**Group # - 7**

Project Name – Transport System

Database – Neo4j

**Introduction:**

What is Neo4j?

Neo4j is an open-source NoSQL graph database.

What is a graph Database?

* In computing, **graph database** is a database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data.
* Neo4j is used by companies like
  + Ebay
  + Walmart
  + Telenor
  + Cisco
  + Justdial

We live in a connected world. There are no isolated pieces of information, but rich, connected domains all around us. Only a database that embraces relationships as a core aspect of its data model is able to store, process, and query connections efficiently. Neo 4j is used in various sectors like Social Networks, Software analytics, Scientific research etc. Neo4j provides full database characteristics including ACID transaction compliance, cluster support, and runtime failover, making it suitable to use graph data in **production scenarios**.

**History of Neo4j:**

* + 2000 : Neo’s founders encountered problems (Performance)with RDBMS and started building the first Neo4j prototype
  + 2002 : They Developed the first ever version of Neo4j
  + 2007 : Formed Swedish-based company behind Neo4j
  + 2010 : Released Neo4j version 1.0

Some particular features make Neo4j very popular among users, developers, and DBAs:

* Materializing of relationships at creation time, resulting in no penalties for complex runtime queries
* Constant time traversals for relationships in the graph both in depth and in breadth due to efficient representation of nodes and relationships
* All relationships in Neo4j are equally important and fast, making it possible to materialize and use new relationships later on to “shortcut” and speed up the domain data when new needs arise
* Compact storage and memory caching for graphs, resulting in efficient scale-up and billions of nodes in one database on moderate hardware
* Written on top of the JVM
* Neo4j is used to build new and innovative applications that leverage connections in data such as recommendations, impact a analysis for network and IT operations, a real-time routing for logistics and the next generation business applications such as a master data management, identity & access management, and content management, fraud detection,a portfolio and risk management **.**

**Query Language used in Neo4j- Cypher Query**

* + SQL QUERY:
    - Select \* from emp where emp\_id=05
  + Cypher Query:
    - MATCH (B:BusStop) WHERE B.name="Burgstrasse“
* Cypher is inspired by number of different approaches and builds upon established practices for expressive querying.
* Cypher is inspired by a number of different approaches and builds upon established practices for expressive querying.
* Most of the words or key words like WHERE and ORDER BY are inspired by SQL query. Pattern matching borrows expression approaches from SPARQL.
* A number of the collection semantics have been borrowed from languages such as Haskell and Python.

**Example of Neo4j:**

Residence Permit

****

Student.name=Sumith

Student.name=Nanditha

**** 

**Transport System:**

* The evolution of Transport system started with human foot.
* Eventually animals started coming into existence as a means of Transport. Donkeys and Horses were used in about 4000 BC to 3000BC which was later evolved by Camels slightly later.
* Later Wheel was invented in Iraq .At the beginning wheels were made of solid pieces of wood put together to form a circle.
* The Egyptians invented the sailing boat which were made of sheets of linen. However it was used when sailing in one direction. When traveling against the wind the boat had to be rowed.
* The Romans are famous for the network of roads which was built by the roman legionaries so that the roman army used to march from one part of the empire to another.
* By the 18th Century roads were greatly used. Acts of Parliament gave right to improve and maintain certain Roads. Travelers had to pay to use them.
* In the 19th Century there was a major revolution by railways which made the travel more quick and faster. The First railway was from Liverpool to Manchester in 1830.By the end of 1840 most of the towns in Britain were connected.
* Although Cars evolved by the end of 19thCentury, it was greatly into existence in the 20th Century.
* Later in the 20th Century came the evolution of Flights which was a Major means of transport between different countries wherein later being evolved between all places of the world.
* By the 21st Century the Jumbo jet and the commercial suborbital space flight is a new means of transport which has evolved .

After having a brief introduction of the evolution of transport system ,we are going to focus on our model which is providing door-to door service.

Our Inspiration of building a Model which provides Door-to-Door Service was from our real time experience and some of the travel websites available now as listed below:

**Existing Systems:**

Before designing our data model we did a case study on the existing travel guide websites and Applications e.g.

* Make my trip
* M-Indicator

**Make my trip** provides users the facility to book flight, train, buses and also provides an option to reserve a hotel online.

* It is recognized as a best travel planner website.

