IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ADVANCED SOFTWARE AND DATA ENGINEERING FOR SECURE SOCIETIES

Received April 20, 2019, accepted May 11, 2019, date of publication May 15, 2019, date of current version June 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916912

Intelligent Data Engineering for Migration to NoSQL Based Secure Environments

SHABANA RAMZAN[1], IMRAN SARWAR BAJWA[2], BUSHRA RAMZAN[2], AND WAHEED ANWAR[2]

[1]Department of Computer Science, Government Sadiq College Women University, Bahawalpur 63100, Pakistan
[2]Department of Computer Science & IT, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan

Corresponding author: Imran Sarwar Bajwa (imran.sarwar@iub.edu.pk)

ABSTRACT In an era of super computing, data is increasing exponentially requiring more proficiency from the available technologies of data storage, data processing, and analysis. Such continuous massive growth of structured and unstructured data is referred to as a ''Big data''. The processing and storage of big data through a conventional technique is not possible. Due to improved proficiency of Big Data solution in handling data, such as NoSQL caused the developers in the previous decade to start preferring big data databases, such as Apache Cassandra, Oracle, and NoSQL. NoSQL is a modern database technology that is designed to provide scalability to support voluminous data, leading to the rise of NoSQL as the most viable database solution. These modern databases aim to overcome the limitations of relational databases such as unlimited scalability, high performance, data modeling, data distribution, and continuous availability. These days, the larger enterprises need to shift NoSQL databases due to their more flexible models. It is a great challenge for business organizations and enterprises to transform their existing databases to NoSQL databases considering heterogeneity and complexity in relational data. In addition, with the emergence of big data, data cleansing has become a great challenge. In this paper, we proposed an approach that has two modules: data transformation and data cleansing module. The first phase is the transformation of a relational database to Oracle NoSQL database through model transformation. The second phase provides data cleansing ability to improve data quality and prepare it for big data analytics. The experiments show the proposed approach successfully transforms the relational database to a big data database and improve data quality.

INDEX TERMS Relational databases, NoSQL, big data, data cleansing.

I. INTRODUCTION

Since the past 3 to 4 decades larger organizations and enterprises have been using conventional databases, such as RDBs (Relational Databases) to store and analyze their data. RDBs have structural model and supports SQL (Structured Query Language) [1]. The RDBs were used comfortably before the big data, because usually the input data was in structured form. If there came across some unstructured data, different mechanisms such as ETL tools were used to convert it into structured data. So, it was not a challenge to handle unstructured data. Now, it has become near to impossible for RDBs to meet pace with the volume, velocity and variety of data creation (3Vs of big data) and storage. However, the social networks are visited by thousands of clients per unit time that become hard for RDBs to handle. These networks ensure all times, availability of services to the clients. It seems that the social network's clients have a variety of data. Alongside there is another paradigm called cloud computing, which also demands data stores for handling massive amounts of data. The data in cloud applications grow beyond the 1TB, Such a huge amount of data creates an issue in terms of response time [2]–[5] Software engineering is facing new challenges with cloud computing because it accumulates the large amount of data on daily basis. These challenges include capturing, querying, sharing, storage, analysis and visualization of large amount of data. Facebook, Yahoo, Google, and Amazon were the websites that realized first, RDBs are unable to handle the types of data and volumes as their data is in petabyte range. RDB is storing that huge amount of data at high cost; cost grows geometrically with the volume of the data. Until a few years ago, RDBs were massively

The associate editor coordinating the review of this manuscript and approving it for publication was Mahmoud Barhamgi.

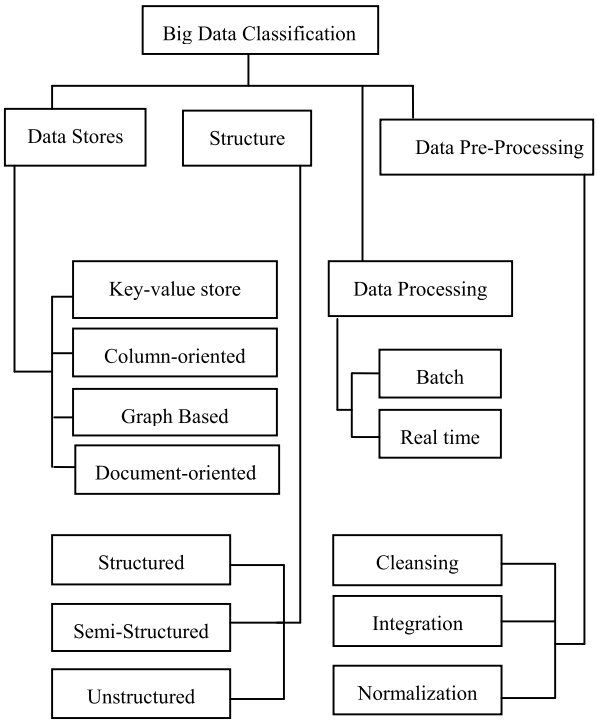


FIGURE 1. Big data classification.

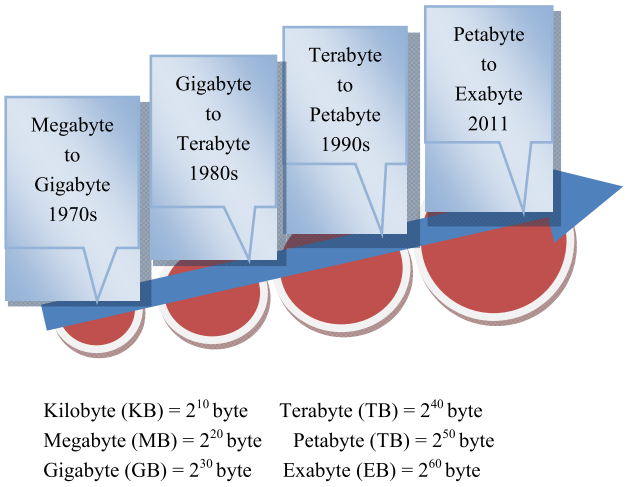


FIGURE 2. The rise in big data.

used by the enterprises to store their huge amount of data. However, the big data in cloud computing is a challenge for RDBMS [6]. It is not possible for RDBs to effectively access petabytes of data. The Big data has major differences with RDBs as it has structured, semi-structured and unstructured data, but RDBs can store only structured data and provides very little support to unstructured and semi-structured data. For better understanding of the big data characteristics, the big data is classified into various categories as shown in Fig. 1. The big data demand horizontal scalability for massive data, but RDBs cannot provide horizontal scalability. A continuous and rapid increase in data is shown in Fig. 2, the term ''big data'' firstly introduced in 1970 and 1980s, when data size increased to gigabytes from megabytes, in the late 1980s the data expanded to several gigabytes or even terabytes, in 1990s companies were holding terabytes or petabytes of data and currently big companies data reach the petabyte or even Exabyte.

This scenario leads developers to investigate the other storage alternatives, big data databases (NoSQL databases) emerged as a better choice, so developers became a part of this hype. NoSQL databases are gaining tremendous popularity as an alternative model of data storage. They were created to overcome the limitations of conventional databases (RDBs). The NoSQL databases have several advantages, including the ability to handle all types of data, whether it is structured, unstructured or semi-structured, highly flexible, distributable and scalable. They can significantly reduce the cost of commodity hardware because it can easily scale out (horizontal scalability) on commodity clusters. The change management is one of the major challenges for RDBMS; if change is not addressed carefully, then the service level will be degraded. NoSQL databases allow the creation of new columns even in rigidly defined BigTable without any fuss. NoSQL databases are cost effective because they allow the storage of massive amount of data at a much lower cost. NoSQL databases have simpler data models, data distribution and automatic repair that leads to lesser maintenance and tuning requirements, so they require less management in contrast with RDBs.

Considering the challenges of big data for RDBs, the modern organizations (having big data systems) are rapidly switching from existing RDBs to NoSQL databases. The successful handling of big data complexity is a great challenge for the conversion of RDBs to NoSQL databases [7]. RDB to RDB conversion like MSSQL to SQL Server is easy because they have a mathematical foundation [8]. NoSQL databases are designed according to the application specific access patterns and queries without using a normalization process and general relation., RDB trained workforce felt it difficult to convert RDB to NoSQL database [9]. Data analysis is another challenge for big data organizations. RDBs can store structured data only and, the data is pre-processed being stored. So, data can easily be retrieved for analysis. However, in case of unstructured data, data is stored first and later on processed for analysis. There is a need to improve data quality in big data systems through data processing techniques. Our research aims to present an automated approach for conversion of existing RDB to NoSQL database and apply data cleansing technique to improve data quality. Data cleansing is necessary to achieve data analysis accuracy. The proposed approach utilized model to model transformation technique to transform SQL Server database into an Oracle NoSQL database. For this transformation we need the metamodels of both, source and target databases. The RDBs metamodel is available, but metamodel of NoSQL databases is not available, so we propose an initial version of the Oracle NoSQL metamodel.

Further, the rest of the paper is structured as follows: review of the related work is presented in Section 2; Section 3 described the basic concepts of RDBs, big data databases and data cleansing; Section 4 presents the transformation module; in Section 5, the data cleansing module is presented, which is followed by result and evaluation in Section 6; finally the conclusion and future directions are given in Section 7.

