ORIGINAL ARTICLE

SAMRIDDHI Volume 14, Special Issue 2, 2022 Print ISSN: 2229-7111 Online ISSN: 2454-5767

A case study: Database migration from RDBMS to NoSQL

Basant Namdeo[1*], Ugrasen Suman[2]

[1]International Institute of Professional Studies, Devi Ahilya University, Indore, India

[2]School of Computer Science & IT, Devi Ahilya University, Indore, India

Abstract

As database technologies progress, a common issue is database migration from one technology to another. RDBMS database has been used for 3-4 decades successfully by companies and various organizations. Since the last decade, the database size of an organization is growing rapidly and the format or the nature of the database is also changing. Therefore, companies are moving from RDBMS to a new type of database. These new types of databases come under the umbrella term, NoSQL. In this paper, we have proposed the overall architecture of database migration from RDBMS to NoSQL database. This includes issues related to database schema design, database reengineering cost calculation, database migration techniques, and middleware which translate the SQL to NoSQL query.

Keywords: RDBMS, NoSQL, Database migration, database reengineering, case study.

SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology (2022); DOI: 10.18090/samriddhi.v14spli02.21

Introduction

A relational database has been used successfully by many organizations for the last 3-4 decades. It follows the relational model for storing the information. It stores the information in a fixed format, i.e. in tabular format, and in the relationship between the columns of tables. As the business grows up of the organization, data generated for an organization also grows rapidly in large volume, and this data may also come from the different number of sources of devices. Relational database is not able the handle this type of data. So, different software companies have developed their own database model and software for the solution of this problem. Some of them are open source and some are proprietary database software. In this regard, Amazon developed DynamoDB, apache developed Cassandra, etc. These new types of database come under the umbrella term, NoSQL. NoSQL databases emerged in response to the limitations of the relational databases in handling the sheer volume, nature and growth of data. The term NoSQL stands mostly for 'Not only SQL', is a group of database, which does not follows the relational database model. Mainly, there are four categories of NoSQL database, namely; Key-Value, Columnar DB, Document DB, and Graph DB[1]. Each category of NoSQL database uses different type of data model for storing information. Organization may choose one from the various NoSQL databases, depending on their requirement and nature of information they want to store. NoSQL databases have many advantages over RDBMS such as efficient, scale-out architecture instead of monolithic architecture, capable of handling high volumes of structured, semi-structured, and unstructured data, well with today's software development methodologies that involve agile sprints, etc.[2] The above-given benefits of NoSQL database over RDBMS attract many organizations for use of NoSQL database.

Organizations which are already using RDBMS database now are migrating their database on NoSQL database due to the advantages given by NoSQL. Various phases are identified by various authors for database migration such as, schema conversion, SQL to NoSQL query translation, and ETL(Extract, Transform and Load).[3], [4] Lots of research works have been performed by various authors in order to improvement in each of the task of database migration.

Corresponding Author: Basant Namdeo, International Institute of Professional Studies, Devi Ahilya University, Indore, India, e-mail: basant_nd@yahoo.com

How to cite this article: Namdeo, B., Suman, U. (2022). A case study: Database migration from RDBMS to NoSQL. *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology,* Volume 14, Special Issue (2), 262-266.

Source of support: Nil

Conflict of interest: None

Database Reengineering Model

NoSQL databases have many advantages over RDBMS such as scalability, availability, schema design flexibility, etc. Growing companies have been attracted to NoSQL databases, due to the advantages given by them. Moving or migrating a

