LESSON

Relational vs. Non-Relational Databases

Google slide deck available here

Overview

Learning Objectives

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

- Identify the key features of relational databases.
- Describe what structured query language (SQL) is and why it is commonly used with relational databases.
- Identify the key features of a non-relational database.
- Understand the background to non-relational databases and the gap they filled in the history of data storage.

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:
25 minutes

Level:
Foundational

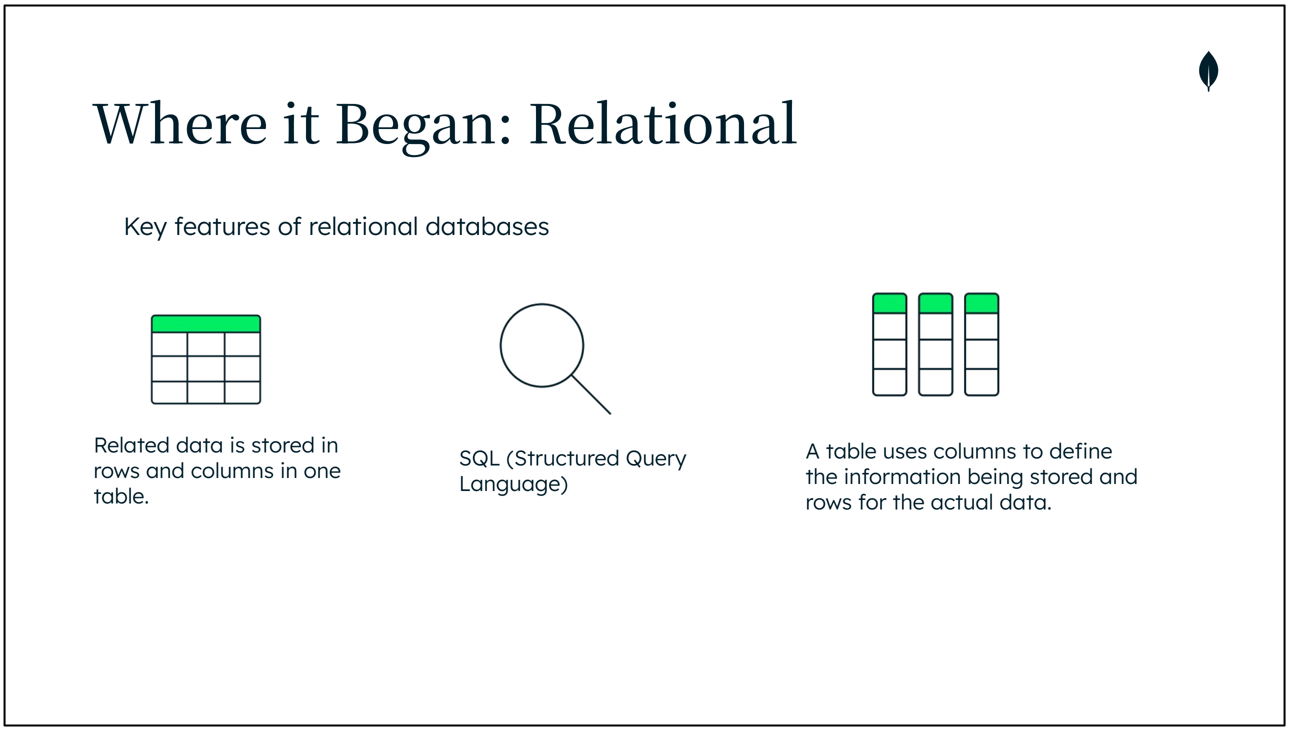
Prerequisites:
None

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

Where it Began



Where it Began: Relational

Key features of relational databases

Related data is stored in rows and columns in one table.

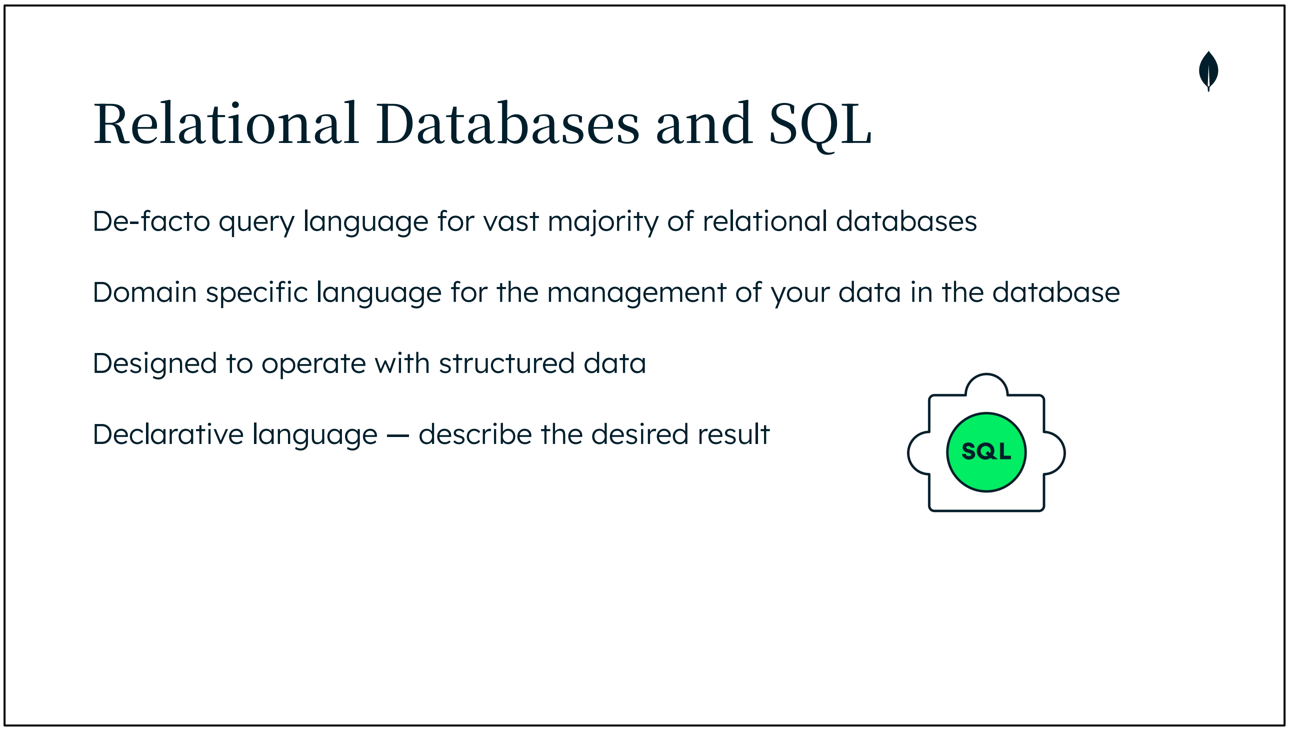
SQL (Structured Query Language)

A table uses columns to define the information being stored and rows for the actual data.

To understand non-relational databases, or “NoSQL” databases, we first need to look at SQL or relational databases.

Key features of relational databases:

- Modeled similarly to an excel spreadsheet with related data being stored in rows and columns in one table.
- SQL (Structured Query Language) is the most common way of interacting with relational database systems. Developers can write SQL queries to perform CRUD (Create, Read, Update, Delete) operations.
- A table uses columns to define the information being stored and rows for the actual data. Each table will have a column that must have unique values—known as the primary key. This column can then be used in other tables, if relationships are to be defined between them. When one table’s primary key is used in another table, this column in the second table is known as the foreign key.



