MEM GRAPH

FREE EBOOK

The Ultimate Introduction to Graph Analytics

INTRODUCTION

This white paper explains the fundamentals of graph modeling, graph analytics, and graph data databases, and how you can apply them to your next project.

If you're skeptical about yet another hyped technology, know that graphs are not a new concept and have been around since the 18th century. Graph analytics withstood the test of time having been applied and battle-tested across industries and applications for decades.

So why should you care and keep on reading? In a nutshell, graphs unlock a whole new world of capabilities and insights that are near-impossible to achieve using traditional data analysis technologies. As a result, an increasing number of businesses are adopting graphs to enhance core capabilities, gaining significant competitive advantages. If you're not seriously considering graph analytics, chances are you will fall behind your competitors.

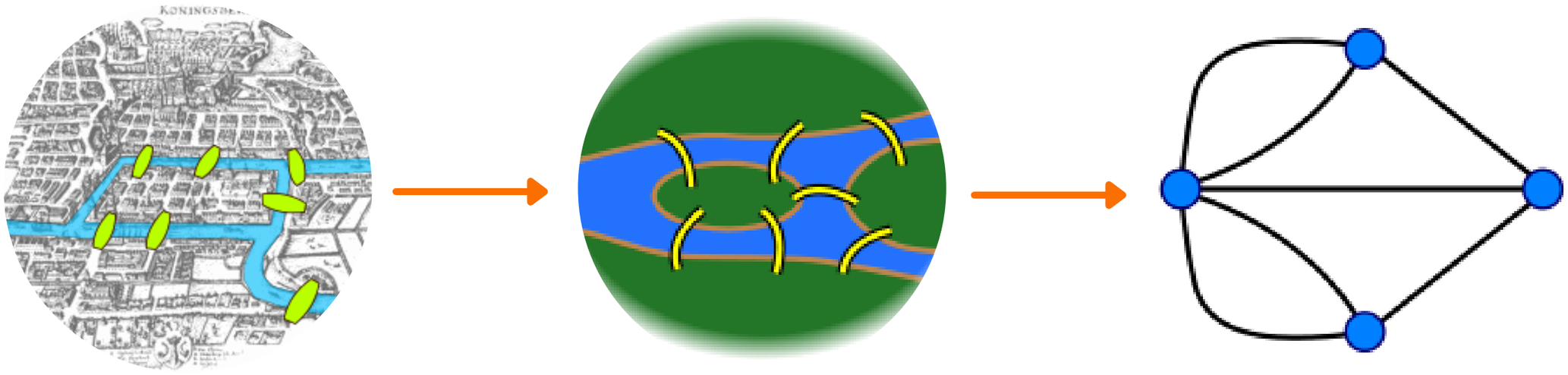
GRAPH THEORY – WHERE IT ALL STARTED

The basic idea of graphs was first introduced in the 18th century by Swiss mathematician Leonhard Euler through his work on the famous "Seven Bridges of Königsberg problem". This is generally considered as the origin of graph theory.

Leonhard Euler
Portrait by Jakob Emanuel Handmann (1753)

The city of Königsberg in Prussia (now Kaliningrad, Russia) was set on both sides of the Pregel River and included two large islands — Kneiphof and Lomse — which were connected to each other, or to the two mainland portions of the city, by seven bridges. The challenge was to devise a walk through the city that would cross each of those bridges once and only once.

Euler quickly realized that the only relevant feature to his problem was the four bodies of land & the seven bridges. This allowed him to reformulate the problem in abstract terms and draw the first known representation of the modern graph; a set of objects, known as vertices or nodes, connected by a set of lines known as edges or relationships.



Original Problem | Relevant Problem Features | First Graph Representation

Today, we define Graph Theory as the study of structures that model objects and their relationships. Graphs can intuitively abstract many real-world dynamic systems from protein to cybersecurity networks, and provide an elegant and easy way to solve complex networking, optimization, pattern matching, and operational problems.

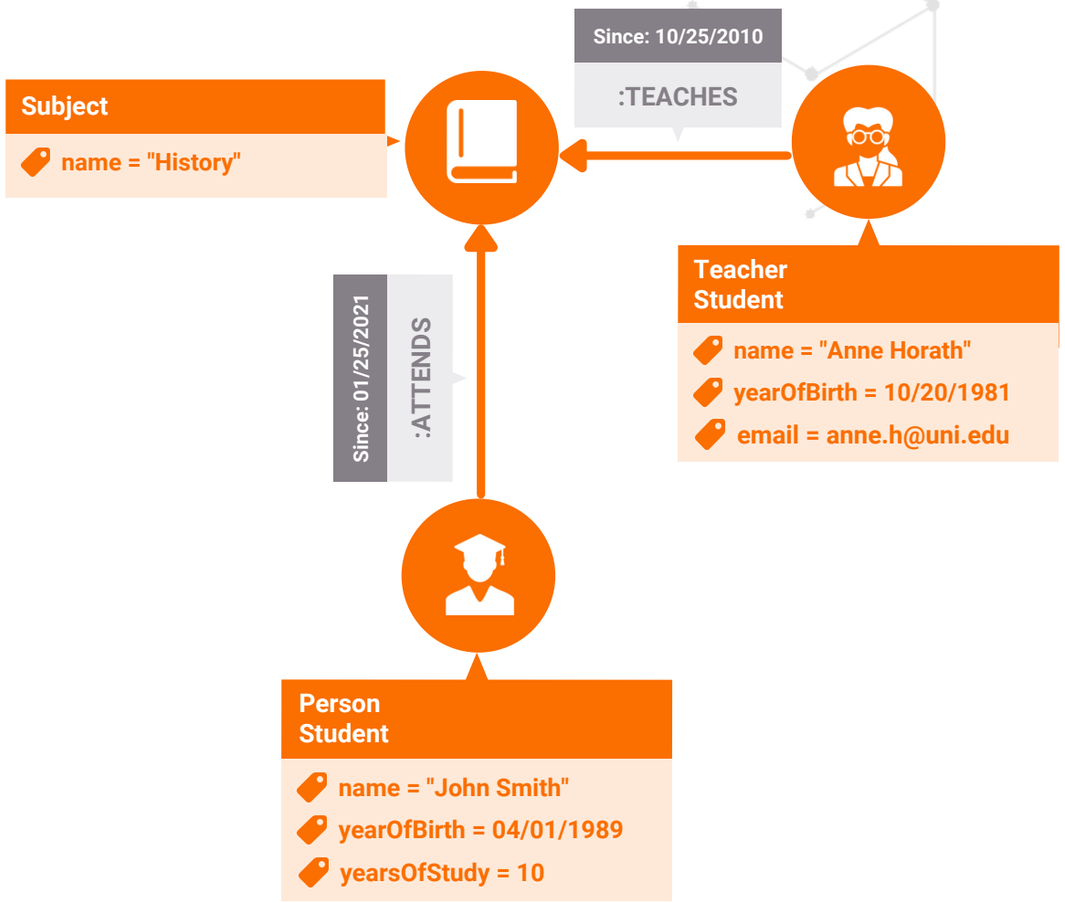
After decades of being an obscure, and theoretical branch of mathematics, only applied by research-focused institutions in highly technical fields, Graph Theory has made the leap to computer science and mainstream applications.

