|  |  |  |  |
| --- | --- | --- | --- |
| **CSY3024: DATABASES 3** | | | |
| Date of submission: | **3rd May 2020** | Module Tutor: | Santosh Khanal |
| Student Name: | Sudish Basnet | | |
| Student ID: | 18413692 | | |
| Tutor Comments |  | | |

Contents

[Report Introduction 1](#_Toc39336217)

[Part 1 2](#_Toc39336218)

[Introduction to Part1 2](#_Toc39336219)

[Overall Reflections 2](#_Toc39336220)

[Problems faced during exercises 2](#_Toc39336221)

[Knowledges Gained 3](#_Toc39336222)

[Skills to improve 3](#_Toc39336223)

[Part 2 3](#_Toc39336224)

[Introduction to Part 2 3](#_Toc39336225)

[Database Design 4](#_Toc39336226)

[Queries 4](#_Toc39336227)

[Part 2 Reflection 8](#_Toc39336228)

[Part 3 9](#_Toc39336229)

[Acknowledgment 14](#_Toc39336230)

[Appendix 15](#_Toc39336231)

[Week 13 15](#_Toc39336232)

[Activities 15](#_Toc39336233)

[Week 14 19](#_Toc39336234)

[Activities 19](#_Toc39336235)

[Reflections 26](#_Toc39336236)

[Week 15 26](#_Toc39336237)

[Activities 26](#_Toc39336238)

[Reflections 38](#_Toc39336239)

[Week 16 38](#_Toc39336240)

[Activities 38](#_Toc39336241)

[Reflection: 39](#_Toc39336242)

[Week 17 40](#_Toc39336243)

[Activities 40](#_Toc39336244)

[Reflection 51](#_Toc39336245)

[Week 18 and 19 51](#_Toc39336246)

[Activities 51](#_Toc39336247)

[Reflection 54](#_Toc39336248)

[Week 20 54](#_Toc39336249)

[Activities 54](#_Toc39336250)

[Reflection 56](#_Toc39336251)

[Week 21 57](#_Toc39336252)

[Activities 57](#_Toc39336253)

[Reflection 59](#_Toc39336254)

[References 59](#_Toc39336255)

# Report Introduction

This is the second assessment provided for Database 3 module by the Northampton University . In this assessment report, it consists of three parts where part 1 covers importation about learning diary reflection for which the codes that have been run practically will be reflected in appendix with the diary log of each task problems that were faced with the steps taken for them. The part 2 consist of assessment task with 6 queries which needs to be solved with mongo dB which needs to be completed after the development of skill through lectures and workshop. The part 3 is basically essay writing covering the topic of “Bigdata analytics and the role of NoSQL databases”.

# Part 1

## Introduction to Part1

In this section all the experience that have been collected during practical work will be reflected with overall reflection of every task in each week. All the task is done properly with 100% accuracy beside some flaws due to being new to mongo DB. The problems faced and knowledge gained will be discuss below in each section of Part 1.

## Overall Reflections

After the completion of every activities which are provided every week and solved which can be found below in appendix with every activity’s solution and their reflections individually. We can reflect that though relational databases are in use for 30 years now with the fact that it’s been giving great performance and flexibility but the major factor which contributed for development of NoSQL DBs like MongoDB is the rapid growth of MEAN stack development, where the developers works from front-end with JavaScript and with database and backend.

The motivation for using MongoDB is because of its pros like data store that provides high performance, automatic scaling and high availability. Mongo DB allows users to query data without having much knowledge and master SQL.

The process and style of querying and inserting data in mongo dB is much like as graph database which was learned through Neo4j. As like in graph database and relational database, MongoDB also provides an aggregation which is a framework which includes utility functions like count, group and distinct. More advance aggregation functions like average , min, max, sum, standard deviation and variance need to be implemented with MapReduce.

Knowledge about the principal of transaction property for the consistency of transaction so that there will not be any problem whenever system fails after transaction or between the transaction. Identifying conflicts pair in the given transaction was very new to me where, I came to acknowledge that how the schedule for transaction conflict serializability.

## Problems faced during exercises

Most of the activities were smooth beside some activities with couples of clauses in same query. Activities from Cypher query were easier to do as because of experience from the previous assignment where we get to involve with CASE WHEN , Unwind clauses as well as union.

Beside cypher query when it came to Mongo DB, all the clauses that are in use are quite as like Cypher and relational database. Problem occurred during Mongo DB was during the listing the result on descending or ascending order where it throws error whenever I use to order result with $orderby inside the find() statement , however it was solved with sort() aggregation outside the find statement and inside the aggregation pipeline.

During the conflict serializability test, I was not aware of making precedence graph which made me think how to test that and I again went through the lectures and identified how to make the precedence graph by separating conflicting pairs.

Overall, the problems were usually due to syntax errors like missing brackets or dot which were shortly solved beside the activity from database transaction management.

## Knowledges Gained

Being new to mongo DB, it did not feel like new because of its flexibility and usual aggregation functions and operators. Skills and knowledge gained after the completion of lectures and tutorials are below:

1. Deeper knowledge on Cypher query clauses like union, with, unwind, case when, range , list etc.
2. Deeper understand about SQL workbench and migration from SQL to graph database.
3. Reason for moving from relational database to mongo DB with the key attributes and differences such as storage, scalability, performance, data volume, cost effective , object oriented and schema less characteristic.
4. Using aggregation for querying data in easier way .
5. Mongo DB actually isn’t for relational data purposes.
6. Mongo DB is fast as because of its scarification on ACID property unless you try to put entire transaction into single record.
7. The use of pretty() operation to return data in formatted way.
8. Knowledge about the database transaction property and its smaller units

## Skills to improve

Beside the facts that every activity was completed but there were always some errors popping out which can be enhance next time if considering below point while querying or developing database in Mongo DB.

* Common errors were due to syntax errors, so it is good to close the bracket or semi colon as well as quotation whenever opening it.
* Know the acceptance accumulator object inside the aggregation .

# Part 2

## Introduction to Part 2

For the assignment 2, part 2 consist if same dataset is given as provided in first assignment which was of English Premier League. This time we have given the dataset in .json format which was .csv previously. We must import the data into our database and query 6 queries using Mongo DB. The dataset consists of team name and team type like either they are playing from home or away and the details of game like referee, goals, scores, draws etc. For the implementation and query, we will be using MongoDB .

## Database Design

For the database coding, we will use mongo dB command prompt with the command mongo if the directory is registered in out shell otherwise, we must run mongod before mongo. Starting with creation of database it’s easy for mongo dB with the command use DATABASE\_NAME and collection with db.createCollection(‘collectionname’).But for our case we can simply create database and collection while importing the file with the command below:

**mongoimport --db PremierLeague --collection EPL --file C:/epl\_1819.json**

With import of data, we have set the database name to PremierLeague and collection to EPL. Now, we can query our 6 queries which are listed below for which we have to switch to database using syntax use DATABASENAME :

## Queries

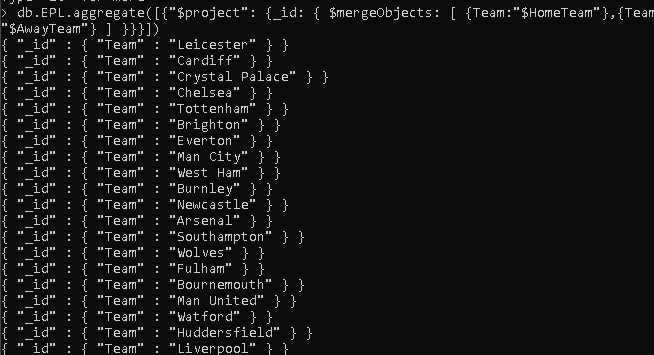
1) Show the EPL teams involved in the season.

For the result of team involved in the season we have to use aggregate function with project function including merge function for objects which will be collected from home team and away team as some team may be playing only as an away team in some cases. The query returned 20 teams name which is as expected.

**db.EPL.aggregate([**

**{"$project":{\_id:{$mergeObjects:[{Team:"$HomeTeam"},{Team:"$AwayTeam"}]}}}**

**]);**



2) How many matches were played on Mondays?

With the aggregation first project function is used to get the converted date using dateFromString function and giving the dateString value and the format of the data date which is then grouped with id as day of week and the sum giving value for each data 1. After grouping, the query is then matched with the id which is 2 for now because normally the Monday is listed as 2 but in ISO Monday is listed as 1. The result is 17 match which is as expected.

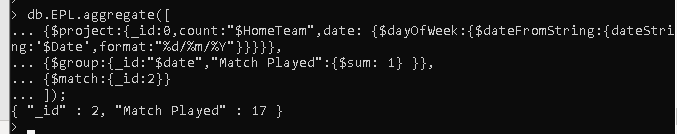
**db.EPL.aggregate([**

**{$project:{\_id:0,count:"$HomeTeam",date: {$dayOfWeek:{$dateFromString:{dateString:'$Date',format:"%d/%m/%Y"}}}}},**

**{$group:{\_id:"$date","Match Played":{$sum: 1} }},**

**{$match:{\_id:2}}**

**]);**



3) Display the total number of goals “Liverpool” had scored and conceded in the season.

For this query, use of match, project and group aggregation is done where at first dataset with matching home team and away team is filter using or operator after which in the project, for each cases calculation is carried either the team is playing from home or as away so that the goals from each side can be sum. After the calculation of data for goals the id is set same for all so that it can be sum in one row which is later done with group. Hence the result is Liverpool scored 89 goals and conceded 22.

**db.EPL.aggregate(**

**{$match:{$or:[{HomeTeam:'Liverpool'},{AwayTeam:'Liverpool'}]}},**

**{$project:{\_id:'Liverpool',**

**Goals:{$cond:{if:{$eq:['$HomeTeam','Liverpool']},then:{$sum:'$FTHG'},else:{$cond:{if:{$eq:['$AwayTeam','Liverpool']},then:{$sum:'$FTAG'},else:{$sum:0}}}}}, Conceded:{$cond:{if:{$eq:['$HomeTeam','Liverpool']},then:{$sum:'$FTAG'},else:{$cond:{if:{$eq:['$AwayTeam','Liverpool']}, then:{$sum:'$FTHG'},else:{$sum:0}}}}}**

**}},**

**{$group:{\_id:'$\_id',Goals:{$sum:'$Goals'},Conceded:{$sum:'$Conceded'}}}**

**);**

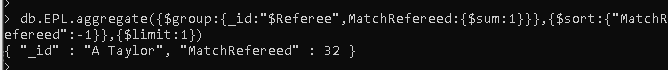


4) Who refereed the most matches?

With the use of aggregation in collection, the data is grouped with the referee and the matched refereed is the calculated with the sum function giving value 1 for each as same in number 2 query which is then sorted with number of matched refereed and limited with maximum match. The maximum match refereed is 32 by A Tylor which is as expected.

**db.EPL.aggregate(**

**{$group:{\_id:'$Referee',MatchRefereed:{$sum:1}}},{$sort:{'MatchRefereed':-1}},**

**{$limit:1});**

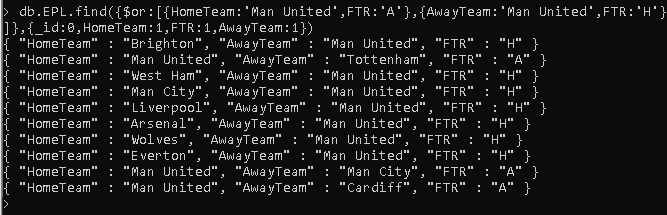
5) Display all the matches that “Man United” lost.

With the use of find the data can be retrieved without aggregation where or operator is used to match the data as home team, away team and result and then to look good output, id is set 0 as false and home team, result and away team is true. We can also use pretty function for this case. The retrieved data is 10 match which is as expected.

**db.EPL.find(**

**{$or:[{HomeTeam:'Man United',FTR:'A'},{AwayTeam:'Man United',FTR:'H'}]},**

**{\_id:0,HomeTeam:1,FTR:1,AwayTeam:1}**

**);**

6) Write a query to display the final ranking of all the teams based on their total points

For this query, which is very complex in compare to other, it needs the combination of different aggregations like project, group, unwind, concatArrays, push and sort. At first the dataset is calculated as like in number 3 query to get the required points using conditional statement after which the collected dataset with two different points as home and as a way is concatenated in array after pushing it where the concatenated array is then unwind which is use to group data on the basic of name of the team and calculating the sum and sorting in descending order for points.

**db.EPL.aggregate(**

**{$project:{\_id:0,H:'$HomeTeam',A:'$AwayTeam',**

**P1:{$cond:{if:{$eq:['$FTR','H']},then:{$sum:3},else:{$cond:{if:{$eq:['$FTR','D']}, then:{$sum:1},else:{$sum:0}}}}}, P2:{$cond:{if:{$eq:['$FTR','A']},then:{$sum:3},else:{$cond:{if:{$eq:['$FTR','D']}, then:{$sum:1},else:{$sum:0}}}}}**

**}},**

**{$group:{\_id:null,details:{$push:{name:'$H',point:'$P1'}},details1:{$push:{name: '$A',point: '$P2'}}}},**

**{$project:{\_id:{$concatArrays:['$details','$details1']}}},**

**{$unwind: '$\_id'},**

**{$group:{\_id:'$\_id.name',Points:{$sum:'$\_id.point'}}},**

**{$sort:{ 'Points':-1}}**

**);**



## Part 2 Reflection

Overall, the queries went smooth beside query number 3 and 6 which made me brainstorm for several days. Beside the two queries others were solved within 2,3 days with the use of different aggregation functions and operators like project, group, merge Objects, or etc. Throughout this queries solution I got very insight view of how the mongo dB works and what are the easier way to query with the huge data in single document. After many tries, query 3 and 6 were solved with nested conditional statement and array bindings and helped me to face such kind of queries in a big dataset mainly with games where each individual team can play from any side. The references that have been followed for the success of these queries; all the references are listed in the end of report.

# Part 3

Bigdata analytics and the role of NoSQL databases

Sudish Basnet

*UoN id : 18413692 ,Department of Computing*

*Northampton University*

sudishbasnet@gmail.com

***Abstract*—Digital world which is growing very fast with the development of big companies and social junction sites, the data that need to be processed become more complex in volume which is terabyte to petabyte which is refer to Big Data, it can be either structured, unstructured , time series, geographic etc. depend upon the source either by humans or robots where human generated data source may be less trustworthy due to interaction with surrounding where they belongs either noisy hence need more focus on that while analyzing data and this trend of big data is becoming a global phenomenon. So big data will no more can be analyze with traditional method like relational database as well as traditional search engines due to its expensive nature for scaling the default volume to next label every time if it come s to process different millions of data each time. That is the reason NoSQL databases plays a vital role for Bigdata analytics with its flexibility and less complex beside the fact of not using any schema or following ACID properties as like in traditional methods.**

***Keywords* --- Bigdata, NoSQL , RDBMS, DBMS**

1. INTRODUCTION

W

ith the rapid development of social networking sites and big companies storing millions of data, relational database is not the case to implement to use due to its complexity with big data and expensive to run single server. So, the running companies or social networking sites like Facebook, Twitter, Google who collect terabytes of user data every single day won’t be possible to manage data collections with relational database and hence need NoSQL to make the system performance high and make it more flexible without any complexity.

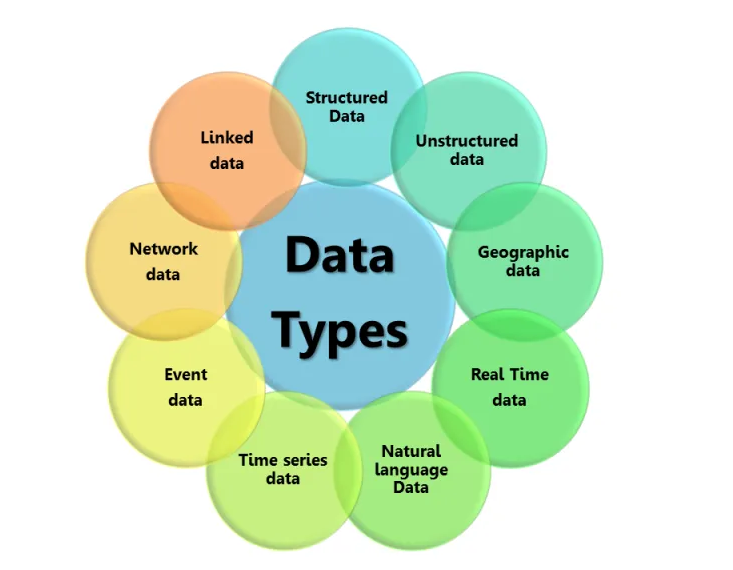
Big data is a field or an area which addresses ways of analyzing, systematically extracts data information from, or simply deal with data sets which are too broad or complex sometime for conventional application software which are for data processing to deal.

