MigDB

Project ID: 16-015

Project Proposal

B.Sc. Special (Honors) Degree in Information Technology

Submitted on <*date*>

MigDB

Project ID : 16-015

Authors:

Student ID	Name	Signature
IT13001476	Liyanaarachchi L.A.G.C	
IT13088156	Madhusanka K.P.L.K	
IT13093938	Kothalawala K.R.M.N	
IT13075422	Padmasiri H.R.K.L	

Supervisor

……

Dr. Anuradha Karunasena

Co-Supervisor

……

<Name of the Co-supervisor>

DECLARATION

We declare that this is our own work and this project proposal does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

……..	……..……..
Liyanaarachchi L.A.G.C	*Madhusanka K.P.L.K*
……..	……..……..
Kothalawala K.R.M.N	*Padmasiri H.R.K.L*

ABSTRACT

Relational databases have been used by the majority for database management purposes for many years. With the ease of operation and sound execution of transactions, relational database is a good fit when it comes to structured data. However, in regards to scalability and emergence of big data which encompasses accelerating amount of structured as well as unstructured data, relational databases are not a proper match. Addressing these issues, NoSQL comes in as a much more flexible alternative to the relational databases being capable of supporting vast amount of complex data which vary in type and structure. With the popularity of NoSQL, the focus moves on to the migration of relational databases into NoSQL databases. Especially with respect to MongoDB, a leading NoSQL document database, many users focus on migrating to MongoDB. In regards to the migration of relational database tables, data along with queries into MongoDB, there are certain issues like current migration systems being incapable of properly mapping table relationships, existing query converters only being capable of converting select queries, the inability of query converters to support table joins and most of the tools requiring MongoDB command knowledge. Hence, the focus of the intended system is to facilitate users to easily transfer from MySQL tables along with proper relationship mapping, data as well as queries into MongoDB without requiring any prior knowledge on MongoDB commands or JSON.

TABLE OF CONTENT

Contents

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Background

With the rapid growth of the software industry and emerging trends like big data, which involves complex, voluminous amount of structured, semi-structured and unstructured data with high velocity, the database management comes to play a vital role in the software industry. When it comes to database management, Relational database has been the main mode of database management for the last few decades and is still used widely all around the world. Relational databases, which follow the **relational model of data** proposed by E.F. Codd in 1970, is a "collection of data items organized as a set of formally-described **tables** from which data can be accessed or reassembled in many different ways without having to reorganize the database tables [1]." Basically relational databases store data in a structured table which is also called a relation, in columns and rows where a column represents an attribute or field and a row aka tuple represents an instance of the entity which is denoted by the table while each record/ tuple in a table is identified by a unique key called the primary key. The most prominent characteristic of relational databases is its support for ACID (Atomicity, Consistency, Isolation, and Durability) properties. Structured Query Language (SQL) is used as the means of interacting with these relational databases. SQL can be used to store and manipulate and retrieve data in relational databases.

Relational databases though with their huge benefits and efficient and reliable execution of transactions which is ensured by its support for ACID properties, is still have limitations in certain areas like scalability and big data. With relational databases, as the amount of data which needs to be stored increased, the database needed to be scaled up which means adding more resources to a given node and customers had to buy bigger, more complex hardware with higher processing speed, memory and storage which was a very costly task and upgrading these databases was also difficult. To improve the scaling process of relational databases, the vendors have introduced some mechanisms like "master-slave" model (where slaves are additional servers to the master server and data is sharded or distributed among these multiple servers), shared storage, in-memory processing, distributed caching. These improvements also does not come free as there could be trade-offs like flexibility, cost and performance. And when it comes

to big data, which consist of massive volume and variety in both type and structure of data, the strictly structured schema-based relational database, is an unsuited solution. Due to these issues, it was decided that relational databases could not be the ideal solution at all times.

NoSQL, which stands for Not Only SQL comes into picture focusing on the issues that the relational databases could not properly address. NoSQL databases, initiated by the necessities of companies like Facebook, Google and Amazon, are best used when dealing with massive amount of semi-structured or unstructured data or data which is stored on multiple servers. NoSQL databases "provide a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases [2]." NoSQL databases assemble related data together unlike tables used in relational databases. NoSQL database types include Key-values database, which associates each value with a key, Column database, where various columns are distributed using clusters while having a pointer which points to the columns, Document database, that consist of key-value collections whilst allowing nested values to be assigned to a key, which is a graph model having edges and nodes that have a capability of storing other items like key-value pairs.

NoSQL follows CAP (Consistency, Availability and Partition Tolerance) enabling users to select any two of the three attributes. NoSQL databases supports scalability well with its distributed architecture and can handle massive amount of data by scaling out horizontally which means by adding new nodes to the distributed system each sharing the workload. This approve is extremely cost-friendly compared to the vertical scaling mechanism of relational databased thus it also eliminates the single point of failure since if one node fails other nodes can take over. NoSQL databases becomes a perfect fit when it comes to big data as its schema-less methodology and horizontally scaling enables it to adapt well to the rapidly increasing and changing semi-structured and unstructured data. With big data also comes the new types of data needs to be added frequently and even though this could be a very troublesome task in relational databased once data is inserted, this task could be easily done with the help of dynamic schema support of NoSQL.

Consequently, due to the high flexibility and scalability and performance advantages of NoSQL, many users are taking interest in the NoSQL databases and those who are facing the issues in the relational databases currently are turning to migrating from relational database to NoSQL

seeking the solutions for these issues. The focus of the intended system is to turn the cumbersome task of converting an existing Relational database to a NoSQL database, into a more manageable and easier task. The proposed system specifically concentrates on the migration of an existing relational MySQL database to a NoSQL open source document type database MongoDB. MongoDB follows stores data in JSON-like documents which can diverge in structure. The objective of the intended system is provide a user-friendly approach for the user, who can be even a novice in NoSQL databases, to migrate databases as well as to generate or convert queries.

