Original Research Paper

Complex SQL-NoSQL Query Translation for Data Lake Management

Nurhadi, Rabiah Abdul Kadir and Ely Salwana Mat Surin

Institute of IR4.0, University Kebangsaan Malaysia, Malaysia

Article history
Received: 28-08-2022
Revised: 05-10-2022
Accepted: 12-10-2022

Corresponding Author:
Rabiah Abdul Kadir
Institute of IR4.0, University Kebangsaan Malaysia, Malaysia
Email: rabiahivi@ukm.edu.my

Abstract: A data lake refers to an extremely large data resource or repository. Data lakes store large amounts of data and use advanced analytics to pair data from multiple sources with different types of structured, semi-structured, and unstructured information. NoSQL databases such as Mongodb, Redis, Neo4j, and Cassandra are nontabular and they store data differently rather than use relational tables. NoSQL databases come in many forms, mostly documents, key values, wide columns, and graphs based on their data model. NoSQL gives less complicated scalability and higher overall performance as compared with traditional relational databases. NoSQL databases can store different types of data, but they cannot fully support Automation, Consistency, Isolation, and endurance (ACID) features, i.e., trigger functions in multi-transaction management, because a NoSQL database uses a non-relational database system. Thus, an interpreter is necessary for SQL-to-NoSQL queries. We used ANTLR (ANother Tool for Language Recognition), which has five main stages: The input SQL query, the tokenizer, the parser, the parser tree, and lastly the generation of the query results in NoSQL. The tool gave users to write a flexible multi-pass language parser that is expected to solve problems in querying complex ACID functions and other problems in complex queries in NoSQL databases. In the measurement, analysis, and evaluation of the translation results through the comparison of each NoSQL criterion against a Relational Database Management System (MySQL), the scores obtained were as follows. The performance criterion achieved the highest score (98.40%) by the MongoDB database, followed by scalability (97.40%) and accuracy (97.00%). The criterion with the lowest score was complexity (91.65%).

Keywords: Translation, SQL, NoSQL, Data Lake, ANTLR

Introduction

The data lake process starts with data absorption and finishes with data transformation for analysis, with every component of the data described in the data lake ecosystem (Davoudian *et al*., 2018). It is allowing huge amounts of data in varied types and formats to be stored (Khine and Wang, 2018; Meena and Meena, 2016). A smart city has several components that use data lake storage technology to store a huge amount of data and has a NoSQL database that is a dataset and consists of several types of data groups (Cha *et al*., 2018). It is a non-relational database management system (non-religious). In other words, NOSQL does not require a sophisticated query and can flexibly handle the database. NoSQL is highly scalable, allowing it to grow per the demands of the data already in existence. Though it should come as no surprise that this database management is thought to be the best option for processing large amounts of constantly changing data (Flores *et al*., 2018). The Document Base, Key Value Base, Graph Store, and Wide Column Store are the types of NOSQL databases (Kamal *et al*., 2019). A data lake is a system for storing structured, semi-structured, unstructured, and binary data of any type in a format that exhibits 4V Big Data is a large fact function inclusive of volume, veracity, variety, and veracity (Khine and Wang, 2018). NoSQL databases can store different types of data, but they cannot fully support ACID (Automation, Integrity, Isolation, and Durability) features, i.e., the trigger functions in multi-transaction management (Lotfy *et al*., 2016), because a NoSQL database uses a non-relational database system (Schreiner *et al*., 2019; Li and Gu, 2019).

The data lake and a data warehouse require data filtering and classification processes to further study complex statement transactions, such as the use of trigger, procedure, group, and join functions in the data lake of a smart city (Hegde and Ravinarayana, 2016). Therefore, we want a translation technique from SQL query into NoSQL query that makes use of ANTLR (or Another Tool for Language Recognition) equipment to enable the proposed translation to support the management of a variety of complex and flexible statement transactions related to managing data lakes for smart cities. Appropriate tools are necessary to build a new translation algorithm for improving complex function queries (i.e., ACID) this has long been a problem in the processing and analysis of online transactions. The translation approach that uses ANTLR grammar definition tools is expected to address the problems in complex function queries (i.e., ACID). ANTLR also simplifies the process of creating lexers and parsers without creating files that resemble the results that ANTLR produces for iterative work.

Related Work

Learn about all aspects of the data in the data lake ecosystem, starting with the manipulation of the data absorption channel and ending with the transformation layer for analysis. To date, a data lake can store a large amount of data with various structures and formats. A data lake can also additionally comprise raw, unstructured records or records with many structures. Most of these data may have unrecognized value for an organization. Data lakes are a comprehensive, statistics-driven methodology primarily based on low-value times that complement the collection, improvement, archiving, and research of uncooked statistics within an organization.

With the abundance of data lakes, an efficient classification method for data sets is necessary to support data analysis and information retrieval. The objective is to use meta-data features that describe data sets to determine whether these data sets are similar. The data collected in a data lake can become inaccessible in the long run when their semantics are no longer available. The diversity of data formats and the amount of collected data in a data lake prohibit the manual cleaning and consolidation of data. A data lake is based on big data technologies, such as Hadoop, Spark, and Thread, which make measurable, distributed, reliable, and intolerable systems incorrect. Data are processed and modified on demand and then integrated with advanced analytics, automation, orchestration, and machine intelligence tools and languages (Hamouda and Zainol, 2017).

The NoSQL database model solves the problem of managing large amounts of data in various formats and types (i.e., constructed, semi-constructed, unconstructed) and processes quickly (Khatibi and Mirtaheri, 2019). Although NoSQL databases have several advantages in storing many categories and types of data, they still suffer from the limited consistency problem in the use of ACID functions and extremely complex data query transactions (Li and Gu, 2019). A NoSQL has four types of databases: Document bases, Key-value Stores, Graph Stores, and Wide Columns (Chen and Lee, 2019). The distinguishing features of a NoSQL include high-performance writing, great scalability, and free-write schemes (Imam *et al*., 2018). NoSQL compliance is less supported in the ACID class and some databases are loosening up concerning data consistency and scaling support to achieve data freshness and resilience (Kamal *et al*., 2019). For this limitation, prior researchers worked on a translator that could execute SQL queries in NoSQL.

