***21BLC1424 - GURSHAAN SINGH BHASIN***

***21BLC1497 – MOHAMED SHOUKAT ALI***

***COURSE CODE : BCSE302L***

***DA-II***

**GRAPHICAL USER DATABASE SYSTEM**

* **AIM**

To create a Database using the Neo4j Graphical Database Management System and incorporate it into Python code by calling the API of the System , to retrieve data as per the users requirements and create a basic GUI Interface that visualizes these outputs as entered and worked upon by the native language of Neo4j – “Cypher” within the Python codes directly linked to the DB via the API key.

* **INTRODUCTION**
* **Neo4j**

Neo4j is a graph database management system designed to store, retrieve, and manage data represented as nodes, relationships, and properties. Unlike traditional relational databases, Neo4j's graph structure allows for flexible and efficient querying of complex, interconnected data. Nodes represent entities, relationships define connections between nodes, and properties provide additional information about nodes and relationships.

Using Neo4j's API, developers can interact with the database programmatically, integrating graph-based functionalities into their applications. This API enables developers to perform operations such as creating, updating, and querying graph data using programming languages like Java, Python, or JavaScript. By leveraging the API, developers can harness Neo4j's powerful graph capabilities within their applications, facilitating tasks such as recommendation systems, social network analysis, and real-time network visualization. Overall, Neo4j's API empowers developers to build sophisticated applications that leverage the advantages of graph databases, including flexible data modelling, efficient traversal of relationships, and powerful graph algorithms.

In this report you as will see ahead , we have worked extensively in Neo4j , applying the concepts of API and JSON directly to utilize the Database management system of Neo4j which is an excellent visualization tool for Databases as compared to the regular MySQL and SQL+ systems. It allows creation of database directly same as these other systems , with a much more efficient graphical node wise format. It further also allows direct CSV file uploads and creation of Database accordingly much simplified and the best of all , allows API exportation to codes which means these Databases once created can also be worked upon through codes to modify and/or even retrieve the required outputs using its native language called “Cypher” for the same.

* **API**

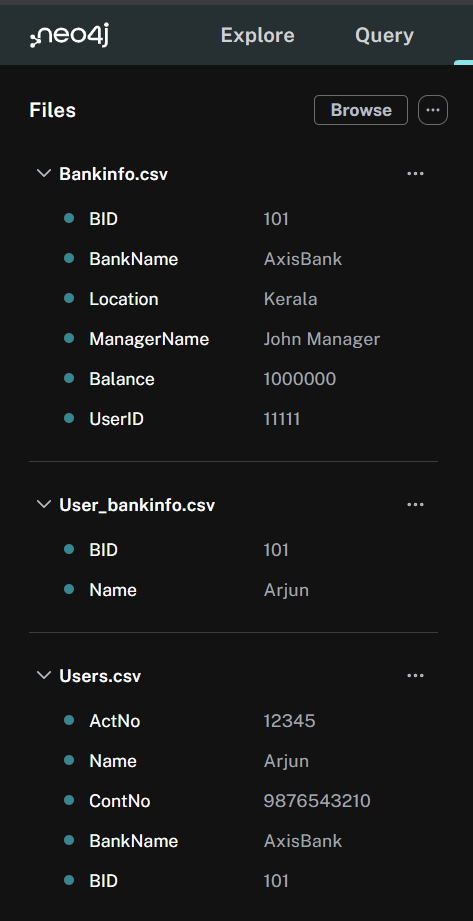
APIs, or Application Programming Interfaces, serve as intermediaries that enable different software applications to communicate and interact with each other. They define the rules, protocols, and tools for exchanging information and accessing functionality. APIs find widespread use across various domains, facilitating integration between systems, platforms, and services.