**M-Indicator** is a mobile application that provides the schedule info about the Mumbai Local trains and Bus Networks across Mumbai and sub urbs of Mumbai.

Our Analysis:

1)We understood that from these systems they will enable the user to select their source and destinations to the destination they wish to travel .e.g Make my trip will book a flight from Mumbai International airport to Frankfurt International Airport but it will not guide the user how to reach to the source airport and to the end point he wish to travel from the destination.

2) Make my trip provides the booking options only for the business firms they are associated with.

Make my trip does not provide the routing information about the Mumbai local trains and municipal bus transport network across the Mumbai as the M-Indicator does and M-Indicator does not guide about the inter city travels.

Our Conclusion:-

These two points lead us to the design of our data model. We have designed our data model

such that :-

* It should provide all available travel network routing information across the world.
* It should help the user to select his nearest starting point to reach the source (Train Station, Bus Stop, Airport) and guide the user how to reach his end point from his Destination.

The Challenge we faced while designing such a data model:

* How to decide the nearest starting point?

After having a Brain Storming Session we decided with below conclusions:

1. **The Address**-We could store the addresses across the globe and will ask user to enter the

address from where he wish to start and address will be mapped to the nearest source. The solutions seems simple at the brain storming but actual implementation would be an headache.

* Storing the addresses across the globe won’t be feasible and user could not query the address in the format it is stored in the database making it impossible to retrieve the answer.
* We dropped the address idea.

1. **The Postal Codes:-**

* Postal codes is the most common thing in the world. Each country has this system which in some extent differ country wise.

We did a case study of the address of 3 separate countries which is India,Germany,UK and analysed how the address in these countries are defined and arrived with the final solution .

Final Solution:

* Finally we decided to relate the source and destination (Train Station, Bus Stop, Airprort ) with the postal codes and ask the user to select the country, city and enter the postal code for the source and destination, so that he can find the nearest start point and simply reach to his end point from his destination, thus providing him with the Door to Door Service.

