

What is NoSQL?

Spin up a NoSQL Cluster Free

Try a NoSQL Database with Atlas →

NoSQL databases (aka "not only SQL") are non-tabular databases 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.

In this article, you'll learn *what* a NoSQL database is, *why* (and when!) you should use one, and *how* to get started.

Overview

This article will cover:

- [What is a NoSQL Database?](#)
 - [Brief History of NoSQL Databases](#)
 - [NoSQL Database Features](#)
 - [Types of NoSQL Database](#)

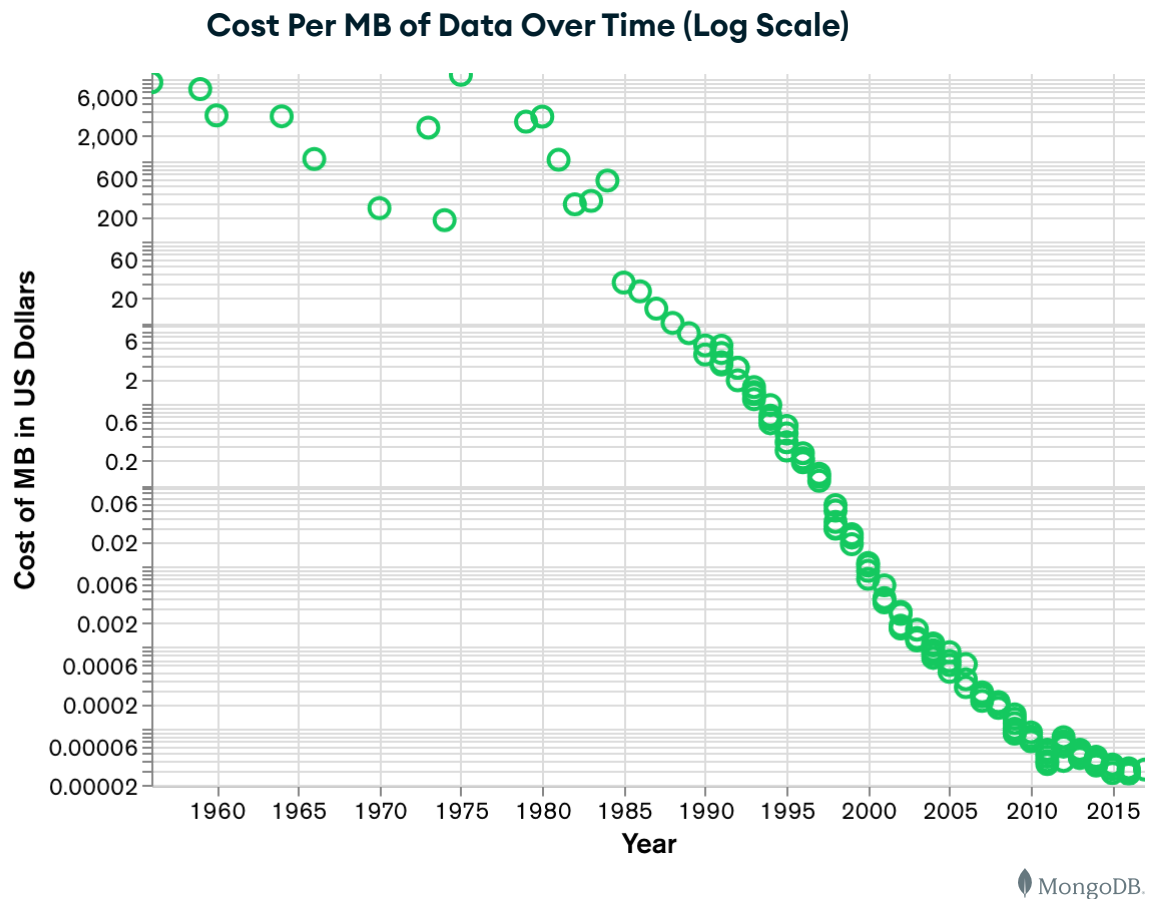
- [Difference between RDBMS and NoSQL](#)
- [Why NoSQL?](#)
- [When should NoSQL be Used?](#)
- [NoSQL Database Misconceptions](#)
- [NoSQL Query Tutorial](#)
- [Summary](#)
- [FAQ](#)

What is a NoSQL database?

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.

Brief history of NoSQL databases

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 in order to avoid data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity.



As storage costs rapidly decreased, the amount of data that applications needed to store and query increased. This data came in all shapes and sizes – **structured**, **semi-structured**, and **polymorphic** – and defining the schema in advance became nearly impossible. NoSQL databases allow developers to store huge amounts of unstructured data, giving them a lot of flexibility.

