

# Implementation of a Java based Software Platform for the Evaluation and Performance Analysis of Open Source Relational Databases

PT Sibanda\*

Jim Chadwick

[psibanda@ufh.ac.za](mailto:psibanda@ufh.ac.za)

[jchadwick@ufh.ac.za](mailto:jchadwick@ufh.ac.za)

Department of Computer Science, University of Fort Hare, Alice 5700

\*Primary Author for correspondence: TEL/FAX 0406022464

Conference Topic: Software    Sub Topics: Software design, Java, electronic commerce

**Abstract** - The number of open source database management systems (DBMS) currently available leave a potential user with the difficult decision of choosing the appropriate system that suits his particular requirements. This paper describes a software platform that allows a potential open source database management system user to carry out a performance evaluation on various open source relational database management systems, with a view of determining the system that best suits his needs. The platform enables the user to specify his own benchmarks, based on his own requirements, or to use established benchmarks for the performance evaluation.

**Index Terms** - Relational Database Management System, Software Platform, Performance Evaluation

## I. INTRODUCTION

Organisations have always had the need to manage data effectively and efficiently. The introduction of database systems has gone a long way in enabling organisations to achieve the above mentioned goal. Commercial database products, while well established and of high quality, are expensive, and not suited to applications in which cost is a major factor. An alternative is Open Source database systems particularly when these products have become established, and are well supported through the use of user newsgroups, contracted organisations and individuals, as well as regular updates. Small commercial enterprises, which lack the funds to install commercial databases, can make use of Open Source database products as an alternative. This has led to the increase in the use of Open Source software, including databases, in many areas, including electronic commerce [1].

However, the number of Open Source database management systems and engines available today leave a data administrator with the difficult task of selecting the appropriate system. A solution would be to subject each database to a set of well defined queries in order to assess the performance of each system. Such a set of queries is referred to as a benchmark. There is wide variety of benchmarks which one can use for the performance

assessment of database systems. Considering the number of database performance assessment benchmarks available, it is quite a daunting task to subject each database system under consideration to all the available benchmark tests and tools and then make a comparison of the results. [2, 3, 4, 5, 6]. Secondly each vendor usually publishes test results of their system using a particular benchmark or benchmarks. However there is always the danger that a vendor will select those benchmarks that reflect well on the product. [7].

In many applications of databases, generic benchmarking results are not ideal, as the operations carried out in practice may be quite different from those used to implement the benchmark. The user may rather wish to compare the performance of various databases using operations that are closely related to the particular application that the user has in mind. In effect, the user may wish to create his own benchmarks, or at least investigate the performance of the database on selected operations closely related to the application at hand. It will clearly be useful to the user if this task is automated as much as possible.

This paper will describe a software platform that achieves the following

- Automate the performance evaluation and comparison of relational database systems
- Allow the user to specify his own benchmarks, based on SQL commands
- Permit the user to use already available generic benchmarks, if this is preferred
- Produce performance and comparison results in graphical format
- Allow for new database systems to be easily added to the system as these become available.

## II. RELATED WORK

Extensive research has been conducted on database performance assessment. Kerstern and Kwakkel mention how it is not possible to apply a single benchmark for all database application areas [5]. The Wisconsin [2] and AS3AP [5] benchmarks mainly test the internal components of a system implementation. The TCP A and TCP B benchmarks are based on a real life situation (a bank teller

network) which renders them inapplicable to other (but not all) application areas [4]. Hence the need for a tool that allows the user to specify their benchmarks specific to the intended database application.

Serious arguments have also been raised on which benchmark is better than the other and researchers are constantly creating or modifying benchmarks [2, 5, 7, 8, 9]. Most benchmarks are presented as systems which can automatically run a test suite on various databases. PolePosition[8] is an open source Java framework for performance testing of databases. This framework was created to allow database administrators to evaluate the performance of databases according to particular needs without having to create a new ad-hoc benchmark specific to the administrator's requirements. PolePosition aims to simplify three tasks that a database evaluator may have to accomplish, namely: building the tests, adding database drivers, and reporting results [10].

Database performance is analysed by measuring the time it takes the system to carry out a selection of transactions. A database system transaction consists of one or more data manipulation statements and queries. An example of a transaction is querying a database system with a SQL statement.

