CALL +91-8048073907

HOME /
HIVE REALTIME USECASES FOR INTERVIEW PREPARATION

# HIVE REALTIME USECASES FOR INTERVIEW PREPARATION

last updated on
29 March, 2019

## HIVE USECASES FOR DEEP LEARNING

*Click here to download the data files for the usecases given below*

### Performing DML statements in Hive

Question is , can we do DML using hive as HDFS doesn't support changes??

1.Create HiveTest table with DML support

```
create table empdml (EmployeeID Int,FirstName String,Designation String, Salary Int,Department String)
clustered by (department) into 3 buckets  stored as orc
TBLPROPERTIES ('transactional'='true');
```

2. Insert

```
insert into table empdml values(1,'Rohit','MD',88000,'Development');
insert into table empdml values(2,'Arun','NJ',75000,'Testing');
```

3. Select

```
SELECT * FROM empdml;
```

4. Update

```
update empdml set salary = 100000 where employeeid = 1;
```

5. Delete

```
delete from empdml where employeeid=2;
```

## ETL Using Hive Queries and functions

Hive can do ETL and ELT, lets explore how can we achieve several business logics in hive by loading staging tables, de-normalized tables with joins, concatenation, summation, aggregations, analytical queries etc.,

Create a database called custdb.
Go inside custdb.
Create the below table in custdb and load customer data

```
Create database custdb;

create table customer(custno string, firstname string, lastname string, age int,profession string)
row format delimited fields terminated by ',';

load data local inpath '/home/hduser/hive/data/custs' into table customer;
```

Create cust transaction table and load the customer and transaction data by joining the 2 tables located in multiple databases such as retail and custdb prefixing schema name, here we are denormalising the tables into a single fat table with added ETL operations and strored as external table

```
create external table cust_trxn (custno int,fullname string,age int,profession string,amount double,product
string,spendby string,agecat varchar(100),modifiedamout float)

row format delimited fields terminated by ',' location '/user/hduser/custtxns';


insert into table cust_trxn
select a.custno,upper(concat(a.firstname,a.lastname)),a.age,a.profession,b.amount,b.product,b.spendby,
case when age<30 then 'low'
when age>=30 and age < 50 then 'middle'
when age>=50 then 'old'
else 'others' end as agecat,

case when spendby= 'credit' then b.amount+(b.amount*0.05) else b.amount end as modifiedamount

from custdb.customer a JOIN retail.txnrecords b

ON a.custno = b.custno;


select * from cust_trxn limit 10;
```

Creating aggregation tables that will be considered as cube and used for quick reporting purposes
Here we are creating a sequence number or a surrogate key column using olap functions like rownumber over and aggregating the age and amount in multiple dimensions with the addition of current_date also.

```
create external table cust_trxn_aggr (seqno int,product string,profession string,level string,sumamt double,
avgamount double,maxamt double,avgage int,currentdate date)
row format delimited fields terminated by ',';

insert overwrite table cust_trxn_aggr
select row_number() over(),product,profession, agecat,
sum(amount),avg(amount),max(amount),avg(age),current_date()
from cust_trxn
group by product,profession, agecat, current_date();
```

