ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 6, Special Issue 5, March 2017

National Conference on Advanced Computing, Communication and Electrical Systems - (NCACCES'17)

24th - 25th March 2017

Organized by

C. H. Mohammed Koya
KMEA Engineering College, Kerala- 683561, India

Relational to NoSQL Database Migration

Aparna Babu [1], Subu Surendran [2]

M. Tech Student, Department of Computer Science, SCT Engineering College, Pappanamcode, Trivandrum, India[1]

Associate Professor, Department of Computer Science, SCT Engineering College, Pappanamcode, Trivandrum, India[2]

ABSTRACT: Relational databases have been the leading model for data storage, retrieval and management for over forty years. Majority of the data comes in semi-structured or unstructured format from social media, video and emails. RDBMS are not designed to accommodate unstructured data. Also the data size has increased tremendously to the range of petabytes and RDBMS finds it challenging to handle such huge data volumes. NoSQL database with their less constrained structure and scalable schema design has been developed to cover the limitations of relational database. With the increasing maturity of NoSQL databases as well as the situation of reading data more than writing on large volumes of data, many applications turn to NoSQL. This paper discusses about various methodologies for migrating the existing data in relational database to NoSQL database. Also different NoSQL databases are compared based on their structure, performance, scalability, consistency, transactional features and read/write operational characteristics.

KEYWORDS: NoSQL, Relational Databases, Unstructured Data, Data Migration, MongoDB, MySQL

I. INTRODUCTION

Relational databases have been the leading model for data storage, retrieval and management for over forty years. Its success is due to the offering of simplicity to developers as well as robustness and high performance. However, more recently, the traditional RDBMSs have been shown its inability to handle efficiently the demands of a new generation software applications, which are especially focused on unstructured data and massive storage. In order to meet the needs for efficient storage and accesses of large amounts of data, the Database Management System's " Not only SQL" , or simply NoSQL, were proposed. We can observe that, today the companies such as Google, Facebook, Twitter and Amazon have been joined to this new way of massive data computing. In most cases, they are trying to adhere to the demands for scalability, high availability and storage of huge amount of unstructured data. The main characteristics that distinguish the NoSQL model from the traditional RDBMS are partitioning of data and data replication. Despite these benefits, most of the existing large and medium scale software applications are still based on RDBMS due to the challenges associated with the migration to NoSQL database. The first one is the volume of data to be migrated. Organizations, in general, decide to migrate their database when the volume of stored data is huge and the RDBMS are no longer able to satisfy the scalability and high availability expectations. Another challenge is related to the relational database model's manner of avoiding data redundancy, for which it is required to maintain the new data model semantically identical to the original one. Thus, all existing relationships have to be adequately represented without loss or distortion of data. Finally, in addition to all data and model migration, there is a cost associated with adapting software applications to communicate properly with the new database model. Approaches for migration proposed by several persons are discussed in this paper.

There are hundreds of readily available NoSQL databases, and each have different usage scenarios. They are usually divided into four categories, according to their data model and storage: Key-Value Stores, Document Stores, Column Stores and Graph databases[7]. This classification is due to the fact that each kind of database offers different solutions for specific contexts. There has been extensive research in the comparison of relational and non-relational databases in terms of their performance for different applications. However, when developing enterprise systems, performance is only one of many quality attributes to be considered.

II. NoSQL COMPARISONS

Some of the existing NoSQL databases are MongoDB, Hbase, Cassandra .etc. These databases are compared against each other in this section based on their structure, performance, scalability, consistency, transactional features and read/write operational characteristics.

MongoDB is an open source NoSQL database. It exists since 2007 and provides a key-value store that manages collections of BSON documents. Cassandra is also an open source NoSQL database. It exists since 2008 and is designed to cope with large amounts of data spread out across many commodity servers. It provides a structured key-value store using the mechanism of eventual consistency. HBase is a Bigtable-like structured store that runs on top of the Hadoop Distributed File System (HDFS). The key characteristics of MongoDB include schema-less structure, full index support for high performance, replication and failover for high performance, auto sharding for easy scalability and rich document based queries for easy readability and the key characteristics of Cassandra include high availability, incremental scalability, eventual consistency and minimal administration[5]. The key characteristics of HBase are distributed and scalable architecture and strong consistency.

MongoDB and Hbase operates in master slave mode whereas Cassandra operates in master-master mode. When compared based on the quality attributes specified in the CAP (consistency, avaailability and performance) theorem, MongoDB provides consistency and performance, Cassandra provides availability and performance and Hbase provides consistency and availability. MongoDB and Hbase have strong consistency whereas Cassandra is only eventually consistent.

