WHITE PAPER

SQL vs. NoSQL vs. NewSQL

"SQL" is used both as the name of a language and as a type of database. SQL the language is a structured query language designed for managing data in relational database management systems (RDBMS). Relational database management systems are often called SQL databases since they use the SQL language. Since the mid-1980s, SQL has been a standard for querying and managing RDBMS data sets.

SQL Databases

Relational databases continue to provide the foundation for the world's transactions. Think about all the credit card transactions being handled by mainframes and large UNIX servers in the data centers of financial services companies.

SQL features

- Rely on relational tables
- Utilize defined data schema
- Reduce redundancy through normalization
- Support JOIN functionality
- Engineered for data integrity
- Traditionally scale up, not out
- Rely on a simple, standardized query language
- Near universal in adoption

NoSQL Databases

While a SQL database is a defined, concrete concept, NoSQL is not. There is enormous variation in technologies that fall under the NoSQL category (though they generally share some characteristics). Thus, NoSQL is a term used for a broad group of data management technologies which vary in features and functionality but which try to solve some key issues of SQL databases.

NoSQL features

- High performance writes and massive scalability
- Do not require a defined schema for writing data
- Primarily eventually-consistent by default
- Support wide range of modern programming languages and tools

VOLTDB

NoSQL Basics

NoSQL solutions emerged as a reaction to frustration with the cost and inflexibility of legacy RDBMS products like Oracle and IBM DB2, which use SQL as a query language. The original NoSQL systems were built for scale, unstructured data, and did not use relational (table-based) schema. Most early NoSQL solutions dumped SQL as a query language, although that tide has turned and proprietary SQL languages are now offered by many of the NoSQL vendors — they discovered that most of their potential users were business analysts who knew and loved SQL.

Implementing a data management platform that can handle "web-scale" loads — millions of simultaneous users — with "engaging" performance (millisecond response times) can be difficult to do and even more difficult to afford with existing legacy products.

Some NoSQL solutions, therefore, focus on consistency models, i.e. availability: the database is always available to accept new data and can always provide an answer when queried, even if that data is not transactionally consistent, i.e., the most recent version written. Examples include DynamoDB and Cassandra.

Other NoSQL databases are designed to be flexible, and focus on data models: they don't enforce a rigid or consistent schema across stored data. These 'document stores' expand upon the traditional key-value store by replacing the values with JSON-structured documents, each able to contain sub-keys and sub-values, arrays of value, or hierarchies of all of the above. There are many document and column-oriented NoSQL databases, e.g. MongoDB, HBase and Couchbase.

There is another class of NoSQL solutions covering products such as Lucene, Solr and ElasticSearch that offer text and document indexing functions, useful, for example, to implement real-time search as users enter terms.

And there is another category, called Graph databases, with products like Neo4J, Titan and Tagged which organize data by relationships instead of by row or document, enabling powerful traversal and graph query capabilities.

The Promise of NewSQL

NewSQL is a term coined by 451 Group analyst Matt Aslett to describe a new group of databases that share much of the functionality of traditional SQL relational databases, while offering some of the benefits of NoSQL technologies.

NewSQL systems offer the best of both worlds: the relational data model and ACID transactional consistency of traditional operational databases; the familiarity and interactivity of SQL; and the scalability and speed of NoSQL. Some offer stronger consistency guarantees than are available with NoSQL solutions, although others limit this to 'tunable' consistency and thus aren't fully ACID-compliant.

Of course, there are differences among NewSQL solutions. SAP HANA handles modest transactional workloads, but without the benefit of native clustering. NuoDB is a cluster-first SQL database with a focus on cloud deployments, but throughput is poor. MemSQL is useful for clustered analytics but its tunable consistency falls down on ACID transactions. Both NuoDB and MemSQL have different types of nodes, that is, different nodes for compute and storage, so data movement and synchronization, especially around transactions, can be issues for them

Four Use Cases Where NoSQL Databases Fall Short

Now that we have noted the basic differences between NoSQL and NewSQL, let's break the differences down specifically by looking at what developers really care about:

- Which problems can I solve with NoSQL?
- Equally important, where is NoSQL a bad fit?
- Where do different approaches — NoSQL and NewSQL — show their strengths?

