
LESSON
When to Use
Non-Relational Databases
Google slide deck available here
This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License
(CC BY-NC-SA 3.0)

Overview

Learning Objectives

At the end of this course, learners will be able to:

- Explain what use cases are non-relational databases most advantageous for and what kind of data it suits best.

Suggested Uses

- Part of one lecture period
- Handouts / asynchronous learning
- Supplemental reading material - read on your own / not part of formal teaching
- Complement to University course Introduction to MongoDB

This lesson is a part of the course Introduction to Modern Databases with MongoDB.

At a Glance

Length:
10 minutes

Level:
Foundational

Prerequisites:
Relational vs. Non-Relational Databases

Share your feedback: We hope these curriculum materials will be a valuable resource for you and your learners. Let us know how the materials work for you, what we can improve on, and how MongoDB for Academia can support you via our brief feedback form.

MongoDB for Academia: MongoDB for Academia offers resources for educators and students to support teaching and learning MongoDB. Check out our educator resources and join the Educator Community. Students can receive $50 in Atlas credits and free certification through the GitHub Student Developer Pack.

Last Update: March 2025

When to Use Non-Relational?

Working with data that changes frequently

Cloud computing

Promoting developer productivity

A very common question when you are considering non-relational databases is when is it appropriate to use them.

When to Use Non-Relational?

Non-relational, particularly the document model, is well suited to polymorphic data that can change frequently.

The document model and non-relational as a whole also supports the easy mapping from the data to the programming language constructs enabling greater developer productivity as there is less code to translate between the database and the application.

Non-relational systems are typically cloud-native and designed as distributed systems.

Let’s discuss some of the aspects around when to use non-relational databases.

When to use Non-Relational?

Non-relational, particularly the document model, is well suited to polymorphic data that can change frequently.

The document model and non-relational as a whole also supports the easy mapping from the data to the programming language constructs enabling greater developer productivity as there is less code to translate between the database and the application.

Non-relational systems are typically cloud-native and designed as distributed systems.

The document model allows for different shapes of data within the same collection, this means that documents with different fields can be present. It is sometimes described as holding multiple schema for the collection.

This is a key feature for enabling developer productivity as it provides rapid iteration of schema versions for data to co-exist within the same database and collection. This means that developers can rapidly change what fields are in a document and not worry about the impact or side effects that occur in the database. Essentially, the database is not a hurdle or an additional burden of work that must be additionally updated for example when a new field is added.

When to use Non-Relational?

Non-relational, particularly the document model, is well suited to polymorphic data that can change frequently.

The document model and non-relational as a whole also supports the easy mapping from the data to the programming language constructs enabling greater developer productivity as there is less code to translate between the database and the application.

Non-relational systems are typically cloud-native and designed as distributed systems.

Another aspect of non-relational databases is that they often offer exact (or close to exact) mappings to what objects the developer desire to use in their application code. This means that the data can be transferred as-is directly to the application without requiring any additional mapping. In the case of relational databases, information for a single object may reside in several tables which then need to be JOINed before being passed to the application. This is the object mapping stage.

When to use Non-Relational?

Non-relational, particularly the document model, is well suited to polymorphic data that can change frequently.

The document model and non-relational as a whole also supports the easy mapping from the data to the programming language constructs enabling greater developer productivity as there is less code to translate between the database and the application.

Non-relational systems are typically cloud-native and designed as distributed systems.

A known pain point for relational databases stems from the initial focus on scaling vertically, where additional resources were added to the machine or a large machine as used to support scaling the database.

Non-relational systems took this pain point and deliberately focused on scaling horizontally, where additional machines were added to the existing machine(s) when scaling the database. This scaling approach has simplified any changes required for these databases to support multiple public cloud providers. It has also focused non-relational systems, firstly on being virtualisation friendly, and more recently on being container friendly, as two key provisioning technologies that assist in deploying the database.

Quiz

Quiz

Which of the following are true for non-relational databases? More than one answer choice can be correct.

- A. Deals with polymorphic data.
- B. Requires more code to translate between application and database.
- C. Requires no object mapping language.
- D. Works well with rapidly changing data.

Quiz

Which of the following are true for non-relational databases? More than one answer choice can be correct.

- ✓ A. Deals with polymorphic data.
- ✗ B. Requires more code to translate between application and database.
- ✓ C. Requires no object mapping language.
- ✓ D. Works well with rapidly changing data.

Quiz

Which of the following are true for non-relational databases? More than one answer choice can be correct.

- ✓ A. Deals with polymorphic data.
- ✗ B. Requires more code to translate between application and database.
- ✓ C. Requires no object mapping language.
- ✓ D. Works well with rapidly changing data.

In non-relational databases, you are able to store differently shaped data in the same collection. Differently shaped data is a key advantage when compared to relational databases

CORRECT: Deals with polymorphic data - In non-relational databases, you are able to store differently shaped data in the same collection. Differently shaped data is a key advantage when compared to relational databases

Quiz

Which of the following are true for non-relational databases? More than one answer choice can be correct.

- ✓ A. Deals with polymorphic data.
- ✗ B. Requires more code to translate between application and database.
- ✓ C. Requires no object mapping language.
- ✓ D. Works well with rapidly changing data.

Less code is required with non-relational databases as the data is stored in constructs that are very similar to the programming language constructs used in applications

INCORRECT: Requires more code to translate between application and database - Less code is required with non-relational databases as the data is stored in constructs that are very similar to the programming language constructs used in applications

Quiz

Which of the following are true for non-relational databases? More than one answer choice can be correct.

- ✓ A. Deals with polymorphic data.
- ✗ B. Requires more code to translate between application and database.
- ✓ C. Requires no object mapping language.
- ✓ D. Works well with rapidly changing data.

The majority of objects used by a programmer will be in similar constructs whether in the application or in the database so no additional mapping is typically required

CORRECT: Requires no object mapping language - The majority of objects used by a programmer will be in similar constructs whether in the application or in the database so no additional mapping is typically required.

Quiz

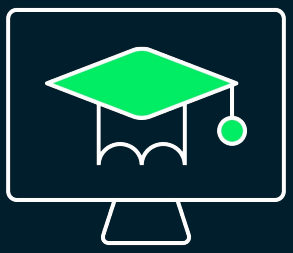
Which of the following are true for non-relational databases? More than one answer choice can be correct.

- ✓ A. Deals with polymorphic data.
- ✗ B. Requires more code to translate between application and database.
- ✓ C. Requires no object mapping language.
- ✓ D. Works well with rapidly changing data.

Each shape change in the structure of data requires a change to the database's corresponding schema if stored in a relational database. This isn't the case for non-relational databases.

CORRECT: Works well with rapidly changing data - Each shape change in the structure of data requires a change to the database's corresponding schema if stored in a relational database. This isn't the case for non-relational databases.

Continue Learning!



MongoDB University has free self-paced courses and labs ranging from beginner to advanced levels.

GitHub Student Developer Pack

Sign up for the MongoDB Student Pack to receive $50 in Atlas credits and free certification!

This concludes the material for this lesson. However, there are many more ways to learn about MongoDB and non-relational databases, and they are all free! Check out MongoDB's University page to find free courses that go into more depth about everything MongoDB and non-relational. For students and educators alike, MongoDB for Academia is here to offer support in many forms. Check out our educator resources and join the Educator Community. Students can receive $50 in Atlas credits and free certification through the GitHub Student Developer Pack.