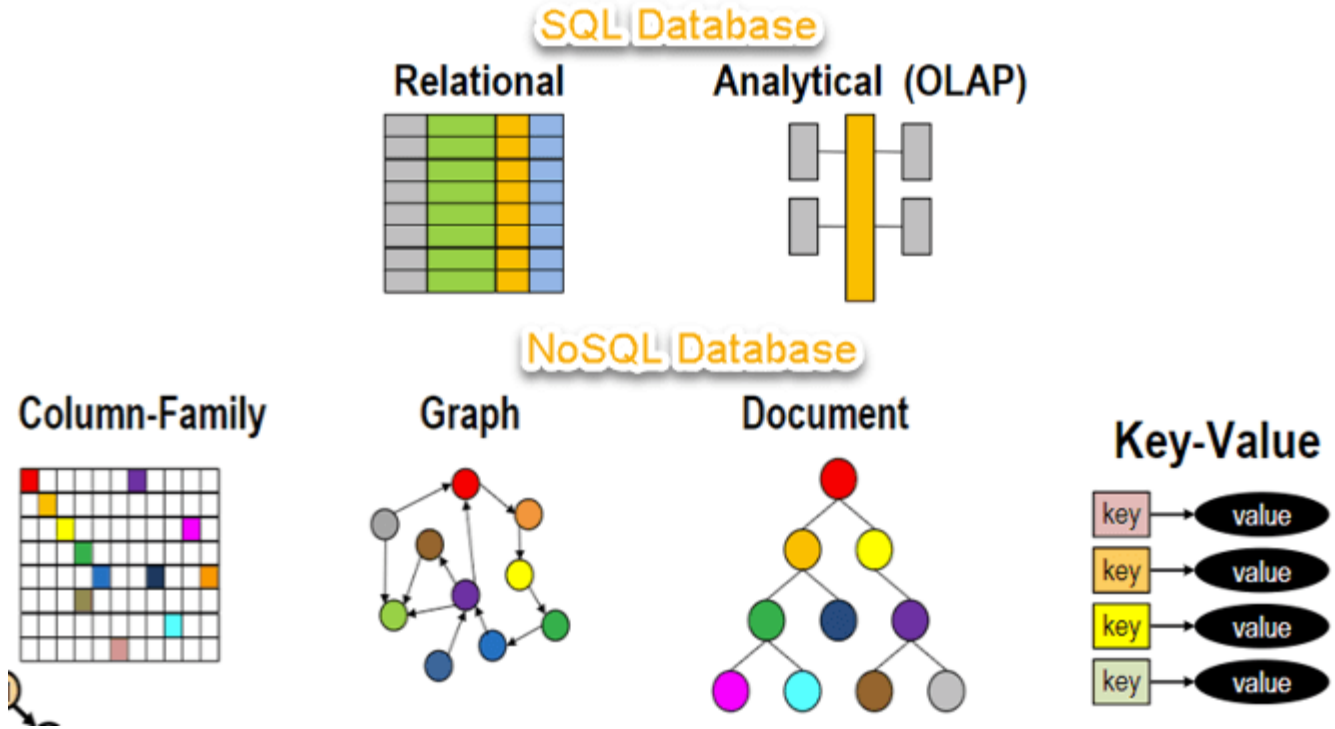
Chapter 5
NoSQL

Dr. Nilesh Madhukar Patil
Associate Professor
Computer Engineering
SVKM’s DJSCE, Mumbai

Introduction to 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.”
- 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.


SQL Database
Relational
Analytical (OLAP)
NoSQL Database
Column-Family
Graph
Document
Key-Value
key
value
key
value
key
value
key
value

Why Should You Use NoSQL Database?

- NoSQL databases frequently enable developers to customize the data format. They are well suited to contemporary Agile production techniques centered on sprints, fast modifications, and frequent software releases.
- It can be time-consuming for a programmer to request a SQL database operator to modify the database structure and afterward discharge and then reload the data. NoSQL databases are frequently more suitable for collecting and analyzing organized, semi-structured, and unstructured data in a centralized database.
- NoSQL databases frequently store information in a format comparable to the entities used during applications.
- Minimizing the requirement for translation between the format in which the information is stored to the format in which the data appears in code.
- As part of its core architecture, NoSQL databases originally were designed to manage large amounts of information.
- Whenever SQL databases are utilized to run web-scale applications, there is no additional engineering required. The road to data scalability is well-defined and straightforward.
- NoSQL databases are frequently built on a scale-out method, which enables scaling to enormous data volumes far less expensive than the scale-up method used by SQL databases.
- Several NoSQL databases utilize a scale-out method, which gives a straightforward way to increase the volume of traffic a library can manage.
- Scale-out structures also provide advantages such as updating or modifying the database structure with no interruption.
- Scale-out architecture is among the most cost-effective methods of handling high traffic levels.

