Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 40 (2019) 170–177

Transportation Research Procedia

www.elsevier.com/locate/procedia

13th International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM 2019), High Tatras, Novy Smokovec – Grand Hotel Bellevue, Slovak Republic, May 29-31, 2019

Comparison of query performance in relational a non-relation databases

Roman Čerešňák*, Michal Kvet

University of Zilina, Univerzitná 8215/1, Žilina 010 26, Žilina

Abstract

Data are the most important treasure of the company. They form the source for the analysis, evaluation and decision making. Complex data management requires sophisticated storage structures and access principles. This paper deals with the database storage architectures, principles and differences, describes the main manipulation operations– insert, update, delete and select. For the evaluation, relational databases, such as Oracle, MySQL, and MS SQL has been selected, which is consecutively compared in the performance section to the non-relational oriented databases – Mongo, Redis, GraphQl and Cassandra, as well. After the theoretical background definition, the experiment section is proposed. For this purpose, a database of the train connections as a combination of all the train stops in Slovakia is designed. For the first performance evaluation model, the only small data portion is used examining the speed of the operation and loading operation into the table. The second model is far more complex, stores a huge number of records. It evaluates the time aspect, as well as storage size demands. Results are summarized in conclusion showing recommendations for the individual database type with emphasis on the speed of the system and limitations.

Peer-review under responsibility of the scientific committee of the 13th International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM 2019).

Keywords: Sql; NoSql; Transformation; Train connection

* Corresponding author. Tel.: 0912 458 469;
E-mail address: roman.ceresnak@fri.uniza.sk

2352-1465
Peer-review under responsibility of the scientific committee of the 13th International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM 2019).
10.1016/j.trpro.2019.07.027

1. Introduction

Nowadays, companies such as Google, Facebook, Twitter and others generate data with various applications. These applications moreover generate huge amount of unstructured data, and that makes it difficult to handle the data. Data are very important for companies what also requires analysis and storing of them into some structures. Relation database can only work with structured data, so there is a need of NoSql Database management system, which can work with no-structured data. Slovak railways have the large database of values representing a connection between cities and villages throughout Slovakia. There were only relational databases like Oracle, MySql, and Microsoft Sql, when this database was created.

1.1. Databases

Database (or data base, rarely data bank) is set of structured data or information stored in a computer system, so a computer program or a person can use a search language to retrieve this information. The information thus obtained can be used in a decision-making process. The computer program used for data management and queries is referred to as SRBD (Database Management System).Computer scientists can classify the database management system according to the database models they support. First mentioned databases (relational) became dominant in the 1980s. They support the using of rows and columns in a series of tables. The most significant majority of them also use Sql for writing and querying data. Non-relational databases have become popular in the 2000s, called NoSql because they use different query language.

1.2. SQL vs NoSQL

The most significant difference between these concepts is that SQL database is relational and contains foreign keys. On the contrary, the NoSql database is irreversible and so does not define the relationships. Table 1 shows the different features of Sql and NoSql databases[1].

Tab 1. Difference between Sql and NoSql databases

Property	Sql	NoSql
The way of storing the data	Tables	Documents, key value,
Organization of data	A predefined scheme	A dynamic scheme
Scalability(increase in performance)	Vertical(larger ram, stronger processor)	Horizontal(more servers, instances)
Query language	Standardized Sql	Own query language
Data intercourse	Foreign keys	Nested documents
Security	Transactions, consistency, isolation	Does not exist

At first glance, Sql may seem to be outweighing the benefits of NoSql, but it may not be true automatically. NoSql does not offer any security features to us, but more it reads and writes data. Therefore, it is used for BigData applications, where terabyte data are expected. NoSql is also an ideal choice if we need to implement a complex full-text search engine that takes similar words, mimicry or other grammar rules into account. NoSql too solves the problem, when we do not know database schematics first. Conversely, we need to address the data consistency on an application side, and that is the downside. Another drawback is that NoSql database does not support transactional processing[2].

There are far more types of NoSQL databases than just those, dealing with the full-text search. They differ by the style of storage as well as by use. As noted in the table, we know NoSql database document, key-value or graph. When it comes to the full-text search, we are talking about document databases. We use the key-value database, because when working with the cache, we need to store the data somewhere. Among the most famous we can find Redis or Memcached. Social networks are shown in the chart database first[3].

