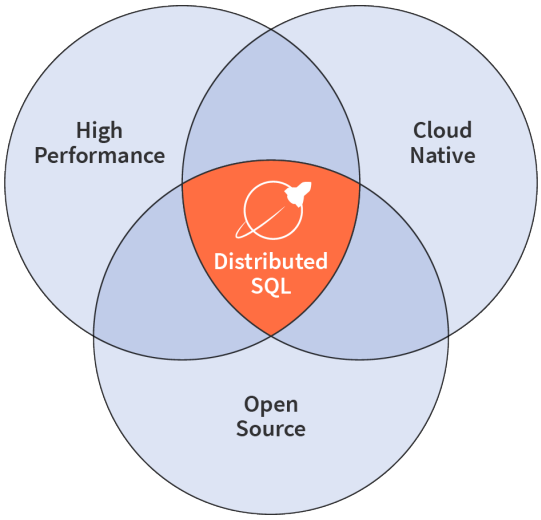
# YUGABYTEDB FUNDAMENTALS CERTIFICATION

## Exam Prep

yugabyteDB




## What is distributed SQL?

**At a minimum, a distributed SQL database should have the following characteristics:**

- A SQL API for accessing and manipulating data and objects
- Automatic distribution of data across nodes in a cluster
- Automatic replication of data in a strongly consistent manner
- Support for distributed query execution so clients do not need to know about the underlying distribution of data
- Support for distributed ACID transactions

*Further reading: **What is Distributed SQL?***

## What is YugabyteDB?

**YugabyteDB is a distributed SQL database with the following additional characteristics:**

- YugabyteDB's architecture and design is inspired by Google Spanner
- YugabyteDB's YSQL API is compatible with PostgreSQL 11.2
- YugabyteDB is a Consistent and Partition Tolerant (CP) database
- YugabyteDB's YSQL API supports serializable and snapshot (repeatable read) isolation levels
- YugabyteDB is completely open source, released under the Apache 2.0 license
- YugabyteDB supports advanced RDBMS features like triggers, stored procedures, foreign keys, and some PostgreSQL extensions
- In YugabyteDB all distributed transactions are guaranteed atomicity, consistency, isolation, and durability

*Further reading: **Design Goals***

> *In YugabyteDB, all distributed transactions are guaranteed atomicity, consistency, isolation, and durability (ACID).*

### What's the difference between monolithic, single-node databases like MySQL or PostgreSQL and distributed SQL databases like YugabyteDB?

By default, in a monolithic, single-node database, all writes must be served from a single node. In order to get more scale from databases like MySQL or PostgreSQL, you typically add more memory, CPU, and storage capacity. In other words, you scale up.

On the other hand, with a distributed, multi-node SQL database like YugabyteDB, writes can be serviced from any node. In order to get more scale from a distributed database, you typically add more nodes to the cluster. In other words, you scale out.

### What guarantees do distributed transactions have in YugabyteDB?

In YugabyteDB all distributed transactions are guaranteed atomicity, consistency, isolation, and durability (ACID).

*Further reading: DocDB Transactions Layer*

### What is the CAP theorem?

The CAP theorem states that in the face of network Partition, it's possible to solve for either Consistency or Availability, but not both. YugabyteDB can be classified as a CP database. This means it will become unavailable if it cannot maintain a consistent data set during a network partition.

*Further reading: CAP Theorem*

### What is Raft and how does YugabyteDB make use of it?

Raft is a consensus algorithm for distributed systems. YugabyteDB makes use of Raft to achieve consensus amongst leaders and followers in a "tablet peer" group. Recall that tablet peers are tablets that hold the same synchronously replicated data.

*Further reading: Raft Replication*

> *YugabyteDB supports synchronous replication between nodes in a cluster and asynchronous replication between clusters.*

## What are the main components of YugabyteDB?

### Universe

A group of nodes (VMs, physical machines, or containers) that collectively function as a clustered database.

*Further reading: **Universe***

### YB-TServer

YugabyteDB process responsible for hosting and serving data from a node.