Features of NoSQL (1/5)

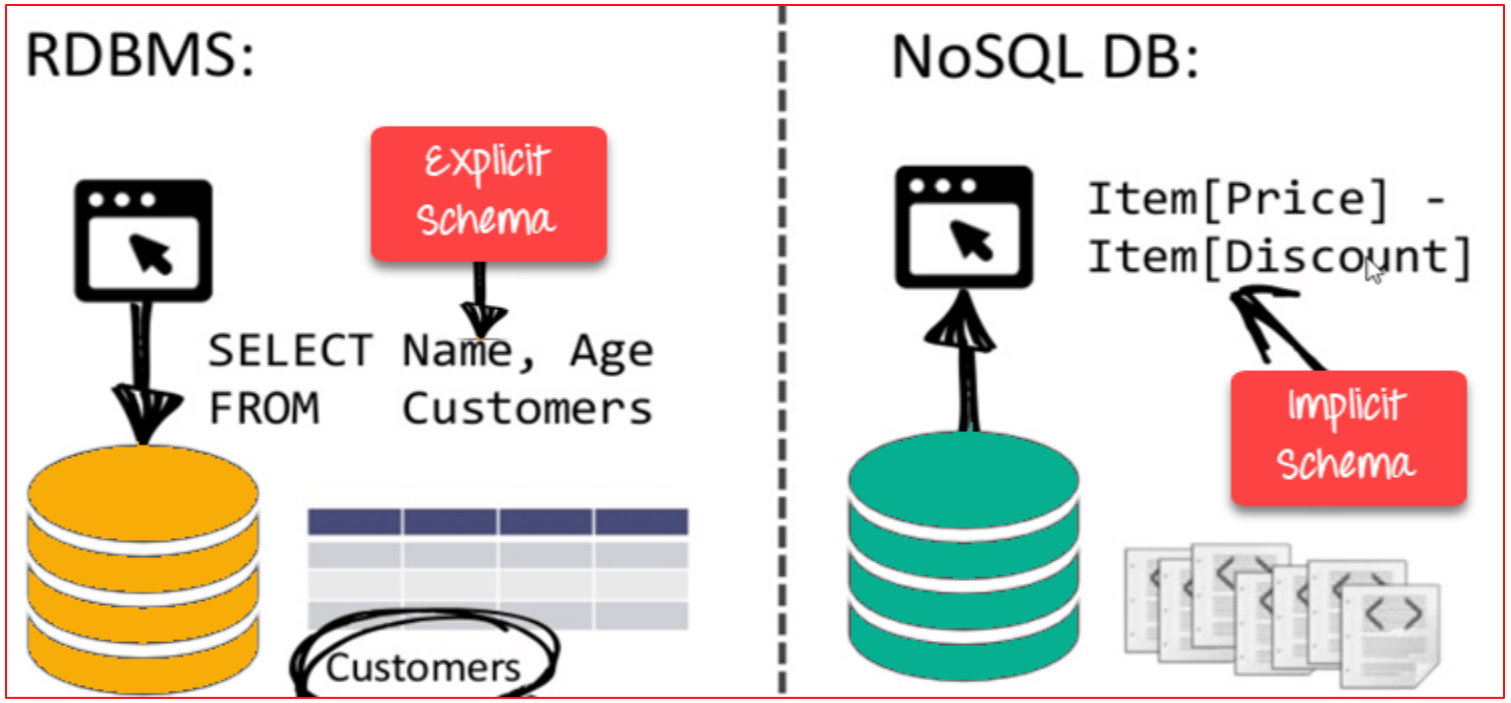
- **Non-relational**
- **Schema-free**
- **Simple API**
- **Distributed**

Non-relational (2/5)

- NoSQL databases never follow the relational model
- Never provide tables with flat fixed-column records
- Work with self-contained aggregates or BLOBs
- Doesn’t require object-relational mapping and data normalization
- No complex features like query languages, query planners, referential integrity joins, ACID properties

Schema-free (3/5)

