Graph Databases - are they really so new

Maleković, Mirko; Rabuzin, Kornelije; Šestak, Martina

GRAPH DATABASES – ARE THEY REALLY SO NEW

[1]MIRKO MALEKOVIC, [2]KORNELIJE RABUZIN, [3]MARTINA SESTAK

[1,2,3]Faculty of Organization and Informatics Varaždin, University of Zagreb, Croatia
E-mail: [1]mirko.malekovic@foi.hr, [2]kornelije.rabuzin@foi.hr, [3]msestak2@foi.hr

Abstract- Even though relational databases have been (and still are) the most widely used database solutions for many years, there were other database solutions in use before the era of relational databases. One of those solutions were network databases with the underlying network model, whose characteristics will be presented in detail in this paper. The network model will be compared to the graph data model used by graph databases, the relatively new category of NoSQL databases with a growing share in the database market. The similarities and differences will be shown through the implementation of a simple database using network and graph data model.

Keywords- Network data model, graph data model, nodes, relationships, records, sets.

I. INTRODUCTION

The relational databases were introduced by E. G. Codd in his research papers in 1970s. In those papers he discussed relational model as the underlying data model for the relational databases, which was based on relational algebra and first-order predicate logic. Since then relational databases gained popularity in the database industry and kept their advantage for many years.

However, the idea of storing and manipulating data in an organized way was present before relational databases were introduced. Several options were available to fulfill that purpose [6]:

- **ISAM (Indexed Sequential Access Model) databases (Reflections on Computing)**

This category of databases is often referred to as file-based databases and they represent the early implementation of the database management system concept. The ISAM acronym represents a technique for managing databases using fixed length table records, indexes and file system locks [7]. ISAM databases were developed and used by IBM during the 1960s and their purpose was to provide the means of sequentially accessing database records in order to perform simple data processing operations (inserting and deleting records, searching for records) [2].

An ISAM database consists of several ISAM files stored on disk, where each file is stored within one disk cylinder of a predefined size. Due to this limitation the structure of an ISAM database is static and inflexible when it comes to inserting new records, since files must be reblocked and records within them rewritten when their assigned disk space is full [2]. This problem becomes even more challenging as the database size grows, because file reblocking is a costly operation requiring time, performance and extra disk space.

Except the aforementioned problem, other major drawbacks of this database type was their lack of support for ad hoc queries and the cross referencing issue when retrieving records from multiple files [2].

- **Hierarchical databases**

This type of databases is based on the hierarchical data model, whose structure can be represented as a layered tree in which one data table represents the root of that tree (top layer), and the others represent tree branches (nodes) emerging from the root of the tree [3]. Database tables are physically connected via pointers with a parent-child relationship in a 1:N ratio, i.e. one parent table can have one or more child tables, whereas one child table can have only one parent table.

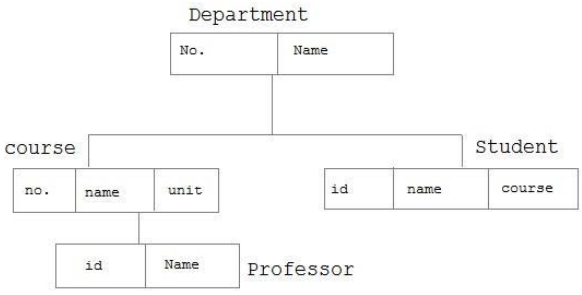

Fig.1. Hierarchical data model example [http://www.studytonight.com/dbms/images/hierarchical-model.jpg]

Since all relationships are established on a physical level, hierarchical databases perform very well when it comes to retrieving data from the database. However, in order to efficiently query the hierarchical database, one must be familiar with the entire database structure, because each query starts at the root table element and continues travelling down the tree structure until it reaches the target element.

Additionally, a common problem, which cannot be ignored, is the problem of data redundancy, which often occurs when implementing more complex, many-to-many (M:N) relationships between tables. For instance, given the database model shown in **Fig.1**, in order to represent a many-to-many relationship between the Course and Professor table (one course can be held by one or more professors, and one professor can teach one or more courses),

one would have to store duplicate data about that relationship in both Course and Professor tables for every professor teaching a given course and every course held by a given professor. The problem of data redundancy and the lack of support for complex relationships tried to be solved by introducing the next database type: network databases.

