Database Management Systems (COP 5725)
(Spring 2010)

Instructor: Dr. Markus Schneider

TA: Hechen Liu, Wenjie Yuan, Ravi Tiwari

Exam 3 Solutions

Name	
UF Id	
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
Question 1	20	
Question 2	30	
Question 3	20	
Question 4	18	
Question 5	12	
Total	100	

Question 1 (Functional Dependencies, Normal Forms) [20 points]

Briefly answer the following questions by giving precise and concise explanations:

1. Give a set of FDs for the relation schema *R(A,B,C,D)* with single valued attributes and primary key *AB* under which *R* is in 1NF but not in 2NF. Explain your selection. [5 points]

   Solution:

   Consider the set of FD: $AB \rightarrow CD$ and $B \rightarrow C$. The non prime attribute C is not fully functionally dependent on the primary key AB. Hence, it is not in 2NF. And as we consider all attribute values as atomic, the schema is in 1NF.

2. Give a set of FDs for the relation schema *R(A,B,C,D)* with primary key *AB* under which *R* is in 2NF but not in 3NF. Explain your selection. [8 points]

   Solution:

   Consider the set of FD: $AB \rightarrow CD$ and $C \rightarrow D$. *AB* is obviously a super key for this relation, since $AB \rightarrow CD$ implies $AB \rightarrow ABCD$. It is a primary key since there are no smaller subsets of keys that hold over *R(A,B,C,D)*. The relation R is in 2NF, as non-prime attributes C and D are fully functional dependent on the primary key AB of the schema, but the functional dependency $C \rightarrow D$ violates 3NF because:

   - $D \in C$ is false; that is, it *is not* a trivial FD
   - *C is not* a superkey
   - *D is not* part of some key for *R*

3. Consider the relation schema *R(A,B,C)*, which has the FD $B \rightarrow C$. If A is a candidate key for R, is it possible for R to be in BCNF? If so, under what conditions? If not, explain why not. [4 points]

   Solution:

   The only way *R* could be in BCNF is if *B* includes a key, *i.e. B* is a key for R.

4. Suppose 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 R has (among others) the FDs $A \rightarrow B$ and $B \rightarrow A$. Explain what such a pair of dependencies means (i.e., what they imply about the relationship that the relation models). [4 points]

   Solution:

   It means that the relationship is one to one. That is, each A entity corresponds to at most one *B* entity and vice-versa. (In addition, we have the dependency $AB \rightarrow C$, from the semantics of a relationship set.)

Question 2 (Lossless Join Decomposition & Dependency Preservation) [30 points]

1. Define loss-less join decomposition. [5 points]

   Solution:
   Refer to the class notes

2. Let a relation R be decomposed into R1,R2, ..., Rn. Let F be a set of functional dependencies on R. Explain precisely what it means for FD to be preserved in the set of decomposed relations. [5 points]

   Solution:

   Let *Fi* denotes the projection of *F D* on *Ri*. *F D* is preserved if the closure of the union of *Fi's* equals *F*.

3. Consider a database of ship voyages with the following attributes: S (ship name), T (type of ship), V (voyage identifier), C (cargo carried by one ship on one voyage), P (port) and D (day). We assume that a voyage consists of a sequence of events where one ship picks up a single cargo, and delivers it to a sequence of ports. A ship can visit only one port in a single day. [20 points]

   i. Based on the above description, identify the set of functional dependencies. [5 points]

      Solution:

      This description implies the following functional dependencies:
      $S \rightarrow T$
      $V \rightarrow SC$
      $SD \rightarrow PV$

   ii. Determine the key(s) for this relation (assuming all attributes are in one table to begin with). [5 points]

      Solution:
      The set of attributes are: (S, T, V, C, D, P). Examining the FD's, we see that D does not appear on the right hand side of any dependency, therefore it must be part of any key. So the smallest set that can be a key is {D}. However, D^+= {D}, so we examine two elements sets. Picking {S, D}, we get $\{S, D\}^+$={ S, T, V, C, D, P } and therefore it is a key. Similarly, $\{V, D\}^+$={S, C, P, V, T, D} and therefore it is a key. Sets {D, C}, {D,T} and {D,P} will not give us a key.