The good news is that, when computer scientists applied graph theory to code and data structures, they didn't change much. So, a lot of the terms you will encounter when working with modern graph tools will be the same you'll find in mathematical references to graph theory.

GRAPH MODELING – AN INTUITIVE WAY TO MODEL THE WORLD

If you try to model many of the real-world problems you are most familiar with (e.g your friend circle, your daily commute, your city layout, etc), you'll end up with some that look like a graph. This is simply because the graph model is arguably the most initiative way to model the world and especially complex systems.

The Property Label Graph Data Model consists of four components: **nodes**, **labels**, **relationships**, and **properties**. While nodes and relationships are the fundamental components, around which everything is designed, labels and properties enable you to enrich your data model with additional information.



Nodes: Represent entities in the graph and can have both labels and properties.

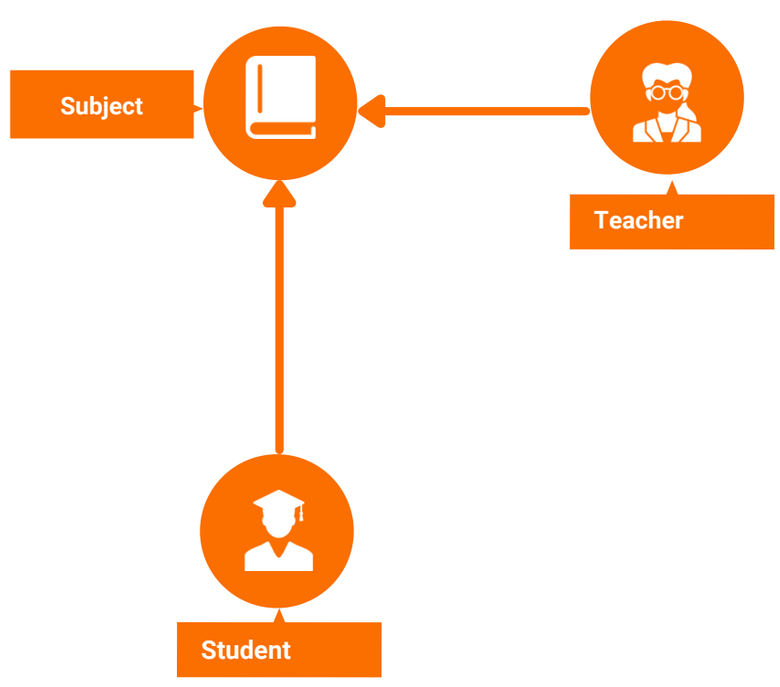
Labels: Used to shape the domain by grouping nodes into sets or categories.

Relationships: Lines that connect nodes to each other and represent a defined connection between them.

Properties: Key-value pairs of data stored on nodes or on relationships. They allow you to store relevant data about the node or relationship with the entity it describes.

NODES

Nodes often represent entities in the graph. They hold specific data in the form of properties represented as key-value pairs. To assign a role to each node, nodes can be tagged with labels.
When working with the model domain, nodes can be easily identified by searching for nouns that represent entities with a unique conceptual identity.
As you can see in our university example, a typical node could represent a university student, a professor, or a subject.

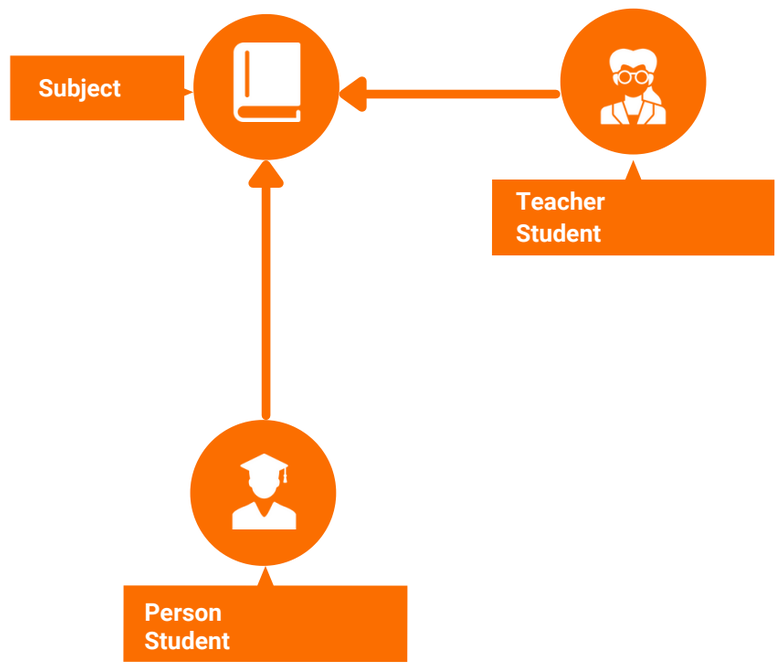


LABELS

Labels are used to shape the domain by grouping nodes into sets or categories. Nodes with the same label belong to the same set. This way of grouping nodes together simplifies database operations significantly. We no longer need to select the whole graph but only the set of nodes we are interested in. Nodes can also have multiple labels attached to them.

