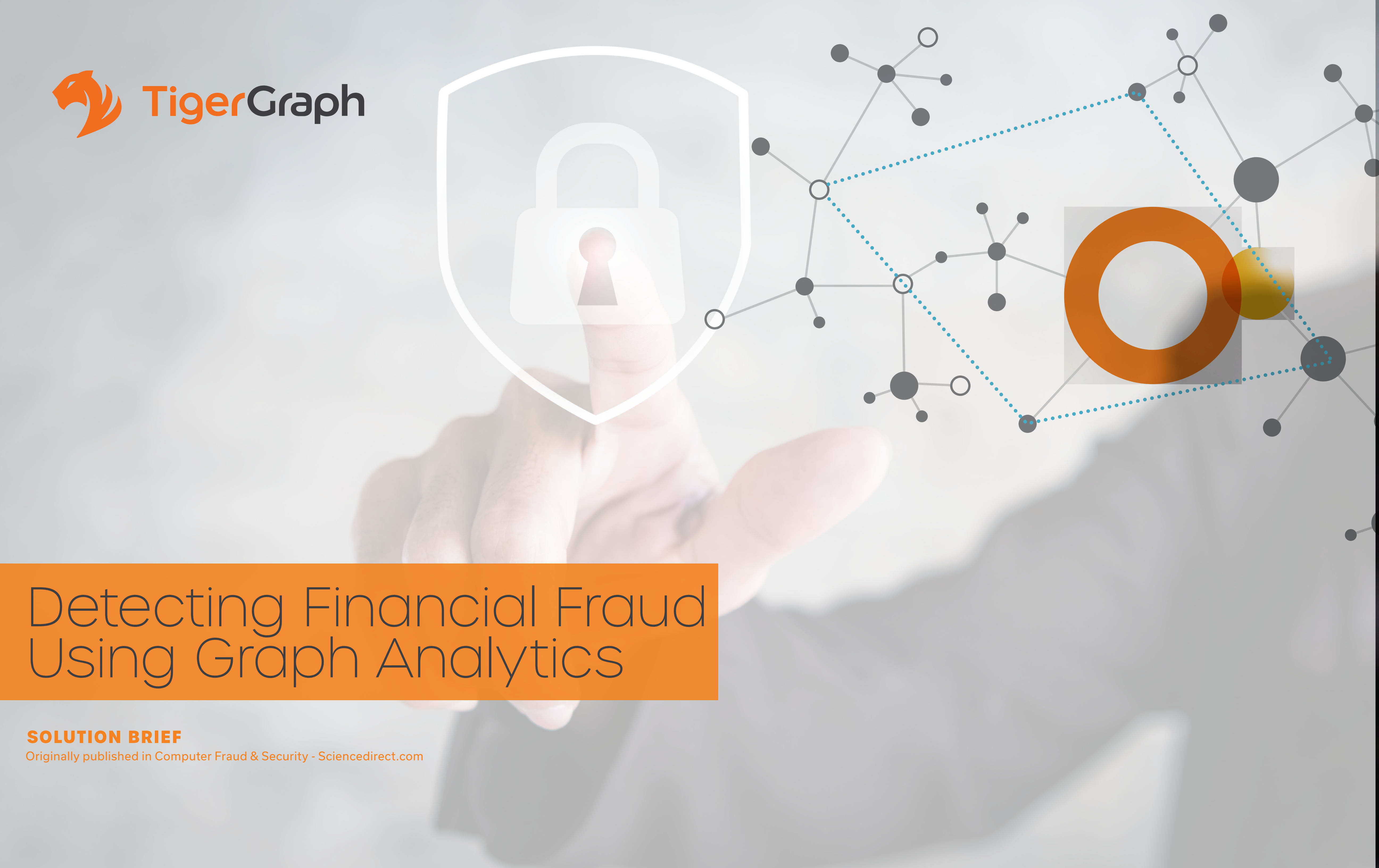


TigerGraph

Detecting Financial Fraud Using Graph Analytics

SOLUTION BRIEF

Originally published in Computer Fraud & Security - Sciencedirect.com

Detecting Financial Fraud Using Graph Analytics

Advanced analytics in graph databases can uncover suspicious patterns of online payment activity in ways that other approaches cannot.

Online fraud will cost businesses more than US$200 billion between 2020 and 2024 according to Juniper Research. This stunning amount is driven by the increased sophistication of fraud attempts and the rising number of attack vectors. And, while banks are fighting back harder than ever, fraudsters have adjusted their techniques to remain below the radar.

