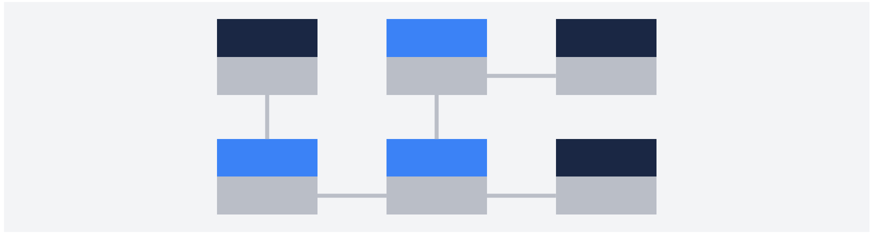
An introduction to NoSQL

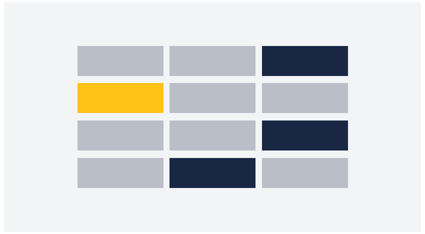
The characteristics of NoSQL databases

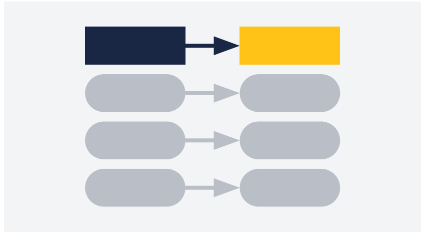
What are NoSQL databases?

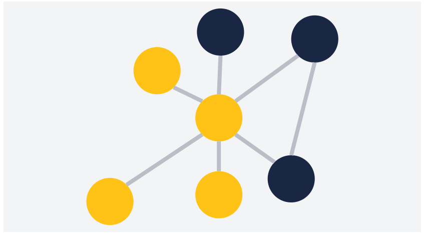
While traditional **SQL databases** excel at storing and retrieving **structured data**, they cannot effectively handle the **complexities of semi-structured** and **unstructured data**.

NoSQL database design is a response to the limitations of traditional SQL databases in handling **modern, dynamic data needs** which are fuelled by the continuous increase in the volumes and complexity of data.









An introduction to NoSQL

What are NoSQL databases?

SQL and NoSQL databases differ on more than just the type data structures they can handle; they have **different advantages**, **disadvantages**, and **uses**.

SQL databases	NoSQL databases
Primarily used for **structured** data.	Can handle various data structures including **structured**, **semi-structured**, and **unstructured** data.
Enforce a **fixed schema**; data structure changes may require migrations and potential downtime.	**Schema-less** or **schema-agnostic** so fields can be added or modified without any downtime.
Vertically scalable, typically requiring more powerful hardware as data increase.	**Horizontally scalable**, efficiently handling larger workloads by distributing tasks across its existing resources.
Excel in scenarios where **data consistency** is crucial, such as banking and e-commerce applications.	Well-suited where **rapid development**, **flexibility**, and **scalability** are priorities.

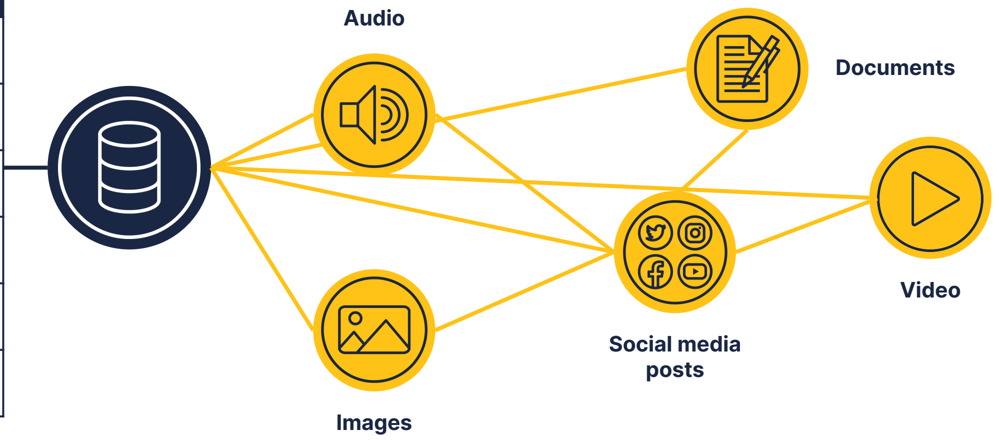
What are unstructured data?

