RFM Analysis Using Neo4j

Saurav Kant Kumar
skumar66@hawk.iit.edu
Illinois Institute of Technology

Abstract

Graph databases represent a paradigm shift from relational databases with a strong support for "relationships". As compared to relational databases which compute relationships at runtime, graph databases persist relationships for fast querying and data retrieval. This work presents RFM analysis on E-commerce data using a graph database, Neo4j application. Given the historical transaction data, this application shows Recency - How recently did the customer purchase?, Frequency: How often do they purchase?, and Monetary Value: How much do they spend? In order to implement this application, transaction data have been taken from UCI Machine Learning repository. The data has been inserted into the Neo4j database and subsequently relationships between Customer, Transaction, and Product nodes have been analyzed. Based on the type of customer, we can recommend certain actions like giving special offers or loyalty points.

1. Introduction

[RWE15]Graph databases are an important member of the NoSQL family of databases. These are primarily a combination of nodes and edges where the nodes denote entities and the relationship amongst the nodes are denoted by the directed edges between nodes. Relationships are given the highest significance in graph databases as compared to other database models, such as the relational database management systems (RDBMS). This implies that graph databases do not require to use foreign keys or references for computing relationships at the time of execution, as in RDBMSs. Since the operations are based on set operations in relational databases, this leads to unexpected latency or memory usage in query execution for the RDBMS. As the size of data grows, the RDBMS operations get slower. However graph databases operate on links or paths and hence are more efficient even for large data sets.

It is cumbersome to represent relationships for connected data in other NoSQL databases since these utilize disconnected aggregates for storing data sets. Therefore, expensive "JOIN" operations may be required which are known to be time-inefficient. Graph databases offer simplicity as well as expressiveness in comparison with RDBMS or other NoSQL families.

The current work has utilized Neo4j[GSB17], an open-source graph database which offers strong support for applications involving connected or linked data. Neo4j has found a number of applications, such as social network analysis, network monitoring and recommender systems etc. This work presents a novel application for RFM analysis on E-commerce based on Neo4j database.

It is important for any business to segment its customers into different categories like 1) Customers who spend the most, buy frequently, and have bought recently 2) Customers who spend much, buy frequently, but there is no recent purchase 3) Customer who spend little, buy occasionally, but there is no recent purchase. 4) Customer who spend little, buy occasionally, and there is a recent purchase. However, the volume and complexity of E-commerce are also tremendous and present a big data challenge. Therefore, in order to store the large volume of the Transaction data, the application has used Neo4j database. Other NoSQL databases like MongoDB and Cassandra do not handle relationship very easily and features similar to foreign key have to be used explicitly to implement it.

The paper is structured as follows. Section 2 presents the background and related work in this domain. Section 3 describes comparative study of Graph Databases. Section 4 Data collection. Section 5 describes the Exploratory Data Analysis. Section 6 describes Data Insertion. Section 7 describes the Data Profiling. Section 8 describes RFM analysis. Section 9 describes Results and Analysis. Section 10 describes milestones.

2. Related Work and Background

A. Related Work

As per the existing literature, apparently there seems a rise in the requirement of a flexible schema. Hence, Relational Database Management System is losing its importance in the context of current big data applications[BT12]. Further, in relational database management systems, the joining of large number of tables leads to a decreased efficiency whereas Graph database is meant for overcoming these kinds of shortcomings.

As per a recent study, the Relational databases are slowly getting replaced by Graph Databases[Mil13]. Graph databases find their applications in wide variety of domains like recommendation engines, semantic web and social networking to list a few. RDBMSs and graph databases such as Neo4J signify their differences in terms of features of the data model, and query structures etc.

As far as information storage is concerned, the Graph databases deals with a fine manner for modelling and traversing the linked data. Applications can be implemented in the Graph database systems, for example Neo4J, independent of the limitations of their counterpart RDBMS.

As per existing literature, the crucial census data in a graph database is stored as graph structure. Here, the nodes acts as persons and the edges represent relationships amongst these nodes. The interpretation observed is that the amount of time taken to process a given query in a graph database is impressive and the graph database is more capable of retrieving relationships over the relational and other NoSQL databases. It is further observed that specifically while handling the highly interconnected data, the graph database provide better performance as compared to the relational and NoSQL databases [YT14].

According to recent studies, the set of predefined queries executes faster in comparison with the relational databases. Further to this, the graph databases leverages more flexibility in terms of restructuring the schemas. This observation proves to be helpful while implementation of commercial systems in different domains [GSB17].

