Secured Data Conversion, Migration, Processing and Retrieval between SQL Database and NoSQL BigData

Alaa Khalil Jumaa

Technical College of Informatics - Sulaimani Polytechnic University- Kurdistan Region of Iraq

alaa.alhadithy@spu.edu.iq

Received: 17 December 2017 **Accepted: 6 March 2018**

Abstract

Data collection is presently managed by using classic relational database systems like SQL, MySQL and Oracle. In recent years, data collection has grown up, and it has become more complex than conventional Relational Database Management System (RDBMS); which are incompetent to deal with it. To handle this problem, organizations and large companies like Google, Facebook, Yahoo and others bring up with new data management technique called NoSQL database; which is designed for a large-scale data storage and analysis. In this paper, a new technique is presented and used to convert SQL to NoSQL database, and also it can migrate, process and retrieve data between them. Because of the NoSQL database (Big Data), it sometimes needs to store in an untrusted or semi-trusted third party, the proposed system allows users to protect their database by encrypting database sensitive attributes before performing conversion and migration processes. Furthermore, the proposed system gives users the ability for retrieving NoSQL data from Big Data storage just like retrieve SQL data; that means users can write a SQL query to retrieve NoSQL data. The proposed system used Apache HBase for NoSQL BigData storage and Apache Sqoop and Hive for data conversion, migration, processing and retrieving processes. The implementation and results of the proposed system are showed the ability of converting, migrating, processing and retrieving data with high efficiency and good performance.

Keywords: NoSQL database, BigData, Data Migration, Sensitive Attributes.

DIYALA JOURNAL FOR PURE SCIENCES

Secured Data Conversion, Migration, Processing and Retrieval between SQL Database and NoSQL Big Data

Alaa Khalil Jumaa

تحويل، ترحيل، معالجة، واسترجاع البيانات الآمنه بين قواعد البيانات العلائقية (المهيكلة) والبيانات الكبيرة (الغير مهيكلة)

علاء خليل جمعة

جامعة السليمانية التقنية – الكلية التقنية المعلوماتية – اقليم كردستان العراق – العراق

الخلاصة

تجميع البيانات وادارتها غالباً ما يتم باستخدام انظمة قواعد البيانات التقليدية مثل SOL, MySQL وORACLE. وفي السنوات الاخيرة تم نمو البيانات المجمعة بشكل كبير واصبحت اكثر تعقيداً من ان تقوم انظمة البيانت التقليدية بالتعامل معها. ولحل هذة المشكلة قامت المؤسسات والشركات العالمية الكبرى مثل جوجل و فيسبووك و ياهو وغيرها باعتماد نظام جديد لأدارة البيانات الكبيرة يدعى البيانات الكبيرة الغير مهيكلة (No-SQL BigData) والتي تم تصميمها لخزن وتحليل البيانات الكبيرة. في هذا البحث تم تقديم واستخدام تقنية جديدة تقوم بتحويل قواعد البيانات المهيكلة (SQL) الى قواعد بيانات غير مهيكلة (NoSQL) وكذلك تقوم بترحيل هذة البيانات ومعالجتها واسترجاعها. وبسبب أن البيانات الكبيرة غالباً ما تخزن في خوادم غير موثوقة او شبه موثوقه، يقوم النظام المقترح بالسماح لمستخدميه بحماية بياناتهم عن طريق منحهم القدرة على تشفير الحقول الحساسة في قواعد بياناتهم قبل القيام بعملية التحويل والترحيل لها. اضافة الى ذلك يوفر النظام المقترح لمستخدمية القدرة على التعامل مع قواعد البيانات الكبيرة الغير مهيكلة بنفس الطريقة التي يتعامل بها مع البيانات المهيكلة وهذا يعني ان المستخدم يستطيع معالجة واسترجاع البيانات الكبيرة الغير مهيكلة باستخدام استعلام SQL الخاص بالبيانت المهيكلة. النظام المقترح استخدم حزمة "HBase" لخزن البيانات الكبيرة الغير مهيكلة وحزم "Sqoop" و"Hive" لتحويل وترحيل ومعالجة واسترجاع هذه البيانات. تنفيذ النظام المقترح والنتائج المستحصلة اظهرت ان النظام له القدرة على تحويل البيانات المهيكلة الى بيانات غير مهيكلة وكذلك قدرته على ترحيل ومعالجة واسترجاع البيانات الكبيرة الغير مهيكلة بكفاءة عالية واداء جيد.

الكلمات المفتاحية : قواعد البيانات الغير مهيكلة، البيانات الكبيرة، ترحيل البيانات، الحقول الحساسة.

Introduction

Data collections are growing rapidly and become more complex in volumes and variety. For this huge increase in the data size, the term of BigData is used to describe these type datasets. Big data refers to the large amounts of structured, unstructured, or hybrid data that flows continuously through and around organizations, including video, audios, texts and transactional records [1]. Big data management has become more difficult and represents a

global challenge. The database is currently managed by using classic relational database systems like SQL and ORACLE which are typically used for data storage, retrieval and management. Its offer a simplicity to users and developers as well as flexibility and high efficiency. However, lately, the traditional database management systems show the incompetence to handle effectively the requirements of new database applications which are essentially concentrated on unstructured data and data storage [2]. Google, Amazon, Facebook, and LinkedIn are among the first companies that discover the serious limitations of relational database technology for supporting big data and big user's requirements. To overcome these limitations, these companies brought up with new data management techniques, their initiatives results in producing a large interest among several developing companies that were facing the related problems. As a result, a new database is designed with novel data management model called as NoSQL. Today, the NoSQL databases are rapidly growing and deployed in many internet companies and other enterprises. It's considered as a viable choice when compared to relational databases, especially the performance and scalability requirements of big users and big data on a cloud environment can be successfully achieved by using NoSQL databases [1]. NoSQL refers to a selective and increasingly familiar data of Non-Relational database systems. Non-Relational database systems mean databases do not need to store on relational tables and not need to use SQL queries for data management and retrieval. This type of database is useful when dealing with a massive quantity of structured and unstructured data [3].

The other sections of this paper are structured as follows: the next section presents the related works. In section 3 the NoSQL database with its advantages and drawback are described. Sections 4 and 5 give a brief explanation for the Hadoop distributed file system and Apache HBase framework. In sections 6 and 7, the proposed system design, implementation, and results are described and discussed. Finally, the paper is concluded in section 8.

