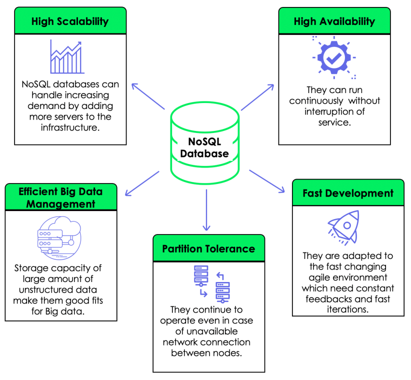
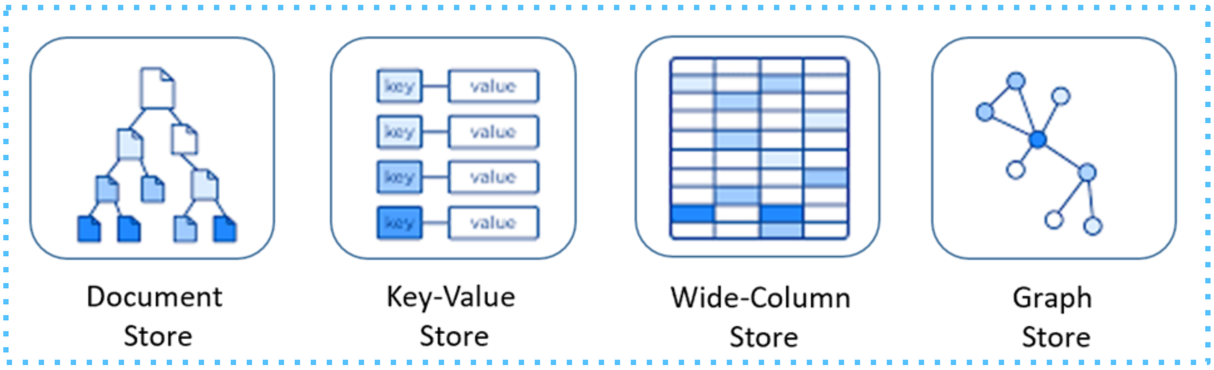
Overview on NoSQL Database


High Scalability
NoSQL databases can handle increasing demand by adding more servers to the infrastructure.
High Availability
They can run continuously without interruption of service.
NoSQL Database
Efficient Big Data Management
Storage capacity of large amount of unstructured data make them good fits for Big data.
Partition Tolerance
They continue to operate even in case of unavailable network connection between nodes.
Fast Development
They are adapted to the fast changing agile environment which need constant feedbacks and fast iterations.

*•NoSQL, also referred to as **"not only SQL" or "non-SQL",** is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data outside the traditional structures.*

- *NoSQL is also type of distributed database, which means that information is copied and stored on various servers, which can be remote or local.*
    - *This ensures availability and reliability of data. If some of the data goes offline, the rest of the database can continue to run.*
- *NoSQL databases are mostly open-source.*
- *Flexible Schema and It supports **Fast-paced Agile** development*
- *NoSQL database supports Key-Value pair(JSON) storage, it also supports other types like Column Store, Document Store, Graph databases.*



- *NoSQL databases prioritizes high performance, high availability, and scalability.*

