WHITE PAPER

How Neo4j Co-Exists with Oracle RDBMS

Strategies for Successful Cooperation

Stefan Kolmar, VP of Field Engineering, Neo4j

White Paper

TABLE OF CONTENTS

How Neo4j Co-Exists with Oracle RDBMS

Strategies for Successful Cooperation

Stefan Kolmar, VP of Field Engineering, Neo4j

Introduction

Let's be honest. No one can argue about the value of the Oracle RDBMS database. Oracle is and continues to be the mainstay for enterprise applications. However, another – but not contradictory – truth remains: IT organizations need capabilities beyond what Oracle and other relational databases provide.

While organizations don't want to rip and replace Oracle, they do want more: They want more features in their applications, more data variety, more agility, more speed and, perhaps most importantly, more ways to rapidly uncover and innovate based on the rich connections inherent in their data.

Connected data enables advanced capabilities that put companies ahead of the competition, including reduced time to market in many development contexts. With connected data, companies can better unify data across many disparate systems with better Master Data Management approaches, and increase positive behaviors and possibly revenue with contextual and relevant real-time product recommendations across ever-growing and evolving datasets. They can also save millions of dollars by analyzing complex connections to fight financial fraud in real time.

Every business needs to leverage these data relationships and leverage them faster. A graph database, like Neo4j, delivers those capabilities – at no risk to your Oracle investments.

What is Neo4j?

In order to understand how a graph database works and the value it brings, it's helpful to understand how relational databases, like Oracle, work. Relational databases store highly structured data in tables with predetermined columns and many rows of the same type of information. Because developers must structure the data used in their applications in this tabular format, the fixed schema works best for problems that are well-defined at the outset.

Relational databases are the perfect tool for highly structured, predetermined schemas. However, these databases don't adapt well to change nor do they provide an efficient approach for traversing relationships between data elements. It's ironic, but relational databases are not good at navigating multi-layered, complex relationships in real time.

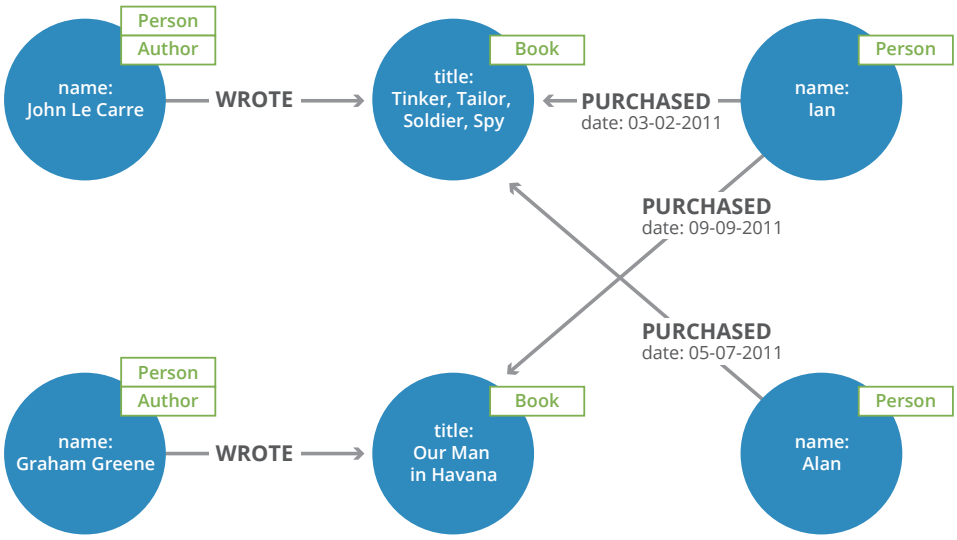
A graph database, like Neo4j, puts data relationships first. Neo4j is an ACID-compliant, transactional database management system with Create, Read, Update and Delete (CRUD) operations working on a graph data model. The graph data model is easy to understand, as it reflects how data naturally exists – as objects and the relationships between those objects.

What's in a Name?

