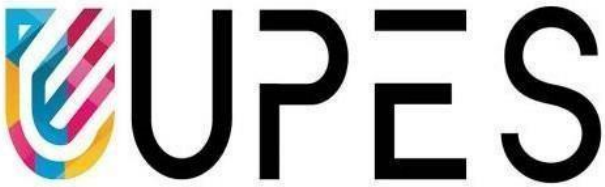


Cassandra A New Reliable Database

SUBMITTED TO:

Mr. GOURAV BATHLA

SUBMITTED BY:

NAME	SAP ID
JYOTI SINGH	500106507
ANJALI SHARMA	500104878
ADITYA PETSHALI	500100956

MASTER OF COMPUTER APPLICATION
SCHOOL OF COMPUTER SCIENCE

Table of Contents

What is a database-

A database is a group of data that has been arranged so that it can be saved electronically on a computer or in the cloud. The term "query language" refers to a type of language that is needed to perform certain tasks, such as inserting data into a database or retrieving data from one. There are numerous varieties of query languages, with SQL (Structured query language) being the most popular.

Database management systems, or DBMS for short, are responsible for managing databases. A DBMS is a piece of system software that enables users to create and maintain databases.

Database and its history-

When data was kept on punched cards, magnetic tapes, or discs in the early days of computing, the idea of a database first emerged. However, the first database management systems (DBMS) were created in the 1960s, which is when the term "database" was first used.

The following are a few of the popular database types:

Relational DBMS: In the 1970s, Edgar Codd introduced a brand-new database paradigm called the relational database, in which data is arranged in tables with rows and columns in each table. Keys and foreign keys can be used to connect the tables by shared attributes. Oracle is an illustration of a relational DBMS.

Object-oriented DBMS: In the 1990s, object-oriented databases also began to gain popularity along with object-oriented programming. In this type of database, the data is organised as objects, which are instances of classes with attributes and methods. Tables or collections can be used to store the objects. O2 is an illustration of an object-oriented DBMS.

NoSQL DBMS: This type of database management system (DBMS) was launched in the 2000s, and it allows data to be organised in a number of different ways that do not adhere to the relational model, such as key-value pairs, documents, graphs, or columns. Large volumes of unstructured or semi-structured data can be handled by NoSQL DBMS with great scalability and performance. MongoDB is a prime example of a NoSQL DBMS.

SQL and NoSQL -

Although both SQL and NoSQL are data bases, they differ from one another in many important ways, such as the fact that SQL works with structured data whereas NoSQL works with unstructured data.

While NoSQL databases are referred to as non-relational or distributed databases, SQL databases are also known as Relational Databases (RDBMS). Structured Query Language (SQL), a robust and flexible alternative for complicated queries, is used by SQL databases to define and handle data. The fixed data structure and preset schemas of SQL databases, however, make updates challenging and disruptive to the entire system.

NoSQL databases, in contrast, support document-oriented, column-oriented, graph-based, or Key Value storage through the use of a dynamic schema for unstructured data. Due to its flexibility, documents can be written without a predetermined structure and each one can have a different structure. NoSQL databases are the best option for big or constantly changing data sets because they are horizontally scalable, or can manage additional traffic by adding more servers.

While NoSQL databases are key-value pairs, document-based, graph databases, or wide-column stores, SQL databases are table-based. While NoSQL databases adhere to the Brewers CAP theory, SQL databases follow the ACID characteristics.

While certain NoSQL databases rely on community help and only a small number of professionals are accessible for large-scale NoSQL deployments, SQL databases provide excellent vendor support and independent consults.

Difference between SQL and NoSQL-

SQL	NoSQL
Relational database management system (RDMS)	Non-relational or distributed database system.
These databases have predefined, fixed, or static schema.	Dynamic schema are present.
Hierarchical data storage is not appropriate for these databases.	These databases work well for the storage of hierarchical data.
Complex queries work best with these databases.	Complex searches are not well suited for these databases.
Vertically scalable	Horizontally scalable
Abides by the ACID (Atomicity, Consistency, Isolation, and Durability) property	Adheres to the CAP (consistency availability, partition tolerance)
EX: MySQL, PostgreSQL,etc	EX: Cassandra, Mongo DB etc

Cassandra Database -

The NoSQL distributed database Cassandra offers high availability and scalability without sacrificing performance by handling enormous quantities of data across numerous servers. Originally created by Facebook and eventually given to the Apache Software Foundation, it is an open-source project. Due to its hybrid masterless architecture, Cassandra can tolerate outages and failures without losing any data. It is suitable for applications that need low latency and worldwide availability because it offers replication across several data centres and cloud regions.

The data model used by Cassandra is based on tables of columns. Each row in each table is identified specifically by a primary key. The partition key, which is made up of one or more columns in the primary key, controls how the data is partitioned among the cluster nodes. The clustering columns, which govern how the data is organised within each partition, can be added after the partition key. To improve the query capabilities, Cassandra additionally allows user-defined types, materialised views, and secondary indexes.