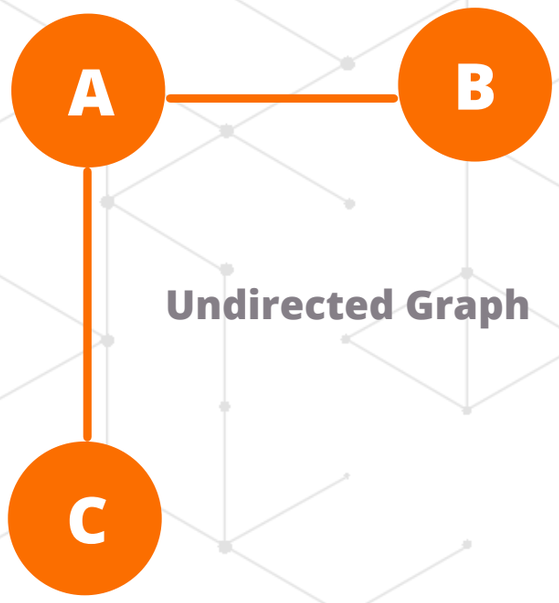
Just as nodes can be easily identified as nouns in the domain description, you can identify labels by generic nouns or groups of people, places, or things.

The node in the example below demonstrates how an entity can belong to multiple groups. A university student can at the same time have the label Person and Student.

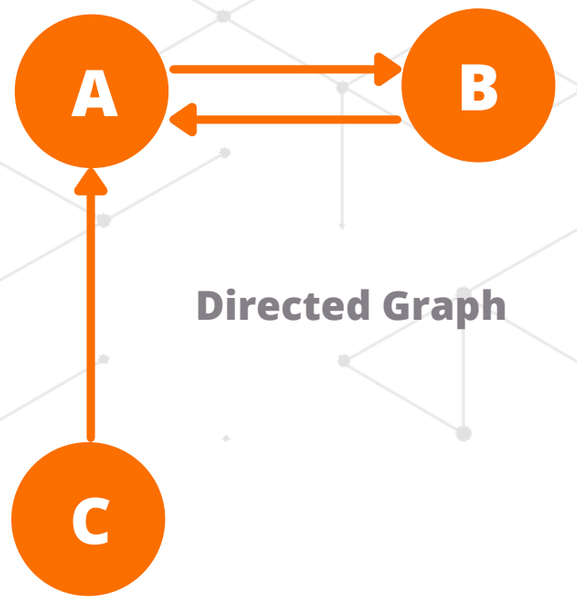


RELATIONSHIPS

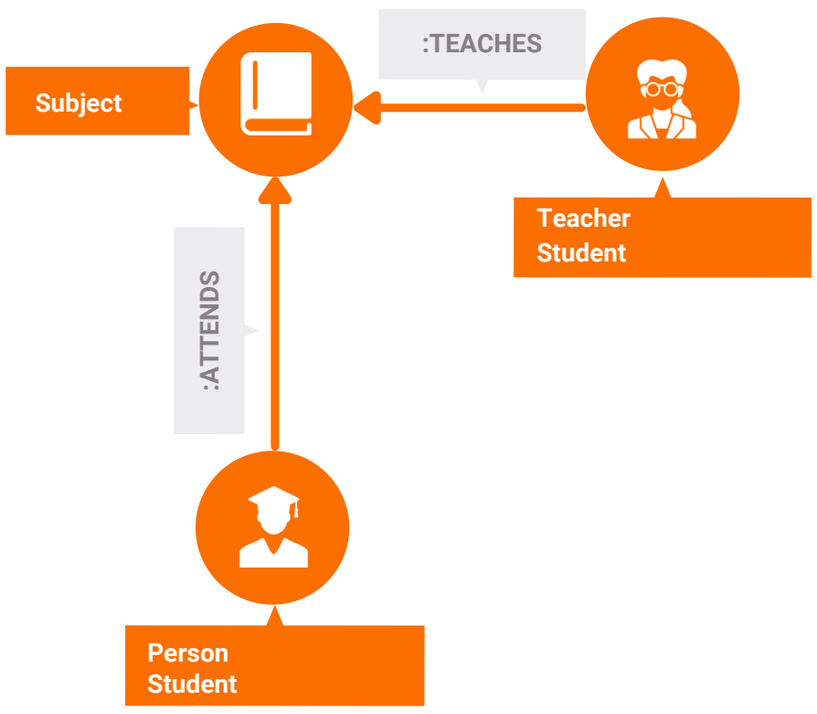
Relationships (or edges) are the lines that connect nodes to each other and represent a defined connection between them. Every relationship has a source node and a target node, in other words, there can't be any loose ends.



If the direction of the relationship is not specified as shown in the graph below, this is called an **undirected graph**. The relationship from node A to B is therefore identical to the relationship from B to A. This approach can be used for situations where the direction of relationships is not important to solving the problem.



If relationship direction is important to the problem you are trying to solve, a **directed graph (DiGraph)** is what you will need. Unlike undirected graphs, in directed graphs, the orientation of relationships is specified. This means that you have to respect the direction of the relationships when navigating your graph.