Relational Databases and SQL

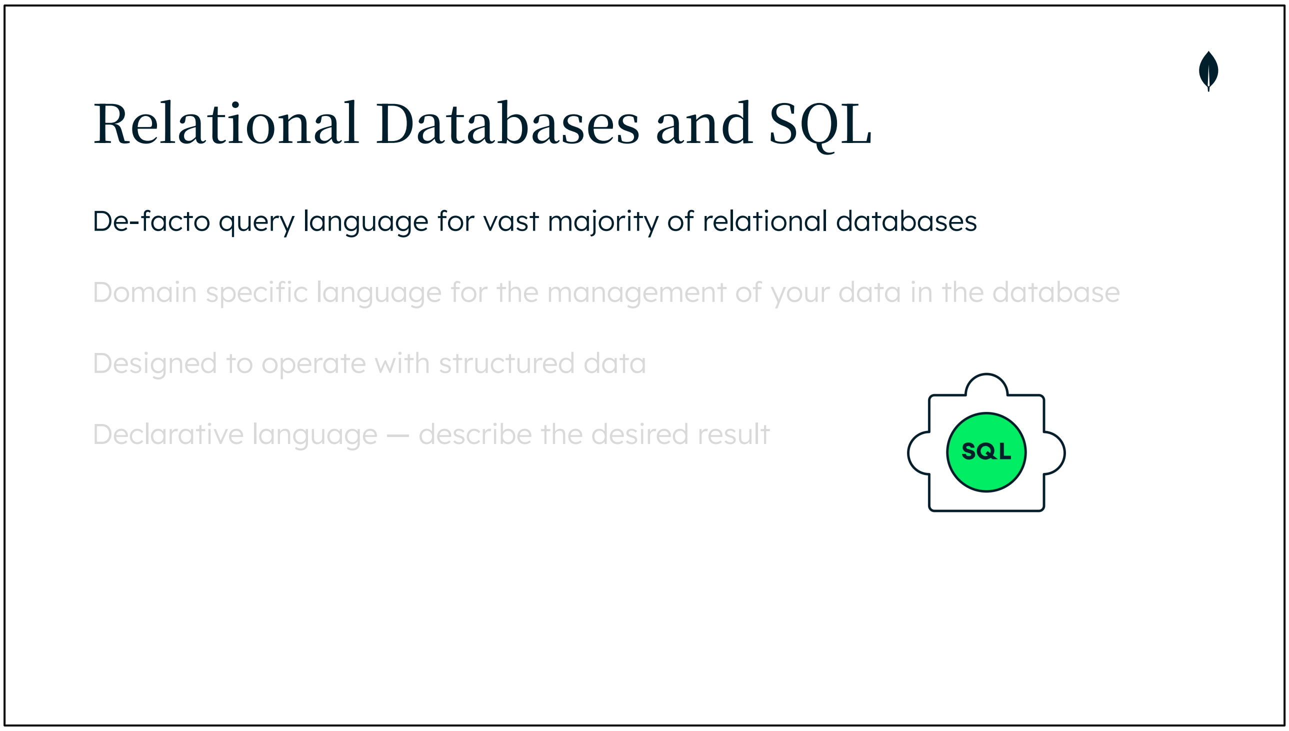
De-facto query language for vast majority of relational databases

Domain specific language for the management of your data in the database

Designed to operate with structured data

Declarative language — describe the desired result

Let’s look at the various aspects of Structured Query Language (SQL) to better understand the history and purpose of this query language.
It is nearly the last fifty years old in the original implementation and has influenced the design of the vast majority of databases in some form or other.



Relational Databases and SQL

De-facto query language for vast majority of relational databases

Domain specific language for the management of your data in the database

Designed to operate with structured data

Declarative language — describe the desired result

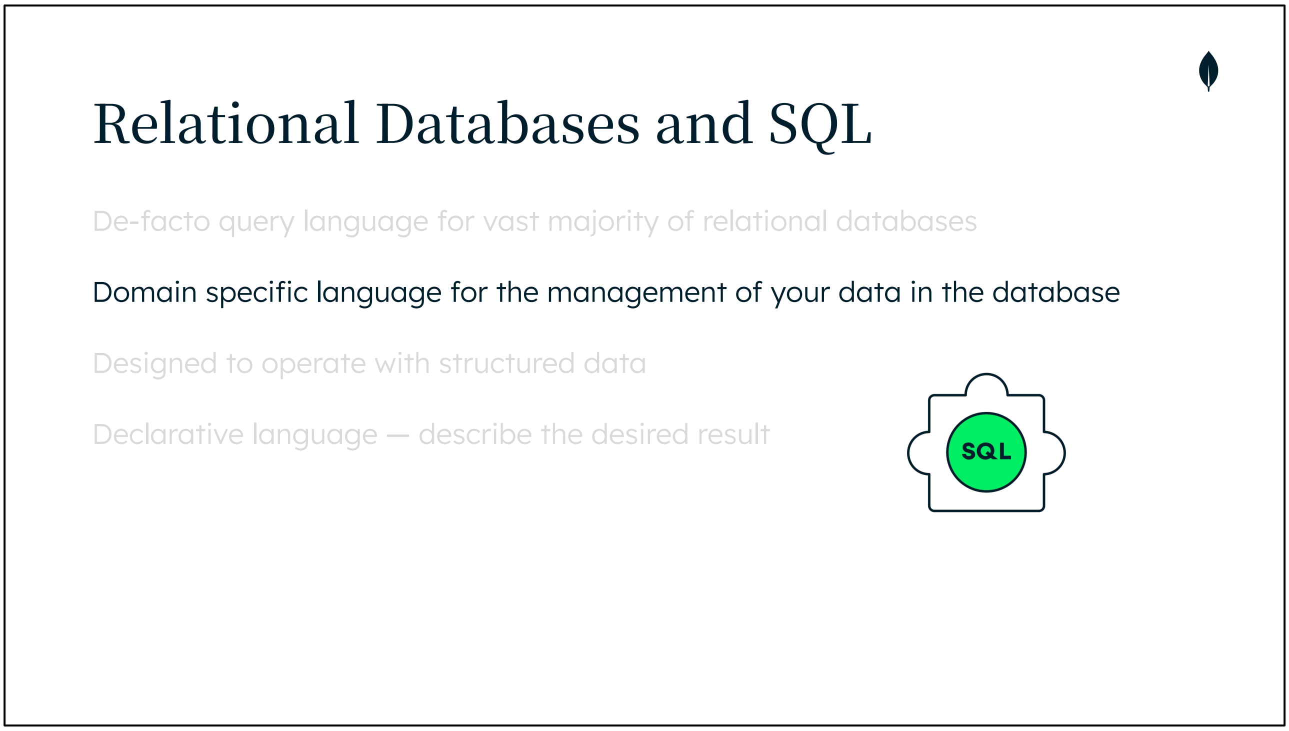
SQL

Structured Query Language, SQL is the most widely used and de-facto query language for relational databases.

It replaced many read-write APIs that were developed prior to it. It had two features that helped it become the main query language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. This helped further solidify its position as the main query language for relational databases.

IBM first used in their products in the early 1970s and Oracle followed in the late 1970s for their products. This ensured that two of the significant database software players had this as their de-facto query language. New entrants into the relational DB space, followed suit using this query language due to the familiarity of developers and DBAs with it.



Relational Databases and SQL

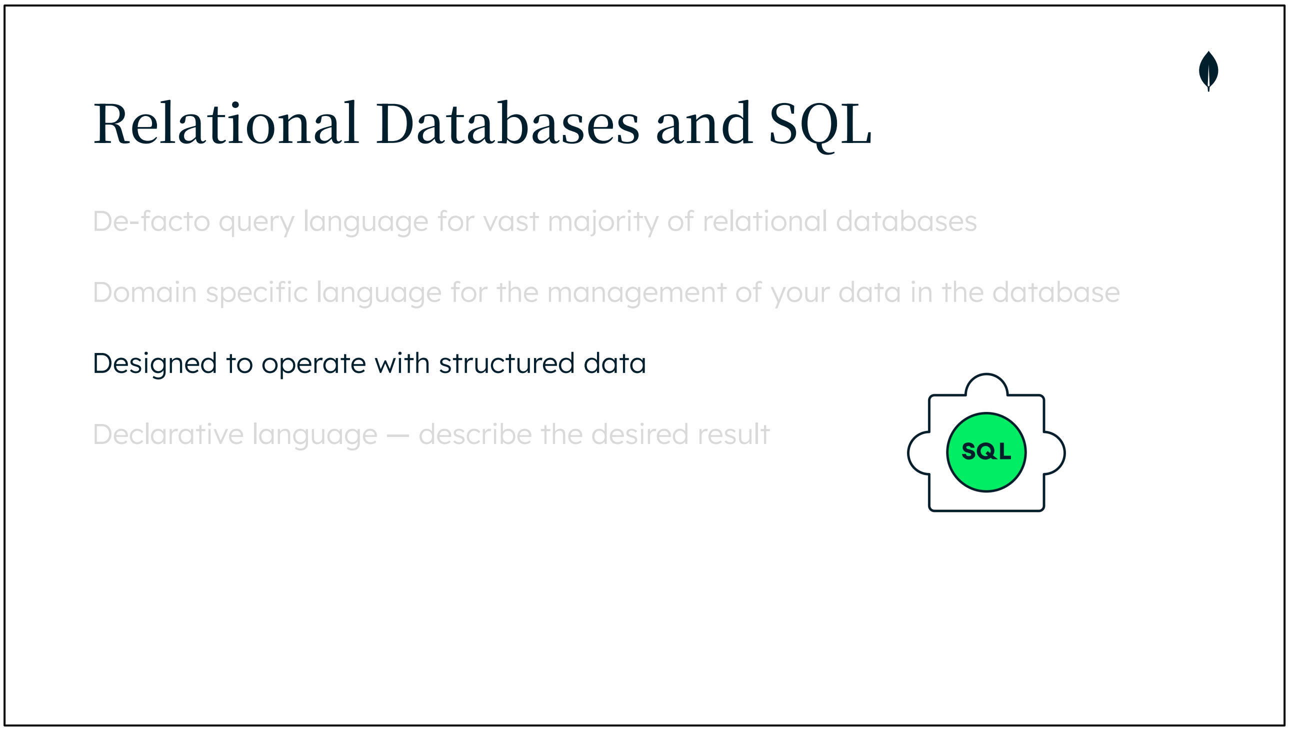
De-facto query language for vast majority of relational databases

Domain specific language for the management of your data in the database

Designed to operate with structured data

Declarative language — describe the desired result

SQL is a domain specific language that supports querying and management of your data. The focus for SQL as a language is towards specialized features for data management and on providing an implementation of Edward F. Codd's relational model which uses relational algebra and tuple relational calculus.



