Hbase_lab

Invoke Hbase

We have to start daemons and zookeeper then invoke HBase shell

(base) [bigdata@localhost]$ start-all.sh
(base) [bigdata@localhost hbase]$ start-hbase.sh
(base) [bigdata@localhost hbase]$ hbase shell
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.3.3, r3e4bf4bee3a08b25591b9c22fea0518686a7e834, Wed Oct 28 06:36:25 UTC 2020
Took 0.0009 seconds
hbase(main):001:0> version
2.3.3, r3e4bf4bee3a08b25591b9c22fea0518686a7e834, Wed Oct 28 06:36:25 UTC 2020
Took 0.0006 seconds
hbase(main):002:0> table_help

```
Help for table-reference commands.
You can either create a table via 'create' and then manipulate the table via commands like 'put', 'get', etc.
However, as of 0.96, you can also get a reference to a table, on which you can invoke commands.
For instance, you can get create a table and keep around a reference to it via:
hbase> t = create 't', 'cf'
Or, if you have already created the table, you can get a reference to it: hbase> t = get_table 't'
You can do things like call 'put' on the table:
hbase> t.put 'r', 'cf:q', 'v'
which puts a row 'r' with column family 'cf', qualifier 'q' and value 'v' into table t.
To read the data out, you can scan the table:
hbase> t.scan
which will read all the rows in table 't'.
Essentially, any command that takes a table name can also be done via table reference.
Other commands include things like: get, delete, deleteall,
get_all_columns, get_counter, count, incr. These functions, along with
the standard JRuby object methods are also available via tab completion.
For more information on how to use each of these commands, you can also just type:
hbase> t.help 'scan'
which will output more information on how to use that command.
You can also do general admin actions directly on a table; things like enable, disable,
flush and drop just by typing:
hbase> t.enable
hbase> t.flush
hbase> t.disable
hbase> t.drop
Note that after dropping a table, your reference to it becomes useless and further usage
is undefined (and not recommended).
Took 0.0003 seconds
```

hbase(main):003:0> whoami

```
bigdata (auth:SIMPLE)
groups: bigdata
Took 0.0599 seconds
hbase(main):004:0> list
TABLE
0 row(s)
```

hbase(main):005:0> status
1 active master, 0 backup masters, 1 servers, 0 dead, 2.0000 average load
Took 0.1548 seconds
hbase(main):006:0> create 'emp', 'personal data', 'professional data'
Created table emp

```
Took 0.7601 seconds
=> Hbase::Table – emp
hbase(main):007:0> list
TABLE
emp
1 row(s)
Took 0.0127 seconds
=> ["emp"]
hbase(main):005:0> create 'course' ,'bigdata'
Created table course
Took 0.7574 seconds
=> Hbase::Table – course
hbase(main):006:0> list
TABLE
course
1 row(s)
Took 0.0121 seconds
=> ["course"]
hbase(main):007:0> describe 'course'
Table course is ENABLED
course
COLUMN FAMILIES DESCRIPTION
{NAME => 'bigdata', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', VERSIONS => '1',
KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', COMPRESSION
=> 'N
ONE', TTL => 'FOREVER', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE =>
'65536', REPLICATION_SCOPE => '0'}
1 row(s)
QUOTAS
0 row(s)
Took 0.2591 seconds
hbase(main):008:0> disable 'course'
Took 0.5467 seconds
hbase(main):009:0> enable 'course'
Took 0.7174 seconds
hbase(main):010:0> show_filters
DependentColumnFilter
KeyOnlyFilter
ColumnCountGetFilter
SingleColumnValueFilter
PrefixFilter
SingleColumnValueExcludeFilter
FirstKeyOnlyFilter
ColumnRangeFilter
ColumnValueFilter
TimestampsFilter
FamilyFilter
QualifierFilter
ColumnPrefixFilter
RowFilter
MultipleColumnPrefixFilter
```

