Lesson 6

Hive Query Language (HiveQL)

HiveQL

- HiveQL is similar to SQL for querying on schema information on the metastore

Processing Engine

- One of the replacements of approach for MapReduce program
- Write an HQL query for MapReduce job and process it instead of writing MapReduce program in Java

Execution Engine

- Bridge between HiveQL process Engine and MapReduce
- Hive Execution engine processes the query and generates results same as MapReduce results
- Uses the flavour of MapReduce

HiveQL

- SQL to extract information from a data warehouse
- HiveQL supports large base of SQL users
- Queries the large datasets which reside in HDFS environment
- HDFS is scalable and Big Data parallel processing environment

HiveQL

HiveQL script commands enable:

1. data definition,
2. data manipulation and
3. query processing

- Data definitions and manipulations creates tables and files

HiveQL Data Definition Language (DDL)

- Database commands for data definition for the DBs and Tables

Creation of Table

- CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [<database name>.] <table name>
- [(<column name><data type> [COMMENT <column comment>], ...)]
- [COMMENT <table comment>]
- [ROW FORMAT <row format>]
- [STORED AS <file format>]

Commands

- CREATE DATABASE, SHOW DATABASE (list of all DBs), CREATE SCHEMA, CREATE TABLE
- for data definition for the DBs and Tables

Example

- Read Example 4.7 for creation of database named toys_companyDB and table named toys_tbl
- Example 4.8 for creation of table toy_products with following fields
- Example 4.9 for CREATE, SHOW and DELETE commands

HiveQL Data Manipulation Language (DML) Commands

- USE <database name>, DROP DATABASE, DROP SCHEMA, ALTER TABLE, DROP TABLE, and LOAD DATA (inserting the data)

Load Command

- LOAD DATA [LOCAL] INPATH '<file path>' [OVERWRITE] INTO TABLE <table name> [PARTITION (partcol1= val1, partcol2= val2 ...)]
- partcol1 refers to column 1, val1 to value

Example

- Example 4.10 for a toy company selling Jigsaws.
- Consider jigsaw_puzzle_info.txt in */home/user* directory
- Creates 4 fields in File : Toy-category, toy-id, toy-name, and Price in US$

HiveQL Commands for Querying

- SELECT [ALL | DISTINCT] <select expression>, <select expression>, ...
- FROM <table name>
- [WHERE <where condition>]
- [GROUP BY <column List>]
- [HAVING <having condition>]
- [CLUSTER BY <column List>| [DISTRIBUTE BY <column List>] [SORT BY <column List>]]
- [LIMIT number];

Partitioning

- Organizes tables into partitions
- Divides the table data into select parts, based on the values of particular set of columns, and create partitions

Partitioning

- Partition makes querying easy and fast
- SELECT actions are then from a smaller number of column-fields
- Recall RC columnar format and serialized records (Section 3.3.3.3)

Concept of partitioning, columnar and file records formats

- Example 4.11 considers a table **T** with eight-columns and four-rows
- Shows how to partition the table, convert in RC columnar format, and serialize

Table partitioning command

- CREATE [EXTERNAL] TABLE <table name> (<column name 1><data type 1>, …….)
- PARTITIONED BY (<column name n><data type n> [COMMENT <column comment>], ...);

Advantages of Partition

- Distribution of execution load horizontally,
- Query response time becoming faster when processing small part of the data instead of searching the entire data set

Examples

- Example 4.12 for how to add, rename, and drop a partition to a table, toys_tbl
- Example 4.13 for the faster selecting a product of specific category using the table during a query- processing action when the table has partitions which are based on category

Limitations

- Creating large number of partitions in table leading to large number of files and directories in HDFS
- Thus create overhead to NameNode since it must keep all metadata for the file system in memory only
-

… Limitations

- Partitions may optimize some queries based on Where clauses, but they may be less responsive for other important queries on grouping clauses

… Limitations

- Large number of partitions leading to large number of tasks (each of which will run in separate JVM) in each MapReduce job
- Lot of overhead in maintaining JVM start up and tear down.
- Larger processing time

Bucketing

- Provide an extra structure to the data that can lead to more efficient query processing
- Bucket is a set of records in column with very large number of fields, put together that stores as a file in the partition directory

Bucketing

- Records with the same bucketed column will always be stored in the same bucket.

Bucket-Records

- Records kept in each bucket provide sorting ease
- Enable Joins at a Map task
- Bucket also usable as sample data set

‘CLUSTERED BY’ Clause

- Divides a table into the buckets
- Example 4.14 how the bucketing enforced
- How a bucketed table partition of toy_airplane_10725 create five buckets
- .

… Example

- How a bucketed column load into toy_tbl?
- How to display bucket data display?

‘CLUSTERED BY’ Clause

- Enables a cluster of records into the bucket

‘View’ in HiveQL

- A query selects from a database
- Database consists of tables
- A query selects from a logic construct
- A complex query selects from number of logical constructs
- Each logical construct is a view of the database from a set of logic A

‘View’ in HiveQL

- A query selects from a database
- Database consists of tables
- A query selects using a logic construct
- The construct is set of logic operations using the data, selected columns and fields, to view just that part of the selection

HiveQL Complex Query

- A complex query selects from number of logical constructs
- Just like viewing the database (set of tables) from an angle (a set of logical operations)
- Each logic construct is a ‘View’ in HiveQL

View Characteristic

- Saves the query and reduce the query complexity,
- Use a View like a table but a View does not store data like a table,

Reference to a View

- Hive query statement when uses references to a view, the Hive executes the View

HiveQL query planner

- Plans to break a query into sub-queries for obtaining the right answer
- Hides the complexity by dividing the query into smaller, more manageable pieces

HiveQL query planner

- Planner combines the information in View definition with the remaining actions on the query

View

- Hides the complexity by dividing the query into smaller, more manageable pieces

Example

- Example 4.15, query with nested sub-queries to a table *toy_tbl*
- The table contains many values for categories of toys
- Example considers a table for Toy_Airplane of product code 10725

Aggregation Function

- Refer Table 4.11 for the Aggregation functions, their return type, syntax and descriptions

JOIN Function

- Refer Section 4.5.5 for the JOIN functions, and
- Examples, their return type, syntax and descriptions

‘GROUP BY’, ‘HAVING’, ‘ORDER BY’ and ‘DISTRIBUTE BY’

- SELECT [ALL | DISTINCT] <select expression>, <select expression>, ...
- FROM <table name>
- [WHERE <where condition>]
- [GROUP BY<column List>]
- [HAVING <having condition>]
- [CLUSTER BY <column List>| [DISTRIBUTE BY <column List>] [SORT BY <column List>]]
- [LIMIT number];

Refer Example 4.16

- SELECT * FROM toy WHERE ProductPrice > 1.5;
- SELECT ProductCategory, count(*) FROM toy_tbl GROUP BY ProductCategory;
- SELECT ProductCategory, sum (ProductPrice) FROM toy_tbl GROUP BY ProductCategory;

Summary

We learnt :

- For querying the large datasets which reside in HDFS environment
- HiveQL script commands enable data definition, data manipulation and query processing
- Processing and Execution Engines

Summary

We learnt:

- Complex Query
- Query Planner

Summary

We learnt uses of:

- Partitioning
- Bucketing
- JOIN
- Cluster By
- Having View
- Group By

End of Lesson 6 on Hive Query Language (HiveQL)