Don't get us wrong — NoSQL databases are great for lots of workloads and applications, but in four areas the weaknesses of NoSQL are pronounced.

This section will discuss those four areas, and identify use cases where NoSQL technology might not be the best choice.

Scalability

NoSQL offerings began to attract attention when they were implemented to address the scalability requirements of web-native companies such as Google, Facebook and Twitter. These organizations were dealing with vast inflows of unstructured data from myriad sources: web searches, mobile devices, user status updates, streams of comments.

In these use cases, the most important consideration was scalability: the database had to scale out massively. The relational schema and challenge of scaling traditional SQL databases was seen as restrictive, and when cost was considered in either dollars or development effort of scaling legacy RDBMSs, using these systems was seen as prohibitive.

The key feature NoSQL systems brought to the development community was the ability to scale applications on inexpensive commodity hardware. If your use case requires horizontal scale-out for unlimited data feeds, NoSQL may be the right choice — unless you want to take action on the data in real-time.

NewSQL Scalability

While legacy relational database systems offer scaling options — at an often-significant cost — NewSQL systems were designed to address this same scalability challenge, all while maintaining the transactionality and SQL-standard interactivity of legacy RDBMSs.

One example is an in-memory, massively parallel, SQL relational database that scales out linearly on inexpensive commodity hardware. Like NoSQL solutions, NewSQL databases should be cloud-friendly and able to scale to meet the demands of web-scale applications. These systems should be designed for low latency high read/write performance, using a shared-nothing, cluster-native, cloud-friendly architecture. They need to provide high availability and fault tolerance, as well as geographic redundancy. Spoiler alert: VoltDB does all these things. With VoltDB, you can build transactional, database-oriented applications against data feeds that were previously limited to stream processing systems.

Availability

NoSQL systems are designed for availability, CAP-theorem style. Availability means the database is always responsive, even in the event of network partitions. NoSQL systems by design prioritize availability over strong consistency (i.e., always correct, exact answers). These systems will return an answer, even if the answer is not the most current at that given instance in time.

This design decision, made famous by Apache Cassandra, was based on the belief that it is less important to be immediately correct than it is to always return a response and/or accept new data.

For example, if your Twitter follower list is not displayed as exactly correct at that moment, that is ok. It is better to display something that is not exactly correct but is not the same for all clients rather than to display nothing at all. Eventually the answer will be correct for all clients.

For many applications, eventual consistency is acceptable. However, when you need to make immediate decisions, or transactions on exact answers, you will need a different solution.

Eventual consistency and thus NoSQL solutions are a poor fit for the following types of applications:

- Let a mobile user's call go through
- Keeping track of (counting) and allocating finite, scarce resources
- Making financial decisions

NewSQL Availability

NewSQL systems favor consistency over availability. A NewSQL system will return the same exact answer to all clients – it won't return different answers. This allows your applications to make decisions, for example on money, airplane seat assignments, inventory, etc., and to know there won't be conflicts.

VoltDB, for example, is a strongly-consistent NewSQL database. If you have a use case that requires speed, scalability and always-correct, ACID-compliant transactions, VoltDB is pretty much the only game in town. No NoSQL offerings can make the same assertion.

Consistency (e.g., ACID-compliant transactions, correct answers)

NoSQL systems were architected for availability (see above). That choice meant they couldn't deliver ACID-based, strong consistency. With CAP, three pillars are described — Consistency, Availability, and Partition tolerance. Brewer's Theorem (aka the CAP theorem) states you can have two, but not all three of these properties.

Thus, NoSQL systems are AP — they are Available, and Partition-tolerant. This makes NoSQL a poor choice for applications or use cases where strong consistency is required:

- Billing
- Rights management, operations support (telco);
- Last dollar (adtech, gaming);
- SLA management, session management
- Trade verification, fraud detection, bid & offer management
- Sensor management.