**Example of our Data Model:**

Next

Node 2

Node 1

Distance=1.6Kms

Name- Bismarckplatz

Name-Bonhoefferstrasse

**Requirements/Queries:**

|  |  |
| --- | --- |
| **Task** | **Cypher Query** |
| Create Node | 1. CREATE(n:Bus {ID:"35"} ); 2. CREATE(n:BusStop {name : "Burgstrasse",seating:"Yes"}); |
| Create Relationship | MATCH (B:BusStop)  WHERE B.name="Burgstrasse"  MATCH (D:Bus)  WHERE D.no = "35"  CREATE (B)-[r:STOPS\_AT]->(D)  MATCH (B:BusStop)  WHERE B.name="Burgstrasse"  MATCH (D:BusStop)  WHERE D.name = "Bismarkplatz"  CREATE (B)-[r:NEXT]->(D) |
| Create Property for Node | MATCH (Train { name: 'S3' })  SET Train.LadiesCompartment = 'Yes'  RETURN Train  MATCH (Tram { name: '5' })  SET Train.Bicycle = 'Yes'  RETURN Train |
| Create Property for Relationsip | MATCH (a:BusStop{name:"Burgstrasse"})  MATCH (b:Bus{no:"35"})  MATCH a-[r:STOPS\_AT]->b  SET r.DeptTime='9:00'  MATCH (a:BusStop{name:"Burgstrasse"})  MATCH (b:BusStop{no:"Bismarkplatz"})  MATCH a-[r:Next]->b  SET r.Distance='3Km' |
| Remove Property value of a Relationship | MATCH (a:BusStop{name:"Burgstrasse"})  MATCH (b:Bus{no:"35"})  MATCH a-[r:STOPS\_AT]->b  REMOVE r.DeptTime  MATCH (a:BusStop{name:"Burgstrasse"})  MATCH (b:BusStop{name:"Blsimarkplatz"})  MATCH a-[r:NExt]->b  REMOVE r.Distance |
| Removing Property value of a Node | MATCH (BusStop { name: 'Bonhoefferstrasse' })  REMOVE BusStop.DepTime  RETURN BusStop  MATCH (Train { name: 'S3' })  REMOVE Train.LadiesCompartment  RETURN Train |
| Update property of a Relationsip | MATCH (a:BusStop{name:"Bonhoefferstrasse"})  MATCH (b:BusStop{name:"NeckarSpitze"})  MATCH a-[r:NEXT]->b  SET r.Distance='3KMS'  MATCH (a:BusStop{name:"Bonhoefferstrasse"})  MATCH (b:Bus{no:"35"})  MATCH a-[r:STOPS\_AT]->b  SET r.DeptTIme='10:00' |
| Update Property of a Node | MATCH (n { name:'Burgstrasse'})  SET n.Seating = 'Yes'  RETURN n  MATCH (n { No:'S3'})  SET n.Bicycle = 'Yes'  RETURN n |
| Updating the label name | MATCH (n {no: '35' })  SET n :Bus  RETURN n  MATCH (n {name: "Burgstrasse" })  SET n :BusStop  RETURN n |
| Delete a Node | MATCH (n:Bus{ID:"35"})  DELETE n; |
| Deleting Relation between nodes | MATCH (n:Train {TrainNo:519})  MATCH (r:TrainStation {name:'HeidelbergHauptbanhof'})  MATCH n-[rel:STOPS\_AT]->r  DELETE rel |
| Delete all nodes and relationship | MATCH (n)  OPTIONAL MATCH (n)-[r]-()  DELETE n,r |
| Calculate distance from Stop A to Stop B | MATCH (a:BusStop {name:'Bonhoefferstrasse'}), (d:BusStop {name:'HeidelBergHauptbanhof'})  MATCH p = allShortestPaths((a)-[:STOPS\_AT\*]-(d)),stops = (a)-[:NEXT\*]->(d)  RETURN EXTRACT(x IN NODES(p) | CASE WHEN x:Bus THEN 'Bus ' + x.id  WHEN x:BusStop THEN 'BusStop ' + x.name  ELSE '' END) AS itinerary,  REDUCE(d = 0, x IN RELATIONSHIPS(stops) | d + x.distance) AS distance |
| Query to return Bus Stops, Bus No, Operted\_by | MATCH (a:BusStop{name:'Bonhoefferstrasse'}),(d:BusStop {name:'HeidelBerg Hauptbanhof'})  MATCH p = allShortestPaths((a)-[:STOPS\_AT\*]-(d))  WITH p, FILTER(x IN NODES(p) WHERE x:Bus) AS buses  UNWIND buses AS Bus  MATCH (Bus)-[:OPERATED\_BY]->(o:TransportOperator)  RETURN EXTRACT(x IN NODES(p) | CASE WHEN x:BusStop THEN 'BusStop' + x.name  WHEN x:Bus THEN 'Bus' + x.id  ELSE '' END) AS itinerary,  COLLECT ('Bus' + Bus.id+ ':' + 'TransportOperator' + o.name) AS Operators |
| Query to return BusStop, Bus No, Departure Time, Arrival Time and Operted\_by | MATCH (a:BusStop{name:'Bonhoefferstrasse'}), (d:BusStop{name:'HeidelBerg Hauptbanhof'})  MATCH p = allShortestPaths((a)-[:STOPS\_AT\*]-(d))  WITH p, FILTER(x IN NODES(p) WHERE x:Bus) AS buses  UNWIND buses AS bus  MATCH (bus)-[:OPERATED\_BY]->(o:TransportOperator)  RETURN EXTRACT(x IN NODES(p) | CASE WHEN x:BusStop THEN 'BusStop ' + x.name  WHEN x:Bus THEN 'Bus ' + x.id  ELSE '' END) AS itinerary,  HEAD(RELATIONSHIPS(p)).DeptTime AS departure\_time,  LAST(RELATIONSHIPS(p)).ArrTime AS arrival\_time,  COLLECT('Bus' + bus.id + ':' + 'TransportOperator' + o.name) AS operators |
| To Calculate all the shortest path from stop A to Stop B | MATCH (a:LandMark {name:'Acharya College' }),(d:BusStop {name: 'Bonhoefferstrasse'})  MATCH p = allShortestPaths((a)-[:STOPS\_AT\*]-(d))  RETURN EXTRACT(x IN NODES(p) | CASE WHEN x:TrainStation THEN 'TrainStation ' + x.name  WHEN x:Train THEN 'Train ' + x.TrainNo  +'(Bicycle Carrying ' + x.Bicycle + 'Pram Support' +x.PramSupport + 'Pantry Car' + x.PantryCar + 'Dedicated Ladies Compartment' + x.LadiesCompartment + ')'  WHEN x:Bus THEN 'Bus No ' + x.id  WHEN x:BusStop THEN 'BusStop ' + x.name  WHEN x:AirPort THEN 'AirPort '+x.name  WHEN x:AirPlane THEN 'AirPlane '+x.FlightNo  WHEN x:LandMark THEN 'LandMark' +x.name  WHEN x:Taxi THEN 'Taxi ' +x.name  WHEN x:AutoRickShaw THEN 'AutoRickshaw' +x.name  ELSE '' END) AS itinerary;   MATCH (a:LandMark {name:'Acharya College'}),(d:BusStop {name:'Bonhoefferstrasse'})  MATCH p=allShortestPaths((a)-[:STOPS\_AT\*]-(d))  MATCH e=(z)<-[:OCCURS\_AT]-() WHERE (z in NODES(p))  MATCH n=(x)-[:OPERATED\_BY]->() WHERE (x in NODES(p))  RETURN p,e,n |
| Main query to get the paths , events and eatery etc | MATCH (a:LandMark {name:'Acharya College'}),(d:BusStop {name:'Bonhoefferstrasse'})  MATCH p=allShortestPaths((a)-[:STOPS\_AT\*]-(d))  MATCH e=(z)<-[:OCCURS\_AT]-() WHERE (z in NODES(p))  MATCH n=(x)-[:OPERATED\_BY]->() WHERE (x in NODES(p))  WITH nodes(p) as pfinal,nodes(e) as efinal,nodes(n) as nfinal  WITH pfinal+efinal+nfinal AS finalfinal  RETURN finalfinal |
| Query to get distances and times. | MATCH (a:BusStop {name:'Bonhoefferstrasse'}), (d:BusStop {name:'HeidelBerg Hauptbanhof'})  MATCH p = allShortestPaths((a)-[:STOPS\_AT\*]-(d)),stops = (a)-[:NEXT\*]->(d)  RETURN EXTRACT(x IN NODES(p) | CASE WHEN x:Bus THEN 'Bus ' + x.id  WHEN x:BusStop THEN 'BusStop ' + x.name ELSE '' END) AS itinerary,  HEAD(RELATIONSHIPS(p)).DeptTime AS departure\_time,  LAST(RELATIONSHIPS(p)).ArrTime AS arrival\_time,    REDUCE(d = 0, x IN RELATIONSHIPS(stops) | d + x.distance) AS distance |