Translator

Apache Flink is an open-supply framework for distributed circulation and batch data processing that runs on distributed clusters. Several interpreters provide SQL queries that translators run on top of NoSQL databases; examples include Hive, YS mart, S2 mart, and Qmapper. The translator was designed to run in Flink to execute advanced SQL queries. This is because Flink's scalability is limited for such queries. The proposed system runs on top of Flink without any changes in the structure of Flink (Das *et al*., 2019). It translates advanced SQL query translations into Flink codes to execute advanced SQL queries in Flink (Gouda *et al*., 2016).

ANTLR is a tool that automatically builds lexers and parsers using only grammar files. One of its tools is a library that can be used as a compiler for converting the source language into the target language (Chen *et al*., 2019). This parser generator library is based on the Java language (Klinbua and Vatanawood, 2017). The lexer is a component for detecting the smallest part (token) of a language. The detection process is called lexing. In general, regular expressions are used to perform the lexing process because the token form is typically extremely simple (Semura *et al*., 2018). Meanwhile, the parser is a component that detects the structure of several tokens. The detection process is called parsing. The process of creating a lexer and parser can be tedious if done manually. To address this issue, Terence Parr created ANTLR, which simplifies the process of generating automated lexers and parsers.

A related study by Pai *et al*. (2019) gives an overview of how to use Language Natural Processing and how to use regular expressions to map SQL queries. It performs various processes such as tokenization, lexical analysis, parsing, and semantic analysis to generate SQL queries equivalent to natural language queries.

Meanwhile (Ge *et al*., 2018) in his studies has proposed a massive facts question machine for custom-designed queries primarily based totally on particular commercial enterprise desires that introduce the additives and shape of the query machine. The ANTLR device is used as a language identifier to lay out and enforces custom-designed SQL dialects. The machine builds a less difficult and simpler question interface in Spark SQL, which meets the query necessities of Internet person conduct evaluation platforms.

Materials and Methods

SQL-NoSQL Translation

This section presents the process for the translation of SQL into NoSQL for data lake management by using ANTLR grammar definitions. Figure 1 shows the basis for implementing the current research framework.

As shown in Fig. 1, the process of the translation SQL-NoSQL is divided into five major stages: Input the SQL query, ANTLR (tokenizer), ANTLR (parser), ANTLR (parser tree), and generate query results in NoSQL. Each stage involves several processes with different functions. Firstly, the lexer detects the smallest part of a SQL statement, known as a token. The detection process is called lexing and the lexing process is typically performed because the token form is generally extremely simple (Semura *et al*. 2018). While the parser detects the structure of some tokens. The detection process is called parsing. Parsing is more complicated than lexing and thus, regular expressions are unsuitable for this research. The Abstract Syntax Tree (AST) represents the abstract syntax structure of source code written in a computer programming language (Mokhtari and Chaoui, 2017). Each node is built on the premise of critical parameters withinside the supply code. In AST, SQL query information can be added to the context of describing data from the created node to be passed down to the code generator. Graphically, a normal movement of tokens from a lexer to a parser would possibly seem like the following at some point throughout the parse, as shown in Fig. 2.

As shown in Fig. 2, the stream object is a filter that generates, processes, aggregates, or separates a stream of tokens for use by consumers (Kate *et al*., 2018). Present lexers and parsers can be combined in new and exciting ways without change.

Circulation-splitting filters are like prisms that cut up white mild into rainbows. The subsequent determine 3 illustrates a situation wherein an unmarried flow of tokens is broken up into three.

Figure 3 describes the flow filter included with ANTLR called the Token Flow Hidden. Token Filter behaves like a coin sorter, which sends coins to one trash can or another trash can.

In Fig. 4, the derivation tree is a non-circular connected graph with one vertex, called the root, which has a path to every other vertex (Semura *et al*., 2018). Descendant/parse trees are useful for explaining how a string (strand) can be obtained by transforming a variable symbol into a terminal symbol (Liu *et al*., 2019). Each variable symbol will be passed down to the terminal until nothing can be replaced. ANTLR can automatically generate lexers and parsers based on the input grammar (Ge *et al*., 2018; Chen *et al*., 2019).

The source of the query statement is entered, and the lexer processes the string and passes the resulting token to the parser (Chen *et al*., 2019). The parser analyzes the token and determines the syntax structure of the source program then an Abstract Syntax Tree (AST) is generated as shown in Fig. 4. Figure 5 explains how SQL to NoSQL translation works. Starting from the input process to the output of NoSQL by mapping to each category of the NoSQL database.

The experimental process used ANTLR 4.0 tool as a translator with 150 complex query data input in SQL on transportation data as a dataset. As shown in Fig. 5, the process begins with input in the complex SQL query and then translates into NoSQL queries. If the translation process is successful and per the grammar adjusted with NoSQL, then the next step is to display the NoSQL query output.

If the SQL query input does not match, then the following error message appears: 'SQL QUERY TYPE IS NOT AVAILABLE'. If the output of the NoSQL query is successful, then the process of mapping to the NoSQL database will continue and will be directed to the respective JDBC NoSQL drivers for database connection. These rules focus on defining the tokens that will form the foundation of parser rules using ANTLR with regular expressions as shown in the example Fig. 6.

These rules implicitly match the characters in the input stream, not the tokens in the token stream. Referenced grammatical elements include token references (Lexer implicit rule references), characters, and strings (Ge *et al*., 2018). Figure 6 indicates the evaluation guidelines advanced for the question scenario. The lexical policies are overlooked for a simplified layout. Lexer rules are treated the same as parser rules, so they can specify arguments and return values. In addition, as shown in Fig. 6, lexer rules have local variables and can also use recursion. Figure 7 shows the data set that is used in this study. The data set was imported into each NoSQL database (Mongodb, Redis, Neo4J, Cassandra). The characteristics and understanding of each NOSQL database are based on its type as follows.

