Assignment9.1_task3(Advanced Hive)

Task 3:

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.

This above blog helps us in uderstanding ACID properties. *ACID* stands for Atomicity, Consistency, Isolation, and Durability. Atomicity means, a transaction should complete successfully or else it should fail completely i.e. it should not be left partially. Consistency ensures that any transaction will bring the database from one valid state to another state. Isolation states that every transaction should be independent of each other i.e. one transaction should not affect another. And Durability states that if a transaction is completed, it should be preserved in the database even if the machine state is lost or a system failure might occur.

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.

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

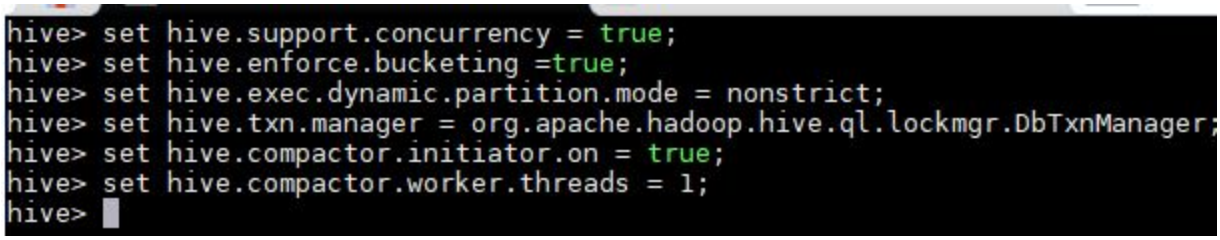
There are numerous limitations with the present transactions available in Hive 0.14. ORC is the file format supported by Hive transaction. It is now essential to have ORC file format for performing transactions in Hive. The table needs to be bucketed in order to support transactions.

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

set hive.support.concurrency = true;

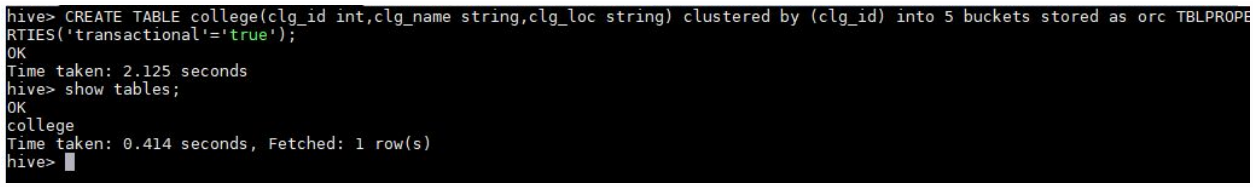
set hive.enforce.bucketing =true;

set hive.exec.dynamic.partition.mode = nonstrict;
set hive.txn.manager = org.apache.hadoop.hive.ql.lockmgr.DbTxnManager;
set hive.compactor.initiator.on = true;
set hive.compactor.worker.threads = 1;



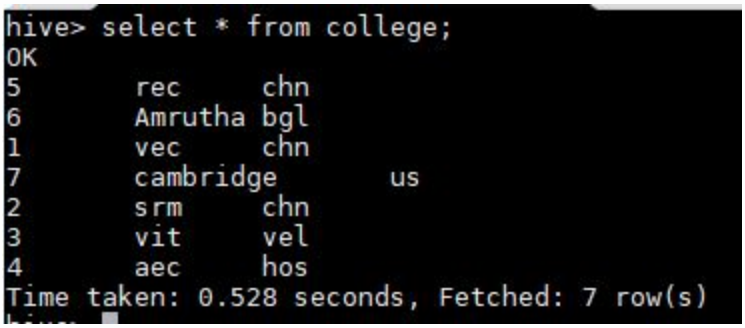
Creating a Table That Supports Hive Transactions

CREATE TABLE college(clg_id int,clg_name string,clg_loc string) clustered by (clg_id) into 5 buckets stored as orc TBLPROPERTIES('transactional'='true');



