PRACTICAL NO. 8

Schema design using HBase

What is HBase?

HBase is a column-oriented non-relational database management system that runs on top of Hadoop Distributed File System (HDFS). HBase provides a fault-tolerant way of storing sparse data sets, which are common in many big data use cases. It is well suited for real time data processing or random read/write access to large volumes of data.

Unlike relational database systems, HBase does not support a structured query language like SQL; in fact, HBase isn't a relational data store at all. HBase applications are written in JavaTM much like a typical Apache MapReduce application. HBase does support writing applications in Apache Avro, REST and Thrift.

An HBase system is designed to scale linearly. It comprises a set of standard tables with rows and columns, much like a traditional database. Each table must have an element defined as a primary key, and all access attempts to HBase tables must use this primary key.

Avro, as a component, supports a rich set of primitive data types including: numeric, binary data and strings; and a number of complex types including arrays, maps, enumerations and records.

A sort order can also be defined for the data. HBase relies on ZooKeeper for high-performance coordination. ZooKeeper is built into HBase, but if you're running a production cluster, it's suggested that you have a dedicated ZooKeeper cluster that's integrated with your HBase cluster. HBase works well with Hive, a query engine for batch processing of big data, to enable fault tolerant big data applications.

HBase Shell

HBase contains a shell using which you can communicate with HBase. HBase uses the Hadoop File System to store its data. It will have a master server and region servers. The data storage will be in the form of regions (tables). These regions will be split up and stored in region servers.

The master server manages these region servers and all these tasks take place on HDFS.

Given below are some of the commands supported by HBase Shell.

We can start the HBase interactive shell using "HBase shell" command as shown below.

$ hbase shell

```
[cloudera@quickstart ~]$ hbase shell
2022-03-24 19:46:39,804 INFO  [main] Configuration.deprecation: hadoop.native.li
b is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.0.0-cdh5.4.2, rUnknown, Tue May 19 17:07:29 PDT 2015

hbase(main):001:0>
```

Restart HBase services if this is not running on terminal

$ sudo su – This commands is to become super user

$ service hbase-master restart – This commands is to restart hbase-master services

$ service hbase-regionserver restart – This commands is to restart hbase-regionserver Services

Once these commands run successfully then Open the browser and refresh the page and see all the HBase serveries will be restarted.

```
[cloudera@quickstart ~]$ sudo su
[root@quickstart cloudera]# service hbase-master restart
no master to stop because kill -0 of pid 2838 failed with status 1
Stopped HBase master daemon:                               [  OK  ]
starting master, logging to /var/log/hbase/hbase-hbase-master-quickstart.clouder
a.out
Started HBase master daemon (hbase-master):                [  OK  ]
[root@quickstart cloudera]# service hbase-regionserver restart
Restarting Hadoop HBase regionserver daemon: Stopping Hadoop HBase regionserver
daemon: no regionserver to stop because kill -0 of pid 3948 failed with status 1
hbase-regionserver.
Starting Hadoop HBase regionserver daemon: starting regionserver, logging to /va
r/log/hbase/hbase-hbase-regionserver-quickstart.cloudera.out
hbase-regionserver.
[root@quickstart cloudera]#
```

Check the shell functioning before proceeding further. Use the list command for this purpose. List is a command used to get the list of all the tables in HBase. It lists all the tables in HBase.

hbase(main):001:0> list

```
hbase(main):002:0> list
TABLE
0 row(s) in 0.4180 seconds

=> []
hbase(main):003:0>
```

Schema Design using HBase

1)Creating a Table using HBase Shell

```
hbase(main):003:0> create 'customer','address','order'
0 row(s) in 0.6330 seconds

=> Hbase::Table - customer
hbase(main):004:0> list
TABLE
customer
1 row(s) in 0.0070 seconds

=> ["customer"]
hbase(main):005:0>
```

