##Creating database

hive>

>

> create database random;

OK

Time taken: 32.215 seconds

hive> use random;

OK

Time taken: 0.138 seconds

hive>

##Creating hive table with all the entities provided in the given dataset##

hive> create table if not exists olympic\_table

> (

> athlete string,

> age int,

> country string,

> year int,

> closing\_date string,

> sport string,

> gold int,

> sliver int,

> bronze int,

> total int

> )

> ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

> with serdeproperties

> (

> "seperatorChar" = "\t"

> )

> stored as textfile;

OK

Time taken: 3.518 seconds

hive>

##Loading the dataset into the hive table olympic\_table##

hive>

> load data local inpath '/home/acadgild/Hive\_data.txt'

> into table olympic\_table;

Loading data to table random.olympic\_table

OK

Time taken: 17.835 seconds

hive>

##Doing Partition and Bucketing Operation##

hive>

> set hive.exec.dynamic.partition=true;

hive> set hive.exec.dynamic.partition.mode=nonstrict;

hive> set hive.exec.max.dynamic.partitions=1000;

hive> set hive.exec.max.dynamic.partitions.pernode=1000;

hive> set hive.enforce.bucketing=true;

##Creating table for bucketing process##

hive> create table if not exists olympic\_bucketed

> (

> athlete string,

> age int,

> year int,

> closing\_date string,

> sport string,

> gold int,

> sliver int,

> bronze int,

> total int

> )

> partitioned by (country string)

> clustered by (year)

> sorted by (year)

> into 3 buckets

> row format delimited

> fields terminated by '\t';

OK

Time taken: 1.921 seconds

hive>

##Inserting the dataset into the table olympic\_bucketed##

hive>

> insert overwrite table olympic\_bucketed

> partition (country)

> select athlete,age,year,closing\_date,sport,gold,sliver,bronze,total,country from olympic\_table;

Loaded : 1/1 partitions.

Time taken to load dynamic partitions: 6.058 seconds

Time taken for adding to write entity : 0.046 seconds

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 3 Cumulative CPU: 84.32 sec HDFS Read: 560064 HDFS Write: 717008 SUCCESS

Total MapReduce CPU Time Spent: 1 minutes 24 seconds 320 msec

OK

Time taken: 525.618 seconds

hive>

**Task 1.1**

1. Write a Hive program to find the number of medals won by each country in swimming.

**Script:**

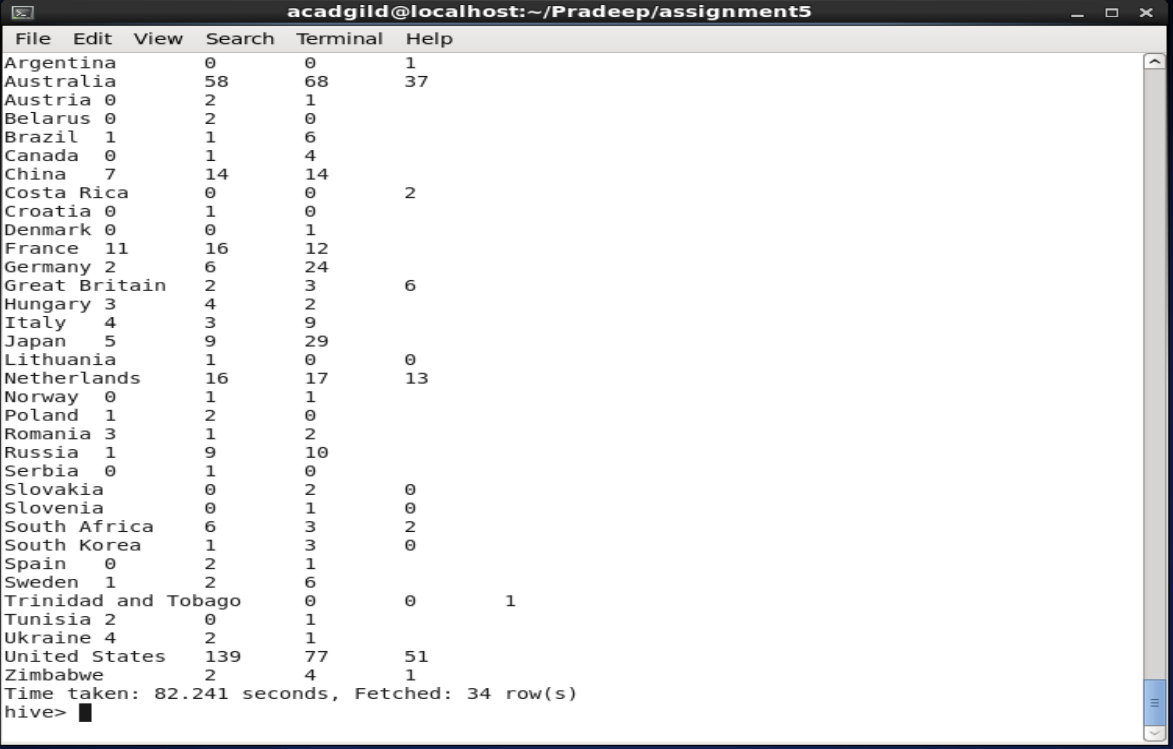
Select country, sum(gold), sum(silver), sum(bronze)

