Database Management Systems (COP 5725)

(Fall 2011)

Instructor:
Dr. Markus Schneider

TA:
Hechen Liu,Virupaksha Kanjilal, Eyup Ayaz

Exam 3

Name:	
UFID:	
Email Address:	

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:

	Maximum	Received
Exercise 1	20	
Exercise 2	35	
Exercise 3	30	
Exercise 4	15	
Total	100	

Exercise 1 (Functional Dependencies) [20 points]

1. What is "functional" in the term of *functional dependency*? [3 points]

It has the characteristic of function: X is functional dependent on Y, or X→Y, if, and only if, each X value is associated with precisely one Y value.

2. Consider the Relational Schema *R* (*A*, *B*, *C*), and FDs *A*→*B* and *B*→*C* hold in *R*. Prove the transitivity rule, that *A*→*C* also holds in *R*. Show your proof in a formal way (using mathematical notations). [4 points]

$\forall t_1, t_2 \in R$
From A→B, we infer if $t_1[A]=t_2[A]$ then $t_1[B]=t_2[B]$
From B→C, we infer if $t_1[B]=t_2[B]$,then $t_1[C]=t_2[C]$,
Then for all $t_1[A]=t_2[A]$, $t_1[C]=t_2[C]$is derived.
Therefore C is functional dependent on A.

3. Assume that we have a relation schema *R*(*A*,*B*,*C*) representing a relationship between two entity sets with keys *A* and *B*, respectively, and suppose that among all the FDs hold on *R*, *A* → *B* and *B* → *A* exist. Explain what such pair of FDs imply in the relationships between *A* and *B* in the Entity-Relationship model. [4 points]

The cardinality of this relationship is 1:1

4. Let *R* (*A*, *B*, *C*, *D*, *E*, *F*) be a relation with FDs *AB*→*C*, *BC*→*AD*, *D*→*E* and *CF*→*B*. Is {*A*, *B*} a candidate key of *R*? Prove if your answer is YES or explain otherwise. [4 points]

AB^+={A, B, C, D, E}
F is not in AB^+
Therefore {A, B} is not a candidate key of R

5. Consider the relational schema *R* {*A*, *B*, *C*, *D*} and the two sets *F* = {*AC*→*B*, *A*→*C*, *D*→*A*} and *G* = {*A*→*B*, *A*→*C*, *D*→*A*, *D*→*B*} of FDs. Use Armstrong's axioms to prove that these two sets are equivalent. [5 points]

We have to show that $G^+ \subseteq F^+$, and $F^+ \subseteq G^+$. The FDs *A*→*C* and *D*→*A* belong to both *F* and *G*; thus, the derivation is trivial.

We show that *A*→*B* ∊ *G* is implied by F. We begin with *A*→*C* ∊ *F*. Applying Armstrong's augmentation axiom leads to *A*→*AC*. Now, in addition, we take *AC* → *B* ∊ *F* and apply Armstrong's transitivity axiom. We obtain A→B

Next, we show that *D*→*B* ∊ *G* is implied by F. We begin with *D*→*A* ∊ *F* and additionally consider *A*→*B* derived in the previous step. By applying Armstrong's transitivity axiom, we obtain *D*→*B*. Therefore $G^+ \subseteq F^+$

Finally, we show that AC→B ∊ F is implied by G. We begin with A → B ∊ G. Applying Armstrong's augmentation axiom leads to AC→BC. Armstrong's decomposition axiom yields AC→B. Therefore $F^+ \subseteq G^+$

You can also solve this question by proving Fc=Gc (canonical covers are equal)

Exercise 2 (Normal Forms) [35 points]

1. Answer the following questions briefly but precisely.

(a) The definition of Normalization shows that the normalization process is to eliminate "anomalies" by decomposing a relation schema into several relation schemas. What are the anomalies? [4 points]

Insert anomaly
Update anomaly
Delete anomaly

(b) Which two important properties does the Third Normal Form guarantee? Explain what these two properties mean. [4 points]

Losslessness (lossless join decomposition): An arbitrary instance must be re-constructable from the decompositions

Dependency preservation: All FDs which hold for schema R must be transferable to the schemas R_1... R_n and must be efficiently checkable

(c) Prove that any relation schema *R* with TWO attributes (*A*, *B*) is in BCNF. [6 points]

1) If there is no non-trivial dependencies in R, then AB is the key, all the FDs are trivial, which satisfies BCNF
2) If A→B exists, then A is the key, which does not violate BCNF
3) If B→A exists, then B is the key, which does not violate BCNF

2. Consider the following relation for published books, answer the questions below.

BOOK (Book_title, Author_name, Book_type, Listprice, Author_affiliation, Publisher)

Suppose the following dependencies exist:

Book_title → Publisher, Book_type

Book_type → Listprice

Author_name→Author_affiliation

(a) What normal form is the relation in? Explain your answer. [4 points]

This relation is in 1NF. The key is {Book_title, author_name}

Dependency Book_type→Listprice violates 2NF

(b) Apply normalization until you cannot decompose the relations further. Which normal form is it in after your final decomposition? State the reasons behind each decomposition. [5 points]

Canonical cover:

FD1: Book_title → Publisher, Book_type

FD2: Book_type → Listprice

FD3: Author_name→Author_affiliation

Key is {Book_title, author_name}

From FD1, we get {Book_title, Publisher, Book_type}

From FD2, we get {Book_type, Listprice}

From FD3, we get {Author_name, Author_affiliation}

Add the candidate key {Book_title, author_name}

The result is in BCNF.

