# Alternative Approaches To Data Storing And Processing

**Speacker's**
**Nikolai**
**Paniavin**
**NRU "MPEI "**

**Authors:**

Mihkel M. Maran

Nikolai A. Paniavin

Ilia A. Poliushkin

National Research University "MPEI"

# Alternative Approaches To Data Storing And Processing



**Abstract**

Most of data processing applications store data using database management system (DBMS). The most widely used DBEs belong to relational. In these engines, data is stored in a number of tables. One of the main relational DBMS interface technologies is Structured Query Language (SQL), but there are alternative approaches, like Query By Example (QBE). Since relational databases have poor performance on some types of data, there have appeared other approaches. They in unite are called “No-SQL databases”, although this approaches are very different from each other. Graph-based and document-based DBMS belongs to them. In this paper, we describe common implementations of relational and No-SQL DBMS and compare their capabilities using a practical example. The differences between them are how they designed, what types of data support, how they store information.

Relational databases store structured data that typically represent real-world objects. It can be information about a person, grouped in tables, the format of which set at the design stage of the store.

Non-relational databases are structured differently. For example, document-oriented databases store information in the form of hierarchical data structures. We can talk about objects with an arbitrary set of attributes. The fact that a relational database divides into several interrelated tables can be stored in a non-relational database as an integral entity.

The internal structure of various database management systems affects the features of working with them. For example, non-relational databases are better scalable.

# Alternative Approaches To Data Storing And Processing



**What kind of database choose in solution?**

In general, Databases divide into two categories:

- **Relational** ( MS SQL Server, IBM DB\2, Oracle, MySQL etc.)
- **Non-Relational** (*Graph* Neo4j, *Document-oriented* MongoDB and others)

The final project may contain not only one kind of database, also SQL and No-SQL but it needs more data control.

# Alternative Approaches To Data Storing And Processing



**History of SQL**

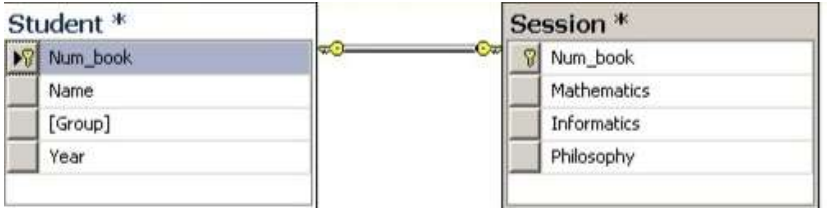
The direction of relational databases was born in the 1970s at IBM.

Dr. E.F. Codd proposed the idea of a new relational data model and specific language.

The research project was named «System R». New the project turned out successful and became widespread from the 1980s till nowdays.

SQL cannot be fully assigned to programming languages. It contains only the data access operators stored in the database, and there are no operators that control program progress.

SQL is based on relational algebra operations and instead of individual table records allows to work with sets (relations whose operands are tables).



# Alternative Approaches To Data Storing And Processing



**Another Language Projects**

*Alternative query language in MS Access:*

*Query By Example (QBE)* for relational databases was developed in parallel with SQL.

This variant was proposed by Moshe M. Zloof.

*SQL may be replaced?*

As a subsequent replacement the developers C.J. Date and H. Darwen in the book « Foundation for Future Database Systems "The Third Manifesto» was proposed a variant of the more relational language «Tutorial D», which supports a much more hierarchical query structure. Still, due to the lack of full recursive data structures, the utility of this capability still minimal.

# Alternative Approaches To Data Storing And Processing



**What about graphs in relational DataBases?**

In relational databases graphs can be represented, for example, as *Closure Table* or as *Nested Sets*.

The essence of this design *Closure Table* is that relationships between entities are stored in a separate table, whereas the main table contains only data from the entities themselves.

It is worth noting that the *Closure Table* method requires more space in the database to store trees than any other method. But Adding new elements to the end of a deep tree hierarchy is less effective than inserting elements near the root of the tree.

*Nested sets* involve assigning two additional keys to each node in the tree, the left key and the right key. To fill in these keys, we will have to completely bypass the entire tree by visiting each node twice. As a result, the selection from the tree will occur fairly quickly. On the other hand, changing the structure requires recalculating all keys in the nodes following the angle being changed.

