Comparison and Data migration of Relational Database with NoSQL

Ravi Kumar V G[1], Chethan K S[2]
Department of Computer Science and Engineering
[1, 2]Assistant Professor, GSSS Institute of Engineering and Technology for Women Mysuru, Karnataka, India

Abstract*-Relational databases have been the dominant model since 1980's, for storing, retrieving and managing data in computer industry. But relational databases do not fit into the current scenario and losing its importance due to fixed schema requirement and inability to scale well. NoSQL databases were built in the need to deal with the increasing amount of complex data (Big Data), required in real-time web applications, and are mostly addressing some of these points: the focus on availability over consistency, horizontally scalable, distributed architecture, and open-source. The purpose of this paper is to present the reasons for a transition from RDBMS to NoSQL databases, to describe the main characteristics of nonrelational databases and to compare and analyze three popular NoSQL solutions –MongoDB, In this paper we aim at comparing both the database options for various operations namely Insert, Select, Update and Delete for small and large datasets.*

Keywords-NoSQL, Relational vs NoSQL, MongoDB, Data Migration

I. INTRODUCTION

Relational databases are great for enforcing data integrity [4]. They are the tool of choice for online transaction processing (OLTP) applications like data entry systems or on-line ordering applications. RDBMS requires that data be normalized so that it can provide quality results and prevent orphan records and duplicates. It uses primary and secondary keys and indexes to allow queries to quickly retrieve data. But all of the good intentions that the RDBMS has for ensuring data integrity come's with a cost. Normalizing data requires more tables, which requires more table joins, thus requiring more keys and indexes. As databases start to grow into the terabytes, performance starts to significantly fall off. Often, hardware is thrown at the problem, which can be expensive both from a capital endpoint and from an ongoing maintenance and support standpoint.

Database can accommodate a very large number of users on an on-demand basis. The main limitations with conventional relational database management systems (RDBMS) are that they are hard to scale with Data warehousing, Grid, Web 2.0 and Cloud applications, have non-linear query execution time, have unstable query plans and have static schema. Even though RDBMS's have provided database users with the best mix of simplicity, robustness, flexibility, performance, scalability and compatibility but they are not able to satisfy the present day users and applications for the reasons mentioned above.

The next generation NonSQL (NoSQL) databases are mostly non-relational, distributed and horizontally scalable and are able to satisfy most of the needs of the present day applications. The main characteristics of these databases are schema-free, no join, non-relational, easy replication support, simple API and eventually consistent. One of the popular Document-oriented databases is MongoDB [5]. It is part of the NoSQL family of database systems. Instead of storing data in tables as is done in a "classical" relational database, MongoDB stores structured data as JSON like documents with dynamic making the integration of data in certain types of applications easier and faster [2].

The aim of this paper is to illustrate how a problem being solved using MySQL will perform when MongoDB is used on a Big data dataset. The results are encouraging and clearly showcase the comparisons made. Queries are executed on a big data airlines database using both MongoDB and MySQL. Select, update, delete and insert queries are executed and performance is evaluated.

II. MODELING AND QUERYING DATA IN MONGODB

NoSQL databases were developed to deal with such large scale data needs. The term "NoSQL" was first used by Carlo Strozzi in 1998 for his RDBMS, Stozzi NoSQL. Recently, the term NoSQL (Not Only SQL) has been used for databases which don't use SQL (Structured Query Language) as its query language and which don't require fixed table schema.

A key feature of NoSQL systems is "Share nothing" horizontal scaling-replicating and partitioning data over many servers. Due to this feature, NoSQL systems can support a large number of simple read/write operations per second. NoSQL systems don't provide ACID (Atomicity, Consistency, Isolation, Durability) guarantees but follow BASE. BASE is acronym for Basically Available, Soft state and eventually consistent. Basically available means that most data is available most of the time. Soft-state means data is not consistent all the time but will be in eventually consistent state. MongoDb by 10gen, Neo4j by Neo Technologies, Cassandra by Facebook, HBase and Google's Big Table are examples of NoSQL databases. NoSQL solutions are divided into four classes [1].

- Key-Value databases
  In key value data stores, data is stored in the form of keys and values. Each key is unique and keys are used to retrieve the values. The query speed of these databases is higher than the relational databases. Amazon's Dynamo and Riak are famous key value data stores. These are used in applications where schema is continuously evolving.
- Column-oriented databases
  In column oriented data stores data is stored by columns and columns of related data is stored in same file which are called column families. These data stores are mostly used in read intensive applications. Hbase and Cassandra are famous col-umn oriented data stores.
- Document databases
  Document data stores are similar to key value data stores but the value is stored in JSON or XML format. It is used for applications in which data is changed occasionally like Customer Relationship Management System
- Graph databases
  Graph Database uses graph structure with nodes, edges and properties of the edges to store the data. They are suited for the applications in which there are more interconnections between the data like social networks. OrientDB and neo4j are popular open source graph databases.