## Benchmarking Hive using different file format storage:

The purpose of doing benchmarking is to identify the best functionality or the feature to be used by iterating with different options, here we are going to create textfile, orc and parquet format tables to check the performance between all these tables and the data size it occupied.

```
create table staging_txn(txnno INT, txndate STRING, custno INT, amount DOUBLE,category STRING, product
STRING, city STRING, state STRING, spendby STRING)
row    format    delimited    fields
terminated by ',' stored as textfile
location '/user/hduser/hiveexternaldata';


create table staging_txn(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product
STRING, city STRING, state STRING, spendby STRING)
row format delimited fields terminated by ',' lines terminated by '\n'
stored as textfile;
```

```
LOAD DATA LOCAL INPATH '/home/hduser/hive/data/txns' OVERWRITE INTO TABLE staging_txn;

create table txn_parquet(txnno INT, txndate STRING, custno INT, amount DOUBLE,category STRING, product STRING, city STRING,
state STRING, spendby STRING)
row format delimited fields terminated by ',' lines terminated by '\n'
stored as parquetfile;

Insert into table txn_parquet select txnno,txndate,custno,amount,category, product,city,state,spendby from staging_txn;

create table txn_orc(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STRING,
city STRING, state STRING, spendby STRING)
row format delimited fields terminated by ',' lines terminated by '\n'
stored as orcfile;

Insert into table txn_orc select txnno,txndate,custno,amount,category, product,city,state,spendby from staging_txn;

select count(txnno),category from staging_txn group by category;

select count(txnno),category from txn_orc group by category;

select count(txnno),category from txn_parquet group by category;

select count(txnno),category from staging_txn group by category;

select count(txnno),category from txn_orc group by category;

select count(txnno),category from txn_parquet group by category;
```

## Hive Sqoop Integration usecases:

1. Start mysql service

2. Login to mysql using root

3. Execute the sql file given below in Mysql to load the customer data
**Note: The required files are attached in the links provided in the top of this page:**

```
source /home/hduser/custpayments_ORIG.sql
```

4. Write sqoop command to import data from customers and products table with 2 mappers, with enclosed by " (As we have ',' in the data itself we are importing in sqoop using enclosed by option).

```
sqoop import --connect jdbc:mysql://localhost/custpayments --username root --password root -table customers -m 2 --split-by
customernumber --target-dir /user/hduser/custdata/ --delete-target-dir --enclosed-by '\"';
```

5. Create a hive external table and load the sqoop imported data to the hive table called custmaster. As we have ',' in the data itself we are using quotedchar option below with the csv serde option as given below as example, create the table with all columns.

```
create external table custmaster (customerNumber int,customername string,contactlastname string,contactfirstname string)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
```

```
WITH SERDEPROPERTIES (
 "separatorChar" = ",",
 "quoteChar" = "\"")
LOCATION '/user/hduser/custdata/';
```

6. Copy the payments.txt into hdfs location /user/hduser/paymentsdata/

```
hadoop fs -mkdir -p /user/hduser/paymentsdata
hadoop fs -put payments.txt /user/hduser/paymentsdata
```

7. Create an external table to point the imported payments data location /user/hduser/paymentsdata/.

8. Create another external table called cust_payments in avro format and join the above 2 tables and load using insert select option.

```
create external table payments (custno int,contactfirstname string,contactlastname string)
row format delimited fields terminated by '~'
stored as avro
location '/user/hduser/paymentsavro';

insert into table payments select customernumber,contactfirstname,contactlastname from custmaster;
```

9. Create a view called custpayments_vw to only display customernumber,concatenated contactfirst and contactlastname,creditlimit, amount.

```
create view custpayments as select custno,upper(concat(contactfirstname,' ',contactlastname)) from payments;
```

## Hive Schema Evolution (Hive Dynamic schema):

*In the below usecase we can directly import the DB data using sqoop in avro format and create hive table without defining the columns by using the avro schema file (avsc) created in the below step, which helps hive create dynamic schema if we don't know the schema upfront. This is a very good feature to anlayse and has a good value in the interview if explained.*

Import the Customer data into hdfs using sqoop import with 3 mappers into /user/hduser/custavro location

Download and copy to /home/hduser/ the avro jar (avro-tools-1.8.1.jar) from the below url to extract the schema from the avro data imported in the above step.

http://central.maven.org/maven2/org/apache/avro/avro-tools/1.8.1/

Copy the customer.avsc (avro schema file extracted from the above step into /user/hduser/custavroschema location.

Create a hive table to read the data from avro data location with the table properties mentioning the avsc location.