- **Network databases**

Network databases and their underlying network model were introduced soon after the hierarchical databases and represented the more progressive option for implementing complex relationships between tables as opposed to their database predecessor. Their characteristics will be discussed in detail in the following chapter.

Hierarchical and network databases have been used for some years, but after E. G. Codd introduced the relational model, the number of their users decreased significantly. One of the reasons for this change was the simplicity of the relational model, which could easily be understood by both users and database designers. The relational model introduced that the users are not required to be familiar with details regarding the physical implementation of the database, i.e. all relational model complexity is hidden from the users. Since then, the era of relational databases began and is still ongoing, because the relational database management systems still have the largest share in the database market.

However, during the last decade the enormous growth of data size and complexity led to the development of new generation of databases assembled under the name of NoSQL databases. One of the database types in this category are graph databases.

Graph databases represent the next generation of databases with the ability to model complex relationships between database entities [8]. Their characteristics will be further discussed in the chapters to come. The strong support for complex relationships provided by the underlying graph data model has made them a very popular solution during the last few years for modelling social networks, fraud detection and many other problems.

The properties of graph databases should inevitably remind us of some properties of network databases, so in this paper we plan to further discuss their similarities and differences followed by an implementation of a sample database in both selected network and graph DBMSs. First we will give an overview of network databases, then we will discuss graph databases and compare them to network databases.

II. NETWORK DATABASES

As we already discussed, network databases and their underlying network data model were developed in the late 1960s as an improved alternative to the hierarchical databases and as an attempt to impose a database standard at that time [1]. Similar to the hierarchical model, the network model is also structured as an inverted tree with owner-member relationship.

The network data model stores all data in record types and their fields stored in data files on disk. Records are connected with set relationships between them, which are also stored in data files [9]. Each owner element stores a physical link to its member elements, which is a more flexible solution for data access as opposed to the hierarchical data access always starting at the tree root element.

The network data model was developed in two variants [2]:

1. Simple network data model, which supports one (owner) to many (member) relationships between records represents the next evolutionary step from hierarchical databases, because it introduced the possibility that one child element has multiple parent elements. Network databases based on this model allow set relationships to be implemented by directly connecting records via pointers or by using indexes. However, many-to-many relationships can be implemented by introducing a composite record between two records, which enables them to be connected via two one-to-many set relationships.
2. Complex network data model supports direct many-to-many relationships (without unnecessary composite records). However, with this data model there is no possibility of storing relationship data, because there is no composite record, i.e. no place to store that data.

The network data model of our sample database is shown in **Fig. 2**. The model consists of five records and four set relationships. Most records are connected with one-to-many set relationships (for instance, an author wrote one or more books), except the many-to-many relationship between User and Book records. This problem is solved by adding a composite record called Borrowing between those records, so in the end we built a simple network data model.

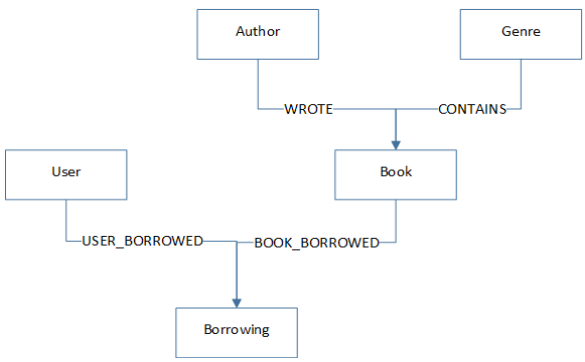

Fig.2. Network data model of sample database

In order to implement this network data model, one can use Raima Database Manager (RDM) developed by Raima Inc., a product which provides APIs, tools and other utilities for database management. It supports both relational and network data model.

