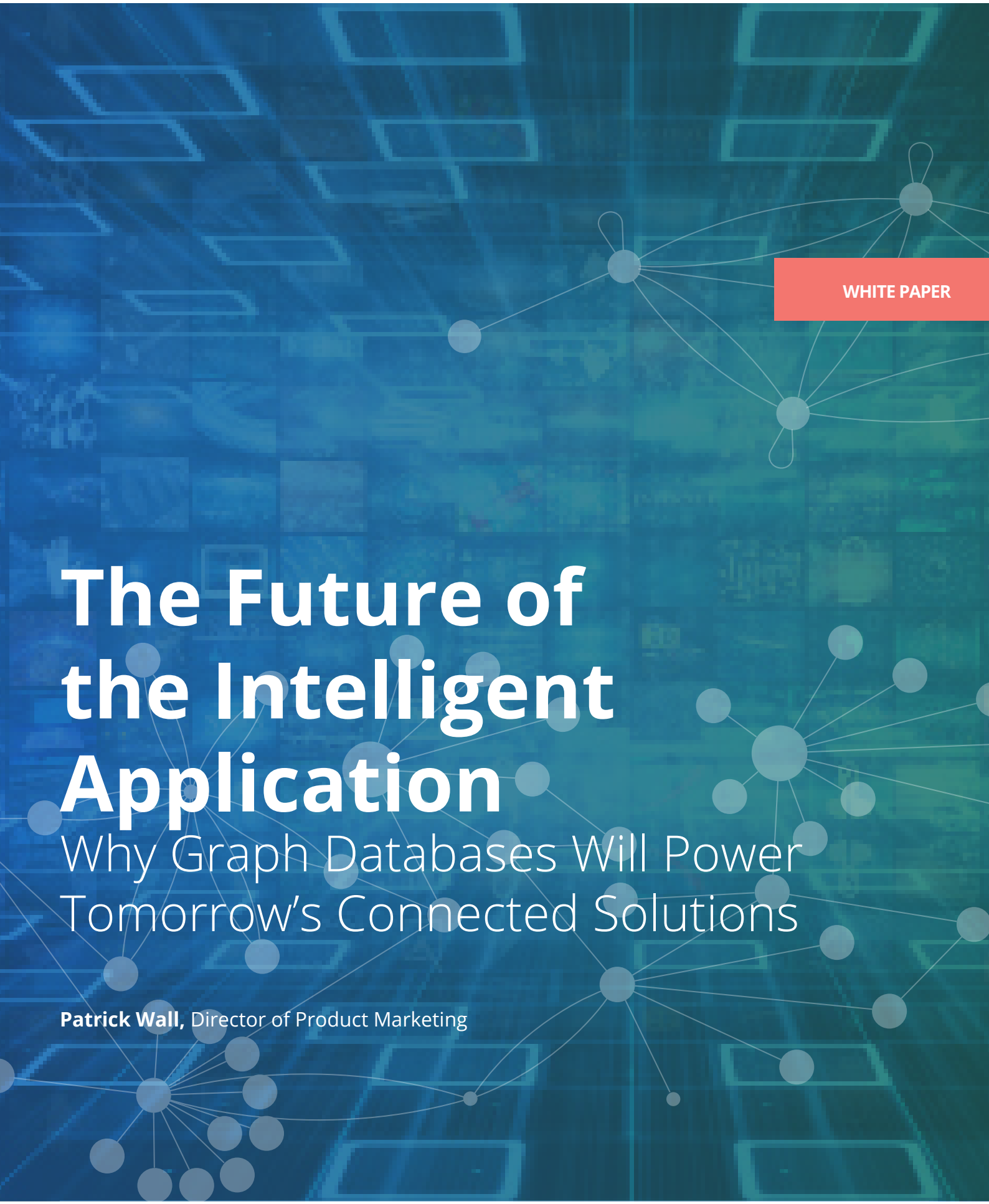


WHITE PAPER

The Future of the Intelligent Application

Why Graph Databases Will Power Tomorrow's Connected Solutions

Patrick Wall, Director of Product Marketing

White Paper

TABLE OF CONTENTS

The Future of the Intelligent Application

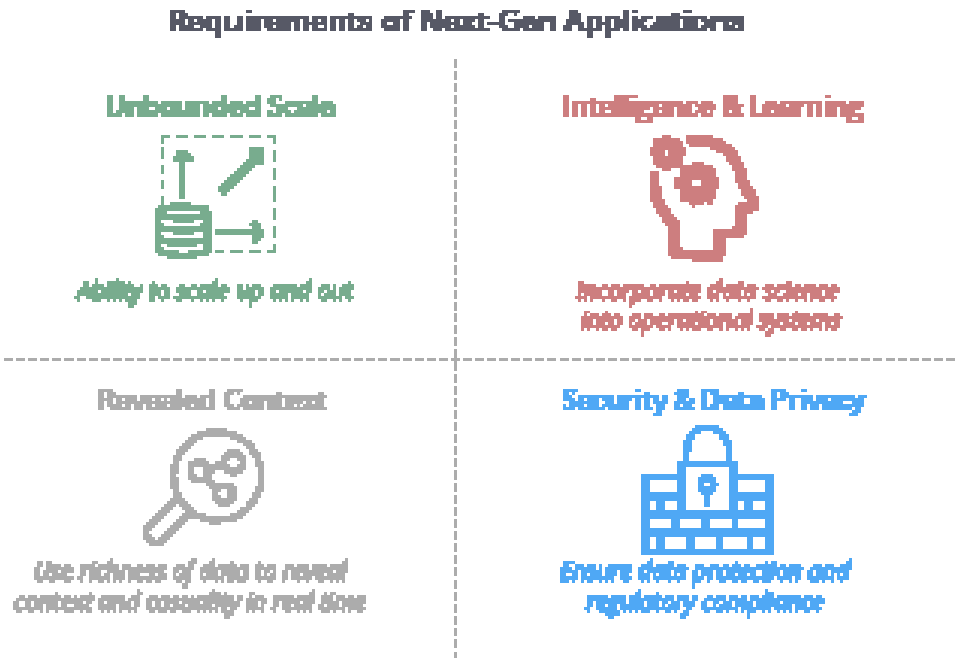
Why Graph Databases Will Power Tomorrow's Connected Solutions

Patrick Wall, Director of Product Marketing

Introduction

Data, AI, intelligent applications. They're no longer separate topics, or even separate conferences. O'Reilly announced that it is merging its data and AI conferences, saying, "Data feeds AI; AI makes sense of data." Further, both power applications. In 2016, Ben Lorica – then program chair of the O'Reilly Strata Data and AI Conferences – predicted that soon "some features of AI will be incorporated into every application that we touch, and we won't be able to do anything without touching an application."

Modern applications come with new requirements. Such applications incorporate intelligence and learning. They require full context to support smart decision-making in real time. These applications must be able to scale without limits to meet unexpected demand. Furthermore, next-generation applications demand straightforward, robust security and flexible architecture to comply with ever-increasing regulations .



Emerging Requirements for Intelligent Applications

Unlimited scalability: Data grows relentlessly and performant applications can't be constrained by data volumes. Applications need to scale up – and out – to handle higher volumes while maintaining performance across a growing diversity of on-premises, hybrid and cloud architectures.

Dynamic, revealed context: Applications increasingly need databases that adapt to the myriad complexities, dynamics and unpredictability of real-world data. Effective applications use the richness of all available data to reveal context and causality in real time.

Security and data privacy: As regulations continue to evolve, individuals and governments are more conscious than ever about how and where consumer and citizen data is used. Developers need to build applications that meet these needs securely and rapidly.

Intelligence and learning: Operational applications are increasingly components of complex systems that incorporate machine learning and artificial intelligence. For actionable AI, applications need to bridge data science across operational systems and leverage context in real time.

This white paper prepares software architects and developers for the challenges of creating intelligent applications.

Why Graph Databases Will Power Intelligent Applications

Just as computing has changed significantly over time, so have databases evolved to support next-generation software applications.

When relational databases were designed, a primary usage pattern was to store and retrieve information that mirrored paper forms. Tables are great for this. The database infrastructures that evolved to support the tabular model grouped data together into fixed tabular buckets. This was a good match for paper forms, and suited the slower pace of data (and business) at the time. It also optimized performance for reading from slow disk rather than fast memory, given the high cost of memory at the time.