Fortunately banks have a powerful weapon in the war against fraud: graph analytics. Advanced analytics in graph databases can uncover suspicious patterns of online payment activity in ways that other approaches cannot – helping to stop fraud before it can be committed.

Gartner has identified graph analytics as one of the ten data and analytics technology trends that can transform businesses. It predicted the sector will grow by 100% annually through 2022 "due to the need to ask complex questions across complex data, which is not always practical or even possible at scale using SQL queries".

It's no surprise, then, that companies like China UnionPay, the largest payment card provider in the world, are investing in graph analytics, along with four of the five top global banks in the world.

Graph techniques can analyse thousands of customer data points – and the crucial relationships between them – to deliver fraud alert scores in real time. Graph can be used for fighting financial fraud by analysing the links between people, phones, and bank accounts (among other things) to reveal indicators of fraudulent behaviour, not only helping banks pinpoint suspicious activity in a sea of data but also giving them the tools to explain what's going on.

A key feature of graph is its ability to perform at speed, especially compared to relational database solutions such as SQL. Banks have been doing fraud detection for years, but one of the things that graph brings to the party – apart from depth of analysis – is speed. While SQL depends on bulky table joins, graph is less memory intensive and able to handle a greater query load.

Online fraud will cost businesses more than US $200 billion between 2020 and 2024 according to Juniper Research

Fraud is Becoming More Complex

Fraud detection systems tend to rely on looking at transactions that exceed preset levels or people who try to max out a credit card with no intention of paying it off. These types of suspicious transactions are easy to detect because they rely primarily on the information in the transaction itself, looking at the amount, the destination or other properties that might generate warning signals.

However, fraud has become more complex than that as fraudsters have learned to work across multiple accounts, including mule accounts not directly controlled by or associated with them, such that individual transactions look ordinary and would not trigger an investigation. This is an especially big problem as banks work to comply with anti-money laundering (AML) regulations, pushing fraudsters to ever greater lengths to obfuscate their activities.

Fraudsters will employ hundreds of accounts to launder money in a technique known as 'smurfing'. In this technique, money that needs to be laundered is disbursed to accounts in quantities small enough to avoid triggering automated reporting limits. The money is then 'layered', that is, mixed, divided and transferred from these accounts to other accounts, with the history of the money becoming more and more difficult to trace with every transfer. Ultimately it is funnelled to the final destination, its origins virtually lost in a long and complex audit trail.

Layering requires a certain amount of setup as the fraudster must create all of the accounts they need with the banks, but once this is in place, it becomes easy to move large amounts of money quickly thanks to electronic banking APIs (application programming interfaces).

This is a network of activity, a pattern which can be traced and matched if you have the computational systems and resources in place to do it.

