Middleware Layer for Replication between Relational and Document Databases

Susantha Bathige
Sri Lanka Institute of Information Technology (SLIIT)
Malabe, Sri Lanka

PPG Dinesh Asanka
Pearson Lanka (Pvt) Ltd
Colombo – 09, Sri Lanka

***Abstract*— Effective and efficient data management is key to today's competitive business environment. Data is a valuable asset for any organization. Data is information and information is knowledge. Today's enterprise applications generate large amount of data especially because of high usage of the internet. As a result, enterprise applications should be able to scale out and perform as required. Otherwise, these applications will not be able to handle their data load, leading to trouble in business continuity. Data replication is one widely used phenomenon in distributed environments, where data is stored in multiple sites, within same or differing geographical areas. Thus enterprise applications will be able to scale out, so that they perform well. Many modern Database Management Systems (DBMSs) provide in-built methods for data replication. Replication is possible between heterogeneous database systems. In this research, the aim is to design and implement a middleware layer for data replication from RDBMS to document oriented DBMS. The middleware layer includes a Java program, source and destination DBMSs. The research approach is to capture DML/DDL changes in source relational DBMS and then convert and stores them in an intermediate XML format. Then the java program continuously looks for such data changes and then push them to the destination DBMS. A replication method like this can address the need of application scalability in situations where both types of DBMSs are used. In live environments, there are areas of possible performance improvements to the middleware layer, especially when dealing with large data volumes.**

Keywords*— *NoSQL, Document Databases, Relational Databases, Data Distribution, Replication

I. INTRODUCTION

Many database systems are being used to store data. Data growth is exponential [1] [2]. Internet growth is almost synonymous to data growth. Main reasons for high usage internet are the introduction of Web 2.0, use of latest devices like Tabs, Smart phones, increased IT literacy, etc.

Data can be categorized into three;

1. Structured data
2. Semi-structured data
3. Unstructured data

RDBMS pioneered in handling structured data which has a fixed structure and less dynamic.

Most of the database systems are based on the well-known, popular concept of relational theory [3]. There are other DBMSs such as object oriented, object relational, NoSQL, etc. Each of these DBMSs has its own strengths and weaknesses. An enterprise has the freedom to use a single database system or multiple database systems to meet the data management needs.

The vast popularity of the Internet caused to generate more data for applications. There should be proper DBMS(s) [4] to manage these data effectively and efficiently. If an enterprise uses many different database systems due to various reasons, there might be a need to exchange data between database systems. Obvious reasons are like performance, scalability and availability.

There are various data distribution methods available in different DBMS, such as import / export, log shipping, replication, database mirroring etc.

The objective of this research is to build a middleware layer to replicate data from a RDBMS to a document DBMS in real-time.

MS SQL Server [5] and MongoDB [6] were chosen as sample DBMSs to demonstrate the model. MS SQL Server is a dominant market database product [7] and it is considered as post-RDBMS whereas MongoDB is an emerging DBMS product [8] which can be categorized as document oriented DBMS.

Presently there is no in-built method of integrating these two DBMSs to replicate data.

II. PREVIUS WORK

Research paper, "MyStore: A High Available Distributed Storage System for Unstructured Data" [9] makes a meaningful attempt to introduce a new methodology and implementation to combine several NoSQL DBMSs and provide a new distributed storage system called MyStore. The objective of the research is to take unique advantages provided by each NoSQL database system (MongoDB, Cassandra and Dynamo) and combine into one. As per the paper, MongoDB provides perfect query functions while Cassandra and Dynamo provides data availability and scalability.

Almost all the popular DBMSs have their own in-built replication mechanisms as a data distribution method. A replication which works within the same DBMS is known as homogenous replication, whereas replication works between different DBMSs is known as heterogeneous replication. Now the question arises about the need of heterogeneous replication. The middleware layer for data replication presented in this

research falls under heterogeneous category, as it distributes data from RDBMS to non-relational document oriented DBMS.