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

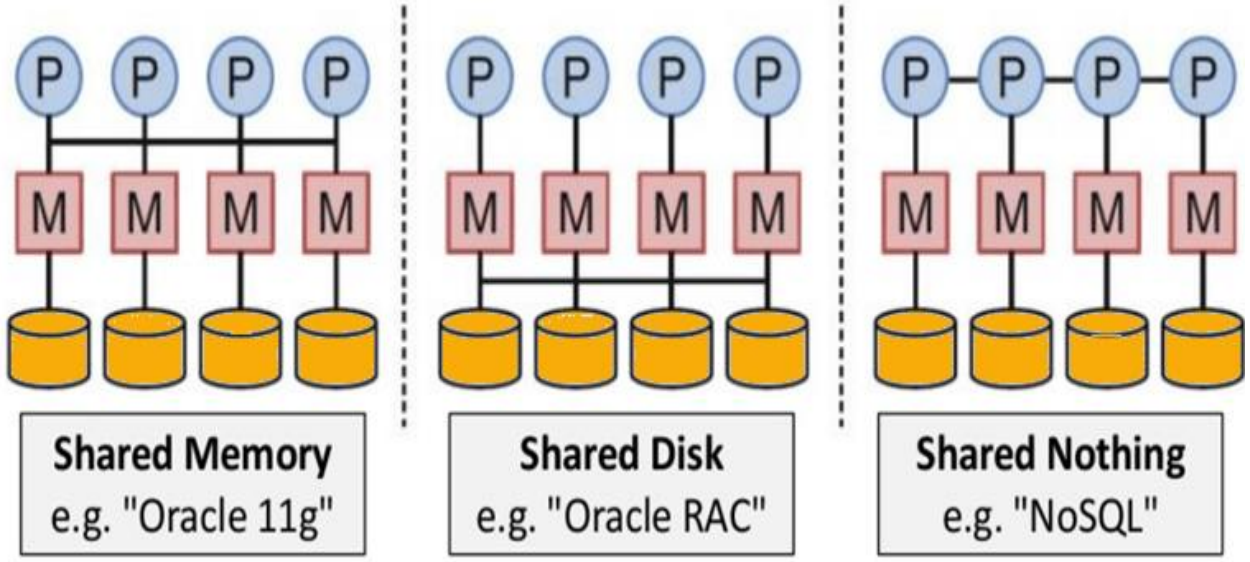


Simple API (4/5)

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

Distributed (5/5)

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

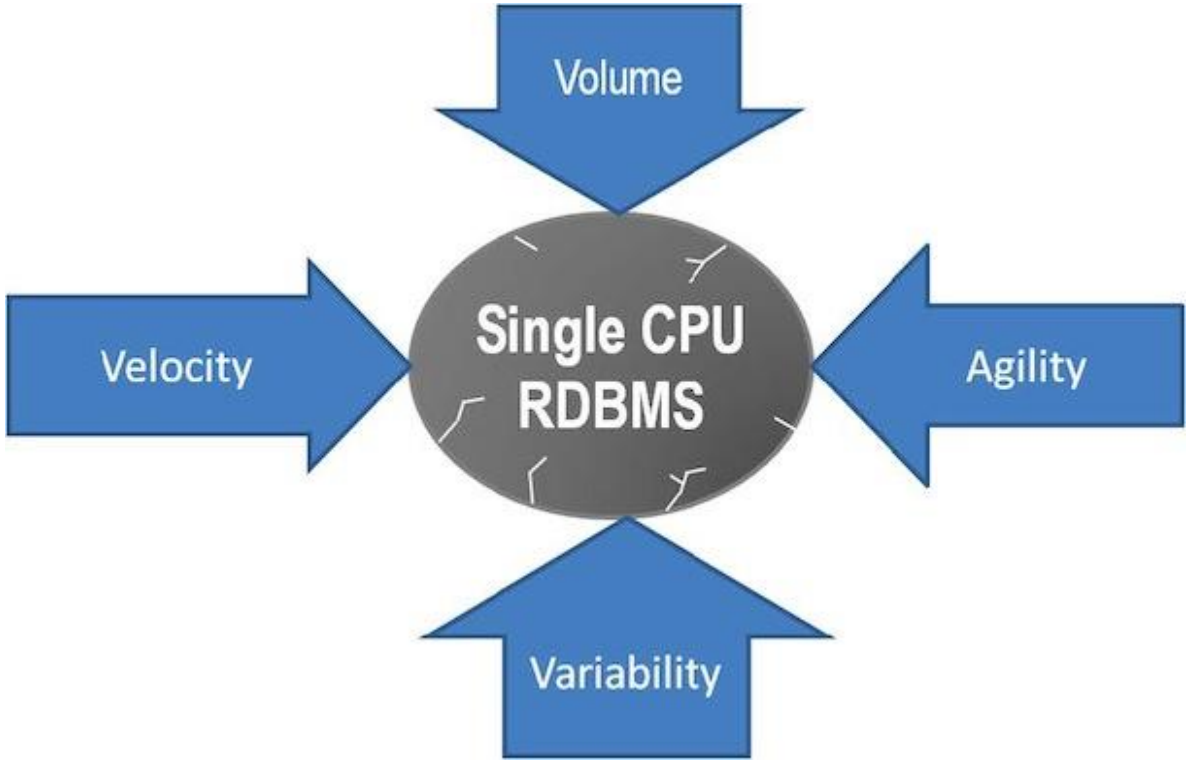


SQL vs NoSQL

SQL	**NoSQL**
RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)	Non-relational or distributed database system.
These databases have fixed or static or predefined schema	They have dynamic schema
These databases are not suited for hierarchical data storage.	These databases are best suited for hierarchical data storage.
These databases are best suited for complex queries	These databases are not so good for complex queries
Vertically Scalable	Horizontally scalable
Follows ACID property	Follows CAP(consistency, availability, partition tolerance)
Examples: MySQL, PostgreSQL, Oracle, MS-SQL Server etc	Examples: MongoDB, GraphQL, HBase, Neo4j, Cassandra etc.

