DOI 10.29042/2018-3701-3704

Helix Vol. 8(5): 3701- 3704

Data Migration From SQL to MongoDB

[1]Nuzhat F. Shaikh, [2]Aditya Jadhav, [3]Chetan Raina, [4]Gaurav Nagoshe, [5]Suraj Kale

[1,2,3,4,5] Dept. of Computer Science and Engineering, Modern Education Society's College of Engineering, Pune

Email: [1]nfshaikh@mescoepune.org, [2]adi98jadhav@gmail.com, [3]chetan5896@gmail.com, [4]gaurav23091996@gmail.com, [5]surajkale6696@gmail.com

Received: 09th July 2018, Accepted: 14th August 2018, Published: 31st August 2018

Abstract

As the Internet related technologies have emerged, a lot of changes were observed in the data collected, stored and processed by the organizations. The new incoming data was unstructured, available from a number of outside areas and accumulated in large volumes within a small time span. It was realized the traditional system is inefficient to handle this data. Solution to that is NoSQL type database. The databases that belong to this category are highly scalable, powerful and efficient. But the problem that occurs with these databases is they don't use the traditional SQL (Structured Query language) to query the data. Hence it is difficult to migrate data from existing traditional databases to the NoSQL databases and develop efficient and optimized functions to query the data under these NoSQL databases. Various tools are available which have the capabilities to address this, but none of them actually provide a complete integrated system for efficiently doing this. This paper proposes an architecture which will provide a GUI Application that will provide the users with a click and use Querying feature with Data Migration and Automatic Query Conversion. MongoDB is the most slanting database for NoSQL, therefore our proposed system focuses on these perspectives with reference to MongoDB.

Keywords: Schema, MongoDB, RDBMS, Migration, Query Syntax, Relational Model

Introduction

The recent increase in the use of the internet and social media sites has emerged as a problem for the relational databases. It has been observed that the relational databases have not been able to scale up with the increased use of databases. The various factors involved such as cost and performance help in determining how the data should be stored and managed. The traditional SQL based databases are preferred for their availability and fault tolerance. But these can be very expensive to maintain and difficult to scale. This has led to corporations migrating from relational databases to NoSQL databases. Cheap hardware and easy scalability are the factors that have favored the migration to these databases. Most of the data on social networking sites and clouds is unstructured and may vary in nature. This data also takes up huge space and requires high availability and scalability. These same tasks can still be accomplished with the traditional RDBMS databases but at a very high cost which makes the NoSQL databases the popular choice as they are comparatively cheaper [10]. MongoDB is a Document based database of NoSQL type that provides us with increased availability and scaling as its major features. It stores the data in JSON format and provides JavaScript functions to query that data. The syntax as well as the data layout being completely different, it introduces the overhead of acquiring efficiency in JavaScript querying and converting queries from all web programs from SQL to MongoDB. The proposed system focuses on 2 major aspects of the database world.

Writing efficient queries can be done only by individuals who have been working on databases from a sufficiently long time and have developed a sense of eminence in the query language. This surely becomes more burdensome if the database to be used does not support the conventional SQL for querying the data. In case of MongoDB, functions of JavaScript are used for querying the data and further a lot of new functions are developed regularly by the MongoDB community. These new functions surely help with the efficient querying of data but for only those who are familiar with this database. The expectation that an entire stack of developers would be efficient in optimized MongoDB querying is not an ideal situation. Hence this paper proposes a Graphical User Interface Application which will eliminate the need to know syntaxes and available method. Moreover the system will also provide us with automatic query correction that we use to minimize the errors by the user.

An essential focus area that prevents companies from switching to powerful databases like MongoDB is the overhead involved in the process of migration. Migration is referred to as the process of transferring data from a database from one server to another or from one database architecture to another.