A. *Heterogeneous data replication*

As stated in above paragraph, heterogeneous replication replicates data between two different DBMSs. One of the industry's leading RDBMS, MS SQL Server supports heterogeneous replication. [10] [12] With this, the users have the option of publishing data from Oracle to MS SQL Server and vice versa. However, in MS SQL Server 2012 which is the latest release of MS SQL Server, has announced that this feature will be removed from the future versions of SQL Server. The alternate method they suggest is to use CDC, change tracking and SSIS. MS SQL Server also supports DB2 for heterogeneous replication [11].

As per the paper, the product DataJoiner is IBM's strategic gateway to enable transparent access to relational and non-relational, IBM and non-IBM databases. This is especially important to the current research because this product claims to be working for both relational and non-relational databases. The further reading of the paper reveals insufficient information on non-relational support of the heterogeneous replications. It could not find sufficient or valuable information on DataJoiner product.

Research abstract of "DataJoiner: A Practical Approach to Multi-Database Access" [13] presented a solution to distribute and migrate data across multiple DBMSs. It is also a middleware. The paper published on 1994 before the NoSQL technology emerged. The paper does not mention whether it supports non-RDBMS. Since it is not mentioned the support for non-RDBMS, here it is assumed that the "DataJoiner" works only for RDBMS.

III. METHODOLOGY

Fig. 1 illustrates the overall replication architecture of the middleware layer.

There are several techniques and methods that can be used to build the replication middleware. The main two methods used in the middleware layer are converting XML to JSON documents and pushing of JSON documents to document DBMS. The services run continuously in background and do the conversion and importing as and when necessary.

Fig. 2 illustrates the main steps of replication process and it can be categorized into two;

1. Activities inside RDBMS
2. Activities outside the DBMSs

The steps of each category are listed below;

A. *Activities inside RDBMS*

1. Identify the table(s) to be replicated.
2. Identify and capture DML/DDL changes.
   DML changes are data modification through CRUD (Create, Retrieve, Update, and Delete) while DDL is limited to add/remove columns.
3. Convert captured DML/DDL changes to XML like format and stores them in a central table named, Document_Repl.

