UCLA
Computer Science Department
Winter 2014

Instructor: J. Cho

Student Name and ID: _____

# CS143 Final: Closed Book, 90 minutes

**(** IMPORTANT PLEASE READ **):**

- There are 3 problems on 7 pages for a total of 60 points to be completed in 90 minutes. *You should look through the entire exam before getting started, in order to plan your strategy.*
- *Simplicity and clarity of your solutions will count.* You may get as few as 0 point for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.
- If you need to make any assumption to solve a question, please write down your assumptions.
- To get partial credits, you may want to write down how you arrived at your answer step by step.
- You may use one-page double-sided cheat-sheet during exam. You are also allowed to use a calculator.
- Please, write your answers neatly. Attach extra pages as needed. Write your name and ID on the extra pages.

| **Problem** | **Score** | |
|---|---|---|
| 1 | 20 | |
| 2 | 20 | |
| 3 | 20 | |
| Total | 60 | |

## Problem 1 (Index and Join): 20 points

Suppose you have 2 relations, $R(\underline{A}, B)$ and $S(\underline{B}, C)$, with the following characteristics:

- The size of one disk block is 1000 bytes.
- Attributes $A$, $B$ are of length 10 bytes. Attribute $C$ is of length 180 bytes.
- The tuples are not spanned across disk blocks
- $|R| = 5,000$ (number of tuples of $R$)
- $|S| = 500$ (number of tuples of $S$)
- We have 30 blocks of memory buffer
- We use one disk block for one B+tree node
- Each pointer in a B+tree index (both a record pointer and a node pointer) uses 10 bytes.

1. (2 points) What is the minimum number of blocks needed to store $R$ and $S$?

   $R$: _____ $S$: _____

   **ANSWER:**
   `R: 100, S: 100.`

2. (2 points) We want to construct a dense B+tree on attribute $B$ of table $S$ by scanning the $S$ table. What is the maximum possible $n$ for the B+tree given the above parameters?

   $n$: _____

   **ANSWER:**
   $10n + 10(n-1) \leq 1000$. `Therefore,` $n = 50$.

3. Assume the numbers computed in the previous problems. Assume that each node in the B+tree contains the minimum number of keys and pointers (as long as it is allowed in our parameter setting).

   (a) (4 points) How many nodes does the constructed B+tree have?

   **ANSWER:**
   21. For leaf nodes, the minimum number of keys per node is 25. Therefore, we will have $500/25 = 20$ leaf nodes and one root node. Any answer with 500/(half of answer 2)+1 would be considered correct assuming that answer 2 is not too small.

   (b) (4 points) How many disk IOs would be incurred during the construction of the B+tree on `S.B`? Assume that you use the main memory buffer in the most efficient way to minimize the number of disk IOs. *In your answer, please include the cost of reading the tuples from $S$ and writing the constructed B+tree to the disk.*

   **ANSWER:**
   121. For index construction, we need to scan the entire $S$ table once. $S$ is stored in 100 blocks, so 100 disk IOs for reading $S$. We also incur 21 disk IOs to write the B+tree. Any answer with S from answer 1 + answer 3(a) would be considered correct assuming that 3(a) is not too big.

4. (4 points) We want to execute the query "`SELECT R.A,R.B FROM R,S WHERE R.B=S.B`" using the B+tree constructed in the previous problems. Since we do not need to return `S.C`, we decide to use the following algorithm to avoid scanning the $S$ table:

```
For each tuple r in R
  Lookup S.B index with r.B
  If r.B exists in the index, return (R.A, R.B)
```

   Compute the cost of this join. In computing your answer, please use the main memory buffers in the most efficient way to minimize the number of disk IOs. Do not include the cost for writing the final answer. Assume that none of the index nodes and the $R$ blocks are cached in main memory in the beginning.

   **ANSWER:**
   121. We cache all B+tree node in the main memory, and scan the $R$ table once. $R$ is stored in 100 blocks. The B+tree index is 21 blocks. Any answer with answer 1S + 3(a) would be considered correct assuming that 3(a) is not too big.