NoSQL Business Drivers (1/5)

- Volume and velocity refer to the ability to handle large datasets that arrive quickly.
- Variability refers to how diverse data types don't fit into structured tables and
- agility refers to how quickly an organization responds to business change.



Volume (2/5)

- There are two ways to look into data processing to improve performance.
- If the key factor is only speed, a faster processor could be used. If the processing involves complex (heavy) computation, the Graphic Processing Unit (GPU) could be used along with the CPU. But the volume of data is limited to onboard GPU memory.
- The main reason for organizations to look at an alternative to their current RDBMSs is the need to query big data.
- The need for horizontal scaling made organizations move from serial to distributed parallel processing where big data is fragmented and processed using clusters of commodity machines.
- This is made possible by the development of technologies like Apache Hadoop, HDFS, MapR, HBase, etc.

Velocity (3/5)

- Velocity becomes the key factor when the frequency in which online queries to the database made by social networking and e-commerce websites have to be read and written in real-time.
- Many single CPU and RDBMS systems are unable to cope up with the demands of real-time inserts.
- RDBMS systems frequently index on many columns that decrease the system performance.
- For example, when online shopping sites introduce great discount schemes, the random bursts in web traffic will slow down the response for every user, and tuning these systems as demand increases can be costly when both high read and write are required.

Variability (4/5)

- Organizations that need to capture and report on certain uncommon data, struggle when attempting to use RDBMS fixed schema.
- For example, if a business process wants to store a few special attributes for a few sets of customers, then it needs to alter its schema definition.
- If a change is made to the schema, all customer rows within the database will also have this column. If there is no value related to this for most of the customers, then the row column representation will have a sparse matrix.
- In addition to this, new columns to an RDBMS require to execute ALTER TABLE command.
- This cannot be done on the fly since the present executing transaction has to be complete and the database has to be closed, and then the schema can be altered.
- This process affects system availability, which means losing business.

Agility (5/5)

- The process of storing and retrieving data for complex queries in RDBMS is quite cumbersome.
- If it is a nested query, data will have nested and repeated subgroups of data structures that are included in an object-relational mapping layer.
- This layer is responsible to generate the exact combination of SELECT, INSERT, DELETE and UPDATE SQL statements to move the object data from and to the backend RDBMS layer.
- This process is not simple and requires experienced developers with the knowledge of object-relational frameworks such as Java Hibernate.
- Even then, these change requests can cause slowdowns in implementation and testing.

CAP Theorem

- CAP theorem is also called Brewer's theorem.
- It states that is impossible for a distributed data store to offer more than two out of three guarantees

1.Consistency

2.Availability

3.Partition Tolerance

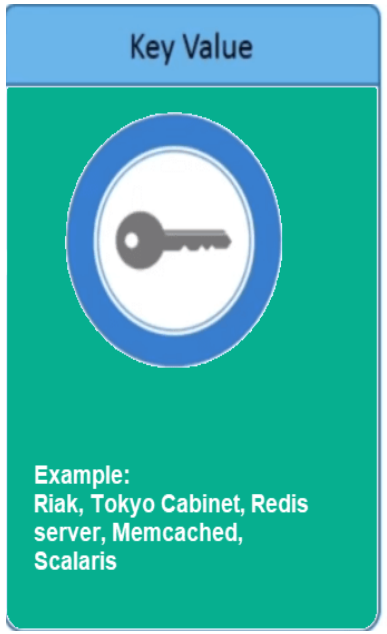
- **Consistency:** The data should remain consistent even after the execution of an operation. This means once data is written, any future read request should contain that data. For example, after updating the order status, all the clients should be able to see the same data.
- **Availability:** The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