You might be using a "relational database," but the truth is it's not really optimized for dealing with relationships. Sure, it's possible to use JOINs to navigate relationships, but relational databases take a performance hit as queries get deeper and add more JOINs. In fact, the term "relational database" has nothing to do with describing relationships between data, but is a reference to the highly specific mathematical notion of a "relation" – a.k.a. a table – as part of E.F. Codd's relational algebra.

You can think of a graph data model as composed of two elements: nodes and relationships. Each node represents an entity, and each relationship represents how two nodes are associated. By assembling the simple abstractions of nodes and relationships into connected structures, Neo4j enables you to build sophisticated models that map closely to a problem domain.

Neo4j enables developers to be more agile and to deliver applications faster due to one simple fact: It's a schema-optional database. This means there's no need to set up elaborate data models based on what you think the business requirements are.



A simple graph data model: Books, authors and owners

Instead, this flexibility abstracts your data model and validation to the application tier, but still inherently provides you with the right level of control to define the data coming in. A data model in Neo4j mirrors whiteboard drawings, closing the gap between business and IT, and enabling you to make changes on the fly as business requirements change.

Because Neo4j uses an intuitive data model that anyone in the organization can understand, querying the data is just as easy to grasp, even for team members without a database background.

Advantages of Neo4j and Oracle working together

There are several advantages to using Neo4j and Oracle RDBMS together, beginning with the simple fact that Oracle is a trusted and well-established technology. With so many enterprise applications running on Oracle, you shouldn't throw away your existing investments in Oracle infrastructure, tools and training. Fortunately, you don't have to. All of your applications can continue to use Oracle RDBMS uninterrupted. By adding Neo4j to your arsenal, you lose nothing while gaining all the strategic capabilities of a graph database.

Using Neo4j and Oracle together also enables organizations to improve application performance by offloading queries that leverage connected data. In an RDBMS, data relationship queries can be answered by creating JOINs between database tables. However, these operations are compute- and memory-intensive, and the performance impact increases as data volumes grow. A graph database can analyze complex, connected data much faster and persist that analysis for future reference.

Finally, there are scenarios where using an RDBMS versus a graph database simply makes sense, and vice versa. Oracle is great for tabular, structured data while unstructured, highly connected or very dynamic data such as hierarchies and networks can be moved out of Oracle and into Neo4j. This enables agility. When architected to coexist, application performance will improve since fast traversals can be done against the graph using Neo4j while transactional data can be retrieved from Oracle.

Graph databases and relational databases each address specific use cases. Neo4j, therefore, doesn't compete with Oracle but coexists with it. Together, Neo4j and Oracle enable developers and architects to use the right type of database for the right use case.

Graph databases and relational databases each address specific use cases. Neo4j, therefore, doesn't compete with Oracle but coexists with it. Together, Neo4j and Oracle enable developers and architects to use the right type of database for the right use case.

How does Neo4j work with Oracle?

There are several ways that Neo4j can work with Oracle (or any other RDBMS). The approach you take depends on what your enterprise is trying to achieve.

Migrate or sync a subset of the data with Neo4j

If there are questions your current application can't efficiently answer because of the depth and complexity of data relationships, you might choose to migrate relevant data to Neo4j, where Neo4j is the transactional ACID-compliant data store for that portion of the data. For example, while the customer and product information can reside in Oracle, the identifiers for customers and products and the way they relate to each other can reside in Neo4j. The calling application can ask Neo4j for relationship information and then use the result set to get details from Oracle.

Another approach is to synchronize a portion of the data from the RDBMS to Neo4j. This was the case for Norwegian telecom Telenor. Telenor's self-service web application was facing a key challenge: login times were increasing, which would ultimately result in losing customers. Logins needed to calculate resource authorization for each user, which, especially for large customers with many users, takes a long time.

At first, Telenor chose to use a stored procedure to pre-calculate resource authorizations for its largest customers in a nightly batch job and then cache the results. This would speed up login for the largest customers; for smaller customers, resource authorizations would be calculated on the fly.

The approach had some drawbacks: the cached data could be up to 24 hours old, so any changes (such as adding new users or deleting terminated employees) could take as much as a day to kick in, and it did nothing to solve the problem for smaller customers. Further looking ahead, Telenor could see that the pre-calculation approach would not ultimately scale as growth continued. The stored procedure alone was extremely complex, with about 1500 lines of SQL code.