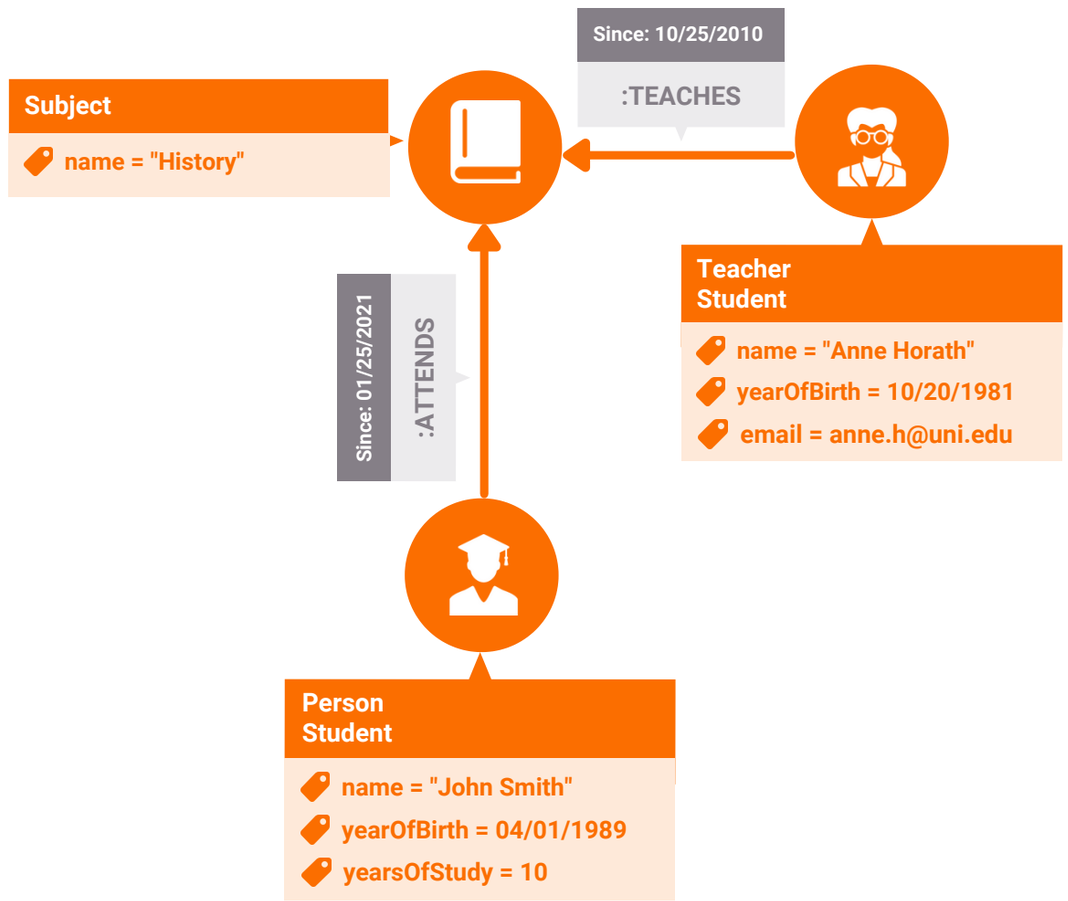
Finally, relationships can also store data in the form of properties, just like nodes. In most cases, relationships store quantitative properties such as weights, costs, distances, ratings, etc. This is called a **weighted graph**, and it could be either directed or undirected.

In our example, the relationship between a Student node and a Subject node could be of the type ATTENDS, while the relationship between Professor and Subject is represented by the type TEACHES.

PROPERTIES

Properties are key-value pairs of data stored on nodes or on relationships. They allow you to store relevant data about the node or relationship with the entity it describes.

Properties support most standard data types like integers, strings, booleans, and many more.



The flexibility and simplicity of properties allow users to easily review the data structure and update it according to their needs. Properties are also very easy to spot. One common way would be asking yourself questions about the nodes and relationships in your model. What information will you need in the future when working with the graph?

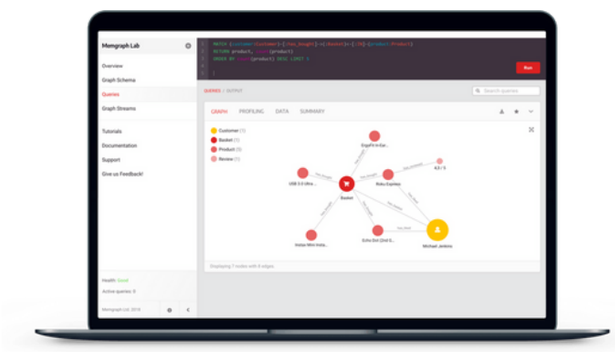
GRAPH ANALYTICS – A WHOLE NEW WORLD OF INSIGHTS

Traditional data analytics are based on statistics and have been used by modern governments, and institutions for decades for a variety of purposes. First, data would be collected and aggregated (e.g. through a census), and then statistical analysis (e.g. regression) would be performed to discover useful information (e.g. where to build the next hospital based on population growth and age by city).

This approach is perfect for analyzing data where relationships between individual data points are not important. However, you are now operating in an ever more connected world where the understanding of complex interdependencies & patterns is what drives many decision-making processes. This is where graph analytics comes into play.

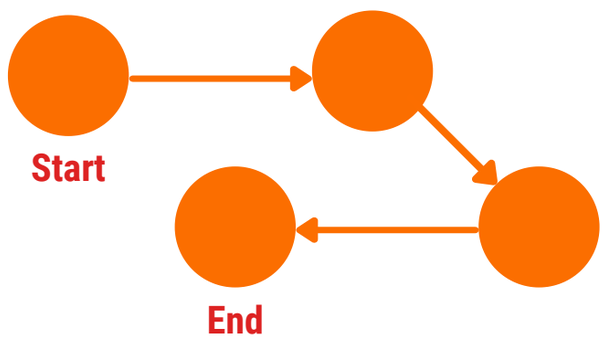
In contrast to statistics-based traditional data analytics, graph analytics (aka network analytics) use different methods to explore connected data leveraging relationships and network structures to extract insights. The main graph analytics methods are graph visualization, pattern-based queries, and graph algorithms.

