MIGRATION OF AN OLTP SYSTEM FROM ORACLE TO MYSQL AND COMPARATIVE PERFORMANCE EVALUATION

Mohit Nanda[1] and Amol Khanapurkar[2]

[1,2]Performance Engineering Research Center,
Tata Consultancy Services Ltd., Mumbai
[1]mohit.nanda@tcs.com, [2]amol.khanapurkar@tcs.com

ABSTRACT

Across the various RDBMS vendors Oracle has more than 60% [6] of market share, with a complete feature-rich and secure offering. This has made Oracle as default choice as the database choice for systems of all sizes.

There many open source databases as MySQL, PostgreS, etc. which has now evolved into complete feature rich offerings and come with zero-licensing fee. This makes it an attractive proposition to migrate from Oracle to an open-source distribution, to cut-down on licensing costs.

Migrating an application from a commercial vendor to open source is based on typical concerns of functionality and performabilty. Though there are various tools and offerings available to migrate but currently there exists no reference points for the exact effort and impact of migration on the application.

Thus we did a study of impact analysis and effort involved in migrating on OLTP application. We successfully migrated the application and did a performance comparison, which is covered in the paper. The paper also covers the tool and methodology used, along with the limitations of MySQL and presents learnings of the entire exercise.

1. INTRODUCTION

The commercial databases used in the applications today come with a high licensing fee and plethora of features. Many a times, an application can fit in a system with smaller subset of features and commercial databases are overkill. At the same time the feature-overload and high licensing fee add to perpetual support and licensing costs.

Over the time, the open-source database systems have caught up with the commercial vendors, in terms of features and performance claims. This has created new opportunities to lower the costs by migrating from commercial vendors to open-source vendors, and enter the zero-licensing-cost bracket.

Migration of a running system is a risky proposition, with a lot of gray areas. Thus we took this exercise of migrating an OLTP application from Oracle to MySQL.

We did a thorough analysis of effort required and functional & performance impact, hence throwing light to the gray areas. Along the course, we also compared the two databases extensively and have documented our experience, learnings and results in this paper.

© CS & IT-CSCP 2011 DOI: 10.5121/csit.2011.1210

2. THE OLTP APPLICATION - EQUIZ

The application we chose to migrate is an Online Quizzing Platform, eQuiz, which is currently being used in production in a large scale IT Company. The application empowers an organisation-wide quizzing platform, as the online quizzing solution for its Qualifying round, for 3 years now.

2.1 Technology Stack

The application is built on a three-tier stack of using JSPs, Java Servlets and Oracle Stored Procedures.

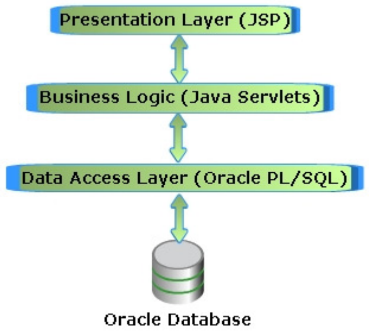

Presentation Layer (JSP)
Business Logic (Java Servlets)
Data Access Layer (Oracle PL/SQL)
Oracle Database

Fig 1. Application Technology Stack

Since the application relies heavily on PL/SQL Stored Procedures for data-access, migrating them accurately was the most vital task.

2.2 Database User Objects

The database user objects in the application had a fair mix of nearly all type of objects [Table 1]. With 256 stored procedures, inside 32 packages, migrating them was the core of the migration process.

Table 1. Database User Objects

User Object	Count
Tables with Data	63
Sequences	22
Triggers	5
Packages [Stored Procedures]	32 [256]
Views	7
Functions	5

3. COMPARISON: ORACLE V/S MYSQL

Oracle is one of the most widely used commercial RDBMS and comes packed with a huge number of features. MySQL is relatively a newer database but it nearly has all the features that Oracle has. We did a detailed comparison of both the systems and found the overlaps and exclusions in both the systems. [Fig 2]

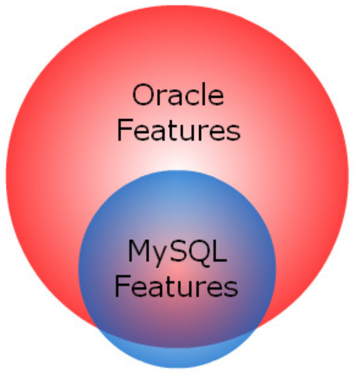


Fig 2. Oracle v/s MySQL Features

3.1 Server Side Features

- MySQL supports only built-in authentication methods, with no support for LDAP, CAS, et al
- There does not exist groups or roles, hence limiting the control on ACL
- Execution plans are not cached globally, only per-connection.

3.2 SQL & Other Database Objects

- MySQL does not have support for Packages enclosing stored procedures.
- There exists no Sequences in MySQL; however an alternate is to use AutoNumber field type.
- Materialized Views are not presents.
- Recursive queries are not supported in MySQL.

