University of Life Science, Prague

Faculty of Economics and Management

Department of Information Engineering

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Diploma Thesis

Graph Databases

Mikulas Regeczy

Declaration

I proclaim that the thesis on the topic of Graph Databases has been written on my own and all resources that have been applied are stated in the thesis.

In Prague on ___________________________

Acknowledgement

I would like to thank to assoc. prof. Vojtech Merunka Ph.D. for his support during the time I was elaborating the thesis.

Graph Databases

Abstract

The main goal of the thesis is to investigate the development of the technology of graph databases. Descriptive and comparative analysis demonstrate the difference between MySQL and Neo4j database. It has been confirmed that graph databases are able to deal with complex relationships between data points much better. They enable to create entities to investigate relationships between data to make it easier to understand them because they are based on one table model.

What is more, graph databases seem to offer profoundly prime results. As a consequence, it is easier to gain the outcome by applying graph databases. In addition, provided that an actor wants to add a new relationship, it is not necessary to reconstruct the database one more time. Finally, as to the time execution of all processes, the retrieval time is faster with graph databases. Therefore, graph databases are more suitable for commercial reasons, such as development of the social network.

Keywords: Graph, Neo4j, MySQL

Grafové databáze

Abstrakt

Hlavním cílem práce je prozkoumat vývoj technologie grafových databází. Popisná a srovnávací analýza demonstruje rozdíl mezi MySQL a databází Neo4j. Bylo potvrzeno, že grafové databáze jsou schopny mnohem lépe řešit složité vztahy mezi datovými body. Umožňují vytvářet entity pro zkoumání vztahů mezi daty, aby bylo snazší jim porozumět, protože jsou založeny na jednom tabulkovém modelu.

A co víc, zdá se, že grafové databáze nabízejí naprosto prvotřídní výsledky. V důsledku toho je snazší získat výsledek použitím grafových databází. Navíc, pokud je potřeba vytvořit nový vztah, není nutné databázi rekonstruovat ještě jednou. A nakonec, pokud jde o časové provedení všech procesů, doba načítání je rychlejší s grafovými databázemi. Proto jsou grafové databáze vhodnější pro komerční účely, jako je rozvoj sociální sítě

Klíčová slova: Graph, Neo4j, MySQL

Contents

Contents of Figures

List of tables

List of Commands

List of Abbreviations

ACID	Atomicity, Consistency, Isolation, Durability
GDB	Graph Database
RDB	Relational Database
PK	Primary Key
FK	Foreign Key
WWW	World Wide Web
NoSQL	Not Only SQL
SQL	Structured Query Language

1 Introduction

Admittedly, storing and processing of the data in graphs has been gaining a large importance over several years. They are regarded as the base of the data structures and they are applied for modelling of results in copious types of relations, for example in biology, social and information systems as it is stated by several researchers, such as AR Mashaghi [1] . They correspond to manifold practical issues and social media of firms tend to utilize models of the graph to practical applications.

As a matter of fact, a graph consists of myriad of nodes and edges. Notes correspond to the actors and relationships between actors are represented by edges. Copious information is implicit in the graph. Several operations are possible to be processed by actors, such as transporting of the graph, product of the graph or graph minor (D.Brent, 2001). Above all, the data offer more opportunities for investigating of the social information. It is essential to store and query the graphs.

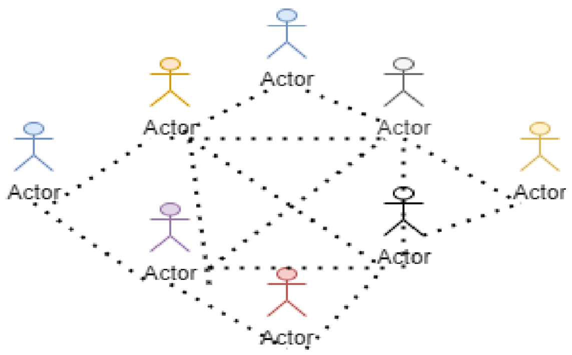


Figure 1: Social Graph2, Source: (J.Korhan, 2011),

[1] AR Mashaghi, Abolfazl Ramezanpour, and V Karimipour (2004), *Investigation of a protein complex network. The European Physical Journal B-Condensed Matter and Complex Systems*, 41(1):113–121 [online]. [cit. 2022-06-12]
[2] Jeff Korhan (2011), *What your business needs to know about social graphs* [online]. [cit. 2022-06-12]

In that case, it is not ideal to apply for the relational databases, which are supposed to store data in the way of tables. They are not suitable for inheriting graph structure. They are incapable for the social graph of the data in spite of the fact that they are ACID. Finally, Graph Databases are considered to be one of the best as for the storing and working with the data.

According to R. Angeles, who deals with graph databases in his research, states that Graph databases apply a graph structure for representing and storing the data. It is possible to use nodes, edges and properties (R.Angles, 2008). They provide a cost-saving result for storing data in comparison to the relational model. In addition, some processes that are profoundly inefficient in the relational databases are easily dealt with in the graph databases.

2 Objectives and Methodology

2.1 Objectives

The main goal of the diploma thesis is to investigate the application development of graph database technology. Moreover, the social network is researched as well.
Another aim of the thesis is to establish a small application made in the Neo4j and MySQL platforms. The final aim is to make a workable application in PHP, Ajax and MySQL.

2.2 Question of the Investigation

With the object of the diploma thesis, the diploma thesis evaluates the reply for the investigated question.
“Considering MySQL and Neo4j which of them is better at dealing with the data?”

2.3 Methodology

The thesis is composed of two parts. Aspects that correspond to the graph databases are involved in the theoretical part. The data has been extracted from sources that are scientifically oriented, for example investigating reports, books and web pages that refer to graph databases.

Social network is researched in the theoretical part as well. MySQL relational database and Neo4j graph database are demonstrated in the practical part. The php application is elaborated at the end of the thesis.

2.4. Limitation of the Investigation

The research of the diploma thesis is limited by several aspects. Subsequently, the diploma thesis does not have a proper sum of data for receiving one of the best solutions.

3 Literature Review

As a matter of fact, relation database is introduced in Section 3.1 that is one of the main points of this research. Practical knowledge about the relational databases is provided in practical part. Several essential technologies and terminologies regarding to the relational databases are demonstrated in this part as well. In addition, ACID (Atomicity, Consistency, Isolation and Durability) that make sure that reliability of the database transaction is guaranteed. Primary Key, Foreign Key and Database Normalization are included into the database transaction. Besides, Primary Key and Foreign Key are the main concepts for building relationships in relational databases. As for the Database Normalization, it is a process for decomposing the data to minimize the redundancy of the data.

Section 3.2 is devoted to the graph databases. Consequently, it is another main topic of the thesis. NoSQL databases are revealed in subsection 3.2.1. and another three graph models are demonstrated in following subsections.

Furthermore, the benchmark of the database is intended for evaluating the performance of the database. Benchmarking databases are revealed in Section 3.3.

3.1 Relational Database

Even so, it has been a huge challenge to discover how to store and safely access data. [3] Edgar F Codd introduced the relational data in 1970. The entire database market was devoted to the relational model until the NoSQL technologies. Admittedly, there are a great number of vendors dealing with the relational database management systems. Their products differ from each other in price and what they are capable to do.

According to E Codd, who writes about relational databases in his book, declares that data is organized into one or more tables of rows and columns[4] . Every row is uniquely identified. Information is stored by using tables. As a result, the software storing is enabled and accessed. The modified data is saved in the server side. A relational database is demonstrated in Table 1.

Table 1: Relational Database, Source: own work

Name	Surname	Gender	Age
Mik	Reg	Male	45
Martin	Reg	Male	37
Amos	Reg	Male	1 week
Anna	Reg	Female	2

There are four relational attributes: “Name”, “Surname”, “Gender” and “Age”. Every attribute corresponds to value. Every row of data, for example “Mik”, “Reg” are for a tuple. Cardinality represents several tuples, and a degree represents a number of attributes. Therefore, 4 corresponds to the cardinality as well as to the degree, which is demonstrated in Table 1. More sizeable relational terminologies are represented [5] in Table 2.

[3] Edgar F Codd (1970). *A relational model of data for large, shared data banks: Communications of the ACM* [online]. [cit. 2022-06-12]
[4] Edgar F Codd (1972). *The Relational Model for Database Management: Version 2. Addison-Wesley Longman* [online]. Publishing Co., [cit. 2022-06-12].
[5] Edgar F Codd (1990). *Further normalization of the data base relational model: Data Base Systems, pages 33–64* [online]. Publishing Co., [cit. 2022-06-12]

Table 2: Terminology of the Relational Model, Source: own work

Terminology	Meaning
Relation	Table in a database
Domain	Type of column in a table
Attribute	Column of a table
Attribute value	Value of the column
Tuple	Table' s row
Entity	Table' s name
Cardinality	Number of rows in a table
Degree	Number of rows in a table

What is more, according to Codd (E. Codd, 1990) properties of the relations are represented:
- A tuple in a relation is represented in a row of the table
- Duplicity is to be avoided by different rows
- The order of rows ought to be insignificant
- The order of columns ought to be insignificant
- All table values ought to be atomic

3.1.1 ACID

A transaction corresponds to a unique logical operation in the database systems. Namely, putting data into a database. A reliable transaction, a set of properties, for example Atomicity, Consistency, Isolation and Durability were implemented in 1983[6].

[6] Theo Haerder and Andreas Reuter (1983). *Principles of transaction-oriented database recovery: ACM Com puting Surveys* (CSUR) [online]. [cit. 2022-06-12

Atomicity

Every transaction ought to be atomic (all or nothing) (R.Elmasri, 2008). In that case, a transaction ought to be considered as a complete unit. Provided that one part of the transaction ceases to work properly, the whole transaction ought to achieve no changes in the database.

Consistency

According to A.T.Clements, who investigates the topic of transactional consistency in his book, confirms that the consistency makes sure that a transaction ought to transfer the database from one stage to another. In the case the transaction is fully completed[7]. Even so, the transaction does not have to be correct, but it must be consistent.

Isolation

Isolation property ascertains that every running transaction ought to be independent during their own transaction unless another one is successfully completed (T.Haeder, 1983). As it is stated by T. Haeder in his research, consequences of not finished transactions ought to be not seen for other transactions.

Durability

For the reason that a transaction is committed, whatever the case the changes ought to stay the same (R.Elmasri, 2008). The property makes sure that execution results are permanent.

The reliability of the transaction is guaranteed by the relational databases trough ACID. Owing to the development of web technologies, the data storage techniques have crucially developed. As for scalability and availability, it has been critically important namely in the environment of distribution. As a consequence of huge data, it is commonly challenging for RDMSs to store and process it. It caused to develop the NoSQL.

[7] Dan RK Ports, Austin T Clements, Irene Zhang, Samuel Madden, and Barbara Liskov (2010). *Transactional consistency and automatic management in an application data cache: In OSDI, volume 10, pages 1–15*, [online]. [cit. 2022-06-12]

3.1.2 Primary Key and Foreign Key

As a matter of fact, Primary Key is applied for identifying every unique record in every column or a set of columns in a table [8]. The tuple of the Private Key attributes ought to be unique for their identification and they cannot be repeated. A constraint is assigned to make sure that the uniqueness of the Primary Key is guaranteed and the reduction of the data in the database is reduced by the Primary Key.

As to Paul De Bra, who researches in relational databases, states that there must be cohesion between the Primary Key and the Foreign Key in a relational database. A field that represents a reference to the Primary Key is the Foreign Key [9]. In fact, there can be several Foreign Keys pointing to several Primary Keys in a database.

However, a data schema of a relational database is demonstrated in Figure 2. There are five tables: students, studentsClass, teachers, classes and departments and a Primary Key. It is considered as id attribute, which is in every table. StudentID, classID and departmentID are in table of studentsClass. Foreign Keys of the attributes of teachersID and departmentID are in table of class. As a final point, Foreign Keys of classID and teachersID are in table of departments. They correspond to id of class and teachers as it is depicted in Figure 2.

[8] Jan Paredaens, Paul De Bra, Marc Gyssens, and Dirk Van Gucht (2012), *The structure of the relational database model, volume 17. Springer Science & Business Media* [online]. [cit. 2022-06-12].

[9] Jan Paredaens, Paul De Bra, Marc Gyssens, and Dirk Van Gucht (2012), *The structure of the relational database model, volume 17. Springer Science & Business Media* [online]. [cit. 2022-06-12].

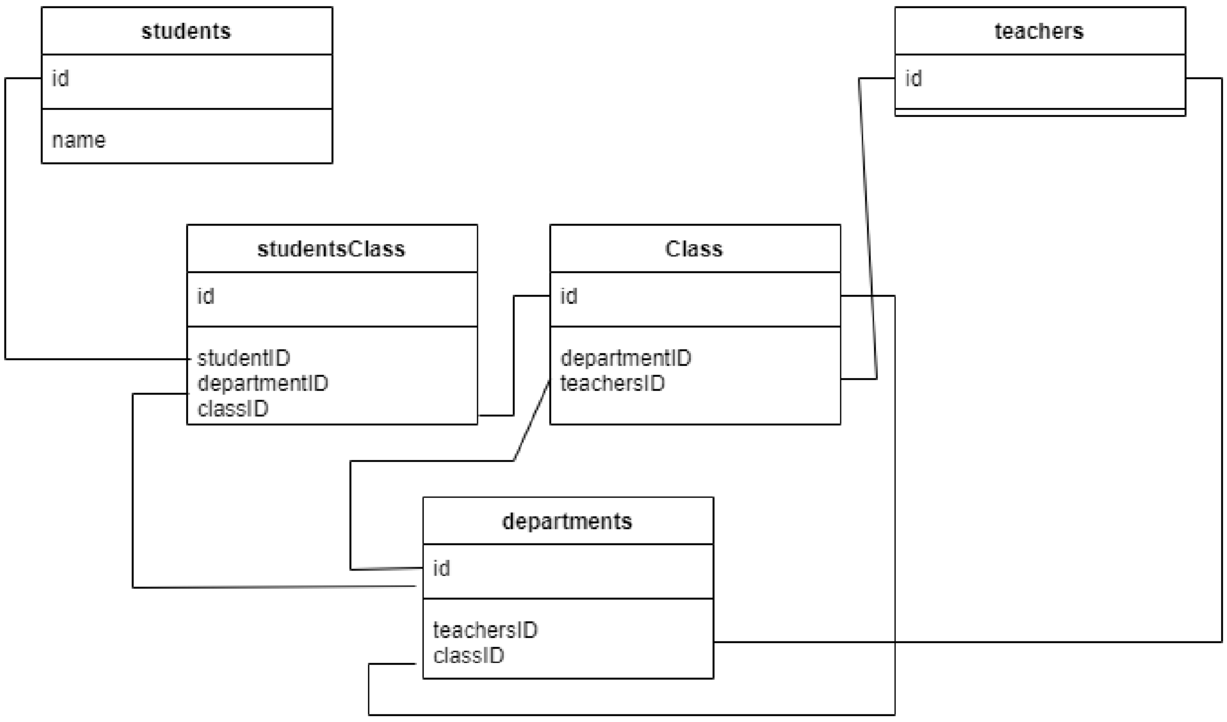


Figure 2 Primary Key and Foreign Key [10], Source: (P. Zaich,2012)

3.1.3 Normalisation of the Database

Begin with the topic, data redundancy can occur in a database system when a field is performed several times. In the case that the data occur several times in a table as it is demonstrated in Figure 3 then the redundancy of data is caused. The data redundancy and the data inconsistency are raised. According to E. Codd, who investigates the data redundancy in his work, states that more sizable storage is required. A concept of the database of normalisation is to be introduced to minimise the data of redundancy (E.Codd, 1970). A complete domain is separated into sub-domain that makes it fully independent. Hence, every sub-domain is connected with each other for the relation between the Foreign Key and the Primary Key (E.Codd, 1972).

[10] Paul Zaich (2012), *Relational databases and deconstructing the learning process* [online]. [cit. 2022-06-12].

Table 3 Data Redundancy, Source: own work

ID	Name	Surname	CourseID	TeacherID
1	Mik	Reg	CMP101 CMP102 ECON203 STAT203	T1001 T2003 T3231 T4013
2	Martin	Reg	CMP101 HIST204 ECON403	T1001 T2243 T3724
3	Anna	Reg	CMP204 CHIN101	T1341 T1291
4	Amos	Reg	CPM101 CPM102	T1001 T2003

Three types of normalization were introduced by Codd (E.Codd, 1972). Namely, First Normal Form, Second Normal form, and Third Normal Form. These three forms fulfil the requirements of the relational datasets as one of the best. Other normal forms, for example EKNF, which stands for Elementary Key. Normal Form or BCNF, which stands for Boyce Codd Normal Form, are applied commonly for academic reasons. Overall, an ideal structure of database normalization is demonstrated in Figure 3.

First Normal Form

Edgar Codd revealed First Normal Form in 1971 (E. Codd, 1970). Provided that a relation exists in the First Normal Form, every attribute of the relation ought to only possess atomic values. The attribute ought to have a single value. Even more, the repeating values are avoided in a table. The data that are related to it are stored in a different table.

Second Normal Form

Edgar Codd introduced the second step of the Normal Form in 1971 (E.Codd, 1970), though. A non-prime attribute ought to be fully dependent on the complete primary key.

Third Normal Form

Admittedly, the third Normal Form is to reduce the duplicity of data. The referential integrity is applied (E.Codd, 1970). Some requirements must be fulfilled, for example that there is a relation in the table and that there is dependency on the primary key. Finally, there is no dependency on other non-prime attributes.

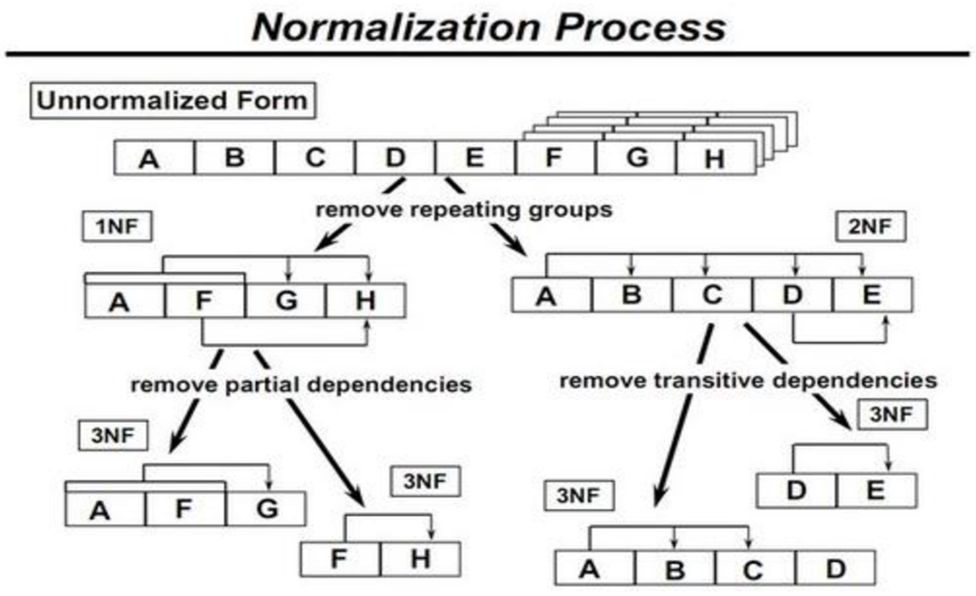


Figure 3 Database Normalization Process, Source: (E.Codd, 1970)

Consequently, the relational model was improved by the normalisation of the database, and it successfully effected the relational database. The data is decomposed into smaller relations. It establishes one of the best relationships between them. Also, according to G. L. Sanders, who investigates denormalization strategies in his book, affirms that it tends to make the model more informative for users [11]. By the way of contrast, it is applied bountifully in RDMSs, but it is not applicable for every situation. For the reason that the relational databases are scalable improvement

[11] Seung Kyoon Shin and G Lawrence Sanders (2006*), Denormalization strategies for data retrieval from data warehouses: Decision Support Systems, 42* [online]. [cit. 2022-06-12].

of readability needs adding some copies of data. In fact, this process is called denormalization and the result of queries can be improved in that case.

