

NoSQL vs. SQL Databases

TL;DR: NoSQL (“non SQL” or “not only SQL”) databases were developed in the late 2000s with a focus on scaling, fast queries, allowing for frequent application changes, and making programming simpler for developers. Relational databases accessed with SQL (Structured Query Language) were developed in the 1970s with a focus on reducing data duplication as storage was much more costly than developer time. SQL databases tend to have rigid, complex, tabular schemas and typically require expensive vertical scaling.

If you're not familiar with what NoSQL databases are or the different types of NoSQL databases, [start here](#).

Overview

Below is an overview of what this article covers.

- [What are the differences between SQL and NoSQL?](#)
- [What are the benefits of NoSQL databases?](#)
- [What are the drawbacks of NoSQL databases?](#)
- [Try a NoSQL database](#)

Why NoSQL Databases



Differences between SQL and NoSQL

The table below summarizes the main differences between SQL and NoSQL databases.

	SQL Databases	NoSQL Databases
Data Storage Model	Tables with fixed rows and columns	Document: JSON documents, Key-value: key-value pairs, Wide-column: tables with rows and dynamic columns, Graph: nodes and edges
Development History	Developed in the 1970s with a focus on	Developed in the late 2000s with a focus on scaling and allowing for

	SQL Databases	NoSQL Databases
	reducing data duplication	rapid application change driven by agile and DevOps practices.
Examples	Oracle, MySQL, Microsoft SQL Server, and PostgreSQL	Document: MongoDB and CouchDB, Key-value: Redis and DynamoDB, Wide-column: Cassandra and HBase, Graph: Neo4j and Amazon Neptune
Primary Purpose	General purpose	Document: general purpose, Key-value: large amounts of data with simple lookup queries, Wide-column: large amounts of data with predictable query patterns, Graph: analyzing and traversing relationships between connected data
Schemas	Rigid	Flexible
Scaling	Vertical (scale-up with a larger server)	Horizontal (scale-out across commodity servers)
Multi-Record ACID Transactions	Supported	Most do not support multi-record ACID transactions. However, some – like MongoDB – do.
Joins	Typically required	Typically not required

	SQL Databases	NoSQL Databases
Data to Object Mapping	Requires ORM (object-relational mapping)	Many do not require ORMs. MongoDB documents map directly to data structures in most popular programming languages.

What are the benefits of NoSQL databases?

NoSQL databases offer many benefits over relational databases. NoSQL databases have flexible data models, scale horizontally, have incredibly fast queries, and are easy for developers to work with.

- **Flexible data models**

NoSQL databases typically have very flexible schemas. A flexible schema allows you to easily make changes to your database as requirements change. You can iterate quickly and continuously integrate new application features to provide value to your users faster.

- **Horizontal scaling**

Most SQL databases require you to scale-up vertically (migrate to a larger, more expensive server) when you exceed the capacity requirements of your current server. Conversely, most NoSQL

databases allow you to scale-out horizontally, meaning you can add cheaper commodity servers whenever you need to.

- **Fast queries**

Queries in NoSQL databases can be faster than SQL databases.

Why? Data in SQL databases is typically normalized, so queries for a single object or entity require you to join data from multiple tables.

As your tables grow in size, the joins can become expensive.

However, data in NoSQL databases is typically stored in a way that is optimized for queries. The rule of thumb when you use MongoDB is **data that is accessed together should be stored together**. Queries typically do not require joins, so the queries are very fast.

- **Easy for developers**

Some NoSQL databases like MongoDB map their data structures to those of popular programming languages. This mapping allows developers to store their data in the same way that they use it in their application code. While it may seem like a trivial advantage, this mapping can allow developers to write less code, leading to faster development time and fewer bugs.

What are the drawbacks of NoSQL databases?

One of the most frequently cited drawbacks of NoSQL databases is that they don't support ACID (atomicity, consistency, isolation, durability) transactions across multiple documents. With appropriate schema design, single-record atomicity is acceptable for lots of applications. However, there are still many applications that require ACID across multiple records.

To address these use cases, MongoDB added support for [multi-document ACID transactions](#) in the 4.0 release, and extended them in 4.2 to span sharded clusters.

Since data models in NoSQL databases are typically optimized for queries and not for reducing data duplication, NoSQL databases can be larger than SQL databases. Storage is currently so cheap that most consider this a minor drawback, and some NoSQL databases also support compression to reduce the storage footprint.

Depending on the NoSQL database type you select, you may not be able to achieve all of your use cases in a single database. For example, graph databases are excellent for analyzing relationships in your data but may not provide what you need for everyday retrieval of the data such as range queries. When selecting a NoSQL database, consider what your use cases will be and if a general purpose database like MongoDB would be a better option.

How to try a NoSQL database

Now that you understand the basics of NoSQL databases, you're ready to give them a shot.

You can check out the [Where to Use MongoDB white paper](#) to help you determine if MongoDB or another database is right for your use case. Then, hop on over to [What Is a Document Database?](#) to learn about the document model and how it compares to the relational model.

For those who like to jump right in and learn by doing, one of the easiest ways to get started with NoSQL databases is to use [MongoDB Atlas](#). Atlas is MongoDB's fully managed, global database service that is available on all of the leading cloud providers. One of the many handy

things about Atlas is that it has a generous, forever-free tier so you can create a database and discover all of the benefits of NoSQL databases firsthand without providing your credit card.

For those who prefer structured learning, [MongoDB University](#) offers completely free online training that will walk you step by step through the process of learning MongoDB.

When you're ready to interact with MongoDB using your favorite programming language, check out the [Quick Start Tutorials](#). These tutorials will help you get up and running as quickly as possible in the language of your choice.

This article was written by [Lauren Schaefer](#), a MongoDB Developer Advocate.

Get Started with MongoDB Atlas

Experience the benefits of using MongoDB, the premier NoSQL database, on the cloud.



[Get Started Free!](#)

Related NoSQL resources

- [AWS re:Invent 2022 Presentation - From RDBMS to NoSQL](#)

- [NoSQL Explained](#)
- [Types of NoSQL Databases](#)
- [Advantages of NoSQL](#)
- [When to Use NoSQL](#)
- [NoSQL Data Modeling](#)
- [NoSQL Examples](#)



 English 

About

[Careers](#)

[Investor Relations](#)

[Legal Notices](#)

[Privacy Notices](#)

[Security Information](#)

[Trust Center](#)

Support

[Contact Us](#)

[Customer Portal](#)


[Atlas Status](#)

[Customer Support](#)


[Manage Cookies](#)

Social

 [GitHub](#)

 [Stack Overflow](#)

 [LinkedIn](#)

 [YouTube](#)



X



Twitch



Facebook

© 2024 MongoDB, Inc.