1.3. Sql features

Tab 2. Difference features in Sql databases

Product/Feature	Microsoft Sql	Oracle RDBMS	MySql
Typical Applications	SharePoint, SCOM, SCCM, WSUS	OBI, SAP	Joomla, WordPress, MyBB, phpBB, Drupal, many open-source
Operating System	Windows Server, Windows Client	Windows, Unix, Linux	Windows, Unix, Linux, Mac and many more
Drivers	ODBC, JDBC, ADO.NET, OLEDB, Microsoft Visual Studio	-	ODBC, JDBC, ADO.NET, Microsoft Visual Studio
Licensing	Closed-source, proprietary	Closed-source, proprietary	Open-source GNU-GPL
Standardized	ANSI-SQL		ANSI-SQL
Transactions	Yes	Yes	Using InnoDB storage engine
Partial Index	Yes	Yes	No
Schema	Yes	Yes	No
Failure			Using MyISAM: UPS required, uninterrupted operation assumed
Graphical Management Tools	Yes: Management Studio and BI Studio	Enterprise Manager	MySQL Workbench. Toad.
Computed columns	Yes	Yes	No
Active/Active clustering	Read-only on the second node	Yes (RAC)	No
Maintenance Plan Wizard	Yes		No
Job Scheduling	Yes (Agent)	Yes (Oracle Scheduler)	v5.1 (Event Scheduler)
History	First release in 1989, Based on Ingress (1974) / Sybase (1987)	1979	1995

2. Sql databases

2.1. Oracle vs. SQL Server vs. MySQL

Apart from the features pointed out in the above table 2, there are some other points on the basis of which we can compare these three databases. We listed them below:

- Typing feature is available in all of three databases, and they support XML and secondary indexes.
- The common APIs for each of the database are ADO.NET, JDBC, and ODBC. While SQL Server also supports OLE DB and TDS, Oracle also supports ODP.NET and OCI.
- For Oracle, the long list of supported programming language includes Delphi, Lisp, Java, C++, C#, Ruby, PHP, Visual Basic and JavaScript also supported by MySQL and SQL Server. Oracle supports many other programming languages not supported by the rest of the two databases. These are Scala, Fortran, and the other languages.
- SQL Server uses transact SQL and >NET for server-side scripting, while Oracle uses PL/SQL languages.
- Triggers are available in all of three databases.
- All of them support concurrency, durability, in-memory capability and foreign key concepts.
- They support ACID properties as far as transaction concepts are concerned.
- In Oracle and MySQL, it can have Master-master and Master-slave replications for replication strategy, and it depends on an edition in SQL Server.

There are three other factors on the basis of which we can compare these databases. A language that is used by them as the main feature of any RDBMS is the language used to execute queries and how it impacts the database

performance. Although all three databases use SQL or structured query language, SQL Server also uses T-SQL developed by Sybase (extension of SQL). Oracle uses PL/SQL or procedural programming language.

Both languages have different syntaxes and capabilities. The main difference between these languages lies in the way they handle stored procedures, variables, and built-in functions. In Oracle PL/SQL, the procedures can also be grouped into packages that cannot be done in SQL Server. Therefore, it can be a bit more complex and much more powerful. T-SQL is quite more straightforward to implement, but on the other hand, MySQL uses the light version of T-SQL and a combination of the procedural languages.

Another factor or feature for comparing these databases is a transaction control. A transaction is a set or a group of more than one instruction, executed altogether as a single unit. SQL Server and Oracle differ mostly here because in SQL Server each command is committed and executed individually and can't be rolled back if any error occurs. It can be used "BEGIN TRANSACTION" command for statement grouping and for committing you can use "COMMIT" command. ROLLBACK command is used to discard any change done in a transaction block.

Each new connection of the database is treated like a new transaction in Oracle. For all queries and command execution, the changes are made only in memory remaining in the cache. Nothing is committed without any explicit COMMIT statement. The changes made in the database remain in the cache and are initially made into memory. When a COMMIT is performed, just by a new instruction, a new transaction is starting. In this way, a more significant transaction feature is offered to the developers and error control can also be done quickly.

In the case of MySQL, transactions are handled by InnoDB. InnoDB is a storage engine, and it is available by default in MySQL. It also provides ACID-complaint features like foreign key support and transaction handling.

3. NoSQL databases

Developers need solutions aligning with the realities of the modern data and the iterative software development practices. In recent years, NoSQL databases have emerged as an answer to the limitations of the traditional relational databases and also, as a provision of performance, scalability, and flexibility, required in modern applications[4].

