Web Application for RDBMS to NoSQL Data Migration and Query Conversion

Manisha Jadhav[1] R. R. Badre[2]
[2]Associate Professor
[1,2]Department of Computer Engineering
[1,2]MIT Academy of Engineering, Alandi, Pune, India

Abstract— With the exponential growth of internet related data, the traditional databases are not enough to handle this huge amount of data. To store and collect data from the internet some changes observed. The newly incoming data was unstructured available from the external resources. Due to the large data, the relational database is not capable of handling the unstructured data, therefore, lots of organization moving toward the NoSQL databases. But the problem with NoSQL database they don't use the structured query language for writing the query. Lots of tools are available in the market that provides data migration facility like Mongify, MongoBooster, Datacenters and Apache sqoop. The problem addressed with this database is that they do not provide the graphical user interface to perform operation easily and efficiently. The proposed web application provides a graphical representation for the one click migration and automatic query conversion with the help of simple click function. The proposed system migrated traditional database to newly available database (MongoDB). Linking process is used to reduce the data migration time.

Key words: MongoDB, RDBMS, Migration, Query Syntax, Relational Model, Schema

I. INTRODUCTION

The relational model having the primary set of the operator like union, intersection, selection, projection and join. Relational databases totally depend upon the relations between the attributes which have the primary and the foreign key. Relations are represented using the tables and the views in the relational database tables having the physical representation of relations and views having the virtual representation of relations. In the relational databases, each row in a table represents as an entity and table represent the group of all entities. The property of the entities represented by using column related to each row. RDBMS is the structured data SQL language is designed for the structural data. SQL Consist of DML language and DDL Language, select, insert, delete and update are the representative operations in the SQL. The Select operation specifies the entities that represent the properties related to the user interest. The join operation is used to join or link to entities and generate the new entities that match the query entered by the user [1].

A. RDBMS and NoSQL Comparison

Due to the exponential growth of internet technology the size of data is increasing and traditional database is not able to handle this data. A relational database is famous to manage the data in the practical scene, but it faces the problem when the data are huge. Relational databases are highly used to store and retrieve data from the application, when the size of data is not large then relational database work best. The Relational database is not suitable for handling a huge data like the internet. NoSQL database is not capable of handling a large amount of data. NoSQL stands "Not Only SQL" this was introduced in 2009. The main advantage of the NoSQL database is that they handle unstructured data like e-mails, documents, social media and multimedia competently. The common features of the NoSQL database are schema is not fixed, NoSQL does not support join operations, NoSQL databases are highly expandable, careful and easy data model as compared to the relational database. NoSQL database having the simple query language, high availability, lower cost. ACID properties of the traditional database, keeping the high cost, but in the exchange of BASE (Base Availability, Soft State, Eventual Consistency) feature, uses the cheap service to manage explosion data and thus achieve the low cost. NoSQL database having the capability to dynamically add the records and distributed data over many servers. Hence the NoSQL database systems famous in the internet companies which had major challenges in dealing with data, relational database are not able to solve this problem.

B. Overview of MongoDB

To collect and fetch the data various applications uses a relational database. RDBMS work well when the size of data is less. RDBMS is inefficient to handle the big data like web data. Newly available databases like NoSQL capable of solving this problem [2] the main benefits of NoSQL database are that they easily handle unstructured records.

MongoDB is document-oriented database not having the schema. MongoDB stands for "humongous" MongoDB database is written in c + + [3]. The basic reason for moving towards the NoSQL database is to make scaling easier. The row in the relational database is replaced with the MongoDB document [4]. Following are the features of MongoDB:

- To store the complex data types MongoDB uses BSON data structure.
- MongoDB supports complicated query language. Mass data is retrieved with high speed.
- MongoDB stores large Binary files.
- Developers easily use and store the JavaScript functions and values on the server side.
- Protocol is easily storing metadata and large files. The response is very fast.
- File mapping is done for the high performance.

