# What is NoSQL?

NoSQL databases (aka "not only SQL") are non tabular, and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.

**Advantages of NoSQL:**  
There are many advantages of working with NoSQL databases such as MongoDB and Cassandra. The main advantages are high scalability and high availability.

1. **High scalability –**  
   NoSQL database use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharding. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra etc. NoSQL can handle huge amount of data because of scalability, as the data grows NoSQL scale itself to handle that data in efficient manner.
2. **High availability –**  
   Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.

**Disadvantages of NoSQL:**  
NoSQL has the following disadvantages.

1. **Narrow focus –**  
   NoSQL databases have very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.
2. **Open-source –**  
   NoSQL is open-source database. There is no reliable standard for NoSQL yet. In other words two database systems are likely to be unequal.
3. **Management challenge –**  
   The purpose of big data tools is to make management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.
4. **GUI is not available –**  
   GUI mode tools to access the database is not flexibly available in the market.
5. **Backup –**  
   Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.
6. **Large document size –**  
   Some database systems like MongoDB and CouchDB store data in JSON format. Which means that documents are quite large (BigData, network bandwidth, speed), and having descriptive key names actually hurts, since they increase the document size.

**Types of NoSQL database:**  
Types of NoSQL databases and the name of the databases system that falls in that category are:

1. MongoDB falls in the category of NoSQL document based database.
2. **Key value store:** Memcached, Redis, Coherence
3. **Tabular:** Hbase, Big Table, Accumulo
4. **Document based:** MongoDB, CouchDB, Cloudant

**When should NoSQL be used:**

1. When huge amount of data need to be stored and retrieved .
2. The relationship between the data you store is not that important
3. The data changing over time and is not structured.
4. Support of Constraints and Joins is not required at database level
5. The data is growing continuously and you need to scale the database regular to handle the data.

## **Overview**

This article will cover:

* [What is NoSQL?](https://www.mongodb.com/nosql-explained#what-is-nosql)
* [What is SQL?](https://www.mongodb.com/nosql-explained#what-is-sql)
* [What are the types of NoSQL databases?](https://www.mongodb.com/nosql-explained#what-are-the-types-of-nosql-databases)
* [How NoSQL Databases Work](https://www.mongodb.com/nosql-explained#how-nosql-databases-work)
* [Try a NoSQL database](https://www.mongodb.com/nosql-explained#try-a-nosql-database)

## **What is NoSQL?**

When people use the term “NoSQL database”, they typically use it to refer to any non-relational database. Some say the term “NoSQL” stands for “non SQL” while others say it stands for “not only SQL.” Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables.

A common misconception is that NoSQL databases or non-relational databases don’t store relationship data well. NoSQL databases can store relationship data—they just store it differently than relational databases do. In fact, [when compared with SQL databases](https://www.mongodb.com/nosql-explained/nosql-vs-sql), many find modeling relationship data in NoSQL databases to be easier than in SQL databases, because related data doesn’t have to be split between tables.

NoSQL data models allow related data to be nested within a single data structure.

NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model simply for the purposes of reducing data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity.

## **What is SQL?**

Now that we have an understanding of NoSQL databases, let’s contrast them with what have traditionally been the most popular databases: relational databases accessed by SQL (Structured Query Language). You can use SQL when interacting with relational databases where data is stored in tables that have fixed columns and rows.

SQL databases rose in popularity in the early 1970s. At the time, storage was extremely expensive, so software engineers normalized their databases in order to reduce data duplication.

Software engineers in the 1970s also commonly followed the waterfall software development model. Projects were planned in detail before development began. Software engineers painstakingly created complex entity-relationship (E-R) diagrams to ensure they had carefully thought through all the data they would need to store. Due to this upfront planning model, software engineers struggled to adapt if requirements changed during the development cycle. As a result, projects frequently went over budget, exceeded deadlines and failed to deliver against user needs.

## **What are the Types of NoSQL Databases?**

Over time, four major types of NoSQL databases emerged: [document databases](https://www.mongodb.com/document-databases), [key-value databases](https://www.mongodb.com/key-value-database), wide-column stores, and graph databases. Let’s examine each type.