Telenor looked to a NoSQL approach to augment their current RDBMS. They decided to sync their data with Solr/Lucene for search capabilities and their resource authorization data, which is highly connected, with Neo4j. Using Neo4j, Telenor could perform resource authorization in real time rather than pre-calculating it, providing users with millisecond-level login times, with changes reflected in near real time.

Syncing data with Neo4j is handled through middleware such as Oracle GoldenGate and related adapters, including those from Oracle as well as open source adapters such as the Oracle GoldenGate and Kafka adapter created by Monsanto and available on GitHub.

Migrating data into Neo4j is handled through CSV files. You export tables out of Oracle and then import them into Neo4j as CSV files using either a command-line tool or syntax inside the Cypher graph query language. This can be performed manually or you can use an ETL tool to automate the migration.

Fully sync Oracle and Neo4j data

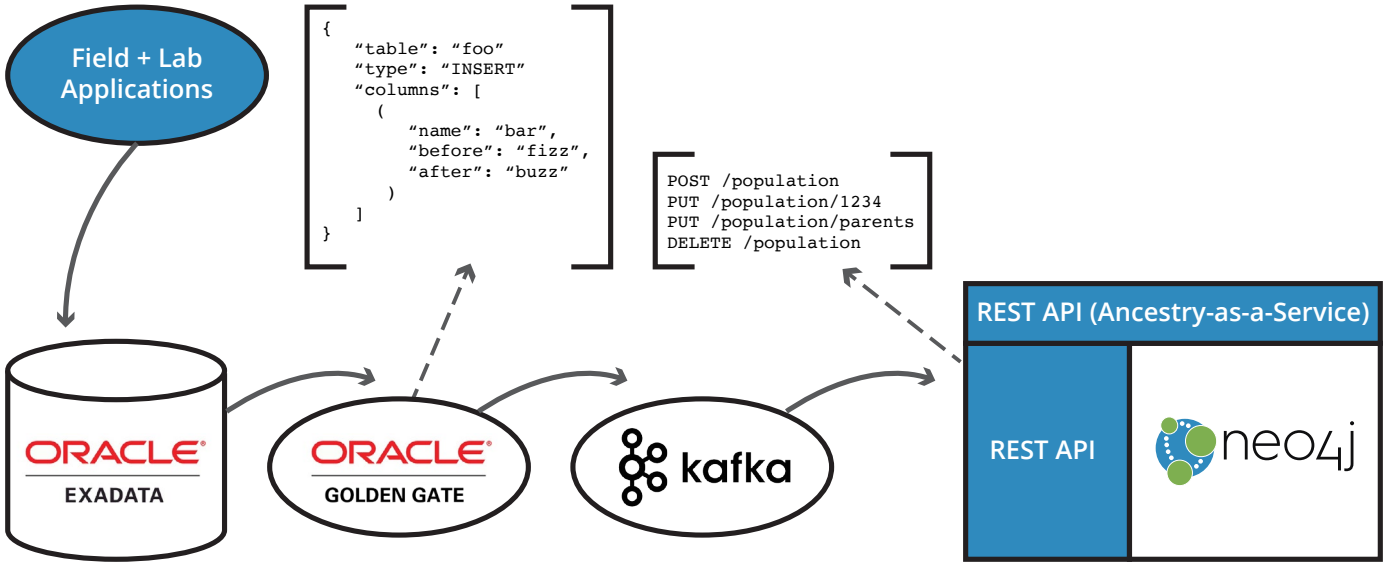
A more complex use case involves synchronizing all data between Oracle and Neo4j. Applications that integrate data from multiple data sources are a common use case for full synchronization. Another use case for a full synchronization arises when you have an existing set of applications writing to an Oracle database and changing those applications is cost prohibitive. For the data to add increasing value, new technologies need to be introduced where the Oracle RDBMS is incapable. This was the case for Monsanto.