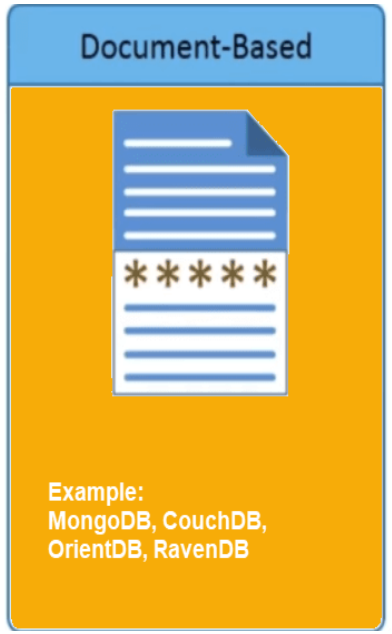
BASE Transactions

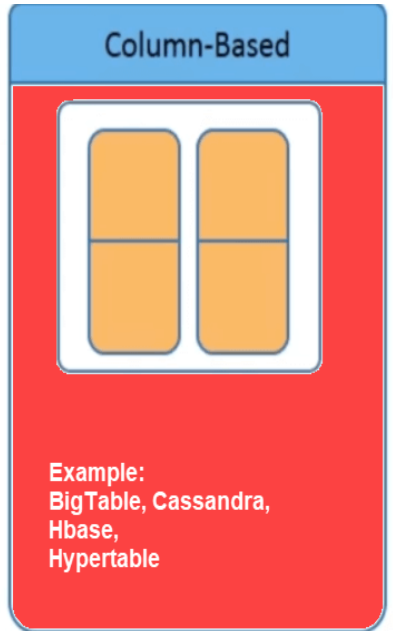
- NoSQL relies upon a softer model known as the BASE model.
- BASE: **B**asically **A**vailable, **S**oft state, **E**ventual consistency
- Basically available means DB is available all the time as per CAP theorem. There will be a response to any request (can be failure too).
- Soft state means even without an input; the system state may change.
- Eventual consistency means that the system will become consistent over time once it stops receiving input.

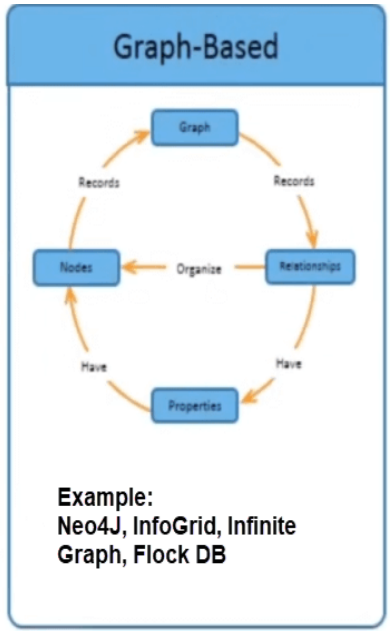
NoSQL Data Architectural Patterns (1/5)

- **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 databases is better to solve all the problems.
- Users should select the database based on their product needs.









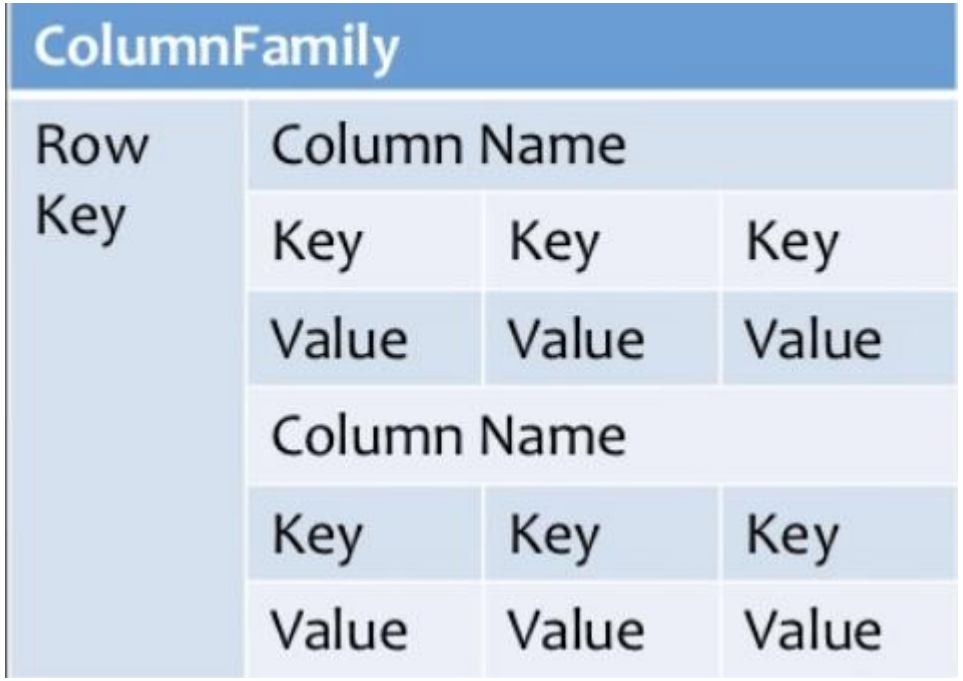
Key Value Pair Based (2/5)

- 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.
- For example, a key-value pair may contain a key like “Website” associated with a value like “Google”.
- It is one of the most basic NoSQL database examples. This kind of NoSQL database is used as a collection, dictionaries, associative arrays, etc.
- Key value stores help the developer store schema-less data. They work best for shopping cart contents.
- Redis, Dynamo, and Riak are some NoSQL examples of key-value store Databases.
- They are all based on Amazon’s Dynamo paper.