**Description of Data model of Database:**

Primary Focus of our Data Model:

The primary focus of our Data Model is to provide travelers Door to Door accessibility to the travel Destination.

Challenges of our Data Model:

* In transport System there are two points ex Point A, Point B which is to be traversed; initially before the implementation of the actual model we were unaware of what are points A and B.
* We were unaware of the “to” between Points A and Point B.
* Establishing connection between the UI and the Database.
* Mention about php and java

Solutions for the challenges:

* After the initial Brain-storming session we decided the Points A and B are the source and Destination respectively Ex: Train stops, bus-stops etc
* The “to” between A and B is the relationship between A and B respectively Ex: Bus, Tram, Roads, auto rickshaw etc
* In order to emphasize the focus of our Model we introduced two things

1. Geography

2.Transport

* Geography is narrowed down from Country->State/Province->City->Street/Area/Landmark

Based on case study of different countries (India, UK, and Germany).

* There are various modes of transport (Tram,Train,Bus,Flight,Autorickaw)

After having a case study on transport system of three different countries i.e. India, UK, and Germany we analyzed how to offer door-to door in these countries with the below input:

1.India – In India the address is identified using postal codes and landmarks

Ex: If we want to traverse from srinagar to vijaynagar where in the source and destination address is :

Source address: 5th Main,Nagendra Block ,Srinagar,Bangalore-560050

Destination Address: 4th Main,GKW Layout,Vijaynagar,Bangalore-560040

The above address is identified using the nearest landmark wherein the nearest landmark for the destination address will be GKW Layout Bus-Stand and the Postal Code is 560040