A NoSQL database is a database that provides a data storage as well as recovery mechanism that is based on means rather than tabular entities relationships that can be found in relational databases. Such databases have been around from the late 1960s, but the term "NoSQL" was coined only in the early 21st century, which was prompted by Web 2.0 companies 'needs.

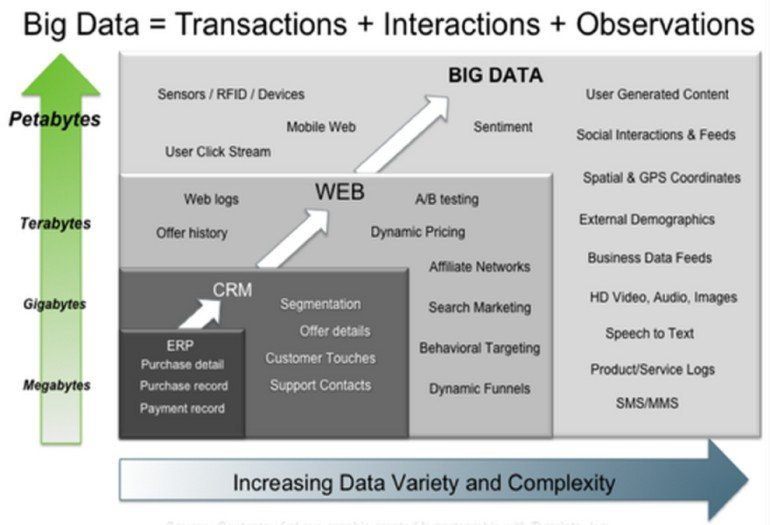
To deal with big data with the requirement of volume, velocity and variety, NoSQL helps effectively. Volume is basically maintaining the ACID properties which consist of Atomicity, Consistency, Isolation, and Durability which is not always necessary and expensive too so for some cases we can deal with minor inconsistencies with our results so that user must able to separate data on multiple sites. Variety in big data refers to varying data where single fixed data model makes it complex to incorporate due to not knowing of schema in some cases so, relational database can be expensive at that time. Next with velocity in big data, it’s not possible to store every data durable to disk all the time if compared to cost efficiency beside some fact of low probability of losing data with related project as memory is not in concerned anymore nowadays due to easy availability, cheap and fast.

1. BIG DATA

With the quick growing of digital world and becoming very complex in nature in terms of volume which extends from terabyte to petabyte, variety like structured , unstructured and hybrid, velocity which is high growth rate of digital data. This means 'Big Data' is a global trend. This is usually seen as a collection of data that has become so large that it cannot be easily handled or manipulated using traditional data management tools like RDBMS or search engines.

(Toward Scalable Systems for Big Data Analytics: A Technology Tutorial - IEEE Journals & Magazine, 2020) Recent technological advancements have led to a deluge of data from distinctive domains (e.g., health care and scientific sensors, user-generated data, Internet and financial companies, and supply chain systems) over the past two decades. Simply it suggests the big data which is coined to capture the value of such emerging digital trend.

Big Data Transactions with Interactions and Observations can be seen in figure below (Source: http://hortonworks.com/blog/7-key-drivers-for-the-big-data-market/ [41]) showing the theorem:



(Moniruzzaman and Akhter Hossain, 2013) Computational and storage requirements of applications such as for Big Data Analytics, Business Intelligence, and social networking over peta-byte datasets have pushed SQL-like centralized databases to their limits [8]. So, this is the fact which explains the lead for the development of NoSQL which is non-relational database.

Big data in analytics involves many types which are structured, unstructured, natural language ,real-time media, geographic, event, network, time series and linked. In order to analyze such big data type it is necessary to know more about each type and able to distinguish either it is human-generated data or robotic where often human data may be less trustworthy, unclean or simply noisy in compare to data generated by robots.Below is the figure showing the data types involved in Big Data from source (Big Data: Types of Data Used in Analytics - Agroknow blog, 2020)

Some of the types of Big Data for analytics with brief description is given below:

* Structured data : Structured data are those data stored in rows and columns like in tabular form, often numerical, where each data object determines its value. It is estimated that this form of data makes up about 10 percent of the total data today and is accessible through database management systems. Examples of standardized or simply traditional data sources includes official databases which are designed by government agencies to store data about persons, companies, real estates, and sensors in industries that collect processes data. Today, sensor data is one of the fast-growing fields, particularly the installation of sensors in plants to track movement, temperature, position, vibration, pressure ,light, liquid and flow.
* Un-Structured data: Un-Structured data are those data of the various forms such as text, image, video, paper, etc. This may also take the form of complaints from clients, contracts, or internal communications. That form of data reflects around 90 per cent of the data that has been generated in this century. Since the middle of the last decade, the explosive growth of social media (e.g. Facebook and Twitter) is responsible for most of the unstructured data we have today. What separates data generated in social media from other forms of data is that social media data has a personal taste. This is the case where un-structured data cannot be store with conventional relational databases.
* Natural language Data : Human-generated data, particularly in the form of a verb. These data differ according to abstraction level and editorial consistency level. Natural language data sources include landlines, mobile phones, speech capture devices, and the Internet of Things which generate large sizes of text-like device-to-device communication.
* Real-time media : Media like Real-time broadcasting of data about live or stored media are real-time media. A special aspect of real-time media is the volume of data being generated which will become more complicated in terms of storage and processing in the future. One of the major media data sources is such services as e.g. Tumblr, Flicker, and Vimeo create an immense amount of content, images, and audio. Video conferencing (or visual collaboration) is another important source or real-time media which allows communication simultaneously like two-way interaction through video and audio transmission over internet from two or more locations.
* Geographic data : Data created from geographical information systems relating to houses, lakes, highways, addresses, citizens, workplaces and transport routes. Such data relate between location, time, and attributes (i.e. descriptive data). Geographic digital data has tremendous advantages over conventional data sources likes charts, such as paper maps, explorer written reports and spoken accounts in that digital data can be quickly replicated, processed, and transmitted. More importantly easy to store, transform and analyze. These data are useful in community planning and environmental impact monitoring. Geo statistics is a branch of statistics that engages in spatial or spatiotemporal data.
* **Linked data : Data which are** based on standard web technologies such as HTTP, SPARQL, RDF as well as URIs for the sharing of information which computers can semi-consult (instead of serving human needs) are linked data. This allows the relation and the reading of data from various sources. The concept was coined in a design note about the Semantic Web project by Tim Berners-Lee, the founder of the World Wide Web Consortium. This project enabled the Web to connect relevant data that was not connected in the past by providing the frameworks and lowering the barriers to connecting currently linked data. Example for big data repositories consists FOAF (friend of a friend), GeoNames ( RDF descriptions of geographical features more than 7 million worldwide), DBpedia ( Dataset of extracted data from Wikipedia) and UMBEL ( Reference structure of 20 k subject classes and relationship derived from Open Cyc)

1. CHALLAGES AND ISSUES FOR BIG DATA ANALYTICS

According to “Top 5 problems with big data - and how to solve them, 2020 “ the problems are finding the signal in the noise ,Data silos , Inaccurate data , Technology moves too fast , Lack of skilled workers. It is hard to get insights out of a big bunch of info. Maksim Tsvetovat, author of the book Social Network Analysis for Startups and Intellect soft’s big data scientist, said that to effectively use big data,” "There has to be a discernible signal in the noise that you can detect, and sometimes there just isn’t one.”

Thus, most common, and biggest challenges that businesses face during managing big data is a classic needle-in - a-haystack question. Tsvetovat stated that big data looks like a hairball in its raw form, and it needs analytical approach to the data. The study by CapGemini claimed that 37 percent of businesses have difficulty getting trained data analysts for using their data. Their bet is to create a common data analysis team which will work for their company, either by retraining the current employees or by hiring big new data-specific staff.

(Kumar and Kirthika, 2017) Meeting the challenges by big data will be difficult. The volume of data is already large voluminous and increasing every day. The velocity of its generation and growth is increasing day by day, driven in part by the proliferation of internet connected devices. Furthermore, the variety of data is being generated and expanding, and organization’s capability to capture and process this data is limited. Big data is an emerging trend and the need for rising in all science and engineering domains.

1. PROPOSED METHODS FOR BIGDATA ANALYTICS

In order to clean and maintain big data, data must go through some of processes which are like Verify new data , remove duplicates , update data and Implement consistent data entry. For the fact that the process can be easier with DBMS so the collection of DBMSs and any other data tuning system can be helpful for Big Data analytics. Below are the prosed methods which can be used to analytics big data with the right selection of system.

1. RDBMS FOR BIG DATA

The RDBMS (relational database management system) used to be only solution for all database needs. Companies like Oracle, Microsoft and IBM are the players leading in relational database. Structured query language is use for RDBMS to define, update and query database but the increasing volume , velocity of data by companies which is changed dramatically within few years it’s impossible to meet the requirement though RDBMS look good with schema but complex to know each relationship when came to big data.

**Limitations of relational database to support big data**

Data size grew tremendously to the petabyte range where relational database finds it difficult to manage such large amounts of data for which it added CPUs as well as more memory for scale up to the DBMS.

* Data from online media, text, video ,audio and mails in a structured as well as unstructured format is outside the reach of RDBMS, as RDBMS literally cannot categorize unstructured data due to design of database to help structured data like sensor, weblog and several financial data.
* The "big data" is also produced at a very high speed but relational database is lacking in high speed as it is built to maintain data more slowly than rapidly expand. Even if relational database is used for the processing and storage of "big data," it leads to be very costly. The failure of RDBMS to manage "big data" has resulted in the advent of emerging technologies.

1. NoSQL FOR BIG DATA

(View of Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics, 2020) This article contrasted SQL versus NoSQL databases and outlined the four NoSQL data models in Big Data Analytics in the context of business situations. These NoSQL data models are understandable and simple to execute and do not involve complicated methods for SQL optimization in Big Data analysis. It is stated that NoSQL acts as a great tool for solving data availability. If compare with RDBMS where data needs to fit into tables and in case of it does not fits, then database structure should be outline again.

There are 4 types of big NoSQL which are key-value store (e.g. Simple DB) , column-oriented database (e.g. Cassandra ), document store (e.g. CouchDB, Big Table, HBase) and graph database (eg:Neo4j) and each database type solves problem which were unsolved by RDBMS.

Organizations that accumulate vast amounts of unstructured data gradually shifting into non-relational databases, also referred to as NoSQL databases as it concentrates on computational analysis of large-scale sample data, helping greater scalability over commodity hardware

The CAP-Theorem suggests that only two out of three individual aspects of scaling which are Strong Consistency, Partition-tolerance and High Availability can be achieved fully at the same time. Below is the figure (Source: nosqltips.blogspot.com) showing characteristic of NoSQL.

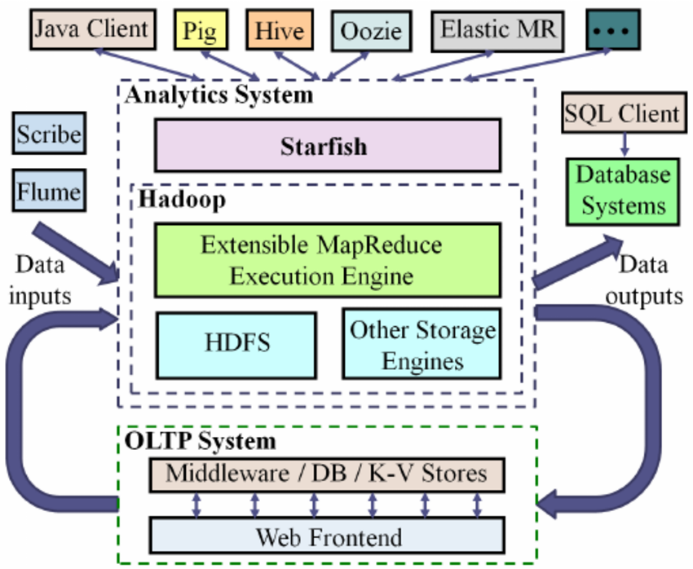
So, the feature of NoSQL for being **Non-relational, Schema-free, Simple API, and Distributed execution made the role vital for Big data analytics.** NoSQL facilitates massive-scale, high-performance, agile information processing. It stores unstructured data across multiple nodes in processing as well as across multiple servers. As such, the distributed NoSQL database architecture has become the chosen approach for few of the warehouses with largest data.

They call NoSQL “The New Darling of the Big Data World “ . According to Dillon, NoSQL is designed for organizational needs— real-time applications that frequently communicate with consumers or parties outside the organization. It offers the ability to query data, so that when it evolves, users can dig down into the data. NoSQL facilitates massive-, high-, agile information processing. As NoSQL databases were developed to better manage and analyze data sets in order to satisfy the demand for data management to handle the increasing complexity as well as inter dependence of big data by internet companies which is a great success too.

1. STARFISH

Starfish: A Self-tuning System for Big Data Analytics (Herodotou et al., 2011) Timely and cost-effective analytics over “Big Data” is now a key ingredient for success in many businesses, scientiﬁc and engineering disciplines, and government endeavors. The Hadoop software stack—which consists of an extensible MapReduce execution engine, pluggable distributed storage engines, and a range of procedural to declarative interfaces—is a popular choice for big data analytics. Most practitioners of big data analytics—like computational scientists, systems researchers, and business analysts—lack the expertise to tune the system to get good performance. Unfortunately, Hadoop’s performance out of the box leaves much to be desired, leading to suboptimal use of resources, time, and money (in pay-as-you-go clouds). For the case of analytics of big data, they introduced Starﬁsh, a self-tuning system . Starfish builds on Hadoop to automatically deliver good performance when adjusting to user needs and device workloads, without the need for users since user do not need to understand and control the tuning knobs in Hadoop. Whereas the framework architecture of Starfish is influenced by research on self-tuning of database systems, they are addressing how current analytical approaches around big data present new challenges with the lead of different c design.

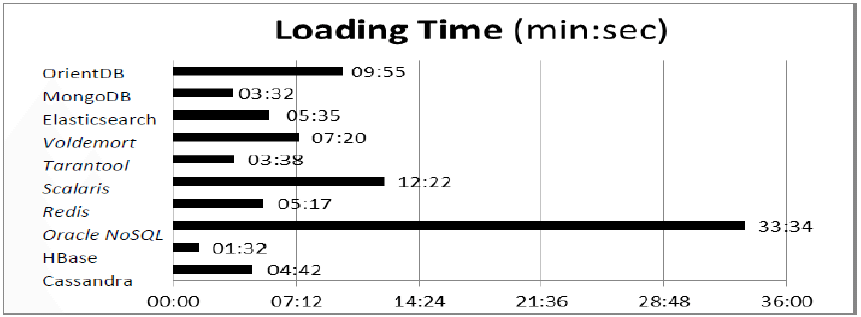
Below is the figure for Starfish in the Hadoop ecosystem.



1. EXPERIMENT DATA AND RESULTS

Experimental Evaluation of NoSQL database (Abramova, Bernardino and Furtado, 2014), the purpose of this paper is to compare different NoSQL databases, to evaluate their performance according to the typical use for storing and retrieving data. For the case study they have tested ten NoSQL databases for big companies like Yahoo Cloud Serving Benchmark with the help of a mix of operations to enhance better understanding of the capability using non-relational databases like NoSQL for the purpose of handling different individual requests as well as to acknowledge the performance either it is affected by the individual non-relational database type and their internal mechanisms.

Below is figure from the above research showing data loading by comparing of execution time with testing database loading 600.000 records.



From the above evaluation the time for the execution of loading 600.000 records by each database was interesting where HBase was in lead with 1:32 min and the MongoDB with the time of 3:38 . So, the result clearly justifies that NoSQL is much faster than any other database while came to execution of big data.

Below is the figure from the same research journal where workload A consist of 50% updates and 50% reads over 600.000 records.



From the above figure when analyzing the execution good performance is given by key-value store database with Tarantool which is the fastest of all due to the mapping of od data into memory. After Tarantool , Redis follow the max performance . Beside the good performance of NoSQL , Oriented DB had given worst performance with 30.9 seconds which is huge for this case with compare to other DBMS which is due to reading of data from the disk with the lower speed of it compare to volatile memory. Overall, the performance of NoSQL is satisfactory with the workload given for this scenario which justifies for the big data analytic, NoSQL role is the main to enhance best performance.

