YUGABYTEDB YSQL DEVELOPMENT FUNDAMENTALS CERTIFICATION

Exam Prep

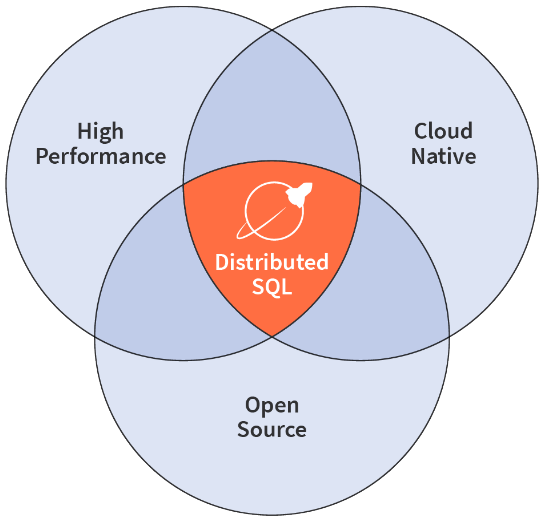
yugabyteDB

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 reuses PostgreSQL's query layer to deliver its YSQL API
- YugabyteDB's YSQL API is compatible with PostgreSQL 11.2
- YugabyteDB's YSQL API is best suited for relational workloads and 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
- YugabyteDB is a Consistent and Partition Tolerant (CP) database
- In YugabyteDB all distributed transactions are guaranteed atomicity, consistency, isolation, and durability (ACID)

Further reading: Design Goals

What are the main components of YugabyteDB?

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

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

What are schemas and tables?

- A YugabyteDB cluster contains one or more databases
- Users and groups of users are shared across the entire cluster, but no other data is shared across databases
- Any given client connection to the server can access only the data in a single database, the one specified in the connection request
- A database contains one or more named schemas, which in turn contain tables
- Schemas also contain other kinds of named objects, including data types, functions, and operators
- Unlike databases, schemas are not rigidly separated: users can access objects in any of the schemas in the database they are connected to, if they have privileges to do so

Working with schemas

- Use CREATE SCHEMA to create a new schema
- Use ALTER SCHEMA to change its properties
- Use DROP SCHEMA to remove it and add CASCADE to delete the dependent objects

“

A primary key is a column or a group of columns used to identify a row uniquely in a table.

Working with databases

- Use CREATE DATABASE to create a database
- Use ALTER DATABASE to change its properties
- Use DROP DATABASE to remove it
- These are commands that can get you details about databases
    - \l
    - \l+
    - SELECT datname FROM pg_database

What is a table?

- A table is a lot like a spreadsheet having rows and columns
- The number and order of the columns is fixed and each column has a name
- The number of rows in a table can vary depending on how much data is stored at a given moment
- When a table is read, the rows will appear in an unspecified order, unless sorting is explicitly requested

What is a primary key?

- A primary key is a column or a group of columns used to identify a row uniquely in a table
- You define primary keys through primary key constraints like NOT NULL and UNIQUE
- A table can only have one primary key

What is a data type?

- Each column has a “data type” that constrains the set of possible values that can be assigned to a column
- For example, data types might only permit certain numbers, characters, or dates to be stored

Working with tables

- Use CREATE TABLE to create a table
- Use \dt to view the tables in a database
- Use \d table_name or \d+table_name to view the tables in a database
- Use ALTER TABLE to change the properties of a table
- Combine ALTER TABLE with ADD COLUMN to add a column to a table
- Use CREATE TABLE AS to create a new table and populate it with the data returned from the query
- Use TRUNCATE TABLE to efficiently delete all the data in a table
- Use DELETE TABLE to delete a table
- Use CREATE TEMP TABLE to create a table that that exists for the duration of a database session

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

What data types does YugabyteDB support?

- YugabyteDB supports over 40 PostgreSQL data types including UUID, timestamps, and dates, serial, JSONB, plus various numeric and char types

Manipulating data in YugabyteDB

- To insert a single or multiple rows of data into an existing table, use INSERT INTO combined with VALUES
- To insert multiple rows of data into an existing table and have default values assigned to columns where a value was not specified use SET DEFAULT
- An INSERT INTO can be combined with a SELECT insert data into table #1, with data from table #2
- An INSERT statement can make use of the optional RETURNING clause, which returns information about the inserted row
- An UPDATE statement is used to manipulate data that has already been written to a table
- An UPDATE statement combined with SET allows us to update all the values in a table for the columns we specify
- Add a RETURNING clause to an UPDATE statement to get back the data that was updated
- To delete a specific row of data use DELETE with a WHERE clause

“

YugabyteDB reuses PostgreSQL’s query layer to deliver its YSQL API.

Querying data in YugabyteDB

- To return all the rows in a table use a SELECT statement with an asterisk and no WHERE clause
- To return the data from a single column specify the column name in the SELECT statement
- To return the data from multiple columns specify the column names in the SELECT statement
- An expression is a combination of one or more values, operators, and functions that evaluate a value. EXPRESSIONS can be of a boolean, numeric, or date types to aid in the querying for specific sets of data. For example, COUNT(*)
- A column or table alias allows you to assign the object a temporary name. Use AS to create the alias.
- To have a SELECT return data in ascending order use an ORDER BY paired with a ASC
- To have a SELECT return data in descending order use an ORDER BY paired with a DESC
- The DISTINCT clause is used in a SELECT statement to remove duplicate rows from a result set and keeps one row for each group of duplicates
- A WHERE clause when combined with an operator allows us to be very specific in defining just the data that we want returned
- You can pair =, AND, OR, IN, LIKE, BETWEEN and <> operators with WHERE clauses
- The LIMIT clause helps us “limit” the number of rows returned
- Use the IN operator in the WHERE clause to check if a value matches any value in a list of values. Use a NOT IN operator to do the opposite.
- The BETWEEN operator allows us to check if a value is within a given range. The NOT BETWEEN operator does the opposite
- The LIKE operator helps us query data using pattern matchings. The NOT LIKE operator does the opposite.
- The ILIKE operator works like the LIKE operator but adds the ability to match values case-insensitively
- The GROUP BY clause divides the rows returned from a SELECT statement into groups
- The HAVING clause allows us to specify a search condition for a group or an aggregate

- The UNION operator helps us to combine the result sets of multiple queries into a single result set
- The UNION operator removes all duplicate rows so to retain the duplicate rows, use UNION ALL
- Like the UNION and EXCEPT operators, the INTERSECT operator combines the result sets of two or more SELECT statements into a single result set
- The EXCEPT operator returns the rows in the first query that do not appear in the output of the second query
- A subquery is a query nested inside another query such as SELECT, INSERT, DELETE, and UPDATE
- The ANY operator allows us to compare a scalar value with a set of values returned by a subquery

Working with JOINS in YugabyteDB

- An INNER JOIN allows you to combine data from multiple tables and identify records that have matching values in both tables
- A FULL OUTER JOIN selects all the records that match either left or right table records
- A LEFT JOIN performs a join starting with the first (left-most) table. Then, any matched records from the second table (right-most) are included.
- A RIGHT JOIN performs a join starting with the second (right-most) table and then any matching first (left-most) table records
- A CROSS JOIN clause allows you to produce a Cartesian product by joining each row of the first table with all the rows of the second table
- A SELF JOIN allows you to join a table to itself. This is useful for querying hierarchical data or comparing rows within the same table.

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. *Join 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.