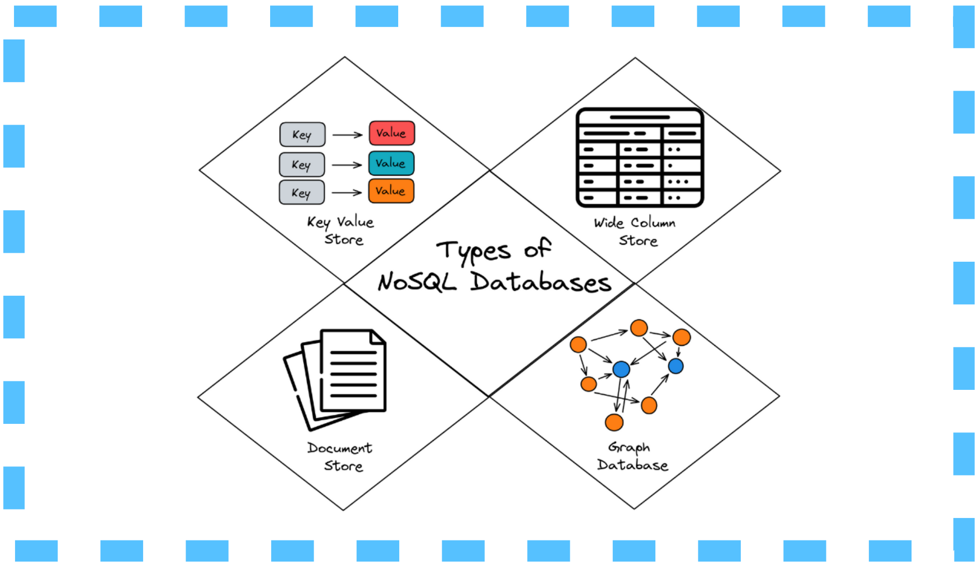
Features of NoSQL Databases

- ***Dynamic schema***
    - *NoSQL databases do not have a fixed schema and can accommodate changing data structures without the need for migrations or schema alterations.*
- ***Horizontal scalability***
    - *NoSQL databases are designed to scale out by adding more nodes to a database cluster, making them well-suited for handling large amounts of data and high levels of traffic.*
- ***Document-based***
    - *Some NoSQL databases, such as MongoDB, use a document-based data model, where data is stored in semi-structured format, such as JSON or BSON(binary encoded Javascript Object Notation (JSON)).*
- ***Key-value-based***
    - *Other NoSQL databases, such as Redis, use a key-value data model, where data is stored as a collection of key-value pairs.*
- ***Column-based***
    - *Some NoSQL databases, such as Cassandra, use a column-based data model, where data is organized into columns instead of rows.*
- ***Distributed and high availability***

   - *NoSQL databases are often designed to be highly available and to automatically handle node failures and data replication across multiple nodes in a database cluster.*

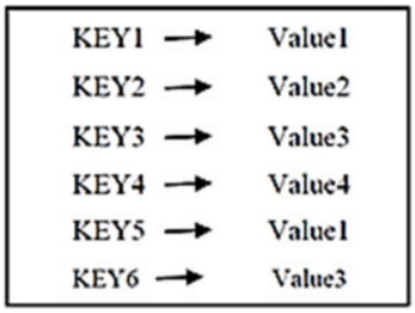
- ***Flexibility***
   - *NoSQL databases allow developers to store and retrieve data in a flexible and dynamic manner, with support for multiple data types and changing data structures.*
- ***Performance***
   - *NoSQL databases are optimized for high performance and can handle a high volume of reads and writes, making them suitable for big data and real-time applications.*

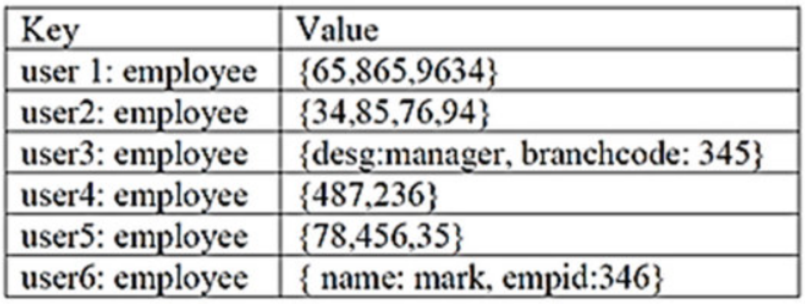
Types of NoSQL databases



- *There are several different NoSQL database systems due to variations in the way they manage and store schema-less data. I'll explain some of the common types below.*

Key-value databases





Key	Value
user 1: employee	{65,865,9634}
user2: employee	{34,85,76,94}
user3: employee	{desg:manager, branchcode: 345}
user4: employee	{487,236}
user5: employee	{78,456,35}
user6: employee	{ name: mark, empid:346}

- *Key-value databases are highly partitionable and allow horizontal scaling at a level that other types of NoSQL databases may not achieve.*
- *A key-value database stores data as a collection of key-value pairs in which a key serves as a unique identifier.*
- *Keys and values can be anything, ranging from simple objects to complex compound objects.*
- *Use cases such as gaming, ad tech, and IoT lend themselves particularly well to the key-value store data design*