Monsanto is a multinational leader in agrochemical and agricultural biotechnology. Prior to adopting Neo4j, Monsanto relied on a 96-CPU Oracle Exadata installation to host its core genetic ancestry data with plenty of stored procedures, JOIN tables, recursive queries and dual indexes to optimize performance. The Monsanto team was well-versed in Oracle tuning and optimization, with over 30 years' experience of tuning Oracle RDBMS between them all. However, the Exadata instance regularly failed to process genetic ancestry data in real time – a prerequisite if the team was to use a new genomic testing technique that could take a full year off of its time-to-market cycle.

The team's first attempt to generate real-time results was to build and parse gigantic in-memory graphs. But once a query was complete, the graphs disappeared. The team looked for a way to persist graph data over the long term and found Neo4j.

Within one day of discovering Neo4j, the team built a prototype with a small dataset. A month later, the team had the entire genetic ancestry dataset in Neo4j for a beta-release application. However, even with the Neo4j deployment in full production, dozens of applications continued to read and write data to the Exadata environment. Instead of turning off these database connections all at once, the team built a custom API layer to sync the stream of information to and from Exadata with Neo4j.

The team then introduced a valuable new query interface where their scientists could execute deeply connected queries in a simple, keyword-driven way that wasn't previously possible with SQL algorithms. The architecture uses Apache Kafka as a distributed commit log to feed Neo4j with live transactional data from Oracle; the team built an Oracle GoldenGate and Kafka connector, which they open sourced and made available on GitHub.



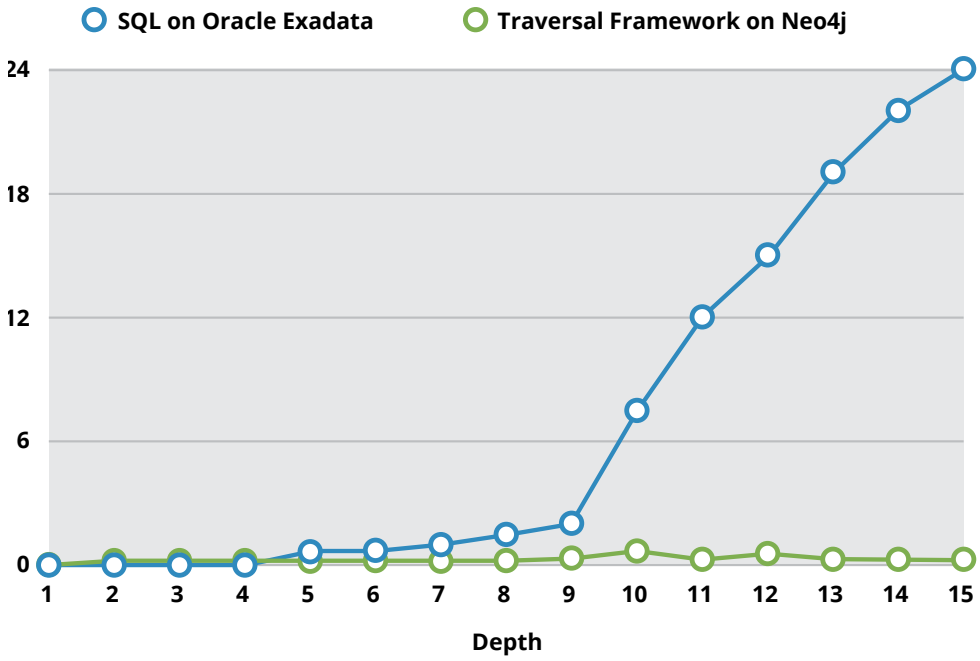
Using Oracle and Neo4j for Ancestry-as-a-Service at Monsanto

Depth and density of connections and data volume impact query times

With Neo4j, you can query data with a depth of millions or tens of millions of connections per second per computer core, which would be the equivalent in the relational database world of millions of JOIN operations per second per core, which is impossible. There's a big difference in speed, and this difference amplifies the tighter your connections are and the more data you have.

The more data you have, the slower it becomes to relate data in other kinds of databases, including Oracle. Using Neo4j, a shortest path query on data with tens of billions of nodes and relationships might take one or two milliseconds to run. The equivalent SQL query would run thousands of times slower if an application was solely using an RDBMS such as Oracle.