2)Put: Inserts a new record into the table with row identified by 'row..'

```
hbase(main):032:0> put 'customer','Akshata','address:city','Mumbai'
0 row(s) in 0.0090 seconds

hbase(main):033:0> put 'customer','Akshata','address:state','Maharashtra'
0 row(s) in 0.0070 seconds

hbase(main):034:0> put 'customer','Akshata','address:country','India'
0 row(s) in 0.0110 seconds

hbase(main):035:0> put 'customer','Akshata','order:number','No-20'
0 row(s) in 0.0070 seconds

hbase(main):036:0> put 'customer','Akshata','order:amount','Rs.1020'
0 row(s) in 0.0090 seconds
```

```
hbase(main):037:0> put 'customer','Vishakha','address:city','Nashik'
0 row(s) in 0.0080 seconds

hbase(main):038:0> put 'customer','Vishakha','address:state','Maharashtra'
0 row(s) in 0.0080 seconds

hbase(main):039:0> put 'customer','Vishakha','address:country','India'
0 row(s) in 0.0090 seconds

hbase(main):040:0> put 'customer','Vishakha','order:number','No-40'
0 row(s) in 0.0070 seconds

hbase(main):041:0> put 'customer','Vishakha','order:amount','Rs.1000'
0 row(s) in 0.0090 seconds

hbase(main):042:0>
```

3)Get: Returns the records matching the row identifier provided in the table

a) a) get 'customer', 'Akshata’

```
hbase(main):042:0> get 'customer','Akshata'
COLUMN                CELL
 address:city         timestamp=1648178061679, value=Mumbai
 address:country      timestamp=1648178112729, value=India
 address:state        timestamp=1648178091382, value=Maharashtra
 order:amount         timestamp=1648178136924, value=Rs.1020
 order:number         timestamp=1648178126641, value=No-20
5 row(s) in 0.0120 seconds

hbase(main):043:0>
```

b) Additional parameters to get only address details

```
hbase(main):043:0> get 'customer','Akshata','address'
COLUMN                CELL
 address:city         timestamp=1648178061679, value=Mumbai
 address:country      timestamp=1648178112729, value=India
 address:state        timestamp=1648178091382, value=Maharashtra
3 row(s) in 0.0100 seconds

hbase(main):044:0>
```

c) Additional parameters to get only city details

```
hbase(main):044:0> get 'customer','Akshata','address:city'
COLUMN                CELL
 address:city         timestamp=1648178061679, value=Mumbai
1 row(s) in 0.0070 seconds

hbase(main):045:0>
```

4)Scan :The scan command is used to view the data in HTable.

Using the scan command, you can get the table data.

scan 'customer'

When we execute above commands in HBase then we will be getting all the table "customer" contents along with additional parameters like timestamp as show in below screenshot.

```
hbase(main):046:0> scan 'customer'
ROW                   COLUMN+CELL
 Akshata              column=address:city, timestamp=1648178061679, value=Mumbai
 Akshata              column=address:country, timestamp=1648178112729, value=India
 Akshata              column=address:state, timestamp=1648178091382, value=Maharashtr
                      a
 Akshata              column=order:amount, timestamp=1648178136924, value=Rs.1020
 Akshata              column=order:number, timestamp=1648178126641, value=No-20
 Vishakha             column=address:city, timestamp=1648178195630, value=Nashik
 Vishakha             column=address:country, timestamp=1648178227806, value=India
 Vishakha             column=address:state, timestamp=1648178212927, value=Maharashtr
                      a
 Vishakha             column=order:amount, timestamp=1648178328240, value=Rs.1000
 Vishakha             column=order:number, timestamp=1648178258553, value=No-40
2 row(s) in 0.0280 seconds

hbase(main):047:0>
```