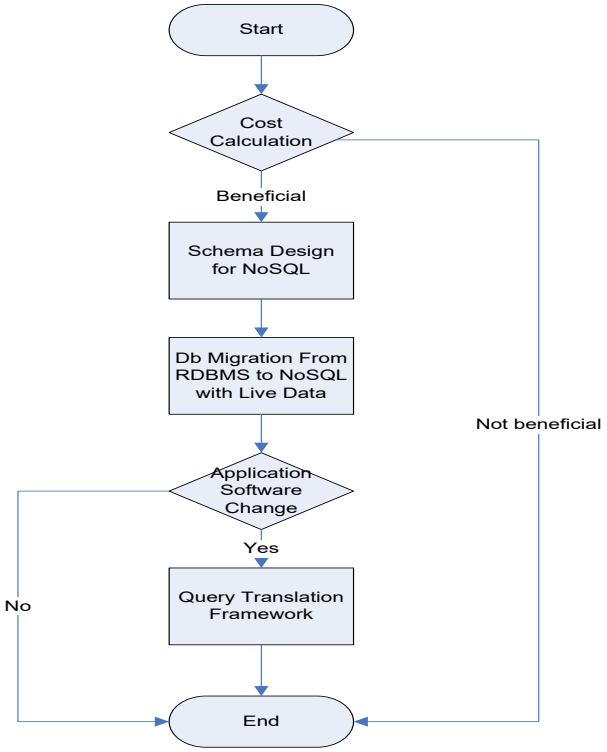

Figure 1: Database migration flow chart

database from one RDBMS software to another is simple, but when the database is migrated from RDBMS to NoSQL it becomes tough. Lots of tasks have to be done for migrating the RDBMS database to the NoSQL database. In this paper, we have suggested some of the tasks that have to be performed for successful database migration from RDBMS to NoSQL database. Figure 1 shows the flow chart for the database migration from RDBMS to NoSQL database.

Cost Benefit Analysis

Cost benefit analysis is the first task which has to be performed for any database reengineering project. Management has to decide whether it is required to performed database migration or not. They decide it by performing the cost benefit analysis. Cost of the migration, and benefit received after migration from the new system is calculated. The cost of database redesign, as well as cost of SQL query translation software for using the old application software with the new NoSQL database, and cost of database migration of static data as well as live data of organization must be considered for cost calculation of database migration.[5] Cost benefit analysis gives the answer, whether we have to go forward in the database reengineering phase or not.

Schema Design

Database schema design is how data is stored or organized in database. Different NoSQL database software stores database in different format, such as, document database stores data in document format, columnar database stores it in group of column-family format, and graph database stores it in property graph format. Schema of new database plays an important role in performance of data retrieval in NoSQL database. Related information is nested in a single document in document database as shown in Figure 2. Many researchers suggested various methods of designing a schema in NoSQL for RDBMS systems.[6][7] Various authors proposed schema suggestion model for NoSQL database by workload driven approach.[8], [9] A prototype, NoSE (NoSQL Schema Evaluator), is proposed which uses conceptual schema of database and SQL queries as input. [9] Binary Integer Program (BIP) method is used for availing the optimized NoSQL database schema. In this approach, appropriate BIP formulation and workload play an important role in creating optimized database schema. Figure 3 shows the schema design advisor model. By this model, user can find all the possible NoSQL database schema of a particular RDBMS database. This model receives the relational table structure, relationship of database tables, select query, and update queries used in the existing system as input, and it gives all the possible NoSQL documents structure with weightage of each database schema as output.

Database Migration

After designing the database schema, actual data from RDBMS to NoSQL has to be transferred. Various tools and software are available for such task.[10]–[12] These software can transfer the static RDBMS database to NoSQL database, but there are some cases in which live data or real time data must also be transferred to NoSQL database. Namdeo et al.[13] proposed a framework for snapshot and live data migration from RDBMS to NoSQL as shown in Figure 4. It uses change data capturer software, and parallelly transferred the static data also known as snapshot of data as well as live data in NoSQL database.

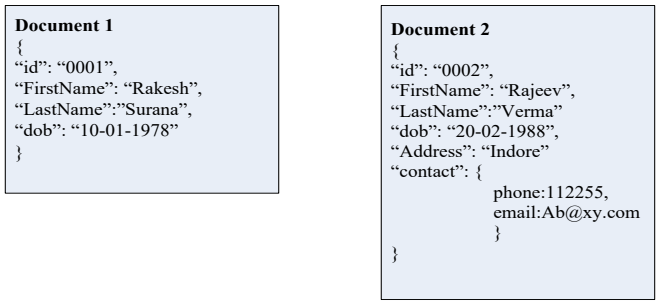

Figure 2: Document Database Schema Design

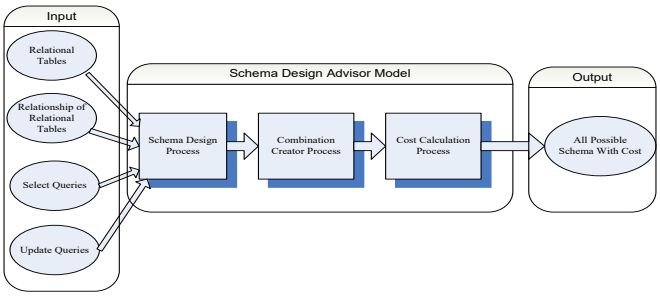

Figure 3: Schema Design Advisor Model [7]