1. CONCLUSION

To conclude whole research, traditional method will be fade out if it comes to bigdata analytic due to it processing speed and scalability which can only be overcome with the help of NoSQL . Bigdata which is the trending issue for every big companies needs NoSQL in order to make their functionality more fast and cheaper with some sacrifices made like ACID properties where the users need to do transaction of big data at a single time not one by one. Overall, the role of NoSQL for big data analytics seems vital due to it features like being **Non-relational, Schema-free, Simple API, and Distributed execution for multiple NoSQL database. The main point to consider is that with the big that thee comes the big responsibilities for companies to analysis data as fast as they can and server accordingly which is the same motive for the development of NoSQL the greater database for big data.**

Acknowledgment

Through the research on Big data analytics with NoSQL writer got the insight view of data structure of big data and how they can be processed or analyzed according the company guidelines where it belongs. Traditional methods like relational database and traditional search engines are unable to analyze big data from big companies with millions of data as effective as NoSQL without sacrificing the performance.

REFERENCES

1. Abramova, V., Bernardino, J. and Furtado, P., 2014. Experimental Evaluation of NoSQL Databases. *International Journal of Database Management Systems*, 6(3), pp.01-16.
2. Journalajrcos.com. 2020. *View Of Comparison Between SQL And Nosql Databases And Their Relationship With Big Data Analytics*. [online] Available at: <http://journalajrcos.com/index.php/AJRCOS/article/view/30108/56497> [Accessed 2 April 2020].
3. Ieeexplore.ieee.org. 2020. *Toward Scalable Systems For Big Data Analytics: A Technology Tutorial - IEEE Journals & Magazine*. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/6842585> [Accessed 2 April 2020].
4. Moniruzzaman, A. and Akhter Hossain, S., 2013. NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *International Journal of Database Theory and Application*, [online] 6(4.2013). Available at: <https://www.researchgate.net/publication/243963821\_NoSQL\_Database\_New\_Era\_of\_Databases\_for\_Big\_data\_Analytics\_-\_Classification\_Characteristics\_and\_Comparison> [Accessed 3 April 2020].
5. Bearden, R., Venkatesh, R., Lipcon, T., Kadar, L., Giraldo, S., Kadar, L., Maheshwari, K., Yao, X., Bearden, R., Venkatesh, R., Raja, V., Lipcon, T. and Lamairesse, M., 2020. *Cloudera Blog -*. [online] Cloudera Blog. Available at: <http://hortonworks.com/blog/7-key-drivers-for-the-big-data-market/ [41]> [Accessed 3 April 2020].
6. Brown, C., Brown, C., Brown, C., Brown, C., Brown, C., Brown, C., Brown, C., Brown, C. and profile, V., 2020. *Nosql Tips And Tricks*. [online] Blog.nosqltips.com. Available at: <http://blog.nosqltips.com/> [Accessed 3 April 2020].
7. Herodotou, H., Lim, H., Luo, G., Borisov, N. and Dong, L., 2011. Starﬁsh: A Self-tuning System for Big Data Analytics. *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, pp.261-272.
8. En.wikipedia.org. 2020. *Big Data*. [online] Available at: <https://en.wikipedia.org/wiki/Big\_data> [Accessed 4 April 2020].
9. SearchDataManagement. 2020. *What Is Nosql (Not Only SQL Database)? - Definition From Whatis.Com*. [online] Available at: <https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL> [Accessed 4 April 2020].
10. Techterms.com. 2020. *RDBMS (Relational Database Management System) Definition*. [online] Available at: <https://techterms.com/definition/rdbms> [Accessed 5 April 2020].
11. Agroknow blog. 2020. *Big Data: Types Of Data Used In Analytics - Agroknow Blog*. [online] Available at: <http://blog.agroknow.com/?p=4690> [Accessed 5 April 2020].
12. Kumar, S. and Kirthika, M., 2017. Big Data Analytics Architecture and Challenges, Issues of Big Data Analytics. *International Journal of Trend in Scientific Research and Development*, Volume-1(Issue-6), pp.669-673.
13. Piesync.com. 2020. *Top 5 Problems With Big Data - And How To Solve Them*. [online] Available at: <https://www.piesync.com/blog/top-5-problems-with-big-data-and-how-to-solve-them/> [Accessed 5 April 2020].
14. Sas.com. (2019). *Big Data Analytics - What it is and why it matters*. [online] Available at: <https://www.sas.com/en_us/insights/analytics/big-data-analytics.html>. [Accessed 5 April 2020].
15. Sharma, S., Tim, U.S., Gadia, S., Wong, J., Shandilya, R. and Peddoju, S.K. (2015). Classification and comparison of NoSQL big data models. *International Journal of Big Data Intelligence*, 2(3), p.201.

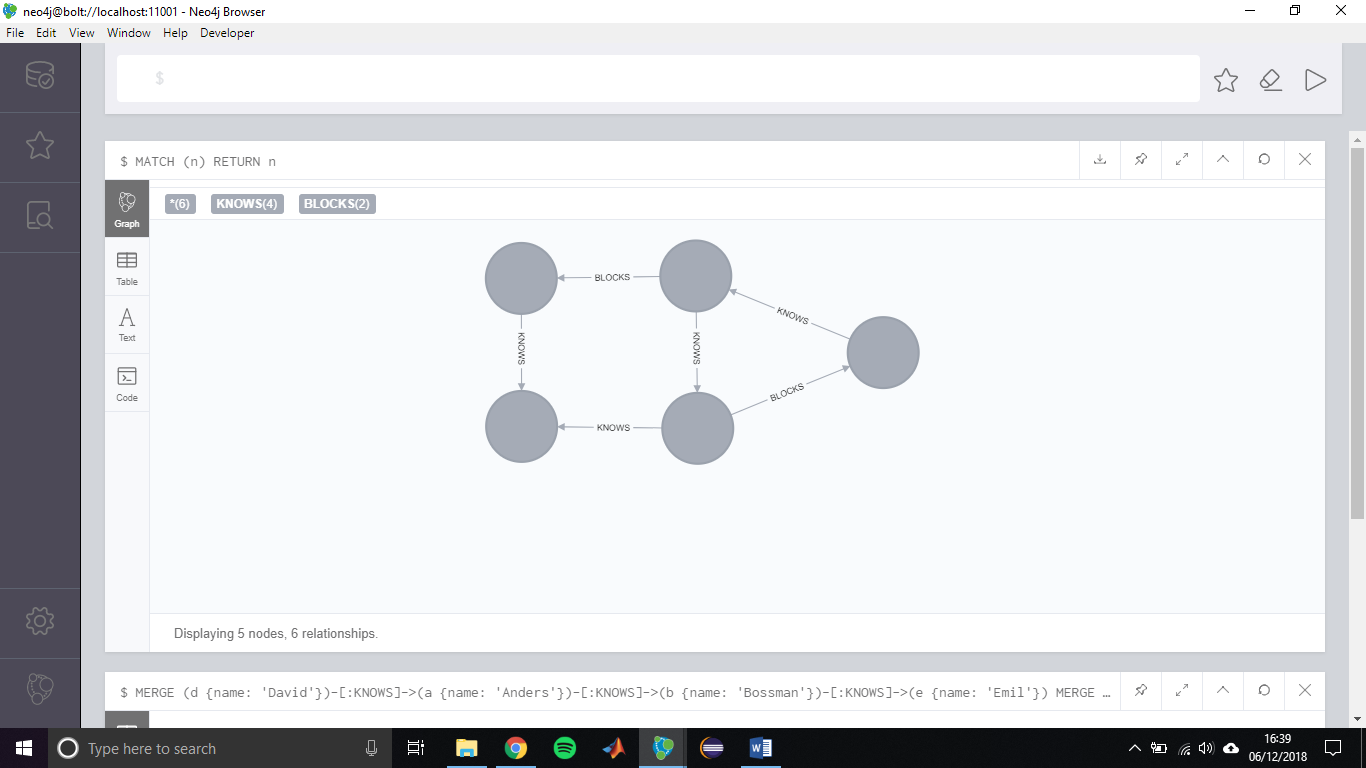
# Appendix

## Week 13

### Activities

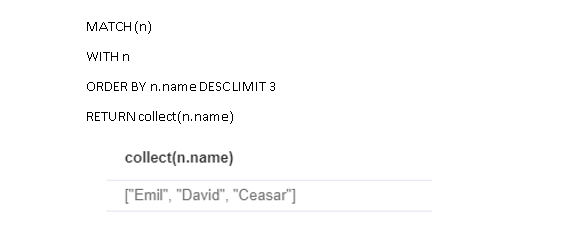
For this week , activity consist of using some of the extra clauses with Cypher which are AGGREGATION, UNWIND, WITH, COLLECT ,CASE and LIST. After the creation of previous week nodes and relationship, in this section all the queries will be done.

Below is the node and relationship from previous week:

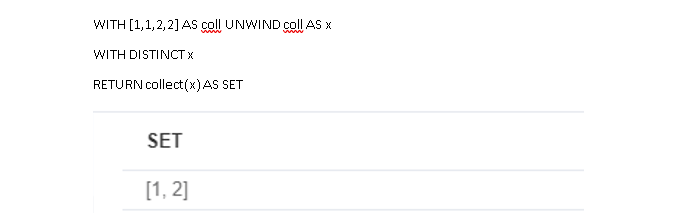


Now all the clause will be used here for querying data on the basic of various conditions.

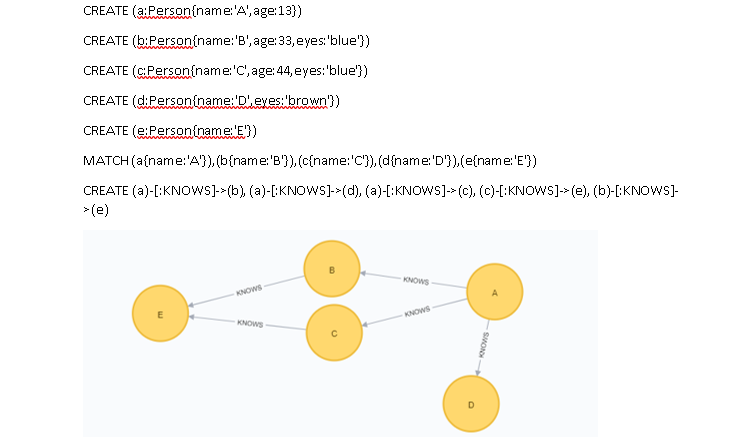
Using WITH clause to connect two functions and then returning data by using collect clause so that all the names will come in list.



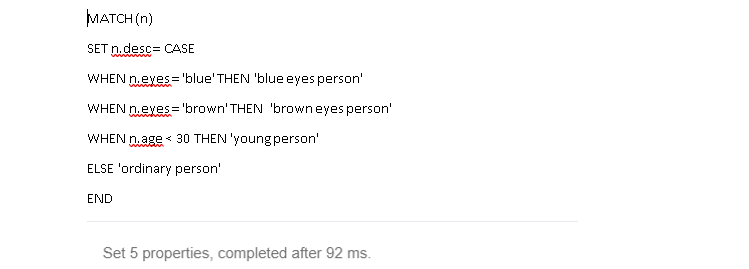
Using UNWIND for simple repeated numbers and retrieving data with distinct value wit collect clause for returning in one list.



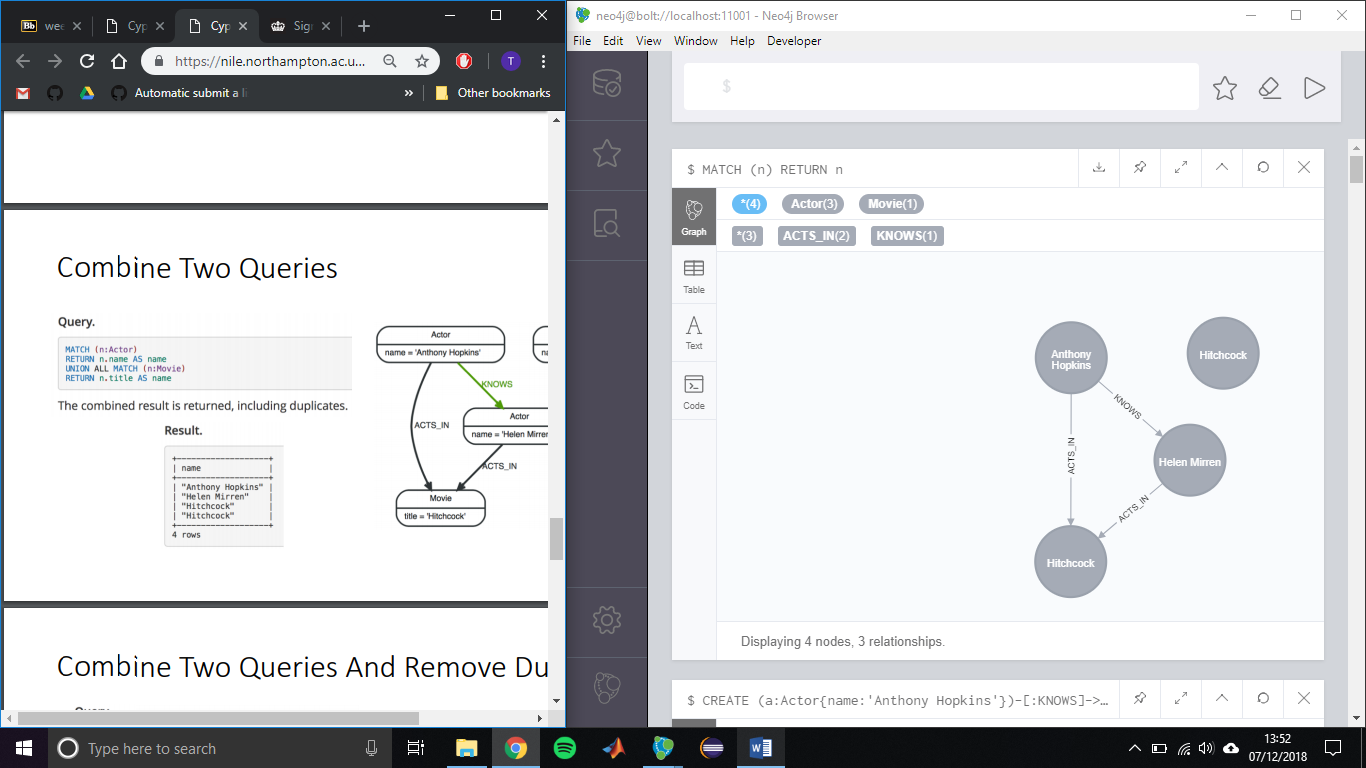
Creating new node and relationship for CASE WHEN clause .



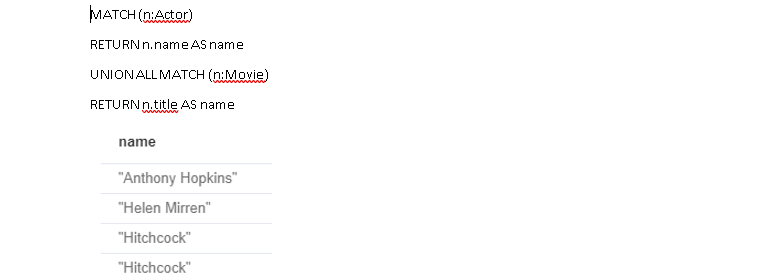
Matching every node at first then setting in descending order and then giving condition for their eyes color and age to compare to set properties for 5 nodes.



