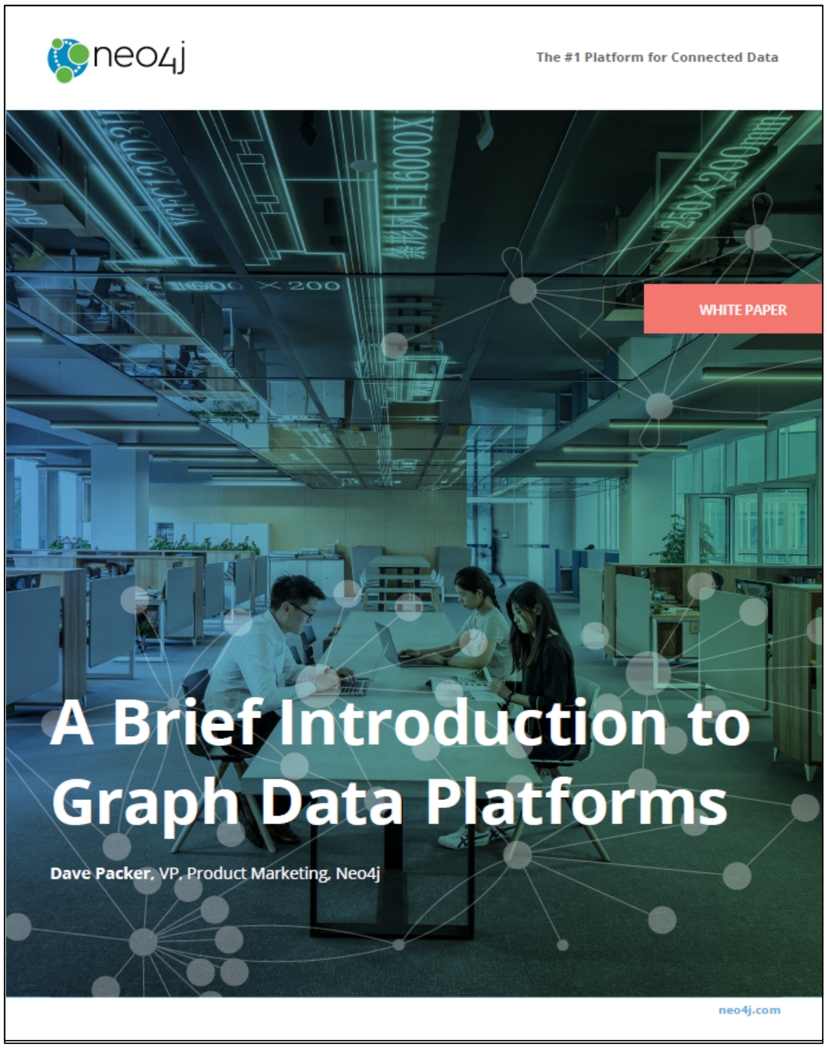
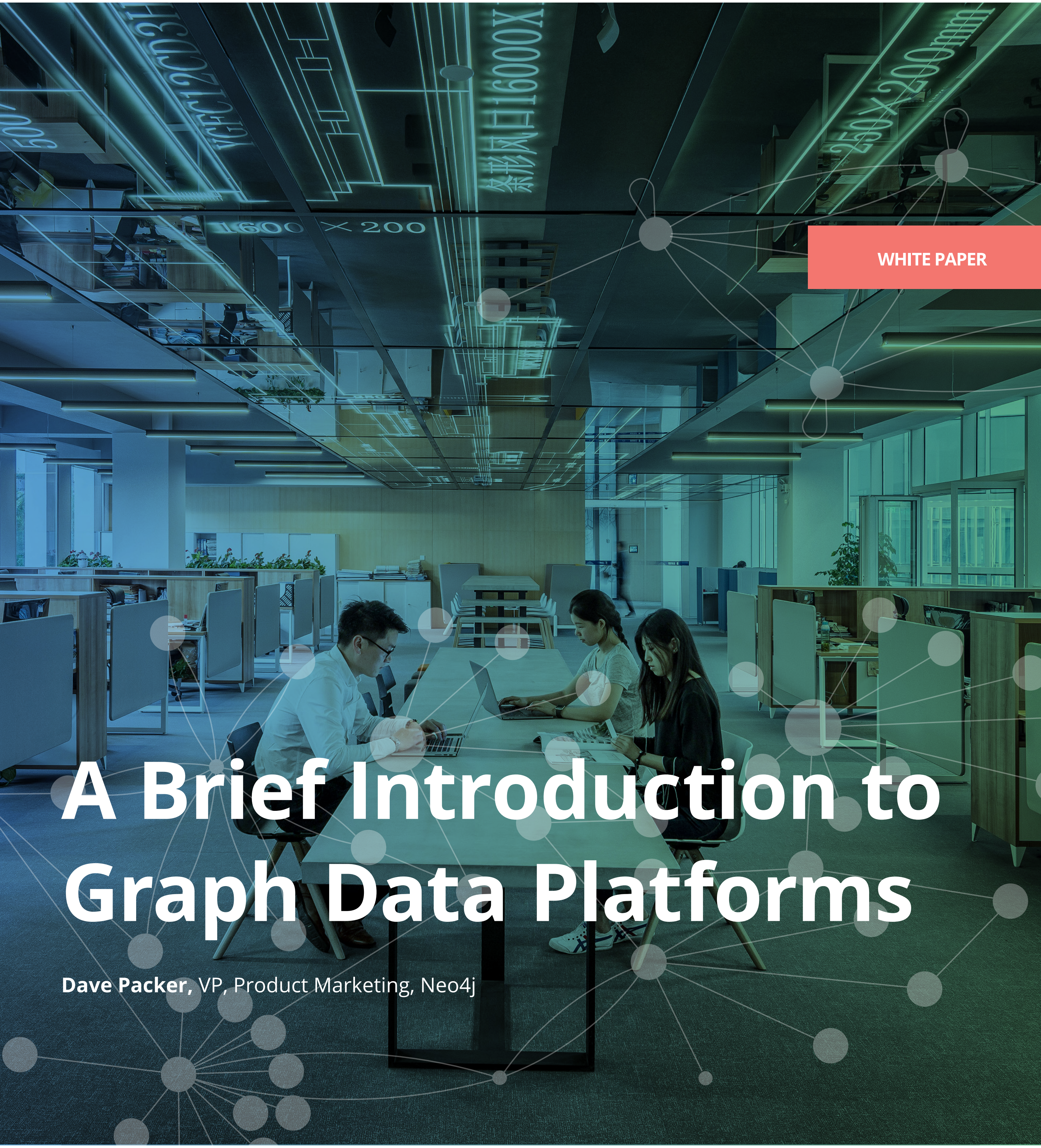
A Brief Introduction to Graph Data Platforms