B. *Activities outside the DBMSs*

4. Extract XML data stored in Document_Repl and convert them to JSON documents.
5. Push JSON documents to document DBMS.

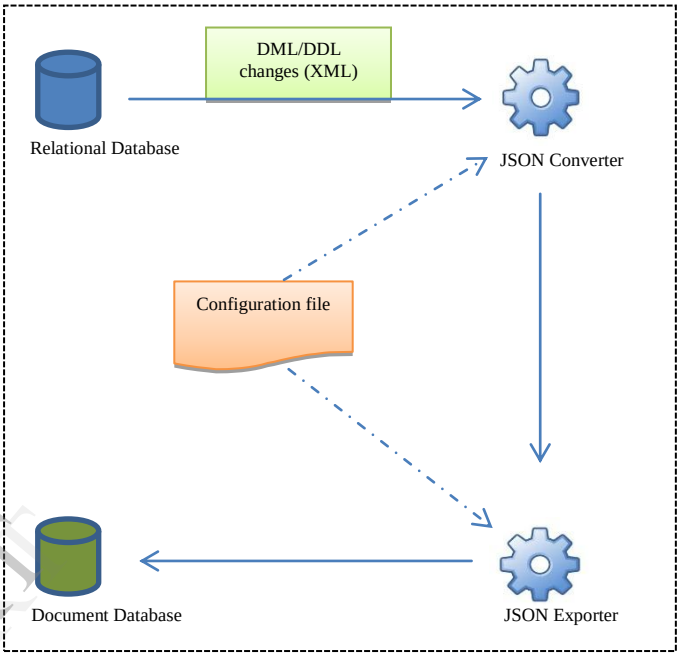

Fig. 1. Replication architecture

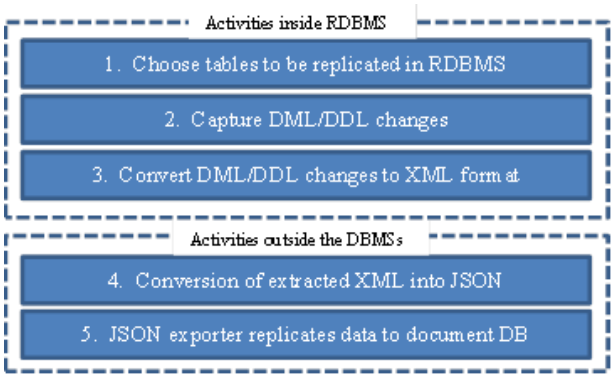

Fig. 2. Replication process

There are several methods and techniques to capture DML/DDL changes. In any RDBMSs, every data change is written to a separate file named, transaction log. There are several mechanisms available in RDBMS to read and capture the transaction log data. However, the data in transaction log is not exposed to users of RDBMS.

After evaluating several options such CDC, Trigger, Change Data Tracking etc., it is decided to use a trigger based mechanism to capture DML/DDL changes. Triggers are used since they are common to many RDBMSs and easy to implement.

There is a performance impact in use of triggers. However, it can be minimized by using an alternative approach. The

performance impact due to triggers is measured for middleware layer and it has been discussed in Section 5.

. In this research, trigger based mechanism is used for prototype validation of the middleware. The implementer has the freedom to use any mechanism of their preference depending their needs.

Data in RDBMS stores in several tables due to normalization. Normalization [14] is a process of database designing in RDBMS.

The relational design of typical Sales Order system is represented in document databases as un-normalized way. The four tables, Customer, Item, SalesOrder and SalesOrderDetails can be modelled in mostly three collections (collection is the terminology for a table in document database) in document DBMS. Fig 3 shows sample data for SalesOrderDetail table in the relational database.

SalesOrderDetail

OrderDetail_PK	Order_PK	Item_PK	Qty	UnitPrice
1	1	1	5	105.00
2	1	2	25	13.50
3	1	3	55	65.50
4	1	10	65	1.50
5	1	20	35	1.50

Fig. 3. Sample data set of SalesOrderDetail table

Figure 4, shows the document oriented database design for sales order system. There are three collections as mentioned below;

1. Item collection

2. Customer collection

3. Order collection

Both SalesOrder and SalesOrderDetail tables are de-normalized into one collection, Order. Customer and Item tables still may remain as separate collections.

When SalesOrderDetail table is marked for replication, it captures DML/DDL changes of the table using triggers.

There has to be four triggers. Two triggers for INSERT and UPDATE need to be added to SalesOrderDetail table while SalesOrder table need another two triggers for UPDATE, DELETE.

The DML/DDL changes which captured by the triggers are not the net change. E.g.: Even a unit price changed for an Order the entire Order is captured.

SQL command "XML AUTO" clause used to convert the captured DML changes to XML format inside the trigger. However the trigger code is not portable without modifications to another RDBMS. Depending on the different syntax and commands the trigger code has to go through some modifications. The reason is a lack of cross-vendor portability in SQL [15].

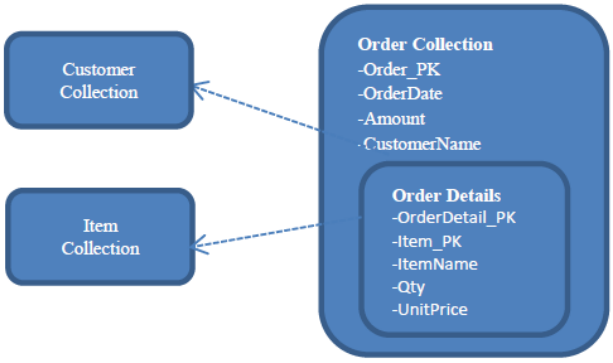

Fig. 4. Document oriented DB design for typical Sales Order system

All DML/DDL changes of tables being marked for replication, writes to a core table, "Document_Repl" in the source database.

