| | |
|---|---|
| Name: | José A Alvarado-Guzmán |
| Affiliation: | Faculty Practice Organization, Columbia Medical Center, Columbia University |
| Email: | Jaa2220@cumc.columbia.edu |
| Presentation type (s): | Software Demonstration |


# Relational to Graph Database: Migration


**José A. Alvarado-Guzmán, MS, Itay Keren, MS**
**Data Science Institute, Columbia University, NY, NY, USA; Electronic Engineering, Columbia University, NY, NY , USA**



**Abstract**

***This paper describes the design and implementation of a Java application use as a data migration tool between a relational database and a graph database. Using the Observational Health Data Science and Informatics (OHDSI) White Rabbit and Rabbit In A Hat as a basis, we developed a user friendly multi-thread GUI that allow the migration of relational data to Neo4j.***


**Introduction**

One of the greatest business trends of this millennium in Information Management is to leveraging high connected data with complex and dynamic relationships. The organization ability to understand and analyze this type of data will be key in determining how successful an organization could be among its competitors. For any significant size, graph databases are the best way of representing and query highly connected data.

The increased popularity of graph data and graph thinking was mainly driven by the tremendous success of social media (Facebook & Twitter) and the commercial success of companies like Google that centered their business model on graph technologies. The easy and free accessibility of general purpose graph databases thanks to the open source community, Neo4j for example, is also impulsing the graph database usage and popularity.

Despite the flexibility and performance of graph databases on high connected data, most developers still prefer to model this data using the relational model due to the learning curve that graph data storage and querying requires. For this reason we design a simple and interactive application that migrates data stored in relational tables into nodes and edges (Neo4j). This migration tool (RD2GD) make it easy and straightforward for a DBA to map tables to the target of the graph database, without having to understand the many intricacies of graph databases, so that they can quickly convert the data and start utilizing the benefits of graph databases. As a test, we will convert a very large dataset in the healthcare industry from MySQL into Neo4j.

***Advantages of GDBMS over RDBMS***

Relational databases are a mature, established technology that is used across many industries. They have been in use in one way or another for over 40 years, and they are certainly not becoming obsolete anytime soon. However, since the advent of ACID (atomicity, consistency, isolation, and durability – the four required properties of a robust database) graph databases in the 2000s, the adoption of graph databases for more industries has grown tremendously.

The prevalent usage of social media sparked the need for extremely interconnected big data – lots of nodes (people) connected to other nodes (friends) and contain many properties and preferences. In order to traverse and retrieve all of these relationships, a RDBMS would require many table joins that take a lot of time and power to compute, while the graph database is optimized and designed to deliver these calculations more directly. In figure 3, we see a comparison between a RDBMS and Neo4j of computing and retrieving different levels of "friends-of-friends", based on the same input data – about 1 million people, each with about 50 friends.

As you can see on Figure 1[2], the execution time of the RDBMS increased quite dramatically between depths, while Neo4j rose fairly linearly. Besides this major advantage, there are other aspects that make graph databases attractive to implement. Since GDBMS's are not bound by schema, they have the flexibility to build from existing data, define new labels, and establish new relationships as needed. The traditional process of database design for RDBMS is extremely involved, and the developer must understand the data and the interaction with the data very well – well before any actual data is entered into it! On the other hand, for GDBMS, as the data grows, the relationships become more complex, and the developer learns the intricacies of the application, it defines how to shape the data – and which new labels and relationships are needed. In general, implementing a graph database from a conceptual ER diagram is direct – whereas for relational databases a complicated table structure is built, which is harder to understand, use and maintain.

| Depth | RDBMS execution time(s) | Neo4j execution time(s) | Records returned |
|---|---|---|---|
| 2 | 0.016 | 0.01 | ~2500 |
| 3 | 30.267 | 0.168 | ~110,000 |
| 4 | 1543.505 | 1.359 | ~600,000 |
| 5 | Unfinished | 2.132 | ~800,000 |

**Figure 1.** Relational & Graph Databases Execution Times Comparison.

## Conclusion

In the constantly-evolving world of databases, developers must keep up with the latest improvements and trends. Developers often are tasked with exploring newer database technologies such as graph databases because new applications exceed the capabilities of the older "trusty" relational databases. In these cases, having a tool for data migration is invaluable. With RD2GD the developer can simply drag and drop, and the app does all the conversion and graph database interfacing in the background, so all that is left to do is enjoy the benefits of graph databases. This helps to lower the learning curve and make the amazing graph database technology much more accessible to new users. Using this tool, we were able to convert a big dataset into Neo4j. In Neo4j, complex relational queries can be run against the data efficiently and easily – because querying the data in SQL would be practically impossible for such a large and interconnected dataset.

## References


1. Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph Databases: New Opportunities for Connected Data.* O'Reilly.
2. Vukotic, A., Watt, N., Abedrabbo, T., Fox, D., & Partner, J. (2014). *Neo4j in Action.* Manning Publications.
3. Dongen, S. v. (2000, May). Graph Clustering by Flow Simulation. *PhD thesis*. University of Utrecht.