A Review to the Approach for Transformation of Data from MySQL to NoSQL

Monika¹ and Ashok²

¹M. Tech. Scholar, Department of Computer Science and Engineering, BITS College of Engineering, Bhiwani, Haryana (India)
²Assistant Professor, Computer Science and Engineering Department, BITS College of Engineering, Bhiwani, Haryana (India)

Abstract
Relational databases have been the dominant model for the several decades, for storing and retrieving data. But in some cases relational database has shown its age and losing its importance due to its dependence on fixed schema and inability to scale well. With the development of the internet and cloud services like Google and Amazon volume of data is increasing exponentially. Data accumulated by social services such as Facebook and Twitter is more connected and semi-structured. Today’s Web and mobile applications are designed to support large number of concurrent users by spreading load across a collection of servers. Traditional relational databases are facing many challenges to contend with these trends. NoSQL (Not Only SQL) databases are developed as a solution to these problems. NoSQL is an umbrella term for all data stores that do not follow RDBMS principles. They do not require fixed schema nor do they use the concept of joins. NoSQL databases are horizontal scalable, flexible to handle semi-structured data and support high availability. NoSQL databases are classified into four classes: Key-value data store, Column oriented data store, Document database and Graph database. These data stores are appropriate for different applications. Key-value data stores are used to handle massive loads. Column-oriented data stores are used to store large amount of data which is distributed over many machines. Document databases are used to store semi-structured data, whereas graph databases are used to store data elements interconnected by many relations. This thesis provides an in-depth knowledge of NoSQL classes, and different tools available of these classes.

Keywords: MySQL, NoSQL, MongoDB, Data Transformation.

Introduction
Database is organized collection of data and the term database management system is used to describe a variety of software systems that allows database functions of storing, retrieving, adding, deleting and modifying data. There are different types of systems as well as models that perform database functions in different ways. The fundamental differences of more significant models are discussed here. Traditional relational databases have been around for decades. Recently, many new NoSQL systems are emerging, which have renewed interest in non-relational storage models. These storage models have been discussed here:

Relational model have been used as the most common data storage model for several decades. The term was originally defined by Edger Codd at IBM Almaden Research Center in 1970 [1]. The software which implements relational model is known as relational database management system or RDBMS. Modern relational database management systems are based on the concept of Codd’s relational model. Examples of RDBMS are MySQL, Oracle, PostgreSQL, Microsoft SQL server. In relational database management systems data is organized into tables consisting of rows and columns. In Codd’s model the table is referred to as relation, columns are referred to as attributes and rows are referred to as tuples. The relation is basic element of relational model. Relation consists of set of tuples having the same attributes. One tuple represent an object and gives information about that object. To distinguish all the tuples in a relation, the set of one or more attributes are chosen as a primary key. To relate two relations, relational database use the concept of foreign key. A foreign key in a relation refers to the primary key in other relation. Compared to primary key, foreign key need not be unique.

Although relational databases have been widely used in industry for many years, there exist some non-relational databases also. Three of these non-relational databases: Object-Oriented databases, XML databases and NoSQL databases. Nonrelational databases are: Object Oriented, XML, NoSQL.
Objectives
1. To study in detail relational and various NoSQL databases and compare them.
2. Installation and hand-on experience on tools available for Document-based data stores.
3. To propose a technique for transformation of data from relational database to document-based NoSQL database.
4. Implementation of the proposed technique using NetBeans Java IDE and analyze the results.