Related Works

F. Zhu et al. (2012) present scalable, fast, and high-performance system to implement read SQL query (no insert or update) by exploiting a number of NoSQL database features. HBase package was used as a data storage and design a partition joined table to perform join

operations that are not supported by the NoSQL. The evaluation of the system performance shows that their system obtains good results with the nineteen SQL queries. The defect for this approach that it has time cost that can be reduced using several approaches like pre-joining tables [4]. S. Khan et al. (2013) proposed a method which provides SQL Query language support to the NoSQL database MongoDB by adding an interface between the application layer and database layer. This interface contains all the routing information as the conversion rules, this will help to communicate with the database layer and to convert from one format to another. This model will achieve the scalability of the big data without affecting the logical implementation [5].

R. Lawrence (2014) proposed a new system that allows NoSQL data to be accessed using SQL queries and simply deals with any software packages supporting Java Database Connectivity package. The proposed system uses MongoDB package as a NoSQL data storage. The results show that the joins operations can be implemented efficiently but it adds a minimal overhead in translating process [6].

M. Hanine et al. (2014) introduce the new processes for migration data from SQL database to NoSQL database. This process consists of two steps: Loading the logical structure of the source database and then mapping between the RDBMS and MongoDB model. This process tries to solve the joins between tables in RDBMS. If the table to be migrated is join to another table, it has to migrate both tables to a single table in NoSQL [7].

M. Potey et al. (2015) developed a system for converting structured database, SQL to the unstructured database, NoSQL BigData. The results of this study show that the database system becomes more reliable and efficient when classical database systems are developed and complemented by using specific features and proprieties for the NoSQL database [8].

Y. Zheng et al. (2015) proposed a method that follows NoSQLs De-normalization, Duplication and Intelligent (DDI) principles keys. In this method, by using the primary and foreign key for the relational database (SQL database) the related tables are aggregated into a big table and then the most suitable key is selected, which is called row key, to identify each row in NoSQL database. The final results show that the suggested method improve access performance about forty-seven percentage [9].

A. Abdullah et al. (2015) implement a prototype system to compare the performance evaluation for data insertion and retrieval speeds between NoSQL databases (MongoDB) and SQL relational database (MySQL). The results (for one server) shows that MongoDB is faster than MySQL in the most of used scenarios, especially when the massive data are used, but this result may be changes for the data sharing across clusters [10].

M. Claudino et al. (2016) proposed a new technique which is used to improve the data conversion process between SQL and NoSQL databases. In this work, the authors present the conceptual model in the relational database system and implemented it in NoSQL database systems. The results show that, by using this model, the obtained NoSQL database is completely related to the source relational one [11].

G. Akansha et al. (2016) proposed a fast and space efficient algorithm to validate data between cross platform databases (RDBMS) and bloom filters (NoSQL) using de-normalized schema structures. The experimental results show that the proposed algorithm has the ability to validate huge datasets and pinpoint the exact corrupted records in constant space and linear time complexity up to the desired error probability. The main limitation of this method is the small probability of false positives which can be eliminated by various optimizations made to the bloom filter. [12].

K. Jeremy et al. (2016) presents the SQL relational model in terms of associative arrays and identifies the key mathematical properties that are preserved within SQL. The experimental results show that the associative arrays can provide a model for polystores to enhance the exchange of data and execution queries. [13]

L. Changqing et al. (2017) proposed efficient techniques for on SQL and NoSQL data transformation based on Espresso heuristic algorithm and depends on four related transformation steps. The final results show that this technique can be reduced the amount of memory usage and transformation execution time [14].

A. Babu et al. (2017) present a comparison of different NoSQL databases like HBase, Casandra, and MongoDB based on their structure, performance, consistency, scalability, and transactional features. Also, discuss various methodologies for migrating the data from SQL relational database to NoSQL database [2].

NoSQL Database

In 1998, the term "NoSQL" is appeared and used to refer to the relational databases that is not use SQL queries. In 2009, the term "NoSQL" is used for conferences which is held by a number of non-relational databases developer, whom organized the NoSQL meetup in San Francisco [15]. NoSQL databases are non-relational databases designed for storing and processing unstructured big data which is distributed over a large number of servers. It grows along with major Internet companies, like Yahoo, Twitter, and Google; which had problems in dealing with huge amounts of data that traditional relational database models could not support the best solution for them [3].

There is a number of challenges associated with the migration data from SQL to NoSQL databases. The first one is the quantity of data needs to be migrated and other challenges is related to database models which are used to avoid data redundancy. Major Internet companies migrate their database from SQL to NoSQL databases because the volume of stored data is massive and the relational database models are increasingly failed to satisfy the scalability, flexibility and high-performance expectations. So, the main advantages that the NoSQL databases can be offered over SQL databases are scalability, high performance, high availability and flexible data models [2].

There are four types of NoSQL databases which are created to support specific requirements, these types are categorized into four groups [1] [16]:

A. **Key-Values Databases**: It is a simple and useful way for NoSQL data storage and retrieval. Every object in the database is stored as a field name along with its value. The object's value can be any types (text, image, audio … etc.) and it can be accessed via a key.

B. **Document databases:** It is similar to the Key-Values databases, but the object value here represents a single document that consists of one or more named fields (Like XML or JSON) that are related to a specific key. This type is a flexible and it allows a dynamic data modification for add or removes fields to/from the documents.

C. **Graph databases:** It uses a flexible graph model to store the data. It stores data as nodes and relationships. These nodes are organized according to the relationship between them.

This type of storage is a good choice for working with connected data, like social networks and public transport links.

D. **Column Databases:** It stores data using column-oriented model. Data is grouped and stored in columns rather than rows, these columns consist of three elements: name, value and timestamp. This type of storage supports fast data access and aggregation for distributed data storage.

In this paper, the last type (column database) is used to store NoSQL data in the Apache HBase data storage during the migration process.

Apache Hadoop Distributed File System (HDFS)

Apache Hadoop is an open source package that is used to store and process a huge data sets across a different number of cluster machines. It can store a massive of structured and unstructured data on a large number of servers while scaling performance by simply expand the system by adding another server. Furthermore, Hadoop system can combines data from multiple sources and run queries against all of the data. Hadoop uses Hadoop File Distributed System (HDFS) to store data across clusters. Central to the scalability of Hadoop is the programming model known as MapReduce. MapReduce helps users to solve problems for data parallel by a sub-divided set of data into small parts called clusters and processed it independently. MapReduce is an important advance because it allows normal developers, not just those have high experience in computing, to use parallel programming structure without worrying about the complex details of clusters communication, job monitoring, and failure treating [17].

