International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 08 | August - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

A Model for Migrating Relational Database to NoSQL Database

Pradeep Shiriyannavar
Department of MCA
Dayananda Sagar College of Engineering
Kumaraswamy layout, Bangalore, India.

Dr. Srinivasan V
Asst professor
Department of MCA
Dayananda Sagar College of Engineering
Kumaraswamy layout, Bangalore, India.

1 Abstract:

Relational Database Management Systems (RDBMS) is widely used to store the Data and Information. As we already know RDBMS can store the data in the form of table but if we want to know the complex data relationships and interconnected information, we have to convert the relational database to graph database. Professionals working with airline databases will be able to quickly query highly connected data thanks to graph databases' ability to quickly uncover and manage new and beneficial associations. Relational databases use expensive huge joins to query related data, whereas graph data-stores contain direct references to their neighboring nodes. In order to handle the enormous quantity of airline data being created at a very high pace, scalability was therefore greatly needed. Additionally, the majority of airline data is semi- or un-structured, necessitating a schema-less database.

In this paper a methodology to issues that occurs while converting an Airline relational to a graph database. By utilizing the source's restrictions and schema. Conjunctive SQL searches over the source are supported by the method and converted into graphs. procedures that traverse the target. To demonstrate the viability of the solution and the effectiveness of query response across the target database, the experimental outcomes are shown. Nodes are mapped to tuples, while edges are mapped to foreign keys.

Constraints were maintained during the change. In order to build the suggested solution, SQL2Neo, popular graph database Neo4j and MySQL as relational databases were both used.

Index Terms—Database migration, Graph, data, attributes, Relationships, tables.

2 Introduction:

The basis for many technical developments and discoveries, data is the lifeblood of the digital age. The simplest definition of data is unprocessed, raw facts, numbers, and statistics. But in the linked world of today, data refers to a wide range of information gathered from many sources, including people, organizations, and technology.

Data gathering and analysis underwent a substantial change in the early 2000s. The creation and storage of data were fundamentally altered by the expansion of the internet and the advancement of digital technologies. The phrase "big data" first appeared when the volume of data produced increased dramatically. During this time, conventional sources like transaction records, surveys, and scientific experiments were the main sources of data generation. But as social media platforms, smartphones, and devices with sensors proliferated, data creation soared to previously unheard-of heights. Data has increased exponentially as a result of the advent of e-commerce, online banking, and digital communication.

As the amount of data increased quickly, so did the difficulties in managing and analyzing it. This led to the rise of new fields like data

Airplane_id	Pilot_id	Departure	Destination	Duration	Price
1881	PI001	Bangalore	New Delhi	3h 15m	4800
1882	PI002	Kolkata	Bangalore	2h 30m	6500
1883	PI003	New Delhi	Goa	2h 50m	4200
1884	PI004	Goa	Kolkata	3h 40m	5200

science and data engineering, which concentrate on deriving useful insights from huge databases.
Traditional databases were increasingly complemented by more sophisticated systems, such as data warehouses and distributed computing frameworks, when it came to of data storage. Large-scale datasets might be stored, processed, and retrieved effectively because to these technologies.

2.1 Relational Database Management System (RDBMS):

Relational Database Management System (RDBMS) is made to manage and arrange data in an organized way. The relational model, on which it is built, arranges data into tables with rows and columns. Data integrity and consistency are ensured by the actions and functionalities offered by RDBMS for storing, retrieving, updating, and manipulating data. It permits the creation of key connections between tables, facilitating effective querying and data retrieval. RDBMS has been widely used in many different fields and applications because of its ease of use, adaptability, and capacity for handling massive volumes of data.

2.2 Graph Database:

A special type of database known as a "graph database" employs a graph data model to represent and store data. Comparatively to conventional relational databases, which use tables, a graph database organizes data using nodes, edges, and characteristics [12]. While nodes alone serve as a representation of things or objects, edges and attributes provide additional information about nodes and edges.

This data architecture is especially useful for describing and querying intricate linkages and interactions between things. In circumstances where connections are essential, such as social networks, fraud detection, recommendation systems and knowledge graphs to comprehending and analyzing the data, graph databases flourish. They are well suited for managing linked and intricately associated data because to their robust traversal and graph-based querying features.