Problem Statement
With the advent of Web 2.0 applications with millions of users reads and writes, a more scalable solution is required. The data stores for these applications need to provide good horizontal scalability. Relational databases cannot solve these problems as they are not horizontal scalable. Similar issues are involved with cloud services, social networks, mobile usage and social media. Due to cloud services data is growing fastly and the data accumulated by social networks is more connected and semi-structured. So, there is need of the database that can store and process big data effectively and can satisfy the demand for high performance while reading and writing. Relational databases have many problems to cope with these trends. Technologies such as NoSQL were developed to meet the reliability and scalability needs. A growing number of developers and users have begun to use NoSQL databases. With this usage, there arose need to transform the existing applications from relational databases to NoSQL databases so that they can run on such platform. Document-oriented databases are one of the categories of NoSQL databases that are appropriate for web-applications which involves storage of semi-structured data and dynamic queries. MongoDB document databases are able to face these new challenges as they allow horizontal scalability, support high-availability and have the flexibility to handle semi-structured data. MongoDB has typical applications in content management systems, mobiles, gaming and achieving. Most of the transformation systems that have been developed till now, transform the data from relational to other non-relational databases (object-oriented databases and XML databases). However, to the best of our knowledge, no work has been done to transform a relational database into NoSQL databases. In this thesis a novel approach has been proposed for the transformation of relational database into document-based class of NoSQL database.

Database Design
Relational databases have been used for data management for decades. But NoSQL databases have gained popularity in the recent years. Now a growing number of organizations and users are using NoSQL databases. They are migrating from relational databases to NoSQL databases. So, there is need to transform the data from relational databases to NoSQL databases. It also needs to be known how data is modeled and queried in NoSQL databases. In the following sections, modeling and querying in NoSQL database is shown. MongoDB document database is chosen as NoSQL database. A transformation algorithm is also proposed. The proposed algorithm is implemented by using NetBeans IDE. For this purpose, MySQL is selected as relational database and MongoDB as document-oriented NoSQL database.

MySQL
MySQL is most popular open source relational database management system. It is owned by Oracle. It is used in a variety of projects and is stable. MySQL is fast, robust, easy to use, multi-user and multi-threaded SQL database server. MySQL support ACID transactions and foreign keys. Because of its simple installation and setup procedure, it is used by small companies as well as large production environments such as Twitter. In Twitter, it is used in the way of a key value store. MySQL is written in C and C++. It has drivers available in most programming languages that allow programs to access the API.
An interactive shell is provided for SQL queries. The programs that use the provided native drivers can also use SQL to interact with the database. The relational schema of database used as example for implementation and design. The database is based on Post-Comment system. A user can post any number of posts at any time on the site. User can also give comments and sub-comments under any post. Tags are used to classify the posts. The database consists of four tables: User, Post, Comment and Tag. User table attributes are uid, username, realname, email, homeurl, pswd. Uid is primary key. The user can login into the system by using username and password. So, user must have valid account. Homeurl is the url of its home page. Post table has pid, uid, title, body, primary url, and time attributes, where pid is the primary key of the post table. Uid is foreign key that refers to the primary key of the user table. Any user can post an article without having valid uid. For an outsider, uid is null. Attributes of comment table are cid, parentid, postid, userid, title, comment, time, score, descriptor, where cid is the primary key of table. Nested comments are also allowed. Nested comments are comment under a comment. Postid is the id of the post under which user has commented. Parentid is the id of the comment under which sub-comment is written. Attributes of tag table are tagid, name, pid. Tagid is primary key. Pid is foreign key which refers to the pid of the post table. Tags are used to categorize the posts.

**MongoDB**

MongoDB is an open source document-oriented NoSQL database. MongoDB stores data in the form of collections. Collection contains documents. Documents are stored in the binary JSON format (BSON). Compared to relational database, collection corresponds to table and document corresponds to relational tuple. However, unlike relational database, collection in MongoDB do not enforce fixed schema. Documents in the same collection do not need to have the same set of fields or structure. Common fields in a collection’s documents may hold different types of data. MongoDB does not use Joins to relate documents like Relational database. In MongoDB data is stored in the denormalized form in which related data is stored in single document. This is known as Embedding. Embedding provides good performance.
for read-intensive applications, as related data can be retrieved in single database operation. The problems with fully embedded structure occur while writing documents to the collection. The need to store the data in different documents arises when data starts to duplicate during embedding. To represent many-to-many relationships data is stored in different documents. To indicate the relationships between the data represented by documents, a reference is stored between two documents. Referencing provides more flexibility than embedding. The database used as example, which has to be transformed from MySQL to MongoDB contains three collections: User, Tag and Post. Post contains an embedded document named comments. Class diagram and JSON format is used for modelling schema of the database.

