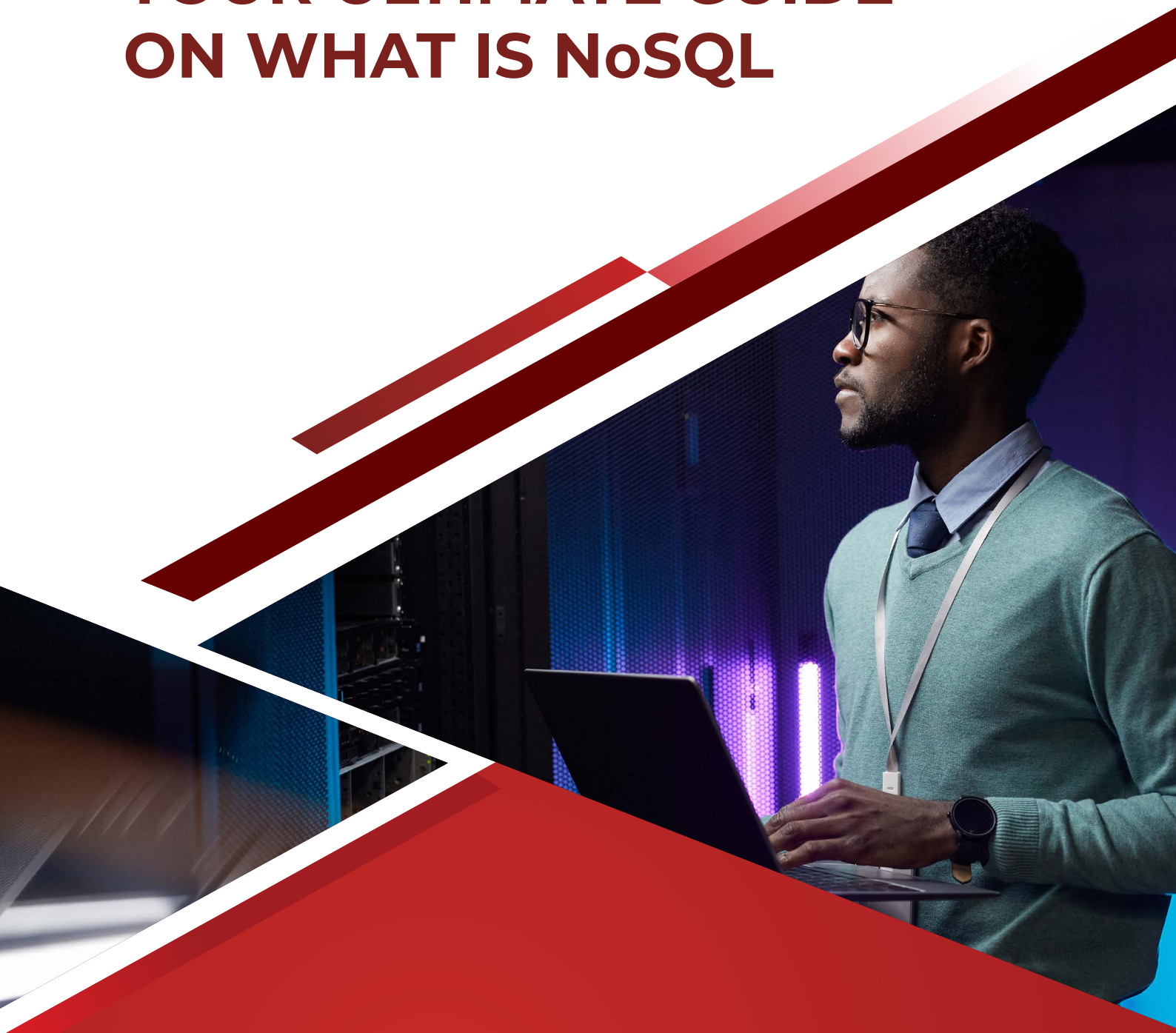
ATLAS SYSTEMS

YOUR ULTIMATE GUIDE ON WHAT IS NoSQL



Exclusive Whitepaper on NoSQL Database Management System

NoSQL describes a philosophical database design that excludes relationship data storage. NoSQL databases tend to be highly specialized and purpose-built. They shouldn't be considered SQL replacements.