5)Delete -Using the delete command, you can delete a specific cell in a table.

```
hbase(main):047:0> delete 'customer','Akshata','address:country'
0 row(s) in 0.0060 seconds

hbase(main):048:0> scan 'customer'
ROW                   COLUMN+CELL
 Akshata              column=address:city, timestamp=1648178061679, value=Mumbai
 Akshata              column=address:state, timestamp=1648178091382, value=Maharashtr
                      a
 Akshata              column=order:amount, timestamp=1648178136924, value=Rs.1020
 Akshata              column=order:number, timestamp=1648178126641, value=No-20
 Vishakha             column=address:city, timestamp=1648178195630, value=Nashik
 Vishakha             column=address:country, timestamp=1648178227806, value=India
 Vishakha             column=address:state, timestamp=1648178212927, value=Maharashtr
                      a
 Vishakha             column=order:amount, timestamp=1648178328240, value=Rs.1000
 Vishakha             column=order:number, timestamp=1648178258553, value=No-40
2 row(s) in 0.0280 seconds

hbase(main):049:0>
```

6)Alter - This command alters the column family schema. To understand what exactly it does, we have explained it here with an example.

```
hbase(main):053:0> alter 'customer','delete' => 'address'
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 2.2070 seconds

hbase(main):054:0>
```

```
hbase(main):054:0> scan 'customer'
ROW                   COLUMN+CELL
 Akshata              column=order:amount, timestamp=1648178136924, value=Rs.1020
 Akshata              column=order:number, timestamp=1648178126641, value=No-20
 Vishakha             column=order:amount, timestamp=1648178328240, value=Rs.1000
 Vishakha             column=order:number, timestamp=1648178258553, value=No-40
2 row(s) in 0.0150 seconds

hbase(main):055:0>
```

7)Describe - This command describes the named table.

```
hbase(main):055:0> desc 'customer'
Table customer is ENABLED
customer
COLUMN FAMILIES DESCRIPTION
{NAME => 'order', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCO
PE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVE
R', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCA
CHE => 'true'}
1 row(s) in 0.0410 seconds

hbase(main):056:0>
```

8)Versions –

A {row, column, version} tuple exactly specifies a cell in HBase. In the Apache HBase you can

have many cells where row and columns are same but differs only in version values. A version is a timestamp values is written alongside each value. By default, the timestamp values represent the time on the RegionServer when the data was written, but you can change the default HBase setting and specify a different timestamp value when you put data into the cell.

```
hbase(main):001:0> create 'customer1',{NAME=>'address',VERSIONS=>3}
0 row(s) in 0.5680 seconds

=> Hbase::Table - customer1
hbase(main):002:0> list
TABLE
customer
customer1
2 row(s) in 0.0190 seconds

=> ["customer", "customer1"]
```

9)Count - You can count the number of rows of a table using the count

command. Its syntax is as follows:

count 'customer'

```
=> ["customer", "customer1"]
hbase(main):003:0> count 'customer'
2 row(s) in 0.0970 seconds

=> 2
hbase(main):004:0>
```

10)Alter -Update the version number for already existing columns family

alter 'customer' , NAME => 'address', VERSIONS =>5

Again using describe command to see version is updated

>desc ‘customer’

```
hbase(main):001:0> alter 'customer',NAME=>'address',VERSIONS=>5
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 2.8110 seconds

hbase(main):002:0> desc 'customer'
Table customer is ENABLED
customer
COLUMN FAMILIES DESCRIPTION
{NAME => 'address', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SC
OPE => '0', COMPRESSION => 'NONE', VERSIONS => '5', TTL => 'FOREVER', MIN_VERSIONS => '
0', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCAC
HE => 'true'}
{NAME => 'order', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOP
E => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER'
, KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE
 => 'true'}
2 row(s) in 0.0650 seconds

hbase(main):003:0>
```

Below are example of adding version of address:city

Currently below is the data in ‘customer’. We will be adding more information in it one byone.

