HBASE :
start hbase shell : hbase shell
Exist hbase shell : exit

To start hbase : ./bin/stat-hbase.sh
To stop hbase : ./bin/stop-hbase.sh

Data Definition Language
---
create: - Creates a table
hbase>create '<table name>','<column family>'
ex: create 'emp', 'personal data', 'professional data'

list:- Lists all the tables in HBase.
hbase>list
table1
table2..

disable: - Disables a table.
hbase>disable '<table-name>'
ex: disable 'emp'
note : 1. To delete a table or change its settings, we need to first disable the table using the disable command
2. After disabling the table, we can still check existence of table through list and exists commands. But we cannot scan(retrieve) it
scan 'emp'
ROW COLUMN + CELL
ERROR: emp is disabled.

is_disabled: - To check whether table is disabled or not. (it returns boolean)
hbase> is_disabled 'table name'
ex: is_disabled 'emp'
true

disable_all: - disable all the tables using regex
hbase> disable_all 'r.*' (disable all the tables start with r)

enable: - Enables a table
hbase>enable 'table-name'
ex : enable 'emp'
0 row(s) in 0.4580 seconds
scan 'emp'
ROW COLUMN + CELL
1 column = personal data:city, timestamp = 1417516501, value = hyderabad
1 column = personal data:name, timestamp = 1417525058, value = ramu
1 column = professional data:designation, timestamp = 1417532601, value = manager
1 column = professional data:salary, timestamp = 1417524244109, value = 50000
2 column = personal data:city, timestamp = 1417524574905, value = chennai ..

is_enabled: - Check whether a table is enabled. (it returns boolean)
hbase> is_enabled 'table name'
ex : is_enabled 'emp'
true

exists: - Verifies whether a table does exists or doesn't exists.
hbase> exists 'table name'
ex1 : exists 'emp'

Table emp does exist
0 row(s) in 0.0750 seconds
ex2 : exists 'student'
Table student does not exist
0 row(s) in 0.0480 seconds

drop: - Drops a table from HBase.
hbase> drop 'table name'
ex : drop 'emp'
0 row(s) in 0.0750 seconds

drop_all: - Drops all the tables using regex
ex : drop_all 'r.*'

describe: - description of the table
hbase> describe 'table name'

alter: - Used change the maximum number of cells of a column family,
set and delete table scope operators, and delete a column family from a table
hbase>alter 'table-name', NAME ==> 'column-family-name', VERSIONS ==> 5
ex: to set maximum number of cells to 5.
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: MAX_FILESIZE, READONLY, MEMSTORE_FLUSHSIZE, DEFERRED_LOG_FLUSH
ex : alter 'emp', READONLY
Delete scope operator : alter 'emp', METHOD ==> 'table_att_unset', NAME ==> 'READONLY'

Add Column family :
alter 'tablename', NAME => 'newcolumnfamily',
Deleting a Column Family :
hbase> alter 'table name', 'delete' ==> 'column family'
ex : alter 'employee','delete'==>'professional'

