Kafeng granted without fee provided that copies are

Business, Industries

Kafeng Wang NO. 201718017729056 Shenzhen College of Advanced Technology University of Chinese Academy of Sciences To respect a relational database is a successful business case, relational database systems are generally e? cient unless the data contains many relationships requiring joins of large tables.

The graph database is a good description of objective data. However, graphs are largely still handled in an ad hoc manner, in part because most data continues to reside in relational-like data management systems, and because graph analytics/querying typically forms a small portion of the overall analysis pipelines. How to make the graph database be used more widely? We compare relational databases, graph databases and combination of two databases, and draw the conclusion that the function of graph database should be integrated into the relational database. Users can only install one database platform, operate relational and graph databases with similar interfaces, and do not need to learn a lot of graph database knowledge. In this gradual way, the popularity of graph database is achieved. CCS CONCEPTS • Computer systems organization ? Database; Relational Database; Graph Database; • Database systems ? business model; KEYWORDS Relational database, graph database, commercial success.

ACM Reference Format: Kafeng Wang. 2017. How to make the graph database more successful? Parasitic to relational databases.

In Proceedings of ACM Woodstock conference (WOODSTOCK’2017). ACM, New York, NY, USA, 4 pages. https://doi. org/10. 1145/nnnnnnn. nnnnnnn1

INTRODUCTION A relational database is a digital database based on the relational model of data, as proposed by E.

F. Codd in 1970 3. A software system used to maintain relational databases is a relational database management system (RDBMS). Virtually all relational database systems use SQL (StructuredPermission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for pro? t or commercial advantage and that copies bear this notice and the full citation on the ? rst page. Copyrights for third-party components of this work must be honored.

For all other uses, contact the owner/author(s). WOODSTOCK’2017, July 2017, Landon, UK © 2017 Copyright held by the owner/author(s). ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. https://doi. org/10. 1145/nnnnnnn.

nnnnnnnQuery Language) for querying and maintaining the database. Relational model organizes data into one or more tables (or “ relations”) of columns and rows, with a unique key identifying each row. Rows are also called records or tuples. Columns are also called attributes.

Generally, each table/relation represents one “ entity type” (such as customer or product). The rows represent instances of that type of entity (such as “ Lee” or “ chair”) and the columns representing values attributed to that instance (such as address or price) 2. In computing, a graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data. A key concept of the

system is the graph (or edge or relationship), which directly relates data items in the store. The relationships allow data in the store to be linked together directly, and in many cases retrieved with one operation. In the past few years, users have begun to reach beyond the relational model. At some point, the cost-bene? t of adopting or creating a new storage model is almost always addressed.

Relational databases have been the workhorse of the database industry for decades, but many new searchintensive applications recently seem to be adopting alternative models 9. 9 is a comparison of the relative usefulness of the relational database MySQL and the graph database Neo4j to store graph data. The goal of 9 was to determine whether a traditional relational database system like MySQL, or a graph database, such as Neo4j, would be more e? ective as the underlying technology for the development of a data provenance system.

A graph is one of the fundamental data abstractions in computer science 9. Research into graph databases was popular in the early 1990s, but died out for a series of reasons including the surge of hypertext and XML research . With the rise of the Internet as a tool for the general public, data began to increase both in volume and interconnectedness.

The graph model was used to represent tremendous amounts of data more often than it had in the past. Traditional data stores were often capable of handling graph data. Yet, they were often neither designed to do so nor e? cient at it. There was a clear desire for a data store tailored to the needs of

graph data 9. This brings us to the question: what happens to the good old relational database systems in the context of graph analytics? do they become obsolete or are they still relevant for large scale graph analytics? do users now need to dumpWOODSTOCK'2017, July 2017, Landon, UK Kafeng Wangtheir data from the relational database to a graph database or can they perform graph analytics (with comparable performance) from within the relational engine? 5. Graph querying and analytics are becoming an increasingly important component of the arsenal of tools for extracting di? erent kinds of insights from data.

Despite an immense amount of work on those topics, graphs are largely still handled in an ad hoc manner, in part because most data continues to reside in relational-like data management systems, and because graph analytics/querying typically forms a small portion of the overall analysis pipelines 10. 2 RELATED WORK 6 investigate which of the databases (Relational or Graph) perform better for Organizational Mining under Process Mining. An intersection of Process Mining and Graph Databases can be accomplished by modelling these Organizational Mining metrics with graph databases. 6 implement SimilarTask and Sub-Contract algorithms on relational and NoSQL (graph-oriented) databases using only query language constructs. We conduct empirical analysis on a large real world data set to compare the performance of row-oriented database and NoSQL graph-oriented database. 6 benchmark performance factors like query execution time, CPU usage and disk/memory space usage for NoSQL graph-oriented database against row-oriented database. 10 describe an end-to-end graph

analysis framework, called GraphGen, that sits atop an RDBMS, and supports graph querying/analytics through.