Relational Databases and SQL

De-facto query language for vast majority of relational databases

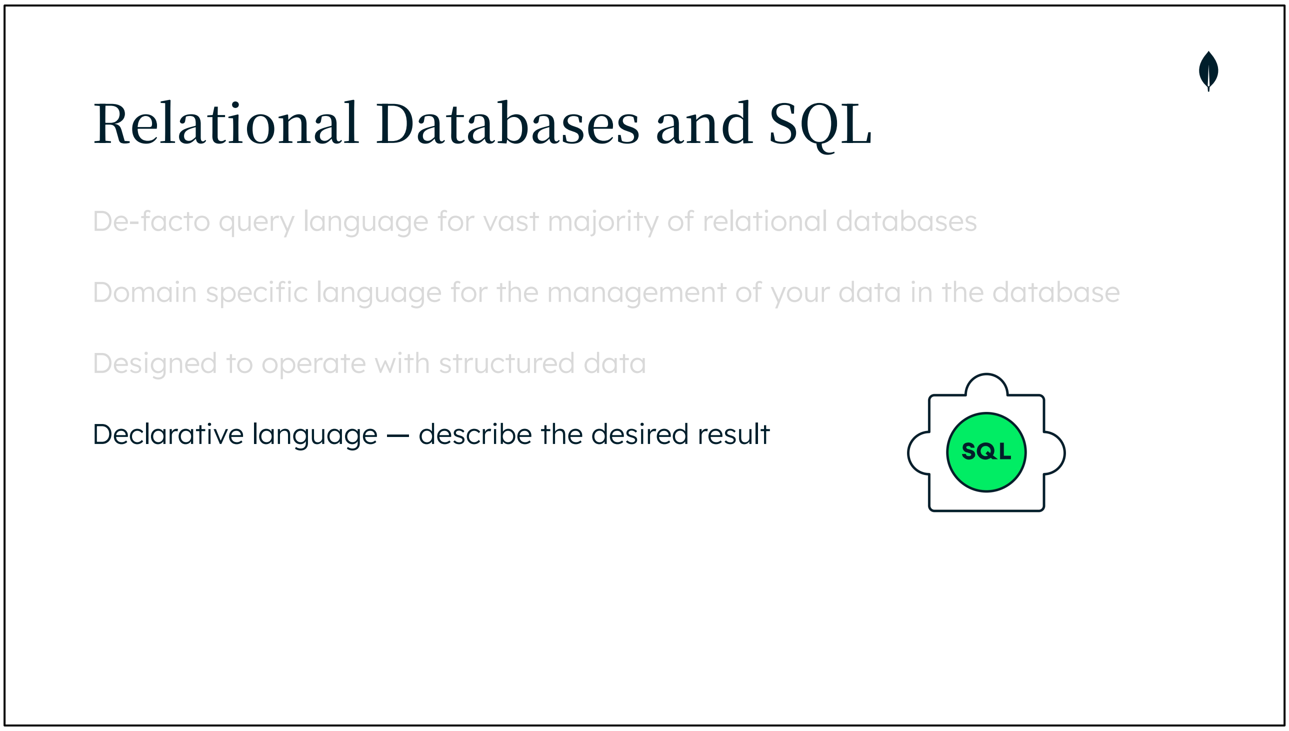
Domain specific language for the management of your data in the database

Designed to operate with structured data

Declarative language — describe the desired result

SQL

SQL was specifically designed to work with tabular data or records as they are also know. These are often categorised as structured data. This is because there can only be a single schema or structure on the data within a relational database.



Relational Databases and SQL

De-facto query language for vast majority of relational databases

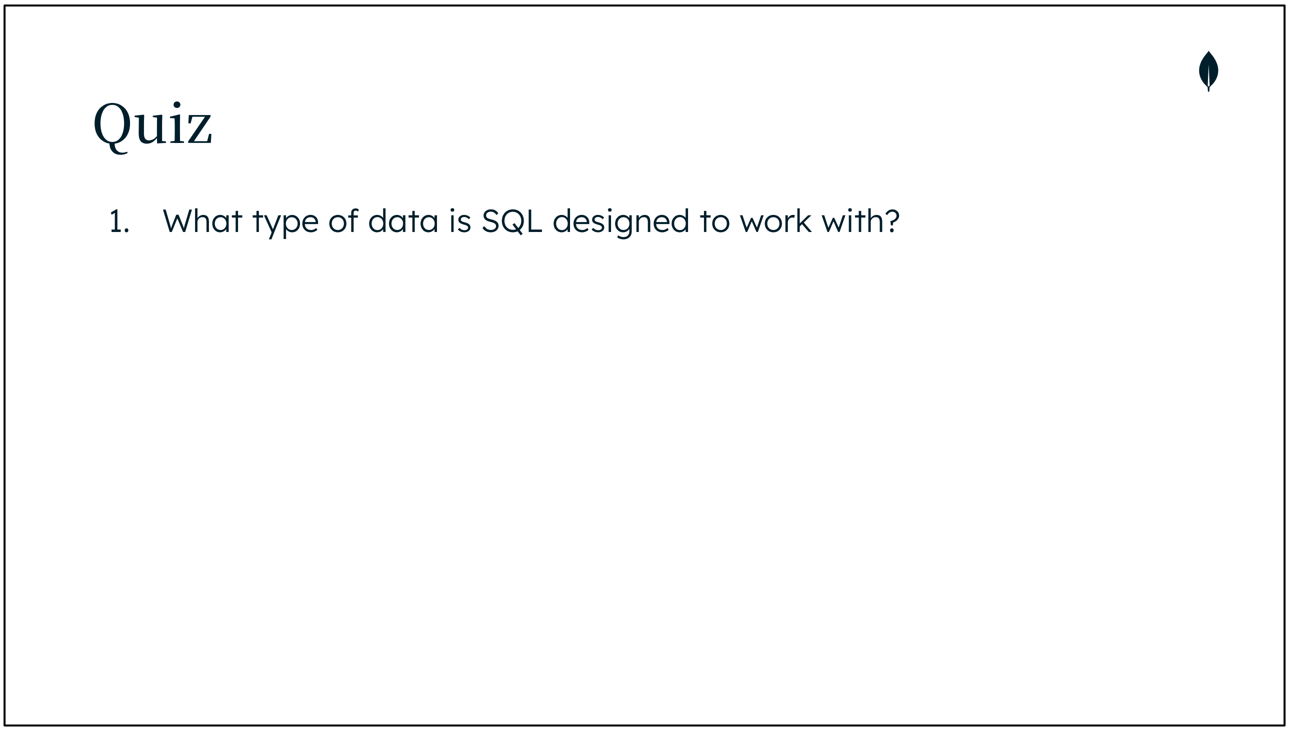
Domain specific language for the management of your data in the database

Designed to operate with structured data

Declarative language — describe the desired result

SQL is a declarative language, this means that you describe in SQL syntax the desired result you wish from the query.
This means that as a language it defines the logic of a computation or query but does not describe the control flow related to the query or logic.