CQL, or Cassandra Query Language, is a proprietary query language for Cassandra that is comparable to SQL but has several limitations. Users can build, change, and query tables, as well as the keyspace (a logical storage space for tables), using CQL. Additionally, CQL provides functions, lightweight transactions, and batch operations. Cassandra offers certain alternatives, such as collections (sets, lists, and maps), counters, tuples, and aggregates, in place of joins, subqueries, and aggregation functions.

Cassandra provides a number of capabilities to guarantee the database's security and observability. To restrict access to information and resources, it offers authentication and authorisation techniques. Additionally, fqltool and audit logs are provided for tracking and replaying user behaviour. Nodetool, JMX, metrics, tracing, and logging are just a few of the tools that Cassandra provides for keeping an eye on cluster health and performance.

For many businesses and organisations that require a dependable and scalable database for their mission-critical applications, like Apple, eBay, Netflix, and Twitter, Cassandra is a preferred option. Among Cassandra's use cases include social media, texting, streaming, internet of things, e-commerce, gaming, and analytics.

History Of Cassandra-

Avinash Lakshman and Prashant Malik created the Cassandra database for the first time at Facebook in 2008 to answer the company's need for a distributed database system to handle their enormous volume of data. The database has the name of the mythical character Cassandra, who was endowed with the power of prophesy but was also cursed with the fact that no one would pay attention to her forecasts.

When Cassandra was made available as an open-source project in 2009, businesses like Twitter and Digg embraced it. Cassandra was elevated to a top-level project of the Apache Software Foundation in 2010, which is now in charge of overseeing its advancement and updates.

Cassandra has experienced a number of upgrades and enhancements over time, including the inclusion of functions like virtual nodes, native secondary indexes, and support for JSON data formats. The initial version of DataStax Enterprise, which includes further capabilities and tools for managing and deploying Cassandra at scale, was launched by DataStax, a business that offers commercial support and services for Cassandra, in 2015.

Today, a wide spectrum of businesses and organisations, from small start-ups to huge corporations, use Cassandra for a variety of use cases, including real-time analytics, e-commerce, messaging, and more. The database is a popular option for businesses with big data workloads due to its capacity to manage enormous amounts of data over several nodes with high availability and fault tolerance.

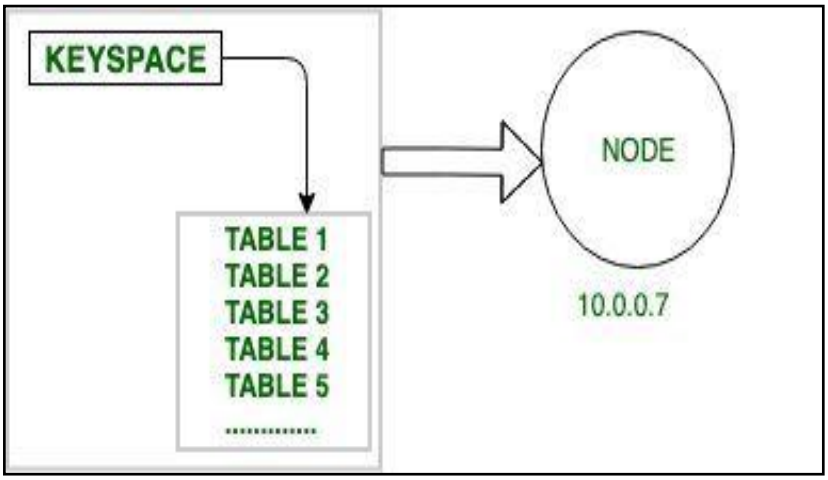
How Does Cassandra work?

We must become familiar with the three primary processes on which Cassandra is based in order to comprehend how it operates.
1. The architecture of Cassandra.
2. The partitioning scheme of Cassandra.
3. The replicatability of Cassandra.

1. Architecture of Cassandra-

A cluster of nodes makes up the main structure of Cassandra. Peer-to-peer systems like Amazon's Dynamo DB and Google Bigtable are examples of how Cassandra is constructed.

Node: The Cassandra component known as the node is where the actual data is kept. A single node can include several tables, and each node can have a maximum size of 4 TB. The size of a node can be determined based on how the user intends to utilise it.



If you look at the diagram, which depicts a keyspace with n different tables that merge into one node, it will become more evident.

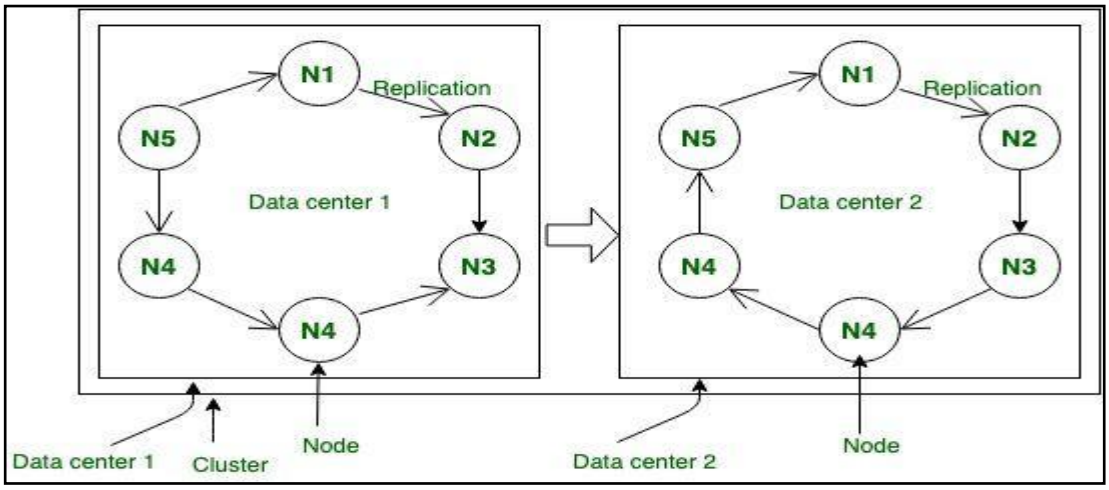
N=T1+T2+T3…..