NewSQL Consistency

Here's a more practical way to think about CAP: In the face of network partitions, you can't always have both perfect consistency and 100% availability. Plan accordingly.

Strongly consistent systems such as VoltDB are CP. They choose Consistency (C) over Availability (A). They also provide Partition-tolerance (P).

Of all NewSQL solutions, VoltDB is the most robust and resilient in the face of many failure scenarios. We've been validated independently by Kyle Kingsbury's Jepsen testing. We've seen customers with production clusters with uptime measured in years.

As you might expect from the above, VoltDB excels in applications or use cases where strong consistency is required:

- Billing
- Rights management, operations support (telco)
- Last dollar (telco, adtech, gaming)
- SLA management, session management (telco)
- Trade verification, fraud detection, bid and offer management (financial exchanges),
- Sensor management (IoT).

Fast Transactional Applications

Today, modern, request-response style applications happen at high volume. While NoSQL solutions often provide datastore speed, they cannot provide strongly consistent transactions at scale.

Applications requiring scalable transactions include:

- Allowing a mobile call to connect while verifying the user's balance
- Placing a trade at the best offer price
- Presenting a mobile ad to potentially thousands of users without blowing through an advertiser's ad budget
- Managing tight SLAs for telco providers
- Detecting a fraudulent card swipe before the transaction is approved

NoSQL databases are generally not the right choice for these types of applications.

These events occur millions of times a day, even millions of times per hour (or even second) all over the world. Businesses in telco, financial services, online gaming, adtech and other industries need to be able to accommodate the varying volume and velocity of these events. They need a scalable, transactionally-consistent solution.

NewSQL Fast Transactions

ACID transactions are a requirement for request-response applications — and because NewSQL systems are relational ACID-based databases, they support ACID transactions. Coupling scalability, strong consistency and the ability to transact on data at scale, NewSQL systems provide the capabilities for modern applications.

Both NoSQL and NewSQL technologies provide datastores on which highly scalable applications can be built. NoSQL datastores are a great choice for applications where availability — getting a response — is valued more highly than getting a consistent, correct response at all times. NewSQL systems provide applications with scalability as well, and also offer strong consistency and transactional interaction, favoring consistency over availability in failure scenarios.

While nearly all NoSQL solutions deliver scalability, VoltDB delivers on scalability and adds strongly consistent transactions; further, few can NoSQL solutions match VoltDB's combination of blazing speed, transactional consistency and scalability.

Conclusion

VoltDB, the most mature NewSQL system, is a cloud-ready SQL operational database. VoltDB combines real-time analytics on inbound data feeds with strong ACID transactions, native clustering, and Hadoop ecosystem support. This allows VoltDB to be the system-of-record for data-intensive applications, while offering an integrated high-throughput, low-latency ingestion engine. It's a great choice for use cases where very high performance and predictably low latency are critical as well as where accurate counting/accounting is important, such as in policy enforcement, personalization, fraud/anomaly detection, and other request-response style fast-decisioning and fast data pipeline applications.

About VoltDB

VoltDB is an in-memory transactional database for modern applications that require the ability to manage data at unprecedented scale and volume, with 100% accuracy.

Unlike OLTP, Big Data, and NoSQL offerings that force users to compromise, only VoltDB supports all three modern application data requirements:

Millions — VoltDB processes relentless volumes of data from users, devices and sources.

Milliseconds — VoltDB ingests, analyzes, and acts on data instantaneously.

100% — Data managed by VoltDB is always accurate, all the time, for all decisions.

Telcos, Financial Services, Ad Tech, Gaming and other companies (including iOT technologies) use VoltDB to modernize revenue-critical applications. VoltDB was founded by a team of world-class database experts, including Dr. Michael Stonebraker, winner of the coveted ACM Turing award.

VOLTDB

 209 Burlington Road, Suite 203, Bedford, MA 01730 voltdb.com