Simple Banking Graphical User DBMS With GUI-Interface Using API

[Gurshaan Singh (21BLC1424)[1],Shoukat Ali (21BLC1497)[2] and Dr.Om Kumar C U[3]], Branch–B.Tech Electronics & Computer Engineering, Sense

VIT® Vellore Institute of Technology (Deemed to be University under section 3 of UGC Act, 1956) CHENNAI

Objectives

Main objectives considered in this work of banking DBMS involves:

- Design a Database using own Banking datasets within the Neo4j Graphical DBMS platform with valid nodes, keys and relationships. Then test its outputs using Cypher language queries.
- Create python code with simple GUI interface to select which users information is desired and using API fetch those details from the Neo4j DB created.

Materials & Methods

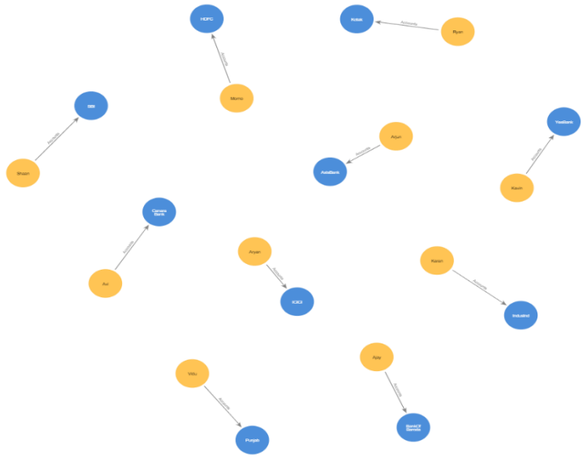
Materials required includes Python 3.10.1, VS Code , Neo4j account libraries including tkinter and py2neo and 3 benchmark datasets for details of User , Bank and Relationship between them.
Proposed method starts with connecting code to Neo4j by invoking API key and account details, then by inputting Cypher language queries (native to Neo4j) for fetching details as per user choice of name using Relationship data to take foreign key connections between the two major datasets being Users Information and Banks Information.
Basic GUI Interface was created which takes Users Input of User Name in drop down box format based on which it returns Users details as well as the Banks details by matching foreign key of Users dataset BankID with BankInfo dataset and then also returns that matched ID attributes record/tuple as Bank details as desired.

Introduction

Regular tabular databases are efficient to work in however tough to completely grasp and visualize information from. In this work, the focus was on solving this issue by incorporating a Graphical DBMS platform Neo4j which allows Node and Edge Graph type representation of data with edges being the Relationships between the datasets. Further by application of its native language Cypher we may work upon this Graph to get desired information of the User and the Bank associated with. A python code is proposed which utilizes API of Neo4j and incorporates a GUI returning the outputs required fetched from the DB

Results 1

Figure 1 shows the Neo4j DB Graph with relationships between the Users and their Banks :
Figure 1 -



RESULTS 2

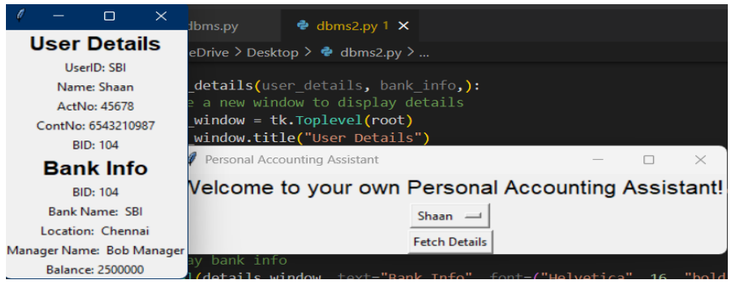
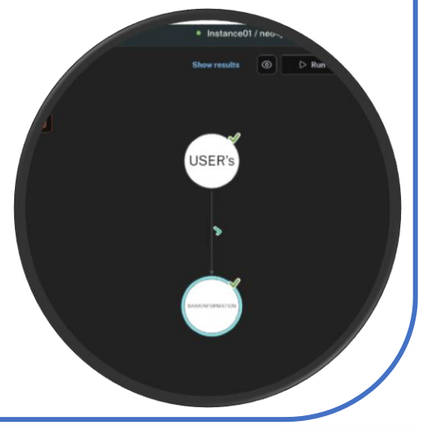


figure 2: GUI output of User and Bank details



From Figure 2. we can see how upon users input of User Name via the drop down menu , the code took the input which was chosen by the user and utilized the code to fetch the attributes of that matched record from the Users dataset and then further matched the BankID foreign Key to the BankInfo dataset by the relationship defined in Neo4j between them of Primary Keys and Foreign Keys , thus allowing the return of Users and Banks Details as desired.

Conclusion

As desired , our project proved to be a capable and efficient method of incorporating a Graphical Database Management with an easy to visually understand and work on Graph. Then , with the help of simple python coding and account details in whose instance we had created this Graph , we invoked the API of Neo4j to apply the very same Cypher Language queries to obtain the Banking details of the User as per the users choice as desired allowing for a quick navigation either through code which can prove useful in larger datasets where the graph is populated with an immense amount of Nodes and Relationships or directly with the very same code in the platform itself , with the advantage here being that of a simple GUI which applies that code to fetch details from the DB via API

References

[1] Massimo Carro. (2013). NoSQL Databases. Springer Ebooks, 1546–1546. doi:10.1007/978-1-4419-9863-7_101051.
.[2] José Guia, Valéria Leite Soares, & Jorge Bernardino. (2017). Graph Databases: Neo4j Analysis, 351–356. doi:10.5220/0006356003510356

Future Research

Further studies can be done by scaling this algorithm and idea to higher levels of data be it real time or existing , where in cases that a graph may be very large in size and thus queries may be the only way to obtain smaller more efficient graphs There this algorithm can work wonders in also optimizing a GUI to show results and not just a graph from the database , either way being more advantageous than the existing simple tabular forms

Contact Information

1.Email – gurshaansingh.bhasin2021@vitstudent.ac.in, Phone – +91–8610737553
2.Email – mohamedshoukat.ali2021@vitstudent.ac.in, Phone - +91-9087701842
3.Email – omkumar.cu@vit.ac.in, Phone – +91–9952035395