# NonRelational Database

1)What is and Why NoSQL

2)NoSQL characteristics

3)NoSQL databases types

4)What is ACID theorem

5)What is CAP theorem

6) NoSQL advantages

# What is and Why NoSQL

-NoSQL is an approach to database design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stands for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built.

-In recent years, the databases have grown into thousands of terabytes, classic databases (RDBMS) are not able to manage this huge volume of Data.

-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 characteristics

-It’s more than rows in tables—NoSQL systems store and retrieve data from many formats: key-value stores, graph databases, column-family (Bigtable) stores, document stores, and even rows in tables.

-It’s free of joins—NoSQL systems allow you to extract your data using simple interfaces without joins.

-It’s schema-free—NoSQL systems allow you to drag-and-drop your data into a folder and then query it without creating an entity-relational model.

-It works on many processors—NoSQL systems allow you to store your database on multiple processors and maintain high speed performance.

-It uses shared-nothing commodity computers—Most (but not all) NoSQL systems leverage low-cost commodity processors that have separate RAM and disk.

-It supports linear scalability—When you add more processors, you get a consistent increase in performance.

-It’s innovative—NoSQL offers options to a single way of storing, retrieving, and manipulating data. NoSQL supporters (also known as NoSQLers) have an inclusive

-Attitude about NoSQL and recognize SQL solutions as viable options. To the NoSQL community, NoSQL means “Not only SQL”.

# NoSQL databases types

-There are four big NoSQL types:

-key-value store

-document store

- column-oriented database

-graph database.

## KEY-VALUE STORES:

-Key-value stores are the least complex of the NoSQL databases. They are, as the name suggests, a collection of key-value pairs, as shown in image , and this simplicity makes them the most scalable of the NoSQL database types, capable of storing huge amounts of data.

-The value in a key-value store can be anything: a string, a number, but also an entire new set of key-value pairs encapsulated in an object. image shows a slightly more complex key value structure. Examples of key-value stores are Redis, Voldemort, Riak, and Amazon’s Dynamo.

## DOCUMENT STORES:

-Document stores are one step up in complexity from key-value stores: a document store does assume a certain document structure that can be specified with a schema. Document stores appear the most natural among the NoSQL database types because they’re designed to store everyday documents as is, and they allow for complex querying and calculations on this often already aggregated form of data.

## COLUMN-ORIENTED DATABASE:

-A column-oriented database stores each column continuously. i.e. on disk or in memory each column on the left will be stored in sequential blocks.

-For analytical queries that perform aggregate operations over a small number of columns retrieving data in this format is extremely fast. As PC storage is optimized for block access, by storing the data beside each other we exploit locality of reference. On hard disk drives this is particularly important which due to their performance characteristics provide optimal performance for sequential access.

## GRAPH DATABASES:

-The last big NoSQL database type is the most complex one, geared toward storing relations between entities in an efficient manner. When the data is highly interconnected, such as for social networks, scientific paper citations, or capital asset clusters, graph databases are the answer.

# What is ACID theorem

-In computer science, ACID (atomicity, consistency, isolation, durability) is a set of properties of database transactions intended to guarantee data validity despite errors, power failures, and other mishaps. In the context of databases, a sequence of database operations that satisfies the ACID properties (which can be perceived as a single logical operation on the data) is called a transaction.

# What is CAP theorem

-CAP theorem also known as Brewer’s theorem was introduced by computer scientist Eric Brewer at Symposium on Principles of Distributed computing in 2000. In CAP theorem, C stands for Consistency, A stands for Availability and P stands for Partition tolerance.

-Today, NoSQL databases are classified based on the two CAP characteristics they support:

-CP database: A CP database delivers consistency and partition tolerance at the expense of availability. When a partition occurs between any two nodes, the system has to shut down the non consistent node (i.e., make it unavailable) until the partition is resolved.

-AP database: An AP database delivers availability and partition tolerance at the expense of consistency. When a partition occurs, all nodes remain available but those at the wrong end of a partition might return an older version of data than others. (When the partition is resolved, the AP databases typically resync the nodes to repair all inconsistencies in the system.)

-CA database: A CA database delivers consistency and availability across all nodes. It can’t do this if there is a partition between any two nodes in the system, however, and therefore can’t deliver fault tolerance.

# NoSQL advantages

## 1)Less need for ETL:

-NoSQL databases support storing data “as is.” Key value stores give you the ability to store simple data structures, whereas document NoSQL databases provide you with the ability to handle a range of flat or nested structures.

-Most of the data flying between systems does so as a message. Typically, the data takes one of these formats:

-A binary object to be passed through a set of layers.

-An XML document.

-A JSON document

-Being able to handle these formats natively in a range of NoSQL databases lessens the amount of code you have to convert from the source data format to the format that needs storing. This is called extract, transform, and load (ETL)

## 2)Support for unstructured text

-The vast majority of data in enterprise systems is unstructured. Many NoSQL databases can handle indexing of unstructured text either as a native feature (MarkLogic Server) or an integrated set of services including Solr or Elasticsearch.

-Being able to manage unstructured text greatly increases information and can help organizations make better decisions. For example, advanced uses include support for multiple languages with facetted search, snippet functionality, and word-stemming support. Advanced features also include support for dictionaries and thesauri.

-Furthermore, using search alert actions on data ingest, you can extract named entities from directories such as those listing people, places, and organizations, which allows text data to be better categorized, tagged, and searched.

-Entity enrichment services such as SmartLogic, OpenCalais, NetOwl, and TEMIS Luxid that combine extracted information with other information provide a rich interleaved information web and enhance efficient analysis and use.

## 3)Ability to handle change over time

-Because of the schema agnostic nature of NoSQL databases, they’re very capable of managing change — you don’t have to rewrite ETL routines if the XML message structure between systems changes.

-Some NoSQL databases take this a step further and provide a universal index for the structure, values, and text found in information. Microsoft DocumentDB and MarkLogic Server both provide this capability.

-If a document structure changes, these indexes allow organizations to use the information immediately, rather than having to wait for several months before you can test and rewrite systems.

## 4)Breadth of functionality

-Most relational databases support the same features but in a slightly different way, so they are all similar.

-NoSQL databases, in contrast, come in four core types: key-value, columnar, document, and triple stores. Within these types, you can find a database to suit your particular (and peculiar!) needs. With so much choice, you’re bound to find a NoSQL database that will solve your application woes.

## 5)Support for multiple data structures

-Many applications need simple object storage, whereas others require highly complex and interrelated structure storage. NoSQL databases provide support for a range of data structures.

## 6)No legacy code

-Because they are so new, NoSQL databases don’t have legacy code, which means they don’t need to provide support for old hardware platforms or keep strange and infrequently used functionality updated.

-NoSQL databases enjoy a quick pace in terms of development and maturation. New features are released all the time, and new and existing features are updated frequently (so NoSQL vendors don’t need to maintain a very large code base). In fact, new major releases occur annually rather than every three to five years.