5. (4 points) How many disk IOs would have been incurred if we used block nested loop join? Is it worthwhile to construct an index on `S.B` to perform the join using the above algorithm considering the index construction overhead?

---

**ANSWER:**
`block nested loop join would have incurred` $100 + 4 \times 100 = 500$ `disk IOs. Even if we take the index construction overhead, the total number of disk IOs was 242. It was definitely worthwhile. Any answer with (1R + 4x1S) or (1R + 100x1S) would be considered correct.`

## Problem 2 (Database Design): 20 points

1. Consider the following table storing temperature readings taken by sensors:

   `Temps(sensorID,time,temp)`

   For example, a tuple (s26,13:42,76) in Temps says that at time 13:42, sensor s26 reported a temperature of 76 degrees. (Don't worry about dates; perhaps our sensors are active for only one day.) Assume for all parts (a)-(c) of this problem that the pair of attributes `(sensorID,time)` is the only key for table `Temps`.

   (a) (2 points) Suppose the functional dependency (FD) "`time`→`temp`" holds on table `Temps`. State in English what real-world property is captured by this FD. (A real-world property is described by a phrase or sentence discussing sensors, times, and/or temperatures, while not mentioning tables, tuples, and/or attributes.) Please be brief; a short phrase or one-sentence answer is sufficient.

   **ANSWER:**
   `At a given time, the temperature readings from all sensors are the same.`

   (b) (2 points) Continuing to assume that FD `time`→`temp` holds for `Temps`, is `Temps` in Boyce-Codd Normal Form? Briefly explain your answer; again, a short phrase or one-sentence answer is sufficient.

   **ANSWER:**
   `No.  The left-hand side does not contain a key.`

   (c) (6 points) Write a SQL query to test whether the FD `time`→`temp` holds on a given instance of Temps. Specifically, your SQL query should return the list of the time values that violate the FD. The query should return an empty answer if the FD does hold.

   **ANSWER:**
   ```
   SELECT time FROM Temps GROUP BY time HAVING COUNT(DISTINCT temp) > 1
   ```

2. (4 points) Suppose we have a relation $R(A, B, C, D, E)$ that satisfies the multivalued dependencies $A \twoheadrightarrow B$ and $B \twoheadrightarrow CD$. When $R$ contains the tuples $(1, 2, 3, 4, 5)$ and $(1, 6, 7, 8, 9)$, write down the list of other tuples that must exist in $R$.

   **ANSWER:**
   $(1, 2, 7, 8, 9)$, $(1, 6, 3, 4, 5)$, $(1, 2, 3, 4, 9)$, $(1, 2, 7, 8, 5)$, $(1, 6, 3, 4, 9)$, $(1, 6, 7, 8, 5)$

3. (6 points) A set of attributes $X$ is said to be closed if $X+ = X$; that is, the closure of $X$ adds nothing to $X$. Relation $R(A, B, C, D, E, F)$ has functional dependencies $A \to B$ and $B \to CD$. In the relation $R$, how many nonempty closed sets of attributes are there? Briefly explain your answer.

   **ANSWER:**
   23. If $X$ contains $A$, it should also contain $B$ and $CD$ to be closed. Depending on whether $E$ and $F$ are in $X$ or now, there are 4 possible $X$'s in this case. If $X$ does not contain $A$, but contains $B$, it should also contain $CD$. Depending on whether $X$ contains $E$ and $F$ or not, there are 4 possible $X$'s. If $X$ does not contain $A$ or $B$, $X$ can be any subset of $\{C, D, E, F\}$. There are 15 possible nonempty $X$'s in this case.

## Problem 3 (Transactions): 20 points

1. Consider the relation `Employee(name, salary)` where `name` is the key. The following three transactions are being executed:

   $T_1$:
   ```
   SELECT SUM(salary) FROM Employee;
   COMMIT;
   ```

   $T_2$:
   ```
   UPDATE Employee SET salary = salary + 200;
   UPDATE Employee SET salary = salary + 1000 WHERE name = 'Tony';
   COMMIT;
   ```

   $T_3$:
   ```
   UPDATE Employee SET salary = salary + 100 WHERE name = 'James';
   UPDATE Employee SET salary = salary + 200 WHERE name = 'Tony';
   COMMIT;
   ```

   The table `Employee` originally has two tuples, ('Tony', 1000) and ('James', 1000). Please assume that individual SQL statements are executed atomically.

