Resumen 7 y Resumen 8
Estudiante: Celina Madrigal Murillo
Carné: 2020059364

5 Use Cases for Graph Technology

When your data is complex, the solution is in the graph.

Most companies have a data problem. The more complicated the problem, the more likely a graph data platform can solve it.

Here are five of the top use cases for graph technology.

1. Fraud Detection: Banks and insurance companies lose billions every year to fraud. Many companies are swapping time-consuming traditional methods with AI and machine learning.

2. Real-Time Recommendations: To be relevant and prevent a potential customer from clicking to a competitor, recommendations must be contextual and instantaneous

3. Bill of Materials: For vehicles, durable goods, and more, tracking every component and its cost, what equipment the components relate to, and the expected product lifespan/average time to failure into a mass bill of materials (BoM) is a behemoth operation for an organization like the United States Army, which deploys a staggering amount of equipment.

4. Track & Trace: Graph data models enable traceability for a variety of industries and use cases, including routing, logistics, supply chain management, and compliance. Track and trace enables us to find out the status of a product (batch, mail/ parcel, train, ship, container, etc.) wherever it is in the supply chain and to identify and verify its path.

5. Network & IT Ops: Graph databases are a natural fit for network and IT ops; after all, a network is a graph. The graph's connected structure enables network managers to conduct sophisticated impact analyses.

A Brief Introduction to Graph Data Platforms

What Are Graph Databases Good for?

Graph database technology is specifically designed and optimized for highly interconnected datasets to identify patterns and hidden connections.

The most common graph use cases and solutions include:

• **Fraud Detection & Analytics:** Real-time analysis of data relationships is essential to uncovering fraud rings and other sophisticated scams.

• **Artificial Intelligence & Machine Learning:** Artificial intelligence (AI) winners and losers will be decided based on who harnesses context within data for a true competitive advantage.

• **Real-Time Recommendation Engines:** Graph-powered recommendation engines help companies personalize products, content and services by building a contextualized map of offers using both historical and real-time data.

• **Knowledge Graphs:** Graph-based search tools tap into your organization's institutional memory.

• **Network & Database Infrastructure Monitoring:** Graph databases are inherently more suitable than RDBMS for making sense of complex interdependencies central to managing networks, data centers, cybersecurity and IT infrastructure.

• **Master Data Management (MDM):** The schema-optional graph database model allows you to organize and manage your master data with flexibility.

Viewed through a technological lens, graph databases tackle the most harrowing of data problems – ones that often linger at the root of project failures and delays. These include:

• Vastly different views of the data model between business and technology teams, which result in misunderstanding and miscommunication.

• Lack of schema flexibility and adaptability, making it hard to respond to changing business requirements both during a project and after a system has gone live.

• The "JOIN problem" which occurs when queries become so tangled that even powerful databases with massive amounts of hardware resources grind to a halt in their attempts to bring the resulting data together

Traditional Technology Choices Do Not Consider How Data Is Interrelated

Traditional RDBMS technology has a difficult time expressing and revealing how real and virtual entities are related. Columns and rows aren't how data exists in the real world. Rather, data exists as rich objects and the relationships between those different objects.

Collections vs. Connections

SQL & NoSQL Systems Focus on Data Aggregation & Collection

Collection-centric storage designs as implemented by SQL and Not only SQL (NoSQL) databases are designed to efficiently divide and store data.

In SQL's case, the normalization of data into a tabular schema aims to minimize storage of duplicate data objects, types and values.

The original relational databases were designed to minimize storage of duplicative data values because disk space was costly. The RDBMS design achieved this consolidated, normalized goal by linking tables of data via foreign keys to their associated records from other tables.

NoSQL systems like document, wide column and key-value data stores carry those concepts forward (and backward) by simplifying their models in exchange for higher levels of scale and simplicity. By eschewing data relationships and providing simple programmatic APIs, NoSQL systems make it easy for developers and administrators to work with simple data in a way that can easily scale.

Graph Systems Focus on Data Connections

By contrast, graph database technologies focus on how data elements are interrelated and contextualized as connected data.

Connected data is the materialization and harnessing of relationships between data elements, which is modeled as a property graph.

A property graph is a data model designed to express data connectedness as nodes connected via relationships to other nodes, where both nodes and relationships can have properties attached to them.

In the graph model, data relationships are persisted so they can be navigated or traversed along connected paths to gain context. Relationships are both typed and directional.

The context provided by these data connections is essential to identifying friendships, making relevant real-time recommendations, attaching adjacent ideas and detecting fraud by following money trails. Without relationships as first-class data entities, all of these use cases become extremely difficult to execute.

Property Graphs Are Intentionally Simple

• You can draw property graphs on whiteboards and map that design directly into a graph database.

• You can change or update a property graph easily, because its agile design eliminates most of the structural overhead of traditional database schemas.

• You can quickly program property graphs because their query language expresses and follows relationships.

• You can visualize and navigate property graphs efficiently by following the relationships on their paths to context.

• You can rapidly determine data context when property graph queries are executed in hyper-fast native graph platforms built on reliable, scalable database architectures.

Benefits of Graph Databases

• **Simple and natural data modeling:** Graph databases provide flexibility for data modeling, depending on relationship types.

• **Flexibility for evolving data structures:** Graph technology provides flexible schema evolution.

• **Simultaneous support for real-time updates and queries:** A graph data store and its model allow real-time updates on graph data while supporting queries concurrently.

• **Better, faster and more powerful querying and analytics:** Graph data stores provide superior query performance with connected data using native storage and native indexed data structure.

The Return on Connected Data

More Connections, More Value

Connections Unlock the Value of Data

The value of data comes from an organization's ability to understand it in relation to other data. By itself, data offers finite value. When connected, data's value is infinite.

Increasing data's connectedness further increases its value through additional context. This is another demonstration of Metcalf's Law of the Network: network value increases exponentially as you add users or nodes to the network.

Enticed by the promise of valuable business insights, companies have invested in big data technologies. But dumping data in a data lake doesn't preserve nor reveal the relationships between data points. And JOIN tables only materialize a relationship when the query is run; relationship information is not a first-class entity in relational nor other types of databases.

Because relational databases don't persist relationship information in storage or any other stage of their analytic exercises, finding connections requires an enormous amount of extra processing. And persisting these connections over their lifetime is next to impossible.

Data, Data Everywhere, Nor any Drop to Drink

Organizations are surrounded by data but they're limited in their ability to do anything with it.