start-hbase.cmd
stop-hbase.cmd

hbase shell
exit

BASIC commands

status → return current status of system
version → return version of HBase
table_help → info for table related commands.
whoami → return.

Creating table

create <table_name>, <column family>
column family : create <tablename>, <column_family1>, <column_family2>
ex emp ← table-name
column families : personal data, professional-data
$create "emp", "presonal data", "professional-data"

* To delete or change its settings, first we need to disable the table.
*disable ing disallow users to perform any operations on table
$disable "table_name"
* To enable table
$enable 'tablename'

$drop "table_name"

to check wether table is enabled or disabled
$ is_enabled "tb_name"
is_disabled "tb_name"
$ dr

disable_all "y*"

→ Inserting new column family:

alter "table_name", {NAME => "New_column_family', VERSIONS => 1}

VERSIONS ↓ no of versions to keep in each cell in a column family

Tell how many previous versions of cell to store.

→ deleting new column family:

alter "table_name", {delete' => "cfname"

→ Renaming ColumnFamily:

① ✗ Not possible.

② → ~~delete~~ and copy one column family to new column family and delete old column family.

---

puting the data

* quantifiers are created while puting the data.

~~# creating table..~~

put "table name", "row-key", ~~column~~ 'column_family': "column_quantifier" } "value"

put "emp", "row1", 'Personal information: name', 'pavan1'

column name = 'column family: column name'

~~column~~

deleting the column family
alter 'tbname', 'delete' => 'column_family name'

exists → verify table exist or not
exits 'tb_name'

* updating column value
put "table name", "row key", "column family: column name", "new_value"

* getting (Reading data from Hbase Shell)
get '<table name>', "row" → complete rowkey

* getting (reading data from specic row, column)
get '<table name>', "row", {COLUMN = 'column family: column name'}

Deleting a specific cell in table.
delete '<table_name>', "row key", "column_name", "time_stamp"

deleting all cells in a row
deleteall "<table_name>", "row"

delete 'tb_name" → delete all the table.

① Specify col name, column family.
② Connect to hbase
③ Create the table

HTableDescriptor → Specify the ~~column~~ hbname, Column family
Connection → Connect hbase
HBAdmin → Create table
HColumnDescriptor → Specifying column

Java Codes

Creation of the table.

```
import org.Apache.hadoop.conf.Configuration;
                hbase.HBaseConfiguration;
                    .client.ConnectionFactory;
                            .Connection;
                            .Admin;
                        .HTableDescriptor;
                        .TableName;
import org.apache.hadoop.Java.util.IOException;
Class CreateTable {                                    throws IOException
    Public static void main(String args[]) {
    Configuration conf = HBaseConfiguration.create();
    Connection conn = HBConnectionFactory.CreateConnection(conf);
    Admin admin = Connect.getAdmin();
TableName Tbl = new HTableDescriptor(TableName.ValueOf("mytable"));
        admin.
    Tbl.addFamily(new HColumnDescriptor(Byte[] bytearray'colfam1'));
if(!admin.TableExists(TableName.ValueOf("mytable"))){
            admin.createTable(Tbl);

    }
```

insertion of data

```
import org.apache.hadoop.conf.Configuration;
                .hbase.HBaseConfiguration;
                .hbase.client.Connection;
                            .ConnectionFactory
                .TableName;
                            .Put;
                            .Table;
                            .Delete;
                .util.Bytes;

Class DeleteTable{                                  throw IOException.
    public static void main(String args[]){
Configuration conf = HBaseConfiguration.create();

Connection conn = ConnectionFactory.createConnection(
                                                    conf);

Table tb1 = conn.getTable(TableName.valueOf("mytab"));
Put put1 = new Put("1");
put1.addFamily(Bytes.toBytes("cf1")) Bytes.toBytes("col1"),
                        Bytes.toBytes("value1"));
tb1.put(put1);
tb1.close()
```

Deleting data.

```
class DeleteTable {                                    throws IOException
    public static void main(String args[]) {
Configuration conf = HBaseConfiguration.create();
Connection conn = ConnectionFactory.createConnection
                                                    (conf);
Table tbl = conn.getTable(TableName.valueOf
                                        ("mtable"));
Delete delete = new Delek(Bytes.toBytes("1"));
delete.addColumn(Bytes.toBytes("colfamily"),
                        Bytes.toBytes("colName"));

tbl.delete(delete);
tbl.close();
```

Retriving data

```
class RetrivingData {
  public static void main(String args[]) throws
                                        IOException{
Configuration conf = HBConfiguration.create();
Connection conn = ConnectionFactory.createConnection
                                        (conf);
Table tbl = conn.getTable(TableName.valueOf(
                                        "mytable"));
//getting row
Get get1 = new Get("1");
//result
Result result= tbl.get(get1);
Byte [] value1 = result.getValue(Bytes
```