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

Column-based (3/5)

- 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.
- 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, HBase, Hypertable are NoSQL query examples of column based database.

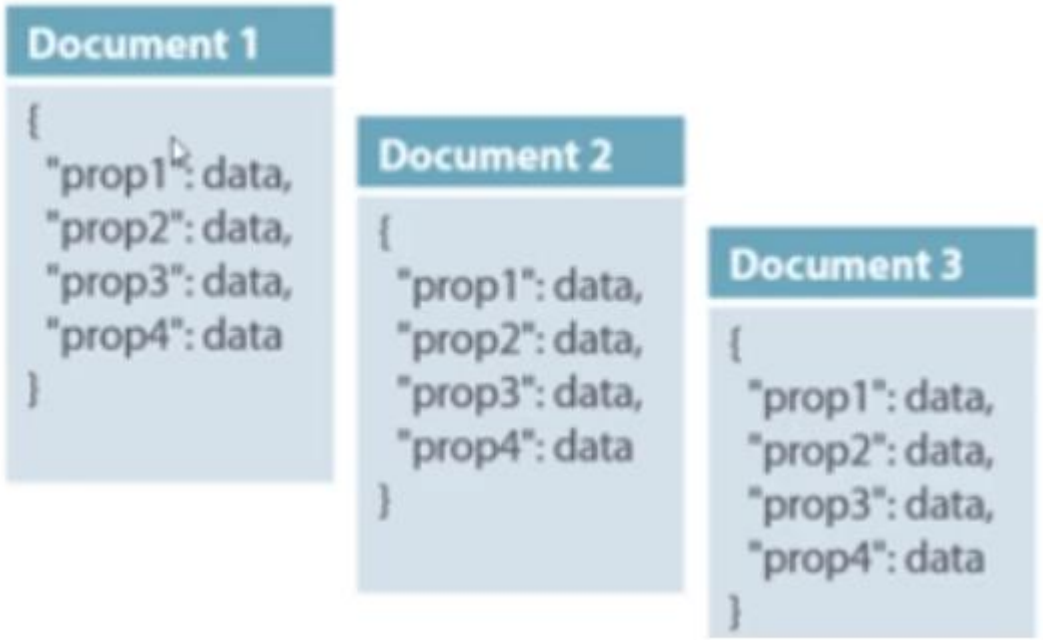


Document-Oriented (4/5)

- 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.
- The document type is mostly used for CMS systems, blogging platforms, real-time analytics & e-commerce applications.
- It should not be used 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.

Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

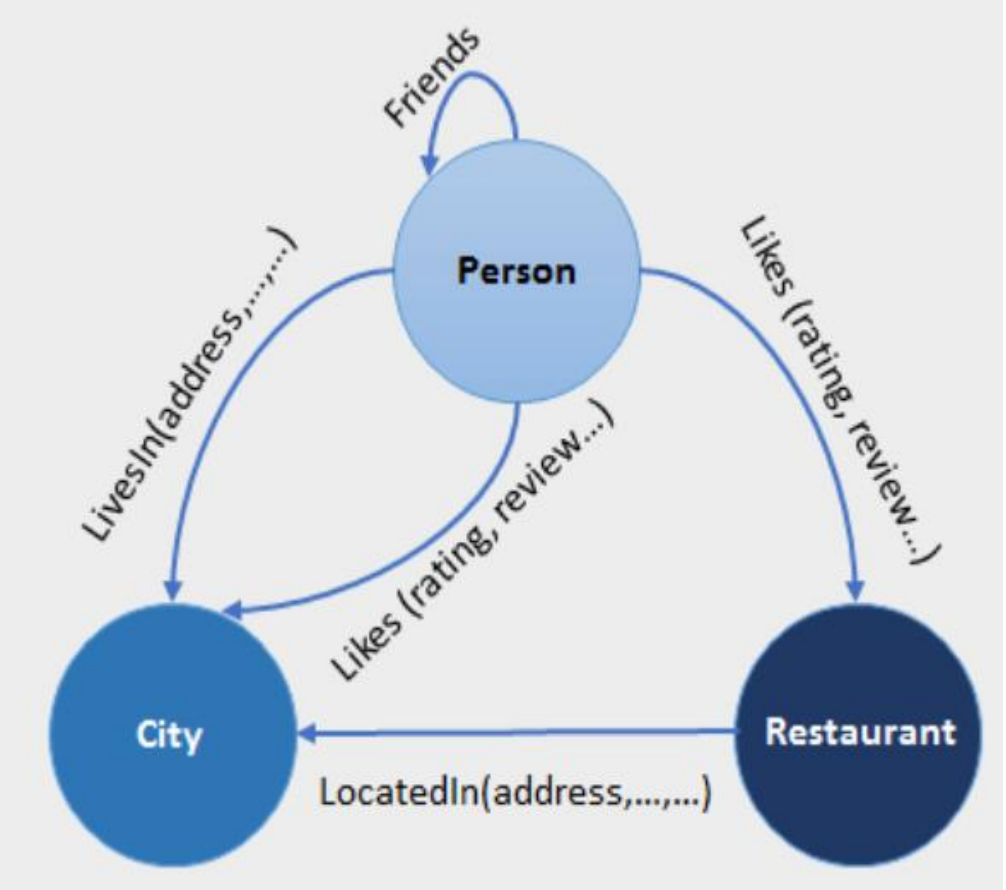
Relational Based



Document Oriented

Graph-Based (5/5)

- 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.
- Graph base databases are mostly used for social networks, logistics, and spatial data.
- Neo4J, Infinite Graph, OrientDB, FlockDB are some popular graph-based databases.



Use of NoSQL in Industry

- Session Store
- User Profile Store
- Content and Metadata Store
- Mobile Applications
- Third Party Data Aggregation
- Internet of Things
- E-commerce
- Social Gaming
- Ad Targeting

Advantages of NoSQL

- Can be used as a Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- No Need for Separate Caching Layer
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect
- Object-oriented programming which is easy to use and flexible
- NoSQL databases don’t need a dedicated high-performance server
- Support Key Developer Languages and Platforms
- Simple to implement than using RDBMS
- It can serve as the primary data source for online applications.
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Eliminates the need for a specific caching layer to store data
- Offers a flexible schema design that can easily be altered without downtime or service disruption

Disadvantages of NoSQL

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

MongoDB

- **MongoDB** is a document-oriented NoSQL database used for high volume data storage.
- Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents.
- Documents consist of key-value pairs which are the basic unit of data in MongoDB.
- Collections contain sets of documents and function which is the equivalent of relational database tables.

Features of MongoDB (1/2)

Features of MongoDB (2/2)

- **Queries:** It supports ad-hoc queries and document-based queries.
- **Index Support:** Any field in the document can be indexed.
- **Replication:** It supports Master-Slave replication. MongoDB uses the native applications to maintain multiple copies of data. Preventing database downtime is one of the replica set's features as it has a self-healing shard.
- **Multiple Servers:** The database can run over multiple servers. Data is duplicated to foolproof the system in the case of hardware failure.
- **Auto-sharding:** This process distributes data across multiple physical partitions called shards. Due to sharding, MongoDB has an automatic load balancing feature.
- **MapReduce:** It supports MapReduce and flexible aggregation tools.
- **Failure Handling:** In MongoDB, it's easy to cope with cases of failures. Huge numbers of replicas give out increased protection and data availability against database downtimes like rack failures, multiple machine failures, and data center failures, or even network partitions.
- **GridFS:** Without complicating your stack, any size of files can be stored. GridFS feature divides files into smaller parts and stores them as separate documents.
- **Schema-less Database:** It is a schema-less database written in C++.
- **Document-oriented Storage:** It uses the BSON format which is a JSON-like format.
- **Procedures:** MongoDB JavaScript works well as the database uses the language instead of procedures.

Document Database in MongoDB

- A record in MongoDB is a document, which is a data structure composed of field and value pairs.
- MongoDB documents are similar to JSON objects.
- The values of fields may include other documents, arrays, and arrays of documents.
- MongoDB stores documents in collections.
- Collections are analogous to tables in relational databases.

