# Fraud Analytics using a different approach, GraphDB data platform

(30 July 2025)

**Overview**

Financial institutions on their own only have what’s commonly referred to as a “on US” view of activity on an account. As it stands Fraud actors are taking advantage of this, hiding their activity by building up accounts/profiles over time, spanning multiple financial service providers and moving funds around between these distributed accounts.

The below scenario is positioned from the viewpoint of a national clearing house (NCH), acting as a Clearing and Settlement interchange, which operates between Financial service providers.

Using this position the NCH in question has the unique position where by they can provide centralized inter-bank Fraud Analytics service, through this they are able to Fraud Score all inter-bank financial transfers, thus they will have a more complete view of the wider flow of funds.

For the Blog below I’ve localized it based for Ireland. As such we’re using Irish personal identification numbers, Banks and addresses, mobile device and land line numbers, etc.

Ireland uses a **Personal Public Service Number (PPS Number)** as their social security identifier. Here's the format:

## PPS Number Format

* **Pattern**: 7 digits + 1 or 2 letters
* **Example**:  1234567A or 1234567AB

## Structure

* **First 7 characters**: Sequential numbers (1234567)
* **8th character**: Always a letter (A-W, excluding I and O to avoid confusion)
* **9th character**  (optional): Second letter for people born after 2013, or in special cases

For this we are primarily interested 5 data products:

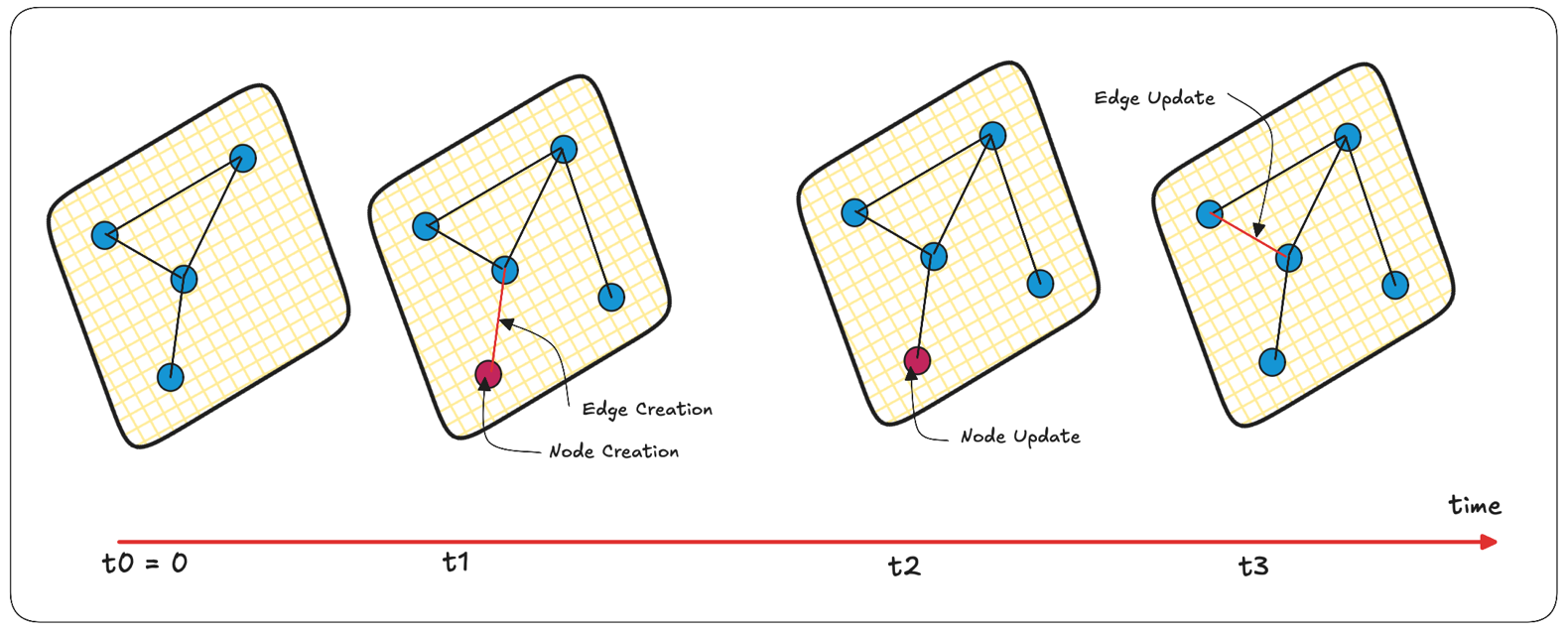
1. Fraud scoring for all transactions.
2. Banks provides for individually held accounts an accoutNumber mapping to PPS/Social Security Number (SSN) list.
3. Banks provide for corporate held accounts and accoutNumber mapping to Company Registration Numbers.
4. National Company register provide a company registration number together with the PPS/SSN of the registered owners, if this includes registered company address would be highly beneficial.
5. National Dept of Home Affairs (Home Office) provide profile for PPS/SSN.
6. A Stretch wish’s, Mobile Telco Operators provide Mobile Device Number (Cell Phone Number) mapped to PPS/SSN. Further add to this the Telco’s “Know your Customer/KYC” information referencing the registered residential address of the owner of the device.

