AkbarAhmed.com

Engineering Leadership

HBase Command Line Tutorial

AKBARSAHMED

8 COMMENTS

AUGUST 13, 2012

Introduction

Start the HBase Shell

All subsequent commands in this post assume that you are in the HBase shell, which is started via the command listed below.

```
hbase shell
```

You should see output similar to:

```
12/08/12 12:30:52 WARN conf.Configuration: hadoop.native.lib 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 0.92.1-cdh4.0.1, rUnknown, Thu Jun 28 18:13:01 PDT 2012
```

Create a Table

We will initially create a table named **test** with one column family named **columnfamily1**.

> *Using a long column family name, such as **columnfamily1** is a horrible idea in production. Every cell (i.e. every value) in HBase is stored fully qualified. This basically means that long column family names will balloon the amount of disk space required to store your data. In summary, keep your column family names as terse as possible.*

```
create 'table1', 'columnfamily1'
```

List all Tables

```
list
```

You'll see output similar to:

```
TABLE
table1 1 row(s) in 0.0370 seconds
```

Let's now create a second table so that we can see some of the features of the **list** command.

```
create 'test', 'cf1'
```

```
list
```

You will see output similar to:

```
TABLE
table1
test
2 row(s) in 0.0320 seconds
```

If we only want to see the **test** table, or all tables that start with "te", we can use the following command.

```
list 'te'
```

or

```
list 'te.*'
```

Manually Insert Data into HBase

If you're using HBase, then you likely have data sets that are TBs in size. As a result, you'll never actually insert data manually. However, knowing how to insert data manually could prove useful at times.

To start, I'm going to create a new table named **cars**. My column family is **vi**, which is an abbreviation of vehicle information.

> *The schema that follows below is only for illustration purposes, and should not be used to create a production schema. In production, you should create a Row ID that helps to uniquely identify the row, and that is likely to be used in your queries. Therefore, one possibility would be to shift the Make, Model and Year left and use these items in the Row ID.*

```
create 'cars', 'vi'
```

Let's insert 3 column qualifies (make, model, year) and the associated values into the first row (row1).

```
put 'cars', 'row1', 'vi:make', 'bmw'
```

```
put 'cars', 'row1', 'vi:model', '5 series'
```

```
put 'cars', 'row1', 'vi:year', '2012'
```

Now let's add a second row.

```
put 'cars', 'row2', 'vi:make', 'mercedes'
```

```
put 'cars', 'row2', 'vi:model', 'e class'
```

```
put 'cars', 'row2', 'vi:year', '2012'
```

Scan a Table (i.e. Query a Table)

We'll start with a basic scan that returns all columns in the **cars** table.

```
scan 'cars'
```

You should see output similar to:

```
ROW           COLUMN+CELL
 row1          column=vi:make, timestamp=1344817012999, value=bmw
 row1          column=vi:model, timestamp=1344817020843, value=5 series
 row1          column=vi:year, timestamp=1344817033611, value=2012
 row2          column=vi:make, timestamp=1344817104923, value=mercedes
 row2          column=vi:model, timestamp=1344817115463, value=e class
 row2          column=vi:year, timestamp=1344817124547, value=2012
2 row(s) in 0.6900 seconds
```

Reading the output above you'll notice that the Row ID is listed under ROW. The COLUMN+CELL field shows the column family after **column=**, then the column qualifier, a timestamp that is automatically created by HBase, and the value.

Importantly, each row in our results shows an individual row id + column family + column qualifier combination. Therefore, you'll notice that multiple columns in a row are displayed in multiple rows in our results.

The next scan we'll run will limit our results to the **make** column qualifier.

```
scan 'cars', {COLUMNS => ['vi:make']}
```

If you have a particularly large result set, you can limit the number of rows returned with the **LIMIT** option. In this example I arbitrarily limit the results to 1 row to demonstrate how **LIMIT** works.

```
scan 'cars', {COLUMNS => ['vi:make'], LIMIT => 1}
```

To learn more about the **scan** command enter the following:

```
help 'scan'
```

Get One Row

The **get** command allows you to get one row of data at a time. You can optionally limit the number of columns returned.

