CS432/532: Final Project Report

**Project Title: Fraudulent Activity Detection Using Graph Database**

**Team Members: Meghana Hubli and Sai Tripura Manne**

# Problem

Finance sectors like banking system and credit card systems, face common identity theft and result in financial loss. Fraudulent activities are carried using these fake identity created using actual mailing addresses, phone number or private confidential identities like Social security numbers. Sometimes, the victims are directly involved in such fraudulent activities instead of creating a fake identity. Credit card details are stolen sometimes at Point of Sale for any vendor sales, and the stolen details are used for bust out credit card sales. Usually these bust out sales are marked as Disputed sales, and tracing back to the identity thief is difficult since the original owner of the credit card has zero knowledge of the identity theft.

To address such fraudulent activities [1], we propose to use a Graph Database approach to plot each identity and the characteristics of the identities as Nodes and establish a relationship between these nodes if any exists. By using this approach, we get a bird’s eye view of the interconnected financial data and try to trace back to origin of identity theft or fake identities.

# Software Design and Implementation

## Software Design and NoSQL-Databse and Tools Used

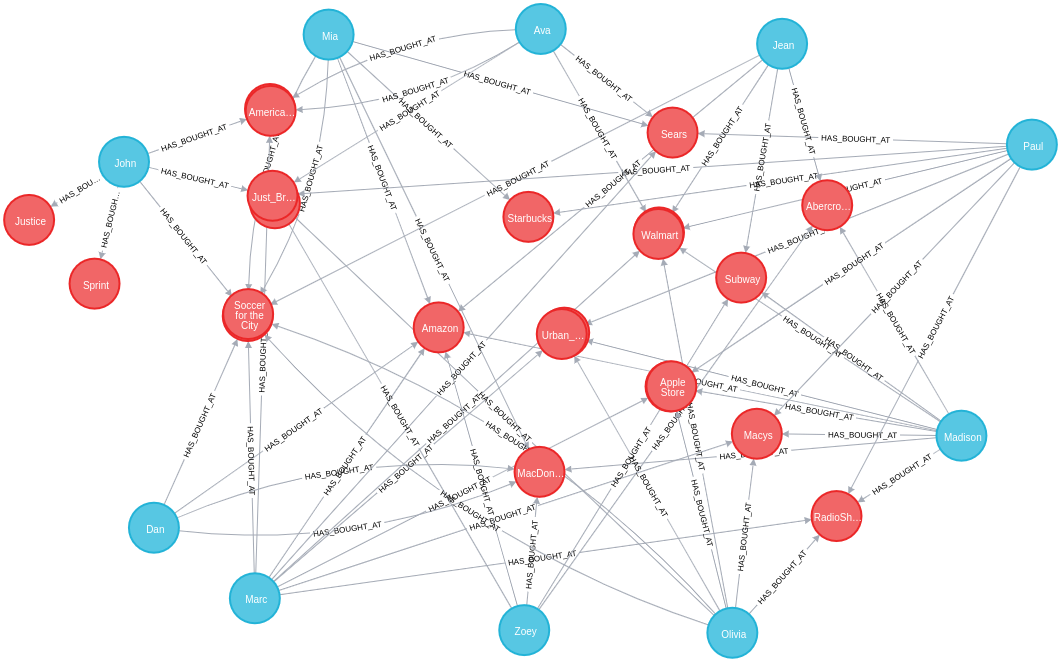
We used an open-source *NoSQL Graph Database called as Neo4j* [2] for our implementation. Neo4j uses it s own query language called as *Cypher Query*. We used *Python Programming Language* to interface with the graph database and automated the process of sending the queries to the database. Each query is stored in a file, and by using python read and write operation, we execute a query, retrieve the output and write the resulting to a csv file.

## Supported Queries

We have considered two types of datasets related to the finance sector. Banking account information datasets and Credit Card transactions datasets between a customer and a merchant. For each one of the datasets, the queries are designed to track fraudulent activities, evaluate the financial losses and look for any false positivies in the outcomes.