1.2. Literature Review

With NoSQL gaining popularity as a much more flexible alternative to relational databases, many researches have been conducted the feasibility and issues of the migration of an existing relational database to a NoSQL database. Moreover when it comes to the specialization of document-oriented MongoDB as the NoSQL database and since MongoDB ranks high among the non-relational databases, there are a few systems available for the purpose of migration of an existing relational database to a MongoDB database. These available products, while having some sound features, still have some shortcomings in certain areas.

In a research [3] conducted by Gansen Zhao, Weichai Huang, Shunlin Liang, Yong Tang, it has been verified that MongoDB is capable of performing relational calculus as it is in relational databases and by providing examples, it has been concluded that theoretically it is easy to migrate data from a relational database to MongoDB. Gansen Zhao, Weichai Huang, Shunlin Liang and Yong Tang states that "given a RDBMS instance with its relation design, it is possible to construct a MongoDB instance that provides the same data management and query features as the RDBMS instance does." To come to the conclusion of the research, the mathematical model of MongoDB has been developed using relational algebra which is the principal model of RDBMS in order to compare MongoDB's capabilities with the relational databases.

The proceeding research [4] by Gansen Zhao, Weichai Huang, Shunlin Liang and Yong Tang provides a mechanism to support join operations in relational databases and to improve query efficiency when migrating to MongoDB by proposing an algorithm based on the nested method.

In the nested method, the related table data is embedded into the collection taking the table which refers to another table as the parent and the table which is referenced as the child. If the child table refers to another table, then that table data is also embedded into the child within the parent. This nested approach which keeps on embedding related data, provides high query efficiency thus also providing a solution for table joining issues. Nonetheless, using nested method in scenarios where a table could be referred by many other tables could result in high data redundancy and can lead to waste of space.

When it comes to the products which focus on migration of an existing relational database to a MongoDB database, Pelica Migrator is one of the most known tools. Pelica Migrator which was launched in 2014 by the organization Techgene, facilitates the users to migrate data from any RDBMS (Relational Database Management System) including MySQL, Oracle, SQL Server 2008, PostgreSQL, DB2 and Sybase to MongoDB. The tool migrates relational database relations/tables into embedded collections or documents in MongoDB and also replicates the relational database data into MongoDB. The users are capable of selecting and unselecting the tables as well as columns that they want to migrate with this tool. Pelica Migrator also provides a feature called Version Controlling which stores and displays migration selection history for future reference. This feature allows users to view and modify their previous selections in order to fit their current need. Pelica migrator, while keenly migrating both relational database table structures and data into MongoDB, still has some drawbacks as it only converts the relational database tables just as they are into collections without considering the relationships among the tables. The tool does not provide capability for either embedding, where the related data is embedded into the same collection, or referencing, where related data is kept in another document and object id is used as a reference, when mapping one-to-one, one-to-many or many-to-many relationships among tables. Hence, the inability to map the relationships appropriately stands out as a major setback of the tool Pelica Migrator.

NoSQL Manager for MongoDB Professional is a sound management, administration and development tool for MongoDB. The tool facilitates users to create collections, add/ modify documents within collections, add/ modify fields within documents using the GUI while also providing the ability to create functions, users and user roles. NoSQL Manager for MongoDB provides the feature of importing data from both CSV and JSON files alongside functionality of exporting data to a CSV, JSON and XML file. The application is also capable of copying an existing database to another database. While the tool

consists of fully functional Mongo shell, it provides auto-completion of Mongo commands as well. Moreover, NoSQL Manager for MongoDB can export Mongo database dump along with restoring an existing Mongo database dump. More importantly, the tool is capable of migrating MS SQL Server and MySQL relational database tables together with data into MongoDB. But then again, just as in Pelica Migrator, NoSQL Manager for MongoDB though it provides many effective features, only transfers tables and data just as they are with no consideration of relationships and therefore is incapable of mapping relationships among tables. Additionally, the tool also requires users' competence in JSON and MongoDB commands.

Another software product associated with MongoDB is RoboMongo. RoboMongo is a shell-centric cross-platform open source software which acts as a Management tool for MongoDB and thus does not have a capability of migrating an existing relational database to MongoDB. The graphical tool provides facilities to create and modify collections, insert documents to collections while also providing the ability to update or delete documents within collections. RoboMongo presents three ways of displaying collections such as tree-mode, where objects are displayed in a hierarchical subdivided structure, table-mode, in which objects are presented in a table and text-mode where objects can be viewed in a JSON-like format. In this tool, multiple shells can be opened and worked on as they operate individually. A major advantage plus point in RoboMongo is its offering of runtime auto-completion of commands while also being available for Linux, OSX and Windows. Even though RoboMongo is a good graphical management tool for MongoDB, it is still difficult to be used by a novice in MongoDB as the tool requires users to have knowledge on JSON and MongoDB commands.

UMongo is an admin GUI management tool for MongoDB. The tool allows users to create and modify databases, create and modify collections while also enabling users to insert, update, and delete documents. UMongo also provides users sharding options to enable sharding, add shard and to view shard collections. A GUI document builder is provided by the tool while also facilitating users to import and export data to various formats like JSON, BSON and CSV. Nevertheless, the tool UMongo does not have sufficient user-friendliness as the tool necessitates good skills of JSON and MongoDB commands from the user.