Quiz



Quiz

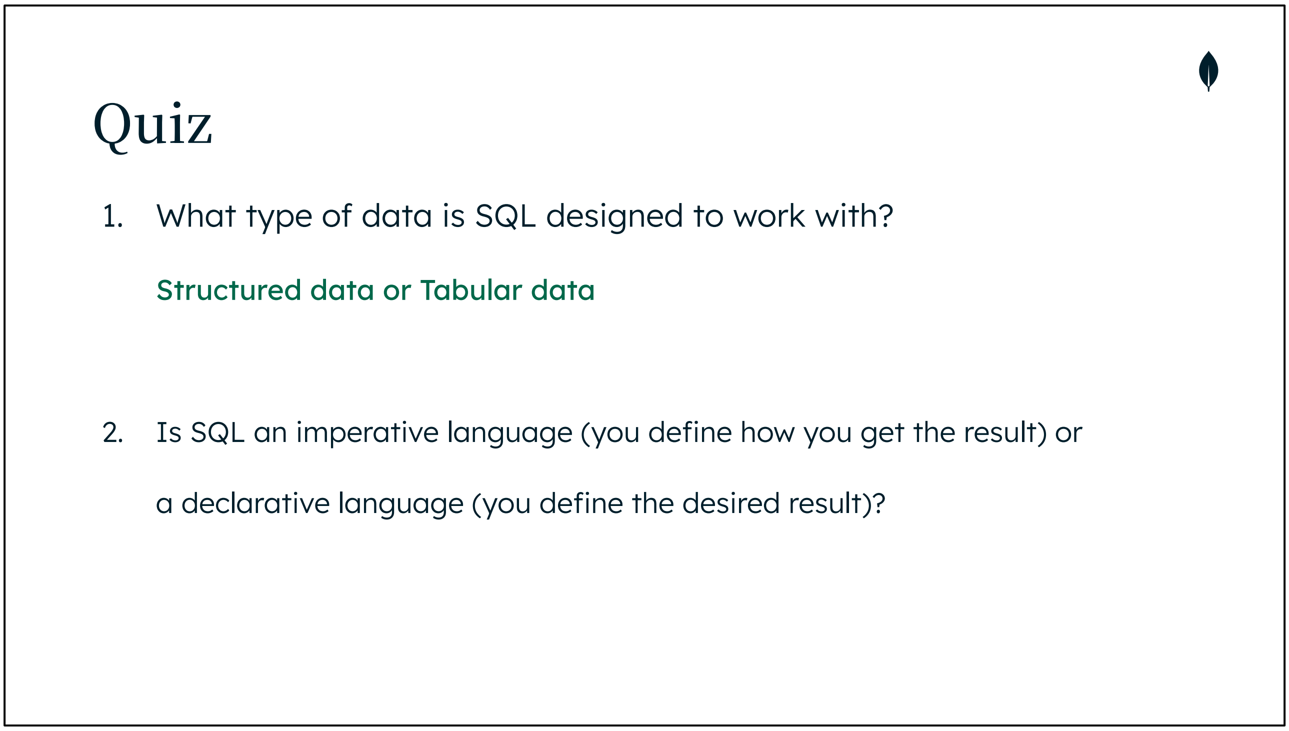
1. What type of data is SQL designed to work with?

Answer the above question

Quiz

1. What type of data is SQL designed to work with?

   Structured data or Tabular data



Quiz

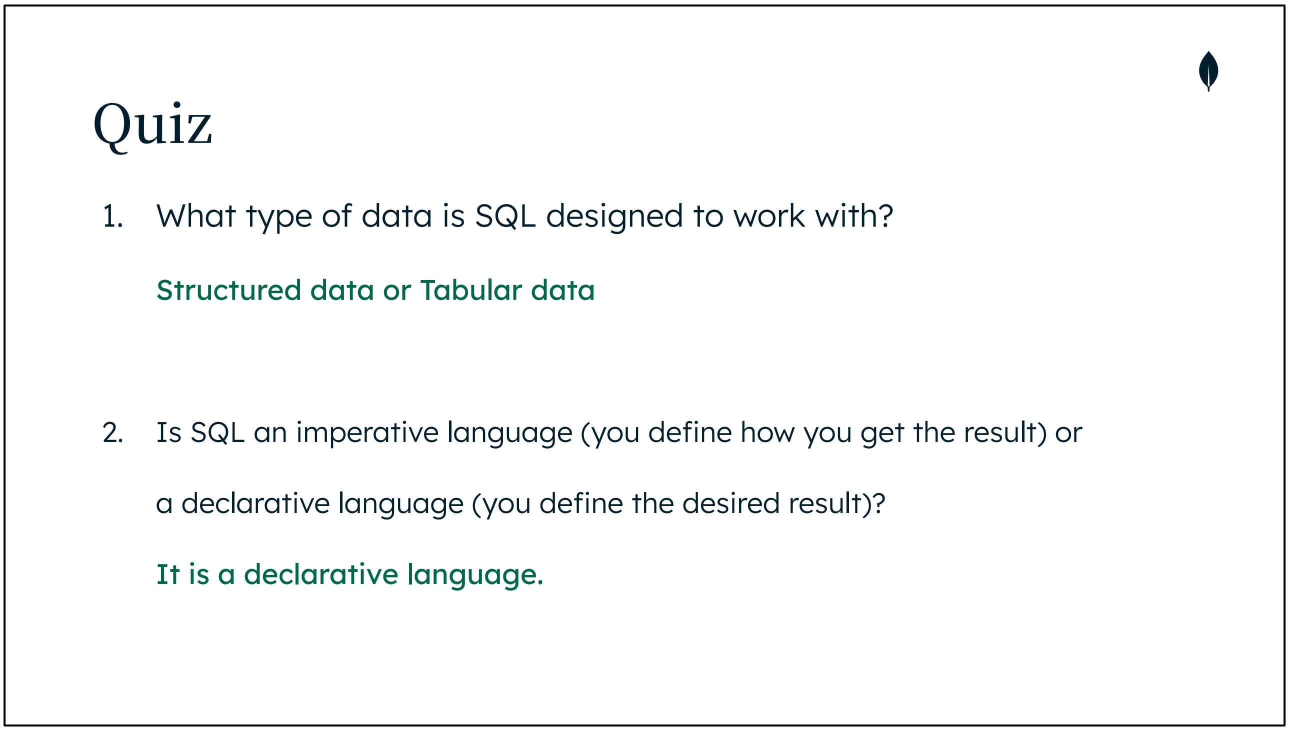
1. What type of data is SQL designed to work with?

   Structured data or Tabular data

2. Is SQL an imperative language (you define how you get the result) or

   a declarative language (you define the desired result)?

SQL is designed to work with structured data or tabular data as it is also known.



Quiz

1. What type of data is SQL designed to work with?

   Structured data or Tabular data

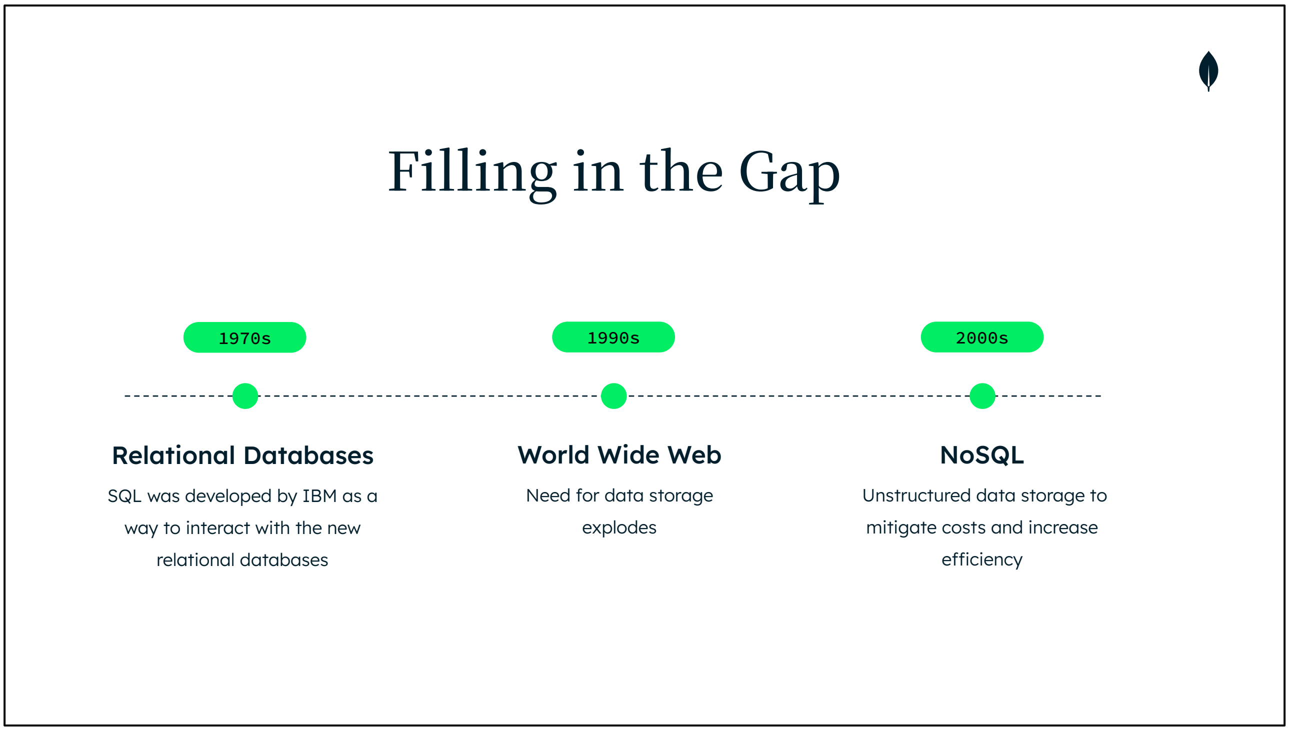
2. Is SQL an imperative language (you define how you get the result) or

   a declarative language (you define the desired result)?

   It is a declarative language.

SQL is an declarative language where you describe using SQL syntax the desired result you wish from the query. SQL defines only the logic of a computation or query, it does not describe the control flow related to the query or logic. Languages that describe the control flow explicitly are categorised as imperative languages.

