Database Management Systems (COP 5725)

Spring 2016

Instructor:
Dr. Markus Schneider

TAs:
Kyuseo Park, Lin Qi

Exam 3 Solutions

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
Question 1	30	
Question 2	25	
Question 3	25	
Question 4	20	
Total	100	

Question 1 (Functional Dependencies) [30 points]

Consider the following set S of functional dependencies:

$A \rightarrow B$	(1)
$AB \rightarrow C$	(2)
$AC \rightarrow B$	(3)
$B \rightarrow C$	(4)

1. [2 points] Given (1) and (4), prove (2) using Armstrong axioms. Please indicate the rules that have used in each step.

$AB \rightarrow BB$	Augmentation of (1)
$AB \rightarrow B$	$BB = B$
$AB \rightarrow C$	Transitivity of (4) and above

2. [2 points] Given (1) and (4), prove (3) using Armstrong axioms. Please indicate the rules that have used in each step.

$AC \rightarrow BC$	Augmentation of (1)
$AC \rightarrow B$	Decomposition of above

3. [3 points] Give all candidate keys of relation *R(ABC)* that satisfies (1-4). Explain your answer.

There is only one candidate key A. $\{A\}^+=\{ABC\}$

4. [3 points] Give a minimal cover for the set *S*.

The minimal cover for *S* is {A->B, B->C}, because the other FDs can be generated from these two.

For the following questions, suppose you are given the functional dependencies set $S = \{AB \rightarrow C, C \rightarrow B, C \rightarrow D\}$.

5. [5 points] Given *S*, is the relation $R_1(A,B,C)$ in 3NF? If yes, justify. If no, specify at least one FD which violates the definition.

The relation R_1 is in 3NF. AB->C is OK because AB is a superkey and C->B is OK because B is part of the superkey AB (C->D is not applicable because D is not in R_1).

6. [5 points] Given *S*, which normal form(s) (BCNF, 3NF) does relation R_2(C,D) obey?

R_2 is in BCNF. C->D is the only applicable FD and BCNF is satisfied because C is a candidate key for R_2. Since BCNF is a stronger normal form than 3NF, it is also in 3NF.

7. [5 points] Is it correct to say that any relation with two attributes is in BCNF? If yes, please prove it. If not, please give a counter-example.

We need to examine the possible nontrivial FD's with a single attribute on the right. In what follows, suppose that the attributes are A and B.

1) There are no nontrivial FD's. Then surely the BCNF condition must hold since only a nontrivial FD can violate this condition. Incidentally, note that {A, B} is the only key in this case.

2) A -> B holds, but B -> A does not. In this case, A is the only key, and each nontrivial FD contains A on the left (in fact the left can only be A). Thus there is no violation of the BCNF condition.

3) B -> A holds, but A -> B does not. This case is symmetric to the above.

4) Both A -> B and B -> A hold. Then both A and B are keys. Surely any FD has at least one of these on the left, so there can be no BCNF violation.

8. [5 points] Usually, a decomposition of a database schema in Boyce-Codd normal form (BCNF) relation schemas is regarded as the "ultimate" decomposition in order to eliminate redundancy and potential update anomalies. Why is the third normal form (3NF) still of interest?

Sometimes, there simply is no decomposition into BCNF that is dependency preserving. For example, consider the relation schema ABC, if we have the FDs AB $\rightarrow$ C, and C$\rightarrow$B, then ABC is not in BCNF because C is not a key. If we try to decompose it, however, we cannot preserve the dependency AB$\rightarrow$C. In contrast, 3NF can guarantee both lossless join decomposition and dependency preservation.

