1. **General Hbase Shell commands**

Show cluster status. Can be ‘summary’, ‘simple’, or ‘detailed’. Thedefault is ‘summary’.

*hbase> status*

*status*

*hbase> status 'simple'*

*hbase> status 'summary'*

*hbase> status 'detailed'*

*version Output this HBase versionUsage:*

*hbase> version*

*whoami Show the CURRENT hbase USER.Usage:*

*hbase> whoami*

1. **Tables Management commands**

Alter column family schema; pass table name and a dictionary specifying new column family schema. Dictionaries are described on the main help command output.Dictionary must include name of column family to alter.For example, to change or add the ‘f1′ column family in table ‘t1′ from current value to keep a maximum of 5 cell VERSIONS, do:

*hbase> ALTER 't1', NAME => 'f1', VERSIONS => 5*

*You can operate ON several COLUMN families:*

*hbase> ALTER 't1', 'f1', {NAME => 'f2', IN\_MEMORY => TRUE}, {NAME => 'f3',*

*VERSIONS => 5}*

*TO DELETE the 'f1' COLUMN family IN TABLE 't1', USE one OF:hbase> ALTER 't1', NAME => 'f1',*

*METHOD => 'delete'*

*hbase> ALTER 't1', 'delete' => 'f1'*

**alter**

You can also change table-scope attributes like MAX\_FILESIZE, READONLY, MEMSTORE\_FLUSHSIZE, DEFERRED\_LOG\_FLUSH, etc. These can be put at the end;

for example, to change the max size of a region to 128MB, do:

*hbase> ALTER 't1', MAX\_FILESIZE => '134217728'*

*You can ADD a TABLE coprocessor BY setting a TABLE coprocessor attribute:*

*hbase> ALTER 't1','coprocessor'=>'hdfs:///foo.jar|com.foo.FooRegionObserver|1001|arg1=1,arg2=2'*

Since you can have multiple coprocessors configured for a table, a sequence number will be automatically appended to the attribute name to uniquely identify it. The coprocessor attribute must match the pattern below in order for the framework to understand how to load the coprocessor classes:

[coprocessor jar file location] | class name | [priority] | [arguments]

You can also set configuration settings specific to this table or column family:

*hbase> ALTER 't1', CONFIGURATION =>{'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}*

*hbase> ALTER 't1', {NAME => 'f2', CONFIGURATION =>{'hbase.hstore.blockingStoreFiles' => '10'}}*

You can also remove a table-scope attribute:

*hbase> ALTER 't1', METHOD => 'table\_att\_unset', NAME => 'MAX\_FILESIZE'*

*hbase> ALTER 't1', METHOD => 'table\_att\_unset', NAME => 'coprocessor$1'*

There could be more than one alteration in one command:

*hbase> ALTER 't1', { NAME => 'f1', VERSIONS => 3 }, { MAX\_FILESIZE =>'134217728' }, { METHOD => 'delete', NAME => 'f2' }, OWNER => 'johndoe',METADATA => { 'mykey' => 'myvalue' }*

**Create table**; pass table name, a dictionary of specifications per column family, and optionally a dictionary of table configuration.

**create**

*hbase> CREATE 't1', {NAME => 'f1', VERSIONS => 5}*

*hbase> CREATE 't1', {NAME => 'f1'}, {NAME => 'f2'}, {NAME => 'f3'}*

The above in shorthand would be the following:

*hbase> CREATE 't1', 'f1', 'f2', 'f3'*

*hbase> CREATE 't1', {NAME => 'f1', VERSIONS => 1, TTL => 2592000,BLOCKCACHE => TRUE}*

*hbase> CREATE 't1', {NAME => 'f1', CONFIGURATION =>{'hbase.hstore.blockingStoreFiles' => '10'}}*

Table configuration options can be put at the end.

describe Describe the named table.

*hbase> describe 't1'*

disable Start disable of named table

*hbase> disable 't1'*

disable\_all Disable all of tables matching the given regex

*hbase> disable\_all 't.\*'*

is\_disabled verifies Is named table disabled

*hbase> is\_disabled 't1'*

drop Drop the named table. Table must first be disabled

*hbase> DROP 't1'*

drop\_all Drop all of the tables matching the given regex

*hbase> drop\_all 't.\*'*

enable Start enable of named table

*hbase> enable 't1'*

enable\_all

Enable all of the tables matching the given regex

*hbase> enable\_all 't.\*'*

is\_enabled verifies Is named table enabled

*hbase> is\_enabled 't1'*

exists Does the named table exist

*hbase> EXISTS 't1'*

list

show\_filters

alter\_status

alter\_async

List all tables in hbase. Optional regular expression parameter could

be used to filter the output

*hbase> list*

*hbase> list 'abc.\*'*

Show all the filters in hbase.