A different tool for handling MongoDB is Admin mongo, which is a web-based application. The application enables users to create and delete collections, create and delete users and view

statistics of the database. Users are also capable of create, update and delete documents as well as to view and insert indexes to collections. However, the application still needs users to have JSON and MongoDB command knowledge.

In regards to conversion of SQL statements into MongoDB statements, there are a few web applications available for the purpose of translating an SQL statement to a MongoDB statement. Query Mongo, while being the most popular query translator, is capable of converting MySQL queries into MongoDB syntaxes. However, whilst the web application translated queries efficiently with managing aggregate functions really well, it is only capable of converting SQL SELECT queries and does not support table joins. Another web application for converting SQL statements to MongoDB statements is Klaus.dk which also translates SQL select queries into MongoDB. The drawbacks of this web application is that it only translated SELECT queries and is incapable of supporting aggregate functions.

Following reviewing of existing researches and existing software products, it can be concluded that though most of the present systems are capable of delivering some decent features and functionalities efficiently, they still have some drawbacks in specific areas due to being incapable of mapping table relationships either by using embedding or referencing and these systems also short in user-friendliness as almost all of the systems discussed requires expertise in JSON and MongoDB from the users.

1.3. Research Gap & Research Problem

The main requirement of this research is to create a system that migrates existing relational database to a MongoDB and manage migrated database with user friendly GUI without requiring any MongoDB related knowledge. As identified in the literature review, there are some crucial limitations in the existing literature which are to be addressed in the indented system.

The major focus of the intended system is to map SQL relationships into mongo database. Because existing products don't have any mechanism to map relations. Here in the intended system best way to map a relation will be suggested by an internal neural network and also with the user's interaction. The user interactions will be learned to the system after a system evaluation about the user's changes. Finally SQL relations will be embed or linked to collections considering previous experiences.

Unlike existing products the system will give the best user experience to the user with graphical user interface. It will provide much easier functionalities like drag and drop. As an example consider table a need to be embed to table B. Dragging table A onto table B will simply do that kind of functionalities.

When someone moves to MongoDB from relational database they may not have good experience with MongoDB shell commands. So novice users will have difficulties with No-SQL Manager for Mongo DB, Robo Mongo like tools because they will need the mongo shell commands and JSON based knowledge to work with above mentioned tools. With the intended system user will have the ability to manage mongo databases with only SQL knowledge. When user types a SQL query system will convert and execute the corresponding mongo commands.

Query translation tools like query mongo are only capable of converting SQL select queries to mongo commands. Even there are limitations like not supporting for tables joins. The intended systems query converter will support for more SQL queries like update, delete and table joins.

A research [4] conducted by Gansen Zhao, Weichai Huang, Shunlin Liang and Yong Tang proposes a relational database to NoSQL database conversion model, which uses nested idea to improve query speed of NoSQL database. The drawback with mechanism is the size of the database with respect to the no of foreign keys being rapidly increased. Since the system proposes the best way to map relationships with previous experiences the relationships will be mapped in the most appropriate way taking both space usage and query efficiency into account.

Features	Research paper [4]	Pelica Migrator	No-SQL Manager	Robo Mongo	UMongo	Admin mongo	Query Mongo	MigDB
Evaluate data changes to the schema								✓
Migration of table structures		✓	✓					✓
Migration of data		✓	✓					✓
Selection of most appropriate relationship mapping option from embedding or referencing	(Only embedding)							✓
User interactions to mapping								✓
Graphical management tool for MongoDB				✓	✓	✓		✓
Query conversion with joining							(only select queries)	✓
Graphical query generation								✓
Does not require JSON or MongoDB knowledge		✓						✓

Table 1 : Comparison of Features

2. OBJECTIVES

2.1. Main Objective

The key objective of this research is make it possible to convert relational database dump into NOSQL (initially MongoDB) dump together with existing data, table structure, relationships and existing database queries consuming incremental machine knowledge and human knowledge.

2.2. Specific Objectives

- Create connection management module to establish connection with MySQL database and MongoDB database in localhost. This module has capability to hold multiple database connections.
- Create module to analyze relational database dump file and understand table structure, references and data. Then map those information into intermediate JSON file.
- Design an interactive User Interface to illustrate complete table structure in well understandable manner and record all changes user expected to have in table structure.

  This module has knowledge to evaluate user changes and identify any violations and report user to recheck changes before submitting. If there are no any violations this module will modify system readable intermediate file according to the user changes.
- Create a Neural network as a separate module to predict about database mapping technique with considering references, column count and data types of columns. This module has capability to trigger scheduled Neural Network supervised learning sessions using a CSV file. This module has capability to retrieve information from Neural Network and send to other modules through a restful interface.
- Provide definition for the mapping of 1 to 1 relationships in relational database tables into NOSQL collections with data migration.
- Provide definition for the mapping of 1 to many relationships in relational database tables into NOSQL collections with data migration.
- Provide definition for the mapping of many to many relationships in relational database tables to NOSQL collections with data migration.

- Design an interactive user interface to illustrate collection structure generated by the system and enable user to change it by applying user experience on database mapping in user friendly way. This module has capability to evaluate user's changes on mapping and report violations to the user. If there are no any violations, catch user changes on mapping and update CSV file which use to learn Neural Network.
- Create individual module to convert existing reference mapping into embedded mapping and update database. This module has subcomponent to convert existing embedded mapping into reference mapping.
- Create module to access NOSQL database on localhost and graphically represent data to the user and enable user to manipulate data and save changes.
- Design an interactive user interface to generate new database queries in user friendly manner. This interface has another functionality to get existing database query as input from user and convert it into NOSQL query in user friendly approach.
- Create user friendly web portal to enable user for accessing above services with getting continuous user interaction. This web portal has capabilities to handle payments, file upload and download, user management and file version management.