Hadoop improves file redundancy by dividing data file into a number of blocks and replicated it across multiple servers, this will prevent the loss of information during future node failures. Hadoop clusters consist of two nodes: master (NameNode) and salves (DataNode). In HDFS cluster there are only single NameNode and different number of DataNodes. The NameNode is responsible for storing meta-data (block location and number of blocks … etc.) and manage file system namespace in memory. However, DataNodes is used to store the actual data in HDFS and to perform read and write requests from the clients [18].

There is a number of big data Applications such as MongoDB, HBase and Casandra are used to store massive amounts of data in HDFS and they support flexible ways for data processing and retrieval. Figure (1) shows the architecture for the HDFS.

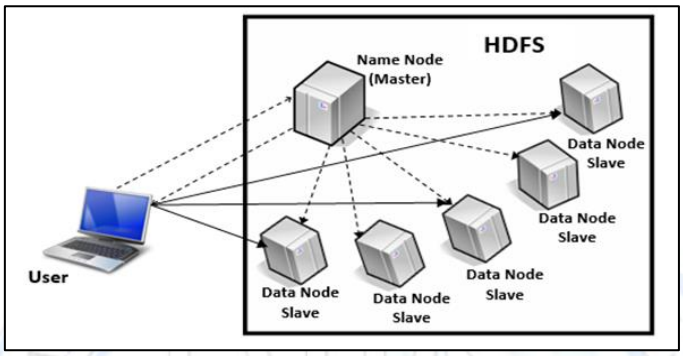

Figure 1: HDFS Architecture [18]

Apache HBase (NoSQL BigData)

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System and it is useful when the user wants to store large volumes of flexible data and query that information [19]. HBase has mainly two types of run modes: Standalone mode and Distributed mode. In standalone mode, the HBase does not use Hadoop distributed file system that means this mode not need to HDFS, but In Distributed mode, the HBase need to be installed and configured across all nodes in the HDFS cluster [9]

The HBase data model can be defined by the following concepts [20]:

- **Table:** HBase store and present data into tables. Table consists of a number of rows.
- **Row**: Data is stored in tables within its rows. Row represents a collection of column families and it can be uniquely specified by its row key.
- **Column Family**: Column family is a collection of columns qualifiers, which is represent a real arrangement of data stored in HBase. All columns families must be created up-front and are not easily modified.

- **Column Qualifier:** they are specific names assigned to data values and it represents the index for a data. Column qualifier is added to a column family and they are treated as arrays of bytes.
- **Timestamp:** It represents the version number for each value within the cell. The timestamp can be generated automatically and it must be unique.

Table 1, shows the HBase structure table for employees which is consist of two column families, Personal and Contacts. The first one has three columns qualifiers (FName, LName, and Gender) and the second has two columns qualifiers (Phone and Address). In this Table, the row key represents the ID for each employee.

Table 1: Structure for the HBase table [20]

Row Key	Column Family: Personal			Column Family: Contacts	
ID	FName	LName	Gender	Phone	Address
00001	Mustafa	Ahmad	Male	789-456-123	Baghdad
00002	Saman	Khalid	Male	779-258-147	Erbil
00003	Maryam	Musa	Female	777-321-654	Baghdad
00005	Aram	Ali	Male	780-263-487	Sulaimani
00006	Julian	Alexi	Female	753-951-852	London
00007	John	Ann	Male	365-412-987	Los Anglos

When HBase BigData storage used in Hadoop, it needs tools, such as Apache Sqoop and Hive, to connect to the relational database server for migrating and retrieving data. Apache Sqoop is used in HDFS to provide interaction with the RDBMS servers. It connects to RDBMS such as MySQL or Oracle database through Java Database Connector (JDBC) and it can import and export data between RDBMSs and HDFS. Sqoop uses MapReduce framework for data transformation. The MapReduce job creates few sessions (mappers) in DB, and each session generates SQL that query its part of the table. If the job or process needs to be quickened, the multiple mappers can be used in this case (by default Sqoop use 4 mappers) [21]. Apache Hive is a Data Warehouse framework that uses a MapReduce to provide facilitates querying and managing massive data storing in HDFS. Instead of writing huge raw map reduce programs in some programming language, Hive provides a SQL-like interface to data stored in HDFS. It has a simple structures database like a relational database, and it provides Hive Query Language (HQL) which is much like SQL [22] [23].

In this paper, the proposed system using Apache Sqoop to import or migrate data from RDBMS (MySQL, Oracle) to the NoSQL BigData (HBase) and Apache Hive for data processing and retrieving from NoSQL BigData (HBase) to user local home directory or to RDBMS. Hive used in the proposed system because it supports users' simple, reliable and dynamic NoSQL data processing and retrieval.

Proposed System

In this paper, a system for data conversion, migration, processing and retrieving between RDBMS (Oracle, MySQL) and NoSQL BigData (HBase) is designed and implemented. The proposed system uses Apache Sqoop to migrate data (tables) from RDBMS to HBase (NoSQL BigData), and Apache Hive to retrieve data from HBase to RDBMS or user local home directory. Figure (3) shows the proposed system architecture.