Mongodb (Document Base)

Each data record is stored in the form of documents. Every document does not have to have the same structure as another document even though it is in one data set (in SQL terminology called a table). The advantage is that there is no need to determine the scheme. The changes of the scheme are needed by the application, meaning it does not require changes in all scheme data and no need for altering tables like SQL:

- Create table: Create a collection
- Create database: Create a database
- Data format: JSON/.BSON files

Redis (Key-Value Store)

Redis or Remote Dictionary Server is a provider of data structures in memory, which generally will be used as a database and cache to streaming machines. Redis can provide a variety of data structures such as strings and hashed and an innate replica:

- Create table: Create hash, list, set, sorter set, and string
- Create Database: Create a database
- Data format: .TXT files

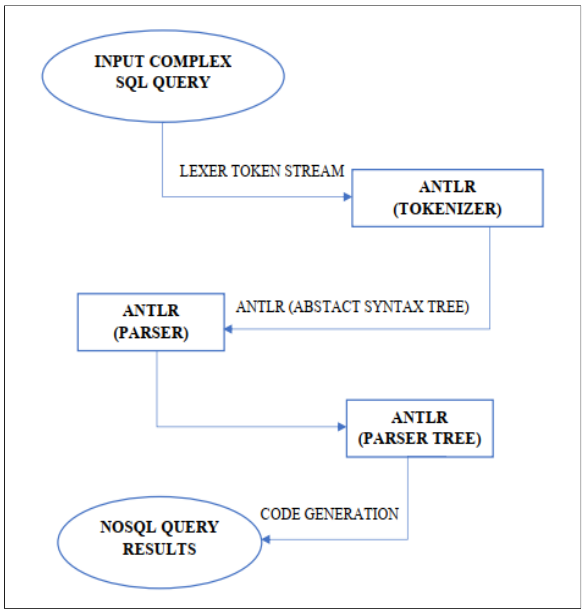


Fig. 1: Process for the translation of SQL into NoSQL

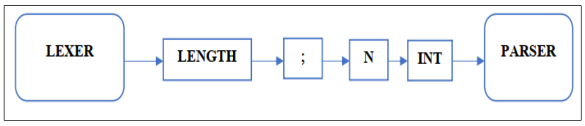


Fig. 2: Normal flow of tokens from lexer to parser in ANTLR

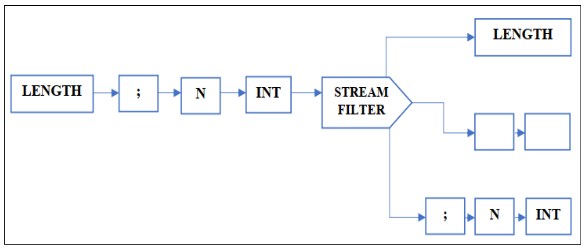


Fig. 3: Circulation-splitting filters of tokens

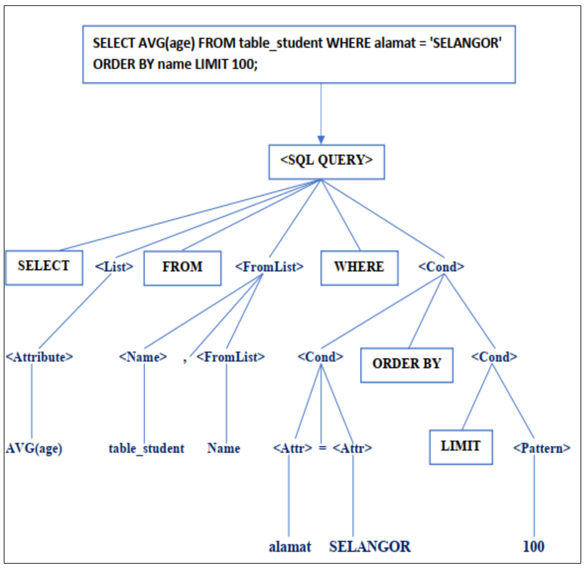


Fig. 4: Parser tree that uses ANTLR

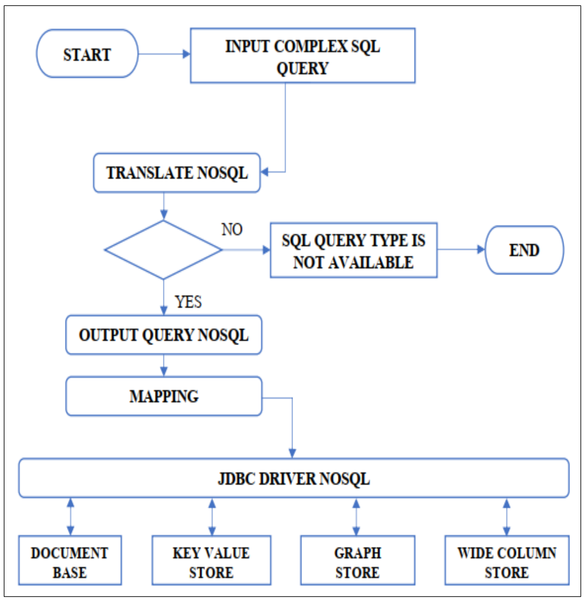


Fig. 5: Translation of SQL into NoSQL

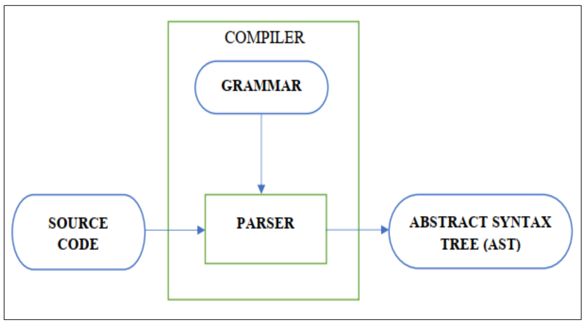


Fig. 6: Parser rules language parsing with ANTLR

Neo4j (Graph Store)

The database is designed for relations representation, such as graphs that consist of nodes and edges. The most popular uses are for social media, public transportation, maps, network topology, and others. Relationships are only in one table. The location points are represented as nodes and between locations as an edge:

- Create table: Create a label
- Create database: Create a database
- Data format: PLT files

Cassandra (Wide Column Store)