Considerable software engineering has gone into generalizing relational databases, and they continue to serve as the dominant database model for information systems.

But relational databases have inherent limitations when it comes to connecting data. JOINing relational tables negatively impacts performance. Despite their name, relational databases are not designed to understand and reveal real-world relationships. The need to define what the data will look like before it shows up is another challenge, particularly as the speed and diversity of data sources accelerates.

Today's applications connect people, companies, assets, devices and even genomes. Mathematically speaking, these connections form a graph. Graph data structures provide a flexible, scalable foundation for these connection-oriented applications.

Graphs are composed of nodes and relationships (or vertices and edges, in traditional mathematics terminology). Nodes are individual data points connected by relationships. Properties can be associated with both nodes and relationships.

Graph data models are intuitive, easily sketched on a white board to share with both business and technical stakeholders. But a native graph database means that the graph, its relationships and their properties do not simply represent a nice model of the data; they represent the way the data is stored.

Neo4j is a native graph database; thus, its internal structures to store and retrieve data are optimized to understand and reveal data relationships naturally, without complex abstractions.

"The modern application development process puts a premium on velocity, which is why ease of use and flexibility for developers have become as critical as performance for database platforms. One of the ways this is achieved is by avoiding the need for cumbersome data abstractions to translate between business needs and relational schemas. Graph databases are a canonical example of this, and Neo4j remains one of the pioneers of the category committed to bringing the benefits of graphs to a wide variety of customer types and use cases."

– Stephen O'Grady, Principal Analyst, RedMonk

The Future of the Intelligent Application

Relational databases are typically optimized for either transactions or analysis. A fully transactional graph database should support the strict requirements of ACID transactions while also flexibly powering applications at scale. It also enables queries that, in the relational world, might take hours to run because of their complexity, but in a native graph context might run in fractions of a second. Graph databases like Neo4j further open the door to the emerging field of graph data science, with graph algorithms as a cornerstone. Graph algorithms enable individual pieces of data to be understood in the context of emergent characteristics of the network, providing a level of intelligence not otherwise available.

The reason why Neo4j supports both graph transactions and graph data science so well is that it has a native graph format specifically for graph storage and local queries and a format specifically for graph analytics. Neo4j automates data transformations between graph storage and graph analytics computation – so you benefit from maximum compute performance for analytics and native graph storage for persistence. This means that there's no duplication or ETL required to run analytics on existing graph data, and that your applications capitalize on a unique virtuous cycle between transactions and data science, within a single database.

Transaction integrity is just as important in graph databases as relational databases. And it's more important in a graph database than in NoSQL databases, which are fundamentally architected to handle data but not relationships between individual points of data. Break one relationship and you have corrupted the database with a relationship that points nowhere. There are countless ways to go wrong in connecting, copying or updating portions of a graph dataset.

In a graph database where everything is connected, ACID transactions are paramount. An atomic transaction must be fully executed or connections could be broken, which results in the graph storing the wrong information. And when a graph database at scale is formed by billions – if not trillions – of connections, even a single broken relationship spells disaster.

What Is Graph Data Science?

Graph data science unlocks insights from connected data. Neo4j provides the first enterprise-grade graph data science platform to harness the natural power of network structures to infer behavior. The Neo4j Graph Data Science Library includes production-ready, validated graph algorithms to help answer previously intractable questions using the connections in your data. These algorithms fall into five categories.

Beyond exploration by domain experts using ad hoc queries, data scientists use graph algorithms for more global analysis and understanding the structure of relationships. Common uses include deep path analytics, influencer identification, community and neighbors detection, fraud detection, link prediction, structural pattern matching and disambiguation.

Furthermore, data scientists use graph feature engineering to improve their predictive accuracy. For example, graph algorithms can produce an influence score for nodes (e.g., PageRank) or label nodes in tightly knit communities and then those scores and labels can be extracted for use in machine learning models. Adding these types of connected features to existing machine learning models improves their accuracy without disrupting the current machine learning pipeline.