While structured data are highly organised into rows and columns, **semi-structured** and **unstructured data** are **without a predefined format** and **cannot be organised into rows and columns**.

Structured data

Name	Region	Income_ group	Pop_n	Pop_u
Madagascar	Sub-Saharan Africa	Low income	27691.01953	38.534
Maji Ndogo	Sub-Saharan Africa	Low income	27628.1	36
Malawi	Sub-Saharan Africa	Low income	19129.95508	17.427
Malaysia	South Asia	Upper middle income	32365.99805	77.159996
Mexico	Latin America & Caribbean	Upper middle income	128932.75	80.730995
Mozambique	Sub-Saharan Africa	Low income	31255.43555	37.0739975

Semi-structured and unstructured data



Characteristics of NoSQL databases

NoSQL databases **prioritise high scalability** and **performance** over rigid data schemas, allowing for quick **development** and **iteration**.

Flexible schema

- Allows dynamic and schema-less data storage, which means **no predefined structure** for storing data is required.
- **Each data record** can have its own **unique structure**, allowing for greater flexibility in handling various data sources.
- This flexibility is advantageous when dealing with semi-structured and unstructured data sources, where the **schema may evolve over time**.

Scalability

- Can be **scaled horizontally**, which means they can handle massive amounts of data by distributing workloads across multiple servers or clusters.
- This scalability is advantageous in **modern applications** such as web applications and other **high-data-volume** and **high-performance requirements**.

Characteristics of NoSQL databases

High availability

- Ensures that **data remain accessible** in the event of server failures or network issues.
- This is achieved through replication and data distribution techniques, which **provide redundancy** and **fault tolerance**.

High performance

- Optimised for **high-throughput** (high capacity) tasks with **low latency** (little delay).
- Efficiently **handles large volumes** of read and write operations in **real-time**, which is critical for social media platforms and other real-time analytics.

Diverse data models

- They **do not rely on tabular**, **highly structured** data models but support **various models** such as key-value, document, columnar, and graph-based data models.
- Each model is **optimised** for **specific types of data** and **use cases**.

NoSQL databases offer a range of characteristics that make them **suitable for handling diverse types of data** and **accommodating** the **scalability** and **performance** demands of modern applications.

Their **schema-less**, **distributed**, and **non-relational** nature provides **flexibility** and **efficiency** in managing data, making them valuable in various use cases.

Types of NoSQL databases

Different NoSQL databases exist, each having its **own query languages** or **APIs** (Application Programming Interface) **tailored** to the specific **data model needs**.

NoSQL databases employ a **variety of query languages** and **data access patterns tailored** to their **specific data models** and **use cases**.

These query languages and patterns are **optimised** to provide **efficient** and **flexible** data retrieval, making them well-suited for their respective application domains.

In the next few slides, we'll look at four different NoSQL databases: **document-based** databases, **key-value** stores, **column-family** stores, and **graph** databases.

We'll take a look at how the same example would be implemented across these four NoSQL databases.

Example: *A water quality report for the province Dahabu in Maji Ndogo sent by Chidi Kunto, commented on by Amara and Bello.*

Types of NoSQL databases

Document-based databases

Data are stored in collections of documents.

MongoDB:

Data are typically stored in **JSON-like documents**, which can accommodate a variety of data structures in a single collection.

MongoDB Query Language (MQL) is primarily used for data retrieval from MongoDB, which allows us to perform queries, filter data, and retrieve documents that match specific criteria.

Example

```
{
   "_id": "2",
   "title": "Water quality report",
   "author": {
      "name": "Chidi Kunto",
      "email": "chidi@ndogo.gov"
   },
   "content": "This report provides an in-depth analysis of water samples.",
   "location": "Dahabu",
   "date_of_issue": "2023-09-20",
   "parameters": {
      "pH": 7.2,
      "Turbidity": "Low",
      "Chlorine": {
         "Free Chlorine": 0.5,
         "Total Chlorine": 1.2
      }
   },
   "comments": [
      {
         "user": "Amara",
         "comment_text": "Important information!"
      },
      {
         "user": "Bello",
         "comment_text": "I have some questions about the chlorine levels."
      }
   ]
}
```

Types of NoSQL databases

Key-value stores

Each piece of data is associated with a unique key.

