Session 8: ADVANCED HIVE

Assignment 8.3

Student Name: Subham Vishal

Course: Big Data Hadoop & Spark Training

Assignment 8.3– Refer the given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

Contents

Introduction

In this assignment you need to implement transaction concepts in Hive. I'm using a dataset created on my own.

Associated Data Files

https://acadgild.com/blog/transactions-in-hive/

Problem Statement

Refer the above given link for transactions in Hive and implement the operations given in the blog using your own sample data set and send us the screenshot.

Transactions in Hive

Transactions in Hive are introduced in **Hive 0.13**, but they only partially fulfill the ACID properties like atomicity, consistency, durability, at the partition level. Here, Isolation can be provided by turning on one of the locking mechanisms available with zookeeper or in memory.

But in **Hive 0.14**, new API's have been added to completely fulfill the ACID properties while performing any transaction.

Transactions are provided at the row-level in Hive 0.14. The different row-level transactions available in Hive 0.14 are as follows:

- Insert
- Delete
- Update

There are numerous limitations with the present transactions available in Hive 0.14. ORC is the file format supported by Hive transaction. It is now essential to have ORC file format for performing transactions in Hive. The table needs to be bucketed in order to support transactions.

Row-level Transactions Available in Hive 0.14

Let's perform some row-level transactions available in Hive 0.14. Before creating a Hive table that supports transactions, the transaction features present in Hive needs to be turned on, as by default they are turned off.

The below properties needs to be set appropriately in hive shell, order-wise to work with transactions in Hive:

```
hive>
    >
    > set hive.support.concurrency=true;
hive> set hive.enforce.bucketing=true;
Query returned non-zero code: 1, cause: hive configuration hive.enforce.bucketing does not exists.
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
hive> set hive.compactor.initiator.on=true;
hive> set hive.compactor.worker.threads=5;
hive> set hive.compactor.worker.threads;
hive.compactor.worker.threads=5
hive>
```

Creating a Table That Supports Hive Transactions

The above syntax will create a table with name **'movie'** and the columns present in the table are ***'movie_id, movie_code, movie_name,screen_number and screen_loc'*** . We are bucketing the table by '***movie_id*** 'and the table format is '**orc'**, also we are enabling the transactions in the table by specifying it inside the **TBLPROPERTIES** as **'transactional'='true'**

HIVE Code:

CREATE TABLE movie

(movie_id int,

movie_code string,

movie_name string,

screen_number int,

screen_loc string)

clustered by (movie_id) INTO 5 buckets STORED as orc TBLPROPERTIES('transactional'='true');

```
hive (Custom)>
            >
            > CREATE TABLE movie
            > (movie_id int,
            > movie_code string,
            > movie_name string,
            > screen_number int,
            > screen_loc string)
            > clustered by (movie_id) INTO 5 buckets STORED as orc TBLPROPERTIES('transactional'='true');
OK
Time taken: 0.377 seconds
hive (Custom)>
```

```
            >
            > Show tables;
OK
college
movie
temperature_data
temperature_data_vw
values__tmp__table__1
values__tmp__table__2
Time taken: 0.19 seconds, Fetched: 6 row(s)
hive (Custom)>
```

```
hive (Custom)> DESCRIBE movie;
OK
movie_id                int
movie_code              string
movie_name              string
screen_number           int
screen_loc              string
Time taken: 0.239 seconds, Fetched: 5 row(s)
hive (Custom)>
```

The table '**movie'** is successfully created which supports row level transactions of HIVE.

Inserting Data into a Hive Table

The below command is used to insert row wise data into the Hive table. Here, each row is separated by '( )' brackets.