InclusiveStopFilter
PageFilter
ValueFilter
ColumnPaginationFilter
Took 0.0084 seconds
hbase(main):011:0> drop 'course'
ERROR: Table course is enabled. Disable it first.
For usage try 'help "drop"'
Took 0.0521 seconds
hbase(main):012:0> is_enabled 'course'
true
Took 0.0082 seconds
=> true
hbase(main):013:0> alter 'course', NAME='learning'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 1.7402 seconds
hbase(main):014:0> alter_status 'course'
1/1 regions updated.
Done.
Took 1.0125 seconds
hbase(main):015:0> count 'course', CACHE=>1000
0 row(s)
Took 0.1021 seconds
=> 0
count can fetch 1000 rows at a time from "course" table if it was created and exist.
We can make cache to some lower value if the table consists of more rows. But by default it will fetch one row at a time.
hbase(main):016:0> count 'course', INTERVAL => 100000
0 row(s)
Took 0.0070 seconds
=> 0

Given below is a sample schema of a table named emp. It has two column families: “personal data” and “professional data”.

hbase(main):035:0> create 'emp', 'personal data', 'professional data'
Created table emp
Took 1.1530 seconds
=> Hbase::Table - emp
hbase(main):036:0> list
TABLE
course
emp
2 row(s)
Took 0.0150 seconds
=> ["course", "emp"]
hbase(main): 037:0> put 'emp', 1 ,'personal data:name' , 'Aya'
hbase(main): 038:0> put 'emp', 2 ,'personal data:name' , 'Marwan'
hbase(main): 039:0> put 'emp', 3 ,'personal data:name' , 'Suzy'

```
hbase(main): 040:0> put 'emp', 4 ,'personal data:name' , 'Fahd'
hbase(main): 041:0> put 'emp', 5 ,'personal data:name' , 'Ayman'
hbase(main): 042:0> put 'emp', 1 ,'personal data:city' , 'Cairo'
hbase(main): 043:0> put 'emp', 2 ,'personal data:city' , 'Alex'
hbase(main): 044:0> put 'emp', 3 ,'personal data:city' , 'Giza'
hbase(main): 045:0> put 'emp', 4 ,'personal data:city' , 'Giza'
hbase(main): 046:0> put 'emp', 5 ,'personal data:city' , 'Aswan'

hbase(main):047:0> scan 'emp'
ROW COLUMN+CELL
0 row(s)
Took 0.0246 seconds

hbase(main):048:0> put 'emp', 1 ,'personal data:age' , '35'
hbase(main):049:0> put 'emp', 2 ,'personal data:age' , '33'
hbase(main):050:0> put 'emp', 3 ,'personal data:age' , '23'
hbase(main):051:0> put 'emp', 4 ,'personal data:age' , '27'
hbase(main):052:0> put 'emp', 5 ,'personal data:age' , '38'

hbase(main):053:0> scan 'emp'

hbase(main):054:0> put 'emp', 1 , 'professional data:position' , 'manager'
hbase(main):055:0> put 'emp', 2 , 'professional data:position' , 'team leader'
hbase(main):056:0> scan 'emp' ,{COLUMNS => 'personal data:name', LIMIT => 2, STARTROW => '3'}
hbase(main):057:0> get 'emp' , '1'
```

```
hbase(main):019:0> get 'emp' , '1'
COLUMN                CELL
 personal data:age    timestamp=2023-03-13T20:39:56.396, value=35
 personal data:city   timestamp=2023-03-13T20:37:42.551, value=Cairo
 personal data:name   timestamp=2023-03-13T20:32:06.515, value=Aya
 professional data:po timestamp=2023-03-13T20:41:11.682, value=manager
 sition
1 row(s)
Took 0.0448 seconds
```

```
hbase(main):058:0> get 'emp', '1', {COLUMN => 'personal data:name'}
```

```
hbase(main):020:0> get 'emp', '1', {COLUMN => 'personal data:name'}
COLUMN                CELL
 personal data:name   timestamp=2023-03-13T20:32:06.515, value=Aya
1 row(s)
Took 0.0481 seconds
```

```
base(main):060:0> scan 'emp',{ COLUMNS => 'personal data:name' , FILTER =>
"ValueFilter(=,'binary:Aya')" }
ROW                           COLUMN+CELL
 1                           column=personal data:name, timestamp=2022-02-03T21:41:49.484, value=heba
1 row(s)
Took 0.0429 seconds
```

```
hbase(main):021:0> scan 'emp',{ COLUMNS => 'personal data:name' ,
FILTER =>  "ValueFilter(=,'binary:Aya')" }
ROW                   COLUMN+CELL
 1                   column=personal data:name, timestamp=2023-03-13T20:32:06.5
                     15, value=Aya
1 row(s)
Took 0.0475 seconds
```