III. DOCUMENT DATABASES

Document database stores data in the form of documents rather than as normalized relational table in relational databases. Data format of these documents can be JSON, BSON or XML. Documents are stored into collections. The relational equivalent of document and collection are record (tuple) and relation (table).But like relation collection does not enforce fixed schema. It can store documents with completely different set of attributes. Documents can be mapped directly to the class structure of programming language but it is difficult to map RDBMS entity relationship data model. This makes easier to do programming with document databases. There is no need of JOINS in document databases as in RDBMS due to embedded document and arrays. That is why today a growing number of developers are moving to document databases. CouchDB by Apache Software Foundation and MongoDB by 10gen are open source databases built for scalability and ease of use. MongoDB is an open source NoSQL document database, initiated by 10gen Company [6]. It was designed to handle growing data storage needs. It is written in C++ and its query language is JavaScript. MongoDB stores data in the form of collections. Each collection contains documents. MongoDB documents are stored in binary form of JSON called BSON format. BSON supports Boolean, float, string, integer, date and binary types. Due to document structure, MongoDB is schema less. It is easy to add new fields to a document or to change the existing structure of a model. MongoDB offers a technique named Sharding to distribute collections over multiple nodes. When nodes contains different amount of data, MongoDB automatically redistribute the data so that load is equally dis-tributed across the nodes. MongoDB also support Master-slave replication. The slave nodes are copies of Master nodes and used for reads or backups.

IV. DATA MIGRATION FROM MYSQL TO MONGODB

The purpose of data migration i.e. transfer existing data into a new environment, is to preserve all the data in the old system. Database migration tool helps to transform traditional applications into user-friendly applications [7]. Fig. 1 shows the approach.

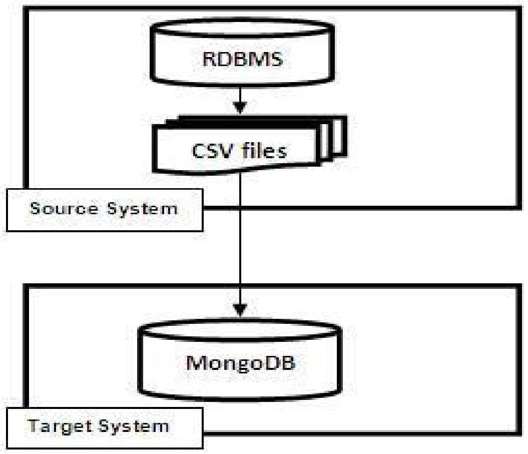

Fig. 1 Flow diagram of migration.

The implementation has two major steps:

A. Extracting the data from MySQL to csv files

Algorithm:
Step1: Select the particular table for migration.
Step2: Generate the csv file for the particular table selected.
Step3: Save the generated csv file in the project folder.

Step4: If successful then go to step 6.
Step5: Else go to step 7.
Step6: Move the generated csv file to the next major step for migrating it to MongoDB.
Step7: Ask the user to once again identify the table for migration.
Step8: End.

B. Dumping the extracted data in the csv files to MongoDB

Input: csv file
Output: Collection of migrated data in MongoDB. Algorithm:
Step1: Start the MongoDB server.
Step2: Start the MongoDB client.
Step3: Set the path of the MongoDB bin file.
Step4: Import the data from csv file.
Step5: Display the imported data to user.
Step6: End.

After executing step A we can view the csv files and the data in the MongoDB after the execution of step B.

V.COMPARISON BETWEEN MYSQL AND MONGODB

A huge airlines database [11] with 1050000 records is considered. The attributes are Year, Month, DayofMonth, DayOfWeek, DepTime, CRSDepTime, ArrTime, CRSArrTime, UniqueCarrier, FlightNum, TailNum, ActualElapsedTime, CRSElapsedTime, AirTime, ArrDelay, DepDelay, OriginDest, Distance, TaxiIn, TaxiOut, Cancelled, CancellationCode, Diverted, CarrierDelay, WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay. The table/ collection is named as project 2016.

A. Relational table insert vs. MongoDB insert

In MongoDB,

db.project2016.insert ( );

In SQL,

Insert into project2016
Values ( );

B. Relational table select vs. MongoDB select

In MongoDB,

db.project2016.find ();

In SQL,

select * from project2016;

C. Relational table update vs. MongoDB update

In Mongodb, db.project2016.update ( );

In SQL,

Update from project2016 set deptime=”957”;

D. Relational table delete vs. MongoDB delete

In MongoDB, db.project2016.remove ( );

In SQL,

delete flightno from project2016 where deptime=”957”;

VI. QUERYING MONGODB

As data modeling is important, it is also important to know how queries are executed in MongoDB. This section describes how queries are written in MongoDB. Six Queries are designed to describe syntax in MongoDB and to show how same queries are written in MySQL [2].