XML to JSON conversion is done using the Java program. It frequently searches the Document_Repl table and filter for records where IsReplicated=0 and then picks them and converts to JSON objects. This table has potential data growth as it contain the data to be replicated. Nevertheless, the data in the table can be cleaned periodically to flush out all the records which has already replicated. This will lead to excessive fragmentation and proper administrative tasks such as rebuilding indexes can solve the problem.

Pushing JSON documents to document database is handled by the same Java program. The converted JSON objects are then pushed immediately to the target document database.

A Java program has been written in a configurable and extendible manner using Factory and Singleton design methods. The source database and target database connection strings are configurable and new DBMSs could be added with minimum changes to the program.

The main reason to use Java as the development platform is to achieve platform independence. As a result the same program could be run in both Windows and Linux platforms without doing any modifications.

IV. SECURITY

Security is one of the major concerns and critical factors when it comes to any software program. Same applies to the replication middleware as well. There are several places that security has to be addressed in the middleware layer. These are;

C. Data being subject to replication

Since Document_Repl is stored in RDBMS, the security features available in underlying RDBMS has to be used to protect the data

D. Replication link

Replication link is the data path from RDBMS to document database. Data replicates from RDBMS has to be protected while transferring in the network path until they reached the target database. There are various types of network data protection which can be applied to secure the data, e.g. Data encryption, Digital Signatures, Authentication

E. Configuration files

Middleware uses configuration files to keep connection string information for both relational and document databases. The connection strings have sensitive data such as user names and passwords. As a result this data also needs to be protected. By design, the middleware layer encrypts the sensitive data in configuration files.

V. Performance Impact Analysis

The replication method has a performance overhead in several aspects. Any method developed to solve business problems has some kind of performance overhead. The same applies to the replication middleware layer too.

Consider the performance overheads to the source DBMS. There are two stages. They are;

1. Capturing of DML/DDL changes
2. Conversion of relational data to unstructured format (XML)

The performance overhead differs with the method used on each of the above two steps. As stated in previous sections, there are various methods to capture DML/DDL changes of a table.

Converting of relational data to unstructured format adds additional overhead to the source RDBMS. The performance overhead differs with the method used for the conversion.

Adding an index to IsReplicated column in Document_Repl table is been considered, and it is noted that the introducing of index slows down the CRUD operations in source tables which are marked for replication. The index will not add much value, if archiving mechanism is implemented. This layer is the only layer that adds performance impact to the transactions in source DBMS.

There will not be any performance impact to the destination database as we only do CRUD operations to the destination database.

There is a delay in getting data to the destination DBMS. This delay consists of the following components;

- Time taken to capture DML/DDL changes of the tables marked for replication. (T1)
- Time taken to convert the relational data to database independent XML format and store them in Document_Repl table. (T2)
- Time take to convert data which is in XML format to JSON documents. (T3)
- Time taken to push JSON documents to target database. (T4)

Total replication latency = T1 + T2 + T3 + T4

This latency is generally a few milliseconds and it depends on the source and target DBMSs driver software performance, which is used to connect to the DBMSs. Time T4 is largely depends on the performance of the network connectivity of the DBMSs. As a result there is little or no control over the T4 time component of the replication latency to the replication method.

There is another performance overhead when the records in source DB changes frequently. In this case it replicated entire record instead of the net change.

Table 1 shows the hardware/software configurations which have been used for the performance testing and Table 2 shows the versions of DBMSs used.