Data Center: Data centres are nothing more than a collection of n nodes; that is, they are referred to as a grouping of nodes when there are many of them.

DC= N1+N2+N3…..

Cluster: Cluster is the term used to describe a grouping of several data centres. When n data centres come together to create a group, it is referred to as a cluster.

C=DC1+DC2+DC3….



The formation of a node, a data centre, and finally a cluster from the data is clearly depicted in the diagram.

2. The partitioning System-

Data is saved and retrieved in Cassandra using a portioning scheme. We specify where the primary copy of the data set is stored in the partitioner system. Data is partitioned among the nodes using a partition key. Cassandra uses a consistent hashing version to distribute data across the storage node.

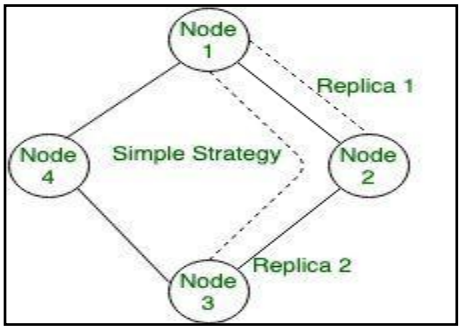
Example to create partitioning
CREATE TABLE T (id int, K int v text, PRIMARY KEY (id));
In the aforementioned example, the primary key serves as the partitioning key. Multiple attributes can be clustered into a primary key and used as the partitioning key.

Following the creation of the partitions, the keys are hashed using a hashing function and the hash values are saved in a hash table, allowing for quick lookup. A query's response time is accelerated if fewer partitions are used.

3. Cassandra's Replicability-

Cassandra works in such a way that it replicates the data of a single node into multiple nodes so that if one node is not currently available, another node can provide the functionality. Because of how reliable this data base is, there are very few chances that it will ever become unresponsive or unavailable.
Replica nodes are secondary nodes, and the replication factor (RF) determines how many replica nodes are required for a certain data collection. The identical data is

present on three nodes when the replication factor is 3.



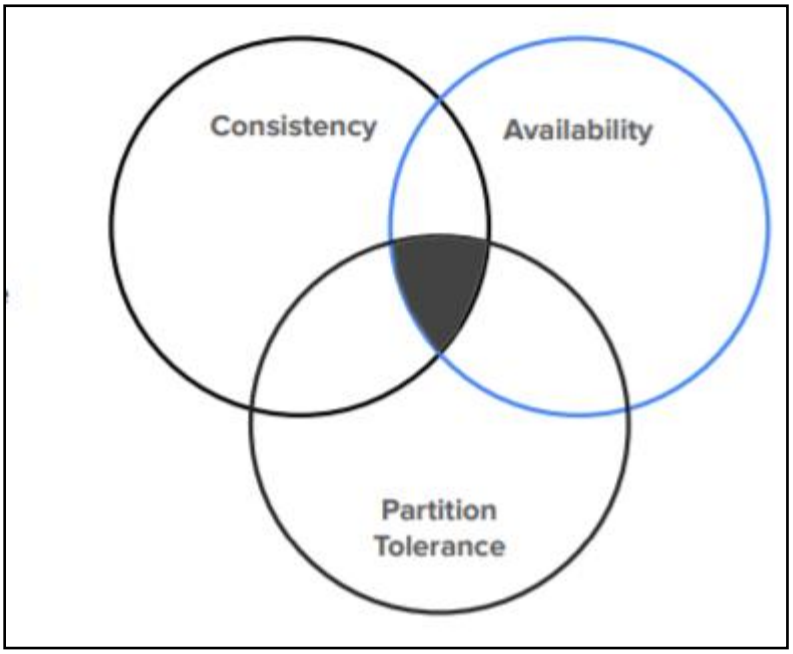
The CAP theorem: is Cassandra AP or CP?

In a failure scenario, a distributed database system can only guarantee two of the three qualities, according to the Consistency, Availability, and Partition Tolerance (CAP) theorem.

1.Consistency: When a query is consistent, it returns the most up-to-date or accurate information that has been updated in the tables. The system is said to be inconsistent if one of the servers returns out-of-date data.

2.Availability: The term "availability" refers to a system's ability to function even when one of its components—such as a server—is malfunctioning or unavailable.