Class Diagram Representation

Class diagram can be used to represent the schema in MongoDB. Documents are represented by classes. Embedded documents and referencing between documents can be represented by relationships between the classes. Composition is used to represent the embedded documents and associations are used to represent the referencing. Document fields are represented by class attributes.

Figure 2: Class Diagram Representation of Sample Database

Representation of database used as example. Post and Comment are embedded documents represented by composition. Comment exists only if the post corresponding to that comment exists that is comment is embedded into the post. Post, User and Tag are different documents. To relate these documents reference is used. In post document, uid is used as reference field of user document and tagid is used as reference of tag document to indicate a relationship between documents. In user document pid is used as reference of the post document and cid is used as reference of comment document. In comment document userid is used as reference of the user document and in tag document pid is used as reference of the post document.
Implementation

The proposed algorithm has been implemented by using NetBeans Java IDE. For this purpose, MySQL is chosen as relational database. Data is transformed from relational database (MySQL) to document database (MongoDB).

The steps involved in the transformation are explained below:

Step 1: In first step, connection with the relational database (i.e. MySQL) is established. Once the successful connection has been made to the MySQL server, then all the existing databases from relational database are extracted.

Then user chooses the database which needs to be created in MongoDB. That is the database, whose data is to be transformed into MongoDB is chosen. Here, project is chosen from existing databases for transformation. The chosen database project has four tables: comment, post, tag, and user.

Step 2: From the table list extracted in step 1, the tables whose corresponding embedded documents are to be created in MongoDB, are chosen. In MongoDB, related data of two or more tables can be stored in a single document. This is known as embedding and documents are known as embedded documents.

Comment and post are chosen as embedded document. To choose comment and post, the tables are first selected and then “select tables for embedding” button is clicked. The tables are displayed into the second list.

Step 3: Comment and post are shown into the second list. From these two documents choose the one in which other document has to be embedded. Here post is chosen as embedded document, it means post contains comment. To choose the document, first the document is selected and then button “Embedded table” is clicked and the table name is shown in the list.

Step 4: In this step, the columns of embedded table (post and comment, which are chosen in second step) on which Join has to be applied are chosen. Joining of tables is required to generate a single embedded document corresponding to two tables. For this purpose, first “Select columns for joins” button is clicked and then the columns of both the selected embedded tables are shown into the combo box. Columns for Joins are chosen from these columns. Here pid column of both the tables is chosen for Join.

Step 5: Pentaho takes the text files (in a specific format) as input to write the data into the MongoDB. This file should contain column names and column values. So, first these files need to be generated. To generate the text files “Generate Text File” button is clicked. The text files corresponding to tables are generated separately but only one file is generated corresponding to embedded documents. The information of all the generated files.

Here, table name is the name of the table, whose files (reference file and text file) are generated. Reference file path is the path of file used for referencing that is where the file is placed. Reference column names are the columns, which exist in the generated reference text file. Generated text file is the file corresponding to which collections are to be created in MongoDB. Files path and column name (attributes of the documents) are shown into the text area.

Step 6: Once the files are generated, the next step is to load the data into MongoDB. Pentaho is used for this purpose. To integrate with pentaho, “integrate” button is clicked. Pentaho gets opened. With the help of Pentaho, data is written into MongoDB. Files which are generated above are given as input to the pentaho. These files are in the format required by Pentaho. It will accept these files and create the collections and load the data into the collection.

Conclusion

NoSQL databases have many advantages over relational databases and more and more organizations are opting for NoSQL databases. Among all types of NoSQL databases, document databases are more appropriate for semi-structured data storage and document databases are highly scalable. In comparison to another document databases MongoDB database is faster and queries in MongoDB are also simpler. So, MongoDB has broader scope than other databases. The proposed technique is effectively transforming data from relational database to document based database.

Future Scope

The proposed technique transforms the data from relational database to document based NoSQL class. This work can be extended for other classes of NoSQL databases also. The presented work has semi-automatic post algorithm steps. The work can be extended in this aspect to make transformation fully automatic.
References


