Query Languages for Graph Databases

D

Kornelije Rabuzin
University of Zagreb, Croatia

INTRODUCTION

One can find more or less complicated definitions, but one could say that a database is an organized collection of data. It is assumed that a database is stored as files on a computer. Actually, any list of items on a paper could be called a collection of data as well, but today it is assumed that databases are stored digitally.

In order to create a text file, some text processor should be used (for example, Word, Wordpad, Notepad, etc.). In the same way, in order to create a database, some Database Management System (DBMS) should be used. Many systems are available, including Oracle, DB2, SQL Server, PostgreSQL, MySQL, Microsoft Access, Neo4j, MongoDB, etc. The majority of DBMSs are relational in nature. "Relational" means that the main structure that is used to store data is a relation, or as users usually call it, a table. One table contains zero or more rows. The idea of storing data into *relations* (tables) is quite old and relational databases have been used for over 40 years. Relations store data in a compact and organized way and redundancy and anomalies that are caused by redundancy are usually avoided. The relational model was introduced by dr. E. F. T. Codd. in the late 1960s and early 1970s.

In order to use a database management system, one has to be familiar with Structured Query Language (SQL). SQL is a standardized language that is supported by all major Relational DBMS software vendors (it does not belong to any vendor in particular). SQL contains many statements to work with data. The CREATE statement is used to create different objects in the database, as well as the database itself. The INSERT, UPDATE and DELETE statements are used for data manipulation purposes. In order to retrieve data from one or more tables, SELECT statement is used. Many other, complex statements are available as well, but for our purposes, we can skip the details. Each DBMS has some (graphical) client program (one or more) that can be used to write queries (in SQL) in order to work with the database (for example, phpMyAdmin, Workbench, Navicat, pgAdmin, SQL Server Management Studio, etc.). More on SQL can be found in (Rabuzin, 2011, 2014).

Relational databases are mature technology; many DBMSs are available, SQL is standardized, etc. But in recent years things have changed. Extremely large amounts of unstructured and/or semi structured data come from different sources and relational databases can not cope with such large amounts of data in a satisfactory manner. Furthermore, certain queries require too much time to execute and sometimes such behavior is not acceptable. Because of this new types of databases are being invented (and used) on a daily basis, especially NoSQL technologies, which are becoming more and more important.

The word NoSQL had different meanings in the past (for example, No to SQL), but today people usually mean "Not Only SQL". When one talks about NoSQL databases, several different types can be distinguished (some authors believe that XML databases should be included as well, as well as some other types, but it doesn't make much difference):

- Document oriented databases,
- Column oriented databases,
- Key Value databases,
- Graph databases.

DOI: 10.4018/978-1-5225-2255-3.ch176

Document oriented databases, like MongoDB, do not use tables to store data. Instead, they use collections that store documents. Since one document contains all the data that are relevant, foreign keys are not used (at least not in a form that one is used to in relational databases). Document databases can use references to link between documents, but usually all the referenced data could/should be included within the document. So basically, collections would correspond to tables and documents would correspond to rows. It is important to have in mind that the document schema is flexible and each document within the collection can have different schema.

Column oriented databases use tables as well, but data is organized and stored differently. Unlike relational databases that store together values that belong to a single row (usually several rows are stored within a single unit of storage), column oriented databases store together values (for different rows) that belong to the same column. This is easy to justify. Namely, when one poses a query, in most cases one doesn't need all the columns in the table. Instead, only a few columns are usually selected. In data warehouses dimension tables can have very large number of columns. So queries that are posed on column oriented databases should be faster because less data has to be read from the hard disk drive since column data is stored together (and not values that belong to a single row).

Key Value (KV) databases store values for defined keys. Values can be simple as well as complex. Key value databases may look trivial, but sometimes they can be very useful.

For now it is important to have in mind that graph databases represent an interesting type as well, and because of that they will be discussed later. Namely, the main idea of this article is to present languages for graph databases so they have to be explored more thoroughly.

Although relational databases are the most commonly used, NoSQL databases have great potential. However, one has to have in mind that other database types (http://db-engines.com/en/) do exists (object oriented databases, content store, event store, etc.), but they are not important for this research. So many different database types are available and they can be used for different purposes; one can store company data, images, sound, biometric data, etc. But the type that one plans to use depends on one's needs because all the database types mentioned above are more or less appropriate for different scenarios. For large amounts of interconnected data graph databases could represent the best choice whereas for large amounts of structured data relational databases may still represent the best choice. The idea behind NoSQL databases is not to suppress relational databases; it is just a question of the user's needs. So, relational databases now have an alternative, but which type to use depends on user's needs.

The main idea of this article is to explore several languages for graph databases. One of the languages that is in use is Cypher Query Language (CQL). However, it has some limitations as it will be shown later on. Because of that a new graphical language is proposed (implemented) that should be easier to learn and to use (at least for end-users). To resolve other limitations (recursion, views) that CQL has, it is shown how a logic programming language (Datalog) could be used as well.

So, the article is organized as follows. First, the background is described. Then CQL is explained. In the next two sections a new language for graph databases is proposed and implemented and then it is shown how a logic programing language could be used to write queries that are not supported in CQL. In the end future research is given and the conclusion is presented. Finally, references are listed.

BACKGROUND

Problems With SQL

When SQL was introduced, it was supposed to be simple. However, it turned out to be quite complex and end-users had problems with more complex queries. Although the idea was that SQL had to be simple so that end users could write the queries