Get the create table statement of the above table created.

sqoop import -Dmapreduce.job.user.classpath.first=true --connect jdbc:mysql://localhost/custpayments --username root --password root -table customers -m 3 --split-by customernumber --target-dir /user/hduser/custavro --delete-target-dir --as-avrodatafile;

hadoop jar avro-tools-1.8.1.jar getschema /user/hduser/custavro/part-m-00000.avro > /home/hduser/customer.avsc

cat ~/customer.avsc

hadoop fs -put -f customer.avsc /tmp/customer.avsc

create external table customeravro

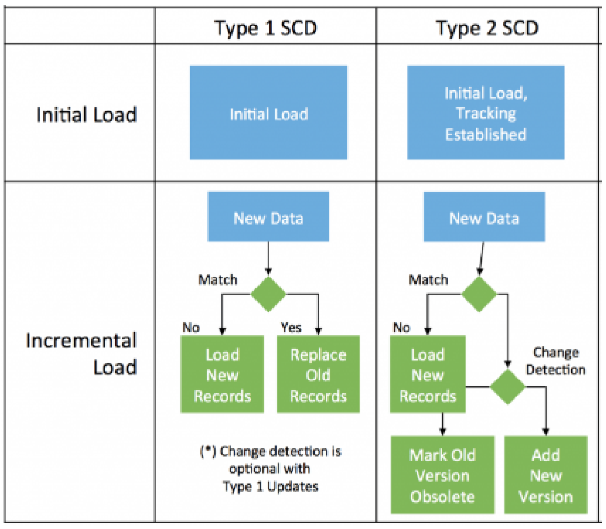
stored as AVRO

location '/user/hduser/custavro'

TBLPROPERTIES('avro.schema.url'='hdfs:///tmp/customer.avsc');

## Slowly Changing Dimension implementation in Hive invoking data using Sqoop:

A Slowly Changing Dimension (SCD) is a dimension that stores and manages both current and historical data over time in a data warehouse. It is considered and implemented as one of the most critical ETL tasks in tracking the history of dimension records.



Source Data readiness:

mysql -u root -p

password: root

mysql

create database stock;

use stock;

create table stockexchange(id integer,exchang varchar(100),company varchar(100),dt date,value float(10,3));

insert into stock.stockexchange values(1,'NYSE','CLI','2019-01-01',35.39),(2,'NYSE','WAG','2019-01-01',24.39),(3,'NYSE','WMT','2019-01-01',145.31),(4,'NYSE','INT','2019-01-01',288.19);

sqoop import --connect jdbc:mysql://localhost/stock --username root --password root --query "select id,exchang,company,value from stockexchange where dt>='2019-01-01' and \$CONDITIONS" --hive-import --hive-overwrite --hive-table stock.managed_stockexchange -m 1 --fields-terminated-by ',' --target-dir /user/hduser/stockexchange

hive> create database stock;

use stock;

drop table ext_stockexchange;

create table stock.ext_stockexchange (id int,ver int,exchang string,company string,value double)

row format delimited fields terminated by ','

location '/user/hduser/stockexchangedata';

Type 1 SCDs - Overwriting

hive

insert into table ext_stockexchange select * from ext_stockexchange where id not in (select id from managed_stockexchange) union select id,1,exchang,company,value from stock.managed_stockexchange;

mysql

insert into stock.stockexchange values(3,'NYSE','WMT','2019-01-03',147.51),(4,'NYSE','INT','2019-01-04',283.77);

sqoop import --connect jdbc:mysql://localhost/stock --username root --password root --query "select id,exchang,company,value from stockexchange where dt>='2019-01-02' and \$CONDITIONS" --hive-import --hive-overwrite --hive-table stock.managed_stockexchange -m 1 --fields-terminated-by ',' --target-dir /user/hduser/stockexchange

select a.* from ext_stockexchange a left outer join managed_stockexchange b on a.id =b.id where b.id is null union select id,1,exchang,company,value from stock.managed_stockexchange;

insert overwrite table ext_stockexchange select * from ext_stockexchange where id not in (select id from managed_stockexchange) union select id,1,exchang,company,value from stock.managed_stockexchange;