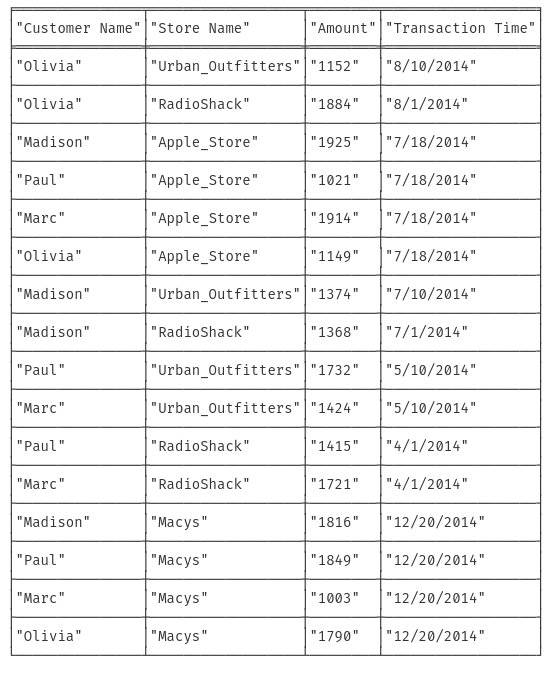
*Credit Card Fraud Detection:*

For the credit card fraud datasets, each transaction made between a customer and the merchant is recorded. Figure 1 shows the graph database made with Customer and Merchant as nodes, and the transactions as Relationship Edges.

  
Figure 1: Graph Database representation of Credit Card activity

Here, the blue notes represent the Customers, Red nodes represent the merchants, and all the nodes are interconnected for the transactions made among these nodes. Each relationship consists of information like transaction date, amount and address.

1. Finding all the “Disputed” transactions made with a merchant by a customer. The following table represents the information extracted by running the Query,



*Query 1:*

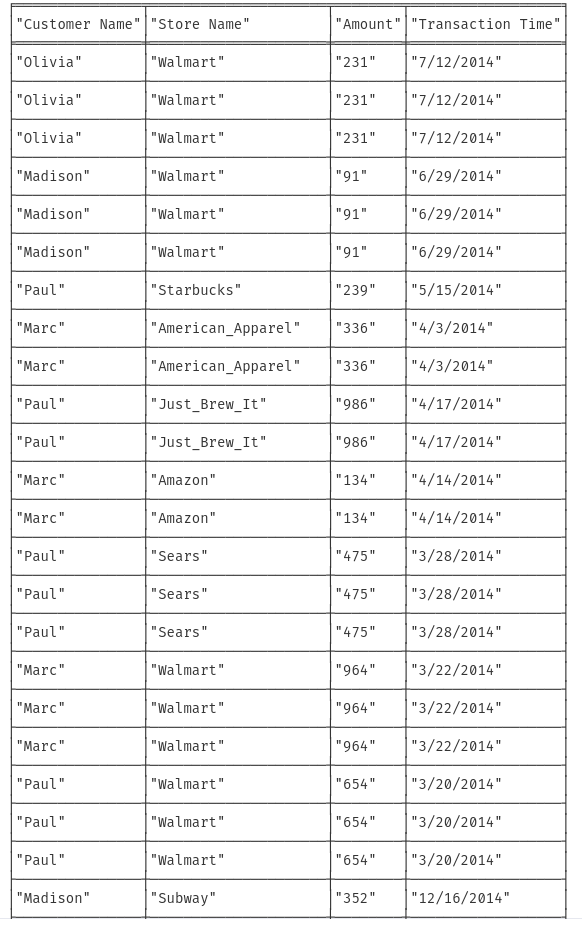
MATCH (victim:Person)-[r:HAS\_BOUGHT\_AT]->(merchant)

WHERE r.status = "Disputed"

RETURN victim.name AS `Customer Name`, merchant.name AS `Store Name`, r.amount AS Amount, r.time AS `Transaction Time`

ORDER BY `Transaction Time` DESC

2. For each one of the “Disputed” transaction, the previously made “UnDisputed” transaction is extracted. Here, the transaction time of the “UnDisputed” transaction is earlier than the “Disputed” transaction. This is to identify to identify the common origin for all the Disputed transactions. The table represents the output from the query,