Query1 :	Find the tag names which have been used in the post under which a particular user has commented. (user=’a1’)
MySQL	mysql> select t.name from comment c,post p,user u,tag t where p.pid=c.postid and c.userid=u.uid and p.pid=t.pid and u.username='a1';
MongoDB	> var u=db.user.findOne({username:"a1"}) >var tag1=db.post.find({"comments.userid":u._id }) > db.tag.find({_id:tag1.tagid},{_id:0,name:1})

The find () method selects documents from a collection that meets the <query> argument. <Projection> argument can also be passed to select the fields to be included in result set. The find () method returns a cursor to the results. This cursor can be assigned to variable.

db.collection.find(<query>,<projection>)

The findOne() method is similar to find() method but it selects only one document from a collection.

In this query first the document with username ‘a1’ is extracted from User collection and stored in variable u. Then the comments corresponding to that user are found by matching u. id field with userid field of subdocument comments (com-ments.userid) in Post Collection and returned cursor is stored in the variable tag1. Then _id is matched in Tag collection. To exclude _id field {_id:0} is written in projection field and the fields which needs to be included like here name is to be in-cluded is written as {name: 1} in projection argument .

Query2	Find the tag(s) which have been used in the posts of each user.
MySQL	mysql>select u.uid,t.tagid from user u,post p,tag t where u.uid=p.uid and p.pid =t.pid;
Mongo DB	db.post.find({},{_id:0,uid:1,tagid:1})

Query3:	Given a cid of a comment find its related post, parent comment and tags associated with the post.
MySQL	mysql> select c.cid, c.postid, c.parentid, t.tagid from comment c,post p ,tag t where c.postid=p.pid and p.pid=t.pid and c.cid='c2';
Mongo DB	db.post.find({"comments.cid":"c2"},{"comments.cid":1, "comments.parentid":1,tagid:1});

Query4:	Find the time of the post under which some user has commented. (user='a1')
MySQL	mysql> select p.time from comment c,post p,user u where p.pid=c.postid and c. erid=u.uid and u.username='a1';
Mongo DB	> var u=db.user.findOne({username:"a1"}) > db.post.find({uid:u._id},{time:1,_id:0})

Query5:	List the Posts of each user.
MySQL	mysql> select u.uid,p.pid from user u,post p where u.uid=p.uid;
MongoD	db.post.find({},{_id:1,uid:1})

Query6:	Find the total number of posts by a particular user. (uid =102)
MySQL	mysql> select count(p.pid) from post p,user u where u.uid=p.uid and u.uid=102;
MongoDB	> db.post.count({uid:102})

VII. CONCLUSION

In this paper data modeling in MongoDB has been shown by using Class diagram and JSON format. It also has been shown that how queries are written in MongoDB. MongoDB does not use JOINs to relate documents like Relational Databases. If it is required to use medium data without complex queries and normal day to day functioning, then MySQL is a better but if the data is non-relational and may involve complex queries and joins if used in SQL, then MongoDB gives better performance for big data.

REFERENCES

[1] Arora, Rupali, and Rinkle Rani Aggarwal. "Modeling and querying data in mongodb." International Journal of Scientific and Engineering Research 4.7 (2013).

[2] Aghi, Rajat, et al. "A comprehensive comparison of SQL and MongoDB databases." International Journal of Scientific and Research Publications 5.2 (2015).

[3] G. Eason, Lara Nichols,"A comparison of object-relational and relational databases", presented to the Faculty of California , chapter 4, pp. 6-7.

[4] Jae Jin Koh,(3-6 October, 2007), Relational database schema integration by overlay and redundancy elimination methods, in International Forum on Strategic Technology(2007), Institute of Electrical and Electronic Engineers, IEEE Computer Society.

[5] https://en.wikipedia.org/wiki/MongoDB . Last Accesses on: Dec 30 2016

[6] "MongoDB" http://www.mongodb.org/. Last Accessed on: Dec 30, 2016

[7] Department of Education Office of Federal Student Aid, Data Migration Roadmap, "A Best Practice Summary", pp.5-6

[8] Satyadhyan Chickerur, "Comparison of Relational Database with Document-Oriented Database (MongoDB) for Big Data Applications", 8th International Conference on Advanced Software Engineering & Its Applications, 2015

[9] Strozzi, Carlo: NoSQL-A relational database management sys-tem.2007-2010-http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql /Home%2520 page

[10] R. Cattell. Scalable sql and nosql data stores. SIGMOD Rec., 39(4):12–27, May 2011.

[11] "The Airline Data Set; http://stat-computing.org/ dataexpo/2009/.