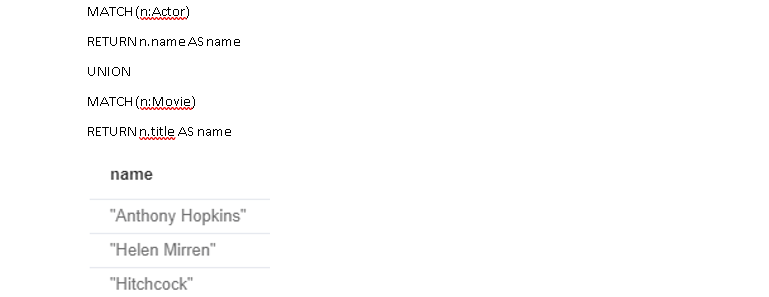
Now creating another node and relationship for UNION clauses



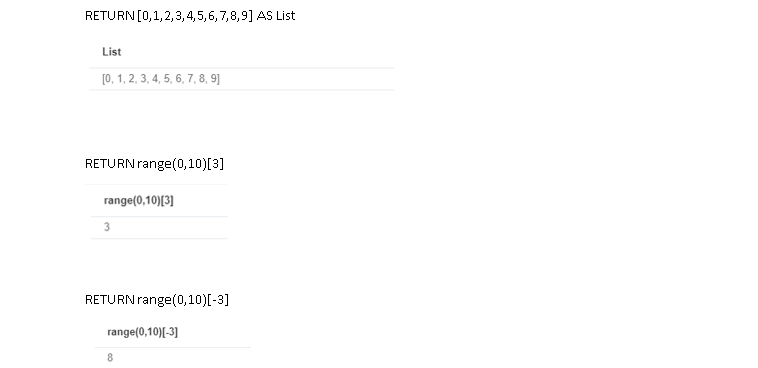
Using UNION ALL to return matching actor and move with two different return statement.



Using UNION clause to return data matching Actor and Movie with title as name



Using LIST function to return every data in one row and then returning data from loop with the rage of 0-10 and difference of 3 and vice versa for next query with -3.



Using extra CALL clause to list labels



#### Reflections

Overall, from this activity I got more deep knowledge on different clause that can be used while retrieving data from node and relationship with Cypher . It was fun to use list to retrieve data in single list so that it will be easier to see and unwind to make the list back into row. CASE WHEN seems very effective to try when it comes to compare all data with different properties. Using range to get data from list when some condition like difference between list with any number like 3 and -3 from above practical . WITH and UNION also makes the query easier to connect two MATCH statement and Return statement when it comes to return data with different properties to be presented in single return statement from same query.

## Week 14

### Activities

This week activity is to use MySQL workbench by connecting it to remote MySQL database and migrating to Neo4j too with the help of given Northwind data with which few queries needs to be carried.

The SQL file already available so that there is no need to write syntax for creating tables , relationships and insert data but for the migration to graph database the SQL dataset given should be first converted to CSV format so that it can be loaded to Neo4j , below is the creating code for nodes and relationship for this activity for graph database according to need of query.

LOAD CSV WITH HEADERS FROM "file:///customers.csv" AS data

MERGE(:Customer{id:data.customerID,company\_name:data.companyName,customer\_name:data.contactName,conta:data.address,country:data.country });

LOAD CSV WITH HEADERS FROM "file:///suppliers.csv" AS data MERGE(:Supplier{id:data.supplierID,company\_name:data.companyName,supplier\_name:data.contactName});

LOAD CSV WITH HEADERS FROM "file:///categories.csv" AS data

MERGE(:Category{id:data.categoryID,name:data.categoryName,description:data.description});

LOAD CSV WITH HEADERS FROM "file:///products.csv" AS data MERGE(p:Product{id:data.productID,name:data.productName,price:toFloat(data.unitPrice)})

MERGE (s:Supplier{id:data.supplierID})-[:SUPPLIES]->(p)

MERGE (p)-[:PART\_OF]->(c:Category{id:data.categoryID});

LOAD CSV WITH HEADERS FROM "file:///employees.csv" AS data

MERGE(e:Employee{id:data.employeeID,first\_name:data.firstName,last\_name:data.lastName,title:data.title})

MERGE (e)-[:REPORTS\_TO]->( m:Employee {id:data.reportsTo});

LOAD CSV WITH HEADERS FROM "file:///orders.csv" AS data

MERGE(o:Order{id:data.orderID})

MERGE (e:Employee{id:data.employeeID})-[:SOLD]->(o)

MERGE(c: Customer {id:data.customerID})

MERGE (c)-[p:PURCHASED]->(o);

LOAD CSV WITH HEADERS FROM "file:///orderdetails.csv" AS data

MATCH (p:Product {id:data.productID})

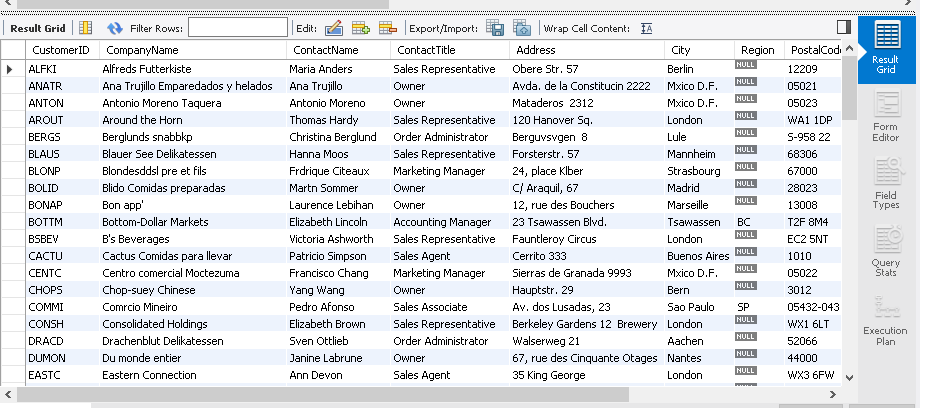
MERGE (o:Orders{id:data.orderID})-[r:ORDER\_DETAILS]->(p)

ON CREATE SET r.price = toFloat(data.unitPrice), r.quantity = toFloat(data.quantity);

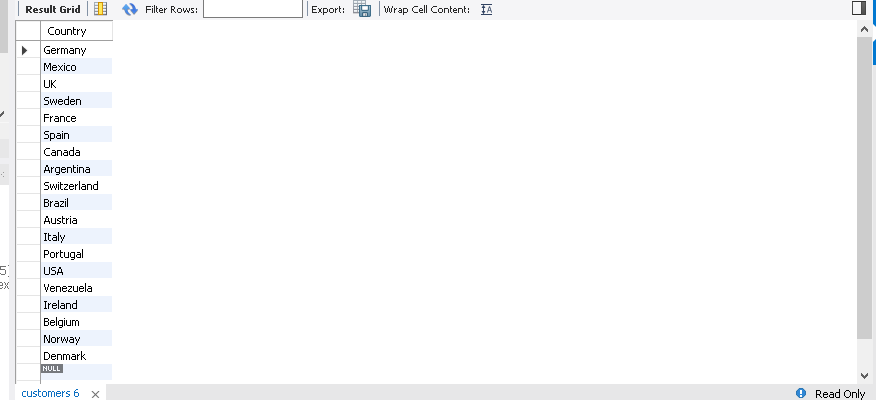
After successful creation of table in workbench with loading schema as well as migration from SQL to Neo4j below are the queries done for this activity as provided side by side.

**QUERIES**

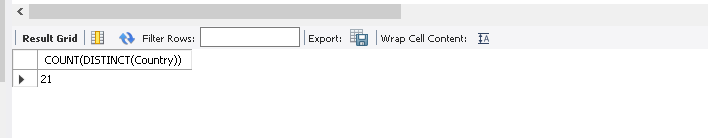
a. Display all customers.

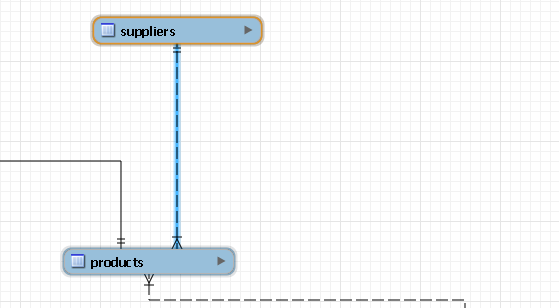
SELECT \* FROM customers;

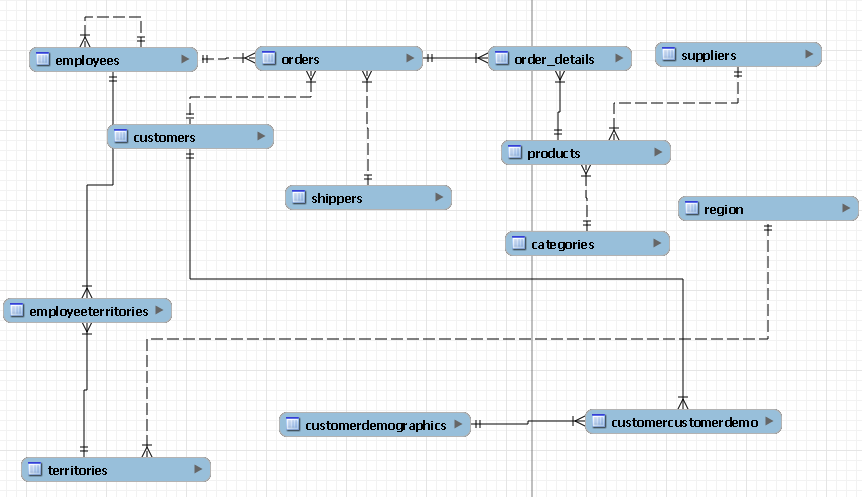
b. Display the unique country names of all Customers.

SELECT DISTINCT Country FROM customers;

c. Display number of countries that all the customers are from.

SELECT COUNT(DISTINCT(Country)) FROM customers;

d. Display the SUPPLIES relationship graphically.

e. Display all the relationships in the north wind database.

f. List all suppliers for the ‘Meat/Poultry’ category.

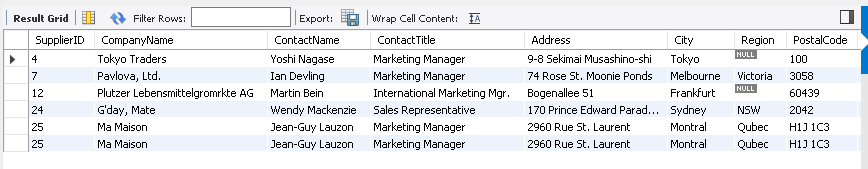
SELECT \* FROM northwind.suppliers ns

JOIN northwind.products np

ON np.SupplierID = ns.SupplierID

JOIN northwind.categories nc

ON nc.CategoryID = np.CategoryID

WHERE CategoryName ='Meat/Poultry';

g. List names of all customers who purchased products in the ‘Meat/Poultry’ category and the total amount purchased.

SELECT nc.ContactName,(nod.Quantity\*nod.UnitPrice)-(nod.Quantity\*nod.UnitPrice)\*nod.Discount

FROM northwind.customers nc

JOIN northwind.orders nor

ON nor.CustomerID = nc.CustomerID

JOIN northwind.order\_details nod

ON nod.OrderID = nor.OrderID

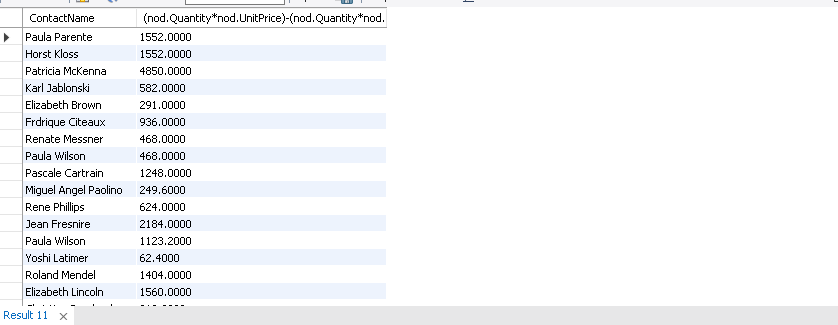
JOIN northwind.products np

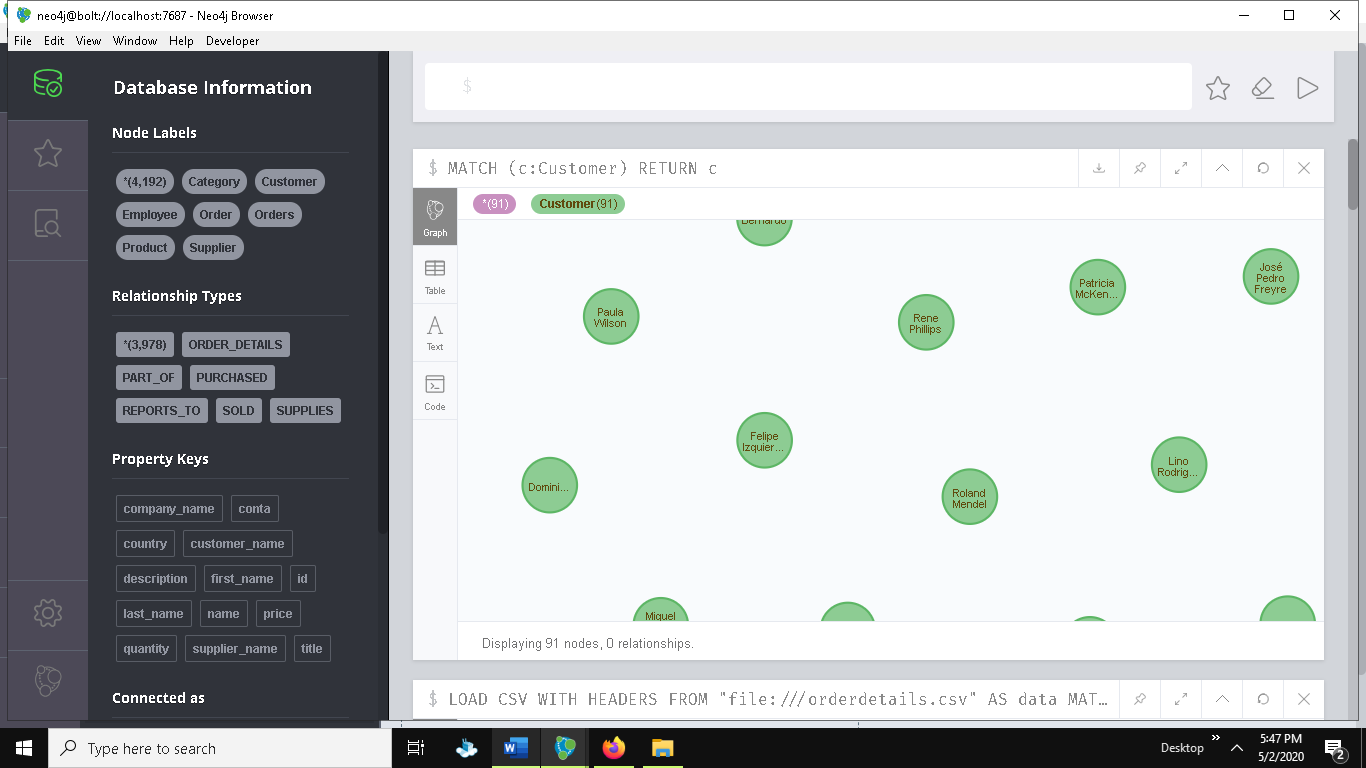
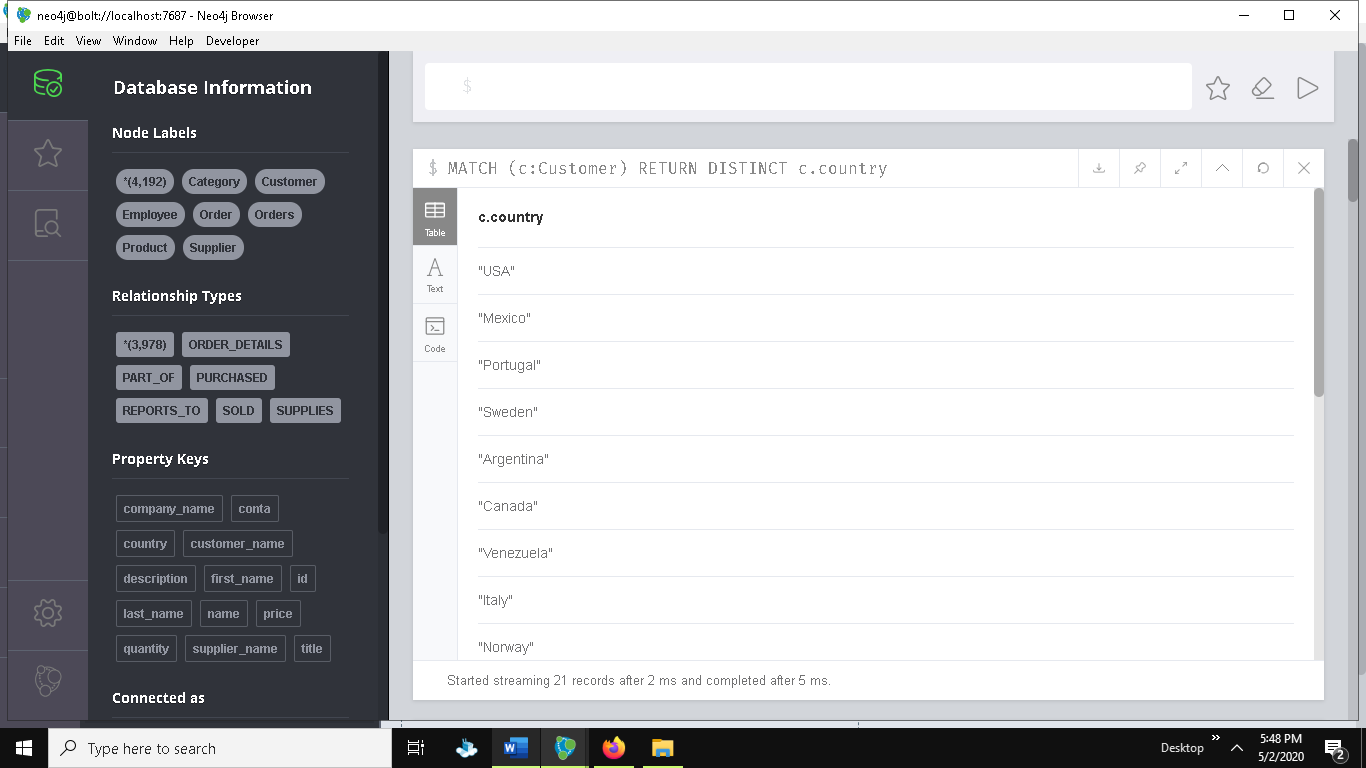
ON nod.ProductID = np.ProductID