GRAPH VISUALIZATION



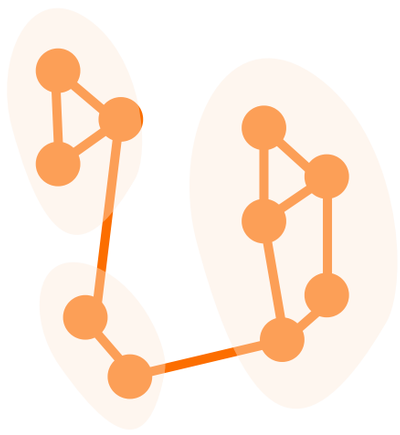
Graph visualization allows you to get a visual representation of your graph data to reveal structures that may be present to help you better understand and analyze the underlying data.

PATTERN-BASED QUERIES

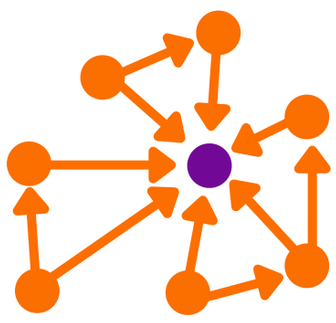


These are the most basic level when it comes to graph analysis, and are usually used for local data analysis. They enable you to find and return an explicit pattern within your graph.

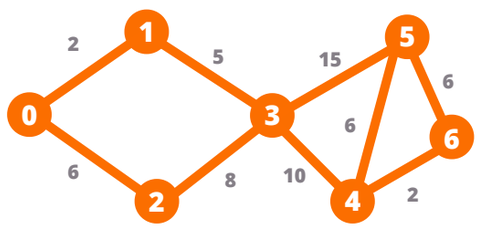
GRAPH ALGORITHMS



Community Detection



Centrality



Path Finding & Search

Graph Algorithms are what really sets graph analytics apart. They have been developed over decades of research to specifically operate on connected data and surface information that is nearly impossible to achieve using traditional analytics methods. They are usually used to run a computation across a large part or the whole graph in an iterative manner and both qualitative and quantitative insights about complex dynamic systems.

GRAPH VISUALIZATION

Graph visualization is a great approach to enable analysts and other non-technical users to intuitively and quickly identify hidden trends or patterns. This can accelerate various investigation activities and help them make decisions quicker.

Graph Visualizations are effective because they're:

- **Intuitive** – They present the information in a way that is very easy for our brains to quickly process and understand.
- **Flexible** – They can be used to abstract and represent a variety of domains with minimal effort.
- **Insightful** – They help the user quickly explore data relationships and gain a deep contextual understanding of that data.

If you are looking to get started with graph visualization you can find a number of powerful Open source tools and libraries that offer great flexibility and features.

Alternatively, you can look at some off-the-shelf graph visualization applications that will allow you to get started quickly and explore your graph data without any programming. These tend to be standalone applications, and are quite pricey.

GRAPH QUERIES

Graph queries are often used to declaratively query a specific part of the graph (subgraph). They have a known starting point and will either find certain nodes/relationships, perform some computation, or update (change/add/delete) information within the graph.

GRAPH ALGORITHMS

Algorithms take graph analytics to a whole new level. Unlike graph queries, graph algorithms are mostly applied to the entire (global) graph structure. This means that they can leverage the overall graph structure and relationships to uncover insights, optimize complex network structures, and predict behaviors.

Graph algorithms have been purposely developed to analyze highly connected data, and therefore offer an easy, elegant, and efficient way to understand and extract valuable insights from your graph data. There are currently hundreds of graph algorithms that are being applied across every major field from Cellular Biology to Satellite Internet Network Optimization and everything in between.

Over the past few years, graph algorithms have been particularly effective in **enhancing machine learning models** by improving their predictive capabilities and accuracy.

Although there are hundreds of graph algorithms that can be categorized into dozens of categories, the most commonly used algorithms belong to three categories:

- **Graph Search –** Graph search algorithms such as Breadth-First Search (BFS) and Depth First Search (DFS) are used to explore paths through the graph to either find a specific node or for general discovery. Note that these algorithms don't necessarily return the most optimal path.
- **Path Finding –** Similar to Graph search, these algorithms are also used to find a path from point A to point B. However, the major difference is that Path Finding algorithms like the Shortest Path algorithm, are used to find optimal paths.
- **Centrality Algorithms –** Centrality algorithms such as PageRank, are used to understand the importance of nodes, and their influence on the network.
- **Community Detection –** Community detection algorithms such as the Girvan-Newman algorithm, help identify similar nodes and group them into communities. These can be used to predict behavior and preferences amongst members of a community.

Unlike graph queries which can run in near real-time, graph algorithms tend to be much more computationally intensive and take minutes or even hours to run depending on the size of the graph. Over the years, many graph processing frameworks such as NetworkX have been developed to efficiently run graph algorithms on static data. However, this is now changing with the introduction of graph databases and graph data streaming platforms which are bringing the power of graph algorithms to real-time applications.