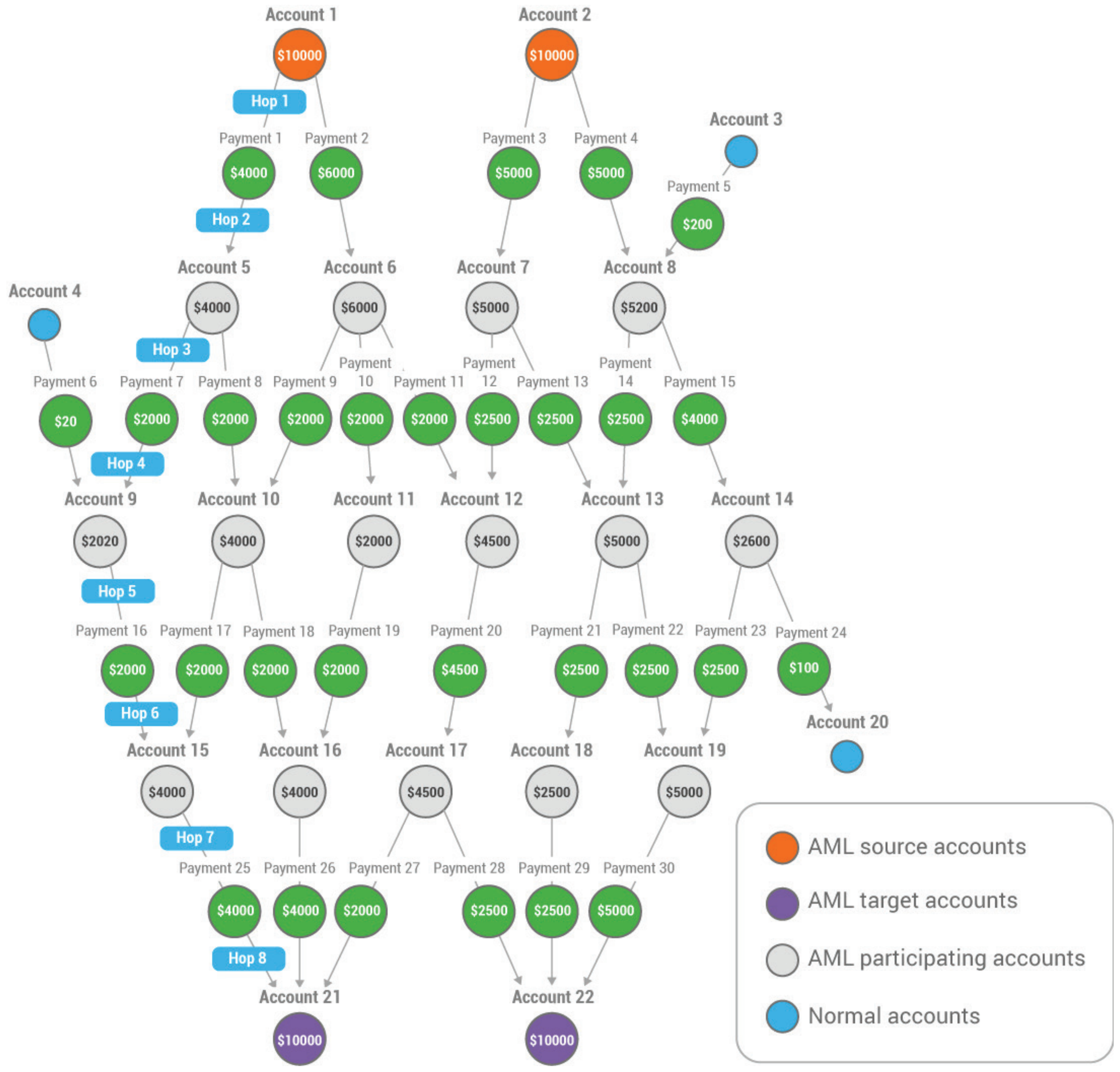


Figure 1: Layering as a technique for disguising the source of funds in money laundering

SQL Versus Graph

Many financial institutions have built their fraud detection systems on the backs of their legacy SQL databases because that's where the data is stored. While these SQL databases are optimal for processing transactions, they struggle with relational analysis as the number of connections or hops between data points grows.

To understand fraud, you have to examine a lot of contextual information. This brings in a number of different types of data – accounts, individuals, transactions, etc – all of which, in a relational database, would be stored in separate tables. To link this data together, you have to make a table join, a temporary construct created in memory which allows you to read across the associated data and extract the information that you require.

Table joins are computationally expensive. If you are looking at just one type of data, e.g. the transaction table, a relational database works fine, but in real-world situations, we work with mixed data which means we are constantly joining different types of data together. So, if you store your mixed data in several different tables, every time you want to link that data together, you have to create a table join. Clearly, the greater the depth of your enquiry, the more tables you are going to have to retrieve which is, in turn, going to affect your search time. In fraud detection – where you are tracing money across many people, accounts, and transactions – it may be necessary to access a number of tables.

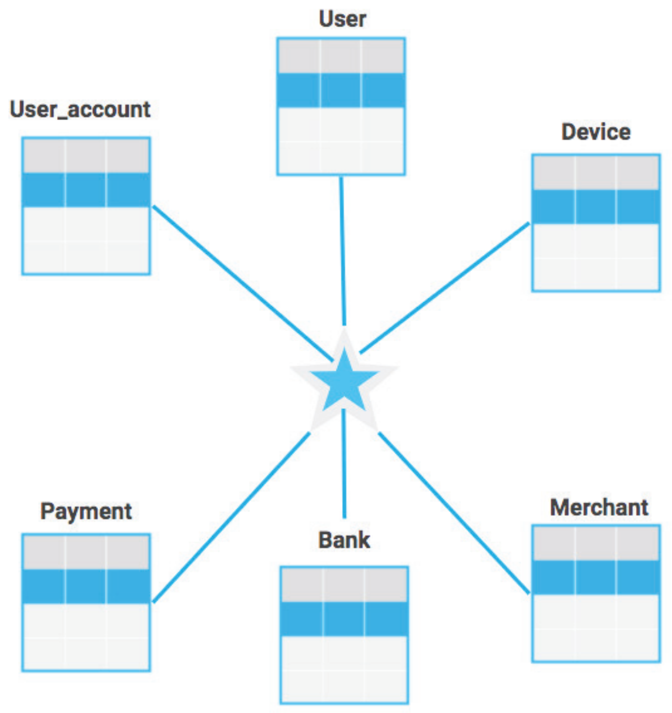


Figure 2: Complex table joins needed with a relational database for fraud analysis

SQL databases are optimal for processing transactions, they struggle with relational analysis as the number of connections or hops between data points grows.