Redis:

Data are stored with a **unique key**, and data can be retrieved by **specifying the key**. Various types of data are supported, including lists, sets, and hashes.

The **Command-Line Interface (CLI)** is used to store and retrieve data, and perform operations on data structures in the database.

We see that the SET command is used to set the value for each key. In the example, we see that the key `report:2:title` has the value `"Water quality report"`.

Example

```
SET report:2:title "Water quality report"
SET report:2:author "Chidi Kunto"
SET report:2:email "chidi@ndogo.gov"
SET report:2:content "This report provides an in-depth
analysis of water samples."
SET report:2:location "Dahabu"
SET report:2:date_of_issue "2023-09-20"
SET report:2:parameters:pH 7.2
SET report:2:parameters:Turbidity "Low"
SET report:2:parameters:Chlorine:FreeChlorine 0.5
SET report:2:parameters:Chlorine:TotalChlorine 1.2
SET report:2:comments:1:user "Amara"
SET report:2:comments:1:comment_text "Important
information!"
SET report:2:comments:2:user "Bello"
SET report:2:comments:2:comment_text "I have some questions
about the chlorine levels."
```

Types of NoSQL databases

Column-family stores

Data are organised into column groups, each containing multiple columns.

Apache Cassandra:

Data are stored in a column-oriented fashion, which allows the handling of **large volumes** of **distributed data** with high availability.

Uses a query language that is somewhat **similar to SQL** but optimised for distributed and columnar data.

Each row in the column family represents a unique dataset, and the row key (in this example, the key is **2**) **uniquely identifies this report** on water quality.

Example

```
Column Family: Water_quality_reports
```

Row key	Column name	Value
2	title	Water quality report
2	author	Chidi Kunto
2	email	chidi@ndogo.gov
2	content	This report provides…
2	location	Dahabu
2	date_of_issue	2023-09-20
2	parameters:pH	7.2
2	parameters:Turbidity	Low
2	parameters:Chlorine:FreeChlorine	0.5
2	parameters:Chlorine:TotalChlorine	1.2
2	comments:1:user	Amara
2	comments:1:comment_text	Important information!
2	comments:2:user	Bello
2	comments:2:comment_text	I have some questions…

Types of NoSQL databases

Graph databases

Data are represented and stored as nodes and relationships.

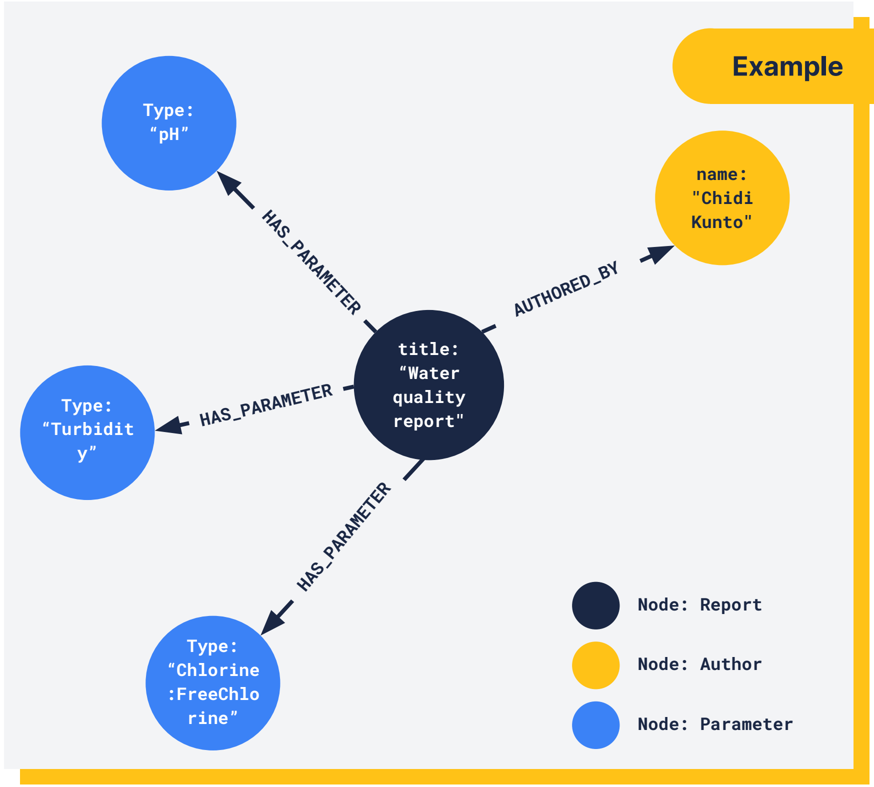
Neo4j:

Designed to manage data with **complex relationships** and **interconnected structures**.

Uses a **graph-specific query language** called **Cypher** for traversing and querying graph data by specifying patterns and relationships between nodes and edges.

Each node represents a **specific entity** in the data. In this example, the report is a node, as well as each of the parameters. **Relationships** (denoted by arrows) **connect nodes** and represent the associations between them.

Example



Use cases of NoSQL databases

Considering the characteristics of NoSQL databases, it’s unsurprising that they **excel in various scenarios** where adaptability, scalability, and real-time processing capabilities are required.

Real-time analytics

- **E-commerce product recommendations:** NoSQL databases can track user behaviour, preferences, and purchase history in real-time. This data are used to make product recommendations, offering personalised shopping experiences as the website is browsed.
- **Log and event analysis:** For applications like server monitoring, NoSQL databases can handle the constant stream of log and event data, allowing administrators to detect issues and make adjustments in real-time.

Content Management Systems (CMSs)

- **News websites:** NoSQL databases can store articles, images, videos, and comments. The flexible schema-less design accommodates different content types and allows for easy updates and revisions.
- **Content delivery:** NoSQL databases provide a robust content delivery framework that includes real-time updates, scalability for high traffic, personalised content delivery, and support for various delivery channels such as websites and mobile applications.

Use cases of NoSQL databases

Internet of Things (IoT) applications

- **Smart home devices:** IoT devices in a smart home generate data related to temperature, security, and energy consumption. NoSQL databases can store and analyse this data in real-time, enabling homeowners to control and optimise their environments.
- **Fleet management:** In logistics and transportation, IoT devices in vehicles provide real-time information on location, fuel consumption, and engine health. NoSQL databases can efficiently handle this data for tracking and maintenance.

Social media platforms

- **Social network graphs:** Social media platforms leverage NoSQL graph databases to model and query intricate social relationships. For example, Facebook uses graph databases to represent connections between users and manage the news feed algorithm.
- **Hashtags and trends:** Twitter also uses NoSQL databases to track trending topics and hashtags in real-time, providing users with up-to-the-minute updates and personalised content.

Other **real-world applications** of NoSQL databases storing and managing massive amounts of data include **Netflix** for customer profiles and viewing histories, **Uber** for rider and driver profiles, trip histories, and real-time location data, and **Airbnb** for property listings and booking histories.

The disadvantages of NoSQL

While there are many advantages to using NoSQL databases, more **frequent adoption is limited** in many cases due to some of the **disadvantages**.

Lack of ACID transactions

Disadvantage: Many NoSQL databases **sacrifice full ACID** (Atomicity, Consistency, Isolation, Durability) transaction support in favour of performance and scalability.

Implication: Strict **data consistency cannot be guaranteed** and in applications where this is crucial, such as financial systems, additional mechanisms have to be put into place to ensure data integrity.

Limited complex queries

Disadvantage: NoSQL databases often have **limited support** for complex queries such as **joins** or **aggregations** across multiple datasets.

Implication: Data may need to be **de-normalised** or combined with other data stores to enable similar types of queries.

Learning curve

Disadvantage: Adopting NoSQL databases can come with a **steep learning curve** because different data models and query languages are required specific to the design.

Implication: Teams may **require upskilling** and **time** to become proficient in NoSQL databases, which can lead to longer development cycles and potential errors during implementation.

Key points to remember about NoSQL

Although SQL is a query language, **NoSQL is not a query language** but rather an approach to database design.

While SQL databases are primarily for structured data, **NoSQL databases** are for **structured**, **semi-structured**, and **unstructured** data.

Different NoSQL databases rely on **different data models** and **query languages**.