II. RELATED WORK

NoSQL is a new breed of databases that do not based on the relational model and do not use SQL for data manipulation [10]. First, we gave an overview of the NoSQL storage of the available literature and then we reviewed the literature related to our research goal of transforming the RDB to NoSQL database and data cleansing of transformed data.

NoSQL databases neither provide the ACID properties, nor base properties, they actually lie in between the spectrum of ACID and BASE [11]. The key features of NoSQL databases are replication, distributed indexes and RAM usage for data storage, horizontal scalability, partition and flexible schema. Therefore, it is possible for these databases to add new columns dynamically in data records. The NoSQL proponents often cite CAP theorem, according to this theorem the system can provide only two properties out of the three properties such as consistency, partition-tolerance and availability [12]. Typically, these databases compromise on consistency. NoSQL data stores are growing rapidly, and it is reported that more than 200 NoSQL stores are available so far.[1] NoSQL databases can handle the big data more efficiently and faster processing as compared to RDBs [13]. RDB Management Systems (RDBMS) are unable to manage the huge growth of data on a single server (vertical scalability).

This is a big challenge for enterprises to transform the existing databases to large scale ones (NoSQL) successfully [14]. For this transformation, it is necessary for the designers and architects have in-depth understanding of RDBs as well as NoSQL databases. Automatic migration from RDB to MongoDB, exploiting the data and query characteristics of RDB [15]. The algorithm is developed using tags: description and action and also a tool is developed for migration from RDB to NoSQL. The automated transformation from SQL Server to HBase is proposed to avoid cross-table queries [16]. The authors have transformed all the tables of RDB involved in a relationship with single HBase table. This approach reduces the complexity of the transformation process by fitting a large and complex relation database into a single table of HBase. There is another transformation approach that has four steps [17]. In the first step, transformed relationships (one-to-one and one-to- many) into a single HBase table. In a second step merging of neighbouring tables is made through recursive method. In the third step, row key is designed and in the fourth step, views are created that based on secondary indexes for required access patterns. One additional step which uses query logs to extract the access patterns. The last step is a key step of this transformation. The authors compared and analyzed the data structures of the input database (RDB) and output database (NoSQL database) to suggest a Graphical User Interface (GUI) tool for transformation [18]. The proposed tool transformed the SQL Server database into Couch database. The framework ''NoSQLayer'' was proposed that perform migration between SQL Server and MonogoDB [19]. This framework has two modules, one is used for the extraction and mapping of metadata and other is used for migration process. The three guidelines are provided to transform the RDB schema to Hbase schema [20] and schema mappings are generated on the bases of these given guidelines. The set of nested schema mappings shows the relationship between the two schemas. These mappings are used to create a program for the transformation of data from the RDB to target NoSQL database. The schema transformation process was proposed for transforming RDBMS to MongoDB [21]. These transformation processes use join operations to support queries in an enhanced manner. This process based on a graph transformation, the schema definition of both databases is read and store as a graph. Sqoop is a tool that works in two ways, migrate the data from the RDB to Hadoop and vice versa [22]. This Apache tool transferred the bulk of data by reading the metadata of RDB and transformed it into Hadoop format using MapReduce and regenerated back to RDB. Most of the web-based applications and Content Management System (CMS) solutions are used RDBs for data management, but the number of users of the internet and clouds are growing rapidly, so it has become difficult for RDBs to handle the huge data traffic. The designed approach maps the real CMS SQL database to a NoSQL database [19]. This approach has two steps, first of all it denormalize the SQL database and then choose a unique identifier i.e. primary key for a big table. In this approach, SQL Server database is migrated to a column-oriented Hbase database. For transformation from RDBs to NoSQL, another application is developed which deals with the transformation of RDB schema to NoSQL schema. This application handles both the DDL and DML commands of relational schema and transform these commands into corresponding commands of NoSQL [23]. The authors proposed approach to transform RDB to document oriented database, using a browser to connect to database [24]. After connectivity, it extracted schema and metadata and extracted data in the form of a graph. Another method is designed to transform data from the RDB (MySQL) to NoSQL (MongoDB) [25]. Migration from relational to NoSQL is comprised of few steps, initially; MySQL database connection is created, afterwards the details of the database are accessed through prototype software. Further mapping is performed between RDB (MySQL) to NoSQL (MongoDB). The layered architecture includes mapping and initial experiment to evaluate the layer. NoSQL layer is proposed for migration of MySQL database to MongoDB [26]. It has two modules, one for migration and the other

[1] 6https://nosql-database.org/.

TABLE 1. Comparison of transformation approaches.

Concepts	Rocha et al. [27]	Hanine et al. [26]	Karnitis et al.[25]	Yoo et al. [28]	Li et al. [19]	Zhao et al.[20]	Jia et al [15]	Proposed Study
Tables	✔	✔	✔	✔	✔	✔	✔	✔
Fields	✔	✔	✔	✔	✔	✔	✔	✔
Data type	✔	✔	✔	✘	✘	✘	✘	✔
Indexes	✔	✘	✘	✘	✘	✘	✔	✔
Join	✔	✔	✘	✔	✘	✔	✔	✔
Primary-Foreign key Relationship	✔	✔	✔	✔	✔	✔	✔	✔
Database Migration Validation	✘	✘	✘	✘	✘	✔	✘	✔

for mapping. The migration module has used various methods for successful migration, whereas mapping module has a persistent layer to translate and run database request in any DBMS to return data in an appropriate format. Experimentally it has proven that proposed layer is an effective solution to handle massive data applications. Similarly, another solution is available to migrate the RDBMS schema to document oriented NoSQL database schema [27]. This method provides atomicity using atomic aggregation and avoids join operations. It uses the column-level denormalization in order to reduce the disadvantages of table-level denormalization. Table 1 shows the comparison between our proposed approach and other transformation approaches.

Data cleansing generally ensures the data consistency and updation [28]. A unified framework of sequential data cleaning approach is proposed for data warehouse [29]. This framework is highly efficient to handle dirty data. It also allows user interaction and provides accurate and consistent data for data analysis. Various rule-based inference algorithms were proposed for RFID (Radio Frequency Identification) data cleansing [30]–[33]. They are applied directly to an RFID data stream and the cleansing technique is used to clean the raw RFID domain data that have many anomalies and is of low quality due to environmental noise and limitations of the devices. The proposed approach used Bayesian inference to clean RFID raw data with high accuracy and efficiency. The detecting crawlers were used for e-commerce data cleansing and de-duping of clients and customers on regular bases [34]. RFID data stream cleansing in a mobile environment is a challenging task. The missing data issue in a mobile environment is addressed by the probabilistic model [35]. The inference based approach is used with model and sequential sampler to sample data that cleans RFID data efficiently. The authors proposed a framework called, BIO-AJAX to clean, correct and standardize biological data in order to improve quality [36]. The framework has two implementations, one for Lineage Path i.e. BIO-AJAX and the other is for Tree-BASE i.e. BIO-JAX.

To the best of our knowledge, the model transformation approach is not used for transformation of RDB to big data databases (such as Oracle NoSQL database). The major contribution of this paper is to present a novel approach that automatically transforms SQL Server database to Oracle NoSQL for both data and schema, afterwards it performs cleansing of transformed data.

III. BACKGROUND: RDBS AND BIG DATA DATABASES

In this section we will give a brief description of the basic concepts about RDBs, big data databases and data cleansing that is relevant to our work.

A. RELATIONAL DATABASES

RDBs are the databases that based on the relational model introduced by E.F. Codd in 1969. The relational model has mathematical foundation: theory of relations and first-order predicate. All mathematical relations have special properties, relation is a mathematical term used for a table. Data is stored in tables using relationships and then accessed through these relationships. These relationships can be of one-to-one, one-to-many and many-to-many type and are called ''entity relationship'' [37]. Each table can have many rows, but all of the same type and can have different content. RDBMS (Relational Database Management System) is used for the creation and management of RDBs. RDBMS defined a language for data definition, data integrity, data manipulation and for transaction control of database [38]. The basic building block of relational model has rows (tuples), columns (attributes) and tables (relations), the order of row and column is insignificant, and each row is unique. There are two main types of RDBMS keys: primary key and foreign key. Primary key can have a single attribute or multiple attribute and uniquely identifies the record. Foreign key can comprise single attribute or multiple attribute that refers to the primary key of another table. The tables or entities are connected to each other through primary key and foreign key relationships. Normalization (set of rules) decomposes the tables into sub-tables to cope up with the problems of complex domains. Another important concept of RDBs is an ACID (Atomicity, Consistency, Isolation and Durability) properties that ensure the reliable transaction process [39]. The data query language of RDBs is Structured Query Language (SQL) to get access of databases. Initially this language is known as Structured English Query Language (SEQL) later on SEQL was replaced with SQL [40].

B. BIG DATA DATABASES

Big data databases (NoSQL databases) are designed with the needs of 'big data' in mind. The evolving landscape of NoSQL databases and its database management systems (NoSQL DBMS) has everything to do with Big Data. The term NoSQL was introduced in 1998 by Carl Strozzi [41] for his open source RDB that was not using SQL to access and manipulate data. NoSQL database has no structured query language interface [42] and have different data models (Document store, column store, key-value store and Graph store). Initially the term NoSQL stood for "No SQL" but recently redefined as "Not Only SQL" means that it complements the RDB management system [43], [44].