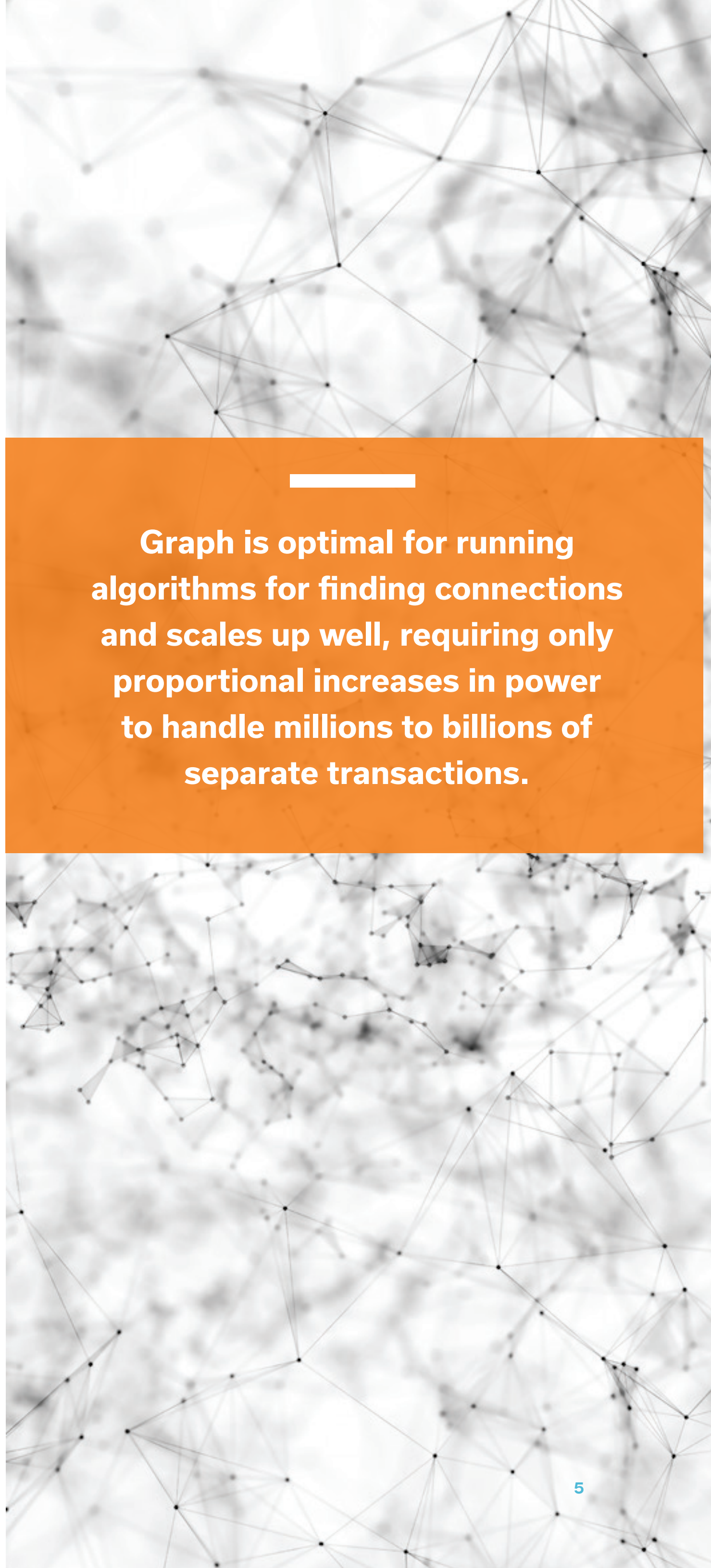
Consider the problem of money laundering. Suppose Alice wants to send $1 million to Bob without setting off internal bank alerts. She divides the money into 110 transactions of around $9,100 each to stay below the mandatory reporting threshold of $10,000. These are then deposited into 110 separate bank accounts. This is step one. Then, in step two, each of these accounts divides its money into transactions of between $100 and $1000 each and sends them to other intermediary accounts, some of which may be new and some of which Alice may have used in step one. Then repeat and repeat several more times. Each step is another layer and adds another hop to the depth of transactions that would have to be analysed to uncover this fraud.

