BIG DATA HADOOP & SPARK TRAINING

Case Study II for session 11

Rashmi Krishna

4/14/18

Hive & Hbase case study

Case Study Description

Let us take up the CUSTOMER and TRANSACTIONS table we have created in the Lets Do

Together section. Let us solve the following use cases using these tables :-

Creating tables:

Go to Hive shell and give the below commands to create a table “CSTRECORDS” and “TXNRECORDSBYCAT”.

CREATE TABLE CSTRECORDS(custno INT, Lname STRING, Fname STRING, productid INT, custdesgntn STRING)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

CREATE TABLE TXNRECORDSBYCAT(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

Loading the tables:

Load the data from local files system to hive tables using the below commands:

LOAD DATA LOCAL INPATH '/home/acadgild/casestudies/hivemovierating/custs.txt' into table CSTRECORDS;

LOAD DATA LOCAL INPATH '/home/acadgild/casestudies/hivemovierating/txns.txt' into table TXNRECORDSBYCAT;

```
hive> CREATE TABLE CSTRECORDS( custno INT, Lname STRING, Fname STRING, productid INT, custdesgntn STRING)ROW FORMAT DELIMITED FIELDS TE
RMINATED BY ',';
OK
Time taken: 1.485 seconds
hive> CREATE TABLE TXNRECORDSBYCAT(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spe
ndby STRING)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
OK
Time taken: 0.231 seconds
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Downloads/custs.txt' into table CSTRECORDS;
Loading data to table simplidb.cstrecords
OK
Time taken: 3.38 seconds
hive> LOAD DATA LOCAL INPATH '/home/acadgild/Downloads/txns.txt' into table TXNRECORDSBYCAT;
Loading data to table simplidb.txnrecordsbycat
OK
Time taken: 1.901 seconds
hive> show tables;
OK
cstrecords
employee_hbase
txnrecordsbycat
txns
Time taken: 0.079 seconds, Fetched: 4 row(s)
hive>
```

1. Find out the number of transaction done by each customer

select a.custno,b.Lname,b.Fname,count(a.amount) from TXNRECORDSBYCAT a join CSTRECORDS b on a.custno=b.custno group by a.custno,b.fname,b.lname;

```
hive> select a.custno,b.Lname,b.Fname,count(a.amount) from  TXNRECORDSBYCAT a join CSTRECORDS b on a.custno=b.custno gro
up by a.custno,b.fname,b.lname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180414181512_3ea889aa-9f4c-4d9c-a92e-516f556890dc
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5
.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2018-04-14 18:15:35     Starting to launch local task to process map join;      maximum memory = 518979584
2018-04-14 18:15:40     Dump the side-table for tag: 1 with group count: 10 into file: file:/tmp/acadgild/bff86f13-be01-
40d5-be8f-091710e0d0f0/hive_2018-04-14_18-15-12_528_5291785545355393029-1/-local-10005/HashTable-Stage-2/MapJoin-mapfile
01--.hashtable
2018-04-14 18:15:40     Uploaded 1 File to: file:/tmp/acadgild/bff86f13-be01-40d5-be8f-091710e0d0f0/hive_2018-04-14_18-1
5-12_528_5291785545355393029-1/-local-10005/HashTable-Stage-2/MapJoin-mapfile01--.hashtable (626 bytes)
2018-04-14 18:15:40     End of local task; Time Taken: 4.448 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1523709148707_0001, Tracking URL = http://localhost:8088/proxy/application_1523709148707_0001/
```

```
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1523709148707_0001, Tracking URL = http://localhost:8088/proxy/application_1523709148707_0001/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523709148707_0001
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-04-14 18:16:08,910 Stage-2 map = 0%,  reduce = 0%
2018-04-14 18:16:26,515 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 4.17 sec
2018-04-14 18:16:42,367 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 7.28 sec
MapReduce Total cumulative CPU time: 7 seconds 280 msec
Ended Job = job_1523709148707_0001
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.28 sec   HDFS Read: 19122 HDFS Write: 451 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 280 msec
OK
4000001 Kristina        Chung   8
4000002 Paige   Chen    6
4000003 Sherri  Melton  3
4000004 Gretchen        Hill    5
4000005 Karen   Puckett 5
4000006 Patrick Song    5
4000007 Elsie   Hamilton       6
4000008 Hazel   Bender  10
4000009 Malcolm Wagner  6
4000010 Dolores McLaughlin      6
Time taken: 91.165 seconds, Fetched: 10 row(s)
hive>
```