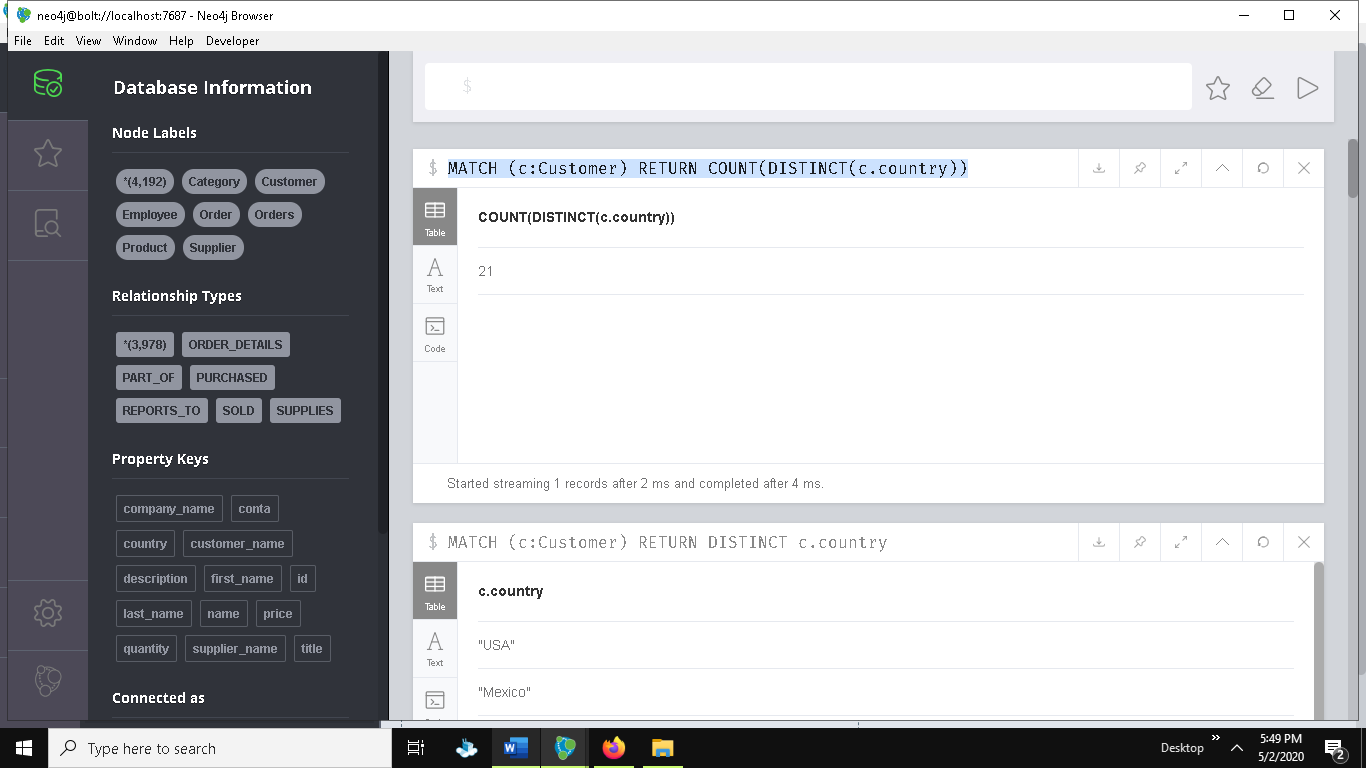
JOIN northwind.categories ncat

ON np.CategoryID = ncat.CategoryID

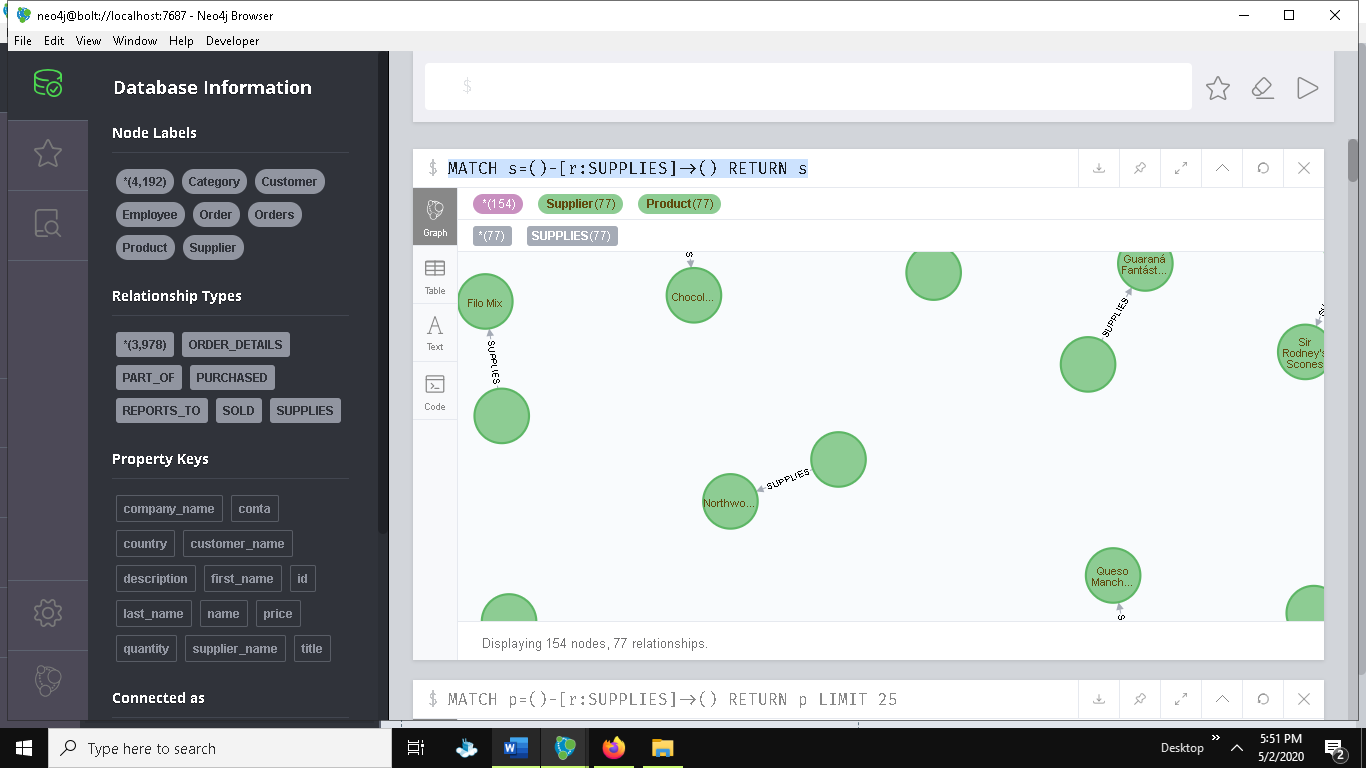
WHERE ncat.CategoryName = 'Meat/Poultry';



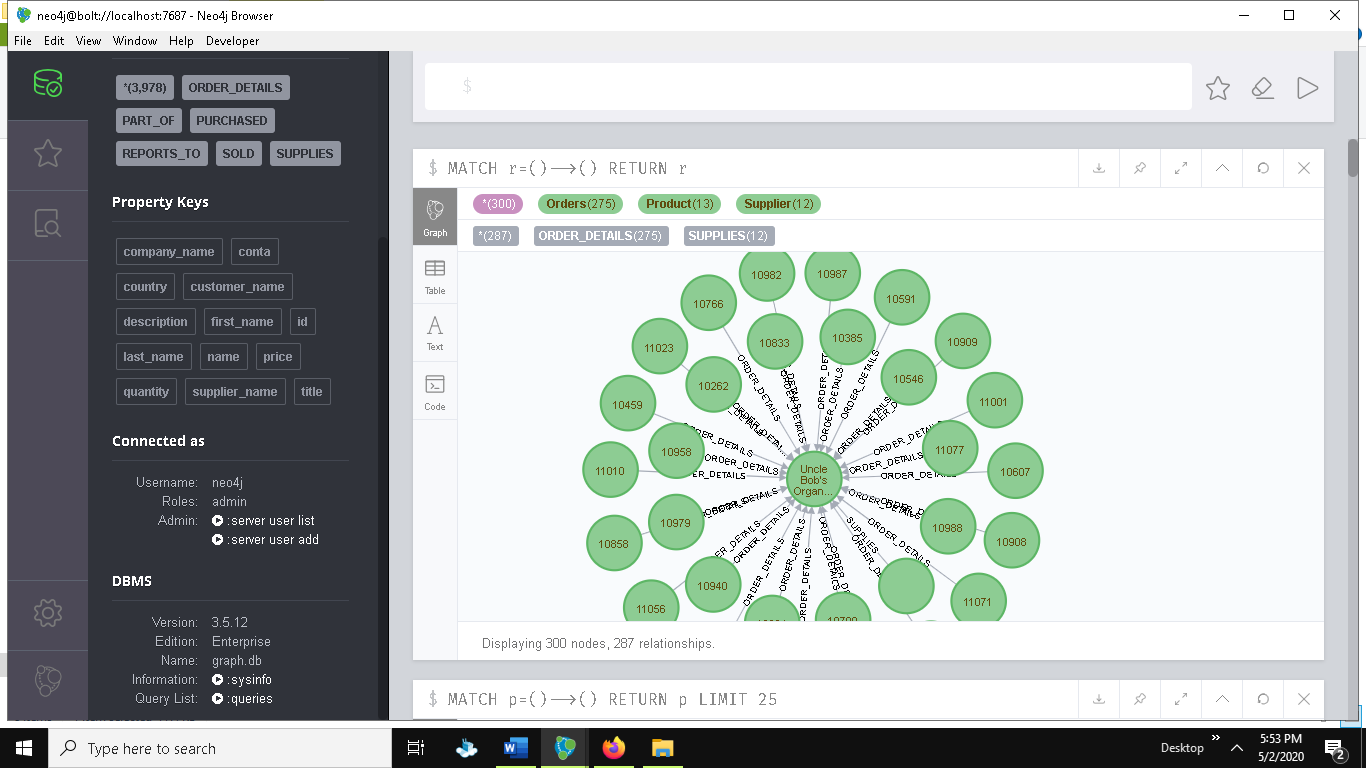
1. MATCH (c:Customer) RETURN c 
2. MATCH (c:Customer) RETURN DISTINCT c.country 
3. MATCH (c:Customer) RETURN COUNT(DISTINCT(c.country))



1. MATCH s=()-[r:SUPPLIES]->() RETURN s



1. MATCH r=()-->() RETURN r



### Reflections

From this week activity, the deeper knowledge on how simple can be the query is known comparing SQL and graph database where the process for loading data in into database in graph database is more easier with the use of CSV file where the huge tables and constraints creation in SQL is solved so that without being known to each relation where in SQL to query data specially with joins, there must be knowledge about the relationship but in graph database it is easy to know with the relationship property through which less entity can be establish to graph database for each storing order details in above graph database as a relationship property. Overall, this activity helped to determine the reason for migration.

## Week 15

### Activities

In this week activity 4 task are given in order to know more about graph database and the actual differences between RDBMS and Graph database. The first task is to develop a database prototype with the given relationship where data can be arbitrary. There is relation between person where person can be staff and the relation between them may different either knows, meets, teaches , reports to.

**At first creating graph database and querying some data where node created is person and staff, relationship created is meets, follows, reports\_to, and teaches as per the given graph.**

MERGE (p:Person {name: 'John'})<-[:MEETS]-(p1:Person {name: 'James'})

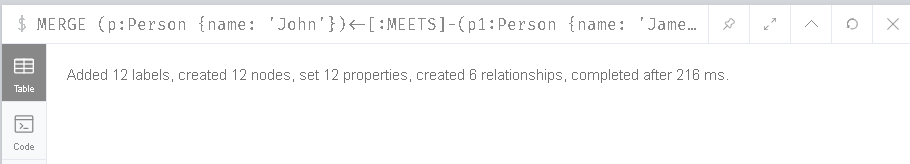
MERGE (p:Person {name: 'James'})-[:MEETS]->(p1:Person {name: 'Scott'})

MERGE (p:Person {name: 'James'})-[:FOLLOWS]->(p1:Person {name: 'Gary'})

MERGE (p:Person {name: 'Gary'})-[:TEACHES]->(p1:Person {name: 'Alice'})

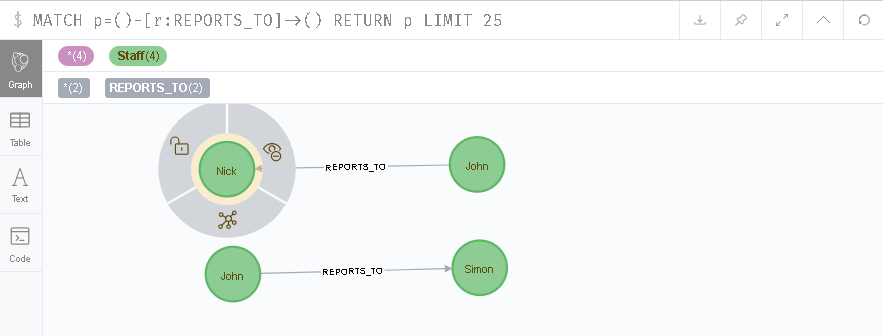
MERGE (p:Staff {name: 'Simon'})<-[:REPORTS\_TO]-(p1:Staff {name: 'John'})

MERGE (p:Staff {name: 'Nick'})<-[:REPORTS\_TO]- (p1:Staff {name: 'John'})

