NoSQL Data Models

Shahwar Saleem, July 19, 2021

1 Non-Relational Databases

Also Not Only SQL.

When NOT to use SQL:

- **Need high Availability in the data**: Indicates the system is always up and there is no downtime Have Large Amounts of Data
- **Need Linear Scalability**: The need to add more nodes to the system so performance will increase linearly
- **Low Latency**: Shorter delay before the data is transferred once the instruction for the transfer has been received.
- **Need fast reads and write**

1.0.1 Apache Cassandra

- Open Source NoSQL DB
- Masterless Architecture
- High Availability
- Linear Scalable
- Used by Uber, Netflix, Hulu, Twitter, Facebook etc
- Major contributors to project: Datastax, Facebook, Twitter, Apple.

1.0.2 Distributed Databases

- In a distributed database, in order to have high availability, you will **need copies of data**
- Scaled out horizontally, made up of multiple nodes.
- High availability i.e no down time. Nodes will fail, therefore, need copies of data.

Eventual Consistency

A consistency model used in distributed computing to achieve high availability that informally guarantees that, if no new updates are made to a given data item, eventuall all accesses to that item will return the last updated value.

1.0.3 CAP Theorem

It is impossible for a distributed data store to **simultaneously provide** more than two out of the following three guarantees: **CONSISTENCY, AVAILABILITY and PARTITION TOLERANCE**.

- **Consistency** Every read from the database gets the latest (and correct) piece of data or an error.
- **Availability** Every request is received and a response is given – without a guarantee that the data is the latest update.
- **Partition Tolerance** The system continues to work regardless of losing network connectivity between nodes.

Apache Cassandra

Apache Cassandra chooses to be highly available at the cost of consistency. It is an AP (Availability and Partition Tolerant) database. [Consistency issues only come up during network failures]

Which of these combinations is desirable for a production system - Consistency and Availability, Consistency and Partition Tolerance, or Availability and Partition Tolerance?

As the CAP Theorem Wikipedia entry says,

"The CAP theorem implies that in the presence of a network partition, one has to choose between consistency and availability." So there is no such thing as Consistency and Availability in a distributed database since it must always tolerate network issues. You can only have Consistency and Partition Tolerance (CP) or Availability and Partition Tolerance (AP). Remember, relational and non-relational databases do different things, and that's why most companies have both types of database systems.

1.0.4 Denormalization in Apache Cassandra

Denormalization of tables in Apache Cassandra is absolutely critical.

New data modeling tasks are required to translate Relational Model to Non-Relational Model.

Data Modeling in Apache Cassandra

- Denormalization is not just okay – it's a must
- Denormalization must be done for fast reads: Apache Cassandra is optimized for fast writes: To have fast reads, a good data model should be formed.
- Apache Cassandra has been optimized for fast writes: Updating multiple tables should not be a concern like it was in SQL DBs.
- ALWAYS think Queries first
- One table per query is a great strategy - Needs data redundancy. One query should access one table, no joins are allowed.
- Apache Cassandra does NOT allow for JOINs between tables

Cassanda Query Language

- CQL is the way to interact with database and is similar to SQL.
- JOINS, GROUPBY, or subqueries are not supported in CQL

1.0.5 Primary Key

Primary Key is how each row can be uniquely identified and how the data is distributed across the nodes (or servers) in our system.

- The first element of PRIMARY KEY is the PARTITION KEY (which will determine the distribution)
- PRIMARY KEY is made up of either just the PARTITION KEY or with the addition of CLUSTERING COLUMNS.
- PARTITION KEY will determine the distribution of data across the system
- The partition key's row value will be hashed (turned into a number) and stored on the node in the system that holds that range of values.

KEY POINTS:

- Primary Key must be UNIQUE. No Duplicates in Apache Cassandra
- Hashing of this value results in placement on a particular note in the system.
- Data distributed by this partition key. (Partition by state will see uneven partitions)
- Composite: More than 1 column.
- May have 1 or more clustering columns.

1.0.6 Clustering Columns

Primary key is made up of either just PARTITION KEY or with addition of CLUSTERING COLUMNS.

- CLUSTERING COLUMNS will determine the sort order within a partition.
- The CLUSTERING COLUMNS will sorte the data in sorted ascending order. e.g alphabetical order.
- More than one CLUSTERING COLUMN can be added.
- CLUSTERING COLUMNS will sort in order of how they are added to primary key. They must also be used IN ORDER in a SELECT statement.

1.0.7 WHERE Clause

Data Modeling in Apache Cassandra is query focused and that focus needs to be on the WHERE clause.

- **PARTITION KEY MUST be included in your query and any Clustering Columns can be used in the order they appear in your PRIMARY KEY**
- **Failure to include a WHERE clause will result in an ERROR**
- Recommended that one partition be queried at a time for performance implications.

SELECT * FROM Table is not recommended

Why do we need to use a WHERE statement since we are not concerned about analytics? Is it only for debugging purposes?

The WHERE statement is allowing us to do the fast reads. With Apache Cassandra, we are talking about big data – think terabytes of data – so we are making it fast for read purposes. Data is spread across all the nodes. By using the WHERE statement, we know which node to go to, from which node to get that data and serve it back. For example, imagine we have 10 years of data on 10 nodes or servers. So 1 year's data is on a separate node. By using the WHERE year = 1 statement we know which node to visit fast to pull the data from.