Their versatility makes APIs invaluable in modern software development. Developers leverage APIs to incorporate functionalities from existing systems into their own applications without needing to understand the internal complexities. This accelerates development cycles, fosters innovation, and enhances the overall user experience. Additionally, APIs promote interoperability, allowing disparate systems to seamlessly exchange data and services, thereby fostering collaboration and integration in the digital ecosystem. In essence, APIs democratize access to technology and empower developers to create more sophisticated and interconnected software solutions.

* **EXPLANATION OF PROJECT WORK**

1. **Neo4j Outlook –**

Here is an overview of how the importing functionality of Neo4j DBMS works :



Thus as seen above , for this project the CSV files taken into consideration were that of Users and their BankInfo with another file called User\_BankInfo created for the purpose of storing the Relationship between the two other “Nodes” of the Graph we are going to form.

Neo4j is similar in working to the other DBMS’ such as MySQL and SQL+ , in the regards that it too creates a Primary key and Foreign key for tables , with the advantage of not having to tediously enter these values while creating the tables and defining primary key , foreign key constraints and inserting values into these tables step by step for the same. This is done directly through CSV import and then the attributes to define as PK or FK are then visually decided upon as seen below :

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

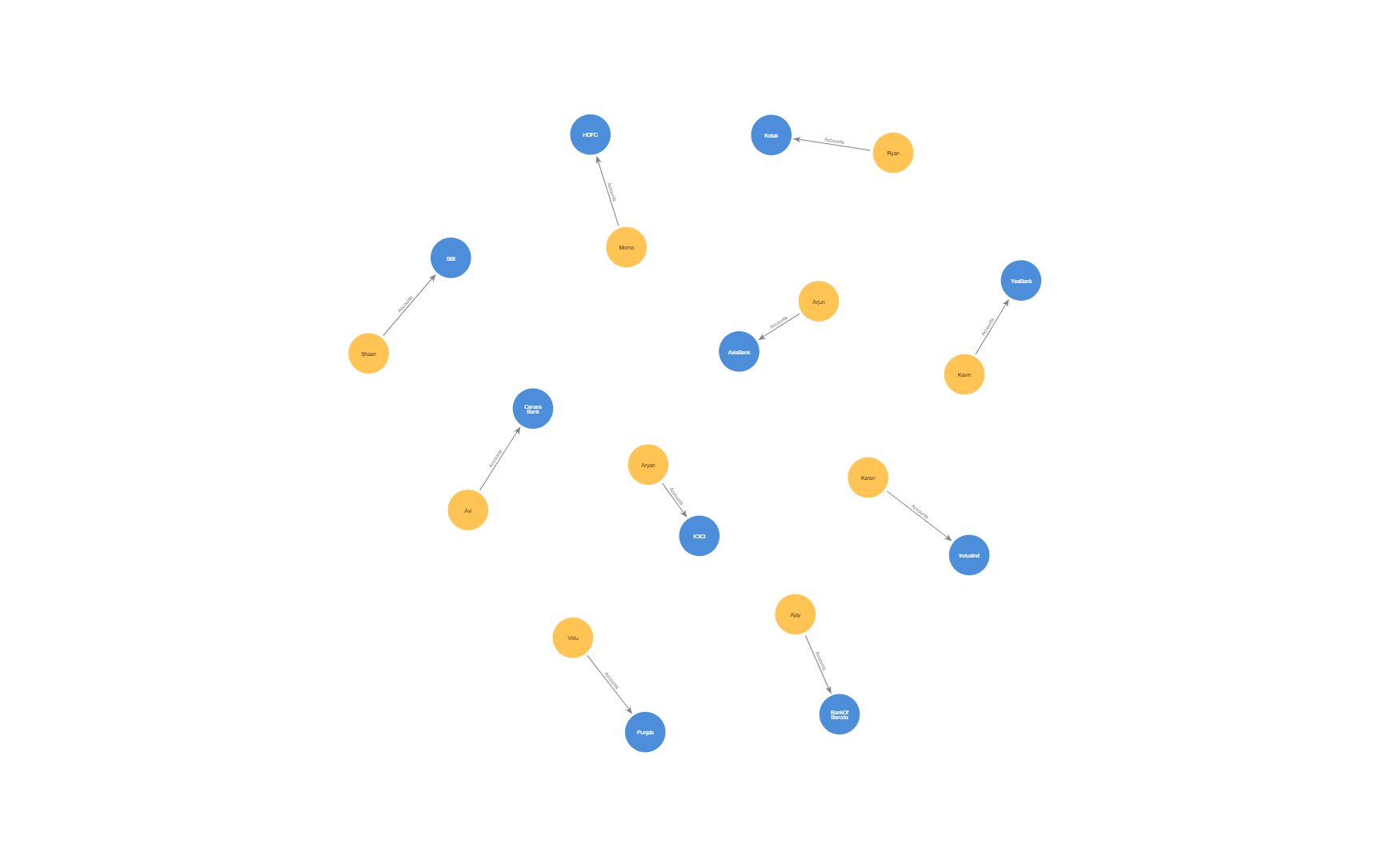
A screenshot of a computer

Description automatically generated

So as seen above , once the CSV Files were imported , we were able to allot the Nodes to their Data and create the Attributes that store them , also allowing to chose which of them to allot as Primary keys , also while doing the same process for the Relationship between them we can choose which attributes to connect from either of the nodes as the keys and create the link between. Just this creation of the DB , saves us ample amount of time in comparison to the traditional method of the same in other DBMS’.

1. **Graph Outlook –**

Once the DB has been created by the imported CSV’s and definition of keys and relationships. There is an option to run import which then stores these values into the DB and allows creation of the Graph , which is the Highlight of this DBMS and this here would look as such :



Thus , from this Graph we can clearly see which user is connected to which product showing direct DB Visualization of data. There are also other layouts of this Graph which can be explored based on the users requirements such as :

