BIGDATA ANALYTICS

UNIT-III

Bigdata Terminologies and Database:

Introduction to NOSQL:

What is NoSQL Databases?

NoSQL is commonly referred to as not only SQL, non SQL, or non relational databases. These non SQL databases are built for extreme high throughput using key value pairs versus relational databases with relative dependence. Using loose dependences and quick indexes NoSQL databases are perfect for Streaming Analytics and IoT applications because data can quickly be stored and referenced.

SQL	NOSQL
Relational dependences	Loose dependences
Updates to tables time consuming	Updates to tables on-demand
Performance dependent on queries & indexes	Performance depend on hardware & network
Rigid scaling	Elastic scaling
Costly	Cost Efficient

Reasons to Use a NoSQL Database

- Storing large volumes of data without structure. A **NoSQL** database doesn't limit storable data types. ...
- **Using** cloud computing and storage. Cloud-based storage is a great solution, but it requires data to be easily spread across multiple servers for scaling. ...
- Rapid development.

Features of NoSQL

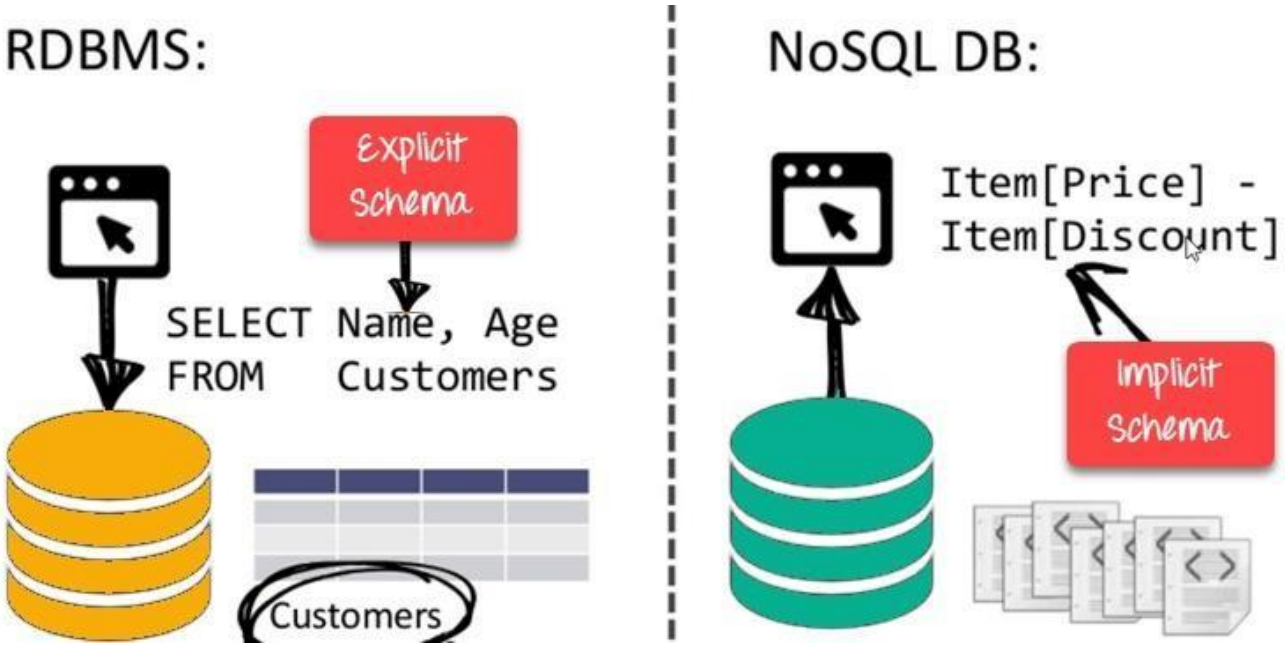
Non-relational

- 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

Schema-free

- 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



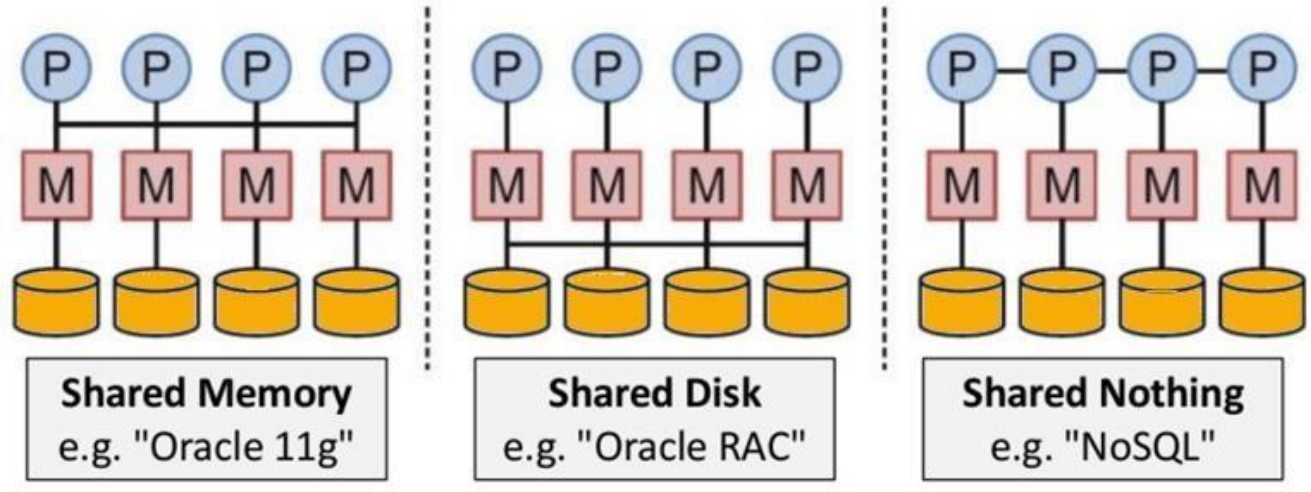
NoSQL is Schema-Free

Simple API

- 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 no standard based query language
- Web-enabled databases running as internet-facing services

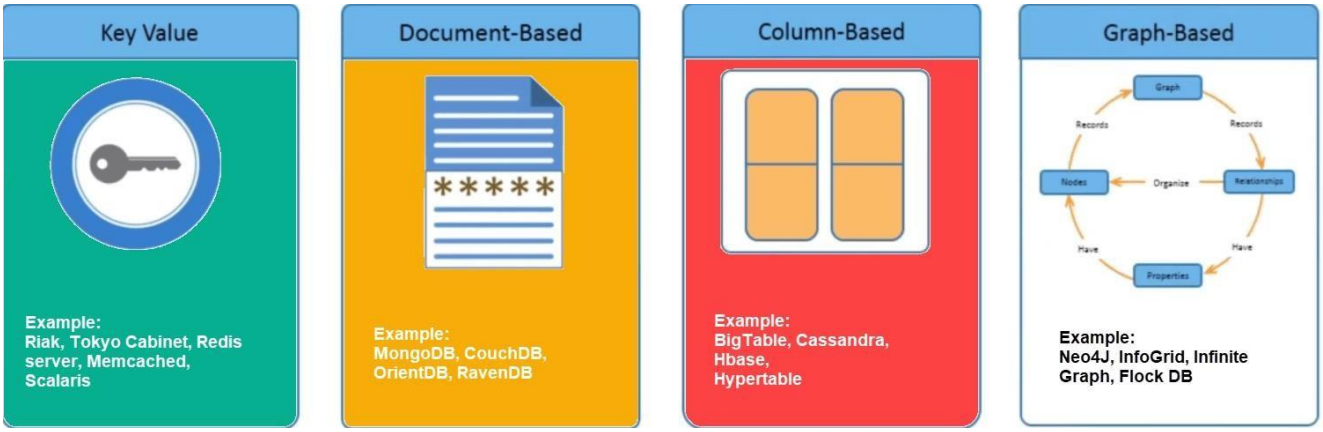
Distributed

- 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
- Shared Nothing Architecture. This enables less coordination and higher distribution.



