An overview of the uses Of Graph Databases in Financial Services – built using Neo4j

Team Members:

1. Naman Jain PES1UG19CS288
2. Hardik Modi PES1UG19CS273
3. Tushar YS PES1UG19CS545

Introduction

Traditionally, banks, credit card companies, insurance firms and other financial services companies have relied on relational databases (RDBMS) to process and manage the massive amount of data produced on a daily basis. While relational databases are useful for handling basic data collection, storage and analytical processing needs, this technology fails when it comes to delivering the speed and responsiveness needed for processing complex queries. Graph databases make it easier to discover insights into relationships, and taking into advantage, this fact, allows for new possibilities in the financial services world.

Accounts and transactions could be modelled as a graph to discover instances of money laundering. By graphing the relationships between these entities, teams can track how and where funds are moving through automated Cypher queries that map to traditional money laundering behaviours.

Standard anti-fraud technologies use discrete data, which is useful for catching individual criminals acting alone, but they fall short when it comes to detecting fraud rings. These methods also produce a lot of false positives which waste time and resources and negatively impact the customer experience. Graph databases can be used to identify fraudulent activity from fraud rings and uncover synthetic or stolen identities in real time with high accuracy.

Proposed Methodology

1. Create and populate the database

```
CREATE (AccountHolder1: AccountHolder {Name:"Tushar",Birthdate:date("2001-10-18") ,UniqueId:10001 })
CREATE (AccountHolder2: AccountHolder {Name:"Hardik",Birthdate:date("2001-04-02") ,UniqueId:10002 })
CREATE (AccountHolder3: AccountHolder {Name:"Naman",Birthdate:date("2001-05-10") ,UniqueId:10003 })
CREATE (AccountHolder4: AccountHolder {Name:"Karthik",Birthdate:date("2001-11-16") ,UniqueId:10004 })


CREATE (pan1:PAN {PanNo: "ABCTY1234D" })
CREATE (pan2:PAN {PanNo: "CXBWY1469D" })
CREATE (pan3:PAN {PanNo: "AXCTY1214Y" })


MATCH(acc1: AccountHolder{UniqueId: 10001}) MATCH(p1: PAN{PanNo: "ABCTY1234D"}) CREATE (acc1)-[:HAS_PAN]->(p1)
MATCH(acc2: AccountHolder{UniqueId: 10002}) MATCH(p2: PAN{PanNo: "CXBWY1469D"}) CREATE (acc2)-[:HAS_PAN]->(p2)
MATCH(acc3: AccountHolder{UniqueId: 10003}) MATCH(p1: PAN{PanNo: "ABCTY1234D"}) CREATE (acc3)-[:HAS_PAN]->(p1)
MATCH(acc4: AccountHolder{UniqueId: 10004}) MATCH(p1: PAN{PanNo: "AXCTY1214Y"}) CREATE (acc4)-[:HAS_PAN]->(p1)


CREATE (addr: Address{city: "Bengaluru", State: "Karnataka"})
CREATE (addr: Address{city: "Mysore", State: "Karnataka"})

MATCH(acc1: AccountHolder{UniqueId: 10001}) MATCH(a1: Address{city: "Bengaluru", State: "Karnataka"}) CREATE (acc1)-[:HAS_ADDR]->(a1)
MATCH(acc1: AccountHolder{UniqueId: 10003}) MATCH(a1: Address{city: "Mysore", State: "Karnataka"}) CREATE (acc1)-[:HAS_ADDR]->(a1)
MATCH(acc1: AccountHolder{UniqueId: 10002}) MATCH(a1: Address{city: "Bengaluru", State: "Karnataka"}) CREATE (acc1)-[:HAS_ADDR]->(a1)
MATCH(acc1: AccountHolder{UniqueId: 10004}) MATCH(a1: Address{city: "Mysore", State: "Karnataka"}) CREATE (acc1)-[:HAS_ADDR]->(a1)
```

```
CREATE (card1: CARD{CardNo: 374245455400126, ExpDate: "2023-05", CVV: "098", Limit: 10000})
CREATE (card2: CARD{CardNo: 374245455400125, ExpDate: "2022-04", CVV: "098", Limit: 20000})
CREATE (card3: CARD{CardNo: 384245455400123, ExpDate: "2025-01", CVV: "048", Limit: 12000})
CREATE (card4: CARD{CardNo: 5684245455401133, ExpDate: "2024-11", CVV: "032", Limit: 16000})

MATCH(acc1: AccountHolder{UniqueId: 10001}) MATCH(c1: CARD{CardNo: 374245455400126}) CREATE (acc1)-[:HAS_CARD]->(c1)
