Full predicate coverage for testing SQL database queries

Javier Tuya, María José Suárez-Cabal and Claudio de la Riva

Department of Computer Science, University of Oviedo, Campus of Viesques, s/n,
33204 Gijón, Spain
`{ tuya, cabal, claudio } @uniovi.es`

Abstract. In the field of database applications a considerable part of the business logic is implemented using a semi-declarative language: the Structured Query Language (SQL). Because of the different semantics of SQL compared to other procedural languages, the conventional coverage criteria for testing are not directly applicable. This paper presents a criterion specifically tailored for SQL queries (SQLFpc). It is based on Masking Modified Condition Decision Coverage (MCDC) or Full Predicate Coverage and takes into account a wide range of the syntax and semantics of SQL, including selection, joining, grouping, aggregations, subqueries, case expressions and null values. The criterion assesses the coverage of the test data in relation to the query that is executed and it is expressed as a set of rules that are automatically generated and efficiently evaluated against a test database. The use of the criterion is illustrated in a case study which includes complex queries.

Keywords: software testing; database testing, MCDC, Full Predicate Coverage, SQL

1 Summary

One of the major fields of study in coverage criteria for testing is related to coverage of the source code using different approaches (e.g. data-flow or control-flow). Control-flow criteria range from path or branch coverage to more sophisticated criteria to thoroughly assess the adequacy of tests according to their logical decisions. These criteria are a powerful tool to evaluate the adequacy of test suites and to assist in the development or completion of test cases. However, in many database applications the software under test interacts with the database in single interaction points where the program passes control to the DBMS that executes an SQL command. In this case, the usual control-flow based criteria can't be used to measure the adequacy of the SQL commands. This is the issue that is addressed in this paper[1]. The primary contributions are:

1) The development of a coverage criterion for SQL based on masking MCDC, considering the specific semantics of a wide range of the SQL syntax and database

[1] Javier Tuya, Mª Jose Suarez-Cabal, Claudio de la Riva. Full predicate coverage for testing SQL database queries. *Software Testing, Verification and Reliability*, 20(3) 237-288, September 2010.

schema constraints. The criterion identifies the requirements that have to be satisfied by the test data used by SELECT queries, including JOIN, WHERE, HAVING and GROUP BY clauses, aggregate functions, subqueries and case expressions. In addition, the criterion incorporates a three-valued logic in order to be able to handle missing information (null values).

2) A complete description of how the test requirements are expressed as a set of coverage rules that is obtained by applying successive transformations on the query under test. Transformations are defined for each of the different clauses that appear in SQL and their combinations. The coverage rules are executable (written in SQL) and then able to determine whether the different situations expressed by the test requirements are covered by the test data. Therefore, by running the coverage rules the adequacy of the test data can be measured and the situations that are not covered identified, enabling to complete the test data, leading to better quality tests.

3) The completely automated generation and evaluation of the coverage rules, which are implemented in a set of tools (SQLFpc: http://in2test.lsi.uniovi.es/sqlfpc/ and SQLRules: http://in2test.lsi.uniovi.es/sqltools/sqlrules/). A web service is also provided to enable their integration with third party applications. As the rules are written as SQL queries they take advantage of all performance improvements implemented in commercial DBMS. Therefore, the evaluation of the coverage is very efficient, even for large databases and complex queries.

4) The application of the coverage criterion to a case study including complex queries taken from Compiere, which is an open source Enterprise Resource Planning (ERP) application. In total, there are 107 queries. The largest queries have up to 24 conditions in the WHERE clause, joins of up to 15 different tables and up to 19 case expressions. Some queries are composed of the union of many queries: the largest ones include 15 queries, and 5 queries with joins over 22 tables. The set of queries uses a total of 136 different tables from the Compiere database schema. The tables have an average of 23 columns, the largest one with 84 columns.

The SQLFpc coverage is evaluated using random databases of different sizes. The generation time of the full set of rules for all the 107 queries is 45.9 seconds, and the evaluation time ranges from 128.5 seconds (4 rows per table) to 170.5 seconds (one thousand rows per table). The coverage is also compared against the mutation score, showing that as the size of the database increases, both the coverage and mutation score increase. Then the coverage criterion is used to assist the development of a test database using eight of the most complex queries. Starting from a blank database, rows are added in order to satisfy each of the coverage rules. The trend shown by the coverage is nearly linear with the increasing of the size of the database beginning near zero for the first step. The mutation score trend is similar, but beginning at a high value.

These results show a good scalability on the database size that allows an interactive evaluation of the coverage even with large databases and queries. Therefore, the criterion may be used both to assess the adequacy of a given test database and to assist the tester in developing new and more complete test data.

Acknowledgements: TIN2007-67843-C06-01, CN-07-168, PAC08-121-1374.