TABLE 1 MACHINE CONFIGURATION (HARDWARE/SOFTWARE)

Configuration	**Value**
PC – Manufacturer	Hewlett-Packard
Model	HP ProBook 4530s
Processor Manufacturer	GenuineIntel
Processor	Intel(R) Core(TM) i7-2670QM CPU @ 2.20 GHz
Number Of Logical Processors	8
Number Of Cores	4
Total Physical Memory	8 GB
Hard disk	300 GB

TABLE 2. DATABASE VERSIONS

Product	**Version**
MS SQL Server	Microsoft SQL Server 2012 - 11.0.2100.60 (X64) Feb 10 2012 19:39:15 Copyright (c) Microsoft Corporation Developer Edition (64-bit) on Windows NT 6.1 <X64> (Build 7601: Service Pack 1)
MongoDB	2.06 64 bit

Followings matrices are collected in performance testing;

1. Process utilization
2. Memory usage
3. Time

"perfmon.exe" is the tool used. By default this tool comes with any version of Windows OS.

Performance matrices are collected in three stages;

1. With no transformation and no replication – Just inserting 1 million sales orders into SalesOrder and SalesOrderDetail tables in MS SQL Server.

2. With transformation and no replication – Inserting one million sales orders into two relational tables (SalesOrder, SalesOrderDetail) while transforming those data into unstructured format and store them in another relational table (Document_Repl) in MS SQL Server.

3. With transformation and replication – This stage has transformation and replication in place, while creating one million sales orders the order data replicate into MongoDB real-time.

Fig. 5 shows the processor utilization of MS SQL Server, MongoDB processes and total processor utilization for stage 3. (With transformation and replication), it has secondary axis to show only the MongoDB processor utilization for clarity purposes.

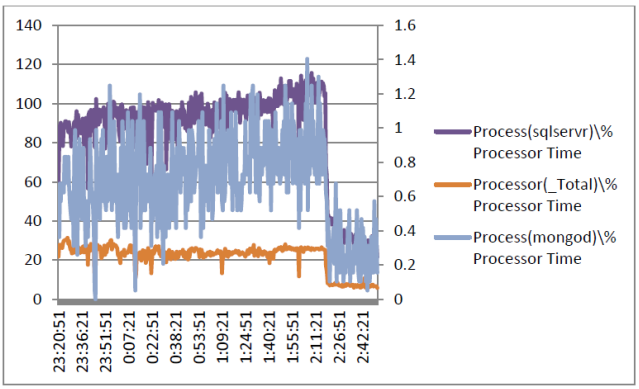

Fig. 5. Processor utilization – "With transformation and replication"

Fig. 6 shows the memory utilization of the computer with regard to stage 1 testing. (With no transformation and no replication)

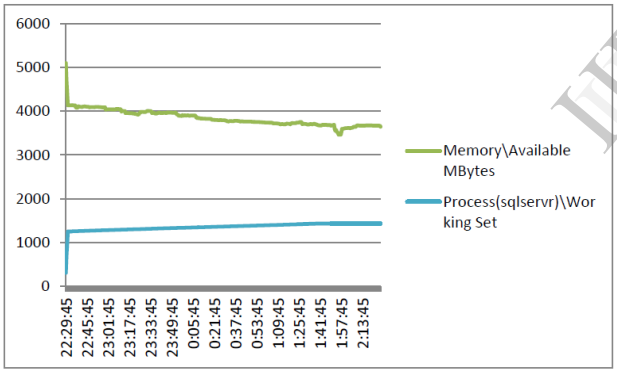

Fig. 6. Memory usage – "With no transformation and no replication"

Table 3 shows the detail time analysis of the performance testing.

TABLE 3
DETAIL TIME ANALYSIS OF STAGE WISE

	Time Taken (milliseconds)		
	No transformation, no replication (Just INSERT)	With transformation, no replication	With transformation, with replication
Generate 1 m Orders / Size (183,222,272 Bytes)	11,407,000	11,704,000	13,558,000
Time taken per record	11.407	11.704	13.558
Time taken per B	0.06225	0.0639	0.074

Table 4 shows the summary of time analysis of the performance testing.

TABLE 4: TIME ANALYSIS - SUMMARY

Layer	Impact (milliseconds)	%
Transformation	0.00165	2.65
Transformation plus replication	0.01175	18.88

VI. LIMITATIONS

Following limitations were identified in the middleware.