2. UK- In UK every post code is connected to every building For Ex:

Address – Barclays Bank PLC,PO Box 12,Poole,Dorset ,BH15 2BB

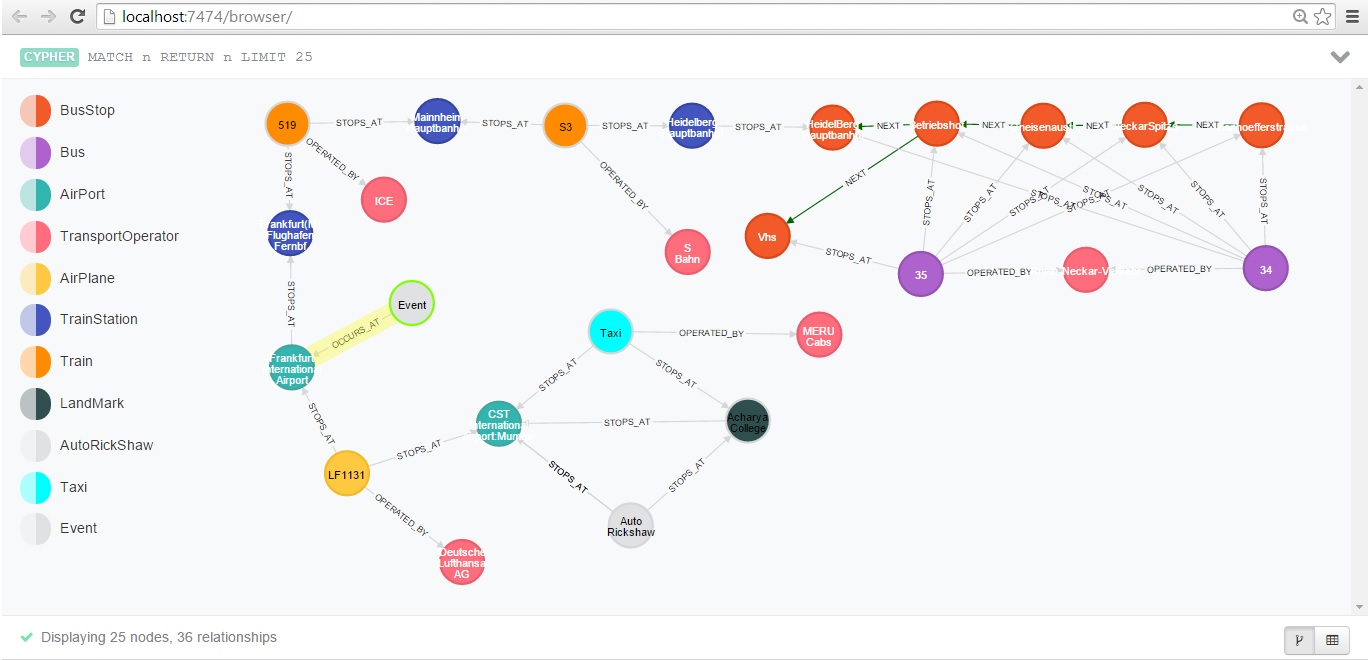
UK Postal Code –Using the postal code BH152BB ,it matches the entire building wherein here with this postal code it matches the exact building Barclays House Poole

3. Germany – In Germany the Postal Code Matches a Town and its associated Street

Address :Schlossstrasse ,Wieblingen ,69123

German post code Matches with a Town and the associated street which directly redirects to the above address

**Design and Implementation of our model: discuss with the team**

****

In the above DataModel if we take an example if we want to travel from Bonhoefferstrasse(Germany) to Acharya Collage(Mumbai),as per our datamodel design if we query we will start as below:

* Start Bonhoefferstrasse ,each stops are linked with relationship Next
* Bus # 34 will be selected as Mode of Transport which is operated by Rhein-Necker-Verkehr Gmbh from Bonhoefferstrasse to Heidelberg Hauptbahnhof as it is related with STOPS\_AT relationship for our destination route
* From Busstop Heidelberg Hauptbahnhof to Train Station Heidelberg Hauptbahnhof ,we have to walk .
* From TrainStation Heidelberg Hauptbahnhof to TrainStation Manheim Hauptbahnhof we have to travel in Train S3 operated by Sbahn
* From TrainStation Manheim Hauptbahnhof to TrainStation Frankfurt Hauptbahnhof ,we have travel in Train 519 operated by ICE
* From TrainStation Frankfurt Hauptbahnhof to Frankfurt International Airport we need to walk.
* From Frankfurt International Airport to CST International Airport,Mumbai we need to travel Using Flight# LF1131 operated by Deutsche Lufthansa AG
* From CST International Airport,Mumbai we have two means of Transport to reach Acharya Collage
  + By Taxi operated by Meru Cabs
  + Auto rickshaw