The software platform described here does not seek to clarify or reinvent database performance analysis benchmarks. Rather it seeks as much as possible to automate the process of database systems performance analysis, while at the same time giving the user flexibility in choosing the transactions that are to be used in the evaluation.

### III. THE SOFTWARE PLATFORM

#### A. Database Products Used for Testing

The software platform is implemented in Java. Java enables simple connection to most database systems through the Java Database Connectivity (JDBC) API. The JDBC API is the industry standard for database independent connectivity between Java and a large number of databases [11]. For this project, we selected a number of database products that can be accessed using Java via JDBC. The Open Source database systems chosen for the pilot testing are MySQL [12], PostgreSQL [13], and HSQLDB [14]. These systems were largely chosen because they are some of the most popular Open Source database systems and they are accessible using Java. For comparison purposes, we have also included the well known Microsoft proprietary MSAccess database [15]. To enable connection using JDBC, the JDBC connector for each database system was acquired for free as a download. It is normally a .jar file that can simply be included in the CLASSPATH. A Java .jar file is an archive which contains various files applicable to a Java application. Table 1 below shows some of the attributes of the database products used.

Database Products	Max database size	Max table size	Max row size	Max columns per row
MySQL Server 5.0	No Limit	64TB	8KB	1000
HSQLDB 1.8.0.7	No Limit (Depends on Memory)	8GB	No Limit (Depends on Memory)	No Limit (Depends on Memory)
PostgreSQL 8.2	No Limit	32TB	1.6TB	1600
Microsoft Access 2007	2GB	2GB	16MB	250 - 1600

Table 1 Attributes of Database Products [16]

#### B. Methodology

The Extreme Programming (XP) software development methodology was used to develop this software platform. It focuses on developing a system iteratively according to the user needs. The initial phase would be to define user stories. A user story describes what the user wants the system to achieve - for example the user would like to input SQL statements into the system.

For this particular system the user problems are defined as follows.

- Create database
  - The system should enable the user to create databases to be tested
  - The database tables can have predetermined sizes according to benchmarks or the user can specify the sizes of the tables.
- Input SQL Statements
  - The system should allow the user to input SQL statements which will be executed on one or more database systems, as selected by the user
- Time transactions
  - All transactions submitted to database systems must be accurately timed by the system
  - The times recorded must displayed to the user in manner which is easy to understand e.g. graphically
- Add database systems
  - The user should be able to easily add any Java compatible database product to the system
- Compare database systems
  - The system should be able to display a comparison of the various database systems
- Use industry standard benchmarks to evaluate database performance
  - The user should be able to evaluate databases automatically using industry standard benchmarks

#### C. Database Connections

One of the most important things this system achieves is connection to multiple independent database systems. Fortunately, the code that executes a transaction does not

change at all from one database to the next. Only the code that connects to the database needs to be varied and this requires writing a separate method for each database that will return a Connection to that database. An example of such a method, which we wrote and implemented for the HSQLDB database is as follows

```
public static Connection getCon(String dbname){
    Connection con=null;
    try {        if(!isRunning())startServer();
                Class.forName("org.hsqldb.jdbcDriver").newInstance();
                //Creating the connection using jdbc
                con=DriverManager.getConnection(
                    "jdbc:hsqldb:hsqldb://localhost/hsqldb","sa","");
            }
        catch(SQLException e){
            startServer();
        }
        return(con);
    }
}
```

Fig. 1: Connecting to a database

Obtaining a connection to a database using the above method is done through the following line of code `Hsqldb.getCon("hsqldb")`. All that is required to add a new database is a short piece of code similar to the above.

#### D. The User Interface

This feature makes it possible for the user to specify various elements of the databases to be tested, namely the number of tables, as well as the number of rows and columns in each table. This is done using the interface shown below.