There is a work in the existing state of art, where recipe is designed based on the ingredients which are at hand available to the user which leads to a direction of decreased food wastage.Such a system bears wide future scope[Lim+14].

Graph databases are a good source for the recommendation of the test cases to generate fruitful results. In specific, the relationship amongst nodes and the integration of entities leads to a unique graph structure which promotes design recommendations as simple graph traversals[Pel17].

B. Background

In Retail sector, various chain of hypermarkets are generating exceptionally large volume of data on a daily basis across the stores. This transactions data can be analyzed to identify different segments of customers, which in turn will help design profitable strategies.

Customer segmentation is a method of dividing customers into groups or clusters on the basis of common characteristics. The market researcher can segment customers into the B2C model using various customer's demographic characteristics such as occupation, gender, age, location, and marital status. Psychographic characteristics such as social class, lifestyle and personality characteristics and behavioral characteristics such as spending, consumption habits, product/service usage, and previously purchased products. In the B2B model using various company's characteristics such as the size of the company, type of industry, and location.

The are four different types of Customer segmentation 1. Demography based 2. Geography based 3. Behavior based 4. Psychography based

RFM (Recency, Frequency, Monetary) analysis is a behavior-based approach grouping customers into segments. It groups the customers on the basis of their previous purchase transactions. How recently, how often, and how much did a customer buy. RFM filters customers into various groups for the purpose of better service. It helps managers to identify potential customers to do more profitable business. There is a segment of customer who is the big spender but what if they purchased only once or how recently they purchased? Do they often purchase our product? Also, It helps managers to run an effective promotional campaign for personalized service.

Recency (R): Who have purchased recently? Number of days since last purchase (least recency)
Frequency (F): Who has purchased frequently? It means the total number of purchases. (high frequency)
Monetary Value(M): Who have high purchase amount? It means the total money customer spent (high monetary value)

Here, Each of the three variables(Recency, Frequency, and Monetary) consists of three equal groups, which creates 27 (3x3x3) different customer segments.

3. Comparative Analysis of Graph Databases

We have considered Neo4j, Amazon Neptune, Apache GraphX for comparative analysis.

A. Description and Use Cases

Neo4j is an open-source graph database, which employs a native graph data store built from the ground up, to leverage not only data but also data relationships and a market leader in current industry. It has a vast platform built around it. Platform includes The Neo4j Graph Data Science Library – the leading enterprise-ready analytics workspace for graph data which includes Graph Algorithms, the graph visualization and exploration tool called Bloom, the Cypher query language, and numerous tools like Browser, integrations, and connectors are all included in the platform to help developers and data scientists quickly build graph-based solutions. It has fueled a slew of game-changing corporate applications in fraud detection, financial services, life sciences, data science, knowledge graphs, and more.
Reference: https://neo4j.com/product/

Amazon Neptune is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets. Amazon Neptune is purpose-built for storing billions of relationships and querying the graph with milliseconds latency. Amazon Neptune is compatible with open graph APIs and supports popular graph models Property Graph and W3C's RDF, and their respective query languages Apache TinkerPop Gremlin and SPARQL. While graph databases usually require extensive hardware management, provisioning, and manual scaling, Amazon Neptune is a fully managed service, without overhead of database management tasks. Deployment can be done in a matter of minutes, with a few clicks in the AWS management console or with the AWS CLI. Amazon Neptune in combination with other tools like Amazon Neptune ML, AWS Graph Notebook makes it easier to implement use cases like fraud detection, identity resolution, knowledge graph and product recommendations, drug discovery, and network security.
Reference: https://aws.amazon.com/neptune/fraud-graphs-on-aws/?pg=ln&sec=uc

GraphX is Apache Spark's API for graphs and graph-parallel computation (graph processing system). It unifies ETL, exploratory analysis, and iterative graph computation within a single system as it is built on top of Apache Spark-a unified analytics engine for Big Data processing. The data can be viewed as both graphs and collections. It provides capability to transform and join graph with RDDs efficiently and write a custom iterative graph algorithm using Pregel API. Like Neo4j's Graph Data Science Library, GraphX also has a growing library of graph algorithms contributed by the community in addition of being a highly flexible API which supports following use case with ease: PageRank, connected components, label propagation, SVD++, Strongly connected components, Triangle count. Currently GraphX is in alpha stage and since its part of Apache's Spark API it is open source as well.
Reference: https://spark.apache.org/graphx/

