UNIT III NOSQL DATABASES 9

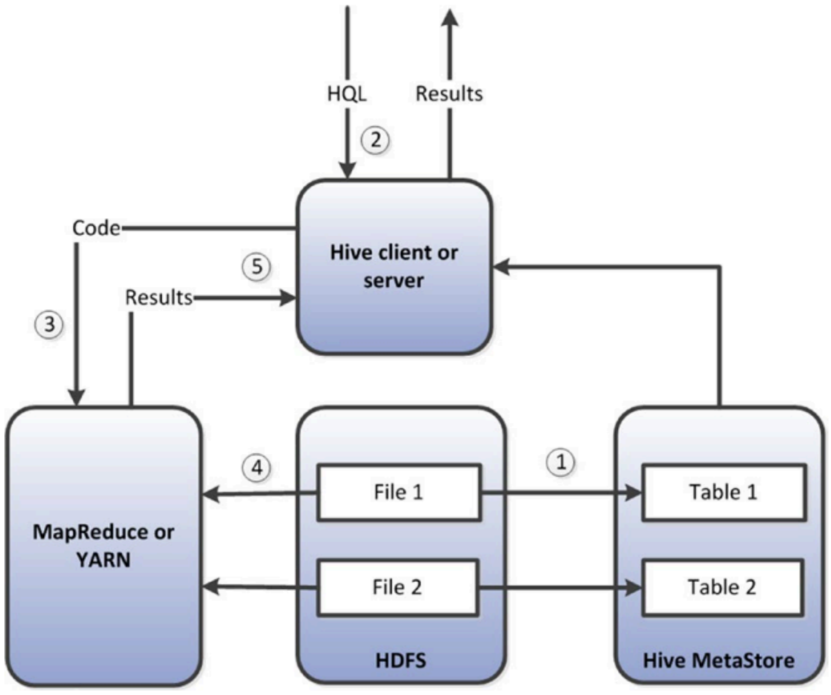
NoSQL – CAP Theorem – Sharding - Document based – MongoDB Operation: Insert, Update, Delete, Query, Indexing, Application, Replication, Sharding–Cassandra: Data Model, Key Space, Table Operations, CRUD Operations, CQL Types – HIVE: Data types, Database Operations, Partitioning – HiveQL – OrientDB Graph database – OrientDB Features

---

HIVE

- Hive is thought of as "SQL for Hadoop," although Hive provides a catalog for the Hadoop system, as well as a SQL processing layer.
- The Hive metadata service contains information about the structure of registered files in the HDFS file system.
- This metadata effectively "schematizes" these files, providing definitions of column names and data types.
- The Hive client or server (depending on the Hive configuration) accepts SQL-like commands called Hive Query Language (HQL).
- These commands are translated into Hadoop jobs that process the query and return the results to the user.
- Most of the time, Hive creates MapReduce programs that implement query operations such as joins, sorts, aggregation, and so on.
- Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.
- Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL-like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.
- Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

HIVE ARCHITECTURE



- The Hive metastore maps HDFS files to Hive tables
- A Hive client or server (depending on the installation mode) accepts HQL commands that perform SQL operations on those tables.
- Hive translates HQL to Hadoop code (3)—usually MapReduce.
- This code operates against the HDFS files (4) and returns query results to Hive (5).

Features of Hive

- Hive is fast and scalable.
- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.
- It supports user-defined functions (UDFs) where user can provide its functionality.

Data types

Hive data types are categorized in numeric types, string types, misc types, and complex types. A list of Hive data types is given below.

Integer:

Type	Size	Range
TINYINT	1-byte signed integer	-128 to 127
SMALLINT	2-byte signed integer	32,768 to 32,767
INT	4-byte signed integer	2,147,483,648 to 2,147,483,647
BIGINT	8-byte signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
FLOAT	4-byte	Single precision floating point number
DOUBLE	8-byte	Double precision floating point number

Date/Time Types

1. **TIMESTAMP**

- It supports traditional UNIX timestamp with optional nanosecond precision.
- As Integer numeric type, it is interpreted as UNIX timestamp in seconds.
- As Floating point numeric type, it is interpreted as UNIX timestamp in seconds with decimal precision.
- As string, it follows java.sql.Timestamp format "YYYY-MM-DD HH:MM:SS.fffffffff" (9 decimal place precision)

2. **DATES**

- The Date value is used to specify a particular year, month and day, in the form YYYY--MM--DD. However, it didn't provide the time of the day. The range of Date type lies between 0000--01--01 to 9999--12--31.