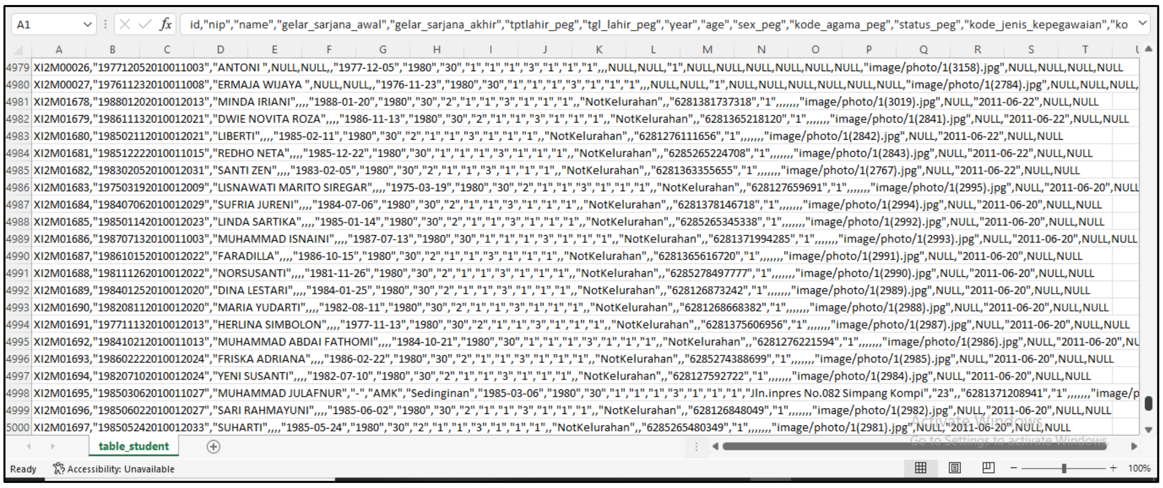
Data storage databases with models such as columns, using the concept of keyspace terms. A keyspace is like a schema in SQL. The key space containing Column Families (such as tables in SQL) consists of rows and columns:

- Create table: Create column family
- Create database: Create a keyspace
- Data format: CSV FILES

Figure 8 is an illustration of the data process from various data sources with various formats according to the NoSQL database type. After going through the metadata format process then the next step is the data import process into a NoSQL database with a different type of data, i.e., mongodb, Redis, Neo4J, and Cassandra.

Then the next step is syntax testing of 150 SQL complex queries. For Example SQL query sintax; " *SELECT id, name FROM table_student WHERE year = '1980' AND age > 30;* " then the results as shown in Fig. 9.

In Fig. 9, the analysis tree is shown in the graphic representation of the symbol. Parse tree follows the precedence of operators. The deepest sub-tree traversed first. Therefore, the operator of the parent node has a lower priority than the operator of the subtree. Figure 10 shows the result of converting a SQL query to NoSQL.



```
A1   fx  id,"nip","name","gelar_sarjana_awal","gelar_sarjana_akhir","tptlahir_peg","tgl_lahir_peg","year","age","sex_peg","kode_agama_peg","status_peg","kode_jenis_kepegawaian","ko

4979 XI2M00026,"197712052010011003","ANTONI ",NULL,NULL,,"1977-12-05","1980","30","1","1","1","3","1","1","1",,,NULL,NULL,"1",NULL,NULL,NULL,NULL,NULL,NULL,"image/photo/1(3158).jpg",NULL,NULL,NULL,NULL
4980 XI2M00027,"197611232010011008","ERMAJA WIJAYA ",NULL,NULL,,"1976-11-23","1980","30","1","1","1","3","1","1","1",,,NULL,NULL,"1",NULL,NULL,NULL,NULL,NULL,NULL,"image/photo/1(2784).jpg",NULL,NULL,NULL,
4981 XI2M01678,"198801202010012013","MINDA IRIANI",,,,"1988-01-20","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6281381737318","1",,,,,,,"image/photo/1(3019).jpg",NULL,"2011-06-22",NULL,NULL
4982 XI2M01679,"198611132010012021","DWIE NOVITA ROZA",,,,"1986-11-13","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6281365218120","1",,,,,,,"image/photo/1(2841).jpg",NULL,"2011-06-22",NULL,NULL
4983 XI2M01680,"198502112010012021","LIBERTI",,,,"1985-02-11","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6281276111656","1",,,,,,,"image/photo/1(2842).jpg",NULL,"2011-06-22",NULL,NULL
4984 XI2M01681,"198512222010011015","REDHO NETA",,,,"1985-12-22","1980","30","1","1","1","3","1","1","1",,"NotKelurahan",,"6285265224708","1",,,,,,,"image/photo/1(2843).jpg",NULL,"2011-06-22",NULL,NULL
4985 XI2M01682,"198302052010012031","SANTI ZEN",,,,"1983-02-05","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6281363355655","1",,,,,,,"image/photo/1(2767).jpg",NULL,"2011-06-22",NULL,NULL
4986 XI2M01683,"197503192010012009","LISNAWATI MARITO SIREGAR",,,,"1975-03-19","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"628127659691","1",,,,,,,"image/photo/1(2995).jpg",NULL,"2011-06-20",NULL
4987 XI2M01684,"198407062010012029","SUFRIA JURENI",,,,"1984-07-06","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6281378146718","1",,,,,,,"image/photo/1(2994).jpg",NULL,"2011-06-20",NULL,NULL
4988 XI2M01685,"198501142010012023","LINDA SARTIKA",,,,"1985-01-14","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6285265345338","1",,,,,,,"image/photo/1(2992).jpg",NULL,"2011-06-20",NULL,NULL
4989 XI2M01686,"198707132010011003","MUHAMMAD ISNAINI",,,,"1987-07-13","1980","30","1","1","1","3","1","1","1",,"NotKelurahan",,"6281371994285","1",,,,,,,"image/photo/1(2993).jpg",NULL,"2011-06-20",NULL,NULL
4990 XI2M01687,"198610152010012022","FARADILLA",,,,"1986-10-15","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6281365616720","1",,,,,,,"image/photo/1(2991).jpg",NULL,"2011-06-20",NULL,NULL
4991 XI2M01688,"198111262010012022","NORSUSANTI",,,,"1981-11-26","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6285278497777","1",,,,,,,"image/photo/1(2990).jpg",NULL,"2011-06-20",NULL,NULL
4992 XI2M01689,"198401252010012020","DINA LESTARI",,,,"1984-01-25","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"628126873242","1",,,,,,,"image/photo/1(2989).jpg",NULL,"2011-06-20",NULL,NULL
4993 XI2M01690,"198208112010012020","MARIA YUDARTI",,,,"1982-08-11","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6281268668382","1",,,,,,,"image/photo/1(2988).jpg",NULL,"2011-06-20",NULL,NULL
4994 XI2M01691,"197711132010012013","HERLINA SIMBOLON",,,,"1977-11-13","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6281375606956","1",,,,,,,"image/photo/1(2987).jpg",NULL,"2011-06-20",NULL,NULL
4995 XI2M01692,"198410212010011013","MUHAMMAD ABDAI FATHOMI",,,,"1984-10-21","1980","30","1","1","1","3","1","1","1",,"NotKelurahan",,"6281276221594","1",,,,,,,"image/photo/1(2986).jpg",NULL,"2011-06-20",NU
4996 XI2M01693,"198602222010012024","FRISKA ADRIANA",,,,"1986-02-22","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6285274388699","1",,,,,,,"image/photo/1(2985).jpg",NULL,"2011-06-20",NULL,NULL
4997 XI2M01694,"198207102010012024","YENI SUSANTI",,,,"1982-07-10","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"628127592722","1",,,,,,,"image/photo/1(2984).jpg",NULL,"2011-06-20",NULL,NULL
4998 XI2M01695,"198503062010011027","MUHAMMAD JULAFNUR","-","AMK","Sedinginan","1985-03-06","1980","30","1","1","1","3","1","1","1","Jln.inpres No.082 Simpang Kompi","23",,"6281371208941","1",,,,,,,"image/p
4999 XI2M01696,"198506022010012027","SARI RAHMAYUNI",,,,"1985-06-02","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"628126848049","1",,,,,,,"image/photo/1(2982).jpg",NULL,"2011-06-20",NULL,NULL
5000 XI2M01697,"198505242010012033","SUHARTI",,,,"1985-05-24","1980","30","2","1","1","3","1","1","1",,"NotKelurahan",,"6285265480349","1",,,,,,,"image/photo/1(2981).jpg",NULL,"2011-06-20",NULL,NULL

table_student
Ready   Accessibility: Unavailable                                                        100%
```