There are various algorithms available for migrating an SQL database to MongoDB. But the user may find difficulty in implementing the functionality of SQL queries on the MongoDB databases. If a large scale company wants to migrate, it will have to invest a huge amount of money in the man power for query

conversion and the process may also take considerable time. We intend to solve this problem by automating this process through our system. The web server will provide an interactive and graphical utility to convert SQL queries to MongoDB queries. This helps a user to implement his operations on the database even if he is unfamiliar with the MongoDB database.

Literature Survey

Cloud suppliers offer their framework, stage and programming to clients who discovered distributed computing as a fitting answer for their necessities. The organizations who need to pick a cloud supplier ought to notice that subsequent to building up their own particular programming in a particular cloud condition, the relocation to another cloud condition would be troublesome or it may be incomprehensible. Cloud databases are in charge of putting away information in an adaptable and high accessible route in cloud situations [1].

NoSQL databases have risen as the answer for handling substantial amounts of client created substance as yet ensuring adaptation to internal failure, accessibility and versatility. Each NoSQL database offers separated properties and qualities and additionally unique information models and structures. Thus, the advancement of users misusing such sort of innovation is entirely reliant on the particular NoSQL arrangement being received, and the relocation from a NoSQL to alternate requires the improvement of specially appointed code dealing with the exchange of information. A relocation framework for columnar NoSQL databases helps to fulfill it. This approach depends on an original meta-model, fit for saving both solid and feeble consistency between information refreshes, auxiliary files and different information sorts. In addition, the approach enables designers to effectively include bolster for new databases [2].

Qing Wang et al. [3] examine information relocation essentials from a hypothetical point of view. Following the system of unique elucidation, we initially examine models and schemata at various solidness, accessibility, question bolster, and different measurements. These frameworks ordinarily give up some of these measurements, e.g. extensive exchange consistency, to accomplish others, e.g. higher accessibility and adaptability.

Mayuri Sadaphule et al. [6] present an interface to implement database queries in spreadsheets. A query conversion tool is used for the translation of SQL queries. Querying data from databases can be difficult for non-technical users. The interface helps such users to use spreadsheets for this purpose that is a more familiar technology to them. This way the users can benefit from the scalability and availability provided by the databases.

levels of reflection to build up a Galois association amongst theoretical and solid models. An inheritance piece is found at an abnormal state deliberation which combines heterogeneous information sources in a heritage framework. At that point it is demonstrated that relocation changes can be determined by means of the arrangement of two subclasses of changes: property-safeguarding changes and property-improving changes. By characterizing the thoughts of refinement accuracy for property-protecting and property-improving changes, a formal system is built up for refining changes happening during the time spent in information movement.

Distributed computing has as of late developed as another figuring worldview empowering on-request and versatile arrangement of assets, stages and programming as administrations. Keeping in mind the end goal to fulfill distinctive capacity necessities, cloud applications typically need to get to and collaborate with various social and NoSQL information sources having different APIs. This APIs heterogeneity incites two primary issues. In the first place it binds cloud applications to particular information stores hampering along these lines their relocation. Second, it expects designers to be comfortable with various interfaces. The paper proposes bland assets characterizing the diverse idea utilized as a part of each kind of information store. These assets are overseen by ODBAPI as a streamlined and a bound together REST API empowering to execute CRUD operations on various NoSQL and social databases [4].

Rick Cattell [5] inspects various SQL and so called "NoSQL" information stores intended to scale basic OLTP-style application stacks over numerous servers. Initially persuaded by Web 2.0 applications, these frameworks are intended to scale to thousands or a great many clients doing refreshes and in addition peruses, as opposed to customary DBMSs and information stockrooms. The new frameworks are differentiated on their information show, consistency systems, stockpiling components,

The query conversion tool is used for translating SQL queries in the .xlsx format. The tool automatically queries the data from the tables in the databases and outputs them in column form for implementation in spreadsheets. First the query is translated into relational algebra expression. Then the relational expression is translated into spreadsheets according to the operator [7].