From Olympic\_bucket

Where sport like ‘Swimming’

Group by country;

**OUTPUT:**

****

1. Write a Hive program to find the number of medals that India won year wise.

**Script:**

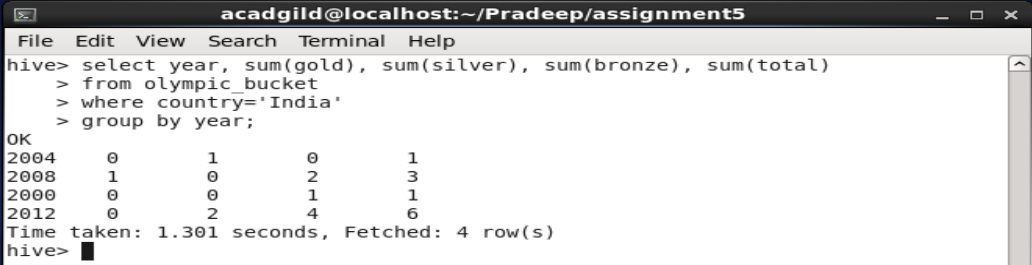
Select year, sum(gold), sum(silver), sum(bronze), sum(total)

From Olympic\_bucket

Where country=‘India’

Group by year;

**OUTPUT:**



1. Write a Hive Program to find the total number of medals each country won.

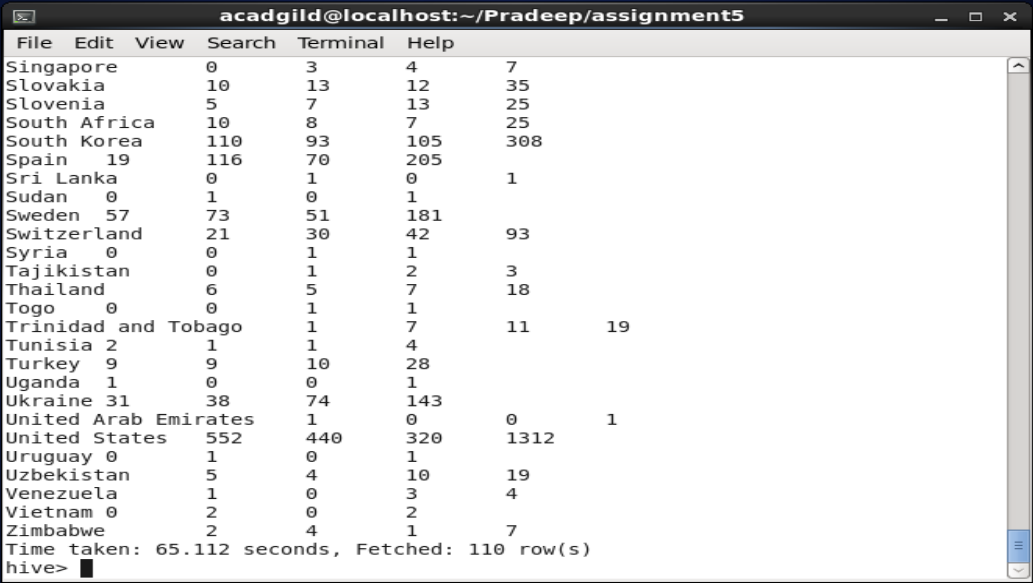
**Script:**

Select country, sum(gold), sum(silver), sum(bronze), sum(total)