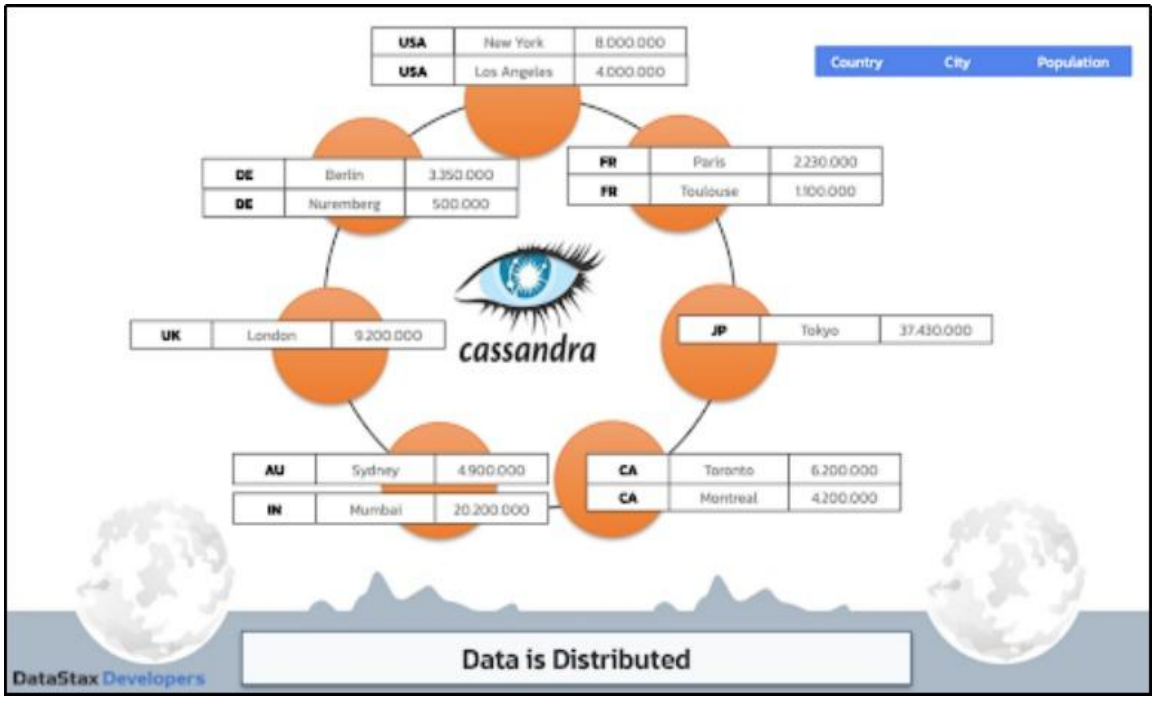
3.Partiton tolerance: Partition tolerance is the ability of a distributed system to tolerate network partitioning. One section of the server cannot connect to another due to network partitioning**.**



As a result, Cassandra is frequently referred to as an AP system, which means that in the event of a problem, the database will prioritise availability and partition tolerance over consistency. However, Cassandra is more adaptable; it may be modified to become either an AP or a CP depending on the user's use case.

Data distribution and structuring in Cassandra-

Without experiencing any downtime, the Cassandra architecture is built to handle and distribute very large amounts of data across thousands of servers. In a cluster known as token-aware, each Cassandra node and even each Cassandra driver is aware of data allocation, allowing the user system to quickly contact any server and obtain the desired result.



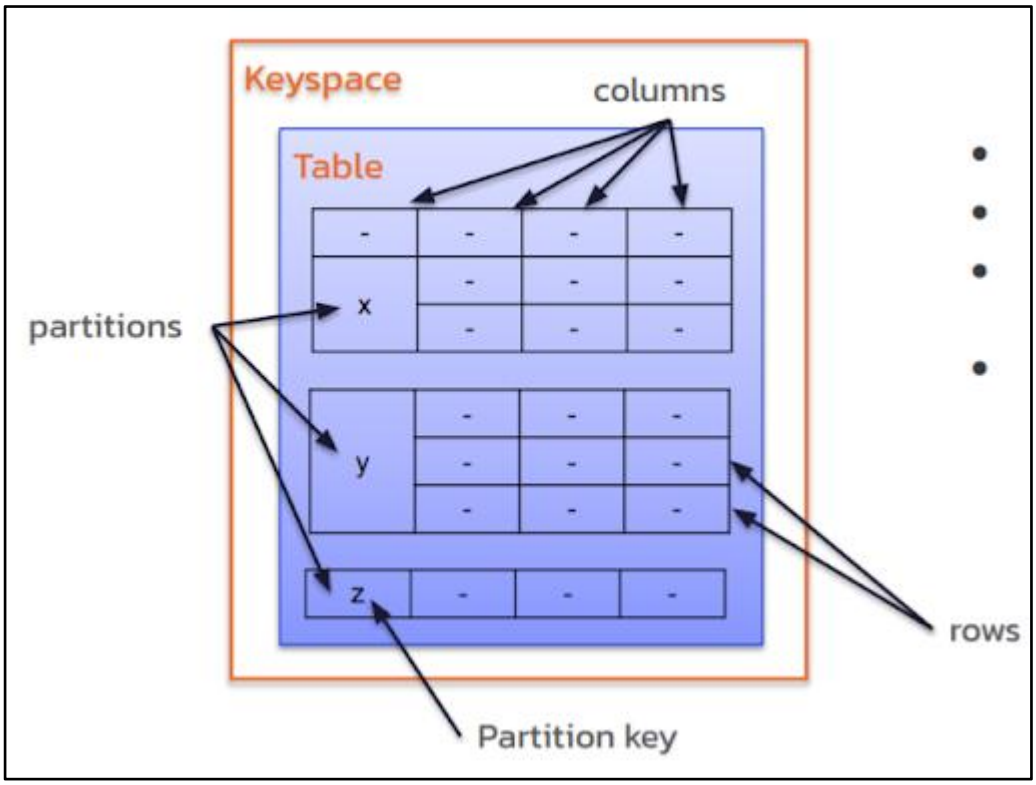
Cassandra employs partitioning based on keys. The key elements of Cassandra's data structure are as follows:

1.Keyspace: a collection of tables that acts as a data container and is comparable to a schema.

2.Table:a set of primary key columns and rows used to store data in partitions.

3.Partition:a collection of rows with the same partition token (Cassandra's fundamental unit of access).

4.Row: one row of structured data in a table.



Cassandra is a database that assists in managing a vast amount of data by partitioning it into manageable chunks. Data is divided over numerous servers using partitions, which each comprise rows of data and a partition key.

The data is divided into smaller pieces and assigned to various cluster nodes when a partition key is set. This facilitates load distribution and makes scaling up or down as necessary easy. The data is automatically redistributed to balance the load when a new node is added.

To ensure that your queries execute quickly and precisely, it's crucial to use a solid partition key. It's crucial to get your table's main key correct from the start since after you've set it, you can't change it.

Read and Write Operation in Cassandra-

Read Operation-

Data can be retrieved from a database and utilised for calculations, functions, or even just to see the data. There are three different types of read operations in Cassandra that a coordinator can send to a replica node. The coordinator is the primary node that receives write requests.

1. **Direct Request:**
   In this type of request, the coordinator contacts one of the replica nodes directly to deliver a read request for data.

2. **Digest Request:**
   In this type of request, the coordinator makes requests to replicas that are indicated by the consistency level; for example, if the consistency level is 2, then any two nodes in the data centre will acknowledge the existence of the data.

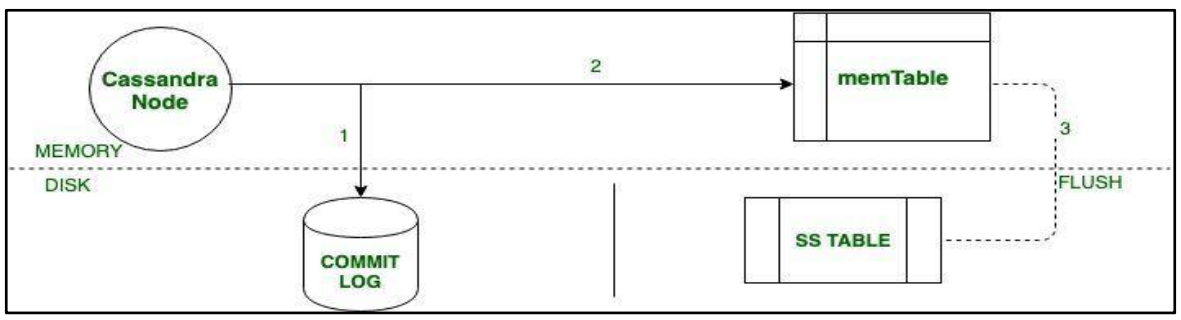
3. **Read Repair Request:** In this request type, if the data that has to be fetched is inconsistent among the nodes, a background read repair request is started that modifies all of the nodes based on the most recent data that is accessible in any node.

Write Operation-

Cassandra is renowned for its quick write operations and is frequently referred to as the Lamborghini of the NoSQL world. This quick storing is only feasible because Cassandra initially saves data in main memory before shifting it to disc, which makes it extremely quick in comparison to other databases.

1. Cassandra stores the data in the commit log in order to ensure that it is saved, as opposed to immediately inserting it into the main database.

2. After the data has been committed in the commit log, it is then saved in the memtable, which keeps the data in storage until it is full before notifying the Cassandra node that the data has been saved.
3. When the MemTable is full, a process called as flush is used to move the data



from the main memory to the disc.

1 Commit log:
Data is initially checked against the commit log before being written to the disc or memTable. In the event that a data node fails, the commit log acts as a synchronisation mechanism to guarantee data consistency.

2 MemTable:
Data is initially recorded to the commit log and then momentarily stored in the MemTable.

3 SS Table:
When the MemTable is full, the data is flushed to a disc file called an SSTable for long-term archival. This procedure makes sure that data is securely saved and that it is still possible to retrieve it in the event of a node failure.

Cassandra query language-

Cassandra created the Cassandra query language CQL to make advantage of Cassandra more effectively.