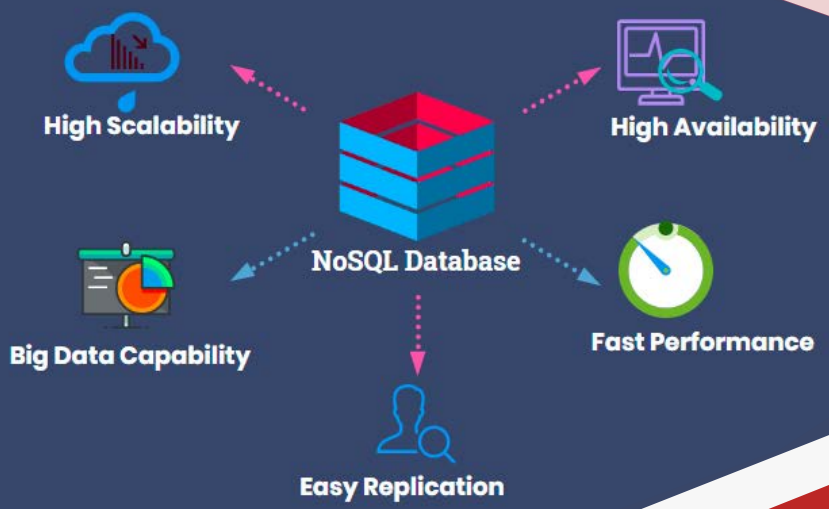
TABLE OF CONTENT

WHAT IS A NoSQL DATABASE?

There is no concrete answer to the question, “What is a NoSQL database?” In the early days, “NoSQL” indicated that a database didn’t support SQL but explaining what something doesn’t support isn’t really a helpful answer to let people know what it does support. The term, however, evolved from there to mean almost the opposite of not supporting SQL when it was updated its meaning to “Not Only SQL.”

In the big picture, NoSQL describes a philosophical database design that excludes relationship data storage. NoSQL databases tend to be highly specialized and purpose-built. They shouldn’t be considered SQL replacements. They could be described as “non-relational” and/or “schema-less,” but the latter isn’t quite correct as a schema can sometimes be used by NoSQL databases, they’re just not required.

For example, in a SQL database, setting up a database for addresses begins with the understanding that the stored records will remain largely unchanged. After evaluating expected query patterns, a SQL database might be expected to optimize storage in two tables, one containing basic information about the customer, with the last name of the customer being the key connecting both tables. In the NoSQL example, however, each database would be designed with a specific customer scenario in mind, complete with a unique technical reason for that organization and scenario. In the simplest case, both the basic information and the customer information would be combined to form one JSON document, which would be accessible under specific querying.



TYPE OF NoSQL DATABASES

The four main types of NoSQL databases are document, as mentioned above, as well as key-value, wide-column, and graph. Document data models are a popular alternative to tabular, relational databases. Storing data in documents similar to JSON objects is popular because these documents are flexible to work with. Documents map to the objects in the developer code, making them easy to work with, which can result in higher productivity and quicker application development. Data doesn't need to be decomposed across tables, combined via JOINs, or integrated through separate object–relational mapping (ORM) layers. Data accessed together are stored together, which requires less code, while also creating a more optimized system.

Each document contains pairs of fields and values, which can be arrays, strings, numbers, booleans, or objects. Documents also have dynamic schemas, which are also self-describing so they don't have to be pre-defined in the database. Fields are not set in stone and can vary from document to document, while the structure can be modified at any time, which helps reduce disruptive schema migrations.

JSON documents are lightweight, language-independent, and human-readable. Today, JSON documents are everywhere, establishing themselves as the standard for data interchange and storage. Documents use a single query language but provide users with a consistent development experience no matter how the data is to be modeled, which can be in ad hoc queries, indexing, or through real-time aggregations.

At its core, document databases are independent units but they are spread across a distributed system. Replication with self-healing recovery keeps applications accessible while giving users the ability to segregate different workloads into a single cluster.