3.3 Indexes, Joins and Storage Engines

- Each storage engine supports different types of indexes with B-Tree indexes, supported by most of them.
- MySQL does not support bitmap indexes.
- Sort-merge joins or hash joins are not present in MySQL, There is only one type of join plan: nested-loop.
- In MySQL, the number of joins per query is limited to 61.
- In MySQL index on an expression are not allowed, you can only index columns.
- In MySQL, each table can have a different storage engine, with each engine having different features.
- InnoDB is one of the most advanced and popular storage engine, with transactions and row-level locking granularity.

4. COMPATIBILITY

Before proceeding with the migration, we did a compatibility check for the application. We did a break-down of all objects, major keywords and features used and checked them for availability in MySQL.

Doing this, we also analyzed the differences in different versions on MySQL and alternatives available if an exact match was not present. Since, lack of an exact match would require manual effort to rewrite/modify existing code, this was an important exercise.

4.1 Features and Versions

The earlier versions of MySQL (before v5.5) did not support Stored Procedures, thus making it difficult to migrate applications, which extensively used Oracle PL/SQL.

From version 5.5 onwards MySQL supports Stored Procedures along with all other Database User Objects that are used in this application. Thus we decided on using MySQL Server 5.5 as the target database.

4.2 Database Objects

Nearly all database objects that our application used, were present in MySQL, except the two listed below:

- **Sequences**
  The application used sequences for generation various recordIDs as primary keys. We overcame that using AutoNumber data-type in MySQL.

- **Utl_raw package for encryption/decryption**
  The application used utl_raw package to encrypt the questions kept in the question bank

- **Functions as RANK() and 4-argument INSTR()**
  The two functions mentioned above did not have an exact match present in MySQL and hence we had to require the functionality using SQL in MySQL Stored Procedures. The effort was not huge and we could achieve desired functionality by implementing the logic in SQL.

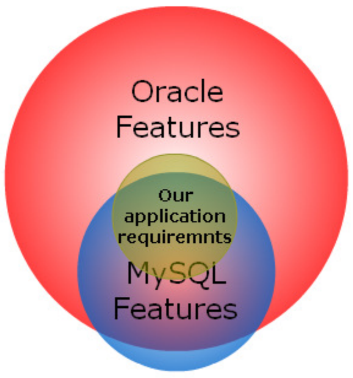


Fig 3. Application requirements w.r.t. Oracle & MySQL

4.3 Keywords

There are several keywords which differ in MySQL against Oracle but most of the time, their purpose remains the same. The tool did most of the job of conversion, but we analyzed the equivalents available and some of them are listed in Table 2.

Table 2. Difference in Keywords

Oracle	MySQL
NVL	IFNULL
SYSDATE	CURRENT_TIMESTAMP
TO_CHAR	DATE_FORMAT / STR
TO_NUMBER	CAST
dateA – dateB	TIMESTAMPDIFF
dateA + dateB	TIMESTAMPADD

5. THE MIGRATION EXERCISE

5.1 The Tool – Ispirer SQLWays

Migrating all database user Objects manually was a mammoth task, hence evaluated a number of tools available in the market and closed on Ispirer SQLWays [5]. It is a fairly mature tool with a large set of supported databases for heterogeneous migrations. It also provided with a whitepaper [3] for the task which covers the basics.

The tool is pretty straightforward to use and uses System DSNs to connect to source and target databases. It reads the source database, gives you a selectable list of objects to migrate. It generates executable SQL scripts for the target database and logs the entire cycle well, for each transformation.

Using the tool, we could easily 90% migrate all database objects including Stored Procedures. The remaining work had to be done manually by going through the cycle of functional testing and debugging.

5.2 Changes in Application Code

- **Stored-Procedure Calls**
  Due to change in procedure names, all calls to stored procedures had to be modified to support the new nomenclature. Rename from PkgName.ProcName to PkgName_ProcName was done using regular expressions:
  ```
  FIND        : (getStatementHandle\([^)]+)(\.)([^)]+\))
  REPLACE     : \1_\3
  ```

- **Resultset Handling**
  The resultset(s) returned by Stored Procedures in MySQL required handling in Java in a different way. Instead of accessing them using getObject(cursorPos), cStmt.getResultSet() was used.

- **Connection String**
  The connection string was modified to use MySQL ODBC drivers.

5.3 Changes at Database End

- **Package-Procedure Nomenclature**
  Since there exists no packages in MySQL, the stored procedures' names were modified in a way that PkgName.ProcName was renamed in MySQL as PkgName_ProcName. It was done automatically by the tool.