Almost all RDBs can be queried with SQL on the other hand each NoSQL database has its own query mechanism. These databases have no standard language to be compared with SQL. NoSQL databases have little to no interoperability due to lack of standardization. Every data type of NoSQL databases uses different data structures. There is no concept of primary or foreign key and no need of schemas or join operations to access data. In this approach we transform SQL server database to big data database such as Oracle NoSQL.

Oracle NoSQL is a distributed key-value store and provides significant features like horizontal scalability, monitoring, transactional semantics for improved data manipulation, and simple administration of data. The Oracle NoSQL has a very simple data model where each row is a key-value pair whose value is associated with a unique key and the value is of arbitrary length. It has tables, rows and fields which are equivalent to tables, rows and columns of RDBs but with a different concept. The Oracle NoSQL table is schema free, but RDBs' tables have predefined schema, each column of Oracle NoSQL has a separate schema, but in RDBs each table has a schema and each row in the Oracle NoSQL database can have unrelated fields but in RDBs each row is a collection of related items.

C. DATA CLEANSING

Data cleansing is a relatively new research field of data analysis, which is an essential part of larger research called data quality. Databases are facing the data cleansing issues throughout their history. The data cleansing is used to improve the quality of data by removing inaccurate, incomplete, or unreasonable data and generally comprises of following steps [45]:

- Define and determine error types;
- Search and identify error instances;
- Correct the errors;
- Document error instances and error types;
- Modify data entry procedures to reduce future errors

The digital information can have serious issues of data quality. For any form of data analysis, data cleansing is the most critical prerequisite [46]. It is computationally expensive process, so suitable for current technology, but near to impossible with previous technology. One can break down the cleansing into six steps: elementizing, standardizing, matching, verifying, house holding, and documenting [44]. Researchers are trying to handle different challenges that arise in the field of data cleansing. There are many issues in the data that need to address, such as data duplication, erroneous data, determining record usability, etc. The term ''dirty data'' is used in literature [47]–[49], here search context is, what it is? Data cleansing has no commonly agreed definition, it depends upon the area that applied the cleansing process. Various definitions of data cleansing are available in the literature. The data cleansing problem is defined as merge/purge problem [48], it is the process to remove data inconsistencies, eliminating the errors and solution for the problem of object identity [50]. The researcher can then guide the operational staff to enhance the validity of study, precision and accuracy of outcomes. Sometimes it is unavoidable to make amendments in the study protocol on a regular basis, regarding design, quality control procedures, observer training and timing. In the worst case, it may require restarting the study again, data extraction and data transformation programming may need to revise.

IV. METHODOLOGY FOR PROPOSED APPROACH

We proposed an approach that has two modules: transformation module and data cleansing module. The first one is to transform RDB to Oracle NoSQL database through model transformation. The latter provides data cleansing methodology to improve data quality.

A. TRANSFORMATION MODULE

Model transformation is the core of the model driven engineering paradigm. It is an automated way to create and exploit conceptual models of the domain. For this purpose, the relational metamodel is available, but there is no metamodel for Oracle NoSQL. In this context, we have proposed primary version of the oracle NoSQL metamodel.

B. MODEL TRANSFORMATION

Model driven Architecture is an approach that deals with models to develop software. The required elements for model transformation are input model, transformation description, transformation engine and transformation rules. The complete framework of model transformation is shown in Fig. 3.

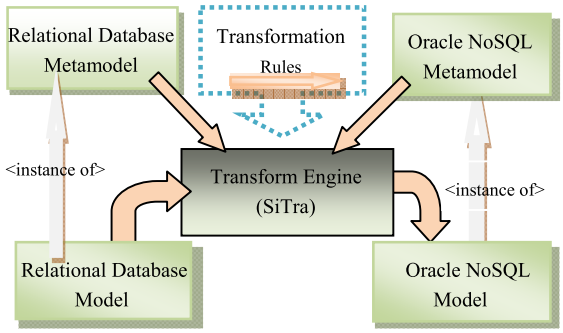


FIGURE 3. Model transformation from relational to oracle NoSQL.

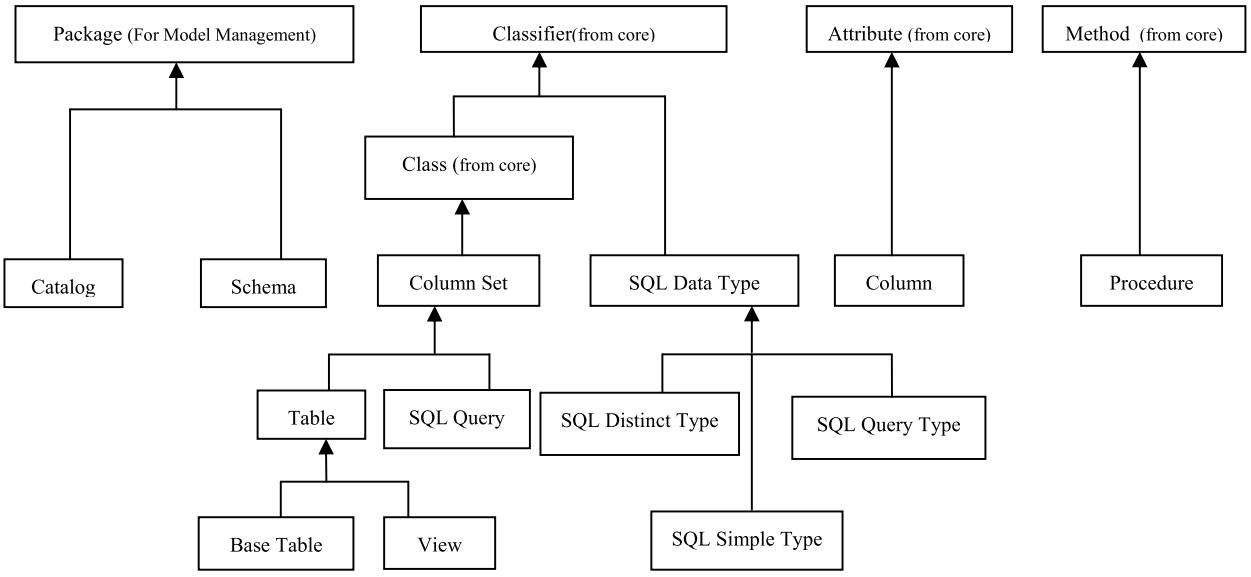


FIGURE 4. Relational metamodel.

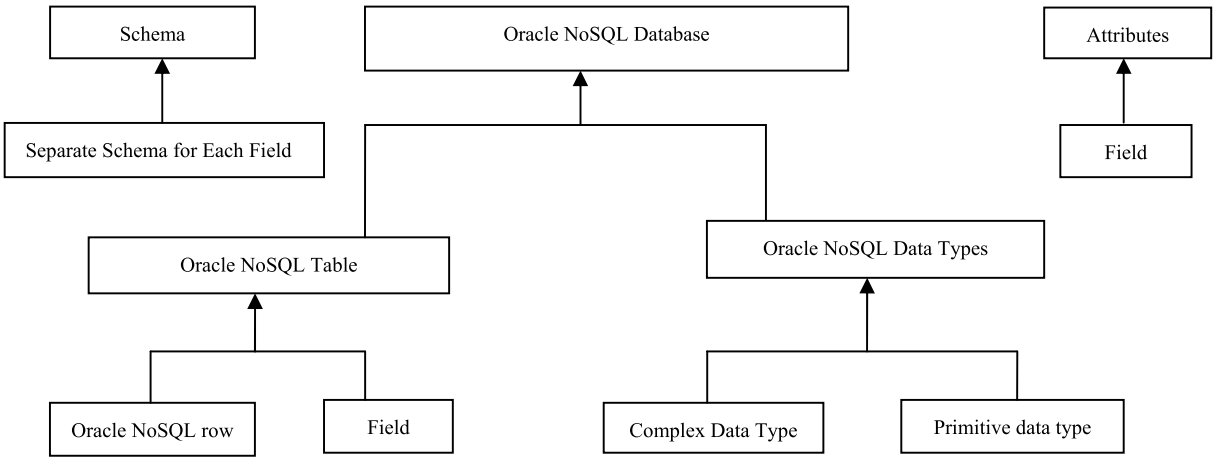


FIGURE 5. Oracle NoSQL metamodel.

In our proposed approach we are using SiTra as a transformation engine to transform the input model to output model and maps the concepts of input model (SQL Server, RDBs model) to output model (Oracle NoSQL model) for the generation of transformation rules that are used by SiTra for transformation. MDA (Model Driven Architecture) is a model based approach that has various types of model transformation, e.g. model to model, model to text and text to model. Model to model transformation is our main concern that transforms a model into another model. The MDA approach maps concepts of source metamodel into corresponding concepts of target metamodel. This mapping is used to generate transformation rules that are used by the transformation engine to generate target metamodel from the source metamodel automatically. The relational metamodel is shown in Fig. 4 and Oracle NoSQL metamodel is shown in Fig. 5.

We proposed Java based MDD (Model-Driven Development) approach to implement model transformations. The two interfaces are used to implement the transformation rules i.e. rule interface and transformer interface. The rule interface is implemented for each transformation rule and it has two methods, check method and build method. The transformation algorithm (Algorithm 1) is used to implement a transformer interface. A transformation algorithm uses check method to determine whether rule is applicable or not and build method is used to create target objects. We extended the transformation algorithm to keep record of transformed objects and avoid duplicate creation of target objects.