>scan ‘customer’

```
hbase(main):003:0> scan 'customer'
ROW                     COLUMN+CELL
 Akshata                column=address:city, timestamp=1648178061679, value=Mumbai
 Akshata                column=address:state, timestamp=1648178091382, value=Maharashtra
 Akshata                column=order:amount, timestamp=1648178136924, value=Rs.1020
 Akshata                column=order:number, timestamp=1648178126641, value=No-20
 Vishakha               column=address:city, timestamp=1648178195630, value=Nashik
 Vishakha               column=address:country, timestamp=1648178227806, value=India
 Vishakha               column=address:state, timestamp=1648178212927, value=Maharashtra
 Vishakha               column=order:amount, timestamp=1648178328240, value=Rs.1000
 Vishakha               column=order:number, timestamp=1648178258553, value=No-40
2 row(s) in 0.1610 seconds

hbase(main):004:0>
```

```
hbase(main):004:0> put 'customer','Ask','address:city','Pune'
0 row(s) in 0.1450 seconds

hbase(main):005:0> put 'customer','Ask','address:city','Delhi'
0 row(s) in 0.0380 seconds

hbase(main):006:0> put 'customer','Ask','address:city','Hydrabad'
0 row(s) in 0.0540 seconds
```

Now we have 4 values for customer Ask for city

>scan 'customer', {COLUMN=> 'address:city', VERSIONS => 2 }

After executing above commands its giving us 2 version of address:city records as shown in below screenshot.

>scan 'customer', {COLUMN=> 'address:city', VERSIONS => 3}

>scan 'customer', {COLUMN=> 'address:city', VERSIONS => 4}

>scan 'customer', {COLUMN=> 'address:city', VERSIONS => 5}

```
hbase(main):007:0> scan 'customer',{COLUMN=>'address:city',VERSIONS=>2}
ROW                   COLUMN+CELL
 Akshata              column=address:city, timestamp=1648178061679, value=Mumbai
 Ask                  column=address:city, timestamp=1648179900063, value=Hydrabad
 Ask                  column=address:city, timestamp=1648179879311, value=Delhi
 Vishakha             column=address:city, timestamp=1648178195630, value=Nashik
3 row(s) in 0.0250 seconds

hbase(main):008:0> scan 'customer',{COLUMN=>'address:city',VERSIONS=>3}
ROW                   COLUMN+CELL
 Akshata              column=address:city, timestamp=1648178061679, value=Mumbai
 Ask                  column=address:city, timestamp=1648179900063, value=Hydrabad
 Ask                  column=address:city, timestamp=1648179879311, value=Delhi
 Ask                  column=address:city, timestamp=1648179858411, value=Pune
 Vishakha             column=address:city, timestamp=1648178195630, value=Nashik
3 row(s) in 0.0540 seconds

hbase(main):009:0> scan 'customer',{COLUMN=>'address:city',VERSIONS=>4}
ROW                   COLUMN+CELL
 Akshata              column=address:city, timestamp=1648178061679, value=Mumbai
 Ask                  column=address:city, timestamp=1648179900063, value=Hydrabad
 Ask                  column=address:city, timestamp=1648179879311, value=Delhi
 Ask                  column=address:city, timestamp=1648179858411, value=Pune
 Vishakha             column=address:city, timestamp=1648178195630, value=Nashik
3 row(s) in 0.0440 seconds

hbase(main):010:0> scan 'customer',{COLUMN=>'address:city',VERSIONS=>5}
ROW                   COLUMN+CELL
 Akshata              column=address:city, timestamp=1648178061679, value=Mumbai
 Ask                  column=address:city, timestamp=1648179900063, value=Hydrabad
 Ask                  column=address:city, timestamp=1648179879311, value=Delhi
 Ask                  column=address:city, timestamp=1648179858411, value=Pune
 Vishakha             column=address:city, timestamp=1648178195630, value=Nashik
3 row(s) in 0.0370 seconds

hbase(main):011:0>
```

