Bhandari and Chitrakar, J Comput Eng Inf Technol 2020, 9:6
DOI: 10.37532/jceit.2020.9(6).241

Journal of Computer Engineering & Information Technology

Review Article

A SCITECHNOL JOURNAL

Comparison of Data Migration Techniques from SQL Database to NoSQL Database

Hira Lal Bhandari*, and Roshan Chitrakar

Abstract

With rapid and multi-dimensional growth of data, Relational Database Management System (RDBMS) having Structured Query Language (SQL) support is facing difficulties in managing huge data due to lack of dynamic data model, performance and scalability issues etc. NoSQL database addresses these issues by providing the features that SQL database lacks. So, many organizations are migrating from SQL to NoSQL. RDBMS database deals with structured data and NoSQL database with structured, unstructured and semi-structured data. As the continuous development of applications is taking place, a huge volume of data collected has already been taken for architectural migration from SQL database to NoSQL database. Since NoSQL is emerging and evolving technology in the field of database management and because of increased maturity of NoSQL database technology, many applications have already switched to NoSQL so that extracting information from big data. This study discusses, analyzes and compares 7 (seven) different techniques of data migration from SQL database to NoSQL database. The migration is performed by using appropriated tools / frameworks available for each technique and the results are evaluated, analyzed and validated using a system tool called SysGauge. The parameters used for the analysis and the comparison are Speed, Execution Time, Maximum CPU Usage and Maximum Memory Usage. At the end of the entire work, the most efficient techniques have been recommended.

Keywords

Data Migration; MySQL; RDBMS; Unstructured Data; SysGauge

Introduction

In 1970, Edgar Frank Codd has introduced architectural framework on the relational database approach in his paper."A relational model of data for large shared data banks" [**1**]. After some time Codd has introduced Structured English Query Language and later has renamed it as Structured Query Language to provide a way to access data in a relational database [**2**]. Since then, relational model has had dominant form in the database market.The most popularly has used database management systems are Oracle, Microsoft SQL server and MySQL [**2**]. All these three DBMS are based on relational database model and use SQL as query language.When NoSQL database has been introduced by Carlo Strozzi in 1998 as a file based database, it has been used to represent relational database without using Structured Query Language. However, it has not be able to compete with relational database. Later Eric Evans an employee in Rackspace Company explained the ambition of the NoSQL movement as a new trend to solve a problem that Relational Databases are not fit. The increasing usage of NoSQL products have energized other companies to develop their own solutions and headed to emerge of generic NoSQL database systems. This way there are more than 150 NoSQL products. These products come with issues like suitability to some areas of application, security and reliability [**3**].

NoSQL databases are emerging from last few years due to its less constrained structure, scalable schema design, and faster access in comparison to relational databases. The key attributes that make it different from relational database are that it does not use the table as storage structure of the data. In addition, its schema is very efficient in handling the unstructured data. NoSQL database also uses many modeling techniques like key-value stores, document data model, and graph databases [**1**].

This research study aims to present comparative study on data migration techniques from SQL database to NoSQL database. This study analyses 7 (seven) recent approaches [**4**] which have been proposed for data migration from SQL database to NoSQL database.

Statement of the problem

There is nothing wrong in using traditional RDBMS for database management. As huge introduction of data from social sites and other digital media, it simply isn't enough for the application dealing with huge databases. Also, NoSQL databases need cheap hardware. Hence, requirement of some of the relational databases need to be converted to NoSQL databases which then enable to overcome drawbacks found in relational databases. Some drawbacks of relational database management systems are:

1. They do not encompass a wide range of data models in data management.
2. They are not easily scalable because of their constrained structure.
3. They are not efficient and flexible for unstructured and semi-structured database.
4. They cannot handle data during hardware failure.

Due to massive use of mobile computing, cloud computing, Internet of Things, and other so many digital technologies, large volume of streaming data is available nowadays. Such huge amounts of data take a great deal of challenges to the traditional relational database paradigm. Those challenges are related to performance, scalability, and distribution. To overcome such challenges enterprises begin to move towards implementing new database paradigm known as NoSQL [**5**].

On the other hand, NoSQL database contains several different models for accessing and managing data, each suited to specific use cases. This is also significant reason to migrate data from SQL database to NoSQL database. The several models are summarized in the **Table 1**.

NoSQL DBMSs are distributed, non-relational databases. They

***Corresponding author: Hira Lal Bhandari, Faculty of Science Health and Technology Nepal Open University, Nepal. E-mail: drw.moon@gmail.com**

Received: November 01, 2020 **Accepted:** December 14, 2020 **Published:** December 21, 2020

doi: 10.37532/jceit.2020.9(6).241

Table 1: NOSQL database models.

Model	Characteristics
Document Store	Data and metadata are stored hierarchi-cally in JSON-based documents inside the database.
Key Value Store	The simplest of the NoSQL Databases, data is represented as a collection of key-value pairs.
Wide-Column Store	Related data is stored as a set of nested-key/ value pairs within a single column.
Graph Store	Data is stored in a graph structure as node, edge, and data properties.

are designed for large-scale data storage and for massive parallel data processing across a large number of commodity servers. They use non-SQL languages and mechanisms to interact with data. Use of NoSQL database systems in database management increased in major Internet companies, such as Google, Amazon, and Facebook; which has aroused challenges in dealing with huge quantities of data with conventional RDBMS solutions could not cope. These systems can support multiple activities, including exploratory and predictive analytics, ETL-style data transformation, and non-mission critical OLTP. These systems are designed so as to scale up thousands or millions of users doing updates as well as reads, in contrast to traditional DBMSs and data warehouses [**6**].

The focus of the study is to get comparative study on different seven techniques to migrate data from relational database to NoSQL database. Migration of data from relational database to NoSQL database refers the transformation of data from structured and normalized database to flexible, scalable and less constrained structure NoSQL database. The main objective of this research is to find out the most efficient data migration technique among seven major migration techniques from SQL database to NoSQL database.