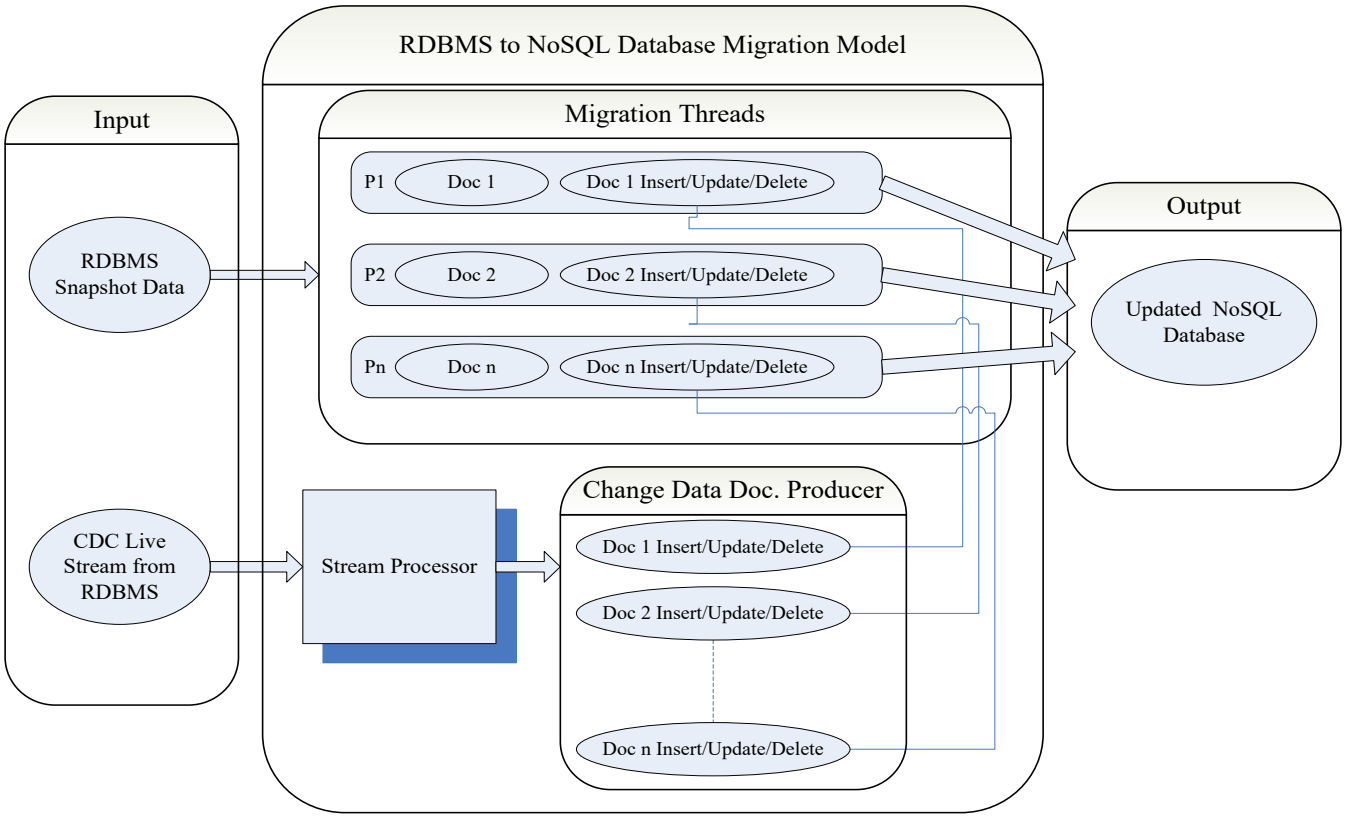

Figure 4: Snapshot-Live Stream Db Migration Model (SLSDMM) [13]

SQL to NoSQL query translation

Some of the organization still wants to use old application software. Their old application software uses relational databases. Now, the organization wants to move their works on new NoSQL database, but they don't want to change their old application software. In this case, there is a need for query translation framework. Query translation framework translates the SQL query into NoSQL query. It works as a middleware between old application software and NoSQL database. The middleware converts SQL queries to NoSQL query syntax and returns the result to the legacy application, which is what they expect from the database. Namdeo et al.[4] proposed a SQL to NoSQL Query Translation Model (SQL-No-QT), which translates the SQL query received from legacy application software to the query in MongoDB formate. It works as middleware layer between the legacy application software and NoSQL database driver. It gets the result from the NoSQL database, and that result is forwarded to the legacy application in the format required by it. Figure 5 shows the SQL to NoSQL Query Translation Model (SQL-No-QT) model.