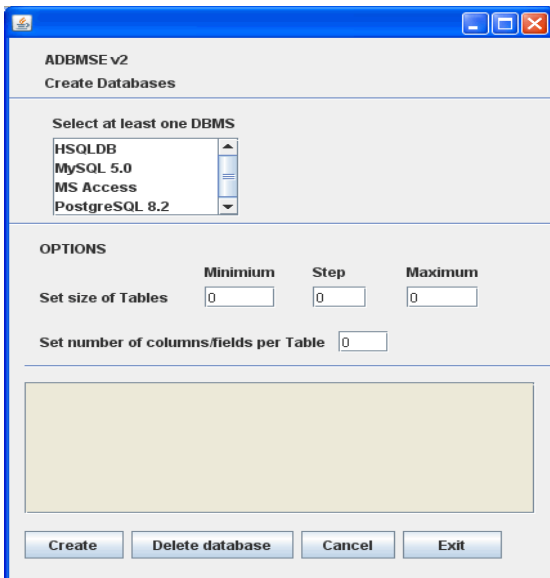


Fig. 2: Create Databases Interface

It is assumed that the user will want to know how transaction timings vary with the size of the database table. The interface allows the user to input the minimum size of the table, the maximum size, and the step. The step is the difference in the number of rows between two adjacent tables. These three numbers enable the system to automatically calculate the number of tables required for

each database. This interface also allows users to create multiple databases at the same time if they wish. The created databases are populated with a random integer in each field. The interface also allows the user to delete the databases after use at the click of a button.

#### E. Performance Evaluation

This is the component of the system that actually allows the user to input test measures to the databases. Here the user is provided with two main options. The first is to choose a generic benchmark incorporated into the system to evaluate the user's choice of databases. With this option the system will automatically create the databases and perform the required tests according to the benchmark's specifications. The second option allows the user to specify their tests through querying the database. This is achieved through allowing the user to enter text SQL query statements.

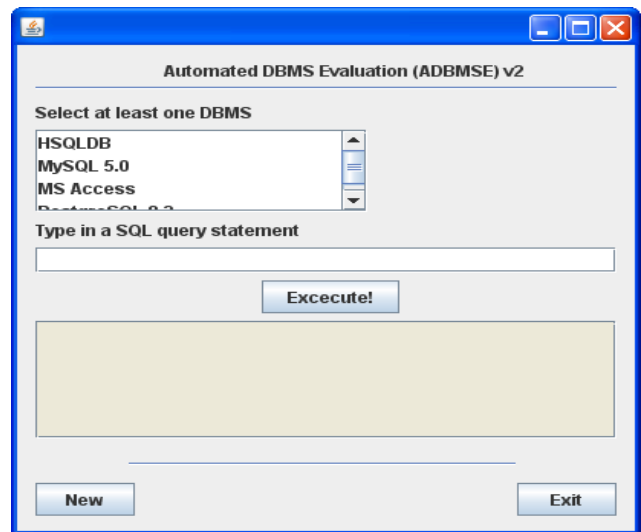


Fig. 3: Submitting and executing a SQL command

The main queries catered for by this component of the system are data manipulation queries. An example of is a *SELECT* query statement. The user can perform as many query statements as required and the system times the duration of this procedure. However a SQL statement requires less than a millisecond to be completed. To obtain meaningful timings, the query is executed one hundred times, as indicated in the code below.

```
if(!con.isClosed()){
    Statement stmt=con.createStatement();
    int k = 0;
    start = System.currentTimeMillis();
    while(k<100){
        stmt.execute(sqlstat);
        k++;
    }
    end = System.currentTimeMillis();
}
```

Fig. 4: Timing a SQL Query Statement

The snippet of code above also clearly indicates that the system specifically measures only the time required to actually execute a query one hundred times. However, the method `System.currentTimeMillis()` utilized in this code has

a granularity dependant on the operating system. On most operating systems the value of time is measured in units of ten milliseconds. The timings are saved to a file which is later subjected to further processing.

#### F. Graphical Output

The platform is designed to require minimal activity on the part of the user. As soon as the SQL command is entered, timings are obtained and sent to a file. This file is then automatically accessed by a built-in graphics package that immediately produces a graphical comparison of performance, in a format that is easily comprehensible. The output files are retained, in case the user needs to perform some more detailed analysis. Immediately after typing the SQL command, and executing it, the user is presented with a graphical comparison of how the selected databases perform on that transaction.

#### IV. SAMPLE OUTPUT

In order to test the platform, we used it to compare the performance of the selected products when subjected to a number of SQL queries. The testing and timings were performed on a Pentium 4 computer with a clock speed of 3192 MHz and 1 Gigabyte of random access memory. The databases used where also stored on the same computer.

Since the system must first create the database tables for the evaluation, it costs nothing extra to time this operation as well.