   iii. Provide possible lossless join decompositions into BCNF and identify whether they are dependency preserving. [10 points]

      Solution:

      The non-BCNF dependencies are $S \rightarrow T$ and $V \rightarrow SC$

      First lossless join decomposition into BCNF is ST, VC, VS, VDP

Att(r1) union Att(r2) union Att(r3) union Att(r4) = Att(R) where r1 = ST, r2 = VC, r3 = VS, r4 = VDP and r = ship voyages (i.e STVCSDP).

VC join VS = VCS → lossless

VCS join VDP = VCSDP → lossless

VCSDP join ST = STVCSDP → lossless

Hence it is lossless join decomposition

Second Lossless join decomposition is ST, VCS, VDP.

Att(r1) union Att(r2) union Att(r3) = Att(R) where r1 = ST, r2 = VCS, r3 = VDP, and r = ship voyages (i.e STVCSDP).

VCS join VDP=VCSDP →lossless

ST join VCSDP= STVCSDP →lossless

Hence this is also lossless join decomposition.

None of the above two lossless join decompositions preserves dependencies, because to be able to check SD → PV (which is SD → P and SD→ V) we need to keep them together.

Question 3 (Normalization) [20 points]

1. Give the definition of 3NF. State whether the statement "A relational schema R is in 3NF if every non-prime attribute of R meets both of the following conditions: (i) It is fully functionally dependent on every key of R, (ii) It is non-transitively dependent on every key of R" is true or false. Please explain your answer in detail. [10 points]

Solution:

A relation schema is in 3NF, if whenever non-trivial functional dependencies X→A in R either hold:
(a) X is a super key of R or
(b) A is a prime attribute of R.

The statement is true, as it is based on the definition of 3NF; a relation schema violates 3NF if a functional dependency X→A holds in R that violates both the conditions (a) and (b) in the definition. Violating (b) means A is a non-prime attribute. Violating (a) means that X is either non-prime or it is a proper subset of a key of R. If X is non-prime, we typically have a transitive dependency that violates 3NF, whereas if X is a proper subset of a key of R, we have partial dependency that violates 3NF.

2. Consider the relation R ($ABCDEFGH$), with FDs $BC \rightarrow AD$, $E \rightarrow FH$, $F \rightarrow GH$. Decompose R into BCNF and list each step of the process.
[10 points]

Solution:

For each functional dependency check for the left reduction, right reduction, and see if the union rule is applicable to further reduce the set of functional dependencies. Finally, the Canonical cover i.e. the minimized set of functional dependencies will be:
FD1: *BC→ AD*
FD2: *E → F (Right reduction on E → FH, as H an extraneous attribute)*
FD3: *F→ GH*

Based on the given Relation and the functional dependencies the Candidate key will be {BCE}. As {B,C,E} are the minimum set of attributes, which together can identify any tuple in the relation R.

Decomposition:
From FD1: R3a = {B, C, A, D}
From FD2: R3b = {E, F}
From FD3: R3c = {F, G, H}
Since no relation contains the candidate key, additionally create a relation

R3d = {B, C, E}

Now as in all the relations R3a, R3b, R3c and R3d the left side of the functional dependencies that hold true for the respective relations are super keys, hence, they are in BCNF.

Question 4 (Database Application Programming) [18 points]

i. Explain how the following steps are performed in JDBC. [6 points]

1. Connect to a data source. [2 points]

   Solution:

   Connecting to a data source in JDBC involves the creation of a *Connection* object. Parameters for the connection are specified using a *JDBC URL* that contains things like the network address of the database server and the username and password for connecting. SQLJ makes calls to the same JDBC driver for connecting to a data source and uses the same type of JDBC URL.