Question 2 (Normalization) [25 points]

StaffNo	DentistName	PatientNo	PatientName	Appointment		SurgeryNo
				date	time	
S001	Michael Pearson	P1005	Jake Mill	23-Sep-13	10:00	SU05X
S001	Michael Pearson	P1129	Nicolas Reese	24-Sep-13	11:00	SU08X
S402	Zheng Lei	P1130	Noah Fuller	10- Aug - 13	09:30	SU19Z
S402	Zheng Lei	P1130	Noah Fuller	17-Aug-13	09:30	SU19Z
S303	Joanna Lawson	P0020	Paul Bates	05-Jun-13	14:00	SU03X
S303	Joanna Lawson	P1005	Jake Mill	08-Oct-13	11:00	SU20P

1. [12 points] The table shown above provides some details of patient dental appointments. It is susceptible to update anomalies. Provide examples of insertion, deletion, and modification anomalies. Please describe the examples concisely.

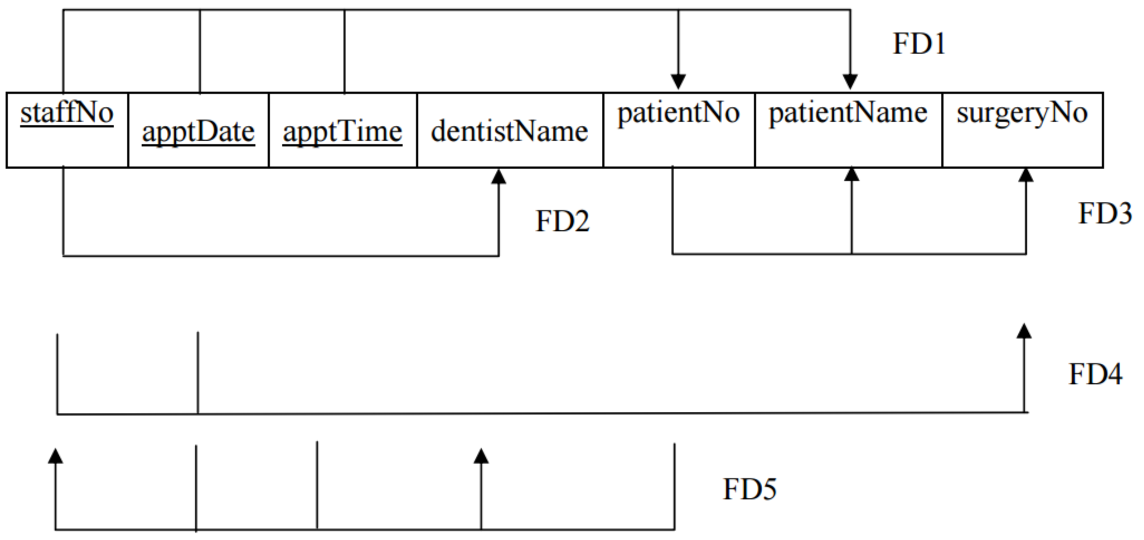
Insertion anomalies: To insert a new patient particular that makes an appointment with the designated dentist, we need to enter the correct detail for the staff. For example, to insert the details of new patient in PatientNo, PatientName and an appointment, we must enter the correct details of the dentist (StaffNo, DentistName) so that the patient details are consistent with values for the designated dentist for example, S011. To enter new patient data that doesn't have dentist to be assigned we can't insert NULL values for the primary key.

Deletion anomalies: If we want to delete a patient named Noah Fuller for example, two records need to be deleted as in row 3 and 4. This anomaly also obvious when we want to delete the DentistName, multiple records needs to be deleted to maintain the data integrity. When we delete a dentist record, for example Michael Pearson, the details about his patients also lost from the database.

Modification anomalies: With redundant data, when we want to change the value of one columns of a particular dentist, for example the DentistName, we must update all the dentist records that assigned to the particular patient otherwise the database will become inconsistent. We also need to modify the appointment schedules because different Dentist has different schedules.

2. [13 points] Describe and illustrate the process of normalizing the table above to 3NF. State any assumptions you make about the data shown in this table.

In the 1NF we remove all the repeating groups (appointment), assign new columns (apptDate and apptTime), and assign primary keys (candidate keys). Then we figure out the functional dependencies (FDs). By using a dependency diagram we represent the table as shown below.