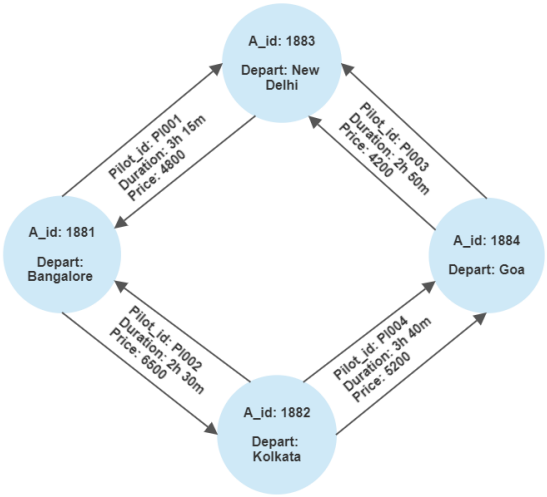


Figure 1: Data Store in graph database

All of this resulted in the development of "unstructured data," which are data that are less related to one another. Because standard RDBMS were unable to adequately store this unstructured data, unstructured data-supporting databases were called upon, which led to the development of NoSQL databases. There are several types of NoSQL databases, including graph databases [9], document-oriented data stores [11], columnar databases, and key-value pair data stores [10]. Only graph databases have been explored in this study.

Nodes and the edges indicating the relationships between them make up a graph. Edges have distinct names and characteristics. In graph databases, data exists as a graph, but queries access the data via graph traversal techniques. This promotes flexibility and enables the expansion of graph databases to very large graph data sets. In Figure 1, a straightforward graph representation

has been shown. The database displayed illustrates how certain friends relate to one another and how that information is kept in a database. Each node has unique attributes that are saved as key-value pairs. A label can be assigned to each node to be used for querying at a later time. The situation with relations is the same. The boundaries specify partnerships and relationship categories. They might also have attributes that could be kept as key-value pairs as well.

3 Literature Survey:

Although there are many applications for the semantic web, blogs, social networking, etc. that should be implemented in graph databases, this paper discusses and implements the transfer of relational medical databases. It is addressed in advance how different tools and technologies were employed throughout implementation.

In a Study of "A Model for Relational to NoSQL database Migration: Snapshot-Live Stream Db Migration Model" By Basant Namdeo and Ugrasen Suman in 2021. They suggested approach is implemented in Java for the MongoDB document database and MySQL (RDBMS). The prototype migrates both live data and a database instance at a certain moment (a database snapshot) concurrently [2]. The major focus of the technique is the migration of two things, specifically RDBMS snapshot data and RDBMS live stream of changed data. The method considers ListOfColls, a collection or array of JSON files containing snapshot data from RDBMS tables for all of MongoDB's collections. Every set is read from listofColls, a new thread is started, and then each collection is sent to the startMigrationThread thread.

One more Paper "Moving Recursion Out of the RDBMS for Transactional Graph Workloads" By Christine F. Reilly and Matthew Clark in 2020. In that Paper they proposed a method that blends straightforward database searches with parallel programming, and they contrast our method with the notoriously inefficient recursive SQL procedures. Our early tests and findings offer direction for the project's future goals, where we will improve the parallel programming strategy and experiment design to more effectively contrast these two techniques. They used the Hadoop MapReduce cluster to run the data generating programme, and they then downloaded the output from the Hadoop file system (HDFS) to a workstation's local disc. The data generator may be set up to produce data sets of various scales[1]. For the initial tests covered in this paper, we created a 1GB data set. For subsequent studies, we'll make use of bigger data sets.

In paper [3] and [4] The Authors To address the drawbacks of separate systems, a hybrid model design was created by combining the two models into a single system. The hybrid system combines relational databases and graph databases by weighing their benefits and drawbacks. A synopsis of earlier studies on the hybrid database idea is presented in that paper. Additionally, a technique is proposed that makes advantage of the restrictions and format of the source create a graphical database by converting a relational database. The technique translates conjunctive SQL searches over the source into target graph traversal operations. To provide evidence of the solution's feasibility and the efficiency of query response throughout the desired database, the experimental outcomes are shown. Nodes are linked to tuples, while edges are mapped to foreign keys.
A graph database has been created using Java software to replace an example relational medical database with 24 tables.

4 Methodology:

Different migration algorithms from relational to non-relational databases have been proposed. Many of them have been put into practise, although they also have drawbacks. They often use foreign keys as edges and tuples as nodes in the RDBMS, however it is clear that this technique is costly in terms of quiring the data. Therefore, a superior method [8] a relational database into a graph database and mapped conjunctive queries launched against a relational

database to queries launched against a graph database, has been used in this work.