White Paper



The #1 Platform for Connected Data



WHITE PAPER

A Brief Introduction to Graph Data Platforms

Dave Packer, VP, Product Marketing, Neo4j

White Paper

TABLE OF CONTENTS

A Brief Introduction to Graph Data Platforms

Dave Packer, VP, Product Marketing, Neo4j

Introduction

You may have heard that graph database technology underpins the latest in machine learning and AI. But what are graph data platforms and why are they rapidly becoming the first choice for application development? Can a shift in database technology really make a big difference?

Relational databases dominate the market with their ability to store structured data in tabular models. While relational databases have their place, their rigid structures fail to adapt easily to the complexity of data, its context and its interconnections. As a result, relational databases are simply the wrong tool for many new business challenges. Real-time recommendations rely on connected data; in the relational world, this requires linking many tables. Relational databases strain under that load while users wait impatiently.

In the past decade alone, the data landscape has shifted significantly. Perhaps it's time to consider new ways of tackling the evolving data landscape and how it's managed.

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.

What Are Graph Databases Good for?

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

Graph data stores are intuitive because they mirror the way the human brain thinks and maps associations via neurons (nodes) and synapses (relationships). A graph database efficiently stores and queries connected data in a node-and-relationships format. As a result, graph technology excels at problems where there is no a priori knowledge of path length or shape by using graph storage and infrastructure to find neighboring data efficiently.

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. They are also used for better digital asset management. Moreover, knowledge graphs are the basis for many natural language processing (NLP) and AI solutions.
- **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. It also lets you harness real-time insights and a 360° view of your customers, products and employees.

What these use cases have in common is that their success requires solving complex problems with dynamic and interconnected datasets. To this end, we should reframe the question, "What are graph databases good for?" as a technical one.

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.

