HBASE - DESCRIBE & ALTER

describe

This command returns the description of the table. Its syntax is as follows:

```
hbase> describe 'table name'
```

Given below is the output of the describe command on the **emp** table.

```
hbase(main):006:0> describe 'emp'
   DESCRIPTION
      ENABLED

'emp', {NAME ⇒ 'READONLY', DATA_BLOCK_ENCODING ⇒ 'NONE', BLOOMFILTER
⇒ 'ROW', REPLICATION_SCOPE ⇒ '0', COMPRESSION ⇒ 'NONE', VERSIONS ⇒
'1', TTL true

⇒ 'FOREVER', MIN_VERSIONS ⇒ '0', KEEP_DELETED_CELLS ⇒ 'false',
BLOCKSIZE ⇒ '65536', IN_MEMORY ⇒ 'false', BLOCKCACHE ⇒ 'true'}, {NAME
⇒ 'personal

data', DATA_BLOCK_ENCODING ⇒ 'NONE', BLOOMFILTER ⇒ 'ROW',
REPLICATION_SCOPE ⇒ '0', VERSIONS ⇒ '5', COMPRESSION ⇒ 'NONE',
MIN_VERSIONS ⇒ '0', TTL

⇒ 'FOREVER', KEEP_DELETED_CELLS ⇒ 'false', BLOCKSIZE ⇒ '65536',
IN_MEMORY ⇒ 'false', BLOCKCACHE ⇒ 'true'}, {NAME ⇒ 'professional
data', DATA_BLO

CK_ENCODING ⇒ 'NONE', BLOOMFILTER ⇒ 'ROW', REPLICATION_SCOPE ⇒ '0',
VERSIONS ⇒ '1', COMPRESSION ⇒ 'NONE', MIN_VERSIONS ⇒ '0', TTL ⇒
'FOREVER', K

EEP_DELETED_CELLS ⇒ 'false', BLOCKSIZE ⇒ '65536', IN_MEMORY ⇒
'false', BLOCKCACHE ⇒ 'true'}, {NAME ⇒ 'table_att_unset',
DATA_BLOCK_ENCODING ⇒ 'NO

NE', BLOOMFILTER ⇒ 'ROW', REPLICATION_SCOPE ⇒ '0', COMPRESSION ⇒
'NONE', VERSIONS ⇒ '1', TTL ⇒ 'FOREVER', MIN_VERSIONS ⇒ '0',
KEEP_DELETED_CELLS

⇒ 'false', BLOCKSIZE ⇒ '6
```

alter

Alter is the command used to make changes to an existing table. Using this command, you can change the maximum number of cells of a column family, set and delete table scope operators, and delete a column family from a table.

Changing the Maximum Number of Cells of a Column Family

Given below is the syntax to change the maximum number of cells of a column family.

```
hbase> alter 't1', NAME ⇒ 'f1', VERSIONS ⇒ 5
```

In the following example, the maximum number of cells is set to 5.

```
hbase(main):003:0> alter 'emp', NAME ⇒ 'personal data', VERSIONS ⇒ 5
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
```

```
Done.
0 row(s) in 2.3050 seconds
```

Table Scope Operators

Using alter, you can set and remove table scope operators such as MAX_FILESIZE, READONLY, MEMSTORE_FLUSHSIZE, DEFERRED_LOG_FLUSH, etc.

Setting Read Only

Below given is the syntax to make a table read only.

```
hbase>alter 't1', READONLY(option)
```

In the following example, we have made the **emp** table read only.

```
hbase(main):006:0> alter 'emp', READONLY
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 2.2140 seconds
```

Removing Table Scope Operators

We can also remove the table scope operators. Given below is the syntax to remove 'MAX_FILESIZE' from emp table.

```
hbase> alter 't1', METHOD ⇒ 'table_att_unset', NAME ⇒ 'MAX_FILESIZE'
```

Deleting a Column Family

Using alter, you can also delete a column family. Given below is the syntax to delete a column family using alter.

```
hbase> alter ‘ table name ’, ‘delete’ ⇒ ‘ column family ’
```

Given below is an example to delete a column family from the 'emp' table.