We'll start by getting all columns in **row1**.

```
get 'cars', 'row1'
```

You should see output similar to:

```
COLUMN                    CELL
 vi:make                  timestamp=1344817012999, value=bmw
 vi:model                 timestamp=1344817020843, value=5 series
 vi:year                  timestamp=1344817033611, value=2012
3 row(s) in 0.0150 seconds
```

When looking at the output above, you should notice how the results under **COLUMN** show the fully qualified **column family:column qualifier**, such as **vi:make**.

To get one specific column include the COLUMN option.

```
get 'cars', 'row1', {COLUMN => 'vi:model'}
```

You can also get two or more columns by passing an array of columns.

```
get 'cars', 'row1', {COLUMN => ['vi:model', 'vi:year']}
```

To learn more about the **get** command enter:

```
help 'get'
```

Delete a Cell (Value)

```
delete 'cars', 'row2', 'vi:year'
```

Let's check that our delete worked.

```
get 'cars', 'row2'
```

You should see output that shows 2 columns.

```
COLUMN    CELL
vi:make   timestamp=1344817104923, value=mercedes
vi:model   timestamp=1344817115463, value=e class
2 row(s) in 0.0080 seconds
```

Disable and Delete a Table

```
disable 'cars'
```

```
drop 'cars'
```

```
disable 'table1'
```

```
drop 'table1'
```

```
disable 'test'
```

```
drop 'test'
```

View HBase Command Help

```
help
```

Exit the HBase Shell

```
exit
```

Categories: Hadoop, HBase • Tags: CDH4, Cloudera, hadoop, HBase •

8 thoughts on “HBase Command Line Tutorial”

Aniket Divekar *says:*
November 23, 2012 at 9:51 am
Hi akbar,
I am new to Hbase. Can you tell me if we can have our own timestamp inserted instead of default. If “YES” the how do we do that….
thanx.

REPLY

akbarsahmed *says:*
November 23, 2012 at 7:26 pm
HI Aniket,

Yes, you can customize the timestamp value. Assuming you’re using the CLI version of put, then just specify the timestamp you want as a Long. A similar capability exists with the API.

put ‘table’, ‘row’, ‘column’, ‘value’, timestamp

However, you should read up on timestamps in detail as other options exist, such as adding your own time column, and since customizing the timestamp has implications worth considering (and too detailed to discuss in a comment).

Akbar

REPLY

Aniket Divekar *says:*
November 24, 2012 at 5:30 am
thanx a lot.
there is another problem i am facing. I am constantly getting an error about Master not running. (Using hbase-0.94.2). but when I check the log files…there its printed that “Master is running”. what could be the reason?

Mohan *says:*
April 20, 2013 at 9:57 am
Hello,

is there any way of getting Hbase epoch time using the linux shell commands.

I knew perl has feature to convert normal timestamp to EPOCH time, but the problem is it is converting into system EPOCH time not the needed Hbase EPOCH.

Reply on this would be appreciated much 🙂

Regards
Mohan

REPLY

mna *says:*
April 25, 2013 at 9:14 am
Hi,
is it possible to "update" multiple row in hbase?
thanks.

REPLY

keystonelemur *says:*
April 18, 2014 at 2:48 pm
One very useful fact that should be mentioned: the default shell is IRB. If you know Ruby, you have a pretty significant amount of power to use in automation.

REPLY

Spiros *says:*
October 19, 2014 at 7:08 pm
I would like to make a comment here in which i am 90% sure. When we say "Delete a cell" we practically mean that we don't delete it rather than setting its contents as non-data (like an empty cell not containing data). Even if wanted to say that it would not be applicable as we delete the content of an entire column and i highlight the CONTENT , NOT THE COLUMN.

P.S.empty cell i don't know what we mean, but iam pretty sure that we don't mean NULL as in hbase there are neither NULL values nor joins.

REPLY

srikanthjk9 *says:*
April 7, 2016 at 7:29 am
Thank you really usefull.

REPLY