EXPERIMENT 4

Hbase Commands

THEORY

What is HBase?

HBase is a column-oriented non-relational database management system that runs on top of the 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 Java™ 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.

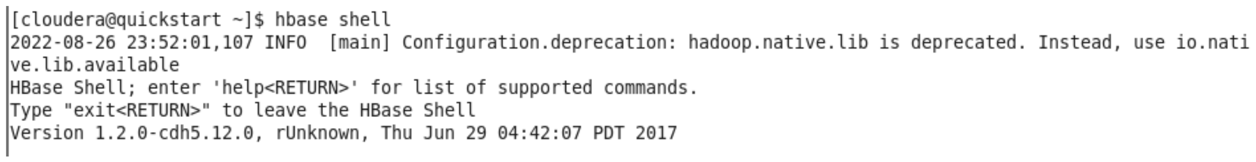
HBase relies on ZooKeeper for high-performance coordination. ZooKeeper is built into HBase, but if you're running a production cluster, it's suggested that you have a dedicated ZooKeeper cluster that's integrated with your HBase cluster.

HBase works well with Hive, a query engine for batch processing of big data, to enable fault-tolerant big data applications.

STEP 1

Hbase shell

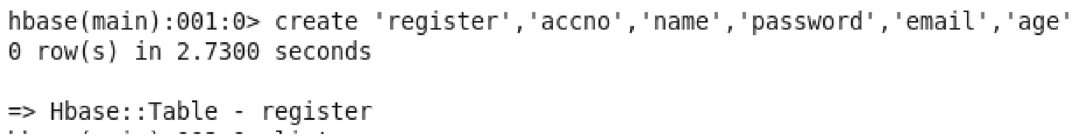
HBase contains a shell using which you can communicate with HBase. HBase uses the Hadoop File System to store its data. It will have a master server and region servers. The data storage will be in the form of regions (tables). These regions will be split up and stored in region servers.
The master server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



```
[cloudera@quickstart ~]$ hbase shell
2022-08-26 23:52:01,107 INFO  [main] Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.nati
ve.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.12.0, rUnknown, Thu Jun 29 04:42:07 PDT 2017
```

STEP 2

Create Table
You can create a table using the create command, here you must specify the table name
and the Column Family name. The syntax to create a table in HBase shell is shown below.
create ‘<table name>’,’<column family>’

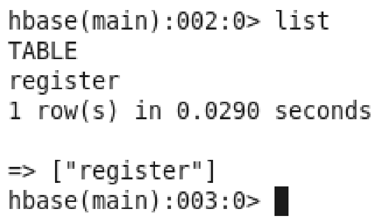


```
hbase(main):001:0> create 'register','accno','name','password','email','age'
0 row(s) in 2.7300 seconds

=> Hbase::Table - register
```

STEP 3

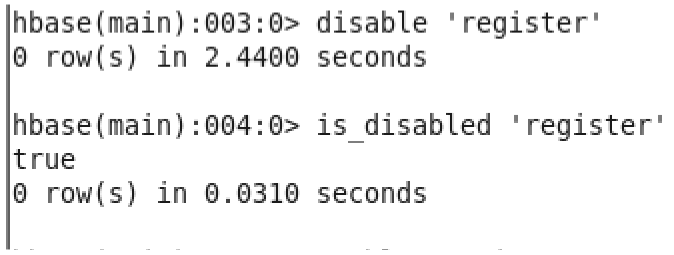
Listing a Table
Liist is the command that is used to list all the tables in HBase. Given below is the syntax
of the list command.

hbase(main):001:0 > list


```
hbase(main):002:0> list
TABLE
register
1 row(s) in 0.0290 seconds

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

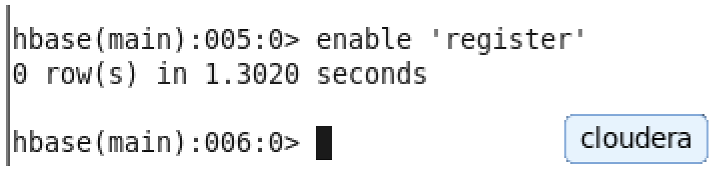
STEP 4

Disabling a Table
To delete a table or change its settings, you need to first disable the table using the disable command. After disabling the table, you can still sense its existence through list and exists commands.Given below is the syntax to disable a table:
disable '<table name>'