Bringing the Fraud score for the transaction together with the above data products will re-write Fraud Analytics for any market.

But, we’re note done, the last component, for which the importance should never be under estimated is time… The ability to identify and analyse patterns as they develop/emerge over time.

Bad actors don’t create a profile in one step and then use them. They build up a profile over time, slowly so as not to attract attention.

For this we need to introduce what’s referred to as a temporal dimension, basically a time based view by decorating all activity/events with a eventTime, be that a financial transaction or a data update for an account, corporate, person, an address etc.



An account as a node and its surrounding nodes and relationships change over time.

Why Graph Database?

For many years relationship databases have been used to model 1 to many relationships, and they are amazing doing that, but it have to be recognised that they do battle with Many to Many.

Many-to-many relationships are an important distinguishing feature between different data models. If you application has mostly one-to-many relationships (tree-structured-data) or no relationships between records, the document model is appropriate or a standard RDBMS.

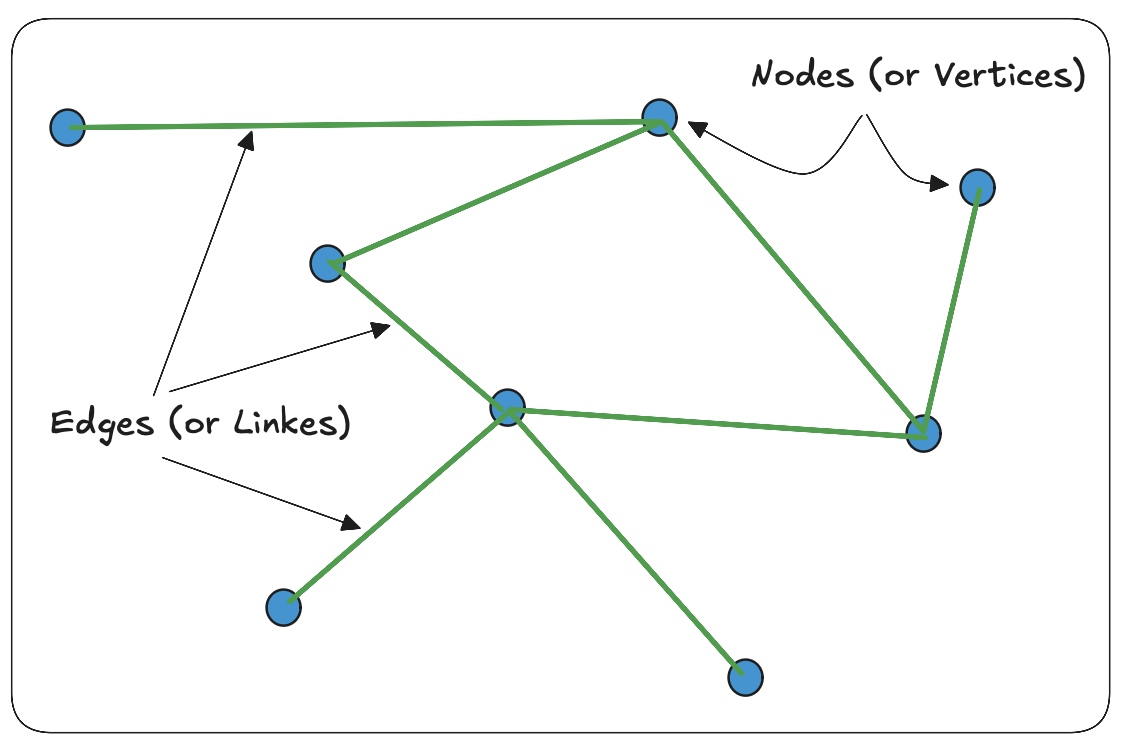
But what if many-to-many relationships are very common in your data? The relational model can handle simple cases of many-to-many relationships, but as the connections within your data become more complex, it becomes more natural to start modelling your data as a graph.

What is a Graph?

Graph databases are a type of NoSQL database that use graph structures (nodes, edges, and properties) to represent and store data, emphasizing relationships between data entities. Unlike relational databases that use tables, graph databases excel at handling complex, interconnected data and are well-suited for tasks like fraud detection, recommendation engines, and knowledge graphs.

