

# 1

## Introduction to MongoDB

# Objectives

After completing this lesson, you should be able to do the following:

- Introduction to NoSQL Databases
- Architecture of NoSQL DB's
- Types of NoSQL DB's
- CAP and BASE





# Introduction

# What is MongoDB ?

- **MongoDB** is a document-oriented NoSQL database used for high-volume data storage.
- It contains the data model, which allows you to represent hierarchical relationships.
- It uses JSON-like documents with optional schema instead of using tables and rows in traditional relational databases.
- Documents containing key-value pairs are the basic units of data in MongoDB.

- Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents.
- Collections contain sets of documents and function which is the equivalent of relational database tables.
- MongoDB is a database which came into light around the mid-2000s.
- Provides high performance, high availability, and automatic scaling.
- In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

## History of MongoDB

- The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to window azure.
- MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. It was initially developed as a PAAS (Platform as a Service). Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.
- The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010.

# Why MongoDB when we so many DB's ?

## Purpose of building MongoDB

- It may be a very genuine question that - "what was the need of MongoDB although there were many databases in action?"

## There is a simple answer:

- All the modern applications require big data, fast features development, flexible deployment, and the older database systems not competent enough, so the MongoDB was needed.

# The primary purpose of building MongoDB is

1. Scalability
2. Performance
3. High Availability
4. Scaling from single server deployments to large, complex multi-site architectures.
5. Key points of MongoDB
6. Develop Faster
7. Deploy Easier
8. Scale Bigger

# MongoDB Features

1. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
3. The rows (or documents as called in MongoDB) doesn't need to have a schema defined beforehand [**Schema less** ]. Instead, the fields can be created on the fly.
4. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
5. Scalability – The MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database

# Why Use MongoDB?

1. Document-oriented – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situation and requirements.
2. Ad hoc queries – MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.
3. Indexing – Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.
4. Replication – MongoDB can provide high availability with replica sets. A replica set consists of two or more mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.

Cont ...

5. Load balancing – MongoDB uses the concept of sharding to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.
11. Supports map reduce and aggregation tools.
6. Uses JavaScript instead of Procedures.
7. It is a schema-less database written in C++.

Cont ...

8. Provides high performance.
9. Stores files of any size easily without complicating your stack.
10. Easy to administer in the case of failures.
11. It also supports:
  - JSON data model with dynamic schemas
  - Built in replication for high availability

## MongoDB Example

- The example shows how a document can be modeled in MongoDB.
- The `_id` field is added by MongoDB to uniquely identify the document in the collection.
- What you can note is that the Order Data (OrderID, Product, and Quantity ) which in RDBMS will normally be stored in a separate table, while in MongoDB it is actually stored as an embedded document in the collection itself. This is one of the key differences in how data is modeled in MongoDB.

```
{  
  _id : <ObjectId> ,  
  
  CustomerName : Rahul ,  
  
  Order : {  
    OrderID: 111  
    Product: ProductA  
    Quantity: 5  
  }  
}
```

Example of  
how data can  
be embedded  
in a document



# Architecture

# Key Components of MongoDB Architecture

1. **\_id** – This is a field required in every MongoDB document. The **\_id** field represents a unique value in the MongoDB document. The **\_id** field is like the document's primary key. If you create a new document without an **\_id** field, MongoDB will automatically create the field. So for example, if we see the example of the above customer table, Mongo DB will add a 24 digit unique identifier to each document in the collection.

<b>_Id</b>	<b>CustomerID</b>	<b>CustomerName</b>	<b>OrderID</b>
563479cc8a8a4246bd27d784	11	Rahul	111
563479cc7a8a4246bd47d784	22	Smith	222
563479cc9a8a4246bd57d784	33	Imran	333

2. **Collection** – This is a grouping of MongoDB documents. A collection is the equivalent of a table which is created in any other RDMS such as Oracle or MS SQL. A collection exists within a single database. As seen from the introduction collections don't enforce any sort of structure.
3. **Cursor** – This is a pointer to the result set of a query. Clients can iterate through a cursor to retrieve results.
4. **Database** – This is a container for collections like in RDMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.
5. **Document** – A record in a MongoDB collection is basically called a document. The document, in turn, will consist of field name and values.
6. **JSON** – This is known as JavaScript Object Notation. This is a human-readable, plain text format for expressing structured data. JSON is currently supported in many programming languages.
7. **Field** – A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases.

## Fields with Key-Value Pair

{

CustomerID : 11

CustomerName : Rahul ,

OrderID : 1234

Example of  
Key Value  
pairs

}



# Data Modeling

- The data in MongoDB has a flexible schema.
- Unlike in SQL databases, where you must have a table's schema declared before inserting data, MongoDB's collections do not enforce document structure.
- This sort of flexibility is what makes MongoDB so powerful.

