IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 6, August 2014.
www.ijiset.com
ISSN 2348 – 7968

Testing of Databases

Taranpreet Kaur[1], Bhupinder Singh[2]

[1](er.taran88@gmail.com), [2](er.batth1988@gmail.com)

ABSTRACT

Database testing involves the tests to check the exact values which have been retrieved from the database by the web or desktop application. Data should be matched correctly as per the records are stored in the database. Database testing is one of the major testing which requires tester to expertise in checking tables, writing queries and procedures. Testing can be performed in web application or desktop and database can be used in the application like SQL or Oracle. There are many projects like banking, finance, health insurance which requires extensive database testing.

Keywords: software testing; database systems; test adequacy criteria

INTRODUCTION

First of all, tester should make sure that he understands all the application totally and which database is being used with the testing application.

Figure out all the tables which exist for the application and try to write all the database queries for the tables to execute since there are many things which are really complex, so you can take the assistance of developers and figure out the queries. Test each and every table carefully for the data added. This is the best process for the testers to perform the DB testing, it can be done for any application and it does not matter application is small or big.

If things are really complex then tester can obtain the query from the developer to test the appropriate functionality.

OVERVIEW

Considering the widespread use of database systems there has been relatively little research into their testing. The work that has been produced differs by a number of factors, not least in the terminology that is used. In order to provide consistency in this paper we use the following terminology:

Application : a software program designed to fulfil some specific requirement. For example, we might have separate application programs to handle the entry of a new customer into the database, and to cancel dormant accounts once a time-limit has passed.

Database : a collection of interrelated data, structured according to a schema, that serves one or more applications.

Database application : an application that accesses one or more databases. A database application will operate on both program and database state. Database system : a logical collection of databases and associated (database) applications.

Testing is more difficult (or, at least, different) when dealing with database applications. The full behaviour of a database application program is described in terms of the manipulation of two very

ISSN 2348 – 7968

different kinds of state: the program state and the database state. It is not enough to search for faults in program state, we must also generate tests that seek for faults that manifest themselves in the database state and in the interaction between the two forms of state. A further complication for testing is that the effects of changes to the database state may persist beyond the execution of the program that makes them, and may thus affect the behaviour of other programs. Thus, it is not possible to test database programs in isolation, as is done traditionally in testing research. For example, a fault may be inserted into the database by one program but then propagate to the output of a completely different program. Hence, we must create sequences of tests that search for faults in the interactions between programs. This issue has not yet been considered by the testing research community. This has been shown to be particularly important for regression testing where the change to the functionality of one program may adversely effect other programs via the database state

Database Testing

Divisions of Database Testing:

- **Data Integrity test**
- **Stored Procedure / Functions test**
- **Type test**
- **Data Size Test**
- **Event Driven Actions Test**
- **Transaction Concurrency Test**

Data Integrity test

To preserve correctness of data, the database imposes various integrity constraints. These ensure the Correctness and Completeness of data and Validity of individual items. Once a value undergoes the Update or Deletion or Insertion or Time bound actions the database should be verified for the changes performed on related entities i.e., Foreign key / Primary key and all dependent entities.

Stored Procedure/Functions Test

Every Stored Procedure is to be tested separately for its functionality. Stored procedures need to be broken up into Action Items based on functions it perform and then each action item needs to be tested separately as the results of complete stored procedure.

The functions needs to be tested for it's return value as well as for the functionality. This testing is a form of database white box testing which helps in validating the modularity and functionality of code by writing another set of procedure and functions.

Procedures Vs Functions	
Procedure	**Function**
• Execute as a PL/SQL statement • Do not contain RETURN clause in the header • Can return none, one, or many values • Can contain a RETURN statement	• Invoke as part of an expression • Must contain a RETURN clause in the header • Must return a single value • Must contain at least one RETURN statement

ISSN 2348 – 7968

Type Test

Type test is performed to verify that the data types suggested by the DBA are same as expected by agreed upon by the developer. The data type mismatches mostly do not effect the functionality and normal execution of code, but prove to be very costly at the following scenarios:

- Updating the data
- Database performance
- Integration of the existing module with other module.
- Modify one routine to affect multiple applications.

Data Size Test

Data size testing should be done at back end to ensure smooth transition while appending functionality and integrating modules as during these phases, the data is passed to the system with direct user interaction and bypassing front end validation.

Event Driven Actions Test

Test Items :-

- Triggers
- Triggering Conditions
- Scheduled Actions
- Database Integrity after Triggers execution.