MongoDB database is nothing but the set of collections which is shown in Fig 1 and Fig 2. No predefined schema for the collection like tables, MongoDB store data in the format of documents. To store document, Binary encoded JSON-like object is used. MongoDB document is a set of fields. Fields can contain a complex list or even document. In the MongoDB, each document has the unique id like a primary key [5]. MongoDB supports "Multikey" this allows using an array as an index. API MongoDB [6] it has own

query language called Mongo Query Language. Following are some examples:

- Select Command
  db.employee.find(f salary: 100000, age: 25 g)
- Drop Command
  db.employee.drop()
- Insert Command
  db.employee.insert(fuser id: 10, age: 25, status:"poor" g)
- Delete Command
  db.employee.remove(fstatus: "good" g)

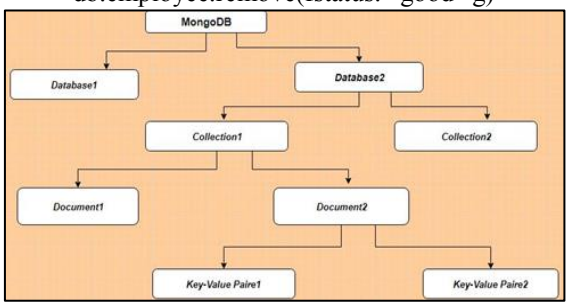

Fig. 1: Structure of MongoDB

One MongoDB cluster build up using three component, shard nodes, configuration servers and routing services.

- Shard Nodes: MongoDB having the set of one or more shards, shard node is used to store the data. Each shard consists of one or more replicated node which can be used to hold the data for that shard. Replicate servers having the one or more servers.
- Configuration Servers: Configuration server is nothing but the set of clusters which is shown in above Figure 2. This server stores the metadata and MongoDB cluster routing information. The metadata and routing information of the MongoDB clusters store in the configuration servers. It also represents the which data is present on which shard. MongoDB having the MongoDB mapped files which can be used to increase the performance. To index the MongoDB database B-tree used. MongoDB used autosharding enabling horizontal scaling across multiple nodes [6].
- Routing Service or Mongos: This server is used to give the response to the client request. The client sends different types of request in the form of queries mongos to send the request to shard and merge the results [4].

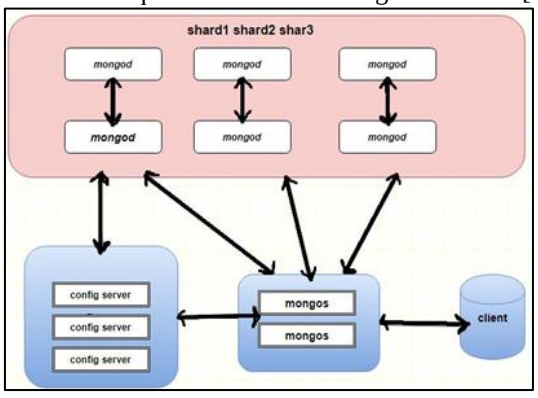

Fig. 2: Architecture of MongoDB

C. Data Migration

Data migration is generally done programmatically to achieve the direct migration, there is no need of human resources that helps to reduce the migration cost. Programmatic migration process includes some phases first one is the extraction, Loading, cleaning and transferring. To improve the data quality need to eliminate redundant or obsolete information. Data migration are of two types direct migration and intermediate migration and different reasons to migrate the database like storage limitations, QoS, upgrades change in the organization policy, security reasons. To achieve the effective data migration there is need to map the existing system data with the new system. Map the data formats with the new system in the proposed system migrate relational data into NoSQL Format there is a need to map the relational database tables, row, columns with the NoSQL database document, collections, fields. To achieve the data migration without human resources convert SQL queries into MongoDB format so query conversion is important in the data migration process. The proposed Query conversion algorithm is used to convert SQL queries into MongoDB format. Data migration is useful for transferring the data from one database to another database or one server to the other server or one architecture to another architecture. Many tools are available in the market to handle the data migration process Apache sqoop is a tool that use to migrate the relational database into NoSQL database. The proposed web application provides the graphical user interface for the query conversion, data migration and CRUD operation. To achieve the data migration from a relational database to the NoSQL uses the data migration algorithm which helps to reduce the migration time. To migrate the database the first step is to get the all metadata related to databases.

Database migration tool which has the auto complete features enabled, they do not provide the automatic query conversion features. The database users or developers need to know about some part of the command so that he can use this migration tool. Working with the traditional database is easy because we work on the relational databases from over 3 decades. But for the new database like MongoDB, the scenario is different. MongoDB introduced the new set of instruction and the structure of MongoDB is also different. Developer not having the deep knowledge of new instruction so it's can't use the all features efficiently. To provide the easy environment to the developer we proposed GUI-based web application for data migration and query conversion.

D. Description of Problem

Data migration is the important aspects of the organization. Lots of organization needs to migrate their relational database into a newly available database due to some inefficient functionality. To achieve data migration, we proposed an architecture for the data migration and the query conversion with GUI, Use to migrate a relational database into the newly available database, i.e. NoSQL, to achieve data migration we use the data migration algorithm. There is a problem with NoSQL Database they don't use the structured query language to write the query so there is a need to provide query conversion utility to satisfy the user requirement. To achieve the query conversion uses the query conversion algorithm. Data migration with the manual process of data migration is difficult, the manual process is time-consuming and also requires man power it directly effect on the cost. But in our proposed system the migration time is less, there is no need of man power because we provide GUI also reduce the migration time.

E. Motivation

Big organization, companies need to manage the big data. The traditional database is not capable of handling the huge amount of data. Working with the available data migration tool some problem is faced. No Graphical user interface is used to migrate the data and auto query conversion features. Motivated from this the system is proposed to build the web application for data migration and query conversion.

Database migration tool which has the auto complete features enabled, they do not provide the automatic query conversion features. The database users or developers need to know about some part of the command so that he can use this migration tool. Working with the traditional database is easy because work on the relational databases from over 3 decades. But for the new database like MongoDB, the scenario is different. MongoDB introduced the new set of instruction and the structure of MongoDB is also different. Developer not having the deep knowledge of new instruction so they can't use all features efficiently. To provide the easy environment to the developer proposed GUI-based web application for query conversion and data migration.

F. Goals and Objectives

The proposed application is to provide a GUI-based web application for data migration (RDBMS to NoSQL) with Minimum time by using proposed algorithm and also provide automatic Query Conversion facility.

- To design a migration model that will help in migration of relational databases to NoSQL databases using Linking Process.
- To create a web-based solution for migrating relational databases to NoSQL databases.
- To provide the GUI for CRUD operation and data migration.
- To reduce the development cost by using automatic query conversion utility. To reduce the migration time.

G. Literature Survey

According to the authors, proposed framework is for the denormalization. The denormalization is used to improve the performance of the data retrieval, data reading and help to improve performance. At the time of data denormalization, there is no data loss integrity. This framework is also helpful to improve the execution speed of query like transferring table, join table [7]. The proposed NoSQL Layer frameworks is used to perform the datasets migration automatically and transparently (i.e the MySQL to MongoDB). The idea behind this framework is to keep the semantics of the traditional database, for the query writing and the data modeled. They offer a framework to the abstract layer that gives permission to the software applications to access data in the NoSQL Format transparently without changing the queries in that application, framework provide the migration successfully and maintained the original database. Each SQL operation is expected on the application and converts request into the NoSQL Database [8]. According to the authors, the proposed work is used to guide the construction design of NoSQL database like MongoDB. MongoDB is a document-oriented database system. Relational calculus is used to prove the relationship between the two databases [9]. According to the author, when migrating the relational database there is no any security related problem in database. To design web application for the data migration the proposed method uses the linking process. Linking process is helpful to create the relationship between primary and foreign key. According to the author, the query execution time of MongoDB is less as compared to a traditional database of embedded data and linking or references data, to reduce the query execution time of MongoDB, they did a lot of experiments to extend the embedding and normalization.

Chung W.C. et al., proposed work is used for the conversion from relational database to HBase separately. They proposed two ideas for the schema conversion. MapReduce framework will convert relational database, each and every table into a single table. When the schema conversion process was done the table is in the relational database is the member of the target database. They use three guidelines for the schema conversion. All related table will be nested transform into a table of HBase. They do not use the multilevel nesting [10].

According to the authors, the proposed system used for the schema conversion. In the proposed work they used to convert relational database schema into the non-relational database schema. They proposed an algorithm for the NoSQL database to other NoSQL database with the help of graph techniques. The graph transformation which identifies the correctness of schema conversion and capable of reading on NoSQL Databases [9].

Bock D.B. et al., Proposed the framework is used to achieve high availability, reading, and scalability of RDBMS. To improve the data migration speed lots of experiment used to perform on the databases with the help of denormalization. The proposed framework is useful to perform the read operation fast and retrieve the data within less time span [10].

II. ARCHITECTURAL DESIGN

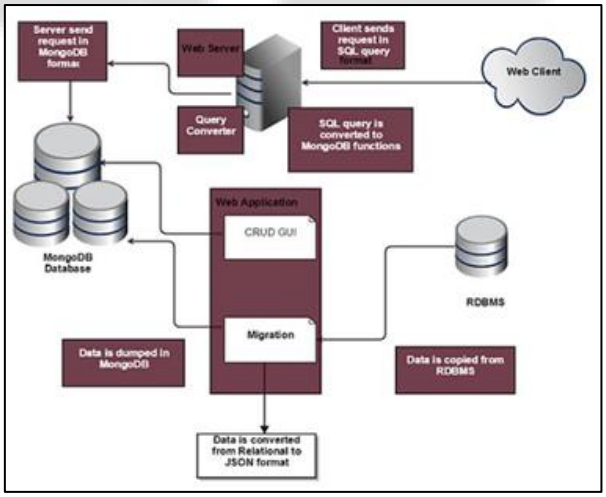

Fig. 3: Proposed System Architecture

Architectural Design gives the overall view of system components and their interface with each other as shown in below Fig 3. The proposed system is divided into two parts:

- Web Based Client Application
- Query Conversion Utility

The web-based client application provides the integrated development environment for the developer to create optimized queries in MongoDB shown in below Fig 3. There is no need to remember any kind of syntaxes, or latest method and any changes in the modules of the MongoDB query language. To process the query web-based IDE will

provide the graphical or symbolic representation of the methods and functions. To achieve this the whole process hierarchy starting from the database and ending to collections, documents, criteria, and projections. At every ordered level display method to the users.

It also displays the structure and data types. At any time, user can be extended, revise or execute his query. The proposed system is capable of handling the errors generated by the users and correct it automatically before the execution. All changes are made internally through the IDE.

A. Data Migration

Migration is the process of transferring either data from a single table or from the entire database from one server to another or from one database architecture to another. Here migrate RDBMS Database into MongoDB shown in above Fig 3. Step first is to choose the data that want to migrate get the metadata of that database. Metadata is nothing but the data gives the information or describe the other data. Metadata having the information about the primary key, foreign key information from that metadata generate the relationship of that RDBMS database and find the joins that maintain the relationship of the primary and foreign key. Step second is to convert the RDBMS data into JSON format.

B. JSON

JSON format can be read by the peoples or machines. JSON having the XML Format for the data interchange. To store the records MongoDB used JSON document. Javascript object notation supports the data types like numbers, Boolean values, array, strings as well as hashes. In our proposed web application when migrate the database first step is to map the dataset tables to a java class. After that, retrieve the records from datasets using JDBC as shown in above Fig 3. Convert all records in the JASON format using JACKSON Library.

C. Mapping Tables, Rows, and Columns

In the relational database the data stored in the tabular format but in the MongoDB, the record is saved in the format of collections. Collections having the set of documents like relational records. Row stores data in the set of columns. Document having the JSON-Like structure.

This document looks like the single row in the column. In the relational database table having the set of rows and in MongoDB collection consist of many documents. MongoDB automatically generates the unique id for the documents. The size of generated unique id is 12 bytes.

D. GUI for the CRUD Operation

In our implemented web application provide the GUI for the create, read, delete and update operation user simply know about the basic things no need to know about the SQL query. The proposed web application provides the simple click functions to create the query and retrieves the data easily as shown in above Fig 3.

III. Methodology

A. Methodology for the Data Migration

- Step 1: Get the metadata of the database.
- Step 2: Map the RDBMS and MongoDB databases by using mapping process.
- Step 3: Make the relations of two databases by using linking process.
- Step 4: Migrate the databases.

B. Methodology for the Query Conversion

- Step 1: Get fired query.
- Step 2: Tokenize the query.
- Step 3: Map the all data with MongoDB format.
- Step 4: Generate the MongoDB query and fetch the results from the MongoDB.

1) Algorithm 1: Algorithm for Data Migration

- Input: Traditional DB and NoSQL Database
- Output: Data migrated successfully

1) Begin:
2) Initialization: select database
3) if exist
4) Drop first and create new
5) then create a new one
6) Get metadata of database
7) divide all tables in two category
8) migrate first tables having primary keys
9) get all metadata of table
10) map (collection <-table, field<-column, document<-row)
11) While exist (row in table)
12) do
13) map(document<-row)
14) select table having foreign key constraint
15) map(collection<-table)
16) map(field<-column)
17) map(document<-row)
18) end While exist (row in table)
19) do
20) map(document<-row)
21) create primary key foreign key relationship by Linking Process
22) exist foreign key
23) map uid of primary key document to foreign key document
24) commit migration
25) end

IV. Algorithm Design

A. Algorithm for Data Migration

The proposed data migration algorithm, used to reduce the data migration time. Algorithm 1 describes the data migration process. RDBMS database is in the tabular format which has the number of tables and NoSQL database i.e. MongoDB is a document oriented database. RDBMS is the source database that need to migrate and MongoDB is the target database where the source database migrated in the format of MongoDB. First Step is to select the database for migration from the available databases if the selected database is already migrated then first drop this database from the MongoDB database and after that create a new database to store the migrated database. The second step is to get the metadata of that database.

B. Algorithm for Query Conversion

The algorithm describes the query conversion from SQL to MongoDB and fetches the results from the MongoDB database.

Algorithm 2 describes the query conversion from MySQL to MongoDB lets take one SQL query select from employee where salary = 20000 and convert this query in MongoDB format i.e db.employee.find("salary": 20000); first step is to get the fired SQL query then tokenize this query if the query is for select operation then proceed else not because the implementation on query conversion only for the select operation, the second step describe that select operation with condition or without condition check and proceed if select query with condition then field name on which condition will apply. Step third describe that to map the collection with table, the document with row and field with column after that your MongoDB query is ready to fire the query and get the results.

1) Algorithm 2: Algorithm for Query Conversion

– Input: SQL Query
– Output: SQL query is converted into MongoDB query Format

1) Begin:
2) Initialization : Get fired query of SQL
3) Tokenize the SQL query
4) if Select
5) then Proceed
6) Get the table name
7) Find out query is of type selection or selection with where clause
8) else
9) Not proceed
10) if Selection Type
11) then converts it into MongoDB query for all documents in the collection
12) map(collection<-table, document<-row, field<-column)
13) Convert it into MongoDB syntax
14) db.collection name.find()
15) if
16) selection with where clause type
17) get field name on which condition will apply
18) Convert it into MongoDB syntax
19) else
20) apply condition on all documents in MongoDB
21) map(collection<-table, document<-row, field<-column)
22) db.Collection Name.find("Field Name": Condition Value); Fire the MongoDB query and see the result
23) end

V. PERFORMANCE ANALYSIS

For the purpose of performance analysis, different scenarios have been considered. The investigation has been carried out by mainly considering the different scenarios:

– Calculating the data migration time between two databases.
– Comparing the data migration time with manual process and internal process.
– Analyzed data migration time and CPU utilization based on a number of records.

A. Results and Discussion

1) Experimental Evaluation

Evolution to validate our RDBMS to NoSQL data migration and deployed on the cloud for effective result. Performed several experiments to analyze the behavior and performance of data migration application. Evaluated our data migration approach on Employee dataset which is available on Launchpad. The experiment uses input as a subset of different number of records (i.e. 2304, 4608, 9216, 18432, 36884, 73728, 110592, 147456, 184320) described in analysis part. The evaluation examines the following three parameters:

a) Migration Time

The average time needed to translate the number of records to target database i.e MongoDB.

Where,

Load = Number of records to migrate
Start = Time taken before start of migration
Overhead Bytes = Number of bytes used for initial communication (Headers)

Following are the equations used to calculate migration time, Equation (1) is used to calculate the transfer rate measure in kb per milli seconds.

Transfer Rate = (load + overhead Bytes) / (end -start) (1)

Migration Time = load × Data Transfer Rate (2)

The above equation (2) is used to calculate the migration time measure milliseconds.

– Procedure to calculate migration time reduction

x = (Migration Time proposed/Migration Time existing) (3)

Equation (3) is used to calculate the migration reduction percentage.

y = x - 100 Reduction = 100 - y

b) CPU Usage

CPU usage is nothing but the percentage of the CPU usage required by the migration system.

– Procedure to calculate CPU Utilization.

Where,

Elapsed Start Time = time at start of migration
CPU Start Time = time at start of current thread execution at CPU
CPU Count = Number of CPUs in a system CPU Count = Available Processors

Total Available CPU Time = CPU Count × (end – elapsed Start Time) (4)

Total Used CPU Time = Current Thread CPU Time – CPU Start Time (5)

CPU utilization in percentage = (Total Used CPU Time ×100) / Total Available CPU Time (6)

The above equation (4), (5) and (6) are used to calculate the CPU utilization at the time of migration process.

– Procedure to calculate CPU Reduction:

x = (CPU usage proposed system / CPU usage existing system)

y = x ×100

Reduction = 100 -y

c) Number of Records

Number of migrated records in the target NoSQL database.

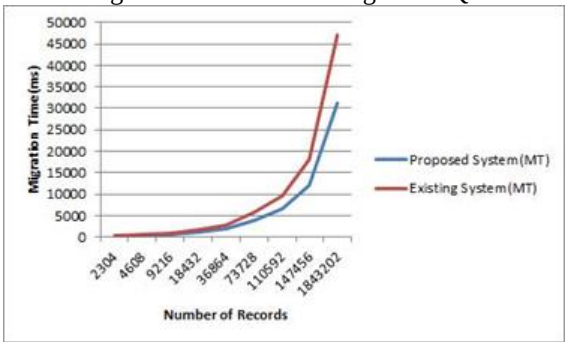

Fig. 4: Overall Migration Performance

The experimental setup is hosted on the cloud environment; perform the evaluation on the JElastic cloud which provided the platform as a service (PaaS). JElastic is a cloud union and ever data is the cloud provider. The configuration of this cloud is it provides platform as a service as per the requirement our requirement is Java 7, Tomcat 7.0.70 server and the OS it also provides the 8 cloudlets i.e 8 virtual PCs, one cloudlet having 128mb RAM and 400mhz CPU.

The above Fig 4 shows the graph for migration performance with respect to migration time. The x-axis represents the number of records and Y-axis represent the migration time in milliseconds. The average reduction of the migration time is 33.30 %.

The above Fig 5 shows the graph for CPU Performance and below Fig 6 shows the graph for overall Reduction with respect to CPU utilization. X-axis represents the number of records and Y-axis represents the CPU utilization in percentage. The average reduction of the CPU utilization in percentage is 15.08 %.

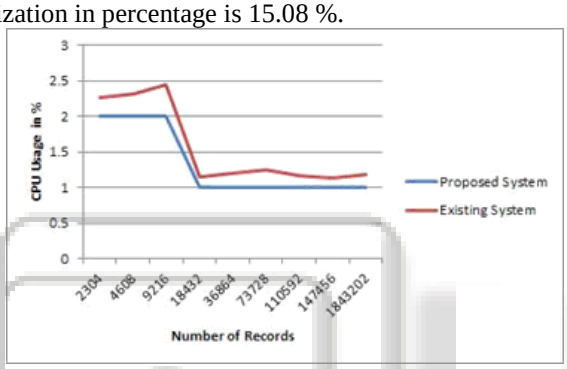

Fig. 5: Overall CPU Performance

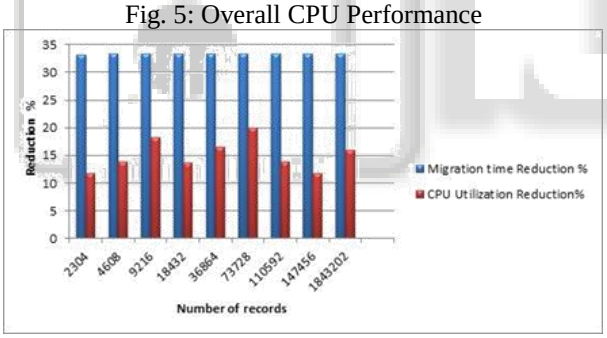

Fig. 6: Overall Reduction

VI. SUMMARY

This paper introduces a Data migration and Query Conversion Algorithm for proposed web application. Linking process is used to reduce the data migration time and the development cost by using automatic query conversion utility. Web application provides the simple click functionality to retrieve the data from the databases, there is no need to know about queries.

VII. CONCLUSION

The proposed web application helps to understand the syntaxes that can help to write the correct query. The web application is developed for data migration from a relational database to NoSQL database. Query conversion utility helps to fetch the result from MongoDB and reduced the development cost. In this project, the GUI-based application for CRUD operations is developed which can be further extended to support higher hierarchies. The cloud environment results show that the proposed web application decreased the overall migration time by 33.30 %, reduced the overall CPU utilization by 15.03%

VIII. FUTURE ENHANCEMENT

Following points extended in the future:

- In this work developed the GUI based tool for CRUD operations which can be extended in the future to support aggregations and indexing operations.
- The proposed Migration application supports the first normal form which can be further expanded to support 2NF, 3NF and so on.
- The proposed application would be migrating RDBMS to MongoDB which can be further extended to migrate from RDBMS to all NoSQL databases.

ACKNOWLEDGMENT

I take immense pleasure in expressing my humble note of gratitude to my project guide Prof. R. R. Badre Associate Professor, Department of Computer Engineering, MIT Academy of Engineering, Alandi, Pune, for his remarkable guidance and useful suggestions, which helped me in completion the paper before the deadline.

REFERENCES

[1] Codd E. F. (1970), "A relational model of data for large shared data banks", Communications of the ACM, 13(6): 377-387.
[2] Bruhn D. 2011, "Comparison of Distribution Technologies in Different NoSQL Database Systems", Karlsruhe Institute of Technology, KIT.
[3] Han J., Haihong E., Le G. and Du J. (2011), "Survey on NoSQL database", In Pervasive computing and applications (ICPCA), 2011 6th international conference: 363-366.
[4] Chodorow K., (2013), "MongoDB: the definitive guide O'Reilly Media, Inc."
[5] Padhy R. P., Patra M. R. and Satapathy S.C.(2011), "RDBMS to NoSQL": reviewing some next-generation non-relational databases. International Journal of Advanced Engineering Science and Technologies, 11(1): 15-30.
[6] Okman L., Gal-Oz N., Gonen Y., Gudes E. and Abramov J. (2011), "Security issues in nosql databases", In Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th Interna-tional Conference: 541-547.
[7] Pinto Y., 2009," A framework for systematic database denormalization", Global Journal of Computer Science and Technology, 9(4): 44-52.
[8] Rocha L., Vale F., Cirilo E., Barbosa D. and Mouro F.(2015)," A Framework for Migrating Relational Datasets to NoSQL1", Procedia Computer Science, 51, 2593-2602.
[9] Zhao G., Huang W., Liang S. and Tang Y. (2013),"Modeling MongoDB with relational model", In Emerging Intelligent Data and Web Technolo-gies (EIDWT), 2013 Fourth International Conference: 115-121.

[10] Bock D. B. and Schrage J. F., 2002. \Denormalization guidelines for base and transaction tables", ACM SIGCSE Bulletin, 34(4):129-133.
[11] Chung W.C., Lin H.P., Chen S.C., Jiang M.F. and Chung Y.C. (2014), "JackHare: a framework for SQL to NoSQL translation using MapRe-duce", Automated Software Engineering, 21(4), pp.489-508.
[12] Bock D. B. and Schrage J. F., 2002, "Denormalization guidelines for base and transaction tables", ACM SIGCSE Bulletin, 34(4):129-133.