The most aspects of these NoSQL technologies vary greatly and have little in common, except for the fact that they do not use a relational data model. There are four types of NoSQL database management systems:

- Key-Value Store – It has Big Hash Table of keys & values {Example- Riak, Amazon S3 (Dynamo)}
- Document-based Store- It stores the documents made up of tagged elements. {Example- CouchDB}
- Column-based Store- Each storage block contains the data from only one column, {Example- HBase, Cassandra}
- Graph-based-A network database that uses edges and nodes to represent and store the data. {Example- Neo4J}[5]

3.1. Key-Value Store

A schema-less format of the key-value database like Riak is just about what is necessary for storage needs. The key can be synthetic or auto-generated while the value can be String, JSON, BLOB (binary large object), etc.

The key-value type basically, is using a hash table where a unique key and a pointer to a particular item of the data are. A bucket is a logical group of the keys – but they do not physically group the data. There can be identical keys in the different buckets.

Performance is enhanced to a significant degree because of the cache mechanisms that accompany the mappings. It is required to know both, the key and the bucket, to read a value, because the real key is a hash (Bucket+ Key).

There is no complexity around the Key-Value Store database model as it can be implemented in a breeze. It is not an ideal method if it is only searching to update a part of the value or query the database.

It is tried and reflected back on CAP theorem, it becomes quite clear, that key-value stores are great around Availability and Partition aspects, but lack in Consistency. The key can be synthetic or auto-generated, while the value can be String, JSON, BLOB (binary large object).

This key/value type database allows clients to read and to write values using the key as follows:

- Get(key), returns the value associated with the provided key.
- Put(key, value), associates the value with the key.
- Multi-get(key1, key2, .., keyN), returns the list of values associated with the list of keys.
- Delete(key), removes the entry for the key from the data store.

While the Key-value type database seems helpful in some cases, it has some weaknesses as well. At first, the model will not provide any traditional database capabilities (like atomicity of transactions or consistency, when multiple transactions are executed simultaneously). The application itself must provide such capabilities[6].

Secondly, as the volume of the data increases, maintaining unique values like keys may become more difficult; addressing this issue requires an introduction of some complexity in the generating character strings that will remain unique among the huge set of keys.

3.2. Document-based Store

The data, that are the collection of the key-value pairs, are compressed as a document store quite similar to the key-value store, but the only difference is, that the values stored (referred to as "documents") provide some structure and encoding of the managed data. XML, JSON (JavaScript Object Notation), BSON (which is a binary encoding of JSON objects) are some common standard encodings.

The following example shows the data values collected as a "document" representing the names of specific retail stores. Note, while all three examples represent locations, the representative models are different.

{ officeName:"Europalace", {Street: "Vysokoskolakov, City:"Zilina", Pincode:"95802"} }
{ officeName:"Mirage", {Street:"Mestska", Block:"B, Ist Floor", City: "Topolcany", Pincode: 59682"} }
{officeName:"Eurovea", {Latitude:"40.748328", Longitude:"-73.985560"} }

Examples of Document Database:

- MongoDB
- Couchbase

3.3. Column-Based Databases

Column-based store database:

In column-oriented NoSQL database, data are stored in the cells grouped in the columns of the data rather than as rows of data. Columns are logically grouped into the column families. The column families can contain a virtually unlimited number of columns that can be created at runtime or the definition of a schema. Reading and writing are done by using the columns rather than the rows.

In comparison, the most relational DBMS store data in the rows. The benefit of storing data in the columns is a fast search/ access and data aggregation. Relational databases store a single row as a continuous disk entry. The different rows are stored in the different places on the disk, while the Columnar databases store all the cells corresponding to the column as the continuous disk entry, thus searches/access faster.

For example: Querying the titles from a bunch of a million articles will be a painstaking task while using relational databases as it will go over each location, is getting off the item titles. On the other hand, the title of all items can be obtained with just a one disk access.

Data Model:

- ColumnFamily: ColumnFamily is a single structure that can group Columns and SuperColumns with ease.
- Key: a permanent name of the record. The Keys have a different number of columns so that the database can be scaled irregularly.
- The best-known examples are Google's BigTable, and HBase & Cassandra inspired from BigTable.

BigTable, for instance, is a compressed high performance and proprietary data storage system owned by Google. It has the following attributes:

- Sparse – some cells can be empty,
- Distributed – data is partitioned across many hosts,
- Multidimensional – more than one dimension,
- Sorted – maps are generally not sorted, but this one is.