When modeling data in Mongo, keep the following things in mind

1. What are the needs of the application – Look at the business needs of the application and see what data and the type of data needed for the application. Based on this, ensure that the structure of the document is decided accordingly.
2. What are data retrieval patterns – If you foresee a heavy query usage then consider the use of indexes in your data model to improve the efficiency of queries.
3. Are frequent inserts, updates and removals happening in the database? Reconsider the use of indexes or incorporate sharding if required in your data modeling design to improve the efficiency of your overall MongoDB environment.

# Difference between MongoDB & RDBMS

RDBMS	MongoDB	Difference
Table	Collection	In RDBMS, the table contains the columns and rows which are used to store the data whereas, in MongoDB, this same structure is known as a collection. The collection contains documents which in turn contains Fields, which in turn are key-value pairs.
Row	Document	In RDBMS, the row represents a single, implicitly structured data item in a table. In MongoDB, the data is stored in documents.
Column	Field	In RDBMS, the column denotes a set of data values. These in MongoDB are known as Fields.
Joins	Embedded documents	In RDBMS, data is sometimes spread across various tables and in order to show a complete view of all data, a join is sometimes formed across tables to get the data. In MongoDB, the data is normally stored in a single collection, but separated by using Embedded documents. So there is no concept of joins in MongoDB.

## Apart from the terms differences, a few other differences are

1. Relational databases are known for enforcing data integrity. This is not an explicit requirement in MongoDB.
2. RDBMS requires that data be normalized first so that it can prevent orphan records and duplicates. Normalizing data then has the requirement of more tables, which will then result in more table joins, thus requiring more keys and indexes. As databases start to grow, performance can start becoming an issue. Again this is not an explicit requirement in MongoDB. MongoDB is flexible and does not need the data to be normalized first.