Scope and Limitations of the Research Study

Scope and limitation of this research covers the following: This study is focused to get analyzed with different techniques to migrate the data from SQL database to NoSQL database to know efficient migration technique so that one can efficiently adapt emerging technology in the database world. Therefore, the study does not include technical discussion of the risks identified, or of the implementation guideline here. The demand for NoSQL databases is increasing because of their diversified characteristics that offer rapid, smooth, scalability, great availability, distributed architecture, significant performance and rapid development agility. It provides a wide range of data models to choose from and is easily scalable where database administrators are not required. Some of the SQL to NOSQL data migrating providers like Riak and Cassandra are programmed to handle hardware failures and are faster, more efficient and flexible. It has evolved at a very high pace.

However, some data migration techniques and NoSQL is still immature and they do not have standard query language. Some NoSQL databases are not ACID compliant. No standard and data loss are the major problems while migrating data from SQL database to NoSQL database.

Review of Related Works

This research study provides the comparative study on different data migration approaches from SQL database to NoSQL databases. This focuses on the study of major migration techniques and suggests the efficient approach for data migration. Migrating process is performed with the help of tools/ framework available.

SQL database and other traditional databases strictly follow structured way to organize the data generated from various applications but NoSQL databases provide flexibility and scalability in organizing the data which makes it easy to access the data. The data generated from social networking sites and real time applications needs flexible and scalable system which increases the requirement of NoSQL. Hence, multidimensional model has been proposed for data migration. The biggest challenge is the migration of existing data residing in data warehouse to NoSQL database by maintaining the characteristics of the data. The growing use of web applications has raised the demand to use NoSQL because traditional databases are unable to handle the rapidly growing data [**4**].

The concept of NoSQL was first used in 1998 by Carlo Strozzi to represent open source database that does not use SQL interface. Strozzi likes to refer to NoSQL as "noseequel" since there is difference between this technology and relational model. The white paper published by Oracle mentions techniques and utilities for migrating non Oracle databases to Oracle databases [**7**]. Abdelsalam Maatuk [**8**] describes an investigation into approaches and techniques used for database conversion. Its origin is also regarded to the invention of Google's BigTable model. This database system, BigTable, is used for storage of projects developed by Google, for example, Google Earth. BigTable is a compressed high performance database which was initially released in 2005 and is built on the Google file system. It was developed using C and C++ languages. It provides consistency, fault tolerance and persistence. It is designed to scale across thousands of machines and it is easy to add more machines to it [**9**]. Later, Amazon developed fully managed NoSQL database service DynamoDB that is used to provide a fast, highly reliable and cost effective NoSQL database services designed for internet scale applications [**9**]. These projects directed a step towards the evolution of NoSQL.

However, the term re-emerged only in 2009, at a meeting in San Francisco organized by Johan Oskarsson. The name for the meeting, NoSQL meetup, was given by Eric Evans and from there on NoSQL became a buzzword [**8**]. Many early papers have talked about the relationship between Relational and NoSQL Databases which gave a brief introduction of NoSQL database, its types and characteristics. They also discussed about the structured and non-structured database and explained how the use of NoSQL database like Cassandra improved the performance of the system, in addition to it can scale the network without changing any hardware or buying bigger server. This result is improving the network scalability with low-cost commodity hardware [**10**].

Sunita Ghotiya [**4**] gave literature review of some of the recent approaches proposed by various researchers to migrate data from Relational to NoSQL databases. Arati Koli and Swati Shinde [**11**] presented comparison among five different techniques to migrate from SQL database to NoSQL database with the help different research paper reviews. Shabana Ramzan, Imran Sarwar Bajwa and Rafaqut Kazmi [**12**] stated the comparison of transformation in tabulated format with different parameters such as source database, target database, schema conversion, data conversion, conversion time, data set, techniques, reference papers which clearly shows the research gap that currently no approach or tool supports automated transformation of MySQL to Oracle NoSQL for both data and

doi: 10.37532/jceit.2020.9(6).241

schema transformation. Arnab Chakrabarti and Manasi Jayapal [**13**] presents empirical comparative study to compare and evaluate data transformation methodologies between varied data sources as well as discuss the challenges and opportunities associated with those transformation methodologies. The database used in transformation was heterogeneous in nature.

In this way, this research study explores the issues regarding relational databases, their features and shortcomings as well as NoSQL and its features. It emphasizes on comparative study on the migration approaches from structured (SQL) database to NoSQL database. In this present scenario maximum application are to be transformed into NoSQL databases because of incremental growth of heterogeneous data. In such condition, SQL database is no more has the ability to handle such complex dataset. So, there is the need of migration of structured and normalized dataset into NoSQL database. In this manner, the research study is focused on performing major migration techniques to transfer data from SQL database i.e. MySQL to NoSQL databases i.e. MongoDB, Hadoop database, etc. Major seven migrating approaches are discussed and used to perform migration task.

This comparative study presented in this research study could be as guide lines for the organizations which are shifting their application towards NoSQL databases. This research will be helpful choose the efficient migration approach to transfer structured and normalized database into NoSQL database.

Methodology

This research study evaluates major migration approaches which have been proposed in the previous research papers. The evaluation is done through comparative study on the migration approaches efficiency measurement with different parameters. They are Speed, Execution Time, Maximum CPU Usage, and Maximum Memory Usage. Migration of data from SQL database to NoSQL database belonging to different migration approaches is done using available framework/tools.

In the **Figure 1** we have presented the workflow that has been followed during the entire process of data transformation. This helps to systematically run and verify each job as it was essential in concluding the study among major migrating approaches performed. This way we can trace the most efficient migration approach to transform data from traditional normalized Database to NoSQL database.

Figure 1 shows how data is migrated from source data store to destination data store i.e. SQL database to NoSQL databases. Here in the diagram each migration approach is planned to implement with the help of respective technology i.e. tools/ framework. Data store 1 signifies SQL database i.e. MySQL and data store 2 implies MogoDB and HBase. Up to the migrating process completion, SysGauge tool is run to check either other processes are run or not. If there are processes running that will be shut down, then only the migration technology run for respective migration approaches using tools/ framework.