From Olympic\_bucket

Group by country;

**OUTPUT:**



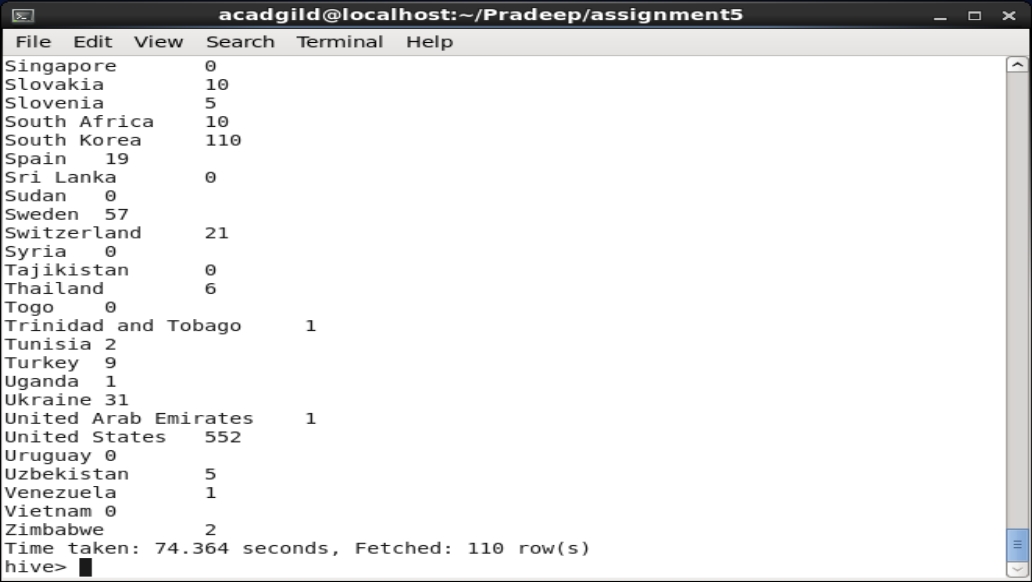
1. Write a Hive program to find the number of gold medals each country won.

**Script:**

Select country, sum(gold)

From Olympic\_bucket

Group by country;

**OUTPUT:** 

**Task 1.2**

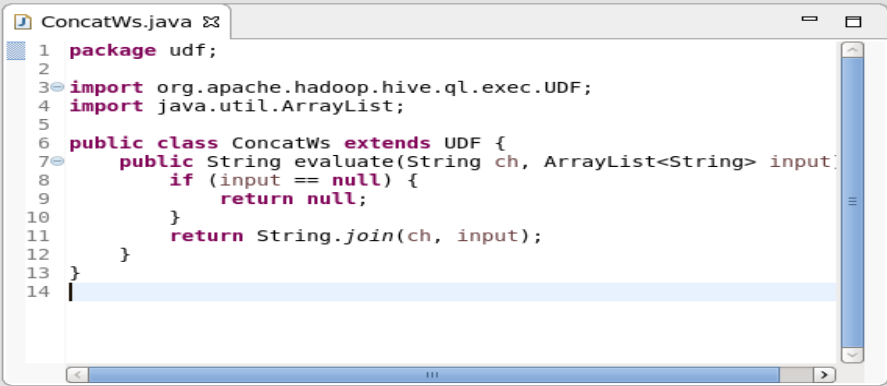
**Task 1.2**

Write a hive UDF that implements functionality of string concat\_ws(string SEP, array<string>). This UDF will accept two arguments, one string and one array of string.

It will return a single string where all the elements of the array are separated by the SEP.