HIVE Code:

INSERT INTO TABLE movie values(1,'AYM','Acham Yenbathu Madamayada',1,'PVR1'),(2,'VTV','Vinnai Thandi Varuvaya',2,'PVR1'),(3,'YA','Yennai Arinthal',3,'PVR1'),(4,'VV','Vettayadu Vilayadu',4,'PVR1'),(5,'KK','Kakka Kakka',5,'PVR1'),(6,'DW','Dark Wind',1,'PVR2'),(7,'TS','Tumhari Sulu',2,'PVR2'),(8,'QQS','Qarib Qarib Single',3,'PVR2'),(9,'GA','Golmal Again',4,'PVR2'),(10,'SS','Secret Superstar',5,'PVR2');

```
          > INSERT INTO TABLE movie values(1,'AYM','Acham Yenbathu Madamayada',1,'PVR1'),(2,'VTV','Vinnai Thandi Varuvaya',2,'PVR1'),(3,'YA',
'Yennai Arinthal',3,'PVR1'),(4,'VV','Vettayadu Vilayadu',4,'PVR1'),(5,'KK','Kakka Kakka',5,'PVR1'),(6,'DW','Dark Wind',1,'PVR2'),(7,'TS','Tumhar
i Sulu',2,'PVR2'),
          > (8,'QQS','Qarib Qarib Single',3,'PVR2'),(9,'GA','Golmal Again',4,'PVR2'),(10,'SS','Secret Superstar',5,'PVR2');
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. s
park, tez) or using Hive 1.X releases.
Query ID = acadgild_20171120120212_55707184-d5ee-4c14-a47e-595be81ce2ff
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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_1511157228616_0003, Tracking URL = http://localhost:8088/proxy/application_1511157228616_0003/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1511157228616_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2017-11-20 12:02:33,257 Stage-1 map = 0%,  reduce = 0%
2017-11-20 12:02:49,189 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.24 sec
2017-11-20 12:03:40,804 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 5.67 sec
2017-11-20 12:03:44,013 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 6.62 sec
2017-11-20 12:03:45,554 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 7.51 sec
2017-11-20 12:03:48,852 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 9.49 sec
2017-11-20 12:03:50,380 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 10.78 sec
2017-11-20 12:03:58,116 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 14.08 sec
2017-11-20 12:04:02,671 Stage-1 map = 100%,  reduce = 76%, Cumulative CPU 16.55 sec
2017-11-20 12:04:04,250 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 17.08 sec
2017-11-20 12:04:05,812 Stage-1 map = 100%,  reduce = 90%, Cumulative CPU 22.41 sec
2017-11-20 12:04:07,200 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 25.39 sec
MapReduce Total cumulative CPU time: 25 seconds 390 msec
Ended Job = job_1511157228616_0003
Loading data to table custom.movie
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 25.68 sec   HDFS Read: 30639 HDFS Write: 5196 SUCCESS
Total MapReduce CPU Time Spent: 25 seconds 680 msec
OK
Time taken: 117.19 seconds
hive (Custom)>
```

```
hive (Custom)>
            > Select * From movie;
OK
movie.movie_id  movie.movie_code       movie.movie_name        movie.screen_number     movie.screen_loc
10      SS      Secret Superstar        5       PVR2
5       KK      Kakka Kakka     5       PVR1
6       DW      Dark Wind       1       PVR2
1       AYM     Acham Yenbathu Madamayada       1       PVR1
7       TS      Tumhari Sulu    2       PVR2
2       VTV     Vinnai Thandi Varuvaya  2       PVR1
8       QQS     Qarib Qarib Single      3       PVR2
3       YA      Yennai Arinthal 3       PVR1
9       GA      Golmal Again    4       PVR2
4       VV      Vettayadu Vilayadu      4       PVR1
Time taken: 0.292 seconds, Fetched: 10 row(s)
hive (Custom)>
            >
```

Now, we have successfully inserted the data into the Hive table and we saw the table data above using ***Select * From movie;***