```
hbase(main):003:0> disable 'register'
0 row(s) in 2.4400 seconds

hbase(main):004:0> is_disabled 'register'
true
0 row(s) in 0.0310 seconds
```

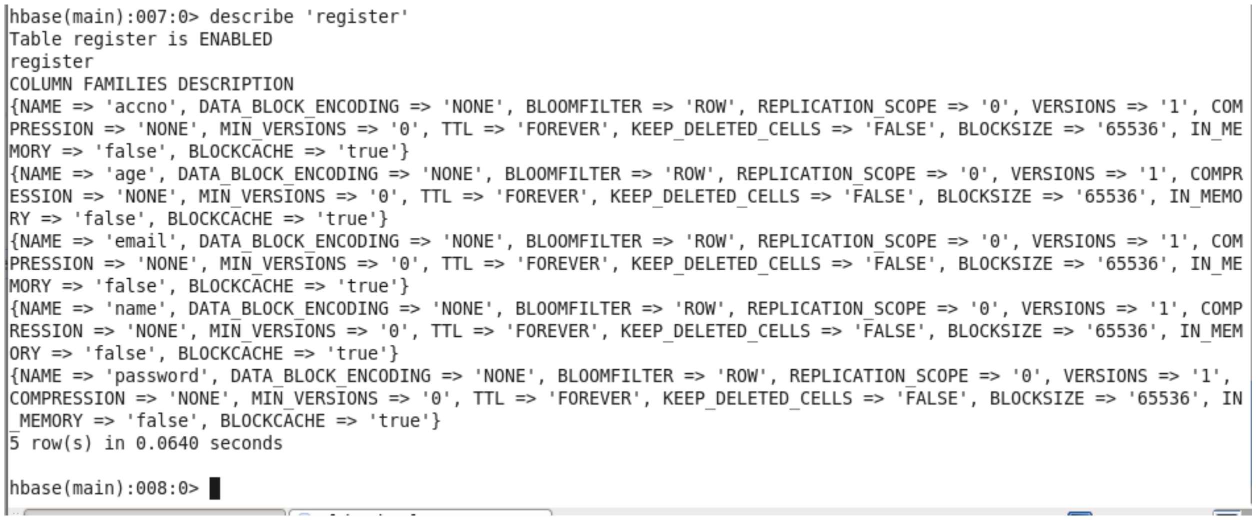
STEP 5

Enabling a Table
Syntax to enable a table:
enable '<table name>'


```
hbase(main):005:0> enable 'register'
0 row(s) in 1.3020 seconds

hbase(main):006:0>
```

STEP 6

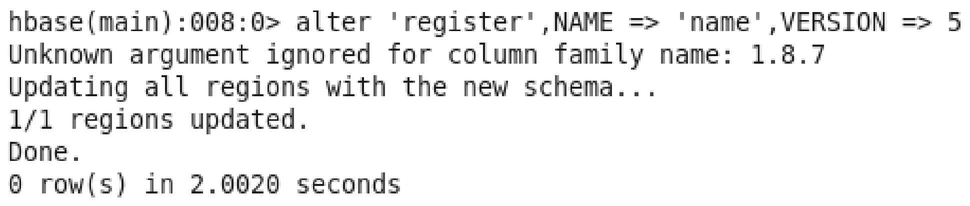
Describe Table
This command returns the description of the table. Its syntax is as follows:
hbase> describe 'table name'
Given below is the output of the described command on the register table:



```
hbase(main):007:0> describe 'register'
Table register is ENABLED
register
COLUMN FAMILIES DESCRIPTION
{NAME => 'accno', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COM
PRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_ME
MORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'age', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPR
ESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEMO
RY => 'false', BLOCKCACHE => 'true'}
{NAME => 'email', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COM
PRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_ME
MORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'name', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMP
RESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN_MEM
ORY => 'false', BLOCKCACHE => 'true'}
{NAME => 'password', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1',
COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE', BLOCKSIZE => '65536', IN
_MEMORY => 'false', BLOCKCACHE => 'true'}
5 row(s) in 0.0640 seconds

hbase(main):008:0> 
```

STEP 7

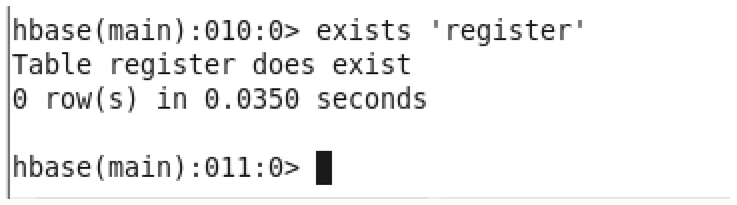
Alter Command
Alter is the command used to make changes to an existing table. Using this command, you can change the maximum number of cells of a column family, set and delete table scope operators, and delete a column family from a table.
★ Changing the Maximum Number of Cells of a Column Family
Given below is the syntax to change the maximum number of cells of a column family.
hbase> alter 't1', NAME ⇒ 'f1', VERSIONS ⇒ 5