NoSQL is Shared Nothing.

Types of NoSQL Databases



There are mainly four categories of NoSQL databases. Each of these categories has its unique attributes and limitations. No specific database is better to solve all problems. You should select a database based on your product needs.

Let see all of them:

- Key-value Pair Based
- Column-oriented Graph
- Graphs based
- Document-oriented

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.

For example, a key-value pair may contain a key like "Website" associated with a value like "Guru99".

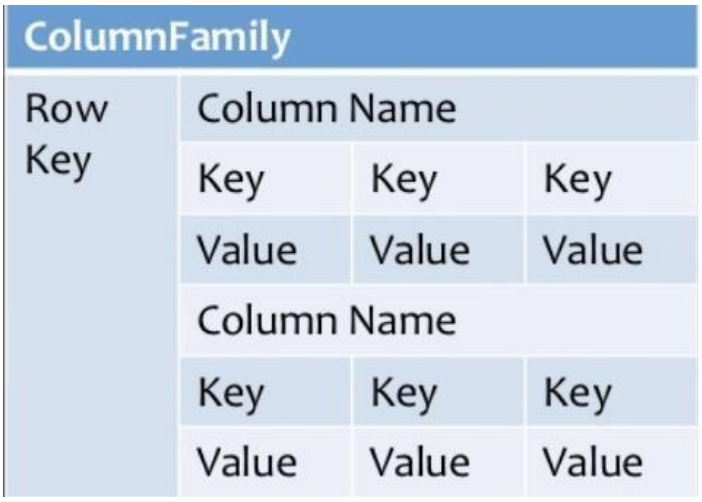
Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

It is one of the most basic types of NoSQL databases. 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 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.



Column based NoSQL database

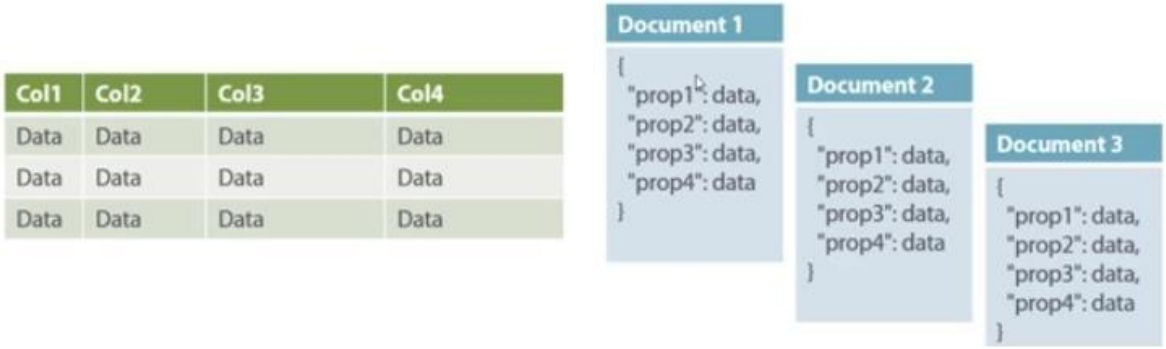
They deliver high performance on aggregation queries like SUM, COUNT, AVG, MIN etc. as the data is readily available in a column.

