SQL vs NoSQL: what to choose for OLAP

Every time choosing database technology you have two main ways: non-relational databases vs relational. In other words there is a NoSQL vs SQL battle. Both options have their advantages, as well as several key features that should be borne in mind when choosing.

And even after years of hibernation do not hurry to write SQL off, especially when we are talking about OLAP. So let's arrange a fair battle and choose a real leader. Here we go.

What are we talking about?

Before comparing NoSQL vs SQL server performance, scalability and other features let's figure out: what are we talking about?

SQL server is primarily a RDBMS (relational database). RDBMS stores structured data that represents real-world objects. For instance personal info, wherein it is grouped in tables. Tables' format is specified at the repository design stage.

NoSQL databases are arranged differently. Information in document-oriented databases stored in the hierarchical form. We can talk about objects with an arbitrary set of attributes..

Of course, there is no one perfect database for everyone. Thus, the problem solution for some companies is to use NoSQL and SQL together. However, if choosing one then which?

What does SQL suggest?

Since the very appearance, NoSQL databases captured everyone's attention. However, despite high speed, scalability and other benefits, in some cases SQL is more efficient. To be precise:

1. The need for database compliance with the requirements of ACID (Atomicity, Consistency, Isolation, Durability). It helps to reduce the possibility of unexpected system behavior and ensures the integrity of the database. Such feature is achieved by a hard determination of exactly how transactions interact with the database. This approach differs from the one used in NoSQL databases, which prioritize flexibility and speed, not 100% data integrity.
2. The data you work with is structured, and the structure is not subject to frequent changes. If your organization is not in the stage of exponential growth, there is probably no convincing reason to use a database that allows you to be fairly free with data types and is aimed at processing huge amounts of information.

What are the top SQL vendors?

The best are: MySQL, Sybase, Microsoft Azure, PostgreSQL.

What does NoSQL suggest?

If you have a doubt the database may become a bottleneck of a certain project based on working with large amounts of information, it is worth looking in the direction of NoSQL databases, which allow something that relational databases cannot do:

1. Large amounts of unstructured information storage. NoSQL database does not impose restrictions on the types of stored data. Moreover, if necessary, in the course of work, you can add new data types.
2. Use of cloud computing and storage. Cloud storage is a great solution, but it requires that data can be easily distributed across multiple servers to ensure scalability. Using local equipment for testing and development, and then transferring the system to the cloud, where it works, is exactly what the NoSQL database was created for.
3. Quick development. If you are developing a system using agile methods, the use of a relational database can slow down. NoSQL databases do not need the same amount of preparatory actions that are usually needed for relational databases.

What are the top NoSQL vendors?

The best are: MongoBD, Hbase, Apache’s Cassandra DB.

To the very beginning

To better understand the nature of both technologies we'd like to rewind time back and touch the roots. So, the story of SQL starts in the late 1970s in IBM Research when language appeared after relational algebra. IBM specialists decided to design a new query language that will be more accessible to users without mathematics and programming background. Thus, the first prototype of the IBM System R relational DBMS was implemented. Later, this language was used in many commercial DBMS and, due to its wide distribution, it gradually became the de facto standard for data manipulation languages in relational DBMS.

In 1989 and 1992 the next SQL versions SQL2 and SQL3 appeared. And everything would be fine further if the main competitor had not came into play. Exactly, we are moving to NoSQL emergence.

What was the push to NoSQL appearance? The Internet spread. After the very coming, the Internet continued to grow and grow. Thus, the software community found that relational databases of the time could not cope with this new workload and it started non-relational systems uprising. The companies started developing their own distributed non-relational systems such as Cassandra, Dynamo, Hadoop, BigTable and others**.**

Interesting fact: in June 2009, Johan Oscarsson organized a meeting in San Francisco, to discuss new developments in the IT data storage and processing market. The main impetus for the meeting were new open-source products like BigTable and Dynamo. For a bright signboard for a meeting it was required to find a succinct and concise term that would fit perfectly into the Twitter hashtag. One of these terms was suggested by Eric Evans from RackSpace - "NoSQL".

It was planned to use the term only for one meeting ,but it so happened that it spread throughout the global network like viral advertising and became the de facto name of a whole trend in the IT industry.

NoSQL became a new hot and desirable until the first problems appeared. The main difficulty was: every NoSQL database offered its own query language as a result you have more languages to learn. This and other problems put the SQL language back into the game. And thus, NoSQL and SQL are still finding out who is cooler.