Fig. 7: Data set to be imported into the database NoSQL

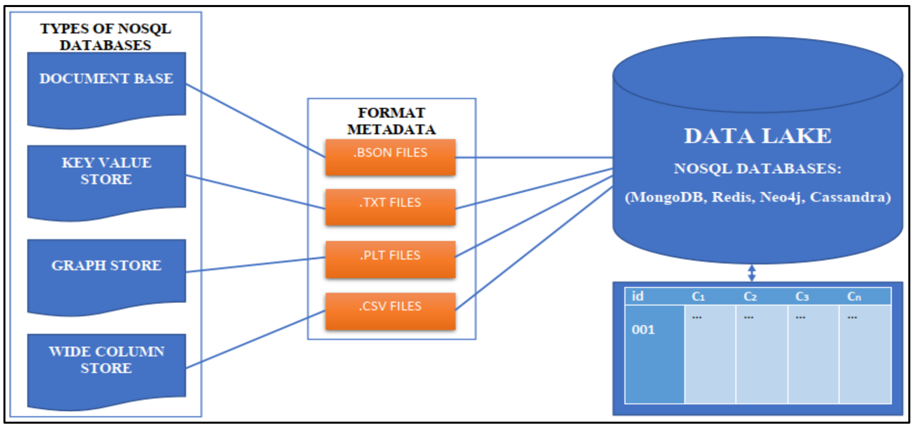


Fig. 8: Data source import process into NoSQL databases

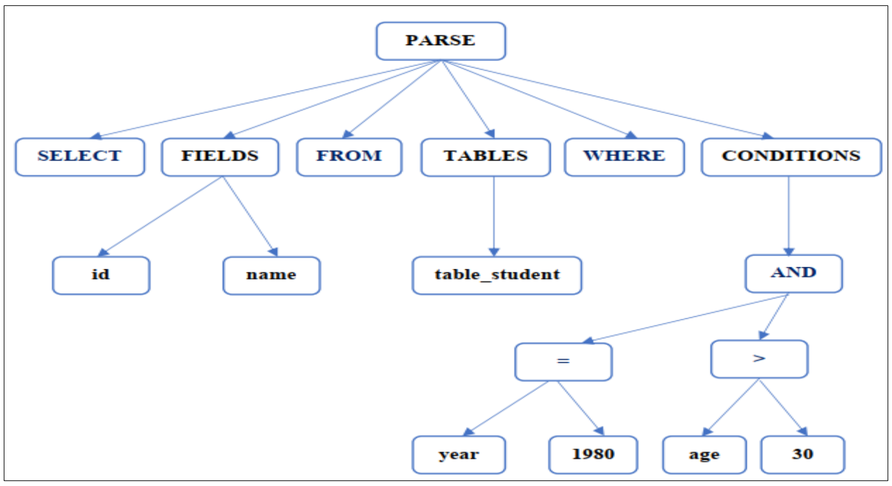
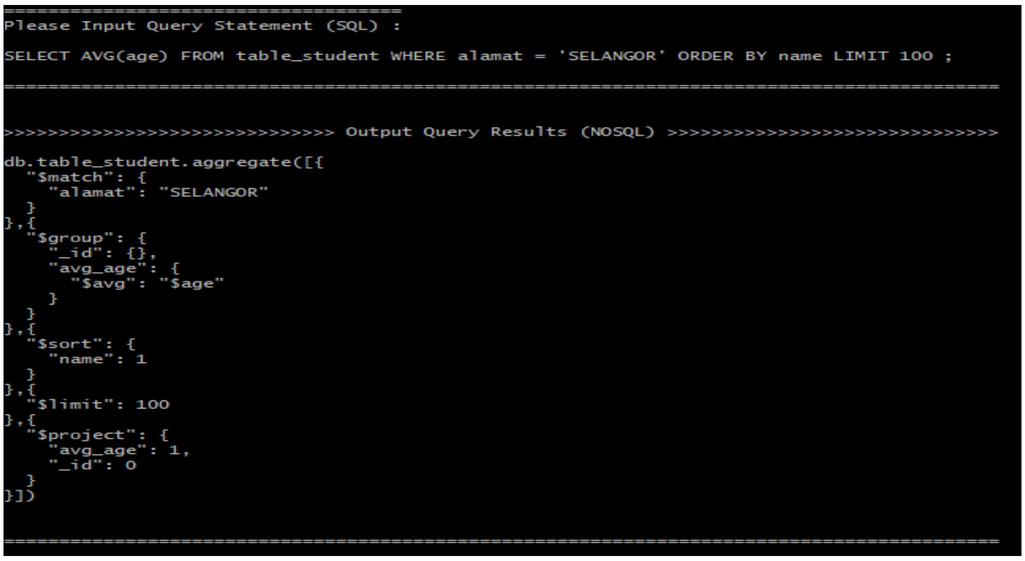