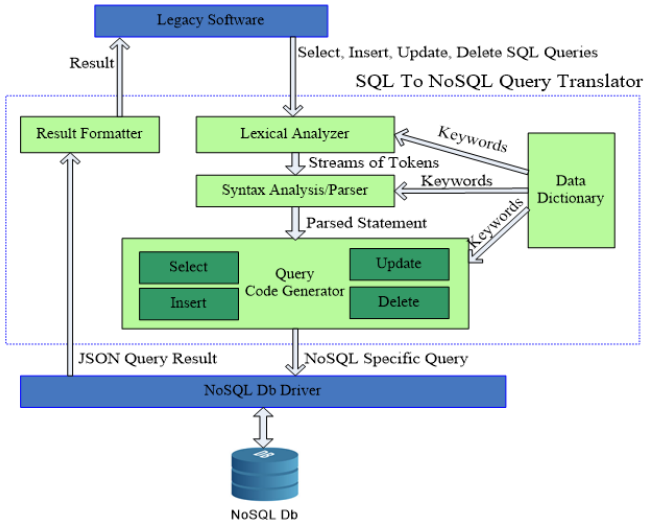

Figure 5: SQL to NoSQL Query Translation Model (SQL-No-QT) [4]

Case Study

In this section, we present a case study of database migration of an educational institute. We have followed the steps given in section 2 for database design and database migration from RDBMS to NoSQL database. Figure 6 gives the subset of database tables that are in RDBMS system. Table 1 describes the details of each database table. Using this table information, and sample select and update queries, which are frequently used in the legacy application, first we find all the possible database schema with its cost. Later, we choose the database schema from the entire possible database schema set, which has the highest cost benefit for the database migration process.