Additionally, the **Agile Manifesto** was rising in popularity, and software engineers were rethinking the way they developed software. They were recognizing the need to rapidly adapt to changing requirements. They needed the ability to iterate quickly and make changes throughout

their software stack – all the way down to the database. NoSQL databases gave them this flexibility.

Cloud computing also rose in popularity, and developers began using public clouds to host their applications and data. They wanted the ability to distribute data across multiple servers and regions to make their applications resilient, to scale out instead of scale up, and to intelligently geo-place their data. Some NoSQL databases like MongoDB provide these capabilities.

NoSQL database features

Each NoSQL database has its own unique features. At a high level, many NoSQL databases have the following features:

- [Flexible schemas](#)
- [Horizontal scaling](#)
- [Fast queries due to the data model](#)
- [Ease of use for developers](#)

Check out [What are the Benefits of NoSQL Databases?](#) to learn more about each of the features listed above.

Types of NoSQL databases

Over time, four major [types of NoSQL databases](#) emerged: document databases, [key-value databases](#), wide-column stores, and graph databases.

- **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.

- **Key-value databases** are a simpler type of database where each item contains keys and values.
- **Wide-column stores** store data in tables, rows, and dynamic columns.
- **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.

To learn more, visit

[Understanding the Different Types of NoSQL Databases.](#)

Difference between RDBMS and NoSQL databases

While a variety of differences exist between relational database management systems (RDBMS) and NoSQL databases, one of the key differences is the way the data is modeled in the database. In this section, we'll work through an example of modeling the same data in a relational database and a NoSQL database. Then, we'll highlight some of the other key differences between relational databases and NoSQL databases.

RDBMS vs NoSQL: Data Modeling Example

Let's consider an example of storing information about a user and their hobbies. We need to store a user's first name, last name, cell phone number, city, and hobbies.

In a relational database, we'd likely create two tables: one for Users and one for Hobbies.

Users

ID	first_name	last_name	cell	city
1	Leslie	Yepp	8125552344	Pawnee

Hobbies

ID	user_id	hobby
10	1	scrapbooking
11	1	eating waffles
12	1	working

In order to retrieve all of the information about a user and their hobbies, information from the Users table and Hobbies table will need to be joined together.

The data model we design for a NoSQL database will depend on the type of NoSQL database we choose. Let's consider how to store the same information about a user and their hobbies in a [document database](#) like MongoDB.

```
1  {
2    "_id": 1,
3    "first_name": "Leslie",
4    "last_name": "Yepp",
5    "cell": "8125552344",
6    "city": "Pawnee",
7    "hobbies": ["scrapbooking", "eating waffles", "working"]
8  }
```



In order to retrieve all of the information about a user and their hobbies, a single document can be retrieved from the database. No joins are required, resulting in faster queries.

To see a more detailed version of this data modeling example, read [Mapping Terms and Concepts from SQL to MongoDB](#).

Other differences between RDBMS and relational databases

While the example above highlights the differences in data models between relational databases and NoSQL databases, many other important differences exist, including:

- Flexibility of the schema
- Scaling technique
- Support for transactions
- Reliance on data to object mapping

To learn more about the differences between relational databases and NoSQL databases, visit [NoSQL vs SQL Databases](#), or [watch From RDBMS to NoSQL presentation from AWS re:Invent 2022](#).

Why NoSQL?

NoSQL databases are used in nearly every [industry](#). Use cases range from the highly critical (e.g., storing [financial data](#) and

[healthcare records](#)) to the more fun and frivolous (e.g., [storing IoT readings from a smart kitty litter box](#)).

In the following sections, we'll explore when you should choose to use a NoSQL database and common misconceptions about NoSQL databases.

When should NoSQL be used?

When deciding which database to use, decision-makers typically find one or more of the following factors lead them to selecting a NoSQL database:

- Fast-paced Agile development
- Storage of structured and semi-structured data
- Huge volumes of data
- Requirements for scale-out architecture
- Modern application paradigms like microservices and real-time streaming

See [When to Use NoSQL Databases](#) and [Exploring NoSQL Database Examples](#) for more detailed information on the reasons listed above.

NoSQL database misconceptions

Over the years, many misconceptions about NoSQL databases have spread throughout the developer community. In this section, we'll discuss two of the most common misconceptions:

- Relationship data is best suited for relational databases.
- NoSQL databases don't support ACID transactions.

To learn more about common misconceptions, read [Everything You Know About MongoDB is Wrong](#).