Key-value databases are simpler types of databases. Each item contains keys and values that can only be retrieved by referencing a key. Key-value databases are useful in cases where large amounts of data need to be stored, but complex queries don't need to be performed to retrieve that data.

Wide-column stores maintain data in tables, rows, and dynamic columns, and each row is not required to have the same column, affording great flexibility when compared to relational databases. Considered by many to be two-dimensional key-value databases, wide-column stores allow the storage of large amounts of data. They are particularly useful when query patterns are predictable. Internet of Things and user profile data are commonly housed in wide-column stores.

Graph databases store information about people, places, and things in nodes while edges contain information about the relationships between those nodes. Graph databases can easily spot patterns that might reveal fraud as well as provide customer recommendations.

NoSQL DATABASE ADVANTAGES

NoSQL databases are scalable and highly available. They are designed to support seamless, online horizontal scalability without a significant single point of failure. Unlike relational databases that have rigid data models that require the precise and rigorous up-front design to ensure adequate performance and resist evolution, NoSQL databases have flexible data models. Changing a schema doesn't mean there will be some system downtime as most schemas can be changed on the fly and developers aren't forced to make specific up-front commitments to any data models.

Unlike with relationship databases, scaling horizontally is simple. High performance is easily achieved because NoSQL databases limit the scope of what a database can do, thereby ensuring what it can actually do is done in an extremely good and highly optimized way.

One of NoSQL's strengths is its scalability. NoSQL databases were conceived for distributed data storage, which is a database system that can be expanded over multiple computers or even across a network of computers by simply adding commodity servers, without worrying about data balance, replication, or node failure. Sharding -- the practice of separating one table's rows into multiple different tables, known as partitions -- is done automatically by a NoSQL database, freeing up developers to focus on an application's logic and content instead of worrying about tricky infrastructure duties.

Sharding breaks the data into two or more smaller chunks, known as logical shards. Each partition replicates the schema and the columns, as well as the different rows. Data held in each partition is unique and siloed from other partitioned data. These logical shards are spread across separate database nodes, known as physical shards, which can contain multiple logical shards. Collectively, the data controlled within all the shards will comprise a full logical dataset.

NoSQL databases are considered a shared-nothing architecture. All shards are autonomous, sharing none of the same data or any of the computing resources. However, in some cases, tables are replicated to serve as reference tables. Sharding is often enabled within the application, where code that defines which shard transmits read and write instructions will be included.

The main appeal of sharding a database is that it can facilitate horizontal scaling, i.e., the addition of machines to an existing stack to spread out the load and allow more traffic and faster processing. This contrasts with vertical scaling which is a lot less flexible as it involves upgrading the hardware of an existing server, usually by adding more RAM or CPU power.

Although it's pretty simple to run a relational database on a single machine and scale it up as necessary by upgrading its computing resources, a non-distributed database will always be limited by storage and compute power. Scaling horizontally is almost as easy as flipping on a switch and it will always make an IT estate more flexible.

A sharded database can speed up query response times. When submitting a query on an unsharded database, every row in the table being queried has to be searched before any results set can be surfaced. Obviously, with massive databases, querying can become exceptionally slow, in some cases prohibitively slow. However, by sharding one table into multiple tables, fewer rows need querying and the results returned will be much quicker.

Sharding can increase application reliability by mitigating the impact of outages. Unsharded databases lack the failsafe inherent in siloed applications. In a sharded database, any disruption would probably only affect a single shard. Obviously, this could certainly affect parts of an application or a website, but the overall impact would be far less disruptive than if an entire database went down.

NoSQL is appropriate for highly flexible data models that have very specific needs which don't fit into relational models. If a lot of unstructured data is being imported and manipulated, MongoDB or CouchDB is preferred. If high-speed access to key-value data is required with strong integrity guarantees, Redis works. Elasticsearch excels if complex and/or flexible searches are needed to churn through lots of data.

NoSQL DATABASE DISADVANTAGES

NoSQL databases are not as reliable as relational databases because they don't inherently support the ACID (Atomicity, Consistency, Isolation, Durability) protocol, which guarantees data validity. In order to support these properties, NoSQL developers have to implement their own workaround code, which can be a tricky, complicated, and far less secure process.