Graph analytics is getting increasingly popular these days and there is a deluge of new systems for graph analytics. However, it is not clear how good or bad are the relational databases for graph analytics. In this talk, I will share our experiences with graph analytics on relational databases. Contrary to the popular belief, modern relational databases can have very good performance over graph analytics 5. Graph data management has received a lot of attention in recent times and a number of graph data management systems have been proposed recently.

These systems address roughly two kinds of query workloads: (i) low latency online graph query processing, e. g. social network transactions, and (ii) high throughput o? ine graph analytics, e. g.

PageRank computation 5. On the other hand, graph analytics di? ers from traditional data analytics in three ways: (i) iterative nature of queries, (ii) maintaining state across iterations, (iii) very large graphs 5. Graph analytics is becoming increasingly popular, driving many important business applications from social network analysis to machine learning. Since most graph data is collected in a relational database, it seems natural to attempt to perform graph analytics within the relational environment. However, SQL, the query language for relational databases, makes it di? cult to express graph analytics operations. This is because SQL requires programmers to think in terms of tables and joins, rather than the more natural

representation of graphs as collections of nodes and edges. As a result, evenrelatively simple graph operations can require very complex SQL queries 4.

3 RELATIONAL DATABASE The relational database management system was ? rst created in the 1970s. Since then, its popularity has skyrocketed, and it has become a primary data storage structure in both academic and commercial pursuits. Relational databases range from small, personal databases like Microsoft Access to large-scale database servers like Oracle, Microsoft SQL Server, and MySQL. This paper focuses on MySQL 9. Relational model organizes data into one or more tables (or “ relations”) of columns and rows, with a unique key identifying each row. Rows are also called records or tuples. Columns are also called attributes. Generally, each table/relation represents one “ entity type” (such as customer or product).

The rows represent instances of that type of entity (such as “ Lee” or “ chair”) and the columns representing values attributed to that instance (such as address or price). To be more speci? c, the user of such a database would begin at the leftmost category and choose the desired entry, then move to the right, gradually focusing in on a particular species, allowing the relational database to assist in the process and eliminate errors. Also, although less e? ciently, using such a system a person could enter a species the rightmost tier of the taxonomic hierarchy and let the database engine complete the identi? cation automatically. The Figure 1 shows the basic idea of a relational database it’s a strict relationship between tables of data able to produces results that other kinds of data orderings cannot 7. 4 GRAPH DATABASE

Graphs are everywhere from the web to social networks to communication topologies to online games to shopping, logistics, and transportation graph structured data is a part of our everyday life.

These applications all yield massive, evolving graphs that capture user activity, intent, and interactions. Analyzing these massive graph datasets is critical to deriving key insights for businesses and organizations. As a result, a plethora of systems for graph analytics have been proposed in the past few years 4.

Graph databases (GDB) are now a viable alternative to Relational Database Systems (RDBMS). Chemistry, biology, semantic web, social networking and recommendation engines are all examples of applications that can be represented in a much more natural form. Comparisons will be drawn between relational database systems (Oracle, MySQL) and graph databases (Neo4J) focusing on aspects such as data structures, data model features and query facilities. Additionally, several of the inherent and contemporary limitations of current o? erings comparing and contrasting graph vs.

relational database implementations will be explored 8. Graph databases really shine when working in areas where information about data interconnectivity or topology is important. In such applications the relations between data and the data itself are usually at the same level. Many companiesHow to make the graph database more successful? Parasitic to relational databases WOODSTOCK’2017, July 2017, Landon, UKFigure 1: Hierarchical relational databasehave developed in-house implementations in

order to cope with the need of graph database systems. Examples would be Facebooks Open Graph, Googles Knowledge Graph, Twitters FlockDB, any many more. The following are a few examples of systems that would bene? t greatly from graph database approach. RDBMS can be used for such systems but in a much more limiting and expensive way (expensive meaning processing power caused by recursive JOINs for friend of a friend type problems) 8. In computing, a graph database is a database that uses graph structures for semantic queries with nodes, edges and properties to represent and store data.

A key concept of the system is the graph (or edge or relationship), which directly relates data items in the store. The relationships allow data in the store to be linked together directly, and in many cases retrieved with one operation 1. This contrasts with relational databases that, with the aid of relational database management systems, permit managing the data without imposing implementation aspects like physical record chains; for example, links between data are stored in the database itself at the logical level, and relational algebra operations (e. g. join) can be used to manipulate and return related data in the relevant logical format. The execution of relational queries is possible with the aid of the database management systems at the physical level (e. g. using indexes), which permits boosting performance without modifying the logical structure of the database 1.

