Credit Card Fraud Detection

Using Neo4j and PostgreSQL

Team: Aryan, Jude, Tarun and David

March 19, 2025

Introduction

UC San Diego

Credit Card Fraud - A Growing Threat

$33bn 2024

60% of consumers experience a fraud attempt

$443bn projected loss by 2032

 (Deloitte)

Financial fraud is increasing for banks, fintechs and credit unions

35% of banks and fintechs experienced 1000+ fraud attempts last year*

1 in 10 report 10,000+ fraud attempts

UC San Diego

 (Alloy Report 2024)

Data Sources

- Synthetically generated dataset with 1.8 million rows *
- 1000 customers
- 800 merchants
- 1st Jan 2019 - 31st Dec 2020

 * Sparkov generator

Data Sources

Transaction details

transaction timestamp,
transaction amount,
merchant,
UUID,
category

Cardholder Information

credit card number
transaction amt,
merchant,
UUID,
category

Location Data

cardholder geocodes
merchant geocodes

Label

is_fraud

Methodology

Methodology

Database Architecture

- PostgreSQL for transactional data storage and basic analytics
- Neo4j for graph-based fraud pattern detection
- Connection between databases (e.g., how data flows between systems)

UC San Diego

Methodology

Data Analysis

- Time-based features (transaction velocity, time between transactions)
- Location-based features (distance between transactions)
- Amount-based Outliers (unusual transaction amounts)
- Merchant category Outliers (unusual merchant categories for a customer)

Methodology

Model Development

- Graph-based pattern detection for fraud rings
- Anomaly detection for individual account fraud

Solution Implementation

Solution Implementation

System Architecture

Databases used

- Postgres - used for structured transaction data
- Neo4j - stores and queries relationships between transactions for fraud detection

