**Session 7: HIVE Operations**

Assignment 7.3

Student Name: Abarajithan SA

Course: Big Data Hadoop & Spark Training

Start Date:  2017-09-09

End Date:  2017-11-26

**Assignment 7.3**– Explain the concepts of Hive Data Definitions, Hive Data Manipulations and HiveQL Manipulations with examples.

Contents

[Introduction 2](#_Toc499068479)

[Problem Statement 2](#_Toc499068480)

[Hive Data Definitions 2](#_Toc499068481)

[Hive Data Manipulations 3](#_Toc499068482)

[HIVEQL manipulation 4](#_Toc499068483)

[Built-in operators 4](#_Toc499068484)

# Introduction

In this assignment we are going to see about the concepts of Hive Data Definitions, Hive Data Manipulations and HiveQL Manipulations.

# Problem Statement

Explain the below concepts with an example in brief.

* Hive Data Definitions
* Hive Data Manipulations
* HiveQL Manipulations

# Hive Data Definitions

**Hive Data Definition Language**. Hive Data Definition Language (DDL) is a subset of Hive SQL statements that describe the data structure in Hive by creating, deleting, or altering schema objects such as databases, tables, views, partitions, and buckets.

The Driver for Apache Hive supports a broad set of DDL, including (but not limited to) the following:

* **CREATE** Database and **DROP** Database
* **CREATE** Table and **DROP** Table
* **ALTER** Table and **ALTER** Partition statements
* **CREATE** View and Drop View
* **CREATE** Function and **Drop** Function

Example:

**CREATE DATABASE acadgild\_db;**

Hive will throw an error if acadgild\_db already exists.

You can suppress these warnings with this variation:

**CREATE DATABASE IF NOT EXISTS acadgild\_db;**

Tables in that database will be stored in subdirectories of the database directory. The exception is tables in the default database, which doesn’t have its own directory. The database directory is created under a top-level directory specified by the property

hive.metastore.warehouse.dir

set hive.metastore.warehouse.dir;

You can override this default location for the new directory as shown:

CREATE DATABASE acadgild\_db

LOCATION '/user/acadgild/mydb';

# Hive Data Manipulations

The ability to manipulate data is a critical capability in big data analysis. Manipulating data is the process of **exchanging**, **moving**, **sorting** and **transforming** the data. This technique is used in many situations, such as cleaning data, searching patterns, creating trends and so on. Hive offers various query statements, keywords, operators and functions to carry out data manipulation.

There are multiple ways to modify data in Hive:

1. LOAD
2. INSERT

* into Hive tables from queries
* into directories from queries
* into Hive tables from SQL

1. UPDATE
2. DELETE
3. MERGE
4. EXPORT and IMPORT commands are also available (as of Hive 0.8).

Example:

1. ***CREATE TABLE students (name VARCHAR(64), age INT, gpa DECIMAL(3, 2))***

***CLUSTERED BY (age) INTO 2 BUCKETS STORED AS ORC;***

***INSERT INTO TABLE students***

***VALUES ('fred flintstone', 35, 1.28), ('barney rubble', 32, 2.32);***

1. ***MERGE INTO <target table> AS T USING <source expression/table> AS S***

***ON <boolean expression1>***

***WHEN MATCHED [AND <boolean expression2>] THEN UPDATE SET <set clause list>***

***WHEN MATCHED [AND <boolean expression3>] THEN DELETE***

***WHEN NOT MATCHED [AND <boolean expression4>] THEN INSERT VALUES<value list>***

# HIVEQL manipulation

Hive provides a CLI to write Hive queries using Hive Query Language (HiveQL). Generally HQL syntax is similar to the SQL syntax that most data analysts are familiar with. Hive's SQL-inspired language separates the user from the complexity of Map Reduce programming. It reuses familiar concepts from the relational database world, such as tables, rows, columns and schema, to ease learning.

Most interactions tend to take place over a command line interface (CLI). Hive provides a CLI to write Hive queries using Hive Query Language (Hive-QL).

Generally, HiveQL syntax is similar to the SQL syntax that most data analysts are familiar with. Hive supports four file formats those are TEXTFILE, SEQUENCEFILE, ORC and RCFILE (Record Columnar File).

* For single user metadata storage Hive uses derby database and
* For multiple user Metadata or shared Metadata case Hive uses MYSQL

## Built-in operators

Hive provides Built-in operators for Data operations to be implemented on the tables present inside Hive warehouse.

These operators are used for mathematical operations on operands, and it will return specific value as per the logic applied.

Types of Built-in Operators in HIVE are:

* Relational Operators
* Arithmetic Operators
* Logical Operators
* Operators on Complex types
* Complex type Constructors

**Relational Operators:**

We use Relational operators for relationship comparisons between two operands. Operators such as **equals**, **Not equals**, **less than**, **greater than** …etc. The operand types are all number types in these Operators.

**Arithmetic Operators:**

We use Arithmetic operators for performing arithmetic operations on operands. Arithmetic operations such as addition, subtraction, multiplication and division between operands we use these Operators.

**Logical Operators:**

We use Logical operators for performing Logical operations on operands. Logical operations such as **AND**, **OR**, **NOT** between operands we use these Operators. The operand types all are BOOLEAN type in these Operators

**Operators on Complex types:**

These are operators which will provide a different mechanism to access elements in complex types.

Example,

Operators Operands Description

**A[n]** A is an Array and n is an integer type it will return nth element in the array A.

**M[key]** M is a Map<K, V> and key has type K It will return the values belongs to the key in the map

**Complex type Constructors:**

It will construct instances on complex data types. These are of complex data types such as **Array**, **Map** and **Struct** types in Hive.

|  |  |  |
| --- | --- | --- |
| Operators | Operands | Description |
| array | (val1, val2, ...) | It will create an array with the given elements as mentioned like val1, val2 |
| Create\_ union | (tag, val1, val2, ...) | It will create a union type with the values that is being mentioned to by the tag parameter |
| map | (key1, value1, key2, value2, ...) | It will create a map with the given key/value pairs mentioned in operands |
| Named\_struct | (name1, val1, name2, val2, ...) | It will create a Struct with the given field names and values mentioned in operands |
| STRUCT | (val1, val2, val3, ...) | Creates a Struct with the given field values. Struct field names will be col1, col2, . |