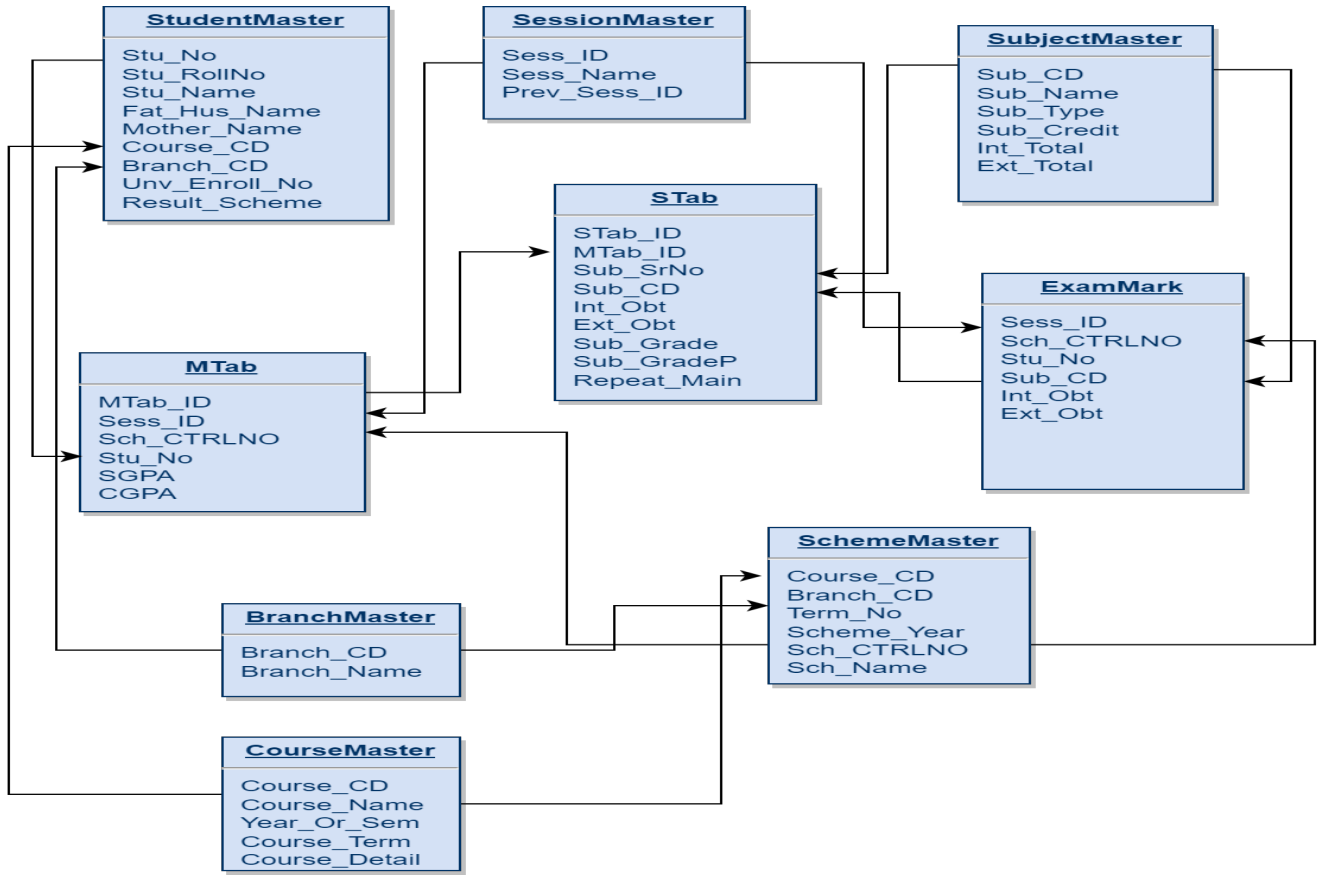

Figure 6: Database Schema of educational institute.

Table 1: Relational Table Description

Relational Table Name	*Description*
StudentMaster	Stores the information of all the students registered in department.
SessionMaster	Stores the information of each session semester wise with previous session information.
SubjectMaster	Stores the master information of subject taught.
CourseMaster	Stores the master information of course offered in department.
BranchMaster	Stores the master information of branches available in each course.
SchemeMaster	Stores the master information of each exam scheme.
ExamMark	Stores the marks obtained by each students in each subject in session and scheme wise.
MTab	Stores the master tabulation information after result prepared.
STab	Stores the child tabulation information after result prepared.

As shown in the above Figure 6, MTab and STab are the two tables which store the Tabulation data of a student in session and scheme wise. MTab is a master table and STab is a child table. Here, we can stores the STab data in MTab as child document as shown in Figure 7.

Thereafter, we have setup the migration system along with change data capturer (CDC) as described in the section 2.3. MySQL database stores the change data in binlogs. Maxwell's daemon[14] application is used for reading the

```
MTab
{
"Mtab_ID": "0002",
"Sess_ID": "202007",
"Sch"_CTRL:"1010101"
"Stu_No": "20200010",
"SGPA": "7.80",
"CGPA": "8.80"
"STab":[ {
                STab_ID:"S010",
                Mtab_ID:"0002",
                Sub_SrNo:1,
                Sub_CD:"CS-101",
                Int_Obt:35,
                Ext_Obt:52,
                Sub_Grade:"A+",
                Sub_GradeP:9,
                Repeat_Main:"M"
                },
        ]
}
```

Figure 7: Relational Table Equivalent Document Structure in MongoDB.

Table 2: SQL to NoSQL Query

Database	*Query*
SQL	Select * from StudentMaster where stu_rollno= 111222
NoSQL	db.StudentMaster.find({stu_rollno:111222})
SQL	insert into BranchMaster(branch_cd, branch_name) values(101, "Computer Sc.")
NoSQL	db.BranchMaster.insertOne({branch_cd:101, branch_name: "Computer Sc."})

change data from MySQL database. Maxwell application sends this changed data to Apache Kafka in a stream of data. This changed data stream is then read by the Java application in separate thread and stores this information in separate temporary database. A separate thread, which is migration a particular document in MongoDB, then reads this temporary database and updates the respective documents in MongoDB. Later, we have also setup the middleware layer as described in section 2.4, which translate the SQL query into NoSQL MongoDB specific query. Table 2 gives the sample SQL to NoSQL query translation code.