B. Features

Neo4j supports lightspeed throughput by relying on distributed high-performance architecture which is fault-tolerant and guarantees data integrity(ACID Compliant).Furthermore, Neo4j Fabric allows it to provide a limitless horizontal scaling without compromising performance using Sharding(Operate over a single large graph) and Fedearation(Query across disjointed graph) . While sharding divides graphs, federation enables queries across disjointed graphs by bringing multiple graphs together. It helps Neo4j achieve High Availability, no single point of failure (fault tolerant), unified view of local and distributed data at a place, increased scalability of read/write operations, data volume and concurrency and predictable response time for query execution, a failover and other infrastructure changes. It supports self-hosted, hybrid, multi-cloud, or our fully managed cloud service.

Security wise neo4j has built-in enterprise level security features like Identity and access control, encrypted communication protocols, Schema- based security to define role-based restrictions and Data Privacy.
Reference: https://neo4j.com/product/neo4j-graph-database/

Amazon Neptune has high throughput low latency for graph queries. Also, Low latency read replicas as well. Easy Scaling of Database Compute Resources and Storage that Automatically Scales. It is also highly available and underlying infrastructure is also durable with features like Instance Monitoring and Repair, Multi-AZ (Availability Zone) Deployments with Read Replicas, Fault-tolerant and Self-healing Storage, Automatic, Continuous, Incremental Backups and Point-in-time Restore, user has ability to take Database Snapshots manually and create a new instance with it whenever required. It also supports RDF and Property graph bulk loading from S3 storage.

Security wise it has features like Network Isolation using Amazon VPC, Resource-Level Permissions using AWS Identity and Access Management (IAM) over actions on specific Neptune resources including Database Instances, Database Snapshots, Database Parameter Groups, Database Event Subscriptions, and Database Options Groups. Encryption of data at rest, automated backups, snapshots, and replicas using AWS Key Management Service (KMS).

Advanced Auditing by logging database events without impacting performance which can later be analyzed using Amazon CloudWatch for various auditing purposes.
Reference: https://aws.amazon.com/neptune/features/

GraphX competes on performance with the fastest graph systems while retaining Spark's flexibility, fault tolerance, and ease of use.

C. Comparison

	Neo4j	Amazon Neptune	Apache GraphX
Software Version	Community/Enterprise	Enterprise	Community but in alpha stage. Not production ready
Query Language	CQL	TinkerPop Gramlin, SPARQL	Python, Java, Scala
Distributed	Yes	Yes	Yes
Fault Tolerant	Yes	Yes	Yes
High Availability	Yes with Sharding and Federation	Yes with replicas and availability zone(AZ)	Yes
Runs on	Linux, Windows, Mac ,Container	AWS cloud	Linux, Windows, Mac ,Container
Security	Built in (encryption provided by third party)	Built in using AWS security products	Built in but must be enabled during deployment by administrator
Performance	Read Replicas, Unbounded Architecture, and Fabric for limitless horizontal scaling without impacting performance	Compute Resource heavy	Inherits Sparks performance
SAAS Support	Yes	Yes	Yes via third party
Self Hosted	Yes	No	Yes

Figure 1. Comparison of different Graph Database

Figure 1 shows comparison of Neo4j, Amazon Neptune, and Apache GraphX based on various parameters.

D. Conclusion

While Amazon Neptune guarantees zero database administration unlike community version of Neo4j it has single point of failure that is AWS cloud services. If AWS cloud service goes down, there will be outage which we have seen to happen. While Neo4j guarantees zero downtime by relying on its distributed architecture pattern used. Even Neo4J has got its own fully managed version like Amazon Neptune called Aruba DB. The query language used by Neo4J called cypher is very easy to learn unlike TinkerPop Gremlin and SPARQL used in Amazon Neptune. Moreover, Cypher is foundation for GQL a W3C standard for graph database query language. Since Amazon Neptune uses outdated standard for query language which will make it harder to get help from community in case of blockage faced while development or maintenance. At present implementing a use-case in Neo4J requires less Lines of Code than Amazon Neptune since Neo4j has vibrant community that has built vast libraries to achieve same so your application will be up and running in less amount of time than in case of Amazon Neptune. There is also a play of vendor lock in case of Amazon Neptune since the code will be written around its DB engine and there is no way to self-host or support for hybrid cloud where as Neo4j is much more portable. Also, Neo4j provides granular security features which is not present in Amazon Neptune. Although encryption is built in Amazon Neptune same can be achieved in case of Neo4j using Thales integration in enterprise version. GraphX is not a graph database. While Amazon Neptune and Neo4j are used to analyze connected nodes and properties in a graph, they allow exploration and discovery from a known starting point, GraphX is used to investigate the graph's nature or topological properties, allowing us to explore and discover links that were previously unknown when the graph was created. Hence GraphX can be used in compliment to graph databases. Mazerunner is a plugin for Neo4J that allows it to be used along with GraphX. We came to conclusion that Neo4j is strides ahead of Amazon Neptune in various features that we compared, and it can be used along with GraphX.