2. Create a new table called TRANSACTIONS_COUNT. This table should have 3 fields - custid, fname and count

CREATE TABLE TRANSACTIONS_COUNT(custno INT, Fname STRING, count INT)ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

```
hive> CREATE TABLE TRANSACTIONS_COUNT( custno INT, Fname STRING, count INT)ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
;
OK
Time taken: 0.274 seconds
hive> describe TRANSACTIONS_COUNT;
OK
custno                  int
fname                   string
count                   int
Time taken: 0.209 seconds, Fetched: 3 row(s)
hive>
```

3.Now write a hive query in such a way that the query populates the data obtained in Step 1 above and populate the table in step 2 above.

INSERT OVERWRITE TABLE TRANSACTIONS_COUNT select a.custno,b.Fname, count(a.amount)as count from TXNRECORDSBYCAT a join CSTRECORDS b on a.custno=b.custno group by a.custno,b.fname;

```
hive> INSERT OVERWRITE TABLE TRANSACTIONS_COUNT  select a.custno,b.Fname, count(a.amount)as count from  TXNRECORDSBYCAT
a join CSTRECORDS b on a.custno=b.custno group by a.custno,b.fname;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180414182134_198982d3-97c8-4e39-8861-79ef9bb513bf
Total jobs = 1
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/acadgild/install/hive/apache-hive-2.3.2-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5
.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
2018-04-14 18:21:50     Starting to launch local task to process map join;      maximum memory = 518979584
2018-04-14 18:21:54     Dump the side-table for tag: 1 with group count: 10 into file: file:/tmp/acadgild/bff86f13-be01-
40d5-be8f-091710e0d0f0/hive_2018-04-14_18-21-34_713_1221870674740354775-1/-local-10003/HashTable-Stage-2/MapJoin-mapfile
11--.hashtable
2018-04-14 18:21:55     Uploaded 1 File to: file:/tmp/acadgild/bff86f13-be01-40d5-be8f-091710e0d0f0/hive_2018-04-14_18-2
1-34_713_1221870674740354775-1/-local-10003/HashTable-Stage-2/MapJoin-mapfile11--.hashtable (553 bytes)
2018-04-14 18:21:55     End of local task; Time Taken: 4.435 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1523709148707_0002, Tracking URL = http://localhost:8088/proxy/application_1523709148707_0002/
```

```
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1523709148707_0002, Tracking URL = http://localhost:8088/proxy/application_1523709148707_0002/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523709148707_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2018-04-14 18:22:14,828 Stage-2 map = 0%,  reduce = 0%
2018-04-14 18:22:30,645 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 3.67 sec
2018-04-14 18:22:45,968 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 8.53 sec
MapReduce Total cumulative CPU time: 8 seconds 660 msec
Ended Job = job_1523709148707_0002
Loading data to table default.transactions_count
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 8.66 sec   HDFS Read: 19250 HDFS Write: 254 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 660 msec
OK
Time taken: 75.815 seconds
hive> select * from TRANSACTIONS_COUNT;
OK
4000001 Chung   8
4000002 Chen    6
4000003 Melton  3
4000004 Hill    5
4000005 Puckett 5
4000006 Song    5
4000007 Hamilton        6
4000008 Bender  10
4000009 Wagner  6
4000010 McLaughlin      6
Time taken: 0.551 seconds, Fetched: 10 row(s)
```