Algorithm 1 Transformation Algorithm

1.Construct: create table object of the source.
2.Construct: Generating a list of target table objects from source table's objects.
3.Mapping: source table objects map to target table objects.
4.Construct: create object of each column of source table.
5.Mapping: Source column object map to target field object.
6. **If** value is true
7. Select the transformation rule
8. // rule is applicable or not, when more than one rule is available for same type object.
9. **endif**
10.Construct: the target table object is created.
11.Construct: target object of each column is created.
12.Select primary key.
13.Keep a history of transformed objects // to avoid duplicate creation of target objects.

C. CONCEPTS MAPPING CHART

The mapping chart shows how to map the concepts of SQL Server database to Oracle NoSQL database, as shown in Fig. 6. Each concept of SQL Server database is mapped to corresponding concepts of Oracle NoSQL. Oracle NoSQL have no join operation, it uses the parent child relationship instead of join operations. Relational databases have the concept of permissions whereas Oracle NoSQL has the concept of privileges.

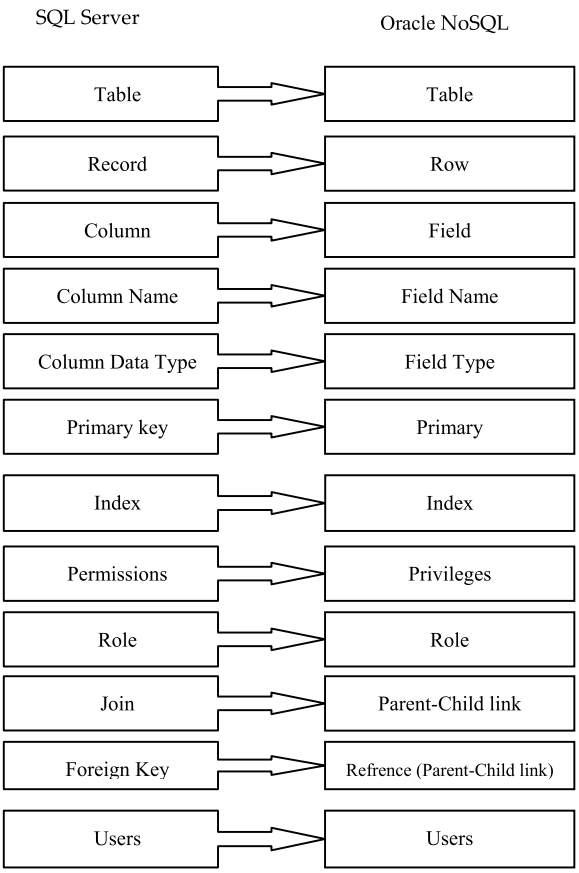


FIGURE 6. Concepts mapping chart.

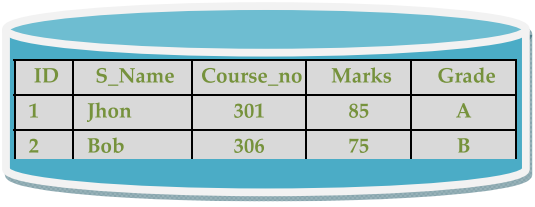
```
{"Id": 1,"Sname": "operating system", "Class": "Ist year"}
{"Id": 2,"Sname": "English literature"
```

FIGURE 7. Oracle NoSQL rows.

D. MAPPING OF TABLES, ROWS AND COLUMNS

The Oracle NoSQL database has tables which are equivalent to tables in SQL Server database. Each Oracle NoSQL table stores data in the form of Oracle NoSQL rows and fields which are equivalent to rows and columns of SQL Server database. Row is a collection of fields and each row is a pair of key and value where key is unique and has value of arbitrary length. Fig. 7 shows the example of Oracle NoSQL rows using fields to store data.

Each Oracle NoSQL table consists of many rows that are equivalent to rows in SQL Server database. Each row of oracle NoSQL is retrieved through a unique key of that particular row. For better understanding of the mapping process, we take an example of SQL Server table and its corresponding table of oracle NoSQL as shown in Figure 8. Each row of SQL Server database table transforms to Oracle NoSQL row and each column of SQL Server table to fields of Oracle NoSQL, as shown in Fig. 8.



ID	S_Name	Course_no	Marks	Grade
1	Jhon	301	85	A
2	Bob	306	75	B

```
{"Id": 1, "S-name": "Jhon", "Course no": "301", "Marks":
"85", "Grade": "A"

{"Id": 2, "S-name": "Bob", "Course no": "306", "Marks": "75",
"Grade": "B"}
```

FIGURE 8. Mapping of SQL server database to oracle NoSQL database.

E. ORACLE NOSQL SCHEMA

Avro is a data schema for the value of records and can be applied to the Oracle NoSQL database. It defines the value fields and their data types. We created avro schema by using JavaScript Object Notation (JSON) format as shown in Fig. 9.

```
KV - > show table-name Student
{
"Type": "Table",
"Name":  "Student",
"Owner": "null",
"shardkey": ["id"],
"primarykey": ["id"],
"fields": [{
      "name": "S-Name", "type": "STRING",
      "default": ""
       },{
       "name": "Coursetitle", "type": "STRING"
       "default": ""
       },{
      "name": "Marks",
      "type": "INTEGER",
      "default": ""   }, ]
}
```

FIGURE 9. Avro schema of student table.

F. TRANSFORMATION RULES

We present an approach for automatic transformation of MySQL server database to oracle NoSQL. For this transformation, siTra engine is used which executes the transformation rules. Each transformation rule has two parts, right side for target model and left side for the source model.

V. DATA CLEANSING MODULE

The methods used for maintaining data quality are data cleansing [51], data quality monitoring and data integration [52]. The data cleansing detects and removes data inconsistencies and errors. The data cleansing approach for transformed database (Oracle NoSQL) is described in the algorithm 2 based on [53] and data cleansing framework is shown in Fig. 10. Data cleansing become necessary with the transformation of legacy database to a new database system [53], such as SQL server database to NoSQL database.

Algorithm 2 Data Cleansing Algorithm

1. Select the database. // dirty data
2. Extract the data.
3. error detection
4. **While** true // apply rule for error detection
5. // clustering algorithm to identify irrelevant data and scrubbing to identify data duplication.
6. error correction // special algorithm for correction.
7. confirm correction
8. store in temporary file
9. **End While**
10. Show to user for manual correction
11. Data Enhancement.
12. Data Harmonization.
13. Data Standardization.
14. Cross check all corrections in temporary file.
15. // Confirm data cleansing by cross checking of data with validated data set
16. Load to the database.

A. ERROR DETECTION

The data that is incorrect, incomplete and inaccurate is detected using different techniques. We have used a parser to detect syntax errors, the technique of clustering to identify irrelevant records and file checksum method to identify duplication of data.

1) PARSING

We used a parser to detect syntax errors. The working of this parser is similar to the parser that used for language and grammar parsing. The parser will decide the acceptance or rejection of data string. It accepts data if it is according to specification (such as JSON format).

2) CLUSTERING

We used clustering that based on Euclidian distance to automatically identify the errors. The clustering algorithm provides a support to identify irrelevant data. The combined clustering algorithm is used that considers the Euclidian distance between records. The algorithm tries to adjust the size of the clusters by making an automatic adjustment of cluster size quickly. The identified clusters combine the records that have certain similarities. So, there is a high probability that cluster records follow the same pattern. The record that has abnormal distance will be identified as an irrelevant data.

3) DUPLICATES IDENTIFICATION

The duplication of data means, one entity has more than one representation. The best way to detect duplicate records is to make a comparison between records; compare each record with all other records. We used File checksum method to identify duplication of data. This method identifies duplication accurately and quickly by comparing the checksum of records.

B. TREATMENT

When the inaccurate, incomplete, irrelevant and inconsistent data is detected, then there is need to decide what to do with detected anomalies. Then there is only three options i.e. deletion, correction or leave unchanged. Inaccurate and incomplete values can never be left unchanged; they have to change with correct ones. Special algorithms used for these corrections.

We can eliminate duplication of data by extraction of file data checksum to exclude the false positives. False positive is avoided by comparing a new chunk with already stored chunks of data. Whenever a new file uploads, calculates the checksum and compared with the checksum of stored data. If the file is new then make a new entry and if it already exists then update the file.

TABLE 2. Transformation rules.

