**Hive Data Partitioning Example**

Now let’s understand data partitioning in Hive with an example. Consider a table named **Tab1**. The table contains client detail: id, name, dept, and yoj(year of joining). Suppose we need to retrieve the details of all the clients who joined in 2012. Then, the query searches the whole table for the required information. But if we partition the client data with the year and store it in a separate file, this will reduce the query processing time. The below example will help us to learn how to partition a file and its data-

Note: Data is independent of schema in Hive. If we change the partition location of a hive table using ALTER TABLE option then the data for that partition in the table has to be moved manually into new location

**Hands on Activity**

**STATIC PARTITION**

1. Create following text files and put the data set using text editor.

file.txt

1,sunny,SC,2009

2,annie,HR,2009

3,jack,HR,2010

4,kevin,TP,2010

file2009.txt

1,sunny,SC,2009

2,annie,HR,2009

file2010.txt

3,jack,HR,2010

4,kevin,TP,2010

2. Upload file2009.txt into hdfs path "/user/maria\_dev/mypartition/"

Upload file2010.txt into hdfs path "/user/maria\_dev/mypartition/"

3. Hive command (Ambari Hive view)

Now, let’s partition above data from two files using static partitioning by year

When we are retrieving the data from the table, only the data of the specified partition will be queried. Creating a static partitioned table is as follows:

CREATE TABLE table\_static\_p

(id INT, name STRING, dept STRING)

PARTITIONED BY (year STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES TERMINATED BY '\n';

LOAD DATA INPATH "/user/maria\_dev/mypartition/file2009.txt" OVERWRITE INTO TABLE table\_static\_p

PARTITION (year='2009');

LOAD DATA INPATH "/user/maria\_dev/mypartition/file2010.txt" OVERWRITE INTO TABLE table\_static\_p

PARTITION (year='2010');

SELECT \* FROM table\_static\_p WHERE year = '2009';

// Following command for checking if a particular partition exists or not?

SHOW PARTITIONS table\_name PARTITION(partitioned\_column='partition\_value')

**DYNAMIC PARTITION**

Now, let’s partition above data into two files using dynamic partitioning by year.

1. Hive command (Ambari Hive view)

Let’s use previously created table\_static\_p table as a source. Hive will determine which data to retrieve from the query result depending on the partition. Creating a dynamic partitioned table is as follows:

\*you need to change partition mode to enable dynamic partitioning.

(Default setting is strict and it prevents all partitions to be dynamic and requires at least one static partition.)

SET hive.exec.dynamic.partition=true;

SET hive.exec.dynamic.partition.mode=nonstrict;

CREATE TABLE table\_dynamic\_p

(id INT, name STRING, dept STRING)

PARTITIONED BY (year STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES TERMINATED BY '\n'

STORED AS textfile

LOCATION "/user/maria\_dev/dynamic\_p";

INSERT OVERWRITE TABLE table\_dynamic\_p

PARTITION (year)

SELECT id, name, dept, year

FROM table\_static\_p;

SELECT \* FROM table\_dynamic\_p WHERE year = '2010';

2. Go to Ambari File View to check the contents of the created files in "/user/maria\_dev/dynamic\_p"

**Question: What Is The Significance Of The Line Set Hive.mapred.mode = Strict;**

**Answer :**

It sets the mapreduce jobs to strict mode. By which the queries on partitioned tables cannot run without a WHERE clause. And it restricts ORDER BY operation without a LIMIT clause. This prevents very large job running for long time.

**Question. How Do You Check If A Particular Partition Exists?**

**Answer :**

This can be done with following query:

SHOW PARTITIONS table\_name PARTITION(partitioned\_col='partition\_val');

**Question. How Can You Stop A Partition Form Being Queried?**

**Answer :**

By using the ENABLE OFFLINE clause with ALTER TABLE Statement.

ALTER TABLE table\_name PARTITION(partitioned\_col='partition\_val') ENABLE OFFLINE;

**Question. How Can You Stop A Partition Form Being Dropped?**

**Answer :**

By using the DISABLE NO\_DROP clause with ALTER TABLE Statement.

ALTER TABLE table\_name PARTITION(partitioned\_col='partition\_val') DISABLE NO\_DROP;