4. Data Collection

The data is taken from UCI Machine Learning repository. It contains 541909 records and 8 features corresponding to 4372 customers. The features are: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country. The transaction data is for the duration of jan 2010 to dec 2011. Each record corresponds to one item purchased by a customer. If a customer buys multiple products at once, each product will appear in separate rows with the same InvoiceNo and CustomerID.

5. Exploratory Data Analysis

The data belongs to 38 different countries. But majority of the data (approx. 91 percent) comes from United Kingdom. For our analysis we have considered all the data from every country.

Figure 2 shows the number of customers for top 10 countries. Clearly the maximum number of customer(around 4000) are from United Kingdom. Germany and France has around 30 customers. Rest of the countries have negligible number of customers.

6. Data Insertion

The data was inserted into Neo4j using Cypher queries. Three different nodes were created, and were labeled as "customer", "product", and "transaction". Two different kinds of relationships were established, which were

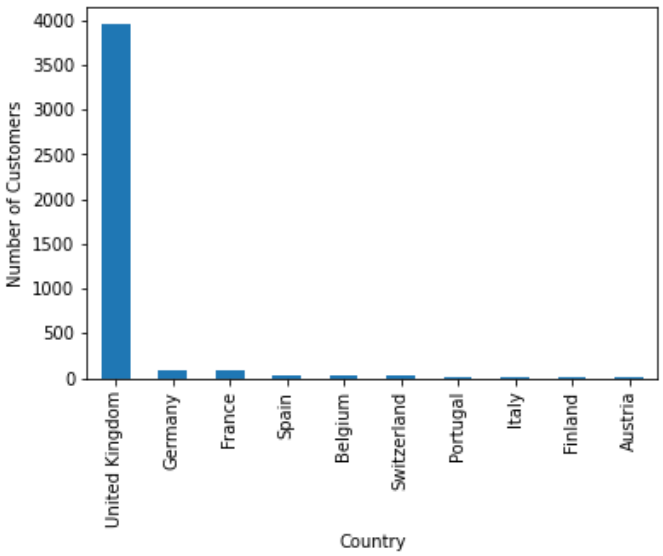

Figure 2. Data Distribution

"MADE" and "CONTAINS". The "MADE" relationship is between customer and transaction nodes. "CONTAINS" relationship is between transaction and product nodes.

A total of 4372 nodes of customer, 4070 nodes of product, 25900 nodes of transaction were created. A total of 22190 "MADE" relationship, 531225 "CONTAINS" relationship were created.

Figure 3 represents all the different types of nodes

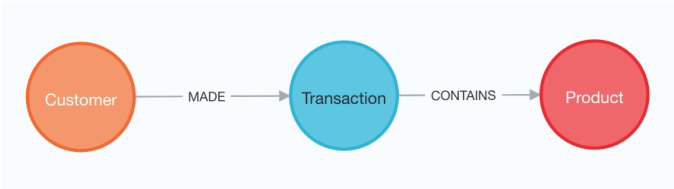

Figure 3. Nodes and Relationships present in Schema

and relationships that exist in the database schema.

Figure 4, 5, and 6 represent sample structure of each of three types of nodes.

7. Data Profiling

The execution has been calculated on mac system with Intel i5 processor and 8GB RAM.

Figure 7 represents node creation time for Customer, Product, and Transaction nodes in Neo4j.


```
{
  "identity": 0,
  "labels": [
    "Customer"
  ],
  "properties": {
"customerID": 17850,
"country": "United Kingdom"
  }
}
```


Figure 4. Attributes of Customer Node


```
{
  "identity": 4372,
  "labels": [
    "Product"
  ],
  "properties": {
"description": "WHITE HANGING HEART T-LIGHT HOLDER",
"stockCode": "85123A"
  }
}
```


Figure 5. Attributes of Product Node


```
{
  "identity": 8442,
  "labels": [
    "Transaction"
  ],
  "properties": {
"transactionDate": "12/1/2010 8:26",
"transactionID": "536365"
  }
}
```


Figure 6. Attributes of Transaction Node

Figure 8 represents relationship creation time for Made, and Contains relationships in Neo4j

Execution time of Cypher query present in Python script is around 2 seconds.