3.2 Graph Database

3.2.1 NoSQL

Because the cloud computing was developed, a new type of database was applied[12]. Consequently, NoSQL corresponds to non-SQL and is exerted for a different storage mechanism. As it is stated by D. McCreary, it retrieves data differently than the tabular relations. In other words, they are utilised by the relational databases [13]. There are four main NoSQL databases, for example key-value stores, column family stores, document stores and graph databases. NoSQL is illustrated graphically in Figure 3. In addition, the rest of NoSQL are depicted below:

Key-value Stores

Admittedly, data is stored as hash table, which is in a schemeless way[14]. Two fields are included in the hash table with a key with its value. So that the data is viewed as a set of Key-Value pairs in the model. More precisely, the data is to be quickly achieved in the database by investigating the key and the key is regarded as the unique identification of the record. The structure of the storage model is profoundly simple. It offers prime availability and scalability. As a final point, since it is easily applicable in applications, it is critically important for distributed environment.

Column Family Stores

As a first point, the data is stored in columns instead of rows. That is how it is performed in the relational databases and, the data is stored in a Key-Value pair. There is a two-dimensional key with a column key and a row key [15]. Even more, three elements are included into a column.

[12] Jing Han, E Haihong, Guan Le, and Jian Du (2011), *Survey on nosql database. In Pervasive computing and applications (ICPCA): Decision Support Systems, 42* [online]. [cit. 2022-06-12].
[13] Dan McCreary and Ann Kelly (2014), *Making sense of nosql: Shelter Island: Manning, pages 19–20* [online]. [cit. 2022-06-12].
[14] Carlo Beltrame (2013), *Key-value stores: Algorithms for Database Systems, pages 1–12* [online]. [cit. 2022-06-12].
[15] Robin Hecht and Stefan Jablonski (2011), *Nosql evaluation. In International Conference on Cloud and Service: Computing, pages 336–41* [online]. [cit. 2022-06-12].

Specifically, a unique name that refers to the column. Value that is the column’ s content and timestamp that determines the validity of the content. The database is demonstrated as a sizeable table, which is a result of a set of rows that possesses several columns. The data is stored as a single entity because of a row-by-row attitude and the information that relates to the attribute due to a column-by-column attitude [16]. Therefore, this model of the database is prime for a large-scale data of the distribution.

Document Stores

For the purpose of documents, applications are allowed to store data in these forms. There is independence between documents in the same way as it is for the rows in a relational database. As for the Key-Value Stores, the Document Databases profit from implementing of the hash table. All data is stored together, and every instance of the data can differ from each other (R. Cattell, 2011). Document Stores provide a prime performance for sizeable data sets due to the document that is semi-structure. Finally, they reveal a large flexibility with data.

[16] ABM Moniruzzaman and Syed Akhter Hossain (2013). *Nosql database: New era of databases for big data analytics-classification, characteristics and comparison*. [online]. [cit. 2022-06-12]

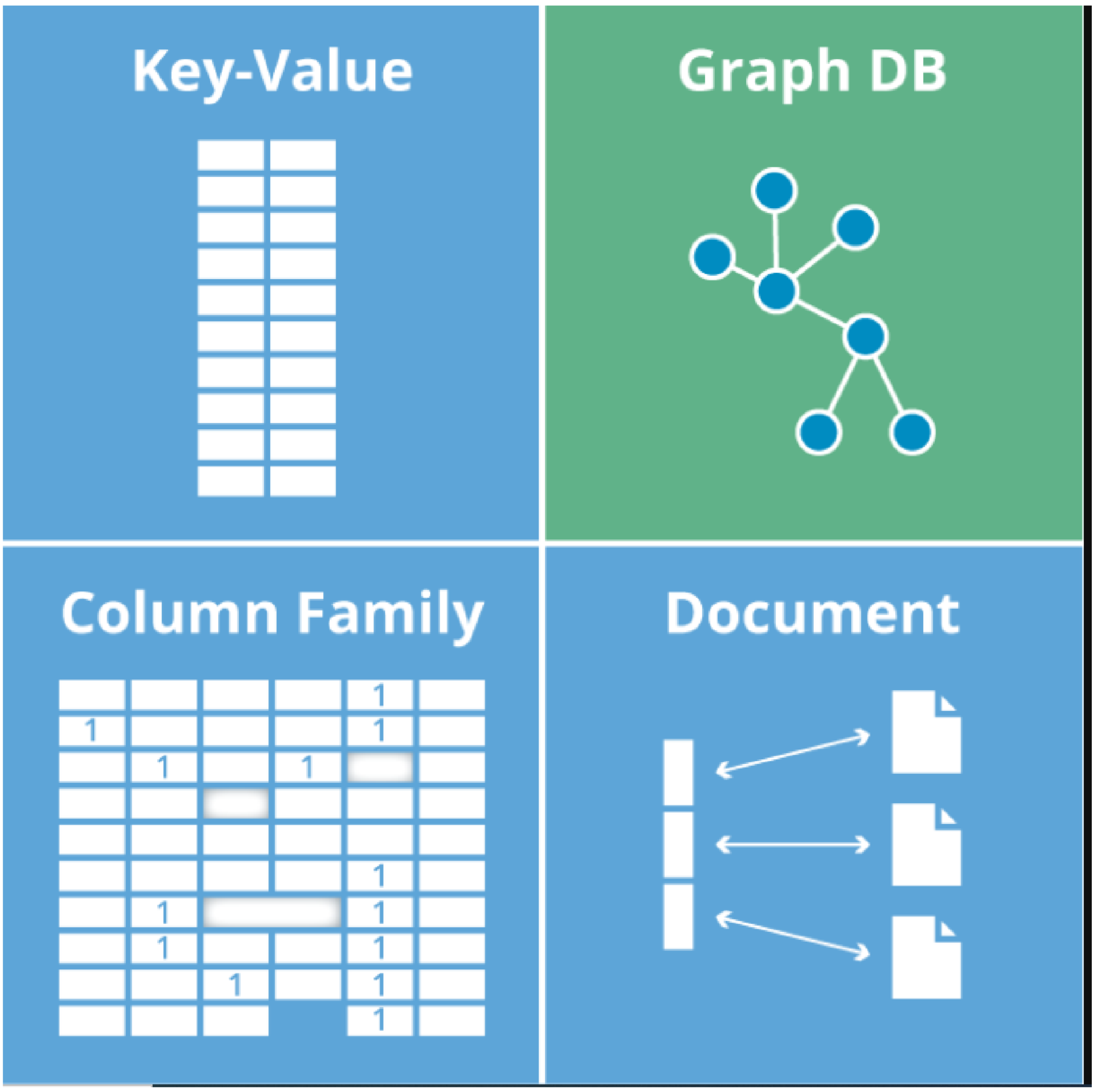


Figure 4 NoSQL Database[17], Source: (M.Sasaki, 2016)

By the way of contrast, the Graph database is another type of NoSQL database. It involves graphs that are composed of notes and edges. They store and manage data. On the other hand, a relational

[17] Bryce Merkl Sasaki and Aspiring Graphista (2016), *Graph databases for beginners: Why we need nosql databases* [online]. [cit. 2022-06-12].

database stores data in tables[18]. In fact, the data is stored as nodes' properties and edges in the database of the graph. As a final point, a large accessibility and scalability are provided by the graphs' structure.

As a matter of fact, a common graph database is demonstrated in Figure 4. There are three nodes in the graph. Every node possesses their property, for instance Id, Name and Age. All nodes are connected with each other. Their links represent a relationship between them. Every edge possesses their property. Id, Label and Timestamps correspond to their properties. By way of contrast, relationships can be illustrated directly as a result that the edges are directional.

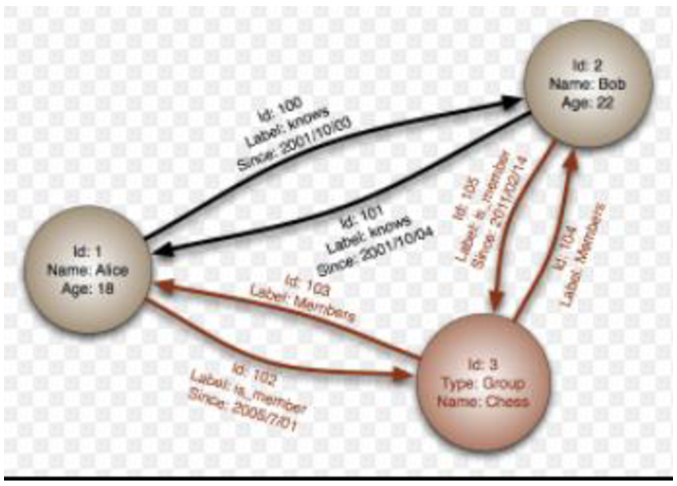


Figure 5 Graph Database, Source: (Renzo Angles,2008)

However, data is stored by graphs that are crucially interconnected into the structure of data. It is profoundly applicable for the networking of the social websites, for instance Facebook and Twitter. Social actors are easily represented as the nodes and edges. They correspond to the relationship between users. Finally, social information about users is represented by properties.

[18] Renzo Angles and Claudio Gutierrez (2008), *Survey of graph database models. ACM Computing Surveys* [online]. [cit. 2022-06-12].

As for the data of modelling in the graph database domain, there has not been a great number of studies regarding to it. It is veritably difficult to offer some rules corresponding to practical applications. Whereas some specific rules are provided by Weber[19].

- Entities are to be represented by nodes.
- Relationships are to correspond to edges to link the entities. In that case, they ought to establish a condition of semantics for every entity.
- Entity of attributes ought to correspond to properties of nodes.
- Strength, weight, and quality of the relationship ought to be defined by properties of the edge.
- Graphs' property
- Hypergraphs
- Resource Description Framework

3.2.2 Property Graphs

However, a graph model is manifold times applied in one of the best systems of the graph database. A graph of property consists of these elements (I. Robinson, 2005):

A pair of vertices:

A) Every vertex uniquely possesses their identifier
B) Every vertex possesses several incoming edges
C) Every vertex possesses several outgoing edges
D) Every vertex possesses several properties that are connected to it. What is more, a map defines properties from key to value.

A pair of edges:

A) Every edge uniquely possesses their identifier
B) Every edge possesses their incoming head vertex
C) Every edge possesses their outgoing tail vertex
D) Every edge possesses several properties whose map defines them from key to value

[19] Ian Robinson, Jim Webber, and Emil Eifrem (2005), *Graph Databases: New Opportunities for Connected Data." O'Reilly Media, Inc."* [online]. [cit. 2022-06-12].

E) Every edge possesses a label to demonstrate the association between the incoming vertex and outgoing vertex

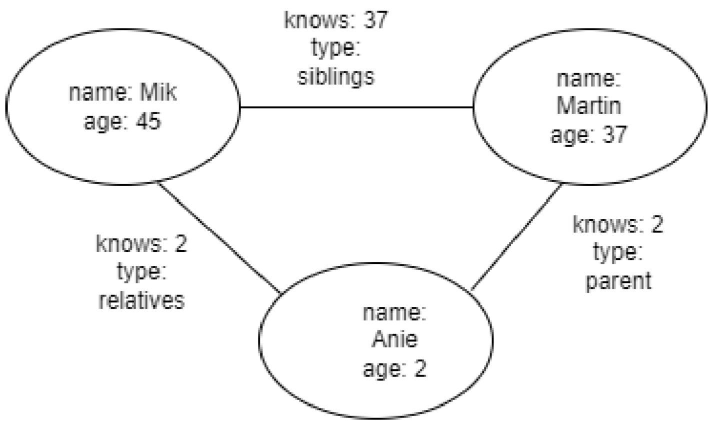


Figure 6 Property Graph, Source: own work

Above all, a property graph is demonstrated in Figure 6. It consists of three actors: Mik, Martin and Anie. They are regarded as nodes in the graph. Further information about their name and age are considered as nodes of property. Edges link them with each other and the relationship between them is viewed as the property of edges. Finally, the defined relationship between them is explained better by hypergraph.

3.2.3 Hypergraphs

As for the graphs, data is stored and applied by databases of graphs that are composed of nodes and edges. According to B. Iordanov, who researches hypergraphs in his book, affirms that the property of graph is considered as one-to-one relationship. Multiple nodes can be linked in the model of the hypergraph[20]. Even more, many-to-many relationships are to be simply stored and processed, which is genuinely difficult to deal with it in a graph of the property.

[20] Borislav Iordanov, (2010*) Hypergraph dB: a generalized graph database. In International Conference on Web Age Information Management, pages 25–36. Springer* [online]. [cit. 2022-06-12].

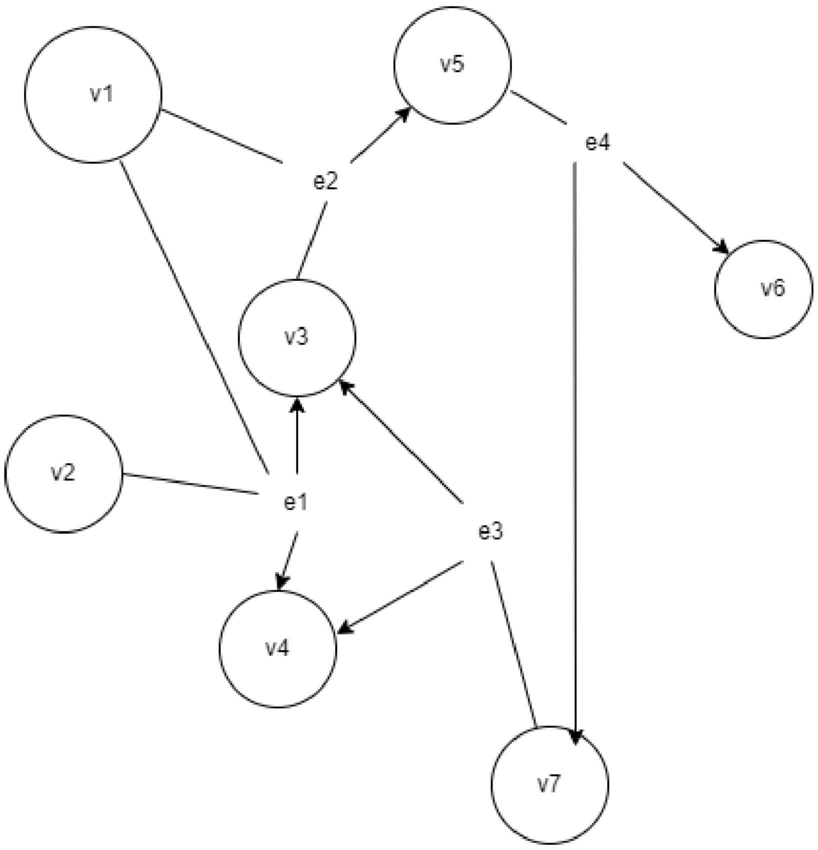


Figure 7 Hypergraph, Source: own work

Besides, a hypergraph is revealed in Figure 7. The graph contains six vertexes. That is: v1, v2, v3, v4, v5 and v6. There are 4 hyper edges, such as e1, e2, e3 and e4 that connect the nodes. Therefore, for example e1 links v1, v2, v3 and v4. E2 connects v1, v3 and v5 as well and e3 links v3, v4 and v7. Finally, v5, v6 and v7 are connected by e4.

While the model is profoundly simple, the relationship must be depicted for better understanding. The cost of storage is increased provided that hyper edges, which are multidimensional, ought to be transferred into a graph of the property.

3.2.4 RDF Triples

However, it is applied for data interchange on the Web[21]. According to G. Antoniou, it manages several syntax notations and formats of data serialization. It is applied for conceptual description and modelling of information that are practised in web resources. Finally, according to S. Powers, who deals with the same issues in his work, declares that the knowledge decays into less sizeable pieces that possess rules of semantics (S.Powers, 2009),(Jeff Z Pan, 2009) , (Graham Klyne, 2004).

- A simple unit of information that is considered as an RDF triple is illustrated in Figure 8.
- It represents expressions known as subject-predicate-object. In addition, it is to refer as resource identifier.
- Finally, the predicate and the triple's subject ought to be applied in URIs for removing unnecessary ambiguity in the information.

[21] Shelley Powers (2009), *Practical RDF. O'Reilly* [online]. [cit. 2022-06-12].