Critical Database Capabilities for Intelligent Applications

Built on the foundation of a native graph database, Neo4j offers critical database capabilities for today's intelligent applications in three areas:

- Scalability
- Security
- Agility

Scalability

Neo4j scales with your data and your business needs, minimizing cost and hardware while maximizing performance across connected datasets.

Relational databases scale, but with limited connections across tables. As a result, JOINs severely degrade performance.

Neo4j invented the graph database. Data integrity is not only fundamental to the graph model, but to the customers who rely on a graph database in mission-critical applications. Enabling unbounded vertical and horizontal scalability required that the graph data would never be corrupted so that enterprise organizations could scale safely and with confidence.

Sharding

A large graph database may have trillions of nodes. Consider LinkedIn, for example. Picture all of the people in your professional network as a single, coherent graph. The physical storage of such a graph is divided, or sharded, across many servers or clusters, despite the fact that it's still a single graph dataset.

While common in the relational database world, sharding is relatively new to graph databases. Sharding is the division of a single logical database into as many physical databases as required.

The ability to divide the graph database across many servers is key to scalability as well as the ability to support use cases such as compliance with data privacy regulations. For example, regulations such as GDPR stipulate that data for a particular country's citizens must be physically stored in that country. The portion of the graph for those geographic customers can thus be sharded to fulfill those requirements.

Federated Graphs

While sharding divides graphs, federated graphs bring multiple graphs together, supporting queries across graph databases that may have different logical structures.

Queries against massive graphs that share a schema

A massive graph sharded across many physical servers can be thought of as a single logical graph. Federated graphs encompass multiple Neo4j servers, creating a logical view of the servers and the databases hosted by those servers.

A federated graph creates a virtual graph that is – from the application or user perspective – one enormous graph database. When a client application or a user runs a query against such a graph, that query interacts with potentially tens or hundreds of machines and databases to store or retrieve data.

"At albelli we regularly deal with petabytes of data, and we are most excited about the new scalability features in Neo4j 4.0. The ability to horizontally scale with the new sharding and federation features, alongside Neo4j's optimal scale-up architecture, will enable us to grow our graph database without barriers. Neo4j factored our requirements into their latest release – a mark of a great vendor and a graph database designed for the future."

– Josh Marcus, Chief Technology Officer, albelli

"The unmatched scalability of Neo4j 4.0 lends itself to emerging AI and machine learning use cases, which require graphs to scale reliably across massive datasets to give learning applications context and make AI more explainable. In addition, the ability to apply fine-grained security on nodes and relationships in a flexible way will have immediate impact across a broad range of use cases."

– Michal Bachman, CEO, GraphAware

Queries against disjointed graphs

Neo4j not only handles queries against large, sharded graphs that share the same data model; it also supports queries across disjointed graphs. In effect, this makes all the data in an enterprise's graph databases searchable with a single query.

The ability to query across disjointed graphs is an entirely new capability for the graph database world. No other graph database on the market offers this feature.

Security

Developers and administrators require granular control over access to data for security and privacy.

For example, a healthcare application can give doctors a view of a patient's diagnosis while an administrator only has access to the patient's appointment schedule, ensuring that only the right users have access to the right data.

Robust data security is critical to the future of any database platform. Neo4j offers granular security that provides role-based access control at the database level to support the needs of different roles and applications.

Building security into the database simplifies secure application development. Rather than tasking developers with backend security, developers write their applications against a scalable and security-conscious backend. That backend is increasingly a database that has both horizontal and vertical scale, as well as extremely fine-grained security controls. As application development changes, today's graph database needs to enforce rigorous enterprise security rules while remaining easy to deploy and manage.

Neo4j offers identity and access control using Kerberos and LDAP. Communications with the database take place over Neo4j's internal binary protocol or using HTTPS requests. Neo4j provides rule-based granular security that can be applied at the level of the data model, as discussed next.