2. Start, commit, and abort transactions. [2 points]

   Solution:

   Each connection can specify how to handle transactions. If the *auto-commit* flag is set, each SQL statement is treated as a separate transaction. If the flag is turned off, there is a *commit ()* function call that will actually commit the transaction. The *auto-commit* flag can also be set in SQLJ. If the flag is not set, transactions are committed by passing a *COMMIT* SQL statement to the DBMS.

3. Call a stored procedure. [2 points]

   Solution:

Stored procedures are called from JDBC using the *CallableStatement* class with the SQL command *{CALL StoredProcedureName}*. SQLJ also uses *CALL StoredProcedureName* to execute stored procedures at the DBMS.

ii. Briefly answer the following questions. [12 points]

1. Explain the following terms: Cursor, Embedded SQL, JDBC, SQLJ, stored procedure. [4 points]

   Solution:

   A **cursor** enables individual row access of a relation by positioning itself at a row and reading its contents. **Embedded SQL** refers to the usage of SQL commands within a host program. **JDBC** stands for Java Database Connectivity and is an interface that allows a Java program to easily connect to any database system. **SQLJ** is a tool that allows SQL to be embedded directly into a Java program. A **stored procedure** is program that runs on the database server and can be called with a single SQL statement.

2. What are the differences between JDBC and SQL? [4 points]

   Solution:

   SQLJ provides embedded SQL statements. These SQL statements are static in nature and thus are preprocessed and precompiled. For instance, syntax checking and schema checking are done at compile time. JDBC allows dynamic queries that are checked at runtime. SQLJ is easier to use than JDBC and is often a better option for static queries. For dynamic queries, JDBC must still be used.

3. Explain the term stored procedure, and give examples why stored procedures are useful. [4 points]

   Solution:

Stored procedures are programs that run on the database server and can be called with a single SQL statement. They are useful in situations where the processing should be done on the server side rather than the client side. Also, since the procedures are centralized to the server, code writing and maintenance is simplified, because the client programs do not have to duplicate the application logic. Stored procedures can also be used to reduce network communication; the results of a stored procedure can be analyzed and kept on the database server.

Question 5 (Data Integrity) [12 points]

1. Discuss the strengths and weaknesses of the trigger mechanism. Contrast triggers with other integrity constraints supported by SQL. [4 points]

Solution:

A trigger is a procedure that is automatically invoked in response to a specified change to the database. The advantages of the trigger mechanism include the ability to perform an action based on the result of a query condition. The set of actions that can be taken is a superset of the actions that integrity constraints can take (i.e. report an error). Actions can include invoking new update, delete, or insert queries, perform data definition statements to create new tables or views, or alter security policies. Triggers can also be executed before or after a change is made to the database (that is, use old or new data).

Following are some disadvantages of using triggers:

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

2. Consider the following relations: [8 points]

Student(<u>*snum:* integer</u>, *sname:* string, *major:* string,
level: string, *age:* integer)
Class(<u>*name:* string</u>, *meets at:* time, *room:* string, *fid:* integer)
Enrolled(<u>*snum:* integer, *cname:* string</u>)
Faculty(<u>*fid*: integer</u>, *fname:* string, *deptid:* integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Write the SQL statements required to create these relations, including appropriate versions of all primary and foreign key integrity constraints.

Solution:

The SQL statements needed to create the tables are given below:

```
CREATE TABLE Student ( snum INTEGER,
sname CHAR(20),
major CHAR(20),
level CHAR(20),
age INTEGER,
PRIMARY KEY (snum))

CREATE TABLE Faculty ( fid INTEGER,
fname CHAR(20),
deptid INTEGER,
PRIMARY KEY (fnum))


CREATE TABLE Class ( name CHAR(20),
meets atTIME,
room CHAR(10),
fid INTEGER,
PRIMARY KEY (name),
```

```
FOREIGN KEY (fid) REFERENCES Faculty)

CREATE TABLE Enrolled (snum INTEGER,
cname CHAR(20),
PRIMARY KEY (snum, cname),
FOREIGN KEY (snum) REFERENCES Student),
FOREIGN KEY (cname) REFERENCES Class)
```