Transaction Concurrency Test

Transaction is a logical unit of program execution that takes a database from one consistent state to another consistent state. The database transactions should be checked for two or more users accessing a database concurrently. It should be verified to ensure that appropriate Concurrency techniques like Locking or Time stamping are in place to make every transaction follow each other.

Test Cases for Database Testing

Points to consider for writing test cases for Database Testing

- The no. of arguments being passed
- The data type of each of the arguments being passed
- The order of the arguments being passed
- The return value
- The data type of the return value
- Testing each and every Trigger, Cursor, Stored Procedure used in the application
- Testing the application w.r.t functionality of the application
- Join used between the back end tables.
- DB testing should be done both from the front end and back end.

Types of Testing

Black Box Testing in Database Testing

Black box testing involves testing interfaces and the integration of the database, which includes:

- Mapping of data including meta data
- Verifying incoming data
- Verifying outgoing data from query functions
- Various techniques such as Equivalence partitioning and boundary value analysis.

With the help of these techniques, the functionality of the database can be tested thoroughly.

ISSN 2348 – 7968

White Box Testing in Database Testing

White box testing mainly deals with the internal structure of the database. The specification details are hidden from the user.

- It involves the testing of database triggers and logical views which are going to support database refactoring.
- It performs module testing of database functions, triggers, views, SQL queries etc.
- It validates database tables, data models, database schema etc.
- It checks rules of Referential integrity
- It selects default table values to check on database consistency.
- The techniques used in white box testing are condition coverage, decision coverage, statement coverage, cyclomatic complexity.

From the survey called The Current State of Data Management Survey intiated between developers, IT management and data professionals it can be concluded that 95.7% of respondents believe that data is a corporate asset, but only 40.3% had a database test suite to validate the data and of those without a test suite only 31.6% had even discussed the concept. 63.7% of respondents indicated that they implemented mission-critical functionality in the database, but only 46% had a regression tests in place to validate the logic of this functionality. Currently, there are tools for database refactoring, testing data consistency, stored procedures, triggers, data validity. Testing is more difficult (or, at least, different) when dealing with database applications. The full behavior of a database application program is described in terms of the manipulation of two very different kinds of state: the program state and the database state. It is not enough to search for faults in program state, we must also generate tests that seek for faults that manifest themselves in the database state and in the interaction between the two forms of state. A further complication for testing is that the effects of changes to the database state may persist beyond the execution of the program that makes them, and may thus affect the behaviour of other programs. Thus, it is not possible to test database programs in isolation, as is done traditionally in testing research. For example, a fault may be inserted into the database by one program but then propagate to the output of a completely different program. Hence, we must create sequences of tests that search for faults in the interactions between programs. This issue has not yet been considered by the testing research community. This has been shown to be particularly important for regression testing where the change to the functionality of one program may adversely effect other programs via the database state. The literature on testing database systems varies in a number of ways. A fundamental difference in the literature is in the understanding as to exactly what a database system is. Each definition is constrained to a particular situation. There is no definition general enough to be applied to the different scenarios in which database systems may be used. The simplest view is when a single application interacts with a single database. This has been moderately extended to handle the situation in which multiple databases exist. Whilst the situation in which multiple applications interact with a database has been considered in a constrained form, there does not exist a generalised definition that is applicable to both this situation and the previous ones.

ISSN 2348 – 7968

Test adequacy criteria for database systems

A test suite is a collection of test cases (or test sequences in our approach) usually targeted towards the verification of the entire system or a specific section of the system. The manner in which a test suite is generated varies between different situations. The simplest is to randomly generate tests for the system. However, it is common to use more formalized techniques based on some aspect of the system, including: the systems specification, observations of the system being used, and the structure of the systems source code. To test every possible input is impractical for anything but the simplest of programs. For database systems this becomes impossible. Thus, if we cannot completely test a system, what is sufficient testing? Current work into software testing has proposed a number of test adequacy criteria that if satisfied will sufficiently test the system according to some characteristic of the specification or implementation. In terms of database testing only one set of criteria exist for determining the quality of a test suite. This approach is based on determining for each database operation an extensional representation of the portion of the database defined or referenced by the application. This approach is limited for a number of reasons: (a) It generates an extensional representation in which non–determinism countered by using a fixed initial state. However, this means that the test suite can only be considered valid for this particular state. (b) It does not consider the effect transaction operators have on the definition–use pairs and so will include unnecessary test cases for interactions that will not occur. (c) Nor does it consider multiple applications or instances of the same application accessing a single database. The only other work on test adequacy for database 5 systems is by Su´arez-Cabal and Tuya in which they present a metric for testing coverage of an SQL SELECT query, and a method for detecting when tuples needed to be added to the database to ensure better coverage, based on analysis of the corresponding query tree1. Their work aims to determine adequacy of a single query (specifically the SELECT query) and does not consider the behaviour of the system as a whole.

