CSE 632 Final Examination (100 points)
19\textsuperscript{th} December: 5pm-7:30pm

December 19, 2002

Open Book + Notes. Printed material other than class notes NOT allowed.

1. **Functional Dependency (10 points)**

Let X, Y be sets of attributes of a relational schema. Prove or disprove the following statements. If true, give a brief proof. If false, provide a set of functional dependencies and sets of attributes X and Y that violates the statement.

(a) $X^+ \cup Y^+ = (X \cup Y)^+$
(b) $X^+ \cap Y^+ = (X \cap Y)^+$

2. **SQL (15 points)**

Suppose we have the following relations:

- `Salary(emp-id, amount)`
- `Dept(emp-id, dept-id)`
- `Name(emp-id, name)`

(a) Write an SQL query that obtains the pairs (name, salary) only for those employees who earn more than the average salary of their dept.

(b) Write an SQL query that returns the set of employee id’s for those employees that earn more money than someone outside their department (i.e., return employee X if there is an employee Y in a department other than X’s, who earns less than X).

3. **Datalog (15 points)**

You will write two datalog queries over the following schema:

- `Subpart(part, sub-part)`
- `Compatible(part1, part2)`
- `PowerReq(part, power)`

The database describes parts that are manufactured by your company. The relation `Subpart` identifies which parts are sub-components of others. The relation `Compatible` identifies pairs of compatible parts (you can assume the relation is symmetric). The relation `PowerReq` identifies the voltage level required for a particular part.

(a) Write a query that finds parts discrepancies in the database. i.e., find all parts X, such that X has a sub-component Y that requires more voltage than X.

(b) Find all parts that have incompatible sub-parts (i.e., find all pairs (X, Y), where X and Y are sub-parts (directly or indirectly) of Z, and X and Y are not compatible).

4. **Indexing (10 points)**

Consider two relations R(A, B) and S(B, C) stored as sequential files. There are two types of most frequently asked queries as follows.
(a) Q1: select * from R where R.A > a1 and R.A < a2
(b) Q2: select * from R, S where R.B = S.B

Suppose that we are allowed to build conventional dense, sparse, or multi-level indexes over the relations to optimize the above queries. What indexes shall we build over R and S to obtain the best query performance? Briefly specify the following aspects for each index you choose to build.

(a) What is the type of the index (dense, sparse, or multi-level)?
(b) Which attribute is the index built on?
(c) How should we use the index for the queries?

5. Recovery (20 points) A database has five elements, X, Y, Z, U, V. The two questions below ask us to recover the database after a system crash.

(a) Assuming we maintain an undo log whose content, after the crash, is:

```
<start T1>
<T1,X,3>
<start T2>
<T2,Y,1>
<start ckpt(T1,T2)>
<T1,X,4>
<start T3>
<commit T1>
<T2,Y,2>
<T3,Z,5>
<commit T2>
<end ckpt>
<start T4>
<T4,U,8>
<start ckpt(T3,T4)>
<start T5>
<T4,U,9>
<T5,X,10>
<commit T5>
<start T6>
<T6,Y,12>
```

Recover the database. Indicate which portion of the log you needed to inspect, and which transactions have to be executed again.

(b) Assuming we maintain a redo log whose content, after the crash, is:

```
<start T1>
<T1,X,3>
<start T2>
<T2,Y,1>
<start T3>
<T3,Z,5>
<T2,Y,2>
<commit T2>
<start ckpt(T1,T3)>
<T1,X,4>
<start T4>
<end ckpt>
<commit T1>
<T3,Y,2>
```
<T4, U, 8>
<start ckpt(T3, T4)>
<start T5>
<T4, U, 9>
<T5, X, 10>
<commit T4>
<start T6>
<T6, Y, 12>

Recover the database. Indicate which portion of the log you needed to inspect, and which transactions have to be executed again.

6. Query Optimization (15 points)

Consider a hash-join of two relations R and S having B(R) = 1000 and B(S) = 500. The values in R and S are skewed such that the hash function assigns twice as many tuples to even-numbered hash buckets as to odd-numbered buckets. How much memory would be required to perform the join in two passes?

7. Transaction Management (15 points)

For each of the following schedules with commit and abort actions, tell whether it is serial, conflict serializable, recoverable, avoids cascading rollbacks, strict 2PL (tell all that applies).

(a) w1(c) r2(a) w2(a) r1(a) w1(a) c1 c2
(b) w1(c) r2(a) w2(a) c2 r1(a) w1(a) c1
(c) w2(c) r2(a) w2(a) w1(c) c2 r1(a) c1