An RDM database consists of the following elements [9]:

- Database dictionary, which stores the information about how database is organized and its content
- Data files, which contain one or more database record types
- Key files, which contain records' key fields and are used for indexing
- Vardata files, which contain data about variable length fields

Data can be modeled with its Data

Definition Language (DDL) in a simple text file with the following sample structure:

```
database booksnetwork
{
data file datfile = "booksnetworkdb.dat" contains author, user, book, genre, borrowing;
key file[1024] keyfile1 = "books.k01" contains author_id;
record author
{
        unique  key     int     author_id;
                        varchar a_firstname[50];
                        varchar a_lastname[100];
}
record  book
{
        unique  key     int     book_id;
                        varchar title[100];
                        int     year_published;
}
set wrote
{
        order   last;
        owner   author;
        member  book;
}
}
```

III. GRAPH DATABASES

Graph databases are a database type that is gaining the most research interest lately, as they can be used to model and store data in different problem domains in which other databases fail to ensure persistent and real time performances. For example, social network analysis is a scenario in which nodes and relationships between nodes represent an excellent choice to implement "friend of a friend (of a friend) concept". At the same time, relational databases could have problems with finding friends on the second and upper levels, as queries become more and more complex as well as time-consuming.

Graph databases are commonly based on the property graph data model, in which data is stored in nodes with a specific label and relationships of a specific type between nodes, and each node and relationship can have its attributes. Since each node contains a physical link to its neighbors it is connected with (this concept is called index-free adjacency), the graph data model supports both one-to-many and many-to-many relationships, which can be implemented directly without creating additional composite nodes or the risk of losing node or relationship data.

The property graph data model of our sample database is shown in **Fig. 3** and consists of four nodes (Author, Book, User, Genre) and three types of relationships (WROTE, BORROWED, PART_OF).

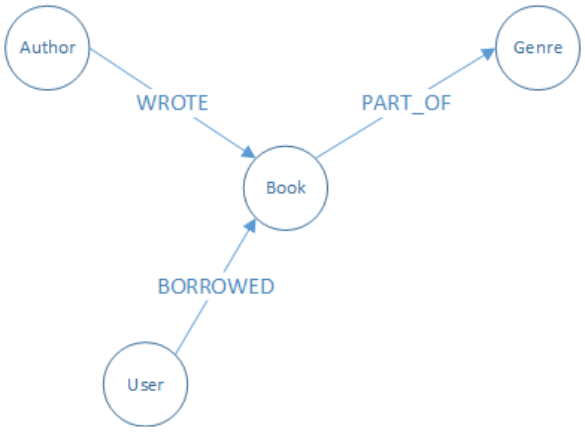

Fig.3. Graph data model of sample database

Basically, two main query languages are used for graph databases, Cypher and Gremlin. They both have some advantages and disadvantages. Cypher is a SQL-like query language based on graph pattern matching, so it is widely used and comprehensive for users and database designers. On the other side, Gremlin is a graph traversal and expressive language which is able to perform more complex graph traversals, because the entire process is divided into a chain of operations (called steps) whose results, unlike Cypher's, do not require additional value conversions [4]. Their performances in retrieving data are quite similar and they both outperform SQL.

We used the Neo4j database to implement the graph data model we created. Neo4j DBMS (Database Management System) is the representative of the graph databases group because it is currently the most widely used graph DBMS on the database market. Even though Neo4j supports queries written in both Cypher and Gremlin language, officially its standard query language is Cypher. For instance, to create the new WROTE relationship between Author and Book nodes in our database the following Cypher query was executed:

CREATE (a:Author{Firstname:'William', Lastname:'Shakespeare'})-[w:WROTE]->(b:Book {Title:'Romeo and Juliet', YearPublished:'1595'})')

The result of this query is shown in **Fig. 4**.

Fig.4. Result of creating a relationship between two nodes in Neo4j database

Since the graph databases are a relatively new category of NoSQL databases, there are still many areas and issues to be researched and solved. One of these areas are integrity constraints. At this point Neo4j provides support only for the node uniqueness constraint, which is used to ensure that there are no duplicate nodes in the database. On the other side, Gremlin does not provide support for any kind of database constraints.

IV. NETWORK AND GRAPH DATABASES – A COMPARISON

Unlike hierarchical databases that have problems with many to many relationships, such relationships can easily be represented in graph databases. They can also be represented in network database by using the complex network data model at the expense of completeness of information. However, it is recommended to use one-to-many relationships by creating composite records.

Both network and graph databases are used to represent complex relationships between entities, which is one of their advantages, as is their good performance with query execution due to the fact that all database entities are connected directly to each other via pointers.