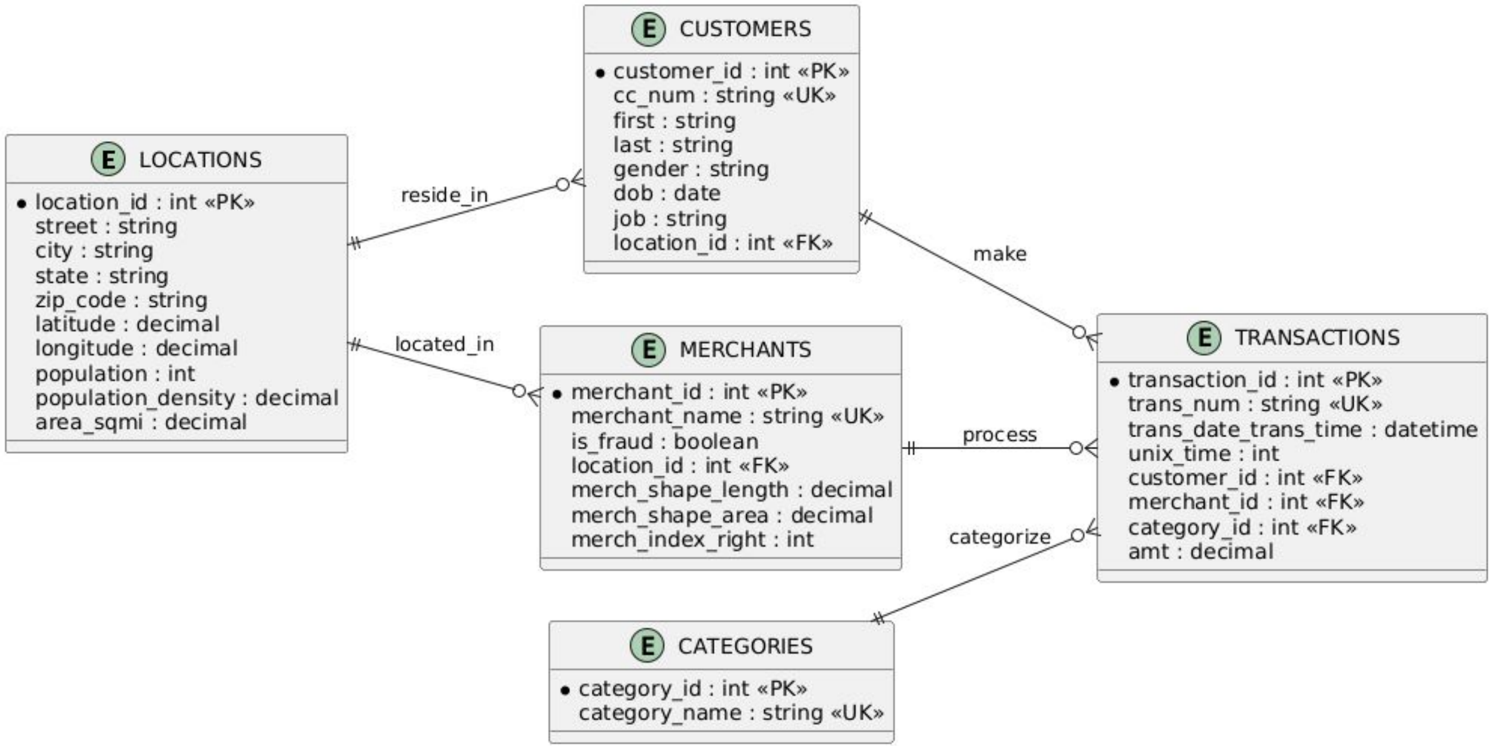
Solution Implementation

SQL Schema Design

Why not store everything in one table?

- creates redundancies
- difficult to update
- slower queries

Solution Implementation



UC San Diego

Demonstration

Use Case Scenarios

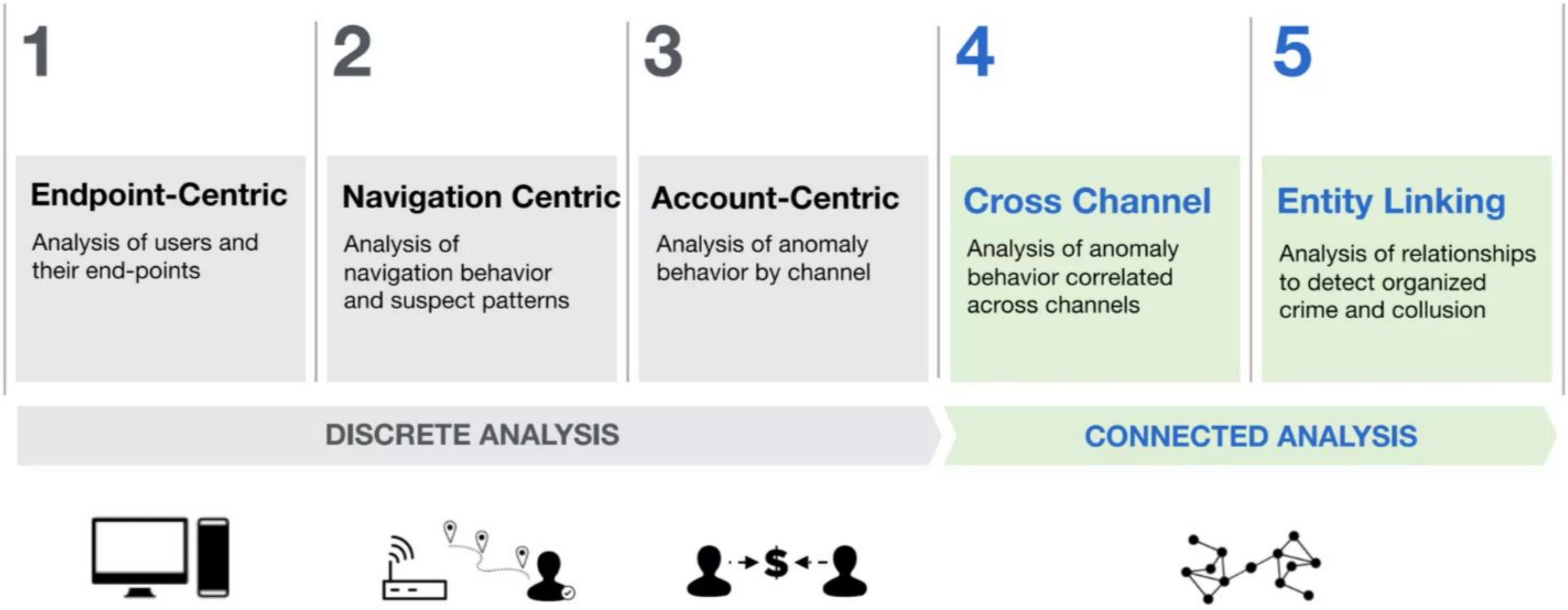
- Example 1: Identifying Transaction Anomalies
  - > 3 standard deviations from average spending
  - Late-night fraud
- Example 2: Geolocation Anomaly Detection
  - Compromised cards in unlikely locations
  - Improbable travel patterns
- Example 3: Flagging suspicious velocity patterns
  - Multiple transactions within 5 minutes
  - identifies cloning operations

