merge algorithm where the relation iterators may jump over (or leapfrog) several records at once. ☐ ☐

Question 2.[Query Languages] Maximum points: 10.

Consider the relational schemas Student(S), Course(C), Enrolls(S, C).

1. Write down an SQL query that returns, for each student present in the database, the student id S and the number of courses s/he is enrolled in. (Hint: this may mean that for some values of S the count is 0!) (5 points)

2. Write down an SQL query that returns all pairs of students (S_1, S_2) such that S_1 has taken (at least) all the courses that S_2 has taken. (5 points)

Question 3.[Query Evaluation] Maximum points: 14.

Consider the following three relations:

X(A, B, C)	Y(A, D)	Z(B, D)
(1, 2, 4)	(1, 5)	(1, 1)
(1, 2, 6)	(1, 7)	(1, 3)
(1, 2, 7)	(2, 1)	(1, 7)
(1, 3, 9)	(2, 3)	(2, 2)
(1, 5, 8)	(2, 5)	(2, 3)
(3, 1, 3)	(3, 1)	(2, 7)
(3, 1, 7)	(3, 3)	(2, 8)
	(3, 8)	(3, 1)
	(3, 9)	(3, 7)

1. We want to evaluate the three-way *natural join* on these relations using the Leapfrog Triejoin algorithm with the join order (A,B,D,C). Draw the trie view of each relation. Write down the first four tuples in the result in the order they are produced. Indicate on your diagram the position of each trie-iterator when each of the first four tuples is emitted. (8 points)

2. Briefly describe an optimization that can be made to the basic block nested-loops join algorithm. Make sure to include when the optimization is useful and describe why the optimized algorithm can be more efficient. (6 points)

Question 4.[Query Optimization] Maximum points: 20.

Consider the three relations R(A,B), S(B,C) and T(C,D), where all attributes are integer attributes and each such integer attribute takes 5 bytes. Thus, each of the input tuples takes 10 bytes, but note that projections and joins produce tuples of different arity and thus different size. You have to take this into account. The page size is 1000 bytes and we assume for simplicity that all of this space can be used to store tuples (the page header does not take space). Assume 22 buffer pages.

- Relation R has 100,000 tuples. Attribute B is the key of R. The domain of attribute A is integer values in the interval [1, 1000]. The domain values of attribute A are uniformly distributed among the tuples in R (i.e., each value is equally likely).
- Relation T has 100,000 tuples. Attribute C is the key of T. The domain of attribute D is integer values in the interval [1, 1000]. The values of attribute D are distributed among the tuples in T according to the following histogram (given in textual form):

  Value of T.D in interval [1 .. 500] : 20000 tuples.

  Value of T.D in interval [501 .. 1000] : 80000 tuples.
- Relation S consists of 1,000,000 tuples. Column B is the key of R referenced by foreign key B of S and column C is the key of T referenced by foreign key C of S.
- There are no index structures.

Consider the following query:

```
SELECT *
FROM R, S, T
WHERE R.B = S.B and S.C = T.C AND R.A <= 10 AND T.D > 100
```

1. Assume that you are using the System R style *dynamic programming query optimization* algorithm. Show in detail the plans that the System R query optimization algorithm produces after the first pass (i.e., "best" one relation plans). Explain why these plans are chosen over other plans. Clearly state any assumptions you make. (3 points)

2. Choose the optimal query plan that the System R algorithm can find (with respect to I/O cost) assuming that the only join operator implemented in the system is block nested-loops join and no index structures are to be used. Assume that the query result does not have to be written to disk but is displayed on screen (which has no cost per se).

   Write down the optimal query plan as an operator tree where the outer loop input of each join is on the left side. Compute, for each node of the query plan, the size of its output in number of tuples and number of pages, plus the cost incurred at that node. In addition, state the overall cost. Write down all the formulas you use. (12 points)

3. Now assume you are allowed to use any join operator implementation (block nested-loops join, sort-merge join, hash join; no index nested-loops join because you must not use indexes). Is there a better query plan that can be obtained by replacing the block nested-loops joins in the previous plan but leaving all else (in particular, the join order) equal? If yes, draw a better (not necessarily optimal) plan and calculate its cost. If no, argue why not. (5 points)

Question 5.[Concurrency control and serializability] Maximum points: 16.

Consider the following transaction schedule, where time increases from left to right:

T_1:		R(A)		W(B)			Commit		
T_2:			R(B)		R(A)			Commit	
T_3:	R(B)					W(A)			Commit

1. Draw the conflict graph for the above schedule. Is the schedule conflict serializable? (4 points)

2. Explain when a schedule can be view serializable but not conflict serializable. Give a small example schedule that has this property. (4 points)

3. Explain in no more than three sentences the main idea of optimistic concurrency control. (4 points)

4. Explain the ACID desiderata for transaction processing. (4 points)

Question 6.[Database Design] Maximum points: 20.

Consider the relation schema $R(A, B, C, D, E, F, G)$ with functional dependencies $F = \{AB \rightarrow CF, CD \rightarrow EA, E \rightarrow ABC, B \rightarrow F, C \rightarrow D\}$.

1. Compute a minimum cover of F. (6 points)

2. Give all the candidate keys of R as inferred from F. (6 points)

3. Is R in BCNF? Is R in 3NF? Explain your answer. (4 points)

4. Consider the decomposition of R into $R_1(A, B, C, D, E, G)$ and $R_2(B, F)$. Is this decomposition (a) dependency-preserving, (b) lossless-join, (c) in 3NF, (d) in BCNF? (4 points)

End of Exam Paper.

Extra Page for Scratch Work (to be discarded and not graded)

Extra Page for Scratch Work (to be discarded and not graded)

Extra Page for Scratch Work (to be discarded and not graded)