ASSIGNMENT 8.3

Problem Statement:

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

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.

Note: Hive 0.14 should be installed to implement the hive transaction property.

Solution:

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:

1. Insert
2. Delete
3. 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 or above:

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;
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>
```

If the above properties are not set properly, the ‘Insert’ operation will work but ‘Update’ and ‘Delete’ will not work and you will receive the following error:

FAILED: SemanticException [Error 10294]: Attempt to do update or delete using transaction manager that does not support these operations.

Creating a Table That Supports Hive Transactions:

```
CREATE TABLE smartphones_data
(phone_id INT,
manufacturer STRING,
phone_model STRING,
phone_price INT)
CLUSTERED BY (phone_id) INTO 5 BUCKETS
STORED AS orc
TBLPROPERTIES('transactional'='true');
```

The above syntax will create a table with name ‘*smartphones_data'* and the columns present in the table are ‘phone*_id, 'manufacturer', 'phone_model’ and 'phone_price'. W*e are *bucketing* the table by ‘phone*_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> CREATE TABLE smartphones_data
    > (phone_id INT,
    > manufacturer STRING,
    > phone_model STRING,
    > phone_price INT)
    > CLUSTERED BY (phone_id) INTO 5 BUCKETS
    > STORED AS orc
    > TBLPROPERTIES('transactional'='true');
OK
Time taken: 0.649 seconds
hive> SHOW TABLES;
OK
employee
olympics_data
smartphones_data
temperature_data
temperature_data_new
Time taken: 0.1 seconds, Fetched: 5 row(s)
hive>
```

We have successfully created a table with name '*smartphones_data'* which supports row-level transactions of Hive.

The create table can be checked using the command ***show tables.***

Inserting Data into a Hive Table:

INSERT INTO TABLE smartphones_data VALUES(101, 'Samsung', 'S6 Edge', 30000), (102, 'Apple', 'iPhone X', 89000), (103, 'Motorola', 'G5 Plus', 15000), (104, 'Coolpad', 'Note 3', 9000);

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

```
hive> INSERT INTO TABLE smartphones_data VALUES(101,'Samsung','S6 Edge',30000),(102,'Apple','iPhone X',89000),(103,'Motorola'
,'G5 Plus',15000),(104,'Coolpad','Note 3',9000);
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20171217115926_6d780310-44ac-488c-93bf-e5ee853c8d37
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_1513491304174_0001, Tracking URL = http://localhost:8088/proxy/application_1513491304174_0001/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1513491304174_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 5
2017-12-17 12:00:40,545 Stage-1 map = 0%,  reduce = 0%
2017-12-17 12:01:13,799 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.21 sec
2017-12-17 12:02:17,813 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.21 sec
2017-12-17 12:03:22,744 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.21 sec
2017-12-17 12:04:23,643 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.21 sec
2017-12-17 12:05:28,785 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.21 sec
2017-12-17 12:05:49,250 Stage-1 map = 100%,  reduce = 41%, Cumulative CPU 24.33 sec
2017-12-17 12:05:57,162 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 27.4 sec
2017-12-17 12:07:00,485 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 29.26 sec
2017-12-17 12:08:17,051 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 29.26 sec
2017-12-17 12:09:22,400 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 29.26 sec
2017-12-17 12:10:00,271 Stage-1 map = 100%,  reduce = 99%, Cumulative CPU 143.86 sec
2017-12-17 12:11:22,142 Stage-1 map = 100%,  reduce = 99%, Cumulative CPU 169.76 sec
2017-12-17 12:11:38,842 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 178.44 sec
MapReduce Total cumulative CPU time: 3 minutes 15 seconds 190 msec
Ended Job = job_1513491304174_0001
Loading data to table custom.smartphones_data
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 5   Cumulative CPU: 195.19 sec   HDFS Read: 29422 HDFS Write: 3953 SUCCESS
Total MapReduce CPU Time Spent: 3 minutes 15 seconds 190 msec
OK
Time taken: 767.224 seconds
```

Now, we have successfully inserted the data into the Hive table.