MongoDB is suitable for RDBMS replacement for web applications, semi-structured content management, real time analytics, high speed logging, caching and high scalability but it is not suitable for highly transactional system and applications with traditional database requirements such as foreign key constraints. Cassandra is an effective solution for fast random read/write and flexible parsing/wide column requirement but it is not efficient in case of relational data, transactional operations, primary and financial record, dynamic queries/searching on column data. Hbase works best for optimized read, range based scan and fast read and write with scalability but it is not good for classic transactional applications or even relational analytics. The usage case of MongoDB, Cassandra and Hbase are Craiglist, Twitter and Facebook message respectively.

Konstantinou et.al[8] performed a study based on elasticity of non-relational database and compared Hbase and Cassandra during execution of read and update operations and concluded that Hbase provides high elasticity and fast reads while Cassandra was capable of delivering fast writes. Van der Veen JS et.al [9]compared Cassandra and MongoDB and proved that Mongodb is capable of providing high throughput, but mainly when it is used as a single server instance. On the other hand, the best choice for supporting a large distributed sensor network was considered Cassandra due to its horizontal scalability. Kashyap et.al[10] compared the performance, scalability and availability of HBase, MongoDB, Cassandra and CouchDB and summarized that Cassandra and Hbase shared similar behaviour, but the former scaled better, and that MongoDB performed better than Hbase by factors in hundreds for their particular workload.

Table 1: NoSQL Comparisons

		MongoDB	Cassandra	HBase
Database Model		Document Store	Wide Column Store	Wide Column Store
Key Characteristics		• Schema-less structure • Full index support for high performance • Replication and Failover for high performance • Rich document based queries • Autosharding	• High Availability • Eventual Consistency • Minimal Administration	• Distributed and Scalable Architecture • Strong Consistency
Operating Mode		Master-Slave	Master-Master	Master-Slave
CAP Theorem	Consistency	Yes	No	Yes
	Availability	No	Yes	Yes
	Performance	Yes	Yes	No
Applications	RDBMS replacement for web applications	Yes	No	No

	Transactional Applications	No	No	No
	Realtime Analytics	Yes	No	No
	High speed logging and caching	Yes	No	No
Indexing	Secondary Indexes	Yes	Yes	Yes
	Composite Keys	Yes	Yes	Yes
	Full Text Search	No	No	No
	Geospatial Indexes	Yes	No	No
Distribution	Horizontal Scalable	Yes	Yes	Yes
	Replication	Yes	Yes	Yes
	Sharding	Yes	Yes	Yes
Usage case		Craiglist	Twitter	Facebook messenger

III. MIGRATION METHODOLOGIES

NoSQL data model are different from Relational model according to their structure and the way they store the information. Compared with the NoSQL databases, the structure of the relational databases is more complex in terms of their concept of normalization. According to the rules of normalization they split their information into different tables with join relationship. On the other hand NoSQL databases store their information in a de-normalized way which is unstructured or semi-structured. Therefore the successful migration with data accuracy and liability from Relational to NoSQL would not be an easy task. Some of the recent methodologies for migration are explained in this section.

A. MIGRATION BASED ON ETL (EXTRACT, TRANSFORM AND LOAD)

A methodology for data migration from MySQL relational database to NoSQL database implemented in MongoDB was proposed by Mohammed Hanine [1]. The migration process consists of two steps: Loading the Logical Structure of the Source Database and then Mapping between the Relational Model and MongoDB Model. After obtaining all information about the source and target database, we have to obtain a representation of the relational model of source database. For this we need to get the names of tables, their attributes and relationships. The information on the relationships can be retrieved from the primary constraint and foreign keys for each table. The next step is dedicated to define the mapping between the relational model of MySQL and document-oriented model of MongoDB. For this we have to load the Database tables and their attributes. The main concern while migrating Relational database to NoSQL is the absence of JOINs in NoSQLdatabase. The document model makes JOINs redundant in many cases. If the table to be migrated is join to another table, we have to migrate both the tables to a single table in NoSQL.

B. LINKED LIST BASED MIGRATION

In order to avoid the cross-table query in the NoSQL database, Yu-Lin Zheng[2] proposed a method that follows NoSQLs DDI principles: Denormalization, Duplication and Intelligent keys. In this method, related tables are aggregated into a big table and then most suitable key is selected, which is called row key, to identify each row. MySQL stores all table schemas in a special table called information schema from where we have to extract each tables primary key(pk) and foreign key(fk) and then let each tables primary and foreign key become the head of a linked list. Then we check the foreign key and continuously concatenate the related tables primary and foreign keys into the corresponding linked lists. After parsing all tables, we are able to get the chained length of all linked lists. The Row key

selected must have the highest cardinality. The Row key with the highest cardinality should be the combination of all primary keys with the longest chained length as shown in Figure 1.