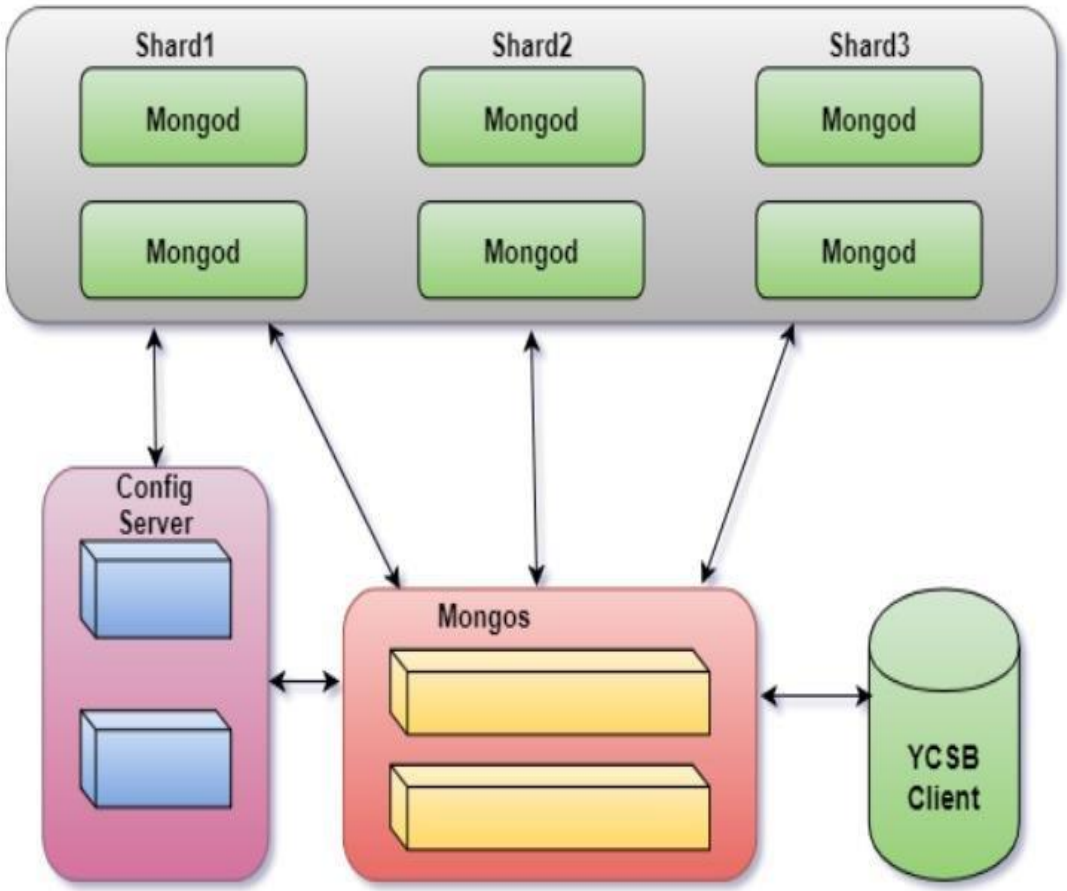
MongoDB Vs RDBMS

MongoDB	**RDBMS**
Document-oriented and non-relational database	Relational database
Document-based	Row-based
Field-based	Column based
Collection based and key-value pair	Table based
Gives JavaScript client for querying	Doesn’t give JavaScript for querying
Relatively easy to setup	Comparatively not that easy to setup
Unaffected by SQL injection	Quite vulnerable to SQL injection
Has dynamic schema and ideal for hierarchical data storage	Has predefined schema and not good for hierarchical data storage
100 times faster and horizontally scalable through sharding	By increasing RAM, vertical scaling can happen

MongoDB Vs MySQL

MongoDB	MySQL
The query language is javascript	The query language is a structured query language
It represents data as JSON documents	It represents data in tables and rows.
Defining the schema is not required	Defining tables and columns is required
It does not support JOIN	It supports JOIN
MongoDB was built with high availability and scalability in mind and offers replication and sharding out of the box.	Although the MySQL idea does not allow for effective replication and sharding, it does let users retrieve related data via joins, which reduces duplication.
Because of its design, MongoDB is less of an attack.	It has a risk of SQL injection attacks.

MongoDB Architecture



Mongod:- It is the Datanode which is used for storing and retrieving the data.

Mongos:- It is the one and only instance that is able to communicate outside of the cluster. It interfaces with clients and routers operations to appropriate shards.

Config Server:- It acts as the container of the metadata(used in case of node failure) about the object stored in the mongod. In a cluster there can be 1 or 3 config-server instances.

Each running component constitutes one node in the MongoDB cluster.

Shard is a group of one or more mongod nodes and is known as replica sets which contains a copy of the data.

The replication factor of a system is determined by the number of datanodes in a shard. It has Master-Slave architecture and within a shard there is only 1 Master which can read and write and others will be slave which can only perform read operations.

The **YC SB (Yugabyte Cloud Service Broker) client** in the MongoDB architecture provides an interface between the application code and the underlying database. The client implements the MongoDB API and translates the commands and queries issued by the application into commands that can be understood by YugabyteDB.

MongoDB Data Types

MongoDB supports a wide range of datatypes, such as:

- **String –** Must be UTF-8 valid
- **Integer –** Stores a numerical value of 32 bit or 64 bit depending upon the server
- **Boolean –** Stores true/ false value
- **Double –** Stores floating point values
- **Min/Max keys –** Compares a value against the lowest and highest BSON elements
- **Arrays –** Stores arrays, lists, or multiple values into one key
- **Date –** Stores the current date or time in UNIX format
- **Timestamp –** Useful for keeping a record of the modifications or additions to a document
- **Object –** Used for embedded documents
- **Object ID –** Stores the ID of a document
- **Binary data –** For storing binary data
- **Null –** Stores a null value
- **Symbol –** Used identically to a string but mainly for languages that have specific symbol types
- **Code –** For storing JavaScript code into the document
- **Regular expression –** Stores regular expression

MongoDB Examples

- MongoDB Commands
- MongoDB Example
- Movie Dataset Example