Evaluation of Reports

I. The test projects

In the current GitHub version, I upload four new projects: taxonomy, SWTBus, SIGULAB and SpringJDBCkong for testing. Besides these four new projects, I keep two projects that DBScribe already tested in order to evaluate my program's ability for detecting SQL-related methods.

II. The discussion about my program's ability for detecting SQL related methods

For project "RiskIt", DBScribe detects 35 methods with SQL local invocations + 9 methods mixing local and delegated SQL invocations. In totally, there are 44 methods contain SQL local invocations.
In my program, we detect 44 methods have SQL statements which is the same number as DBScribe.

For project "xinco", DBScribe detects 26 methods with SQL local invocations + 21 methods mixing local and delegated SQL invocations. In totally, there are 47 methods contain SQL local invocations.
In my program, we detect 47 methods have SQL statements which is the same number as DBScribe.

Since other projects do not have the data to compare with, the conclusion for this ability is:
For the known projects, DBScribeSchema could detect as many SQL-related methods as DBScribe.

However, for the other projects, I could not guarantee the ability about detecting methods is same as DBScribe because the grammars of SQL statements are defined in the library of Irony. All I could do is modifying the original grammar to make it detect more methods. Now I have fixed all the cases I known by comparing the results between Irony and DBScribe. For some unknown statements might appear in future, maybe we still need to modify the grammar in Irony. However, the grammar is pretty easy to modify once you understood the code of Irony.
In the rest part, I would show some details about each projects and their reports.

III. The details of test projects

1. Taxonomy
Link: https://github.com/haint/taxonomy.
Database schema name: taxonomy.
The number of tables we covered: 10.
The percentage of tables we covered: 10/10 = 100%.

2. SWTBus
Link: https://github.com/gorannovotny/SWTBus.
Database schema name: swtbus.
The number of tables we covered: 13.
The percentage of tables we covered:13 / 13 = 100%.

3. SpringJDBCkong
Link: https://github.com/namratajavactp/spring_final_demos/tree/master/Spring_JDBC_kong.
Database schema name: citius_schema.
The number of tables we covered: 1.
The percentage of tables we covered: 1/1 = 100%.

4. SIGULAB
Link: https://github.com/esteoliver/SIGULAB.
Database schema name: modulo2.

The number of tables we covered: 6.
The percentage of tables we covered: 6/7 = 85.7%.
The table we missed: gestiona. There are two files in the projects are operating related to "gestiona" but I do not see there is any SQL statements in the files. Since there is no SQL statement shows in code about this table we could not show this table.

5. RiskIt

Link: http://www.cs.wm.edu/semeru/data/ICSE16-DBScribe/subjects/RiskIt.zip.
Database schema name: riskit.
The number of tables we covered: 11.
The percentage of tables we covered: 11 / 13 = 84.6%.
The table we missed: geo, stateabbv. I searched the code of the RiskIt but I do not see any code contains these two tables name. So our program is fine.

6. Xinco

Link: http://www.cs.wm.edu/semeru/data/ICSE16-DBScribe/subjects/xinco.zip.
Database schema name: xinco.
The number of tables we covered: 14.
The percentage of tables we covered: 14 /23 = 60.9%
The table we missed: nine tables that are named as "*_t". However, for these nine tables there are no code related to them in the project. These tables look like just the back up of the original table. For example, many sql statements access table "xinco_add_attribute" but there is no code about accessing table ""xinco_add_attribute_t". So our program is still fine.

As the conclusion:
For all six projects we test, for any table that is accessed by the SQL statements in code, we could successfully show them in the report.