The distributed, scalable, and highly available NoSQL database system Cassandra uses the Cassandra Query Language (CQL) as its query language. Data is kept in tables with rows of columns in CQL's model, and queries can be run using the language's well-known syntax. The use of partition keys and clustering columns to specify the primary key and the physical layout of the data, as well as the lack of joins, group by, or subqueries, are some of the distinctions between CQL and SQL.

Users can design secondary indexes, materialised views, functions, triggers, and security roles as well as create, insert, update, delete, and manipulate data in Cassandra tables using CQL. Additionally, CQL has operators for filtering, sorting, paging, and aggregating data, and it supports the JSON format for data insertion and retrieval. The command-line programme cqlsh shell, which connects to a Cassandra node and enables users to interact with the database, can be used to run CQL. Drivers for several programming languages, such as Java, Python, Ruby, or C#, can also be used to run CQL.

CQL is intended to offer a straightforward and expressive way to work with Cassandra data while simultaneously utilising its distributed and fault-tolerant architecture. Users can model their data to suit the requirements of their applications and query it effectively and reliably with the aid of CQL. Additionally, CQL enables users to take advantage of Cassandra's capabilities, like user-defined types, lightweight transactions, and consistency levels that may be adjusted.

Cassandra query language syntax-

Similar to SQL, Cassandra Query Language (CQL) is a query language for Apache Cassandra. Here are a few fundamental CQL commands:

- CREATE TABLE: used to create a new table. For example:

  CREATE TABLE users (id int PRIMARY KEY, name text, age int);
  creates a new table called users with the specified columns and primary key.

- INSERT: used to insert data into a table. For example:

  INSERT INTO users (id, name, age) VALUES (1, 'John', 25);
  inserts a new row into the users table with the specified values.

- SELECT: used to query data from a table. For example:

  SELECT * FROM users;
  retrieves all columns and rows from the users table.

- UPDATE: used to update data in a table. For example:

  UPDATE users SET age = 26 WHERE id = 1;
  updates the age column for the row with id equal to 1 in the users table.

- DELETE: used to delete data from a table. For example:

  DELETE FROM users WHERE id = 1;
  deletes the row with id equal to 1 from the users table.

- CREATE KEYSPACE: used to create a new keyspace. For example:

  CREATE KEYSPACE mykeyspace WITH replication = {
  'class': 'SimpleStrategy',
  'replication_factor': 3};
  creates a new keyspace called mykeyspace with the specified replication strategy and replication factor.

Use Case of Cassandra-

Cassandra is a strong database that is used for many different applications and can manage a lot of data. Here are some typical Cassandra use cases and the businesses that employ it in these fields:

1. **Messaging:** Cassandra is a wonderful database for businesses that offer mobile phones and messaging services since it can manage massive amounts of data.Cassandra is used by businesses like Twitter and Cisco WebEx for storage.

2. **High-speed applications:** Cassandra is a wonderful database for applications where data is flowing in very quickly from multiple devices or sensors. It can manage high-speed data.Cassandra is used by businesses like IBM and Cloudkick for storage.

3. **Product catalogues and retail apps:** Cassandra allows merchants to create robust product catalogues with quick input and output.For storage, Cassandra is used by organisations like OpenX and Rackspace.

4. **Analytics and recommendation engines for social media:** Social media service providers can utilise Cassandra for analysis and suggestions.Cassandra is used by businesses like Facebook, Instagram, and Reddit for storage.

5. **Transaction logging:** Cassandra can be used to log transactions like purchases, test results, watched movies, and movie theatre locations.Cassandra is used by businesses like Netflix and Spotify for storage.

6.Time series data: As long as you create your own aggregates, Cassandra can be utilised to store time series data.Cassandra is used for storage by organisations like Digg and Formspring.

Features that make Cassandra so powerful-

1.Open-source availability: Cassandra is a free, open-source solution that is hosted by Apache.

2.Distributed footprint: Cassandra is built with a distributed architecture and may operate on a number of nodes, each of which is equally important.

3.Scalability: Cassandra's elastic scalability allows it to adapt its size to changing demands.

4.Cassandra Query Language (CQL): Although similar to SQL, Cassandra's query language, CQL, was developed specifically for the architecture of the database.

5.Fault tolerance: Cassandra's distributed architecture and data replication between nodes make it exceptionally fault-tolerant and able to withstand node or data centre failures.

6.High performance: Due to its core architecture decisions, Cassandra consistently outperforms several prominent NoSQL competitors in benchmarks and practical applications.

Brief How companies are using Casandra-

How Netflix is Using Cassandra Database-

One of the top entertainment services in the world, Netflix has more than 200 million subscribers spread over more than 190 nations. Users can stream Netflix's extensive library of films, TV series, documentaries, and original material whenever they want on a variety of devices. Netflix depends on a strong and scalable data infrastructure that is capable of handling enormous amounts of data and traffic in order to provide its users with such a high-quality and individualised experience.

Cassandra, an open source NoSQL database that offers high availability, fault tolerance, linear scalability, and low latency, is one of the fundamental elements of Netflix's data architecture. The distributed database Cassandra stores data in a cluster of nodes, with each node handling a portion of the data. Cassandra employs a peer-to-peer architecture in which nodes coordinate reads and writes with one another and duplicate data across many nodes for consistency and redundancy.