```
hbase(main):008:0> alter 'register',NAME => 'name',VERSION => 5
Unknown argument ignored for column family name: 1.8.7
Updating all regions with the new schema...
1/1 regions updated.
Done.
0 row(s) in 2.0020 seconds
```

STEP 8

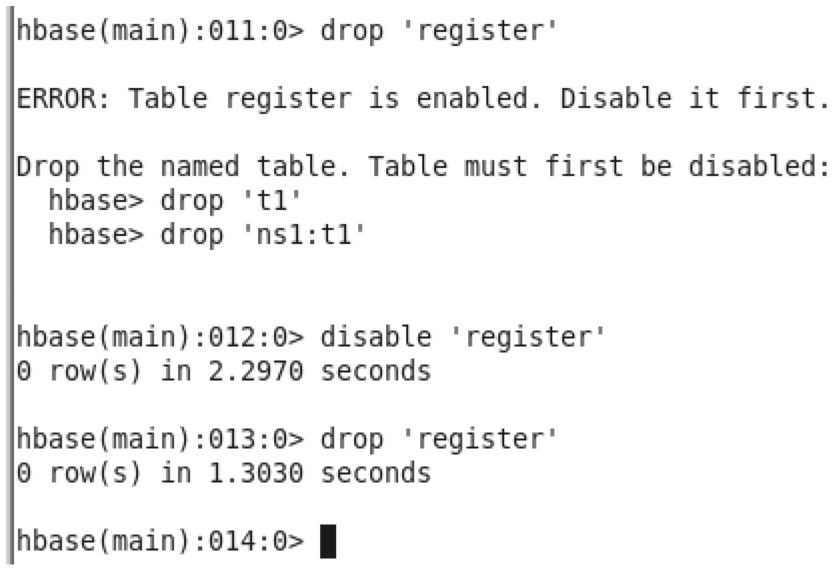
Exist Command
You can verify the existence of a table using the exists command. The following example shows how to use this command.



```
hbase(main):010:0> exists 'register'
Table register does exist
0 row(s) in 0.0350 seconds

hbase(main):011:0> █
```

STEP 9

Drop Table
Using the drop command, you can delete a table. But before dropping a table, you have to disable it.

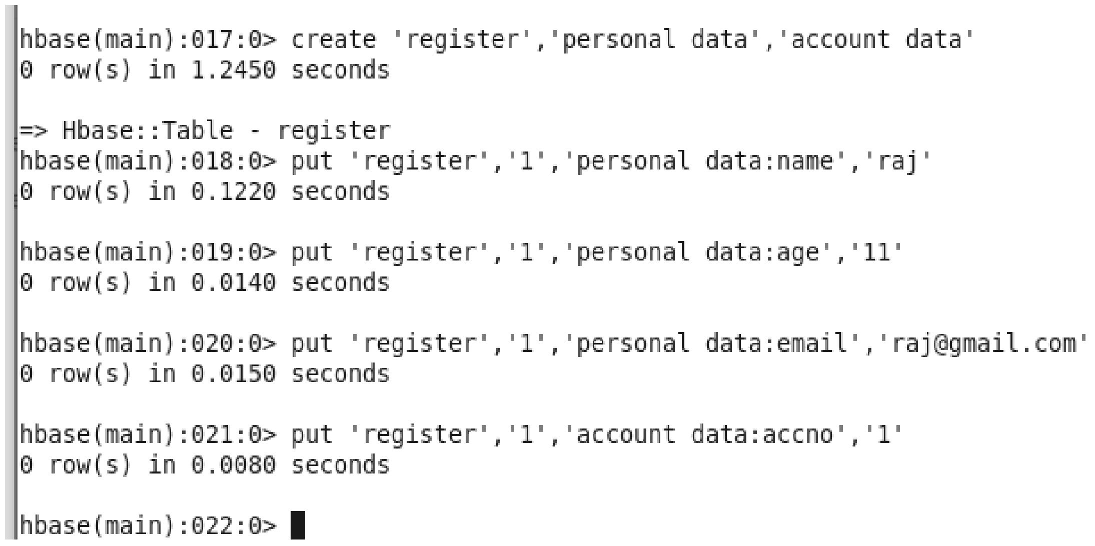


```
hbase(main):011:0> drop 'register'

ERROR: Table register is enabled. Disable it first.