Overview of Non-Relational Databases



Filling in the Gap

1970s — **Relational Databases**
SQL was developed by IBM as a way to interact with the new relational databases

1990s — **World Wide Web**
Need for data storage explodes

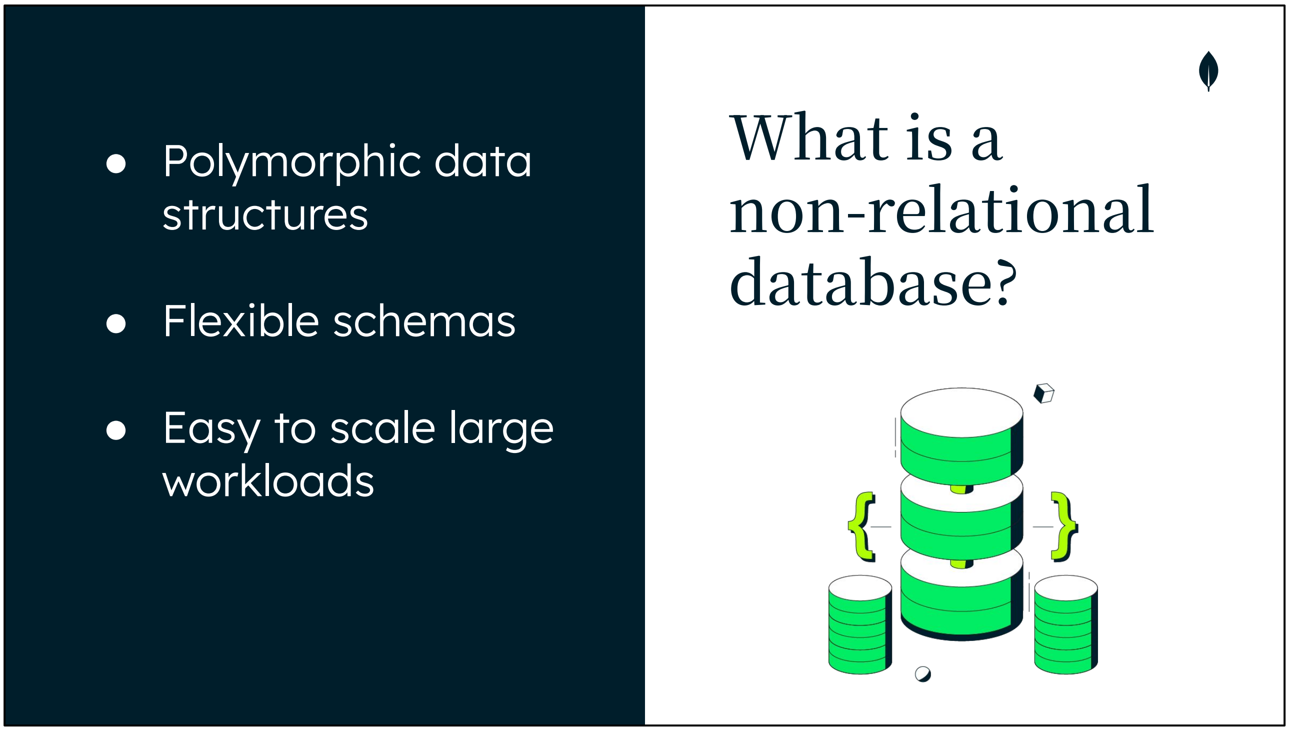
2000s — **NoSQL**
Unstructured data storage to mitigate costs and increase efficiency

When traditional relational databases were introduced, they were able to handle the growth of the data size by running on bigger machines.

With the emergence of the web came a huge data explosion that made it difficult to scale with hardware. You could not scale the database by running it on a bigger server, so companies were left to horizontally scale by distributing data across multiple servers or by running on more powerful servers. However, these scaling options were often complex and costly to maintain.

1. Relational Databases: To mitigate the cost of the first navigational databases and allow for searching, E.F Codd released his paper on a new way of storing data, relational. SQL was added to the field.
2. World Wide Web: The invention of the web fueled the demand for client-server database systems and high efficiency. Companies forced to scale use more servers at a high cost.
3. NoSQL: NoSQL (non-relational) databases were created to allow for faster processing of larger, more varied datasets. Emphasis on flexibility.

To fix the problem, various technology and software companies introduced new databases referred to as NoSQL or non-relational.



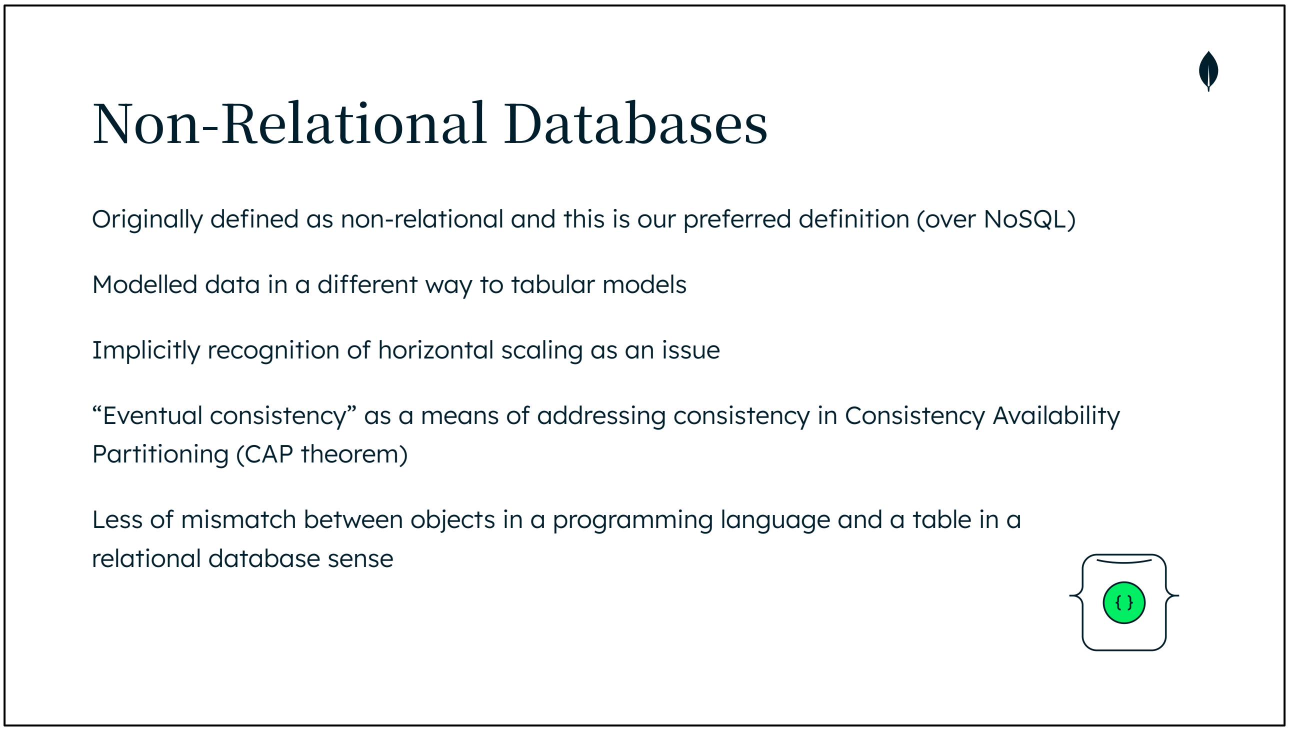
What is a non-relational database?

- Polymorphic data structures
- Flexible schemas
- Easy to scale large workloads

Non-relational databases differ from relational databases in that they do not store data in a tabular form.

Instead, non-relational databases might be based on data structures like documents, graphs, or dictionaries. NoSQL databases also come in a variety of types based on their data model.

They provide flexible schemas and scale easily with large amounts of data and high user loads. They were designed when it was expected that data would be partitioned across multiple machines to scale, in contrast to relational databases which assumed the data would stay on a single machine.