After ten layers, there have been thousands of transactions and the source of the money in each account (which never exceeded $10,000) is well and truly obscured. Then all the accounts disburse their money to Bob. A cursory examination of the bank records would show Alice's company paying a lot of suppliers and Bob's company being paid by a lot of customers. No red flags because these scenarios fit the pattern of normal business behaviour.

However, if you could map it visually, it would show the money flowing out from a single artery into a lot of capillaries and then back into one main artery, revealing that there was a transaction for $1 million between Alice and Bob. Easy if you knew in advance which customers were fraudsters, but you don't. Alice's operation involves a lot of data stored in a number of different tables. There are bank accounts, bank customers (Alice and all of her aliases), and transactions. As a fraud investigator, you are looking at thousands of Alices paying money to suppliers and asking yourself, which one of these Alices is a fraudster? Most of them will be honest business people but you'll have to examine all of their transactions to a certain depth to understand which one of them warrants a red flag for further investigation.

Computationally, you have to create massive table joins to understand what is happening in this scenario – a table join that is big enough to track all transactions from all of your Alices through ten layers of payments to all of your Bobs. That is an unfeasibly large table join. And the problem is, if you tried to make your table join more manageable by reducing the number of layers, you would completely miss the connection between Alice and Bob.

In graph, you don't analyse connections by creating joins because the data is stored in a pre-linked format. In the money laundering scenario, the connections between the person, the bank account and the transaction are all directly linked to the transaction itself. Graph is optimal for running algorithms for finding connections. Community detection algorithms, for instance, would help to detect the fact that Alice was using a host of mule accounts to wash her money. Path detection would quickly identify that Bob was frequently the final destination for her money. And in the case where you found a community, but didn't know who was running it, centrality algorithms would help identify Alice as the kingpin. Graph also scales up well in this scenario, requiring only proportional increases in power to handle millions to billions of separate transactions.



Graph is optimal for running algorithms for finding connections and scales up well, requiring only proportional increases in power to handle millions to billions of separate transactions.

Graph for Large Dataset Analysis

A key concept in understanding graph databases is index-free adjacency. While SQL systems have processing overheads associated with every index lookup, in graph the links to associated data are loaded with every node. A key distinction here is the difference between native and non-native graph systems. There are a number of non-native graph solutions built on top of other databases which don't have index-free adjacency and are therefore not optimised for fast graph traversal. Relationship lookups in non-native graph solutions are slower because of the need to fetch the address of the related node from a central index. By contrast, native graph solutions store this in the node itself.

Relational databases are optimised to retrieve data row by row, with link analysis not a primary concern. With graph, link analysis is its primary function. Another feature of index-free adjacency is that it doesn't put extra overheads on your system as your database grows. While relational databases get slower as they expand due to indexes adding more and more layers of indirection, each requiring another expensive lookup, native graphs enjoy constant performance making them ideal for collecting ever greater amounts of contextual data which is essential in fraud analysis.

Scale is an issue to consider in graph. A small bank might have six or seven million customers, while a large bank might have fifty to one hundred million customers, with terabytes of data. For this you need not just a native graph database, you need a distributed graph database. This allows you to put some of your data on one server, some more data on another server, and so on, without the user having to worry about how it is partitioned. A distributed, native graph solution allows you to implement algorithms for search, pathfinding, clustering, centrality, dependency analysis and similarity in ways that simply won't perform at scale in a relational database.

A small bank might have six or seven million customers, while a large bank might have fifty to one hundred million customers, with terabytes of data. For this you need a distributed graph database.