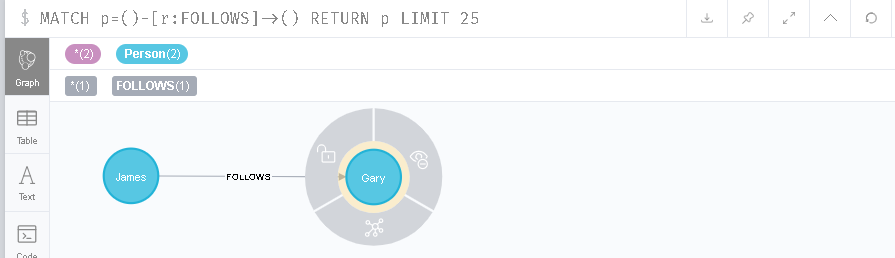


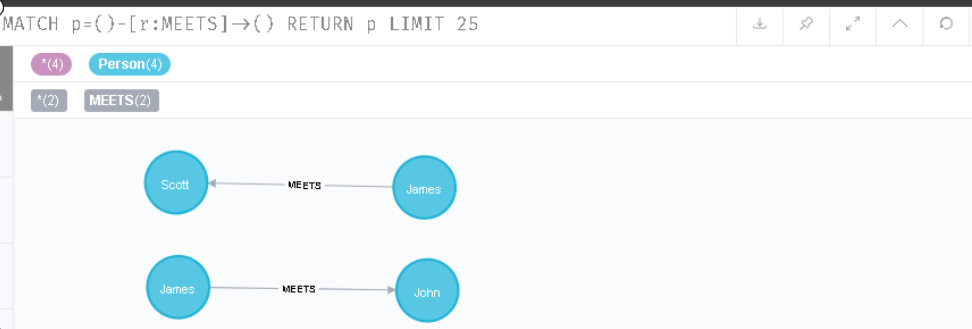
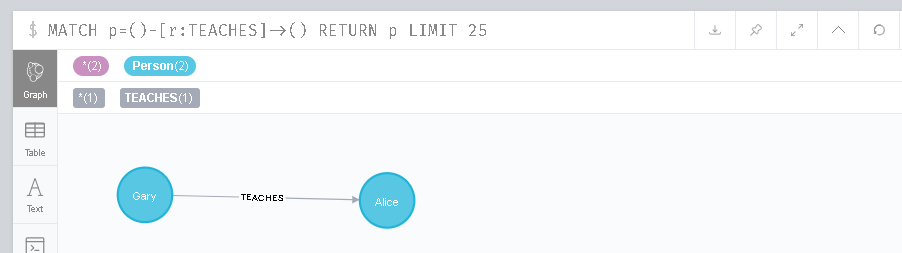
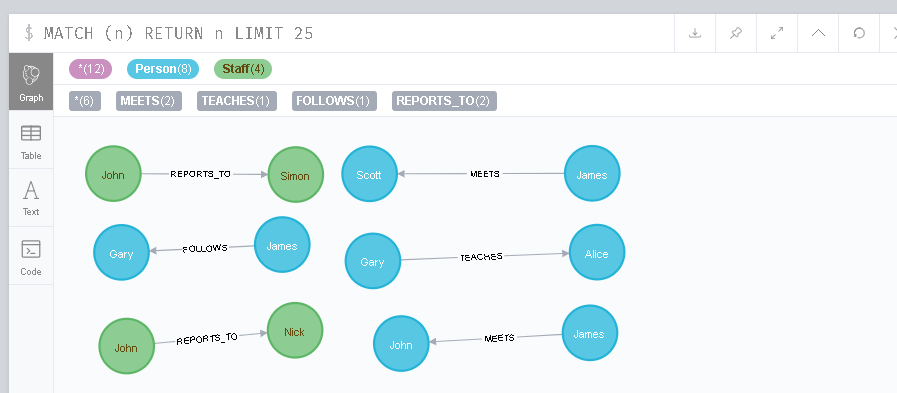
Returning nodes with the matching relationships

1. Returning nodes and relationship with the relationship of reports to



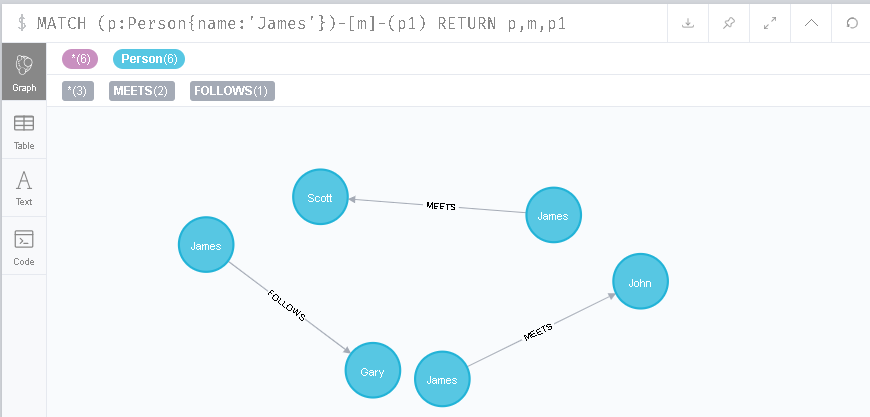
1. Returning nodes and relationship with the relationship of follows

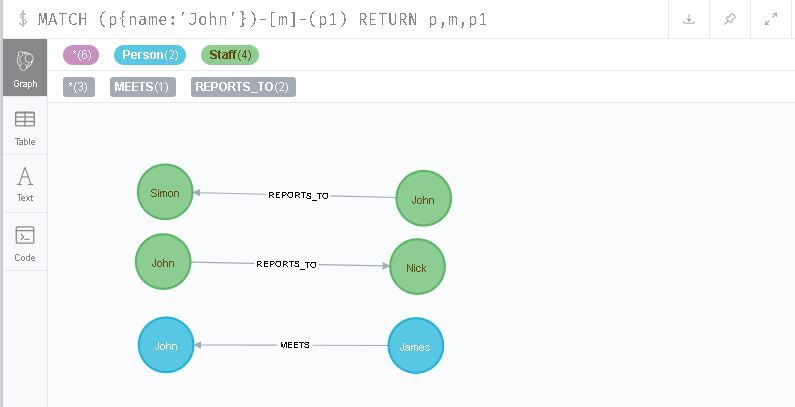


1.  Returning nodes and relationship with the relationship of meets
2. Returning nodes and relationship with the relationship of teaches
3. Returning all the nodes and relationships in the database.
4. Updating node information with match and set statement





1. Returning the relationship and nodes with the matching node having node label person and property with name James
2. Returning nodes and relationship with a node of having property with name John



1. Updating related nodes information with the matching node property with name John



1. Creating new graph database with same information of data but in single node person and relationship connection

MERGE (p:Person {name: 'John'})<-[:CONNECTION]-(p1:Person {name: 'James'})

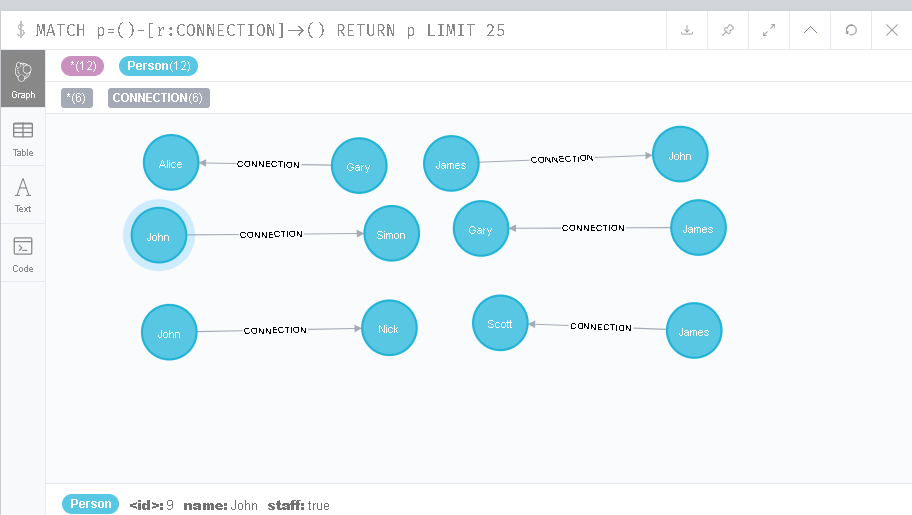
MERGE (p2:Person {name: 'James'})-[:CONNECTION]->(p3:Person {name: 'Scott'})

MERGE (p4:Person {name: 'James'})-[:CONNECTION]->(p5:Person {name: 'Gary'})

MERGE (p6:Person {name: 'Gary'})-[:CONNECTION]->(p7:Person {name: 'Alice'})

MERGE (p8:Person {name: 'Simon', staff:true})<-[:CONNECTION]-(p9:Person {name: 'John',staff:True})

MERGE (p10: Person {name: 'Nick',staff:true})<-[:CONNECTION]-(p11:Person {name: 'John'})



**Creating new database with the use of relational database for the same data set and querying accordingly with the help of SQL in localhost phpMyAdmin.**

1. Creating tables for the database person, staff, relationship, relationship1

CREATE TABLE person(

name VARCHAR2(15));

CREATE TABLE staff(

name VARCHAR2(15));

CREATE TABLE relationship(

person1 VARCHAR2(15),

name VARCHAR2(15),

person2 VARCHAR2(15));

CREATE TABLE relationship1(

staff1 VARCHAR2(15),

name VARCHAR2(15),

staff2 VARCHAR2(15));

1. Creating primary keys for the tables

ALTER TABLE person

ADD CONSTRAINT pk\_person

PRIMARY KEY (name);

ALTER TABLE staff

ADD CONSTRAINT pk\_staff

PRIMARY KEY (name);

1. Creating foreign key for the relations

ALTER TABLE relationship

ADD CONSTRAINT fk\_r

FOREIGN KEY (person1)

REFERENCES person(name);

ALTER TABLE relationship

ADD CONSTRAINT fk\_r1

FOREIGN KEY (person2)

REFERENCES person(name);

ALTER TABLE relationship1

ADD CONSTRAINT fk\_r3

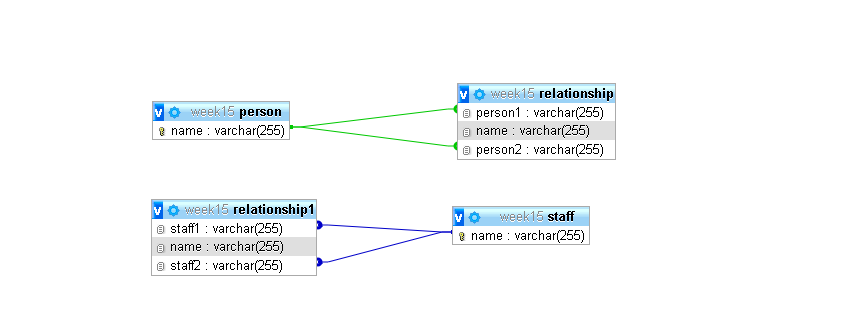
FOREIGN KEY (staff1)

REFERENCES staff(name);

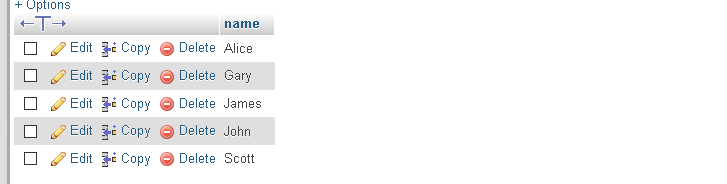
ALTER TABLE relationship1

ADD CONSTRAINT fk\_r4

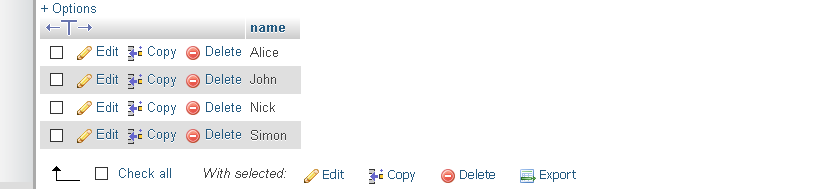
FOREIGN KEY (staff2)

REFERENCES staff(name);

1. Now inserting the data into the database and querying the database .

INSERT INTO `person` (`name`) VALUES ('Scott'), ('Alice'),('Gary'),('John'), ('James');

INSERT INTO `staff` (`name`) VALUES ('Simon'), ('Alice'),('John'), ('Nick');

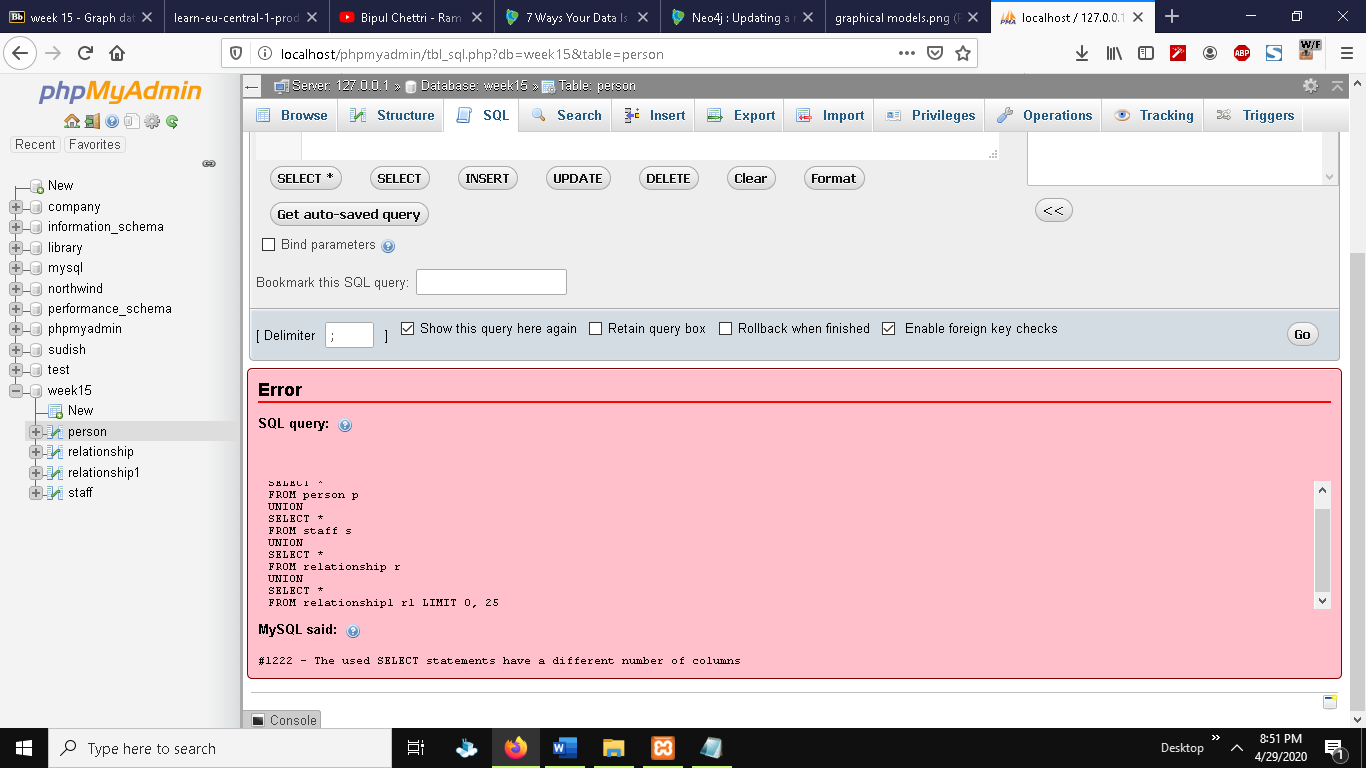


INSERT INTO `relationship` (`person1`, `name`, `person2`) VALUES ('James', 'knows', 'John'), ('James', 'meets', 'Scott'),('James', 'follows', 'Gary'), ('Gary', 'teaches', 'Alice');

INSERT INTO `relationship1` (`staff1`, `name`, `staff2`) VALUES ('John', 'reports\_to', 'Simon'), ('John', 'reports\_to', 'Nick');



1. Now querying the inserted data according to data queried from graph database.
2. Selecting all the data in the database