Basic GraphDB Concepts

A graph consists of two kinds of objects: vertices (also known as nodes or entities) and edges (also known as relationships or arcs).



Data in a graph database is primarily stored as 3 different objects:

Based on Property Graphs model & the triple-store concept which defines 3 primary “actors”, namely:

Subject, Predicate, Object resulting in a Relationship Graph, modelled as:

* Nodes (Subject and Objects) - a node is the thing you are storing in the database. Thinking relationally, this is similar to a record in a table. If the table holds details on Transactions in this case, each record about a single Transaction will be their own node. A node can be an instance of any entity, a person, a place, an address, a thing, etc and the same graph database can hold instances of multiple types of these entities.
* Edges (Predicates) - an edge is the relationship between nodes. Again, thinking relationally, they are similar to a foreign key between nodes. Relationships are not mandatory but they can be many-to-many. The same nodes can be related to each other in multiple different ways.
* Properties (Properties associated) - extra non-mandatory attributes that can be added to either a node or an edge.

For our use case, at a simple level.

Subject (p): Who => Outgoing Account/the Debtor

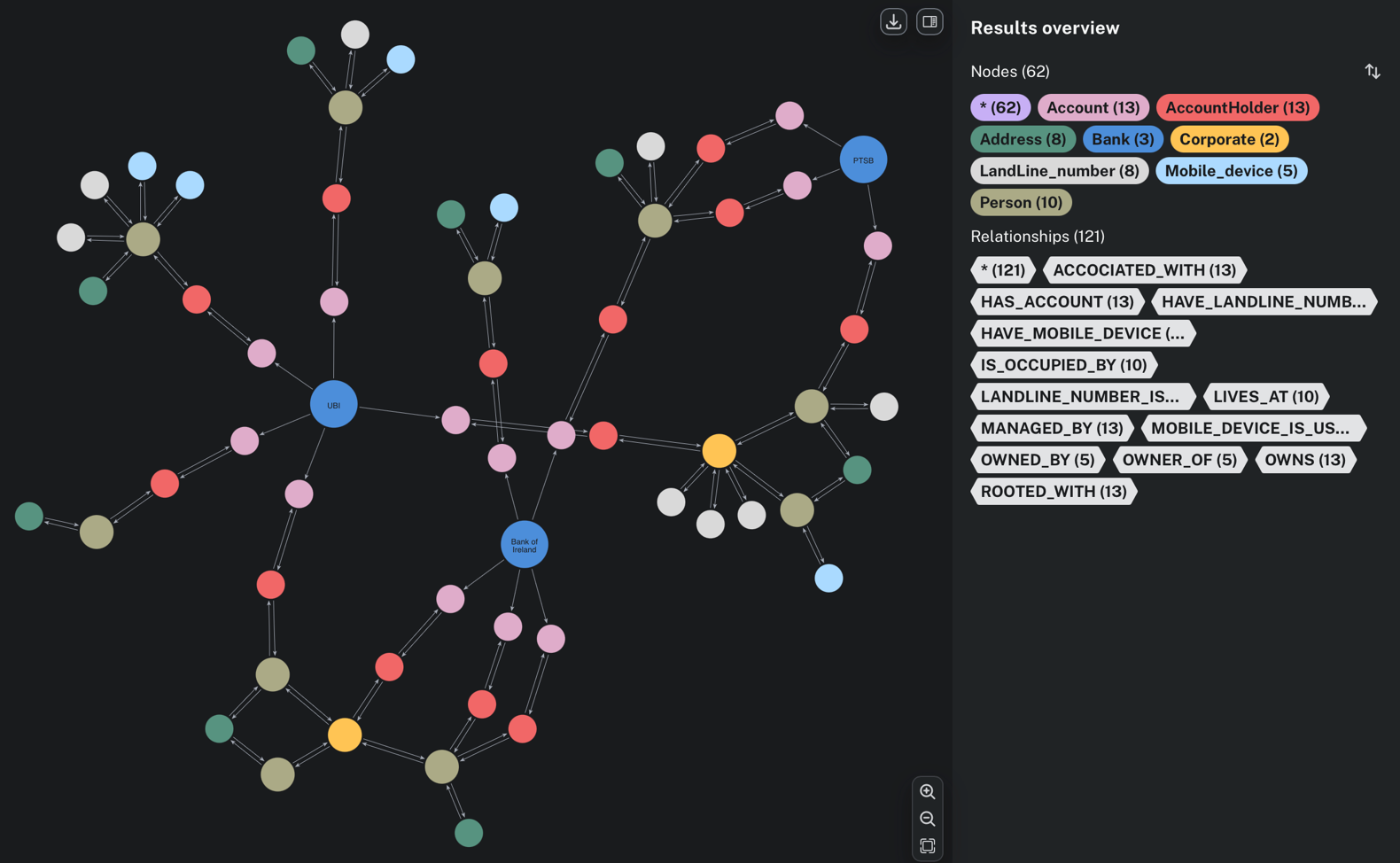
Predicate (s): What => Payment/Transaction also referred to as the Verb