In this section we present a number of test adequacy criteria based on our intensional specification of database state and intensional descriptions of the behaviour of database operations. The criteria presented, are focussed on the structural and data-oriented elements of database systems. Briefly, structural elements include branches, loops and procedure calls. Data-oriented elements are the points at which data is defined and used in the program. However, first we present a brief discussion about the types of faults that can occur within a database system.

From a simple perspective this can be seen as determining if the output from a database system matches its required output. Bearing in mind that database state is persistent, it can be observed that the output database state is dependent not only on the input to the program but also the initial (or input) database state. Therefore, a test case execution can be seen as moving the database from one state to another. It is this transition that we aim to verify. The first type of fault is simply that the implemented functionality does not match the specified functionality. Other faults include: attempting to access a database entity that does not exist, operations attempting to violate the databases constraints (such as primary key or referential integrity), and transactions being aborted or

ISSN 2348 – 7968

committed incorrectly. These types of faults manifest themselves either in the database state or as a result of an interaction between program and database state.

Conclusions and future work

This paper is just a review of tesing in databases. But as we found, there is not good tool for testing. After this, I would like to design a good testing tool for databases which may save time, money and data redundancy.

Tool for Data Base Testing Using:

Swing is the primary Java GUI widget kit. It is a part of Oracle's Java Foundation Classes (JFC)—an API for providing a graphical user interface (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check box and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables and lists. Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent.The term lightweight is used to describe such an element.

MySQL ("My S-Q-L", officially, but also called "My Sequel") is the world's most used open source relational database management system (RDBMS) as of 2008 that runs as a server providing multi-user access to a number of databases. It is named after co-founder Michael Widenius' daughter, My. The SQL phrase stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used lamp open source web application software stack (and other 'AMP' stacks) LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python." Free-software-open source projects that require a full-featured database management system often use MySQL.

For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases. Applications which uses MYSQL databases, Includes: TYPO3,Joomla, WordPress, phpBB, MyB B, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google (though not for searches), Facebook,Twitter, Flickr, Nokia.com, and YouTube

REFERENCES

1. Haller, K. (2009), "White-Box testing for Database-driven applications: A Requirement Analysis", Zurich, Switzerland.
2. Willmor, D. and Embury, M. ,S. (2005), "Exploring test adequacy for database

ISSN 2348 – 7968

system", Informatics Process Group, School of Computer Science.

3. Walek, B. and Klimes, C. (2012), "A tool for database testing and optimization", International Journal of Computer and Communication Engineering, Vol. 1, No. 3, September 2012.
4. D. Chays, S. Dan, P. G. Frankl, F. I. Vokolos, and E. J. Weber. A framework for testing database applications. In Proceedings of the International Symposium on Software Testing and Analysis (ISSTA), pages 147–157, August 2000.
5. D. Chays, Y. Deng, P. G. Frankl, S. Dan, F. I. Vokolos, and E. J.Weyuker. An AGENDA for testing relational database applications. Software Testing, Verification and Reliability, 14(1):17–44, 2004.
6. E. F. Codd. A relational model of data for large shared data banks. Communications of the ACM (CACM), 13(6):377–387, 1970.
7. T. Connolly and C. Begg. Database Systems. Addison-Wesley, 3 edition, 2002.
8. Y. Deng and D. Chays. Testing Database Transactions with AGENDA. In Proceedings of the 27th International Conference on Software Engineering (ICSE). IEEE Computer Society, May 2005.
9. Y. Deng, P. G. Frankl, and Z. Chen. Testing database transaction concurrency. In Proceedings of the International Conference on Automated Software Engineering (ASE), pages 184–195. IEEE Computer Society, October 2003.
10. P. G. Frankl and E. J. Weyuker. An applicable family of data flow testing criteria. IEEE Transactions on Software Engineering, 14(10):1483–1498, 1988.
11. M. J. Harrold, J. A. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, and A. Gujarathi. Regression test selection for java software. In Proceedings of the 16th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), pages 312–326. ACM, October 2001.
12. W. E. Howden. Methodology for the generation of program test data. IEEE Transactions on Computers, 24(5):554–560, 1975.