The following graph shows the time taken to create a database in each database system.. In this graph, the X-Axis is the number of tables in the database. There is very minimal difference between HSQLDB, PostgreSQL and MS Access. However MySQL takes the longest time to complete this process.

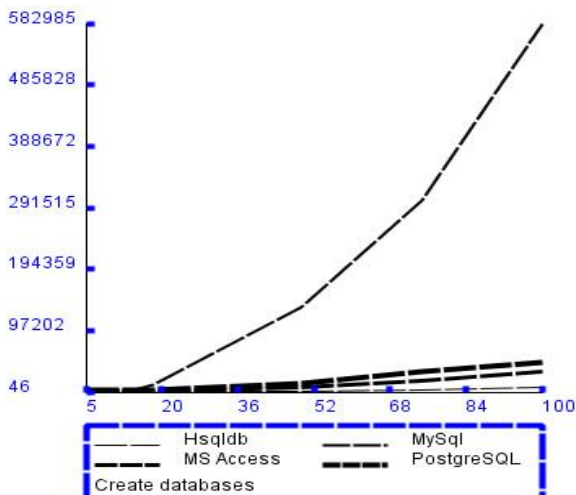


Fig. 5 Create databases

Fig. 6, 7 and 8 are graphs showing the results of timing some simple data manipulation queries. In these graphs the X-Axis represents the number of rows in a table.

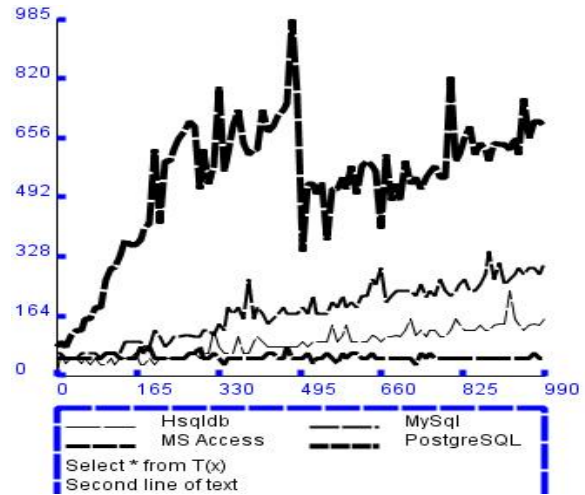


Fig. 6: SELECT \* FROM t(x)

Fig. 6 shows the results of timing a query that selects all the data in each table. In this case, MS Access is the most efficient, and PostgreSQL gives the worst performance.

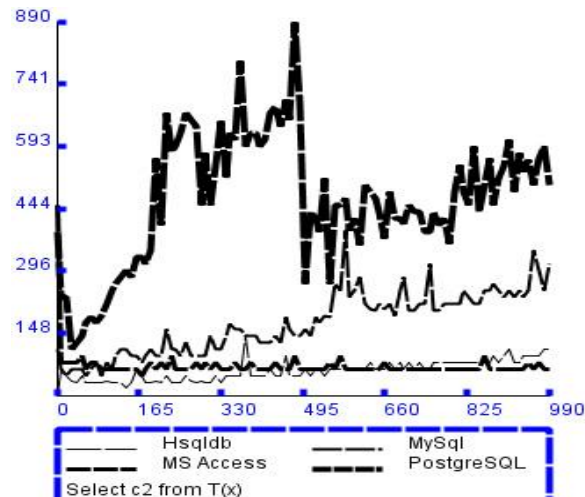


Fig. 7: SELECT column FROM t(x)

Fig. 7 shows results of timing a query that selects all the data in the second column of each table. As one might expect, the results differ little from those in Fig 4.

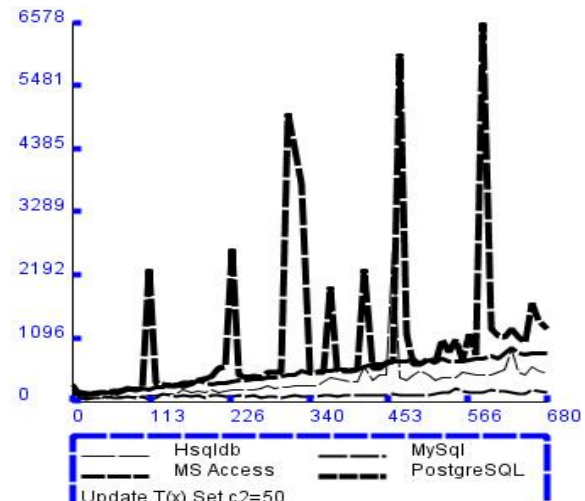


Fig. 8: UPDATE table SET column = 20

Fig. 8 shows the results of timing an update query. This query updates all the data in the second column of each table to a different value. From the above graphs one can clearly see that, once again, PostgreSQL has the worst performance.