The contents of the table can be viewed using the command **select * from smartphones_data;**

```
hive> SELECT * FROM smartphones_data;
OK
101     Samsung S6 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
104     Coolpad Note 3  9000
```

From the above image, we can see that the data has been inserted successfully into the table.

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

```
hive> SELECT * FROM smartphones_data;
OK
101     Samsung S6 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
104     Coolpad Note 3  9000
101     Samsung S6 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
104     Coolpad Note 3  9000
Time taken: 6.508 seconds, Fetched: 8 row(s)
hive>
```

Updating the Data in Hive Table:

UPDATE smartphones_data

SET phone_id = 105

WHERE phone_id = 104;

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

```
hive> UPDATE smartphones_data
    > SET phone_id = 105
    > WHERE phone_id = 104;
FAILED: SemanticException [Error 10302]: Updating values of bucketing columns is not supported.  Column phone_id.
```

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 ***'phone_id'*** column and performing the Update operation on the same column, so we have got the error:

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

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

UPDATE smartphones_data
SET phone_model = 'S7 Edge'
WHERE manufacturer = 101;

```
hive> UPDATE smartphones_data
    > SET phone_model = 'S7 Edge'
    > WHERE manufacturer = 'Samsung';
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20171217123645_6acdbe56-f6c2-4981-b84f-e975fbef2558
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_1513491304174_0003, Tracking URL = http://localhost:8088/proxy/application_1513491304174_0003/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1513491304174_0003
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5
2017-12-17 12:37:30,420 Stage-1 map = 0%,  reduce = 0%
2017-12-17 12:38:40,823 Stage-1 map = 0%,  reduce = 0%

2017-12-17 12:44:24,629 Stage-1 map = 0%,  reduce = 0%
2017-12-17 12:45:31,360 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 72.77 sec
2017-12-17 12:47:15,468 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 72.77 sec
2017-12-17 12:48:55,174 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 72.77 sec
2017-12-17 12:50:57,080 Stage-1 map = 27%,  reduce = 0%, Cumulative CPU 130.37 sec
2017-12-17 12:51:24,562 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 224.94 sec
2017-12-17 12:52:28,646 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 224.94 sec
2017-12-17 12:53:30,583 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 224.94 sec
2017-12-17 12:54:32,024 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 224.94 sec
2017-12-17 12:55:43,221 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 236.99 sec
2017-12-17 12:55:53,878 Stage-1 map = 100%,  reduce = 55%, Cumulative CPU 243.64 sec
2017-12-17 12:56:56,261 Stage-1 map = 100%,  reduce = 55%, Cumulative CPU 244.48 sec
2017-12-17 12:58:06,516 Stage-1 map = 100%,  reduce = 55%, Cumulative CPU 244.48 sec
2017-12-17 12:58:49,291 Stage-1 map = 100%,  reduce = 99%, Cumulative CPU 326.18 sec
2017-12-17 12:59:55,632 Stage-1 map = 100%,  reduce = 99%, Cumulative CPU 336.87 sec
2017-12-17 13:00:05,665 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 358.12 sec
MapReduce Total cumulative CPU time: 5 minutes 59 seconds 270 msec
Ended Job = job_1513491304174_0003
Loading data to table custom.smartphones_data
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 359.27 sec   HDFS Read: 55845 HDFS Write: 1971 SUCCESS
Total MapReduce CPU Time Spent: 5 minutes 59 seconds 270 msec
OK
Time taken: 1422.942 seconds
```

We have successfully updated the data.

The updated data can be checked using the command ***select * from smartphones_data***

```
hive> SELECT * FROM smartphones_data;
OK
101     Samsung S7 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
104     Coolpad Note 3  9000
101     Samsung S7 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
104     Coolpad Note 3  9000
Time taken: 3.208 seconds, Fetched: 8 row(s)
```

We can see that the data has been updated successfully.