1. Selecting persons and relationship with the person name of James

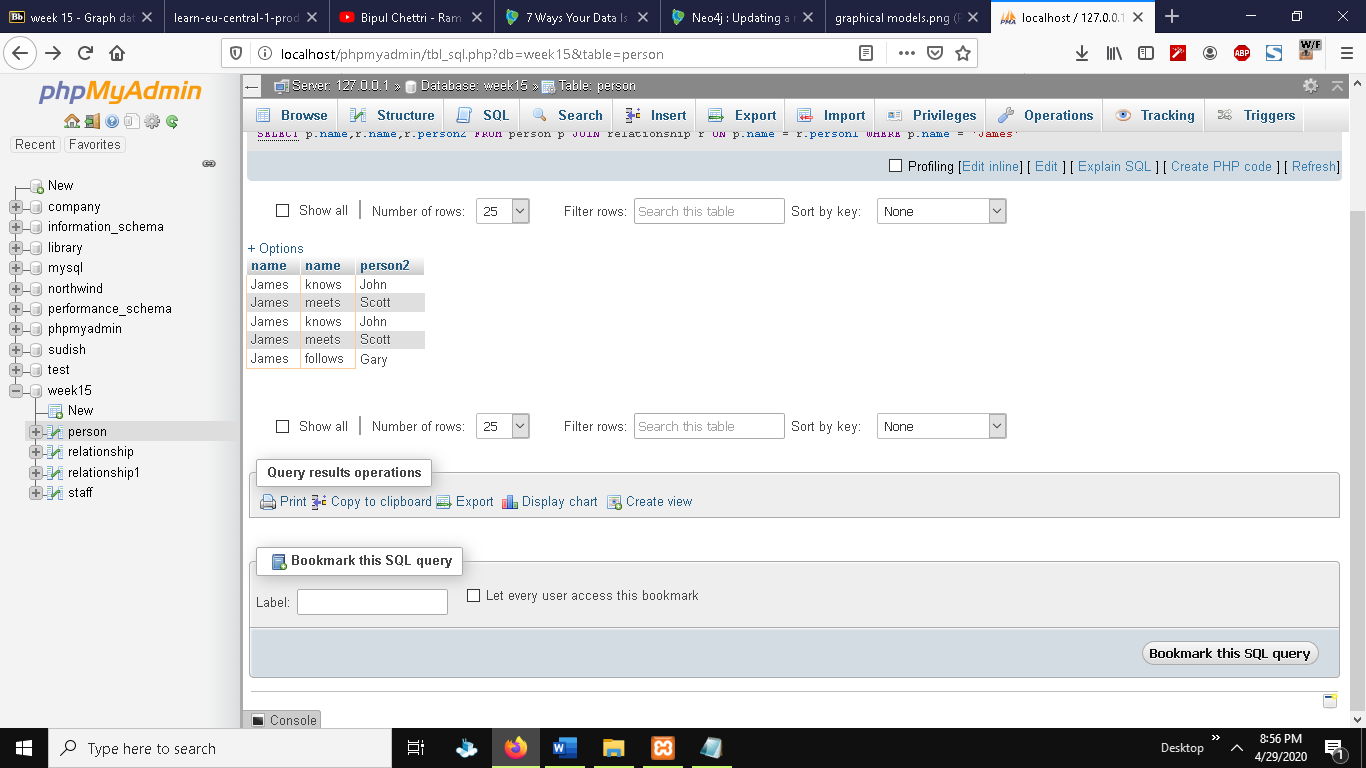
SELECT p.name,r.name,r.person2

FROM person p

JOIN relationship r

ON p.name = r.person1

WHERE p.name = 'James';



1. Selecting person and staff with name john and their respective relationships

SELECT p.name,r.name,r.person1

FROM person p

JOIN relationship r

ON p.name = r.person2

WHERE p.name = 'John'

UNION

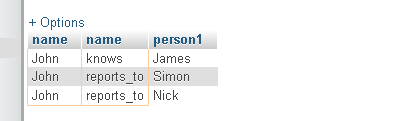
SELECT s.name,r1.name,r1.staff2

FROM staff s

JOIN relationship1 r1

ON s.name = r1.staff1

WHERE s.name = 'John';



1. Returning person names with their relationship of meets

SELECT r.person1,r.name,r.person2

FROM relationship r

WHERE r.name = 'meets' ;



1. Updating person data which is in relation with james making their age 20
2. First adding age attribute to person table

ALTER TABLE person ADD(

age INTEGER(3));

1. Now querying data and updating the age

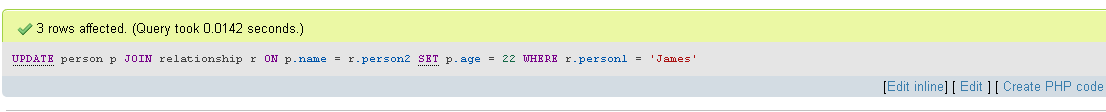
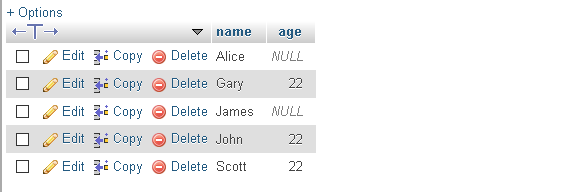
UPDATE person p

JOIN relationship r

ON p.name = r.person2

SET p.age = 22

WHERE r.person1 = 'James';



The activity consists of another question which is about comparing pros and cons with the carried solution.

|  |  |
| --- | --- |
| Relational database | Graph database |
| * It is time consuming to create tables and crating relationship in between them. * The database design or simply calling table and relation format is easy to understand. * As compared with increasing data it is difficult to understand or remind all the tables and relationship. * RDBMS from the above section was unable to retrieve every data in the database with single command. * While updating data because there may not be the require attribute in the table then user need to alter the table and then update the table. * It is very difficult to query data with the use of union, joins for retrieving data related to relationship. | * It is very fast to create nodes and relationships. * The database is designed in such a way that it may not easy to understand as there is not designed tables and many relationships as RDBMS. * It does not vary with large data as it is managed in same way for every size of data which gives the advantage of not reminding all the relationship. * Graph data returned every data with single line of code. * It is easy to merge and in case of absence of attribute or field it is added while inserting any new data to node or relationship. * With the node and relationship, it is easy and fast to query without complex functions. |

From the above table it is already justified that Graph database has concurred in this session as it was very time consuming and brainstorming for relational database creation and query but in case of graph database it was just opposite where the relational database creation statement is of 37 line but in case of graph database it is just 6 lines which is very important to notice.

Another question given in the same activity is to read a linked page which is (“7 Ways Your Data Is Telling You It’s a Graph”). After reading the whole article which can be found at neo4j , I can reflect few points that I have taken from this are as : graphs are changing the world and one of the way of showing love to our data is providing right home for them as we haven’t seen the evolution of relational database and graph database at once which is true because we had just applied RDBMS to our database where graph database innovation was suspected by us and we need to know that hierarchies in the real world are actually the graphs as claimed by karen lopez. There is not any data with single parent where every child is directly indirectly connected to each other which means a entity having relationship permanent as well as temporary which is not appliable in real world as we cannot work in the same place with two roles at same time. The real definition of relational database which meant to be the line or relation between tables not only table as they are only constraints so for the relationship in RDBMS we have to create and put them in that place which is not further practicable for big data with multiple relationship whereas another drawback that is claimed at the session is about the relationship property or name which is not visible in RDBMS and just opposite to it Graph database have given as much priority as nodes to relationships. So, the purpose of RDBMS can be considered as things that consists of constraints with them for data integrity. The classic tradeoff also focuses on relationship either discovering them or documenting which is usually asked by business. The actual purpose of session for knowing our data is graph are that the names used like network, tree etc. simply meaning important of data and their relationship, hierarchical database , vendor telling it won’t work, path-y questions, if some team member say it can be slow, developer ate not available with asked questions and a proof work done with neo4j which works. So, this all are the things to be considered for knowing the data are graphs. So, the main point to understand is that whenever there is question of recursion in the data, and there need to be master the data management as compared with social medias where one can have 2 followers but at the same time another can have millions of followers so there is the case to be use with graph database. Other cases for it are like for the purpose of IT network operations, real-time recommendations, if there is chance of any fraud or forensic, resource optimization and lastly promotion building.

### Reflections

To conclude the overall reflection about this week activity is about practical understanding of RDBMS and Graph database with the same dataset that how it can be used accordingly where the RDBMS comes to be very time consuming and complex during quiring but graph database justified with simple and fast solution for the given dataset. With the help of Neo4j session article, it gave the more insight view of the hierarchies’ database which are not actually that so that it should be known when to use graph database although relational database is not useless. Optimizing the competing advantage for the system with the existing data on data relationships.

## Week 16

### Activities

This is the first activity with mongo DB after graph database with Neo4j .The objective for this week is to know about mongo DB. After knowing mongo DB then we must create database and collection to operations like insert, update , read and delete.

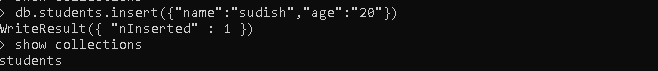
* Creating database with use command



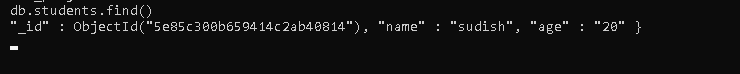
* View database with show dbs command



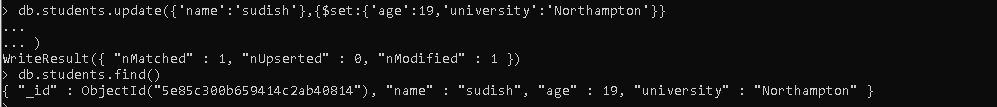
* Creating collection student with createCollection command and inserting with insert command



* Reading collection data with find command



* Updating collection data with name matching condition and $set to update data and if data do not present add it to the collection



* Deleting data from collection with matching column using remove command and then deleting collection with drop command.



* Similarly, database can be deleted with db.deleteDatabase() command .

### Reflection:

Though mongo DB is new us, but it didn’t even feel like new because of its easiness that I don’t even have to pressure myself to remind the queries. Queries use in mongo DB are much easier and shorter in compared with any other database like RDBMS .

Overall, the exercise was much more comfortable that I thought . Using command like use to create database if not created otherwise to switched into the database, createCollection command to create collection where we can create collection indirectly while inserting data into collection with db.collectionname.insert syntax. The idea of deleting is same as for other DBMS with drop, remove command.

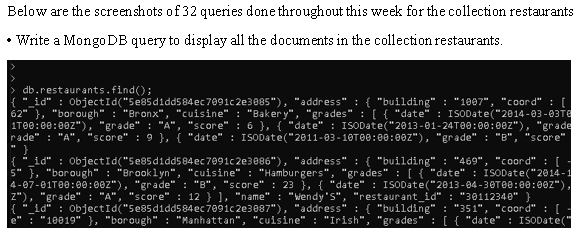
## Week 17

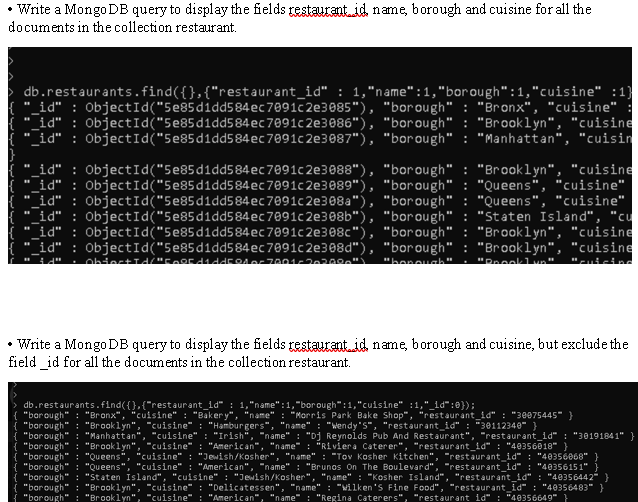
### Activities

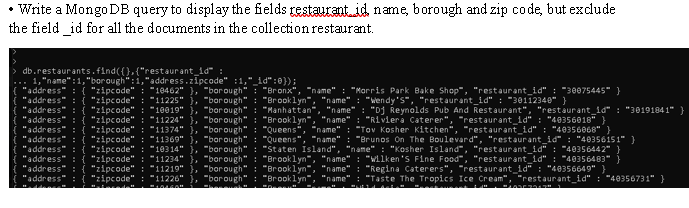
The aim of this activity is to import .json file which is given to us about restaurant . After importing, several queries are to be done to get deeper knowledge on filtering data. Below are the querying and importing syntax for this week.

* Importing json file into database with console. We can also import file with mongo db compass which is much easier.

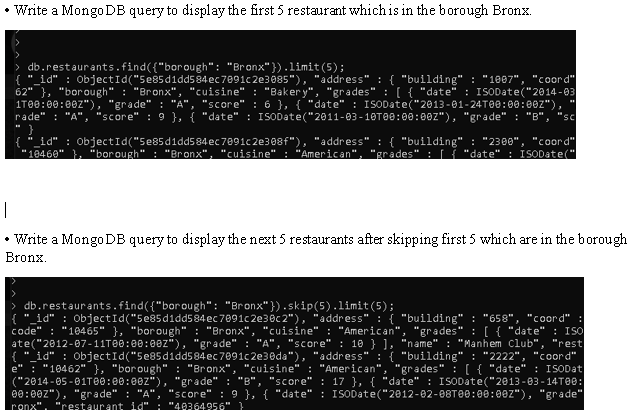
mongoimport --db testDb --collection restaurants --file C:/restaurants.json



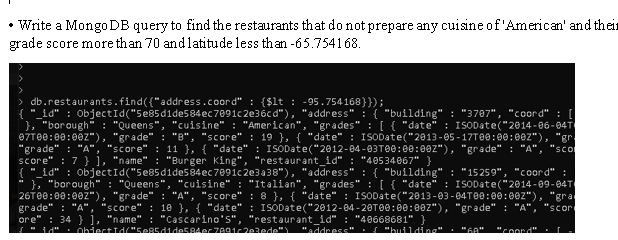




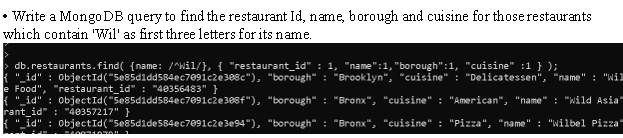




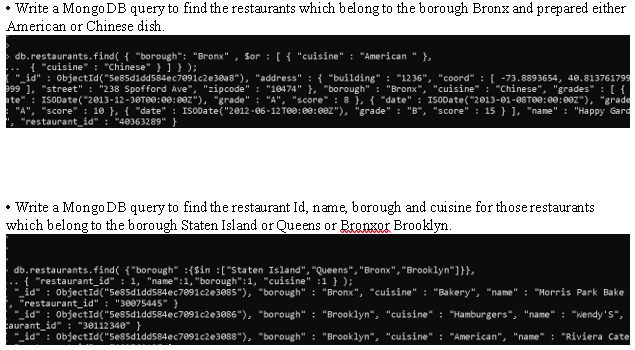




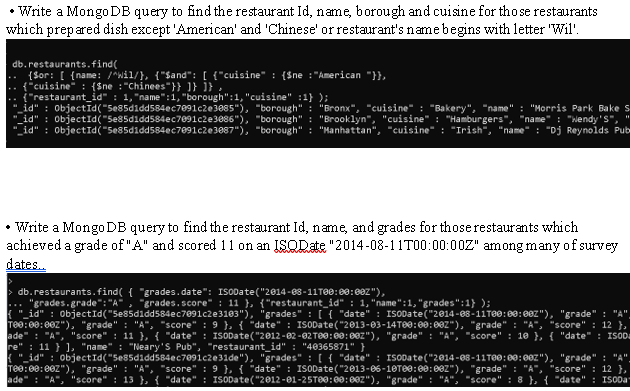


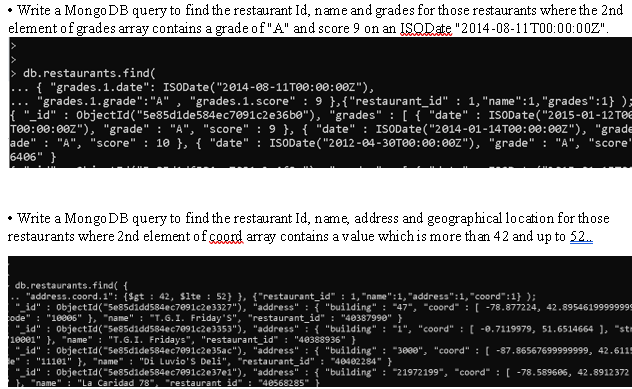


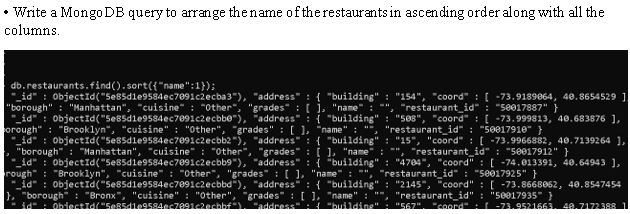




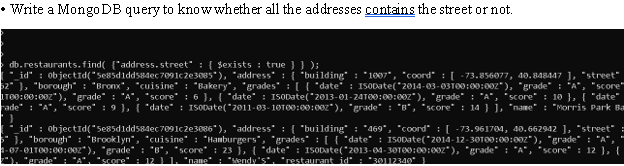


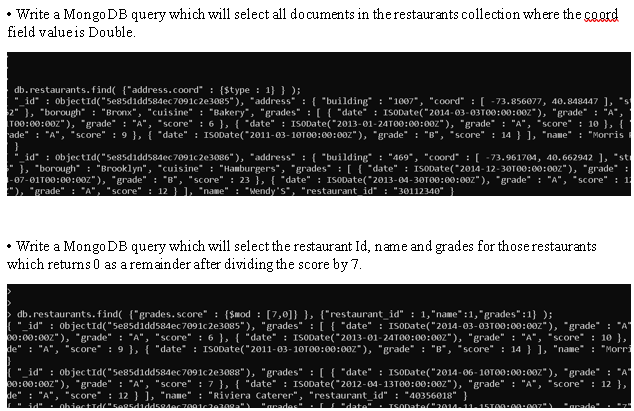


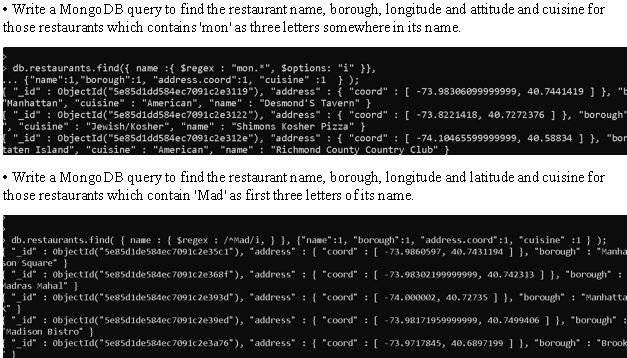












### Reflection

Overall query went smooth beside of syntax error usually due to missing of curly brackets after property list which were overcomes as soon as the error throw by the system. Aggregations or Operators like limit , skip was known. As I am friendly with python Django so some aggregation like gte, seems known. To include all the summary, this activity helped me a lot to know more deeply about different types of aggregations to filter data. With compare to relational database and graph database the aggregations look familiar without the addition of $ sign here. $regex operator was new to me, so I got to know about its property to match string pattern . $mod operator was also new which can help to get odd or even number. Likewise, $type was also new which is for querying document by their BSON type.