   (a) (4 points) Assume that all three transactions run under the isolation level SERIALIZABLE. List all possible values that can be returned by $T_1$. Briefly explain your answer.

   **ANSWER:**
   2000, 2300, 3400, 3700.
   When all transactions run under SERIALIZABLE, any possible schedule is conflict equivalent to a serial schedule. Possible schedules for $T_1, T_2$ and $T_3$ are: $T_1T_2T_3$, $T_1T_3T_2$, $T_2T_1T_3$, $T_2T_3T_1$, $T_3T_1T_2$, and $T_3T_2T_1$. The outputs from $T_1$ are 2000, 2000, 3400, 3700, 2300, 3700, respectively.

   (b) (6 points) Assume that $T_1$ runs under the isolation level READ UNCOMMITTED and $T_2$ under REPEATABLE READ and $T_3$ under SERIALIZABLE. List all possible values that can be returned by $T_1$. Briefly explain your answer.

   **ANSWER:**
   2000, 2100, 2300, 2400, 2700, 3400, 3500, 3700.
   Only $T_2$ and $T_3$ are updating values, so let us focus on these two transactions first. Under REPEATABLE READ, the only exception to ACID is phantom, but because $T_2$ and $T_3$ do not insert any tuple, we do not need to worry about ACID exceptions for the two. So the possible schedules for $T_2$ and $T_3$ are equivalent to $T_2T_3$ or $T_3T_2$. Regarding $T_1$, since it is READ UNCOMMITTED, its SELECT statement may do a dirty read.
   Now let us consider the schedule $T_2T_3$. Under this schedule, the total salay value changes from $2000 \rightarrow 2400 \rightarrow 3400 \rightarrow 3500 \rightarrow 3700$. $T_1$ may read any of these salary sum values.

For the schedule $T_3T_2$, the total salay value changes from $2000 \rightarrow 2100 \rightarrow 2300 \rightarrow 2700 \rightarrow 3700$. Again, $T_1$ may read any of these salary sum values. Therefore, possible outputs from $T_1$ are 2000, 2100, 2300, 2400, 2700, 3400, 3500, 3700.

2. Consider the following schedule:

$$r_1(X)\, w_1(X)\, r_2(X)\, c_2\, r_3(Y)\, w_3(Y)\, c_3\, r_1(Y)\, w_1(Y)\, c_1$$

Circle YES or NO for the questions below and provide brief explanation. Be careful with your answers for this problem. For every correct answer, you will get 2 points, but for every incorrect answer, you will get 1 point deducted.

(a) Is the schedule serial? YES / NO
**ANSWER:**
`No`

(b) Is the schedule conflict serializable? YES / NO
**ANSWER:**
`Yes. The precedence graph is` $T_2 \leftarrow T_1 \leftarrow T_3$ `and has no cycle.`

(c) Is the schedule recoverable? YES / NO
**ANSWER:**
`No.` $c_2$ `appears before` $c_1$ `even though` $T_2$ `reads a value written by` $T_1$`.`

(d) Should the above schedule be allowed when all three transactions run under the isolation level SERIALIZABLE? YES / NO

**ANSWER:**
`No.` $T_2$ `does dirty read from` $T_1$ `and commits before` $T_1$`.`

Depending on your answer, provide an answer to one of the following questions. (2 points)

- If your answer is yes, write down the serial schedule that is equivalent to the above schedule.
- If your answer is no, what is the the minimum relaxation necessary to the isolation level of each transaction to allow this schedule? The isolation levels of the three transactions may or may not be the same after the relaxation.

$T_1$ isolation level: _____

$T_2$ isolation level: _____

$T_3$ isolation level: _____

**ANSWER:**
`No. Change the isolation level of` $T_2$ `to READ UNCOMMITTED.`