scan 'customer', {VERSIONS => 5 } – This for all columns not for any specific column

```
hbase(main):011:0> scan 'customer',{VERSIONS=>5}
ROW                   COLUMN+CELL
 Akshata              column=address:city, timestamp=1648178061679, value=Mumbai
 Akshata              column=address:state, timestamp=1648178091382, value=Maharashtra
 Akshata              column=order:amount, timestamp=1648178136924, value=Rs.1020
 Akshata              column=order:number, timestamp=1648178126641, value=No-20
 Ask                  column=address:city, timestamp=1648179900063, value=Hydrabad
 Ask                  column=address:city, timestamp=1648179879311, value=Delhi
 Ask                  column=address:city, timestamp=1648179858411, value=Pune
 Vishakha             column=address:city, timestamp=1648178195630, value=Nashik
 Vishakha             column=address:country, timestamp=1648178227806, value=India
 Vishakha             column=address:state, timestamp=1648178212927, value=Maharashtra
 Vishakha             column=order:amount, timestamp=1648178328240, value=Rs.1000
 Vishakha             column=order:number, timestamp=1648178258553, value=No-40
3 row(s) in 0.0430 seconds

hbase(main):012:0>
```

11)Disable -This command will start disabling the named table If table needs to be deleted or dropped, it has to disable first

Syntax: disable <tablename>

disable 'customer'

```
hbase(main):012:0> disable 'customer'
0 row(s) in 1.3950 seconds

hbase(main):013:0>
```

12)Drop– It drops a table from HBase. Drop means complete deletion of table. For this first disable the table then drop it.

drop 'customer'

```
hbase(main):013:0> drop 'customer'
0 row(s) in 0.3220 seconds

hbase(main):014:0>
```

Sparks by example

1) **Create Table**

```
hbase(main):014:0> create 'emp','office'
0 row(s) in 0.4910 seconds
```

2) **List table**

This returns all users table in the database, you can also use with a regular expression to filter the results

```
hbase(main):015:0> list
TABLE
customer1
emp
2 row(s) in 0.0170 seconds

=> ["customer1", "emp"]
hbase(main):016:0>
```

3) Describe Table

```
hbase(main):016:0> describe 'emp'
Table emp is ENABLED
emp
COLUMN FAMILIES DESCRIPTION
{NAME => 'office', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCO
PE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER
', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACH
E => 'true'}
1 row(s) in 0.0240 seconds

hbase(main):017:0>
```

4) Insert Data to Table

```
hbase(main):017:0> put 'emp','1','office:name','Akshata'
0 row(s) in 0.0500 seconds

hbase(main):018:0> put 'emp','2','office:name','Kajal'
0 row(s) in 0.0080 seconds

hbase(main):019:0> put 'emp','3','office:name','Sarita'
0 row(s) in 0.0550 seconds

hbase(main):020:0> put 'emp','2','office:gender','F'
0 row(s) in 0.0190 seconds
```

```
hbase(main):022:0> put 'emp','2','office:gender','F'
0 row(s) in 0.0070 seconds

hbase(main):023:0> put 'emp','2','office:age','21'
0 row(s) in 0.0100 seconds

hbase(main):024:0> put 'emp','2','office:age','22'
0 row(s) in 0.0190 seconds

hbase(main):025:0>
```

Internally, HBase doesn't do an update but it assigns a column with new timestamp and scan fetches the latest data from columns.

```
hbase(main):025:0> put 'emp','1','office:salary','20000'
0 row(s) in 0.0250 seconds

hbase(main):026:0> put 'emp','1','office:salary','30000'
0 row(s) in 0.0450 seconds

hbase(main):027:0> put 'emp','1','office:salary','40000'
0 row(s) in 0.0490 seconds

hbase(main):028:0>
```

```
hbase(main):028:0> put 'emp','3','office:age','20'
0 row(s) in 0.0390 seconds

hbase(main):029:0> put 'emp','3','office:age','24'
0 row(s) in 0.0500 seconds

hbase(main):030:0>
```

```
hbase(main):002:0> put 'emp','3','office:gender','F'
0 row(s) in 0.1890 seconds

hbase(main):003:0> put 'emp','1','office:gender','F'
0 row(s) in 0.0450 seconds

hbase(main):004:0> put 'emp','1','office:age','21'
0 row(s) in 0.0480 seconds
```

```
hbase(main):008:0> put 'emp','2','office:salary','30000'
0 row(s) in 0.0400 seconds

hbase(main):009:0> put 'emp','3','office:salary','32000'
0 row(s) in 0.0160 seconds
```

