getindata

Apache Hive

Working with Apache Hive

During this exercies, you become an Apache Hive Developer/Analyst who will create physical objects in Hive and perform some basic operations on them.

Select The Username

Please use your Linux account at the edge node (cdh00.cl.ii.pw.edu.pl)

Using Hive command line interface

1. **Login using ssh to edgenode cdh00.cl.ii.pw.edu.pl using your Linux account.**
2. **In this exercise we are going to use the CSV file we uploaded to HDFS earlier today. First of all check if the file is in your HDFS home folder:**

```
hdfs dfs -ls /user/${USER}/external/measured_data/measured_data.csv
```

3. **Login to hive and create your own Hive database**

```
hive
hive> create database ${USER};
```

4. **Open another ssh connection or if your are using terminal manager like byobu/tmux/screen just open another window. Check if your database exists:**

```
hive -S -e "show databases;" 2>/dev/null| grep ${USER}
```

In this step you you executed a hive command directly from Linux cmd line. What do you think – where can it be useful?

5. **Create an external Hive table in your database using a CSV file you have in your HDFS home folder:**

```
CREATE EXTERNAL TABLE ${USER}.MEASURED_DATA_CSV_EXTERNAL
(
 `md_lsb_id` int,
  `md_timestamp` string,
  `md_lsb_id2` int,
  `md_value` double,
  `md_unit` string,
  `md_timetype` string,
  `md_quality_mark` string,
  `md_desc` string
)
```

```
ROW FORMAT DELIMITED FIELDS
TERMINATED BY '|' LINES TERMINATED BY '\n' STORED as TEXTFILE
LOCATION '/user/${USER}/external/measured_data';
```

6. **Change your working database to yours and list all the tables “stored” in it:**

```
use ${USER};
OK
Time taken: 0.318 seconds

hive> show tables;
OK
measured_data_csv_external
```

7. **Print and analyze table metadata:**

```
hive> desc formatted measured_data_csv_external;
OK
# col_name              data_type               comment

md_lsb_id               int
md_timestamp            string
md_lsb_id2              int
md_value                double
md_unit                 string
md_timetype             string
md_quality_mark         string
md_desc                 string

# Detailed Table Information
Database:               xmwiewio
Owner:                  xmwiewio
CreateTime:             Thu May 26 13:33:52 CEST 2016
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs:///cdh01.cl.ii.pw.edu.pl:802/user/xmwiewio/external/measured_data
Table Type:             EXTERNAL_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   false
        EXTERNAL                TRUE
        numFiles                1
        numRows                 -1
        rawDataSize             -1
        totalSize               331035256
        transient_lastDdlTime   1464262432

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
```

```
Num Buckets:           -1
Bucket Columns:         []
Sort Columns:          []
Storage Desc Params:
        field.delim             |
        line.delim              \n
        serialization.format    |
Time taken: 0.389 seconds, Fetched: 41 row(s)
```

8. **Generate a DDL (data defintion) for a table:**

```
hive>show create table ${USER}.measured_data_csv_external;
OK
CREATE EXTERNAL TABLE `training.measured_data_csv_external`(
  `md_lsb_id` int,
  `md_timestamp` string,
  `md_lsb_id2` int,
  `md_value` double,
  `md_unit` string,
  `md_timetype` string,
  `md_quality_mark` string,
  `md_desc` string)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY '|'
  LINES TERMINATED BY '\n'
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  'hdfs://HDFS-HA/user/sar_wim/external/measured_data'
TBLPROPERTIES (
  'COLUMN_STATS_ACCURATE'='false',
  'numFiles'='1',
  'numRows'='-1',
  'rawDataSize'='-1',
  'totalSize'='331035256',
  'transient_lastDdlTime'='1464262432')
Time taken: 0.276 seconds, Fetched: 25 row(s)
```

9. **Try to run a simple query to get an average value od md_value column grouped by md_lsb_id and a year and month for the last few months;**

```
hive> SELECT md_lsb_id,date_format(md_timestamp,'yyyy-MM') as
year_month,avg(md_value) as avg_val FROM ${USER}.measured_data_csv_external
GROUP BY  md_lsb_id,date_format(md_timestamp,'yyyy-MM') ORDER BY
md_lsb_id,year_month desc limit 15;
```

```
OK
1       2017-05 0.49873148036001064
1       2016-05 0.49768154708603957
1       2015-05 0.497515021264509
2       2017-05 0.4988676805140444
2       2016-05 0.49742044523741197
2       2015-05 0.5056076317793062
3       2017-05 0.49836022891096265
3       2016-05 0.4991462674179214
3       2015-05 0.500562383337185
4       2017-05 0.4993031206736664
4       2016-05 0.4955264848762207
4       2015-05 0.49834381017309315
5       2017-05 0.4983890152540982
5       2016-05 0.498885413611568
5       2015-05 0.4989340026906018
Time taken: 69.092 seconds, Fetched: 15 row(s)
```

10. **Create a copy of the ${USER}.measured_data_csv_external table as managed by Hive and stored in ORC format. In another ssh session or terminal window compare the sizes of both tables:**

```
hive>CREATE TABLE ${USER}.MEASURED_DATA_ORC STORED AS ORC AS
SELECT * FROM ${USER}.MEASURED_DATA_CSV_EXTERNAL;
```

```
hdfs dfs -du -h -s /user/${USER}/external/measured_data
482.0 M  1.4 G  /user/xmwiewio/external/measured_data

hdfs dfs -du -h -s /user/hive/warehouse/${USER}.db/measured_data_orc
52.4 M  157.3 M  /user/hive/warehouse/xmwiewio.db/measured_data_orc
```

