**database operations in Hadoop platform using pig**

1. Make Directory User

[cloudera@localhost ~]$ hdfsdfs -mkdir /tmp/user

2. Make DirectoryZipcodes

[cloudera@localhost ~]$ hdfsdfs -mkdir /tmp/zipcodes

3.Create users.txt and enter the data

[cloudera@localhost ~]$ gedit users.txt

930,28,F,scientists,07310

922,29,F,admin,21114

620,28,M,technician,21114

3.Create zipcodes.txt and enter the data

[cloudera@localhost ~]$ gedit zipcodes.txt

98801,SanDiego,USA

21114,NewYork,USA

07310,London,UK

33556,Delhi,India

4. [cloudera@localhost ~]$hdfsdfs -put zipcodes.txt /tmp/zipcodes

5. [cloudera@localhost ~]$hdfsdfs -put users.txt /tmp/user

6.[cloudera@localhost ~]$geditjoin\_lab.pig

users = LOAD '/tmp/user' USING PigStorage(',') AS (userid:chararray, age:int,gender:chararray, occupation:chararray,zipcode:chararray);

zipcodes = LOAD '/tmp/zipcodes' USING PigStorage(',') as (zipcode:chararray,city:chararray,country:chararray);

jned = JOIN users BY zipcode, zipcodes BY zipcode;

STORE jned INTO '/tmp/joined-data/';

7. [cloudera@localhost ~]$ pig -f join\_lab.pig

8.[cloudera@localhost ~]$hdfsdfs -ls /tmp/joined-data

9.[cloudera@localhost ~]$hdfsdfs -cat /tmp/joined-data/part-r-00000

**Step-1:**

[cloudera@localhost ~]$ hive

Create user table

hive> CREATE TABLE user(useridint, age int, gender string, occupation string, zipcode string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

To create index in hive for table

hive> create index user\_gender\_index on table user1(gender) as 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;

**Step-2:**

Create one file and put data into that file

[cloudera@localhost ~]$ gedit user.txt

**Input text**

121, 21, m, seo analyst, 12147

232, 22, f, HR, 25467

525, 23, f, clerk, 46587

858, 24, m, clerk, 98801

646, 25, f, doctor, 21114

Put the data into hadoop

[cloudera@localhost ~]$ hdfsdfs -put user.txt /user/hive/warehouse/user

**Step-3:**

hive> ALTER INDEX user\_gender\_index ON user REBUILD;

**Step-4:**

hive> select \* from user where gender='M';

**Step-5:**

hive> show indexes on user;

**Step-6:**

hive>drop index user\_gender\_index on user;

**Step-7:**

hive> show indexes on user;

**Create The Database in Hive:**

**Step-1:**

Open terminal and type hive for opening the hive

[cloudera@localhost ~]$hive

hive>

To dispay existed databases

hive> show databases;

**Step-2:**

To create new database

hive> create database lbrce ;

To use created database

hive> use lbrce;

**Step-3:**

To create new table

hive> CREATE TABLE user(useridint, age int, gender string, occupation string, zipcode string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

hive> describe user;

**Step-4:**

Open the another new terminal for creating text input data

[cloudera@localhost ~]$ gedit e.txt

[cloudera@localhost ~]$ hdfsdfs -put e.txt /user/hive/warehouse/emp

**Step-5:**

Open previous terminal for loading the data into table

hive> load data inpath '/user/hive/warehouse/emp/' overwrite into table user;

**Step-6:**

To rename table by using alter command

hive> alter table user rename to emp;

To display table content

hive> select \* from emp;

**Step-7:**

To display particular gender from given table

hive>select \* from emp where gender='F';

**Step-8:**

To drop the existed table by using drop command

hive> drop table emp;

To drop the existed database by using drop command

**hive> drop database lbrce;**

**Step-1:**

Open the eclipse

1. Create the package

File->new->java project give the name Lower

2. Create class

right click on package Lower and

new->class and give the name

**Step-2:**

Add the code

import org.apache.hadoop.io.Text;

importorg.apache.hadoop.hive.ql.exec.UDF;

public class Lower extends UDF

{

public Text evaluate(Text s)

{

if (s == null)

{

return null;

}

return new Text(s.toString().toLowerCase());

}

}

**Step-3:**

Add jar files

right click on Lower go the buildpath->add externaljar files

gotofilesystem->usr/lib/hive/lib

select all except py and php and click ok.