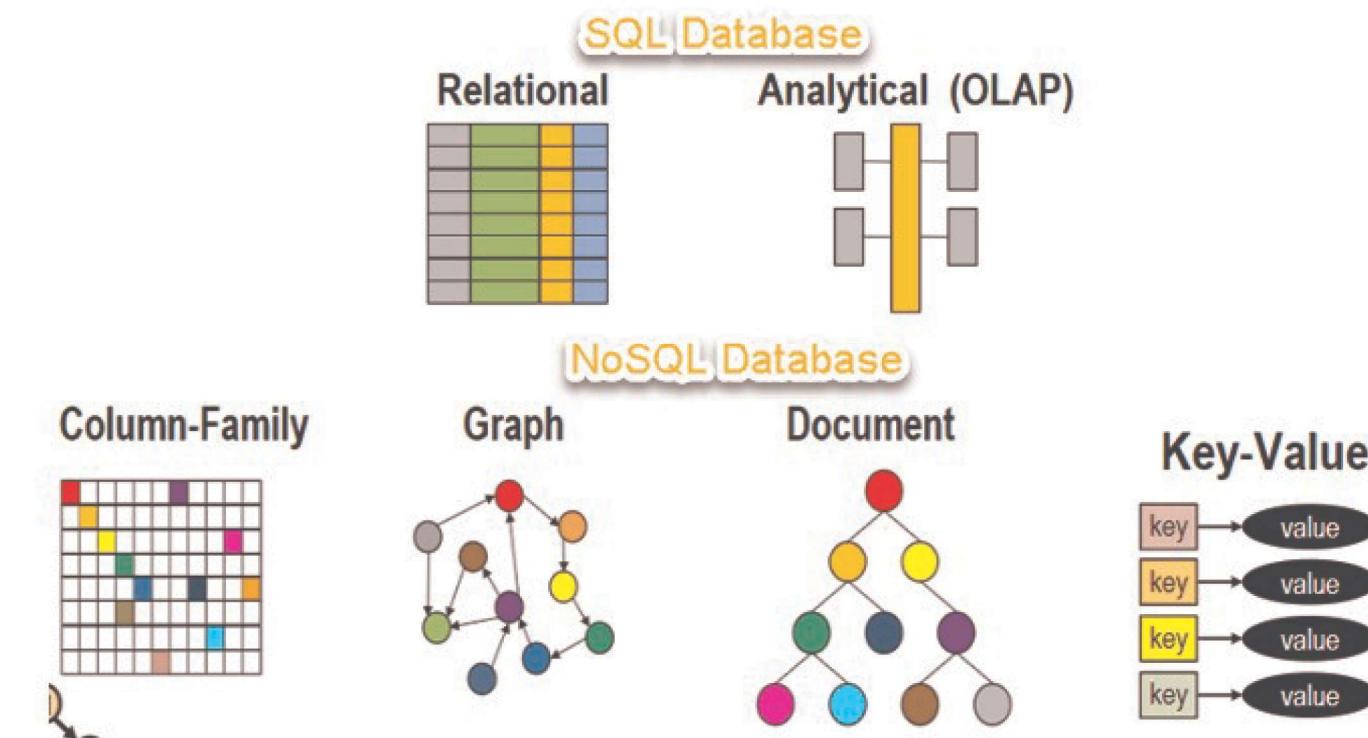


# NoSQL Databases

# What is NoSQL?

- **NoSQL** Database is a non-relational Data Management System, that does not require a fixed schema.
- It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs.
- NoSQL is used for Big data and real-time web apps.
  - For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.
- **NoSQL database** stands for “Not Only SQL” or “Not SQL.” Though a better term would be “NoREL”, NoSQL caught on. Carl Strozz introduced the NoSQL concept in 1998.

- Traditional RDBMS uses SQL syntax to store and retrieve data for further insights. Instead, a NoSQL database system encompasses a wide range of database technologies that can store structured, semi-structured, unstructured and polymorphic data.

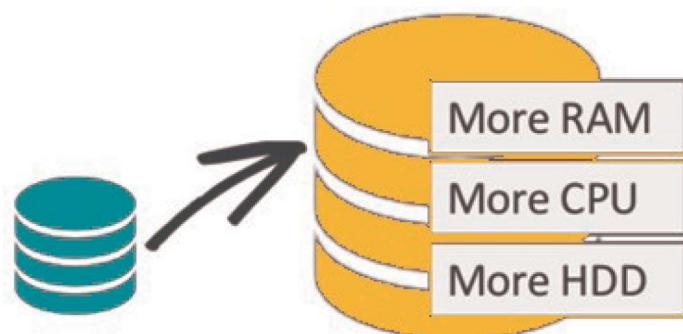


## Why NoSQL?

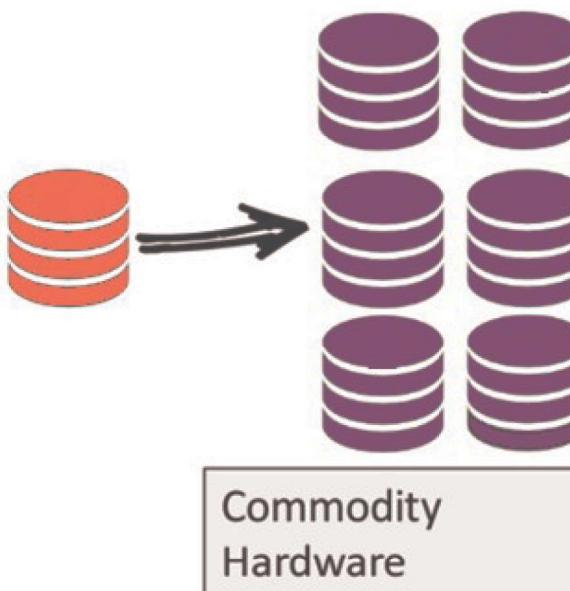
- The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with huge volumes of data. The system response time becomes slow when you use RDBMS for massive volumes of data.
- To resolve this problem, we could “scale up” our systems by upgrading our existing hardware. This process is expensive.

- The alternative for this issue is to distribute database load on multiple hosts whenever the load increases. This method is known as “scaling out.”

**Scale-Up (vertical scaling):**



**Scale-Out (horizontal scaling):**



- NoSQL database is non-relational, so it scales out better than relational databases as they are designed with web applications in mind.

## Brief History of NoSQL Databases

- 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
- 2000- Graph database Neo4j is launched
- 2004- Google BigTable is launched
- 2005- CouchDB is launched
- 2007- The research paper on Amazon Dynamo is released
- 2008- Facebooks open sources the Cassandra project
- 2009- The term NoSQL was reintroduced