Assume there is a table named employee in HBase. It contains the following data:

```
hbase(main):006:0> scan 'employee'

   ROW                    COLUMN+CELL

row1 column = personal:city, timestamp = 1418193767, value = hyderabad

row1 column = personal:name, timestamp = 1418193806767, value = raju

row1 column = professional:designation, timestamp = 1418193767, value = manager

row1 column = professional:salary, timestamp = 1418193806767, value = 50000

1 row(s) in 0.0160 seconds
```

Now let us delete the column family named **professional** using the alter command.

```
hbase(main):007:0> alter 'employee','delete'⇒'professional'
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 2.2380 seconds
```

Now verify the data in the table after alteration. Observe the column family ‘professional’ is no more, since we have deleted it.

```
hbase(main):003:0> scan 'employee'
   ROW              COLUMN &plus; CELL
row1 column = personal:city, timestamp = 14181936767, value = hyderabad

row1 column = personal:name, timestamp = 1418193806767, value = raju

1 row(s) in 0.0830 seconds
```

Adding a Column Family Using Java API

You can add a column family to a table using the method **addColumn** of **HBAseAdmin** class. Follow the steps given below to add a column family to a table.

Step 1

Instantiate the **HBaseAdmin** class.

```
// Instantiating configuration object
Configuration conf = HBaseConfiguration.create();

// Instantiating HBaseAdmin class
HBaseAdmin admin = new HBaseAdmin(conf);
```

Step 2

The **addColumn** method requires a table name and an object of **HColumnDescriptor** class. Therefore instantiate the **HColumnDescriptor** class. The constructor of **HColumnDescriptor** in turn requires a column family name that is to be added. Here we are adding a column family named “contactDetails” to the existing “employee” table.

```
// Instantiating columnDescriptor object

HColumnDescriptor columnDescriptor = new
HColumnDescriptor("contactDetails");
```

Step 3

Add the column family using **addColumn** method. Pass the table name and the **HColumnDescriptor** class object as parameters to this method.

```
// Adding column family
admin.addColumn("employee", new HColumnDescriptor("columnDescriptor"));
```

Given below is the complete program to add a column family to an existing table.

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.MasterNotRunningException;
import org.apache.hadoop.hbase.client.HBaseAdmin;

public class AddColoumn{

   public static void main(String args[]) throws MasterNotRunningException, IOException{

      // Instantiating configuration class.
      Configuration conf = HBaseConfiguration.create();
```

```
      // Instantiating HBaseAdmin class.
      HBaseAdmin admin = new HBaseAdmin(conf);

      // Instantiating columnDescriptor class
      HColumnDescriptor columnDescriptor = new HColumnDescriptor("contactDetails");

      // Adding column family
      admin.addColumn("employee", columnDescriptor);
      System.out.println("coloumn added");
   }
}
```

Compile and execute the above program as shown below.

```
$javac AddColumn.java
$java AddColumn
```

The above compilation works only if you have set the classpath in " **.bashrc** ". If you haven't, follow the procedure given below to compile your .java file.

```
//if "/home/home/hadoop/hbase " is your Hbase home folder then.

$javac -cp /home/hadoop/hbase/lib/*: Demo.java
```

If everything goes well, it will produce the following output:

```
column added
```

Deleting a Column Family Using Java API

You can delete a column family from a table using the method **deleteColumn** of **HBAseAdmin** class. Follow the steps given below to add a column family to a table.

Step1

Instantiate the **HBaseAdmin** class.

```
// Instantiating configuration object
Configuration conf = HBaseConfiguration.create();

// Instantiating HBaseAdmin class
HBaseAdmin admin = new HBaseAdmin(conf);
```

Step2

Add the column family using **deleteColumn** method. Pass the table name and the column family name as parameters to this method.

```
// Deleting column family
admin.deleteColumn("employee", "contactDetails");
```

Given below is the complete program to delete a column family from an existing table.

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.MasterNotRunningException;
import org.apache.hadoop.hbase.client.HBaseAdmin;

public class DeleteColoumn{
```

```
   public static void main(String args[]) throws MasterNotRunningException, IOException{

      // Instantiating configuration class.
      Configuration conf = HBaseConfiguration.create();

      // Instantiating HBaseAdmin class.
      HBaseAdmin admin = new HBaseAdmin(conf);

      // Deleting a column family
      admin.deleteColumn("employee","contactDetails");
      System.out.println("coloumn deleted");
   }
}
```

Compile and execute the above program as shown below.

```
$javac DeleteColumn.java
$java DeleteColumn
```

The following should be the output:

```
column deleted
```