String Types

1. **STRING**

The string is a sequence of characters. It values can be enclosed within single quotes (') or double quotes (").

2. **Varchar**

The varchar is a variable length type whose range lies between 1 and 65535, which specifies that the maximum number of characters allowed in the character string.

3. **CHAR**

The char is a fixed-length type whose maximum length is fixed at 255.

<u>Complex Type</u>

Type	Size	Range
Struct	It is similar to C struct or an object where fields are accessed using the "dot" notation.	struct('James','Roy')
Map	It contains the key-value tuples where the fields are accessed using array notation.	map('first','James','last','Roy')
Array	It is a collection of similar type of values that indexable using zero-based integers.	array('James','Roy')

Database Operations,

Hive - Create Database

In Hive, the database is considered as a catalog or namespace of tables. So, we can maintain multiple tables within a database where a unique name is assigned to each table. Hive also provides a default database with a name default.

- Initially, we check the default database provided by Hive. So, to check the list of existing databases, follow the below command: -

```
hive> create database demo
hive> show databases;
```

Hive - Drop Database

In this section, we will see various ways to drop the existing database. drop the database by using the following command.

```
hive> drop database demo;
```

Hive - Create Table

In Hive, we can create a table by using the conventions similar to the SQL. It supports a wide range of flexibility where the data files for tables are stored. It provides two types of table:

- Internal table
- External table

Internal Table

The internal tables are also called managed tables as the lifecycle of their data is controlled by the Hive. By default, these tables are stored in a subdirectory under the directory defined by hive. metastore. warehouse.dir (i.e. /user/hive/warehouse). The internal tables are not flexible enough to share with other tools like Pig. If we try to drop the internal table, Hive deletes both table schema and data. Let's create an internal table by using the following command:-

```
hive> create table demo.employee (Id int, Name string , Salary float)
row format delimited
fields terminated by ',' ;
```

Let's see the metadata of the created table by using the following command:-

```
hive> describe demo.employee
```

External Table

The external table allows us to create and access a table and a data externally. The external keyword is used to specify the external table, whereas the location keyword is used to determine the location of loaded data. As the table is external, the data is not present in the Hive directory. Therefore, if we try to drop the table, the metadata of the table will be deleted, but the data still exists. Let's create an external table using the following command: -

```
hive> create external table emplist (Id int, Name string , Salary float)
row format delimited
fields terminated by ','
location '/HiveDirectory';
```

we can use the following command to retrieve the data: -

```
select * from emplist;
```

Hive - Load Data

Once the internal table has been created, the next step is to load the data into it. So, in Hive, we can easily load data from any file to the database.

Let's load the data of the file into the database by using the following command: -

```
load data local inpath '/home/codegyani/hive/emp_details' into table demo.employee;
```

Hive - Drop Table

Hive facilitates us to drop a table by using the SQL drop table command. Let's follow the below steps to drop the table from the database. Let's check the list of existing databases by using the following command: -

```
hive> show databases;
hive> use demo;
hive> show tables;
hive> drop table new_employee;
```

Hive - Alter Table

In Hive, we can perform modifications in the existing table like changing the table name, column name, comments, and table properties. It provides SQL like commands to alter the table.

Rename a Table

If we want to change the name of an existing table, we can rename that table by using the following signature: -

```
Alter table old_table_name rename to new_table_name;
```

Now, change the name of the table by using the following command: -

```
Alter table emp rename to employee_data;
```

Adding column

In Hive, we can add one or more columns in an existing table by using the following signature:

```
Alter table table_name add columns(column_name datatype);
```

Now, add a new column to the table by using the following command: -

```
Alter table employee_data add columns (age int);
```

Change Column

In Hive, we can rename a column, change its type and position. Here, we are changing the name of the column by using the following signature: -

```
Alter table table_name change old_column_name new_column_name datatype;
```

Now, change the name of the column by using the following command: -

```
Alter table employee_data change name first_name string;
```

Delete or Replace Column

Hive allows us to delete one or more columns by replacing them with the new columns. Thus, we cannot drop the column directly. Let's see the existing schema of the table.

alter table employee_data replace columns(id string, first_name string, age int);

Partitioning

The partitioning in Hive means dividing the table into some parts based on the values of a particular column like date, course, city or country. The advantage of partitioning is that since the data is stored in slices, the query response time becomes faster.

The partitioning in Hive can be executed in two ways –

- Static partitioning
- Dynamic partitioning

Static Partitioning

In static or manual partitioning, it is required to pass the values of partitioned columns manually while loading the data into the table. Hence, the data file doesn't contain the partitioned columns.