# Features of NoSQL

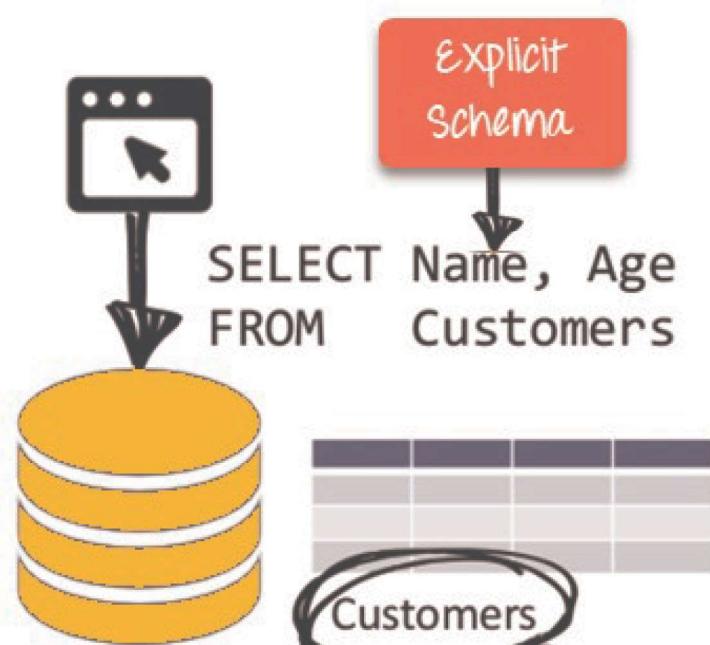
## Non-relational

1. NoSQL databases never follow the [relational model](#)
2. Never provide tables with flat fixed-column records
3. Work with self-contained aggregates or BLOBs
4. Doesn't require object-relational mapping and data normalization
5. No complex features like query languages, query planners, referential integrity joins, ACID

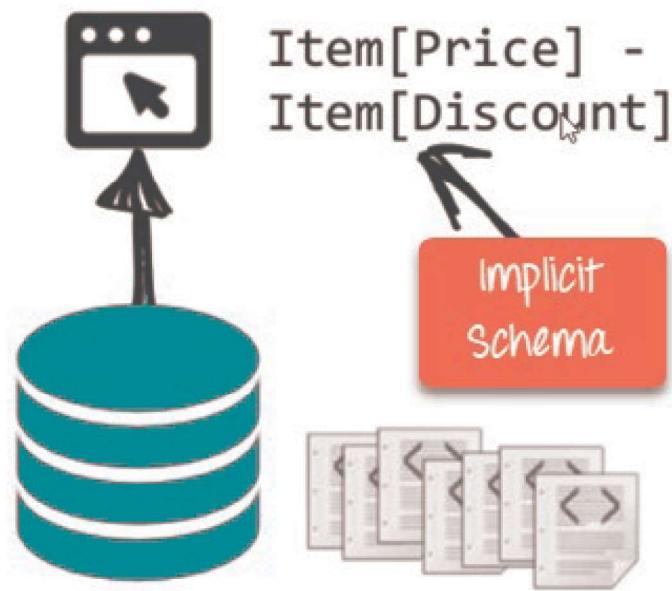
# Schema-free / Schema Less [ NoSQL is Schema Free ]

1. NoSQL databases are either schema-free or have relaxed schemas
2. Do not require any sort of definition of the schema of the data
3. Offers heterogeneous structures of data in the same domain

RDBMS:

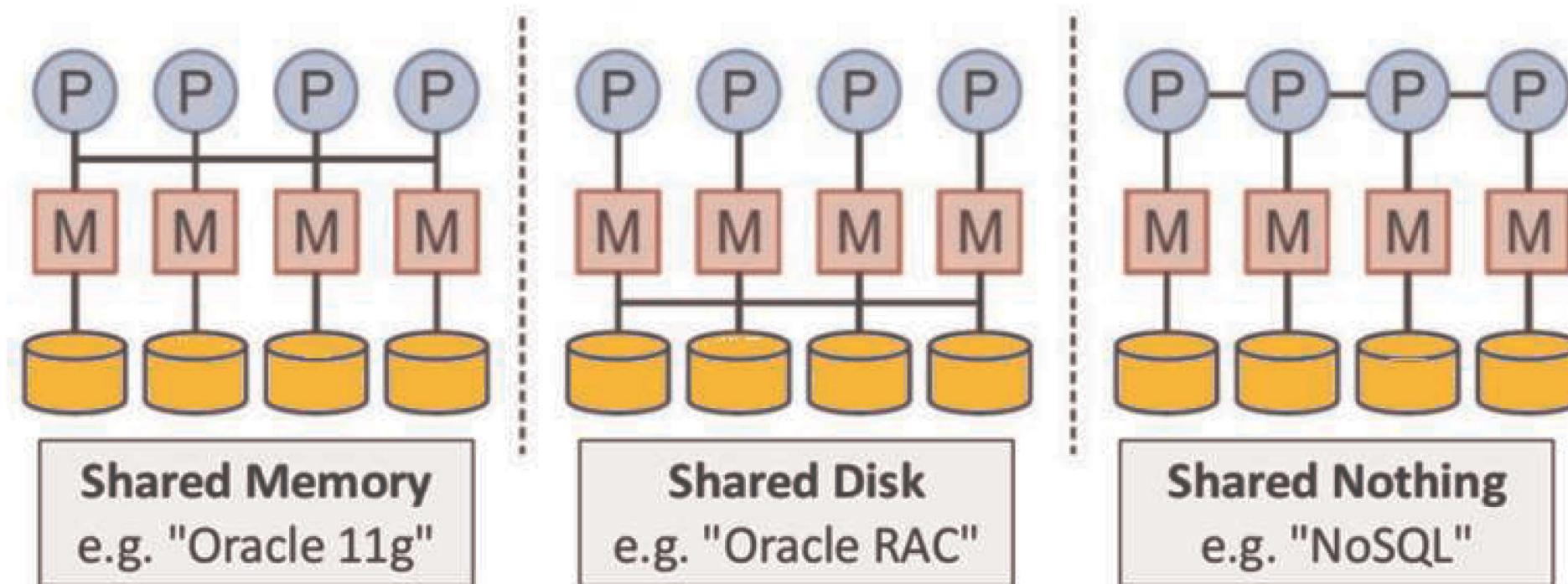


NoSQL DB:



1. Offers easy to use interfaces for storage and querying data provided
2. APIs allow low-level data manipulation & selection methods
3. Text-based protocols mostly used with HTTP REST with JSON
4. Mostly used no standard based NoSQL query language
5. Web-enabled databases running as internet-facing services

1. Multiple NoSQL databases can be executed in a distributed fashion
2. Offers auto-scaling and fail-over capabilities
3. Often ACID concept can be sacrificed for scalability and throughput
4. Mostly no synchronous replication between distributed nodes Asynchronous Multi-Master Replication, peer-to-peer, HDFS Replication
5. Only providing eventual consistency
6. Shared Nothing Architecture. This enables less coordination and higher distribution.

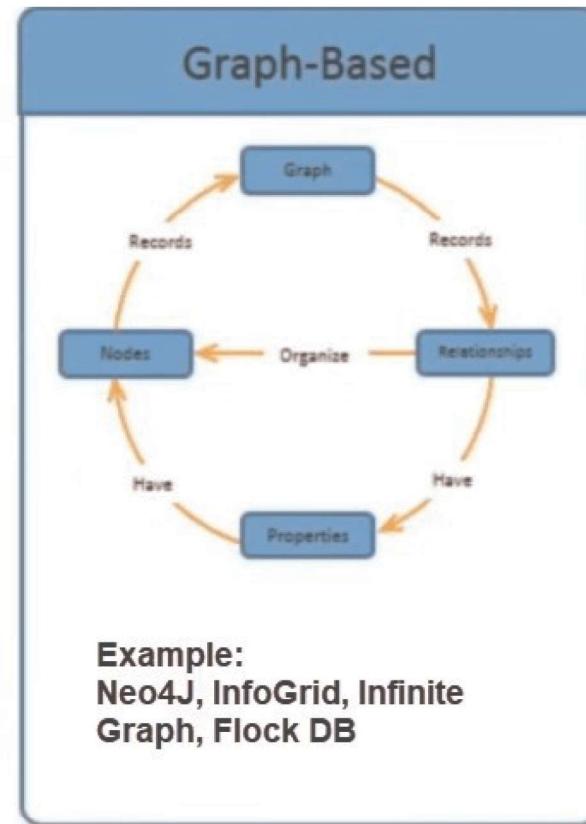
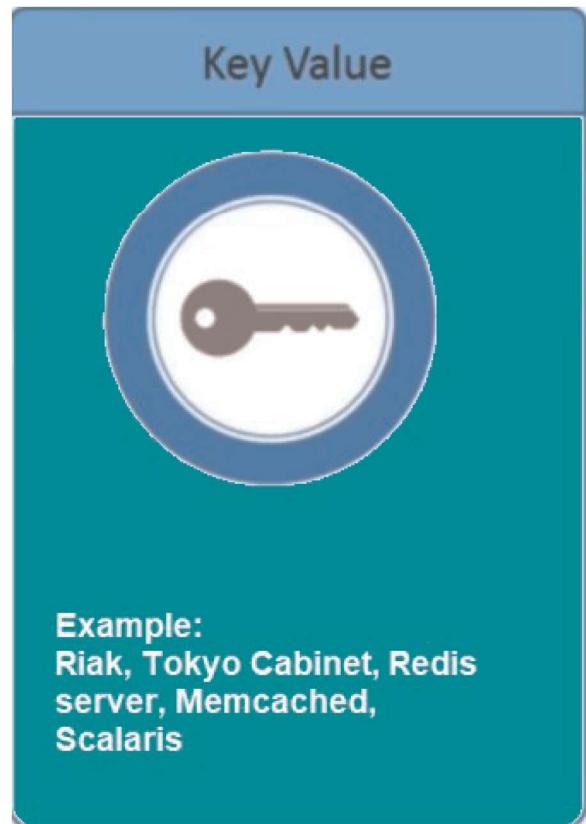