Drop the named table. Table must first be disabled:
  hbase> drop 't1'
  hbase> drop 'ns1:t1'


hbase(main):012:0> disable 'register'
0 row(s) in 2.2970 seconds

hbase(main):013:0> drop 'register'
0 row(s) in 1.3030 seconds

hbase(main):014:0> █
```

STEP 10

Creating Data (Put command)
Using the put command, you can insert rows into a table. Its syntax is as follows:

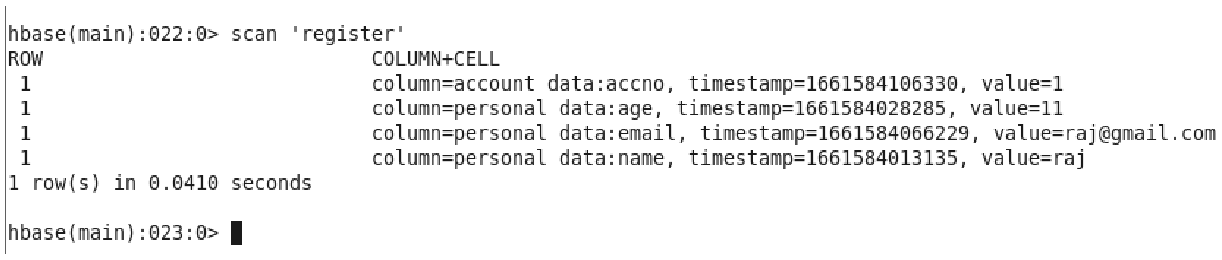
put ’<table name>’,’row1’,’<colfamily:colname>’,’<value>’



```
hbase(main):017:0> create 'register','personal data','account data'
0 row(s) in 1.2450 seconds

=> Hbase::Table - register
hbase(main):018:0> put 'register','1','personal data:name','raj'
0 row(s) in 0.1220 seconds

hbase(main):019:0> put 'register','1','personal data:age','11'
0 row(s) in 0.0140 seconds

hbase(main):020:0> put 'register','1','personal data:email','raj@gmail.com'
0 row(s) in 0.0150 seconds

hbase(main):021:0> put 'register','1','account data:accno','1'
0 row(s) in 0.0080 seconds

hbase(main):022:0> 
```

STEP 11

Scan Table
The scan command is used to view the data in HTable. Using the scan command, you can get the table data. Its syntax is as follows:
scan ‘<table name>’

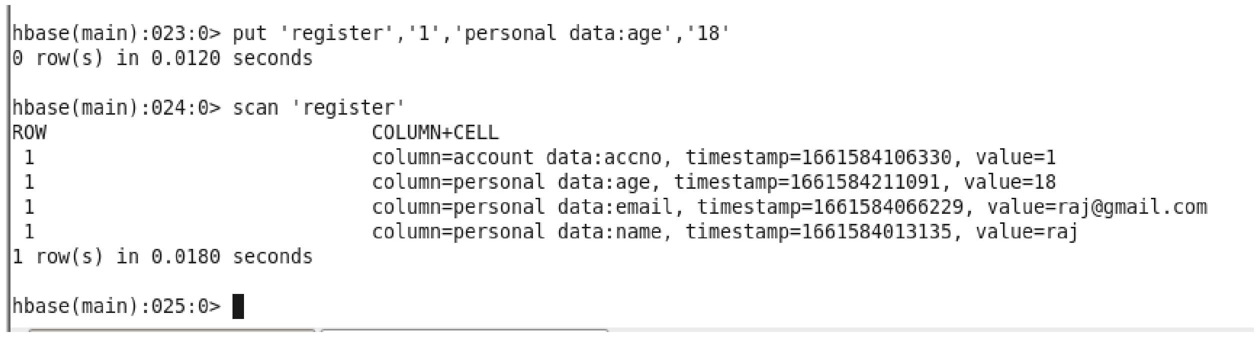