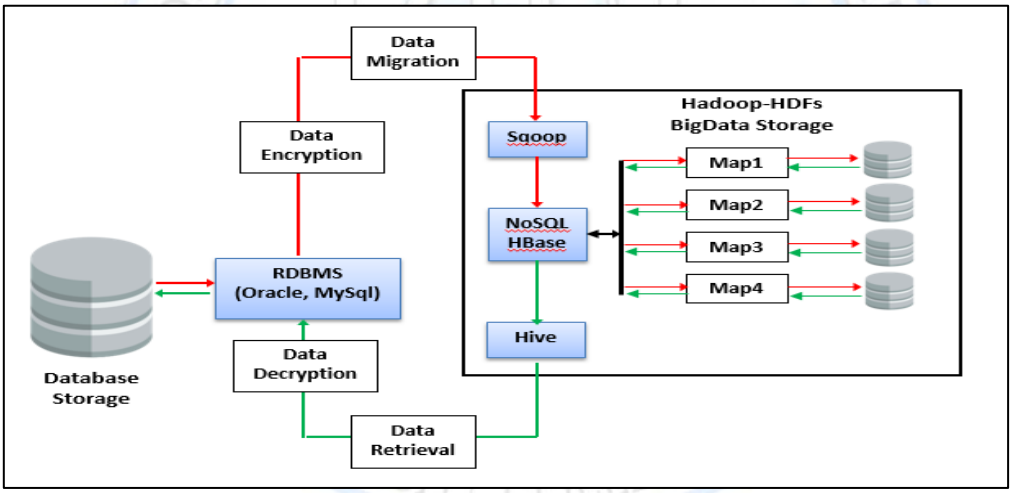

Figure 3: Proposed system architecture

In general, proposed system consists of three processes:

- **Data Encryption and Decryption Process**: In this phase, users can encrypt the sensitive attributes (Columns) for database's table. The users need to this phase before uploading or migrating databases from RDBMS to HBase BigData. This process will protect sensitive attribute from the other system users and HDFS system administrator. The proposed system supports two cryptographic algorithms (DES and AES). Users can select one of those algorithms according to their requirements. Also, after retrieving a data from

(HBase); which is big data storage; users need for decrypting those data using the same cryptographic algorithms and keys.

- **Data Conversion and Migration Process:** To perform data conversion and migration from RDBMS (SQL) to HBase big data (NoSQL), Apache Sqoop is used. Apache Sqoop imports SQL tables from RDBMS to HBase and saved in NoSQL table in the Hadoop's HDFS. Because the proposed system uses HBase with HDFS fully distributed system, the migrated database will be distributed and saved across all nodes in the cluster.

The algorithms in below shows the main steps for this process

1. Connect to HBase (NoSQL) server.
2. Create HBase tables (NST_i) and specify the row key and columns families for each table, where $NST_i \in NST_1, NST_2 \ldots NST_n$, i = 1 to n, and n = number of tables within NoSQL database.
3. Identify each SQL table's columns to the related NoSQL table's columns families.
4. Specify the number of mappers (sessions) on HDFS.
5. Connect to RDBMS server.
6. For each table (ST_i) in SQL database, where $ST_i \in ST_1, ST_2 \ldots ST_n$, i = 1 to n, and n = number of tables within SQL database.
   - Enter SQL table information (SN_i, $Col_{i,j)}$), where
     SN_i = SQL table name, i = 1 to n, and n = number of tables within SQL database.
     $Col_{i,j}$ = columns' names in table T_i, j = 1 to m, and m= number of columns for each table.
   - Enter NoSQL table information (NN_i, K_i, $ColF_{i,j}$), where
     NN_i = NoSQL table's name, i = 1 to n, and n = number of tables within NoSQL database.
     K_i = Row key for each NoSQL table.
     $ColF_{i,j}$ = Columns families' names for NoSQL table, j = 1 to m, and m= number of columns families for each NoSQL table.
   - Import and transform SQL table from RDBMS to NoSQL data and save it in HBase table.
7. Close all connections.

Figure (4) shows the flowchart of conversion and migration processes.

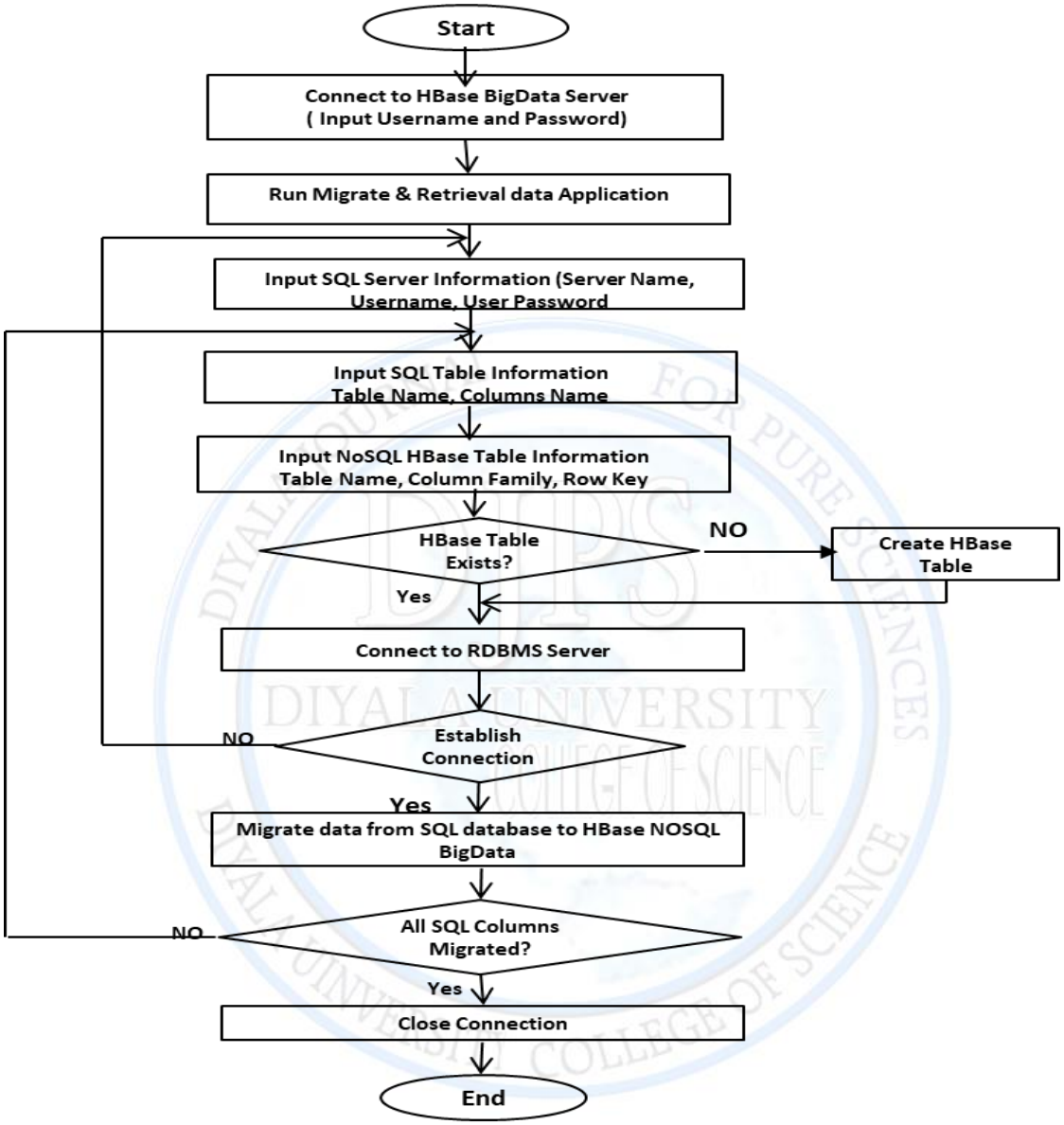

Figure 4: Flowchart of data conversion and migration process.