# Types of NoSQL DB's

- **NoSQL Databases** are mainly categorized into four types: Key-value pair, Column-oriented, Graph-based and Document-oriented. Every category has its unique attributes and limitations. None of the above-specified database is better to solve all the problems. Users should select the database based on their product needs.
- Types of NoSQL Databases:
  1. Key-value Pair Based
  2. Column-oriented Graph
  3. Graphs based
  4. Document-oriented

# Types



## Key Value Pair Based

- Data is stored in key/value pairs. It is designed in such a way to handle lots of data and heavy load.
- Key-value pair storage databases store data as a hash table where each key is unique, and the value can be a JSON, BLOB(Binary Large Objects), string, etc.

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Cont ...

- It is one of the most basic NoSQL database example. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc. Key value stores help the developer to store schema-less data. They work best for shopping cart contents.
- Redis, Dynamo, Riak are some NoSQL examples of key-value store DataBases. They are all based on Amazon's Dynamo paper.

## Column-based

- Column-oriented databases work on columns and are based on BigTable paper by Google. Every column is treated separately. Values of single column databases are stored contiguously.

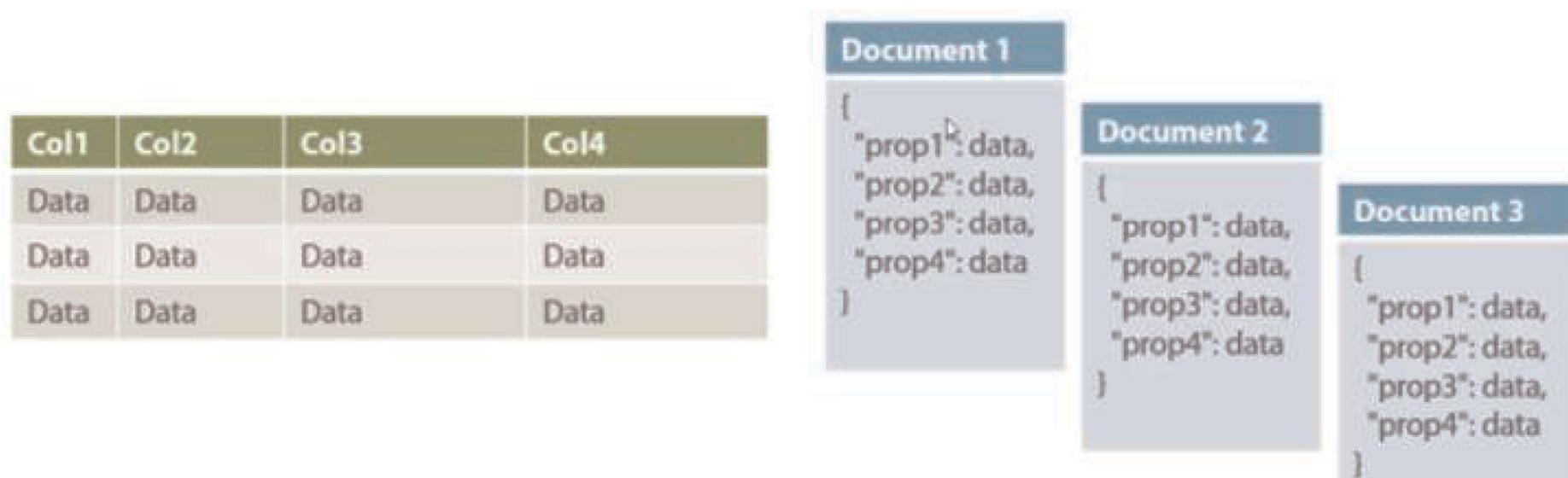
ColumnFamily			
Row Key	Column Name		
	Key	Key	Key
Value	Value	Value	Value
	Column Name		
Key	Key	Key	Key
	Value	Value	Value

Cont ...

- They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.
- Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,
- HBase, Cassandra, Hypertable are NoSQL query examples of column based database.