Column-based NoSQL databases are widely used to manage data warehouses, business intelligence, CRM, Library card catalogs,

HBase, Cassandra, HBase, Hypertable are 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 this diagram on your left you can see 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

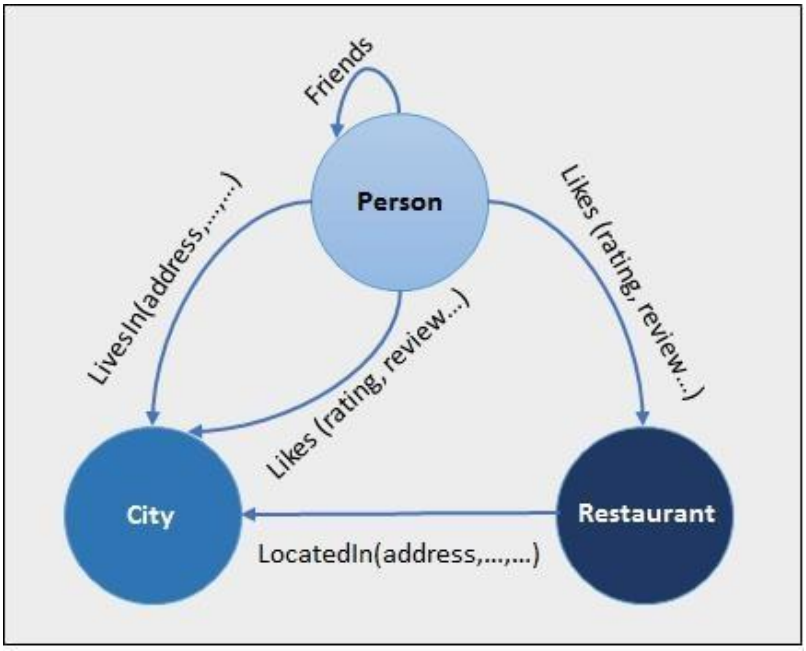
columns you have and so on. However, for a document database, you have data store like JSON object. You do not require to define which make it flexible.

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.



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.

Advantages of NoSQL

- Can be used as 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 which 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.

NoSQL Applications

Facebook messaging platform

Apache Cassandra was created by Facebook to power their Inbox. It did this for a number of years. Cassandra worked by doing the following:

- Cassandra indexed users' messages and the terms (words, and so on) in the messages and drove a search over all the content in those messages. The user ID was the primary key. Each term became a super column, and the message IDs were the column names.

Amazon DynamoDB

Amazon originally published the Dynamo paper, thereby launching the concept of NoSQL key-value stores. Since then, Amazon has created a separate database called **DynamoDB** as a service offered on the Amazon Web Services marketplace site.

Google Mail

Google's Bigtable was created to provide wide-column storage for a range of Google's applications, including Orkut, Google Earth, web indexing, Google Maps, Google Books, YouTube, blogger.com, Google Code and **Google Mail.**

LinkedIn

LinkedIn has used Hadoop to churn information about relationships overnight and to push the latest graph information to the Voldemort key-value NoSQL store for query the next day. In this way, LinkedIn maintained a rolling view of all data in the service.

BBC iPlayer online media catalog

The British Broadcasting Corporation has an online service to provide UK citizens with a free catchup service called the iPlayer for BBC television and radio shows.

The information for episodes, series, and brands is updated by a different team from that responsible for scheduling episodes for TV.

BBC Sport and Olympics platforms

In 2011, the BBC realized that its journalists were spending a lot of time deciding where to publish stories on the BBC Sport website. This cost a lot of time and money and stories weren't consistently available to users in different areas of the sports website.

HealthCare.gov

Healthcare.gov has been called the most complex IT system implementation of all time. Building it required several systems, with the most visible one being the HealthCare.gov marketplace.

UK NHS Spine 2 Backbone

The UK National Health Service comprises hundreds of organizations, all under one national umbrella. For example, general practice surgeries and hospitals each have their own systems.

Secure information sharing

In many situations, you need to provide access to information while also maintaining its security. Here are several examples:

- A book publisher providing access to summaries so that you can verify the relevance of a book before purchase, but only view the full book after purchase

Citizen engagement

Governments use NoSQL databases to empower citizens with information about how their country is governed. A good example is Fairfax County in Virginia, which uses MarkLogic Server to provide geospatial information through an online browse and search interface to government agencies and residents.

The service covers a range of information — for example, **geographic points in the county and police-related events**.