Today, the genetic ancestry solution services approximately 120 different applications and data scientists, handling over 600 million REST requests across approximately a billion nodes, and results are returned in just tenths of milliseconds.



Finding all of a plant's ancestors at Monsanto: Query times for Oracle Exadata versus Neo4j

Migrate all the data into Neo4j

In some cases, an existing application may be hitting its architectural and performance limits. Often, this means you are faced with developing a new application. If the data in question is highly connected, it may make sense to use Neo4j rather than Oracle as the sole database store, and in that case you would migrate all the data to Neo4j.

There are several scenarios where it may indeed make sense to do a rip and replace for a particular application. If business users are complaining about prohibitively slow application performance, for example, then you may want to consider moving entirely to Neo4j since doing so can greatly improve query times (see sidebar).

Similarly, if most of the queries in an application are joining varying sets and depths of data in real time, where the sets cannot be easily predicted or pre-computed, then the data would be much more efficiently handled in Neo4j. Finally, you may choose to do a rip and replace if your queries have become exceedingly complex and it's taking too long to train new staff members on them. With a graph database, connected data queries are much simpler and easy enough for even beginners to grasp with minimal training.

Replacing Oracle RDBMS with Neo4j is similar to migrating a subset of data, only you're migrating all of the data. The migration also involves editing the application code that interacts with the database and rewriting those queries.

A number of companies have moved traditional systems, like supply chain management, to Neo4j. Schleich GmbH, one of the largest toy companies in Germany, found that its RDBMS-powered product data management (PDM) solution wasn't delivering the flexibility, performance and operational efficiency their supply chain required. Schleich decided to create its next generation PDM system in Neo4j.

Benefits of Neo4j

Performance

- Much faster queries on connected data

Flexibility and agility

- Change the data model any time, without risk

Ease of understanding

- Intuitive data model you can draw on a whiteboard

Ease of coding

- 50 to 200 lines of SQL translates into 3 to 7 lines of the Cypher graph query language

Within six months, the company's technology partner, Structr, had migrated all the data from the previous PDM system into the company's open source graph application program, which leverages Neo4j. Traceability and compliance across the entire value chain are key aspects of the toy industry, and the new PDM system meets all of these requirements, including allowing third parties to gain visibility where needed. The graph-based PDM system is being integrated with SAP and introduced into key areas of the company.

Another use case where migrating all the data into Neo4j made sense involved integrating numerous systems from acquired companies into a single application. Billes, a printing house in business since 1939, together with its technology partner NetConsult, built a system for planning, ordering, shipping and billing based on Neo4j. A variety of mergers and acquisitions had led to a patchwork of tools and parallel IT systems. Relevant portions of all of these systems were integrated seamlessly into the Neo4j-based system, which is called Poff.

Because Neo4j has a flexible data model that can be changed at any time, system integration efforts were greatly reduced. In total, about 40 external systems receive orders and process them through Poff. The success and ease of administration of the system have enabled the company to add new business areas, including online orders from consumers.

Conclusion

It's time to get real about databases. Relational databases can't do everything you need them to do. But more importantly, you don't have to sacrifice the capabilities of a graph database to preserve your investment in Oracle RDBMS – or vice versa.

With Neo4j you can use best-of-breed technology as your applications dictate – whether that be a graph database or a relational database. As the world's first graph database built with native graph storage and processing, Neo4j delivers unparalleled performance even as your data grows. And we continue to get better, thanks to a large community of graph database enthusiasts who contribute to the Neo4j ecosystem.

To learn more about adding Neo4j to your environment, contact a Neo4j expert or download a 30-day free trial of Neo4j Enterprise Edition.

Neo4j is the leader in graph database technology. As the world's most widely deployed graph database, we help global brands – including Comcast, NASA, UBS, and Volvo Cars – to reveal and predict how people, processes and systems are interrelated.

Using this relationships-first approach, applications built with Neo4j tackle connected data challenges such as analytics and artificial intelligence, fraud detection, real-time recommendations, and knowledge graphs. Find out more at neo4j.com.

Questions about Neo4j?

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