Fig. 9: Example of a query using a parser tree



```
=====================================
Please Input Query Statement (SQL) :
SELECT AVG(age) FROM table_student WHERE alamat = 'SELANGOR' ORDER BY name LIMIT 100 ;
===========================================================================================

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Output Query Results (NOSQL) >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

db.table_student.aggregate([{
   "$match": {
     "alamat": "SELANGOR"
   }
},{
   "$group": {
     "_id": {},
     "avg_age": {
       "$avg": "$age"
     }
   }
},{
   "$sort": {
     "name": 1
   }
},{
   "$limit": 100
},{
   "$project": {
     "avg_age": 1,
     "_id": 0
   }
}])

===========================================================================================
```

Fig. 10: The result of converting a SQL query to NoSQL

Results of Experiment

The ratio between the results of the RDBMS queries and the NOSQL Database is the focus of this study's results. The capability to translate SQL queries into NoSQL is identified by comparing the results between the MySQL and NoSQL databases, which are measured and evaluated. The measuring range is based on four characteristics and functions:

- Performance: Refers to the capability to find, analyze and then resolve various database congestion that can affect application response time or hinder application performance (Mahajan *et al*., 2019)
- Scalability: Scalability indicates the ability of a system to handle an increasing amount of work, or the ability to do more total work in the same elapsed time as the processing power grows to keep up with growth (Khasawneh *et al*., 2020; Flores *et al*., 2018)
- Accuracy: Accuracy denotes the capability to represent the right data in a form that is consistent, unambiguous, and most relevant to the historical records stored in computer-accessible digital media (Sánchez-de-Madariaga *et al*., 2018)
- Complexity: Complexity is the capability of the query to evaluate the function and size of an expression (Gil *et al*., 2019)

Results of the comparison for the average scores that used the functions and operations available in the Relational Database Management System (RDBMS) (MySQL) with NoSQL are obtained from the experiment as shown in Figs. 11-15. The result of NoSQL is referred to Mong DB, Redis, Neo4j, and Cassandra.

Performance

Comparison of RDBMS (MySQL) with NoSQL consisting of the following operations: Select, insert, update, delete and search. The average results are presented in Fig. 11, where the average total score of the NoSQL database (MongoDB, Redis, Neo4j, Cassandra) performance is 97.6% that of RDBMS (MySQL).

Scalability

Figure 11 shows operations, such as data storage (write), data retrieval (read), data sharding, data caching, and cluster management, from a comparison of RDBMS (MySQL) with NoSQL.

Accuracy

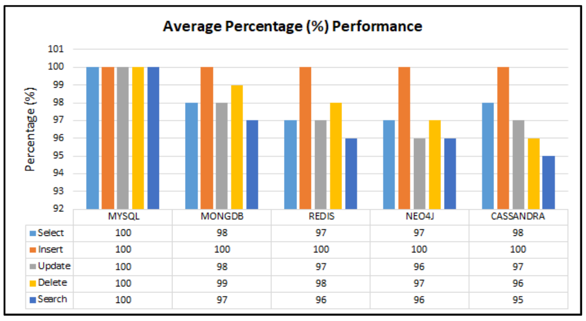
Comparison between RDBMS (MySQL) and NoSQL that includes operations, such as importing exporting, and loading data. The average results are illustrated in Fig. 13, where the average total score of the NoSQL database (MongoDB, Redis, Neo4j, Cassandra) accuracy criterion is 96.5% that of RDBMS (MySQL).

Complexity

Comparison of RDBMS (MySQL) with NoSQL that consists of operations, such as complex queries, functions, and combinations. Details of the results are shown in Table 1 and Fig. 14.

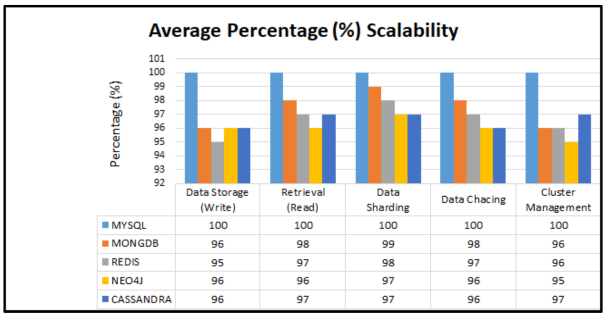
From the data comparison of the average percentage (%) of complexity indicated in Table 1 between RDBMS (MySQL) and NoSQL, the use of complex query operations can also be presented visually through the graphic image in Fig. 14.

The findings show that the comparison between RDBMS (MySQL) and NoSQL indicates that MySQL has a higher score than NoSQL in all the measurement criteria. For the measurement of each specific criterion for the NoSQL databases, the performance criteria reached the highest score (98.40%) by the MongoDB database, followed by scalability (97.40%) and then accuracy (97.00%). The lowest score criteria are complexity (91.65%), as shown in Fig. 15.



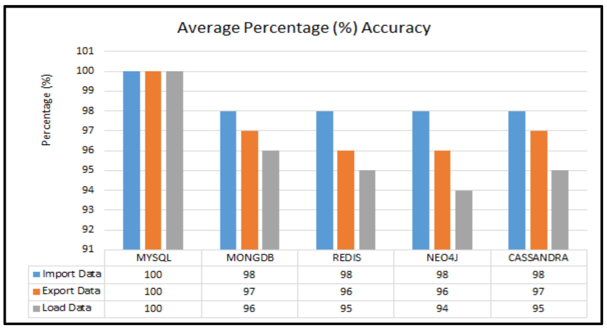
	MYSQL	MONGDB	REDIS	NEO4J	CASSANDRA
Select	100	98	97	97	98
Insert	100	100	100	100	100
Update	100	98	97	96	97
Delete	100	99	98	97	96
Search	100	97	96	96	95