Rule ID	Rule definition	
T_1	Description:	The objective is to transform the relational database (SQL Server) table into Oracle NoSQL table.
	Expression:	[table (t-name)] = "TABLE"+ "CREATE"+ "-NAME"+ t-name.
T_2	Description:	The objective is to transform the row of relational database (SQL Server) table to row of Oracle NoSQL.
	Expression:	[ROW (Row)] = Record
T_3	Description:	The objective is to transform the columnof relational database (SQL Server) table to field of Oracle NoSQL.
	Expression:	[Fields (fields)] = Column
T_4	Description:	The objective is to transform the column name of relational database (SQL Server) to field name of Oracle NoSQL.
	Expression:	[FIELDS (data-type, item-name)] = "ADD"+ "-FIELD"+ "-TYPE"+ data-type+ item-name
T_5	Description:	The objective is to transform the data types of columns of relational database (SQL Server) into data types of fields of Oracle NoSQL.
	Expression:	[data-type (data-type)] = data-type data-type(integer) = "int" data-type(integer) = "bigint" data-type(double) = "double" data-type(string) = "string" data-type(Blob) = "Binary
T_6	Description:	The objective is to transform the primary key of relational database (SQL Server) table to primary key of Oracle NoSQL.
	Expression:	PRIMARY-KEY(primary-key)] = "PRIMARY"+ "-KEY" + " "+"-FIELD" + item-name
T_7	Description:	The objective is to transform the index of relational database (SQL Server) to index of Oracle NoSQL.
	Expression:	INDEX (index)] = "PLAN" + " " + "ADD"+ "-INDEX"+ "-NAME"+ index-name + "-TABLE" + t-name +"-FIELD" + item-name
T_8	Description:	The objective is to transform the users of relational database (SQL Server) to users of Oracle NoSQL.
	Expression:	[USERS (users)] = "CREATE" +"USER" + user-name + "IDENTIFIED"+ "BY"
T_9	Description:	The objective is to transform the permissions of relational database (SQL Server) to privileges of Oracle NoSQL.
	Expression:	[PRIVILIGES (privilges)] = "GRANT" + permission-name +"TO"+ role-name
T_{10}	Description:	The objective is to transform the of role relational database (SQL Server) to role of Oracle NoSQL.
	Expression:	[ROLE (role)] = "CREATE" +"ROLE"+ role-name
T_{11}	Description:	The objective is to transform the join of relational database (SQL Server) to parent child link of Oracle NoSQL.
	Expression:	[PARENT-CHILD LINK (parent-child link)] = "TABLE" + "CREATE" + -NAME" + t-name + "."+ childt-name
T_{12}	Description:	The objective is to transform the foreign key of relational database (SQL Server) to parent child link of Oracle NoSQL.
	Expression:	[REFRENCE-PARENT-CHILD LINK (parent-child link)] = "TABLE" + "CREATE" + -NAME" + t-name + "."+ childt-name

C. CONFIRMATION

Correction confirmation is necessary to ensure that data is accurate, consistent, relevant and complete. The corrections of data can be confirmed by comparing it with valid data set. After confirmation, we add a correction in a temporary file.

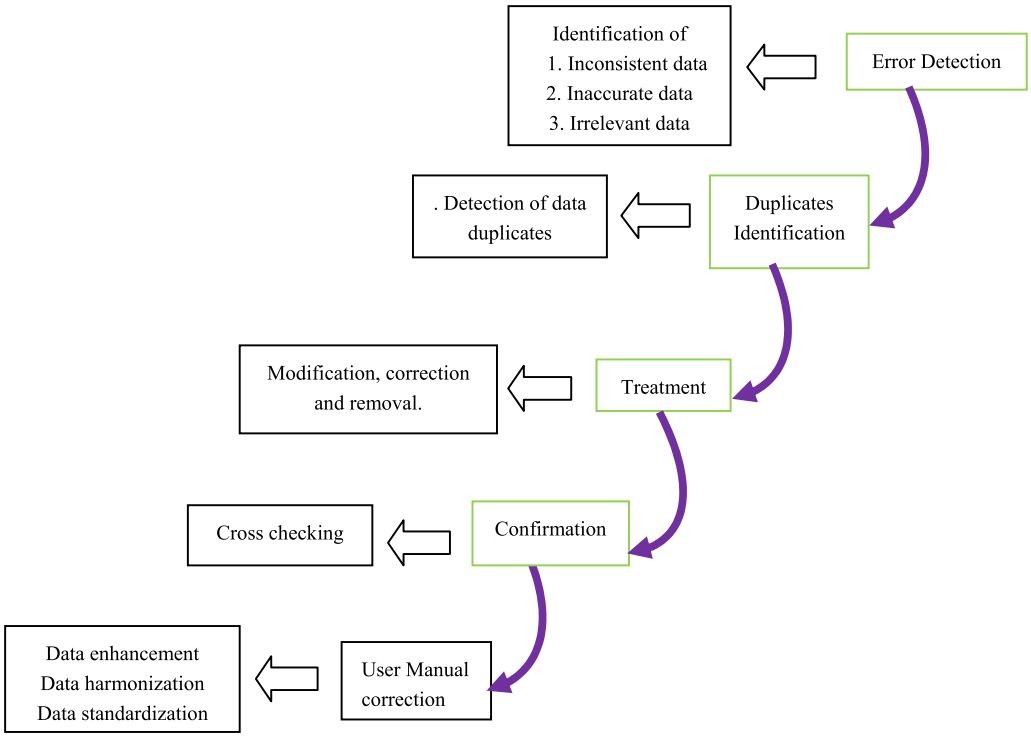


FIGURE 10. Data cleansing framework.

D. USER MANUAL CORRECTION

As a final step, allow the user to do manual correction: data enhancement, data harmonization and data standardization. Data enhancement made data more complete by adding related information, data harmonization: place actual code (student, address) instead of short codes (Stu, addr etc.) and data Standardization: uses standard codes. After all corrections made in the final step, data will be loaded to big data database i.e. Oracle NoSQL.

The proposed approach has two modules, transformation module and data cleansing module. The architecture of the proposed approach is shown in Fig. 11.

VI. RESULTS AND EVALUATIONS

SQL server is used as a source model and Oracle NoSQL is used as a target model. The following different databases are used for the evaluation of our proposed methodology:

- DB1: Users
  This database has four tables (Normal_Users, Facebook_Users, Users_Permissions, Users-systemadmin) [size: 1.4GB]
- DB2: Publications
  This database has five tables (Sample, Quat_result, Qula_result, Labname, Articles) [size: 2.1GB]
- DB3: Rentalproducts
  This database has eight tables (Customer, Address, City, Inventory, Staff, Store, Payment, Rental). [size: 3.5GB]
- DB4: Employees:
  This database contains five tables (Emp, Departments, Dept_Manager, salaries, Titles). [size: 1.2 GB]
- DB5: ShoppingTrade

This database has ten tables (Customers, Shippers, Emp, Orders, Orderdet, Suppliers, Regions, Country, Emp_Country, Products). [size: 1.6 GB].

The RDBs have joins that access data from multiple tables, but NoSQL databases have no join. The Oracle NoSQL database uses the parent-child relationship instead of join. These sample databases have single table, parent-child table and multiparent-child table. The multiparent-child tables take more time in conversion as compare to parent-child tables and parent child tables take more time in conversion than a single table, as shown in Fig. 12.

A. TRANSFORMATION EVALUATION METRICS

DBMS performance has been evaluated through different cost metrics. These metrics are also used to evaluate database transformation processes, but in a limited way. These metrics evaluate (a) schema information preservation, (b) data equivalence, and (c) system Performance [54].

1) SCHEMA INFORMATION PRESERVATION

Several user-based evaluation metrics have been proposed to verify the correctness of the schemas that are generated by schema mapping techniques. The comparison is performed between source and target schema to ensure that the target schema reflects all the semantics of the source schema. If the syntax and semantics of the source model

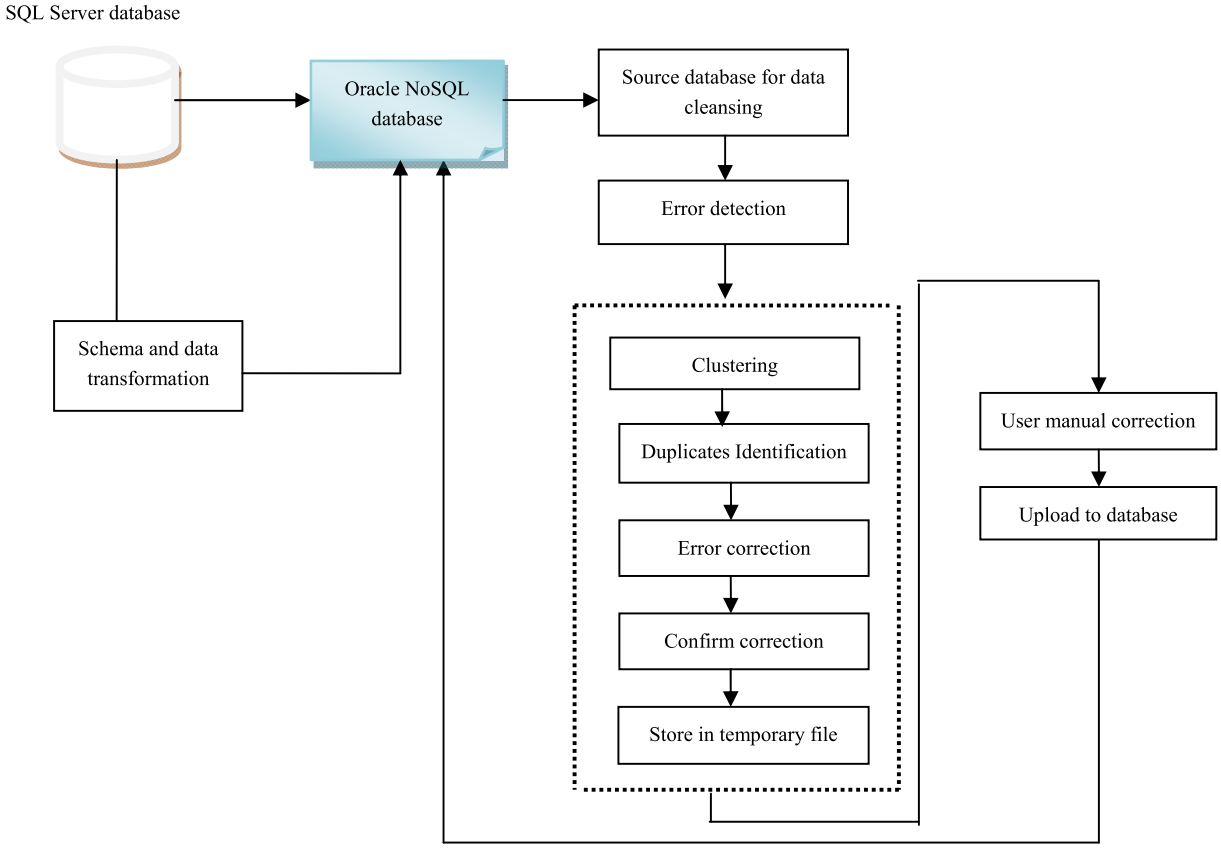


FIGURE 11. Architecture of proposed approach.