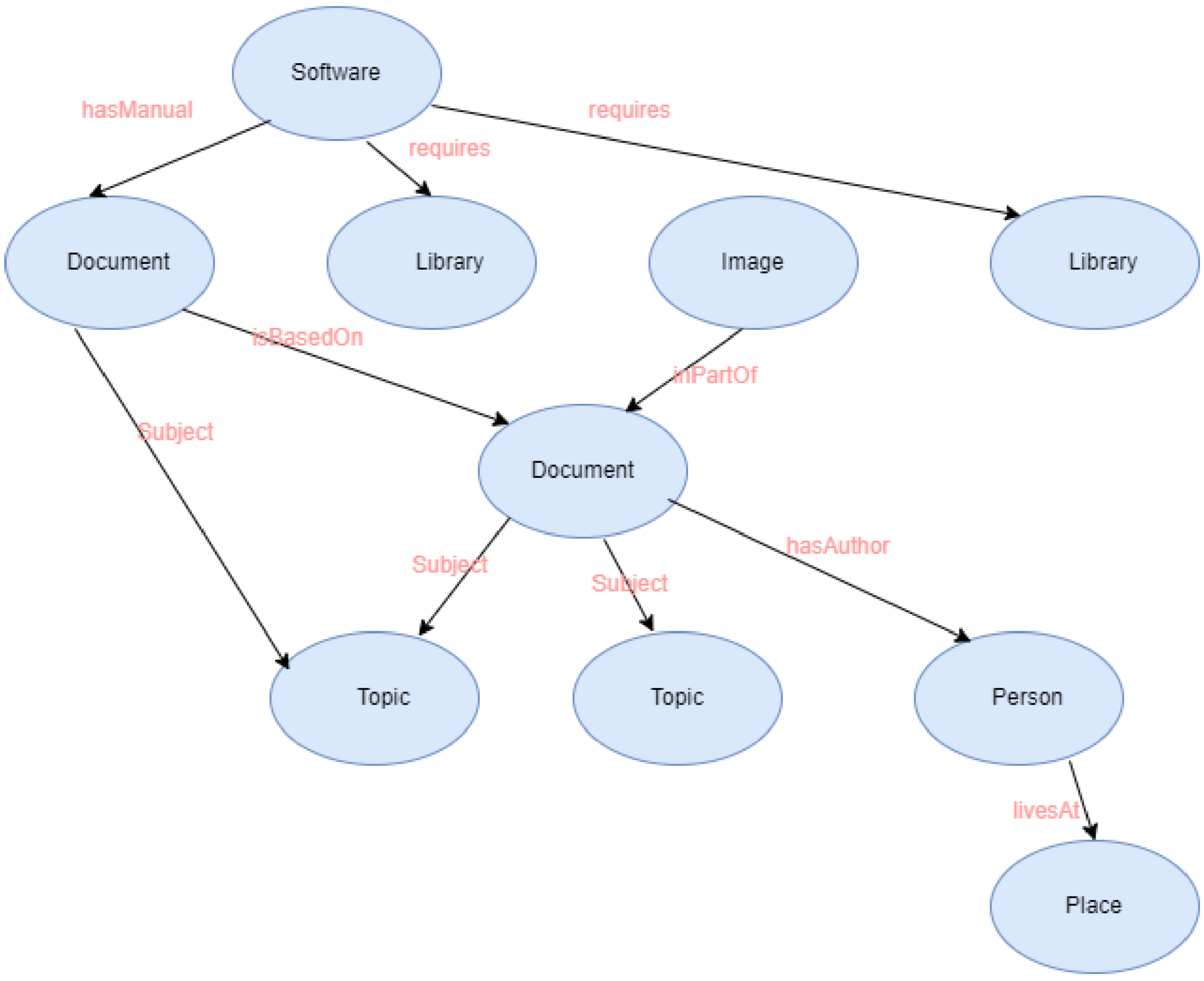


Figure 8 RDF Triple, Source: own work

The graph of the RDF, which is demonstrated in Figure 8, is converted into the subject-predicate-object. The result is illustrated in Table 4, though.

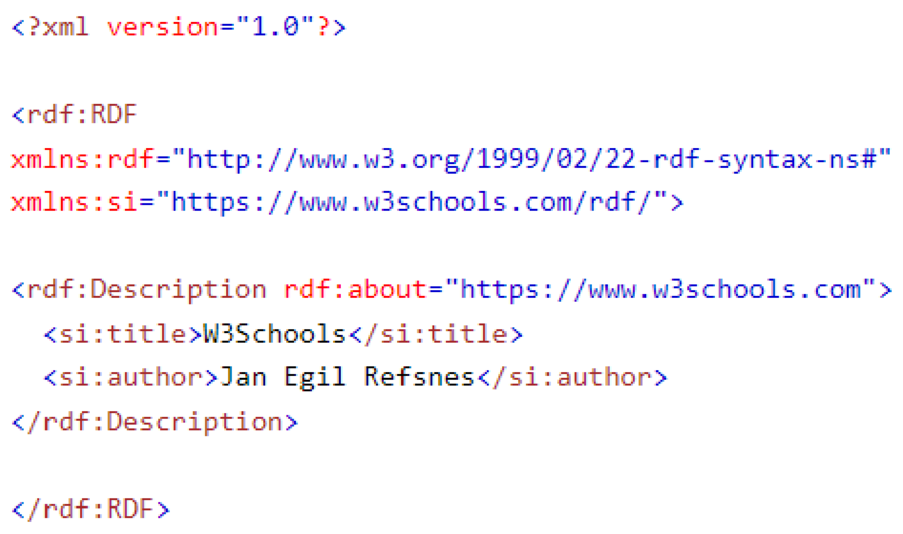
Table 4 RDF subject-predicate-object, Source: own work

Subject	Predicate	Object
Software	hasManual	Document
Software	requires	Library
Document	isBasedOn	Document
Document	Subject	Topic
Image	inPartOf	Document
Document	hasAuthor	Person
Person	livesAt	Place

Admittedly, RDF is considered as one of the best Semantic Web technologies. It is commonly applied in the Semantic Web as a graph database. According to T. Berners-Lee, who investigates Semantic Web in his research, declares that there is not manifold information about the Semantic Web to identify whether two nodes are same or not. For solving the issue, notion of the URI is utilised by RDF. The URI is crucial for stating identity on the WWW[22]. It is easily identified whether two people are referring to the same condition. Furthermore, the property of URI enables common organisations to determine specific terms, for example xmlns that refers to the definition for XML schema (S.Powers, 2009),(G.Antoniou, 2004).

Even so, the infrastructure of the web is used by the Web to demonstrate how to correspond to a specific entity. The namespace infrastructure is taken as an advantage by the RDF standards. It identifies a small number of identifiers in the namespace, which are called rdf. Above all, it indicates when another identifier is ought to be applied as a predicate (G.Antoniou, 2008),(J.Pan, 2009). A typical RDF file that is written by XML is illustrated in Figure 9.

[22] Tim Berners-Lee, James Hendler, Ora Lassila, et al (2001*), The semantic web. Scientific American, 284(5):28– 37* [online]. [cit. 2022-06-12].



```xml
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="https://www.w3schools.com/rdf/">

<rdf:Description rdf:about="https://www.w3schools.com">
  <si:title>W3Schools</si:title>
  <si:author>Jan Egil Refsnes</si:author>
</rdf:Description>

</rdf:RDF>
```

Figure 9 RDF/ XML, Source: (G.Antoniou, 2008)

Benchmark of the Database

Databases of hardware, software and configuration are evaluated to realise how they perform, and it is regarded as one of the main issues. In fact, several standard tests are considered as a prime tool for determining the performance of the database. The benchmarking is viewed as one of the best for comparing the performance between applications, services, and processes to identify the best way to enhance the outcome. The benchmarking can be commonly separated into two categories[23], such as generic benchmarking and custom benchmarking.

Generic Benchmarking

Above all, the benchmarking is for measuring organizations' processes and performance. It is supposed to focus on a specific application domain, and it is used for upgrading performance and processes. Finally, it is for establishing of new standards.

[23] Hoe Jin Jeong and Sang Ho Lee. (2005) *An integrated database benchmark suite. In 2005 First International Conference on Semantics, Knowledge and Grid, pages 60–60. IEEE* [online]. [cit. 2022-06-12].

Custom Benchmarking

However, specific customers implement a custom benchmark for a particular application. The performance of the database is accurately measured based on the needs of a certain customer. It is only for a particular application. It is profoundly costly to implement and design the benchmark.

As a result of the development of techniques of graph databases, the performance of several graph databases has been compared. The comparison between scalable graphs databases has emerged, such as Neo4j[24] compared to Jena, Hypergraph DB and DEX. Every database's performance is tested by HPC scalable graph analysis. However, 1k, 32k and 1M nodes out of 8.4 million relations are tested, which is a small fraction of a sizeable social network.

Moreover, the way how to compare nowadays graph databases was suggested by Renzo Angles (C. Gutierrez, 2008). Nine data stores are compared as for their basic features. This comparison includes for example Allegro Graph, DEX, G-Store, Hypergraph DB or Neo4j. The information about storing the data and querying offers a general concept about the systems of database. Above all, it is difficult to realise the performance of the systems of graph database provided that it is dealt with graph data.

According to G. Amstrong, who investigates in benchmarking, reveals that the performance of graph databases was compared to the relational databases and Linkbench. It is focused on the traces from the databases. It stores graph data from Facebook [25]. For being able to construct Linkbench, the data and query workload are depicted by this work.

[24] Hoe Jin Jeong and Sang Ho Lee (2010), *Survey of graph database performance on the hpc scalable graph analysis benchmark. In International Conference on Web-Age Information Management, pages 37–48. Springer* [online]. [cit. 2022-06-12].

[25] Timothy G Armstrong, Vamsi Ponnekanti, Dhruba Borthakur, and Mark Callaghan (2013), Linkbench*: a database benchmark based on the facebook social graph. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pages 1185–1196. ACM* [online]. [cit. 2022-06-12].

3.4 MySQL

As a matter of fact, MySQL is regarded as one of the best open-source databases. It is owned by Oracle Corporation. It was developed by MySQL AB in 1995 and it is veritably appreciated by academic and scientific fields. In addition, it is supported for MySQL by users of the manufactures and the community.

Besides, MySQL is part of relational databases. The relational databases separate data into tiny blocks, which are given into one source of the data. Accessibility and flexibility are maximised by referencing the data with other tables.

By way of contrast, MySQL was originally established to deal with sizeable databases. The data of MySQL is stored in the table as rows. It is undoubtedly similar to an Excel spreadsheet. In fact, a particular category is stored in a column by every table and there is a predefined list of properties for each element. Provided that there is no data, Null is applied instead of it.

The relational model is controlled by a set of principles, which are called as ACID. It stands for atomicity, consistency, isolation and durability. Finally, MySQL is ideal robust and simple interaction with the database.

Caching and Buffering

- Begin with the topic, storage engines are utilised for storing, dealing with and obtaining data from the database of MySQL. MySQL supports the storage engine. Foreign key constraints for maintaining integrity and the recovery of the crash are reinforced as well.
- A buffer pool signifies a linked sheet of pages. It commonly saves obtained data in the upper part of the list by using algorithm (LRU). As a result of accessing manifold of threads, the suffocation occurs in the buffer pool. For preventing it, the maximum of 64 instances are allowed to be achieved.
- A query cache is applied by MySQL for storing SELECT statements and their outcomes. The outcome is obtained from the cache. In case that the announcement is queried one more time, the announcement is gained from the cache. Provided that the table is partially changed, all queries are to be taken away.

Availibility and Scalability

- Even so, the scalability corresponds to the balance between several servers. The charge of the server, hardware, and processing power are linked to it.
- The availability is regarded as the opportunity for reaching the database to ascertain that availability is not to be influenced by any failure of either hardware or software.
- Techniques for offering availability and scalability are applied differently by MySQL.

Replication

-Begin with the topic, the method for occurring of the replication is used by a technique called MySQL Replication. It can restart and stop itself at once. The master node replicates the data. However, it is not sure that the data is to be replicated by the master node to the other nodes called slaves.

3.5 Neo4j

Due to the data model, it is considered as a graph database. It tends to store nodes, which are linked with relationships. The user also states properties for both constructs.

Above all, it is an open source that was established in 2007 by Neo Technology[26]. Since it is embeddable, it is possible to join it with an application to apply it as another library. It consists of nodes, edges, and properties for storing data. Index free adjacency is provided. In that case, every node is considered as a pointer. Labelled property is supported by the graph model, and it has several characteristics. It is composed of nodes and relationships. Properties and nodes possess more than one label[27].

[26] The Neo4j (2022) Getting started Guide v4.0 [online]. [cit. 2022-06-12].

[27] W. Khan (2017), *Predictive Performance Comparison Analysis of Relational & NoSQL Graph Databases* [online]. [cit. 2022-06-12].

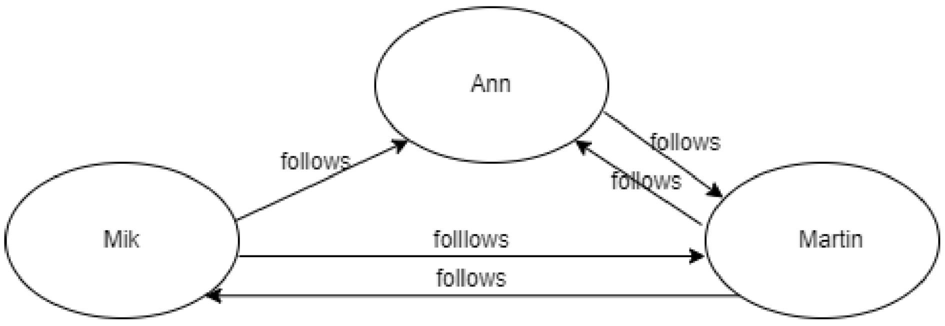


Figure 10 Property Graph Model, Source: own work

As for the Figure 10, it demonstrates three nodes, 5 relationships and one label. Mik, Ann and Martin are Users. They are nodes and the names are their properties. Follows is a relationship between them. Edges are bidirectional in Neo4j as it is revealed in Figure 10.

3.6 Social Network

Admittedly, a social network is regarded as a structure consisting of individuals, groups, and organisations. Every actor of the structure is linked with each other. They are connected by edges (Stanley Wasserman, 1994), (John Scott, 2011). The structure of social network is profoundly similar to the semantics of social network. In that case, the social network offers a prime way for investigating the complete society (P.J.Carrington, 2005). A graph can represent a social network as it is revealed in Figure 11. Links between nodes (Linton C Freeman, 2004), (S. Wasserman, 2005) correspond to the map of the social network. Finally, ties are between nodes either strong, weak, positive or negative.

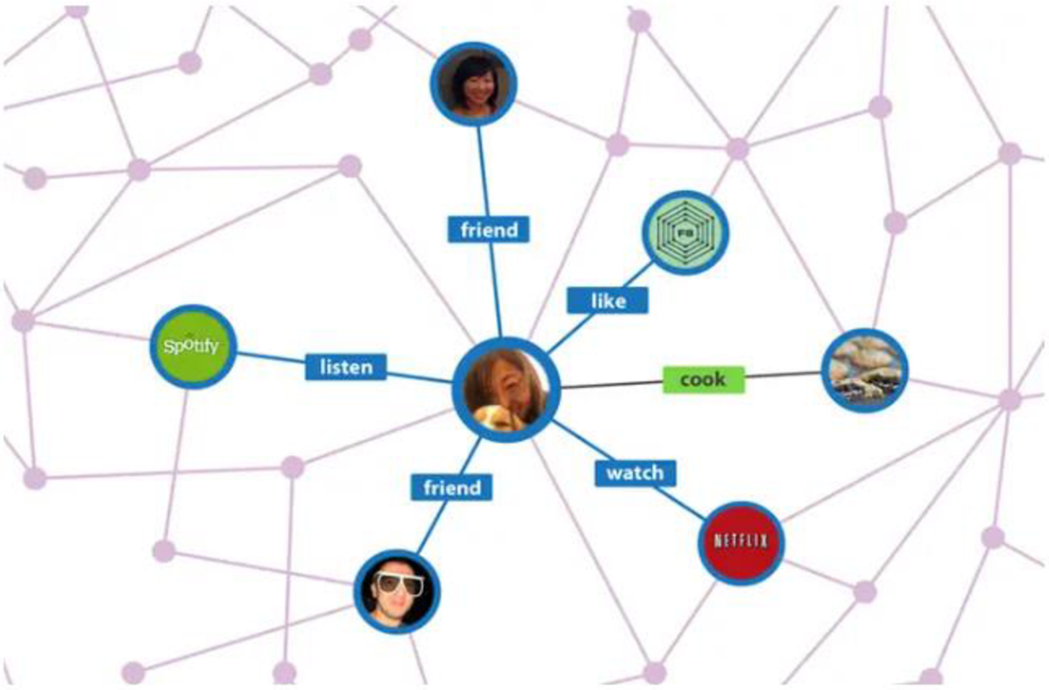


Figure 11 social media, Source: [37]Dickinson, 2012

Strong connections

According to R.Nelson, who affirms that the relationships whose ties are strong, are commonly connected with interactions (D.Krackhardt, 1992),(R. Nelson, 1989), (M.Ruef, 2002). Due to the interactions, people tend to transfer their knowledge pretty fast between each other.

However, strong connections between people are crucially important during time of uncertainty or change (D. Krackhardt, 1992). People are inclined to avoid uncomfortable situations. Strong links are to reduce the refusal. It offers a state of physical ease during the time of uncertainty. The change is not influenced by weak ties, but it is rather affected by a specific type of strong links (D. Krackhardt, 1992).

- Interaction: Provided that A and B are strong links, there must be interaction between them.

- Affection: In case that A and B are strong links, A must be affected by B.
- Time: Granted that A and B are strong ties, links A and B must have some history
- interactions must have lasted for a while.

Weak connections

The weak links between people are regarded as connections that are not in the same social circle of people. Moreover, they are considered to have infrequent interactions (M.Granovetter, 1973), (M. Granovetter, 1983). According to the hypothesis about weak connections (A.Rapoport, 1957), it states that in the case that A is connected with B and C then there is a large chance that B and C are linked mutually as it is demonstrated in Figure 12.

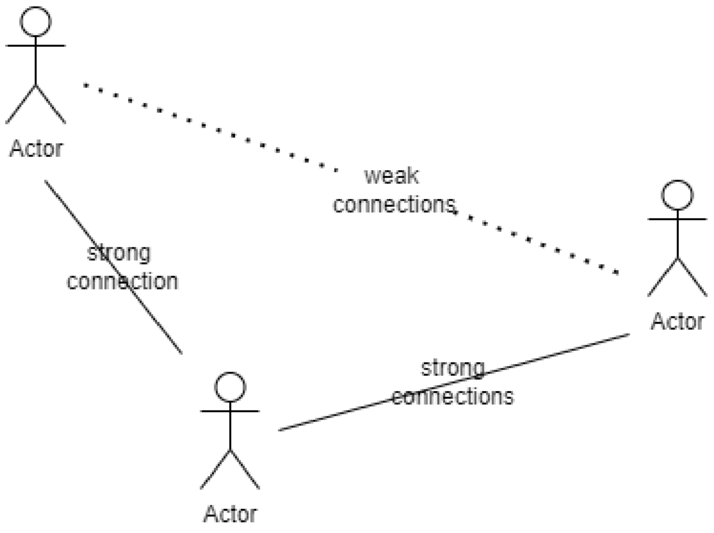


Figure 12 Connections, Source: own work

Positive and negative connections

Admittedly, friendship is regarded as a positive connection. While unfriendly relationship is considered as a negative link between people (A. Rapoport, 1957)]. According to F. Harary, who writes about structural models, states that a signed graph was established with a structure theorem to demonstrate both connections [28]. Provided that relationships are positive in every cycle, the theorem is viewed as balanced. Granted that connections are negative; the theorem is regarded as unbalanced. In the case that the theorem corresponds to a network of positive and negative connections, it might be still balanced. In that case the network is composed of two subnetworks. Every subnetwork possesses positive connections between their nodes and negative connections between other nodes but in different networks.

Whatever the case, there is a specific graph in which one of the best nodes reach each other by doing several small steps. The network is regarded as a small-world network. The theory is part of the social network analysis. It helps to discover connections between people[29].

People profit from the social network daily as for the search and transfer of the information shared between people. There is a profoundly sizeable transfer of information. Besides, people share manifold information on the internet. As a result, the information is expanding profoundly fast.

There are specific aspects for the social network sites:

Actors:

Provided that the user wants to participate in the social network, the user must register their account and log into the network. Several types of users, such as groups or companies, ought to be supported by the social network.