Data Description

The source of sample database to migrate from SQL database to NoSQL data. Database used in the migrating process is structured database. Data set containing in the database table consists of 1000 number of records. The database table schema is presented below

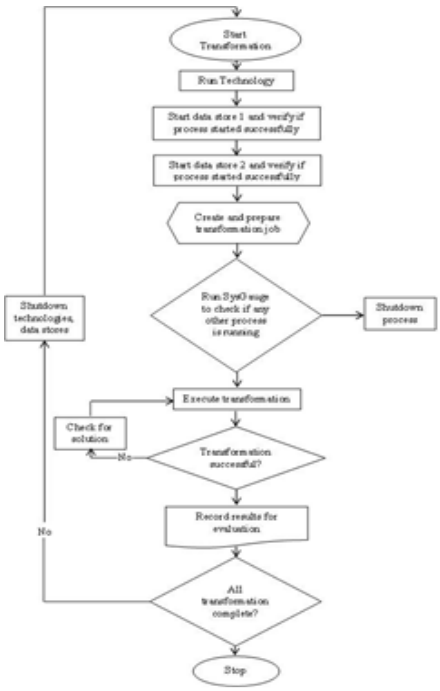

Figure 1: Workflow to run the transformation.

which clarifies the structure of data. **Table 2** includes six different columns and seven different rows. First column consists of fields such as user id, user name, last name, Gender, password and Status. They have int and varchar data type. int basically the numeric data type and varchar is the character data type.

Environment and Comparison Characteristics

Implementation Details: This section includes the details of implementation of the study in which an experiment to execute the data migration between the data stores was setup. Microsoft Windows machine with the following configuration is used to run all type of data migration approaches using respective tools **Table 2.1**.

Only the migrating tools and concerned database were allowed to run whereas all others shut down to make sure that no other variable had impact on the result. After the completion of each job, the tools and databases were restarted. SysGauge tool was used to analyse the processes running on the machine with respect to the CPU and memory utilization. The process specific to the technology was studied using 'SysGauge' and the quantitative characteristics like maximum CPU, Memory and Time are documented as Maximum CPU load, Maximum Memory Usage and CPU Time respectively. **Figure 2** shows an instance of the SysGauge tool in which the characteristics are highlighted.

Characteristics of Comparison: In this section, a set of well-defined characteristics have been discussed which can be considered for comparative study. Previous study stated NoSQL databases are often evaluated on the basis of scalability, performance and consistency. In addition to that system or platform dependent characteristics there could be complexity, cost, time, loss of information, fault tolerance and algorithm dependent characteristics could be real time processing, data size support etc. To meet the scope of this research, quantitative characteristics are considered hence actual values are

doi: 10.37532/jceit.2020.9(6).241

Table 2: NOSQL database models.

Field	Type	Null	Key	Default	Extra
user_id	Int(11)	No	PRI	Null	auto_increment
user_name	varchar(255)	Yes		Null	-
last_name	varchar(50)	Yes		Null	-
Gender	varchar(50)	Yes		Null	-
password	varchar(50)	Yes		Null	-
Status	varchar(50)	Yes		Null	-

Table 2.1: NOSQL database models.

Processor	Intel® Core(TM)i3-3217U CPU@1.80 GHZ
Installed Memory (RAM)	2.00 GB
Operating System	Windows 7 Professional
Processor type	64-bit
Hard disk	500 GB

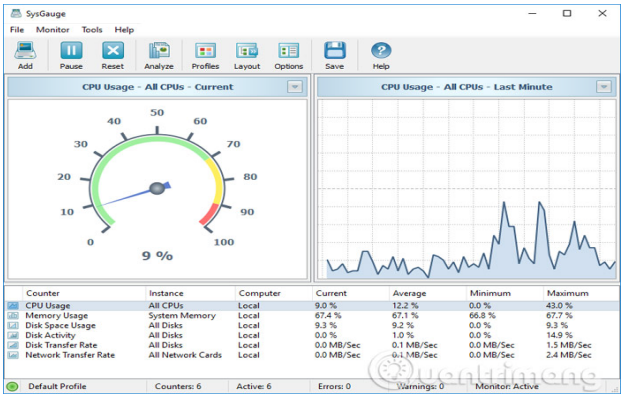

Figure 2: SysGauge Instance.

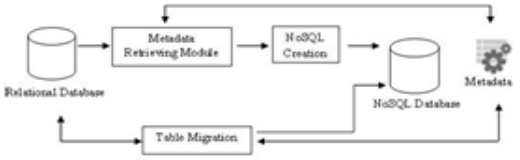

Figure 3: Migration Module Working Diagram.

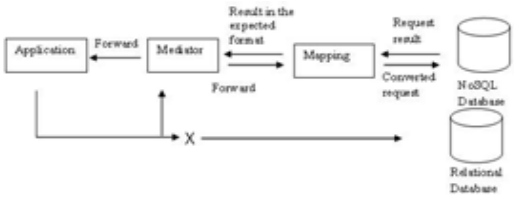

Figure 4: Data Mapping Module Working Diagram.

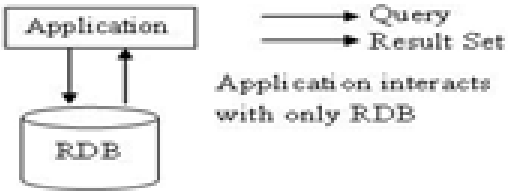

Figure 5: Original system with RDB only.

retained and can be traced actual result observed from performing the migration of data from SQL database to NoSQL database. These numerical aspects were carefully studied before collecting the data to give the best comparative **Figures 3-5**. We present the metrics that have been used to evaluate our results.

Maximum CPU Load: This refers maximum load percentage of the processor time used by the processor during the data migration. This is a key performance metric and useful for investing issues was monitored by shutting down all other unnecessary processor technologies management.