*Query 2:*

MATCH (victim:Person)-[r:HAS\_BOUGHT\_AT]->(merchant)

WHERE r.status = "Disputed"

MATCH (victim)-[t:HAS\_BOUGHT\_AT]->(othermerchants)

WHERE t.status = "Undisputed" AND t.time < r.time

WITH victim, othermerchants, t ORDER BY t.time DESC

RETURN victim.name AS `Customer Name`, othermerchants.name AS `Store Name`, t.amount AS Amount, t.time AS `Transaction Time`

ORDER BY `Transaction Time` DESC

3. From the combination list of all the “Disputed” and “UnDisputed” transactions, each merchant where the last Undisputed transaction was carried out was counted to see which is a common source for most of the credit card frauds. This can identify theft at Point of Sale, and actions can be taken accordingly.

MATCH (victim:Person)-[r:HAS\_BOUGHT\_AT]->(merchant)

WHERE r.status = "Disputed"

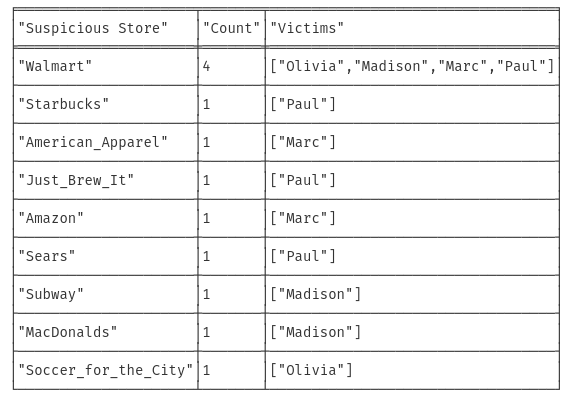
MATCH (victim)-[t:HAS\_BOUGHT\_AT]->(othermerchants)

WHERE t.status = "Undisputed" AND t.time < r.time

WITH victim, othermerchants, t ORDER BY t.time DESC

RETURN DISTINCT othermerchants.name AS `Suspicious Store`, count(DISTINCT t) AS Count, collect(DISTINCT victim.name) AS Victims ORDER BY Count DESC

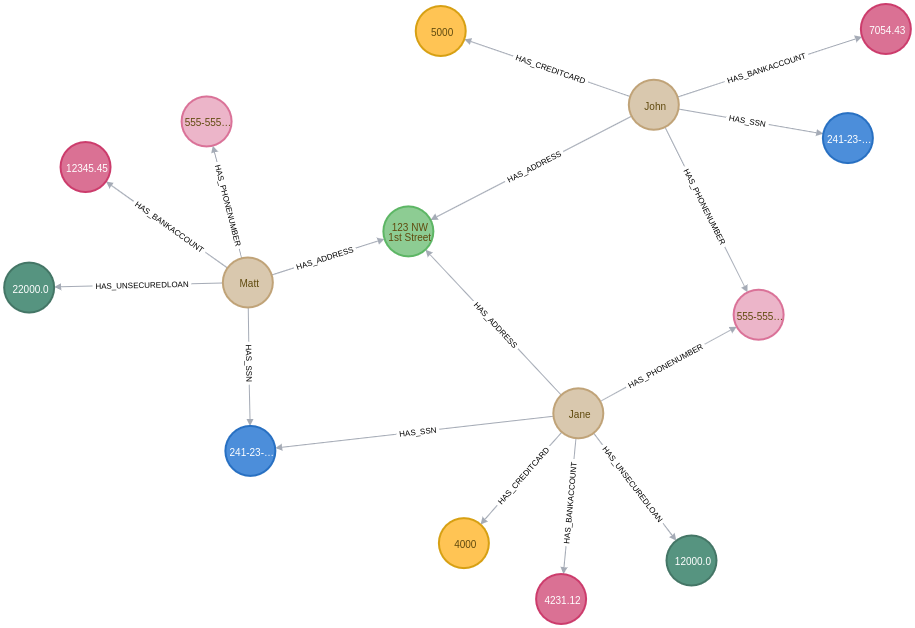
From the table below, we can see that “Walmart” is a common place where the fradulent activity starts from, using the count.



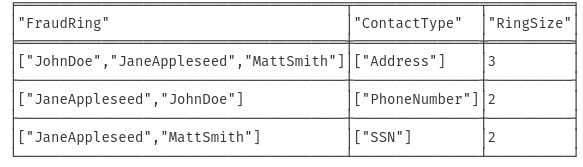
*Query 3:*

*Bank Fraudulent Ring Detection:*

Another case of identity theft is by forging fake identities based on actual information from a group of fradulent users. For example, a shared address, phone number or SSN. The following queries should identify any fraudulent rings which are interconnected. The graph database view of the dataset is,

  
Figure 2: Interconnected Bank Account Details

1. Looking for any formation of closed loop fradulent rings which are sharing private information like address, phone number and SSN.



*Query 1*

MATCH (accountHolder:AccountHolder)-[]->(contactInformation)

WITH contactInformation,count(accountHolder) AS RingSize

MATCH (contactInformation)<-[]-(accountHolder)

WITH collect(accountHolder.UniqueId) AS AccountHolders,

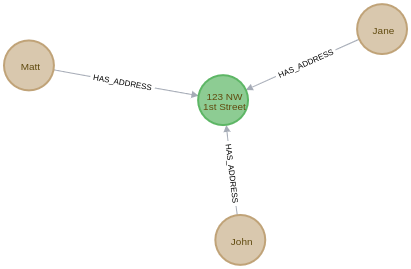
contactInformation, RingSize

WHERE RingSize > 1

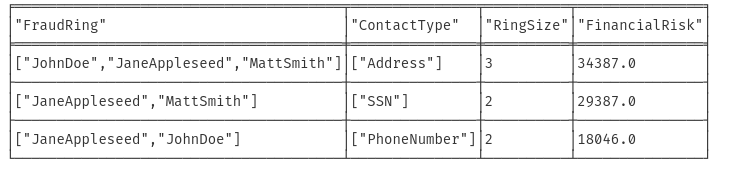
RETURN AccountHolders AS FraudRing,labels(contactInformation) AS ContactType, RingSize

ORDER BY RingSize DESC

2. Evaluate if the above mentioned query is a false positive, given that address can be shared by multiple users in a household, or multiple phone numbers can be shared by guardian supporting a bank account.

  
Figure 3: Account holders sharing same address, possible false positive

3. Computing the financial losses suffered due to the fraudulent rings by considering personal loans, bank account statements and credit card payments due. This can be used to avoid the losses which can outcome from these rings, and closely monitor such groups.



*Query 3*

MATCH (accountHolder:AccountHolder)-[]->(contactInformation)

WITH contactInformation, count(accountHolder) AS RingSize

MATCH (contactInformation)<-[]-(accountHolder),

(accountHolder)-[r:HAS\_CREDITCARD|HAS\_UNSECUREDLOAN]->(unsecuredAccount)

WITH collect(DISTINCT accountHolder.UniqueId) AS AccountHolders,

contactInformation, RingSize,SUM(CASE type(r)

WHEN 'HAS\_CREDITCARD' THEN unsecuredAccount.Limit

WHEN 'HAS\_UNSECUREDLOAN' THEN unsecuredAccount.Balance

ELSE 0

END) as FinancialRisk

WHERE RingSize > 1

RETURN AccountHolders AS FraudRing,labels(contactInformation) AS ContactType, RingSize,round(FinancialRisk) as FinancialRisk

ORDER BY FinancialRisk DESC

The query looks for all the finance related to the fraud ring, and evaluates the sum of borrowed credit.

# Project outcomes

* Provide the link (URL) to your source code on GitHub here
* Provide the link (URL) to your YouTube video here
* Provide the link to your presentation slides here

##### References

1. Sadowski, Gorka, and Philip Rathle. "Fraud detection: Discovering connections with graph databases." *White Paper-Neo Technology-Graphs are Everywhere* 13 (2014).
2. Neo4j Graph Database, <https://neo4j.com/>