Relationships

Copious social relationships exist in the social network. A relation must be confirmed granted that it is bidirectional, such as in Facebook. On the other hand, a relation does not have to be confirmed

[28] Frank Harary (2005*), Structural models: An introduction to the theory of directed graphs*. [online]. [cit. 2022-06-12].

[29] Duncan J Watts and Steven H Strogatz (1998), *Collective dynamics of small-world networks. nature, 393(6684):440–442* [online]. [cit. 2022-06-12]

in Twitter because the relationship is unidirectional. Furthermore, another type of relation is known as blocking other users.

Content:

There is bountiful of content in the social network, for example text posts, pictures and videos. Some particular content can be offered by a specific oriented network. Whereas social networks are focused on more common content.

Timeline of activities

It is a stream of actions acted by social actors. The home activities are considered as one of the best timelines in which the social actors are included.

Analysis of the Social Network

Begin with the topic, analysis of the social network is social network analysis. It is to investigate the theory of the network in the case of the social network. The actors are to be referred to nodes. Connections are to correspond to relationships between individuals, for example friendship, company, ...and so on. It is a pair of methods for analysing the social structures. They investigate the links of the structures [30]. More precisely, the methods are dependent on the accessibility of relational data.

The relationship between actors is observed by the analysis of the social network. In addition, the importance of the relations between actors depends on the fact how they are reflected. There are manifold of studies focusing on the relationship between entities, such as (Wasserman, 1994).

- Actors are considered as interdependent units.
- Connections between actors are viewed as channels for transferring resources.
- Network models are considered as the environment for offering chances for constraints

According to R. Hanneman, who investigates in the field of the social network, states that analysis of the social network is regarded as the field of mathematical sociology (R. Hanneman, 2001), (R. Hanneman, 2005) and the data is considered by mathematical approaches as conclusive. Subsequently, the final status is appropriately reflected.

[30] John Scott (1988), *Social network analysis. Sociology, 22(1):109–127* [online]. [cit. 2022-06-12].

There are copious ways how to measure the relationships within the social network. In fact, full network method is one of the methods. It is capable of obtaining maximum of information and it is highly priced. It is profoundly difficult to do it. All connections are considered, which offer a complete standpoint of relations in the social network. On the other hand, it is necessary to investigate the full structure of the network (R. Hanneman, 2005).

By way of contrast, the snowball method is not prolific, but it is less expensive. According to H. Hanneman, easier generalisation is allowed to do from the sample. It starts with a focal actor. The actor is asked to name another actor they are connected with. The method stops provided that all actors are mentioned (R. Hanneman, 2005), (H. Hanneman, 2001).

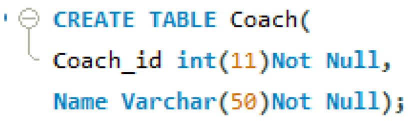
Nevertheless, the snowball method has some weaknesses. Previously, some actors can be isolated. As a result, connections become overstated. It is not guaranteed that all actors are to be linked. It is essential to start at a proper place with selected initial nodes. Finally, this method can overlook isolators, but it is profoundly productive to obtain the information of the network (R. Hanneman, 2005), (H. Hanneman, 2001).

4 Practical Part

Begin with the topic, a simple experiment with a dataset of players has been performed. The dataset consists of information about players, teams they play for, players they play with, coaches they are coached by and teams they played against. The implementation was also done in MySQL80 that is illustrated in Figure 13. Subsequently, JSon code was created and inserted into Neo4j. The process of implementation is demonstrated in Figure 14-23. The complete connections between them are revealed in Figure 24.

Mainly players from the same data set are used for implementing a players' live search. The whole application has been made in php 8, AJAX and MySQL. A sample result of the live search is presented in Figures 32-33.

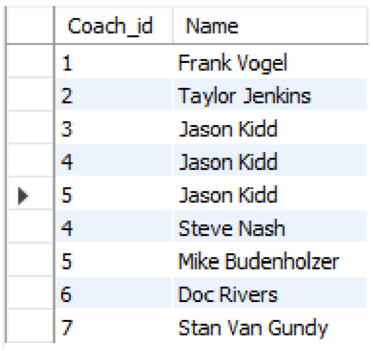
Command 1- Create table Coach



```
CREATE TABLE Coach(
Coach_id int(11)Not Null,
Name Varchar(50)Not Null);
```

Source: own work

Table 5- Coach, Source: own work

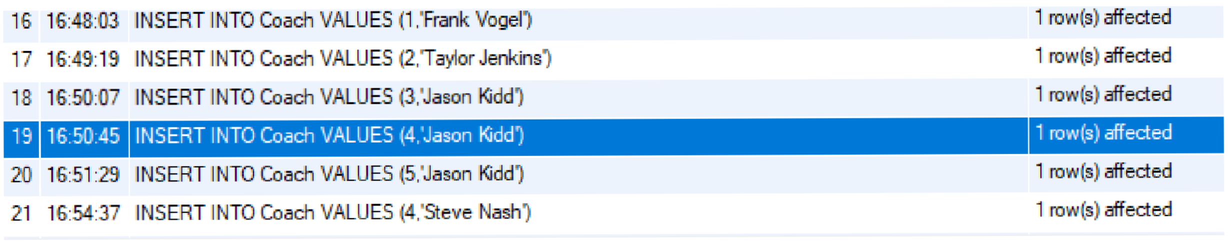


Coach_id	Name
1	Frank Vogel
2	Taylor Jenkins
3	Jason Kidd
4	Jason Kidd
5	Jason Kidd
4	Steve Nash
5	Mike Budenholzer
6	Doc Rivers
7	Stan Van Gundy

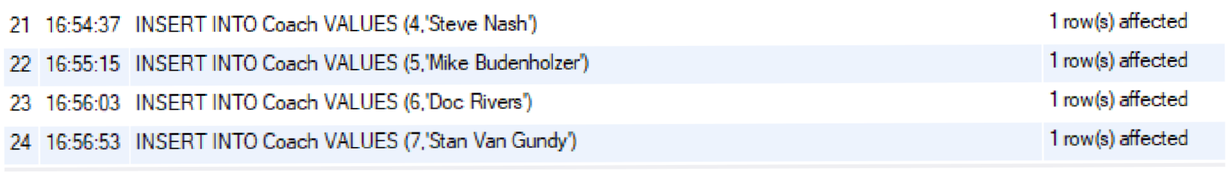
Source: own work

Command 2- INSERT INTO Coach VALUES

Equation 1



```
16  16:48:03  INSERT INTO Coach VALUES (1,'Frank Vogel')        1 row(s) affected
17  16:49:19  INSERT INTO Coach VALUES (2,'Taylor Jenkins')     1 row(s) affected
18  16:50:07  INSERT INTO Coach VALUES (3,'Jason Kidd')         1 row(s) affected
19  16:50:45  INSERT INTO Coach VALUES (4,'Jason Kidd')         1 row(s) affected
20  16:51:29  INSERT INTO Coach VALUES (5,'Jason Kidd')         1 row(s) affected
21  16:54:37  INSERT INTO Coach VALUES (4,'Steve Nash')         1 row(s) affected
```



```
21  16:54:37  INSERT INTO Coach VALUES (4,'Steve Nash')         1 row(s) affected
22  16:55:15  INSERT INTO Coach VALUES (5,'Mike Budenholzer')   1 row(s) affected
23  16:56:03  INSERT INTO Coach VALUES (6,'Doc Rivers')         1 row(s) affected
24  16:56:53  INSERT INTO Coach VALUES (7,'Stan Van Gundy')     1 row(s) affected
```

Source: own work

Command 3- CREATE table team

Equation 2

```
CREATE table TEAM(
team_ID int(11)NOT NULL,
Name VARCHAR(50)NOT NULL);
```

Source: own work

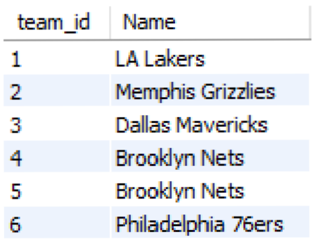
Command 4- SELECT*FROM team

Equation 3

```
select*from team;
```

Source: own work

Table 6- Team, Source: own work

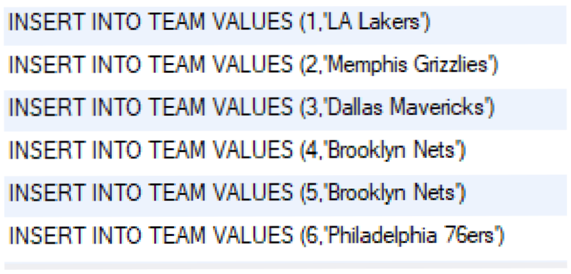


team_id	Name
1	LA Lakers
2	Memphis Grizzlies
3	Dallas Mavericks
4	Brooklyn Nets
5	Brooklyn Nets
6	Philadelphia 76ers

Source: own work

Command 5- INSERT INTO team VALUES

Equation 4

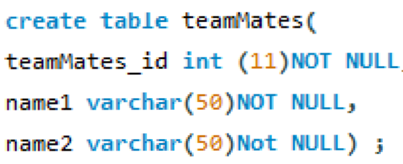


```
INSERT INTO TEAM VALUES (1,'LA Lakers')
INSERT INTO TEAM VALUES (2,'Memphis Grizzlies')
INSERT INTO TEAM VALUES (3,'Dallas Mavericks')
INSERT INTO TEAM VALUES (4,'Brooklyn Nets')
INSERT INTO TEAM VALUES (5,'Brooklyn Nets')
INSERT INTO TEAM VALUES (6,'Philadelphia 76ers')
```

Source: own work

Command 6- CREATE TABLE teamMates

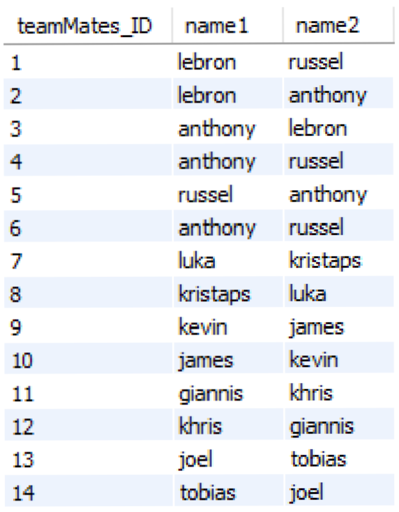
Equation 5



```
create table teamMates(
teamMates_id int (11)NOT NULL,
name1 varchar(50)NOT NULL,
name2 varchar(50)Not NULL) ;
```

Source: own work

Table 7- teamMates, Source: own work

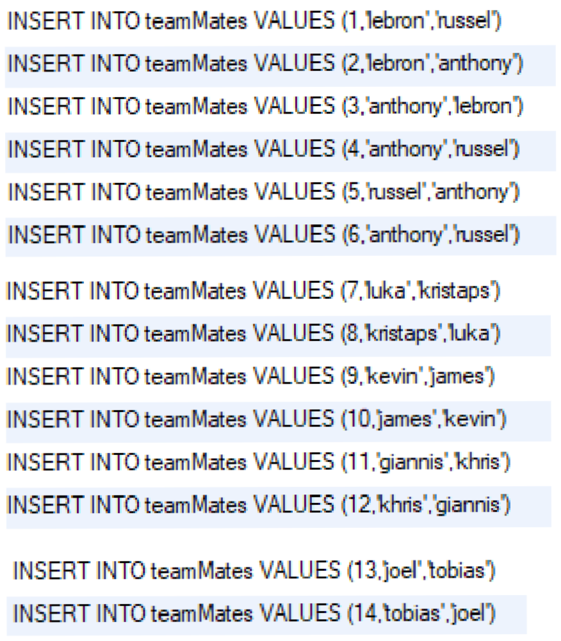


teamMates_ID	name1	name2
1	lebron	russel
2	lebron	anthony
3	anthony	lebron
4	anthony	russel
5	russel	anthony
6	anthony	russel
7	luka	kristaps
8	kristaps	luka
9	kevin	james
10	james	kevin
11	giannis	khris
12	khris	giannis
13	joel	tobias
14	tobias	joel

Source: own work

Command 7- INSERT INTO teamMates VALUES

Equation 6



```
INSERT INTO teamMates VALUES (1,'lebron','russel')
INSERT INTO teamMates VALUES (2,'lebron','anthony')
INSERT INTO teamMates VALUES (3,'anthony','lebron')
INSERT INTO teamMates VALUES (4,'anthony','russel')
INSERT INTO teamMates VALUES (5,'russel','anthony')
INSERT INTO teamMates VALUES (6,'anthony','russel')

INSERT INTO teamMates VALUES (7,'luka','kristaps')
INSERT INTO teamMates VALUES (8,'kristaps','luka')
INSERT INTO teamMates VALUES (9,'kevin','james')
INSERT INTO teamMates VALUES (10,'james','kevin')
INSERT INTO teamMates VALUES (11,'giannis','khris')
INSERT INTO teamMates VALUES (12,'khris','giannis')

INSERT INTO teamMates VALUES (13,'joel','tobias')
INSERT INTO teamMates VALUES (14,'tobias','joel')
```

Source: own work

Command-CREATE TABLE coaches

```
create table coaches(
coaches_id int(11)NOT NULL,
coachName varchar(50)NOT NULL,
playerName varchar(50)NOT NULL);
```

Source: own work

Command 8- INSERT INTO coaches VALUES

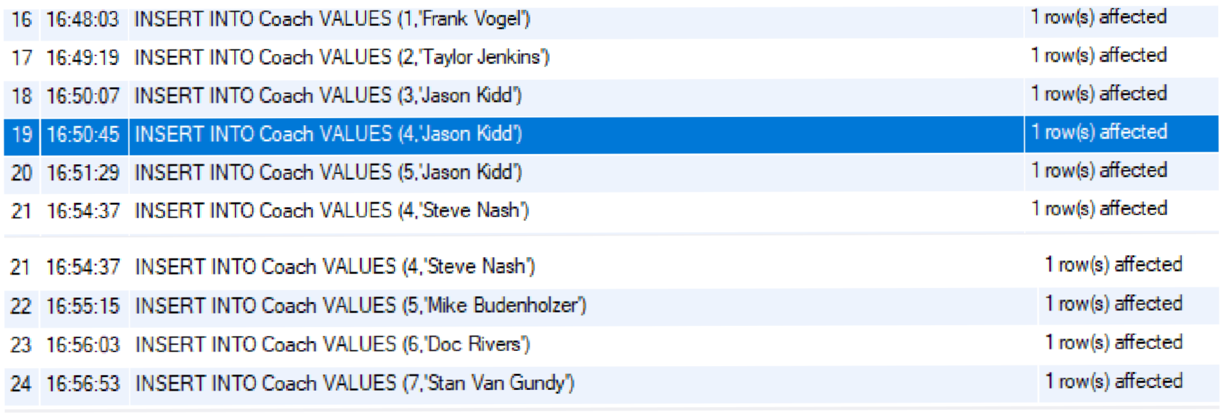
Equation 7

```
INSERT INTO coaches VALUES (1,'frank','lebron');
INSERT INTO coaches VALUES (2,'frank','anthony');
INSERT INTO coaches VALUES (3,'frank','russel');
INSERT INTO coaches VALUES (4,'taylor','ja');
INSERT INTO coaches VALUES (5,'jason','luka');
INSERT INTO coaches VALUES (6,'jason','kristaps');
INSERT INTO coaches VALUES (7,'steve','kevin');
INSERT INTO coaches VALUES (8,'steve','jame');
INSERT INTO coaches VALUES (9,'mike','giannis');
INSERT INTO coaches VALUES (10,'mike','khris');
INSERT INTO coaches VALUES (11,'doc','tobias');
INSERT INTO coaches VALUES (12,'doc','joel');
```

Source: own work

Command 9- INSERT INTO coach VALUES

Equation 8

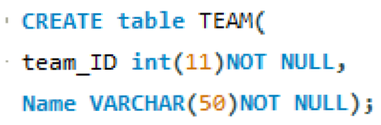


```
16  16:48:03  INSERT INTO Coach VALUES (1,'Frank Vogel')          1 row(s) affected
17  16:49:19  INSERT INTO Coach VALUES (2,'Taylor Jenkins')       1 row(s) affected
18  16:50:07  INSERT INTO Coach VALUES (3,'Jason Kidd')           1 row(s) affected
19  16:50:45  INSERT INTO Coach VALUES (4,'Jason Kidd')           1 row(s) affected
20  16:51:29  INSERT INTO Coach VALUES (5,'Jason Kidd')           1 row(s) affected
21  16:54:37  INSERT INTO Coach VALUES (4,'Steve Nash')           1 row(s) affected

21  16:54:37  INSERT INTO Coach VALUES (4,'Steve Nash')           1 row(s) affected
22  16:55:15  INSERT INTO Coach VALUES (5,'Mike Budenholzer')     1 row(s) affected
23  16:56:03  INSERT INTO Coach VALUES (6,'Doc Rivers')           1 row(s) affected
24  16:56:53  INSERT INTO Coach VALUES (7,'Stan Van Gundy')       1 row(s) affected
```

Source: own work

Command 10- CREATE table team

Equation 9

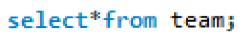


```
CREATE table TEAM(
team_ID int(11)NOT NULL,
Name VARCHAR(50)NOT NULL);
```

Source: own work

Command 11- SELECT*FROM team

Equation 10



```
select*from team;
```

Source: own work

Command 12- INSERT INTO team VALUES

Equation 11

```
INSERT INTO TEAM VALUES (1,'LA Lakers')
INSERT INTO TEAM VALUES (2,'Memphis Grizzlies')
INSERT INTO TEAM VALUES (3,'Dallas Mavericks')
INSERT INTO TEAM VALUES (4,'Brooklyn Nets')
INSERT INTO TEAM VALUES (5,'Brooklyn Nets')
INSERT INTO TEAM VALUES (6,'Philadelphia 76ers')
```

Source: own work

Command 13- CREATE TABLE teamMates

Equation 12

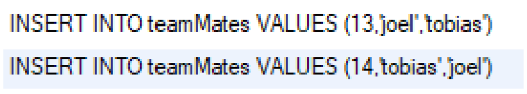
```
create table teamMates(
teamMates_id int (11)NOT NULL,
name1 varchar(50)NOT NULL,
name2 varchar(50)Not NULL) ;
```

Source: own work

Command 14- INSERT INTO teamMates VALUES

Equation 13

```
INSERT INTO teamMates VALUES (1,'lebron','russel')
INSERT INTO teamMates VALUES (2,'lebron','anthony')
INSERT INTO teamMates VALUES (3,'anthony','lebron')
INSERT INTO teamMates VALUES (4,'anthony','russel')
INSERT INTO teamMates VALUES (5,'russel','anthony')
INSERT INTO teamMates VALUES (6,'anthony','russel')
INSERT INTO teamMates VALUES (7,'luka','kristaps')
INSERT INTO teamMates VALUES (8,'kristaps','luka')
INSERT INTO teamMates VALUES (9,'kevin','james')
INSERT INTO teamMates VALUES (10,'james','kevin')
INSERT INTO teamMates VALUES (11,'giannis','khris')
INSERT INTO teamMates VALUES (12,'khris','giannis')
```