When faced with the problem of moving its data from a single Oracle database in a data centre to the cloud, Netflix decided to embrace Cassandra in 2010. Netflix needed a database that could handle the growing volume and variety of data, as well as the unexpected and international nature of its traffic, in order to take advantage of the advantages of cloud computing, including as flexibility, scalability, and cost effectiveness. Cassandra was chosen by Netflix because it satisfied their demands for performance, dependability, and flexibility.

Cassandra manages virtually all of Netflix's database needs, including user accounts, movie ratings, metadata, movie bookmarks, logs, and more. Netflix uses Cassandra to power more than 50 Cassandra clusters with more than 750 nodes, where it stores 95% of its data. Netflix also makes use of other technologies, such as Spark for analytics, Kafka for streaming, Elasticsearch for full-text search, and JanusGraph for graph queries, in addition to Cassandra.

Some of the benefits that Netflix has gained from using Cassandra are:

1. **High availability:** Even in the event of node failures or network partitions, Cassandra makes sure that Netflix's service is constantly accessible to its users. Depending on the level of availability and consistency required, Netflix can specify how many copies of the data are stored in various nodes or regions using Cassandra's replication mechanism. For instance, Netflix replicates data three times inside each region and three times between regions to provide cross-region redundancy.

2. **Scalability:** Without experiencing any downtime or performance degradation, Cassandra enables Netflix to extend its data architecture horizontally by adding more nodes to the cluster. Data is evenly spread throughout the cluster thanks to Cassandra's partitioning strategy, and the nodes take care of load balancing automatically. By boosting each node's capability or performance, Netflix can also scale its data architecture vertically. For instance, Netflix boosts memory size for caching and employs SSDs for speedier reads and writes.

3. **Low latency:**For Netflix's read and write operations, Cassandra offers quick and reliable response times, which are essential for offering its users a seamless and customised viewing experience. The write performance of Cassandra is particularly excellent because it doesn't require locking or synchronisation between nodes. A commit log is used as a first step in the writing process before writing to memtable, an in-memory structure that is periodically flushed to disc as an immutable file called SSTable. Reads are served from memtables, SSTables, or, if a cache is configured, from that as well.

4. **Flexibility:** With Cassandra, Netflix can model its data without being restricted by a fixed schema or preset data model, allowing it to better meet its business goals. A wide variety of data types, including strings, numbers, lists, sets, maps, and user-defined types are supported by Cassandra. Netflix can add new characteristics to existing rows using Cassandra's dynamic columns without changing the schema. Additionally, Cassandra supports a variety of query languages, including Gremlin (Graph Query Language), which is used for graph queries, and CQL (Cassandra Query Language), which is similar to SQL.

Finally, for storing and processing massive volumes of data in the cloud, Netflix uses Cassandra as its main database solution. Through the use of Cassandra, Netflix is able to offer a dependable, scalable, and quick service to its consumers all over the world while also allowing it to adjust to shifting business needs and consumer preferences.

How Facebook is Using Cassandra Database-

With about 3 billion monthly active members, Facebook is one of the most widely used social networking sites in the world. Facebook need a scalable, high-performance, and trustworthy database system to manage such a vast volume of data. Facebook employs a number of database systems, including the distributed NoSQL database Cassandra, which offers high availability and fault tolerance.

Facebook created Cassandra in 2008 to power its Inbox Search function, which allowed users to look up messages and chats. Every node in Cassandra may accept requests and communicate with other nodes using a peer-to-peer protocol because it was created as a peer-to-peer system. Additionally, Cassandra provides a column family data model, enabling adaptable and dynamic schema construction.

When Facebook's new Messaging platform, based on HBase, another NoSQL database, was introduced in late 2010, it instead chose to use Inbox Search instead of Cassandra. For several use cases that required strict consistency assurances, Facebook discovered that Cassandra's eventual consistency model was challenging to work with. Contrarily, HBase provided consistent reads and writes across numerous data centres.

However, this does not imply that Facebook has completely stopped using Cassandra. In fact, Facebook continues to use Cassandra for some of its products and services, including Instagram, which it purchased in 2012. Instagram stores user information, including profiles, photographs, likes, comments, and follows, in Cassandra. Instagram, which has more than 1 billion monthly active users and more than 100 million photo uploads each day, also uses Cassandra to manage its enormous growth and size.

Some of the benefits that Cassandra provides for Instagram are:

1. **Horizontal scalability:** Without any downtime or data loss, Cassandra can quickly expand out by adding more nodes to the cluster.

2. **High availability:** Cassandra can withstand node failures and network splits without compromising the service or data availability.

3. Low latency: Cassandra distributes the data across numerous nodes and data centres so that requests can be processed quickly.

4. Tuneable consistency: Depending on the requirements and trade-offs of the application, Cassandra supports varying degrees of consistency.

For several of its services and applications that need scalability, high availability, and low latency, Facebook uses the Cassandra database, in conclusion. For storing vast amounts of unstructured or semi-structured data that may be accessed by key-value queries, Cassandra is an appropriate option. Facebook also makes use of various database systems for a variety of use cases that call for distinct features and capabilities because Cassandra is not a one-size-fits-all solution.