**Java Program:**

Following is the java program that will take two arguments: separator string as ch and array list of string as input and returns the joined string to the calling command:



Then I have created a jar file from the above program with the name **ConcatWS.jar**

Now I have created the emp\_skills.txt file which contains the following information:



Now I have created a table to store the details of employee with the following command:

* Create table emp
* (
* eIdint,
* eName String,
* eSkillsarray<String>
* )

Row format delimited

Fields terminated by ‘\t’

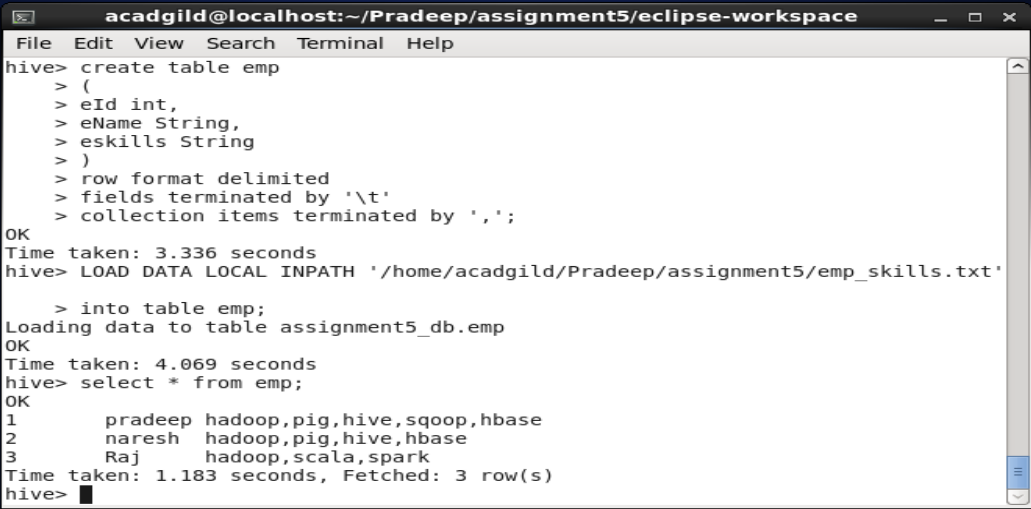
Collection items terminated by ‘,’;

Then I have loaded the data into the emp table with the following command:

* LOAD DATA LOCAL INPATH ‘/home/acadgild/Pradeep/assignment5/emp\_skills.txt’
* Into table emp;

Then I have verified whether the data is loaded as per expectations with the following command:

* Select \* from emp;

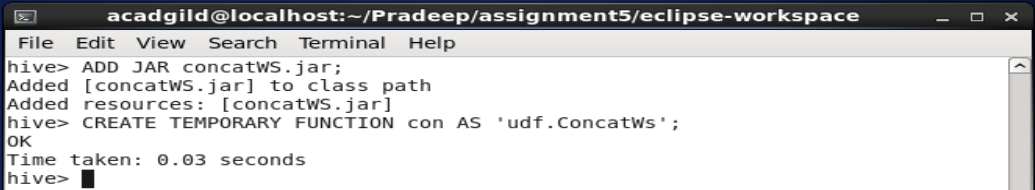


Then I have registered this jar to existing hive session with the following command:

* ADD JAR ConcatWS.jar;

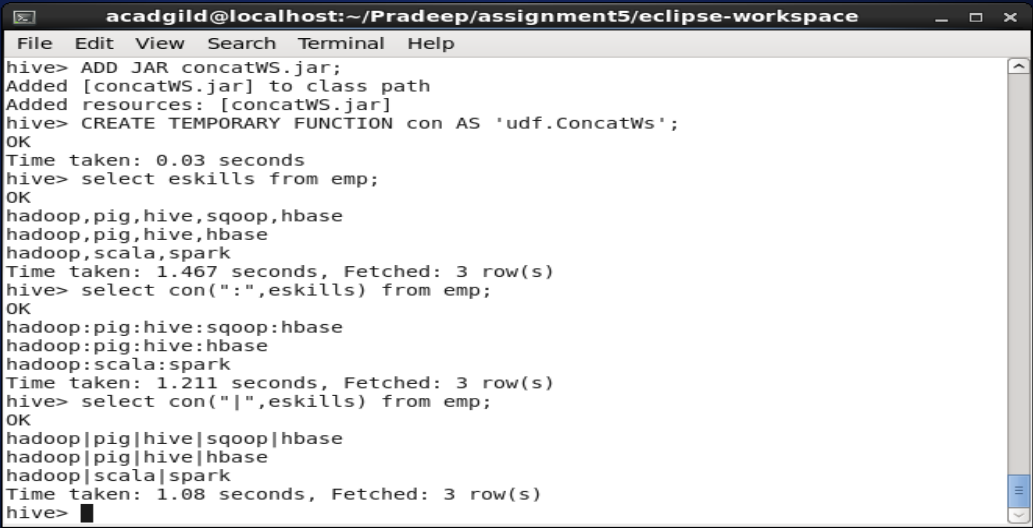
Then I have created a temporary function with the following command:

* CREATE TEMPORARY FUNCTION con AS ‘udf.ConcatWs’;



Then I have executed the following command to work with udf function:

* Select eskills from emp;
* Select con(“:”,eskills) from emp;
* Select con(“|”,eskills) from emp;



As you can see in the above screen shot I am able to add the different separators for the eskills string array that I am passing to the Hiveudf function.

TASK 1.3

\*\*\*\*\*\*\*\*

###Practice of Acadgild hive blog with own data##

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>

hive> create table collage

> (

> clg\_id int,

> clg\_name string,

> clg\_location string

> )

> clustered by (clg\_id)

> into 5 buckets

> stored as orc

> TBLPROPERTIES('transactional' = 'true');

OK

Time taken: 61.056 seconds

hive>

hive> insert into table collage values

> (1,'BIT','Sathy'),

> (2,'CIT','CBE'),

> (3,'GCT','CBE'),

> (4,'VIT','Vellore'),

> (5,'NEC','KVP'),

> (6,'UIT','KVP'),

> (7,'SRM','CHE')

> ;

MapReduce Jobs Launched:

Stage-Stage-1: Map: 1 Reduce: 5 Cumulative CPU: 60.83 sec HDFS Read: 27135 HDFS Write: 3952 SUCCESS

Total MapReduce CPU Time Spent: 1 minutes 0 seconds 830 msec

OK

Time taken: 535.974 seconds

hive>

hive>

> select \* from collage;

OK

5 NEC KVP

6 UIT KVP

1 BIT Sathy

7 SRM CHE

2 CIT CBE

3 GCT CBE

4 VIT Vellore

Time taken: 2.616 seconds, Fetched: 7 row(s)

hive>

##Updating the table with own data##

hive> update collage

> set clg\_name = 'IIT'

> where clg\_id = 3;

hive> select \* from collage;

OK

5 NEC KVP

6 UIT KVP

1 BIT Sathy

7 SRM CHE

2 CIT CBE

3 IIT CBE

4 VIT Vellore

Time taken: 4.568 seconds, Fetched: 7 row(s)

hive>

hive> delete from collage

> where clg\_id= 2;

hive> select \* from collage;

OK

5 NEC KVP

6 UIT KVP

1 BIT Sathy

7 SRM CHE

3 IIT CBE

4 VIT Vellore

Time taken: 1.149 seconds, Fetched: 6 row(s)

hive>

##Task 2.2##

[acadgild@localhost ~]$ start-hbase.sh

localhost: starting zookeeper, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-zookeeper-localhost.localdomain.out

starting master, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-master-localhost.localdomain.out

starting regionserver, logging to /home/acadgild/install/hbase/hbase-1.2.6/logs/hbase-acadgild-1-regionserver-localhost.localdomain.out

You have new mail in /var/spool/mail/acadgild

[acadgild@localhost ~]$ jps

3408 DataNode

10322 HMaster

5330 RunJar

3573 SecondaryNameNode

3877 NodeManager

10439 HRegionServer

10583 Jps

10253 HQuorumPeer

3773 ResourceManager

3311 NameNode

[acadgild@localhost ~]$

[acadgild@localhost ~]$ hbase shell

2018-09-07 23:13:47,120 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/home/acadgild/install/hbase/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/home/acadgild/install/hadoop/hadoop-2.6.5/share/hadoop/common/lib/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

HBase Shell; enter 'help<RETURN>' for list of supported commands.