Maximum Memory Usage: Maximum memory usage refers maximum percentage of the physical RAM used by the process during data migration. An important metric to keep a track of resource consumption and impact it has on the time.

Analysis of changes in the resource consumption is an important performance metric. Maximum CPU load, CPU time and maximum memory usage were calculated for each of the migration approaches using SysGauge tool in Windows operating system.

Execution Time: It is the total time taken to complete the data migration. This was measured using the respective tools for the migration techniques to compare the faster means of migrating data between SQL databases to NoSQL databases. This time included the time taken to establish a connection to the source and destination databases, reading data from the source and writing data to the destination. As a common unit, all the results were converted into second. However, some migration took long time to complete, were expressed in minutes.

Speed: speed is computed as the size of data transformed per second. For each of the migration techniques, this value was obtained from the tools using which migration was performed. The value of speed was important, for example, in the migration of data from MySQL to MongoDB database.

Methods of Migration

While comparing SQL databases with NoSQL databases, the structure is more complex because they use structured way to access and store data as well as the concept of normalization. According to the rules of normalization they split their information into different tables with join relationship. On the other hand, NoSQL Databases store their information in a de-normalized way which is unstructured or semi-structured. Therefore the successful migration with data accuracy and liability from relational to NoSQL would not be an easy task. To come to the conclusion, comparison of major data migration techniques is done with the help of different tools such as MysqlToMongo, phpMyAdmin, Sqoop, Mysq l2 etc. Speed, Execution Time, Maximum CPU Usage and Maximum Memory Usage are checked for the comparison of major approaches for data migration from relational to NoSQL database.

Mid-model Approach using Data and Query Features: This model is used for transition and for migration of data from SQL database to NoSQL database. This model works on two basic concepts: Data features and query features. First mid model is migrated to the physical model which is destination database and when it is successfully performed the data is migrated from SQL to NoSQL Databases [**4**].

doi: 10.37532/jceit.2020.9(6).241

To perform the migration task an application 'MysqlToMongo' is used to perform migration of data using its data and query features.

Algorithm 1 Mid-model approach (For executed transaction)

Goal:

Execute Transaction
Assumption:
Once Transaction start execution it's not interrupted
Input: Keys:
in which transactions will operate
Operations:
(kind of operation required for each key read, write) Data of each Sub Transaction and operations resides in memory of layer
Output:
Transaction Data Steps
1. Inform data migration to get data of the required keys.
2. If data is ok in memory
3. Inform secondary middle layer to start execution.
4. Lock data in key status by saving the required operation on it.
5. For each key in SubTransactionKeys
6. do the required operation using current data in memory
7. write operation with data
8. If Transaction.status=="Running"
9. Transaction.status="Completed" so no transaction can in-terrupt it
10.Update data in layer Memory
11. If(updated data status is delete)
12. State Change current data status to delete
13. Else if(current data status is insert)
14. Leave it Insert
15. Else If (current data status is update)
16. Leave it update
17. End return selected data
18. Else
19. Go to Transactions In waiting

Algorithm 2 Mid-model approach (For waiting transaction)

Goal:

Execute Transaction

Input: Keys:

In which transactions will operate

List of currently locked keys and operations in each key (Read or

Write) reside in locked table

List of Waiting Transactions (transactions in waiting that arrive

Before current transaction and use any of keys associated with

Current transaction) for current transactions

Output:

Locking keys of transaction and go to transaction execution

Steps:

1. while (transaction.status=="waiting")
2. if(no keys were locked)
3. transaction.status="running"
4. go to Execute Transaction
5. else
6. for each transaction in waiting transaction
7. if (all transaction status==completed or errored)
8. remove all keys from locked table
9. current transaction.status=running
10. go to execute transaction

NoSQLayer Approach: This migration approach works on the basis of two modules: Data Migration Module and Data Mapping Module. In this data migration module the elements for example, column and row are identified from source database and then they are mapped automatically into NoSQL model. Data-mapping module consists of the persistence layer, designed to be an interface between the application and the DBMS, which monitors all SQL transactions from the application, translates these operations and redirects to the NoSQL model created in the previous module. Finally, the result of each operation is treated and transformed to the standard expected by the SQL application. The pictorial representations presented below describe each of these modules [**11**].

This migration approach migrate dataset from MySQL to MongoDB. To perform the NoSQLayer migrating process, software 'MysqlToMongo' is used so that data is migrated from MySQL to MongoDB. MysqlToMongo is data conversion software that helps database user to convert MySQL database data to MongoDB.

Content Management System Approach for Schema De-normalization: Almost all web-based applications and Content Management System (CMS) solutions are using Relational databases for data management. But, when users of internet and clouds are growing rapidly, it is difficult for relational databases to handle the huge data traffic. This is why database design approach has transformed the real CMS SQL database to a NoSQL database. This approach consists of two steps, first to de-normalize the SQL database and then to choose a unique identifier key as a primary key for a big table [**12,13**]. Conversion from RDBMS TO NOSQL by schema mapping and migration, centered on two forms of analysis: qualitative and quantitative. In the evaluation, goal of qualitative is to provide a proof of concept by showing the schema migration and mapping framework execution in practice, in the quantitative one we aim to verify whether the application of NoSQL, with our framework, leverages the system performance [**14**].

Schema migration and query mapping framework consist of: Schema Migration Layer, Reverting Normal Forms and Row-key Selection, and Schema Migration.

Algorithm below shows a schema migration algorithm that uses table-level de-normalization. We first generate a schema graph from the relational schema and make it acyclic if needed. We then transform the schema graph into a set of schema trees. For each schema tree, we create a collection for the root node and replace a foreign key in each node with the child node that the foreign key refers to (i.e., primary key table).

