Hive Query Language

Hive Query Language (HQL)

1. **Create and manage tables and partitions.**
2. **Support various Relational, Arithmetic, and Logical Operators.**
3. **Evaluate functions.**
4. **Download the contents of a table to a local directory or result of queries to HDFS directory.**

DDL and DML statements

Database

To create a database named “STOCKS” with comments and database properties.

CREATE DATABASE IF NOT EXISTS STOCKS COMMENT 'STOCK Details' WITH DBPROPERTIES ('creator' = 'JOHN');

Database

To display list of databases

SHOW DATABASES;

To describe a database.

DESCRIBE DATABASE STOCKS;

To describe extended database.

DESCRIBE DATABASE EXTENDED STOCKS;

Database

To drop database.

DROP DATABASE STOCKS;

To alter the database properties

ALTER DATABASE STOCKS SET DBPROPERTIES('edited-by'='JAMES')

To make database as the current working database

USE STOCKS

Tables

Hive provides two kinds of tables:

- Managed Table
  - Hive Stores the table under the warehouse folder under Hive
  - Complete life cycle of table is managed by Hive
  - When internal table is dropped, it drops the data as well as the metadata
- External Table
  - When table is dropped, it retains the data in the underlying location. But drops metadata

Tables

To create managed table named ‘STOCK’.

CREATE TABLE IF NOT EXISTS STOCK(symbol STRING, reported STRING, price FLOAT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

To create external table named ‘**EXT_STOCK**’.

CREATE EXTERNAL TABLE IF NOT EXISTS EXT_STOCK(symbol STRING, reported TIMESTAMP, price FLOAT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/user/hive/warehouse/stock';

Tables

To load data into the table from file named msft-s.csv. Copy msft-s.csv into Downloads folder.

LOAD DATA LOCAL INPATH '/home/cloudera/Downloads/msft-s.csv' OVERWRITE INTO TABLE EXT_STOCK;

To retrieve the student details from “EXT_STOCK” table.

SELECT * from EXT_STOCK;

Partitions

Partitions

Partitions split the larger dataset into more meaningful chunks.

Hive provides two kinds of partitions: Static Partition and Dynamic Partition.

Use static partition if the value is known at compile time

- To create static partition based on “price” column.

CREATE TABLE IF NOT EXISTS STATIC_PART_STOCK (symbol STRING, reported TIMESTAMP) PARTITIONED BY (price FLOAT) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

- Load data into partition table from table.

INSERT OVERWRITE TABLE STATIC_PART_STOCK PARTITION (price=27.0) SELECT symbol, reported from EXT_STOCK where price=27.0;

Partitions

- To create dynamic partition on column date.

CREATE TABLE IF NOT EXISTS DYNAMIC_PART_STOCK(symbol STRING, reported TIMESTAMP, price FLOAT) PARTITIONED BY (repyear TIMESTAMP) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

- To load data into a dynamic partition table from table.

SET hive.exec.dynamic.partition = true;

Note: The dynamic partition strict mode requires at least one static partition column. To turn this off,
set hive.exec.dynamic.partition.mode=nonstrict

INSERT OVERWRITE TABLE DYNAMIC_PART_STOCK PARTITION (repyear)
SELECT symbol,reported, price, YEAR(reported) AS repyear from EXT_STOCK;

Bucketing In Hive

- Partitioning based on geographic locations like country, some bigger countries will have large partitions (ex: 4-5 countries itself contributing 70-80% of total data) where as small countries data will create small partitions (remaining all countries in the world may contribute to just 20-30 % of total data).
- To overcome the problem of over partitioning, Hive provides Bucketing concept.
- Features
  - Bucketing concept is based on **(hashing function on the bucketed column) mod (by total number of buckets)**. The hash_function depends on the type of the bucketing column.
  - Records with the same bucketed column will always be stored in the same bucket.
  - We use **CLUSTERED BY** clause to divide the table into buckets.
  - Physically, each bucket is just a file in the table directory, and Bucket numbering is 1-based.
  - Bucketed tables will create almost equally distributed data file parts.

Advantages of Bucketing

- Bucketed tables offer efficient sampling. With sampling, we can try out queries on a fraction of data for testing and debugging purpose when the original data sets are very huge.
- As the data files are equal sized parts, map-side joins will be faster on bucketed tables than non-bucketed tables. In Map-side join, a mapper processing a bucket of the left table knows that the matching rows in the right table will be in its corresponding bucket, so it only retrieves that bucket (which is a small fraction of all the data stored in the right table).
- Similar to partitioning, bucketed tables provide faster query responses than non-bucketed tables.
- Bucketing concept also provides the flexibility to keep the records in each bucket to be sorted by one or more columns. This makes map-side joins even more efficient, since the join of each bucket becomes an efficient merge-sort.