```
hbase(main):022:0> scan 'register'
ROW                          COLUMN+CELL
 1                           column=account data:accno, timestamp=1661584106330, value=1
 1                           column=personal data:age, timestamp=1661584028285, value=11
 1                           column=personal data:email, timestamp=1661584066229, value=raj@gmail.com
 1                           column=personal data:name, timestamp=1661584013135, value=raj
1 row(s) in 0.0410 seconds

hbase(main):023:0> 
```

STEP 12

Updating Data (Put Command)
You can update an existing cell value using the put command. The newly given value replaces the existing value, updating the row. To do so, just follow the same syntax and mention your new value as shown below.

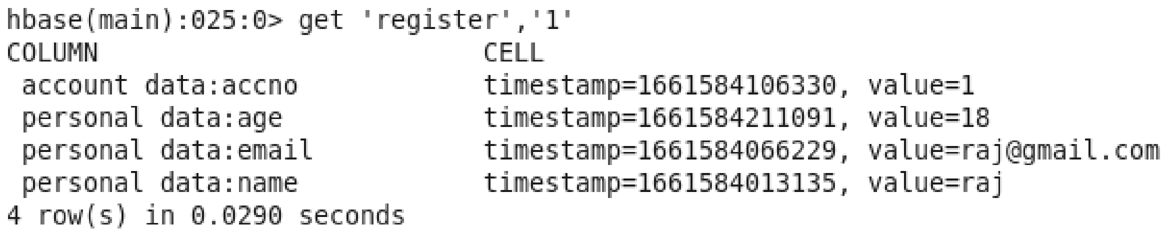
put ‘table name’,’row ’,'Column family:column name',’new value’



```
hbase(main):023:0> put 'register','1','personal data:age','18'
0 row(s) in 0.0120 seconds

hbase(main):024:0> scan 'register'
ROW                           COLUMN+CELL
 1                            column=account data:accno, timestamp=1661584106330, value=1
 1                            column=personal data:age, timestamp=1661584211091, value=18
 1                            column=personal data:email, timestamp=1661584066229, value=raj@gmail.com
 1                            column=personal data:name, timestamp=1661584013135, value=raj
1 row(s) in 0.0180 seconds

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

STEP 13

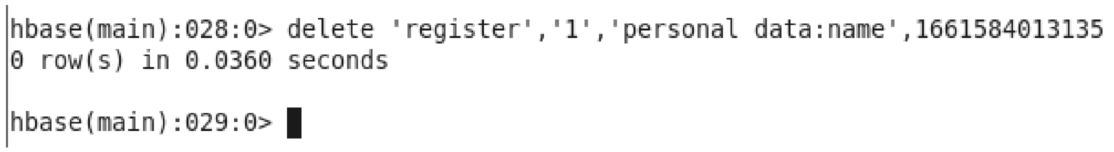
Read Data (Get Command)
The get command and the get() method of HTable class are used to read data from a table in HBase. Using the get command, you can get a single row of data at a time. Its syntax is as follows:
get ’<table name>’,’row1’



```
hbase(main):025:0> get 'register','1'
COLUMN                         CELL
 account data:accno            timestamp=1661584106330, value=1
 personal data:age             timestamp=1661584211091, value=18
 personal data:email           timestamp=1661584066229, value=raj@gmail.com
 personal data:name            timestamp=1661584013135, value=raj
4 row(s) in 0.0290 seconds
```

STEP 14

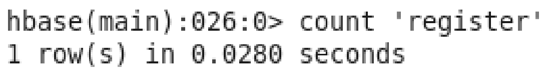
Delete Data (Delete Command)
Deleting a Specific Cell in a Table
Using the delete command, you can delete a specific cell in a table. The syntax of delete command is as follows:
delete '<table name>', '<row>', '<column name >', '<time stamp>'



```
hbase(main):028:0> delete 'register','1','personal data:name',1661584013135
0 row(s) in 0.0360 seconds

hbase(main):029:0> 
```

STEP 15

Count Command
You can count the number of rows of a table using the count command. Its syntax is as follows:
count '<table name>'

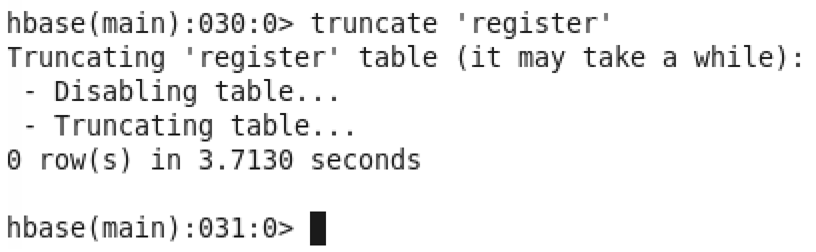


```
hbase(main):026:0> count 'register'
1 row(s) in 0.0280 seconds
```

STEP 16

Truncate Command
This command disables drops and recreates a table. The syntax of truncate is as follows:

hbase> truncate 'table name'



```
hbase(main):030:0> truncate 'register'
Truncating 'register' table (it may take a while):
 - Disabling table...
 - Truncating table...
0 row(s) in 3.7130 seconds

hbase(main):031:0> 
```