Fig. 11: Comparison of mean values using performance criteria



	Data Storage (Write)	Retrieval (Read)	Data Sharding	Data Chacing	Cluster Management
MYSQL	100	100	100	100	100
MONGDB	96	98	99	98	96
REDIS	95	97	98	97	96
NEO4J	96	96	97	96	95
CASSANDRA	96	97	97	96	97

Fig. 12: Comparison of common values primarily based totally on the scalability criteria



	MYSQL	MONGDB	REDIS	NEO4J	CASSANDRA
Import Data	100	98	98	98	98
Export Data	100	97	96	96	97
Load Data	100	96	95	94	95

Fig. 13: Comparison of mean values according to accuracy criteria

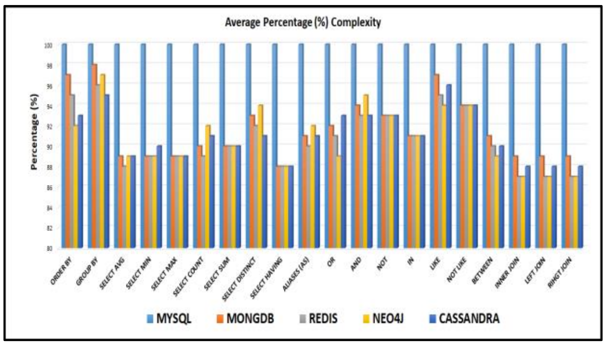
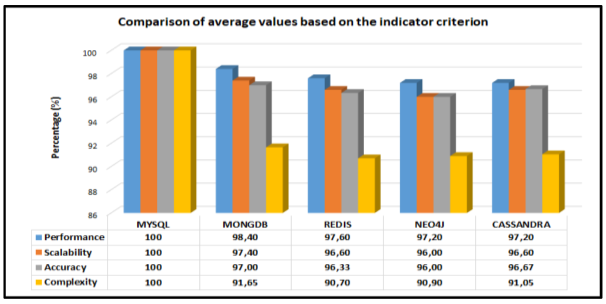


Fig. 14: Comparison of mean values according to complexity criteria



	MYSQL	MONGDB	REDIS	NEO4J	CASSANDRA
Performance	100	98,40	97,60	97,20	97,20
Scalability	100	97,40	96,60	96,00	96,60
Accuracy	100	97,00	96,33	96,00	96,67
Complexity	100	91,65	90,70	90,90	91,05

Fig. 15: Comparison of average values according to index criteria

Table 1: Average percentage (%) complexity

		RDBMS		NoSQL		
Number	Operation	MySQL	MongoDB	Redis	Neo4j	Cassandra
1	ORDER BY	100	97	95	92	93
2	GROUP BY	100	98	96	97	95
3	SELECT AVG	100	89	88	89	89
4	SELECT MIN	100	89	89	89	90
5	SELECT MAX	100	89	89	89	89
6	SELECT COUNT	100	90	89	92	91
7	SELECT SUM	100	90	90	90	90
8	SELECT DISTINCT	100	93	92	94	91
9	SELECT HAVING	100	88	88	88	88
10	ALIASES (AS)	100	91	90	92	91
11	OR	100	92	91	89	93
12	AND	100	94	93	95	93
13	NOT	100	93	93	93	93
14	IN	100	91	91	91	91
15	LIKE	100	97	95	94	96
16	NOT LIKE	100	94	94	94	94
17	BETWEEN	100	91	90	89	90
18	INNER JOIN	100	89	87	87	88
19	LEFT JOIN	100	89	87	87	88
20	RIGHT JOIN	100	89	87	87	88

Discussion

By comparing complex queries from two different database results types, namely RDBMS (MySQL) and NoSQL Database, it is feasible to conclude that the complexity characteristics remain relatively low compared to performance, scalability, and accuracy. As a result, it is required to refine and enhance the SQL to NoSQL translation in future research, particularly for complicated queries.

Conclusion

This study measures and evaluates the translation results of SQL into NoSQL by using ANTLR grammar definitions for the data lake of a smart city that includes several complex data query functions. The evaluation results are based on quantitative analysis through experiments. The results of the evaluation and analysis of NoSQL databases with appropriate RDBMS (MySQL) are based on performance, scalability, accuracy, and complexity. The most important technical translation output characteristics of any NoSQL database are studied when selecting the appropriate level of each database for the given features and capabilities. Although each NoSQL database nearly has the same scores in terms of performance and scalability, their scores in complexity and accuracy exhibit differences. From the test results, the criterion with the highest score is performance (97.6%), whilst the criterion with the lowest score is complexity (91.1%). The future work of this research will involve algorithm optimization and supporting complex transaction features to translate from SQL into NoSQL by using better methods.

Acknowledgment

We would like to express gratitude to the Universiti Kebangsaan Malaysia (UKM) for providing the opportunity and funding for this research under the Student Scheme Code TAP-20558.

Funding Information

This research article was supported by the Student Scheme provided by Universiti Kebangsaan Malaysia with the code TAP-20558.

Author's Contributions

Nurhadi: Involve in the implementation of the translation of complex SQL-NoSQL Query model development and experiment, analysis and write the draft of the manuscript.

Rabiah Abdul Kadir: Responsible for designing the translation of complex SQL-NoSQL Query model, examining the implementation, reviewing and writing the manuscript.

Ely Salwana Mat Surin: Review the manuscript.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the authors have read and approved the manuscript and that no ethical issues are involved.

References

Cha, B., Park, S., Kim, J., Pan, S., & Shin, J. (2018). International network performance and security testing based on distributed abyss storage cluster and a draft of data lake framework. *Security and Communication Networks*, *2018*. https://doi.org/10.1155/2018/1746809

Chen, J. K., & Lee, W. Z. (2019). An Introduction of NoSQL Databases based on their categories and application industries. *Algorithms*, *12*(5), 106. https://doi.org/10.3390/a12050106

Chen, W., Yang, Q., Xing, J., Zhao, S., Xue, G., & Wang, S. (2019, November). An ANTLR-Based Approach to Compiling the Programing Language of Insect Intelligent Building. In *2019 Chinese Automation Congress (CAC)* (pp. 3964-3969). IEEE. https://doi.org/10.1109/CAC48633.2019.8996500

Das, N., Paul, S., Sarkar, B. B., & Chakrabarti, S. (2019). NoSQL overview and performance testing of HBase over multiple nodes with MYSQL. In *Emerging technologies in data mining and information security* (pp. 269-279). Springer, Singapore. https://doi.org/10.1007/978-981-13-1498-8_24