Document databases

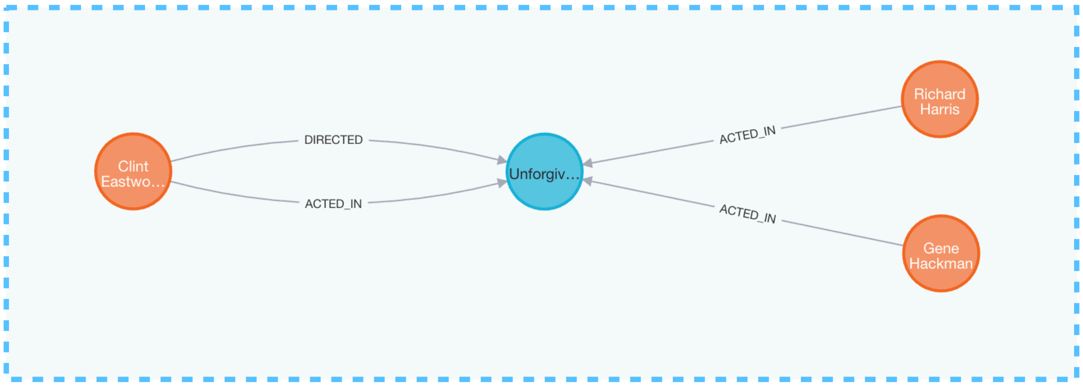
- *Document databases have the same document model format that developers use in their application code.*
- *They store data as JSON objects that are flexible, semi-structured, and hierarchical in nature..*
- *The flexible, semistructured, and hierarchical nature of documents and document databases allows them to evolve with applications' needs.*

- *The document database model works well with catalogs, user profiles, and content management systems, where each document is unique and evolves over time.*

```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  }
  "hobbies": ["surfing", "coding"]
}
```

Graph databases

- *Graph databases are purpose-built to make it easy to build and run applications that work with highly connected datasets.*
- *They use nodes to store data entities and edges to store relationships between entities.*
- *An edge always has a start node, end node, type, and direction. It can describe parent-child relationships, actions, ownership, and the like.*
- *There is no limit to the number and kind of relationships a node can have. You can use a graph database to build and run applications that work with highly connected datasets.*



- *Typical use cases for a graph database include social networking, recommendation engines, fraud detection, and knowledge graphs*

In-memory databases

- *While other non-relational databases store data on disk or SSDs, in-memory data stores are designed to eliminate the need to access disks.*
- *They are ideal for applications that require microsecond response times or have large spikes in traffic.*
- *You can use them in gaming and ad-tech applications for features like leaderboards, session stores, and real-time analytics.*

Search databases

- *A search-engine database is a type of non-relational database that is dedicated to the search of data content, such as application output logs used by developers to troubleshoot issues.*
- *They use indexes to categorize similar characteristics among data and facilitate search capability.*
- *Search-engine databases are optimized for sorting unstructured data like images and videos.*

When NoSQL should be used?

The decision to use a relational database versus a non-relational database is largely contextual, and it varies depending on the use case.

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

Advantages of NoSQL Databases

- *High scalability*
- *Distributed Computing*
- *Lower cost*
- *Can support easy updates to schemas.*
- *Can process large volumes of data at high speed*
- *Can store structured, semi-structured, and unstructured data*
- *No complicated Relationships*
- *Easy for developers*

Disadvantages of NoSQL Databases

- *Lack of SQL and standardization*

- *They don't support ACID transactions except for Mongodb.*
- *Large document size*
- *No Backup*

Differences between SQL and NoSQL

Features	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.
Data Storage	These databases are not suited for hierarchical data storage.	These databases are best suited for hierarchical data storage.
Query Support	These databases are best suited for complex queries	These databases are not so good for complex queries
Schema	These databases have fixed or static or predefined schema	They have a dynamic schema
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
Examples	Oracle, MySQL, Microsoft SQL Server, and PostgreSQL	Document: MongoDB and CouchDB, Key-value: Redis and DynamoDB, Wide-column: BigTable, Cassandra and HBase, Graph: Neo4j and Amazon Neptune