HBASE - UPDATE DATA

Updating Data using HBase Shell

You can update an existing cell value using the **put** command. To do so, just follow the same syntax and mention your new value as shown below.

```
put ‘table name’,’row ’,'Column family:column name',’new value’
```

The newly given value replaces the existing value, updating the row.

Example

Suppose there is a table in HBase called **emp** with the following data.

```
hbase(main):003:0> scan 'emp'
 ROW               COLUMN &plus; CELL
row1 column = personal:name, timestamp = 1418051555, value = raju
row1 column = personal:city, timestamp = 1418275907, value = Hyderabad
row1 column = professional:designation, timestamp = 14180555,value = manager
row1 column = professional:salary, timestamp = 1418035791555,value = 50000
1 row(s) in 0.0100 seconds
```

The following command will update the city value of the employee named ‘Raju’ to Delhi.

```
hbase(main):002:0> put 'emp','row1','personal:city','Delhi'
0 row(s) in 0.0400 seconds
```

The updated table looks as follows where you can observe the city of Raju has been changed to ‘Delhi’.

```
hbase(main):003:0> scan 'emp'
  ROW          COLUMN &plus; CELL
row1 column = personal:name, timestamp = 1418035791555, value = raju
row1 column = personal:city, timestamp = 1418274645907, value = Delhi
row1 column = professional:designation, timestamp = 141857555,value = manager
row1 column = professional:salary, timestamp = 1418039555, value = 50000
1 row(s) in 0.0100 seconds
```

Updating Data Using Java API

You can update the data in a particular cell using the **put** method. Follow the steps given below to update an existing cell value of a table.

Step 1: Instantiate the Configuration Class

Configuration class adds HBase configuration files to its object. You can create a configuration object using the **create** method of the **HbaseConfiguration** class as shown below.

```
Configuration conf = HbaseConfiguration.create();
```

Step 2: Instantiate the HTable Class

You have a class called **HTable**, an implementation of Table in HBase. This class is used to communicate with a single HBase table. While instantiating this class, it accepts the configuration object and the table name as parameters. You can instantiate the HTable class as shown below.

```
HTable hTable = new HTable(conf, tableName);
```

Step 3: Instantiate the Put Class

To insert data into HBase Table, the **add** method and its variants are used. This method belongs to **Put**, therefore instantiate the **put** class. This class requires the row name you want to insert the data into, in string format. You can instantiate the **Put** class as shown below.

```
Put p = new Put(Bytes.toBytes("row1"));
```

Step 4: Update an Existing Cell

The **add** method of **Put** class is used to insert data. It requires 3 byte arrays representing column family, column qualifier *columnname*, and the value to be inserted, respectively. Insert data into HBase table using the **add** method as shown below.

```
p.add(Bytes.toBytes("coloumn family "), Bytes.toBytes("column
name"),Bytes.toBytes("value"));
p.add(Bytes.toBytes("personal"),
Bytes.toBytes("city"),Bytes.toBytes("Delih"));
```

Step 5: Save the Data in Table

After inserting the required rows, save the changes by adding the put instance to the **put** method of the HTable class as shown below.

```
hTable.put(p);
```

Step 6: Close HTable Instance

After creating data in HBase Table, close the **HTable** instance using the close method as shown below.

```
hTable.close();
```

Given below is the complete program to update data in a particular table.

```
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 UpdateData{

   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"));

      // 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 Updated");

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

Compile and execute the above program as shown below.

```
$javac UpdateData.java
$java UpdateData
```

The following should be the output:

```
data Updated
```