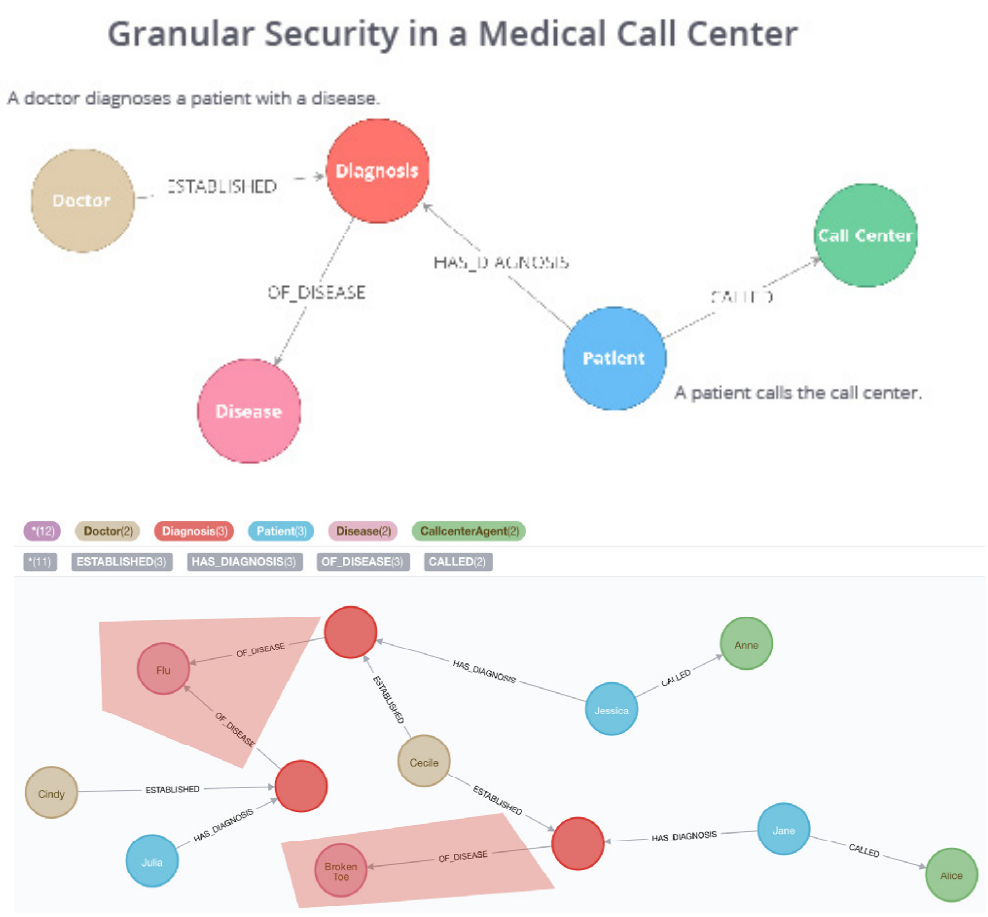
Granular Security

Security applied to a specific schema allows multiple users to have different views of the same database. Each user is assigned a role, and that role has permissions to read – or read and write – certain types of data.

In a graph data structure, this greatly simplifies the task of assigning permissions. For example, with schema-based security you define the kind of access that a given user has against a graph. One user may have access to all the objects, nodes and relationships. Another user may have access only to some objects or some properties of given objects.

Granular security empowers security staff to create rules about sensitive data and know that they will be applied at the database level. Administrators do not need to worry about managing a complex distributed set of security capabilities; they declare specific role-based permissions at the schema level and know that everything is protected in accordance with those permissions. Granular security offers security configuration consistency across Neo4j databases, clusters and shards.

Neo4j restricts what data is seen by different users, supporting role-based access control. The restrictions are defined using database and schema information. Think of a human resources database. With the right security rules in place, the same database serves as a staff directory for everyone, as well as a database where sensitive HR information is kept.



Neo4j offers fine-grained security down to individual objects and their properties. Neo4j grants permission to traverse, read or write data based on node labels, relationship types or database and property names.