Object (o): Who => Incoming Account/the Creditor

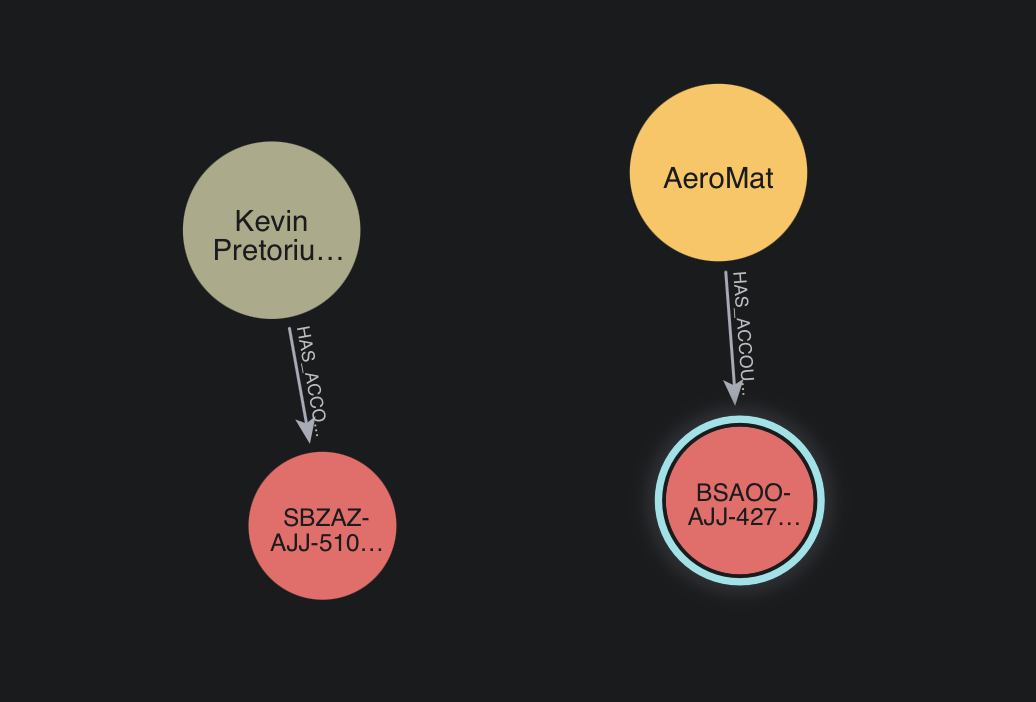
Properties (p): Additional detail.

Below is an imaginary graph depicted, mapping out:

* 4 Banks,
* Accounts,
* Account Holders (ah), ah being either a Corporate or individual depicted as a Person’s.
* A Corporate also owned by 1 or more individuals model as person nodes, all associated with
* A Person.
* Address/s, associated with
* mobile devices and further
* land line numbers.
* … we can add more here like Internet Service Providers, IP’s, if available location based tracking of transactions.



The above can be replicated by following the following GIT REPO: [financial\_txn\_flow\_using\_graphDB](https://github.com/georgelza/financial_txn_flow_using_graphDB.git)

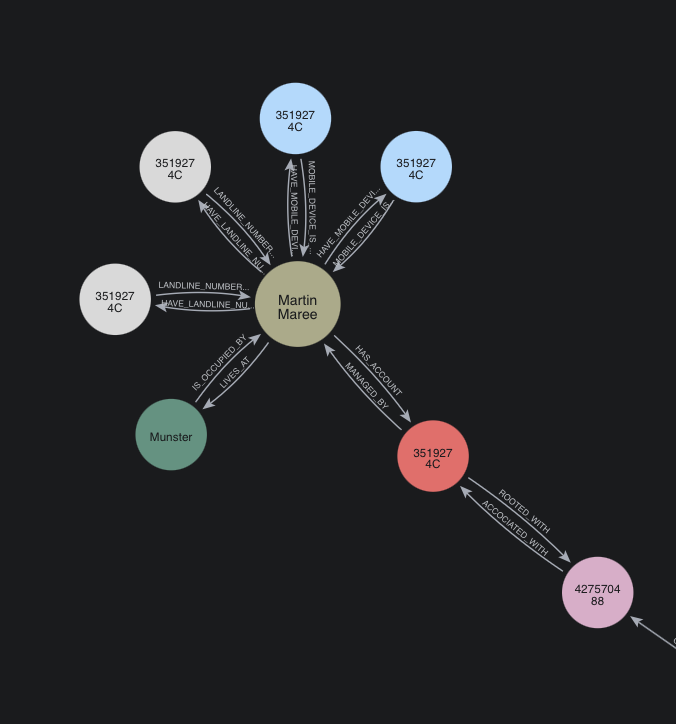


At the simplest here we’ve showing an individual as “Has Account” and a corporate as   
Has Account”