Type "exit<RETURN>" to leave the HBase Shell

Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0>

##Creating the table in hbase using the following commands##

hbase(main):001:0> create 'clicks' , NAME => 'hits' , VERSIONS => 5

0 row(s) in 3.6610 seconds

=> Hbase::Table - clicks

hbase(main):002:0>

##Adding the record into the 'clicks' table##

hbase(main):002:0> put 'clicks' , '192.168.127.128', 'hits:NoOfHits' , '12'

0 row(s) in 0.7350 seconds

hbase(main):003:0>

##Scanning the table 'clicks'##

hbase(main):003:0> scan 'clicks'

ROW COLUMN+CELL

192.168.127.128 column=hits:NoOfHits, timestamp=1536342491172, value=12

1 row(s) in 0.1560 seconds

hbase(main):004:0>

##Update the same row key for 5 times using the following command##

hbase(main):004:0> put 'clicks' , '192.168.127.128' , 'hits:NoOfHits' , '1'

0 row(s) in 0.0350 seconds

hbase(main):005:0> put 'clicks' , '192.168.127.128' , 'hits:NoOfHits' , '2'

0 row(s) in 0.0350 seconds

hbase(main):006:0> put 'clicks' , '192.168.127.128' , 'hits:NoOfHits' , '3'

0 row(s) in 0.0200 seconds

hbase(main):007:0> put 'clicks' , '192.168.127.128' , 'hits:NoOfHits' , '4'

0 row(s) in 0.0390 seconds

hbase(main):008:0> put 'clicks' , '192.168.127.128' , 'hits:NoOfHits' , '5'

0 row(s) in 0.0260 seconds

hbase(main):009:0>

##Scanning the last Qualifier inside the column family##

hbase(main):009:0> scan 'clicks' , {COLUMN => 'hits:NoOfHits' , VERSION => 5}

ROW COLUMN+CELL

192.168.127.128 column=hits:NoOfHits, timestamp=1536342644317, value=5

1 row(s) in 0.0690 seconds

hbase(main):010:0>

Answer in your own words with example.

1. What is NoSQL data base?

NoSQL is an approach to database design that can accommodate a wide variety of data models, including key-value, document, columnar and graph formats. NoSQL, which stand for "not only SQL," is an alternative to traditional relational databases in which data is placed in tables and data schema is carefully designed before the database is built. NoSQL databases are especially useful for working with large sets of distributed data.

1. How does data get stored in NoSQL database?

Data Storage is not based on a single data model. Most outstanding ones are key-value pair, graph, document, and columnar.

1. What is a column family in HBase?

A column family defines shared features to all columns that are created within them (think of it almost as a sub-table within your larger table). You will notice that HBase columns are composed of a combination of the column family and column qualifier (or column key): 'family:qualifier'  
  
Where the qualifier can be an arbitrary array of bytes, the column family has to be composed of printable characters. Also, on HDFS, the column family is what is stored in human-readable format, example: '/hbase/table/region/<colfamX>'  
  
Remember - all columns families must be created up-front whereas columns can be added on the fly. Hence understanding the design of your data access pattern is crucial when first building your HBase instance.

1. How many maximum number of columns can be added to HBase table?

It has been recommended to keep the number of column families under three. But there is no magic number like this. Why not two? Why not four? Technically, HBase can manage more than three of four column families. However, you need to understand how column families work to make the best use of them.

1. Why columns are not defined at the time of table creation in HBase?

An HBase table is made of column families which are the logical and physical grouping of columns. The columns in one family are stored separately from the columns in another family.

A single column family contains one or more columns, Column families must be defined at table creation time but columns can be added dynamically after table creation (if an insert statement states a column that does not exist for a column family it will create it).

1. How does data get managed in HBase?

The data in the HBASE is managed in such a way that,

* **WAL:** Write Ahead Log is a file on the distributed file system. The WAL is used to store new data that hasn't yet been persisted to permanent storage; it is used for recovery in the case of failure.
* **BlockCache:** is the read cache. It stores frequently read data in memory. Least Recently Used data is evicted when full.
* **MemStore:** is the write cache. It stores new data which has not yet been written to disk. It is sorted before writing to disk. There is one MemStore per column family per region.
* **Hfiles** store the rows as sorted KeyValues on disk.

All the above are sub components of Region server which manages the data in the HBASE.