Now if we try to re-insert the same data again, it will be appended to the previous data as shown below:

```
            > INSERT INTO TABLE movie values(1,'AYM','Acham Yenbathu Madamayada',1,'PVR1'),(2,'VTV','Vinnai Thandi Varuvaya',2,'PVR1'),(3,'YA',
'Yennai Arinthal',3,'PVR1'),(4,'VV','Vettayadu Vilayadu',4,'PVR1'),(5,'KK','Kakka Kakka',5,'PVR1'),(6,'DW','Dark Wind',1,'PVR2'),(7,'TS','Tumhar
i Sulu',2,'PVR2'),
            > (8,'QQS','Qarib Qarib Single',3,'PVR2'),(9,'GA','Golmal Again',4,'PVR2'),(10,'SS','Secret Superstar',5,'PVR2');
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. s
park, tez) or using Hive 1.X releases.
Query ID = acadgild_20171120121137_82c62685-925d-45ae-b5da-f0b2bc9c82e8
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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_1511157228616_0004, Tracking URL = http://localhost:8088/proxy/application_1511157228616_0004/
Kill Command = /home/acadgild/hadoop-2.7.2/bin/hadoop job  -kill job_1511157228616_0004
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2017-11-20 12:12:09,656 Stage-1 map = 0%,  reduce = 0%
2017-11-20 12:12:28,293 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.68 sec
2017-11-20 12:13:24,707 Stage-1 map = 100%,  reduce = 13%, Cumulative CPU 7.09 sec
2017-11-20 12:13:28,081 Stage-1 map = 100%,  reduce = 27%, Cumulative CPU 8.04 sec
2017-11-20 12:13:29,649 Stage-1 map = 100%,  reduce = 40%, Cumulative CPU 8.97 sec
2017-11-20 12:13:32,924 Stage-1 map = 100%,  reduce = 53%, Cumulative CPU 10.97 sec
2017-11-20 12:13:36,040 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 12.97 sec
2017-11-20 12:13:43,945 Stage-1 map = 100%,  reduce = 73%, Cumulative CPU 15.92 sec
2017-11-20 12:13:46,988 Stage-1 map = 100%,  reduce = 76%, Cumulative CPU 18.32 sec
2017-11-20 12:13:48,844 Stage-1 map = 100%,  reduce = 80%, Cumulative CPU 18.76 sec
2017-11-20 12:13:50,583 Stage-1 map = 100%,  reduce = 87%, Cumulative CPU 21.35 sec
2017-11-20 12:13:52,117 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 26.95 sec
MapReduce Total cumulative CPU time: 26 seconds 950 msec
Ended Job = job_1511157228616_0004
Loading data to table custom.movie
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 27.72 sec   HDFS Read: 30414 HDFS Write: 5203 SUCCESS
Total MapReduce CPU Time Spent: 27 seconds 720 msec
OK
_col0   _col1   _col2   _col3   _col4
Time taken: 137.432 seconds
hive (Custom)>
```

```
hive (Custom)>
            > select * from movie;
OK
movie.movie_id  movie.movie_code        movie.movie_name        movie.screen_number     movie.screen_loc
10      SS      Secret Superstar        5       PVR2
5       KK      Kakka Kakka     5       PVR1
10      SS      Secret Superstar        5       PVR2
5       KK      Kakka Kakka     5       PVR1
6       DW      Dark Wind       1       PVR2
1       AYM     Acham Yenbathu Madamayada       1       PVR1
6       DW      Dark Wind       1       PVR2
1       AYM     Acham Yenbathu Madamayada       1       PVR1
7       TS      Tumhari Sulu    2       PVR2
2       VTV     Vinnai Thandi Varuvaya  2       PVR1
7       TS      Tumhari Sulu    2       PVR2
2       VTV     Vinnai Thandi Varuvaya  2       PVR1
8       QQS     Qarib Qarib Single      3       PVR2
3       YA      Yennai Arinthal 3       PVR1
8       QQS     Qarib Qarib Single      3       PVR2
3       YA      Yennai Arinthal 3       PVR1
9       GA      Golmal Again    4       PVR2
4       VV      Vettayadu Vilayadu      4       PVR1
9       GA      Golmal Again    4       PVR2
4       VV      Vettayadu Vilayadu      4       PVR1
Time taken: 0.301 seconds, Fetched: 20 row(s)
hive (Custom)>
```

Earlier, we inserted **10** rows, now the same command has been executed and the same data is appended to the previous data and we have fetched **20** rows.

Updating the Data in Hive Table

HIVE Code:

UPDATE movie set movie_id = 8 where movie_id = 7;

The above command is used to update a row in Hive table.

```
hive (Custom)>
            >
            > UPDATE movie set movie_id = 8 where movie_id = 7;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is not supported.  Column movie_id.
hive (Custom)>
```

From the above image, we can see that we have received an error message. This means that the Update command is not supported on the columns that are bucketed.