- **Data Processing and Retrieval Process:** Because of the complexity of processing and querying data with HBase, the proposed system has used the Apache Hive for processing and retrieving data from HBase big data to user's local home directory or to SQL RDBMS. Apache Hive gives users a reliable and flexible NoSQL data processing and retrieving. It allows the user to access NoSQL database just like SQL database, it means the user can write any SQL query to process and retrieve data from NoSQL database. So the proposed

system combines HBase structure with Hive for allowing users to access big data in a simple and a dynamic way. This process can be explained by the following:-

1. Connect to HBase (NoSQL) server.
2. Create external HBase tables (EST_i) according to the related existing HBase table (NST_i) and specify the row key and columns families using Apache Hive.
3. Create CSV file as a destination file.
4. Retrieve all data from (EST_i) table (or specific data columns from (EST_i) table) using SQL queries and saved the result in the CSV file.
5. Import data from CSV file to any RDBMS using "import" command which already exists RDBMSs.
6. Close all connections.

Figure (5) shows the flowchart for data processing and migration process.

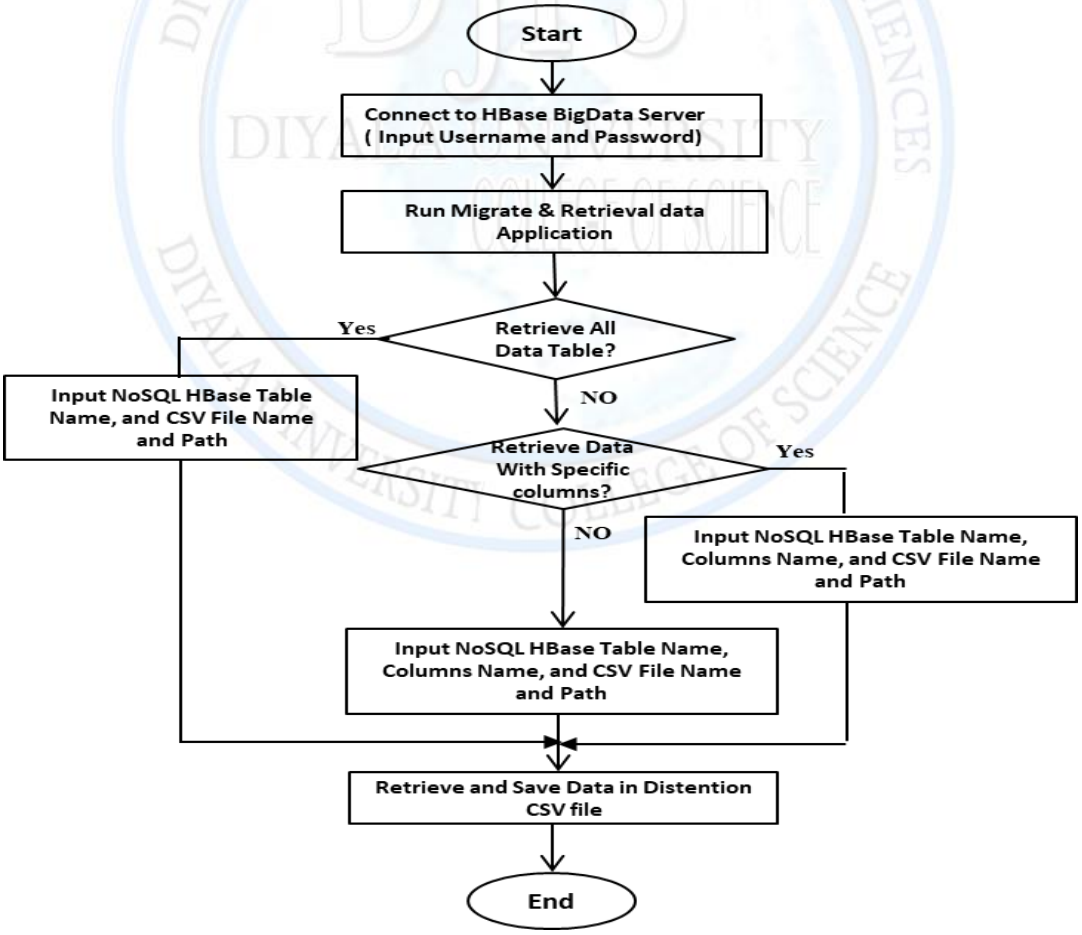

Figure 5: Flowchart for Data processing and retrieval process.

Proposed System Implementation and Results

Implementation of the proposed system needs to download, install and configure Hadoop, HBase, Hive, and Sqoop packages. Three Linux (Debian ver-8.3) servers are used. The first one (Core i7, RAM 4-GB) is used as a Master (Name Node), and the other two servers (Core i7, RAM 4-GB) are used as Slaves (Data Node).

Apache Hadoop (Ver-2.7) was downloaded, installed and configured on these servers. Also, Apache HBase (Ver-1.2.6), Hive (Ver-1.2.2), and Sqoop (Ver-1.4.6) are downloaded, installed, and configured on the same servers. Those packages are respectively downloaded from the following sites [24] [25] [26] [27]. The RDBMS part for the proposed system is Oracle 10g Express edition at the client side.

The application programs that are used to convert, migrate and retrieve data between RDBMS (SQL) and HBase BigData (NoSQL) are written in Java Programming Language.

The application programs need to download the Java Database Connectivity driver (JDBC) in order to perform the connection between java application and Apache HBase, Sqoop, and Hive.