3. RESEARCH METHODOLOGY

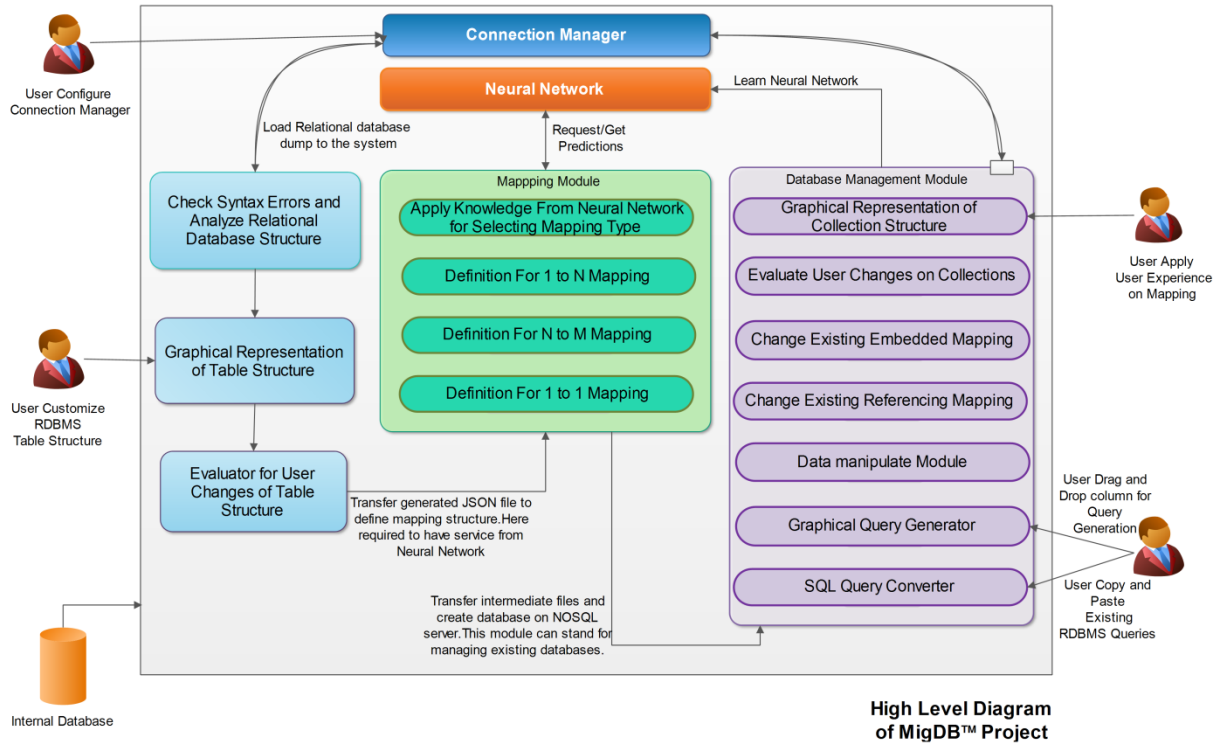


Figure 1 : System Diagram

The main goal of the intended system is to facilitate users to easily transfer from MySQL tables, data along with queries into MongoDB without requiring any prior knowledge on MongoDB commands or JSON. In order to accomplish the objective, the system uses a step by step approach while interacting with the user in the process.

For the migration process to start, the system requires the SQL database to be migrated. The SQL database can be provided either by uploading the MySQL dump file or by selecting the database to be migrated after providing MySQL connection parameters. The system then analyzes the dump file and creates a JSON file with table and data structures. Using the JSON file, the system represent MySQL table structures graphically and allows users to remove tables/ columns that they do not wish to be migrated. User changes are validated and the updated JSON file is forwarded for relationship mapping. Initially, all tables and data are mapped into document-like objects and store in a separate JSON file. If the table structures contain one-to-one relationships, the one-to-one mapping module analyzes the table structures and embeds data in an appropriate

manner. The one-to-many mapping module then uses the neural network in selecting the most suitable mapping option to map the one-to-many relationships within the database to be migrated by passing the necessary parameters. After comparing neural network response with its own evaluations, the one-to-many mapping module uses embedding or referencing appropriately. The re-structured JSON file containing document objects is then analyzed by the many-to-many mapping module. Many-to-many mapping modules sends a request to the neural network in order to decide on the best mapping option and according to the response, the module re-arranges the JSON file containing document objects. Following the completion of relationship mapping, the collections are created in MongoDB and documents are inserted into the collections. The management module of the system facilitates users to modify the way that the relationship mapping is done while also enabling users to modify data within documents. The system aids users in generating MongoDB commands graphically by only selecting items and typing values which helps users with a fewer knowledge in MongoDB commands or JSON. Moreover, the system facilitates users to convert existing MySQL queries into MongoDB commands easily and instantly.

The development process is to follow waterfall model and the technologies which are to be used are Java8, JavaFX, MongoDB, Sqlite, JSON, Apache Derby, Neuroph framework, JAX-RS and Jersy. The project would also use SceneBuilder, Eclipse, Apache Tomcat, Maven builder, Netbeans and MySQL Workbench as the tools.