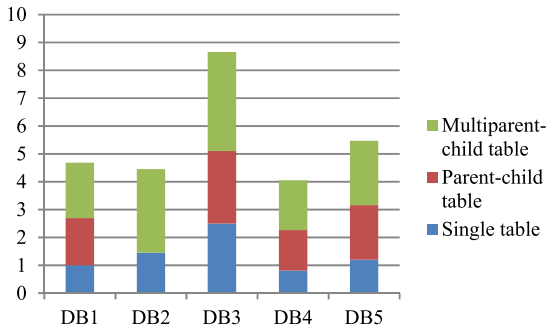


FIGURE 12. Database transformation time.

concepts have been used correctly, then the schema will be correct [55]. The automatic database migration approach could be validated by comparing its result with the result of an expert's manual approach [56]. The expert user who has an acquaintance of both databases can evaluate the target schema syntactically. The resulting schema can be compared with the existing literature that translated schema from the same input database [57]. The resulted schema cannot be compared because there is no example available in the existing literature that transform SQL Server database to Oracle NoSQL.

2) DATA EQUIVALENCE

The migration results also have been evaluated by querying source and target database. The data returned by the queries is compared to validate the generated schemas. Fong and Cheung evaluate their approach on the basis of query results, observing the differences between source and target schema [58]. We have evaluated the types of datasets and the key features of database with the help of the set of queries (insert, delete, retrieve, update for RDB and for Oracle NoSQL use, Put, Remove and Get methods instead of queries).

We tested the proposed system through a defined set of queries on three databases, DB3, DB4 and DB5. The results are manually inspected and the data equivalence is confirmed. The role of queries is defined as follows,

Query 1: SINGLE-INSERT

This query inserts a single row. Oracle NoSQL used put operation to create/insert new row by adding a key value pair.

Query 2: Delete

This query is used to remove single row. Oracle NoSQL used delete method to remove a particular record.

Query 3: MULTI –DELETE

This query removes multiple rows. Oracle NoSQL used multiDelete method to remove multiple rows.

TABLE 3. Query elapsed time in milliseconds for DB3.

No	Query	Elapsed time SQL Server	Elapsed time Oracle NoSQL
1.	Insert	7.38	10.9
2.	Delete	5.93	6.51
3.	Multidelete	6.36	8.51
4.	Retrieve	8.53	10.73
5.	Multiple retrieve	10.45	11.01
6.	Retrieve with single join	27.68	8.57
7.	Retrieve with Multiple join	30.12	16.71
8.	Update	7.78	7.38

TABLE 4. Query elapsed time in milliseconds for DB4.

No	Query	Elapsed time SQL Server	Elapsed time Oracle NoSQL
1.	Insert	6.38	8.79
2.	Delete	4.53	4.39
3.	Multidelete	5.43	5.98
4.	Retrieve	7.49	8.33
5.	Multiple retrieve	9.34	10.43
6.	Retrieve with single join	20.68	9.54
7.	Retrieve with Multiple join	28.12	15.32
8.	Update	6.17	6.33

Query 4: RETRIEVE
This query retrieves the single row. Oracle NoSQL used get operation to retrieve data.
Query 5: MULTIPLE-RETRIEVE
This query retrieves the multiple rows. Oracle NoSQL used multiGet operation to retrieve multiple records at once.
Query 6: RETRIEVE WITH SINGLE JOIN
This query retrieves records as join query of RDB. Oracle NoSQL don't have join, it has a parent - child relationship that works as a join. This query includes primary key of the parent table row as well as primary key of child table row.
Query 7: RETRIEVE WITH MULTIPLE JOIN
This query retrieves rows from the parent table and more than one child table.
Query 8: UPDATE
This query updates the row. Oracle NoSQL updates the row by using put operation if the key is already present.

3) SYSTEM PERFORMANCE COMPARISON

The quality of translation is ensured by evaluating system performance. Query elapsed time, the times that query takes to execute, is used as a performance evaluation metric by most of the DBMS. The performance and different aspects of the functionality of key-value stores is examined and measured through the developed set of queries. The queries are implemented to evaluate the system performance by measuring the elapsed time of each query. We performed experiments on Ci7 3.2 GHz processor with 8GB RAM and Ubuntu 14. The elapsed time (milliseconds) of each query for source and target databases is shown in Table 3 for DB3, Table 4 for DB4 and Table 5 for DB5.

We have run each query five times and the measurement is the average elapsed time of these runs. There is a possibility of caching results from the database, so we have changed the value in the predicate before submitting the query each time. The evaluation results are shown in Fig. 13. for DB3, Fig.14. for DB4 and Fig. 15.for DB5.

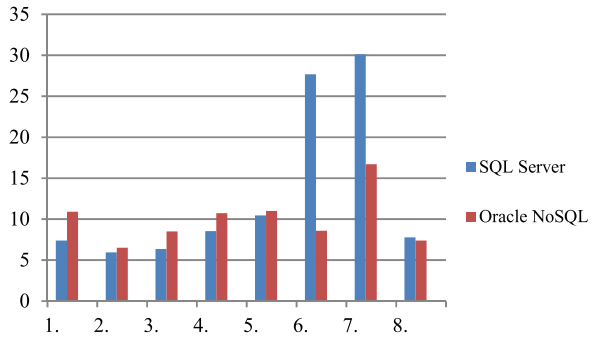


FIGURE 13. Performance analytics (Query elapsed time for DB3).

B. DATA CLEANSING EVALUATION METRICS

There are different parameters to evaluate the data cleansing methodology Recall, Precision and F-measure. The accuracy

TABLE 5. Query elapsed time in milliseconds for DB5.

No	Query	Elapsed time SQL Server	Elapsed time Oracle NoSQL
1.	Insert	8.21	12.69
2.	Delete	6.48	6.99
3.	Multidelete	7.89	9.12
4.	Retrieve	6.46	10.14
5.	Multiple retrieve	9.48	15.32
6.	Retrieve with single join	29.35	10.23
7.	Retrieve with Multiple join	33.48	18.10
8.	Update	8.79	9.76

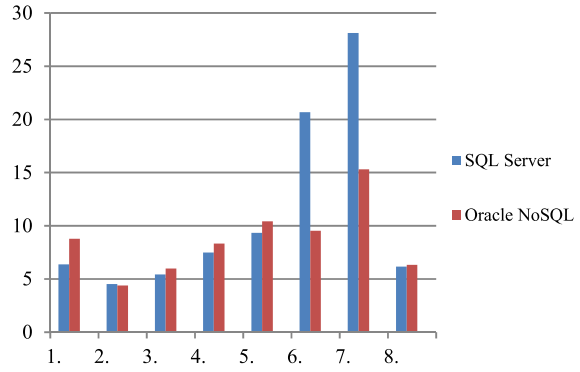


FIGURE 14. Performance analytics (Query elapsed time for DB4).

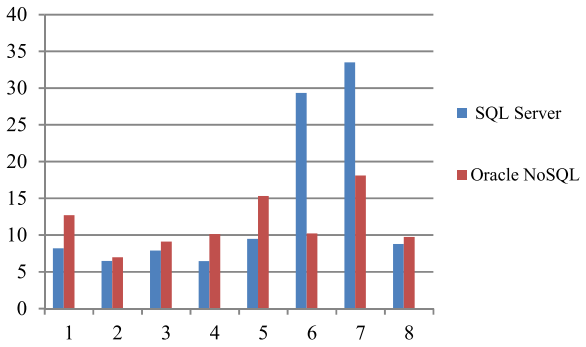


FIGURE 15. Performance analytics (Query elapsed time for DB5).

of the algorithm is determined by the number of errors detected. The correct detection of error is considered as correct positive (CP), when there is no error, but detected as an error is considered as incorrect positive (IP) and when there is an error but not detected as an error is considered as incorrect negative (IN).

The three exemplary accuracy measurement tools are used as assessment measurements: Precision, Recall and F-measure.