This aids in storing the most often accessible entities close at hand, greatly reducing fetching time. The detailed implementation process and algorithm are explained in the text that follows.

Consider:

- **R** is a Relational Database which is going to convert graph database.
- **i** is attributes set
- **r** set of relation on R
- **G** is a Graph wit Node N and Edge E.
- **V** is Vertex
- Edge (Vi,Vj) belongs to R

Algorithm for Creating a Graph database

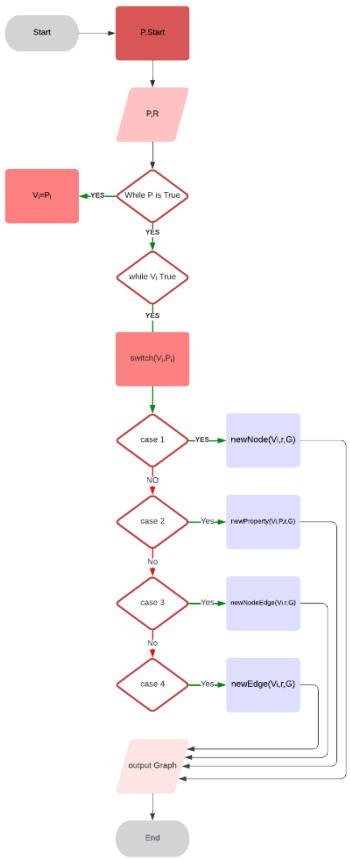
Input: R, P
Output: Graph

```
G ← <0,0>
P ← P.Initial
    while P do
      Vi ← Pi
            while Vi do
                  switch (condition (Pi,Vi))
                  case 1: newNode(Vi,r,G);
                  case 2: newProperty(Vi,P,r,G);
                  case 3: newNodeEdge(Vi,r,G);
                  case 4: newEdge(Vi,r,G);
                  end switch
            end while
      end while
            return Graph
```

Each column in a specific relational database, such as one for an airline record, is represented as a node in a graph. As a result, certain primary keys may relate to other primary keys to provide foreign key restrictions. We have certain nodes in this situation that are referred to as sources and only have outgoing edges. Sinks are nodes that have only incoming edges. Additionally, hubs are nodes with several incoming edges. These qualities are helpful when moving to a graph database based on conjunctive searches over relational databases. Full paths (P) are all feasible routes from a source node to a sink node.

Relational databases allow for 1:1, 1:N, or N:N (n2n) relationships between two tables. As indicated in the algorithm, each path P_i in P is travelled through individually, and then g(N,E) is formed at the conclusion. Let's suppose P symbolizes the traverse from source to sink and is analogous to A1 A2.... Ak. An attribute in r equates to Ai in P. The fundamental function getAll(r,V_i) is used to retrieve every value (v) connected to V_i in r. These values must be correctly connected to the graph's nodes. Active domain of V_i is the term used to describe the attribute values now being considered. V_i is marked as Visited once all of these active locations have been covered.

Flow Chart:



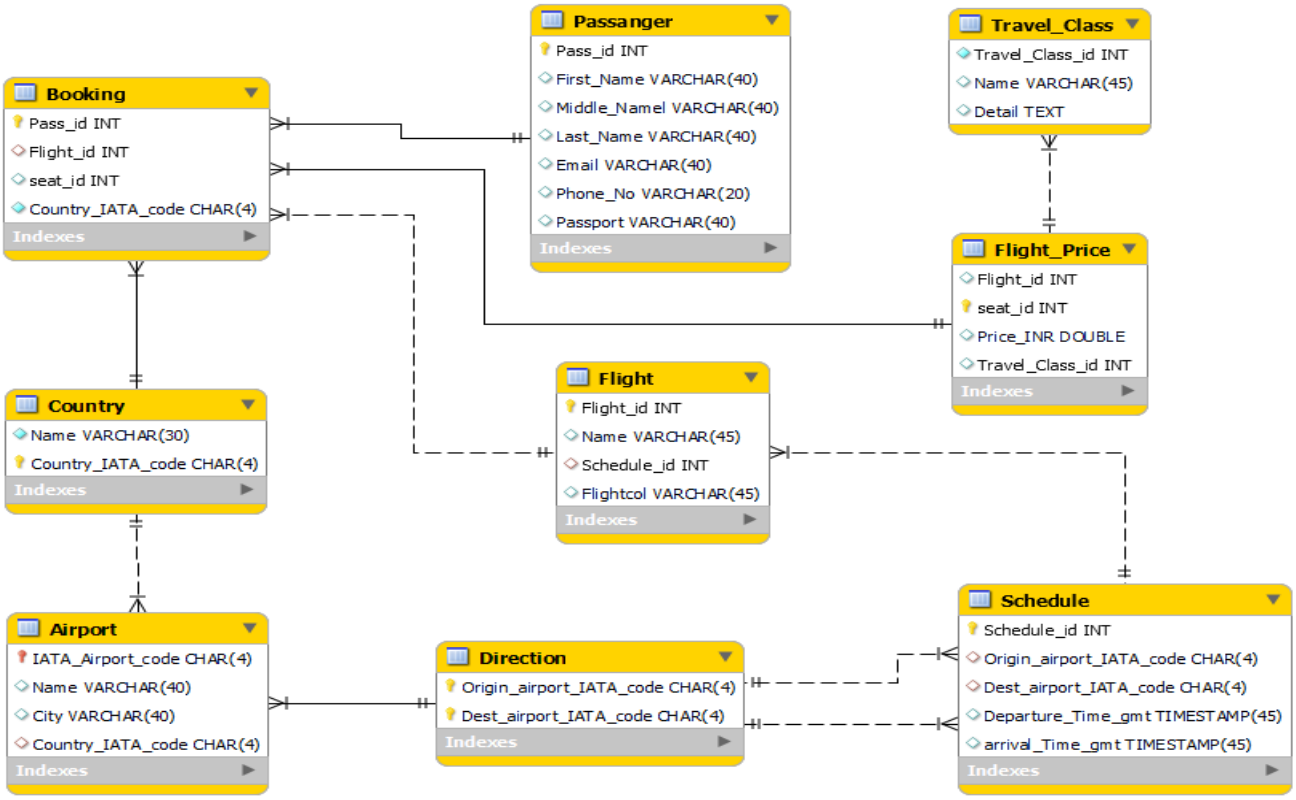


Figure 3: Relational Schema for airline Database

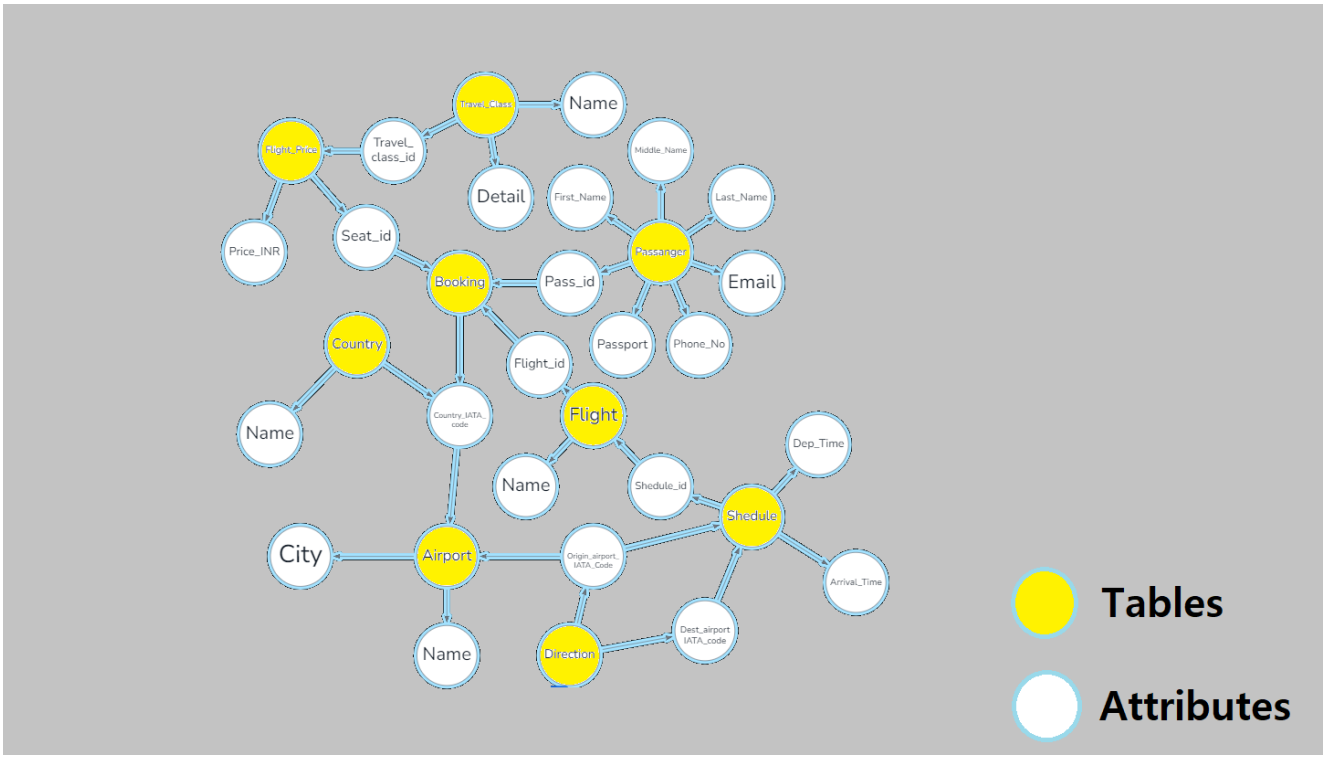


Figure 3: Resultant graph database

Case 1: V_i is the source, and neither V_i nor V_i+1 are visited. The migration has started, and new nodes are being produced. newNode() is in charge of doing this. New nodes are added to g for each value of the data in V_i. The data attributes are updated in the properties of the newly generated nodes.

Case 2: V_i is unvisited and not a hub node. In this instance, visit to the nodes with the property label V_i-1 and, if V_i and V_i-1 are related, add that value to the reached nodes; otherwise, leave them as they are. The node property in this instance is created using newProperty() with an additional parameter as the travelled path P. Since V_i is the source and hasn't been visited; V_i+1 has a foreign key constraint from V_i ->V_i+1 is indicated by this. The major column values are already present in the node; hence it only needs to add additional column values. Add any values for each v when v belongs to getAll(r, Vi), as done by newProperty() in the Algorithm for each v.

Case 3: V_i is an unvisited hub/n2n node. Similar to CASE 2, navigate to the last attribute V_{i-1} and produce as many values as those that were retrieved for each one. Then, enter each value as an attribute in the associated nodes. Now renew the edge between the newly created nodes and name them with the label of V_i before connecting them to V_{i-1} using its label. The Algorithm's newNodeEdge() function does this.

Case 4: V_i is either (hub and visited and last attribute is not a source) or (n2n and unvisited). It appears at the end of a schema route of attributes. Extract the V associated with V_i in the tuple if the relationship between V_i and V_{i-1} is similar as they are both in the identical tuple (i.e., there is no foreign key). If a foreign key exists instead, V fetches value V_1 from V_{i-1}. The nodes matching to every v in V are then retrieved, and if an identical value is found, the nodes are linked with the label that is belonging to V_i. Algorithm's newEdge() function is in charge of doing all of these tasks.

5. Result and Discussion:

An Airline database used to implement the method of migrating the Relational database to graph database by using 9 tables and Their attribute. In the Figure 2 Relational Schema of Airline database is shown. Then, Neo4j is used to access this database, and the Neo4j browser instance retrieves all the nodes that have migrated. Figure 3 depicts the final graph database that was created.

Additionally, the migration procedure offers a chance to review your data model and refine it for graph-based storage and retrieval. Compared to the relational approach, it may result in better performance, scalability, and easier maintenance.

Data management and query performance might be significantly impacted by switching relational database to graph database. Although relational databases, which are based on the conventional tabular layout, excel in managing organized data, they can be ineffective when handling relationships between data points that are complicated. On the other hand, graph databases are a better option for some use cases since they are made expressly to manage complex relationships.

An improved and more potent data management system that can handle complex and interrelated data with enhanced flexibility and query capabilities is the overall outcome of moving from a relational database to a graph database.

5 Conclusion:

The job of migrating data from an RDBMS to a graph database is highly difficult since the data is always changing and growing rapidly in size. Both data migration methods are used in the Neo4j paradigm that is described for data migration from RDBMS to Graph[namdio]. This model has four components for migration those are NewNode, NewProperty, NewNodeEdge and NewEdge.This methodology effectively migrates as well as streams of updated data in real time in Graph. Experiments further demonstrate that the suggested

model operates more quickly than other models already in use and migrates real data [13].

A method for building a conversion database system between a client and two different user's databases is thereby suggested. To shorten response times by allowing the databases to collaborate, this system would have the capacity to break down and run queries on numerous databases. in ascending order to provide the results, the aforementioned idea will be attempted to be executed in the next phase. This paper carefully examines a number of hybrid database approaches [3].

