**NoSQL** (non-SQL/non-relational) dbs are a type of system for managing and storing data, and they're different from the traditional db. In regular databases, we use tables with fixed structures to organize information. However, NoSQL databases are more flexible. They can handle large amounts of data that might not fit neatly into those tables.

* These databases don't follow strict rules of traditional databases. Can handle various data models and structures.
* Additionally, they can scale horizontally, meaning they can grow and handle more data by adding more servers.
* They don't use the usual SQL language for querying data. Instead, they have their own ways of retrieving and manipulating information. Some well-known examples of NoSQL databases include MongoDB, Neo4J, and HyperGraphDB. Each of them has its own strengths and is suitable for different types of data and applications.

In traditional relational databases, the ACID (Atomicity, Consistency, Isolation, Durability) properties are essential for maintaining data integrity and ensuring reliable transactions. However, NoSQL databases often prioritize scalability and flexibility over strict adherence to ACID principles. Instead, they adhere to the **BASE** model.

* ***Basically Available:*** NoSQL databases prioritize availability over consistency. This means that even in the face of network partitions or failures, the system remains operational and responsive. Users can access the data and perform operations even if the system is experiencing some issues.
* ***Soft state:*** NoSQL databases allow for flexibility in data consistency. Unlike ACID-compliant systems where data must always be in a fully consistent state, NoSQL databases may permit temporary inconsistency among distributed replicas. This flexibility allows for improved performance and availability.
* ***Eventually Consistent:*** NoSQL databases prioritize eventual consistency rather than immediate consistency. In other words, while updates to the database may not be immediately reflected across all nodes or replicas, they will eventually propagate and converge to a consistent state over time. This approach allows for high availability and partition tolerance in distributed systems.

NoSQL databases prioritize scalability and speed over strict consistency, which means they can handle big data loads efficiently. However, this flexibility can lead to temporary data inconsistencies. So, developers must design their systems to handle these issues.

**NoSQL databases are generally classified into four main categories (do diagrams from sir’s notes):**

**1. Document Database**

* *Data Storage:*

Data is stored in flexible documents.

Documents are like nested boxes with keys and values.

Values (DT) can be simple (atomic) or complex (lists, arrays, nested objects, or collections).

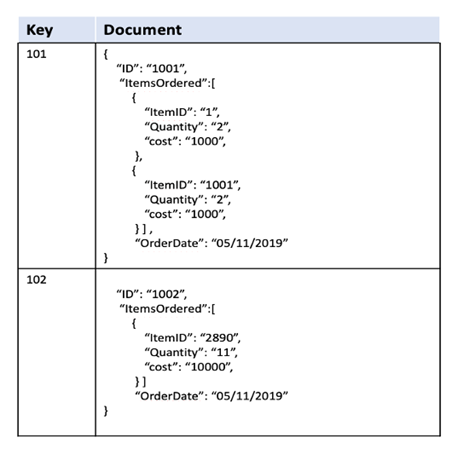
* *Retrieval and Querying:*

Each document has a unique key for access.

Retrieve specific parts easily, like finding the cost of an item.

Supports secondary indexes for faster searches.

* Data Modeling:  
  Your code models can directly become documents. Formats include JSON, BSON, or XML. Amazon DocumentDB works like MongoDB.

*Use Cases:*

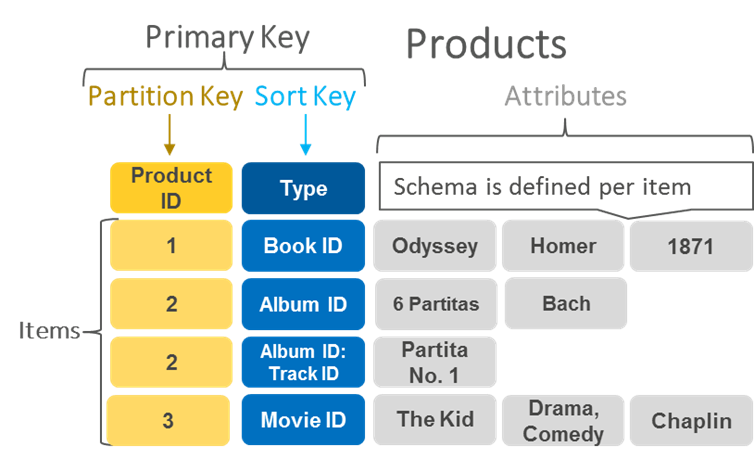
NoSQL databases, like document databases, are used in places where lots of data needs quick handling. They're popular in social media analysis, online shopping sites, games, managing content and documents, and keeping track of customer relationships.

*Considerations:*

NoSQL databases are flexible and can handle a lot of data, but they might not always ensure data consistency or transactional guarantees like traditional databases do. When choosing a database, it's crucial to assess what your application needs carefully.

*Comparison with Relational Databases:*

In relational databases, information is often organized and split into different tables. For example, user details might be in one table, and hobbies in another. To get all the details about a user and their hobbies, you need to join data from these tables. However, this joining process can slow down performance.