With the right choice of technologies, graph databases introduce a new way of looking at data by promising to significantly address all of these issues. They seek to take you beyond

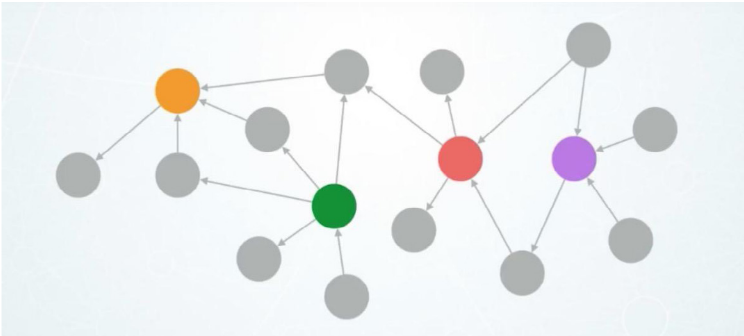
Well-established entities such as NASA, eBay, UBS and many of their peers use graph technology to improve their customer experiences and increase their competitiveness.

handling sheer volumes of relatively simple data and towards revealing the interconnected complexities latent in your data – and deriving bottom-line value from them.

Traditional Technology Choices Do Not Consider How Data Is Interrelated

Today's data and applications require elasticity, agility, speed and interconnectivity. Despite the name, relational databases are not well-suited for modeling and storing today's highly connected and agile datasets. RDBMS demand slow and expensive schema redesigns that hurt agile software development processes and hinder your ability to scale and innovate quickly.

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.



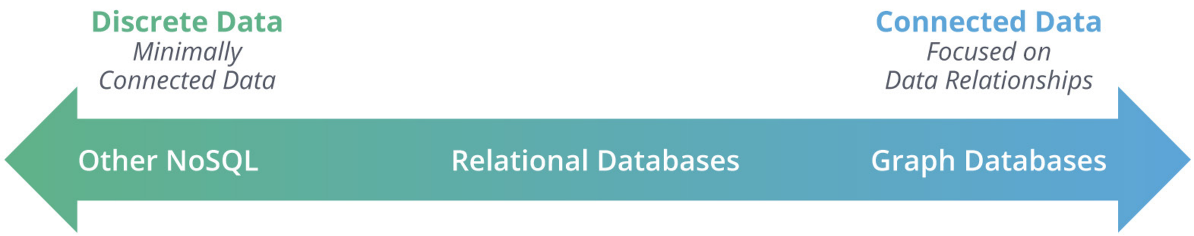
The theory of connected networks (graph theory) has been a mathematical discipline for nearly three centuries, but few technologies have harnessed these theoretical models for the purpose of data storage and analysis.

Early on, a handful of companies have truly tapped into the power of graph technology as the driver of their businesses, including Google, Facebook, LinkedIn and Microsoft as well as upstarts like Airbnb and Uber. Well-established entities such as NASA, eBay, UBS and many of their peers use graph technology to improve their customer experiences and increase their competitiveness.

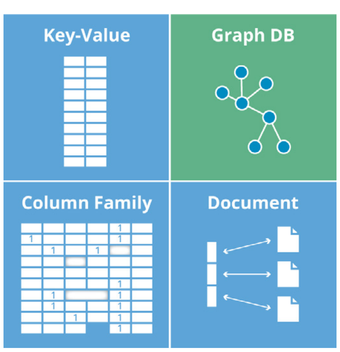
Today, graph-powered applications are used by more than 75% of the Fortune 500, including:

- 7 of the world's top 10 retailers
- 3 of the top 5 aircraft manufacturers
- 8 of the top 10 insurance companies
- All of North America's top 20 banks
- 8 of the top 10 automakers
- 3 of the world's top 5 hotels
- 7 of the top 10 telcos

These successes serve as a strong indicator of graph technology's impact on both innovation and the bottom line.



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.