11. **Rerun the query from step 9 using ORC table and compare the time taken:**

```
hive> SELECT md_lsb_id,date_format(md_timestamp,'yyyy-MM') as
year_month,avg(md_value) as avg_val FROM ${USER}.measured_data_orc GROUP BY
md_lsb_id,date_format(md_timestamp,'yyyy-MM') ORDER BY md_lsb_id,year_month
desc limit 15;

OK
1       2017-05 0.4987314803600108
1       2016-05 0.4976815470860367
1       2015-05 0.497515021264509
2       2017-05 0.49886768051404695
2       2016-05 0.4974204452374088
2       2015-05 0.5056076317793062
3       2017-05 0.4983602289109608
```

```
3       2016-05 0.499146267417921
3       2015-05 0.500562383337185
4       2017-05 0.49930312067366467
4       2016-05 0.4955264848762195
4       2015-05 0.49834381017309315
5       2017-05 0.4983890152540996
5       2016-05 0.498885413611569
5       2015-05 0.4989340026906018
Time taken: 52.808 seconds, Fetched: 15 row(s)
```

What do you think – what are the reasons for the speedup, if any?

12. Create a partitioned by year table:

```
CREATE TABLE ${USER}.MEASURED_DATA_ORC_PART
(
 `md_lsb_id` int,
  `md_timestamp` string,
  `md_lsb_id2` int,
  `md_value` double,
  `md_unit` string,
  `md_timetype` string,
  `md_quality_mark` string,
  `md_desc` string
)
PARTITIONED BY (dt int)
STORED AS ORC;

hive> desc ${USER}.MEASURED_DATA_ORC_PART;
OK
md_lsb_id               int
md_timestamp            string
md_lsb_id2              int
md_value                double
md_unit                 string
md_timetype             string
md_quality_mark         string
md_desc                 string
dt                      int

# Partition Information
# col_name              data_type               comment

dt                      int
Time taken: 0.411 seconds, Fetched: 14 row(s)
```

13. **Load the year partitions using Hive "INSERT OVERWRITE PARTITION"**

```
INSERT OVERWRITE TABLE ${USER}.MEASURED_DATA_ORC_PART
PARTITION(dt=2015)
SELECT * FROM ${USER}.MEASURED_DATA_ORC WHERE
year(md_timestamp)=2015;
```

```
INSERT OVERWRITE TABLE ${USER}.MEASURED_DATA_ORC_PART
PARTITION(dt=2016)
SELECT * FROM ${USER}.MEASURED_DATA_ORC WHERE
year(md_timestamp)=2016;

INSERT OVERWRITE TABLE ${USER}.MEASURED_DATA_ORC_PART
PARTITION(dt=2017)
SELECT * FROM ${USER}.MEASURED_DATA_ORC WHERE
year(md_timestamp)=2017;
```

14. List table partitions:

```
hive> show partitions ${USER}.measured_data_orc_part;
OK
dt=2013
dt=2014
dt=2015
```

15. Overwrite all the partitions at once using dynamic partitioning option:

```
set hive.exec.dynamic.partition.mode=nonstrict
INSERT OVERWRITE TABLE ${USER}.MEASURED_DATA_ORC_PART
PARTITION(dt)
SELECT md_lsb_id,
md_timestamp,
md_lsb_id2,
md_value,
md_unit,
md_timetype,
md_quality_mark,
md_desc,
year(md_timestamp) as dt FROM ${USER}.MEASURED_DATA_ORC;

         Time taken for load dynamic partitions : 505
        Loading partition {dt=2017}
        Loading partition {dt=2016}
        Loading partition {dt=2015}
         Time taken for adding to write entity : 1
Partition xmwiewio.measured_data_orc_part{dt=2015} stats: [numFiles=1,
numRows=2400000, totalSize=18330755, rawDataSize=1140000000]
Partition xmwiewio.measured_data_orc_part{dt=2016} stats: [numFiles=1,
numRows=2400000, totalSize=18331378, rawDataSize=1140000000]
Partition xmwiewio.measured_data_orc_part{dt=2017} stats: [numFiles=1,
numRows=2400000, totalSize=18330712, rawDataSize=1140000000]
MapReduce Jobs Launched:
OK
```

16. Compare the execution time, CPU time and HDFS reads of non-partitioned and partitioned table:

```
hive> select count(*) from ${USER}.measured_data_orc_part where dt=2015 and
md_lsb_id=100;


Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-05-18 21:39:51,763 Stage-1 map = 0%,  reduce = 0%
2017-05-18 21:39:57,951 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.36 sec
2017-05-18 21:40:04,128 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.19
sec
MapReduce Total cumulative CPU time: 3 seconds 190 msec
Ended Job = job_1495125662310_0016
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.19 sec   HDFS Read: 57176
HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 190 msec
OK
24000
Time taken: 20.56 seconds, Fetched: 1 row(s)
------------
hive> select count(*) from training.measured_data_orc where year(md_timestamp)=2015
and md_lsb_id=100;


Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-05-18 21:40:16,918 Stage-1 map = 0%,  reduce = 0%
2017-05-18 21:40:23,099 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.96 sec
2017-05-18 21:40:29,291 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.99
sec
MapReduce Total cumulative CPU time: 3 seconds 990 msec
Ended Job = job_1495125662310_0017
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.99 sec   HDFS Read: 169556
HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 990 msec
OK
24000
Time taken: 19.509 seconds, Fetched: 1 row(s)
```