In databases that do not support recursive queries (such as MySQL), tree selection is faster than if it were done using a stored procedure.

# Alternative Approaches To Data Storing And Processing



**Non- Relational Graph DBMS**

Neo4j is a NoSQL open source graph DBMS that was first implemented in Scala and Java languages in 2003 by the company Neo Technology. The source stored in the public domain on the GitHub along with the full documentation and description. Currently, Neo4j is the most common graph DBMS. Neo4j uses its storage format, which was specifically designed to store information as a graphical structure. As uses examples can be maps, transport routes, semantic networks etc.

The DBMS uses its own query language — Cypher.

**Big data processing**

When processing a graph, only a part of it must be placed in RAM, so that sufficiently large graphs can be processed.

# Alternative Approaches To Data Storing And Processing

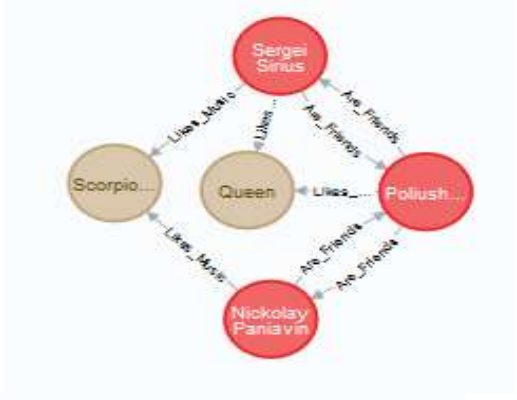


**Example of implementation in Neo4j in semantic network.**

(part of code)

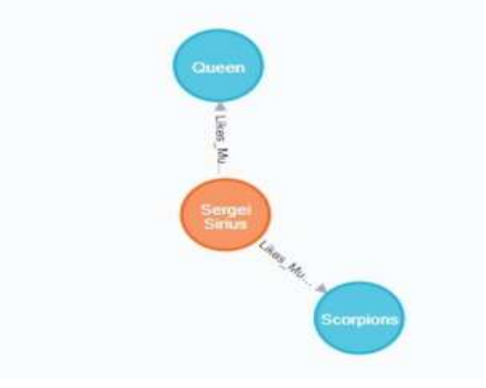
```
CREATE (User1:Person {name:'Nickolay Paniavin', born:1996})
CREATE (MusicGroup1: Musician {name:'Scorpions', founded:1965})
CREATE (User1)-[:Are_Friends]->( User2)

MATCH (n) RETURN (n)
```




Let 's get out all the musical groups Sergei likes.

```
MATCH(person{name :"Sergei Sirius" } ) - [: Likes_Music] -> (ms: Musician)
RETURN person, ms
```



# Alternative Approaches To Data Storing And Processing



**How to find the path?**

(part of code)

CREATE (a)-[:ROAD {cost:5}]->(b)

For example let`s find with *Dijkstra algorithm* all the shortest paths from the starting point to all the others.

MATCH (start:Loc {title: 'A'}), (end: Loc {title: 'F'})

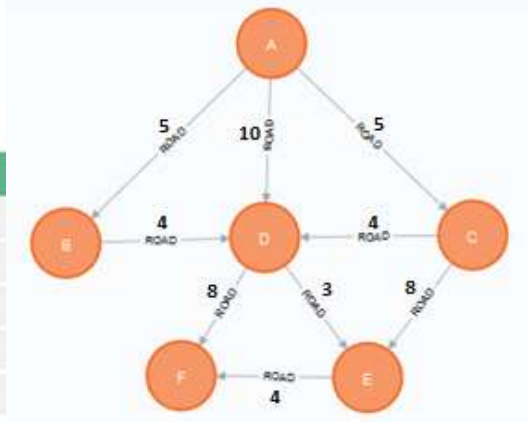
CALL apoc.algo.dijkstra(start, end, 'distance', 'value')

YIELD Name, Cost

RETURN Name, Cost

**Results**

| Name | Cost |
|---|---|
| A | 0 |
| C | 5 |
| D | 9 |
| E | 12 |
| F | 16 |



# Alternative Approaches To Data Storing And Processing



**Comparing relational and non relational graph solutions.**

For test there were created 100 000 vertexes and 1 000 000 edges.