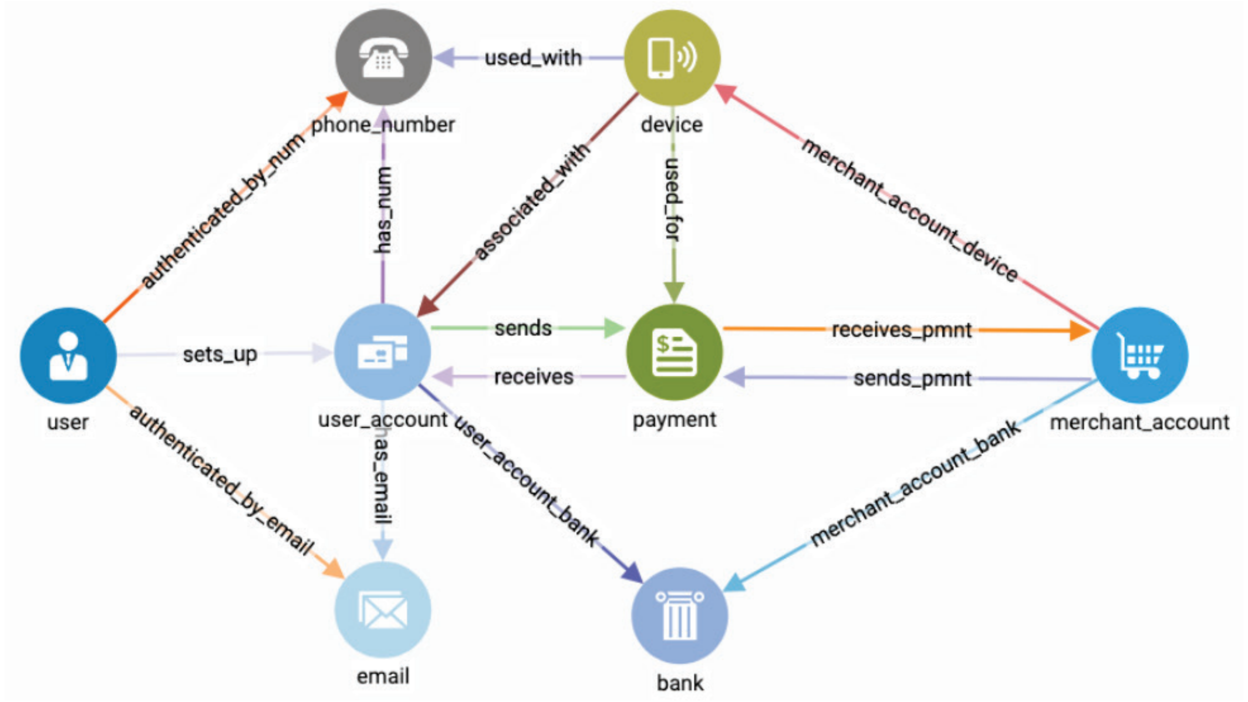


Figure 3: Example of graph schema used in fraud detection (screenshot of TigerGraph)

Graph at Work

One of the ways to detect fraud is to find groups of transactions or persons that have an unusually high number of interconnections. To detect such groups or communities, you need an algorithm which can efficiently study and assess the entire graph's structure. One such community detection algorithm is Louvain modularity or, simply, the Louvain method.

Louvain has so far been the most effective version of a host of algorithms which try to maximize a score called modularity. Other algorithms, or algorithms which try to shortcut Louvain by using random sampling, either don't get as good scores, don't get consistent scores, or take longer to run.

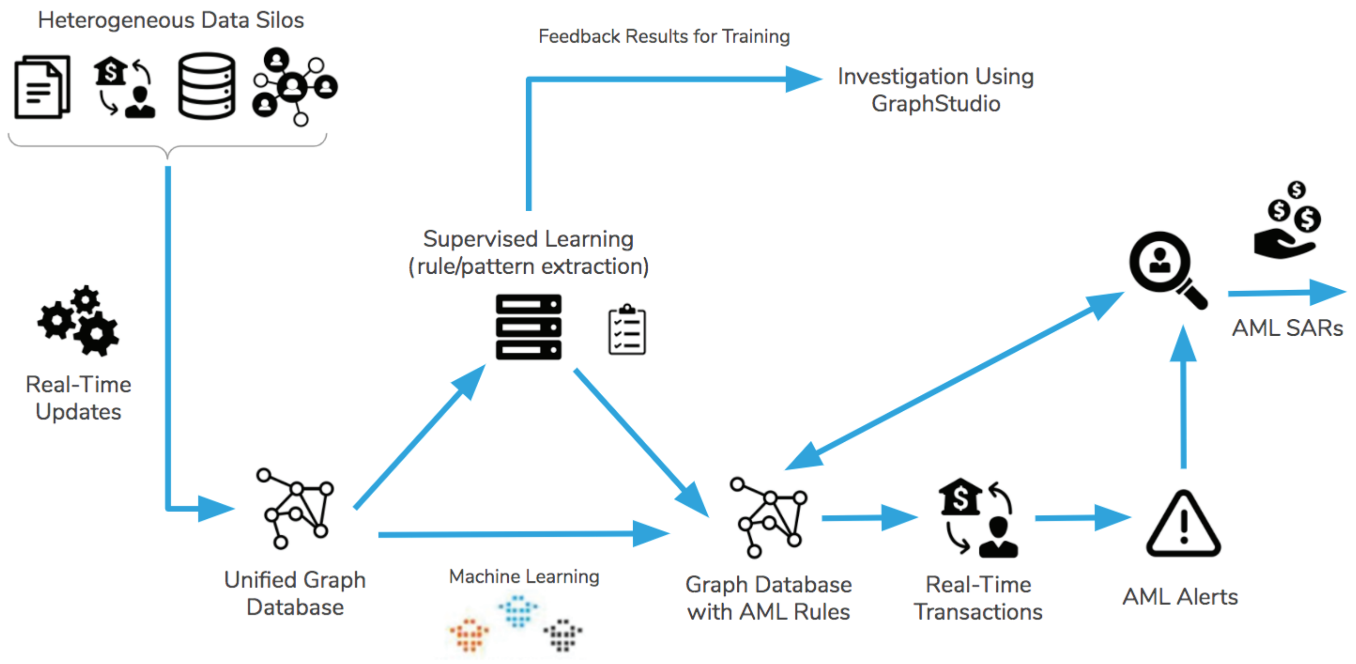
Other algorithms for working on fraud include PageRank. As an algorithm for determining the influence of web pages based on page-to-page referral, it can also identify who is "pulling the strings" in a community of suspicious financial transactions.

One of the points that is sometimes raised about graph is that people say we can already do fraud detection with SQL queries, which is true when you are looking at a limited number of lookups. This is fine for doing backward analysis where you have an idea of where you want to look for suspicious activity.

However, if you want to examine all connections – as in the case where you don't know which transactions are suspect and you want to find out – that is another case altogether. The SQL query quickly becomes large and unmanageable as the dataset grows because now you want to jump through all of those hops and find out where they went. The problem becomes even more complex because as you work your way through the data, you may need to look left (upstream) and right (downstream) while performing a what-if query – all of which would be complex and difficult to maintain with SQL.

So, three things that push SQL over its limit are the history, the present and the what-if question – together they kill SQL which is why people are looking at graph to solve these problems. As SQL works its way through a tree or network diagram, examining the nodes, fraud queries and algorithms will want to examine nodes to the left and the right of the active node, and a separate SQL query has to be written for each of these hops. To do this, you have to know in advance – which you won't – the structure of the network and which branches and nodes will need to be explored to be able to write the SQL. In some cases, you will need to backtrack and traverse other branches as part of your query which may in turn throw up new paths to traverse.

Anti-Money Laundering Workflow with Machine Learning



Heterogeneous Data Silos
Feedback Results for Training
Investigation Using GraphStudio
Supervised Learning (rule/pattern extraction)
Real-Time Updates
AML SARs
Unified Graph Database
Machine Learning
Graph Database with AML Rules
Real-Time Transactions
AML Alerts

Figure 4: The anti-money laundering workflow using machine learning

Graph and Machine Learning

Graph and machine learning can work together to deliver even better results. Machine learning (ML) alone may take the accuracy of fraud signals as high as 70-80%, but combining it with graph can raise it to 90% or higher.

It's usually not feasible to run ML algorithms directly on live data because it's computationally expensive and the data is constantly changing. However, ML systems offline can ingest historical data generated from the graph. From this data, patterns can be generated which are indicative of suspect behaviour. The patterns can then be loaded into the graph and run against pattern-matching algorithms to flag suspect activity.

You can also use AI to create entirely new decision trees, starting by generating large numbers of random rules. This 'forest of decision trees' is then applied to data to see which ones work and which don't, with rules weeded out based on their effectiveness. It is a useful technique for creating unique rules that would otherwise not be created, but it's only possible if you have all the data available at the same time, which is what graph is good at.

A big concern with automated fraud detection systems is the issue of false positives. Firstly, it can absorb a lot of time and effort from human analysts dealing with the false positives and filtering out the hits that really need more investigation. And secondly, there is a concern about the impact on customers – as one telecoms company explained to us, they want to take automated fraud detection slowly because one of their biggest fears is getting a false positive and losing a legitimate customer over it. If you approach fraud detection with graph without sufficient preparation, you can run the risk that suddenly everything looks like fraud.

However, graph feeds very well into machine learning – it is very good at generating data for training ML systems because it is very good at producing explainable models of what it has detected. Rather than simply giving something a score based on heuristics, graph can generate data on the links between different objects in the database which can be fed into ML systems for further analysis. This explainability extends to showing humans what's going on, as the linking data can be used to draw detailed diagrams from which humans can infer the relationships between the objects and ask further questions. The region of confidence (ROC) curve tells you how confident you are in the decision that a machine made. Some decisions you can accept from the machine alone based on the score. But if your confidence is below 50-60%, and depending on the value of the transaction, you might want to forward this to a human analyst. The analyst really needs to understand why the machine thought there was a 50% chance that the transaction was bad – you don't want to force the human to go back to the beginning to work out why. You want them to take a case from the system and work it quickly and confidently with all the information to hand.