Misconception: relationship data is best suited for relational databases

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 relational databases](#), many find modeling relationship data in NoSQL databases to be easier than in relational 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.

Misconception: NoSQL databases don't support ACID transactions

Another common misconception is that NoSQL databases don't support ACID transactions. Some NoSQL databases like MongoDB do, in fact, support [ACID transactions](#).

Note that the way data is modeled in NoSQL databases can eliminate the need for multi-record transactions in many use cases. Consider the earlier example where we stored information about a user and their hobbies in both a relational database and a document database. In order to ensure information about a user and their hobbies was updated together in a relational database, we'd need to use a transaction to update records in two tables. In order to do the same in a document database, we could update a single document – no multi-record transaction required.

NoSQL query tutorial

A variety of NoSQL databases exist. Today, we'll be trying MongoDB, the world's most popular NoSQL database according to [DB-Engines](#).

In this tutorial, you'll load a sample database and learn to query it – all without installing anything on your computer or paying anything.

Authenticate to MongoDB Atlas

The easiest way to get started with MongoDB is [MongoDB Atlas](#). Atlas is MongoDB's fully managed database-as-a-service. Atlas has a forever free tier, which is what you'll be using today.

1. Navigate to [Atlas](#).
2. [Create an account](#) if you haven't already.
3. [Log into](#) Atlas.
4. Create an Atlas organization and project.

For more information on how to complete the steps above, visit the official MongoDB documentation on [creating an Atlas account](#).

Create a cluster and a database

A cluster is a place where you can store your MongoDB databases. In this section, you'll create a free cluster.

Once you have a cluster, you can begin storing data in Atlas. You could choose to manually [create a database](#) in the [Atlas Data Explorer](#), in the [MongoDB Shell](#), in [MongoDB Compass](#), or using [your favorite programming language](#). Instead, in this example, you will import Atlas's sample dataset.

1. Create a free cluster by following the steps in the [official MongoDB documentation](#).

2. Load the sample dataset by following the instructions in the [official MongoDB documentation](#).

Loading the sample dataset will take several minutes.

While we don't need to think about database design for this tutorial, note that database design and [data modeling](#) are major factors in MongoDB performance. Learn more about best practices for modeling data in MongoDB:

- [MongoDB Schema Design Patterns Blog Series](#)
- [MongoDB Schema Design Anti-Patterns Blog Series](#)
- [Free MongoDB University Course: M320 Data Modeling](#)

Query the database

Now that you have data in your cluster, let's query it! Just like you had multiple ways to create a database, you have multiple options for querying a database: in the [Atlas Data Explorer](#), in the [MongoDB Shell](#), in [MongoDB Compass](#), or using [your favorite programming language](#).

In this section, you'll query the database using the Atlas Data Explorer. This is a good way to get started querying, as it requires zero setup.

- 1. Navigate to the Data Explorer (the Collections tab), if you are not already there. See the [official MongoDB documentation](#) for information on how to navigate to the Data Explorer.

The left panel of the Data Explorer displays a list of [databases](#) and [collections](#) in the current cluster. The right panel of the Data Explorer displays a list of documents in the current collection.

The screenshot shows the MongoDB Data Explorer interface. At the top, there are tabs for Overview, Real Time, Metrics, Collections (selected), Search, Profiler, Performance Advisor, Online Archive, and Command. Below the tabs, it says 'DATABASES: 8 COLLECTIONS: 21'. On the right, there are buttons for 'VISUALIZE YOUR DATA' and 'REFRESH'.

On the left sidebar, there is a '+ Create Database' button and a search bar labeled 'NAMESPACES'. Below this, a list of databases is shown: sample_airbnb (expanded), sample_analytics, sample_geospatial, sample_mflix, sample_restaurants, sample_supplies, sample_training, and sample_weatherdata. Under 'sample_airbnb', the 'listingsAndReviews' collection is selected.

The main panel displays the 'sample_airbnb.listingsAndReviews' collection. It shows 'COLLECTION SIZE: 89.99MB', 'TOTAL DOCUMENTS: 5555', and 'INDEXES TOTAL SIZE: 572KB'. Below this, there are tabs for Find (selected), Indexes, Schema Anti-Patterns (0), Aggregation, and Search Indexes (0). On the right, there is an 'INSERT DOCUMENT' button.

Below the tabs, there is a 'FILTER' bar with the text '{ field: 'value' }' and buttons for 'OPTIONS', 'Apply', and 'Reset'. Below the filter bar, it says 'QUERY RESULTS 1-20 OF MANY'.