Overview of NewSQL

While various variants of the NoSQL database continue to be used, there is another paradigm arising in parallel to NoSQL — **NewSQL**. NewSQL promises to combine benefits from RDBMS (strong consistency) with benefits from NoSQL (scalability); it mainly achieves this through new architecture patterns and efficient SQL storage engines.

Most current NewSQL databases are based on Google's Spanner database and the theories in academic papers such as Calvin: Fast Distributed Transactions for Partitioned Database Systems from Yale. Spanner is Google's scalable, multi-version, globally-distributed, and synchronously-replicated database. It was the first system to distribute data at global scale and support externally-consistent distributed transactions. The *Calvin* academic paper from Yale was on leveraging determinism to guarantee active replication and full ACID-compliance of distributed transactions without two-phase commit.

Comparing SQL, NoSQL, and NewSQL

Feature	**SQL**	**NoSQL**	**NewSQL**
Relational Property	Yes, follows relational modeling to a large extent.	No, it doesn't follow a relational model, it was designed to be entirely different from that.	Yes, since the relational model is equally essential for real-time analytics.
ACID	Yes, ACID properties are fundamental to their application	No, rather provides for CAP support	Yes, Acid properties are taken care of.

SQL	Support for SQL	No support for old SQL	Yes, proper support and even enhanced functionalities for Old SQL
OLTP	Inefficient for OLTP databases.	Supports such databases but it is not the best suited.	Fully functionally supports OLTP databases and is highly efficient
Scaling	Vertical scaling	Vertical scaling	Vertical + Horizontal scaling
Query Handling	Can handle simple queries with ease and fails when they get complex in nature	Better than SQL for processing complex queries	Highly efficient to process complex queries and smaller queries.
Distributed Databases	No	Yes	Yes

What Is MongoDB?

MongoDB is an open-source document database built on a horizontal scale-out architecture. Founded in 2007, MongoDB has a worldwide following in the developer community.

Instead of storing data in tables of rows or columns like SQL databases, each row in a MongoDB database is a document described in JSON, a formatting language.

Here's a simple JSON document describing contact information:

```
{
 "name" : "Carlos Smith",
 "title" : "Product Manager",
 "location" : "New York, NY",
 "twitter" : "@MongoDB",
 "facebook" : "@MongoDB"
}
```

Document databases are extremely flexible, allowing variations in the structure of documents and allowing storage of documents that are partially complete. One document can have others embedded in it.

Fields in a document play the role of columns in a SQL database, and like columns, they can be indexed to increase search performance.

Best of all for many developers, the programmer can change the structure of the database easily as needs change. Some say this turns data into code.

From its founding, MongoDB was built on a scale-out architecture, a structure that allows many small machines to work together to create systems that are fast and handle huge amounts of data.

MongoDB has always focused on providing developers an excellent user experience, which, in addition to all its other properties, has made MongoDB a favorite of developers worldwide for a huge variety of applications.

Why Use MongoDB?

MongoDB is a document database built on a scale-out architecture that has become popular with developers of all kinds who are building scalable applications using agile methodologies.

MongoDB was built for people who are building internet and business applications who need to evolve quickly and scale elegantly. If you are doing that, you should consider MongoDB.

Companies and development teams of all sizes use MongoDB because:

- The document data model is a powerful way to store and retrieve data that allows developers to move fast.
- MongoDB's horizontal, scale-out architecture can support huge volumes of both data and traffic.
- MongoDB has a great user experience for developers who can install MongoDB and start writing code immediately.
- MongoDB can be used everywhere by anyone:
    - For free through the open source community edition
    - In the largest data centers through the enterprise edition
    - In any of the major public clouds through MongoDB Atlas
- MongoDB has developed a large and mature platform ecosystem, which means:
    - *MongoDB has a worldwide community of developers and consultants, so it is easy to get help.*

- MongoDB works on all types of computing platforms, both on-premise and in the cloud.
- MongoDB can be used from all major languages.
- MongoDB can be accessed from all major ETL and data management systems.
- MongoDB has enterprise-grade support.