MATCH(acc2: AccountHolder{UniqueId: 10002}) MATCH(c2: CARD{CardNo: 374245455400125}) CREATE (acc2)-[:HAS_CARD]->(c2)
MATCH(acc3: AccountHolder{UniqueId: 10003}) MATCH(c3: CARD{CardNo: 384245455400123}) CREATE (acc3)-[:HAS_CARD]->(c3)
MATCH(acc4: AccountHolder{UniqueId: 10004}) MATCH(c3: CARD{CardNo: 5684245455401133}) CREATE (acc4)-[:HAS_CARD]->(c3)

CREATE (bankAcc1: BankAccount{AccNo: 50120123456789, Balance: 29000})
CREATE (bankAcc2: BankAccount{AccNo: 50120123456129, Balance: 80000})
CREATE (bankAcc3: BankAccount{AccNo: 51120123456123, Balance: 40000})

MATCH(acc1: AccountHolder{UniqueId: 10002}) MATCH(a1: BankAccount{AccNo: 50120123456129}) CREATE (acc1)-[:HAS_ACCOUNT]->(a1)
MATCH(acc1: AccountHolder{UniqueId: 10001}) MATCH(a1: BankAccount{AccNo: 50120123456789}) CREATE (acc1)-[:HAS_ACCOUNT]->(a1)
MATCH(acc1: AccountHolder{UniqueId: 10003}) MATCH(a1: BankAccount{AccNo: 51120123456123}) CREATE (acc1)-[:HAS_ACCOUNT]->(a1)
MATCH(acc1: AccountHolder{UniqueId: 10004}) MATCH(a1: BankAccount{AccNo: 50120123456129}) CREATE (acc1)-[:HAS_ACCOUNT]->(a1)

CREATE (transfer: TRANSACTION{Amount: 5000, Type: "UPI", Time: datetime("2021-11-18T16:46:50"), Tid: 12345})
CREATE (transfer: TRANSACTION{Amount: 3000, Type: "Net Banking", Time: datetime("2021-11-19T15:06:10"), Tid: 12346})
CREATE (transfer: TRANSACTION{Amount: 2500, Type: "UPI", Time: datetime("2021-11-20T17:16:10"), Tid: 12376})
CREATE (transfer: TRANSACTION{Amount: 3500, Type: "UPI", Time: datetime("2021-11-10T11:13:04"), Tid: 12196})
```

```
MATCH(a1: BankAccount{AccNo: 50120123456789}) MATCH(t1: TRANSACTION{Tid: 12345}) CREATE (a1)-[:DEBIT]->(t1)
MATCH(a2: BankAccount{AccNo: 50120123456129}) MATCH(t1: TRANSACTION{Tid: 12345}) CREATE (t1)-[:CREDIT]->(a2)

MATCH(a1: BankAccount{AccNo: 51120123456123}) MATCH(t1: TRANSACTION{Tid: 12346}) CREATE (a1)-[:DEBIT]->(t1)
MATCH(a2: BankAccount{AccNo: 50120123456789}) MATCH(t1: TRANSACTION{Tid: 12346}) CREATE (t1)-[:CREDIT]->(a2)

MATCH(a1: BankAccount{AccNo: 51120123456123}) MATCH(t1: TRANSACTION{Tid: 12376}) CREATE (a1)-[:DEBIT]->(t1)
MATCH(a2: BankAccount{AccNo: 50120123456789}) MATCH(t1: TRANSACTION{Tid: 12376}) CREATE (t1)-[:CREDIT]->(a2)

MATCH(a1: BankAccount{AccNo: 50120123456789}) MATCH(t1: TRANSACTION{Tid: 12196}) CREATE (a1)-[:DEBIT]->(t1)
MATCH(a2: BankAccount{AccNo: 51120123456123}) MATCH(t1: TRANSACTION{Tid: 12196}) CREATE (t1)-[:CREDIT]->(a2)

CREATE (purchase1: PURCHASE{Amount: 3999, Time: datetime("2021-11-10T18:21:36"), Pid: 23456, Merchant: "Amazon"})
CREATE (purchase2: PURCHASE{Amount: 10999, Time: datetime("2021-11-12T13:29:26"), Pid: 23356, Merchant: "Amazon"})
CREATE (purchase3: PURCHASE{Amount: 15000, Time: datetime("2021-11-13T13:29:26"), Pid: 21357, Merchant: "Flipkart"})
CREATE (purchase3: PURCHASE{Amount: 4000, Time: datetime("2021-10-12T12:19:46"), Pid: 21557, Merchant: "Flipkart"})
CREATE (purchase4: PURCHASE{Amount: 4699, Time: datetime("2021-11-14T17:29:26"), Pid: 22157, Merchant: "Flipkart"})
CREATE (purchase4: PURCHASE{Amount: 2999, Time: datetime("2021-10-12T17:29:26"), Pid: 24571, Merchant: "D-Mart"})

MATCH(c1: CARD{CardNo: 374245455400125}) MATCH(p1: PURCHASE{Pid: 23456}) CREATE (p1)-[:WITH_CARD]->(c1)
MATCH(c1: CARD{CardNo: 384245455400123}) MATCH(p1: PURCHASE{Pid: 23356}) CREATE (p1)-[:WITH_CARD]->(c1)
MATCH(c1: CARD{CardNo: 374245455400126}) MATCH(p1: PURCHASE{Pid: 21357}) CREATE (p1)-[:WITH_CARD]->(c1)
MATCH(c1: CARD{CardNo: 384245455400123}) MATCH(p1: PURCHASE{Pid: 21557}) CREATE (p1)-[:WITH_CARD]->(c1)
MATCH(c1: CARD{CardNo: 374245455400126}) MATCH(p1: PURCHASE{Pid: 22157}) CREATE (p1)-[:WITH_CARD]->(c1)
MATCH(c1: CARD{CardNo: 5684245455401133}) MATCH(p1: PURCHASE{Pid: 24571}) CREATE (p1)-[:WITH_CARD]->(c1)

CREATE (ip1: IP{ip_addr: "192.256.68.11"})
CREATE (ip1: IP{ip_addr: "192.256.68.21"})
CREATE (ip1: IP{ip_addr: "192.256.48.23"})
```

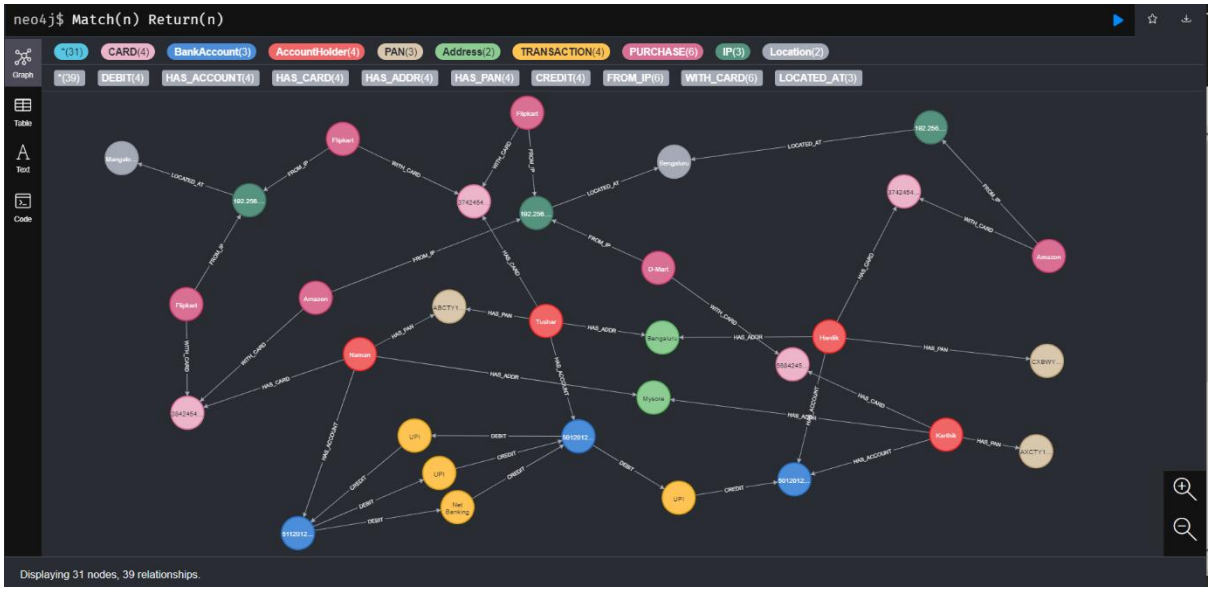
```
CREATE (ip1: IP{ip_addr: "192.256.68.11"})
CREATE (ip1: IP{ip_addr: "192.256.68.21"})
CREATE (ip1: IP{ip_addr: "192.256.48.23"})


MATCH(ip1: IP{ip_addr: "192.256.68.11"}) MATCH(p1: PURCHASE{Pid: 23456}) CREATE (p1)-[:FROM_IP]->(ip1)
MATCH(ip1: IP{ip_addr: "192.256.68.21"}) MATCH(p1: PURCHASE{Pid: 23356}) CREATE (p1)-[:FROM_IP]->(ip1)
MATCH(ip1: IP{ip_addr: "192.256.68.21"}) MATCH(p1: PURCHASE{Pid: 21357}) CREATE (p1)-[:FROM_IP]->(ip1)
MATCH(ip1: IP{ip_addr: "192.256.48.23"}) MATCH(p1: PURCHASE{Pid: 21557}) CREATE (p1)-[:FROM_IP]->(ip1)
MATCH(ip1: IP{ip_addr: "192.256.48.23"}) MATCH(p1: PURCHASE{Pid: 22157}) CREATE (p1)-[:FROM_IP]->(ip1)
MATCH(ip1: IP{ip_addr: "192.256.68.21"}) MATCH(p1: PURCHASE{Pid: 24571}) CREATE (p1)-[:FROM_IP]->(ip1)

CREATE (loc: Location{name:"Bengaluru"})
CREATE (loc: Location{name:"Mangalore"})

MATCH(ip1: IP{ip_addr: "192.256.68.11"}) MATCH(l1: Location{name:"Bengaluru" }) CREATE (ip1)-[:LOCATED_AT]->(l1)
MATCH(ip1: IP{ip_addr: "192.256.68.21"}) MATCH(l1: Location{name: "Bengaluru"}) CREATE (ip1)-[:LOCATED_AT]->(l1)
MATCH(ip1: IP{ip_addr: "192.256.48.23"}) MATCH(l1: Location{name: "Mangalore"}) CREATE (ip1)-[:LOCATED_AT]->(l1)
```

Result -



Results

1. Executing queries

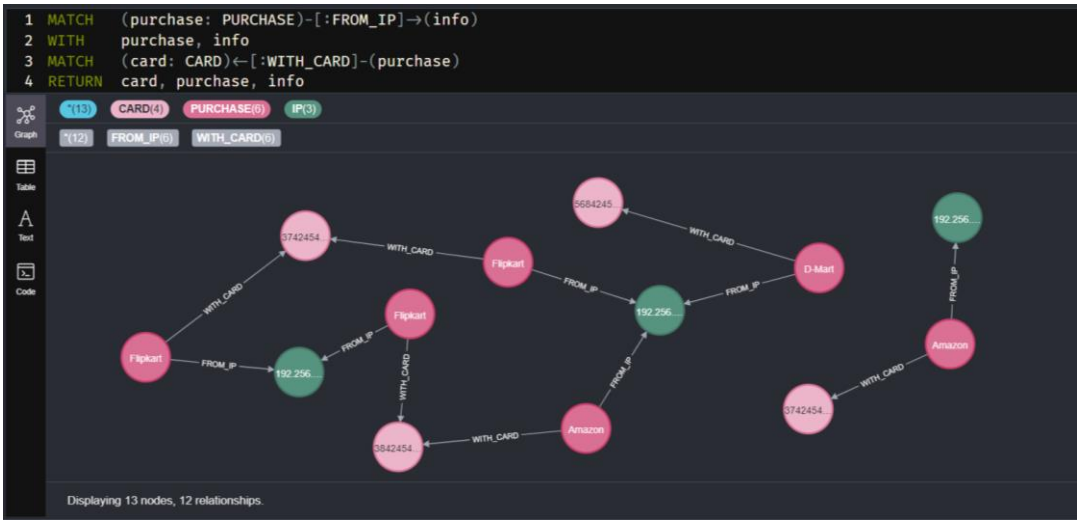
I. Finding potential fraud ring by extracting details of customers sharing a PAN number.

```
MATCH   (accountHolder:AccountHolder)-[:HAS_PAN]→(info)
WITH    info, count(accountHolder) AS RingSize
MATCH   (info)←[]-(accountHolder)
WITH    collect(accountHolder.UniqueId) AS AccountHolders,
        info, RingSize
WHERE   RingSize > 1
RETURN  AccountHolders AS FraudRing,
        labels(info) AS Identification,
        RingSize
ORDER BY RingSize DESC
```

	FraudRing	Identification	RingSize
1	[10003, 10001]	["PAN"]	2

II. Order the IP addresses by network traffic. Places where there is a high network traffic could indicate a potential fraudster, or it could also be a public network.

```
MATCH   (purchase: PURCHASE)-[:FROM_IP]→(info)
WITH    purchase, info
MATCH   (card: CARD)←[:WITH_CARD]-(purchase)
RETURN  card, purchase, info
```




```
MATCH (purchase: PURCHASE)-[i:FROM_IP]→(ip)
RETURN ip.ip_addr as IP_address, count(i) as Purchases
ORDER BY Purchases DESC
```

	IP_address	Purchases
1	"192.256.68.21"	3
2	"192.256.48.23"	2
3	"192.256.68.11"	1

III. Purchase trend in different locations. Useful for market segmentation.

```
MATCH (purchase: PURCHASE)-[i:FROM_IP]→(ip)
WITH purchase, i, ip
MATCH (ip)-[:LOCATED_AT]→(loc)
RETURN loc.name as Place, count(purchase) as Purchases
ORDER BY Purchases DESC
```

Table | Text | Code

	Place	Purchases
1	"Bengaluru"	4
2	"Mangalore"	2

IV. Retailer popularity at a specific location.

```
MATCH (purchase: PURCHASE)-[i:FROM_IP]→(ip)
WITH purchase, i, ip
MATCH (ip)-[:LOCATED_AT]→(loc)
WHERE loc.name = "Bengaluru"
RETURN purchase.Merchant as Merchant, count(purchase) as Purchases
ORDER BY Purchases DESC
```

Table | Text | Code

	Merchant	Purchases
1	"Amazon"	2
2	"D-Mart"	1
3	"Flipkart"	1

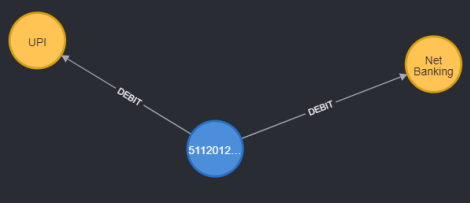
V. Account transaction history for a specific account.

```
MATCH (acc)←[:HAS_ACCOUNT]-(accholder)
WITH acc, accholder
MATCH (acc)-[:DEBIT]→(transaction)
WHERE acc.AccNo = 51120123456123
RETURN acc, transaction
```

Graph | Table | Text | Code

*(3) BankAccount(1) TRANSACTION(2)

*(2) DEBIT(2)



Conclusion

Today’s digitally-savvy customers expect companies to deliver personalised service that reflects an understanding of who they are and what they might be interested in the future. Financial service firms need a complete 360-degree view of customer data to provide powerful, real-time services. The power of connected data in the form of data relationships in graph databases enable these institutions to retain and improve their competitive edge.

With graph databases, financial services firms can detect fraud rings more accurately, connect various systems and data sources and oversee identity and access management, while at the same time boost customer engagement and drive sales revenue. Graphs are emerging as an incredibly powerful way of helping deliver just that, at scale, in real-time.