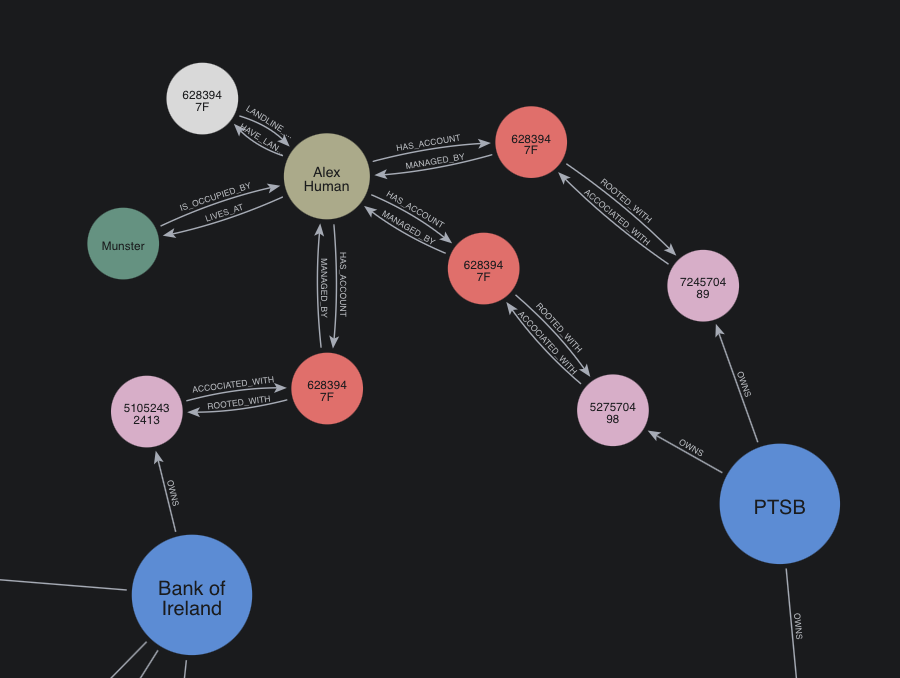
If we now take a closer Look:



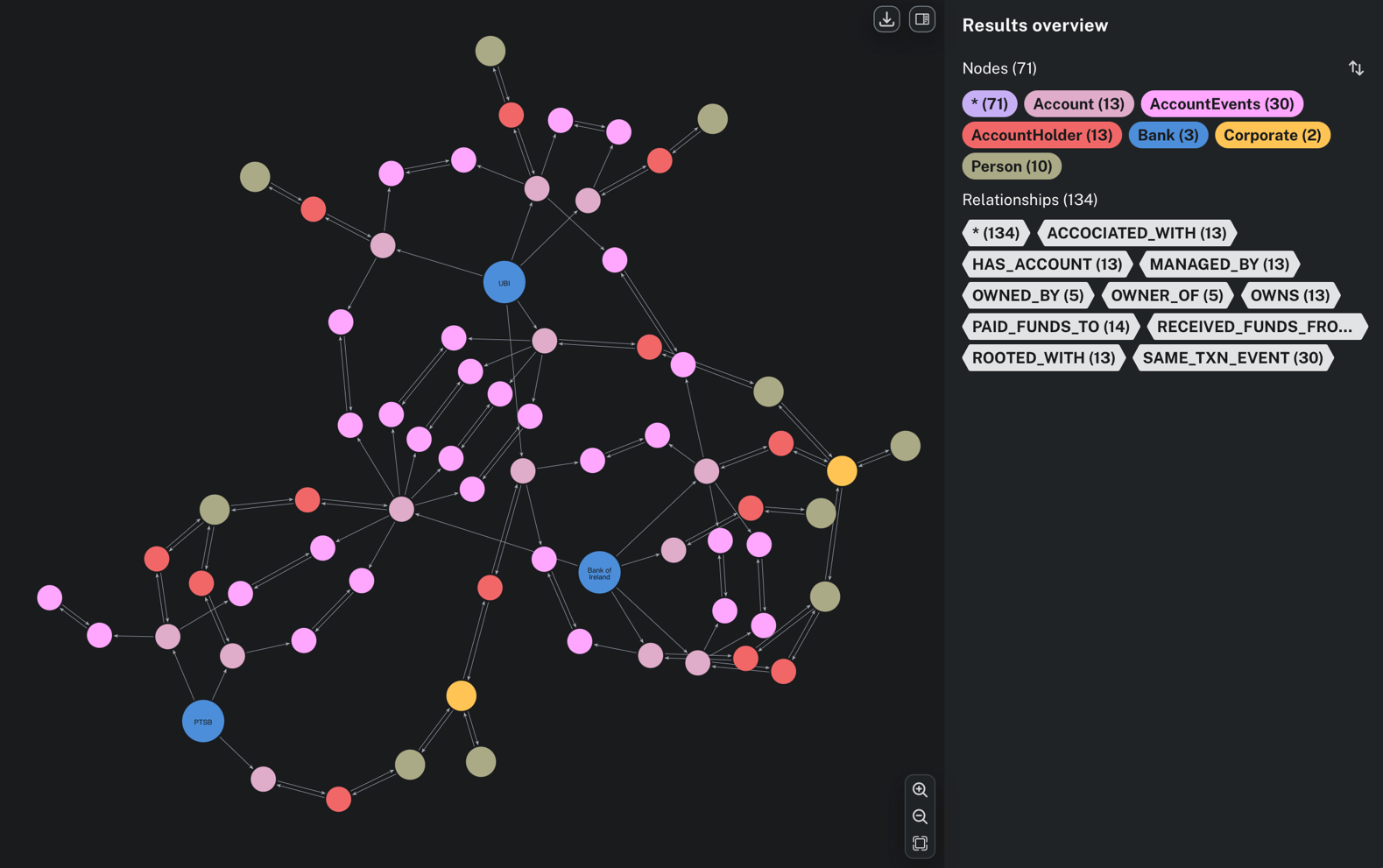
We can see some interesting facts around Alu Engineering, ownership and residential commonality (Notice: Ben and Karen share an address as depicted by the parcel number 3003).



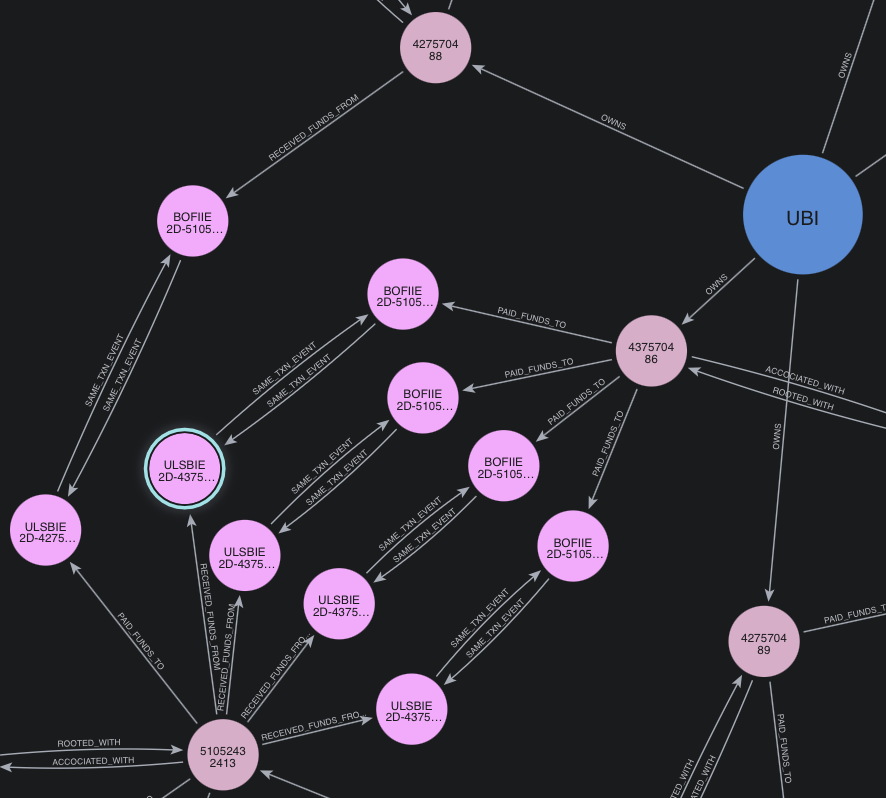
Above we can see an overview of Martin Maree, which mobile devices he owns, account, landline number and address.



