Graph databases by the example of Neo4j

# Introduction

A great deal of today’s software applications are built around databases and perform various tasks on large data sets to serve certain business interests. Relational databases have been around for decades and have been optimized to serve this purpose best. However, with the growing of the internet and the vastly increasing amount of data (big data), new types of databases have emerged under the designation NoSQL. Common representatives of this kin are key-value stores and document-oriented stores, but also the less established graph databases. In contrast to its relatives, the latter focuses strongly on relations and the topology between entities instead of the entities itself. Graph databases therefore provide customized query languages with rich expressiveness regarding relations and also feature algorithms from graph theory. They are therefore suitable for problems where efficient graph traversal and related graph algorithms are essential.

The subject of this document is to examine the usability, performance and possible application scenarios of graph databases. As an example for a graph-related problem, two games based on the Wikipedia will be demonstrated:

1. Wikipedia Philosophy game  
   Select a random article. Follow the first non-italic, non-parenthesis-enclosed link of the main text and repeat this step until a cycle is detected or the article Philosophy is reached.  
   Several persons analyzed this issue und came to the conclusion that roughly 95% of all articles link to Philosophy this way[[1]](#footnote-1).
2. 5 clicks to Jesus

Select a random article. Using all links of the main text, try to reach the article Jesus in 5 clicks (links) or less.

# Data sets

Three dumps of the Wikipedia articles have been used throughout the tests:

|  |  |  |  |
| --- | --- | --- | --- |
| Language | Date of dump | # Articles | # Links |
| English | 2014-12-08 | 15113788 | 164379808 |
| Russian | 2015-01-10 | 3039356 | 36903313 |
| Plattdütsch | 2015-01-08 | 607331 | 31814 |

The English Wikipedia was used initially with the goal to solve the original philosophy game. However, during preprocessing we came to the conclusion that the data set was far too large to build a fair database for benchmarks and tests. The reason for this is that, although all articles have ids, Wikipedia stores its links using the textual name of the linked page, and not its id. Still, well-designed relational databases use integral ids to refer to foreign entities. Considering a fair comparison, we tried to resolve all textual links (several million) to their page ids. However, with <10 resolved ids per second due to the large dataset, we aborted this process as it would have probably taken months to complete. It would be possible to resolve the ids without a database using e.g. a custom coded program and a hash map with enough size to store 10GiB strings as keys with associated values. Nevertheless, this would not solve the equivalent problem when inserting into a graph data base where relationships can only be created after still matching the destination page by either string or integer. The ultimate choice was to drop the large data set and try a smaller one. First experiments with the Russian Wikipedia showed that despite all indexes resolving all pages would still require approximately 11 days. Furthermore, due to the language’s nature, Unicode support became an essential requirement, doubling the storage size of the database. While inserting all links, we discovered that our SQL database (LocalDb) has a limit of 10 GiB per table. Thus also the Russian Wikipedia was too large for our experiment. Finally, we have chosen a quite small Wikipedia which can still somehow be understood, the Plattdüütsch Wikipedia.

# Handling large files

# Method

# Input preprocessing

# MS SQL Database (LocalDb)

# Neo4j Database

# Benchmarked queries

Insert script took (eng)

00:46:00.1983008 (b)

00:41:19.1609778 (p)

# Conclusion

1. http://matpalm.com/blog/2011/08/13/wikipedia-philosophy/ [↑](#footnote-ref-1)