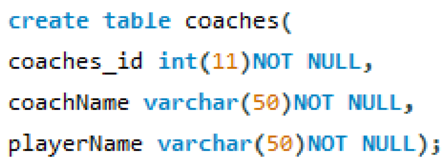


```
INSERT INTO teamMates VALUES (13,'joel','tobias')
INSERT INTO teamMates VALUES (14,'tobias','joel')
```

Source: own work

Command 15-CREATE TABLE coaches

Equation 14

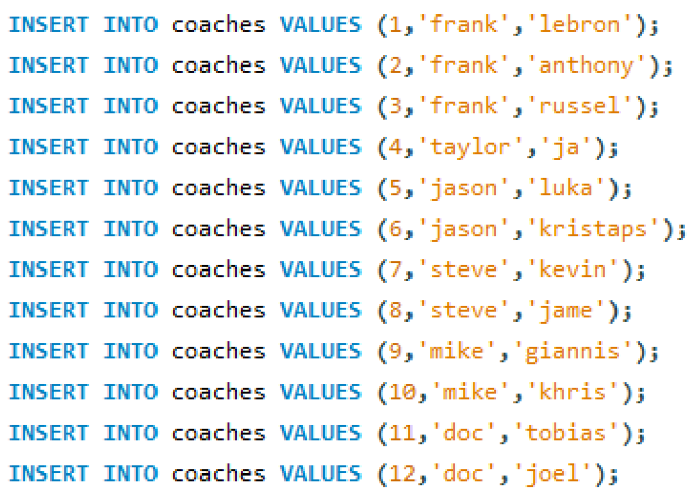


```
create table coaches(
coaches_id int(11)NOT NULL,
coachName varchar(50)NOT NULL,
playerName varchar(50)NOT NULL);
```

Source: own work

Command 16- INSERT INTO coaches VALUES

Equation 15



```
INSERT INTO coaches VALUES (1,'frank','lebron');
INSERT INTO coaches VALUES (2,'frank','anthony');
INSERT INTO coaches VALUES (3,'frank','russel');
INSERT INTO coaches VALUES (4,'taylor','ja');
INSERT INTO coaches VALUES (5,'jason','luka');
INSERT INTO coaches VALUES (6,'jason','kristaps');
INSERT INTO coaches VALUES (7,'steve','kevin');
INSERT INTO coaches VALUES (8,'steve','jame');
INSERT INTO coaches VALUES (9,'mike','giannis');
INSERT INTO coaches VALUES (10,'mike','khris');
INSERT INTO coaches VALUES (11,'doc','tobias');
INSERT INTO coaches VALUES (12,'doc','joel');
```

Source: own work

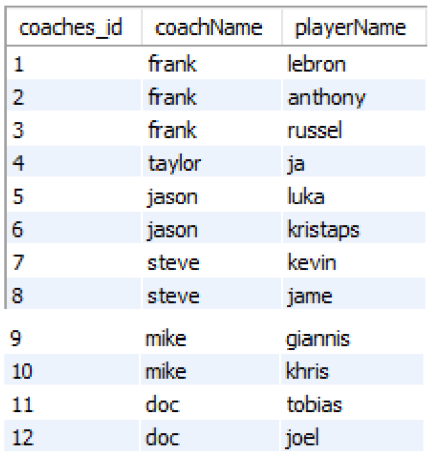
Command 17- SELECT*FROM coaches

Equation 16

```
SELECT*from coaches;
```

Source: own work

Table 8- coaches, Source: own work



coaches_id	coachName	playerName
1	frank	lebron
2	frank	anthony
3	frank	russel
4	taylor	ja
5	jason	luka
6	jason	kristaps
7	steve	kevin
8	steve	jame
9	mike	giannis
10	mike	khris
11	doc	tobias
12	doc	joel

Source: own work

Command 18- SELECT*FROM players

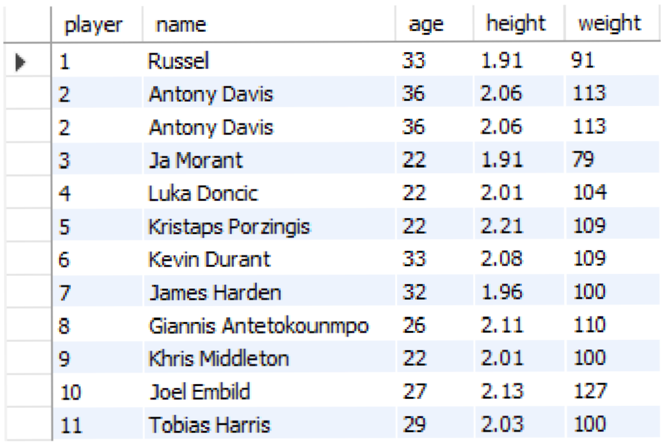
Equation 17

```
SELECT*from players;
```

Source: own work

Table 9- player, Source: own work

Table 1

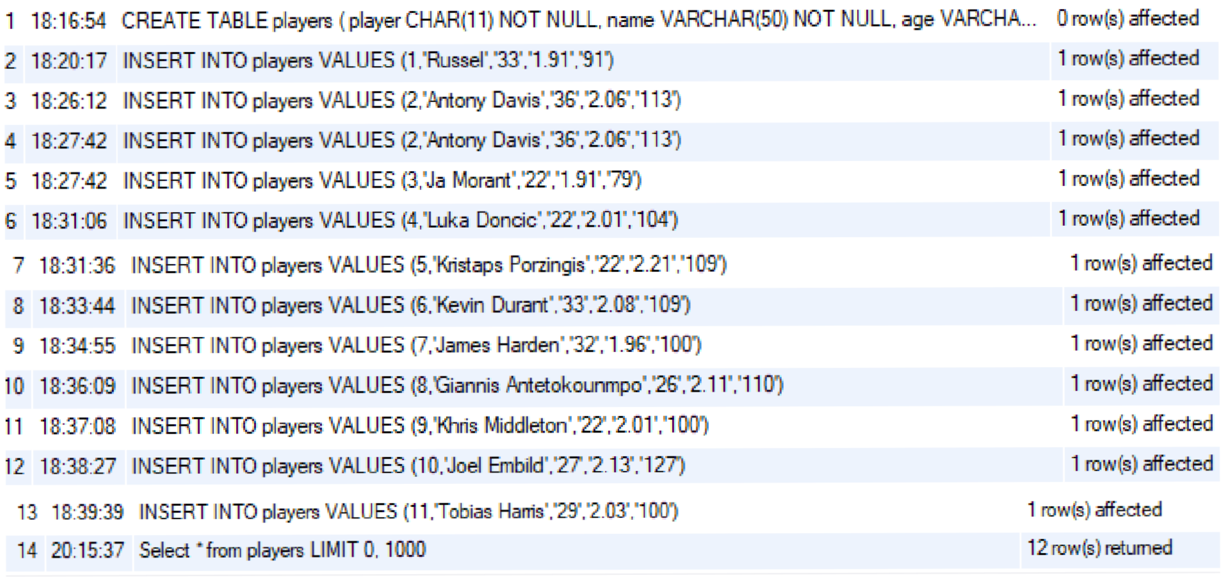


player	name	age	height	weight
1	Russel	33	1.91	91
2	Antony Davis	36	2.06	113
2	Antony Davis	36	2.06	113
3	Ja Morant	22	1.91	79
4	Luka Doncic	22	2.01	104
5	Kristaps Porzingis	22	2.21	109
6	Kevin Durant	33	2.08	109
7	James Harden	32	1.96	100
8	Giannis Antetokounmpo	26	2.11	110
9	Khris Middleton	22	2.01	100
10	Joel Embild	27	2.13	127
11	Tobias Harris	29	2.03	100

Source: own work

Command 19- CREATE TABLE players.

Equation 18

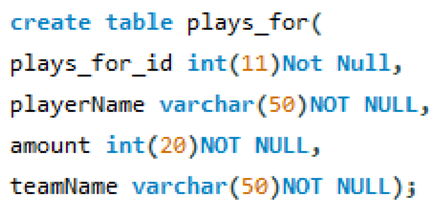


```
1   18:16:54  CREATE TABLE players ( player CHAR(11) NOT NULL, name VARCHAR(50) NOT NULL, age VARCHA...   0 row(s) affected
2   18:20:17  INSERT INTO players VALUES (1,'Russel','33','1.91','91')                                  1 row(s) affected
3   18:26:12  INSERT INTO players VALUES (2,'Antony Davis','36','2.06','113')                           1 row(s) affected
4   18:27:42  INSERT INTO players VALUES (2,'Antony Davis','36','2.06','113')                           1 row(s) affected
5   18:27:42  INSERT INTO players VALUES (3,'Ja Morant','22','1.91','79')                               1 row(s) affected
6   18:31:06  INSERT INTO players VALUES (4,'Luka Doncic','22','2.01','104')                            1 row(s) affected
7   18:31:36  INSERT INTO players VALUES (5,'Kristaps Porzingis','22','2.21','109')                     1 row(s) affected
8   18:33:44  INSERT INTO players VALUES (6,'Kevin Durant','33','2.08','109')                           1 row(s) affected
9   18:34:55  INSERT INTO players VALUES (7,'James Harden','32','1.96','100')                           1 row(s) affected
10  18:36:09  INSERT INTO players VALUES (8,'Giannis Antetokounmpo','26','2.11','110')                  1 row(s) affected
11  18:37:08  INSERT INTO players VALUES (9,'Khris Middleton','22','2.01','100')                        1 row(s) affected
12  18:38:27  INSERT INTO players VALUES (10,'Joel Embild','27','2.13','127')                           1 row(s) affected
13  18:39:39  INSERT INTO players VALUES (11,'Tobias Harris','29','2.03','100')                         1 row(s) affected
14  20:15:37  Select * from players LIMIT 0, 1000                                                       12 row(s) returned
```

Source: own work

Command 20- CREATE TABLE plays_for

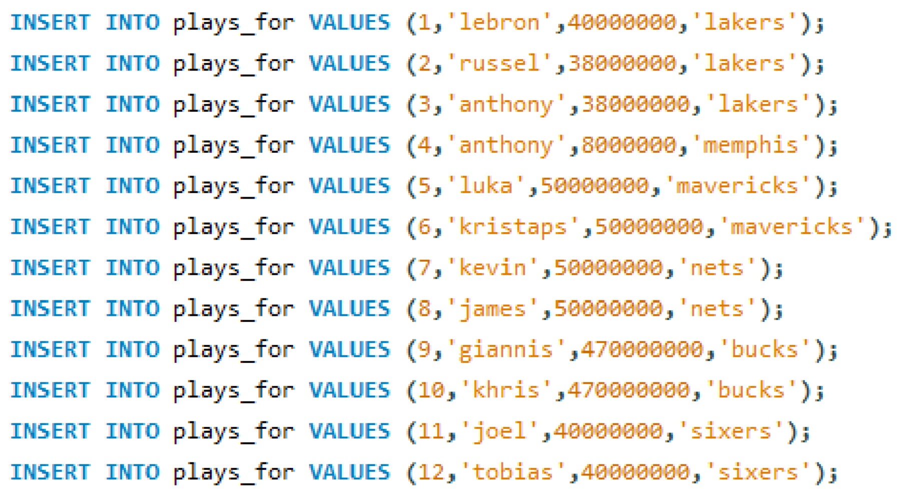
Equation 19



```
create table plays_for(
plays_for_id int(11)Not Null,
playerName varchar(50)NOT NULL,
amount int(20)NOT NULL,
teamName varchar(50)NOT NULL);
```

Source: own work

Command 21- INSERT INTO plays_for VALUES

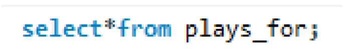


```
INSERT INTO plays_for VALUES (1,'lebron',40000000,'lakers');
INSERT INTO plays_for VALUES (2,'russel',38000000,'lakers');
INSERT INTO plays_for VALUES (3,'anthony',38000000,'lakers');
INSERT INTO plays_for VALUES (4,'anthony',8000000,'memphis');
INSERT INTO plays_for VALUES (5,'luka',50000000,'mavericks');
INSERT INTO plays_for VALUES (6,'kristaps',50000000,'mavericks');
INSERT INTO plays_for VALUES (7,'kevin',50000000,'nets');
INSERT INTO plays_for VALUES (8,'james',50000000,'nets');
INSERT INTO plays_for VALUES (9,'giannis',470000000,'bucks');
INSERT INTO plays_for VALUES (10,'khris',470000000,'bucks');
INSERT INTO plays_for VALUES (11,'joel',40000000,'sixers');
INSERT INTO plays_for VALUES (12,'tobias',40000000,'sixers');
```

Source: own work

Command 22- SELECT*FROM plays_for

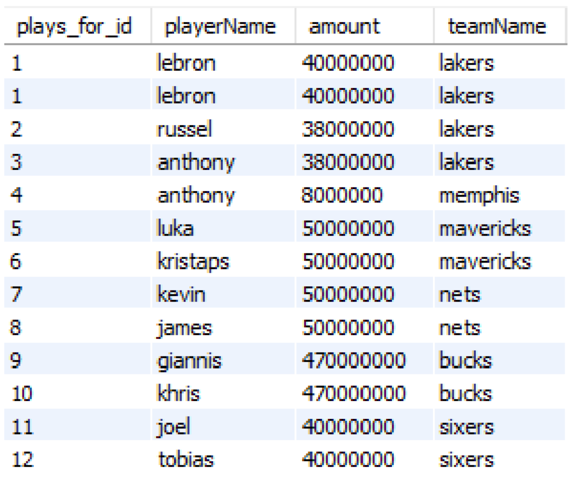
Equation 20



```
select*from plays_for;
```

Source: own work

Table 10- plays_for, Source: own work

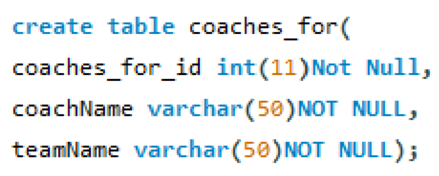


plays_for_id	playerName	amount	teamName
1	lebron	40000000	lakers
1	lebron	40000000	lakers
2	russel	38000000	lakers
3	anthony	38000000	lakers
4	anthony	8000000	memphis
5	luka	50000000	mavericks
6	kristaps	50000000	mavericks
7	kevin	50000000	nets
8	james	50000000	nets
9	giannis	470000000	bucks
10	khris	470000000	bucks
11	joel	40000000	sixers
12	tobias	40000000	sixers

Source: own work

Command 23- CREATE TABLE coaches_for

Equation 21

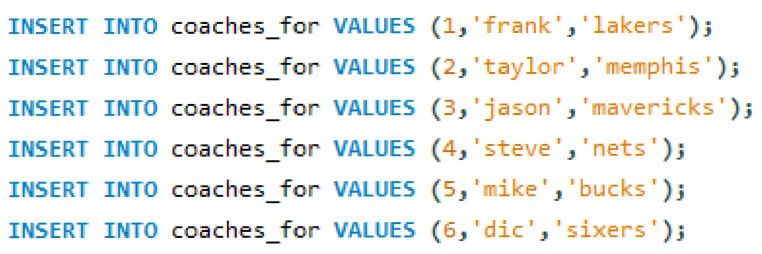


```
create table coaches_for(
coaches_for_id int(11)Not Null,
coachName varchar(50)NOT NULL,
teamName varchar(50)NOT NULL);
```

Source: own work

Command 24- INSERT INTO coaches_for

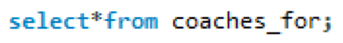
Equation 22



```sql
INSERT INTO coaches_for VALUES (1,'frank','lakers');
INSERT INTO coaches_for VALUES (2,'taylor','memphis');
INSERT INTO coaches_for VALUES (3,'jason','mavericks');
INSERT INTO coaches_for VALUES (4,'steve','nets');
INSERT INTO coaches_for VALUES (5,'mike','bucks');
INSERT INTO coaches_for VALUES (6,'dic','sixers');
```

Source: own work

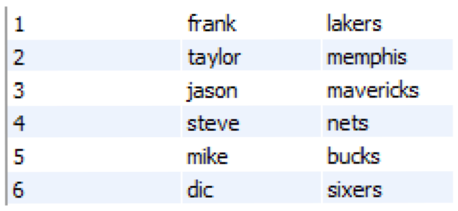
Command 25- SELECT*FROM coaches_for



```sql
select*from coaches_for;
```

Source: own work

Table 11- coaches_for, Source: own work

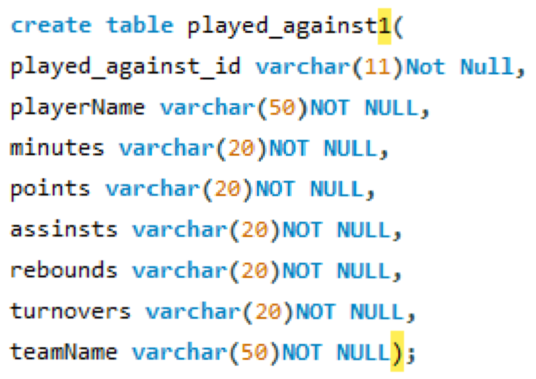


1	frank	lakers
2	taylor	memphis
3	jason	mavericks
4	steve	nets
5	mike	bucks
6	dic	sixers

