NoSQL Databases

Daniel Rutz and Leon Schürmann (Eds.)

March 25, 2019

Contents

1 GraphDB

Thore Krüss, Lennart Purucker, Johanna Sommer

1.1 Motivation/Introduction

First paragraph to state the general context and make important points for the motivation and why its important. Refer to Wood Paper here.

But Graph Database research has its beginnings already in the early 90s. During this time, numerous proposals came up, describing a semantic network to store data about the database. That was, because contemporary systems were failing to take into account the semantics of a database. (letzten Satz ändern) The Locial Data Model [3] was proposed, trying to combine the advantages of relational, hierarchical and network approaches in that they modeled databases as directed graphs, with leaves representing attributes and internal nodes posing as connections between the data.
Similar to that, the Functional Data Model [6] was proposed with the same goal, focusing specifically on providing a conceptually natural database interface [1].
During this period, most of the underlying theory of Graph Databases was created. It was most likely because of insufficient hardware support for big graphs that this research declined, only to be picked up again now, powered by improved hardware. Todays focus in Graph Theory research lies primarily on actual practical systems and on the theoretical analysis of graph query languages [1].

Especially practical implementations of Graph Database Theory have gained traction, as real world problems are more often than not interrelated - hence graphs are extremely useful in unterstanding the wide diversity of real-world datasets.[5]
The emerging of social networks have naturally helped graph database models come up, with big players like Twitter and their implementation FlockDB entering the field. In those social network situations, a so-called social graph can effortlessly model attributes of a person as well as relationships between people. While in traditional RDBMS the apparent friend-of-a-friend-problem would be solved with a join over all relevant tables, in graph database technology this can be achieved with a traversal, which is far more cost inexpensive (letzten Teilsatz ändern) [4].

Another meaningful topic today are recommender systems, where most work focusing on optimizing machine learning algorithms. But also in database theory, this specific context poses challenges. However again, the graph model gracefully maps item similarities and correlations between user behaviour [2].

These application fields bring very distinct workloads that require specific query languages to process. There are two different kinds of workload: in social network transactions low-latency online graphs are processed while for example link analysis algorithms evaluate high-throughput offline graphs [1]. Many query language proposals have come up recently, differing mainly in the underlying graph data structure/model and the functionality provided [7].

A deeper description of the theory behind graph databases will be given in subsection 2, aiming to connect the data model to its fields of application as well as comparing it to RDBMS. This comparison will be picked up in subsection 3, where a use case from the SQL world will be applied to Graph Database Theory, focusing in particular on Neo4j. Lastly, our findings will be stated in subsection 4 with a general conclusion.

1.2 Graph Database Theory

1.2.1 Description of data model and functionality

1.2.2 fields of application

1.2.3 CAP Theorem

1.2.4 GraphDB vs. RDBMS

einleitung, there is lots of literature on it and the main comparison points seem to be:

support

performance

in xx VIcknais et al. did a detailed performance survey. For that, mySQL version 5.1.42 and neo4j 1.0-b11hoch2 was compared. [**?**] The queries simulate the types that are

used in provenance systems. So for one dataset or node, you traverse the graph to find out their herkunft. Similarily if a data object/node is marked as incorrect, this information needs to be propagated to all is descendants/child nodes. The queries then were partitioned int structureal queries referencing the graph but not the paylos, an data queries using the payload data. integer payload data For the traversal queries, S0, S4, and S128, Neo4j was clearly faster, sometimes by a factor of 10. Though relational databases are not designed to do traversals. Neo4j also has a built in framework for traversals whereas mysql used a standard breadth first search. in the data queries mysql turned out to be more efficient, partly because neo4j uses Lucene for querying, which treats all payload as text, eventhough this payload was integer. Also with a text payload mysql outperformed neo4j. Though the researchers also took into account data closer to the real world that has spaces. Surprisinliy, at a large enough scale, neo4j outperformed mysql by a large amount.

flexibility

security

1.3 Implementation with Neo4j

1.3.1 Use Case from the SQL world

1.3.2 Installation

1.3.3 modelling

1.3.4 usage, query language

1.3.5 short conclusion, summary

1.4 Reflection

1.4.1 alternative popular graphdbs

1.4.2 conclusion

reflect on advantages disadvantages with implementation references

Bibliography

[1] Renzo Angles and Claudio Gutierrez. An introduction to graph data management. In *Graph Data Management*, 2018.

[2] Zan Huang, Wingyan Chung, Thian-Huat Ong, and Hsinchun Chen. A graph-based recommender system for digital library. In *Proceedings of the 2Nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02, pages 65–73, New York, NY, USA, 2002. ACM.

[3] Gabriel Kuper. The logical data model : a new approach to database logic /. 09 1985.

[4] Justin J. Miller. Graph database applications and concepts with neo4j. 2013.

[5] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases*. O'Reilly Media, Inc., 2013.

[6] David W. Shipman. The functional data model and the data language daplex. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, SIGMOD '79, pages 59–59, New York, NY, USA, 1979. ACM.

[7] Peter T. Wood. Query languages for graph databases. *SIGMOD Record*, 41:50–60, 2012.