8. RFM Analysis

A python script was created to fetch relevant information from the Neo4j database using Cypher query. For each customer recency, frequency, and monetary values were cal-

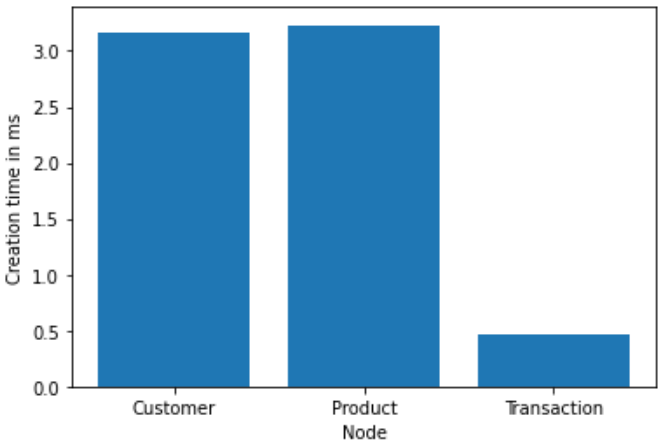

Figure 7. Performance on creating Nodes in ms.

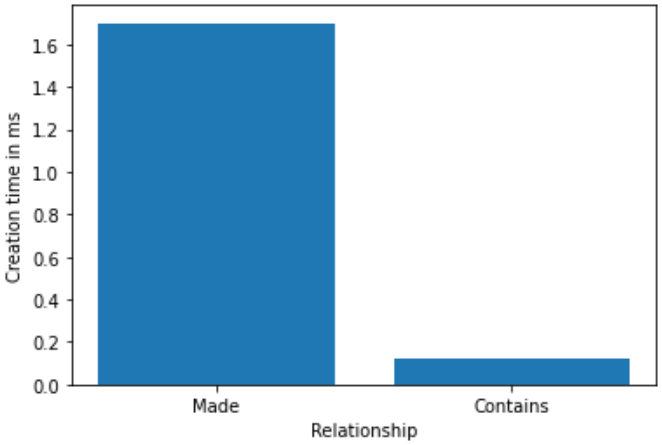

Figure 8. Performance on creating Relationships in ms.

culated in the query.

```
MATCH (c:Customer)-[r1:MADE]->(t:Transaction)-[r2:CONTAINS]->(:Product)
WITH SUM(r2.price) AS monetary,
  COUNT(DISTINCT t) AS frequency,
   c.customerID AS customer,
   MIN(
    duration.inDays(
    date(datetime({epochmillis: apoc.date.parse(t.transactionDate, 'ms', 'MM/dd/yyyy')})),
    date()
   ).days
  ) AS recency
RETURN customer, recency, frequency, monetary
```

Figure 9. Query 1

Figure 9 represents the cypher query used in python script. Monetary value is calculated by summing the price of all products for each customer across every transaction. Frequency value is calculated by counting number of times a person has made a purchase. Recency value is number of days between last transaction date of the customer and current date. But since the data is between Jan 2010 and Dec 2011, it makes sense to update the recency value by subtracting the minimum recency value in the dataframe from each recency value.

The customers are segmented into 27 different groups based on the recency, frequency, and monetary value. Out of these 27 different segments we are considering only following four segmants:

1. Best Customers - Customers with recency=3, frequency=3, monetary=3
2. No recent purchase Customers - Customers with recency=1, frequency=3, monetary=3
3. Low Loyalty Customers - Customers with recency=1, frequency=1, monetary=1
4. New Customers - Customers with recency=3, frequency=1, monetary=1

Here, recency = 3 represents the most recent customer and recency = 1 represents the least recent customer. frequency = 3 represents most frequent customer and frequency = 1 represents least frequent customer. monetary = 3 represents those customer who spent maximum amount and monetary = 1 represents those customer who spent least amount of money.