Non-Relational Databases

Originally defined as non-relational and this is our preferred definition (over NoSQL)

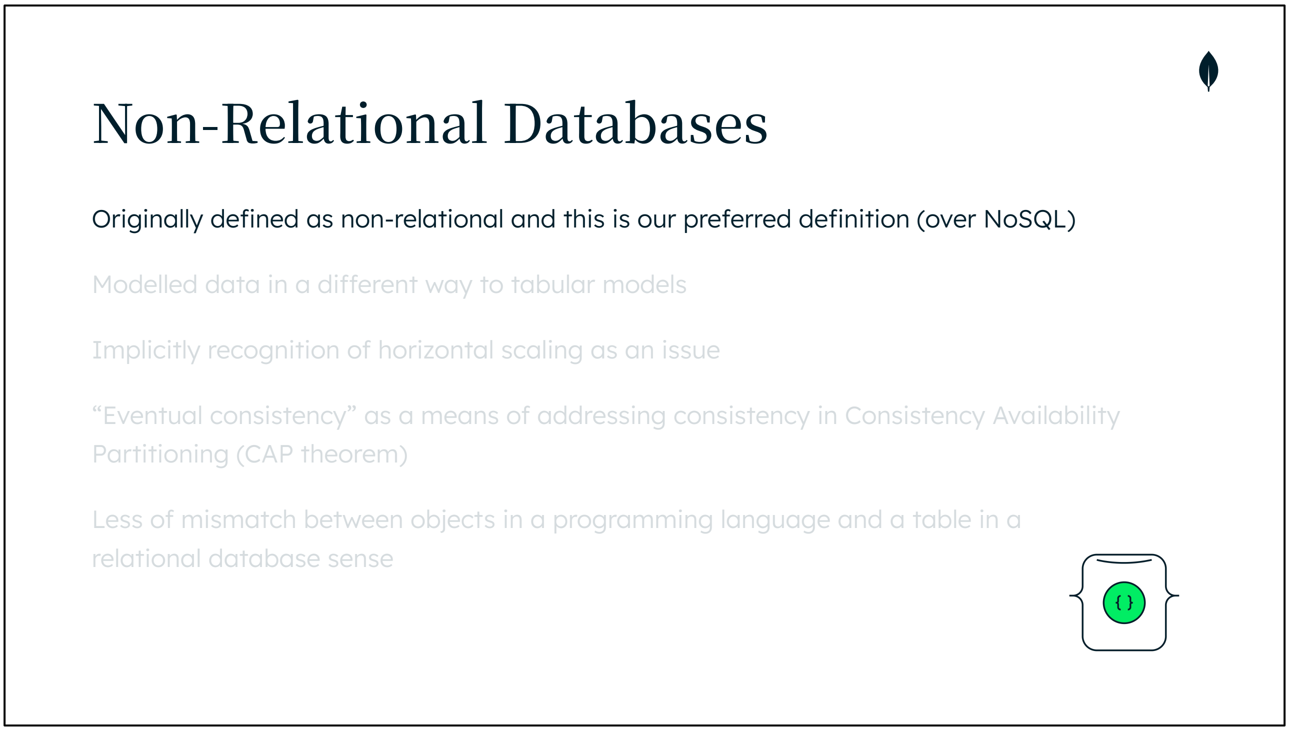
Modelled data in a different way to tabular models

Implicitly recognition of horizontal scaling as an issue

“Eventual consistency” as a means of addressing consistency in Consistency Availability Partitioning (CAP theorem)

Less of mismatch between objects in a programming language and a table in a relational database sense

Let’s look at the various aspects of non-relational databases to better understand the history and purpose of this data model.



Non-Relational Databases

Originally defined as non-relational and this is our preferred definition (over NoSQL)

Modelled data in a different way to tabular models

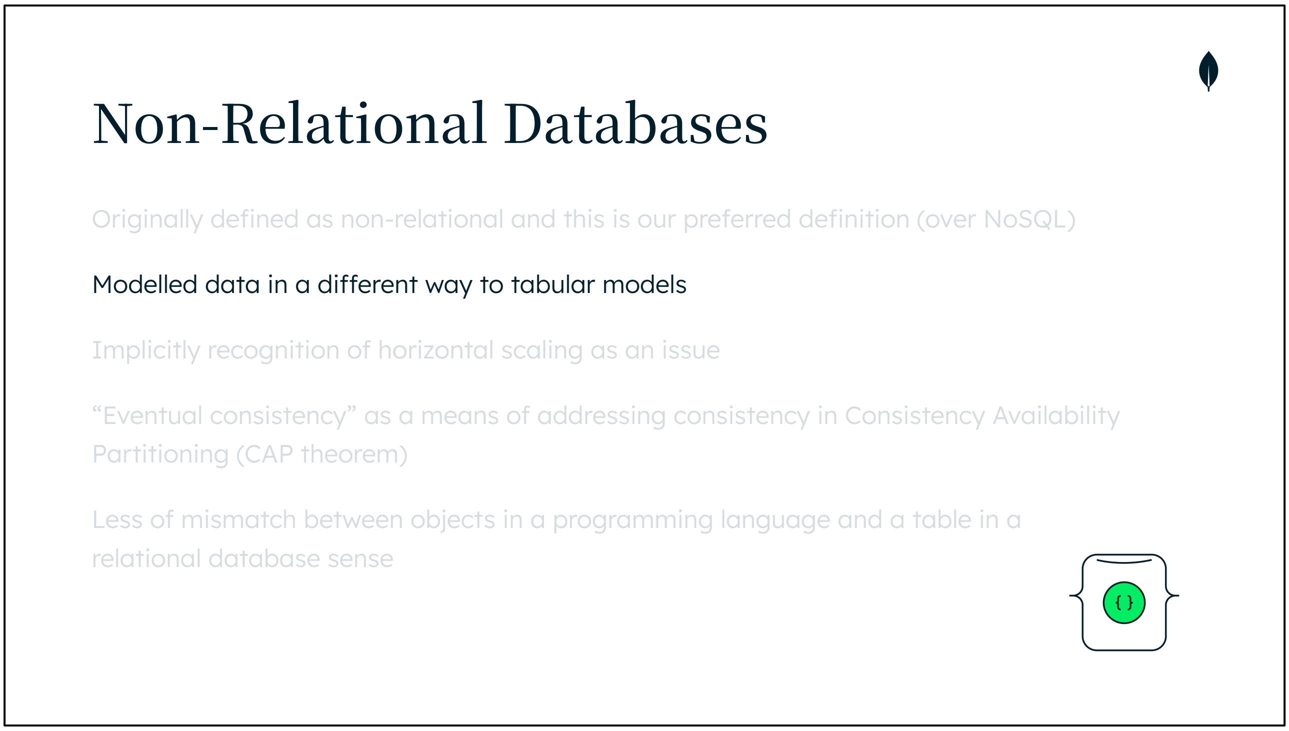
Implicitly recognition of horizontal scaling as an issue

“Eventual consistency” as a means of addressing consistency in Consistency Availability Partitioning (CAP theorem)

Less of mismatch between objects in a programming language and a table in a relational database sense

Query languages other than SQL were originally described as non-relational, later this was modified to “not just SQL”. That said we (at MongoDB) prefer to define them as non-relational.
The definition was broadened as these query languages may support SQL query languages or sit beside SQL databases in polyglot persistence situations.

Polyglot persistence is where multiple different types of specialized databases are used to store types of data, graph data being stored in a graph database, whilst records being stored in a relational database. The belief behind this reasoning is that by using the most appropriate specialized database for the specific type of data would would be more performant than having a single database (general purpose) try to store all kinds of data.



Non-Relational Databases

Originally defined as non-relational and this is our preferred definition (over NoSQL)

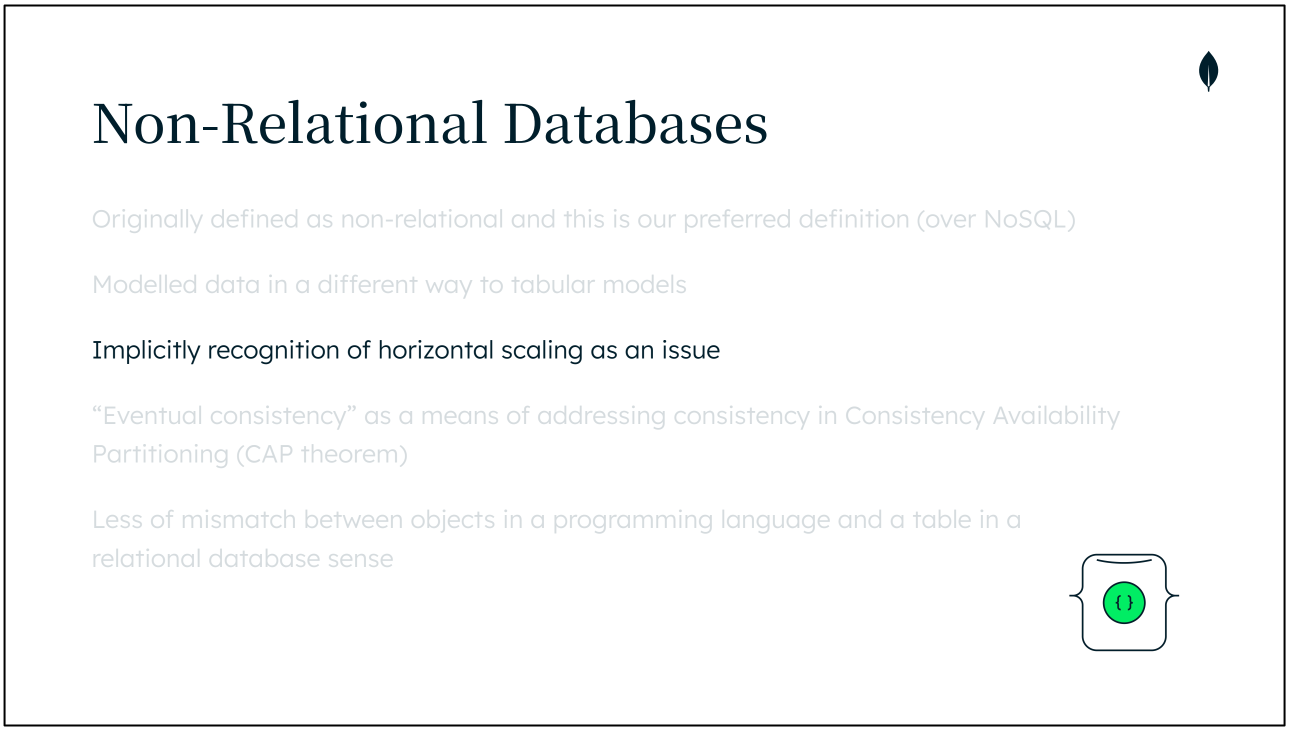
Modelled data in a different way to tabular models