OS Windows 10, Intel Core i5 3.2 GHz, 8 Gb DDR3,

1 Tb 7200 rpm hard drive,

Windows Server 2008 x64, MS SQL Server 17.3, Neo4j Desktop 4.0

| ----- | | Time, s | |
|---|---|---|---|
| Generated database | RAM, Mb | Data import | search for all common neighbors for two vertexes |
| MS SQL Server | 1594 | 13 | 39 |
| Neo4j | 3126 | 24 | 87 |

# Alternative Approaches To Data Storing And Processing



**Document-oriented DBMS**

Document-oriented DBMS provide more opportunities than relational ones. In such systems, the unit of data presentation is a document that has a set of individual attributes. This document can be presented in JSON format (for example, as in MongoDB).

Documentary systems usually do not support ACID semantics, but they vary considerably among themselves at the level of data consistency, atomic operations, and methods of controlling parallel access to documents.

Documentary DBMS has a flexible data model, which in some cases is more convenient than a fixed circuit, and is better combined with object-oriented programming, reducing the interlayer between DBMS and the programming language.

# Alternative Approaches To Data Storing And Processing



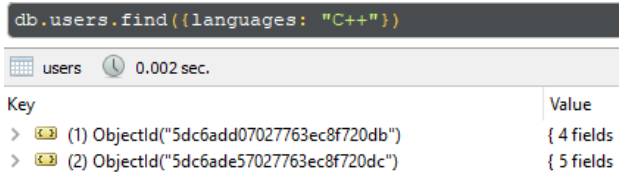
**Simple queries in MongoDB**

db.users.insertOne({ "name": "Nick",  "age": 23, languages: ["C#", "C++"]} )

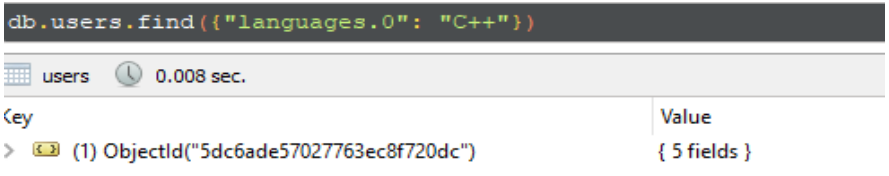
db.users.insertOne({ "name": "Ilya","age": 23, languages: ["C++", "Prolog","Lisp"],prefMusic: "Queen"} )

Get out everyone who knows the language C++

db.users.find({languages: ["C++"]})




Let us bring out those who have language C++ in the first place :



# Alternative Approaches To Data Storing And Processing



**Conclusion**

The part of this work, alternative approaches to data processing were considered. The study assumes supplementing relational databases and not insists their replacement.

All data, which in one way or another presented in the form of graphs, can be entered into the Neo4j and with it is possible to work. You can also fill in any other database, relational or non-relational, with this data. The advantage of a graph DB represented on systems with a large number of nodes and links.

Despite the advantages, the graph database has a significant disadvantage - the amount of disk space consumed. The programmer faces a new task of choosing between a quick solution and the organization of available memory, which is limited.

Document-Oriented DBMS allow to find documents or parts of them on demand. Documents can be grouped into collections. Therefore, they can be considered a distant analogue of relational DBMS tables, but collections can contain other collections. Because documents in a collection can be arbitrary, it is best to combine documents with a similar structure into a collection for more efficient indexing.

If the data structure is strictly fixed and will not change in the process, then you should give preference to a relational database.

Choosing a DBMS, it is necessary to proceed from the initial data and sets of requirements for the solution of the task.

INFORINO 2020

Moscow, Russia
14-17 April, 2020

# Thank you for attention!

**Speacker's contacts:**


**Nikolai A. Paniavin**

**NRU "MPEI"**

**e-mail PaniavinNA@mpei.ru**


IEEE

МЭИ NATIONAL RESEARCH UNIVERSITY

CROC