The usefulness and usability of the graphbase query system also require a participant-based evaluation. However, features like Facebook's GraphSearch imply increasing interest in providing users with tools to make complex inquiries about their social networks. Possible user groups networks include academic librarians, intelligence analysts, and scholars in the humanities. Thus, participants from a range of fields with diverse degrees of knowledge should be included in an evaluation of the method. When applicable, it is important to evaluate the usability of the hypergraph query strategy as well as its value as an analytical tool. The former may be guided by observing users perform numerous query definition tasks while developing simple and complex query graphs. By providing participants in a number of areas with access to visual analytic tools that incorporate the hypergraph query system and conducting follow-up interviews with the users after a period of tool use in their study, the latter may be evaluated longitudinally [7].

In conclusion, Numerous advantages and chances for improved data administration and analysis come with switching from a relational database to a graph database. For some use situations, graph databases offer a more effective and adaptable option due to their superior ability to handle intricate and related data structures. When working with heavily related data, graph databases offer enhanced query performance. There is less need for expensive joins and challenging SQL queries as a result of the ability to traverse relationships directly in the database. In situations where connections are important, including social networks, recommendation systems, and fraud detection, this can result in considerable performance benefits [1].

6 References:

[1] Christine F. Reilly and Matthew Clark. 2020. Moving Recursion Out of the RDBMS for Transactional Graph Workloads. *https://ieeexplore.ieee.org/document/9298122*

[2] Basant Namdeo and Ugrasen Suman. 2021. A Model for Relational to NoSQL database Migration: Snapshot-Live Stream Db Migration Model. *https://ieeexplore.ieee.org/document/9441829*

[3] H.R.Vyawahare, Dr.P.P.Karde and Dr.V.M.Thakare 2018. A Hybrid Database Approach Using Graph and Relational Database. *https://ieeexplore.ieee.org/document/8509057*

[4] Manpreet Singh and Karamjit Kaur. 2015. SQL2Neo : Moving Health-care Data From Relational To Graph Databases. https://ieeexplore.ieee.org/document/7154801

[5 DENNIS DOSSO and GIANMARIA SILVELLO. 2020. Search Text to Retrieve Graphs: A Scalable RDF Keyword-Based Search System. https://ieeexplore.ieee.org/document/8960366

[6] Sofoklis Floratos, Yanfeng Zhang, Yuan Yuan, Rubao Lee and Xiaodong Zhang. 2018. SQLoop: High Performance Iterative Processing in Data Management https://ieeexplore.ieee.org/document/8416367

[7] Rachel Shadoan and Chris Weaver, Member, IEEE Computer Society. 2013. Visual Analysis of Higher-Order Conjunctive Relationships in

Multidimensional Data Using a Hypergraph Query System.
https://ieeexplore.ieee.org/document/6634154

[8] R. De Virgilio, A. Maccioni, and R. Torlone. 2013. Converting relational to graph databases. *https://dl.acm.org/doi/10.1145/2484425.2484426*

[9] K. Kaur and R. Rani, "Modeling and querying data in nosql databases," in Big Data, 2013 IEEE International Conference on. IEEE, 2013, *https://ieeexplore.ieee.org/document/6691765*

[10] K. K.-Y. Lee, W.-C. Tang, and K.-S. Choi, "Alternatives to relational database: Comparison of nosql and xml approaches for clinical data storage," Computer Methods and Programs in Biomedicine, vol. 110, no. 1, pp. 99–109, 2013. http://www.shealth.info/thomas/research/alternatives-to-relational-database/.

[11] K. Chodorow, MongoDB: the definitive guide. " O'Reilly Media, Inc.", 2013. *https://dl.acm.org/doi/10.5555/2544030*

[12] C. Lee and Y. Zheng, "Automatic SQL-to-NoSQL Schema Transformation over the MySQL and HBase Databases," in IEEE International Conference on Consumer Electronics - Taiwan, 2015, *https://www.researchgate.net/publication/321482775_Migration_from_relational_to_NoSQL_database*

[13] Z. G. Ives, Y. Zhang, S. Han, and N. Zheng, "Dataset relationship management," in Conference on Innovative Data Systems Research (CIDR), Asilomar, California, USA, 2019.. *https://www.cidrdb.org/cidr2019/papers/p55-ives-cidr19.pdf*