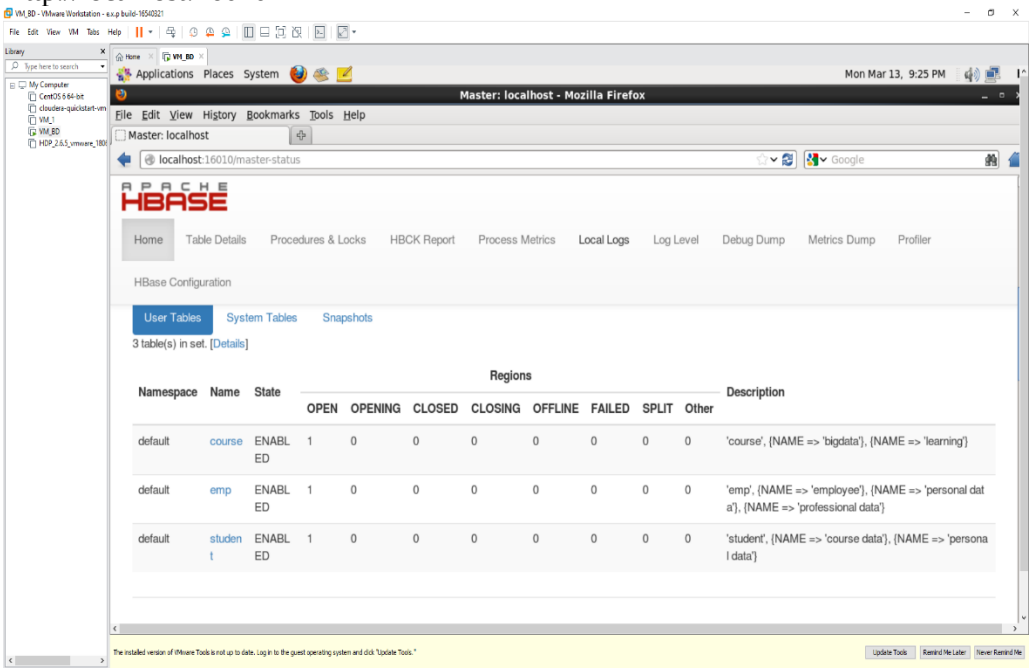
Row key	personal data	professional data

	personal data			professional data	
Rowkey	**Name**	**City**	**age**	**Rowkey**	**position**
1	Aya	Cairo	35	1	manager
2	Marwan	Alex	33	2	Team leader
3	Suzy	Giza	23		
4	Fahd	Giza	27		
5	Ayman	Aswan	38		

hbase(main):038:0> exists 'student'
Table student does not exist
Took 0.0102 seconds
=> false
hbase(main):039:0> exists 'emp'
Table emp does exist
Took 0.0114 seconds
=> true

Start Hbase web interface :

http://localhost:16010



hbase(main):040:0> exit
hbase$./bin/stop-hbase.sh

bigdata@localhost$ stop-all.sh
--The HBase directory can be seen through HDFS as shown

--By exploring HBase we can view and monitor its content with details as permissions owner, group size , date and replication if exist.

Tables Managements commands

These commands will allow programmers to create tables and table schemas with rows and column families.The following are Table Management commands

- Create
- List
- Describe
- Disable
- Disable_all
- Enable
- Enable_all
- Drop
- Drop_all
- Show_filters
- Alter
- Alter_status

Data manipulation commands

These commands will work on the table related to data manipulations such as putting data into a table, retrieving data from a table and deleting schema, etc.

The commands come under these are

- Count
- Put
- Get
- Delete
- Delete all
- Truncate
- Scan