Algorithm 3 A schema migration using table-level de-normalization

Input: relational schema RS

Output: MongoDB schema

doi: 10.37532/jceit.2020.9(6).241

1. Generate a schema graph G from RS
2. Make G acyclic based on user's decision if needed
3. Transform G into a set ST of schema trees
4. for (each schema tree T ST) {
5. create a collection for the root of T
6. for (each non-root node n of T)
7. embed n into the parent node np of n
8. remove the foreign key in np that refers to n
9. }
10. }

HBase Database Technique: HBase is the Hadoop database, a distributed and scalable big data store. HBase consists of some features such as linear and modular scalability, strictly consistent reads and writes convenient base classes for backing Hadoop Map Reduce jobs with Apache HBase tables [**15**]. By using Sqoop we can import information from a NoSQL database from social website framework into HDFS. The information to the import procedure is a database table. Sqoop read the table column by line into HDFS [**16**].

When direct access is available to the RDBMS source system, we may choose for either a File Processing method if not we may choose RDBMS processing while database client access is available [**17**].

Algorithm 4 Migration from MySQL to HBase

1. Steps to migrate from MySQL to HBase
2. Setup Hadoop on the system.
3. Use Sqoop to migrate data (tables) from MySQL to Hadoop Distributed File System.
4. Convert the data stored in HDFS to a designated data store format such as XML or CSV etc.
5. Setup HBase on top of the Hadoop framework.
6. Map the data onto tables created on the HBase – column oriented database based on the data access needs of the applications.

Data Adapter Approach: The data adapter system is highly modularized, layered between application and databases. It is basically lies on the concept of performing queries from applications and data transformation between databases at the same time. This system provides a SQL interface to parse query statements that enables to access both a Relational database and a NoSQL database.

This approach offers a mechanism to control the database transformation process and to let applications perform queries whether target data (table) are being transformed or not. After data are transformed, we get a patch mechanism to synchronize inconsistent tables [**18**]. We present the data adapter system with its design and implementation in following manner.

Without using adapter i.e. mysq l2, available system only allows application to connect to a relational database. **Figure 6** depicts

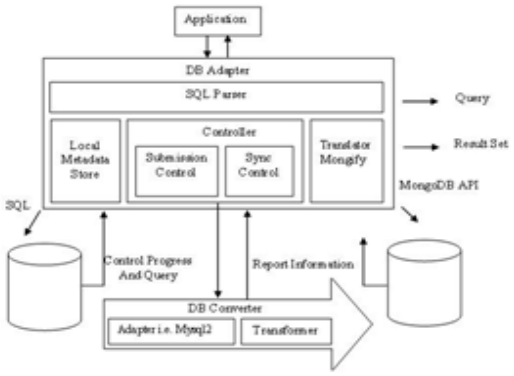

Figure 6: System architecture with data adapter and its components.

the architecture of data adapter system consisting of: a Relational Database, a NoSQL Database, DB Adapter, and DB Converter. Above mentioned system is the coordinator between applications and two databases. It controls query flow and transformation process. The DB Converter is needed for transformation of data and reporting transformation progress to DB Adapter for further actions.

Application i.e. Ruby on rails access databases through the DB Adapter i.e. mysq l2. The DB Adapter parses query, submits query, and gets result set from databases. The system needs some necessary information such as transformation progress from DB Converter, and then decides when the query can be performed to access database. DB Converter migrate data from a relational database to a NoSQL database. The data adapter system accepts queries while the transformation is performed, but the data in two databases may not be consistent. The DB Adapter will detect and ask DB Converter to perform synchronization process to maintain data consistency. Automatic Mapping Framework: This approach of migration provides a framework which is generally used for automatic mapping of Relational databases to a NoSQL database. Data migration to a Column-oriented database is beneficial for several cases because the data can be appended on one dimension that is technically simpler and faster: the data are added one after the other, thus arouses much higher write speeds with very low latency. This technique consists of better scalability since the development of data is done only on one dimension their partitioning is simpler to perform and can be distributed across multiple servers [**12**].

Framework 'NoSQLBooster' is used for MongoDB for automatic database mapping from MySQL to MongoDB. NoSQLBooster for MongoDB (formerly MongoBooster) is a shell-centric cross-platform GUI tool for MongoDB, which provides comprehensive server monitoring tools, fluent query builder, SQL query, ES2017 syntax support and true intelligence experience.

Here is an algorithm of automatic mapping of MySQL relational databases to MongoDB. The algorithm uses the MySQL INFORMATION SCHEMA that provides access to database metadata. Metadata is data about the data, such as the name of a database or table, the data type of a column, or access privileges. INFORMATION SCHEMA is the information database, the place that stores information about all the other databases that the MySQL server maintains. Inside INFORMATION SCHEMA there are several read-only tables. They are actually views, not base tables.

Algorithm 5 Automatic Migration Framework

1. Creating the MongoDB database. The user must specify the MySQL database that will be represented in MongoDB. The database is created with the following MongoDB command: use DATABASE NAME.
2. Creating tables in the new MongoDB database. The algorithm verifies for each table in what relationships is involved, if it has foreign keys and/or is referred by other tables.
3. If the table is not referred by other tables, it will be represented by a new MongoDB collection.
4. If the table has not foreign keys, but is referred by another table, it will be represented by a new MongoDB collection.
5. If the table has one foreign key and is referred by another table, it will be represented by a new MongoDB collection. In our framework, for this type of tables we use linking method, using the same concept of foreign key.
6. If the table has one foreign key but is not referred by another table, the proposed algorithm uses one way embedding model. So, the table is embedded in the collection that represents the table from the part 1 of the relationship.
7. If the table has two foreign keys and is not referred by another table, it will be represented using the two way embedding model, described in section 2.4.
8. If the table has 3 or more foreign keys, so it is the result of a N:M ternary, quaternary relationships, the algorithm uses the linking model, with foreign keys that refer all the tables initially implied in that relationship and already represented as MongoDB collections. The solution is good even the table is referred or not by other tables.

Extract-Transform-Load approach: The term ETL came into existence from data warehousing and is an acronym for Extract-Transform-Load. ETL insists a process of how the data are loaded from the source system to the data warehouse [**19, 20**]. In these days, the ETL enhances a cleaning step as a separate step. The sequence is then Extract-Transform-Load.