HBaseAdmin and HTableDescriptor are important

```
import java.io.IOException;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.conf.Configuration;

public class DDLTest {
    public static void main(String[] args) throws IOException {

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

        // Instantiating HbaseAdmin class
        HBaseAdmin admin = new HBaseAdmin(con);

        // Instantiating table descriptor class
        HTableDescriptor tableDescriptor = new HTableDescriptor(TableName.valueOf("emp"));
```

```
        // Adding column families to table descriptor
        tableDescriptor.addFamily(new HColumnDescriptor("personal"));
        tableDescriptor.addFamily(new HColumnDescriptor("professional"));
        // Execute the table through admin
        admin.createTable(tableDescriptor);
        System.out.println("Table created ");

         //Getting all the list of tables using HBaseAdmin object
        HTableDescriptor[] tableDescriptor = admin.listTables();
        // printing all the table names.
        for (int i=0; i<tableDescriptor.length;i++ ){
            System.out.println(tableDescriptor[i].getNameAsString());
        }

         // Verifying weather the table is disabled
        Boolean bool = admin.isTableDisabled("emp");
        System.out.println(bool);
        // Disabling the table using HBaseAdmin object
        if(!bool){
            admin.disableTable("emp");
            System.out.println("Table disabled");
        }

         // Verifying weather the table is enabled
        Boolean bool = admin.isTableEnabled("emp");
        // Disabling the table using HBaseAdmin object
        if(!bool){
            admin.enableTable("emp");
            System.out.println("Table Enabled");
        }

          // Verifying the existance of the table
        boolean bool = admin.tableExists("emp");
        System.out.println( bool);

        // disabling table named emp
        admin.disableTable("emp12");
        // Deleting emp
        admin.deleteTable("emp12");

         // Instantiating columnDescriptor class
        HColumnDescriptor columnDescriptor = new HColumnDescriptor("contactDetails");
        // Adding column family
        admin.addColumn("employee", columnDescriptor);

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

Data Manipulation Language

==========

put:- Puts a cell value at a specified column in a specified row in a particular table

put '<table name>','row1','<colfamily:colname>','<value>'

ex : put 'emp','1','personal_data:name','raju'

Note : same command for update as well

put '<table name>','row1','<colfamily:colname>','<new-value>'

get: - Fetches the contents of row or a cell.

get '<table name>','row1'

ex: get 'emp', '1'

Reading a Specific Column : get 'table name', ‘rowid’, {COLUMN ==> ‘column family:column name ’}

delete:- Deletes a cell value in a table
delete '<table name>', '<row>', '<column name >', '<time stamp>'
ex : delete 'emp', '1', 'personal data:city', 1417521848375

deleteall: - Deletes all the cells in a given row
deleteall '<table name>', '<row>'
ex : deleteall 'emp','1'

scan: - view table data
scan '<table name>'
ex : scan 'emp'

count:- Counts the number of rows in a table
count '<table name>'

truncate: - Disables, drops, and recreates a specified table
truncate '<table name>'

HTable, Put,Get, Delete, Result are important classes

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.util.Bytes;

public class InsertData{
    public static void main(String[] args) throws IOException {

        // Instantiating Configuration class
        Configuration config = HBaseConfiguration.create();

        // Instantiating HTable class
        HTable hTable = new HTable(config, "emp");

        // Instantiating Put class
        // accepts a row name.
        Put p = new Put(Bytes.toBytes("row1"));

        // adding values using add() method
        // accepts column family name, qualifier/row name ,value
        p.add(Bytes.toBytes("personal"),
        Bytes.toBytes("name"),Bytes.toBytes("raju"));
        p.add(Bytes.toBytes("personal"),
        Bytes.toBytes("city"),Bytes.toBytes("hyderabad"));

         // Updating a cell value
        p.add(Bytes.toBytes("personal"),
        Bytes.toBytes("city"),Bytes.toBytes("Delih"));

        // Saving the put Instance to the HTable.
        hTable.put(p);
        System.out.println("data inserted");

         // Instantiating Get class
        Get g = new Get(Bytes.toBytes("row1"));
        // Reading the data
        Result result = table.get(g);
        // Reading values from Result class object
        byte [] value = result.getValue(Bytes.toBytes("personal"),Bytes.toBytes("name"));
```

```
        byte [] value1 = result.getValue(Bytes.toBytes("personal"),Bytes.toBytes("city"));
        // Printing the values
        String name = Bytes.toString(value);
        String city = Bytes.toString(value1);
        System.out.println("name: " + name + " city: " + city);

          // Instantiating Delete class
        Delete delete = new Delete(Bytes.toBytes("row1"));
        delete.deleteColumn(Bytes.toBytes("personal"), Bytes.toBytes("name"));
        delete.deleteFamily(Bytes.toBytes("professional"));
        // deleting the data
        table.delete(delete);

          // Instantiating the Scan class
        Scan scan = new Scan();

        // Scanning the required columns
        scan.addColumn(Bytes.toBytes("personal"), Bytes.toBytes("name"));
        scan.addColumn(Bytes.toBytes("personal"), Bytes.toBytes("city"));
        // Getting the scan result
        ResultScanner scanner = table.getScanner(scan);
        // Reading values from scan result
        for (Result result = scanner.next(); result != null; result = scanner.next())
        System.out.println("Found row : " + result);

        // closing HTable
        hTable.close();
    }
}
```