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.**

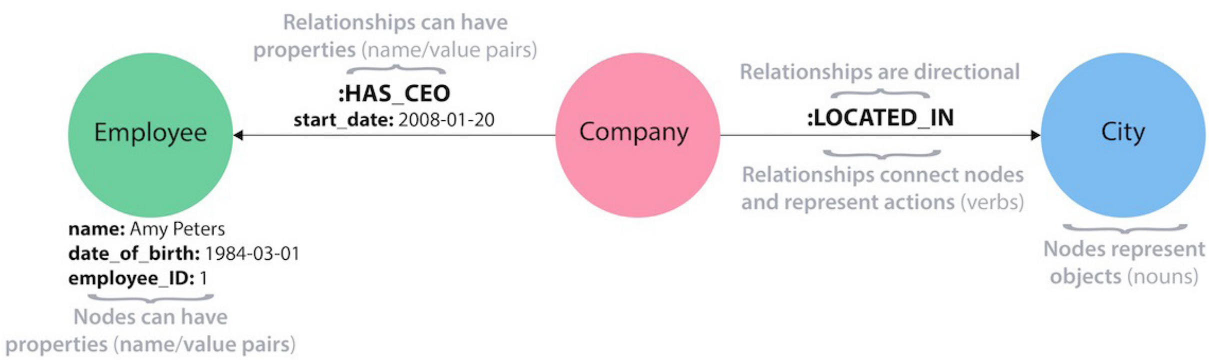
In SQL's case, the normalization of data into a tabular schema aims to minimize storage of duplicate data objects, types and values. These systems were born during the era of scarce physical memory and expensive disk-based storage, designed to avoid managing often-redundant data objects such as physical location addresses for shipping, billing, homes, offices, destinations, businesses, etc. For example, all of these data objects included common, redundant data such as a country and its provinces or states, or telephone area codes and postal codes.

The original relational databases were designed to minimize storage of duplicative data values because disk space was costly. (Ironically, each normalization incurs a cost in relationship storage in the form of JOIN tables, which native graph databases have managed to eliminate through the use of pointers.) The RDBMS design achieved this consolidated, normalized goal by linking tables of data via foreign keys to their associated records from other tables. This is why a relational dataset modeled into a graph often shrinks by an order of magnitude or more, maintaining the full richness of the data without redundant data storage.

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.

A lack of concern about relationships leads to looser data guarantees, plain APIs and straightforward scaling schemes. Data is easily spread out and just as easily retrieved, without the need to maintain the integrity of related data that's written across a distributed storage or a cluster of machines and without needing to concern itself with the performance of distributed JOINs across those machines.

These systems take on the "store and retrieve" problem at scale for simple data, and their architectures reflect this, as does the set of problems they are equipped to address. However, none of these systems focus on interrelated, contextualized data or how that data might be traversed to reveal unobvious relationships, as explained below.



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 (which in turn can be indexed).

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 (which in turn can be indexed). For example, devices on an enterprise network might be modeled as nodes, with properties for their attributes (such as throughput) and relationships pointing to other adjacent devices.

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.

Connected data in property graphs enables you to illustrate and traverse many relationships and find context for your next breakthrough application or analysis.

Benefits of Graph Databases

- **Simple and natural data modeling:** Graph databases provide flexibility for data modeling, depending on relationship types. Since the graph model comes with no inherent rules, graph data stores add as much or as little semantic meaning as the domain requires. This occurs without any constraints like normalization or restructuring of the data using denormalization.
- **Flexibility for evolving data structures:** Graph technology provides flexible schema evolution. In a constantly changing data environment, you need the option to add or drop data entities or relationships as well as extend or modify your data model. Graph databases allow for evolving data structures that match today's agile development environments.
- **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.

Connected data in property graphs enables you to illustrate and traverse many relationships and find context for your next breakthrough application or analysis.

Conclusion

Today, businesses need to bring together data from many different systems, which is driving a huge shift from on-premises to cloud as well as investments in new data integration models. Traditional rigid systems aren't nimble enough to keep up with the pace of change.

Supporting complex data environments often requires real-time data analysis to make split-second decisions to optimize the business.

Making the most of your data requires the ability to make connections across all of it. As the fastest growing database technology in the past decade, graph databases provide a much more scalable, secure and flexible platform than traditional databases for enterprises to deploy their use cases – from real-time recommendation engines to fraud detection to knowledge graphs to AI and machine learning.

What will your first use case be?

Find out more about Neo4j. Sign up for a 30-minute demo with a live Q&A.

Thank you for downloading this Neo4j Introduction to Graph Data Platforms datasheet! Carahsoft serves as Neo4j's Master Government Aggregator®.

To learn how to take the next step toward acquiring Neo4j's solutions, please check out the following resources and information:

For additional resources:
carah.io/Neo4jResources

For upcoming events:
carah.io/Neo4jEvents

For additional Neo4j solutions:
carah.io/Neo4jSolutions

To purchase, check out the contract vehicles available for procurement:
carah.io/Neo4jContracts

To set up a meeting or for more information:
Neo4j@carahsoft.com
877-742-8468

To request a quote:
carah.io/RequestQuote