1. **NoSQL Databases**

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases

Relational databases rely on tables, columns, rows, or schemas to organize and retrieve data. In contrast, NoSQL databases do not rely on these structures and use more flexible data models. NoSQL can mean “not SQL” or “not only SQL.” As RDBMS have increasingly failed to meet the performance, scalability, and flexibility needs that next-generation, data-intensive applications require, NoSQL databases have been adopted by mainstream enterprises. NoSQL is particularly useful for storing unstructured data, which is growing far more rapidly than structured data and does not fit the relational schemas of RDBMS. Common types of unstructured data include: user and session data; chat, messaging, and log data; time series data such as IoT and device data; and large objects such as video and images.

Benefits of NoSQL Databases

**Scalability**: NoSQL databases use a horizontal scale-out methodology that makes it easy to add or reduce capacity quickly and non-disruptively with commodity hardware.

**Performance**: By simply adding commodity resources, enterprises can increase performance with NoSQL databases.

**High Availability**: NoSQL databases are generally designed to ensure high availability and avoid the complexity that comes with a typical RDBMS architecture that relies on primary and secondary nodes.

**Global Availability**: By automatically replicating data across multiple servers, data centers, or cloud resources, distributed NoSQL databases can minimize latency and ensure a consistent application experience wherever users are located.

**Flexible Data Modeling**: NoSQL offers the ability to implement flexible and fluid data models.

Example of NoSQL Database: Cassandra, Hbase, MongoDB

2. **What are different types of No SQL databases**

There are four different type of No SQL Databases

1. **Key-value stores**

Key-value stores, or key-value databases, implement a simple data model that pairs a unique key with an associated value. Because this model is simple, it can lead to the development of key-value databases, which are extremely performant and highly scalable for session management and caching in web applications. Implementations differ in the way they are oriented to work with RAM, solid-state drives or disk drives. Examples include Aerospike, Berkeley DB, MemchacheDB, Redis and Riak.

1. **Document databases**

Document databases, also called document stores, store semi-structured data and descriptions of that data in document format. They allow developers to create and update programs without needing to reference master schema. Use of document databases has increased along with use of JavaScript and the JavaScript Object Notation (JSON), a data interchange format that has gained wide currency among web application developers, although XML and other data formats can be used as well.  Document databases are used for content management and mobile application data handling. Couchbase Server, CouchDB, DocumentDB, MarkLogic and MongoDB are examples of document databases.

1. **Wide-column stores**

Wide-column stores organize data tables as columns instead of as rows. Wide-column stores can be found both in SQL and NoSQL databases. Wide-column stores can query large data volumes faster than conventional relational databases. A wide-column data store can be used for recommendation engines, catalogs, fraud detection and other types of data processing.  Google BigTable, Cassandra and HBase are examples of wide-column stores.iv.

1. **Graph stores**

Graph data stores organize data as nodes, which are like records in a relational database, and edges, which represent connections between nodes. Because the graph system stores the relationship between nodes, it can support richer representations of data relationships. Also, unlike relational models reliant on strict schemas, the graph data model can evolve over time and use. Graph databases are applied in systems that must map relationships, such as reservation systems or customer relationship management. Examples of graph databases include AllegroGraph, IBM Graph, Neo4j and Titan.

3. CAP Theorem

**CAP theorem**, also named **Brewer's theorem** after computer scientist [Eric Brewer](https://en.wikipedia.org/wiki/Eric_Brewer_(scientist)), states that it is impossible for a [distributed data store](https://en.wikipedia.org/wiki/Distributed_data_store) to simultaneously provide more than two out of the following three guarantees:

Consistency: Every read receives the most recent write or an error. Performing read operation will see the value of most recent write causing all nodes to return the same data. A system is in consistent state if a transaction starts with a consistent state and ends with a system in consistent state.

Typical Relational Database are consistent: MySQL, PostgresSQL

Availability: Every request receives a (non-error) response – without guarantee that it contains the most recent write. Availability in a distributed system means system remains available 100% of time.

Partition Tolerance: The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes. A system which is partition tolerant can sustain any amount of network failure that does not result in failure of entire network.

Storage systems that fall under Consistency and Partition Tolerance (CP) are :MongoDB, Redis

Storage systems that fall under Availability and Partition Tolerance are : Cassandra, CouchDB, DynamoDB

4. Hbase architecture

Physically, HBase is composed of three types of servers in a master slave type of architecture. Region servers serve data for reads and writes. When accessing data, clients communicate with HBase RegionServers directly. Region assignment, DDL (create, delete tables) operations are handled by the HBase Master process. Zookeeper, which is part of HDFS, maintains a live cluster state.

The Hadoop DataNode stores the data that the Region Server is managing. All HBase data is stored in HDFS files. Region Servers are collocated with the HDFS DataNodes, which enable data locality (putting the data close to where it is needed) for the data served by the RegionServers. HBase data is local when it is written, but when a region is moved, it is not local until compaction.

The NameNode maintains metadata information for all the physical data blocks that comprise the files.

## **Regions**

HBase Tables are divided horizontally by row key range into “Regions.” A region contains all rows in the table between the region’s start key and end key. Regions are assigned to the nodes in the cluster, called “Region Servers,” and these serve data for reads and writes. A region server can serve about 1,000 regions.

## **HBase HMaster**

Region assignment, DDL (create, delete tables) operations are handled by the HBase Master.

A master is responsible for:

* Coordinating the region servers

- Assigning regions on startup , re-assigning regions for recovery or load balancing

- Monitoring all RegionServer instances in the cluster (listens for notifications from zookeeper)

* Admin functions

- Interface for creating, deleting, updating tables

## ZooKeeper: The Coordinator

HBase uses ZooKeeper as a distributed coordination service to maintain server state in the cluster. Zookeeper maintains which servers are alive and available, and provides server failure notification. Zookeeper uses consensus to guarantee common shared state. Note that there should be three or five machines for consensus.

## HBase Meta Table

* This META table is an HBase table that keeps a list of all regions in the system.
* The .META. table is like a B tree.
* The .META. table structure is as follows:

- Key: region start key,region id

- Values: RegionServer

## **Region Server Components**

A Region Server runs on an HDFS data node and has the following components:

* WAL: Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.
* BlockCache: is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.
* MemStore: is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStore per column family per region.
* Hfiles store the rows as sorted KeyValues on disk.

5. Hbase vs Relational Database

|  |  |
| --- | --- |
| Hbase | RDBMS |
| HBase is schema less. It does not concept of fixed columns | An RDBMS is governed by its schema, which describes the whole structure of tables. |
| It is built for wide tables. HBase is horizontally scalable. | It is thin and built for small tables. Hard to scale. |
| No transactions are there in HBase. | RDBMS is transactional. |
| It has de-normalized data. | It will have normalized data |
| It is good for semi-structured as well as structured data. | It is good for structured data. |