CSC 174 Spring 2024 Final Exam

Total: 100 points (100*0.4=400 Canvas Points)

First Name _____ Last Name _____

Section I

1. (18 points) Given relation R (x, y, a, b, c, g), where {x, y} is the only key. Given a set of functional dependencies F={fd1,fd2}, where
fd1: {x,y} → {a,b,c,g}
fd2: {c,g}→ b
Please decompose R into BCNF relations with Lossless join property. Present all the steps that help you to achieve your answer.

R(x, y, a, b, c, g) Key = (x, y)

fd1 : x, y → a, b, c, g

fd2 : c, g → b

step①: (x, y) is only candidate key

(x, y)+ = { a, b, c, g, x, y }

step②: Determine closure

(x, y)+ = { a, b, c, g, x, y }

(c, g)+ = { c, g, b }

hence,

Prime Attributes: x, y

non-Prime Attributes: a, b, c, g

step③: Find FD which violates BCNF

FD2 violates BCNF as c, g is not a super key

Relation R will decompose to

R1: x, y, a, c, g with FD: x, y → a, c, g

R2: c, g, b with FD: c, g → b

conclusion: Both R1 & R2 are in BCNF & decomposition ensures Lossless join.

2. (18 points) Given relation R(u,e,r,t,w, f), where the only key of R is (u,r,w). Given a set of functional dependencies E = {fd1: u → e, fd2: r→ t, fd3: u→f}. Given the minimal cover of E is F, and F=E.
Decompose R into 3NF relations with dependency preservation and lossless join properties. Present all the steps that help you to achieve your answer.

Step 1: Find minimal covers.

fd1: u → e
fd2: r → t
fd3: u → f

F = E

Step 2

candidate key (u, r, w)

Step 3, for each FD create new relations R

R1 (u, e)
R2 (r, t)
R3 (u, f)
R4 (u, R, w)

Step 4

All dependencies in E can be derived from new relations

fd1 is ~~derived~~ covered by R1
fd2 is covered by R2
fd3 is covered by R3

R is decomposed to R1 (u,e) R2 (r,t), R3(u, f) } R4(u, r, w)

3 (17 points) Given the following two tables to keep track of the students that a professor is advising. One professor can advise many students, while one student can only be advised by one professor. noOfStudent is a derived attribute that keeps how many students are advised by the professor.
Professor (PID, pname, office, noOfStudent)
Student (SID, sname, address, advisor) foreign key (advisor) references Professor (PID)
Note that "advisor" attribute does NOT have "not null" constraint.
Using MySQL, create an "after" trigger to maintain this derived attribute upon INSERT operation, that is, when a new tuple is inserted to the student table, the corresponding Professor table should be updated.

```
DELIMITER $
CREATE TRIGGER NoStu_insert
After Insert ON student
FOR EACH ROW
BeGIN
    IF NEW.advisor is NOT NULL THEN
        UPDATE PROFESSOR
        SET noOfStudent = noOfStudent+1
        WHERE PID = New.advisor;
    End IF;
END $$

DELIMITER ;
```

4. (16 points) Given the following execution sequence of transactions:

[start-trans T1]
...
[start-trans T2]
...
[commit T2]

[start-trans T3]
....
A:
[commit T1]
[checkpoint]
B:
[start-trans T4]
...
[start-trans T5]
...
C:
[commit T4]
[checkpoint]
...
[commit T3]
...
D:

If the above execution fails and corresponding log entries are retrieved at A, B, C, or D, what redo/undo operations are necessary at each point? If neither redo nor undo needed, please write "do nothing".

	Deferred	immediate
A	Redo T1, T2	Redo T1, T2
B	Redo T1, T2, T3	Redo T1, T2, T3
C	Redo T1, T2, T3, T4	Redo T1, T2, T3, T4
D	Redo T1, T2, T3, T4	Redo T1, T2, T3, T4

Section II The schema is shown in the last page
1. (15 points) Write a MySQL function: Input a department name, return the highest salary in this department. Please fill in blanks. Domain of Salary is DECIMAL(10,2).

```
delimiter $
CREATE FUNCTION high_sal (dept_name VARCHAR(20))

RETURNS DECIMAL(10,2)
BEGIN

    DECLARE max_salary DECIMAL(10,2);

SELECT MAX(SALARY) INTO max_salary

From Employee

JOIN Department ON Employee.DNO = Department.DNumber

WHERE Department.DNAME = dept_name;

RETURN mat_salary;

END$
delimiter ;
```

3. (16 points) Create a view to present all employees who have more than 2 dependents.
View name: Emp_more_dep
Attribute of the view: emp_ssn, first_name, last_name, no_of_dependents

```
Create View Emp_more_dep AS
Select
    e.ssn AS emp_ssn,
    e.FNAME AS first_name,
    e.LNAME AS last_name,
    Count (d.DEPENDENT_NAME) AS no_of_Dependent
FROM
    Employee e
JOIN
    Dependent d ON e.SSN = d.ESSN
GROUP BY
    e.SSN, e.FNAME, e.LNAME
Having
    Count(d.Dependent_NAME) > 2;
```