Volume 5, Issue 12, December 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A GUI Based Unified Utility for Data Migration

[1]Nikhilesh Chaudhari, [2]Umair Baig, [3]Aditya Bhandare, [4]B. B. Gite
[1, 2, 3] BE Computer, [4] HOD of Computer,
[1, 2, 3, 4] University of Pune, Maharashtra, India

Abstract— Now a days we are watching at a transition in the style and type of storage of data. Traditionally, the data was stored in a relational model. The relational model is a tabular representation of data. The relational model was a very successful model for storage and representation of data. It was and still is predominant for databases where transactions are involved. However these days, NoSQL databases are becoming increasing popular and are being widely accepted. New technologies such as Internet of Things (IoT) have added to the increasing significance of NoSQL. Also most of these NoSQL databases are open source like the MongoDB or CouchDB platform. And they have various different styles of how the data will be stored in them. For example, MongoDB is a document-based platform. The data in MongoDB is represented in the form of JSON document. JSON is acronym for JavaScript Object Notation. Its nature is of key-value pair. Another database platform Apache Cassandra is a column oriented database, where the columns can be dynamically added or increased as per requirement. In today's world, we are even having some instances where data is not only migrated from SQL platform only, but also from one NoSQL platform to another NoSQL platform. No database platform is ideal for all. We must choose the database platform as per requirement. For the traditional databases, to derive the benefits of the NoSQL, the respective relational data needs to be migrated to the newer platforms. The NoSQL platform provides benefits like scalability, schema-less data storage, no complex joins, tuning, faster access to data etc. over the relational platforms. This paper proposes a GUI based open source system to migrate the data from relational database platform to the NoSQL platform. The system provides facility to allow migration to multiple platforms as the client requires it. The client of the system is required to give which platform he needs the database to be migrated. It automates the process of data migration.

Keywords— JSON, ACID, BASE, NoSQL, SQL, DATA TRANSLATOR.

I. INTRODUCTION

The tremendous use of internet, social media sites and cloud computing has challenged the relational model of data storage. The large amount of data really raises the question whether the relational model is still capable of handling it or not. The speed at which data is being generated nowadays and also the amount of data generated everyday plays an important role in deciding how and where the data should be stored and managed. The data that is generated by social websites, cloud storage and mobile phones in unstructured and of different types. This large volume of data can be actually handled by the relational databases, but the cost associated for handling and maintaining the data with it is very high and consume resources in a haphazard manner. The rigid schema nature of the relational model is the major reason for such a high resource consumption.

SQL stands for Structured Query Language and is used for accessing and manipulating relational databases. It was the first commercial languages for Edgar Codd's relational model. SQL is an ANSI standard however different versions of SQL are available in the market. They follow ACID properties. ACID stands for Atomicity, Consistency, Isolation and Durability. The SQL databases are dominant in systems where data consistency is the major requirement. It is still heavily used in systems involving transactions.

The NoSQL databases outperform SQL databases in most of the features except for data consistency. The NoSQL databases compromise data consistency in order to gain high availability, scalability and performance. But a large and dominant number of organizations do not require data consistency in their requirements.

The NoSQL databases are particularly schemaless. Hence they are very efficient in handling such unstructured data. It is a myth that NoSQL databases is only useful for websites where only unstructured data is to be stored. But this is not the reality. The websites and organizations using relational database for this purpose will soon feel the necessity and have the requirement of NoSQL databases as their clients and data starts growing. The major factors for the popularity of NoSQL databases are:

1. Low cost 2. High Performance 3. High Scalability. 4. High Availability. 5. Flexible Schema

II. CAP THEOREM

The RDBMS systems support ACID properties while the NoSQL systems follows BASE properties. Horizontal scalability is easily achieved by using NoSQL databases. Implementation of database in a distributed environment makes it difficult to achieve three major properties of databases like Consistency, Availability and Partition tolerance. This is the Brewster's theorem or the CAP theorem [2].

Consistency means that same data is visible to all clients in a cluster of nodes, even while having concurrent updates. Availability means that all database clients are available to access some version of data in a cluster even if a node in cluster goes down. Partition tolerance means that the system keeps working even if nodes in a cluster are unable to communicate with each other [2].