**2. Key-value:**

Stores data as key-value pairs. Key: This is a unique identifier within the pair. Value: This is the data associated with the key.   
{

"user\_id": "12345",

"name": "John Doe",

"age": 30,

"email": "john@example.com"

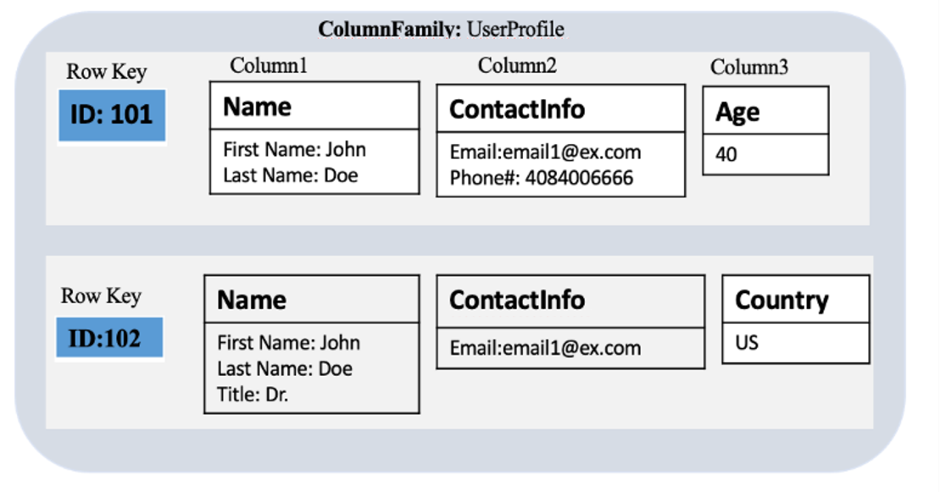
}

Each data item is identified by a unique key.

The associated value can be a string, number, object, or another data structure.

Ideal for scenarios where simple and fast retrieval of data is crucial.

Example service: Amazon DynamoDB, provided by AWS, is a managed key-value database service.



**3. Wide-column**:

*Column-Oriented Storage:* Instead of rows, data is organized into columns. Columns are grouped into families with similar attributes. Each row has a unique key for easy retrieval.

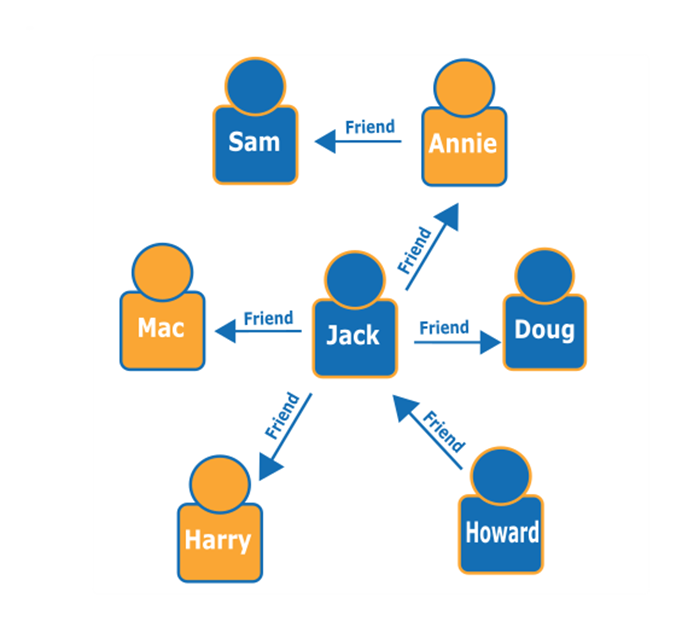
*Flexible Schema and Data Types:* Supports varying numbers of columns and multiple data types. No rigid schema requirements, making it adaptable to changing data needs.

*Efficient Storage and Retrieval:* Large data storage within a single column reduces disk usage. Retrieval operations are faster due to the column-oriented structure.

*Example Data:* Ideal for diverse datasets like time-series data or user profiles. For instance, user profiles can have columns for ID, name, email, etc.

*AWS Offering:* Amazon Keyspaces (based on Apache Cassandra) offers managed wide-column storage. It's scalable and ideal for high-throughput applications.

In short, wide-column stores organize data into columns instead of rows, offering flexibility, efficient storage, and fast retrieval for various types of data, supported by services like Amazon Keyspaces.

**4. Graph:-**

Graph databases are designed for data with complex relationships. Data is modeled as entities (nodes) and the relationships between them (edges).

Users can explore the data by moving from one node to another along defined relationships.

For example, in a social network, you can find friends of friends by traversing the graph.

Graph databases provide query languages tailored for navigating and analyzing graph structures.

These languages help in finding patterns or specific relationships within the data.

Example: Social Network Graph:

Nodes represent people, and edges represent relationships like friendship.

By traversing the graph, you can find connections like friends of friends.

Amazon Neptune is a managed graph database service provided by AWS.

It helps in storing and querying highly interconnected data efficiently.

In essence, graph databases are perfect for exploring complex relationships in data, such as social networks or network infrastructures. They offer specialized tools and services like Amazon Neptune to manage and query graph data effectively.