4. Now lets make the TRANSACTIONS_COUNT table Hbase complaint. In the sence, use Ser Des And Storate handler features of hive to change the TRANSACTIONS_COUNT table to be able to create a TRANSACTIONS table in Hbase.

```
create table TRANSACTIONS_Hbase(custid String, f_name string, count int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties ("hbase.columns.mapping"=":key,customerdetails:fname,customerdetails:count")
tblproperties("hbase.table.name"="Transactions");
```

```
hive> create table TRANSACTIONS_Hbase(custid String, f_name string, count int) STORED BY 'org.apache.hadoop.hive.hbase.H
BaseStorageHandler' with serdeproperties ("hbase.columns.mapping":key,customerdetails:fname,customerdetails:count") tb
lproperties("hbase.table.name"="Transactions");
OK
Time taken: 5.804 seconds
hive>
```

```
hbase(main):002:0> list
TABLE
clicks
1 row(s) in 0.0280 seconds

=> ["clicks"]
hbase(main):003:0>
hbase(main):003:0> list
TABLE
Transactions
clicks
2 row(s) in 0.0260 seconds

=> ["Transactions", "clicks"]
hbase(main):004:0>
```

before using storage handling feature in hive

After using storage handler feature in hive to create a table "transactions"

5. Now insert the data in TRANSACTIONS_COUNT table using the query in step 3 again, this should populate the Hbase TRANSACTIONS table automatically

INSERT OVERWRITE TABLE transactions_Hbase select * from transactions_count;

```
hive> INSERT OVERWRITE TABLE transactions_Hbase select * from transactions_count;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20180414182842_d3f860e1-7c04-422c-b452-debeaa0fbe88
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1523709148707_0003, Tracking URL = http://localhost:8088/proxy/application_1523709148707_0003/
Kill Command = /home/acadgild/install/hadoop/hadoop-2.6.5/bin/hadoop job  -kill job_1523709148707_0003
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2018-04-14 18:29:05,221 Stage-3 map = 0%,  reduce = 0%
2018-04-14 18:29:20,926 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 4.51 sec
MapReduce Total cumulative CPU time: 5 seconds 270 msec
Ended Job = job_1523709148707_0003
MapReduce Jobs Launched:
Stage-Stage-3: Map: 1   Cumulative CPU: 5.27 sec   HDFS Read: 11219 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 270 msec
OK
Time taken: 42.311 seconds
hive>
```

```
hbase(main):004:0> scan 'Transactions'
ROW                                    COLUMN+CELL
0 row(s) in 0.2820 seconds

hbase(main):005:0> scan 'Transactions'
ROW                                    COLUMN+CELL
 4000001                               column=customerdetails:count, timestamp=1523710761521, value=8
 4000001                               column=customerdetails:fname, timestamp=1523710761521, value=Chung
 4000002                               column=customerdetails:count, timestamp=1523710761521, value=6
 4000002                               column=customerdetails:fname, timestamp=1523710761521, value=Chen
 4000003                               column=customerdetails:count, timestamp=1523710761521, value=3
 4000003                               column=customerdetails:fname, timestamp=1523710761521, value=Melton
 4000004                               column=customerdetails:count, timestamp=1523710761521, value=5
 4000004                               column=customerdetails:fname, timestamp=1523710761521, value=Hill
 4000005                               column=customerdetails:count, timestamp=1523710761521, value=5
 4000005                               column=customerdetails:fname, timestamp=1523710761521, value=Puckett
 4000006                               column=customerdetails:count, timestamp=1523710761521, value=5
 4000006                               column=customerdetails:fname, timestamp=1523710761521, value=Song
 4000007                               column=customerdetails:count, timestamp=1523710761521, value=6
 4000007                               column=customerdetails:fname, timestamp=1523710761521, value=Hamilton
 4000008                               column=customerdetails:count, timestamp=1523710761521, value=10
 4000008                               column=customerdetails:fname, timestamp=1523710761521, value=Bender
 4000009                               column=customerdetails:count, timestamp=1523710761521, value=6
 4000009                               column=customerdetails:fname, timestamp=1523710761521, value=Wagner
 4000010                               column=customerdetails:count, timestamp=1523710761521, value=6
 4000010                               column=customerdetails:fname, timestamp=1523710761521, value=McLaughlin
10 row(s) in 0.3530 seconds

hbase(main):006:0>
```