SQL DB Analyzer

This is the component for analyzing SQL database structure along with the creation of intermediate JSON (JavaScript Object Notation). With this module users can provide their database to migrated either by providing SQL connection and selecting the database from the synchronized MySQL database list or by uploading a MySQL dump file. If the user selects the option of uploading the dump file, the uploaded SQL file is inspected for any syntax errors by the module. In case of a syntax error, the module would alert the user with an error message. After verifying the SQL dump with no syntax errors, the component would start analyzing the SQL dump file to create the intermediate JSON file. In reference to JSON file creation, **SQL DB Analyzer** module would first read the create table statements without considering foreign key constraints and would create JSON objects for each and every table including properties such as table name, column count, primary key and columns with data types. Once all table objects have

been created, the module would then move to reading of foreign key constraints. When a foreign key constraint statement is read, the system would add another object that includes the referencing details like to which table and which column the foreign key refers to, which column is the foreign key as well as the relationship type to the table object of the table containing the foreign key. Meanwhile, it would add an object containing referenced details to the table object which is referred by the foreign key such as which column of that table is referenced, which column and table refers to that table along with the relationship type. To decide on the relationship type, the module would read randomly selected data. Following the completion of table structure mapping to JSON, the system would insert existing data as objects to the JSON file. A JSON file containing all the table structures and data would be the output of this component.


```
"tables":[
   {
      "name":"customer",
      "colCount":4,
      "primaryKey":"id",
      "columns":[
         {
            "colName":"id",
            "dataType":"int"
         },
         { },
         { },
         { }
      ],
      "referencedBy":[
         {
            "referencedCol":"id",
            "referencingTab":"contactInfo",
            "referencingCol":"customerId",
            "relationshipType":"OneToOne"
         },
         { }
      ]
   },
```


Figure 2 : JSON File Structure

SQL DB Modification Evaluator

This component would aid users in the selection of which tables and columns they wish to migrate as well as which needs to be removed. Taking the JSON file outputted by the previous component as the input, SQL DB Modification Evaluator component graphically represent the table structures of the MySQL database. Through the graphical representation, the component allows users to select/ remove existing tables as well as columns in those tables that are to be migrated. When a user selects a table to be removed from the migration process, the evaluator component checks whether the removed table is referenced by any other table with the assistance of the JSON file. If the table that the user wishes to be removed is referenced by an existing table, the component provides the user a warning message saying that the table cannot be removed since it is referenced by a specific table and would not allow user to remove the table. However, if the referencing table has already been removed by the user, the module would update the JSON file's referencing table object by updating the corresponding name/value pair indicating that the table has been removed by the user thus facilitating users to undo the removals, while also updating the referenced table object's referenced details implying that the relationship has been removed. The same process would be applied in the removal of table columns. After all modifications have been done and evaluated, an updated JSON file would be outputted from the module.

Figure 3 : JSON File indicating Removed Column

1:1 Mapping

This module is responsible for mapping of one-to-one relationships in the database that is to be migrated. The module uses embedding is used as a mapping option since it is the preferred method of mapping one-to-one relationships in MongoDB. The module takes the previously created JSON file as the input and analyzes the file in order to identify the tables and columns which are participating in one to one relationships. One-to-one mapping module then converts data in first JSON file and then converts them into document objects and stores them in a new JSON file. The module then embeds the table document containing fewer relationships with other tables into the table document containing more relationships with other tables. The JSON file containing document objects would be the output of the component.

1: M Mapping

The module is responsible for mapping of one-to-many relationships within the migrating database appropriately. The one-to-many mapping module would receive JSON file with table and object structures along with the JSON file containing collections and documents which was outputted from the 1:1 relationship mapping module. To begin with, the component sends a request to the neural network providing the necessary parameters like number of columns, relationship type, and number of foreign keys to get the most appropriate way to map the relationship out of embedding and referencing. Following the response from the neural network, the component does its own evaluation to decide on which mapping method to use. After the mapping method has been decided, the module then starts re-structuring the JSON file containing collections. If embedding is selected as the mapping option, the one-to-many mapping module embeds each and every many side documents into the one side documents while removing the foreign key name/value pair and the many side documents from the JSON file. However, if the mapping option selected is referencing, the module would decide on the most suitable document to store the reference based on the number of many side records for a single one side record. In case of the reference being stored in the one side document, the reference would be added to the document as a name/value pair containing an array of many side object ids as the value. If the reference is stored in the many side document, a new name/value pair would be inserted to the document containing the object id of the one side document while removing the foreign key.

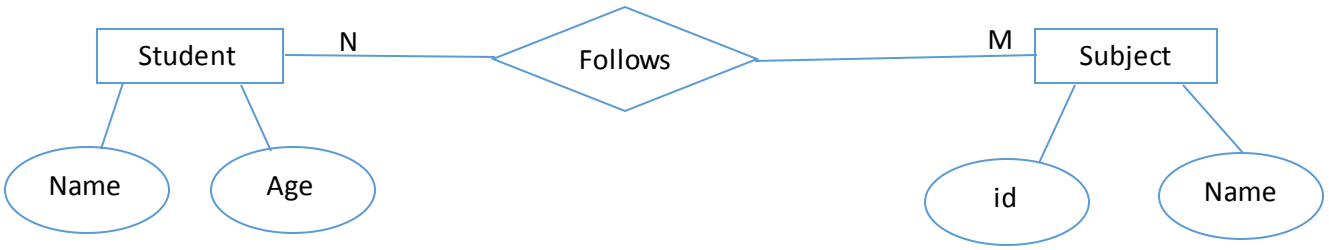
After mapping all one-to-many relationships within the database to be migrated, the updated JSON file which is having all one-to-many relationships suitably mapped, would be outputted from the module.

N : M Mapping

This module is capable of mapping SQL many-to-many relationships to MongoDB. The JSON modified in the I : N mapping module will be given an as a input to the this module. The JSON file will contain the many-to-many relationships information inside “referencedBy” array. The information about many-to-many relationships will be sent to neural network along with arguments for analyzing and selection of the best mapping option. Considering the response of the neural network, the component will map the relationships. If embedding is suggested by the neural network, the component will decide which collection need to be embedded to which collection. Then the documents which embed will be stored as an array of the parent document. After embedding all the objects the collection containing embedded documents will be deleted.

If referencing is selected as the mapping module the referenced documents object ids will be added to referencing document as an array of objects.

Consider the following sql N : M relationship as an example.



If the mapping option is embedding the data will be arranged as below:

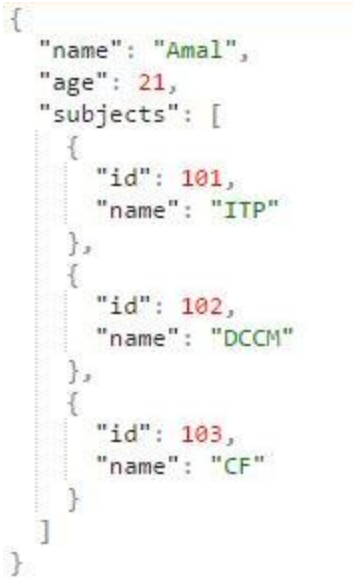


Figure 4 : Many-to-many Exmple using Embedding

If the mapping option is referencing data will be arranged like below:

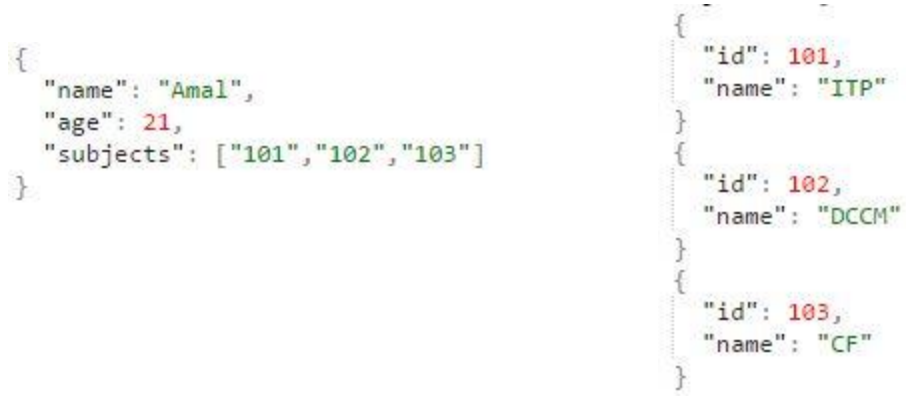


Figure 5 : Many-to-many Example using Referencing

Data Analyzer

This module is capable of analyzing a Mongo Database and show it as a graphical representation. Since the entire system is focused on novice MongoDB users they will be having trouble with MongoDB commands. All the Mongo databases will list down and user can select a

database. When user select a database the module will show available collections. If the database is a migrated database from sql it will only have structured data. But since MongoDB is an unstructured database it may contain different key value pairs in different documents in same collection. So the module will analyze what are the most common fields to collections and that data will represent in a table structure. After selecting a particular document it will show all the all the key value pairs and embedded structures in a graphical manner. The arrays and embedded documents will be shown in a nested table structure.

The other feature of this module is that the users can manage data in the same graphical view in a user friendly way. Users can add new documents to a collection. It will load a form like view with names of the common key value pairs. Then user will be able to add new key value pairs, delete key value pairs and add values to those keys. When the user presses the save button system will generate and execute corresponding command. Likewise user will be able to edit or delete existing document.

User Changes Evaluator

Mapping evaluator will evaluate the mapping changes to the MongoDB before making changes to the database. Since users has ability to change the mapping structure they will change the embedded and referenced mappings. But there will be scenarios that the selected collection can't be mapped as user's wish. This module is capable of identifying those scenarios and disabling or undoing these changes. The evaluator is also capable of identifying mistakes like data type errors, duplicate key value pairs done in data manipulation module.

Change Embedded Mapping

When user change the embedded mapping to referencing the change will evaluate by evaluator. Then request will pass to this module. It will read all the documents in the collection and create embedded documents as new documents under new collection name. Then the collections will linked by object id and the embedded documents will be deleted from the collection

Graphical Query Generator

The module facilitates users who do not have any MongoDB command or JSON knowledge to generate MongoDB commands using the graphical user interface provided. Basically the user is capable of generating a MongoDB command by only clicking and typing values with the aid of the component. The Graphical Query Generator module takes the connected MongoDB databases as the input and graphically lists the databases and collections which currently exist. For querying, users can first select the collection they want to query on, select operators or keywords and/or input the values. The module would track the user actions on the graphical interface, convert the actions into MongoDB commands and would generate the corresponding MongoDB command as the output. The Graphical Query Generator module is capable of generating MongoDB command for creating/ dropping databases, creating/ modifying collections, insert/ update/ delete documents as well as querying and projection of documents. Generation of all these command does not require any MongoDB or JSON based commands.

SQL Query Converter

The Query converter module would assist users in easily converting their MySQL queries into MongoDB commands. Unlike existing query converters, the module would support table-level create, alter and drop statements, record-level insert, update, delete statements along with join statements. Taking the MySQL query as the input, the component would first assess the user inputted MySQL query for any syntax errors. In case of any syntax errors, the component is to output an error message to the user indicating that a syntax error has been identified and that it needs to be corrected prior to the conversion. Following the verification of a syntax error-free statement, the module move on to the conversion of the statement to a MongoDB statement, which involves identifying each keyword in the MySQL statement and mapping it into the corresponding MongoDB statements. When converting join table statements, the module checks for table availability and checks documents for embedded table data. The fully converted MongoDB command would be the output of the module.

Neural Network

This module is defined to have incremental machine knowledge with the interaction of human to make decision to select NOSQL collection modeling type without having traditional procedural programming technique. This module has capability to predict and make assumption about the

future patterns on top of existing data. This Neural Network consists with input neurons namely reference type, column count of the table, data type count and so on. Neural Networks not have intelligent to understand text patterns so it is required to use text to number mapping or bipolar conversions. See below diagram for the description of the Neuron in Neural Network.

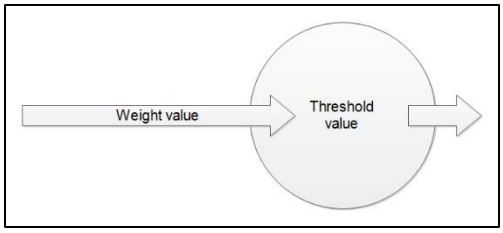


Figure 6: Process of Neuron

It is completely same as biological neuron system. It is required to have enough amount of weight value greater than threshold value in neuron to fire neuron and go through the path. This weight values are playing major role in this module and it was proposed to handle those weights using java Matrix class. We need to perform addition of matrix, dot product of vector matrix, multiplication of matrix and so on. We are planning to use supervised learning methodology to train neural network. Supervised learning methodology means we provide input data sets and expected output data sets. Then neural network try to learn itself by changing weight values dynamically and get close to the given output, then there is a different between expected output and given output. We call that difference as error of the neural network. So we need to implement another process to tune-up neural network using an algorithm like Backpropagation algorithm and minimize the amount of error.

This Neural Network is a Feedforward network. This network should consist with input layer and output layer. It could have one or more hidden layers or zero hidden layer. Feedforward network means only flow forward, no any neuron has connected itself or with previous layer.

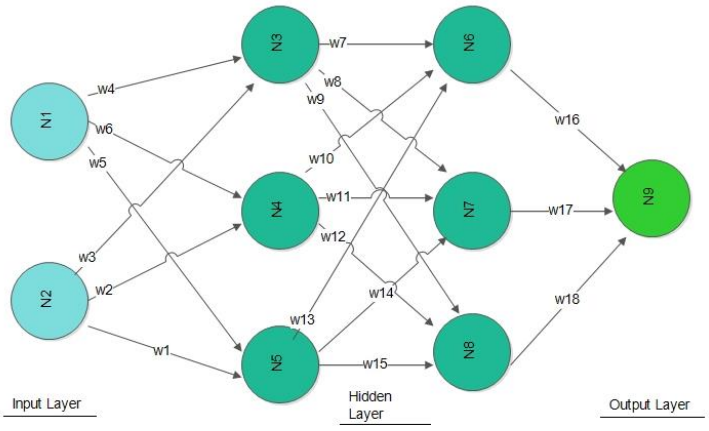
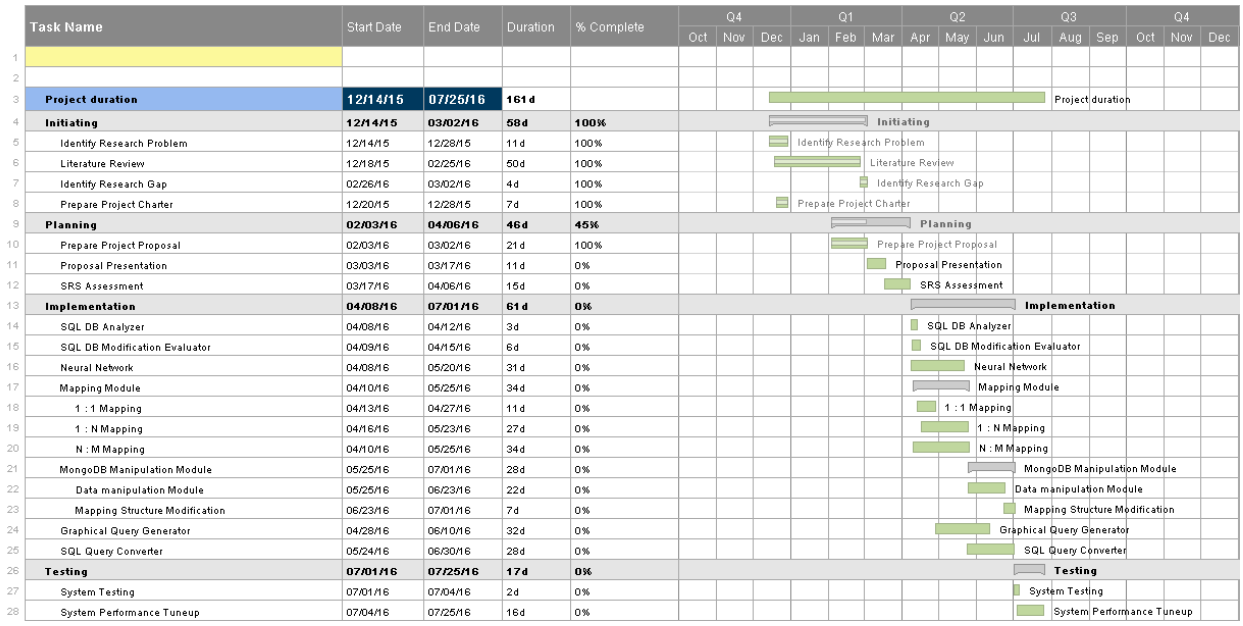


Figure 7: Neuron Map

The response returning from neural network need to map to a particular range of numbers. We use Activation function like linear activation function to scale output of neural layer. This module consist with restful application interface to distribute services of neural network to the other modules. This module will be able to transfer response as a JSON file. This Neural Network will be hosted in remote apache server and connect to the client applications via internet.