5.4 Effort Required

The tool generates executable SQL scripts and nearly 90% of the Database User objects got migrated out-of-the-box using these scripts. However, the remaining 10% required manual effort to analyze and modify the SQL to fit the need. This was done using Regression testing of modules, along with debugging.

The entire process tool close to 33 PD for the process, where time was spent in Analysis, Migration, Functional Testing and Debugging.

5.5 Limitations of tool

- The tool could not convert date formats used in TO_DATE and TO_CHAR statements, and hence those had to be modified manually.

- Oracle supports || operator for String Concatenation while MySQL does not and treats it as logical OR. The tool could not transform String Concatenations automatically. They had to be manually modified using STRCAT.

- It converted statement with 4-argument INSTR() to 3-argument LOCATE(), which is not-an-exact equivalent and thus resulting in logical errors. Had to be fixed later manually.

6. IMPACT ANALYSIS

6.1 Functional Verification

The system did undergo a full round of regression testing to assert that all functionalities were still behaving in the same way. For the few functions which did not pass the tests, the Stored Procedure code had to be modified / rewritten to accommodate compatibility issues. These were chiefly due to:

- Lack of sequences in MySQL

- Lack of inbuilt Encryption/Decryption functions

After the required changes, the application with MySQL now behaves as desired with no compromises on functionality.

6.2 Performance Tests

We conducted performance tests to compare the performance of the system with MySQL against Oracle. The tests were conducted using a suite with Open Source Load Testing Tool Grinder [7]. The tests were conducted in the same test environment to get apples-to- apples comparison.

6.2.1 Load Tests

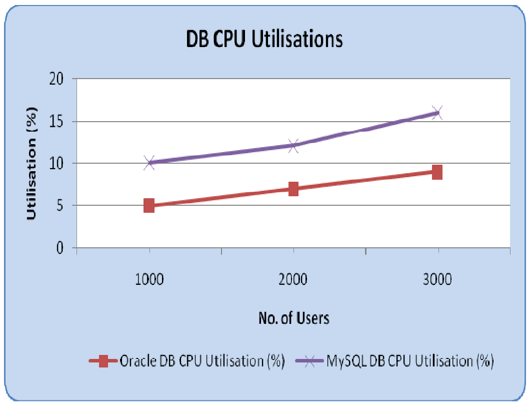
The load tests were conducted using real-time scenarios:

Workload	: 1000 – 3000 concurrent users
Think-time	: 30 seconds
Transaction Mix	: Login -> Take Test -> Logout

Table 3. System Utilisations under workload

	Oracle		**MySQL**	
#Users	**App CPU Utilisation (%)**	**DB CPU Utilisation (%)**	**App CPU Utilisation (%)**	**DB CPU Utilisation (%)**
1000	5	5	8	10
2000	6	7	10	12
3000	8	9	12	16

The CPU utilizations were slightly higher, in case of MySQL as compared to Oracle. However all workloads executed within acceptable utilizations close to 15%. [Table 3]



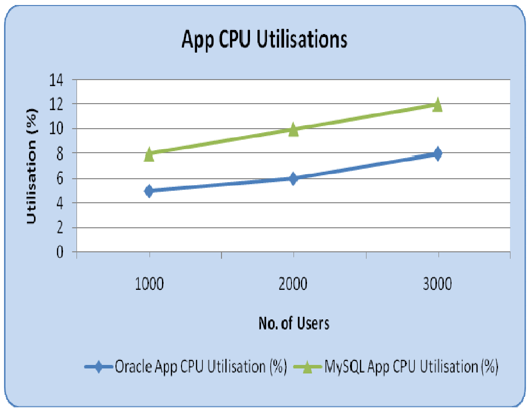


Fig 4. DB & App CPU Utilisations (%)

The tests depict that the MySQL system gives a competitive performance with approx 5% less Response Times and Throughput [Fig 5] than Oracle.

Table 4. Avg. Response Time & Throughput

	Oracle		**MySQL**	
#Users	**Response Time (ms)**	**Throughput (pps)**	**Response Time (ms)**	**Throughput (pps)**
1000	0.28	34.49	0.27	32.9
2000	0.73	68.85	0.71	64.62
3000	0.86	103.2	0.82	101.3

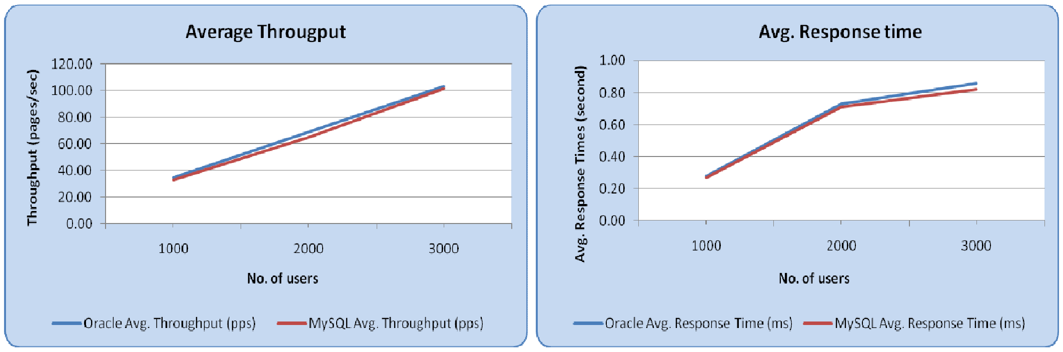


Fig. 5. Average Throughput (Pages/sec) & Response Times (ms)