A group of circles with text

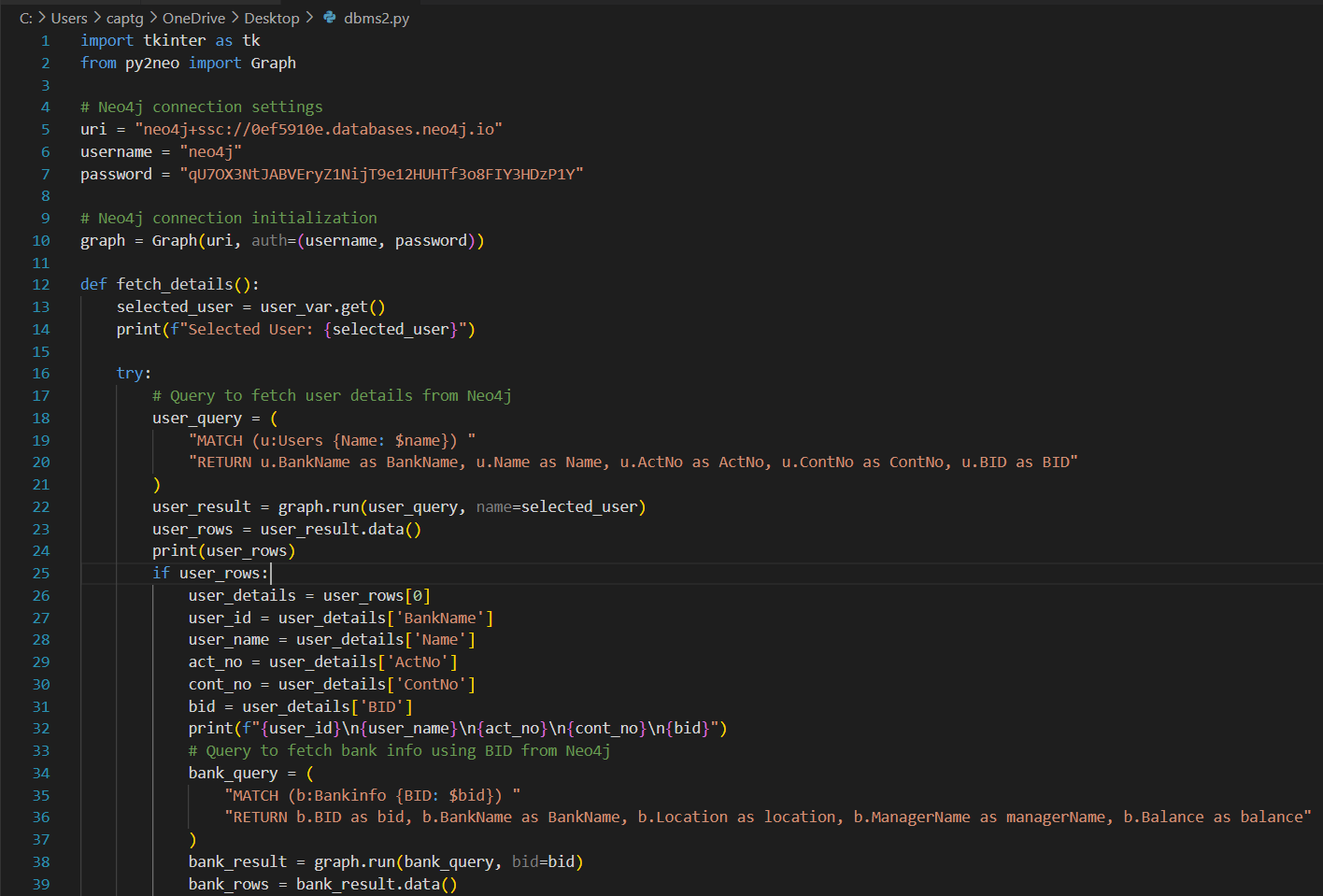
Description automatically generated

Just to show another example of the utility of this DBMS , let us explore how this graph can further be worked upon directly through the site using the “Cypher” Language. For this we have taken the example of data between Orders and Products , let us see the different operations that can be worked upon this DB using Cypher Queries directly in the system :

This here is an example of how the user could find all orders of the Product : Chai , in a Graphical format , this could also be done in a tabular format for a more statistical view :

1. **INCORPORATION IN PYTHON CODE USING API IMPLEMENTATION**

Keeping in Mind the Aim of this project , after understanding the Basic aspects of how Neo4j works and how we have created our DB for reference here in this project. Let us now import the very same database into Python code , using the API facility provided by Neo4j and also create a simple GUI using tkinter which for this project , aims to ask the User to select the Name of the person whose Bank Information they wish to retrieve via the provision of a drop down box. This is parsed into the DB through the Python code by implementing a similar cypher query as seen above within the python code itself which executes the same in the Neo4j DB itself to retrieve the required result as per our requirement and then further finally project it into the GUI defined.

Here is how our code looks for the same :

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

Thus from this code we can understand the following steps were undertaken :

* Tkinter was imported for GUI generation and py2neo was imported to work upon the Neo4j DB using our py code
* The connection settings i.e. the URI , key and passwords were all defined as per the Instance of which our DB was generated
* The logic of getting the Users selection and then defining the MATCH Cypher query was defined to fetch the values corresponding to that User chosen
* Once fetched , it understand that through the “BID” attribute present within the Users table (Here Node) , which is the Foreign Key in this case and PK in the BankInfo node , it creates a link between the two nodes through this relationship and then for the selected User , retrieves the BankInformation provided within that table (node)
* The final part of the logic is the GUI generation with its basic drop down box definitions and further the looks and general layout for how the boxes generated on output are to be seen

A screenshot of a computer

Description automatically generatedHere is how the output would then be generated upon running this code :

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

* **CONCLUSION**

Thus our Aim of creating a Database using the Graphical DBMS – Neo4j and incorporating it upon creation in python code to then work on it and retrieve user queries while presenting it with the help of a simple GUI was successfully achieved as seen and discussed above.

**xXx**