The tool can also be used to perform the validation and correction of the queries entered by the user. Then a parse tree is generated according to the database schema selected by the user. The system then identifies the SQL Query and converts it into the form of a Relational Algebra Expression. The graph is

implemented by using BFS and DFS algorithms for traversal [8].

Proposed System

The system that we propose can be subdivided into an online database application and a query conversion utility. The client application acts as an environment that allows the users to select and convert the databases from SQL to MongoDB. To fulfill this purpose, the process of querying is expected to be in a hierarchical form starting from databases and ultimately lowering down to Collections, Documents, Criteria and Projections. The structure and data types of the keys will also be considered. The query conversion utility provides the user with graphical user interface that allows him to choose from some basic predefined SQL queries or write his own SQL query. This way the user can implement his queries even if he is not familiar with the MongoDB database.

Also, the system is expected to detect syntax errors in the queries before execution. These errors will be automatically corrected by the utility after which the queries will be executed. Fig.1 shows the block diagram of proposed system.

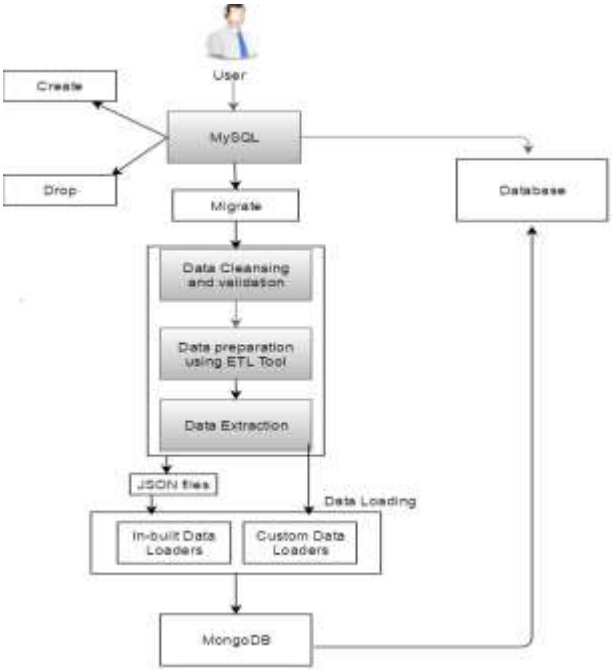

Fig. 1: Block Diagram

Mathematical Model

Data migration is the process of transforming data modeled by schema of source database D1 into target database D2,

Input = {D1, D2}

Where D1 = source database RDBMS

D2 = target database MongoDB

T = {T1, T2, T3……Tn}

Where T is the set of available tables in the database D1

M1 = {Pk, Fk, Row, Column, Records}

where M1 is the metadata of database D1

M2 = {id, Document, Collections, fields}

Where M2 is the metadata of database D2

f = M1->M2 this function is used to map the two sets of metadata

Collection->Tables

Document-> Row

Field->Column

Output = D1->D2

Query Conversion

Let S be the Whole System Consisting of

S = {I, P, O}

I = Input

O = output

I = {U, Q, D}

O= {Qm, Dm}

U = User

U = {u1, u2….un}

Where Q is the Query Entered by user

Q = {q1, q2, q3…qn}

D = {d1, d2, d3…dn}

Where D is the Dataset on which we perform the query conversion

Qm is the converted query in MongoDB

Output = Data retrieved from the MongoDB database

System Implementation

Data Migration

The process of transferring data from the entire database of one server to another or from one database architecture to another is known as data migration. Here we migrate RDBMS Database into MongoDB. First step is to choose the data that we need to migrate and get the metadata of that database. Metadata is nothing but the data which gives the information or describes the other data. Metadata has the information about the primary key and foreign key information from which it generates the relationship of that RDBMS database and find the joins that maintain the relationship of primary and foreign key. The second step is to convert the RDBMS data into JSON format.

Query Conversion

Query conversion is the process of converting SQL queries into MongoDB format. Thus without writing a MongoDB query we can fetch the data from MongoDB database. The benefit of this query conversion utility is that user does not need to know about the MongoDB syntax. The conversion of queries is hosted on the server containing the two databases.