Source: own work

Command 26- CREATE TABLE played_against

Equation 23

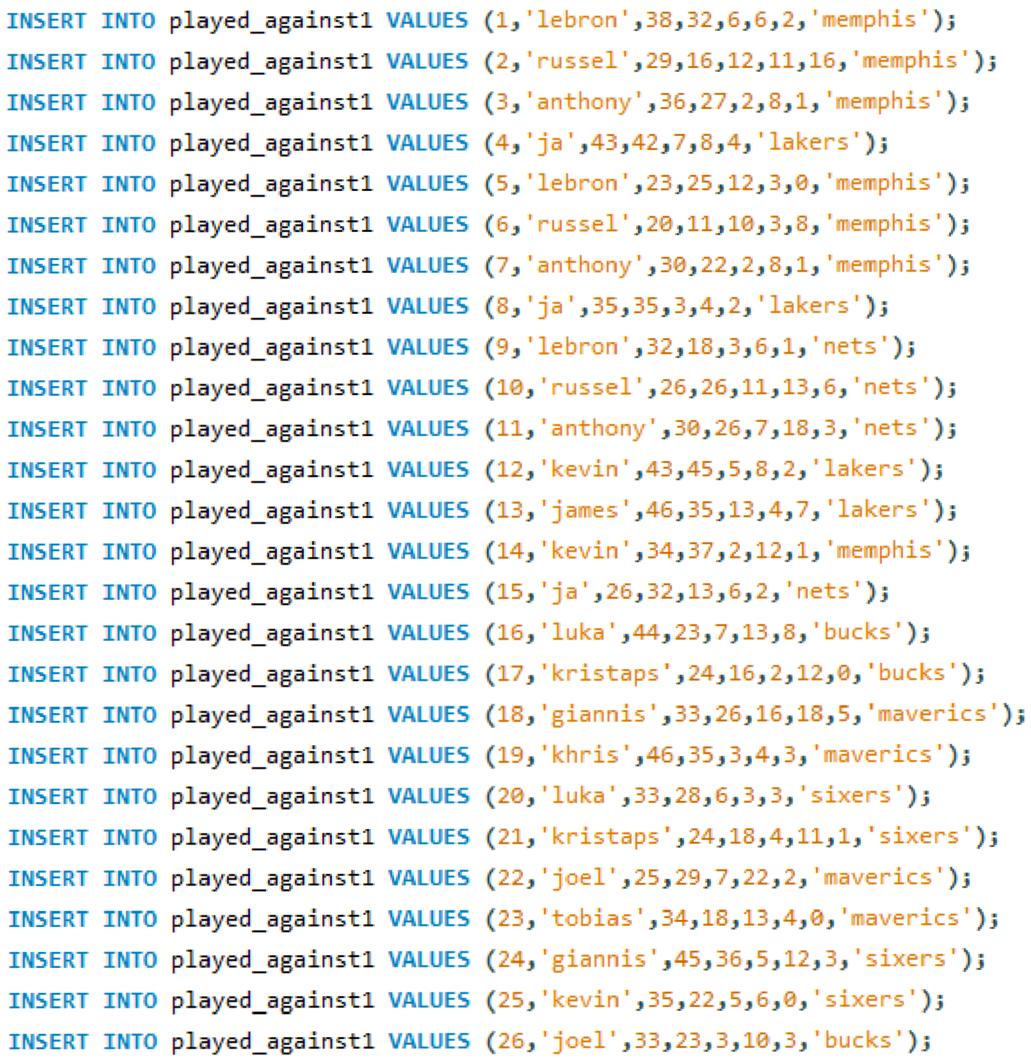


```
create table played_against1(
played_against_id varchar(11)Not Null,
playerName varchar(50)NOT NULL,
minutes varchar(20)NOT NULL,
points varchar(20)NOT NULL,
assinsts varchar(20)NOT NULL,
rebounds varchar(20)NOT NULL,
turnovers varchar(20)NOT NULL,
teamName varchar(50)NOT NULL);
```

Source: own work

Command 26- INSERT INTO played_against1 VALUES

Equation 24

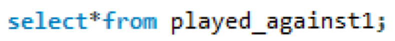


```
INSERT INTO played_against1 VALUES (1,'lebron',38,32,6,6,2,'memphis');
INSERT INTO played_against1 VALUES (2,'russel',29,16,12,11,16,'memphis');
INSERT INTO played_against1 VALUES (3,'anthony',36,27,2,8,1,'memphis');
INSERT INTO played_against1 VALUES (4,'ja',43,42,7,8,4,'lakers');
INSERT INTO played_against1 VALUES (5,'lebron',23,25,12,3,0,'memphis');
INSERT INTO played_against1 VALUES (6,'russel',20,11,10,3,8,'memphis');
INSERT INTO played_against1 VALUES (7,'anthony',30,22,2,8,1,'memphis');
INSERT INTO played_against1 VALUES (8,'ja',35,35,3,4,2,'lakers');
INSERT INTO played_against1 VALUES (9,'lebron',32,18,3,6,1,'nets');
INSERT INTO played_against1 VALUES (10,'russel',26,26,11,13,6,'nets');
INSERT INTO played_against1 VALUES (11,'anthony',30,26,7,18,3,'nets');
INSERT INTO played_against1 VALUES (12,'kevin',43,45,5,8,2,'lakers');
INSERT INTO played_against1 VALUES (13,'james',46,35,13,4,7,'lakers');
INSERT INTO played_against1 VALUES (14,'kevin',34,37,2,12,1,'memphis');
INSERT INTO played_against1 VALUES (15,'ja',26,32,13,6,2,'nets');
INSERT INTO played_against1 VALUES (16,'luka',44,23,7,13,8,'bucks');
INSERT INTO played_against1 VALUES (17,'kristaps',24,16,2,12,0,'bucks');
INSERT INTO played_against1 VALUES (18,'giannis',33,26,16,18,5,'maverics');
INSERT INTO played_against1 VALUES (19,'khris',46,35,3,4,3,'maverics');
INSERT INTO played_against1 VALUES (20,'luka',33,28,6,3,3,'sixers');
INSERT INTO played_against1 VALUES (21,'kristaps',24,18,4,11,1,'sixers');
INSERT INTO played_against1 VALUES (22,'joel',25,29,7,22,2,'maverics');
INSERT INTO played_against1 VALUES (23,'tobias',34,18,13,4,0,'maverics');
INSERT INTO played_against1 VALUES (24,'giannis',45,36,5,12,3,'sixers');
INSERT INTO played_against1 VALUES (25,'kevin',35,22,5,6,0,'sixers');
INSERT INTO played_against1 VALUES (26,'joel',33,23,3,10,3,'bucks');
```

Source: own work

Command 28- SELECT*FROM played_against

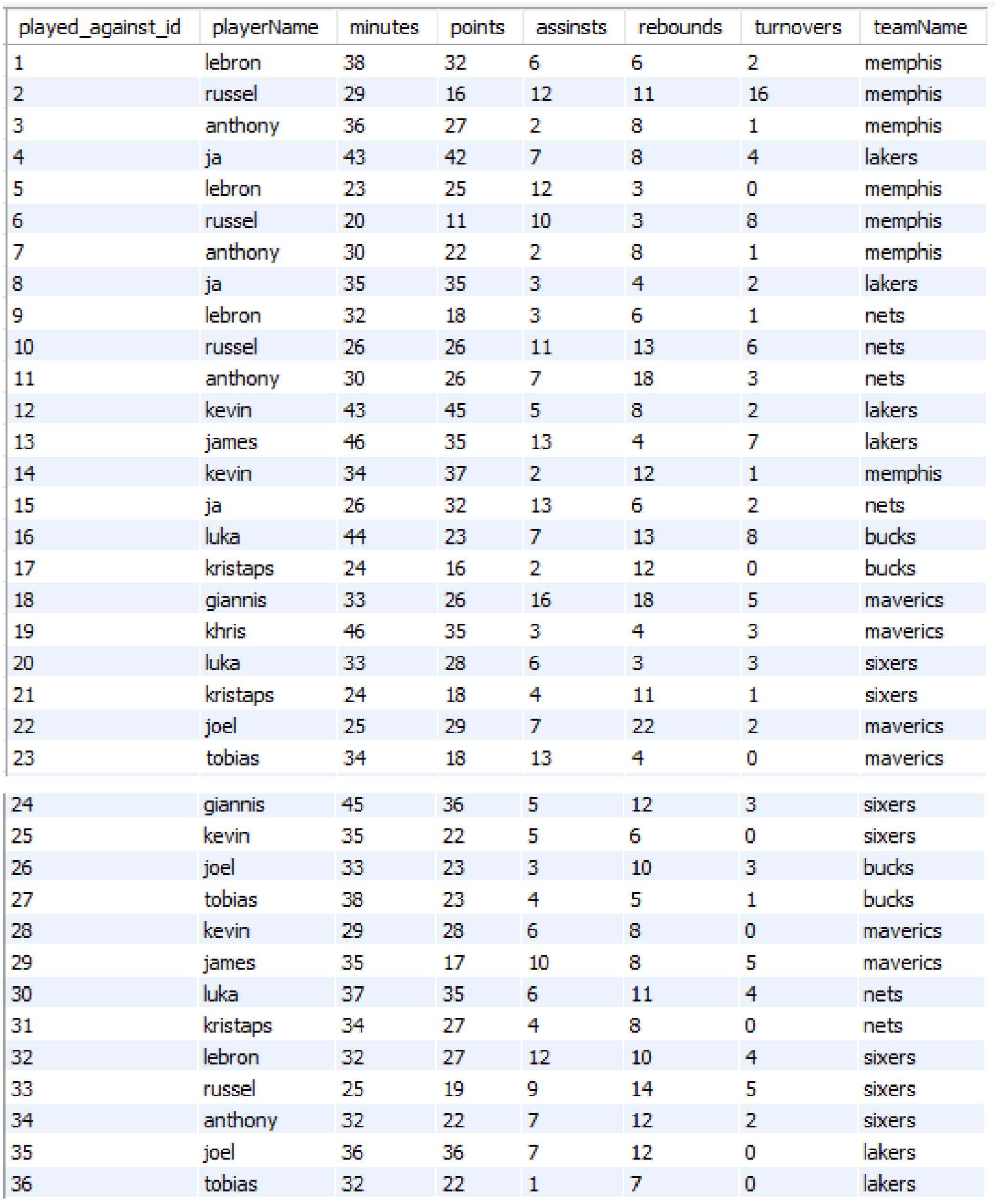
Equation 25



```
select*from played_against1;
```

Source: own work

Table 12- played_against, Source: own work



played_against_id	playerName	minutes	points	assinsts	rebounds	turnovers	teamName
1	lebron	38	32	6	6	2	memphis
2	russel	29	16	12	11	16	memphis
3	anthony	36	27	2	8	1	memphis
4	ja	43	42	7	8	4	lakers
5	lebron	23	25	12	3	0	memphis
6	russel	20	11	10	3	8	memphis
7	anthony	30	22	2	8	1	memphis
8	ja	35	35	3	4	2	lakers
9	lebron	32	18	3	6	1	nets
10	russel	26	26	11	13	6	nets
11	anthony	30	26	7	18	3	nets
12	kevin	43	45	5	8	2	lakers
13	james	46	35	13	4	7	lakers
14	kevin	34	37	2	12	1	memphis
15	ja	26	32	13	6	2	nets
16	luka	44	23	7	13	8	bucks
17	kristaps	24	16	2	12	0	bucks
18	giannis	33	26	16	18	5	maverics
19	khris	46	35	3	4	3	maverics
20	luka	33	28	6	3	3	sixers
21	kristaps	24	18	4	11	1	sixers
22	joel	25	29	7	22	2	maverics
23	tobias	34	18	13	4	0	maverics
24	giannis	45	36	5	12	3	sixers
25	kevin	35	22	5	6	0	sixers
26	joel	33	23	3	10	3	bucks
27	tobias	38	23	4	5	1	bucks
28	kevin	29	28	6	8	0	maverics
29	james	35	17	10	8	5	maverics
30	luka	37	35	6	11	4	nets
31	kristaps	34	27	4	8	0	nets
32	lebron	32	27	12	10	4	sixers
33	russel	25	19	9	14	5	sixers
34	anthony	32	22	7	12	2	sixers
35	joel	36	36	7	12	0	lakers
36	tobias	32	22	1	7	0	lakers

Source: own work

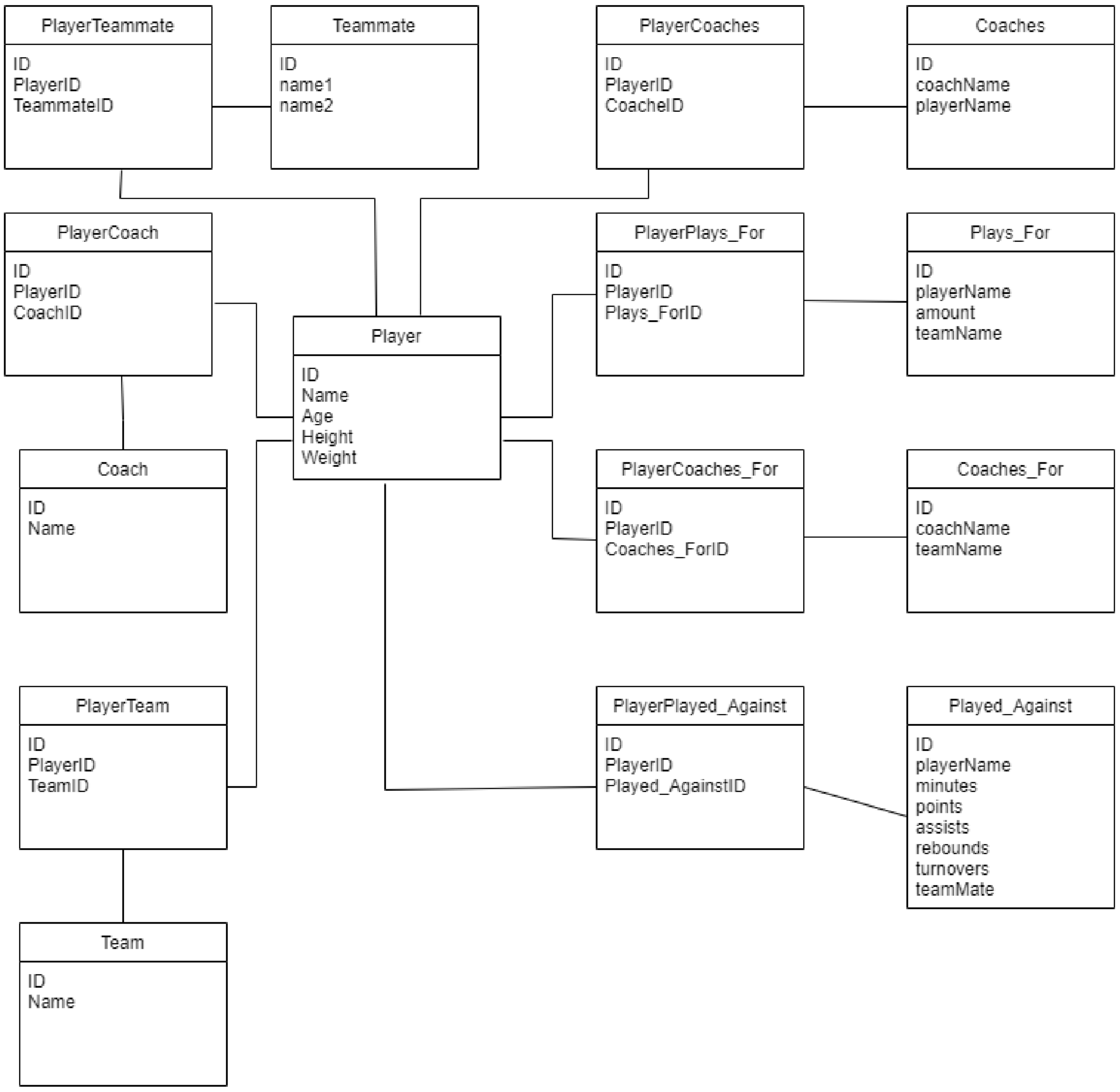


Figure 13 NBA database – MySQL server, Source: own work

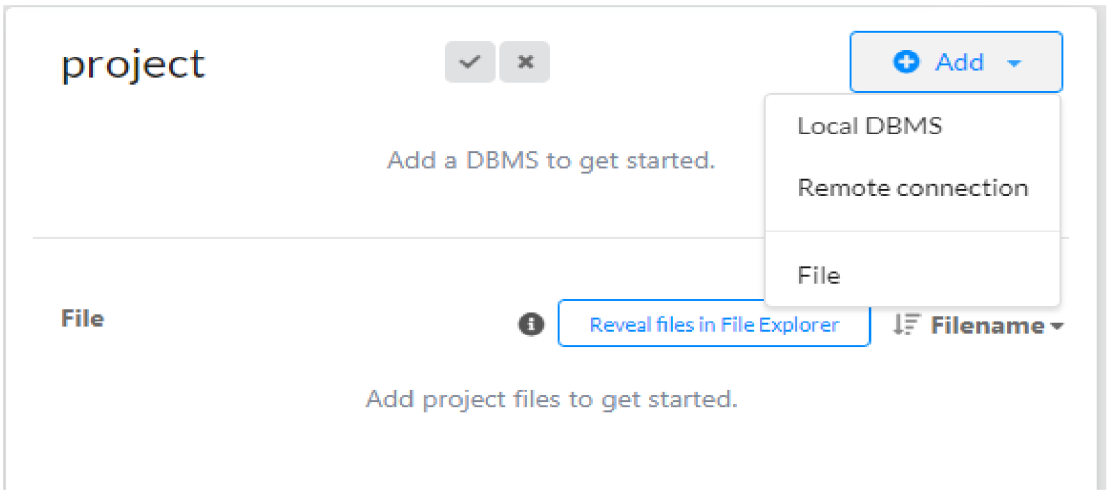

project
Add
Local DBMS
Remote connection
File
Add a DBMS to get started.
File
Reveal files in File Explorer
Filename
Add project files to get started.