Inserting Data into a Hive Table

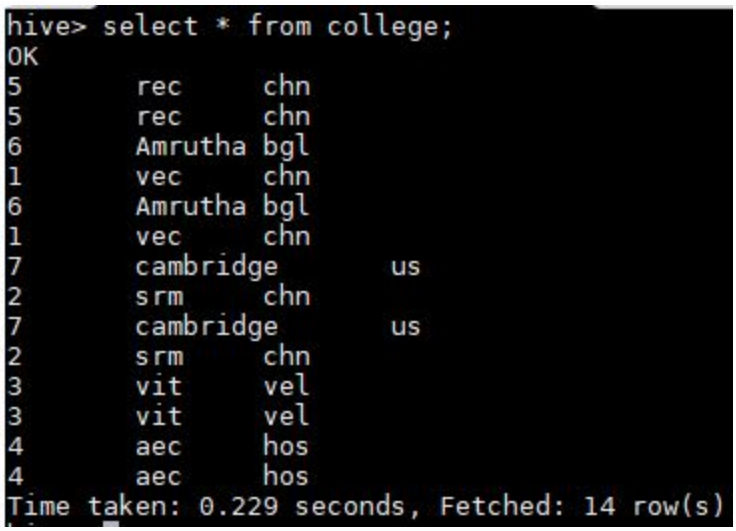
INSERT INTO table college
values(1,'vec','chn'),(2,'srm','chn'),(3,'vit','vel'),(4,'aec','hos'),(5,'rec','chn'),(6,'Amrutha','bgl'),(7,'cambridge','us');



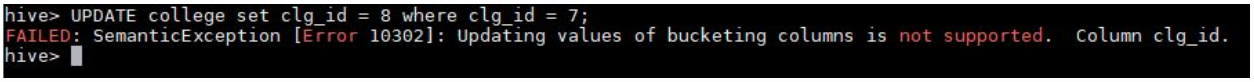
Re-inserting same data again:

----

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



Updating the Data in Hive Table

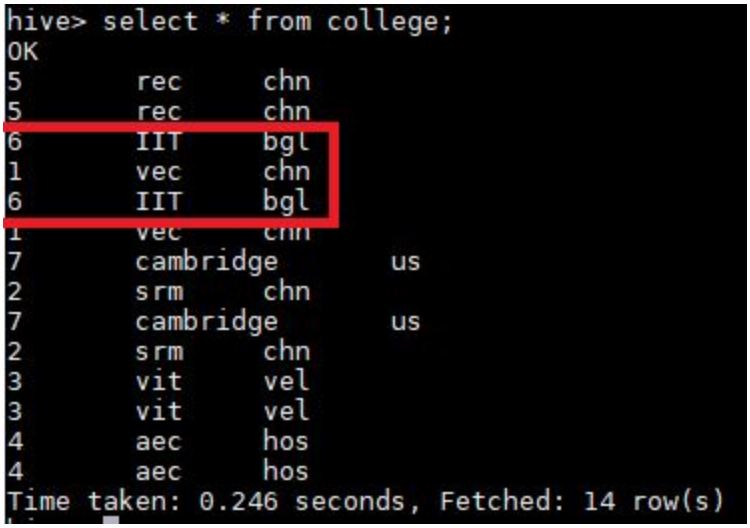


From the above image, we can see that we have received an error message. This means that the Update command is not supported on the columns that are bucketed.

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

Let's perform the update operation on Non bucketed column as below:

UPDATE college set clg_name = 'IIT' where clg_id = 6;

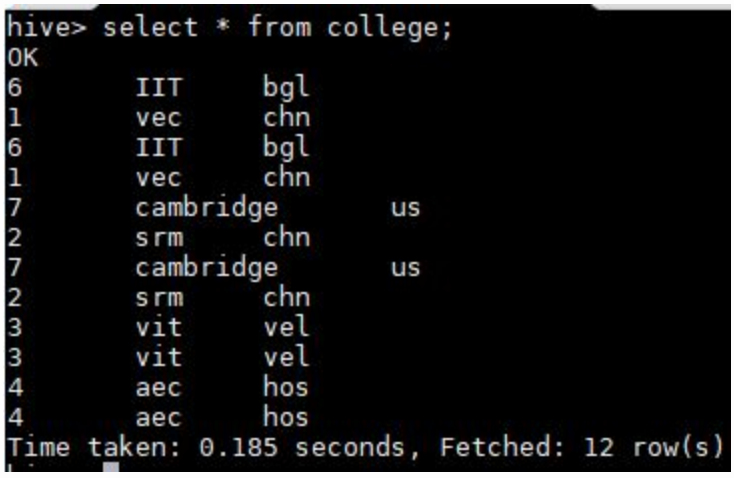


We can see that the data has been updated successfully as IIT.

Deleting a Row from Hive Table

Let's perform the Delete operation on the same table.

delete from college where clg_id=5;



We have now successfully deleted a row from the Hive table. This can be checked using the command select * from college data having clge_id as 5 is deleted.