DBSchemaScribe—Final Results

Shengye Wan

I. System Architecture

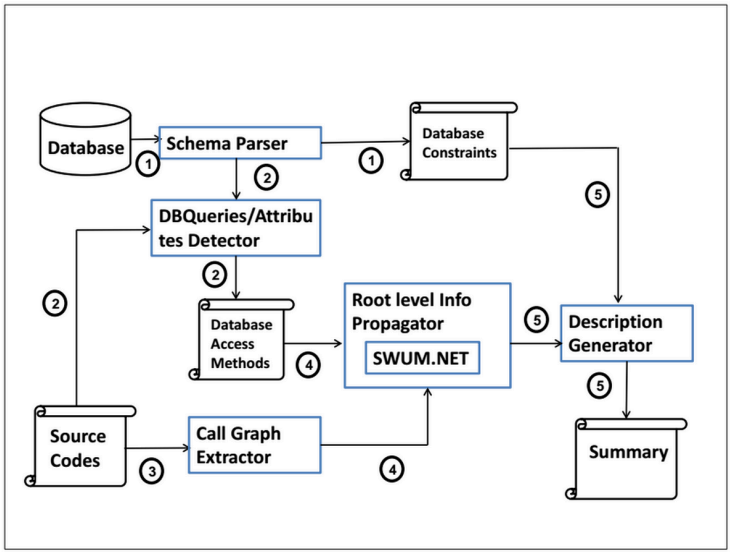

Figure 1: DBSchemaScribe System Architecture

II. Technical Details

1. Schema Parser

This component is developed based on namespace "**MySql.Data.MySqlClient**" which contains INFORMATION_SCHEMA TABLE of each schema in MySql database. These tables provide access to database metadata, information about the MySQL server such as the name of a database or table, the data type of a column, or access privileges.

2. DB Queries / Attributes Detector

This component is developed based on the only one open source solution I found. It's called "**Irony Kit**" (refer: https://gridwizard.wordpress.com/2014/11/08/looking-for-a-sql-parser-for-c-dotnet/). Here is an example:
For SQL query: **SELECT ID, Title FROM Shows WHERE ID=1**, It could generate a parsing tree looks like figure 2.

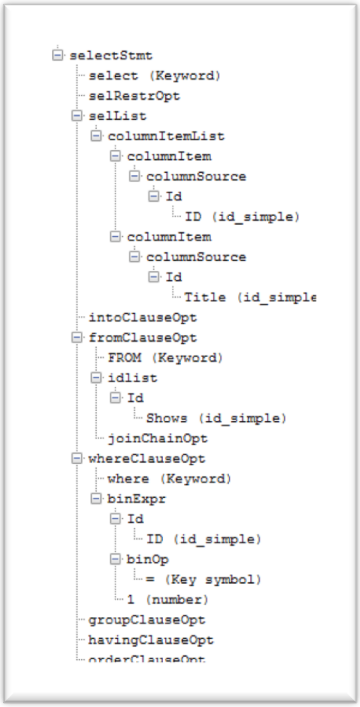

Figure 2: SQL Parsing Tree

3. Call Graph Extractor

This component is based on the namespace "**ABB.SrcML**". The classes of namespace could help us get detailed information of each methods in project source code. Then we could build the CallGraph based on the methods relationship of SrcML and find out what's the top-level methods of each method.
*Acknowledgement: The function about generating call graphs of methods is written by Boyang.

4. Root Level Info Propagator

This component is developed based on the namespace "**ABB.Swum**". The classes of this namespace could automatically generate natural language descriptions for each method.

5. Description Generator

This component is developed based on the the namespace "**Antlr3.ST**". Basically we could pass values from our application the the script of a webpage and automatically generate a page as report with defined data.

III. Code Comments

Program class: This class is the start point of the DBSchemaScribe. Every time we want to analyze a new project, we need:

1) Set up the database;
2) Set the myConnectionString with proper database login information and the database should be the database of target project;
3) set the LocalProj to point at the location of the project.

After these steps we could start the program to automatically analyze a project.

ExtractMethodSQL class: This class is the most important project of DBSchemaScribe. This class has several important jobs and basically it connects with all other classes. So I'll describe its function with other classes together:

1) It would rebuild all the sql statement in the source code by RebuildString method. Then ExtracMethodSQL class would parse all strings we get with sqlStmtParser class (the sql grammar are pre-defined in the sqlGrammar class). If we think this is a sql statement, then we could check what's the table names and columns names with sqlStmtParser class.

2) It would save all necessary information of each method in desMethod class. This class is wrapped with MethodDefinition class of ABB.SrcML.Data. We also saved the natural language description in desMethod class. The description is generated by SwumSummary class. Furthermore, we use call graph to get all the call paths of this method (This step is realized by CallGraph class, CGManager class and InvokeCallGraphGenerator class). Then we use attribution followmethods to save all the methods in the call graph paths but not the top-level methods. The top-level methods in call graph are most important because they are most possible to show how user interact with database so we save them alone in attribution finalmethods.

3) It would save all necessary information of database in dbTable class and dbColumn class. These two classes could represent the table and column. I use thes two classes to reflect the relationship between tables and columns and relationship between database and methods. Similary in these two classes we mark methods into three categories: Directly methods refer to the methods call the sql statement to operate this table/column directly; Follow methods represent methods in the call graphs of direct methods but not in the top level and final methods are top level methods in call graphs. We extract the information of table/column with the class dataSchemer.

4) Finally, It would generate a SingleSummary class for each table and column. Then we pass these summaries to FinalGenerator class and this class could generate the final HTML-based report.

IV. Report Evaluation

For the detecting issue, I checked all the methods of RiskIt found in DBScribe project and our project also found those methods. One interesting point of our reports is that actually many top-level methods are named like "MainClass.main", actually this method does not include too much information and it could be many final methods for different tables or columns. In this case, I think the second top-level methods might be more useful to report the relationship between database and human behavior.