Difference between Cassandra and Other Database-

Depending on the use cases and circumstances, Cassandra and other DBMS offer distinct capabilities and advantages. Unstructured data and high data velocity can be handled by Cassandra, a NoSQL database. It offers decentralised installations and straightforward transactions. Collections are another feature that Cassandra offers to show relationships between data.

Relational databases (RDBMS) and other NoSQL databases are examples of additional DBMS. RDBMS employ SQL for database maintenance and querying and are built for structured data. They have master-slave systems with set schemas and a single point of failure. They enable nested, intricate transactions as well as centralised deployments. They make use of a database containing rows and columns, where rows stand in for individual records and columns for attributes. Keys and joins are also used by RDBMS to represent data relationships.

MongoDB, Redis, Memcached, DynamoDB, and more NoSQL databases are available. Depending on the kind of data they store and the performance they provide, they have various data models and features. For instance, the document-oriented database MongoDB stores data in documents that resemble JSON. It has a single master node with automatic fail-over and a customizable schema. It offers horizontal scaling and straightforward transactions. Data is stored using documents and collections, where documents resemble rows in a relational database management system but may include alternative fields. Additionally, MongoDB enables arrays and embedded documents to represent data relationships.

Difference between Cassandra and MySQL-

Cassandra	MySQL
Cassandra is a NoSQL database .	MySQL is a relational database.
It uses a distributed, large column store format to store data.	It uses the SQL query language to store data in tables.
Cassandra's high availability and scalability are made possible by its architecture to manage massive volumes of data across numerous servers.	Because it offers referential integrity and ACID features, MySQL is better suited for smaller datasets and single-server deployments.
Cassandra provides quick and effective read and write operations but does not enable complicated queries or joins.	Although MySQL allows a wide range of queries and joins, doing so takes more effort and resources.
provides techniques for ensuring data accuracy in a distributed system, including eventual consistency and immediate consistency.	only offers immediate consistency, which ensures constant data accuracy.
Any operating system that supports Java can run Cassandra because it is developed in Java.	MySQL may be used on FreeBSD, Linux, OS X, Solaris, and Windows and is written in C and C++.
Companies like Netflix, Instagram, Reddit, and Spotify use Cassandra.	Businesses like Airbnb, Pinterest, Slack, Twitter, and Udemy use MySQL.

Difference between Cassandra and PostgreSQL-

Cassandra	PostgreSQL
Cassandra uses a wide-column data model ,(column-family mode).	PostgreSQL is the data model.
Data is arranged into rows with dynamic columns grouped into column families or tables, allowing for flexible schema design.	It makes use of tables with a predetermined row-and-column schema.
Large datasets and heavy read/write workloads are no problem for this horizontally scalable system.	The primary method of vertical scaling for PostgreSQL is resource expansion on a single server.
Cassandra use the query language CQL (Cassandra Query Language).	SQL (Structured Query Language) is used by PostgreSQL.
Sub-queries, sophisticated joins, or aggregation operations are not supported by CQL.	A well-known and effective language for data manipulation is PostgreSQL.
Turntable consistency in Cassandra enables users to set the degree of consistency needed for their applications.	Strong consistency is emphasised by PostgreSQL, making sure that alterations are immediately apparent to subsequent reads.
With Cassandra, the replication factor can be chosen.	In PostgreSQL, master-master replication is supported.

Difference between Cassandra and Oracle-

Cassandra	Oracle
Cassandra can keep a lot of structured data on numerous servers.	On a single server or in a cluster, Oracle is capable of handling complicated queries and transactions.
The Apache Software Foundation created Cassandra in 2008.	The Oracle Corporation created the RDBMS in 1980.
composed in Java.	composed in C and C++.
Cassandra is an open-source software.	Oracle is a commercial software.
The broad column store model is used by Cassandra.	Using a relational model, Oracle
Both eventual consistency and instantaneous consistency strategies are supported by Cassandra.	Only the immediate consistency approach is supported by Oracle.
The shade partitioning approach is used by Cassandra.	Oracle uses horizontal partitioning method.
With Cassandra, the replication factor can be chosen.	Both master-slave and master-master replication techniques are supported by Oracle.
Transaction ideas are absent in Cassandra.	Oracle complies with ACID transactional properties.

Difference between Cassandra and MongoDB-

Cassandra	MongoDB
Columns and tables are used by Cassandra to store data.	MongoDB stores data in documents that resemble JSON.
For straightforward queries, Cassandra can provide a faster and more reliable response time.	More sophisticated and dynamic data structures are supported by MongoDB.
The query language (CQL) used by Cassandra is unique.	Python and Java are examples of third-party languages used by MongoDB.
A variety of master nodes are available in Cassandra.	MongoDB only employs one master node.
Cassandra offers the option to customise each query's consistency level.	Only eventual or strong consistency is supported by MongoDB.
It can be tailored to ACID attributes but does not support ACID transactions.	It offered snapshot isolation along with multi-document ACID transactions.

References-

1. Geeks for geeks
2. Data flair
3. Cassandra apache.org
4. Spice works
5. Data stax
6. Oracle