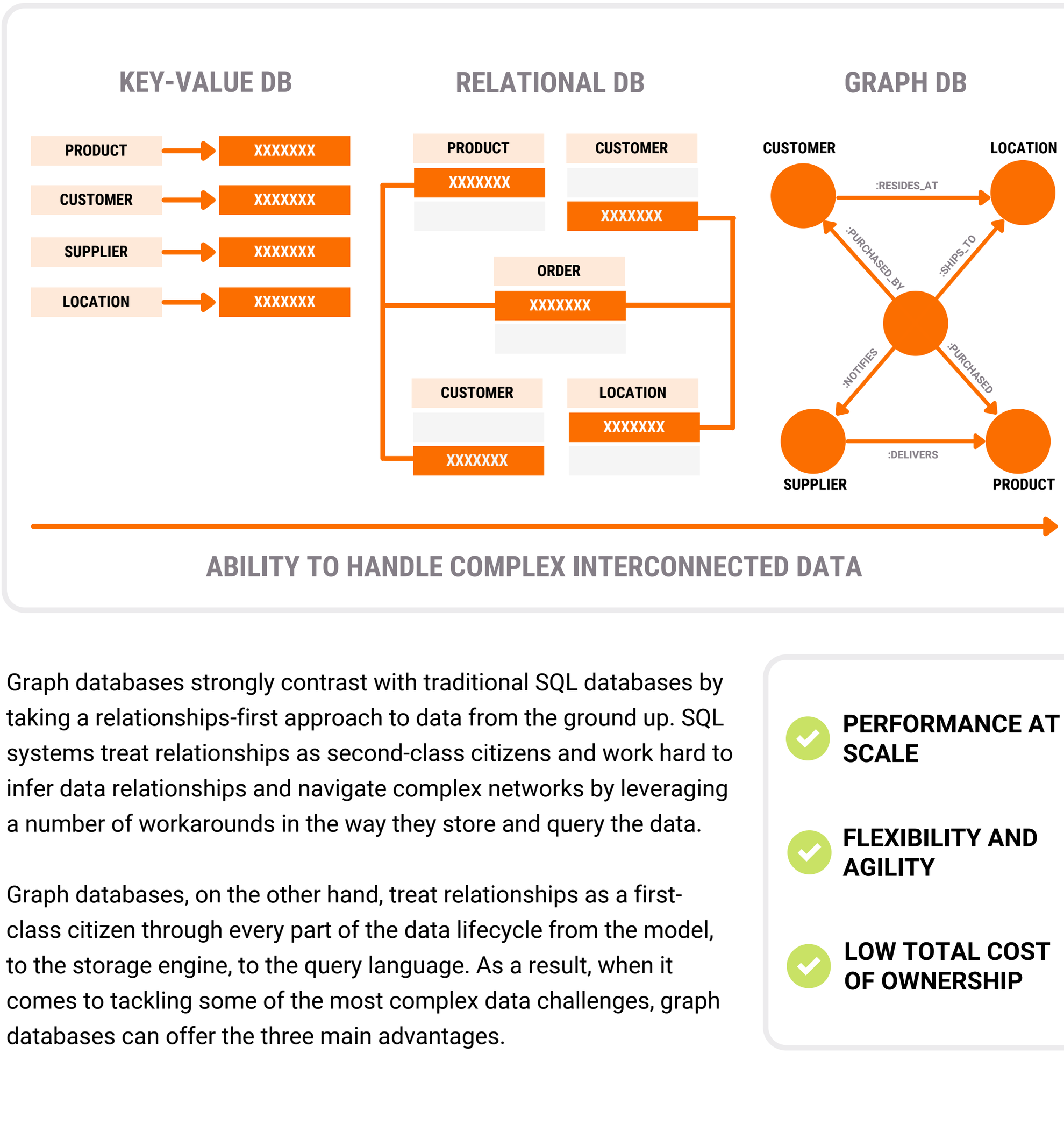
In the next section, you learn about graph databases and graph data streaming platforms. We will explore how they work, what are their advantages and disadvantages, and where they can be applied in real-world applications.

GRAPH DATABASES – PRODUCTIONIZING GRAPH ANALYTICS

Over the past few decades, SQL (Relational Databases) and NoSQL databases have been developed to accelerate, simplify, scale, and eventually productionize traditional analytics. Graph databases were invented to do the same for graph analytics.

Simply put, a graph database is a data management system specifically engineered and optimized to store and analyze graph data. As a result, they offer a highly efficient, flexible, and overall elegant way to run graph analytics.

Although modeling, storing and analyzing graph data is possible using a variety of tools, including traditional databases, it will likely be a laborious, slow, and frustrating processing for your development team resulting in sub-optimal results and serious limitations. The reason behind this is very simple, SQL and other NoSQL databases were not built to deal with highly complex interconnected data.



Graph databases strongly contrast with traditional SQL databases by taking a relationships-first approach to data from the ground up. SQL systems treat relationships as second-class citizens and work hard to infer data relationships and navigate complex networks by leveraging a number of workarounds in the way they store and query the data.

Graph databases, on the other hand, treat relationships as a first-class citizen through every part of the data lifecycle from the model, to the storage engine, to the query language. As a result, when it comes to tackling some of the most complex data challenges, graph databases can offer the three main advantages.