In this table, we have bucketed the **'movie_id'** column and performing the Update operation on the same column, so we have go the error

"FAILED: SemanticException[Error 10302]: Updating values of bucketing columns is not supported. Column clg_id"

Now let's perform the update operation on Non bucketed column,

UPDATE movie set movie_code = 'AM' where movie_id = 1;

```
FAILED: SemanticException [Error 10004]: Line 1:30 Invalid table alias or column refere
e, movie_name, screen_number, screen_loc)
hive (Custom)> UPDATE movie set movie_code = 'AM' where movie_id = 1;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future vers
park, tez) or using Hive 1.X releases.
Query ID = acadgild_20171120122424_ee45ea75-ad3f-4756-a2ac-41f1d9b5f9ae
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 5
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>
```

```
hive (Custom)>
            >
            > select * from movie;
OK
movie.movie_id  movie.movie_code        movie.movie_name        movie.screen_number     movie.screen_loc
10      SS      Secret Superstar        5       PVR2
5       KK      Kakka Kakka     5       PVR1
10      SS      Secret Superstar        5       PVR2
5       KK      Kakka Kakka     5       PVR1
6       DW      Dark Wind       1       PVR2
1       AM      Acham Yenbathu Madamayada       1       PVR1
6       DW      Dark Wind       1       PVR2
1       AM      Acham Yenbathu Madamayada       1       PVR1
7       TS      Tumhari Sulu    2       PVR2
2       VTV     Vinnai Thandi Varuvaya  2       PVR1
7       TS      Tumhari Sulu    2       PVR2
2       VTV     Vinnai Thandi Varuvaya  2       PVR1
8       QQS     Qarib Qarib Single      3       PVR2
3       YA      Yennai Arinthal 3       PVR1
8       QQS     Qarib Qarib Single      3       PVR2
3       YA      Yennai Arinthal 3       PVR1
9       GA      Golmal Again    4       PVR2
4       VV      Vettayadu Vilayadu      4       PVR1
9       GA      Golmal Again    4       PVR2
4       VV      Vettayadu Vilayadu      4       PVR1
Time taken: 0.352 seconds, Fetched: 20 row(s)
hive (Custom)>
```

We have successfully updated the data **movie_code** where the **movie_id =1**. It can be seen above that the ***movie_code*** for the ***movie_id=1*** was **'AYM'** and now it is updated to **'AM'**

Deleting a Row from Hive Table

HIVE Code:

DELETE FROM movie WHERE movie_id=8;

```
hive (Custom)>
            >
            > DELETE FROM movie WHERE movie_id=8;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the fu
park, tez) or using Hive 1.X releases.
Query ID = acadgild_20171120122956_03ed1bc5-8dd2-4066-8a0e-09e7f9efbf6e
Total jobs = 1
```

We have now successfully deleted a row from the Hive table. This can be checked using the command ***select * from movie.*** We can see only **18** rows where our actual data is **20** rows. We can see there is not ***movie_id=8.***

```
hive (Custom)>
            > select * from movie;
OK
movie.movie_id  movie.movie_code        movie.movie_name        movie.screen_number     movie.screen_loc
10      SS      Secret Superstar        5       PVR2
5       KK      Kakka Kakka     5       PVR1
10      SS      Secret Superstar        5       PVR2
5       KK      Kakka Kakka     5       PVR1
6       DW      Dark Wind       1       PVR2
1       AM      Acham Yenbathu Madamayada       1       PVR1
6       DW      Dark Wind       1       PVR2
1       AM      Acham Yenbathu Madamayada       1       PVR1
7       TS      Tumhari Sulu    2       PVR2
2       VTV     Vinnai Thandi Varuvaya  2       PVR1
7       TS      Tumhari Sulu    2       PVR2
2       VTV     Vinnai Thandi Varuvaya  2       PVR1
3       YA      Yennai Arinthal 3       PVR1
3       YA      Yennai Arinthal 3       PVR1
9       GA      Golmal Again    4       PVR2
4       VV      Vettayadu Vilayadu      4       PVR1
9       GA      Golmal Again    4       PVR2
4       VV      Vettayadu Vilayadu      4       PVR1
Time taken: 0.338 seconds, Fetched: 18 row(s)
hive (Custom)>
```

----------***xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx***----------