29%

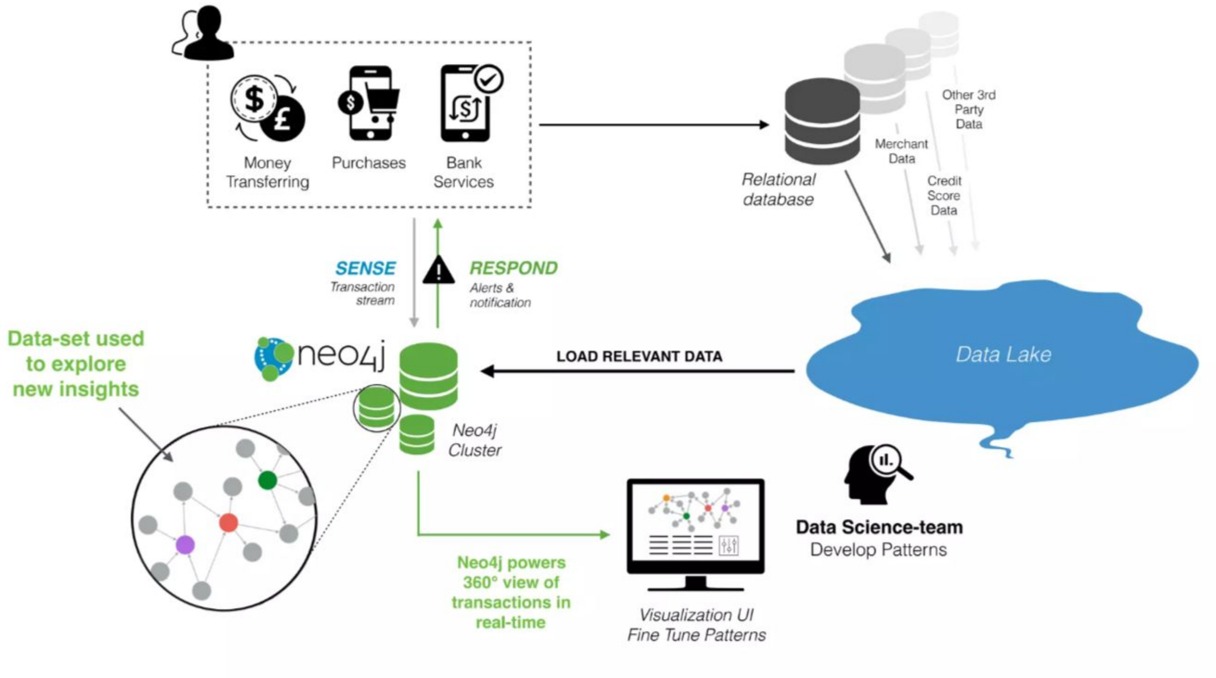
26%

54%

Postgres vs Neo4j

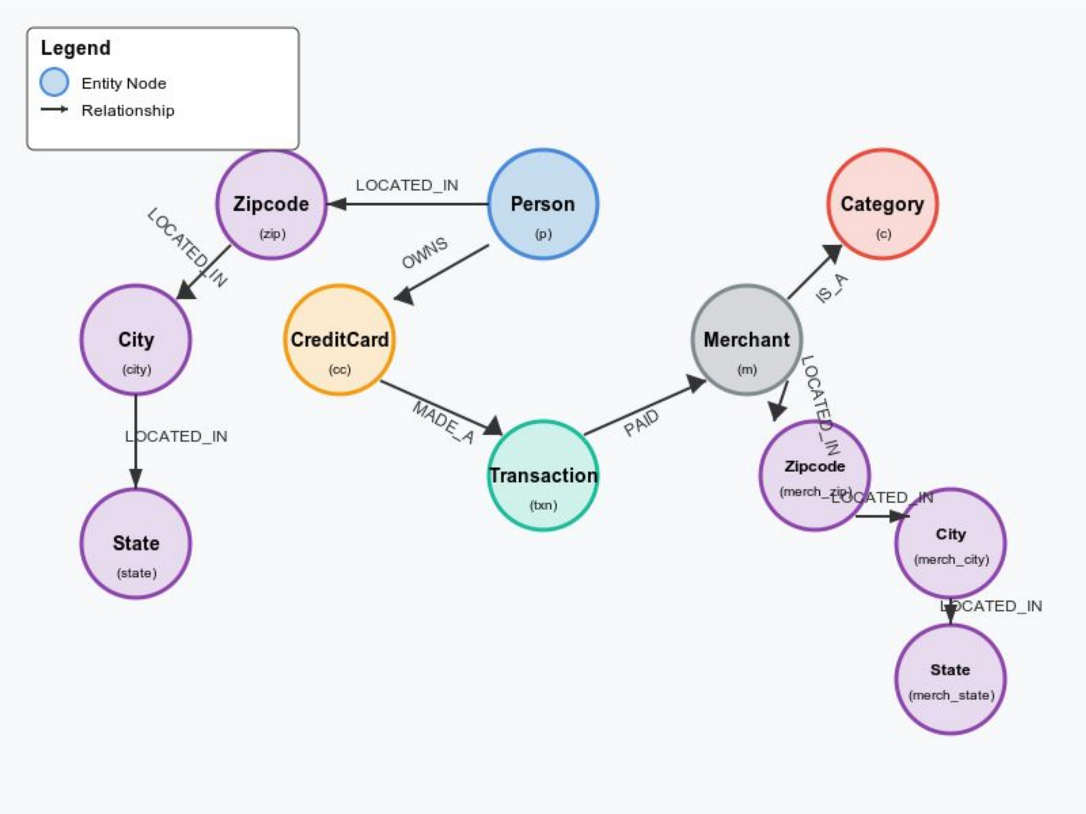


Neo4j



Money
Transferring
Purchases
Bank
Services
Relational
database
Merchant
Data
Other 3rd
Party
Data
Credit
Score
Data
SENSE
Transaction
stream
RESPOND
Alerts &
notification
Data-set used
to explore
new insights
neo4j
LOAD RELEVANT DATA
Data Lake
Neo4j
Cluster
Neo4j powers
360° view of
transactions in
real-time
Visualization UI
Fine Tune Patterns
Data Science-team
Develop Patterns