9. Results and Analysis

Best customers:

	recency	frequency	monetary
count	725.00	725.00	725.00
mean	8.62	15.61	7057.82
std	6.90	18.66	19078.76
min	0.00	5.00	1146.96
25%	3.00	7.00	1959.38
50%	8.00	11.00	3147.31
75%	14.00	17.00	5452.17
max	24.00	248.00	279489.02

Figure 10. Statistics for Best Customers

Figure 10. illustrates that for Best Customers mean recency is 8.62 days, mean frequency is 15.61 days, mean monetary value is 7057 USD. Since most of the revenue is generated by high value customers it is important to provide offers to these customers if they do not buy anything for 24 days(maximum value of recency)

Figure 11. illustrates that for Customers with no recent purchase, mean recency is 145.44 days, mean frequency is 7.73 days, mean monetary value is 2703.51 USD. These are the customers who buy rarely from the store but they buy in bulk and spend huge amount of money to the store. These

No purchases recently:

	recency	frequency	monetary
count	70.00	70.00	70.00
mean	145.44	7.73	2703.51
std	56.63	4.24	1693.39
min	91.00	5.00	1143.27
25%	103.75	5.00	1564.56
50%	126.00	6.00	2097.25
75%	169.00	9.00	3327.58
max	330.00	35.00	10217.48

Figure 11. Statistics for no recent purchase Customers

customers are slightly less valuable than Best Customers. So if the customer does not buy for 330 days the store can provide some lucrative offer to them.

Low loyalty:

	recency	frequency	monetary
count	746.00	746.00	746.00
mean	239.35	1.22	175.15
std	82.16	0.41	220.64
min	91.00	1.00	-4287.63
25%	172.00	1.00	115.69
50%	241.00	1.00	179.63
75%	304.00	1.00	276.52
max	373.00	2.00	374.57

Figure 12. Statistics for Low Loyalty Customers

Figure 12. illustrates that for Low loyalty Customers mean recency is 239.35 days, mean frequency is 1.22 days, mean monetary value is 175.15 USD. These are the customers who come rarely to the store and spend negligible amount of money. These customers can be given little discount for a shorter duration of time.

Figure 13. illustrates that for New Customers mean recency is 13.28 days, mean frequency is 1.43 days, mean monetary value is 216.78 USD. These customers have recently started coming to the store, so it is the best time to gain their trust by giving small offers on the items that are never bought before.

10. Milestones

1. Business Understanding - To understand the problem we are trying to solve
2. Data Understanding - To analyse the data in order to understand the features and patterns in the data

New customers:

	recency	frequency	monetary
count	207.00	207.00	207.00
mean	13.28	1.43	216.78
std	7.00	0.50	93.77
min	1.00	1.00	-17.45
25%	8.00	1.00	159.10
50%	14.00	1.00	216.30
75%	19.00	2.00	298.91
max	24.00	2.00	373.75

Figure 13. Statistics for New Customers

3. Designing Cypher Queries - To design the cypher queries to create nodes and relationships and insert data into neo4j database.
4. Phython Code - To connected neo4j database with Python in order to derive features like recency, frequency, and monetary value. Derive 27 different segements of customers.
5. Identify segments of customers - To derive insights about different kinds of customers. So that lucrative offers can be designed for different customer segment.
6. Comparative Study of different Graph Databases
7. Exploratory Data Analysis
8. Data Profiling

References

[BT12] Shalini Batra and Charu Tyagi. “Comparative analysis of relational and graph databases”. In: *International Journal of Soft Computing and Engineering (IJSCE)* 2.2 (2012), pp. 509–512.

[Mil13] Justin J Miller. “Graph database applications and concepts with Neo4j”. In: *Proceedings of the southern association for information systems conference, Atlanta, GA, USA*. Vol. 2324. 36. 2013.

[Lim+14] Veranika Lim et al. “Design implications for a community-based social recipe system”. In: *World Congress on Sustainable Technologies (WCST-2014)*. IEEE. 2014, pp. 19–26.

[YT14] Kay Thi Yar and Khin Mar Lar Tun. “Data relationship query in relational db, nosql db and graph db”. In: *International Journal of Computer Applications* 105.17 (2014).

[RWE15] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases: new opportunities for connected data*. ” O’Reilly Media, Inc.”, 2015.

[GSB17] José Guia, Valéria Gonçalves Soares, and Jorge Bernardino. “Graph Databases: Neo4j Analysis.” In: *ICEIS (1)*. 2017, pp. 351–356.

[Pel17] Joseph Pellegrino. “Flexible recommender systems based on graphs”. In: *AISR2017*. 2017.