However, it is worth accenting the price for achieving this advantage. Clearly, the physical independence is ruined or reduced. When so often using the word 'implementation' regarding any data model, and the users need to know the implementation details, we have the right to say 'what we have here is nothing of the physical independence. All of these problems were the main Codd's motivation for developing the relational model.

The concept of nodes and relationships in graph databases is very similar to records and set relationships in network databases.

To build our sample database we used the RDM solution for network databases and Neo4j graph database. In the first case, the database structure is created in the beginning in only one step by defining its structure with the syntax of the DDL inside a simple text file and compiling that file. We can say that the database schema is fixed and must be known in advance, because each change in the schema requires the text file with its definition to be entirely recompiled. On the other side, when it comes to graph databases, the database structure is built dynamically by creating concrete nodes and relationships in the database by executing Cypher or Gremlin queries. Graph databases as a category of NoSQL databases are schema-less, so the query languages have no built-in support (commands) for managing the database structure.

Additionally, network databases provide no support for ad hoc queries, so the RDM C++ API generated a fixed set of methods to be executed against the database (creating and retrieving records and their keys etc.). Conversely, graph query languages offer a wide set of clauses, which can be used to build complex or custom queries in real time.

As already discussed, network databases have not developed the concept of integrity constraints and have no support for ensuring the database integrity, while the same issue still needs to be further developed and researched in graph databases.

CONCLUSIONS

In this paper we have made a comparison of network databases used before relational databases and graph databases, a category of NoSQL databases with a growing popularity on the database market. As we have discussed, network and graph databases (their underlying data models) share some similarities (importance of connection between entities, concept of nodes and relationships, ability to model complex relationships, direct links between nodes), but they also have some differences (unlike network databases, graph databases can implement many-to-many relationships without the third, "dummy", node, and support ad hoc queries etc.).

All these facts say that the network model, in the right sense, did not exist. Some authors tried to make something in that area, but only partially. The same thing, for now, can be said for the graph data model. Therefore, if we wanted to be correct, we would have said 'the incomplete network data model' and the 'incomplete graph data model'. All the time we have to have in mind that the physical independence is reduced or destroyed.

The implementation approach also varies, as we have shown when implementing a sample database with both technologies.

REFERENCES

[1] C. Coronel, and S. Morns, "Database Systems: Design, Implementation, & Management", Course Technology, vol. 11, pp. 39-45, 2014.
[2] J. L. Harrington, "Relational Database Design Clearly Explained", Morgan Kaufmann, vol. 2, pp. 50-71, 2002.
[3] M. J. Hernandez, "Database Design for Mere Mortals", Addison-Wesley Professional, vol. 2, pp. , 2003.

[4] F. Holzschuher, and R. Peinl, “Performance optimization for querying social network data”, 2014, Workshop proceedings of the EDBT/ICDT 2014 Joint ConferenceAthens, URL: http://ceur-ws.org/Vol-1133/paper-38.pdf [accessed August 17, 2016]

[5] G. O’Regan, “Introduction to the History of Computing: A Computing History Primer”, Springer, vol. 1, pp. 276-278, 2016.

[6] A. Tatnall, “Reflections on the History of Computing”, Springer-Verlag Berlin Heidelberg, vol.1, pp.119-120, 2012.

[7] R. L. Bague, “Explore the differences between ISAM and relational databases”, 2004, URL: http://www.techrepublic.com/article/explore-the-differences-between-isam-and-relational-databases/ [accessed August 17, 2016]

[8] Neo4j documentation, “From Relational to Neo4j”, URL: https://neo4j.com/developer/graph-db-vs-rdbms/ [accessed August 17, 2016]

[9] W. Warren, “Raima Database Manager v12.0 Architecture and Features”, 2013, URL: http://raima.com/wp-content/uploads/RDM-v12-Technical-Whitepaper-.pdf [accessed August 17, 2016]

[10] C. Bachman, "The Programmer as Navigator", 1973, ACM Turing Award lecture, Communications of the ACM, vol. 16, no. 11, November 1973. (pdf)

[11] C. Bachman, “Implementation Techniques for Data Structure Sets”, 1974, Data Base Management Systems.

[12] K. Rabuzin, M. Šestak, and M. Novak (in press), “Integrity constraints in graph databases”, 2016, 27thCentral European Conference on Information and Intelligent Systems Proceedings.

★★★