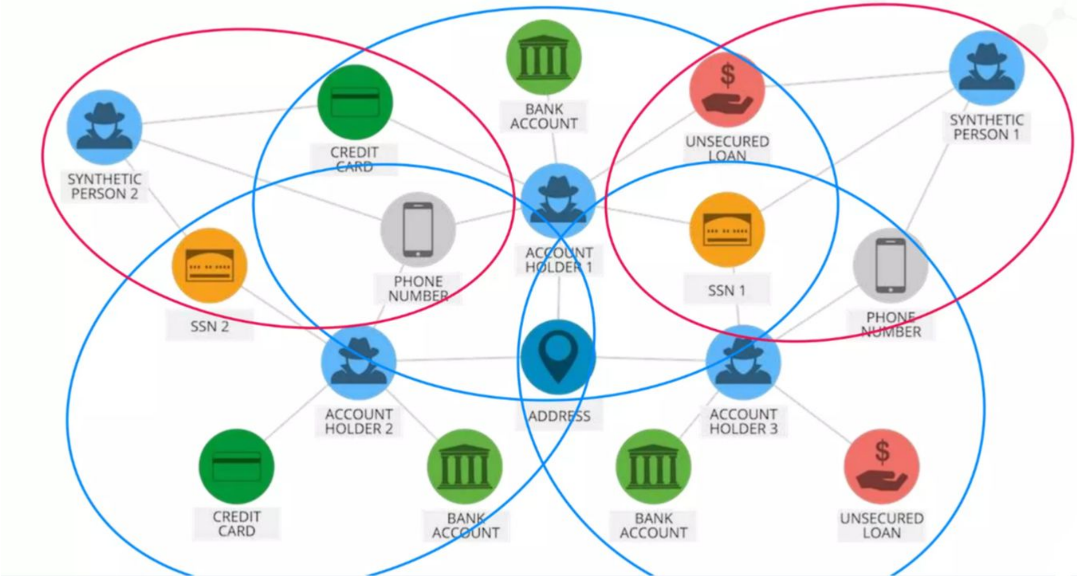
Graph Relationships

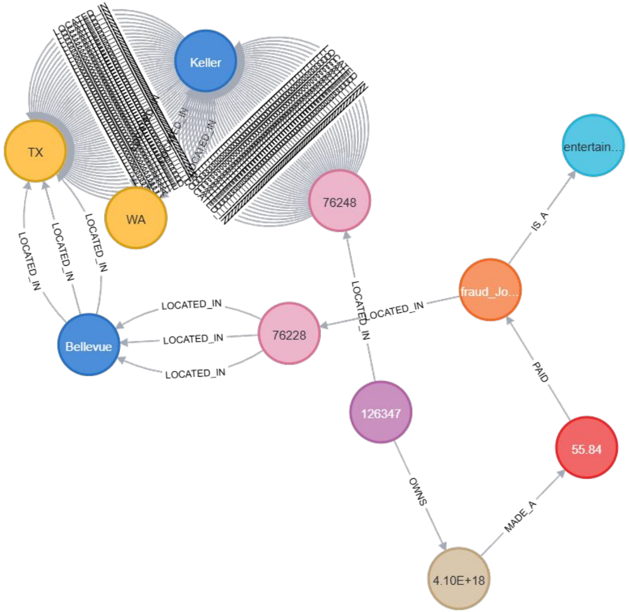


Attributes

```
MERGE (p:Person {first: row.first, last: row.last,
                 gender: row.gender, job: row.job})
MERGE (m:Merchant {merchantName: row.merchant})
MERGE (c:Category {categoryType: row.category})
MERGE (txn:Transaction {txnID: row.trans_num,
                        amount: toFloat(row.amt),
                        timestamp: row.trans_date_trans_time,
                        isFraud: row.txn_is_fraud})
MERGE (cc:CreditCard {ccNum: row.cc_num})
MERGE (personLoc:Geocode {latitude: toFloat(row.lat),
                          longitude: toFloat(row.long)})
MERGE (zip:Zipcode {code: row.zip})
MERGE (city:City {name: row.city, pop: toInteger(row.city_pop)})
MERGE (state:State {name: row.state})
MERGE (merchantLoc:Geocode {latitude: toFloat(row.merch_lat),
                            longitude: toFloat(row.merch_long)})
MERGE (merch_zip:Zipcode {code: row.merch_zip_code})
MERGE (merch_city:City {name: row.merch_po_name})
MERGE (merch_state:State {name: row.merch_state})
```

Graphical Representation of Fraudulent Connections





Initial Exploration

- Transaction Paths: Person → Credit Card → Transaction → Merchant → Category
- Merchant Fraud Networks
- Geographic Fraud Clusters
- Credit Card Multi-Hop Fraud Risk
- Circular Fraud Patterns
- Common Fraud Path Analysis

Advanced Exploration

Shortest Path Between Fraudulent Transactions

Connect seemingly unrelated fraud events:

- Detect hidden intermediaries.
- Reveal coordinated attacks.

```
MATCH (t1:Transaction), (t2:Transaction)
WHERE t1.isFraud = '1' AND t2.isFraud = '1' AND t1.txnID < t2.txnID
WITH t1, t2
MATCH path = shortestPath((t1)-[*..6]-(t2))
WHERE LENGTH(path) > 1
```