Type 2 SCDs - Creating another dimension record

sqoop import --connect jdbc:mysql://localhost/stock --username root --password root --query "select id,exchang,company,value from stockexchange where dt>='2019-01-03' and \$CONDITIONS" --hive-import --hive-overwrite --hive-table stock.managed_stockexchange -m 1 --fields-terminated-by ',' --target-dir /user/hduser/stockexchange

mysql:

insert into stock.stockexchange values(3,'NYSE','WMT','2019-01-04',157.81),(5,'NYSE','INT','2019-01-04',83.77);

Hive:

insert overwrite table ext_stockexchange

select a.id,coalesce(b.ver1+row_number() over(partition by a.id),row_number() over(partition by a.id)) as ver,a.exchang,a.company,a.value from managed_stockexchange a

left outer join (select id,max(ver) as ver1 from ext_stockexchange group by id) as b on a.id=b.id

union

select id,ver,exchang,company,value from ext_stockexchange;

## Hive Static Partition Load Automation script:

Create some sample data in the /home/hduser/hivepart location as given below:

```
cp /home/hduser/hive/data/txns /home/hduser/hivepart/txns_20181212_PADE

cp /home/hduser/hive/data/txns /home/hduser/hivepart/txns_20181212_NY

cp /home/hduser/hive/data/txns /home/hduser/hivepart/txns_20181213_PADE

cp /home/hduser/hive/data/txns /home/hduser/hivepart/txns_20181213_NY
```

Create a shell script namely hivepart.ksh in the /home/hduser/hivepart location, give execute permission and execute as *bash hivepart.ksh /home/hduser/hivepart retail.txnrecsbycatdtreg*

```
set -x

#!/bin/bash

#Script to create hive partition load

#bash hivepart.ksh /home/hduser/hivepart retail.txnrecsbycatdtreg


echo "$0 is starting"

echo "$1 is the source path of txn data sent by the source system"

echo "$2 is the hive table with schema prefixed will be loaded with the above files generically"


rm -f $1/partload.hql


if [ $# -ne 2 ]

then

echo "$0 requires source data path and the target table name to load"

exit 2

fi


echo "$1 is the path"
echo "$2 is the tablename"


for filepathnm in $1/txns_*; do

echo "file with path name is $filepathnm"

filenm=$(basename $filepathnm)

echo "$filenm"

dt=`echo $filenm |awk -F'_' '{print $2}'`

dtfmt=`date -d $dt +'%Y-%m-%d'`

echo $dtfmt

reg=`echo $filenm |awk -F'_' '{print $3}'`

echo $reg

echo "LOAD DATA LOCAL INPATH '$1/$filenm' OVERWRITE INTO TABLE $2 PARTITION
(datadt='$dtfmt',region='$reg');" >> $1/partload.hql

done


#cp /home/hduser/hivepart/partload.hql /home/hduser/partload.hql

echo "loading hive table"

hive -f $1/partload.hql
```

## OTHER INFO

BIGDATA STUDENT CONTENTS

Inceptez Reviews

Inceptez Pig, MapReduce, HBase & Phoenix

1 Data Science Introduction

Introduction to BigData

INCEPTEZ AWS EC2, S3, Kafka & Spark Integration

HIVE WORKOUTS UPDATED

Inceptez Apache Pig Workouts and Usecases

HIVE REALTIME USECASES FOR INTERVIEW PREPARATION

Kafka Nifi Architecture Tutorial Interview

HADOOP HDFS INTERVIEW USECASES

APACHE HIVE ARCHITECTURE DETAIL DOCUMENT sqoop interview questions with incremental usecase

vmware installation steps pdf

Full Stack Cloud Based Bigdata Hadoop & Spark Engineer

## Quick links

Services

Latest Updates

Industry News

More Info

Contact us

## 'Site Sense' keywords

mail

hi team

big data spark developer

3

aws

Open now

Like our Facebook page

Inceptez Technologies
320 likes

Like Page

Share

Be the first of your friends to like this

## Student helpline

+91-8048073907

info@inceptez.com

http://www.inceptez.com

## Share us on

## Subscribe to our updates

your e-mail address

SUBSCRIBE

Site