One problem that occurred was while ordering the data in descending order which shows error like unknown top level operator $ query while I was using command to order with $order by command inside the find aggregation which was solved with .sort operation outside the find aggregation.

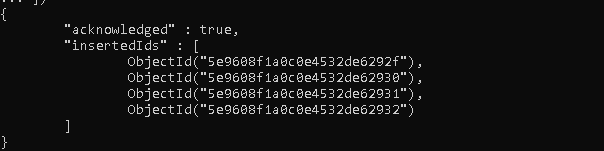
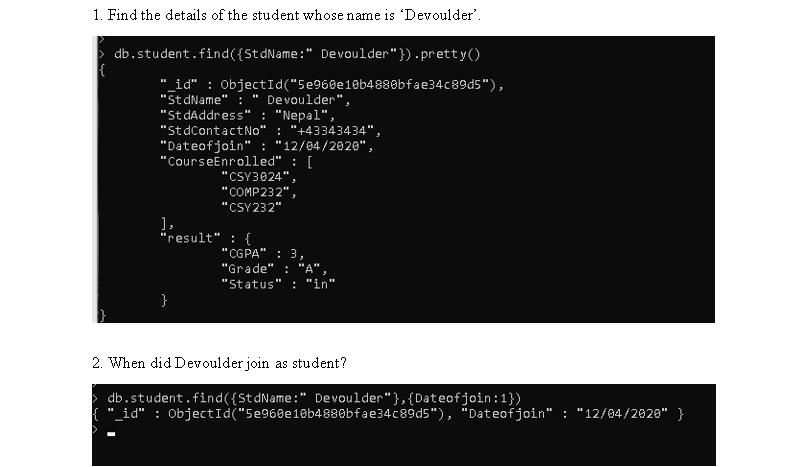
To conclude in one sentence, this activity developed my skills a lot regarding query and the operators and aggregation used while using mongo DB.

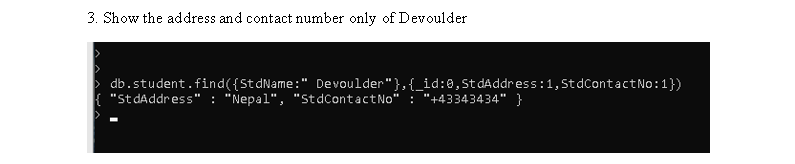
## Week 18 and 19

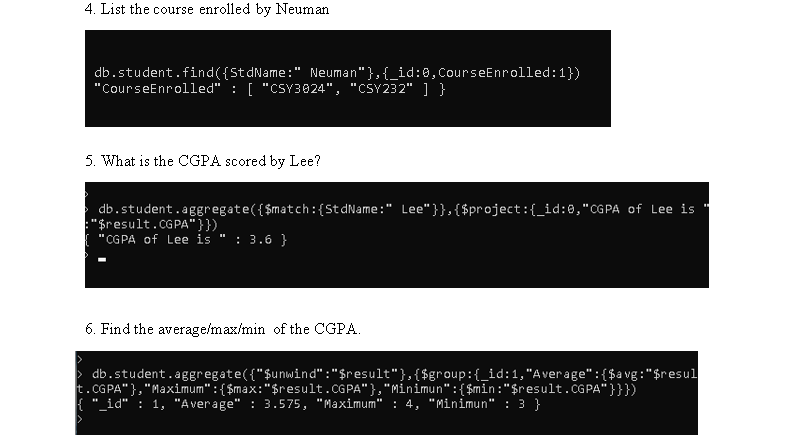
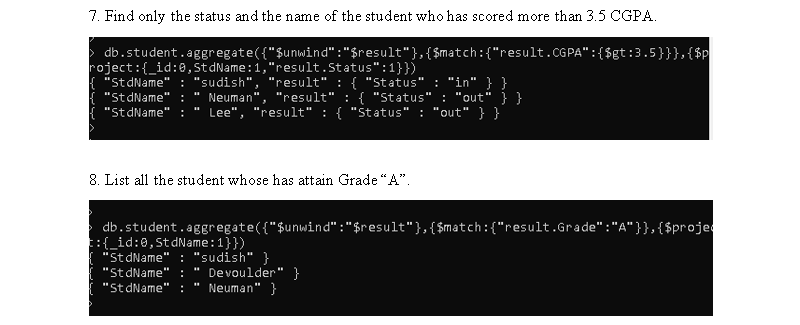
### Activities

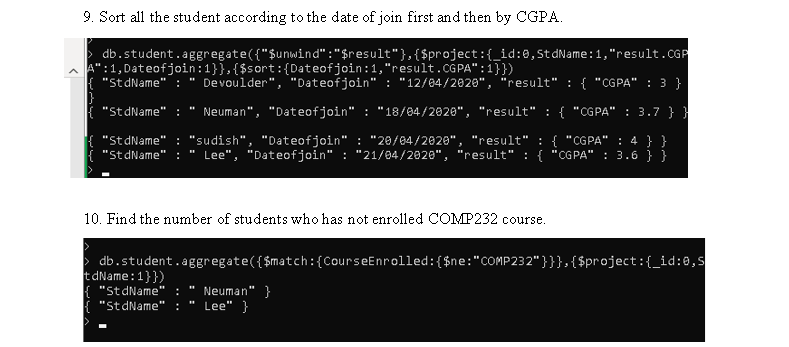
For this week, activity is given by our tutor to create database for student with provided attributes and query 10 set of questions. After the completion of this activity student will be able to get more knowledge about mongo DB function and operations.











### Reflection

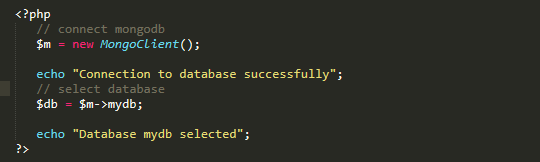
The 10 queries were successfully done without errors although project and group aggregation are new to me, but small research helped me a lot. The query 6 was confusing for me because I didn’t knew that whole collection data can be grouped by specifying value 1 to its id inside group function but at last all the queries were smoothly completed and I am able to know more about mongo DB queries and its aggregation with different technique. As I used to insert data one by one but with the help of insert many it made insertion a lot easier. Overall, this activity helped me to explore more on mongo DB functionalities.

## Week 20

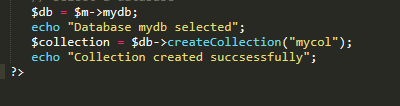
### Activities

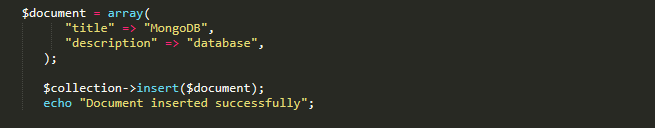
The activity for this week is to use mongo DB with PHP where first the installation of mongo DB php driver is to be installed for which “php\_mongo-1.6.8” version is used. The php\_mongo.dll is save to the ext. folder which is default for PHP extension directory upon which we must add the extension in the php.ini file. After this we can start our activity. For the output check, output data will be echo accordingly with given statement as because of the easiness to check the result.

Starting with establishing connect with database upon success of this the output in the echo is displayed which is connection successful and database selected. If the driver not installed properly then it shows error as “ Uncaught error Mongo dB class not found”.

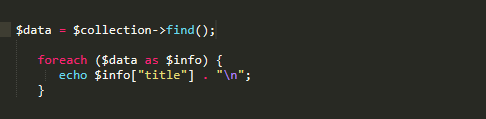


With the database connection establishment now, we can create new collection where data can be created with the syntax below which is as same as in mongo dB command shell.



After the collection creation, it is easy to insert data into collection although we have not designed database structure yet but on insertion of data to collection it automatically adds the field to the database. Below document of array contain the data which is inserted with insert function.

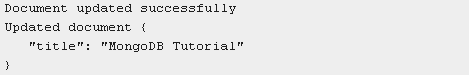
Before we have inserted data, which was successful as it echoed the expected given statement but to ensure if it either inserted or not, let us read the collection with the find function and looping the collection result so that if there are huge data all can be collected. The output data is given below the actual code.



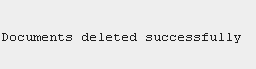


CRUD is the operation which we do in the database to manage the data. The read operation is done already after creation (insert) and now in order to update the present data we can simply use update function where we matched the data with the title field and then updated the array with set function. Below is the syntax to update and the output of the updated data.





Deleting the record from the collection using remove function after matching the title field of the collection data.



### Reflection

Overall, the activity went smooth with the idea of how the mongo DB can be used with php. There was no problem while doing the CRUD operation for the database as it is as same as queuing or doing CRUD operation the mongo DB command shell which we already did in the previous week . The use of aggregation and other method are as same as with the above process which can be further workout after the completion of each exercise and assessment.

## Week 21

### Activities

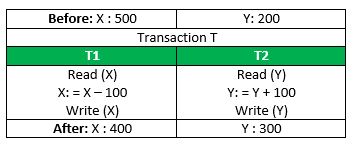
This is the last week for the semester where the database transaction management is introduced. For the activity session after the end of class, two question have been given which is related to transaction properties and he actions involved in them.

**List some database transaction properties, using an appropriate example to illustrate what they mean and why they are important.**

A transaction is a very small and logical work unit that accesses and manages data in database which may contain several low-level tasks. . Transactions use read and write operations to access data to ensure the consistency within a database, such properties are observed before and after the transaction which are called ACID properties.

Below are the properties explained with examples:

**Atomicity**

This property explains that the entire process takes place at once or does not happen at all. For the example consider transaction T, consisting of T1 and T2: 100 switches from account X to account Y.

From the table above, t1 and t2 contains several process inside one operation, if the transaction fails or stops after T1 completion where T2 is not completed yet then this causes inconsistent because the amount from T1 is already deducted but never switches to another account for which the transaction must be executed complete otherwise none. So, this is the principal of automaticity.

**Consistency**

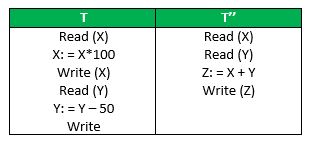
Consistency defines the correctness of database before and after the transaction where integrity constraints must be maintained. From the above example we can clarify that, the total amount before any action is 700 where after the transaction X has 400 and Y has 300 and on total they have 700 so there is consistency .If second transaction was failed there wouldn’t be any consistency.

**Isolation**

Isolation defines the multiple transaction without interference maintaining consistency of the database. From the table below, we can experiment that T and T” are two transaction and if T has been executed until Read (Y) which then passes to another transaction T” then T” reads X correctly but Y is read incorrectly due to which the result at end with both transaction is like below:

Suppose X = 100 and Y = 100

T’’ : (X+Y = 50, 000+100=50, 100)  
T : (X+Y = 50, 000 + 50 = 50, 50).  
So , there is inconsistent between T and T” due to 50 units loss. That is why isolation should take place and the visibility should only made after completion of whole operation.



**Durability**

This property explains that the once the transaction is completed then the update is written to the database even if there is some system failure which will take permanent effect. So, from the above example, the transactions made if completed without any system failure in between then it will be in effect otherwise no transaction will be made.

**The following is a serial schedule, which involves actions in several transactions.**

W3(z) → R4(z) → W4(z) → R1(x) → R2(y) → W3(y) → R1(z) → W5(x) →R 5(y)→R4(y)→W4(u)→R5(v)

**• Identify conﬂict pairs in above schedule.**

<W3(z),R4(z)>,<W3(z),W4(z)>,<R1(x),W5(x)>,<W3(z),R1(z)>,<W4(z),R1(z)>,

<R2(y),W3(y)>,<W3(y),R5(y)>,<W3(y),R4(y)>

**• Draw precedence graph.**

**• Test conﬂict serializability of the schedule.**

It is conflict serializability because the graph is not cyclic which can be seen above.

### Reflection

From this last activity, the knowledge about database transaction and the serial schedule is gained. Though we look as the transaction is happening with one operation but there are several small units happening in between the transaction. Principal of transaction property for the consistency of transaction so that there won’t be any problem whenever system fails after transaction or between the transaction. Identifying conflicts pair in the given transaction was very new to me where, I came to acknowledge that how the schedule for transaction conflict serializability. Overall , the activity has given very insight view about the database transaction management.

# References

1. Tutorialspoint.com. 2020. *Mongodb - PHP - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com/mongodb/mongodb\_php.htm> [Accessed 14 April 2020].
2. Docs.mongodb.com. 2020. *Aggregation — Mongodb Manual*. [online] Available at: <https://docs.mongodb.com/manual/aggregation/> [Accessed 14 April 2020].
3. Kb.objectrocket.com. 2020. *Mongodb Group By Multiple Fields Using Aggregation Function | Objectrocket*. [online] Available at: <https://kb.objectrocket.com/mongo-db/mongodb-group-by-multiple-fields-using-aggregation-function-464> [Accessed 15 April 2020].
4. Docs.mongodb.com. 2020. *Aggregation Pipeline — Mongodb Manual*. [online] Available at: <https://docs.mongodb.com/manual/core/aggregation-pipeline/> [Accessed 15 April 2020].
5. Kb.objectrocket.com. 2020. *$Project In Mongodb – What Is $Project And How To Use It In Mongodb | Objectrocket*. [online] Available at: <https://kb.objectrocket.com/mongo-db/project-in-mongodb-what-is-project-and-how-to-use-it-in-mongodb-462> [Accessed 16 April 2020].
6. Neo4j Graph Database Platform. (2015). *7 Ways Your Data Is Telling You It’s a Graph*. [online] Available at: https://neo4j.com/blog/7-ways-data-is-graph/ [Accessed 20 Apr. 2020].
7. https://github.com/mongodb/docs/blob/v4.2/source/reference/operator/aggregation/cond.txt. (n.d.). *$cond (aggregation) — MongoDB Manual*. [online] Available at: https://docs.mongodb.com/manual/reference/operator/aggregation/cond/ [Accessed 25 Apr. 2020].
8. “Aggregation Framework - What’s the $unwind Operator in MongoDB?” *Stack Overflow*, stackoverflow.com/questions/16448175/whats-the-unwind-operator-in-mongodb. [Accessed 25 Apr. 2020].
9. “$concatArrays (Aggregation) — MongoDB Manual.” *Https://Github.Com/Mongodb/Docs/Blob/v4.2/Source/Reference/Operator/Aggregation/ConcatArrays.Txt*, docs.mongodb.com/manual/reference/operator/aggregation/concatArrays/. [Accessed 25 Apr 2020.