Fraud Communities Detection

Identify communities of connected entities:

- Common neighbors indicate fraud rings.
- Helps identify networked activity.

```
MATCH (t:Transaction)
WHERE t.isFraud = '1'
MATCH (t)-[*1..2]-(entity)
WITH COLLECT(DISTINCT entity) AS fraudConnectedEntities
UNWIND fraudConnectedEntities AS e1
UNWIND fraudConnectedEntities AS e2
WHERE ID(e1) < ID(e2)
MATCH (e1)-[*1..3]-(commonEntity)-[*1..3]-(e2)
```

Centrality-Based Fraud Risk

Identify key players in the fraud network:

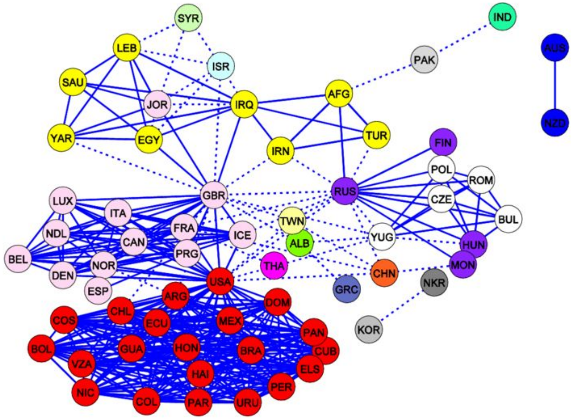
- Centrality measures highlight influential merchants and persons.
- Predict high-risk nodes.

```
MATCH (t:Transaction)
WHERE t.isFraud = '1'
MATCH (t)-[:PAID]-(m:Merchant)
WITH COLLECT(DISTINCT m) AS fraudMerchants
UNWIND fraudMerchants AS fraudMerchant
MATCH (fraudMerchant)<-[:PAID]-(t:Transaction)<-[:MADE_A]-(cc:CreditCard)
```

UC San Diego

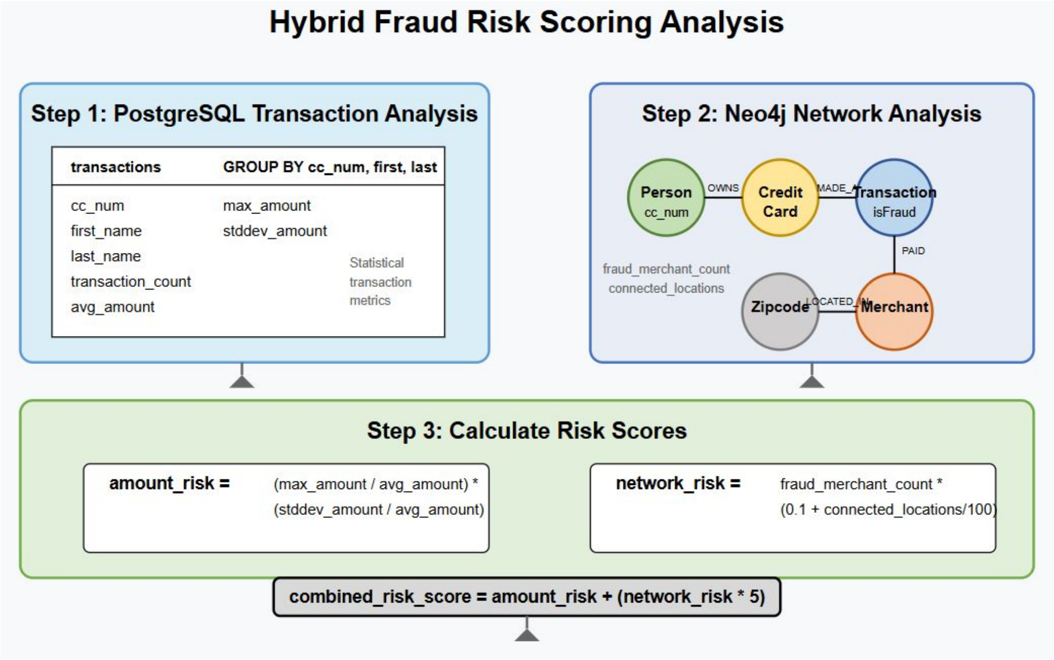
Graph Analytics

- PageRank for Key Fraud Entities
- Louvain Community Detection
- Triangle Counting for Fraud Network Analysis