* **Document databases** store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects, and their structures typically align with objects developers are working with in code. Because of their variety of field value types and powerful query languages, document databases are great for a wide variety of use cases and can be used as a general purpose database. They can horizontally scale-out to accomodate large data volumes. MongoDB is consistently ranked as the world’s most popular NoSQL database according to [DB-engines](https://db-engines.com/en/ranking) and is an example of a document database. For more on document databases, visit [What is a Document Database?](https://www.mongodb.com/document-databases).
* **Key-value databases** are a simpler type of database where each item contains keys and values. A value can typically only be retrieved by referencing its key, so learning how to query for a specific key-value pair is typically simple. Key-value databases are great for use cases where you need to store large amounts of data but you don’t need to perform complex queries to retrieve it. Common use cases include storing user preferences or caching. Redis and DynanoDB are popular key-value databases.
* **Wide-column stores** store data in tables, rows, and dynamic columns. Wide-column stores provide a lot of flexibility over relational databases because each row is not required to have the same columns. Many consider wide-column stores to be two-dimensional key-value databases. Wide-column stores are great for when you need to store large amounts of data and you can predict what your query patterns will be. Wide-column stores are commonly used for storing Internet of Things data and user profile data. Cassandra and HBase are two of the most popular wide-column stores.
* **Graph databases** store data in nodes and edges. Nodes typically store information about people, places, and things while edges store information about the relationships between the nodes. Graph databases excel in use cases where you need to traverse relationships to look for patterns such as social networks, fraud detection, and recommendation engines. Neo4j and JanusGraph are examples of graph databases.

## **How NoSQL Databases Work**

One way of understanding the appeal of NoSQL databases from a design perspective is to look at how the data models of a SQL and a NoSQL database might look in an oversimplified example using address data.

**The SQL Case.** For an SQL database, setting up a database for addresses begins with the logical construction of the format and the expectation that the records to be stored are going to remain relatively unchanged. After analyzing the expected query patterns, an SQL database might optimize storage in two tables, one for basic information and one pertaining to being a customer, with last name being the key to both tables. Each row in each table is a single customer, and each column has the following fixed attributes:

* Last name :: first name :: middle initial :: address fields :: email address :: phone number
* Last name :: date of birth :: account number :: customer years :: communication preferences

**The NoSQL Case.** In the section Types of NoSQL Databases above, there were four types described, and each has its own data model.

Each type of NoSQL database would be designed with a specific customer situation in mind, and there would be technical reasons for how each kind of database would be organized. The simplest type to describe is the document database, in which it would be natural to combine both the basic information and the customer information in one JSON document. In this case, each of the SQL column attributes would be fields and the details of a customer’s record would be the data values associated with each field.

For example: Last\_name: "Jones", First\_name: "Mary", Middle\_initial: "S", etc

## **Try a NoSQL Database**

If you’d like to try a NoSQL database, [MongoDB Atlas](https://www.mongodb.com/cloud/atlas/register?tck=nosql-explained) is a great place to start. Atlas is a database service that is fully managed by MongoDB and available on all of the leading cloud providers. Atlas has a forever-free tier that you can use to kick the tires and discover the basics.

Not sure what to do now that you have an Atlas account? Head over to [MongoDB University](https://university.mongodb.com/) where you can get free online training from MongoDB engineers. MongoDB University has surpassed 1.4 million course registrations. The [Quick Start Tutorials](https://www.mongodb.com/blog/channel/quickstart) are another great place to start as they will help you get up and running quickly with your favorite programming language.

NoSQL vs SQL Databases

**TLDR:** 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.

## **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 reducing data duplication | Developed in the late 2000s with a focus on scaling and allowing for 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 |
| 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 is 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](https://www.mongodb.com/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 whitepaper](https://www.mongodb.com/collateral/use-case-guidance-where-to-use-mongodb) 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?](https://www.mongodb.com/document-databases) 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](https://www.mongodb.com/cloud/atlas?tck=NoSQLvsSQL). 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 first hand without providing your credit card.

For those who prefer structured learning, [MongoDB University](https://university.mongodb.com/) is 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](https://www.mongodb.com/blog/channel/quickstart). These tutorials will help you get up and running as quickly as possible in the language of your choice.