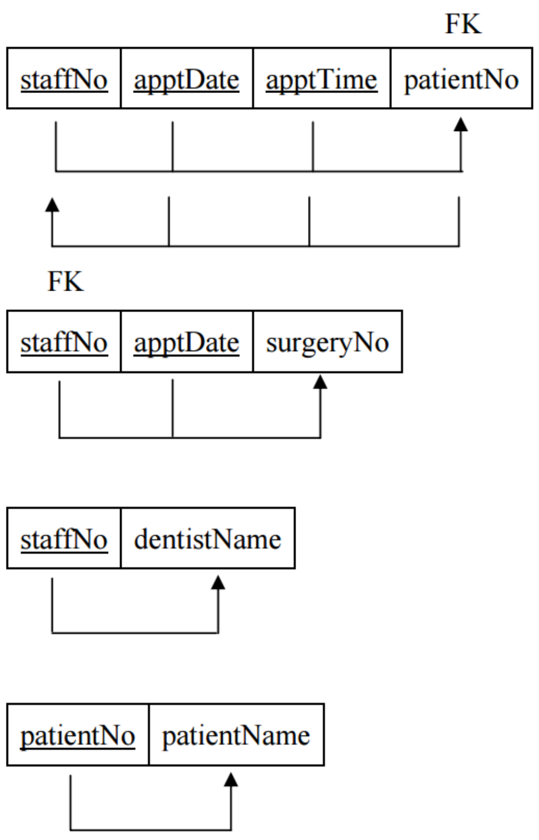
FD1 is already in 2NF. In this case, we can see that FD2 (just depends on staffNo) and FD4 (just depends on staffNo and apptDate) violate the 2NF. These two NFs are partially dependent on the candidate keys but not the whole keys. FD2 can stand on its own by depending on the staffNo and meanwhile FD4 also can stand on its own by depending on the staffNo. FD3 violates the 3NF showing the transitive dependency where surgeryNo and patientName depend on patientNo while patientNo depends on the staffNo that is the non-key depending on another non-key. So we need to remove the FD2 and FD4 by splitting into new tables and at the same time creating foreign keys. The new tables that are in 2NF are shown below.

staffNo	apptDate	apptTime	patientNo	patientName

staffNo	apptDate	surgeryNo

staffNo	dentistName

Finally in 3NF we must remove the transitive dependencies. In this case we remove FD3 by splitting the first relation into two new tables. The transitive dependency left is the patientName that depends on the patientNo, so we split this into a new table while creating a foreign key.



By re-arranging and giving names to the tables, we get the following 3NF decomposition:

staffNo	dentistName

FK

staffNo	apptDate	surgeryNo

patientNo	patientName

FK

staffNo	apptDate	apptTime	patientNo

Question 3 (Data Integrity) [25 points]

Consider the following database concerning World War II ships:

Classes (class, type, country, numGuns, weight)
Ships (name, class, launched)
Battles (name, date)
Outcomes (ship, battle, result)

Ships are built in "classes" of the same design, and a class is usually named after the first ship of that class. The relation Classes records the name of the class, the type ('bb' for battleship, 'bc' for battlecruiser), the country that built the ship, the number of main guns, and the weight (in tons). Relation Ships records the name of the ship, the name of its class, and the year in which the ship was launched. Relation Battles gives the name and date of battles involving these ships, and relation Outcomes gives the result (sunk, damaged, or ok) for each ship in each battle.

1. Write **assertions** for each of the following conditions.

(a) [6 points] No class may have more than 3 ships.

(b) [6 points] No ship may be launched before the ship that bears the name of this class.

Solution:

```
a) CREATE ASSERTION CHECK
  ( 3 >= ALL
            (SELECT COUNT(*) FROM Ships GROUP BY class)
  );

b) CREATE ASSERTION CHECK
  ( NOT EXISTS
   (SELECT s1.name FROM Ships s1
    WHERE s1.launched < (SELECT s2.launched FROM Ships s2
                          WHERE s2.name = s1.class
                  )
     )
   );
```

2. Write the following triggers. In each case, disallow or undo the modification if it does not satisfy the stated constraints.