After implementing all the steps with java application software, we have successfully performed the migration of database from RDBMS to NoSQL, and also successfully developed the middleware to facilitate the translation of SQL query to NoSQL query for legacy application software.

Discussion

NoSQL database have many of advantages over RDBMS for big data such as, schema design flexibility, availability, scalability etc. In this case study, database migration from RDBMS to NoSQL database gives better performance in terms of execution time, ease of handling large data set, and flexibility in schema design for legacy application software and database. After migration, legacy applications works as it was working before the database migration with the help of SQL to NoSQL query translation middleware layer. These steps of migration could be helpful for those organizations which are planning to migrate their relational database to NoSQL database, and still wants to use legacy application software as front end application.

Conclusion

This paper emphasises and discussed the various phases such as, database schema design recommendation model, data migration, SQL to NoSQL query translation layer, of database migration from RDBMS to NoSQL database. In this paper, we have presented a case study of database migration from RDBMS to NoSQL database of an educational institute. This database migration process not only migrate the static database of the particular time or instance, but also live data which is captured by the change data capturer from the RDBMS database.

References

[1] Advantages of NoSQL | MongoDB. (n.d.). Retrieved March 5, 2022, from https://www.mongodb.com/nosql-explained/advantages

[2] Huseini, B., Ajdari, J., Raufi, B., Zenuni, X., & Ismaili, F. (2019). Transformation From Relational Database To Nosql Database : Migration and Integration. In I. Hashi (Ed.), *Proc. of 3rd International Conference on Business and Economics* (pp. 479–489). Retrieved from https://www.researchgate.net/profile/Desislava-Kalcheva-3/publication/337155638_ISCBE2019_Conference_Proceedings_FINAL/links/5dc870724585151435006dfc/ISCBE2019-Conference-Proceedings-FINAL.pdf#page=479

[3] Maxwell's daemon. (n.d.). Retrieved February 5, 2020, from https://maxwells-daemon.io/

[4] Mior, Michael J. (2014). Automated schema design for NoSQL databases. *Proc. of the ACM SIGMOD International Conference on Management of Data*, 41–45. https://doi.org/10.1145/2602622.2602624

[5] Mior, Michael Joseph, Salem, K., Aboulnaga, A., & Liu, R. (2017). NoSE: Schema design for NoSQL applications. *IEEE Transactions on Knowledge and Data Engineering*, *29*(10), 2275–2289. https://doi.org/10.1109/TKDE.2017.2722412

[6] mongoimport. (n.d.). Retrieved June 5, 2020, from https://docs.mongodb.com/manual/reference/program/mongoimport/

[7] Moniruzzaman, A. B. M., & Hossain, S. A. (2013). NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *International Journal of Database Theory and Application*, *6*(4), 1–13. https://doi.org/10.1016/S0262-4079(12)63205-9

[8] Namdeo, B., & Suman, U. (2020a). Performance analysis of schema design approaches for migration from RDBMS to NOSQL databases. In *Lecture Notes in Networks and Systems* (Vol. 94, pp. 413–424). https://doi.org/10.1007/978-981-15-0694-9_39

[9] Namdeo, B., & Suman, U. (2020b). Schema design advisor model for RDBMS to NoSQL database migration. *International Journal of Information Technology (Singapore)*, *13*(1), 277–286. https://doi.org/10.1007/s41870-020-00515-8

[10] Namdeo, B., & Suman, U. (2021). A Model for Relational to NoSQL database Migration: Snapshot-Live Stream Db Migration Model. *Proc. of 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 199–204. Retrieved from https://ieeexplore.ieee.org/document/9441829

[11] Namdeo, B., & Suman, U. (2022a). A Middleware Model for SQL to NoSQL Query Translation. *Indian Journal of Science and Technology*, *15*(16), 718–728. Retrieved from https://indjst.org/articles/a-middleware-model-for-sql-to-nosql-query-translation

[12] Namdeo, B., & Suman, U. (2022b). Cost Model for Database Reengineering from RDBMS to NoSQL. *Proc. of 4th IEEE International Conference on Recent Trends in Computer Science and Technology*. https://doi.org/10.1109/ICRTCST54752.2022.9781890

[13] Sqoop -. (2019). Retrieved June 5, 2020, from https://sqoop.apache.org/

[14] The Professional Client, IDE & GUI for MongoDB | Studio 3T. (n.d.). Retrieved May 15, 2021, from https://studio3t.com/