Conclusion

The demand of NoSQL databases is increasing rapidly because of their rapid scalability and distributed architecture. Now we have a GUI that allows to implement SQL queries on migrated databases. SQL to MongoDB query conversion is now possible and getting of data from MongoDB database through SQL query is possible. The methods through which relational databases and NoSQL databases manage their information are entirely different. In RDBMS, the schema is fixed and data is stored in the form of tables where relationships may exist between multiple tables, while NoSQL databases do not follow a definite schema. In NoSQL, data is stored in an unstructured form and in a varied format of databases that includes documents, key-value pairs, columns and graphs [9].

Recently a lot of enterprises have started to migrate from SQL to NoSQL databases. The main aim of our system is to provide an environment to these enterprises to migrate data from relational database to the NoSQL data store. The data migration allows enterprise's Online Analytical Processing (OLAP) which is a significant part of the broader category of Business Intelligence [12]. The methods used by relational and NoSQL databases to store data are absolutely different and this produces a challenging task for corporations to migrate between the two databases.

The difference in the structure and nature of the RDBMS and NoSQL database make the migration process difficult. From the different choices available we have selected migration from MySQL in the SQL group to MongoDB in the NoSQL group as our test case.

References

[1] M. Shirazi, H. C. Kuan, and H. Dolatabadi, "Design patterns to enable data portability between clouds' databases," in ICCSA 2012. Salvador: IEEE, Jun. 2012, pp. 117–120.

[2] M. Scavuzzo, E. Di Nitto and S. Ceri, "Interoperable data migration between NoSQL columnar databases," in EDOCW 2014. Ulm: IEEE, Sep. 2014, pp. 154–162.

[3] B. Thalheim and Q. Wang, "Data migration: A theoretical perspective," Data and Knowledge Engineering, vol. 87, pp. 260–278, 2013.

[4] R. Sellami, S. Bhiri, and B. Defude, "ODBAPI: A unified REST API for relational and NoSQL data stores," in Big Data Congress 2014. Anchorage: IEEE, Jun. 2014, pp. 653–660.

[5] R. Cattell, "Scalable SQL and NoSQL data stores," SIGMOD Rec., vol. 39, no. 4, pp. 12–27, May 2011.

[6] M. Sadaphule and N. F. Shaikh, "A Survey: A Tool for Database Query Translation into Spreadsheets", International Journal of Engineering Science and Technology (IJEST), Vol. 7 No.11, pp 401-406 ", ISSN : 0975-5462, Nov 2015

[7] M. Sadaphule and N. F. Shaikh, "An Application of Database Query Translation into Spreadsheets", IEEE International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) - 2016, 2nd-4th March 2016

[8] M. Sadaphule and N. F. Shaikh, "SQL Query Parser: An Automated Tool for Translating the Queries into Spreadsheets ", IJCSIS International Journal of Computer Science and Information Security, IJCSIS, Pittsburgh, PA, USA ESCI - IP & Science - Thomson Reuters - Web of Science. Impact Factor 0.519 Vol. 14 No. 8ISSN 1947-5500August 2016

[9] Z. Chen, S. Yang, H. Zhao, and H. Yin, "An objective function for dividing class family in NoSQL database," in CSSS 2012. Nanjing: IEEE, Aug. 2012, pp. 2091–2094.

[10] H. Dharmasiri and M. Goonetillake, "A federated approach on heterogeneous NoSQL data stores," in ICTer 2013. Colombo: IEEE, Dec 2013, pp. 234–239.

[11] L. Rocha, F. Vale, E. Cirilo, D. Barbosa, and F. Mourao, "A framework for migrating relational datasets to NoSQL," Procedia Computer Science, vol. 51, pp. 2593–2602, 2015.

[12] K. North, "The NoSQL alternative," InformationWeek, no. 1268, pp. 33–35; 38–39, May 2010.