Graph databases are based on graph theory, and employ nodes, edges, and properties in Figure 2. 5 INTEGRATION Most users of graph systems run two

di? erent platforms. This is cumbersome because users must put in signi? cant e? ort to learn, develop, and maintain two separate environments.

Thus, a natural question to ask is how bad it would be to simply use a SQL-based relational system for both storing raw data and performing graph analytics 4. Figure 2: Graph databases employ nodes, properties, and edges. 4 present GRAPHiQL, an intuitive query language for graph analytics, which allows developers to reason in terms of nodes and edges. GRAPHiQL provides key graph constructs such as looping, recursion, and neighborhood operations. At runtime, GRAPHiQL compiles graph programs into e? cient SQL queries that can run on any relational database. We demonstrate the applicability of GRAPHiQL on several applications and compare the performance of GRAPHiQL queries with those of Apache Giraph (a popular vertex centric graph programming language). These purpose-built graphs systems are typically used in conjunction with a storage system such as a relational database.

A common usage pattern involves collecting raw data, e. g., about sales transactions, friend relationships, locations visited, etc in a relational table, exporting this relational data into a graph system, running some analysis, and thenWOODSTOCK'2017, July 2017, Landon, UK Kafeng Wang(possibly) reloading the data into the database for presentation or aggregation. Two systems are typically used because of the perception that 1) it is di? cult or impossible to express many graph analytics in SQL, 2) even if it is possible to express graph analytics in SQL, relational databases are ine? cient at the kind of iterative algorithms that comprise many graph analytics (e. g., page

rank, shortest paths), and 3) graphs systems lack features that make them usable as primary data stores, such as the ability to e? ciently perform in-place updates, provide transactions, or e? ciently subset or aggregate records 4. 4 presented GRAPHiQL, a graph intuitive query language for relational databases. GRAPHiQL allows developers to reason about more natural graph representations, i.

e. nodes and edges, rather than dealing with relational tables and joins. GRAPHiQL provides several key graph constructs, such as looping and neighborhoods. As a result, GRAPHiQL shifts the focus of the developer back to his analysis. At runtime, GRAPHiQL compiles user queries to optimized SQL queries, which can run on any relational engine. Our experiments show that GRAPHiQL queries run signi? cantly faster on a relational database compared to Giraph: 12x faster on small graph and 4: 3x faster on large graph. Essentially, GRAPHiQL combines the best of both worlds ease-of-use of procedural languages and good performance of declarative languages.

In our future work, we will build an optimizer to tune the performance of GRAPHiQL queries to the underlying RDBMS 4. Thus, there is an absence of a comparative study of modern relational databases in the context of graph analytics. In our e? ort, we are trying to benchmark and understand the performance of relational databases for graph analytics. In this paper, we report our ? ndings so far. Our key observations are: (1) parallel graph exploration can signi? cantly reduce the number of joins, a key bottleneck in relational databases, (2) relational databases can match or even signi? cantly outperform a graph database, and (3) column stores have superior

performance over graph analytics as well. Furthermore, apart from expressing graph analytics as SQL queries, we also consider exposing a more natural vertex-centric programming interface on top of a relational database. The vertexcentric interface means that programmers can now actually think in terms of a graph while still running their queries in a relational engine 5. 10 presented a uni? ed framework for extraction and analysis of graph-structured data stored in an RDBMS or similar structured storage engines.

Apart from providing an intuitive means of specifying various graph structures within the relational data without compromising on the ability to write complex graph algorithms, our high-level abstractions enable a wide variety of adaptive optimizations for conducting these types of analyses. There are many interesting and di? cult challenges here in terms of deciding where to execute graph queries/tasks, re-writing the SQL queries, making materialization decisions, and handling inaccuracies of the query optimizer and database statistics exposed by natural graph extraction and analysis tasks 10. Graph analytics is emerging as an important application area and several graph data management systems have been proposed recently. However, this requires users to switch to yet another data management system. In this paper, we revisited relational databases in the context of graph analytics. We implemented two popular graph queries, PageRank and Shortest Paths, on di? erent relational database engines.

Our results show that relational databases can match or even outperform a graph database. In particular, column-oriented database can perform really well. Furthermore, we showed vertex-centric programming interface, which

allows programmers to easily specify their graph queries, on top of a relational database. Our results demonstrate that graph analytics over relational databases is promising and we could do both traditional as well as graph data management within a single system 5.

6 CONCLUSION Since most graph data is collected in a relational database, it seems natural to attempt to perform graph analytics within the relational environment. According to GRAPHiQL and some experimental result, it is possible to integrate the graph database into relational data. However, relational databases still have the most market share, and most of data is collected in a relational model. If graph database want be used widely like relational databases, the best method is embedded into the latter.