Figure 14 Local DBMS, Source: own work

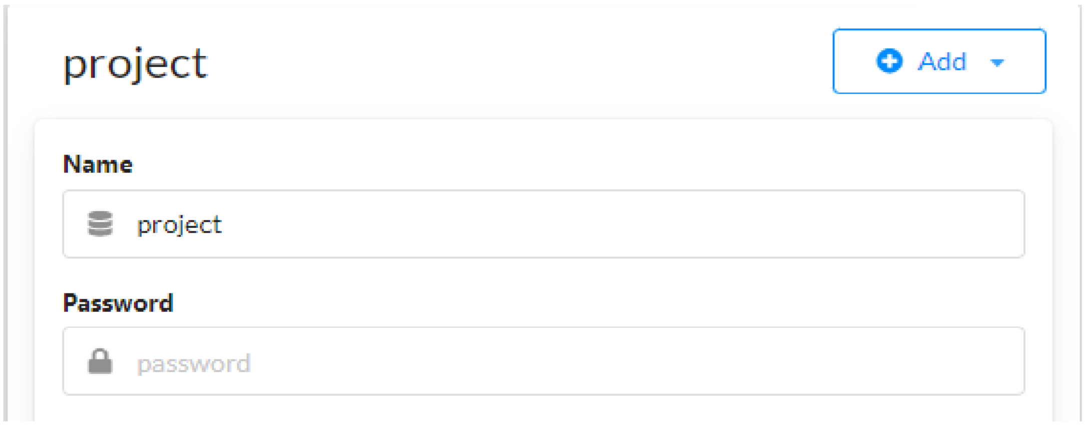
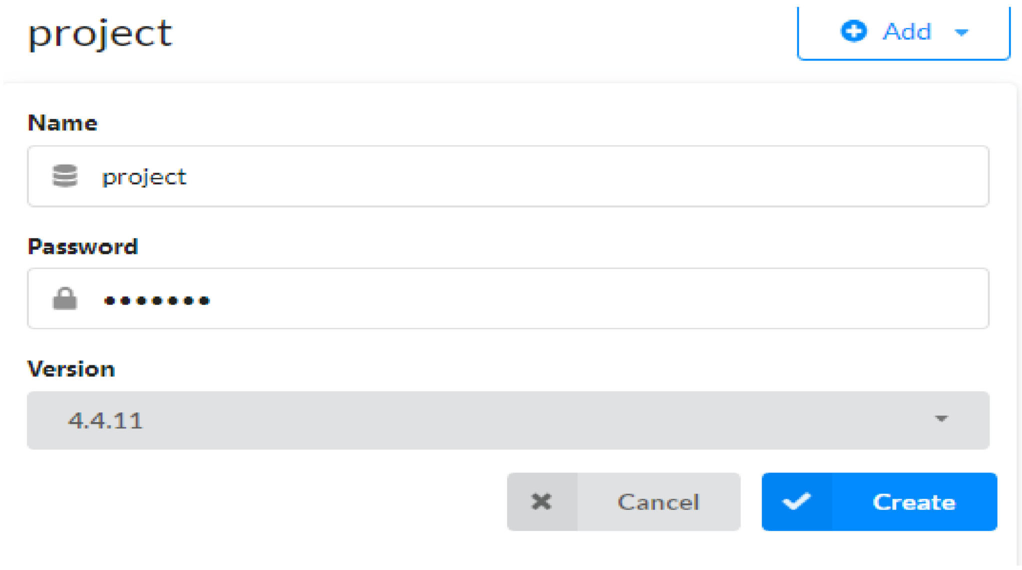


Figure 15 Giving a name, Source: own work

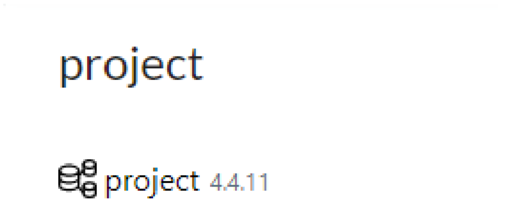

project

Name: project

Password: •••••••

Version: 4.4.11

Figure 16 Password, Source: own work


project

project 4.4.11

Figure 17 Database, Source: own work

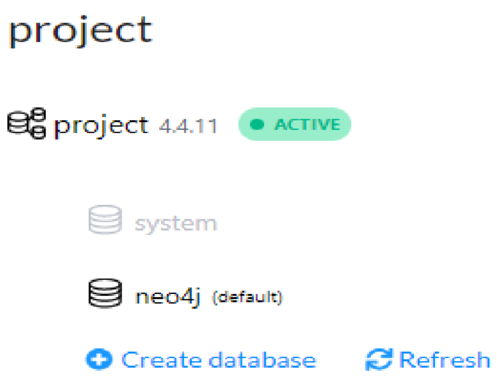


Figure 18 Created database, Source: own work

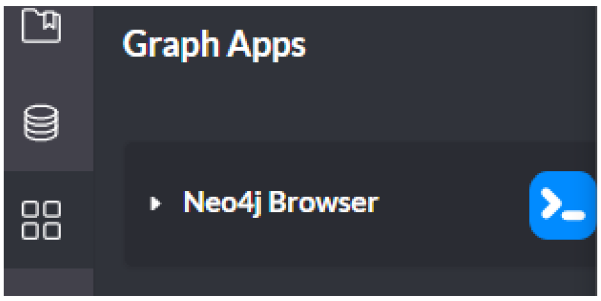
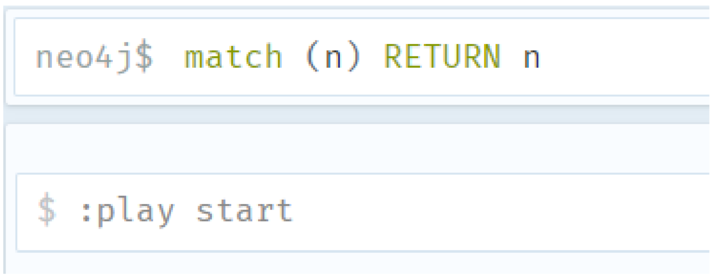
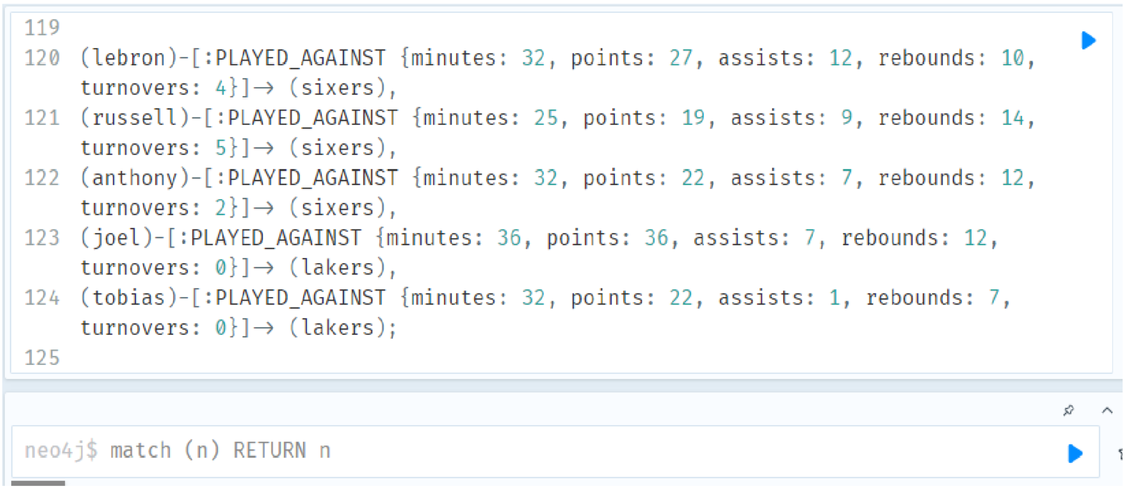


Figure 19 Neo4j Browser, Source: own work



```
neo4j$ match (n) RETURN n

$ :play start
```

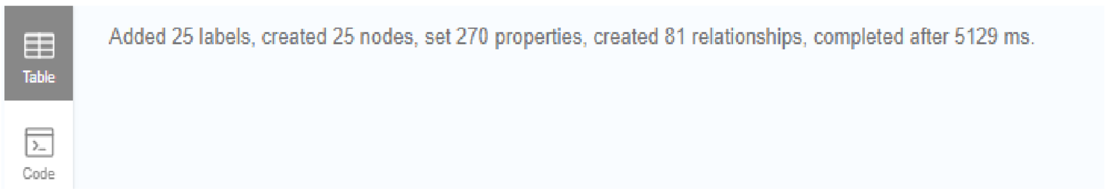
Figure 20 Neo4JQuerry, Source: own work



```
(lebron)-[:PLAYED_AGAINST {minutes: 32, points: 27, assists: 12, rebounds: 10,
turnovers: 4}]→ (sixers),
(russell)-[:PLAYED_AGAINST {minutes: 25, points: 19, assists: 9, rebounds: 14,
turnovers: 5}]→ (sixers),
(anthony)-[:PLAYED_AGAINST {minutes: 32, points: 22, assists: 7, rebounds: 12,
turnovers: 2}]→ (sixers),
(joel)-[:PLAYED_AGAINST {minutes: 36, points: 36, assists: 7, rebounds: 12,
turnovers: 0}]→ (lakers),
(tobias)-[:PLAYED_AGAINST {minutes: 32, points: 22, assists: 1, rebounds: 7,
turnovers: 0}]→ (lakers);
```

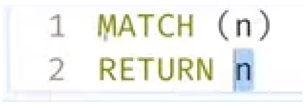
```
neo4j$ match (n) RETURN n
```

Figure 21 Inserted Code, Source: own work



```
Added 25 labels, created 25 nodes, set 270 properties, created 81 relationships, completed after 5129 ms.
```

Figure 22 Summary of the Task, Source: own work



```
MATCH (n)
RETURN n
```

Figure 23 Final Querry, Source: own work

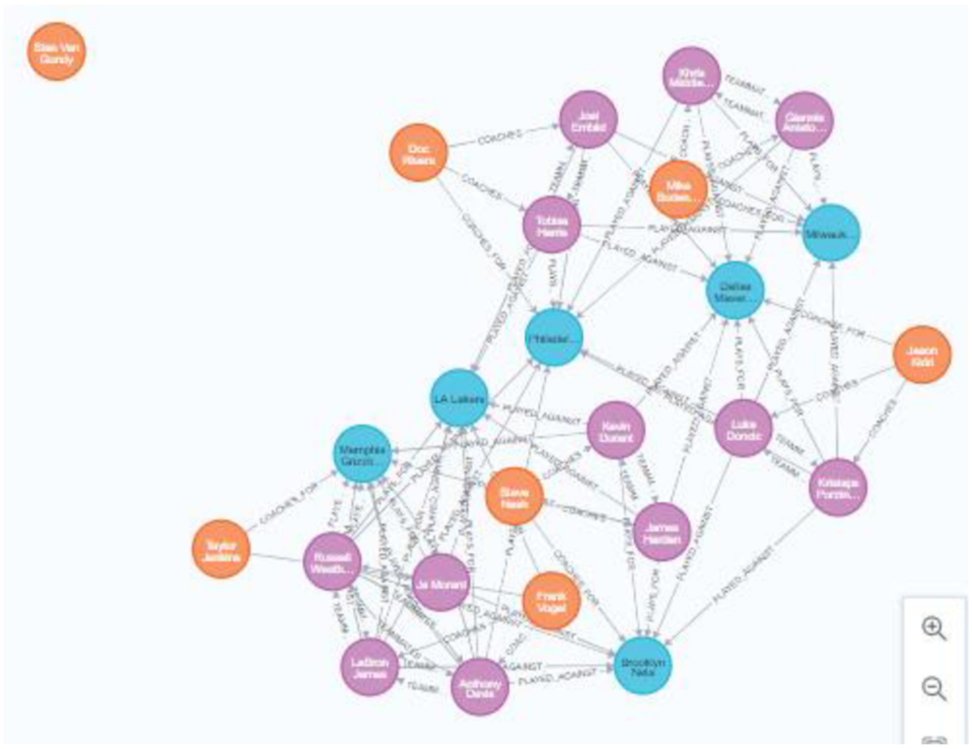


Figure 24 Neo4j, Source: own work

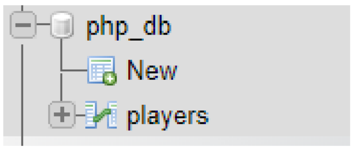
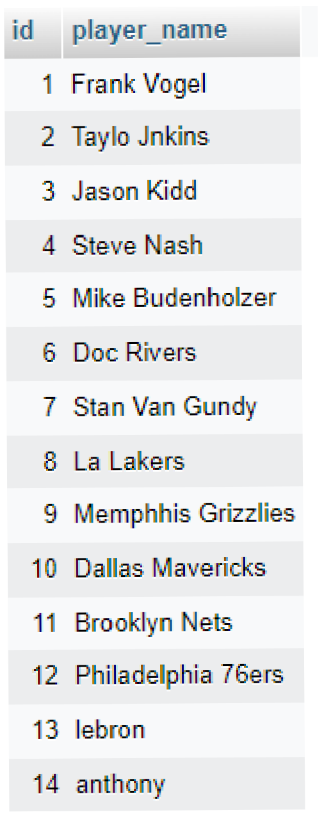


Figure 25 Database, Source: own work

```
CREATE TABLE players (
  id int not null,
  player_name varchar(50) not null);
```

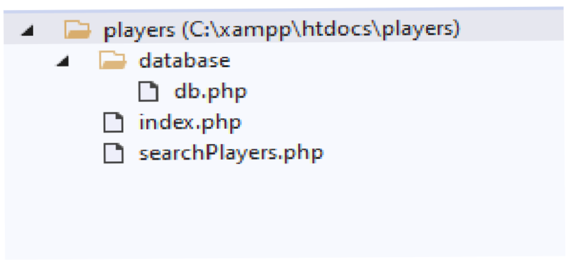
```
INSERT INTO players VALUES(1,'Frank Vogel');
INSERT INTO players VALUES(2,'Taylo Jnkins');
INSERT INTO players VALUES(3,'Jason Kidd');
INSERT INTO players VALUES(4,'Steve Nash');
INSERT INTO players VALUES(5,'Mike Budenholzer');
INSERT INTO players VALUES(6,'Doc Rivers');
INSERT INTO players VALUES(7,'Stan Van Gundy');
INSERT INTO players VALUES(8,'La Lakers');
INSERT INTO players VALUES(9,'Memphhis Grizzlies');
INSERT INTO players VALUES(10,'Dallas Mavericks');
INSERT INTO players VALUES(11,'Brooklyn Nets');
INSERT INTO players VALUES(12,'Philadelphia 76ers');
INSERT INTO players VALUES(13,'lebron');
INSERT INTO players VALUES(14,'anthony');
```

Figure 26 Create table players, Source: own work



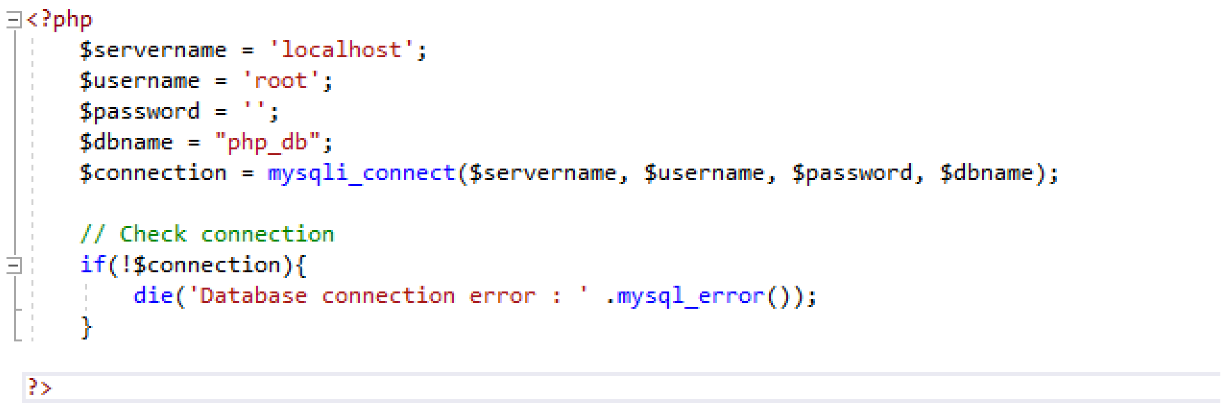
id	player_name
1	Frank Vogel
2	Taylo Jnkins
3	Jason Kidd
4	Steve Nash
5	Mike Budenholzer
6	Doc Rivers
7	Stan Van Gundy
8	La Lakers
9	Memphhis Grizzlies
10	Dallas Mavericks
11	Brooklyn Nets
12	Philadelphia 76ers
13	lebron
14	anthony

Figure 27 Table players, Source: own work



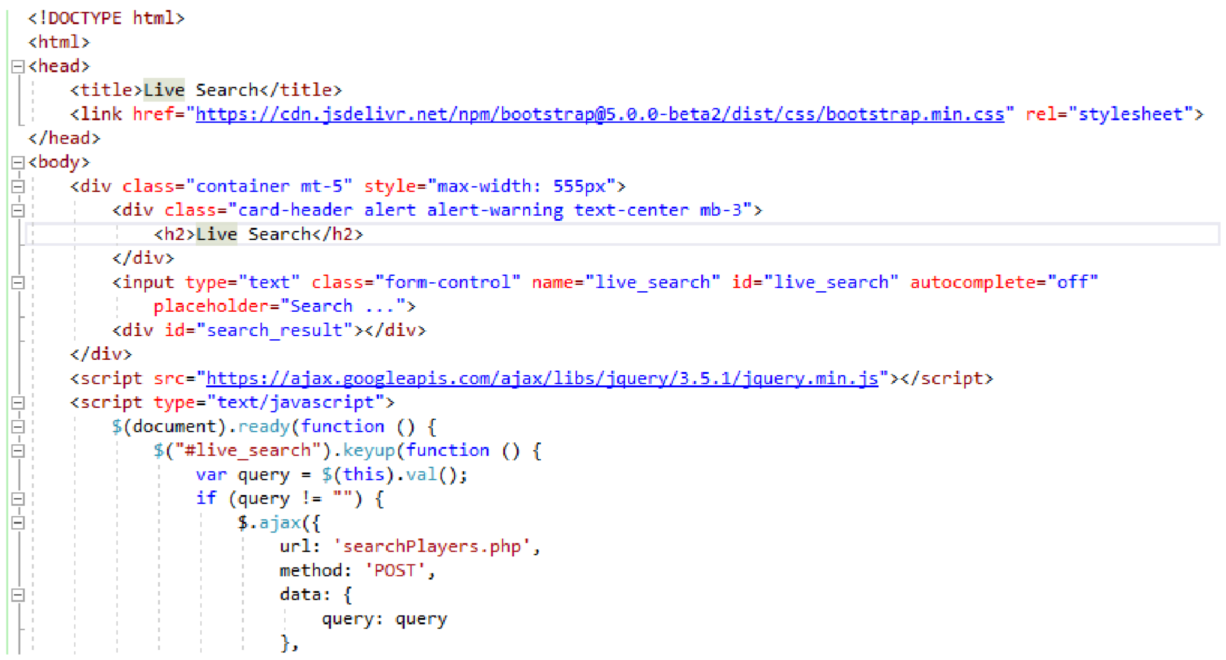
```
◢ players (C:\xampp\htdocs\players)
    ◢ database
          db.php
      index.php
      searchPlayers.php
```

Figure 28 Search Explorer, Source: own work

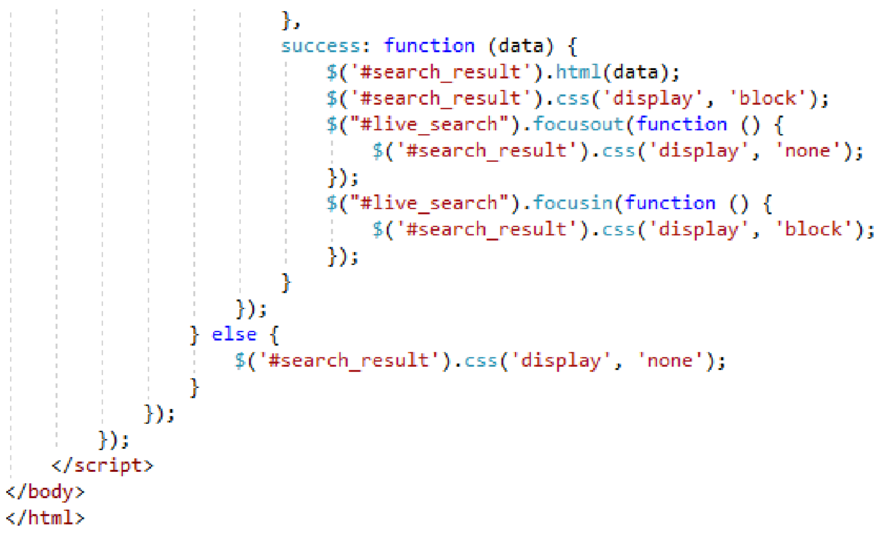


```php
<?php
    $servername = 'localhost';
    $username = 'root';
    $password = '';
    $dbname = "php_db";
    $connection = mysqli_connect($servername, $username, $password, $dbname);

    // Check connection
    if(!$connection){
        die('Database connection error : ' .mysql_error());
    }

?>
```