Encryption, Migration and Retrieving processes can be implemented by following steps:

1. A user at the client side needs to connect to the Oracle database system and encrypt the sensitive attributes in the database table. Then, save it to a new secure table.
2. For the conversion and migration process, the users need to connect to the Hadoop server with their name and password. Java application program with Putty application program allows users to perform a Secure Shell (SSH) connection to the Hadoop server.

Figure (6) shows the application program interface which is used to perform these two steps.

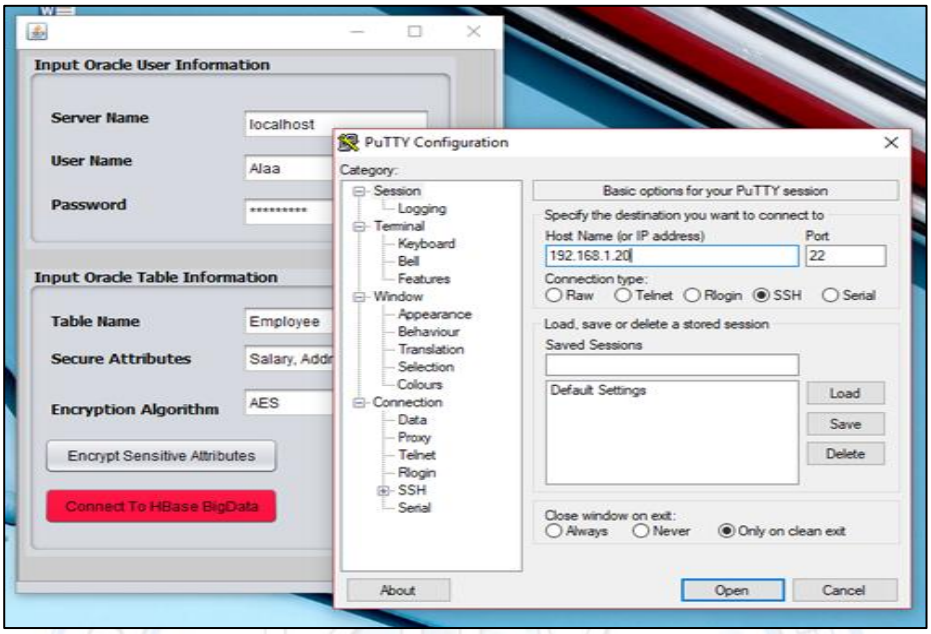

Figure 6: User Encrypt Sensitive attribute and connect to Hadoop (BigData) Server

3. After successfully connection, users can use another Java Application program that is already saved at the user local home directory for data conversion, migration, processing and retrieval between RDBMS and HBase BigData. This application program includes three options
   - Migrate Data to new HBase NoSQL Table.
   - Migrate Data to Existing HBase NoSQL Table
   - Process and Retrieve data from HBase NoSQL Table
4. For Conversion and migration process, users need to select one of the two migration options (that are mentioned in above) and they need to input required information for Oracle server (Server Name or IP, User Name, and User Password) and for HBase NoSQL database (Table Name, Columns Families, Row key, and Number of Mappers). After that, the system directly creates an External Hive table with the same name of HBase NoSQL table and with the same structure for the SQL table, this table will be used in the data processing and retrieval. Then, the user needs to click “Upload Data” command in the application program interface.

During this process, the SQL table structure will be automatically converted to the HBase NoSQL table structure. In the application program interface, the user just needs to specify columns for the SQL table and the related NoSQL column Family. For example, in tables (2) and (3), columns F-name, L-name, and Age will be specified to "Personal" Family and the Address and Phone-Number columns will be specified to the "Contact" Family.

Table 2: Columns in the SQL table

ID	F-Name	L-Name	Age	Address	Phone-Number	-----	

Table 3: Equivalent NoSQL HBase Table

Row Key: ID	Personal: F-Name	Personal: L-Name	Personal: Age	Contact: Address	Contact: Phone-Number	-----

After conversion operation, the Oracle table is converted, migrated and saved in HDFS. Figure (7) shows the application program interface for the conversion and migration processes.

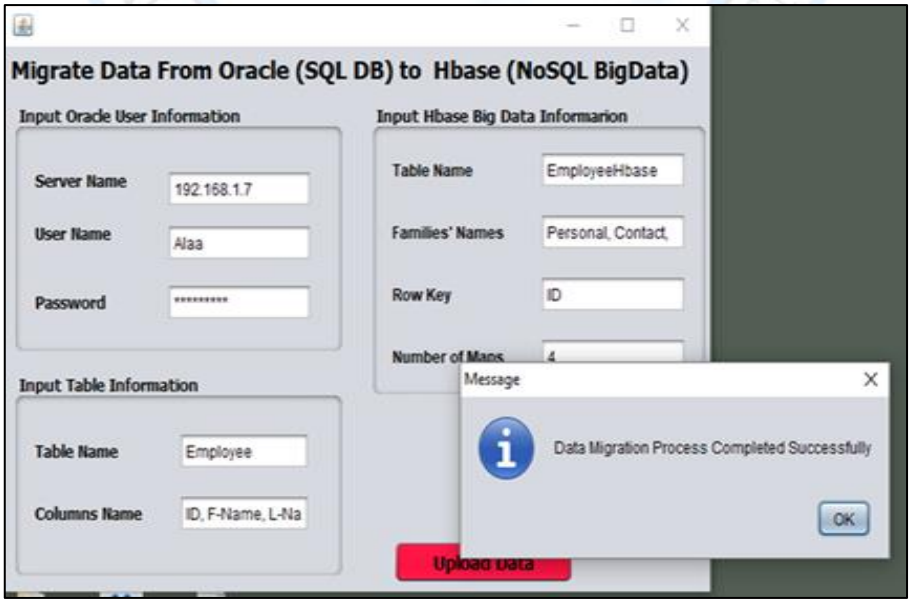

Figure 7: Convert and Migrate data from Oracle SQL to HBase NoSQL

5. There are three ways for data processing and retrieval processes. The systems' users can select one of them from the application program interface and give the required information (HBase or Hive table Name, text or CSV file name and location). Three facilities are available to the users when they want to process or retrieve data from HBase table, these are:
   - Downloading all data table.
   - Retrieving data table for the specific columns.
   - Writing a SQL query for database processing (insert, update, delete …) and optionally retrieving data with the specific condition.

