**Introduction:**

Apache Cassandra is a highly scalable, high-performance distributed database designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. It is a type of NoSQL database. Let us first understand what a NoSQL database does.

**What is NoSQL?**

A NoSQL database (Not Only SQL) is a database that provides a mechanism to store and retrieve data other than the tabular relations used in relational databases. These databases are schema-free, support easy replication, have simple API, eventually consistent, and can handle huge amounts of data.

The primary objective of a NoSQL database is to have:

* Simplicity of design
* Horizontal scaling is possible
* finer control over availability

**Types of NoSQL Databases:**

* Key-Value store
* Document Databases
* Column Databases
* Graph Databases

**NoSQL vs. Relational Databases:**

**NoSQL:**

* Supports very simple query language.
* No fixed schema.
* No “ACID”, it is only “eventually consistent”.
* Does not support transactions.

**Relational Databases:**

* Supports powerful query language.
* It has a fixed schema.
* Follows “ACID”.
* Supports transactions.

**NP:** *There is no winner in this comparison, it really depends on the kind of application you are building and the data you are storing.*

**What is Apache Cassandra?**

* Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for handling huge amounts of structured data spread out across the world. It provides highly available service with no single point of failure.
* Cassandra is a NoSQL database and can be described as a mix between a “*Key-Value Store”* and *“Column-Orientated”* database.

**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:**

* **Scalability** - New machines can easily be added with no downtime or interruption.
* **Decentralized** -No master & no single point of failure. Data is distributed across the cluster
* **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).
* **Cassandra Query Language (CQL)** – An SQL-like alternative

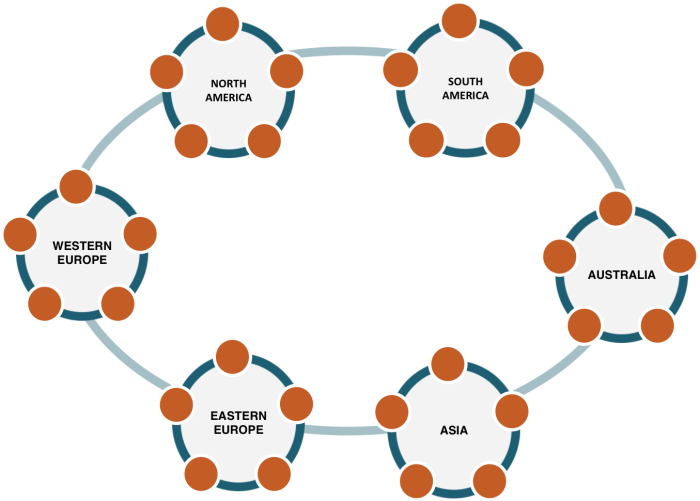
**Cassandra Query Language (CQL):**

* *CQL* is very similar to *SQL* in terms of syntax and commands*.*
* Statements directly change data and/or change the way data is stored.
* All statements end with semi-colon “**;**”.

SELECT \* FROM tablename;

**Cassandra Architecture:**

* Built with the understanding that hardware & software failures can happen.
* Per to Per Architecture.
* All the nodes are the same.
* Read/Write anywhere.
* Gossip protocol.
* Commit Log Captures All Activity.



**How data is stored?**

* **Keyspaces** – A Keyspace is similar to a **“Schema”**
* **Columns** – A “**Column Family”** is similar to a **“Table”** in a relational database except much more flexible
* Cassandra encourages **“Denormalization”** as oppose to relational **“Normalization”**
* **Columns** can be created on the fly.

**The Data Model:**

The data model of Cassandra is significantly different from what we normally see in an RDBMS. Let’s see how Cassandra stores its data.

**Cluster:**

Cassandra database is distributed over several machines that operate together. The outermost container is known as the Cluster. For failure handling, every node contains a replica, and in case of a failure, the replica takes charge. Cassandra arranges the nodes in a cluster, in a ring format, and assigns data to them.

**Keyspace:**

Keyspace is the outermost container for data in Cassandra. The basic attributes of a Keyspace in Cassandra are:

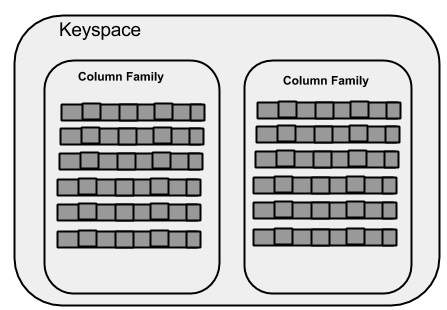
* **Replication factor:** It is the number of machines in the cluster that will receive copies of the same data.
* **Replica placement strategy:** It is nothing but the strategy to place replicas in the ring. We have strategies such as simple strategy (rack-aware strategy), old network topology strategy (rack-aware strategy), and network topology strategy (datacentre-shared strategy).
* **Column families:** Keyspace is a container for a list of one or more column families. A column family, in turn, is a container of a collection of rows. Each row contains ordered columns. Column families represent the structure of your data. Each keyspace has at least one and often many column families.

The syntax of creating a keyspace is as follows:

CREATE KEYSPACE Keyspace name

WITH replication = {'class': 'SimpleStrategy', 'replication factor': 3};

The following illustration shows a schematic view of a Keyspace.



**Column Family:**

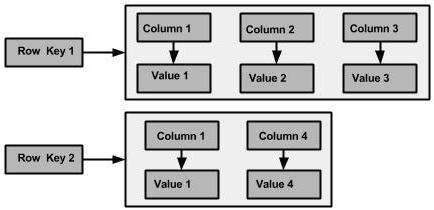
* A *“Column Family”* is similar to a *“Table”* in a RDBMS because it has columns and rows
* Relational database tables use a predefined, fixed schema. Column families do not which makes them very flexible

**A Cassandra column family has the following attributes:**

* **keys\_cached:** It represents the number of locations to keep cached per SSTable.
* **rows\_cached:** It represents the number of rows whose entire contents will be cached in memory.
* **preload\_row\_cache**: It specifies whether you want to pre-populate the row cache.

**Note:** Unlike relational tables where a column family’s schema is not fixed, Cassandra does not force individual rows to have all the columns.

The following figure shows an example of a Cassandra column family.



**Column:**

A column is the basic data structure of Cassandra with three values, namely key or column name, value, and a time stamp. Given below is the structure of a column.

**SuperColumn:**

A super column is a special column, therefore, it is also a key-value pair. But a super column stores a map of sub-columns.

Generally column families are stored on disk in individual files. Therefore, to optimize performance, it is important to keep columns that you are likely to query together in the same column family, and a super column can be helpful here.