Extract: The Extract step consists of the data extraction from the source system and makes it accessible for further processing. The main aim of the extract step is to fetch all the necessary data from the source system with as minimal amount of resources as possible.

Transform: The transform step applies a set of rules to transform the data from the source to the target. This includes converting any measured data to the same dimension using the same units so that they can later be joined. The transformation step also requires joining data from several sources, generating aggregates, generating surrogate keys, sorting, deriving new calculated values, and applying advanced validation rules.

Load: During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible. The target of the Load process is often a database. In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes. The referential integrity needs to be maintained by ETL tool to ensure consistency.

Steps: -

1. Lock the target database in source system.
2. Lock the target database in destination system.
3. Extract information from target database from Source system.
4. Transform information to destination database.
5. Release lock of source and destination systems.

Discussion

In this section we discuss the results of the experiment and also report the challenges that we faced during the entire phase.

Comparing Quantitative Characteristics of Migration Approaches: This determinative evaluation was used to check if the study is going in the right direction. The data migration methodologies which were implemented in this research study are compared with one another and evaluated in the matrix as described. Since each aspect cannot be predicted at the initial of the study and due to unexpected changes that happened at different phases, a revision of the methodologies was necessary at every stage.

Migrating Results

An implementation details as described earlier was environmental setup; the values of maximum CPU load, CPU time, and maximum memory usage are retrieved using the SysGauge tool, outcome of execution time, speed are documented from the respective technology used in the migration process and the results are compiled as shown in the **Table 3**. There were 3 target data stores such as MongoDB, CMS Database and Hadoop Database used in the research study. The tools and framework involved in the transformation were MysqlToMongo, phpmyadmin, mysq l2, NoSQLBooster for MongoDB, Sqoop and Studio 3T.

Transformation result varies from one migration technique to another technique that was evaluated according to the values retained from execution of respective methodologies. That execution was performed with the help of tools or framework which belongs to different migration approaches. Evaluated result of different migration approaches are discussed below:

Mid-model Approach using Data and Query Features: MongoDB using MysqlToMongo Framework): MysqlToMongo tool is used to migrate data from MySQL to MongoDB. It uses data and query features. It transforms structured data of size 2833.3 KB per second from MySLQL to MongoDB. Data set having size 85 KB and including data 1000 rows is transformed in 0.03 sec. At the time of data transformation from MySQL to MongoDB using MysqlToMongo tool, Maximum CPU Usage is 23 percentage and Maximum memory consumption is 9.1 percentage and after transformation and conversion of SQL database is 4 Kb.

doi: 10.37532/jceit.2020.9(6).241

Table 3: NOSQL database models.

Approaches	Speed (Kb/ sec.)	Execution time (sec.)	Maximum CPU Usage	Maximum Memory Usage
Mid-model approach	2833.3	0.03	23	9.1
NoSQLayer approach	8500	0.01	21	7.1
Content Management System approach	44.97	1.89	26	50
HBase database Technique	0.39	215.4	84	59.4
Data Adapter Approach	850	0.1	14	5.4
Automatic Mapping Framework	56.67	1.5	63	16.9
Extract-Transform-Load approach	1214.29	0.07	70	17.6

NoSQLayer Approach: MysqlToMongo tool was used migrating data from MySQL to MongoDB. It uses data and query features. It transforms structured data of size 8500 KB per second from MySLQL to MongoDB. Data set having size 85 KB and including data 1000 rows is transformed in 0.01 sec. At the time of data transformation from MySQL to MongoDB using MysqlToMongo tool, Maximum CPU Usage is 21 percentage and Max-imum memory consumption is 7.1 percentage and after transformation and conversion of SQL database is 1 Kb.

Content Management System Approach for Schema De-normalization: It transforms structured data of size 44.97 KB per second from MySLQL to Word press. Data set having size 85 KB and including data 1000 rows is transformed in 1.89 sec. At the time of data transformation from MySQL to Word press using phpmyadmin, Maximum CPU Usage is 26 percentages and Maximum memory consumption is 50 percentages and after transformation and conversion of SQL database is 84.7 Kb.

HBase Database Technique: It transforms structured data of size 0.39 KB per second from MySLQL to Hadoop database using Sqoop. Data set having size 85 KB and including data 1000 rows is transformed in 215.4 sec . At the time of data transformation from MySQL to Hadoop database, Maximum CPU Usage is 84 percentages and Maximum memory consumption is 59.4 percentages, and after transformation and conversion of SQL database is 65.1 KB.

Data Adapter Approach: It transforms structured data of size 850 KB per second from MySLQL to MongoDB Database using mysq l2 data adapter on ruby on rails. Data set having size 85 KB and including data 1000 rows is transformed in 0.1 sec. At the time of data transformation from MySQL to Hadoop database using Sqoop, Maximum CPU Usage is 14 percentage and Maximum memory consumption is 5.4 percentage, and after transformation and conversion of SQL database is 88 KB.

Automatic Mapping Framework: It transforms structured data of size 56.67 KB per second from MySLQL to MongoDB Database using NoSQLBooster for MongoDB. Data set having size 85 KB and including data 1000 rows is transformed in 1.5 sec. At the time of data transformation from MySQL to Hadoop database using sqoop, Maximum CPU Usage is 63 percentage and Maximum memory consumption is 16.9 percentage, and after transformation and conversion of SQL database is 1 KB.

Extract-Transform-Load Approach: It transforms structured data of size 1214.29 KB per second from MySLQL to MongoDB Database using Studio 3T. Data set having size 85 KB and including data 1000 rows is transformed in 0.07 sec. At the time of data transformation from MySQL to MongoDB database, Maximum CPU Usage is 70 percentages and Maximum memory consumption is 17.6 percentages, and after transformation and conversion of SQL database is 88 KB.