## Document-Oriented:

- Document-Oriented NoSQL DB stores and retrieves data as a key value pair but the value part is stored as a document. The document is stored in JSON or XML formats. The value is understood by the DB and can be queried.

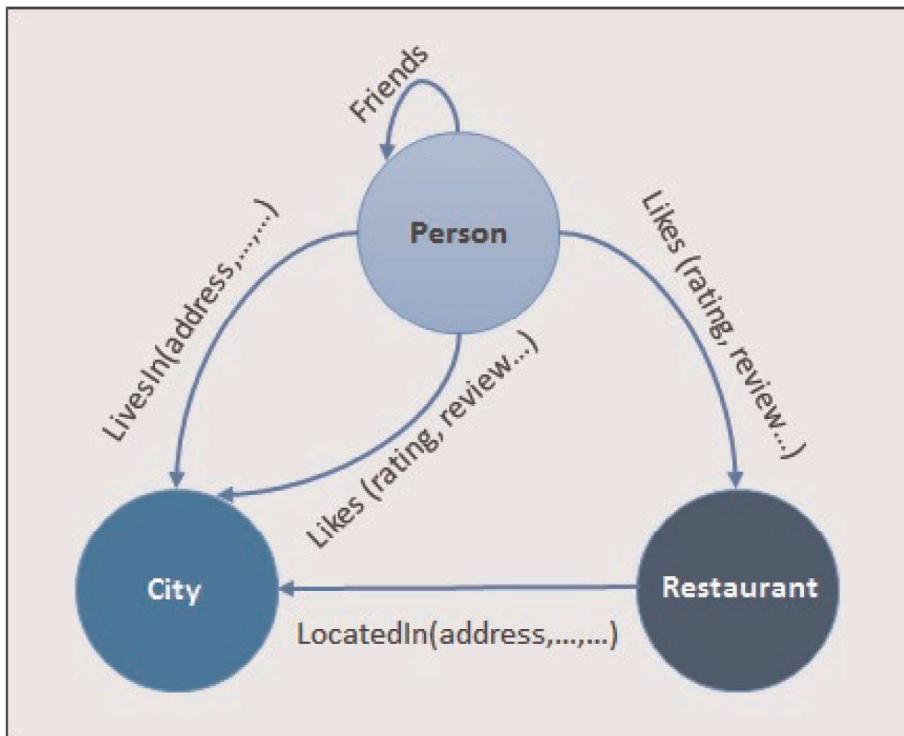


## Relational Vs. Document

- In RDBMS we have rows and columns, and in the right, we have a document database which has a similar structure to JSON. Now for the relational database, you have to know what columns you have and so on. However, for a document database, you have data store like JSON object. You do not require to define which make it flexible.
- The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications. It should not use for complex transactions which require multiple operations or queries against varying aggregate structures.
- Amazon SimpleDB, CouchDB, MongoDB, Riak, Lotus Notes, MongoDB, are popular Document originated DBMS systems.

# Graph-Based

- A graph type database stores entities as well the relations amongst those entities. The entity is stored as a node with the relationship as edges. An edge gives a relationship between nodes. Every node and edge has a unique identifier.



Cont ...

- Compared to a relational database where tables are loosely connected, a Graph database is a multi-relational in nature. Traversing relationship is fast as they are already captured into the DB, and there is no need to calculate them.
- Graph base database mostly used for social networks, logistics, spatial data.
- Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.

## Query Mechanism tools for NoSQL

- The most common data retrieval mechanism is the REST-based retrieval of a value based on its key/ID with GET resource
- Document store Database offers more difficult queries as they understand the value in a key-value pair. For example, CouchDB allows defining views with MapReduce

## What is the CAP Theorem?

- CAP theorem is also called brewer's theorem. It states that it is impossible for a distributed data store to offer more than two out of three guarantees
  1. Consistency
  2. Availability
  3. Partition Tolerance

## **Consistency:**

- The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.

## **Availability:**

- The database should always be available and responsive. It should not have any downtime.