Gantt Chart



	Task Name	Start Date	End Date	Duration	% Complete
1					
2					
3	**Project duration**	**12/14/15**	**07/25/16**	**161 d**	
4	**Initiating**	**12/14/15**	**03/02/16**	**58d**	**100%**
5	Identify Research Problem	12/14/15	12/28/15	11 d	100%
6	Literature Review	12/18/15	02/25/16	50d	100%
7	Identify Research Gap	02/26/16	03/02/16	4d	100%
8	Prepare Project Charter	12/20/15	12/28/15	7d	100%
9	**Planning**	**02/03/16**	**04/06/16**	**46d**	**45%**
10	Prepare Project Proposal	02/03/16	03/02/16	21 d	100%
11	Proposal Presentation	03/03/16	03/17/16	11 d	0%
12	SRS Assessment	03/17/16	04/06/16	15d	0%
13	**Implementation**	**04/08/16**	**07/01/16**	**61 d**	**0%**
14	SQL DB Analyzer	04/08/16	04/12/16	3d	0%
15	SQL DB Modification Evaluator	04/09/16	04/15/16	6d	0%
16	Neural Network	04/08/16	05/20/16	31 d	0%
17	Mapping Module	04/10/16	05/25/16	34d	0%
18	1 : 1 Mapping	04/13/16	04/27/16	11 d	0%
19	1 : N Mapping	04/16/16	05/23/16	27d	0%
20	N : M Mapping	04/10/16	05/25/16	34d	0%
21	MongoDB Manipulation Module	05/25/16	07/01/16	28d	0%
22	Data manipulation Module	05/25/16	06/23/16	22d	0%
23	Mapping Structure Modification	06/23/16	07/01/16	7d	0%
24	Graphical Query Generator	04/28/16	06/10/16	32d	0%
25	SQL Query Converter	05/24/16	06/30/16	28d	0%
26	**Testing**	**07/01/16**	**07/25/16**	**17d**	**0%**
27	System Testing	07/01/16	07/04/16	2d	0%
28	System Performance Tuneup	07/04/16	07/25/16	16d	0%

Figure 8 : Gantt Chart

4. DESCRIPTION OF PERSONAL AND FACILITIES

Member	Component	Task
IT13001476	Neural Network	• Create a Neural Network. • Create restful api to access neural network. • Create module to learn neural network.
	Web Portal	• Create website to manage user accounts, payments and file version control.
	NOSQL Mapping Module	• Create module to convert existing referencing mapping into embedded mapping. • Create Interactive UI to handle NOSQL collection structures.
IT13088156	NOSQL Management Module	• Evaluate user changes on database mapping structure. • Access data in

		database and enable users to operate with data. • Change existing embedded mapping to reference mapping.
	Many to Many Mapping	• Select many to many tables, connect with neural network and proceed with referencing or embedding modeling method.
IT13093938	SQL DB Modification Evaluator	• Graphically represent table structures and relationships in the SQL dump file • Allow users to remove unwanted tables/ columns prior to migration • Evaluate user modifications for any violations of constraints • Update JSON file recording user changes
	One-to-many Mapping	• Analyze tables with one-to-many relationship and

		compare with neural network response • Update JSON file embedding or storing references accordingly
	SQL Query Converter	• Analyze inputted SQL query for any syntax errors • Output the converted query as a MongoDB command
IT13075422	SQL DB Analyzer	• Creation of connection manager connecting MySQL and MongoDB • Allow users to upload SQL file • Analyze SQL file for any syntax errors • Creation of JSON file including table structured and data • Update relationships types of tables analyzing data
	One-to-one Mapping	• Analyze tables and select the tables to be embedded • Creation of JSON file containing document objects

	Graphical Query Generator	• List connected MongoDB databases • Allow users to select, click and type values accordingly • Track user actions and generate MongoDB command accordingly

Table 2 : Personal and Facilities

5. BUDGET (If Any)

Budget		
Expenses	Per-person	Total
Communication Charges	2000	8000
Document printings	2000	8000
Travelling	4600	23000
Total	8600	39000

Table 3 : Budget

6. REFERENCES

[1] Margaret Rouse, "relational database," *searchsqlserver.techtarget.com*, para. 1, April, 2006. [Online]. Available: http://searchsqlserver.techtarget.com/definition/relational-database . [Accessed: Feb.18, 2016].

[2] Wikipedia Contributors, "NoSQL", *Wikipedia, The Free Encyclopedia,* Wikipedia, The Free Encyclopedia, [online document], Feb 18, 2016. Available: Wikipedia, https://en.wikipedia.org [Accessed: Feb.21, 2016].

[3] Gansen Zhao, Weichai Huang, Shunlin Liang and Yong Tang, "Modeling MongoDB with Relational Model", in *4th International Conference on Emerging Intelligent Data and Web Technologies,* Xi'an, China, Sept. 9-11, 2013, p. 115 – 121. Available: IEEE Xplore, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6631603&newsearch=true&queryText=sql%20to%20nosql%20data%20migration. [Accessed: Feb.10, 2016].

[4] Gansen Zhao, Weichai Huang, Shunlin Liang and Yong Tang, "Schema Conversion Model of SQL Database to NoSQL", in 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Guangdong, China, Nov. 8-10, 2014, p. 355 - 362. Available: IEEE Xplore, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7024609&newsearch=true&queryText=sql%20to%20nosql%20data%20migration . [Accessed: Feb.10, 2016].

7. APPENDICES (If Any)