*Further reading: **YB-TServer Service***

### YB-Master

This node process stores system metadata and records such as which tables, users, and permissions exist. It is also responsible for coordinating background operations (such as load-balancing or initiating re-replication of under-replicated data) and performing a variety of administrative operations such as creating, altering, and dropping tables.

*Further reading: **YB-Master Service***

## What forms of replication does YugabyteDB support?

YugabyteDB supports synchronous replication between nodes in a cluster and asynchronous replication between clusters.

*Further reading: **DocDB Replication Layer***

## What is DocDB?

DocDB is YugabyteDB's distributed document store responsible for transactions, sharding, replication, and persistence. Its design has roots in RocksDB.

*Further reading: **DocDB Storage Layer***


*Yugabyte Structured Query Language (YSQL) is a* ***fully-relational*** *SQL API*


*Yugabyte Cloud Query Language (YCQL) is a* ***semi-relational*** *SQL API*

### How does sharding and data distribution work in YugabyteDB?

- Database sharding is the horizontal partitioning of rows distributed across nodes
- Database sharding increases both read and write scalability of the database
- Rows are automatically sharded by primary key
- YugabyteDB supports the partitioning of data by either HASH or RANGE
- Tables are split into tablets
- Replicated tablets (tablets that contain identical data) form a Raft group with a leader and followers
- Replicated tablets always contain identical data because the data is synchronously replicated

*Further reading: **DocDB Sharding Layer***

### What data access APIs does YugabyteDB support?

**YugabyteDB supports two modes of data access:**

- **YSQL:** A PostgreSQL-compatible API best suited for relational workloads
- **YCQL:** A semi-relational API inspired by Cassandra, best suited for NoSQL workloads

*Further reading: **YSQL** **and** **YCQL***

### How do distributed SQL databases tolerate failures?

Distributed SQL databases are able to tolerate failures because no single node in the cluster holds all data. This means you can lose one or more nodes (depending on your replication factor) and still have data available to clients.

### What is a network partition and how does YugabyteDB handle them?

In a network partition, clients can connect to the database nodes, but some of the nodes cannot communicate with each other. A network partition can quickly introduce data inconsistencies as soon as the nodes are able to communicate with each other again. YugabyteDB handles a network partition by allowing shard followers to serve timeline-consistent reads and making them available for a leader election. Meanwhile, the shard leaders are able to accept writes and serve strong reads.

*Further reading: How YugabyteDB Handles Network Partitions*

> *Distributed SQL databases are able to tolerate failures because no single node in the cluster holds all data.*

### What formula helps us determine the replication factor (RF) required to achieve a fault tolerance (FT) of k nodes?

The formula is **RF = (2k + 1).**

For example, to achieve the fault tolerance of 1 node we'd need a replication factor of at least 3.

3 = (2 * 1 + 1)

In another example, to achieve the fault tolerance of 2 nodes we'd need a replication factor of at least 5.

5 = (2 * 2 + 1)

*Further reading: Replication Factor*

### Under what circumstances would an RF=3 YugabyteDB cluster always become unavailable?

Using the formula RF = (2k + 1), if an RF=3 cluster's node count drops to 1, the YugabyteDB cluster would become unavailable.

### What are secondary indexes and why are they advantageous to use in YugabyteDB?

Secondary indexes enable an additional mode of fast data access besides indexing by primary key. Additionally, they can be used to enforce the uniqueness of column values.

*Further reading: **Secondary Indexes***

### What are Colocated Tables in YugabyteDB?

Colocated tables have all of their data in a single tablet, but the tablet is replicated on multiple nodes.

*Further reading: **Colocated Tables***

### What is Change Data Capture (CDC)?

YugabyteDB's CDC ensures that any changes in data are identified, captured, and automatically applied to another datasource or made available for consumption by applications and other tools.

*Further reading: **Change Data Capture***

yugabyteDB

### Questions?

YugabyteDB's Community Slack is the easiest and fastest way to get your questions answered about requirements, installation, class content, exam preparation and certification. ***Register here:***

***yugabyte.com/slack***

...after introducing yourself on **#introductions,** please join the **#training** channel, this is where we discuss all things training and certification related.