1. What happens internally when new data gets inserted into HBase table?

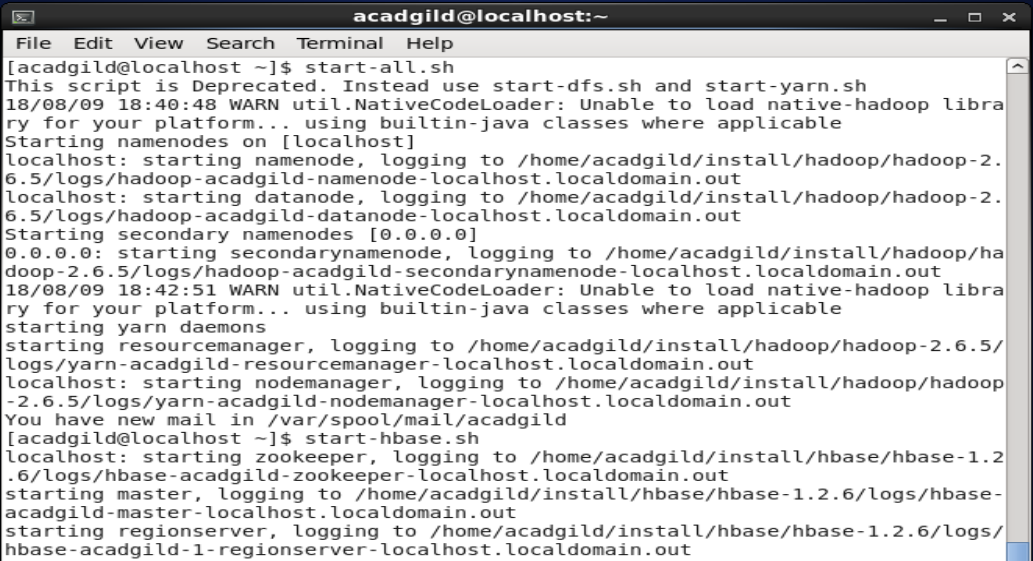
* The client gets the Region server that hosts the META table from ZooKeeper.
* The client will query the .META. Server to get the region server corresponding to the row key it wants to access. The client caches this information along with the META table location.
* It will get the Row from the corresponding Region Server.

For future reads, the client uses the cache to retrieve the META location and previously read row keys. Over time, it does not need to query the META table, unless there is a miss because a region has moved; then it will re-query and update the cache.

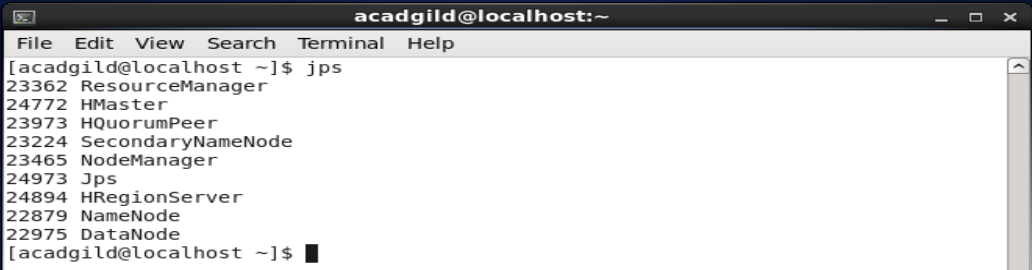
**Task 2.2**

Before working with HBase actual commands we need to start the Hadoop and HBase services with the following commands:

* Start-all.sh
* Start-hbase.sh



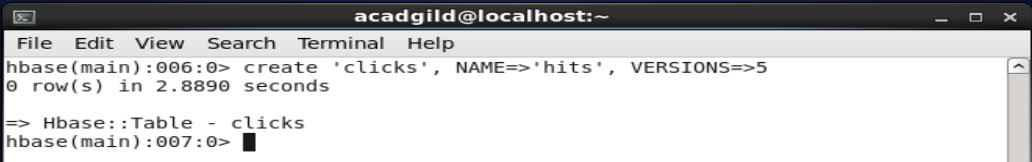
* Jps command to check whether the services started or not



1. Create an HBase table named 'clicks' with a column family 'hits' such that it should be able to store last 5 values of qualifiers inside 'hits' column family.

