COSC6340: Database Systems
Project: Finding Candidate Keys and Foreign Keys in SQL

Instructor: Carlos Ordonez

1 Introduction

You will develop a Java program that generates SQL code to find candidate keys and foreign keys in a database.

You will use a DBMS supporting SQL. All keys are assumed to have one column. The database is simple a set of tables in SQL.

2 Program requirements

Your program will generate SQL code based on a list of tables given in a text file. Basically, you will have to compute projections π and joins $\bowtie$ between two tables in order to "guess" which columns may be candidate keys and foreign keys. Remember a foreign key K in T_1 is valid if all its values exist in some referenced table T_2 where K is a PK. (i.e. all T_1 rows with different K values match some row in T_2).

Relational queries will be evaluated with SQL. You have freedom to program your SQL queries (nested, derived tables, views, etc). Cursors are not necessary. If the size of the result table is non-empty (one or more rows) you will assume there is a foreign key between both tables (or ER relationship). Remember that joins are computed over columns (attributes) from the same data type (domains). Columns (attributes) may have, but are not required to have the same name. That is, the program cannot assume the a potential foreign column has the same name in two tables. On the other hand, if a column has the same name in two tables it could be a foreign key in one table, but it cannot be guaranteed. However, your program needs to assume that both columns have the same data type. For instance an integer column cannot be joined with a string column. Your program only needs to handle integers and strings (dates, float not required).

In most cases, there will be a foreign key referring to one table, but there may database instances where a foreign key may refer to two tables. In general, a foreign key will have the same name on both related tables, BUT that is not a requirement. That is, your program must still test every column, even if the column names are different. Also, data types must be the same in order to join two tables, but your program CANNOT assume a column with the same name on two tables has the same data type.

3 Program call, input and Output

The program basically takes a text file with the database schema and a threshold for the number Here is a specification of input/output. of rows in a successful join.

- syntax for call: **"java FindKeys database=db.txt;minjoinrows=integer|all"**.
- Notice parameters have a specific name (case insensitive).

- The **minjoinrows** threshold has default value 1 (if no value provided). Integer can be some minimum number of rows of the referencing table (for instance size of one table if known). ”all” means the referencing table rows match with no row excluded from the referencing table (i.e. the foreign key table). Zero is not acceptable (think why). If the parameter is ”all” then the foreign key column must not have nulls. However, there may be test cases with nulls in a column and thus your program can state that such column is not likely a FK.
- Notice parameters are not separated by spaces, but by ;.
- the output is another text file called ”foundKeys.txt”, which provides primary key as ”PK”, candidates keys labeled with ”CK” and foreign keys with ”FK”. You should label ”PK” the first candidate key. It is desirable tables appear in the same order as the input file.

3.1 Programming details

1. 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.
2. The program will be developed in Java; any version is OK. Your program will be compiled by the TA from the command line with ”javac”. You can use any GUI for development (e.g. Eclipse).
3. 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 TA will provide the IP address of the DBMS server. Your program will be tested with tables in this DBMS server.
4. You can include an optional GUI, but output in simple text form is the only requirement. You can get +1 if you include a nice/intuitive GUI (with a README file). Although, you can only get this extra credit if your program finds foreig keys correctly.
5. You can create temporary tables. Always remember to drop any derived table in advance.
6. This program is a single phase project. You can ask any questions by the Google newsgroup (preferably) or email (you must have a good reason not to use the Google newsgroup).

   You can also come to the TA or my office hours with specific programming questions (if the question is not posted or this homework description is unclear). In addition, we will have short disucssion on the homework every week.
7. Each table will have no more than 10 columns. The database will have no more than 20 tables.
8. Tables may be empty and input text files may be empty.
9. 8 test cases will be ”clean” databases. 2 test cases may contain nulls. There will be at most 2 test cases with input files with errors (table name starts with number, table with no columns, commas missing, etc). Your program should not crash under input files with errors.

4 Examples of input and output

Here are two examples of database schemas. Notice there is one table schema per line. Notice data type is not specified. 1st example of input file with integer columns:

```
T1(A,B,C)
T2(B,C,Z)
T3(C,D,E,F)
```

Example of output for 1st example (db-easy.txt):

```
T1(A(PK),B(FK=T2.B)),C(FK=T3.C))
T2(B(PK),C(FK=T3.C),Z(CK))
T3(C(PK),D,F(CK),E,F)
```

2nd Example of input file (db-store.txt):

```
Employee(employeeid,empname,empsalary,departmentid)
Department(deptid,deptname,deptSalesAmt,deptItems)
Product(SKU,productName,productPrice,deptid)
```

Output for store database:

```
Employee(employeeid(PK),empname(CK),empsalary,departmentid(FK=department.deptid)
Department(deptid(PK),deptname,deptSalesAmt,deptItems)
Product(SKU(PK),productName,productPrice,deptid(FK=DEPARTMENT.deptid))
```

5 Turn in project

One email with a zip file attached. The subject should clearly specify course and project #. Do not include class files or Java libraries; only your Java and SQL code. Include a readme file to compile/run (instructions, notes, errors if any) and small database example where your program runs. It is OK to inclue some saample SQL and output generated by your program. Do not include a large Java library in your zip file. Your program should work with the existing JDK (1.5, 1.6). Therefore, your zip file is expected to be **small**.

If you developed a GUI explain how to run it in the README file. The GUI is at most +10% and will count only if your program passes most test cases. Also, your program should work from the command-line without the GUI.

Your project must be sent by 10pm on the due date. No late submissions are accepted, unless there is a previous discussion and acceptable justification with the professor (the TA cannot make exceptions).

Example: email with subject "COSC6340: Project1Team01" (two digits for team number). The zip file should be called "Project1Team01.zip" and must contain all .java files and the README.TXT file.