## **Partition Tolerance:**

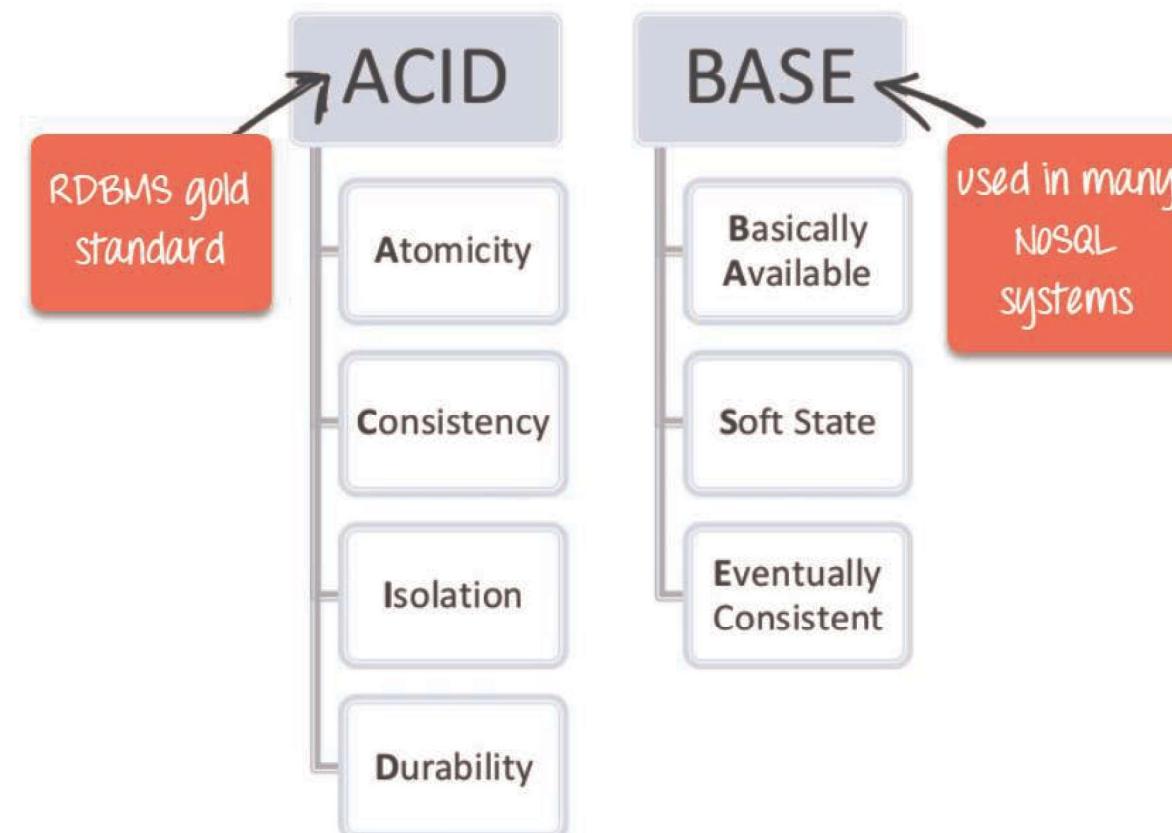
- Partition Tolerance means that the system should continue to function even if the communication among the servers is not stable. For example, the servers can be partitioned into multiple groups which may not communicate with each other. Here, if part of the database is unavailable, other parts are always unaffected.

# Eventual Consistency

- The term “eventual consistency” means to have copies of data on multiple machines to get high availability and scalability. Thus, changes made to any data item on one machine has to be propagated to other replicas.
- Data replication may not be instantaneous as some copies will be updated immediately while others in due course of time. These copies may be mutually, but in due course of time, they become consistent. Hence, the name eventual consistency.

# BASE: Basically Available, Soft state, Eventual consistency

- Basically, available means DB is available all the time as per CAP theorem
- Soft state means even without an input; the system state may change
- Eventual consistency means that the system will become consistent over time



## Advantages of NoSQL

1. Can be used as Primary or Analytic Data Source
2. Big Data Capability
3. No Single Point of Failure
4. Easy Replication
5. No Need for Separate Caching Layer
6. It provides fast performance and horizontal scalability.
7. Can handle structured, semi-structured, and unstructured data with equal effect
8. Object-oriented programming which is easy to use and flexible
9. NoSQL databases don't need a dedicated high-performance server
10. Support Key Developer Languages and Platforms

Cont ...

11. Simple to implement than using RDBMS
12. It can serve as the primary data source for online applications.
13. Handles big data which manages data velocity, variety, volume, and complexity
14. Excels at distributed database and multi-data center operations
15. Eliminates the need for a specific caching layer to store data
16. Offers a flexible schema design which can easily be altered without downtime or service disruption

## Disadvantages of NoSQL

1. No standardization rules
2. Limited query capabilities
3. RDBMS databases and tools are comparatively mature
4. It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
5. When the volume of data increases it is difficult to maintain unique values as keys become difficult
6. Doesn't work as well with relational data
7. The learning curve is stiff for new developers
8. Open source options so not so popular for enterprises.

# Summary

In this lesson, you should have learned how to:

- Introduction to NoSQL Databases
- Architecture of NoSQL DB's
- Types of NoSQL DB's
- CAP and BASE