   When a system's user clicks on "Download Data" command in the application program interface, the data will be retrieved and saved in the specified location. Figure (8) shows the data processing and retrieval process.

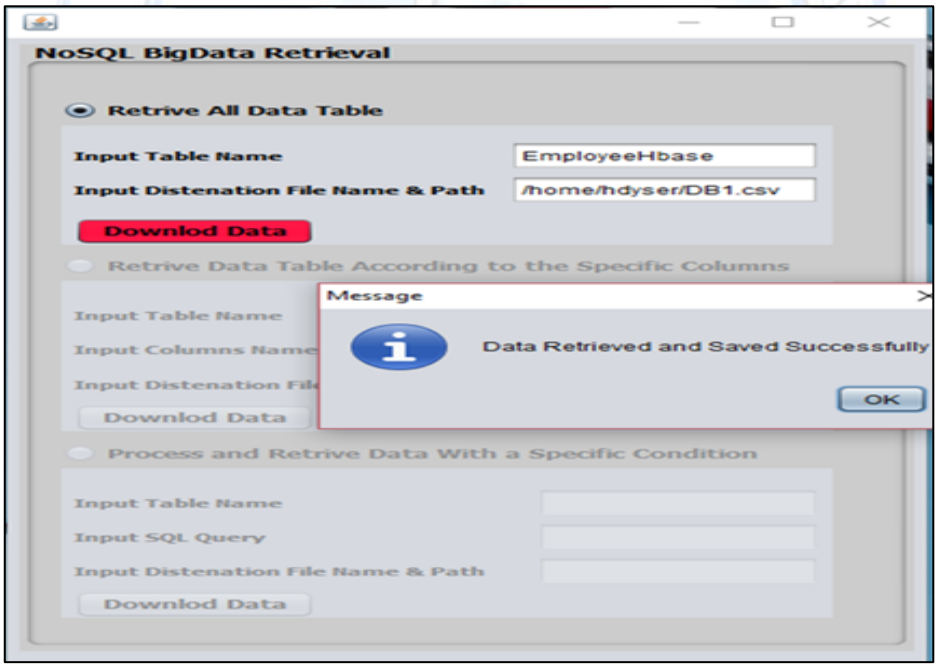

Figure 8: Data processing and retrieved from HBase NoSQL

The proposed system is used to convert and to migrate different databases (SQL Tables) with different size [28]. These database tables are converted and migrated from Oracle 10g that is installed in Windows-10 to HBase BigData (NoSQL Table) that is installed in Linux. Table

(4) shows these three different databases with their size, mappers and needed conversion and migration time.

Table 4: Tables proprieties with the conversion-migration time

Table Name	No. of Columns	No of Rows	DB Size (MB)	No. of Maps	Migration Time (sec)
BCD_HPSA_FCT_DET_DH	10	59609	6.2	1	24.32
BCD_HPSA_FCT_DET_DH	10	59609	6.2	4	25.5
EHB_AWARD_GRANT_FA_AGR1	80	24710	17	1	39.5
EHB_AWARD_GRANT_FA_AGR1	80	24710	17	4	43.7
EHB_AWARD_GRANT_FA_AGR2	27	37032	23	1	45.3
EHB_AWARD_GRANT_FA_AGR2	27	37032	23	4	60.3

Figures (9) and (10) show the relations between database size (Table Size) and data conversion and migration time for 4 and 1 mapper respectively. Based on the two figures, it can be observed that the conversion and migration time depends on the Table size and number of mappers. It can be seen that data conversion and migration needed time with 4 mappers is less than needed time for 1 mapper because of increasing the number of mappers (run with more than one sessions) lead to faster job completion. The difference in time will be increased and become very clear when the size of migrated data is increased.The proposed system in this paper differs from the other related work systems because it uses Apache HBase as a big data storage, and it exploits the facilities that are supported by the Apache Hive (HQL) to make the system more flexible and reliable when the NoSQL big data are processed and retrieved.

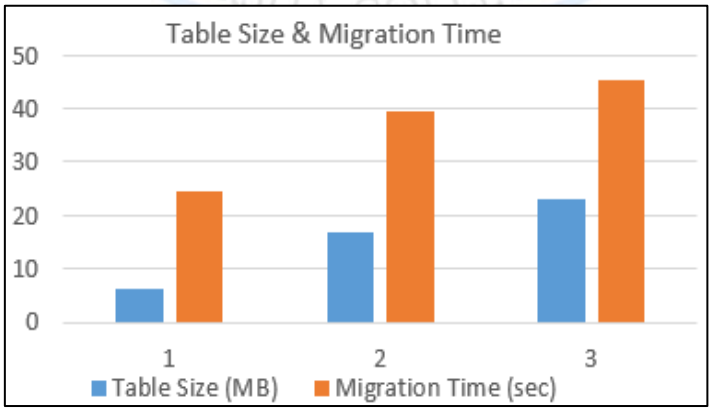

Figure 9: Relationship between Table size and Conversion-Migration needed Time for 4 Mappers

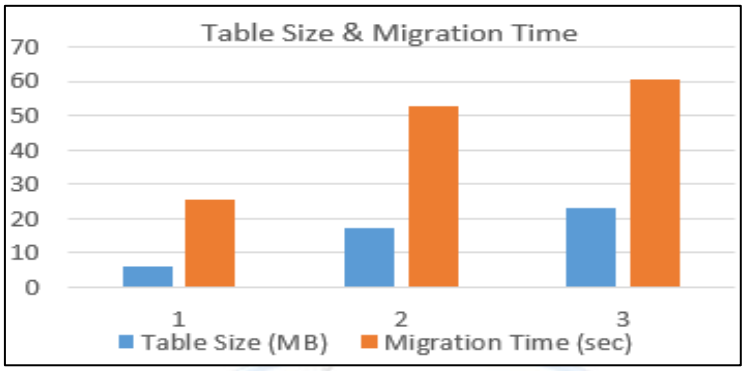

Figure 10: Relationship between Table size and Conversion-Migration needed Time for 1 Mapper

Conclusion

Recently, a NoSQL database has become a major part of the database systems because of its handful advantages. Many organization and large internet companies have been joined to use these types of the database as a new way of massive data computing by exploiting NoSQL data proprieties. The system, which is proposed in this paper, converts, migrates, process, and retrieve secure data between SQL and NoSQL data storage. The most important points that are concluded from this paper are:

1. Users can protect the sensitive attributes from the database table by encrypting data for these attributes using the cryptography algorithms (DES and AES).
2. The proposed system can convert and migrate SQL database to NoSQL data using Apache Sqoop. Then, save the data in the BigData Storage (HBase).
3. The proposed system supports NoSQL data processing and retrieving in a reliable and flexible way by using Apache Hive package as an intermediate data storage between users and HBase NoSQL BigData. Also, it allows users to write a SQL query to process and retrieve NoSQL database.
4. The proposed system can be used to migrate and retrieve BigData (Giga or Tera byte) using specific software tools like Hadoop, HBase, Hive and Sqoop; and using high-level computer performance (CPU/Memory/RAM).