Figure 29 Db, Source: own work

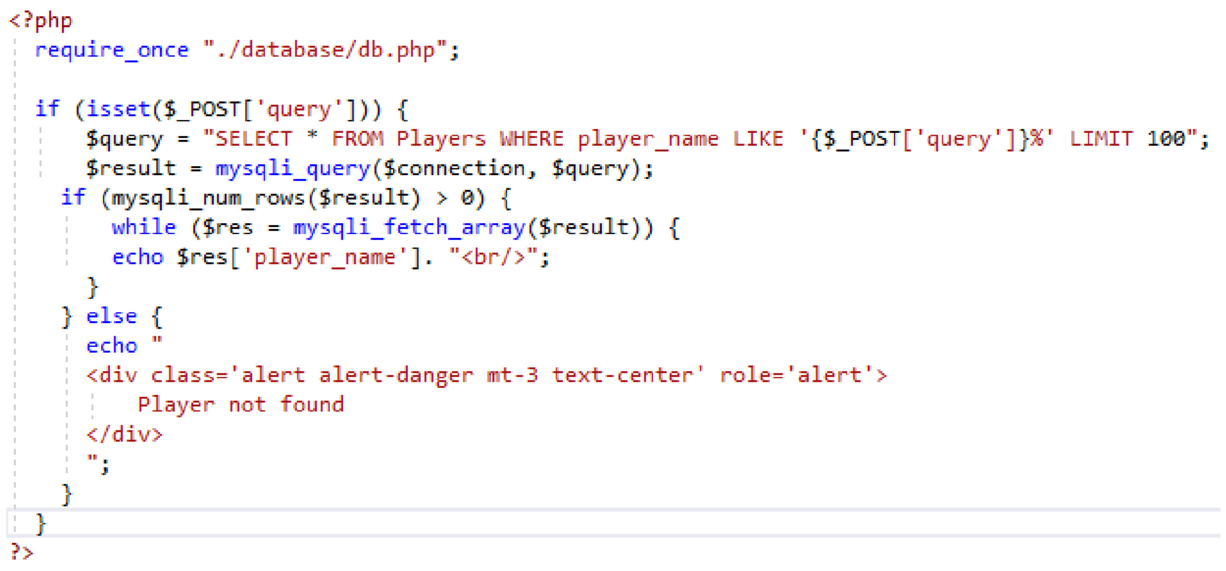


```html
<!DOCTYPE html>
<html>
<head>
    <title>Live Search</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <div class="container mt-5" style="max-width: 555px">
        <div class="card-header alert alert-warning text-center mb-3">
            <h2>Live Search</h2>
        </div>
        <input type="text" class="form-control" name="live_search" id="live_search" autocomplete="off"
            placeholder="Search ...">
        <div id="search_result"></div>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script type="text/javascript">
        $(document).ready(function () {
            $("#live_search").keyup(function () {
                var query = $(this).val();
                if (query != "") {
                    $.ajax({
                        url: 'searchPlayers.php',
                        method: 'POST',
                        data: {
                            query: query
                        },
```



```
                        },
                        success: function (data) {
                            $('#search_result').html(data);
                            $('#search_result').css('display', 'block');
                            $("#live_search").focusout(function () {
                                $('#search_result').css('display', 'none');
                            });
                            $("#live_search").focusin(function () {
                                $('#search_result').css('display', 'block');
                            });
                        }
                    });
                } else {
                    $('#search_result').css('display', 'none');
                }
            });
        });
    </script>
</body>
</html>
```

Figure 30 Index, Source: own work



```
<?php
  require_once "./database/db.php";

  if (isset($_POST['query'])) {
      $query = "SELECT * FROM Players WHERE player_name LIKE '{$_POST['query']}%' LIMIT 100";
      $result = mysqli_query($connection, $query);
    if (mysqli_num_rows($result) > 0) {
        while ($res = mysqli_fetch_array($result)) {
        echo $res['player_name']. "<br/>";
      }
    } else {
      echo "
      <div class='alert alert-danger mt-3 text-center' role='alert'>
          Player not found
      </div>
      ";
    }
  }
?>
```

Figure 31 searchPlayer, Source: own work

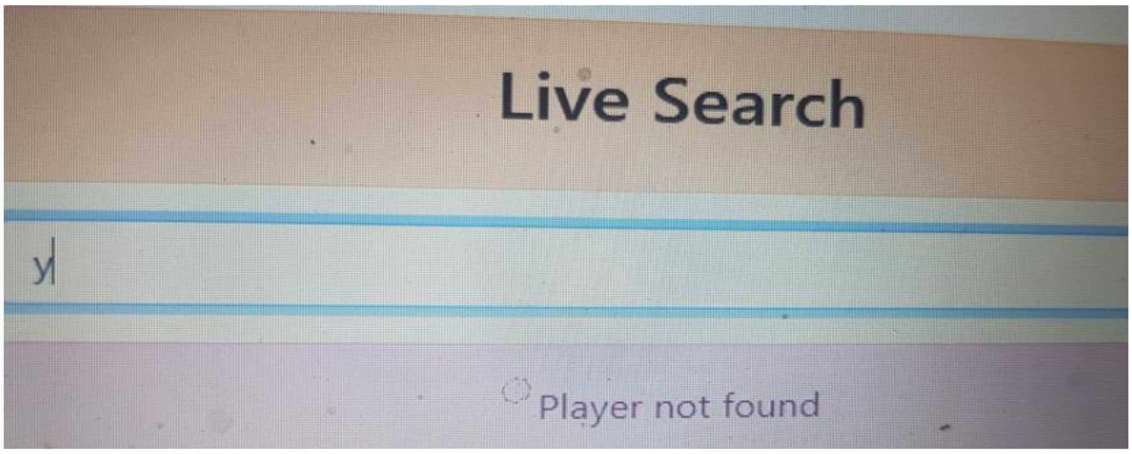
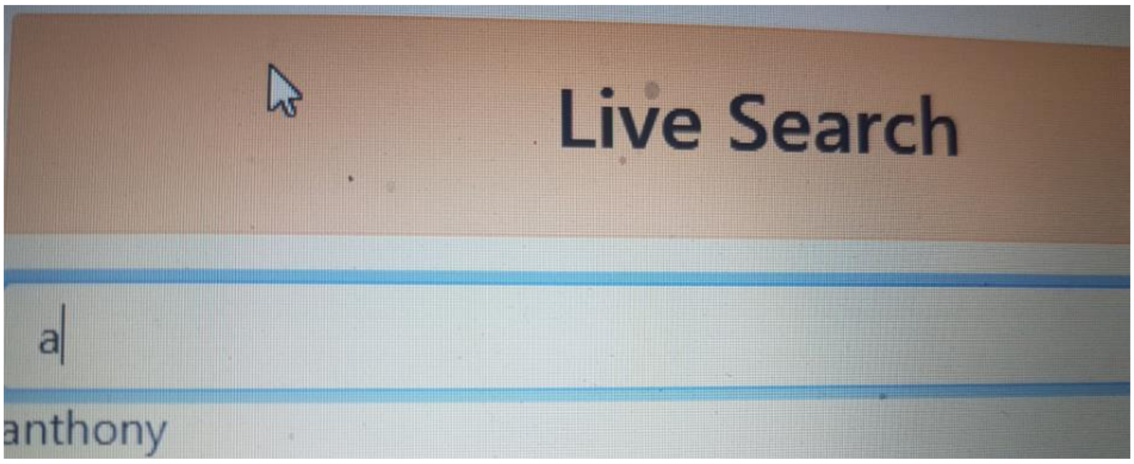


Figure 32 Live search, Source: own work



Live search 1, Source: own work

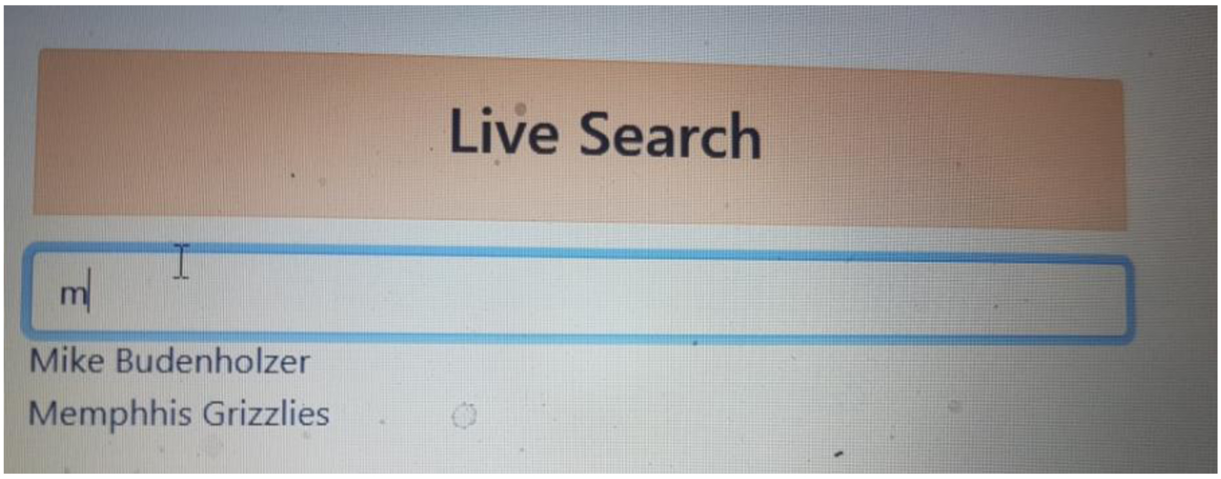


Figure 33: Live search 2, Source: own work

5 Conclusion

As a matter of fact, as for the representing the data a relational database as well as a graph database are ones of the best. However, as for the researching question which was to consider MySQL and Neo4j and find out which of them is able to deal with data better graph databases are somewhat better at storing and obtaining highly linked data. What is more, they are able to deal with complex relationships between data points much better. They enable to create entities to investigate relationships between data to make it easier to understand them because they are based on one table model. The table is called graph and it contains all information between entities and their relationships. In addition, graph databases seem to offer profoundly prime results. As a consequence, it is easier to gain the outcome by applying graph databases. Provided that an actor wants to add a new relationship, it is not necessary to reconstruct the database one more time. Finally, as to the time execution of all processes, the retrieval time is faster with graph databases. Therefore, graph databases are more suitable for commercial reasons, such as development of the social network.

.

6 References

Angles Renzo and Gutierrez Claudio. *Survey of graph database models*. ACM Computing Surveys (CSUR), 40(1):1–39 [online]. 2008 [cit. 2022-06-12].

Antoniou Grigoris. *A semantic web primer*. the MIT Press [online]. 2004 [cit. 2022-06-12].

Armstrong Timothy G, Ponnekanti Vamsi, Borthakur Dhruba, and Callaghan Mark. Linkbench: *a database benchmark based on the facebook social graph*. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pages 1185–1196. ACM [online]. 2013 [cit. 2022-06-12].

Beltrame Carlo. *Key-value stores: Algorithms for Database Systems, pages 1–12* [online]. 2013 [cit. 2022-06-12].

Berners-Lee Tim, Hendler James, Lassila Ora, et al. *The semantic web. Scientific American, 284(5):28– 37* [online]. 2001 [cit. 2022-06-12].

Carrington Peter J, Scott John, and Wasserman Stanley. *Models and Methods in Social Network Analysis*. Cambridge University Press [online]. 2005 [cit. 2022-06-12].

Cartwright Dorwin and Harary Frank. *Structural balance: a generalization of heider's theory*. Psycho logical review, 63(5):277 [online]. 1956 [cit. 2022-06-12].

Cattell Rick. *Scalable sql and nosql data stores*: Acm Sigmod Record, 39(4):12–27 [online]. 2011 [cit. 2022-06-12].

Codd Edgar F (1970). *A relational model of data for large, shared data banks*: Communications of the ACM [online]. [cit. 2022-06-12]

Codd Edgar F (1972). *The Relational Model for Database Management*: Version 2. Addison-Wesley Longman [online]. Publishing Co., [cit. 2022-06-12].

Codd Edgar F (1990). *Further normalization of the data base relational model*: Data Base Systems, pages 33–64 [online]. Publishing Co., [cit. 2022-06-12]

Dickinson Boonsri. Social media marketing [online]. 2012 [cit. 2022-06-12]. http://www.businessinsider.com/

Elmasri Ramez (2008). *Fundamentals of Database Systems* [online]. Pearson Education India, [cit. 2022-06-12].

explainer-what-exactly-is-the-social-graph-2012-3.

Freeman Linton C. *The Development of Social Network Analysis*. Empirical Press Vancouver [online]. 2004 [cit. 2022-06-12].

Granovetter Mark S. *The strength of weak ties*. American Journal of Sociology, pages 1360–1380 [online]. 1973 [cit. 2022-06-12]

Granovetter Mark S. *The strength of weak ties*: A network theory revisited. Sociological theory, 1(1):201– 233 [online]. 1983 [cit. 2022-06-12].

Graves Mark, Bergeman Ellen R, and Lawrence Charles B. *Graph database systems*. IEEE Engineering in Medicine and Biology Magazine [online]. 1995 [cit. 2022-06-12].

Haerder Theo and Reuter Andreas (1983). *Principles of transaction-oriented database recovery*: ACM Com puting Surveys (CSUR) [online]. [cit. 2022-06-12].

Han Jing, Haihong E, Le Guan, and Du Jian. *Survey on nosql database*. In Pervasive computing and applications (ICPCA): Decision Support Systems, 42 [online]. 2011 [cit. 2022-06-12].

Hanneman Robert A and Riddle Mark. *Introduction to social network methods* [online]. 2005 [cit. 2022-06-12]

Hanneman Robert A and Riddle Mark. *Social network analysis*. Riverside: University of California, [online]. 2001 [cit. 2022-06-12].

Harary Frank. *Structural models: An introduction to the theory of directed graphs*. [online]. 2005 [cit. 2022-06-12].

Hecht Robin and Jablonski Stefan. *Nosql evaluation*. In International Conference on Cloud and Service: Computing, pages 336–41 [online]. 2011 [cit. 2022-06-12].

Iordanov Borislav. *Hypergraphdb: a generalized graph database. In International Conference on Web Age Information Management*, pages 25–36. Springer [online]. 2010 [cit. 2022-06-12].

Jeong Hoe Jin and Lee Sang Ho. *An integrated database benchmark suite*. In 2005 First International Conference on Semantics, Knowledge and Grid, pages 60–60. IEEE [online]. 2005 [cit. 2022-06-12].

Jeong Hoe Jin and Lee Sang Ho. *Survey of graph database performance on the hpc scalable graph analysis benchmark*. In International Conference on Web-Age Information Management, pages 37–48. Springer [online]. 2010 [cit. 2022-06-12].

Khan W. Predictive Performance Comparison Analysis of Relational & NoSQL Graph Databases [online]. 2017 [cit. 2022-06-12].

Klein Hans K and Daniel Lee Kleinman. *The social construction of technology*: *Structural considerations*. Science, Technology & Human Values, 27(1):28–52 [online]. 2002 [cit. 2022-06-12]

Klyne Graham, Carroll Jeremy J, and McBride Brian. *Resource description framework (rdf): Concepts and abstract syntax.* W3C Recommendation,10 [online]. 2004 [cit. 2022-06-12]

Korhan, Jeff. *What your business needs to know about social graphs* [online]. 2011 [cit. 2022-06-12].

Krackhardt David. *The strength of strong ties: The importance of philos in organizations*. Networks and Organizations: Structure, Form, and Action [online]. 1992 [cit. 2022-06-12].

MashaghAR i, Ramezanpour Abolfazl, and Karimipour V. *Investigation of a protein complex network*. The European Physical Journal B-Condensed Matter and Complex Systems, 41(1):113–121 [online]. 2004 [cit. 2022-06-12]

McCreary Dan and Kelly Ann. *Making sense of nosql*: Shelter Island: Manning, pages 19–20 [online]. 2014 [cit. 2022-06-12].

Moniruzzaman ABM and Hossain Syed Akhter (2013). *Nosql database*: new era of databases for big data analytics-classification, characteristics and comparison. [online]. [cit. 2022-06-12]

Nelson Reed E. *The strength of strong ties: Social networks and intergroup conflict in organizations*. Academy of Management Journal [online]. 1989 [cit. 2022-06-12].

Pan Jeff Z. *Resource description framework*. In Handbook on Ontologies, pages 71–90. Springer [online]. 2009 [cit. 2022-06-12].

Paredaens Jan, Bra Paul De, Gyssens Marc, and Dirk Van Gucht. *The structure of the relational database mode*l, volume 17. Springer Science & Business Media [online]. 2012 [cit. 2022-06-12].

Paredaens Jan, Bra Paul De, Gyssens Marc, and Gucht Dirk Van. *The structure of the relational database model, volume 17*. Springer Science & Business Media [online]. 2012 [cit. 2022-06-12].

Ports Dan RK, Clements Austin T, Zhang Irene, Madden Samuel, and Liskov Barbara (2010). *Transactional consistency and automatic management in an application data cache*: In OSDI, volume 10, pages 1–15, [online]. [cit. 2022-06-12].

Powers Shelley *Practical RDF. O'Reilly* [online]. 2009 [cit. 2022-06-12].

Rapoport Anatol *Contribution to the theory of random and biased nets*. The Bulletin of Mathematical Biophysics [online]. 1957 [cit. 2022-06-12]

Robinson Ian, Webber Jim, and Eifrem Emil. *Graph Databases: New Opportunities for Connected Data*. " O'Reilly Media, Inc." [online]. 2005 [cit. 2022-06-12].

Ruef Martin. *Strong ties, weak ties and islands: structural and cultural predictors of organizational innovation*. Industrial and Corporate Change [online]. 2002 [cit. 2022-06-12]

Sasaki Bryce Merkl and Graphista Aspiring. *Graph databases for beginners*: Why we need nosql databases [online]. 2016 [cit. 2022-06-12].

Scott John and Carrington Peter J. *The SAGE Handbook of Social Network Analysis*. SAGE publications [online]. 2011 [cit. 2022-06-12]

Scott John. *Social network analysis*. Sociology, 22(1):109–127 [online]. 1988 [cit. 2022-06-12].

Shin Seung Kyoon and Sanders G Lawrence. *Denormalization strategies for data retrieval from data warehouses*: Decision Support Systems, 42 [online]. 2006 [cit. 2022-06-12].

The Neo4j Getting Started Guide v4.0 [online]. [cit. 2022-06-12].

Wasserman Stanley. *Social Network Analysis: Methods and Applications, volume 8. Cambridge univer sity press* [online]. 1994 [cit. 2022-06-12].

Watts Duncan J and Strogatz Steven H. *Collective dynamics of small-world networks. nature, 393(6684):440–442* [online]. 1998 [cit. 2022-06-12]

West Douglas Brent et al. *Introduction to Graph Theory, volume 2. Prentice hall Upper Saddle River*, [online]. 2001 [cit. 2022-06-12]

Zaich Paul. *Relational databases and deconstructing the learning process* [online]. 2012 [cit. 2022-06-12].