Davoudian, A., Chen, L., & Liu, M. (2018). A survey on NoSQL stores. *ACM Computing Surveys (CSUR)*, *51*(2), 1-43. https://doi.org/10.1145/3158661

Flores, A., Ramírez, S., Toasa, R., Vargas, J., Urvina-Barrionuevo, R., & Lavin, J. M. (2018, April). Performance Evaluation of NoSQL and SQL Queries in Response Time for the E-government. In *2018 International Conference on eDemocracy & eGovernment (ICEDEG)* (pp. 257-262). IEEE. https://doi.org/10.1109/ICEDEG.2018.8372362

Ge, W., He, G., & Liu, X. (2018). Business-oriented customized big data query system and its SQL parser design and implementation. In *MATEC Web of Conferences* (Vol. 232, p. 01004). EDP Sciences. https://doi.org/10.1051/matecconf/201823201004

Gil, D., Johnsson, M., Mora, H., & Szymański, J. (2019). Review of the complexity of managing big data of the internet of things. *Complexity*, *2019*. https://doi.org/10.1155/2019/4592902

Gouda, Y. G., Mohammed, H. S., & Khafagy, M. H. (2016). Advanced SQL Query To Flink Translator. *Int. J. Appl. Inf. Syst*, *10,* 11-15. https://doi.org/10.5120/ijais2016451532

Hamouda, S., & Zainol, Z. (2017, August). Document-oriented data schema for relational database migration to NoSQL. In *2017 International conference on big data innovations and applications (innovate-data)* (pp. 43-50). IEEE. https://doi.org/10.1109/Innovate-Data.2017.13

Hegde, S. D., & Ravinarayana, B. (2016). Survey Paper on Data Lake. *International Journal of Science and Research*. 5(7), 1718-1720.

Imam, A. A., Basri, S., Ahmad, R., Watada, J., & González-Aparicio, M. T. (2018). Automatic schema suggestion model for NoSQL document-stores databases. *Journal of Big Data*, *5*(1), 1-17. https://doi.org/10.1186/s40537-018-0156-1

Kamal, S. H., Elazhary, H. H., & Hassanein, E. E. (2019). A qualitative comparison of nosql data stores. *International Journal of Advanced Computer Science and Applications*, *10*(2). https://doi.org/10.14569/ijacsa.2019.0100244

Kate, A., Kamble, S., Bodkhe, A., & Joshi, M. (2018, March). Conversion of natural language query to SQL query. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 488-491). IEEE. https://doi.org/10.1109/ICECA.2018.8474639

Khasawneh, T. N., AL-Sahlee, M. H., & Safia, A. A. (2020, April). Sql, newsql and nosql databases: A comparative survey. In *2020 11th International Conference on Information and Communication Systems (ICICS)* (pp. 013-021). IEEE. https://doi.org/10.1109/ICICS49469.2020.239513

Khatibi, E., & Mirtaheri, S. L. (2019). A dynamic data dissemination mechanism for Cassandra NoSQL data store. *The Journal of Supercomputing*, *75*(11), 7479-7496. https://doi.org/10.1007/s11227-019-02959-7

Khine, P. P., & Wang, Z. S. (2018). Data lake: A new ideology in the big data era. In *ITM web of conferences* (Vol. 17, p. 03025). EDP Sciences. https://doi.org/10.1051/itmconf/20181703025

Klinbua, K., & Vatanawood, W. (2017, November). Translating tosca into docker-compose yaml file using antlr. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (pp. 145-148). IEEE. https://doi.org/10.1109/ICSESS.2017.8342884

Li, C., & Gu, J. (2019). An integrated approach of hybrid databases based on SQL in a cloud computing environment. *Software: Practice and Experience*, *49*(3), 401-422. https://doi.org/10.1002/spe.2666

Liu, Y., Kumar, M., Katul, G. G., & Porporato, A. (2019). Reduced resilience as an early warning signal of forest mortality. *Nature Climate Change*, *9*(11), 880-885. https://doi.org/10.1038/s41558-019-0583-9

Lotfy, A. E., Saleh, A. I., El-Ghareeb, H. A., & Ali, H. A. (2016). A middle-layer solution to support ACID properties for NoSQL databases. *Journal of King Saud University-Computer and Information Sciences*, *28*(1), 133-145. https://doi.org/10.1016/j.jksuci.2015.05.003

Mahajan, D., Blakeney, C., & Zong, Z. (2019). Improving the energy efficiency of relational and NoSQL databases via query optimizations. *Sustainable Computing: Informatics and Systems*, *22*, 120-133. https://doi.org/10.1016/j.suscom.2019.01.017

Meena, S. D., & Meena, M. S. V. (2016). Data lakes-a new data repository for big data analytics workloads. *International Journal of Advanced Research in Computer Science*, *7*(5), 65-66.

Mokhtari, R., & Chaoui, A. (2017). Compiling, verifying, and simulating dynamic software architectures using ANTLR and colored-ADL. *International Journal of Communication Networks and Distributed Systems*, *19*(4), 406-433. https://doi.org/10.1504/IJCNDS.2017.087389

Pai, L. I. U., Yan, S. U. N., & Wei, R. E. N. (2019). An ANTLR-Based Feature Extraction and Detection System for Scratch3. 0. *Journal of Beijing University of Posts and Telecommunications*, *42*(6), 70. https://doi.org/10.13190/j.jbupt.2019.125

Sánchez-de-Madariaga, R., Muñoz, A., Castro, A. L., Moreno, O., & Pascual, M. (2018). Executing complexity-increasing queries in relational (MySQL) and noSQL (MongoDB and EXist) size-growing ISO/EN 13606 standardized EHR databases. *JoVE (Journal of Visualized Experiments)*, (133), e57439. https://doi.org/10.3791/57439

Schreiner, G. A., Duarte, D., & Mello, R. D. S. (2019). When relational-based applications go to NoSQL databases: A survey. *Information*, *10*(7), 241. https://doi.org/10.3390/info10070241

Semura, Y., Yoshida, N., Choi, E., & Inoue, K. (2018, December). Multilingual detection of code clones using antlr grammar definitions. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 673-677). IEEE. https://doi.org/10.1109/ICCIDS.2019.8862080