From the evaluated results during migration of data set from SQL Database to NoSQL database. In totality, 'Data Adapter Approach' was found the most efficient from the point of CPU Usage and Memory Usage. On the other hand, NoSQLayer Approach is the most efficient from execution time and data migration speed point of view. Basis of comparison were Speed, Maximum CPU Usage percentage, Maximum Memory Usage percentage and Execution Time. The resource consumption of migrating procedure was evaluated using 'SysGauge' tool. Data conversion/ transformation speed and total execution time were evaluated using framework/ tools regarding respective migration approach.

Migrating Efficiency of Transformation Techniques: The overall evaluation of all transformation techniques involved in transforming data from SQL Database i.e. MySQL to NoSQL Databases such as MongoDB, Hadoop Database and CMS Database have been plotted as shown from **Figure 7-10**. This provides a clear picture of which technology was the most efficient in comparison to the others. The average data size per second, Database size, Maximum CPU Usage

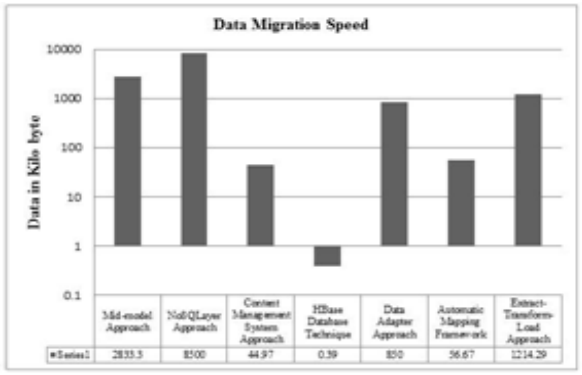

Figure 7: Data Migration Speed.

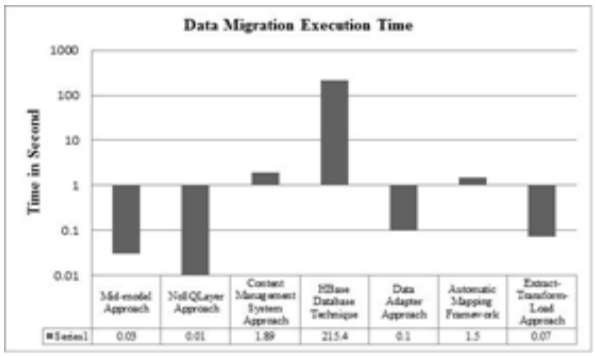

Figure 8: Data Migration Execution Time.

doi: 10.37532/jceit.2020.9(6).241

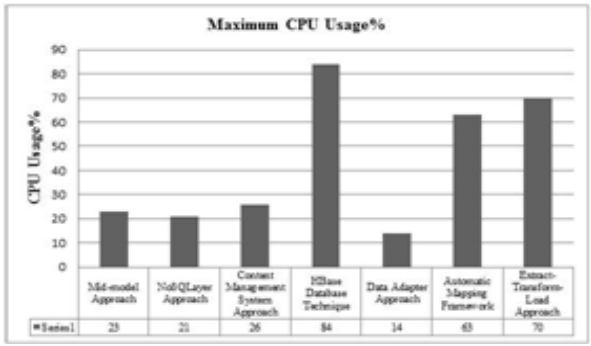

Figure 9: Maximum CPU Usage Percentage.

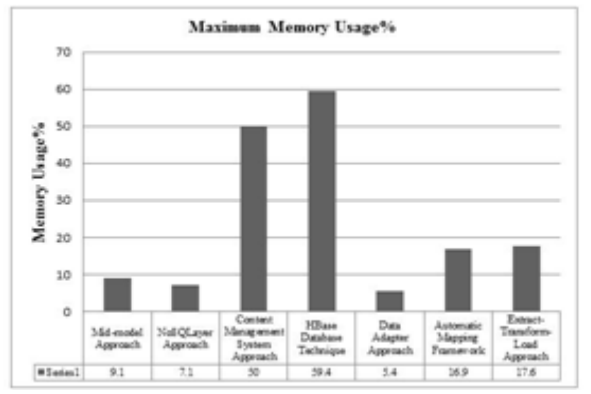

Figure 10: Maximum Memory Usage Percentage.

percentage, Maximum Memory Usage percentage transformed per second for each migration approach have also been plotted to convey the efficiency of each migrating technique.

Summarization of the Results: Although, a final result for migrating speed amongst major migration techniques has been drawn, there were other results which further verify the efficiency of the migration techniques which has helped validate our results to measure the efficiency of the transformation techniques: To depict clear picture for migrating techniques' efficiency, the results for each parameter has been presented.

In the **Figure 7**, horizontal axis shows the techniques that are used in migration and vertical axis is used to represent data in byte to be migrated in a second during the migrating process from SQL Database to NoSQL Database. From the **Figure 7**, NoSQLayer Approach is migrating largest data size i.e. 8,500 kilo byte per second from SQL database to NoSQL database. Then Mid-model Approach, Extract Transform-Load Approach and Data Adapter Approach are better from data migrating speed point of view. The migrating speed of these approaches is 2833.3 KB, 1214.29 KB and 850 KB per second respectively. Thus, we can come to the conclusion that NoSQLyer is the migrating technique which is the most efficient from the migrating speed point of view.

In the **Figure 8**, horizontal axis shows the techniques that are used in migration and vertical axis is used to represent total execution time which is consumed during the completion of data migrating process from SQL database to NoSQL database. From the Figure 8, NoSQLayer Approach has taken 0.01 Sec. to migrate 1000 number of records from SQL database to NoSQL database. Then there are other techniques such as Mid-model Approach, Extract-Transform-Load Approach and Data Adapter Approach are the techniques which consume lesser time in data migration. The execution time during the completion of data migration by them are are 0.03 Sec., 0.07 Sec. and 0.1 Sec . respectively. Thus, we can come to the conclusion that NoSQLayer is the migrating technique which is the most efficient from the execution time point of view.