## V. FUTURE WORK

The system so far has some limitations. When the user is creating databases to test with their own custom benchmark the databases are populated with a random integer in each field. However the type of data stored in a database affects query performance. Therefore there is need to include other types of data for these databases. There is also need to allow users to execute scripts or stored procedures to enable execution of multiple statements on the databases to be evaluated. The system has only been implemented on the Windows XP operating system, it can also be tested on other operating system since Java can run on most operating systems.

## VI. SUMMARY

The software platform described in this paper provides a user friendly approach to the evaluation and comparison of database products that can be accessed using Java. It allows the user to design their own benchmarks, using SQL transactions that may be tailored to the application that the user has in mind. The databases selected by the user are evaluated and compared with the results being displayed in a graphical format that is easy to interpret. It is envisaged that this platform would be particularly useful to those who may be considering the use of Open Source database products in commercial applications, especially electronic commerce.

## VII. REFERENCES

- [1] Joe McKendrick, *Open Source in the Enterprise: New Software Disrupts the Technology Stack*, September 2007, Available from [http://www.ioug.org/IOUG\\_Open\\_Source\\_07.pdf](http://www.ioug.org/IOUG_Open_Source_07.pdf)
- [2] David J. DeWitt, *The Wisconsin Benchmark: Past, Present, and Future*, 1993, *The Benchmark Handbook for Database and Transaction Systems* (2<sup>nd</sup> Edition), Jim Gray ed., Morgan Kaufman.
- [3] Brian L, *The TPC-C benchmark – What does it really mean*, 1997, Sunworld, Web Publishing. Retrieved April 07 2008 from <http://sunsite.uakom.sk/sunworldonline/swol-08-1997/swol-08-database.html>
- [4] The Transaction Processing Council(TPC) website. <http://www.tpc.org>
- [5] M. L. Kersten, F. Kwakkel, *Design and Implementation of a DBMS Performance Assessment Tool*, Computer Science/Department of Algorithmics and Architecture Report CS-9270 December, Stichting Mathematisch Centrum.
- [6] Vuk Ercegovic, David J. DeWitt, Raghu Ramakrishnan, 2005, *The TEXTURE Benchmark Measuring Performance of Text Queries on a relational DBMS*, Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.

- [7] Burleson Consulting, *Database benchmark wars: What you need to know*, 2002 updated 2006, Burleson Enterprises, Retrieved 07 March 2008 from [http://www.dba-oracle.com/art\\_db\\_benchmark.htm](http://www.dba-oracle.com/art_db_benchmark.htm)
- [8] The PolePosition benchmark website. <http://www.polepos.org>
- [9] The Open Source Database Benchmark website <http://osdb.sourceforge.net/>
- [10] Rick Grehan, *An open source database benchmark*, Sun Microsystems, Inc. Retrieved April 07 from <http://today.java.net/pub/a/today/2005/06/14/poleposition.html>
- [11] Sun Microsystems Java website <http://java.sun.com/javase/technologies/database/>
- [12] MySQL Database Management System <http://www.mysql.com>
- [13] PostgreSQL Database Management System <http://www.postgresql.org>
- [14] HSQLDB Database Engine <http://www.hsqldb.org>
- [15] Microsoft MSAccess Database Management System <http://office.microsoft.com/access>
- [16] David Leung, *Open Source database Comparison*, September 2007. Retrieved July 11 2008 from <http://www.encyclopedia.com/display/encyclopedia/Open+Source+Databases+Comparison>

## BIOGRAPHICAL NOTE.

**Phillip T Sibanda** is from Zimbabwe, and completed his BSc Honours(Cum Laude) degree in Computer Science at the University of Fort Hare in 2006. He is currently studying for the MSc degree at the University of Fort Hare, funded by NRF.