(a) [6 points] When a new class is inserted into "Classes", also insert a ship with the name of that class and a NULL launch data.

(b) [7 points] If a tuple is inserted into "Outcomes", check that the battle is listed in "Battles", and if not, insert tuples into one or both of these relations, with NULL components where necessary.

Solution:

a)

```
CREATE TRIGGER TriggerA
AFTER INSERT ON Classes
REFERENCING
        NEW ROW AS NewRow
FOR EACH ROW
BEGIN
        INSERT INTO Ships (name, class, lunched)
                VALUES (NewRow.class, NewRow.class, NULL);
END;
```

b)

```
CREATE TRIGGER TriggerB
AFTER INSERT ON Outcomes
REFERENCING
        NEW ROW AS NewRow
FOR EACH ROW
WHEN (NewRow.battle NOT IN (SELECT name FROM Battles))
   INSERT INTO Battles (name, date)
        VALUES (NewRow.battle, NULL);
```

Question 4 (Data Integrity) [20 points]

Consider the following relational schema:

Emp(eid: integer, ename: string, age: integer, salary: real)
Works(eid: integer, did: integer, pct time: integer)
Dept(did: integer, budget: real, managerid: integer)

1. [5 points] Define a table constraint on Emp that will ensure that every employee makes at least $10,000. You should include the SQL statements of creation of the table.

```
CREATE TABLE Emp (eid INTEGER,
ename CHAR(10),
age INTEGER ,
salary REAL,
PRIMARY KEY (eid),
CHECK (salary >= 10000 ))
```

2. [5 points] Define a table constraint on Dept that will ensure that all managers have age > 30.

```
CREATE TABLE Dept (did INTEGER,
buget REAL,
managerid INTEGER ,
PRIMARY KEY (did),
FOREIGN KEY (managerid) REFERENCES Emp,
CHECK( (SELECT E.age FROM Emp E, Dept D)
WHERE E.eid = D.managerid ) > 30 )
```

3. [5 points] Define an assertion on Dept that will ensure that all managers have age > 30. Compare this assertion with the equivalent table constraint. Explain which is better.

```
CREATE TABLE Dept ( did INTEGER,
budget REAL,
managerid INTEGER ,
PRIMARY KEY (did))

CREATE ASSERTION managerAge
CHECK ((SELECT E.age
FROM Emp E, Dept D
WHERE E.eid = D.managerid ) > 30 )
```

Since the constraint involves two relations, it is better to define it as an assertion, independent of any one relation, rather than as a check condition on the Dept relation. The limitation of the latter approach is that the condition is checked only when the Dept relation is being updated. However, since age is an attribute of the Emp relation, it is possible to update the age of a manager which violates the constraint. So the former approach is better since it checks for potential violation of the assertion whenever one of the relations is updated.

4. [5 points] Write SQL statements to delete all information about employees whose salaries exceed that of the manager of one or more departments that they work in. Be sure to ensure that all the relevant integrity constraints are satisfied after your updates.

To write such statements, it is necessary to consider the constraints defined over the tables. We will assume the following:

```
CREATE TABLE Emp ( eid INTEGER,
ename CHAR(10),
age INTEGER,
salary REAL,
PRIMARY KEY (eid) )

CREATE TABLE Works ( eid INTEGER,
did INTEGER,
pcttime INTEGER,
PRIMARY KEY (eid, did),
FOREIGN KEY (did) REFERENCES Dept,
FOREIGN KEY (eid) REFERENCES Emp,
ON DELETE CASCADE)

CREATE TABLE Dept ( did INTEGER,
buget REAL,
managerid INTEGER ,
PRIMARY KEY (did),
FOREIGN KEY (managerid) REFERENCES Emp,
ON DELETE SET NULL)
```

Now, we can define statements to delete employees who make more than one of their managers:

```
DELETE
FROM Emp E
WHERE E.eid IN ( SELECT W.eid
FROM Work W, Emp E2, Dept D
WHERE W.did = D.did AND D.managerid = E2.eid AND E.salary > E2.salary)
```