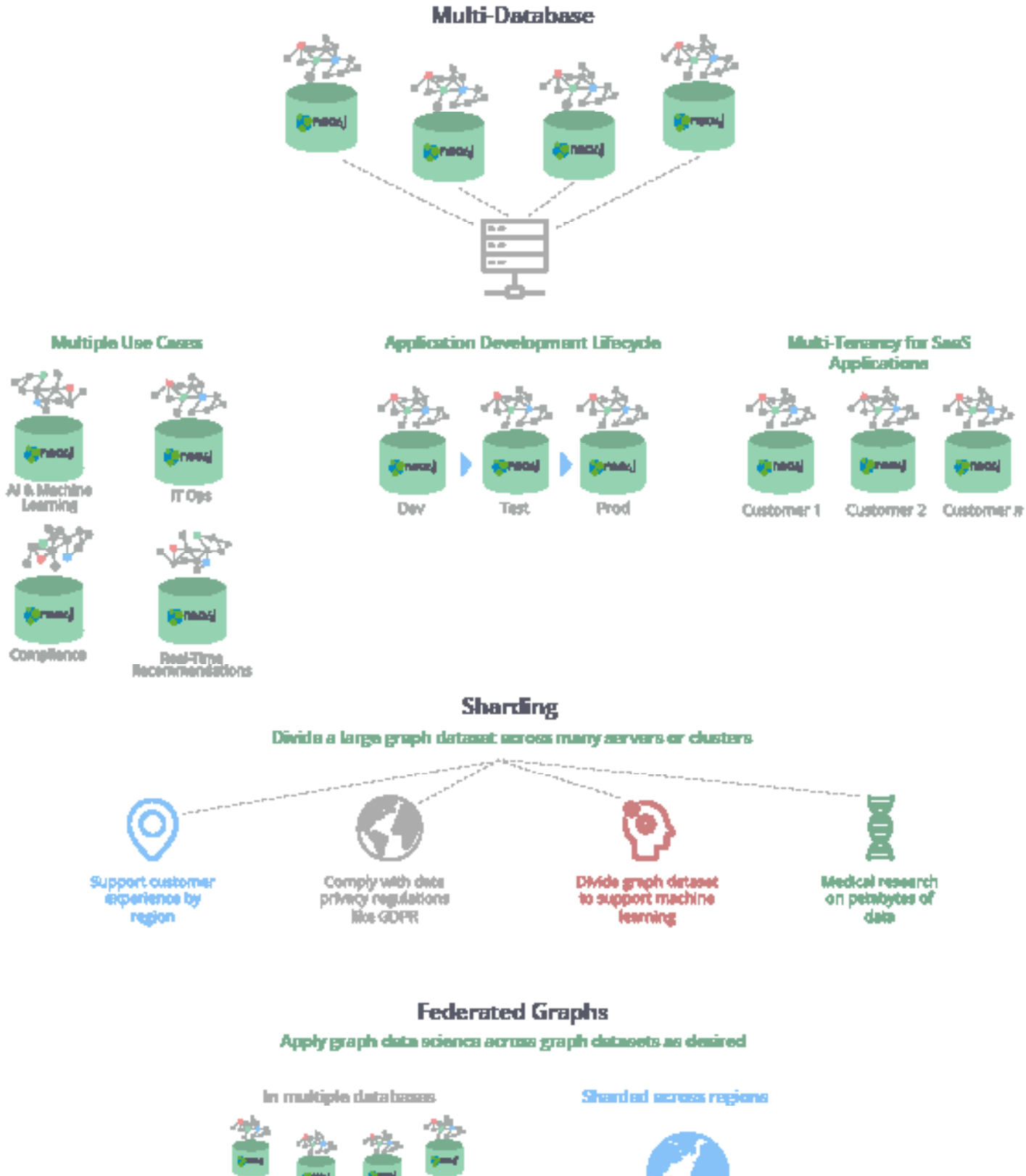
It defines and replicates security rules across the cluster using roles that are associated with users. These rules apply to everyone, even those who query the database using Cypher.

Consider a large research hospital connected with a university. Doctors, healthcare workers, researchers and administrative staff all have different security profiles. Doctors may have access to individual patient data but not billing information. Call center staff who schedule appointments need access to contact information for patients but not their diagnoses or other health information. Researchers gauging the impact of certain medications may have access to test results and selected demographic characteristics of a large group of patients but not to personally identifiable information.

Modern applications require embedded security that is applied no matter where data physically resides. Schema-based security offers this level of protection, regardless of whether there are multiple on-premise graph databases or an enormous graph sharded across multiple cloud repositories around the world.

A graph database may be sharded across hundreds of machines or clusters, but the schema describes all of the data in them. Organizations declare security rules based on that schema, and those rules are inherited by all users and applications that access the graph.