before loading the data in hive

After loading the data in hive

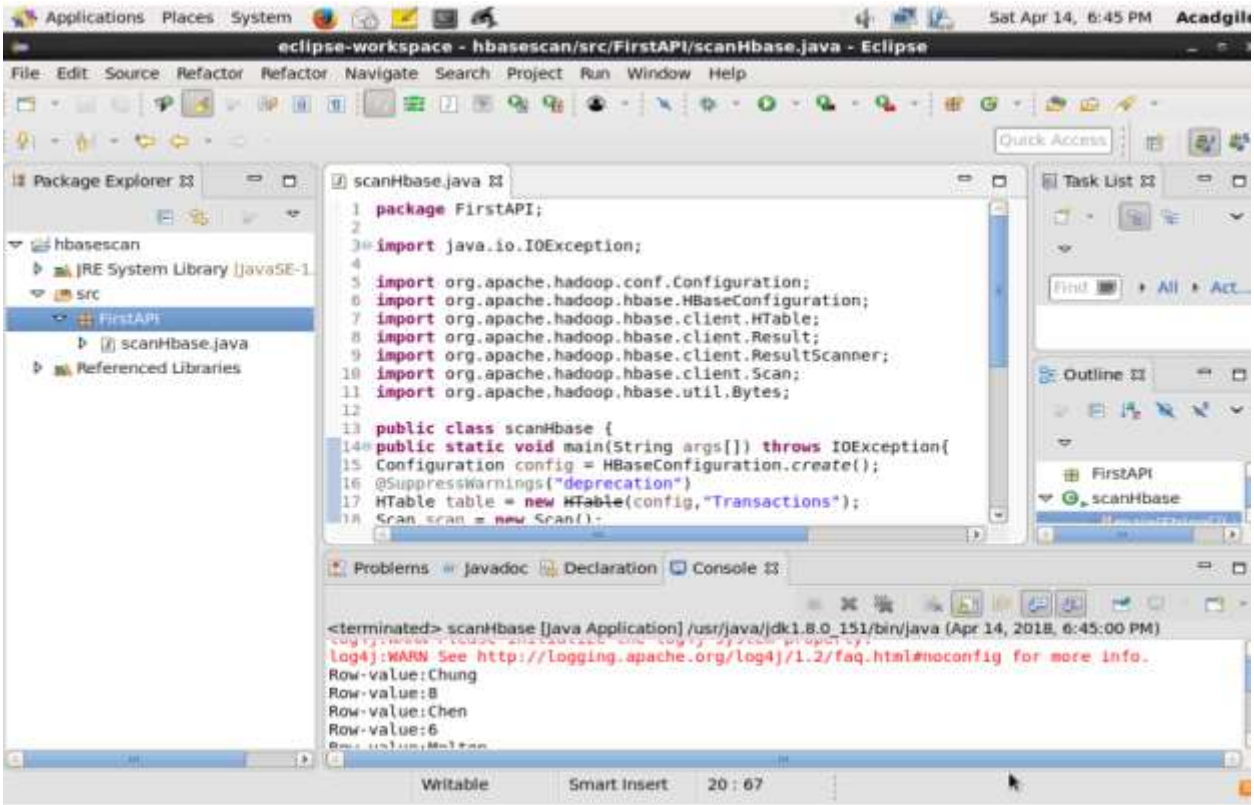
6. Now from the Hbase level, write the Hbase java API code to access and scan the TRANSACTIONS table data from java level

```
package FirstAPI;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.util.Bytes;
public class scanHbase {
public static void main(String args[]) throws IOException{
Configuration config = HBaseConfiguration.create();
@SuppressWarnings("deprecation")
HTable table = new HTable(config,"Transcations");
Scan scan = new Scan();
scan.addColumn(Bytes.toBytes("customerdetails"), Bytes.toBytes("fname"));
scan.addColumn(Bytes.toBytes("customerdetails"), Bytes.toBytes("count"));
ResultScanner result = table.getScanner(scan);
for(Result res:result){
byte[]
val=res.getValue(Bytes.toBytes("customerdetails"),Bytes.toBytes("fname"));
byte[]
val1=res.getValue(Bytes.toBytes("customerdetails"),Bytes.toBytes("count"));
System.out.println("Row-value:"+Bytes.toString(val));
System.out.println("Row-value:"+Bytes.toString(val1));

}table.close();}}
```

Explanation:

- Here we are scanning the Hbase table “Transactions” created using storage handler in hive.
- We are selecting the column family “Customerdetails” and column qualifiers “fname and count”.
- We are converting their values from Bytes to string and printing them, as shown below



This is the output: