BLOG@CACM

NoSQL: A Beginner's Guide

By Alex Williams
January 28, 2021
Comments

The rise of big data over the years, especially with all the social networks these days, have compelled big companies to reassess their data access and collection methods — and this is where NoSQL databases come into play.

But what exactly is NoSQL? And why should it matter in today's hyper-connected society?

Defining NoSQL

A NoSQL database refers to a non-relational database, but it isn't the antithesis to SQL (Structured Query Language) either. NoSQL isn't a query language like SQL — it's the database. Second, while "NoSQL" may suggest it means "not SQL", it's actually "not only SQL".

NoSQL databases don't have a go-to query language like an RDBMS (Relational Database Management System), which usually uses SQL or one of its variations. As its name implies, the RDBMS has a relational model: data is arranged into logically related tables of fields, rows, and columns. NoSQL databases don't organize data in tabular form, hence their main "non-relational" characteristic.

NoSQL vs SQL Databases

Both database types have their distinct set of pros and cons. To make an informed decision, you need to see how NoSQL and SQL databases compare in different aspects:

Scalability

NoSQL databases use horizontal scaling: the installation of additional servers to help the main server. Such pieces of hardware can be costly, but they're meant for long-term performance and are better for big and/or constantly shifting datasets. Issues brought about by growing traffic are addressed with servers. On the other hand, SQL databases employ vertical scaling: improving RAM, SSD, and CPU components.

Flexibility

An SQL database has related tables and predefined schemas — ensuring consistent data. The downside is that it isn't as flexible as a NoSQL database: you can't just modify schema definitions of SQL databases. They're rigid and all data in the database must adhere to them.

A NoSQL database has flexible schemas. This is excellent for unstructured and semi-structured data. You can place datasets here that tabular structures aren't designed to conveniently handle. Moreover, data can be stored from a variety of database types, which will be covered in a later section.

ACID vs BASE Principles

Many SQL databases follow the ACID model:

***A**tomicity*: All statements in a database transaction must succeed for data changes to take effect. If one part fails, then the whole process is canceled. For example, all statements are reverted if the delete or insert operation doesn't work. Essentially, atomicity protects data integrity.

***C**onsistency*: A database should only have data that adheres to all defined rules or schemas. A transaction that leads non-compliant data to be saved will cause a rollback instead. Consistency prevents failed or incomplete transactions from corrupting the database.

***I**solation*: Transactions can take place simultaneously, but a transaction currently running cannot affect others. Imagine you're attempting to purchase four VIP concert tickets and there are only five left on the ticketing website. Another individual won't succeed in purchasing two tickets if you've already placed four tickets in your cart. Similarly, if that person gets two tickets first, you won't be able to order four tickets anymore.

***D**urability*: System crashes and power failures don't ruin successful transactions. As long as it's complete, the transaction's changes to the database will remain. The committed data won't be lost, since there will already be transaction logs.

In contrast, NoSQL databases have the BASE model, which lends itself more to easy scalability and data flexibility than data consistency:

***B**asically **A**vailable*: Consistency in data is present in BASE, too, but it takes less priority over data being available. A NoSQL database isn't perfect, but it's designed to function nearly all the time.

***S**oft State*: The BASE model doesn't mind if stores or replicas are inconsistent at one point.

***E**ventual Consistency*: Mutual consistency eventually occurs among the database stores. While the ACID model demands data consistency at all times, the BASE model accepts temporary store inconsistencies since they'll be consistent when the system is done with all the inputs.

Types of NoSQL Databases

Many NoSQL database models exist, but these are the four general classifications:

Document-Oriented Database

This database model uses documents, which are data structures having distinct key-value pairs or nested documents. All the metadata of a particular data item is kept in one place. Documents are considered in their entirety: breaking them into segments of their key-value pairs isn't suggested. This way, the database can group distinct documents into one collection. Moreover, a document-oriented NoSQL database can index documents by their primary identifier and properties.

This type of database is considered a subtype of a key-value database. However, what sets the two apart is how they process data. A documented-oriented database uses the internal document structure to pull metadata. The database engine then utilizes the metadata to optimize the database. On the other hand, data is viewed as

MORE NEWS & OPINIONS
The AI-Powered, Totally Autonomous Future of War Is Here
Wired
Why You Should Be Able to Make Your Own Individualized, Digital Nano-Currency
Wesley Sheh, Joel Nishimura
How to Ace IT Product Localization: The 101 Guide
Alex Tray

inherently non-transparent to a key-value database.

Document-oriented databases offer a stark contrast to SQL or relational databases. The latter keeps data in related tables — and one object can be utilized in many of these tables. Document-oriented databases keep every detail of an object in just one instance. Likewise, stored objects in the database don't need to have a shared attribute, so the database can retrieve and store data without object-relational mapping.

MongoDB, IBM Domino, Cosmos DB, ArangoDB, eXist-db, RethinkDB, and Clusterpoint are examples of document-oriented databases. There are many others worth looking into, so consider looking into the comprehensive list of NoSQL databases. Each system offers certain advantages and may differ in terms of API, query method, protocol, and programming language.

Graph-Based Database

While the graph-based database is a type of NoSQL database, it puts great value in relationships as an RDBMS does. Other database models associate pieces of data with underlying connections. Conversely, graph-based databases use dependencies between nodes for naming and handling relationships — and even assigning them properties.

A graph-based database has objects known as nodes and edges. Nodes are items similar to relations or records in an RDBMS or a document in a document-oriented database. Edges are lines that link nodes. Otherwise known as graphs, they signify node relationships and are a core concept of graph-based databases. InfiniteGraph, Trinity, Onyx DBMS, and InfoGrid are graph-based databases. Overall, this type of NoSQL database is great for content management and the integration and grouping of massive data sets.

Wide-Column Store Database

Similar to SQL databases, a wide-column store database has tables filled with rows and columns. The big difference, however, is that this NoSQL database can have tables where the respective columns from row to row don't share names and formats. Likewise, each row can have different total numbers of columns. You can insert a column to a single row at any moment without having to do the same for all other rows.

Also known as an extensible record store, column-oriented DBMS, or columnar database, a wide-column store database arranges related information into columns. These columns can be bunched into column families. With this type of database, you can store information that would've required many rows in an SQL database in just one column. Notable wide-column store databases include Cosmos DB, ScyllaDB, Hypertable, and Apache Cassandra.

Key-Value Database

Every piece of data in a key-value database or key-value store has a lookup key, which corresponds to a particular value. This database is considered one of the simpler NoSQL database models — making it a good pick for horizontal scaling. Sets of records or objects are located in a hashtable or dictionary, the structure of key-value stores. Each object has many data-rich fields. To access a specific object in the database, the user must have a unique key for it.

So how does the key-value database differ from a relational database? Instead of a predefined structure where related tables have fields with strictly defined data types, a key-value store looks at data as one opaque set composed of many fields handling each record. Data isn't relationally stored — the key-value database stores pointers and their associated data with unique keys. Thus, key-value databases are more flexible and require much less memory for database management, which can improve performance in intensive workloads.

Berkeley DB is a key-value store written in C. Its API binding includes Python, Java, and Ruby. Similarly, Redis and RocksDB are written in C. Other key-value database examples include LevelDB, GenieDB, and Chordless.

Conclusion

A NoSQL database is a welcome solution in an age where businesses must keep up with rapid digitization and the rise of people connecting to the Internet. Relational databases aren't viable methods for storing and managing unstructured data. On the other hand, NoSQL databases offer efficient horizontal scaling instead of expensive vertical scaling that may sometimes even lead to considerable downtime.

This is not to say that SQL and relational databases no longer have a place in business today. Rigid data models that prioritize data consistency and integrity at all times are valuable. But if agile development and scalability without downtime in the era of web applications are more crucial to your business goals, then NoSQL databases are worth looking into.

Alex Williams *is a full-stack developer with over 15 years of experience, and the owner of Hosting Data UK.*

No entries found