COSC6340: Database Systems
Project: Certifying 3NF and BCNF Normal Forms in SQL

Instructor: Carlos Ordonez

1 Introduction

You will develop a Java program that generates SQL code to check 3NF and BCNF. As an advanced option your program can normalize a database. 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 DBMS will be HP Vertica running on Linux. The client Java program can run on Windows or Linux. The connection interface will be JDBC. All input and output of your program will be text files, but 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 4 attributes and there are at most 16 nonkey (nonprime) attributes. The main goal of your program is to certify (verify) normal forms 3NF and BCNF. Your program must answer yes/no for 3NF and BCNF.

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 valid 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 the 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 2 attributes on the left hand side of an FD.

3. For BCNF you should check partial FDs between 2 non-key attributes and one key attribute in CK.

2.1 Programming details

1. Input and output are text files. 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 and an evaluation option parameter (certification or decomposition). If any of these parameters are missing, your program should return an error message and stop.
3. 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, temp tables and views. You can name your tables/columns any way you want, and assume unique column names in the database. The only constraints that you have are not to modify the input table in any way and the output table should be called “NF”.

3 Program call and Output

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):

```
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” file regardless the selected option, and dump the SQL generated through your program.
- The SQL of your generated script should be properly formated and indented (one SELECT term per line, JOIN/WHERE/GROUP in different line).
- A summary result table, called ”NF.txt”, with 4 columns should be created in the DBMS. The first column should specify the table name, the second column should include the evaluated NF, the third column should say if the table was or not in the specified form, the following column should state the reason of why the form was violated. You should include all violations (define the column as char(200)). Call the output table “NF” and you can do a “SELECT * FROM NF;” as your last statement.
- Use SQL SPJA queries with to certify each normal form.
- Display the final NF certification getting the results with JDBC and displaying them on the screen.
- The output in a simple text file is the only requirement. Your program will be tested with scripts. Therefore, a GUI is discouraged and unnecessary.

4 Examples of input and SQL

Recall that all tables are given in the input database file. Here are some examples of input tables thast need to be checked for normal forms. The tables have an ”artificial” PK in order to allow storage in the DBMS. Notice i is NOT considered in the normalization certification, but it is used just to have a practical valid PK.

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/output is:

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

Table	Form	Y_N	Reason
R	3NF	N	not 2NF, $K2 \rightarrow A$, $K2 \rightarrow B$
R	BCNF	N	not 3NF
S	3NF	Y	
S	BCNF	Y	
T	3NF	Y	
T	BCNF	N	$A \rightarrow K2$