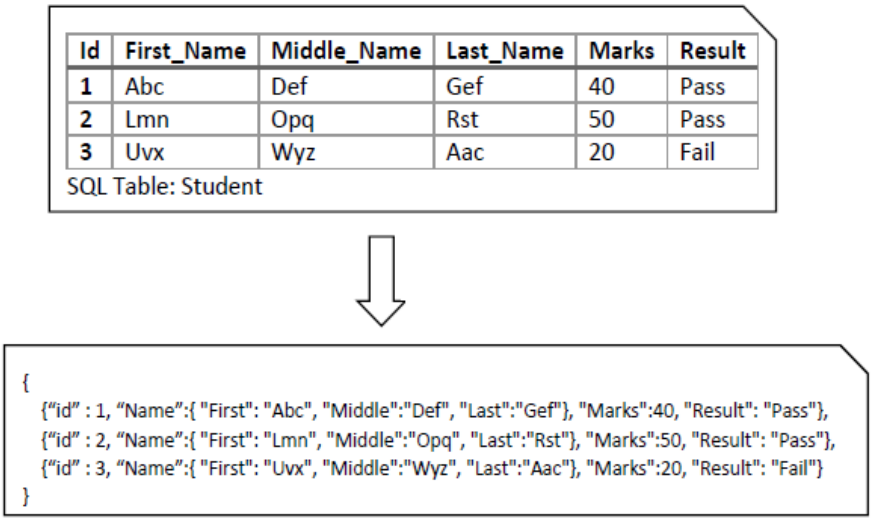
NoSQL database platform are classified as per the properties they consist of or give priority to. We here, are focused on NoSQL platforms named MongoDB, Cassandra and CouchDB. MongoDB is considered under consistent, partition tolerant systems (CP). Cassandra and CouchDB are classified under available, partition tolerant systems (AP).

III. MODELING MONGODB

Zhao [1] has proposed about modeling the MongoDB with the relational model. He has also given the comparison of semantic expression of both models and analysing the feasibility of the migration of a relational database to MongoDB and its evaluation. As said, MongoDB is a document oriented database. A collection in MongoDB is similar to table in SQL database. And the document's model in MongoDB is the same with the relational model in the relational database systems. One row of the table is equivalent a document in MongoDB, the keys are similar to fields in tables, and lastly, value in the table is equivalent to the value of the key.

The diagram given below gives exactly how the modeling will be done in the system. Thus, proves the feasibility of migration between relational database and MongoDB.

CouchDB [3] is also a NoSQL document based database by Apache software foundation. It also stores data in JSON format. The explanation and modeling given above about MongoDB migration is, hence, also applicable for CouchDB.



Id	First_Name	Middle_Name	Last_Name	Marks	Result
1	Abc	Def	Gef	40	Pass
2	Lmn	Opq	Rst	50	Pass
3	Uvx	Wyz	Aac	20	Fail

SQL Table: Student

Figure 1. Modeling MongoDB with Relational SQL Model

IV. MODELING CASSANDRA

Cassandra [4] is a column-oriented database by Apache software foundation. It is a distributed database and has no single point of failure by Apache. In Cassandra's terminology, node is a place where data is stored. Data center is a collection of related nodes, while cluster is a component that contains one or more data centers. Cassandra works on the Gossip protocol which allows the nodes to communicate with each other and detect any faulty nodes in the cluster. Unlike relational tables where a column family's schema is not fixed. Cassandra never forces individual rows to have all the columns.

V. NOSQL TO NOSQL MIGRATION

While the database platform is selected on the basis of requirement, many organizations have felt the need for migration of data from one NoSQL platform to another one. This is, majorly, because of any database cannot provide consistency, availability and partition tolerance at the same time. As the organizations feel of adding some new functionalities or features to their products, the database may require features that the present platform is unable to provide it. Also collaborations between organizations many a times require data sharing among organizations, and hence, indirectly require data migration. Clients many a time face problems, or many times are not happy with features of their working platform. For ex, many a time organization have migrated from their existing platform to another because of scalability problems. [7]

VI. EXISTING SYSTEMS

Mongify [5] [11] is one of the existing systems which migrates data from SQL to NoSQL database system. However it allows migration only to MongoDB database. It is based on self-defining routes. Tungsten Replicator is another such utility, but specifically used only for migrating to MongoDB platform. These systems work for only one specific database and does not give the client any other options that client may wish to migrate. The above systems only work for MongoDB.

VII. PROPOSED SYSTEM

As we have seen the existing system, they allow migrating only to a specific NoSQL platform. They do not have GUI for the system and thus they are complex to handle.