In the **Figure 9**, horizontal axis shows the techniques that are used in migration and vertical axis is used to rep-resent Maximum CPU Usage percentage which is consumed during the completion of data migrating process from SQL Database to NoSQL Database. Maximum CPU Usage of Data Adapter Approach has 14 percentages which is comparatively the least among seven migration techniques. Then, NoSQLayer Approach and Mid-layer Approach have 21 percentage and 23 percentage CPU Usage respectively. They are two other techniques which have lesser CPU Usage. Thus, we can come to the conclusion that Data Adapter Approach the most efficient from the CPU Usage point of view i.e. it uses only the 14 percentage of the CPU Load during the complete migration of 1000 number of records from SQL Database to NoSQL Database.

In the **Figure 10**, horizontal axis shows the techniques that are used in migration and vertical axis is used to represent Maximum Memory Usage percentage which is consumed during the completion of data migrating process from SQL Database to NoSQL Database. Maximum CPU Usage of Data Adapter Approach has 5.4 percentages which is comparatively the least among seven migration techniques. Then, NoSQLayer Approach and Mid-layer Approach has 7.1 percentage and 9.1 percentage Memory Usage respectively. These are the two other techniques which have lesser Memory Usage. Thus, we can come to the conclusion that Data Adapter Approach is the most efficient from the Memory Usage point of view i.e. it uses only the 5.4 percentage of the Memory Load during the complete migration of 1000 number of records from SQL database to NoSQL databases.

The experiments, results, analysis and comparisons show that HBase Database Technique, Content Management System Approach, Automatic Mapping Framework and ETL Approach Technique reached a higher maximum CPU and memory loads than other techniques during the migration process. It is also seen from the viewpoint of Speed of Data migration and Execution time, the NoSQLayer Approach is the most efficient. And, from CPU Usage and Memory Usage point of view, the Data Adapter is the most efficient technique.

Conclusion

The main objective of this study is to compare various approaches of data migration from SQL to NoSQL by using well defined characteristics and datasets. In order to address the growing demands of modern applications to manage huge / big data in an efficient manner, there emerges a need of schema-less NoSQL databases that is capable of managing large amount of data in terms of storage, access and efficiency. The main focus of this research is to carry out a comparative study and analysis of most common migrating approaches using most appropriate tools (other than commercially available ones) that prefer basic and practical conversion from structured data to unstructured data. In this work, 7 (seven) migrations procedures have been performed one-by-one and separatley by using freely available resources (data and tools) and then performance analysis of each procedure has been evaluated

doi: 10.37532/jceit.2020.9(6).241

on the basis of performance parameters. Further, all the challenges faced during the course of this work have been documented for future reference. The main contribution of this work is that it will serve as guidelines for organizations looking for migrating data from structured to semi or unstructured repository in the most efficient way.

References

1. Mohamed H, Omar B, Abdesadik B (2015) "Data Migration Methodology from Relational To NoSQL Databases. Inter J Comp App, 9: 2511–2515.
2. Pretorius D (2013) "NoSQL database considerations and implications for businesses" Inter J Comp App.
3. Mughees M (2013) "NoSQL, Data migration from standard SQL to NOSQL.
4. Ghotiya S, Mandal J,Kandasamy S (2017) "Migration from relational to NoSQL database," IOP Conf Ser Mater Sci Eng 263: 1-4.
5. Yassine F,Awad M (2018) "Migrating from SQL to NOSQL Database: Practices and Analysis," Proc 13th Int Conf Innov Inf Technol 58-62.
6. Moniruzzaman B, Akhter H (2013) "NoSQL Database: New Era of Databases for Big data.
7. Potey M,Digrase M, Deshmukh G, Nerkar M (2015) "Database Mi-gration from Structured Database to non- Structured Database," Int J Comput Appl, 8975–8887.
8. Abramova P, Veronika B, Jorge F (2014) "Experimental Evaluation of Indoor Visual Comfort," Int J Database Manag Syst. 6:1-16.
9. Ameya N, Anil P, Dikshay P (2013) "Type of NOSQL databases and its comparison with relational databases" Int J Appl Inf Syst 5: 16-19.
10. Mohamed A, Altrafi G,Ismail O (2014) "Re-lational Vs. NoSQL databases: A survey," Int J Comput Inf Technol, 2279–2764.
11. Koli A, Shinde S (2017) "Approaches used in efficient migration from Relational Database to NoSQL Database," Proc Second Int Conf Res Intell Comput Eng, 10: 223–227.
12. Ramzan S, Bajwa S, Kazmi R (2018) "An intelligent approach for han-dling complexity by migrating from conventional databases to big data," Symmetry (Basel), 10:1-12.
13. Chakrabarti A, Jayapal M (2017) "Data transformation methodologies between heterogeneous data stores: A comparative study," Data 2017 – Proc 6th Int Conf Data Sci Technol Appl, 241–248.
14. Kuderu N, Kumari V (2016) "Relational Database to NoSQL Conversion by Schema Migration and Mapping," Int J Comput Eng Res Trends, 3: 506.
15. Khourdifi Y,Bahaj M, Elalami A (2018) "A new approach for migration of a relational database into column-oriented nosql database on hadoop," J Theor Appl Inf Technol, 96: 6607.
16. Tiyyagura N,Rallabandi M, Nalluri R (2016) "Data Migration from RDBMS to Hadoop,"184.
17. Seshagiri V, Vadaga M, Shah J, Karunakaran P (2016) "Data Migration Technology from SQL to Column Oriented Databases (HBase)," 5:1-11.
18. Liao T (2016) "Data adapter for querying and transformation between SQL and NoSQL database," Futur Gener Comput Syst, 65: 111–121.
19. Lalitha R (2016) "Classical Data Migration Technique in Multi-Database Systems (SQL and NOSQL)," Int J Comput Sci Inf Technol,7: 2472–2475.
20. Yangui R, Nabli A, Gargouri F (2017) "ETL based framework for NoSQL warehousing," Lect Notes Bus Inf Process, 299: 40-53.

Author Affiliation Top

Faculty of Science Health and Technology, Nepal Open University, Nepal