**Criteria Fulfillment:**

For the result to be fetched from database at least one of the field has to be entered:

1. Postal code of the source and destination has to be entered

2. Source has to be entered depending on the country

Eg: INDIA : Landmark, Bus stop or the station name

Germany: Postal code or street name

UK : Postal Code

3.Destination has to be entered depending on the country

Eg: INDIA : Landmark, Bus stop or the station name

Germany: Postal code or street name

UK : Postal Code

4.Design - As we had planned to provide user with door2door transport solution we had to model our data Modelas,

Nodes:

Bus Stop - Every bus stop in the world is a node in the database. Bus stops will be the label and the description name will consist of stop name.

Ex: **BUS STOP**

NAME : Bonhoefferstrasse

Train - Every train in the world is a node in the database. Train Station will be the label and the description name will consist of stop name.

Properties:

Train Number : S3

Bicycle : Yes/No

Pram Support: Yes/No

Panty Car: Yes/No  
Ladies Compartment: Yes/No

Ex: **Train**

Transport Operator - All transport medium will have an operator associated with it, hence in our model operators are node and this nodes have relationship with the transport medium.

Ex: **Transport Operator**

Properties:

name : Rhein-Neckar-Verkehr

Bus- Every Bus in the world is a node in the database. Bus will be the label and the property name will consist of Bus number.

Properties:

Id: 34  
Pram Support :Yes/No  
Reservation Senior Citizens: Yes/No

**BUS**

Properties:

name : Rhein-Neckar-Verkehr

Flight-Every Flight in the world is a node in the database. Airplane will be the node and the property name will consist of Flight number.

Properties:

Id: 34  
Pram Support :Yes/No  
Reservation Senior Citizens: Yes/No

**FLIGHT**

Taxi- Every Taxi in the world is a node in the database. Taxi will be the node and the property name will consist of Name.

**TAXI**

Properties:

Name:Taxi

Auto rickshaw- Every auto rickshaw in the world is a node in the database. Auto rickshaw will be the node and the property name will consist of Name.

Properties:

Name:Autorickshaw

**AUTORICKSHAW**

**Relationships:**

STOPS\_AT- Relationship is established between Transport Medium to Stops(Ex:BUS – STOPS\_AT-BusStop)

NEXT- Relationship is established between Two stops(Ex: Bonhoefferstarsse- NEXT-NeckarSpitze)

OPERATED\_BY-Relationship is established between transport medium and Transport operator(Ex: Bus-OPERATED\_BY-RNV)

OCCURS\_AT-Relationship is established between Events and Stations wherein Relationship OCCURS\_AT will have a property type(Ex: Events-OCCURS\_AT(Snowstorm)-Frankfurt Airport)

**Advantages of our Database:**

* All Transport networks under one umbrella
* Finds nearest starting point
* Automatically search shortest path
* Any node can be added, updated and deleted on the go
* Easy to maintain
* Schema free
* No complex joins as in SQL
* Easily scalable

**Dis- advantages of our Database:**

* Need knowledge of Cypher query language.
* Syntax is case sensitive.
* No Sharding at Neo4j level.
* Limited colors in Neo4j web GUI.
* Unlike SQL where we can specify user and access, Neo4j does not have user management.
* Complex Indexes which are not deleted.
* Complex Neo4j JSON object response.

**Comparison of Neo4j with other Databases:**

* Neo4j Schema in transport is easy to visualise and represent, as the core concept of a graph and the transport is the same.
* The Cypher query and web GUI interface is easy to understand
* In Document databases we would have created documents for each bus stop, train stations and then put all the train, bus details on them (for larger stops this could be a problem , so we would need to split the document).
* In Key Value we would have would have to create a Key value bucket for geography(post codes, town ) and with the associate geography key to other buckets of train )

**CONCLUSION:**