- The data in RDBMS cannot be replicated to multiple document databases.
- The developed prototype depends on the XML features in RDBMSs. As a result the same prototype could not be adapted as it is for other RDBMSs which do not provide XML commands to XML conversions.

VII. FUTURE WORK

The following can be considered to improve the replication model further.

- Platform independent solution to convert relational data to XML. This could have been achieved by writing a separate database independent component. However, this would definitely increase the latency of replication. The advantage of such a system would be that the conversion process is immune to changes even with different types of source DBMSs.
- Further improvement to Java program which builds the replication link, so that replication latency could be minimized.
- Implementation of synchronous replication mechanism. This ensures high data integrity between both databases.
- Bi-directional replication can be implemented to improve the functionality of the middleware.

VIII. DISCUSSION AND CONCLUSION

The research question was to find an effective and efficient way to replicate data from RDBMS to document oriented DBMS. During the development of prototype application, it is realized that it would take a lot more time and effort to create an enterprise level application of this kind. However, the research gap has been addressed substantially with the middleware layer introduced in the research.

There could be more performance testing carried out in various levels including multi user testing, load testing and stress testing.

IX. REFERENCES

[1] "Internet World Stats," Miniwatts Marketing Group, 25 7 2011. [Online]. Available: http://www.internetworldstats.com/emarketing.htm. [Accessed 8 1 2012]. [1]
[2] D. Connolly, "A Little History of the World Wide Web," W3C, 07 10 2011. [Online]. Available: http://www.w3.org/History.html. [Accessed 8 1 2012]. [2]
[3] E. F. Codd, "A relational model of data for large shared data banks," Magazine, pp. 377-387, 6 6 1970. [4]
[4] "Relational Model," Wikimedia Foundation, Inc., [Online]. Available: http://en.wikipedia.org/wiki/Relational_model. [Accessed 12 1 2012]. [5]

[5] P. Kahn, "Object Database Management Systems," IBM Academic Initiative, Courseware, 2005-2012, [Online]. Available: http://www.odbms.org/. [Accessed 12 1 2012]. [7]
[6] "NoSQL," [Online]. Available: http://nosql-database.org/. [Accessed 12 1 2012]. [8]
[7] "TOP 10 Enterprise Database Systems To Consider," ServerWatch, 20 5 2012. [Online]. Available: http://www.serverwatch.com/trends/article.php/3883441/Top-10-Enterprise-Database-Systems-to-Consider.htm. [Accessed 29 9 2012]. [9]
[8] 10gen, "mongoDB Customers," [Online]. Available: http://www.10gen.com/customers. [Accessed 5 1 2012]. [10]
[9] L. Z. W. Q. H. J. Y. P. Wenbin Jiang, "MyStore: A High Available Distributed Storage System for Unstructured Data," in 2012 IEEE 14th International Conference on Communication, Networking & Broadcasting ; Computing & Processing (Hardware/Software), Wuhan, China, 2012. [17]
[10] "Heterogeneous Database Replication," Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/ms151149.aspx. [Accessed 10 7 2012]. [19]
[11] Microsoft, "Non-SQL Server Subscribers," Microsoft, [Online]. Available: http://msdn.microsoft.com/en-us/library/ms151864.aspx. [Accessed 06 12 2012]. [20]
[12] P. Haase, "Heterogeneous Data Replication," University of Rostock, Department of Computer Science. [21]
[13] P. Gupta, "DataJoiner: a practical approach to multi-database access," in Proceedings of the Third International Conference, San Jose, 1994. [23]
[14] T. J. Teorey, S. S. Lightstone, T. Nadeau and H. Jagadish, "Normalization," in Database Modeling and Design, Morgan Kaufmann, 2011, p. 352. [25]
[15] ITL Education Solutions Limited, "Structured Query Language," in Introduction to Database Systems, Pearson Education India, 2008, p. 592. [26]