Database Management Systems (COP 5725)

Spring 2017

Instructor: Dr. Markus Schneider
TAs: Keke Zhai, Man Mohan Devineni

Homework 5

<table>
<thead>
<tr>
<th>Name:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>UFID:</td>
<td></td>
</tr>
<tr>
<td>Email Address:</td>
<td></td>
</tr>
</tbody>
</table>

Pledge (Must be signed according to UF Honor Code)

On my honor, I have neither given nor received unauthorized aid in doing this assignment.

_______________________________________________
Signature

For scoring use only:

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Maximum</th>
<th>Received</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercise 1</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>Exercise 2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Exercise 3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Exercise 4</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>Exercise 5</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Exercise 1 (Functional Dependencies) [24 points]

1. Given the attribute set \( U = \{ \text{name, color, category, department, price} \} \) and a set \( F = \{ \{ \text{name} \} \rightarrow \{ \text{color} \}, \{ \text{category} \} \rightarrow \{ \text{department} \}, \{ \text{color, category} \} \rightarrow \{ \text{price} \} \} \) of functional dependencies (FDs). Find the closure of the attribute set \{name\}, the closure of the attribute set \{name, category\}, and the closure of the attribute set \{color\}. Describe the single steps for computing the attribute closures. [4 points]

2. Check the equivalence of the following two sets \( F = \{ B \rightarrow CD, AD \rightarrow E, B \rightarrow A \} \) and \( G = \{ B \rightarrow CDE, B \rightarrow AC, AD \rightarrow E \} \) of FDs by using the Armstrong axioms. Describe each single axiom that you apply. [6 points]

Consider a relation schema \( R(A,B,C,D,E,F) \) and a set of functional dependencies \( \{ A \rightarrow D, AC \rightarrow DE, B \rightarrow ACD, D \rightarrow B \} \) (for questions 3, 4, and 5)

3. Compute a canonical cover for the above set of functional dependencies (show each step of your derivation with an explanation). [6 points]

4. Compute the closure of \( AF \) and then determine whether or not \( AF \) is a candidate key. [4 points]

5. Determine whether or not \( (A, E, F) \) is in BCNF and justify your answer using the transitive closure of a set of attributes. [4 points]
Exercise 2 (Chase Test) [20 points]

1. Perform the Chase test over the given decomposition. Show the steps by specifying the functional dependency and underlining the value that changes in each step, and conclude if the given decomposition is a lossless join decomposition.

   a) \( R(A, B, C, D, E) \), \( F = \{AB \rightarrow C, D \rightarrow E, C \rightarrow D\} \). \( P1 = \{R1(A, B, C), R2(C, D), R3(D, E)\} \) is a decomposition of \( R \). [6 points]

   b) Now we assume \( P2 = \{R1(A, C), R2(B, C, D), R3(D, E)\} \) and keep the other assumptions. Perform the Chase test for \( P2 \). [6 points]

2. Let \( R(A,B,C,D,E) \) be decomposed into relations with the following three sets of attributes: \( \{A,B,C\} \), \( \{B,C,D\} \), and \( \{A,C,E\} \). For the following set \( F=\{A \rightarrow D, D \rightarrow E, B \rightarrow D\} \) of FDs, use the Chase test to tell whether the decomposition of \( R \) is lossless. [8 points]

Exercise 3 (Normalization) [10 points]

Suppose we have a relation schema \( R(A, B, C, D, E, F, G) \) and a set of functional dependencies \( F = \{BCD \rightarrow A, BC \rightarrow E, A \rightarrow F, F \rightarrow G, C \rightarrow D, A \rightarrow G, A \rightarrow B\} \). Decompose \( R \) into 3NF. Show all steps and argue precisely. Is this decomposition also in BCNF? If so, why. If not, why not? [10 points]
Exercise 4 (Triggers) [24 points]

Considering the following tables in a database:

```sql
CREATE TABLE EMP
(EMPNO NUMBER(4) NOT NULL,
ENAME VARCHAR2(10),
JOB VARCHAR2(9),
MGR NUMBER(4),
HIREDATE DATE,
SAL NUMBER(7, 2),
COMM NUMBER(7, 2),
DEPTNO NUMBER(2));

CREATE TABLE DEPT
(DEPTNO NUMBER(2),
MGRNO NUMBER(2),
DNAME VARCHAR2(14),
LOC VARCHAR2(13) );

CREATE TABLE BONUS
(ENAME VARCHAR2(10),
JOB VARCHAR2(9),
SAL NUMBER,
COMM NUMBER);

CREATE TABLE SALGRADE
(GRADE NUMBER,
LOSAL NUMBER,
HISAL NUMBER);

CREATE TABLE GRADENUM
(GRADE NUMBER,
NUM NUMBER,
HIGH NUMBER);
```

1. Add a constraint to table 'EMP' that checks whether 'DEPTNO' refers to column 'DEPTNO' in table 'DEPT'. The constraint should also guarantee that once a department in the 'DEPT' table is deleted, the employee records of that department in table 'EMP' are also deleted. [3 points]
2. Create a constraint that checks the attribute 'SAL' of table 'EMP' whether the salary of employees is higher than 2000 and lower than 10000. Also, create a primary key constraint on the empno and create a constraint on the ename to be unique. [3 points]

3. Create a trigger that displays the employees' average salary, before an employee's record in table 'EMP' is updated or before new employee’s record is inserted. [4 points]

4. Create a trigger that sets the value of 'COMM' in table 'EMP' equal to the value of 'COMM' in 'BONUS' for employees whose bonus record is updated in table 'BONUS'. If the employee is not in the 'EMP' table, insert a new record into the 'EMP' table with that employee's name, set 'EMPNO' to the current highest 'EMPNO' + 1, and leave the other columns as null. [6 points]

5. Create a trigger that does the following: If a department’s deptno is updated then update the employees’ depno also who works for the department. [4 points]

6. Create a trigger that does the following: When the employees’ salary is modified (less than before or more than 20%), then display the error message. [4 points]
Exercise 5 (Function, Block and Procedure) [22 points]

We assume the same database schema as in Exercise 3.

1. Create a function that calculates the summation of all employees' salaries of a given department.
   - input: PDNAME: department name
   - return: SUM_SAL: summation of all employees' salaries

Then write an anonymous block to call your function and output the result. (function, block) [6 points]

2. Write a block which updates the 'EMP' table. If an employee's salary is less than 3000, then increase his salary by 5 percent, and output the employee’s name, original salary and new salary. Make use of the cursor concept. [5 points]

3. Create a procedure that takes department number and changes the manager for the department to the employee in the department with highest salary. [5 points]

4. Write a procedure which first groups the salaries of employees. If one's salary falls into a salary boundary bounded by 'LOSAL' and 'HISAL' in table 'SALGRADE', then we consider this employee in the group of this 'GRADE'. Then insert records to table 'GRADENUM', set 'NUM' as the number of employees whose salary fall into its boundary. If 'NUM' > 3, set 'HIGH' = 1, otherwise, set 'HIGH' = 0; (procedure, cursor, if else) [6 points]