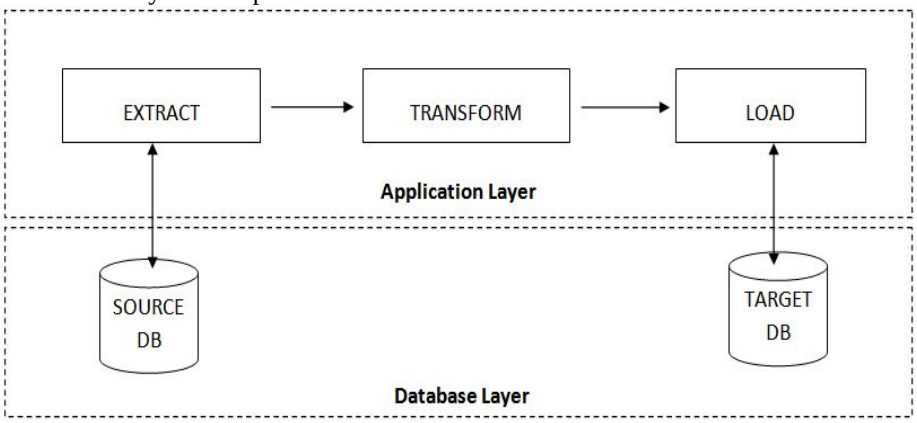


Figure 2 System Architecture

As we have seen the modeling of various schemas and migration feasibility above, a system where migration to multiple NoSQL database platform is possible. The system allows client to select which database he needs his data to migrate to. The system shows a complete monitoring of the migration process over the GUI. The monitoring shows the number of records in source database to be migrated, number of records migrated successfully, estimated time remaining and time of start for migration. The system is made dynamic so as to add more platform options to migrate.

Given above is the architecture of the system. It works essentially as a 2-tier database system .When the client gives the platforms of the source and target databases, the specified platforms are connected to the system by using the appropriate drivers. The client then gives the name of database to be migrated along with its authentication details. Now, as a connection is established between two platforms at the application layer of the system, a new database is created at the target platform. Now, the phases that take place are

1. EXTRACTION
2. MIGRATION
3. LOADING

These are the same activities which are also carried out during data integration and data warehousing processes. Extraction phase copies one record at a time from the source database into the application layer and passes it to the transformation phase. The transformation phase is actually where the modeling of the data takes place, and creates a new record that is fit to be loaded into the target database. Loading phase is one where the new record generated at the transformation phase is copied into the target database. This process is repeated till every record of the source database is migrated to the target database. The system should be open for adding up new platforms for migration.[6]

VIII. CONCLUSION

Data migration is current an important practice of importing legacy data to a new platform. It is a fairly time consuming process, if done manually. The number of NoSQL databases are increasing day by day [8]. Many of them are open source. Certainly, there is a requirement of a tool which allows migration to database platforms quickly and as per client needs. The automation of this practice certainly will be required to save time and deliver quickly to client's requirement. Also having seen the advent of Internet of Things and other technologies, there is a high need of migration techniques. Since SQL and NoSQL, both databases have their own advantages, it is likely that there is a need to use both databases simultaneously. Hence, we have come up with a solution for migration of data which helps both the databases to exist in correlation with each other. However with NoSQL databases now coming up with features of ACID properties, scalability and distributed nature. Probably, NoSQL will be the platform of choice for most of the clients and organization.

REFERENCES

[1] Schema Conversion Model of SQL Database to NoSQL, P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2014 Ninth International Conference on Gansen Zhao; Qiaoying Lin; Libo Li; Zijing Li IEEE 8-10 Nov. 2014.

[2] "Brewer's CAP Theorem", julianbrowne.com, Retrieved 02-Mar-2010 Brewers CAP Theorem on distributed systems: 2010. Available: http://www.royans.net/arch/brewerscap-theorem-on-distributed-systems/comment-page-1/

[3] S. CouchDB the definitive guide: 2013.Available:http://guide.couchdb.org/draft/consistency.html#figure/3.

[4] Getting started with Cassandra: 2010. Available: http://nosql.mypopescu.com/post/573604395/tutorial-getting-started-withcassandra.

[5] ETL available at http://www.dataintegration.info/etl

[6] Mongify available at http://mongify.com/
[7] Mughees Thesis on Data Migration From Standard Sql To Nosql Muhammad Mughees http://hdl.handle.net/10388/ETD-2013-11-1342
[8] Making the Shift from Relational to NoSQL: 2013. e_Whitepaper Transitioning Relational to NoSQL. http://www.couchbase.com/
[9] NoSQL: 2013. Available: http://en.wikipedia.org/wiki/NoSQL
[10] Making Shift from Relational to NoSql http://www.couchbase.com/sites/default/files/uploads/all/whitepapers/Couchbase_Whitepaper_Transitioning_Relational_to_NoSQL.pdf
[11] MongoDb - The Definitive Guide at http://mongify.com/
[12] Modeling MongoDB with Relational Model - Gansen Zhao , Weichai Huang , Shunlin Liang , Yong Tang