5) Reading Data from a table

This returns all rows from table.

```
hbase(main):010:0> scan 'emp'
ROW                   COLUMN+CELL
 1                    column=office:age, timestamp=1648181408456, value=21
 1                    column=office:gender, timestamp=1648181400259, value=F
 1                    column=office:name, timestamp=1648180470228, value=Akshata
 1                    column=office:salary, timestamp=1648180828615, value=40000
 2                    column=office:age, timestamp=1648180724289, value=22
 2                    column=office:gender, timestamp=1648180662949, value=F
 2                    column=office:name, timestamp=1648180489592, value=Kajal
 2                    column=office:salary, timestamp=1648181491322, value=30000
 3                    column=office:age, timestamp=1648180900801, value=24
 3                    column=office:gender, timestamp=1648181384705, value=F
 3                    column=office:name, timestamp=1648180519436, value=Sarita
 3                    column=office:salary, timestamp=1648181503066, value=32000
3 row(s) in 0.0490 seconds
```

This scan's the 'emp' table to return name and age columns from starting row 1 and ending row 3.

```
hbase(main):011:0> scan 'emp',{COLUMNS=> ['office:name','office:age'],STARTROW=>'1',STOPROW=>'3'}
ROW                   COLUMN+CELL
 1                    column=office:age, timestamp=1648181408456, value=21
 1                    column=office:name, timestamp=1648180470228, value=Akshata
 2                    column=office:age, timestamp=1648180724289, value=22
 2                    column=office:name, timestamp=1648180489592, value=Kajal
2 row(s) in 0.0480 seconds

hbase(main):012:0>
```

This returns all columns for row ‘2’ from ’emp’ table.

```
hbase(main):012:0> get 'emp','2'
COLUMN                          CELL
 office:age                     timestamp=1648180724289, value=22
 office:gender                  timestamp=1648180662949, value=F
 office:name                    timestamp=1648180489592, value=Kajal
 office:salary                  timestamp=1648181491322, value=30000
4 row(s) in 0.0220 seconds

hbase(main):013:0>
```

We can also specify which columns to return.

```
hbase(main):014:0> get 'emp','1',{COLUMNS=> ['office:age','office:name']}
COLUMN                          CELL
 office:age                     timestamp=1648181408456, value=21
 office:name                    timestamp=1648180470228, value=Akshata
2 row(s) in 0.0190 seconds

hbase(main):015:0>
```

Disabling Table

Use disable to disable a table. Prior to delete a table or change its setting, first, you need to disable the table. The syntax to disable the table is as follows.

Let’s disable the ’emp’ table and then will see how to check if the table disabled.

Use is_disabled to check if the table is disabled. When it disabled it returns ‘true

Let’s check if the table disabled by using describe

```
hbase(main):015:0> disable 'emp'
0 row(s) in 1.4210 seconds

hbase(main):016:0> is_disabled 'emp'
true
0 row(s) in 0.0150 seconds

hbase(main):017:0> describe 'emp'
Table emp is DISABLED
emp
COLUMN FAMILIES DESCRIPTION
{NAME => 'office', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1'
, COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '6553
6', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.0710 seconds
```

Enabling Table

Use enalbe to enable a disabled table. You need to enable a disabled table first to perform any regular commands., The syntax to enable the table is as follows.

Deleting Rows

Use delete command, to remove a column at a row from a table.

```
hbase(main):018:0> enable 'emp'
0 row(s) in 0.6650 seconds
```

Dropping Table

Use drop command to delete a table. You should disable a table first before you drop it.

```
hbase(main):019:0> deop 'emp'
```