5. The proposed system differs from another existing system because it uses Apache HBase as a big data storage, and exploits the Apache Hive (HQL) facilities for supporting flexibility and reliability during data retrieving and processing.
6. The implementation of the proposed system shows that the data conversion, migration, processing and retrieving between RDBMS (SQL) and HBase (NoSQL) are performed successfully with high efficiency and performance. The results also showed that the conversion and migration time depends on data size and a number of mappers.

References

1. Dadapeer, N. M.Indravasan, and G. Adarsh "A Survey on Security of NoSQL Databases". International Journal of Innovative Research in Computer and Communication Engineering. ISSN (Online): 2320-980, Vol. 4, Issue 4, April 2016.
2. A. Babu and S. Surendran. "Relational to NoSQL Database Migration", International Journal of Innovative Research in Science, Engineering and Technology. ISSN (Online): 2319 – 8753. Volume 6, Special Issue 5, March 2017.
3. A B M Moniruzzaman and S. A. Hossain. "NoSQL Database: New Era of Databases for Big data Analytics, Classification, Characteristics and Comparison". International Journal of Database Theory and Application, ISSN: 2005-4270, Vol. 6, No. 4, August, 2013.
4. F. Zhu, J. Liu, and L. Xu: "A Fast and High Throughput SQL Query System for Big Data". International Conference on Web Information Systems Engineering (WISE), pp 783-788-2012.
5. S. Khan and Prof.Vanita Mane. "SQL Support over MongoDB using Metadata", International Journal of Scientific and Research Publications, Volume 3, Issue 10, October 2013.
6. R. Lawrence. "Integration and virtualization of relational SQL and NoSQL systems including MySQL and MongoDB". International Conference in Computational Science and Computational Intelligence (CSCI), volume 1, pp 285-290, March 2014.

7. M. Hanine, A. Bendarag and O. Boutkhoum,"Data Migration Methodology from relational to NoSQL Databases", International journal of Computer, Electrical, Automation, Control and Information Engineering, Vol: 9, No: 12, 2014
8. M. Potey, M. Digrase, G. Deshmukh, and M. Nerkar. "Database Migration from Structured Database to Non-Structured Database". International Conference on Recent Trends & Advancements in Engineering Technology (ICRTAET) 2015.
9. Chao- Hsien Lee, and Yu-Lin Zheng, "Automatic SQL-to-NoSQL Schema Transformation over the MySQL and HBase Databases", IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW) 2015.
10. A. Abdullah and Q. Zhuge. "From Relational Databases to NoSQL Databases: Performance Evaluation". Research Journal of Applied Sciences, Engineering and Technology, Vol: 11 No: 4 pp: 434-439, 2015
11. W. L. Low, J. Lee, and P. Teoh., "Conceptual Mappings to Convert Relational into NoSQL Databases". 18th International Conference on Enterprise Information Systems, January 2016.
12. A. Goyal, A. Swaminathan, R. Pande and V. Attar, "Crosplatform (RDBMS to NoSQL) database validation tool usingbloom filter," 2016 International Conference on Recent Trends in Information Technology (ICRTIT), Chennai, 2016, pp. 1-5.
13. J. Kepner et al., "Associative array model of SQL, NoSQL, and NewSQL databases," 2016 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, 2016, pp. 1-9.
14. Changqing Li and Jianhua Gu, "A distinctive transformation approach of NoSQL's SQL conditions based on Espresso," 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, 2017, pp. 61-69.
15. E. Lai, "Researchers: Databases still beat Google's MapReduce". The Computerworld magazine reports in an article about the NoSQL, San Francisco-USA. April 2009.
16. P. Sareen and P. Kuma, "NoSQL Database and its Comparison with SQL Database". International Journal of Computer Science & Communication Networks, Vol: 5 No: 5, pp 293-298, 2011.

17. E. Sahafizadeh and M. Nematbakhsh. "A Survey on Security Issues in Big Data and NoSQL". Advances Computer Science International Journal (ACSIJ), Vol: 4, No: 16, July 2015.

18. V. Patil and N. Venkateshan , "Review on Big Data Security in Hadoop". International Journal of Engineering and Computer Science. Vol: 3, No: 12, pp: 9507-9509, December 2014.

19. HBase Tutorial by TUTORIAL LIBRARY, Available in https://www.tutorialspoint.com/ hbase /index.htm. Last access at 8/10/2017.

20. A. Khurana, "Introduction to HBase Schema Design". The USENIX Journal of Election Technology and Systems, Vol: 37, No: 5, pp: 29-36, October 2012.

21. Sqoop Tutorial by TUTORIAL LIBRARY, Available in https://www.tutorialspoint.com/ sqoop /sqoop_introduction.htm. Last access in 8/10/2017.

22. A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu and R. Murthy, "Hive - A Petabyte Scale Data Warehouse Using Hadoop", Proceedings of the 26th International Conference on Data Engineering, Long Beach, California, USA, March 2010,

23. N. Pushpalatha and P. Sudheer. "Data Preprocessing in BigData by using Hive Interface". International of Advance Research in Computer Science and Management Studies. Vol: 3, No:4 2015.

24. http://hadoop.apache.org/releases.html. Last access in 8/10/2017.

25. http://www-eu.apache.org/dist/hbase/. Last access in 8/10/2017.

26. http://archive.apache.org/dist/sqoop/1.99.7/. Last access in 8/10/2017.

27. https://hive.apache.org/downloads.html. Last access in 8/10/2017.

28. https:// datawarehouse.hrsa.gov / data / datadownload.aspx # MainContent_ ctl00_gvDD_ lbl_dd_topic_ttl_0. Last access at 8/10/2017.