3.4. Graph-Based NoSql Database

In Graph Base NoSQL Database, it is not possible to find a rigid format of SQL or the tables and a columns representation. A flexible graphical representation is used instead, perfect to address scalability concerns. Graph structures are used with edges, nodes, and properties, providing index-free adjacency. The data can be easily transformed from one model to another, using Graph Base NoSQL database [7].

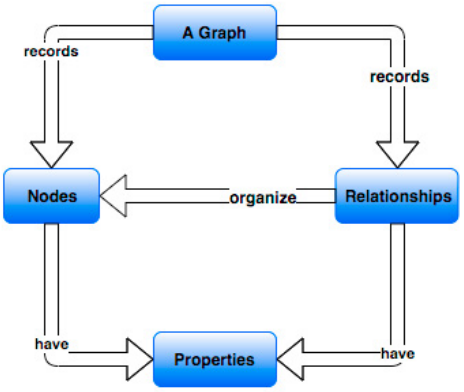

Fig 1. Organization o Graph-based database

These databases use the edges and the nodes to represent and store the data.

- Some relationships with one another organize these nodes, what is represented by the edges between the nodes.
- Both, the nodes and the relationships, have some defined properties

Some features of the graph-based database are explained on the basis of the example below:

Labelled, directed, attributed multi-graph: The graph contains the nodes labeled properly with some properties. These nodes have some relationship with one another what is shown by the directional edges. For example: in the following representation, “Alice knows Bob” it is shown by the edge, that that also has some properties.

While the relational database models can replicate the graphical ones, the edge would require a join which is a costly proposition.

4. Experiments

Tab 3. Query performance of databases with 10 000 records in milliseconds

Type/Operation	Oracle	MySql	MsSql	Mongo	Redis	GraphQL	Cassandra
Insert	0.076	0,093	0.093	0.005	0.009	0.008	0.011
Update	0.077	0,058	0.073	0.008	0.013	0.007	0.013
Delete	0.059	0.025	0.093	0.01	0.021	0.017	0.018
Select	0.025	0,093	0.062	0.009	0.016	0.010	0.014

All operation that have been tracked and recorded in the previous table will be tracked even when creating an index above the requested attributes. Sometimes, when the user visits the webpage http://www.slovakrail.sk/. In this website, person defines the place, where they want to get to, where they want to get out and when the travel will be. The page gives a result with many connections. If the connection does not exist page shows a connection with more stop transition. An index will be created above this data for the faster search.

All previous tests were conducted at the size of the range of 10 000 records. The speed and the effectiveness of NoSql database are shown in the higher number of records, for example, 10 000 and more. For these quality testing need, created fake records and consequently tested them.

We have 90 000 records for the latest testing, in all tables. We decided to use the range scan in the table, which is very needed to speed up the queries. We documented all the results in the spreadsheet.

All the experiments were performed on MacBook pro year 2015 8GB RAM and I5 processor.

Tab 4. Query performance of database with 100 000 records in milliseconds

Type/Operation	Oracle	MySql	MsSql	Mongo	Redis	GraphQL	Cassandra
Insert	0.091	0.038	0.093	0.005	0.010	0.008	0.011
Update	0.092	0.068	0.075	0.009	0.013	0.012	0.014
Delete	0.119	0.047	0.171	0.015	0.021	0.018	0.019
Select	0.062	0.067	0.060	0.009	0.015	0.011	0.014

