COSC6340: Database Systems
Project: Discovering Functional Dependencies and Certifying Normal Forms with SQL Queries

Instructor: Carlos Ordonez

1 Introduction

You will develop a Java program that generates SQL code to check normal forms. The input is a set of tables (relations) and a potential "candidate key (CK)" for each table (it is initially unknown if the given CK is indeed a CK).

The output consist of: A summary table that states for each table if it complies with 3NF (Y/N), and a decomposition of the input table into multiple tables when it is not in 3NF.

The DBMS will run on Linux, available on a UH server. Your client Java program will run in Linux, too. The connection interface will be via the JDBC protocol. The schema of the tables will be specified as a text file, and the actual tables will be stored in the database. While your program must generate the output as text files, the processing will be done with SQL queries.

2 Program requirements

Your program will generate SQL code based on input tables whose basic schema is defined in a text file. Basically, you will initially have to check normal forms against a given candidate key and several nonkey attributes. You can assume the "candidate" key has at most 3 attributes and there are at most 17 nonkey (nonprime) attributes. The main goal of your program is to certify (verify) normal forms 3NF. Your program must answer yes/no for 3NF.

1. You must first check 1NF and 2NF.

   For 1NF, you must check duplicates and a clean key. There is no need to verify there are no atomic values since a table in the DBMS will be given as input. The given key may be composite. Therefore, every attribute must be clean. Keep in mind you are given a candidate key and the table itself already has another "fake" primary key. However, this "candidate key", proposed by the user, may have duplicates or nulls. In short, you are checking whether you have a potential new key.

   For 2NF check partial functional dependency (FD) on the candidate key. If there is a simple key given then, the table is in 2NF. If you have a composite key, you must analyze every subset of columns for partial FDs against every subset of the composite key. Rows with nulls should be skipped. You can assume the candidate key has at most 4 attributes. If thke validation fails, and the decomposition option is selected, you should create additional tables to propose the lossless decomposition in 2NF. This is accomplished by moving the defined attributes of the source table into a new table, and using the subset of the candidate key as the new primary key of this table.

2. For 3NF, you need to check transitive FDs on nonkey attributes. You must check up to 3 attributes on the left hand side of an FD.

2.1 Programming details

1. Input and output are text files. All information to be processed is in SQL database tables.

2. You need to use JDBC to submit queries, but your program should be able to generate an SQL script file for all the queries used in your program. The user must provide an input text file. If any of these parameters are missing, your program should return an error message and stop.
3. The input text file specifies the tables to be analyzed, and their schema. If you find and inconsistence between the schema and the actual table, you must report the problem an continue to the next table.
4. Always remember to drop any derived table in advance to avoid exceptions.

2.2 Theory

Remember a relation in 3NF (BCNF) is also in 2NF and 1NF and a relation in 2NF in also in 1NF. Therefore, if the program is only certifying, when a lower NF is not satisfied the program should not continue checking further NFs (since that would be unnecessary).

SQL code generation: Your program will get just one key without specifying other (secondary) candidate keys. You are required to check normal forms based on that CK only. You can use any combination of SELECT statements, temporary tables and views. You can name your tables/columns any way you want, and assume unique column names in the database. You are not allowed to modify the input table in any way.

3 Program call and output specification

Here is a specification of input/output. There is only one input parameter: the database schema file.

- syntax for call: "java CertifyNF database=dbxyz.txt".
- Example of input file (dbxyz.txt) below. Notice the "fake" primary key is not included in the table schemas.

```
Employee(employeeid(k),name,salary)
Department(deptid(k),subdeptid(k),deptname,deptSalesAmt)
```

- If the connection to the DBMS fails, you have to send an error message. If the connection to the DBMS is successful, you should generate the "NF.sql"script file regardless the selected option, and dump the SQL generated through your program.
- Input database/tables checking:

  Your program will be tested with several database schema text files. You must make elementary error checking for non-existent tablefs and invalid column names. The input tables have numbers or strings.
- The SQL of your generated script file should be properly formated and indented (one SELECT term per line, JOIN/WHERE/GROUP in different line).
- Use SQL SPJA queries with to certify each normal form.

  Decomposition is required for tables not in 3NF. When decomposing into m tables use an m-join query to verify decomposition is correct.
- Output:

  A summary result table, called "NF.txt", with 4 columns should be created. The first column should specify the table name, the second column should say if the table was or not in the 3NF, the third column should specify the what normal form has failed (when the table is not in 3NF), the following column should state the reason of why the normal form stated on the third column was violated. You should include all violations (define the column as char(200)).

  Write to a text file ("Decomposition.txt") the decomposition of each denormalized table as a list of tables, one per line (format specified below). Decomposition is not needed if table is in 3NF, of course.

  Write to a text file "NF.sql" all the SQL statements generated during the checking of normalization.
- Optional for extra credit (10%): Check if the input table is in BCNF. If not, report the problem.

4 Examples of input and output

Recall that all tables are given in the input database file. Here are some examples of input tables that must be checked for normal forms. The tables have a "fake" PK i in order to allow storage in the DBMS. Notice i is IGNORED in the normalization certification.

R

i	K1	K2	A	B
1	1	1	1	0
2	2	1	1	0
3	3	2	2	0
4	4	2	2	0
5	1	2	2	0
6	3	1	1	0

S

i	K1	A
1	10	2
2	20	1
3	30	2
4	40	2
5	50	1
6	60	2

T

i	K1	K2	A
1	10	1	2
2	10	2	1
3	20	1	2
4	30	1	3
5	30	2	1
6	40	1	2

4.1 Output: NF certification

The input is:

```
dbxyz.txt
--------------------------------------
R(K1(k),K2(k),A,B)
S(K1(k),A)
T(K1(k),K2(k),A)
```

The output is the summary tables and the list of decompositions. Note that the lines starting with # are optional comment lines.

```
#Table      3NF        Failed       Reason
R              N       2NF          K2->A,K2->B
S              Y
T              Y
```

Decomposition into normalized tables:

```
#R decomposition:
R1(K1,K2,A)
R2(A,B)
#Verification:
R=join(R1,R2)? YES
```

5 Suggested work load distribution

Since the teams are comprised of three students, each of the team members will be in charge of one of the following tasks:

- Designing of all the SQL queries necessary to verify 3NF. The student in charge of this task should write SQL queries and try them using scripts.
- Java Programming to generate SQL for 3NF verification
- Java Programming to generate decomposition, and its verification by a join.

6 Grading

The TA will provide the input file and the expected output for sample test cases. This test cases will give the students the opportunity of evaluate their own projects, and make corrections before the official submission. The TA will use scripts for automatic grading. Your project will be tested with 10-20 test cases, with various difficulty levels. After the automatic grading is ready, the curse instructor will meet every team, and will make questions to the team members, according with the responsibilities enumerated in the section 5 of this document.