Now let's perform the Delete operation on the same table.

Deleting a Row from Hive Table:

DELETE FROM smartphones_data

WHERE phone_id = 104;

The above command will delete a single row in the Hive table.

```
hive> DELETE FROM smartphones_data
    > WHERE phone_id = 104;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execu
tion engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = acadgild_20171217135153_816a9e06-8a8b-49fa-8981-95fa3c008bcd
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_1513491304174_0004, Tracking URL = http://localhost:8088/proxy/application_1513491304174_0004/
Kill Command = /home/acadgild/hadoop-2.6.0/bin/hadoop job  -kill job_1513491304174_0004
Hadoop job information for Stage-1: number of mappers: 5; number of reducers: 5

2017-12-17 13:52:56,740 Stage-1 map = 0%,  reduce = 0%
2017-12-17 13:54:02,562 Stage-1 map = 0%,  reduce = 0%
2017-12-17 13:55:04,658 Stage-1 map = 0%,  reduce = 0%
2017-12-17 13:56:06,749 Stage-1 map = 0%,  reduce = 0%
2017-12-17 13:57:10,628 Stage-1 map = 0%,  reduce = 0%
2017-12-17 13:58:15,452 Stage-1 map = 0%,  reduce = 0%
2017-12-17 13:59:18,242 Stage-1 map = 0%,  reduce = 0%
2017-12-17 14:00:21,141 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 64.93 sec
2017-12-17 14:01:45,450 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 64.93 sec
2017-12-17 14:04:02,303 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 64.93 sec
2017-12-17 14:05:33,090 Stage-1 map = 0%,  reduce = 0%, Cumulative CPU 64.93 sec
2017-12-17 14:06:24,672 Stage-1 map = 60%,  reduce = 0%, Cumulative CPU 146.2 sec
2017-12-17 14:06:31,806 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 199.98 sec
2017-12-17 14:07:34,818 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 199.98 sec
2017-12-17 14:08:39,103 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 199.98 sec
2017-12-17 14:09:41,445 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 199.98 sec
2017-12-17 14:10:48,121 Stage-1 map = 100%,  reduce = 47%, Cumulative CPU 215.54 sec
2017-12-17 14:10:56,934 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 220.21 sec
2017-12-17 14:12:00,219 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 220.21 sec
2017-12-17 14:13:14,625 Stage-1 map = 100%,  reduce = 67%, Cumulative CPU 220.21 sec
2017-12-17 14:14:10,285 Stage-1 map = 100%,  reduce = 99%, Cumulative CPU 254.89 sec
2017-12-17 14:15:12,277 Stage-1 map = 100%,  reduce = 99%, Cumulative CPU 324.32 sec
2017-12-17 14:15:23,441 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 335.24 sec
MapReduce Total cumulative CPU time: 5 minutes 35 seconds 990 msec
Ended Job = job_1513491304174_0004
Loading data to table custom.smartphones_data
MapReduce Jobs Launched:
Stage-Stage-1: Map: 5  Reduce: 5   Cumulative CPU: 335.99 sec   HDFS Read: 53960 HDFS Write: 1342 SUCCESS
Total MapReduce CPU Time Spent: 5 minutes 35 seconds 990 msec
OK
Time taken: 1426.764 seconds
```

We have now successfully deleted a row from the Hive table. This can be checked using the command select * from smartphones_data.

```
hive> SELECT * FROM smartphones_data;
OK
101     Samsung S7 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
101     Samsung S7 Edge 30000
102     Apple   iPhone X        89000
103     Motorola        G5 Plus 15000
Time taken: 2.931 seconds, Fetched: 6 row(s)
hive>
```

We can see that there is no row with ***phone_id =104***. This means that we have successfully deleted the row from the Hive table.

This is how the transactions or row-wise operations are performed in Hive.