PostgreSQL + Neo4j Combined Analysis

Creating fraud risk scores using transaction and relationship metrics



Lessons Learned & Future Work

- Challenges
- Key Insights
- Future Extensions

UC San Diego

Lessons Learned & Future Work

Challenges

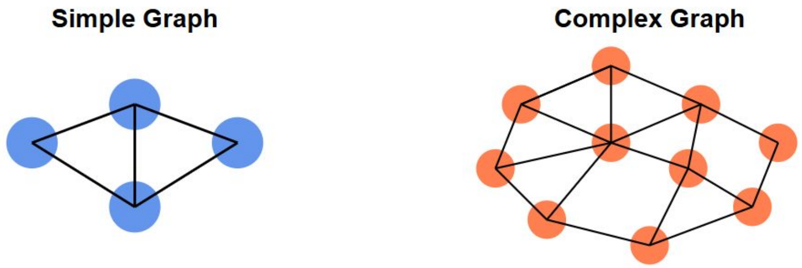
- Data quality issues
  - populating data and zip to city translation
- Data ingestion
  - dealing with memory/time constraints
  - graph structure
- Querying
  - optimization

UC San Diego

Lessons Learned & Future Work

Key Insights

- Tradeoffs between simpler and complex graph structure, even for same data
- Multiplicity can be dangerous
  - requires consideration when querying (counting distinct paths)
  - or alternative (reduced) graph complexity



Criteria	Simple Graph	Complex Graph
Creation Speed	Fast ✓	Slow ✗
Query Performance	Fast ✓	Potentially Slower ✗
Relationship Coverage	Minimal ✗	Complete ✓
Maintenance Effort	Easy ✓	Challenging ✗
Data Insights	Basic ✗	Rich ✓
Adaptability	Limited ✗	Highly Flexible ✓

UC San Diego

Multiplicative Effect

Original Data

We have **2** actual transactions

Transaction ID	Amount	Merchant
TX001	$120.29	Bahringer-Streich
TX002	$49.52	Smitham-Boehm

Merchant Locations

Bahringer-Streich

- Zipcode: **32321**, City: **Bristol**
- Zipcode: **32321**, City: **Grand Ridge**
- Zipcode: **32440**, City: **Bristol**

This merchant has **3** location combinations

Smitham-Boehm

- Zipcode: **80137**, City: **Watkins**
- Zipcode: **80137**, City: **Littleton**

This merchant has **2** location combinations

Multiplicative Effect Demonstration

When we query paths through the graph, each transaction creates multiple paths:

TX001 → Bahringer-Streich → **3 paths**

Path 1: TX001 → Bahringer-Streich → Zipcode:32321 → City:Bristol
Path 2: TX001 → Bahringer-Streich → Zipcode:32321 → City:Grand Ridge
Path 3: TX001 → Bahringer-Streich → Zipcode:32440 → City:Bristol

TX002 → Smitham-Boehm → **2 paths**

Path 1: TX002 → Smitham-Boehm → Zipcode:80137 → City:Watkins
Path 2: TX002 → Smitham-Boehm → Zipcode:80137 → City:Littleton

Actual Transactions: 2 **Counted Paths:** 5

Multiplication Factor: 2.5x

Solutions:

- Create direct relationships when appropriate
- Ensure no duplicate nodes through MERGE instead of CREATE
- Use COUNT(DISTINCT) when querying paths

UC San Diego

Lessons Learned & Future Work

Future Work

- Real-time implementation
    - query caching for quick retrieval
- Additional data sources
    - more data on cc transactions could allow for additional databases, tables, graphs, etc. and more insightful analysis
- Advanced ML integration
    - employ pagerank and community detection for fraud entity identification
    - model fraud detection through combination of tabular and graphical data

UC San Diego

THANK YOU

UC San Diego