Implicitly recognition of horizontal scaling as an issue

“Eventual consistency” as a means of addressing consistency in Consistency Availability Partitioning (CAP theorem)

Less of mismatch between objects in a programming language and a table in a relational database sense

Non-relational databases typically has mechanisms for storage and retrieval of data which are modeled in a different way to tabular models.
In MongoDB’s case, we’ll look at BSON and JSON for the storage and for the presentation respectively as well as the document model.



Non-Relational Databases

Originally defined as non-relational and this is our preferred definition (over NoSQL)

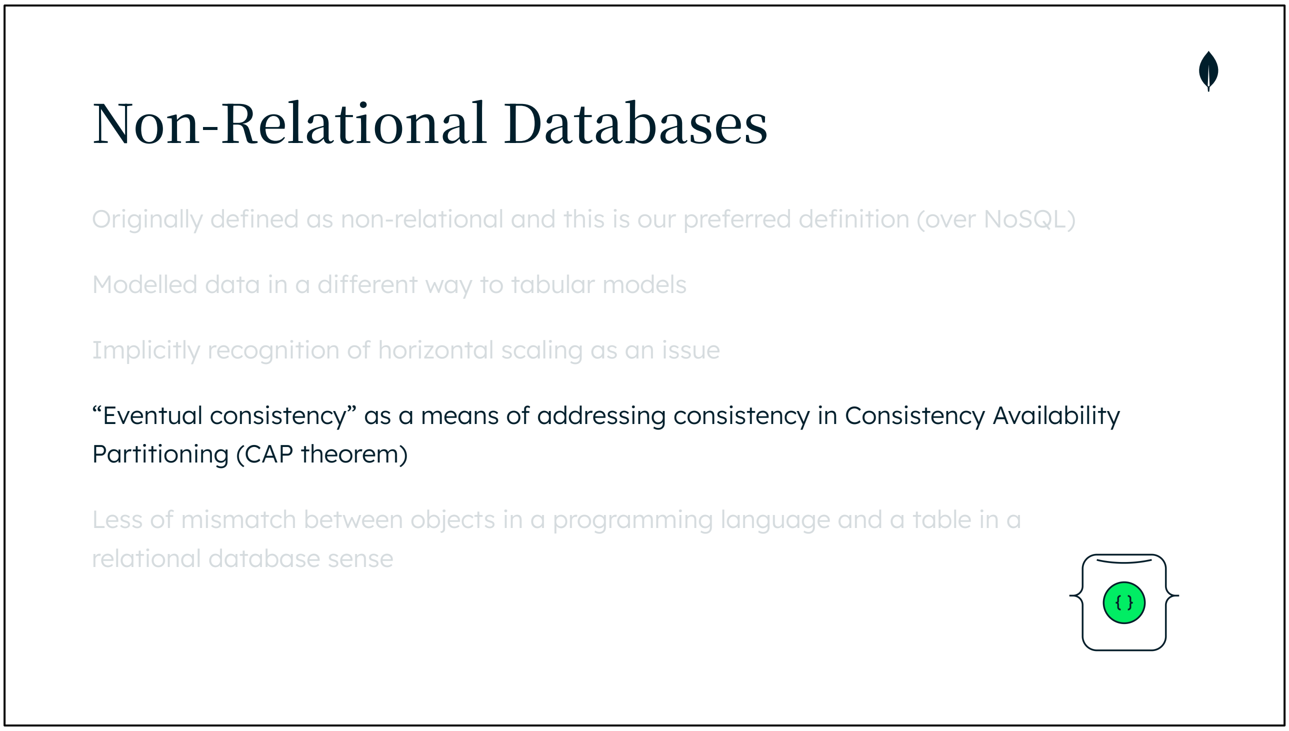
Modelled data in a different way to tabular models

Implicitly recognition of horizontal scaling as an issue

“Eventual consistency” as a means of addressing consistency in Consistency Availability Partitioning (CAP theorem)

Less of mismatch between objects in a programming language and a table in a relational database sense

Non-relational databases also implicitly included horizontal scaling of the data to clusters of machines as a clear requirement. This built on the experience of relational databases which typically used vertical scaling to move to bigger and bigger instances of a single machine. Both are valid approaches to scaling data, however vertical scaling can hit limits of hardware for single machines. In the case of horizontal scaling to many machines, these can be smaller and potentially cheaper to run than one very large single machine.



Non-Relational Databases

Originally defined as non-relational and this is our preferred definition (over NoSQL)

Modelled data in a different way to tabular models

Implicitly recognition of horizontal scaling as an issue

“Eventual consistency” as a means of addressing consistency in Consistency Availability Partitioning (CAP theorem)

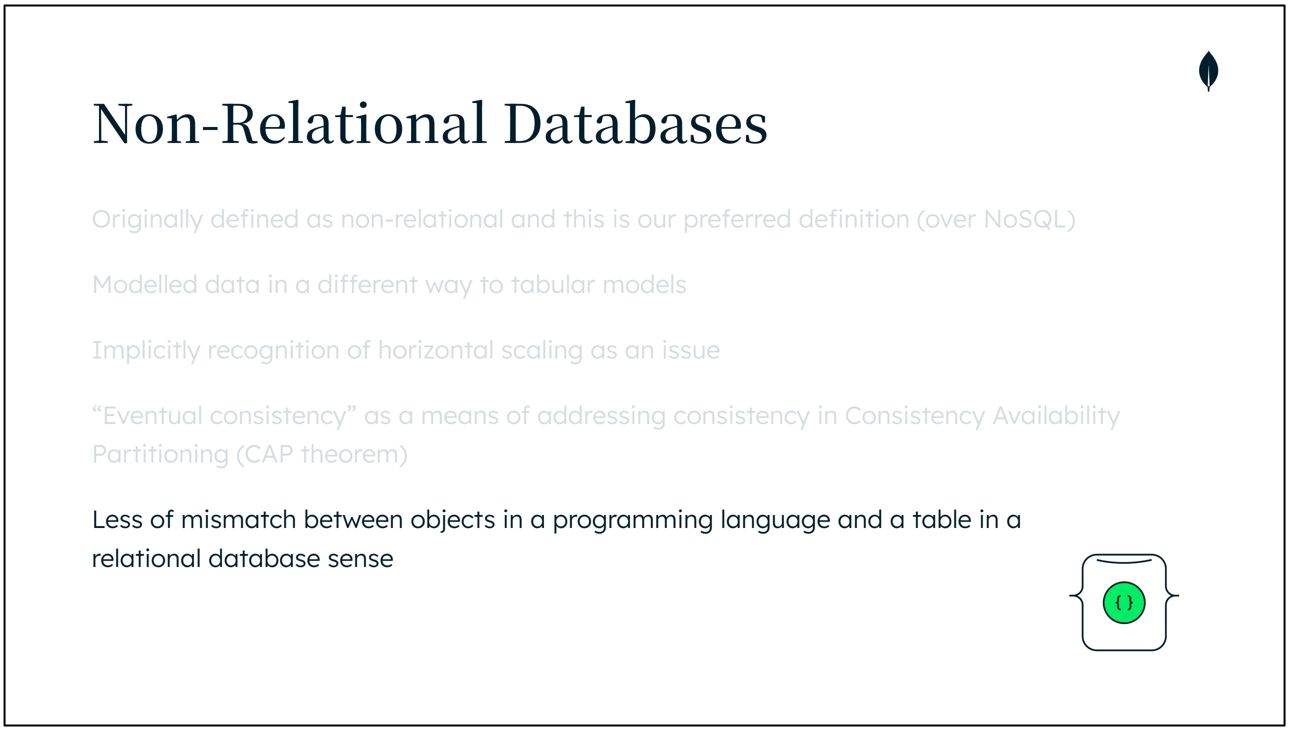
Less of mismatch between objects in a programming language and a table in a relational database sense

The CAP theorem or Brewer’s theorem states that it is impossible for any distributed data store to simultaneously provide more than two out of the three guarantees. These guarantees are consistency, availability, and partition tolerance. This consistency differs from that in the definition of ACID, in the CAP definition it defines that any read should receive the most recent written data or an error.