Graph databases are good at showing results graphically, making explainability one of its key strengths. This combined with the ability to explore contextual data makes it an asset in fraud detection.Feedback from the analysts can also be fed into ML and graph systems, creating a virtuous circle between humans and machines which can make fraud teams more efficient.

Banks are expected to spend $7.1 billion on AI in 2020, growing to $14.5 billion by 2024, on initiatives such as fraud analysis and investigation, according to market research firm International Data Corp.

Wall Street Journal, August 26, 2020

"Visa unveils more powerful AI tool that approves or denies card transactions"

Case Study - China UnionPay

China UnionPay is the world's largest payment card provider. In 2017 it reportedly moved the equivalent of $15 trillion over its network. For the bank, the switch to graph was driven by the need for massive scale and efficiency. With its previous relational database system, it was infeasible to do any fraud checking or even in many cases to detect when accounts went bad.

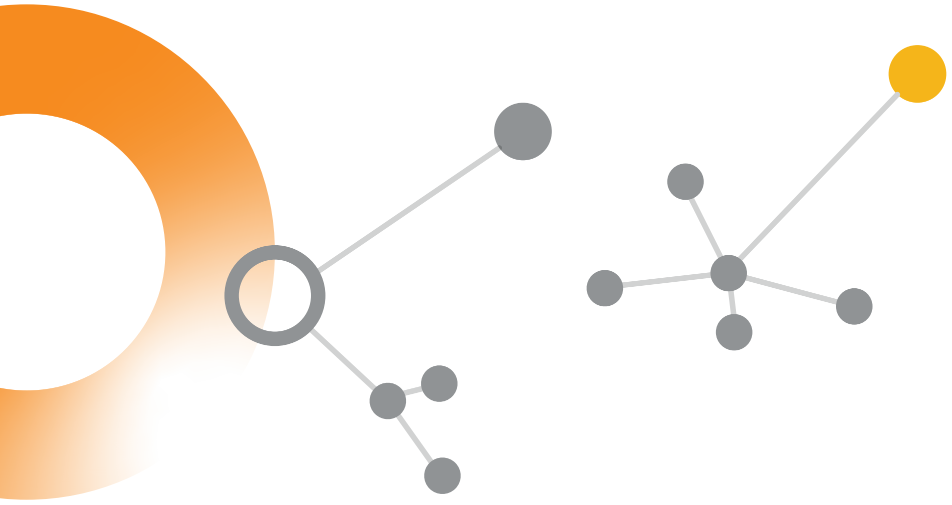
Graph is allowing the bank to perform these checks at speed and scale in ways it couldn't before. But it goes beyond simple fraud detection and account healthchecking, because once they had this in place, they were able to begin applying rules to more than just the transaction itself which allowed them to add context to their investigations. One system analyses historical transaction data and develops rules which can then be loaded into the live production database in the form of patterns to be matched. Once they have defined a subset of accounts and transactions that they are interested in, they are able to run complex algorithms like centrality and PageRank against a subgraph to expose new business intelligence.

Based on the confidence level of the result, suspect transactions can either be blocked automatically or passed to a human investigator to make the final determination. They in turn can use subgraphs to conduct further investigations. Of course, each financial institution operates within its own domain, so approaches will vary considerably. But the combination of graph and ML-based pattern mining provides an invaluable tool for combating fraud in a rapidly evolving financial environment.

Graph is allowing China UnionPay to perform fraud checking at speed and scale in ways it couldn't before.

Learn more and download TigerGraph for free

https://www.tigergraph.com/solutions/fraud-detection/



About TigerGraph

TigerGraph is the only scalable graph database for the enterprise. TigerGraph's proven technology connects data silos for deeper, wider and operational analytics at scale. Four out of the top five global banks use TigerGraph for real-time fraud detection. Over 50 million patients receive care path recommendations to assist them on their wellness journey. 300 million consumers receive personalized offers with recommendation engines powered by TigerGraph. The energy infrastructure for 1 billion people is optimized by TigerGraph for reducing power outages. TigerGraph's proven technology supports applications such as fraud detection, customer 360, MDM, IoT, AI, and machine learning.

The company is headquartered in Redwood City, California, USA.
Follow TigerGraph on Twitter at @TigerGraphDB or visit www.tigergraph.com

© TigerGraph, Inc. 2020 All Rights Reserved.