NoSQL vs SQL difference

Rivalry, battles, but we still did not figured out the clear and principle differences between systems. So, from words to actions:

Feature	SQL	NoSQL
Language	Relational databases use Structured Query Language. On the one hand, this opens up great opportunities for development: SQL is one of the most	Uses different languages. Non-relational databases, in turn, offer a dynamic data structure that can be stored

	flexible and common query languages, so its choice minimizes a number of risks, and it will be especially useful if you have to work with complex queries. SQL, on the other hand, has a number of limitations as a subsequent change in the data structure can be disastrous for the entire system.	in several ways: column-oriented, document-oriented, in the form of graphs, or based on key-value pairs. This flexibility means the following: You can create documents without setting their structure in advance; Each document may have its own structure; Each database can have its own syntax; You can add fields directly while working with data.
Structure	Data is presented in the form of tables.	Data is presented in the form of documents, key-value pairs, graphs or wide-column repositories.
Scalability	SQL databases are vertically scalable, thus, you can increase the load on a single server by increasing the power of central processors, RAM, or storage systems.	NoSQL databases are horizontally scalable. This means that you can increase traffic by distributing it or adding more servers to your database.
Indexing	In both SQL and NoSQL databases, indexes serve the same purpose — to speed up and optimize data retrieval. But how exactly they work is different due to the different database architectures and peculiarities of information storage in the database. In SQL, indices are represented as B-trees, which reflect the hierarchical	In NoSQL databases, they point to documents, or to parts of documents, between which there is basically no relationship.

structure of relational data.

Facts: advantages and disadvantages of SQL vs NoSQL

Now we see four clear differences between servers and ready to move on to the next round: SQL vs NoSQL advantages and disadvantages.

SQL pros and cons:

Pros

Time-proven: SQL server is a highly developed DBMS, which means there is a large community around it, many examples and high reliability;

Compatibility: SQL is available on all major platforms, including Linux, Windows, Mac, BSD and Solaris.

Payback: SQL are usually open source databases that are freely available;

Replicability: SQL database can be distributed among several nodes, thus reducing the load and improving the scalability and availability of the application.

Cons

Complexity: although SQL was conceived as an end-user tool, in the end the majority of them became complex and are more programmer's tool;

Deviations from standards: despite the availability of the international standard, many companies involved in the development of DBMS, make changes to the SQL language used in the developed database, thereby departing from the standard. Thus, there appear SQL-specific dialects specific to each particular DBMS.

NoSQL pros and cons:

Pros

Dynamic scheme: this DBMS allows working flexibly with the data scheme without the need to change the data itself;

Easy to manage: the DBMS does not need a separate database administrator. Due to sufficient usability, it can be easily used by both developers and system administrators;

Speed: high performance when operating simple queries;

Flexibility: it is possible to add fields or columns without harming existing data, their structure and DBMS performance.

Cons

Limited built-in query language capacity: almost all the query languages and API methods of the NoSQL repositories were created based on certain SQL functions, but they have much less functionality;

NoSQL solutions do not require defining a database schema before starting work: therefore, during the development process, you may encounter unforeseen difficulties that may lead to the abandonment of this

NoSQL solution.

So what is better?

And who is the winner? Let's sum it up:

SQL databases are ideal for projects with the following features:

- Logical data requirements can be defined in advance;
- When data integrity is very important;
- A need for well-established technology based on well-established standards, using

which one can count on extensive experience of developers and technical support.

You should choose NoSQL databases if in your products:

- Data requirements are vague, uncertain, or evolving with project development;
- The goal of the project can be adjusted over time, with the possibility of starting development immediately;
- One of the main database requirements is data processing speed and scalability.

And of course, you always can use those systems together and have everything you need at once. However, if we are talking about such system as for example Ranet OLAP what is better then?

Your option will be definitely SQL. The deal is using NoSQL for OLAP - is not for what this server intended. NoSQL databases provide very high read and write throughput for large objects. As a rule, they are not intended to be used in atomic-level data which OLAP system requires. Though, OLAP with NoSQL use is possible if we are talking about MongoBD and hBase. In other cases, SQL wins.

Business Intelligence Glossary

Most helpful

OLAP overview

OLAP operations

OLAP cubes

OLAP in data mining

BI technologies overview

Recent Posts

Faster analysis with OLAP, basic technical terms and how they are related to each other

Development prospects for Ranet OLAP Technology using Artificial Intelligence

OLAP: outdated technology or a modern trend in the development of business intelligence platforms?

OLAP technology in BI solutions

Xafari Framework hot price

The summertime is coming soon and bringing Great Summer Discounts!

What is the Galaktikasoft team’s planning for Xafari Framework and Ranet OLAP development in 2023

MDX report template

Share the Link

PARTNERS

RESOURCES

Ranet OLAP Blog
Xafari Framework Blog
Xafari Documentation
Ranet OLAP Documentation
Support (Tickets)
Development services

OUR RATING

5

based on 3 reviews

sitemap