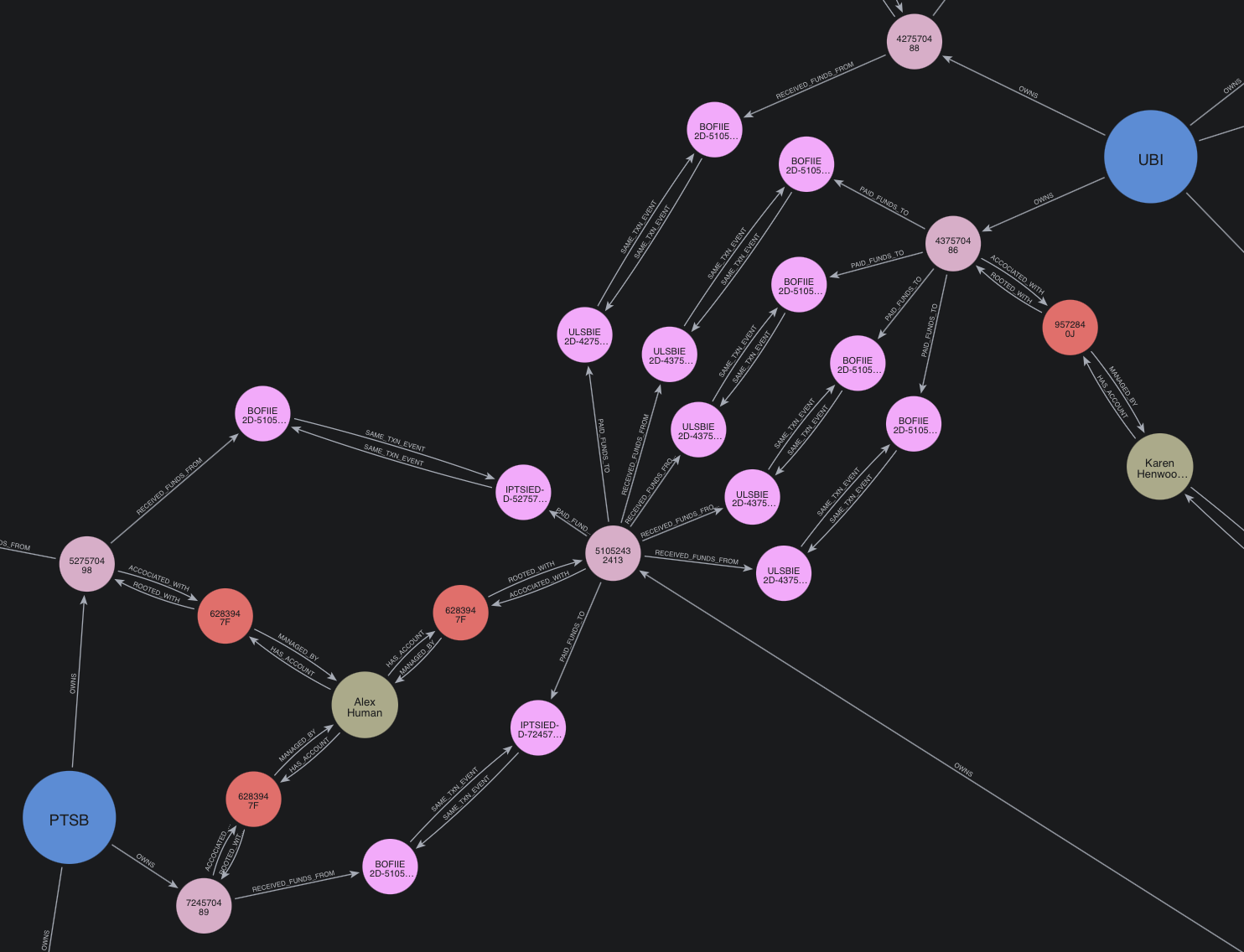
The above diagram becomes interesting when we notice Alex have 3 bank accounts, 2 at PTSB and one at Bank of Ireland. We can also see how landline number and physical address.



So in the above diagram I removed landline numbers and mobile device information as well as addresses... But we now have is Transactions, depicted as AccountEvents.



So here we show how multiple transactions have occurred between the 2 accounts, each depicted as brown circles.



More interesting it shall become, we zoom out a bit, we can now see Alex Human, and the 3 accounts he has, and then account right in the centre which has been used to move money via 4 transactions from Karen to Alex.

At the simplest, this is some of the power of GraphDB’s, exposing relationships, intersect points.