3. Let R (A, B, C, D, E, G, H, I) be a relation schema with the set of FDs $F=\{H\to GD, E\to D, HD\to CE, BD\to A\}$ of functional dependencies on this schema. Answer the following questions.

(a) Compute the canonical cover of F. [4 points]

We first get
$H\to CEGD, E\to D, BD\to A$

After right reduction, we get

$\{H\to CEG, E\to D, BD\to A\}$

(b) What is (are) the key(s) of R? [4 points]

BHI is the key

(c) Find a BCNF decomposition of R. [4 points]

{CEGH} {ED} {ABD} {BHI}

Exercise 3 (Integrity Constraints and Triggers) [30 points]

1. List 3 problems of triggers that must be paid attention to when you design them. [6 points]

   1) User must control that triggers do not contradict each other
   2) A trigger can activate another trigger. Cycles should be avoided
   3) Termination of events
   4) If a consistency condition can be formulated by an integrity constraint, triggers should not be used

2. One important integrity constraint you learned in the class is the foreign key constraint. Correct adding of foreign key will keep the consistence of the data. Assume you have the following database schemas for an online shop.

Customer (cid: integer, cname: varchar2(100), email: varchar2(100), address: varchar2(100), phone_num: varchar2(15));
Product(pid: integer, pname:varchar2(100), price: numeric (7, 2), category: varchar2 (100));
Orders (cid: integer, pid: integer, amount: integer)
Favorite (cid: integer, pid: integer)

The *Orders* and *Favorite* tables are tables of relationships. They both refer to the Customer table and the Product table respectively. The difference is the deletion operation. For any deletion on users or products, there is no need to keep the orders any longer. However, the company would like to do further analysis on the number of records in the Favorite table. Write the SQL statements to create the *Orders* and *Favorites* tables respectively. Assume that all the other tables have already existed. Add the foreign key constraints according to the above explanation. State clearly how the foreign key constraints will perform on the deletion operation. [8 points]

```
Create table Orders
(cid:integer, pid:integer,
amount:integer,
primary key (cid: pid),
foreign key cid references Customer (cid) on delete cascade,
foreign key pid references Product (pid) on delete cascade);

Create table Favorite
(cid:integer, pid:integer,
primary key (cid: pid),
foreign key cid references Customer (cid) on delete set null,
foreign key pid references Product (pid) on delete set null);
```

3. Consider the following SQL declarations:

```
Create table Employee
(ID integer primary key,
 salary integer,
 dept_num integer references Department(number));

Create table Department
(number integer primary key,
salaryCap integer);

Create assertion Policy check(
     Not exists (select *
               From Employee, Department
               Where Employee.dept_num  = Department.number
               And Employee.salary > Department.salaryCap))
```

(Hint: salaryCap is a limit on the amount of money a department spends on salary per employee.)

(a) State in English the policy enforced by the assertion. [4 points]

After any insertion or update on the Employee or the Department table, there should not be any employee having a salary greater than the salary cap in the corresponding department of the employee

(b) Revise the above table declarations to use table constraints instead of assertions. Your constraints should be defined so that under no circumstances the policy can be violated. [8 points]

In the employee table define the following tuple based check constraint:

```
CHECK ( salary <=  (Select salaryCap  from Department
                   where Department.number=dept_num) )
```

In the department table add the following check constraint:

```
CHECK ( salaryCap >= ALL (Select salary from Employee
                         where  Employee.dept_num=number) )
```

(c) Compare your solution in (b) with the given assertion solution. Which one is better? Why? [4 points]

Both methods work. They will check for potential violation of the assertion whenever one of the relations is updated. However an assertion will introduce more overhead than a table constraint.

Exercise 4 (Database Application Programming) [15 points]

1. In PL/SQL, you can create stored functions. For example

```
SQL> create or replace function square (x in int)
  2  return int as
  3  begin
  4   return x*x;
  5  end square;
  6  /
```

Will create the Square operation for integer values. Briefly answer the following questions,

(a) Create the *Add* operation for two input integers. [2 points]

```
SQL> create or replace function addop (x in int, y in int)
  2  return int as
  3  begin
  4   return x+y;
  5  end addop;
  6  /
```

(b) What are the advantages of creating your own stored functions in PL/SQL? [3 points]

Can use more powerful functions that have not been defined in the database query language.

(c) Find out two limitations using stored functions in PL/SQL (Hint: think about the input and output data types of the functions). [4 points]

1) Cannot perform aggregation queries on groups
2) All data types of the input must be known in the DBMS
3) All data types of the output must be known in the DBMS

2. Compare the differences between PHP and JSP in web-based database applications. List their advantages and disadvantages. [6 points]

JSP:

Advantages:

Java classes can be used in JSP pages as beans to create dynamically driven web pages.
Response time is faster because the byte code is already compiled
Manageable and scalable when the application gets larger
Much more support available and major DBMS providers tend to support JSP because it is one of the industry standards
Plat form independent
Used to develop n-tier (data layer, middle layer, front and back-end) web applications
…

Disadvantages:

Not very useful for small applications
More costly than PHP or CGI because hardware support is more demanding
Will run slower if the hardware is not up to date
Project deployment might get very complicated for larger applications (i.e. Setting up class paths etc)
Requires tomcat extension to the Apache server to run JSP pages
…

PHP:

Advantages:

Useful for smaller applications
Strong scripting power and string manipulation
Very easy to configure and deploy
Application server Apache is freely available
…

Disadvantages:

Response time might be slow because the script has be to interpreted as it runs
Gets messy when the application size gets bigger
Not very scalable or manageable
Mostly used for light weight web projects with light weight DBMS like MySQL etc.
…

*** END ***