The first measure is precision that is the measure of correctly detected errors to all declared errors. Precision is represented by the following formula.

$$P = \frac{CP}{(CP + IP)} \tag{1}$$

Recall is a measure of correctly detected error to all positive detected errors. It is the completeness of the produced results. Recall is represented by the following formula.

$$R = \frac{CP}{(CP + IN)} \tag{2}$$

The third accuracy measure used here is called as F-measure. F-Measure is the harmonic mean of Recall and Precision. So, it is the measure of accuracy of the system and is represented by the following formula.

$$F - Measure = \frac{2PR}{(P + R)} \tag{3}$$

A set of five cases was selected to test the accuracy of the proposed cleansing methodology. The selected cases have a set of databases with different numbers of the respective tables in each database. We processed all these cases with our proposed methodology for data cleansing. Table 6 shows the results of database transformation and data cleansing whereas, each detected error was considered. Table 6 also shows that the rate of success of data cleansing for all five cases in terms of Recall, was as high as 84 to 91%. The proposed approach supported at the maximum for detection and correction of errors.

Fig. 16 shows the results of recall, precision and F-measure for all five different case studies. The results of these measurements show the accuracy of the proposed approach under the different database loads. The results depicted that our data cleansing approach is carried out successfully. The comparative analysis of our proposed data cleansing module with existing approaches presented in the literature shown in Table 7. The experimental results show that our approach provides the promising results in terms of efficiency and performance.

The used approach is novel and important that it automatically transforms the existing SQL Server database to Oracle NoSQL database and performs cleansing of the data of transformed database. The used approach uses model transformation for the schema as well as data transformation and proposed data cleansing module to improve data quality of the big data database (Oracle NoSQL). The result of the experiments shows the correctness of our approach and proved that our approach outperforms the other similar approaches.

TABLE 6. Accuracy metrics.

Case	Model Transformation details				Data cleansing evaluation					
	Total Transform-mations	Correct Transfom-mations	Incorrect Transfom-ations	Missed Transform-ations	CP	IP	IN	Precision (P) %	Recall (R) %	F-Measure (F)%
1	31	29	1	1	11	04	02	73.33	84.61	79.70
2	19	16	2	1	10	02	01	83.33	90.91	86.95
3	29	25	2	2	15	03	02	83.33	88.23	85.71
4	20	18	1	1	09	01	01	81.81	90.00	78.25
5	24	21	2	1	16	04	03	80	84.21	82.05

TABLE 7. Comparison of data cleansing approaches.

	Rational	Ease of implementation	Interactional	Protractile	Performance
Unified and sequential [30]	✗	✗	✓	✗	✗
AJAX [37]	✓	✗	✗	✓	✓
Alliance Rules [59]	✓	✗	✗	✓	✗
Potter's Wheel [60]	✗	✗	✓	✗	✓
Proposed approach	✓	✓	✓	✓	✓

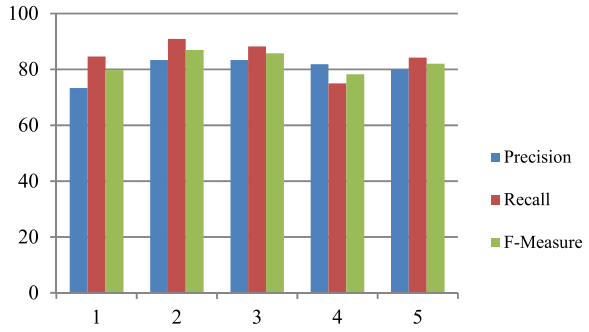


FIGURE 16. Evaluation results for data cleansing.

VII. CONCLUSION AND FUTURE WORK

This study proposed an approach that has two modules i.e. transformation module and data cleansing module. The transformation module automatically transforms SQL Server database into NoSQL key-value store. The developed system does the mapping of SQL Server database concepts to Oracle NoSQL concepts. Terminology of both databases is similar but each term has a different concept. Transformation rules are generated by this mapping and SiTra engine is used to execute these rules to perform the automatic transformation of SQL Server to Oracle NoSQL. For this transformation, we need metamodel of both databases such as source and target database. RDBs have standard metamodel but NoSQL databases have no metamodel therefore, we also proposed an initial version of the Oracle NoSQL metamodel. Data cleansing module proposed the methodology to clean the data of transformed database. It detects the error through parsing, clustering and duplicates identification techniques and afterward applies the corrections. Afterwards it stores all corrections in the temporary file and allow user to make manual corrections. Finally, upload the temporary file into the Oracle NoSQL database. The system has been implemented the modules of the proposed approach on various databases. Evaluation metrics are used to evaluate the modules of our approach. The results proved that the proposed approach is highly suitable for database transformation and data cleansing.

As a future direction, the approach can be extended to allow the multiple data inputs in case of huge volume of data. The transformation time can further be reduced by using advanced technologies. The data transformation module can be enhanced as a future work to support other RDBs and NoSQL graph databases. The data cleansing module can be further enhanced using statistical techniques for duplicate identification.

REFERENCES

[1] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[2] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland, "The end of an architectural era: (it's time for a complete rewrite)," in *Proc. 33rd Int. Conf. Very Large Data Bases*, Sep. 2007, pp. 1150–1160.

[3] M. Stonebraker and U. Cetintemel, ""One size fits all": Idea whose time has come gone," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, Apr. 2005, pp. 2–11.

[4] M. Stonebraker, D. Abadi, D. J. DeWitt, S. Madden, E. Paulson, and A. Pavlo, "MapReduce and parallel DBMSs: Friends or foes?" *Commun. ACM*, vol. 53, no. 1, pp. 64–71, Jan. 2010.

[5] M. Stonebraker, "What does 'big data' mean?" *Commun. ACM*, Sep. 2012. Accessed: Nov. 15, 2018. [Online]. Available: https://cacm.acm.org/blogs/blog-cacm/155468-what-does-big-data-mean/fulltext

[6] D. J. Abadi, "Data management in the cloud: Limitations and opportunities," *IEEE Data Eng. Bull.*, vol. 32, no. 1, pp. 3–12, Mar. 2009.