The query results show a single document with the following fields:

```
{
  "_id": "10006546",
  "listing_url": "https://www.airbnb.com/rooms/10006546",
  "name": "Ribeira Charming Duplex",
  "summary": "Fantastic duplex apartment with three bedrooms, located in the histori...",
  "space": "Privileged views of the Douro River and Ribeira square, our apartment ...",
  "description": "Fantastic duplex apartment with three bedrooms, located in the histori...",
  "neighborhood_ov...": "In the neighborhood of the river, you can find several restaurants as ...",
  "notes": "Lose yourself in the narrow streets and staircases zone, have lunch in...",
  "transit": "Transport: • Metro station and S. Bento railway 5min; • Bus stop a 50 ...",
  "access": "We are always available to help guests. The house is fully available t...",
  "interaction": "Cot - 10 € / night Dog - € 7,5 / night",
  "house_rules": "Make the house your home...",
  "property_type": "House"
}
```

The Data Explorer displays a list of documents in the listingsAndReviews collection.

- 2. Expand the **sample_mflix database** in the left panel. A list of the database's collections is displayed.
- 3. Select the **movies collection**. The Find View is displayed in the right panel. The first twenty documents of the results are displayed.
- 4. You are now ready to query the movies collection. Let's query for the movie Pride and Prejudice. In the query bar input { **title: "Pride and Prejudice"** } and click **Apply**.

Two documents with the title “Pride and Prejudice” are returned.

QUERY RESULTS 1-2 OF 2

```
_id: ObjectId("573a1397f29313caabce7030")
plot: "While the arrival of wealthy gentleman sends her marriage-minded mothe..."
> genres: Array
runtime: 265
> cast: Array
  num_mflix_comme... : 1
poster: "https://m.media-amazon.com/images/M/MV5BMjA5MTg2OTYyN15BMl5BanBnXkFtZT..."
title: "Pride and Prejudice"
fullplot: "Mrs. Bennet is determined to find husbands for her five daughters. The..."
> languages: Array
released: 1980-01-13T00:00:00.000+00:00
> awards: Object
lastupdated: "2015-09-16 09:17:18.283000000"
year: 1980
> imdb: Object
> countries: Array
type: "series"
```

```
_id: ObjectId("573a1399f29313caabceeead")
plot: "Jane Austen's classic novel about the prejudice that occurred between ..."
> genres: Array
runtime: 327
> cast: Array
poster: "https://m.media-amazon.com/images/M/MV5BMDM0MjFLOGYtNTg2ZC00MmRkLTg5OT..."
title: "Pride and Prejudice"
fullplot: "Jane Austen's classic novel about the prejudice that occurred between ..."
> languages: Array
released: 1996-01-14T00:00:00.000+00:00
```

The results for querying for movies with the title "Pride and Prejudice".

Congrats! You've successfully queried a NoSQL database!

Continue exploring your data

In this tutorial, we only scratched the surface of what you can do in MongoDB and Atlas. Continue interacting with your data by using the Data Explorer to insert new documents, edit existing documents, and delete documents.

When you are ready to try more advanced queries that aggregate your data, [create an aggregation pipeline](#). The aggregation framework is an incredibly powerful tool for analyzing your data. To learn more, take

the free MongoDB University Course
[M121 The MongoDB Aggregation Framework](#).

When you want to visualize your data, check out [MongoDB Charts](#). Charts is the easiest way to visualize data stored in Atlas and Atlas Data Lake. Charts allows you to create dashboards that are filled with visualizations of your data.

Summary

NoSQL databases provide a variety of benefits including flexible data models, horizontal scaling, lightning fast queries, and ease of use for developers. NoSQL databases come in a variety of types including document databases, key-values databases, wide-column stores, and graph databases.

MongoDB is the [world's most popular NoSQL database](#).
[Learn more about MongoDB Atlas](#), and give the free tier a try.

Excited to learn more now that you have your own Atlas account? Head over to [MongoDB University](#) where you can get free online training from MongoDB engineers and earn a [MongoDB certification](#). The [Quick Start Tutorials](#) are another great place to begin; they will get you up and running quickly with your favorite programming language.


Follow this tutorial with MongoDB
Atlas

- [What are the 4 different types of NoSQL databases?](#)
- [NoSQL Databases Advantages](#)
- [NoSQL data modeling and schema design](#)
- [NoSQL Database Examples](#)

Learn More

- [MongoDB Compatibility](#)
- [MongoDB Basics](#)
- [Learn About Databases](#)
- [Languages compatible with MongoDB](#)



 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.