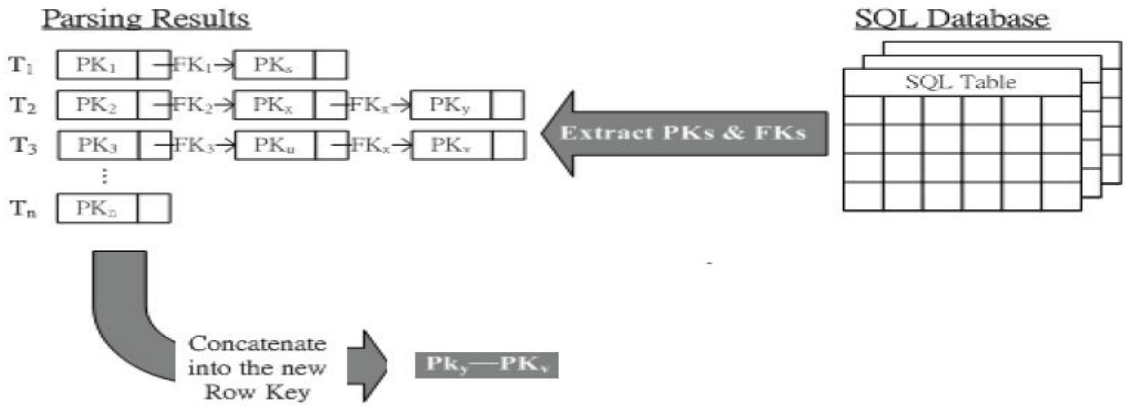

Fig.1: Row key selection after parsing of tables [2]

C. GRAPH BASED MIGRATION

Ganzen Zhao et.al[3] presented a methodology for autonomous SQL-to-NoSQL schema denormalization and migration. Denormalization means to re-aggregate related SQL tables into a big NoSQL table even if some data should be duplicate or redundant in the NoSQL table. Each row should then select one intelligent key called row key for unique identification. A graph model is created to describe general database schema. Through this graph model, schema conversion procedure and a series of extension operations are done to convert original schema into final schema. A graph model is shown in the Figure 2, where T represents table names, A represents attributes, FK represents foreign keys, t1, t2, t3 and t4 represent four tables in this database, (t1,t2), (t2,t3) and '(t3,t4) represent three relations of dependency. When a table reference another table by foreign key, then there exists a table dependency.

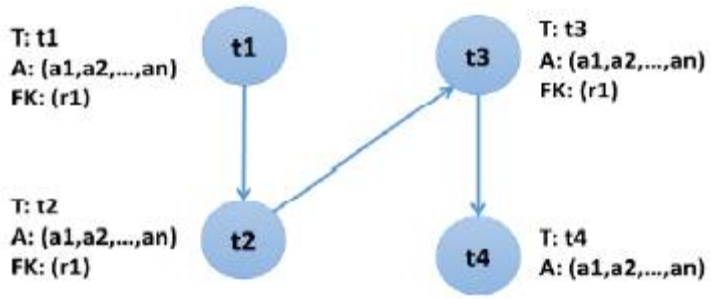

Fig.2: Graph representation of mysql tables and relationships[3]

A node with child in the graph means that it possess incomplete information. In order to gather missing information, the data set needs to integrate data from its children point. We must also make sure that the child node has already possessed complete information as we integrate from that child node, or it might cause some data loss. We first convert all leaf nodes to NoSQL database, and remove the edges that reference to them. This process is called graph transformation. We have to recursively process the graph until all nodes are processed. The edges in the graph are removed through the table extension. Table extension of a table means adding data or extra information into the table. Figure 3 shows the graph transformation. The final nodes in the graph after removing all the edges are the final tables that we have to migrate to NoSQL.

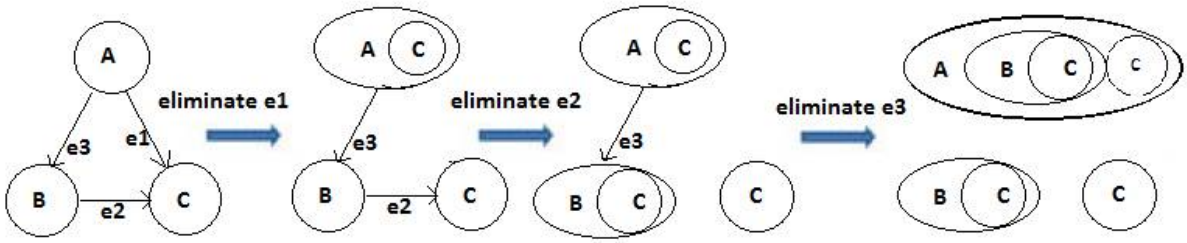

Fig.3: Graph transformation using extension operation [3]

