Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 2019

Methods for Automatic Selection of Database Systems for Optimized Query Performance

Matthäus Zloch [1]

Abstract: On a database system query runtime performance is, besides the data volume, mainly affected by the data model and the index structures that one database system utilizes to store its data in. There are lots of different implementations of the traditional relational model, triples stores, and yet more with the rising NoSQL technologies. For a developer who needs to choose one database system, it is key to understand how the data, that one application will store, looks like (data model and schema implementation) and what typical use-cases the system will be addressed with (queries), in order to make sure the system is truly suited to reach optimal overall query performance. This thesis will introduce methods and applications, in order to detect the optimal database system for a given data model and its queries.

Keywords: database systems, data modeling, machine learning, software architecture

1 Motivation and Problem Statement

Relational databases have been around for several decades. It has been a first choice technology for developers, when it comes to the decision of how to store the data of an application. In the past years, developers tend to move to technologies outside the relational model, that comprise different solutions and ways of storing data. The core assumption in this thesis is that a developer cannot know in advance, which of the available database systems are suited best for the data and the queries of an application. If the decision shall be made with care, an evaluation must be made each time an application is developed. In this thesis a closer look at the structures of data models and queries is taken, in order to find the database technology that is suited for the data of an application.

A data model is an extremely organized and selective abstraction of knowledge [Ev03] and with rising size and complexity, it tends to have different structural characteristics. For instance, there are parts with lots of many-to-many relationships and tree-like structures, which are likely to be seen in graph-based models. Those models can be found mostly in applications which need a high interconnected-ness of their entity types, e.g. social media platforms. On the other hand, we can find models which store pure numerical or string-based data in a plain tabular format, e.g. an address book or a financial application. Every database system utilizes a different kind of internal data structure that deals with the data it stores. A data model that is composed mainly of characteristics that are typical for graph-structures, a graph-database might be suited best. However, since each query addresses

[1] Heinrich-Heine Universität, Lehrstuhl für Datenbanken und Informationssysteme, Universitätsstrasse 1, 40589 Düsseldorf, matthaeus.zloch@hhu.de

different parts of a data model in the database, a developer cannot tell in advance which database system is suited best for an application.

The research idea presented in this thesis is to look at data models prior to implementation of the application and to utilize their structure, queries and data, in order to predict a suitable database system. A suitable database system is one that serves with as best performance as possible every query of an application, compared to other database systems. In order to predict the runtime of some queries which address a schema, an algorithm by means of a cost-function has to be developed, which takes a number of features into account. The features need to be identified on the schema level, query level, and, if available, on the data level. The algorithm is supposed to compute a value which will represent a degree of suitability of the given set of queries addressing the given schema, to a list of a fixed set of database systems of different types. It is planned to use a method from the area of machine learning to solve that problem.

From this aim arise several research questions that should be answered throughout the thesis, which are among others:

- Is it reasonable to study the structure of the data model and the queries before making a decision about the database system?
- What makes a data model be graph-based? What makes it be not graph-based?
- How ”good” can the runtime of a query be predicted?
- What are the factors that influence a decision about a database?

2 Related Work

Related work is to be found from the time when triple stores were introduced to store RDF data [MMM04]. Research on alternative database systems is as old as database systems itself. With the growing number of database systems available and the recent change in pragmatism (from full ACID to BASE, i.e. from traditional to new models), researchers start to think about which database system is the ”right” one for their first of application. Usually, comparison papers look within a group of similar database types, e.g. the group of NoSQL databases or RDF data stores, [Ro07][An12][JV13][Ga14]. In the area of RDF data stores, recent works reach out to compare among database systems of other types, like relational databases and NoSQL databases in theory and applied informatics [AG05][Vi10][Cu13]. *polyglot persistence*[Fo11] is a term which came up in the context of applied NoSQL databases. It describes the fact that different kinds of data is stored in different types of database systems. Related work and similar methods is also expected in this area.

A *recommender* for database systems according to a given data model and a set of queries, what this thesis is trying to accomplish, is not known to us. There are lots of one-to-one comparisons of features [Ga12] of database systems and articles which document experiences of usage [Ri16]. However, most likely the developer is left with picking one

system according to the application needs. This decision is rather backed up by experience and not by characteristic numbers.

Other related work is expected in the area of database systems and query evaluation plans. We expect to learn from that area of research.

3 Research Plan

Related Work, from now. Find more similar works in this area.

Evaluation, by mid 2016. First, a motivating paper shall show that it is feasible to investigate in further research. It will cover an evaluation with different types of database systems on one specific data model and example data from a real-life application, together with a fixed set of use-case queries for that application and data model. By the time of writing this paper, there are already results available, which will be described in more detail in Section 4.

Implementation Work, from mid 2016. Regarding the cost-function, in order to learn from as much queries and query runtimes for all database systems considered as possible, it is either (1) aimed to use a data generator that creates as much as realistic test data as possible for any arbitrary data model, or (2) use benchmarks with test data and queries for that. Since benchmarks are designed to be run on a specific database system, it has to be evaluated, if their use is reasonable at all. A data generator would suite for any data model and generate data with different complexity. For instance, it might be interesting how different probabilities and relations influence the cost-function's output. The method and the algorithm will constitute the second paper of the thesis.

Implementation Work, by mid 2017. The implemented algorithm shall be integrated in an application that is more user-friendly. A lightweight web application is considered. Also considerable is an integration into another already existing application. As input data, the user would pass a data model in a specific format and a set of queries that are likely to be executed on the system. This implementation will wrap up the implementation work for this thesis. It is planned to publish a comprehensive article about the final product.

4 Current Status and Results

The evaluation paper has started to be written and some results of the evaluation are still to be interpreted.

In preparation for the first paper the evaluation data has been extracted from a real-life application called MISSY[2], which is a web application that presents official statistics data about European and national surveys. MISSY implements the DDI-RDF Discovery Vocabulary (DISCO in short) [Bo13] as its native data model. DISCO is of sufficient complexity

[2] The Microdata Information System MISSY - http://www.gesis.org/missy

for the evaluation, since it consists of graph-like data (highly connected social science survey information) on one hand and of plain tabular data (statistics data) on the other.

For evaluation there are currently four systems taken into consideration:

- Relational database MySQL Community Edition - an open source database of wide acceptance. The original data is stored in a MySQL database and this implementation forms the baseline.
- Triple store Virtuoso OpenSource[3] - a famous triple store implementation in Java. Since the original model is supposed to be a linked data model, this database has been chosen because of its capability to store RDF data.
- Graph databases: Since graph databases make a dominant part of NoSQL technologies, two different candidates were chosen from this category: (1) Neo4j Community Edition[4] - a native graph database with Cypher as its own query language and (2) Stardog[5] - a graph database that allows to query with SPARQL.

The queries were composed from three different sources (around 40 in total): the project's web application, official usage-cases document available for DISCO[Vo15], and validation queries for DISCO for functional data integrity[Ha15]. Since all types of database systems listed above allow different query languages, they had to be translated into the query language each system supports. The queries were executed according to two different approaches: (1) equal distribution, where each query is executed 25 times within a shuffled list of queries (leading to 1000 executions overall), and (2) probability distribution, were each query is assigned to one (out of four in total) application usage scenarios according to a specific weight. In approach (2) some queries will be executed more often then others in one usage scenario, respectively. Like in approach (1), the total number of executions is set to 1000.

An example of results can be seen in Figure 1. It shows the results for an evaluation run with different distributions of queries on usage-scenario 1. Usage-scenario 1 contains queries which relate to a user's navigation behaviour of the web application MISSY. There are five different weights per usage scenario, here 50/50, 60/40, up to 90/10, where 50/50 means that 50% of the queries are specific to usage-scenario 1 and 50% are not. In 60/40 there are 60% of the queries specific to usage-scenario 1 and 40% are not, and so on. From usage scenario 1 we can see in Figure 1 that the higher the frequency of returning queries is, the faster is MySQL database and slower Neo4j. Else is with Stardog and Virtuoso: Stardog's performance is slightly worse, where Virtuoso's merely differ.

The results remain to be interpreted, but are expected to contribute on different levels. For approach (1) we want to learn from each query individually: how does it look like? Which concepts does it use (aggregation-functions, relations covered, size of where-clause, etc.)? Which parts of the data model does it address, and according to that, on which database

[3] https://github.com/openlink/virtuoso-opensource
[4] http://neo4j.com
[5] http://stardog.com/

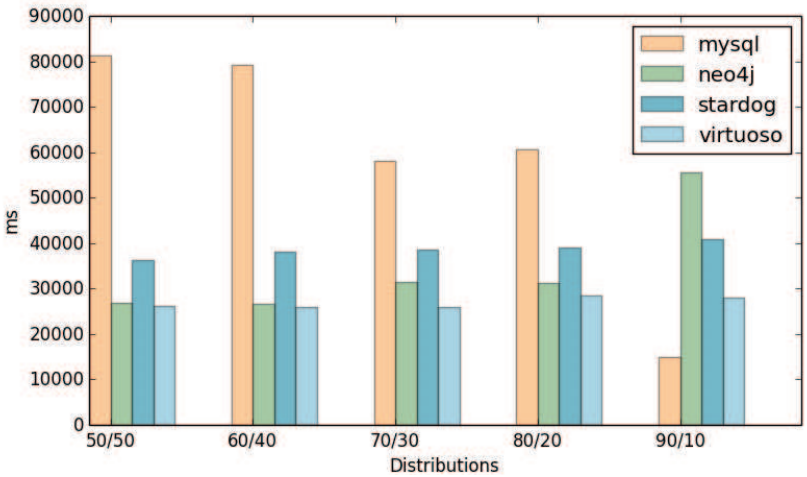

Abb. 1: Increasing the frequency of queries in usage-scenario 1 results in better overall query runtime for MySQL and worse performance for Neo4j. Stardog and Virtuoso perform slightly worse.

system is it executed faster? For approach (2) we will look at overall query runtime performance to motivate that it is reasonable to look at the data model and the queries used in advance.

References

[AG05] Angles, Renzo; Gutierrez, Claudio: Querying RDF Data from a Graph Database Perspective. In: Proc. of 2nd European Semantic Web conference. ESWC, Crete, pp. 346–360, 2005.

[An12] Angles, Renzo: A Comparison of Current Graph Database Models. In: ICDE Workshops. IEEE Computer Society, pp. 171–177, 2012.

[Bo13] Bosch, Thomas; Cyganiak, Richard; Gregory, Arofan; Wackerow, Joachim: DDI-RDF Discovery Vocabulary: A Metadata Vocabulary for Documenting Research and Survey Data. In: Proceedings of the 6th Linked Data on the Web (LDOW) Workshop at the World Wide Web Conference (WWW). CEUR Workshop Proceedings. CEUR-WS.org, 2013.

[Cu13] Cudr-Mauroux, Philippe; Enchev, Iliya; Fundatureanu, Sever; Groth, Paul T.; Haque, Albert; Harth, Andreas; Keppmann, Felix Leif; Miranker, Daniel P.; Sequeda, Juan; Wylot, Marcin: NoSQL Databases for RDF: An Empirical Evaluation. In: International Semantic Web Conference (2). Lecture Notes in Computer Science. Springer International Publishing, pp. 310–325, 2013.

[Ev03] Evans, Eric: Domain-driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Longman Publishing Co., Inc., 2003.

[Fo11] Fowler, Martin: , Polyglot Persistence. Available from http://martinfowler.com/bliki/PolyglotPersistence.html, 2011.

[Ga12] Garshol, Lars Marius: , RDF triple stores - an overview. Available from http://www.garshol.priv.no/blog/, September 2012.

[Ga14] Gandini, Andrea; Gribaudo, Marco; Knottenbelt, WilliamJ.; Osman, Rasha; Piazzolla, Pietro: Performance Evaluation of NoSQL Databases. In: Computer Performance Engineering. Lecture Notes in Computer Science. Springer International Publishing, pp. 16–29, 2014.

[Ha15] Hartmann, Thomas; Zapilko, Benjamin; Wackerow, Joachim; Eckert, Kai: Constraints to Validate RDF Data Quality on Common Vocabularies in the Social, Behavioral, and Economic Sciences. Computing Research Repository (CoRR), abs/1504.04479, 2015. http://arxiv.org/abs/1504.04479.

[JV13] Jouili, Salim; Vansteenberghe, Valentin: An Empirical Comparison of Graph Databases. In: SocialCom. IEEE Computer Society, pp. 708–715, 2013.

[MMM04] Manola, Frank; Miller, Eric; McBride, Brian: , RDF Primer. Available from https://www.w3.org/TR/rdf-primer/, 2004.

[Ri16] Richey, Clark: , From Good to Graph: Choosing the Right Database. Available from http://neo4j.com/blog/good-to-graph-right-database, 2016.

[Ro07] Rohloff, Kurt; Dean, Mike; Emmons, Ian; Ryder, Dorene; Sumner, John: An Evaluation of Triple-Store Technologies for Large Data Stores. In: OTM 2007 Workshop SSWS. pp. 1105–1114, 2007.

[Vi10] Vicknair, Chad; Macias, Michael; Zhao, Zhendong; Nan, Xiaofei; Chen, Yixin; Wilkins, Dawn: A comparison of a graph database and a relational database: a data provenance perspective. In: ACM Southeast Regional Conference. ACM, p. 42, 2010.

[Vo15] Vompras, Johanna; Gregory, Arofan; Bosch, Thomas; Wackerow, Joachim: Scenarios for the DDI-RDF Discovery Vocabulary. DDI Working Paper Series, 2015. http://dx.doi.org/10.3886/DDISemanticWeb02.