The implicit aspect of the CAP theorem is that in the case of a network partition, a choice must be made between consistency and availability.

Eventual consistency or optimistic replication as it is also known, is a model in distributed systems that informally guarantees that where no new updates are made to a given piece of data, that eventually all of the reads for that data item will return the last written update value for the data item.

MongoDB supported strong consistency for single documents with ACID transaction guarantees from very early versions. It is only in more recent versions since MongoDB 4.0 that ACID transaction guarantees were possible for multiple documents operations.



Non-Relational Databases

Originally defined as non-relational and this is our preferred definition (over NoSQL)

Modelled data in a different way to tabular models

Implicitly recognition of horizontal scaling as an issue

“Eventual consistency” as a means of addressing consistency in Consistency Availability Partitioning (CAP theorem)

Less of mismatch between objects in a programming language and a table in a relational database sense

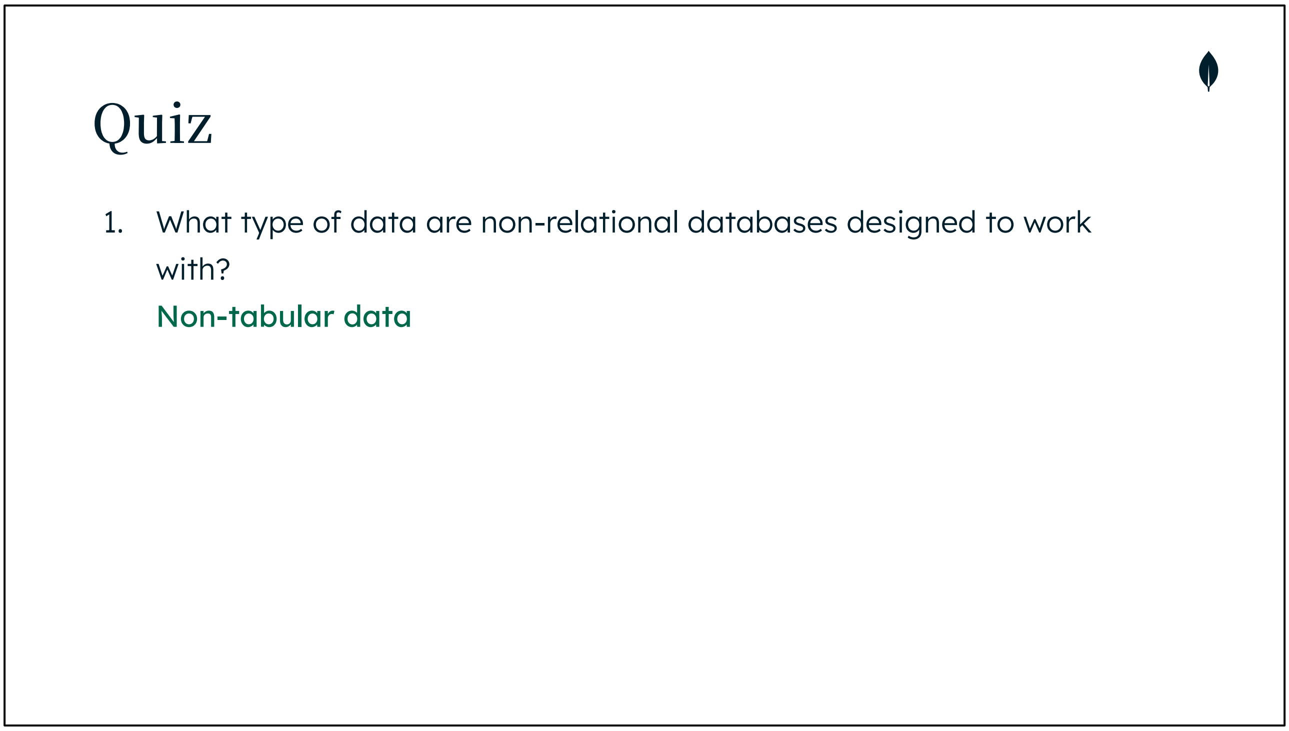
In terms of a mismatch, this is related to the issue that programmers prefer to deal with the concept of an object of say a car.
In relational databases, it might be that some of the car parts are kept in different tables, for instance a separate table with records may hold the data around the wheels for all cars. This means one or more joins may be required when using a relational database to create a single object holding a car’s information. In relational databases, this is often done through an object mapping layer.

In the case of MongoDB and particularly with documents, all of the related information for a car is typically stored in a single document. This is similar for other non-relational systems which mean that the data can typically be directly mapped from the database to the programming language without any object mapping required.

Quiz

Quiz

1. What type of data are non-relational databases designed to work with?



Quiz

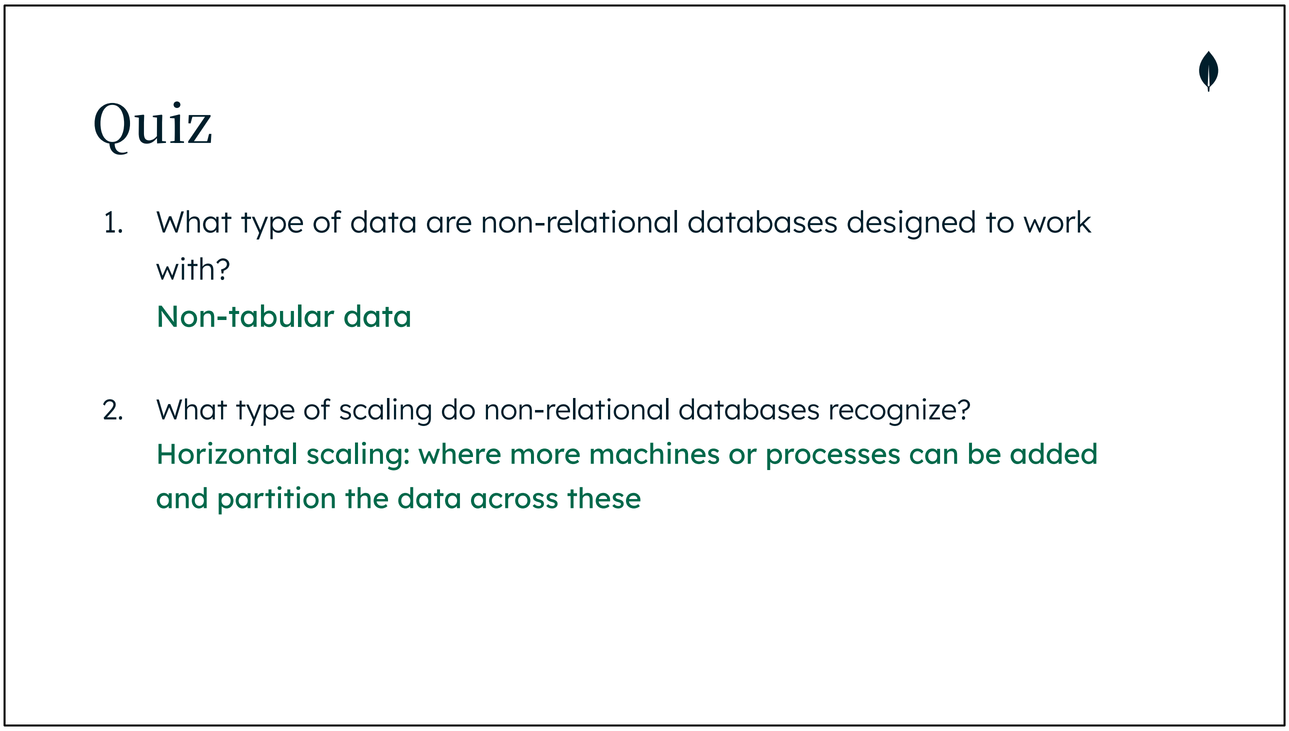
1. What type of data are non-relational databases designed to work with?
   Non-tabular data

Non-relational databases are designed to work with non-tabular or non-structured data, specifically in non-relational there is no hard requirement to have a single schema or single structure on the data. This allows for various different or multiple structures to exist in a given collection or table.

Quiz

1. What type of data are non-relational databases designed to work with?
   Non-tabular data

2. What type of scaling do non-relational databases recognize?



Quiz

1. What type of data are non-relational databases designed to work with?
   Non-tabular data

2. What type of scaling do non-relational databases recognize?
   Horizontal scaling: where more machines or processes can be added and partition the data across these

Non-relational databases recognised horizontal scaling as the type of scaling they should support. They can definitely benefit from vertical scaling where the capacity of the machine they are based on is increased. However, non-relational databases learnt from problems in the earlier generation of relational databases where there was a finite vertical scaling possible due to the hardware constraints on any single given machine.

This led non-relational databases to recognise that adding more machines and partitioning data across multiple machines or instances was the key to successfully scaling. Hence, most non-relational databases focus primarily on horizontal scaling.

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.