HBASE

Hbase Practical Session 1

Semi colon is not needed incase of hbase.

hbase is Case Sensitive.

Connect to Hbase = hbase shell

List of tables = list

If you are getting an error, then you need to start some services.

To see status of all services (Run it from terminal)

sudo service --status-all - (Run this on terminal)

These two services should be running -

1. HBase master daemon is not running
2. hbase-regionserver is not running.

Run this commands to make these services on

sudo service hbase-master restart

sudo service hbase-regionserver restart

Create a table

create 'students','personal_details','contact_details','marks'

students = table name

personal_details,contact_details,marks = Column families

Inserting a records

put 'students','student1','personal_details:name','shubham'

put 'students','student1','personal_details:email','shubham@gmail.com'

students = table name

student1 = Row Key

personal_details:name = Column families:Column

shubham = Value

When we were using hive it used to take lot of time to insert it used to invoke map reduce job. But in Hbase it is inserting the records in few seconds.

To see the complete content of table

scan 'students'

scan will give a range of rows or all the rows.

To get specific row

get 'students', 'student1'

get'tablename','rowkey'

To get info about specific Column Family.

get 'students', 'student1', {COLUMN => 'personal_details'}

To get info about specific Column

get 'students', 'student1', {COLUMN => 'personal_details:name'}

Delete Column

delete 'students', 'student1', 'personal_details:email'

Describe a table

describe 'students'

Check if table exists or not

exists 'students'

Drop a table

drop 'students'

To drop a table, we need to disable it first.

The data is present in memstore for now and flush has not happened means hfile is not created. When we disable the table the content of memstore is flushed to the disk and then we can drop the table.

We can only delete the table when the data of it is available in HDFS.

disable 'students'

Hbase Practical Session 2

Create table

create 'cencus','personal','professional'

Insert the data.

Querying the data

scan 'census' (gives a range or records)

scan 'census', (COLUMNS => ['personal: name']} (to get name column from all records in a table)

scan 'census', {COLUMNS> ['personal:name'], LIMIT => 1}

scan 'census', (COLUMNS => ['personal: name'], LIMIT=> 1, STARTROW => "2"}

scan 'census', (COLUMNS => ['personal:name'], LIMIT => 3, STARTROW => "2"}

scan 'census', (COLUMNS => ['personal:name'], STARTROW => "2", STOPROW => "3"}

startrow is included however stoprow is excluded.

STARTROW => "2", STOPROW => "3" --> It is used to mention range of records.

delete 'census', 1, 'personal: marital_status'

exists 'census'

drop 'census'

disable 'census' (removes the index from memory and flushes the recent changes to disk)

Location of HBASE table

separate directory is created for each column family.

hadoop fs -ls /hbase/data/default/cencus/c34d2f1ec34b02a9e19890a7fe202cfc/personal

hadoop fs -ls /hbase/data/default/cencus/c34d2f1ec34b02a9e19890a7fe202cfc/professional

For now, we cannot see what is inside the column families. Because data is not yet flushed to the disk.

We can try that by using disable table option.

Disable flushes the in memory changes to the disk.

Get data based on a filter condition

In HBase, fetching data based on a filtering condition is achieved using filters.

These filters are like Java methods which take two input parameters a logical operator and a comparator.

The logical operator specifies the type of the test, i.e., equals, less than, etc.

The comparator is the number/value against which you wish to compare your record.

Some commonly used filter functions are:

1. ValueFilter

scan 'cencus', {FILTER => "ValueFilter(=,'binary:Maria')"}

In ValueFilter we search for Value.

2. QualifierFilter

scan 'cencus', {FILTER => "QualifierFilter(=,'substring:Name')"}

In ValueFilter we search for Column.

3. FamilyFilter

scan 'cencus', {FILTER => "FamilyFilter(=,'substring:professional')"}

In FamilyFilter we search for Column Family.

Count number of records

count 'cencus'

HIVE-HBASE INTEGRATION

Creating a table which we can access both from hive as well as hbase.

Why Hive?

We want to access the table from hive when we want to do some aggregation like group by, order by.

Why Hbase?

We want to access the table from hbase when we want to do quick searching or when we want to do inserts updates or deletes.

Use case of Hive-Hbase table - Hbase table managed by hive.

1. If on a hbase table we want to do any processing like groupby aggregation or any other kind of map reduce activity then it's better to create a hbase table managed by hive.

2. you are doing some processing in hive. after the processing you want to dump the data in hbase table, for quick searching. In this case you can create hbase table managed by hive.

Steps to create

step 1: we will have a dataset kv1.txt

step 2: we will be creating a hive table based on the structure of kv1.txt file (2 columns)

step 3: we will be loading the file kv1.txt in the hive table created in step 2.

step 4: we will be verifying the data in the table.

step 5: you will create a hbase table managed by hive.

step 6: load the data from normal hive table created in step 2 and put it in special table created in step 5.

Step 1: Make sure file kv1.txt file is placed inside Downloads folder in cloudera VM

(the delimiter in this file is CTRL-A character which is default in delimiter in hive)

Step 2: Create a table in hive.

Step 3: Load the data in hive table.

load data local inpath '/home/cloudera/Desktop/shared1/kv1.txt' into table pokes;

Step 4: Verify the data in hive table.

select * from pokes;

There is no concept of primary key in hive. So, there can be multiple values for same key.

HBase tables cannot have multiple identical keys, only unique keys.

Step 5: Create a Hive-Hbase table.

```
CREATE TABLE hbase_table_1(key int, value string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,cfl:val")
TBLPROPERTIES ("hbase.table.name" = "xyz");
```

Note: The TBLPROPERTIES command is not required, but those new to Hive-HBase integration may find it easier to understand what's going on if Hive and HBase use different names for the same table.

In this example, Hive will recognize this table as "hbase_table_1" and HBase will recognize this table as "xyz".

Data will be stored in HBase in the form of HFiles, and Metadata is stored in Hive.

When you get an error during creating hive-hbase table try this -

```
hadoop fs -rm -r -f /hbase/WALs
hadoop fs -rmdir /hbase/WALs
sudo service hbase-master restart
sudo service hbase-regionserver restart
hbase shell #new shell
```

Step 6: From the Hive prompt, insert data from the Hive table pokes into the Hive-HBase table hbase_table_1

```
INSERT OVERWRITE TABLE hbase_table_1 SELECT * FROM pokes WHERE foo=98;
```