Create a jar file

export->add jar give the name and click ok

**Step-4**:

Open the terminal

[cloudera@localhost ~]$ geditemp

[cloudera@localhost ~]$ hdfsdfs -ls /user/hive/warehouse/emp

ls: `/user/hive/warehouse/emp': No such file or directory

[cloudera@localhost ~]$ hdfsdfs -put emp /user/hive/warehouse/emp

[cloudera@localhost ~]$ hdfsdfs -ls /user/hive/warehouse/emp

[cloudera@localhost ~]$

**Step-5:**

open the another terminal

type hive

hive>create database hive;

hive> use hive;

hive>CREATE TABLE user(empnamestring,salint) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

hive>load data inpath '/user/hive/warehouse/emp' overwrite into table user;

hive> add jar /home/cloudera/hiveudf.jar;

hive>create temporary function tolower as 'Lower';

hive> select tolower(empname) from user;

**Step-1:** Open the Hbase shell.

[cloudera@quickstart ~]$ hbase shell

**Step-2:** Enter list to see the tables available.

hbase(main):001:0> list

TABLE

**Step-3:** See the status of the master servers.

hbase(main):002:0> status

1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load

**Step-4:** See the version.

hbase(main):003:0> version

1.2.0-cdh5.8.0, rUnknown, Thu Jun 16 12:46:57 PDT 2016

**Step-5:** Know the current user.

hbase(main):004:0>whoami

**Step-6:** Create the table student.

hbase(main):005:0> create 'student', 'studentID','dept'

0 row(s) in 1.5020 seconds

=> Hbase::Table – student

**Step-7:** Scan the table created.

hbase(main):006:0> scan 'student'

222333333333ROW COLUMN+CELL

0 row(s) in 0.1700 seconds

**Step-8:** Enter the records into the created table.

hbase(main):007:0> put 'student','1','studentID:id','148W1A05A2'

0 row(s) in 0.1210 seconds

hbase(main):008:0> scan 'student'

ROW COLUMN+CELL

1 column=studentID:id, timestamp=1488982002053, value=148W1A

05A2

1 row(s) in 0.0280 seconds

hbase(main):008:0> put 'student','1','studentID:name','Vamsi'

0 row(s) in 0.0200 seconds

hbase(main):009:0> scan 'student'

ROW COLUMN+CELL

1 column=studentID:id, timestamp=1488982002053, value=148W1A

05A2

1 column=studentID:name, timestamp=1488982231678, value=Vamsi

1 row(s) in 0.0140 seconds

**Step-9:** Now disable the table created.

hbase(main):010:0> disable 'student'

0 row(s) in 2.2600 seconds

**Step-10:** Drop the created table.

hbase(main):011:0> drop 'student'

0 row(s) in 1.2870 seconds

hbase(main):012:0> list

TABLE

0 row(s) in 0.0080 seconds

=> []

hbase(main):013:0>

**Hbase Tables:**

HBase tables are way different [compared to the relational database tables](http://dwgeek.com/hadoop-and-netezza-comparison-netezza-vs-hadoop.html/). HBase organizes all data into tables. Table names are Strings and composed of characters that are easy and safe for use in a file system path. In this article, we will check create tables using HBase shell commands and examples.

We will create the sample tables will couple of columns and insert some sample values to verify the tables.

### Create Tables using HBase Shell

You can create a table using the create command in HBase, table name and the Column Family name are mandatory fields to create table. All the columns in the HBase are grouped into column family.

The HBasedata model design for the table is given below:

|  |  |  |  |
| --- | --- | --- | --- |
| RowKey | personal\_data:name | personal\_data:city | personal\_data:age |
| 1 | Ram | Bengaluru | 25 |

### Syntax to Create Tables using HBase Shell

The syntax to create table in HBase shell is given below:

create '<table\_name>', '<column\_family\_name>'

### Create Tables using HBase Shell Example

Below is the examples of creating ‘personal’ table with column family name personal\_data:name, personal\_data:city and personal\_data:age:

hbase(main):031:0> create 'personal','personal\_data'

0 row(s) in 2.3850 seconds

=> Hbase::Table - personal

### Check Created HBase Table

You can check if the table created using ‘list’ command. Below is the sample output of the list command when executed by passing above created table name as argument:

hbase(main):035:0> list 'personal'

TABLE

personal

1 row(s) in 0.0080 seconds

=> ["personal"]

### Insert sample records to the HBase table

For now to test your created table, follow below steps to insert records to HBase table:

hbase(main):037:0> put 'personal',1,'personal\_data:name','Ram'

0 row(s) in 0.0230 seconds

hbase(main):038:0> put 'personal',1,'personal\_data:city','Bengaluru'

0 row(s) in 0.0050 seconds

hbase(main):039:0> put 'personal',1,'personal\_data:age','25'

0 row(s) in 0.0030 seconds

### Display the Content of HBase Table

You can use the ‘scan’ command to display the content of tables. Follow below step:

hbase(main):040:0> scan 'personal'

ROW COLUMN+CELL

1 column=personal\_data:age, timestamp=1505200304953, value=25

1 column=personal\_data:city, timestamp=1505200291903, value=Bengaluru

1 column=personal\_data:name, timestamp=1505200278059, value=Ram

1 row(s) in 0.0090 seconds

hbase(main):041:0>