Why use MongoDB? Simply to go further and faster when developing software applications that have to handle data of all sorts in a scalable way.

Thousands of companies like Bosch, Barclays, and Morgan Stanley run their businesses on MongoDB, and use it to handle their most demanding apps in areas like IoT, Gaming, Logistics, Banking, e-Commerce, and Content Management.

MongoDB is a great choice if you need to:

- Represent data with natural clusters and variability over time or in its structure
- Support rapid iterative development.
- Enable collaboration of a large number of teams
- Scale to high levels of read and write traffic.
- Scale your data repository to a massive size.
- Evolve the type of deployment as the business changes.
- Store, manage, and search data with text, geospatial, or time series dimensions.

MongoDB as a company has grown because the number of use cases with these characteristics keep growing.

Characteristics of MangoDB

High Performance

MongoDB provides high performance data persistence. In particular,

- Support for embedded data models reduces I/O activity on database system.
- Indexes support faster queries and can include keys from embedded documents and arrays.

Rich Query Language

MongoDB supports a rich query language to support read and write operations (CRUD) as well as:

- Data Aggregation
- Text Search and Geospatial Queries.

SEE ALSO

- SQL to MongoDB Mapping Chart
- SQL to Aggregation Mapping Chart

High Availability

MongoDB's replication facility, called replica set, provides:

- *automatic* failover
- data redundancy.

A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.

Horizontal Scalability

MongoDB provides horizontal scalability as part of its *core* functionality:

- Sharding distributes data across a cluster of machines.

- Starting in 3.4, MongoDB supports creating zones of data based on the shard key. In a balanced cluster, MongoDB directs reads and writes covered by a zone only to those shards inside the zone. See the Zones manual page for more information.

Support for Multiple Storage Engines

MongoDB supports multiple storage engines:

- WiredTiger Storage Engine (including support for Encryption at Rest)
- In-Memory Storage Engine.

In addition, MongoDB provides pluggable storage engine API that allows third parties to develop storage engines for MongoDB.

What is Apache Cassandra?

Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for managing very large amounts of structured data spread out across the world. It provides highly available service with no single point of failure.

Listed below are some of the notable points of Apache Cassandra −

- It is scalable, fault-tolerant, and consistent.
- It is a column-oriented database.
- Its distribution design is based on Amazon's Dynamo and its data model on Google's Bigtable.
- Created at Facebook, it differs sharply from relational database management systems.
- Cassandra implements a Dynamo-style replication model with no single point of failure, but adds a more powerful "column family" data model.
- Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Features of Cassandra

Cassandra has become so popular because of its outstanding technical features. Given below are some of the features of Cassandra:

- **Elastic scalability** − Cassandra is highly scalable; it allows to add more hardware to accommodate more customers and more data as per requirement.
- **Always on architecture** − Cassandra has no single point of failure and it is continuously available for business-critical applications that cannot afford a failure.
- **Fast linear-scale performance** − Cassandra is linearly scalable, i.e., it increases your throughput as you increase the number of nodes in the cluster. Therefore it maintains a quick response time.

- **Flexible data storage** − Cassandra accommodates all possible data formats including: structured, semi-structured, and unstructured. It can dynamically accommodate changes to your data structures according to your need.
- **Easy data distribution** − Cassandra provides the flexibility to distribute data where you need by replicating data across multiple data centers.
- **Transaction support** − Cassandra supports properties like Atomicity, Consistency, Isolation, and Durability (ACID).
- **Fast writes** − Cassandra was designed to run on cheap commodity hardware. It performs blazingly fast writes and can store hundreds of terabytes of data, without sacrificing the read efficiency.

History of Cassandra

- Cassandra was developed at Facebook for inbox search.
- It was open-sourced by Facebook in July 2008.
- Cassandra was accepted into Apache Incubator in March 2009.
- It was made an Apache top-level project since February 2010.

****