[7] S. Ramzan, I. Bajwa, and R. Kazmi, "An intelligent approach for handling complexity by migrating from conventional databases to big data," *Symmetry*, vol. 10, no. 12, p. 698, Nov. 2018.
[8] N. Li, B. Xu, X. Zhao, and Z. Deng, "Database conversion based on relationship schema mapping," in *Proc. Int. Conf. Internet Technol. Appl. (iTAP)*, Aug. 2011, pp. 1–5.
[9] R. Ouanouki, A. April, A. Abran, A. Gomez, and J. Desharnais, "Toward building RDB to HBase conversion rules," *J. Big Data*, vol. 4, p. 10, Dec. 2017.
[10] Y. Huang and T.-J. Luo, "NoSQL Database: A scalable, availability, high performance storage for big data," in *Proc. Joint Int. Conf. Pervasive Comput. Networked World*, vol. 2013, pp. 172–183.
[11] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011.
[12] E. A. Brewer, "Towards robust distributed systems," in *Proc. PODC*, Jul. 2000, pp. 1–12.
[13] A. Abdullah and Q. Zhuge, "From relational databases to NoSQL databases: Performance evaluation," *Res. J. Appl. Sci., Eng. Technol.*, vol. 11, no. 4, pp. 434–439, Oct. 2015.
[14] A. Davoudian, L. Chen, and M. Liu, "A survey on NoSQL stores," *ACM Comput. Surv.*, vol. 51, no. 2, p. 40, Apr. 2018.
[15] T. Jia, X. Zhao, Z. Wang, D. Gong, and G. Ding, "Model transformation and data migration from relational database to MongoDB," in *Proc. IEEE Int. Congr. Big Data (BigData Congr.)*, Jun./Jul. 2016, pp. 60–67.
[16] C.-H. Lee and Y.-L. Zheng, "Automatic SQL-to-NoSQL schema transformation over the MySQL and HBase databases," in *Proc. IEEE Int. Conf. Consum. Electron.-Taiwan*, Jun. 2015, pp. 426–427.
[17] D. Serrano, D. Han, and E. Stroulia, "From relations to multi-dimensional maps: Towards an SQL-to-HBase transformation methodology," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun./Jul. 2015, pp. 81–89.
[18] M. Mughees, "Data migration from standard SQL TO NoSQL," M.S. thesis, Univ. Saskatchewan, Nov. 2013. [Online]. Available: http://hdl.handle.net/10388/ETD-2013-11-1342
[19] F. Vale and L. Rocha, "Nosqlayer: A framework for migrating relational datasets to nosql models," *Revista Eletrônica de Iniciação Científica em Computação*, vol. 14, no. 3, pp. 1–10, 2011.
[20] C. Li, "Transforming relational database into HBase: A case study," in *Proc. IEEE Int. Conf. Softw. Eng. Service Sci.*, Jul. 2010, pp. 683–687.
[21] G. Zhao, Q. Lin, L. Li, and Z. Li, "Schema conversion model of SQL database to NoSQL," in *Proc. 9th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput.*, Nov. 2014, pp. 355–362.
[22] J. Anuradha, "A brief introduction on big data 5Vs characteristics and Hadoop technology," *Procedia Comput. Sci.*, vol. 48, pp. 319–324, Jan. 2015.
[23] G. A. Schreiner, D. Duarte, and R. dos Santos Mello, "Sqltokeynosql: A layer for relational to key-based NOSQL database mapping," in *Proc. 17th Int. Conf. Inf. Integr. Web-Based Appl. Services*, Dec. 2015, p. 74.
[24] G. Karnitis and G. Arnicans, "Migration of relational database to document-oriented database: Structure denormalization and data transformation," in *Proc. 7th Int. Conf. Comput. Intell., Commun. Syst. Netw.*, Jun. 2015, pp. 113–118.
[25] M. Hanine, A. Bendarag, and O. Boutkhoum, "Data migration methodology from relational to NoSQL databases," in *Proc. World Acad. Sci., Eng. Technol., Int. J. Comput., Elect., Automat., Control Inf. Eng.*, vol. 9, no. 12, pp. 2566–2570, Feb. 2016.
[26] L. Rocha, F. Vale, E. Cirilo, D. Barbosa, and F. Mourão, "A framework for migrating relational datasets to NoSQL," *Procedia Comput. Sci.*, vol. 51, pp. 2593–2602, Jan. 2015.
[27] J. Yoo, K.-H. Lee, and Y.-H. Jeon, "Migration from RDBMS to NoSQL using column-level denormalization and atomic aggregates," *J. Inf. Sci. Eng.*, vol. 34, no. 1, pp. 243–259, Jan. 2018.
[28] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*, Amsterdam, The Netherlands: Elsevier, 2011.
[29] J. J. Tamilselvi and V. Saravanan, "A unified framework and sequential data cleaning approach for a data warehouse," *Int. J. Comput. Sci. Netw. Secur.*, vol. 8, no. 5, pp. 117–121, May 2008.
[30] J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby, "A deferred cleansing method for RFID data analytics," in *Proc. 32nd Int. Conf. Very Large Data Bases*, Sep. 2006, pp. 175–186.
[31] H. Gonzalez, J. Han, and X. Shen, "Cost-conscious cleaning of massive RFID data sets," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 1268–1272.
[32] H. Chen, W.-S. Ku, H. Wang, and M.-T. Sun, "Leveraging spatio-temporal redundancy for RFID data cleansing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2010, pp. 51–62.
[33] C. Bornhövd, T. Lin, S. Haller, and J. Schaper, "Integrating automatic data acquisition with business processes experiences with SAP's auto-ID infrastructure," in *Proc. 13th Int. Conf. Very Large Data Bases*, vol. 30, Aug. 2004, pp. 1182–1188.
[34] R. Kohavi, L. Mason, R. Parekh, and Z. Zheng, "Lessons and challenges from mining retail e-commerce data," *Mach. Learn.*, vol. 57, nos. 1–2, pp. 83–113, Oct. 2004.
[35] Z. Zhao and W. Ng, "A model-based approach for RFID data stream cleansing," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2012, pp. 862–871.
[36] K. G. Herbert and J. T. L. Wang, "Biological data cleaning: A case study," *Int. J. Inf. Qual.*, vol. 1, no. 1, pp. 60–82, Apr. 2007.
[37] E. Dede, B. Sendir, P. Kuzlu, J. Hartog, and M. Govindaraju, "An evaluation of cassandra for Hadoop," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun./Jul. 2013, pp. 494–501.
[38] E. F. Codd, *The Relational Model for Database Management: Version 2*. Reading, MA, USA: Addison-Wesley, 1990.
[39] C. J. Date, *An Introduction to Database Systems*. London, U.K.: Pearson Education India, 2006.
[40] D. D. Chamberlin and R. F. Boyce, "SEQUEL: A structured english query language," in *Proc. ACM SIGFIDET (Now SIGMOD) Workshop Data Description, Access Control*, May 1974, pp. 249–264.
[41] C. Strozzi, "Nosql—A relational database management system," *Lainattu*, vol. 5, 1998. Accessed: Jan. 11, 2019. [Online]. Available: http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page
[42] A. Lith and J. Mattsson, "Investigating storage solutions for large data—A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data," Chalmers Univ. Technol., Gothenburg, Sweden, 2010. Accessed: Dec. 6, 2018. [Online]. Available: http://publications.lib.chalmers.se/records/fulltext/123839.pdf
[43] K. Orend, "Analysis and classification of NoSQL databases and evaluation of their ability to replace an object-relational persistence layer," *Architecture*, vol. 1, pp. 1–100, Apr. 2010.
[44] S. D. Kuznetsov and A. V. Poskonin, "NoSQL data management systems," *Program. Comput. Softw.*, vol. 40, no. 6, pp. 323–332, Nov. 2014.
[45] J. I. Maletic and A. Marcus, "Data cleansing: Beyond integrity analysis," in *Proc. Conf. Inf. Qual.*, MIT Sloan School Manage., Cambridge, MA, USA, 2000, pp. 200–209.
[46] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, "Advances in knowledge discovery and data mining," in *Advances in Knowledge Discovery and Data Mining*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996, pp. 1–34.
[47] R. Kimball, "Dealing with dirty data," *Proc. DBMS*, vol. 9, no. 10, pp. 55–60, Sep. 1996.
[48] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data Mining Knowl. Discovery*, vol. 2, no. 1, pp. 9–37, Jan. 1998.
[49] C. Fox, A. Levitin, and T. Redman, "The notion of data and its quality dimensions," *Inf. Process. Manage.*, vol. 30, no. 1, pp. 9–19, Jan./Feb. 1994.
[50] H. Galhardas, D. Florescu, D. Shasha, and E. Simon, "An extensible framework for data cleaning," in *Proc. ICDE*, Feb. 2000, p. 312.
[51] E. Rahm and H. H. Do, "Data cleaning: Problems and current approaches," *IEEE Data Eng. Bull.*, vol. 23, no. 4, pp. 3–13, Dec. 2000.
[52] M. Lenzerini, "Data integration: A theoretical perspective," in *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, Jun. 2002, pp. 233–246.
[53] M. M. Hamad and A. A. Jihad, "An enhanced technique to clean data in the data warehouse," in *Proc. Develop. E-Syst. Eng. (DeSE)*, Dec. 2011, pp. 306–311.
[54] A. Maatuk, "Migrating relational databases into object-based and XML databases," Ph.D. dissertation, Northumbria Univ., Newcastle upon Tyne, U.K., 2009. [Online]. Available: http://nrl.northumbria.ac.uk/id/eprint/3374
[55] C. Fahrner and G. Vossen, "A survey of database design transformations based on the entity-relationship model," *Data Knowl. Eng.*, vol. 15, no. 3, pp. 213–250, Jun. 1995.
[56] R. H. Chiang, T. M. Barron, and V. C. Storey, "A framework for the design and evaluation of reverse engineering methods for relational databases," *Data Knowl. Eng.*, vol. 21, no. 1, pp. 57–77, Dec. 1996.

[57] D. Lee, M. Mani, F. Chiu, and W. W. Chu, "Nesting-based relational-to-XML schema translation," in *Proc. WebDB*, May 2001, pp. 61–66.
[58] J. Fong and S. K. Cheung, "Translating relational schema into XML schema definition with data semantic preservation and XSD graph," *Inf. Softw. Technol.*, vol. 47, no. 7, pp. 437–462, May 2005.
[59] R. Arora, P. Pahwa, and S. Bansal, "Alliance rules for data warehouse cleansing," in *Proc. Int. Conf. Signal Process. Syst.*, May 2009, pp. 743–747.
[60] V. Raman and J. M. Hellerstein, "An interactive framework for data transformation and cleaning," Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/CSD-0-1110, 2000.

SHABANA RAMZAN received the M.C.S. degree from the University of Engineering and Technology Lahore, Pakistan, in 2004, and the M.S. degree in computer science from The Islamia University of Bahawalpur, Pakistan, in 2014, where she is currently pursuing the Ph.D. degree. She is also an Assistant Professor with the Government Sadiq College women University, Bahawalpur, Pakistan. Her current research interests include NoSQL databases, NoSQl databases security, cloud networking, and cloud computing. She is a Professional Member of ACM and the Vice Chair of ACM Chapter Bahawalpur.

IMRAN SARWAR BAJWA received the Ph.D. degree in computer science from the University of Birmingham, U.K. He is currently an Assistant Professor with the Department of Computer Science & IT, The Islamia University of Bahawalpur. He has published more than 130 papers in well-reputed peer-refereed conferences and journals. He has more than 850 citations in Google Scholar. In addition, he has more than 40 articles in ISI Web of Science and more than 70 articles in Scopus and 300 citations of his work in Scopus. He has more than 15 years of teaching research experience at various universities of Pakistan, Portugal, and U.K. He has very good programming skills in Java and C#. He was the Chair of ACM Bahawalpur Chapter, from 2016 to 2017.

BUSHRA RAMZAN received the B.S. and M.S. degrees in computer science from The Islamia University of Bahawalpur, Pakistan, in 2005 and 2014, respectively, where she is currently pursuing the Ph.D. degree. She is currently an Active Research Member of the AI Research Group, DCS & IT, The Islamia University of Bahawalpur. She is also an IT Officer with the National Bank Regional Head Office Bahawalpur, Pakistan. She has journal publications and also presented papers in conferences. Her current research interests include big data, NoSQL databases, cloud computing, and virtual telemedicine.

WAHEED ANWAR received the master's degree in computer science from The Islamia University of Bahawalpur Punjab, Pakistan (IUB), and the M.S. degree in computer science from IUB, where he is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science and Information Technology (DCS & IT). He has over 10 years of teaching, research and development experience. He is currently a Lecturer with the Department of Computer Science and Information Technology, The Islamia University of Bahawalpur. In addition, he is sun certified java programmer SCJP2. He published three research papers in recent years. His current research interests include text mining, web mining, machine learning, and deep learning.