D. A FRAMEWORK FOR MIGRATION AND QUERY CONVERSION

Lefonardo Rocha, Fernando Vale and Elder Cirilo[4] proposed a framework NoSQLayer, capable to support migration from relational (MySQL) to NoSQL DBMS (MongoDB). The framework is divided into 2 main modules. Data Migration Module is responsible for identifying automatically all the elements from the original database, creation of equivalent structure using NoSQL data model, and finally, completely migrate the data. Data Mapping Module is designed to be an interface between the application and the DBMS, which monitors all SQL transactions from the application, translates these operations and redirects to the NoSQL model created in the data migration module. The data migration module starts with the analysis of the relational database and identify all elements of the database and then construct a new schema. The construction of the new schema includes creating a table in MongoDB for each table in MySQL. The records of each table are retrieved and mapped as documents, where each attribute of the tables are represented by fields in these documents. The relationships among the documents are represented by documents references, to facilitate the mapping module in transactions involving table joins. Last step performs the data migration from a relational database to NoSQL which consists of mapping the complete requests on each table of MySQL (select * from table) onto data insertions in the corresponding collections of MongoDB NoSQL database. The goal of data mapping module is to allow a seamless database migration, preventing any change in the application code before changing the data model used so that application developers can continue to create queries in the relational model, but the data will be fetched from a NoSQL database. The first task of this module is to intercept queries issued by the application to the DBMS and forward the intercepted queries to another sub-module in the form of xml files. The xml file will then be evaluated to find the query type and identify what should be the next procedure to be executed, according to the operation type (Select, Insert, Update or Delete). For each operation, a corresponding method is responsible for performing the operation onto the NoSQL database and returning the result of operation.

IV.CONCLUSION

The demand for NoSQL databases is increasing because of their diversified characteristics that offer rapid smooth scalability, great availability, distributed architecture, significant performance and rapid development agility. RDBMSs follow strictly a predefined schema and store their data in different tables through relationship according to the structure of the schema, whereas schema-less NoSQL have a different way for storing and retrieving their flexible, unstructured or semi-structured data available in the different format of databases that include document, key-value, columnar and graph data store group. In this paper, a comparison of different NoSQL databases like Hbase, MongoDB, Cassandra .etc. are explained. Inorder to utilize the advantages offered by NoSQL, we have to migrate the existing data from Relational database to NoSQL database. We have presented an overview of several approaches for performing relational to NoSQL migration.

REFERENCES

[1]. Mohammed Hanine, Abdesadik Bendarag and Omar Boutkhoum, 'Data Migration Methodology from relational to NoSQL Databases' , International journal of Computer, Electrical, Automation, Control and Information Engineering, Vol:9, No:12, 2014.
[2]. Chao-Hsien Lee, Yu-Lin Zheng, ' Automatic SQL-to-NoSQL Schema Transformation over the MySQL and HBase Databases', 2015 International Conference on Consumer Electronics.
[3]. Ganzen Zhao, Qiaoying Lin, Libo Li, Zijing Li, ' Schema Conversion Model of SQL Database to NoSQL' , 2014 Ninth International Conference on P2P, Parallel, Grid, CLoud and Internet Computing, IEEE.
[4]. Leonardo Rocha, Fernando Vale and Elder Cirilo, 'A Framework for Migrating Relational Datasets to NoSQL' , ICCS 2015 International Conference On Computational Science.
[5]. Yishan Li and Sathiamoorthi Manoharan, ' A performance comparison of SQL and NoSQL databases' , 2013.
[6]. Chao-Hsien Lee, Yu-Lin Zheng, 'SQL-to-NoSQL Schema Denormalization and Migration: A study on Content Management Systems' , 2015 IEEE International Conference on Systems, Man, and Cybernetics.
[7]. Jing Han, Haihong E and Guan Le, 'Survey on NoSQL Database' , IEEE 2011.
[8]. Konstantinou I, Angelou E, Boumpouka C, Tsoumakos D, Koziris N, 'On the elasticity of nosql databases over cloud management platforms' , Proceedings of the 20th ACM international conference on Information and knowledge management, 24-28, Glasgow, 2011.
[9]. Van der Veen JS, van der Waaij B, Meijer RJ, ' Sensor data storage performance: Sql or nosql, physical or virtual' , Cloud Computing (CLOUD), 2012 IEEE 5th International Conference, 431-438, IEEE.
[10]. Kashyap S, Zamwar S, Bhavsar T, Singh S, ' Benchmarking and analysis of nosql technologies' , International Journal of Emerging Technologies, Adv Eng 3, 422-426, 2013.