Buckets

set hive.enforce.bucketing=true;

- To create a bucketed table having 3 buckets.

```
CREATE TABLE IF NOT EXISTS STOCK_BUCKET(symbol STRING,
reported TIMESTAMP, price FLOAT)
CLUSTERED BY (price) into 3 buckets;
```

- Load data to bucketed table.

```
FROM EXT_STOCK
INSERT OVERWRITE TABLE STOCK_BUCKET
SELECT symbol, reported, price;
```

- To display the content of first bucket.

```
SELECT DISTINCT price FROM STOCK_BUCKET
TABLESAMPLE(BUCKET 1 OUT OF 3 ON price);
```

Aggregations

Hive supports aggregation functions like avg, count, etc.

To write the average and count aggregation function.

SELECT avg(price) FROM EXT_STOCK;

SELECT count(*) FROM EXT_STOCK;

Group by and Having

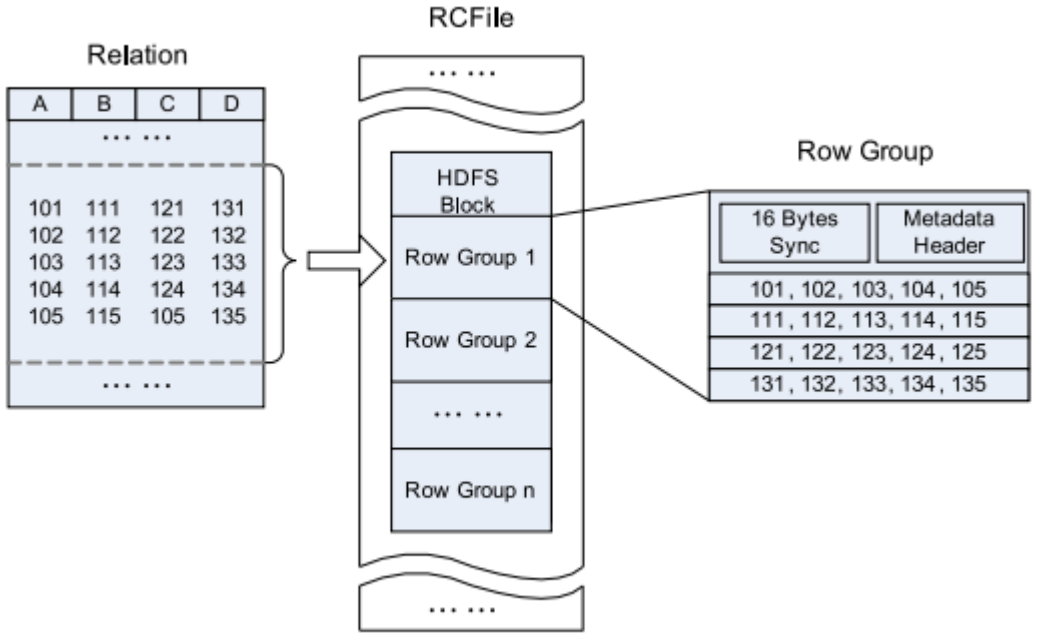
To write group by and having function.

```
SELECT symbol, reported,price
FROM EXT_STOCK
GROUP BY symbol, reported,price
HAVING price> 27.0;
```

Hive File Format

- **Text File -** The default file format is text file.
- **Sequential File - S**equential files are flat files that store binary key-value pairs.
- **RCFile (Record Columnar File) -** RCFile stores the data in **Column Oriented Manner** which ensures that **Aggregation** operation is not an expensive operation. RCFile first partitions horizontally and then vertically to serialize data.

  CREATE TABLE STOCK_RC (symbol STRING, reported TIMESTAMP, price FLOAT) STORED AS RCFILE;
  INSERT OVERWRITE TABLE STOCK_RC SELECT symbol, reported, price FROM STOCK;
  SELECT SUM(price) FROM STOCK_RC;



Complex Types

```
CREATE TABLE complex (
    c1 ARRAY<INT>,
    c2 MAP<STRING, INT>,
    c3 STRUCT<a:STRING, b:INT, c:DOUBLE>);
```

SELECT c1[0], c2['b'], c3.c FROM complex;

SerDer

- SerDer stands for Serializer/Deserializer.
- Contains the logic to convert unstructured data into records.
- Implemented using Java.
- Serializers are used at the time of writing. Serializer takes java object and translates it into something Hive can write to HDFS
- Deserializers are used at query time (SELECT Statement). Deserializer converts binary representation or a string of record to java object that Hive can manipulate

```
CREATE TABLE stations (usaf STRING, wban STRING, name STRING) ROW FORMAT
SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe' WITH
SERDEPROPERTIES ("input.regex" = "(\\d{6}) (\\d{5}) (.{29}) .*");
```

Manipulate XML Data

```
CREATE TABLE XMLSAMPLE(xmldata string);

LOAD DATA LOCAL INPATH '/home/cloudera/Downloads/employee.xml' INTO TABLE
XMLSAMPLE

CREATE TABLE xpath_table AS
SELECT xpath_int(xmldata, 'employee/empid'),
Xpath_string(xmldata,'employee/name')
FROM xmlsample;

SELECT * FROM xpath_table;
```

Further Readings

- http://en.wikipedia.org/wiki/RCFile
- https://cwiki.apache.org/confluence/display/Hive/DynamicPartitions
- https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL
- https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DML