Following is the command to create a table ‘clicks’ with column family ‘hits’ and having 5 versions:

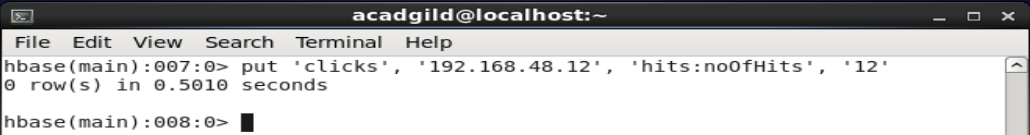
* Create ‘clicks’, NAME=>‘hits’, VERSIONS=>5



2. Add few records in the table and update some of them. Use IP Address as row-key. Scan the table to view if all the previous versions are getting displayed.

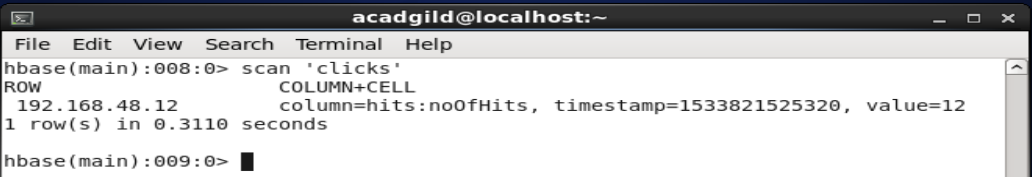
To add records in the clicks table we use below put command and we use row ip address 192.168.48.12 as row key:

* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘12’



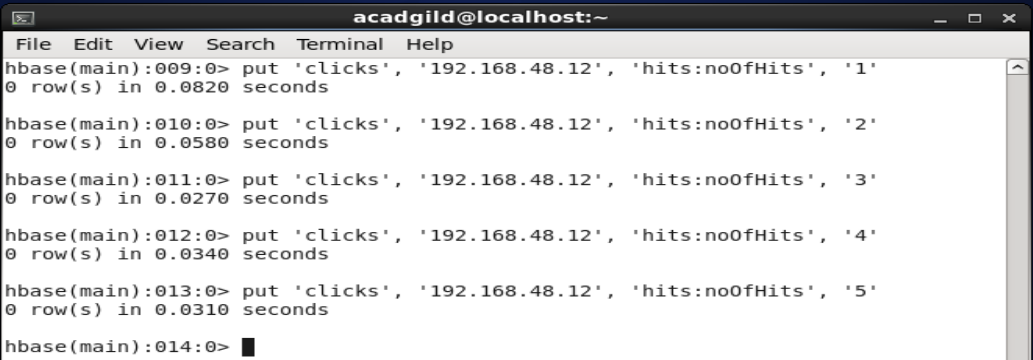
To view the contents of the table use scan command as show below:

* Scan ‘clicks’



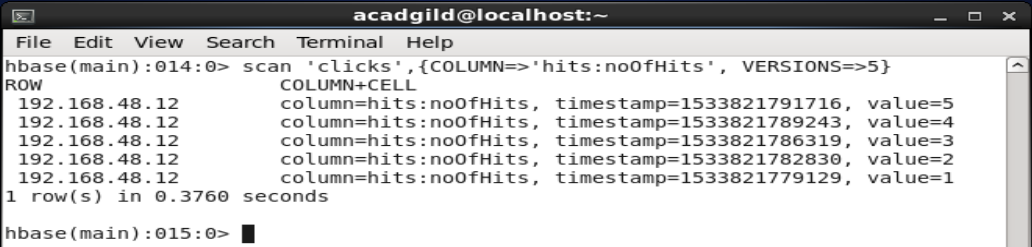
Now we will try to update the same row key for 5 times and check whether we are getting all the version values or not with the help of following commands:

* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘1’
* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘2’
* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘3’
* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘4’
* Put ‘clicks’, ‘192.168.48.12’, ‘hits:noOfHits’, ‘5’



Now we are going to see the last 5 qualifiers inside hits column family with the help of the following command:

* Scan ‘clicks’,{COLUMN=>'hits:NoOfHits',VERSIONS=>5}



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!