5. The result of experiment graphically

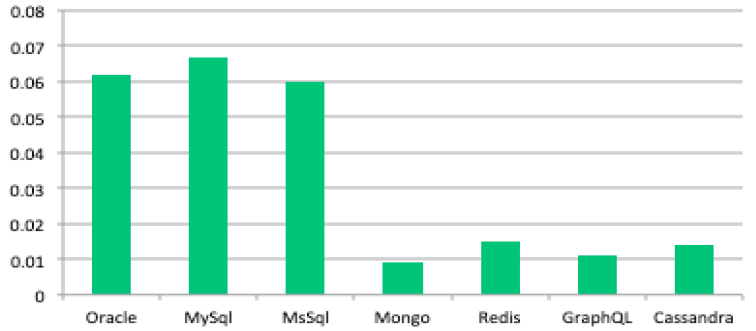

Fig 2. Query performance in milliseconds

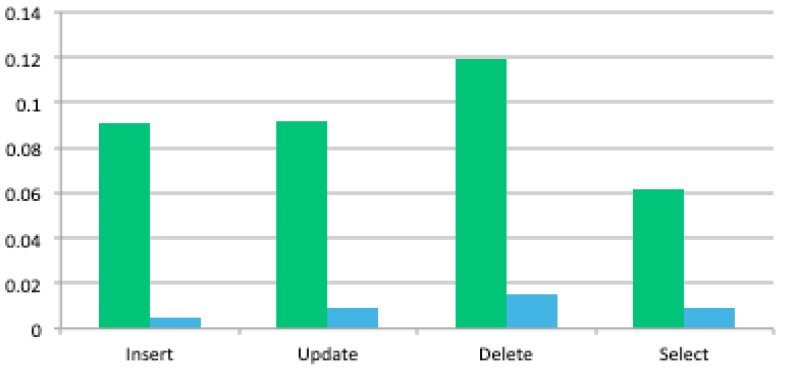

Fig 3. Query performance for Oracle and MongoDB

In figure 2. result time for a select query in relational databases(Oracle, MySql, and MsSql) and nonrelational databases (Mongo, Redis, Cassandra a GraphQL) are shown. This result was expected. It is caused by way of data stored. In NoSql such as Redis, Mongo, Cassandra or GraphQL data are stored in the form of flat collections where this data are duplicated again and again, and a single piece of data is hardly ever partitioned off, but instead it is stored

in the form of an entity. It means, reading or writing operations to a single entity has become more accessible and faster [8].

On the other hand in relation databases data need to be broken into several small logical tables to avoid duplicity and redundancy. For this needs help normalization. Normalization provide managing data correctly and efficiently. Divide data to several related tables involved with normalization hampers the performance of data processing in relation databases using Sql and therefore time needed for receive value from relation databases is much higher than from nonrelation as we can see in Image 2.

The right column shows us how query performance in relational database Oracle. The left column represents nonrelation database Mongo. In comparison, a ratio between databases for receive data is almost 1:3 for select performance, 1:6 for the delete operation, 1: 9 for update operation and 1:15 for insert operation.

6. Conclusion

When are compared two groups, the difference between relational and nonrelated databases in query time is not huge, as it can be seen in table 3. The difference was adjusted after the range scan implementation on a key attribute. At this point, the database has consisted of 10 000 records after inserting 500 000 records to the database, the speed and the effectiveness of nonrelated database. Relation database can still be used in the case of the size of a train station in Slovakia. In the case of countries with a higher number of train stations and stops such as Germany, Netherlands, and others, using the non-relation databases would be more effective.

Acknowledgement

This publication is the result of the project implementation: Centre of excellence for systems and services of intelligent transport II., ITMS 26220120050 supported by the Research & Development Operational Programme funded by the ERDF.

References

[1] Lourenço, J., Cabral, B., Carreiro, P., Vieira, M. and Bernardino, J. (2015). Choosing the right NoSql database for the job: a quality attribute evaluation. Journal of Big Data, 2(1).

[2] Mehmood, N., Culmone, R. and Mostarda, L. (2017). Modeling temporal aspects of sensor data for MongoDB NoSql database. Journal of Big Data, 4(1).

[3] Sánchez-de-Madariaga, R., Muñoz, A., Lozano-Rubí, R., Serrano-Balazote, P., Castro, A., Moreno, O. and Pascual, M. (2017). Examining database persistence of ISO/EN 13606 standardized electronic health record extracts: relational vs. NoSql approaches. BMC Medical Informatics and Decision Making, 17(1).

[4] Mehmood, N., Culmone, R. and Mostarda, L. (2017). Modeling temporal aspects of sensor data for MongoDB NoSql database. Journal of Big Data, 4(1).

[5] Srdja Bjeladinovic (2018) A fresh approach for hybrid Sql/NoSql database design based on data structuredness, Enterprise Information Systems, 12:8-9, 1202-1220

[6] R. Deari, X. Zenuni, J. Ajdari, F. Ismaili and B. Raufi, "Analysis And Comparision of Document-Based Databases with Relational Databases: MongoDB vs MySQL," 2018 International Conference on Information Technologies (InfoTech), Varna, 2018, pp. 1-4

[7] S. Li, H. Jiang and M. Shi, "Redis-based web server cluster session maintaining technology," 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, 2017, pp. 3065-3069.

[8] Celesti, M. Fazio, A. Romano, A. Bramanti, P. Bramanti and M. Villari, "An OAIS-Based Hospital Information System on the Cloud: Analysis of a NoSql Column-Oriented Approach," in IEEE Journal of Biomedical and Health Informatics, vol. 22, no. 3, pp. 912-918