Basic Introduction to GraphDB

* [Learning Graph Databases - Part 1](https://medium.com/@wrisovkarmakar/cypher-on-aws-neptune-graph-db-ef20484a3e29)
* [Learning Graph Databases - Part 2](https://medium.com/@wrisovkarmakar/learning-graph-databases-c5aa54bb9807)
* [Learning Graph Databases - Part 3](https://medium.com/@wrisovkarmakar/learning-graph-databases-a694f43ea0db)

Popular GraphDB Technologies

* [Neo4J](https://neo4j.com/product/graph-data-science/))
* [Amazon Neptune](https://aws.amazon.com/neptune/)
* [ArangoDB](https://arangodb.com/)
* [Azure Cosmos DB](https://azure.microsoft.com/en-us/products/cosmos-db)
* [JanusGraph](https://janusgraph.org/)
* [TigerGraph](https://www.tigergraph.com/) and
* [OrientD](https://orientdb.dev/)
* …

As per a cursory Google search ;)

Here's a more detailed look at some of the leading graph database platforms:

1. Neo4j: Often considered the most popular graph database, Neo4j is a native graph database known for its performance, scalability, and support for ACID transactions. It uses the Cypher query language, which is optimized for graph traversal and querying.
2. Amazon Neptune: A fully managed graph database service, Amazon Neptune supports both Property Graph and RDF models, offering flexibility for different graph modelling needs. It's a good choice for cloud environments and regulated industries.
3. ArangoDB: A multi-model database, ArangoDB supports graph, document, and key-value data models. It uses the AQL query language, allowing for complex graph traversals and joins.
4. Azure Cosmos DB: Another multi-model database, Azure Cosmos DB offers Gremlin API support (based on TinkerPop) for graph needs within Microsoft stacks.
5. JanusGraph: A distributed, open-source graph database designed for large-scale graphs, JanusGraph supports various storage backends like Apache Cassandra and HBase.
6. TigerGraph: A native graph database, TigerGraph focuses on deep link analytics and real-time recommendations.
7. OrientDB: A multi-model database that combines graph and document models.
8. [Memgraph](https://memgraph.com/): An in-memory graph database that excels in real-time analytics and stream processing.
9. [Stardog](https://www.stardog.com/): A knowledge graph platform that combines graph database capabilities with semantic reasoning.
10. [Dgraph](https://docs.hypermode.com/dgraph/overview): A distributed graph database designed for scalability and performance.

When choosing a graph database, consider factors like the size and complexity of your data, the types of queries you need to perform, your existing infrastructure, and the level of support and scalability required.

Summary

At this stage, we’re just scratching the surface of what’s possible. For those that follow my blogs will know I love real time data.

If we were to bring that dimension also into scope, using technologies like i.e.:

* [Apache Kafka](https://kafka.apache.org/) (with either direct publish onto a topic or using Apache Kafka Connect framework to source data),
* [Apache Flink](https://flink.apache.org/) for streaming processing (think [PyFlink](https://nightlies.apache.org/flink/flink-docs-master/docs/dev/python/overview/) calling external API’s to compute Embeddings).
* Add to that additional technologies like Document stores e.g.:
  + [DocumentDB](https://aws.amazon.com/documentdb/) by Amazon or
  + [Mongo Atlas](https://www.mongodb.com/) by MongoDB for storing our unstructured data,
* Ingesting reference data from distributed data sources using Database CDC (Change Data Capture) technologies by [Debezium](https://debezium.io/).

#### All coming together to create a highly enriched real time data stream leading into our GraphDB environment, allowing us to us AI/ML, GNN’s ([Graph Neural Networks](https://neo4j.com/blog/developer/demystifying-graph-neural-networks/)), SNA (Social Network Analysis ), etc to expose patterns, relationships, the world quickly becomes very interesting.

Further Reading

Now that I got your attention, some further reading.

[Neo4J: industry-use-cases/data-models/transactions/transactions-base-model](https://neo4j.com/developer/industry-use-cases/data-models/transactions/transactions-base-model/?_gl=1*10za4cz*_gcl_au*MTc2MjA3MzA3NS4xNzUzMjY3Mzc5*_ga*NzU1MTc3ODQwLjE3NTMyNjczNzk.*_ga_DL38Q8KGQC*czE3NTMzNzE3MDUkbzUkZzEkdDE3NTMzNzE4OTUkajUwJGwwJGgw*_ga_DZ)

[A Comprehensive Guide to Temporal Graphs in Data Science](https://www.analyticsvidhya.com/blog/2023/12/a-comprehensive-guide-to-temporal-graphs-in-data-science/)

[Mastering Fraud Detection With Temporal Graph Modeling](https://neo4j.com/blog/developer/mastering-fraud-detection-temporal-graph/)

[Build real-time fraud detection solutions using Amazon Neptune ML](https://aws.amazon.com/blogs/database/build-a-real-time-fraud-detection-solution-using-amazon-neptune-ml/)



**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

<https://medium.com/@georgelza>