Example of Static Partitioning

Select the database in which we want to create a table.

```
hive> use test;
```

Create the table and provide the partitioned columns by using the following command: -

```
hive> create table student (id int, name string, age int, institute string)
partitioned by (course string)
row format delimited
fields terminated by ',';

hive> describe student;
```

Load the data into the table and pass the values of partition columns with it by using the following command: -

```
hive> load data local inpath '/home/codegyani/hive/student_details1' into table student partition(course= "java");
```

Here, we are partitioning the students of an institute based on courses.
Load the data of another file into the same table and pass the values of partition columns with it by using the following command: -

```
hive> load data local inpath '/home/codegyani/hive/student_details2' into table student partition(course= "hadoop");
```

```
hive> select * from student;
```

Retrieve the data based on partitioned columns by using the following command: -

```
hive> select * from student where course="java";
```

Dynamic Partitioning

In dynamic partitioning, the values of partitioned columns exist within the table. So, it is not required to pass the values of partitioned columns manually.
First, select the database in which we want to create a table.

```
hive> use show;
```

Enable the dynamic partition by using the following commands: -

```
hive> set hive.exec.dynamic.partition=true;
hive> set hive.exec.dynamic.partition.mode=nonstrict;
```

Create a dummy table to store the data.

```
hive> create table stud_demo(id int, name string, age int, institute string, course string) row format delimited fields terminated by ',';
```

load the data into the table.

```
hive> load data local inpath '/home/codegyani/hive/student_details' into table stud_demo;
```

Create a partition table by using the following command: -

```
hive> create table student_part (id int, name string, age int, institute string)
partitioned by (course string) row format delimited fields terminated by ',';
```

Insert the data of dummy table into the partition table.

```
hive> insert into student_part
partition(course)
select id, name, age, institute, course
from stud_demo;
```

HiveQL

Hive is the original SQL on Hadoop. From the very early days of Hadoop, Hive represented the most accessible face of Hadoop for many users. Hive Query Language (HQL) is a SQL-based language that comes close to SQL-92 entry-level compliance, particularly within its SELECT statement. DML statements—such as INSERT, DELETE, and UPDATE—are supported in recent versions, though the real purpose of Hive is to provide query access to Hadoop data usually ingested via other means. Some SQL-2003 analytic window functions are also supported. HQL is compiled to MapReduce or—in later releases—more sophisticated **YARN-based DAG algorithms**.

The following is a simple Hive query:

```
0: jdbc:Hive2://> SELECT country_name, COUNT (cust_id)
0: jdbc:Hive2://> FROM countries co JOIN customers cu
0: jdbc:Hive2://> ON(cu.country_id=co.country_id)
0: jdbc:Hive2://> WHERE region = 'Asia'
0: jdbc:Hive2://> GROUP BY country_name
0: jdbc:Hive2://> HAVING COUNT (cust_id) > 500;
2015-10-10 11:38:55 Starting to launch local task to process map join; maximum
memory = 932184064
<<Bunch of Hadoop JobTracker output deleted>>
2015-10-10 11:39:05,928 Stage-2 map = 0%, reduce = 0%
2015-10-10 11:39:12,246 Stage-2 map = 100%, reduce = 0%, Cumulative CPU 2.28 sec
2015-10-10 11:39:20,582 Stage-2 map = 100%, reduce = 100%, Cumulative CPU 4.4 sec
+---------------+------+--+
| country_name | _c1 |
+---------------+------+--+
| China | 712 |
| Japan | 624 |
| Singapore | 597 |
+---------------+------+--+
3 rows selected (29.014 seconds)
```

HQL statements look and operate like SQL statements. There are a few notable differences between HQL and commonly used standard SQL, however:

- HQL supports a number of table generating functions which can be used to return multiple rows from an embedded field that may contain an array of values or a map of name:value pairs. The **Explode()** function returns one row for each element in an array or map, while json_tuple() explodes an embedded JSON document.

- Hive provides a SORT BY clause that requests output be sorted only within each reducer within the MapReduce pipeline. Compared to ORDER BY, this avoids a large sort in the final reducer stage, but may not return results in sorted order.
- DISTRIBUTE BY controls how mappers distribute output to reducers. Rather than distributing values to reducers based on hashing of key values, we can insist that each reducer receive contiguous ranges of a specific column. DISTRIBUTE BY can be used in conjunction with SORT BY to achieve an overall ordering of results without requiring an expensive final sort operation. CLUSTER BY combines the semantics of DISTRIBUTE BY and SORT BY operations that specify the same column list. Hive can query data in HBase tables and data held in HDFS.