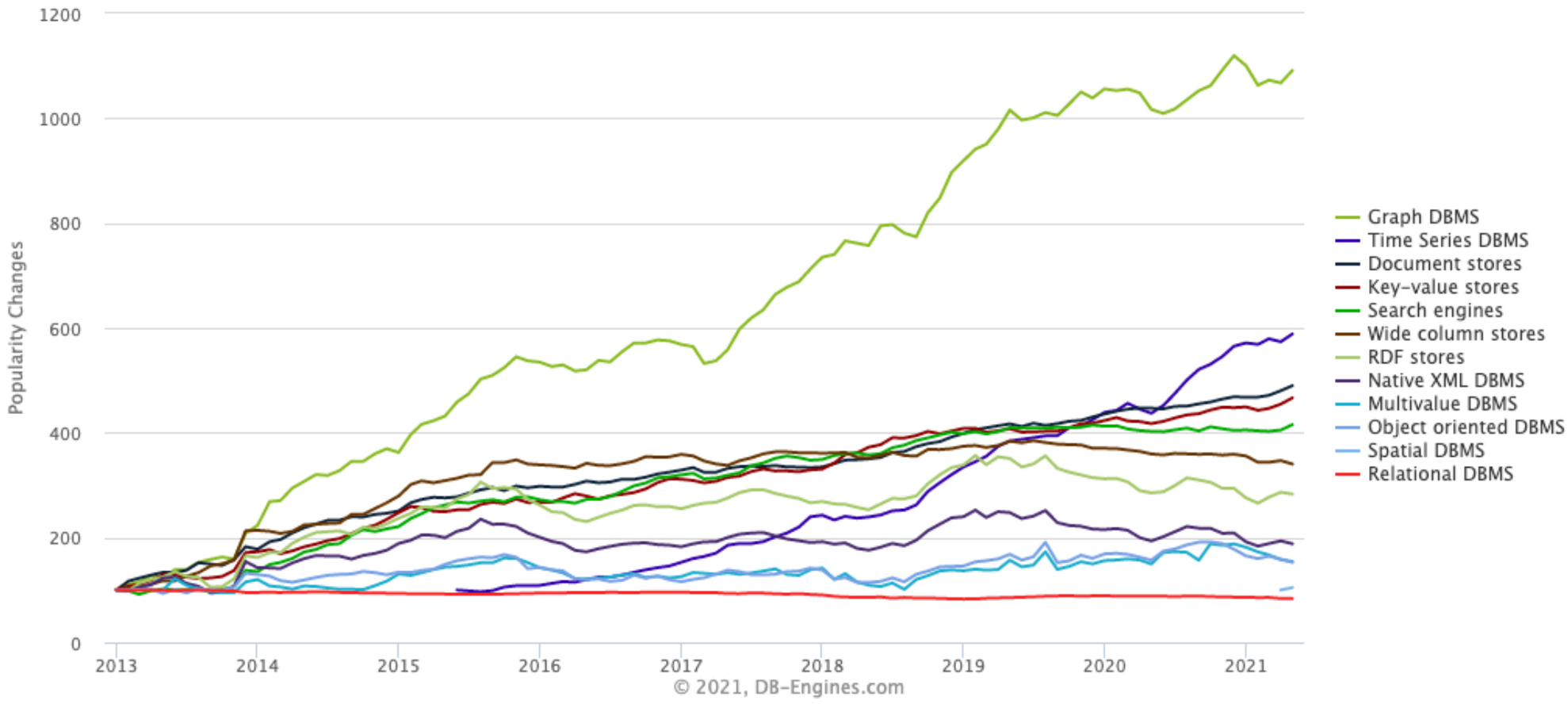
- **PERFORMANCE AT SCALE**
- **FLEXIBILITY AND AGILITY**
- **LOW TOTAL COST OF OWNERSHIP**

Performance at scale – One of the main drawbacks of relational databases when dealing with interconnected data is the severe degradation in performance as the number and depth of relationships increase. This means that with the right optimization, a SQL database can perform acceptably for small datasets on queries with a depth of 2 or even 3 hopes, but it will most definitely fail as your data scales. In contrast, graph databases are designed to maintain predictable performance for queries with 3+ hopes on datasets with millions and even billions of relationships.

Flexibility and agility – With a graph database, your team can easily model complex business scenarios in an intuitive and easy manner due to the fact that the graph data model maps very well to the way we think about real-world problems. A modeling process that could take days in SQL could be done in a few hours. But it doesn't stop there. The graph data model is also very flexible and can be easily changed at any time. This means that unlike with the relational model, your team doesn't have to spend hours to get the model right ahead of time, and then spend a long time adjusting it when the business use-case changes.

Low total cost of ownership – When using a database that was not built to store and query highly interconnected data, your team will need to put in place several workarounds and redundancies in an effort to get the job done. This might lead to the need to combine different systems which could be expensive to maintain or use much more hardware to improve performance and scalability. With a graph database, you will avoid operational complexity and keep the total cost of ownership low and predictable.

It's for these reasons that graph databases have been the fastest growing database category for the past 7 years. According to Gartner, By 2023, graph technologies will facilitate rapid contextualization for decision making in 30% of organizations worldwide



GRAPH ANALYTICS USE CASES

Given that businesses are operating in an ever more connected world where the understanding of complex relationships and interdependencies between different data points is crucial to many decision-making processes, graph analytics have found their way into every major industry. In this section, we will explore real-world graph use cases.

SOCIAL NETWORK ANALYSIS

Some of the main tasks in social network analysis include the detection and identification of influencers and decision-makers. Ensuring the continued engagement of these influencers can lead to a better overall engagement of the whole network or specific communities that have been formed around such individuals.

This kind of information can greatly benefit the sales and marketing processes which heavily rely on community-related information and the general growth of the social network.

Social network analysis can also help to get insights that would otherwise be counterintuitive. Graph algorithms aren't under the influence of human bias and can therefore derive much clearer conclusions from the data. Common usages in social network analysis include:

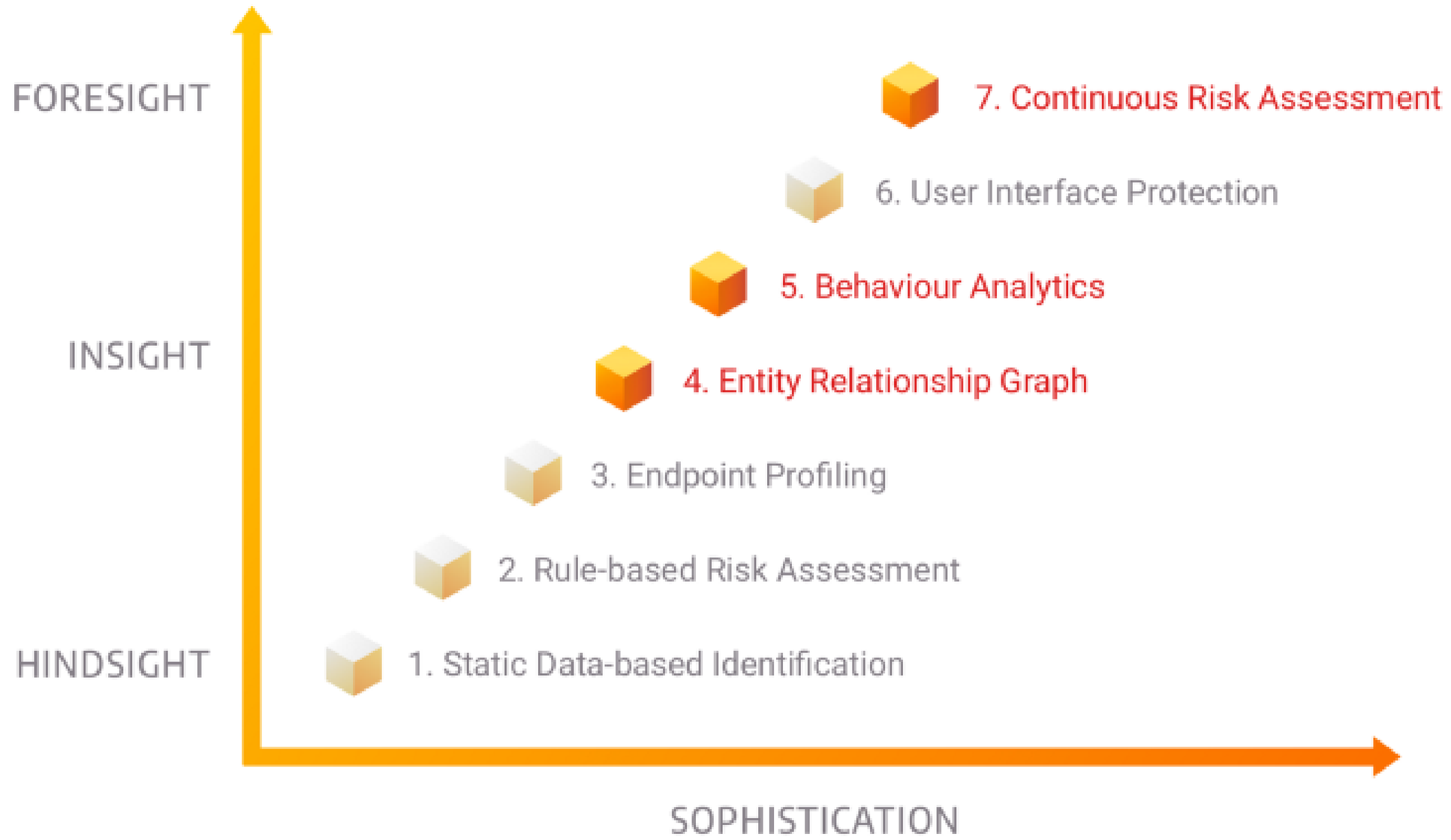
- Identifying communities.
- Identifying users with the most influence.
- Finding connections between specific entities.
- Identifying malicious bots.

FRAUD DETECTION IN FINANCIAL SERVICES

Financial organizations very often have critical vulnerabilities due to their size, complex structure, and the number of services they offer. As a result, they are often a target for fraudsters and cybercriminals. According to a Juniper Research study, merchants and financial services organizations will spend $9.3 billion annually on fraud detection and prevention by 2022. Advanced graph analytics provides deeper insights, complementing Business Intelligence, and helps organizations preempt and prevent potential fraud while protecting customers.

Fraud detection involves identifying fraudulent transactions and bad actors in financial networks. It can be used to search for already committed crimes or to analyze incoming transactions and make real-time decisions concerning their legitimacy.

According to a recent Gartner report, traditional rule-based and siloed fraud detection systems need to be quickly updated to adapt to more sophisticated and complex fraud attacks. In their report, they highlight seven key capabilities as best practices in financial fraud systems. These range from the most basic security measures, such as static data-based identification, to the more sophisticated ones, such as behavior analytics and continuous risk assessment.



In contrast to the one-dimensional traditional rule-based systems, graph databases allow for the aggregation of multiple data silos to integrate all customer interaction channels and easily map complex relationships between data points. This unlocks the capability for fine-grained, sophisticated, and real-time entity relationships and behavioral analytics as well as for a 360-degree continuous risk assessment.

Adopting a holistic approach to fraud prevention, by combining the unique capabilities of graph databases with existing rule-based approaches, has the potential to dramatically improve the accuracy, cost, and efficiency of fraud detection systems. According to Forbes, a relationship approach to fraud detection using graph databases reduces false positives, improves false-negative detection, eases investigations, and reduces overall fraud investigation costs.

TRANSPORTATION NETWORK ANALYSIS & OPTIMIZATION

A transportation network usually implies the transport of people and goods from one location to another by traversing a transportation network. Such networks include street networks, railroad networks, river networks, utility networks, and pipeline networks. These types of networks are comprised of linear features which can be viewed as edges and points of intersection between them which are essentially nodes. In such terms, a transportation network quickly turns into a graph that can be traversed and analyzed using standard graph-based methods.

Many transportation problems can be addressed through a network. A few notable examples include:

- Identifying the fastest and/or shortest path between two points in the network.
- Determining the most efficient way for a delivery vehicle to visit a series of stops (also known as the traveling salesman problem).
- Defining the service area around a given location.
- Identifying an optimal store location to supply the largest number of potential customers effectively.

SUPPLY CHAIN MANAGEMENT

Globalization has without a doubt changed the production network. With the world at our fingertips, we are able to deliver top-notch products more rapidly and affordably than ever before. Globalization has also made supply chains more complex, in turn making supply chain management a more challenging task.

In an ideal world, every supply chain would be straightforward: one provider supplies another in an easy-to-understand progression to the final finished good. However, with globalization, brands can source from different providers from everywhere in the world, making the inventory network somewhat incomprehensible and overconnected.

We need constant innovations to deal with this intricacy and graph databases are well suited to handle such problems. Instead of relational databases, a typical information model which stores rigidly organized information into tables, graph databases collect nodes and the relationships between them into connected structures that are perfect for analyzing.