6.2.2 Point-load Stress Test

We also did a point-load stress test with zero think time to see if the system can sustain high peak loads with zero think times.

The stress tests were conducted as the following real-time scenario:

Workload	: 500 concurrent users (no ramp-up) x 1 iteration each
Think-time	: 0 seconds
Transaction Mix	: Login -> Take Test -> Logout

The system handled the load with acceptable response times and CPU utilizations [Table 5].

Table 5. Point-load Stress test

500 concurrent users x 15 iterations with Zero Think Time				
Database	**App CPU Utilisation (%)**	**CPU Utilisation (%)**	**Avg. Response Time (ms)**	**Avg. Throughput (pps)**
Oracle	12	35	1.73	272.00
MySQL	15	30	1.90	262.30

6.2.3 Test Environment

The test environment comprised of two server class machines used for Application and DB Server Respectively. The load testing tool was deployed on a separate server along with other machines, used as load generators. For monitoring system utilizations, system utilities as sar and top were used.

Table 6. Application & Database Server Configuration

Hardware Configuration	
CPU	16 core 1.7 GHz Xeon
Memory	8 GB
Software Configuration	
OS	Ubuntu Linux
App Server	Apache Tomcat 6.0.14

Hardware Configuration	
CPU	4 core 2.66 GHz Xeon
Memory	4 GB
Software Configuration	
OS	CentOS 5.0
DB Server (1)	Oracle 10.2.0.1
DB Server (2)	MySQL 5.5

7 CONCLUSIONS

The entire exercise of analysis and migration took close to 33 PD, with almost 11 PD spent in analysis of feature compatibility and feasibility of migration. The results suggested that we could successfully migrate such type of an OLTP application with no compromises on features. The performance evaluation proved that for this application there would be no major impact on performance.

All the database objects were successfully migrated, with most of the effort spent in migrating stored procedures. The effort chiefly was spent in functional testing and writing equivalent logic where an exact match of Oracle PL/SQL function was not available.

The effort spent in the migration activity is justified by the cost benefit achieved by it. This has brought the application to Zero-cost licensing fee bracket. Using MySQL as the database for our application which is not mission critical, we have been able to cut down on total cost of ownership.

Other peripheral advantages of this migration include easier deployment on cloud. This is because lot of cloud service providers, provide standard off-the-shelf images with MySQL installed.

The paper has given analysis of feature compatibility and performance comparison between Oracle & MySQL. If the suitability of migration of an application/project to MySQL is established, then its attractive proposition to enter zero-cost licensing fee bracket.

Table 7: Abbreviations & Terms used

Abbreviation/Term	Definition
PL/SQL	Procedural Language / Structured Query Language
pps	Pages per second
DSN	Data Source Name
RDBMS	Relational Database Management System
LDAP	Lightweight Directory Access Protocol
CAS	Central Authentication Service
OLTP	Online Transaction Processing
OLAP	Online Analytical Processing

ACKNOWLEDGEMENTS

We would like to thank Dr. Rajesh Mansharamani, VP & Head, Performance Engineering Research Center, TCS for supporting us with his able and effective guidance.

We would also like to thank Mr. Ajeet Ojha from TCS and Mr. Jayanti Murthy from Global Consulting Practice–Performance Engineering, TCS for their expert guidance on databases throughout the process.

REFERENCES

[1] Jutta Horstmann, "Migration to Open Source Databases", Technical University Berlin, September 2005

[2] [Online] AmazonTech Inc., "Database Migration to MySQL" http://www.mysql.com/why-mysql/white-papers/mysql_amazontech_mssql2mysql_paper.pdf

[3] [Online] Ispirer's whitepaper on "Oracle to MySQL migration", March 2009 http://www.ispirer.com/doc/sqlways-oracle-to-mysql-whitepaper.pdf

[4] [Online] MySQL Migration Toolkit http://dev.mysql.com/doc/migration-toolkit/en/index.html

[5] [Online] Ispirer SQLWays http://www.ispirer.com/products

[6] [Online] Oracle India Pvt. Ltd. Flyer http://www.ibef.org/attachdisplay.aspx?cat_id=432&art_id=6021

[7] [Online] The Grinder, a java load testing framework http://grinder.sourceforge.net