Architectural Agility to Scale Connected Applications



Agility

From data models to infrastructure, the Neo4j Graph Database makes it easy to fluidly evolve your solution as business requirements change.

Neo4j offers full multi-database capabilities with separation of data, allowing multiple databases to be run inside a single Neo4j cluster. This provides significant operational efficiency, security and agility for B2B SaaS multi-tenancy, development convenience and regulatory compliance.

Multi-Database

Many production applications need physical data isolation underneath shared infrastructure.

In a single server or cluster, Neo4j supports multiple databases, whether those databases have the same schema or distinct schemas.

The ability to support multiple databases with the same schema is critical for multitenant environments, such as SaaS applications where each user's data is kept in a separate database. Such a multitenant environment offers physically separated storage and replication.

Multitenancy enables you to optimize resources in a typical cloud environment. Suppose a server is set up to handle 10 tenants. If one of those tenants is heavily using the system and requires more resources, the database for that tenant can be easily moved to a dedicated server with more resources.

Another use case for multiple databases with the same schema is enhanced support for the application development lifecycle. A separate database may be used for development, another for testing and yet another for production. This ensures developers do not impact testing or production, and those involved in testing have a separate environment.

A single Neo4j instance may include multiple databases with entirely different schemas. For example, multiple databases supporting the needs of different departments may reside in one cluster. A graph database used for real-time recommendations may coexist with graph databases for HR, marketing, sales and finance, all of which have different schemas and security requirements.

Flexible Deployment

Organizations want to be able to deploy graph solutions on-premise or in the cloud, depending on their requirements.

Neo4j offers multiple cloud deployment options. Many customers manage Neo4j themselves on their cloud of choice. Neo4j is also available as a managed service, whether that's a white-glove service with 24x7 support for a global enterprise, or a self-serve, pay-as-you-go managed service to make it easy for developers to get started with graphs.

Organizations often manage Neo4j themselves – either in the data center or in the cloud – when they have a high need for compliance or data governance. These organizations require tight control over not just the infrastructure, but the location of their data. The expense of maintaining the expertise to manage every unique piece of software in their enterprise ecosystem is offset by the importance of isolating their data and their infrastructure from the outside world.

However, large enterprises have become increasingly comfortable with the security available in the cloud and the benefits of operating in cloud IaaS environments, both in terms of cost savings as well as flexibility. Today, even security-conscious companies have moved to the cloud in part to offload the burden of infrastructure management to cloud providers, but also to offload the risk and expertise involved in managing each element of their software architecture to providers. Neo4j works with multiple public cloud partners, including GCP, Azure and AWS.

Large companies may use Neo4j Cloud Managed Services, hiring the experts who built Neo4j to manage it in their own dedicated private cloud or virtual private cloud.

Smaller companies that want a production-quality instance of Neo4j have the option of choosing Neo4j Aura. Neo4j Aura is a fully managed graph database as a service, built specifically for startups and small technology companies. Neo4j Aura manages operations and security and offers 24x7 availability. Neo4j Aura allows smaller organizations to start working with a production-quality instance of Neo4j in one click.

Conclusion

The future of the database is the graph data model, reflective of the highly connected world we live in.

The more an organization understands the power of graph queries and graph algorithms, the more demand they have for graph database technology.

Once you decide to add graph technology to your infrastructure, you need enterprise-grade capabilities, not just a graph layer on top of an existing data store. Neo4j has invested time and expertise in solving the difficult computer science problems inherent in scaling a graph database. As a result, Neo4j is the world's leading graph database, positioned to handle the most complex enterprise use cases.

Neo4j is the leading graph database platform that drives innovation and competitive advantage at Airbus, Comcast, eBay, NASA, UBS, Walmart and more. Hundreds of thousands of community deployments and more than 300 customers harness connected data with Neo4j to reveal how people, processes, locations and systems are interrelated.

Using this relationships-first approach, applications built using Neo4j tackle connected data challenges including artificial intelligence, fraud detection, real-time recommendations and master data. Find out more at Neo4j.com.

Questions about Neo4j?

Contact us around the globe:
info@neo4j.com
neo4j.com/contact-us