SQL is a technology that is 30+ years older than NoSQL, which means it is much more functional and is considerably more stable than NoSQL. The two languages are not compatible at all, which is not a surprise. NoSQL is a specialized database, extremely good for highly specific use cases, but it is not necessarily meant as a catch-all language like SQL is. This means using NoSQL could mean multiple types of databases and data models might be required to fill all niches and unique use cases. Even then, some form of SQL still might be needed for streamlined processing purposes.

Because NoSQL isn't designed to remove data duplication, databases can grow and grow into behemoths. However, with storage getting cheaper and cheaper by the day, this is becoming less and less of an issue. With no real single source of the truth, data quality can also be questionable.

CONCLUSION

It's not really a question of whether one should choose SQL or NoSQL. The NoSQL data model doesn't use relational databases, but it is not meant as a replacement, it's more a complementary or enhancing tool. NoSQL is exceptionally versatile and has limitless use cases and almost limitless scaling capabilities, but NoSQL is more of a design philosophy than a full-on replacement for SQL, which is a good thing as SQL is such an established legacy database system that it would be almost impossible to get rid of it. NoSQL is an answer to the problem of Big Data and it helps users build models on the massive amounts of data that are now flowing through so many enterprise systems these days.

SQL databases include Oracle, MySQL, Microsoft SQL Server, and PostgreSQL. NoSQL databases include three types; Document -- MongoDB and CouchDB; Key-value -- Redis and DynamoDB; and Wide-column -- Cassandra and HBase; Graph: Neo4j and Amazon Neptune. Schemas are rigid for SQL databases but flexible for NoSQL. SQL databases scale vertically, with larger servers, while NoSQL databases scale horizontally, scaling out across servers, which is a much faster and more optimized way to scale. The former database requires joins while the latter doesn't.

Both SQL and NoSQL have their own advantages and disadvantages and, while the relational database community has recognized that NoSQL was addressing inherent deficiencies in technology, it has responded to this challenge with products like NewSQL, which attempts to provide the scalability of a NoSQL system for online transaction processing (OLTP) workloads while maintaining the ACID guarantees of a traditional database system.

Technology constantly evolves. IT departments are always looking out for new systems and tech manufacturers are happy to oblige, which has resulted in a proliferation of systems that address one fundamental data warehouse problem after the next. In many ways, this is great for developers. Not all applications have relational database-shaped problems or would choose to trade-off data duplication issues for a more unified analytics platform that also contains a single version of data truth. However, as mentioned at the top, NoSQL is as much a philosophical database design as a physical one. Anyone deciding between SQL and NoSQL and even amongst the NoSQL database options should be armed with a deep understanding of the advantages and disadvantages of each as in some cases trade-offs won't be needed.

Both SQL and NoSQL have their own advantages and disadvantages and, while the relational database community has recognized that NoSQL was addressing inherent deficiencies in technology, it has responded to this challenge with products like NewSQL, which attempts to provide the scalability of a NoSQL system for online transaction processing (OLTP) workloads while maintaining the ACID guarantees of a traditional database system.

HOW ATLAS CAN HELP?

Atlas Systems is a leading IT & Technology company and helping mid to large size companies to transform and optimize their IT operations using AI & ML. We are also assisting Fortune 1000 clients with cost-effective IT operations support.

We have implemented NoSQL solutions with all of the leading platforms. We have assisted Fortune 1000 clients to build private clouds, public clouds, and hybrid clouds. Our highly qualified and experienced team at Atlas has the expertise to assist you to define the roadmap, strategize, and execute your journey to the cloud.

We provide comprehensive database support services that manage all your database support requirements. Our highly qualified and experienced team of experts will work with you to recommend and perform every database support task required. We support:

Apache CASSANDRA

mongoDB.

Couchbase NoEQUAL

snowflake

PARTNERS

Microsoft Gold Partner

ORACLE

Couchbase

UiPath

snowflake

103 Carnegie Ctr Ste 300 Princeton, NJ 08540

+1 609 256 4556 sales@atlassystems.com AtlasSystems.com