*hbase> show\_filters*

Get the status of the alter command. Indicates the number of regions of the table that have received the updated schema Pass table name.

*hbase> alter\_status 't1'*

Alter column family schema, does not wait for all regions to receive the schema changes. Pass table name and a dictionary specifying new column family schema. Dictionaries are described on the main help command output. Dictionary must include name of column family to alter. To change or add the ‘f1′ column family in table ‘t1′ from defaults to instead keep a maximum of 5 cell VERSIONS, do:

You can also change table-scope attributes like MAX\_FILESIZE MEMSTORE\_FLUSHSIZE, READONLY, and DEFERRED\_LOG\_FLUSH.

For example, to change the max size of a family to 128MB, do:

*hbase> ALTER 't1', METHOD => 'table\_att', MAX\_FILESIZE => '134217728'*

*There could be more than one alteration IN one command:*

*hbase> ALTER 't1', {NAME => 'f1'}, {NAME => 'f2', METHOD => 'delete'}*

To check if all the regions have been updated, use alter\_status <table\_name>

1. **Data Manipulation commands**

count

Count the number of rows in a table. Return value is the number of rows. This operation may take a LONG time (Run ‘$HADOOP\_HOME/bin/hadoop jar hbase.jar rowcount’ to run a counting mapreduce job). Current count is shown every 1000 rows by default. Count interval may be optionally specified. Scan caching is enabled on

count scans by default. Default cache size is 10 rows. If your rows are small in size, you may want to increase this parameter. Examples:hbase> count ‘t1′

*hbase> COUNT 't1', INTERVAL => 100000*

*hbase> COUNT 't1', CACHE => 1000*

*hbase> COUNT 't1', INTERVAL => 10, CACHE => 1000*

The same commands also can be run on a table reference. Suppose you had a reference t to table ‘t1′, the corresponding commands would be:

*hbase> t.COUNT*

*hbase> t.COUNT INTERVAL => 100000*

*hbase> t.COUNT CACHE => 1000*

*hbase> t.COUNT INTERVAL => 10, CACHE => 1000*

delete

Put a delete cell value at specified table/row/column and optionally timestamp coordinates. Deletes must match the deleted cell’s coordinates exactly. When scanning, a delete cell suppresses older versions. To delete a cell from ‘t1′ at row ‘r1′ under column ‘c1′ marked with the time ‘ts1′, do:

*hbase> t.COUNT*

*hbase> t.COUNT INTERVAL => 100000*

*hbase> t.COUNT CACHE => 1000*

*hbase> t.COUNT INTERVAL => 10, CACHE => 1000*

The same command can also be run on a table reference. Suppose you had a reference t to table ‘t1′, the corresponding command would be:hbase> t.delete ‘r1′, ‘c1′, ts1 Delete all cells in a given row; pass a table name, row, and optionally a column and timestamp. Examples:

*hbase> deleteall 't1', 'r1'*

*hbase> deleteall 't1', 'r1', 'c1'*

*hbase> deleteall 't1', 'r1', 'c1', ts1*

deleteall

The same commands also can be run on a table reference. Suppose you had a reference t to table ‘t1′, the corresponding command would be:

*hbase> t.deleteall 'r1'*

*hbase> t.deleteall 'r1', 'c1'*

*hbase> t.deleteall 'r1', 'c1', ts1*

get

Get row or cell contents; pass table name, row, and optionally a dictionary of column(s), timestamp, timerange and versions. Examples:

*hbase> get 't1', 'r1'*

*hbase> get 't1', 'r1', {TIMERANGE => [ts1, ts2]}*

*hbase> get 't1', 'r1', {COLUMN => 'c1'}*

*hbase> get 't1', 'r1', {COLUMN => ['c1', 'c2', 'c3']}*

*hbase> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}*

*hbase> get 't1', 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts2], VERSIONS=> 4}*

*hbase> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VERSIONS => 4}*

*hbase> get 't1', 'r1', {FILTER => "ValueFilter(=, 'binary:abc')"}*

*hbase> get 't1', 'r1', 'c1'*

*hbase> get 't1', 'r1', 'c1', 'c2'*

*hbase> get 't1', 'r1', ['c1', 'c2']*

Besides the default ‘toStringBinary’ format, ‘get’ also supports custom formatting by column. A user can define a FORMATTER by adding it to the column name in the get specification. The FORMATTER can be stipulated:1. either as a org.apache.hadoop.hbase.util.Bytes method name (e.g, toInt, toString) 2. or as a custom class followed by method name:

e.g.

‘c(MyFormatterClass).format’.Example formatting cf:qualifier1 and cf:qualifier2 both as Integers:

*hbase> get 't1', 'r1' {COLUMN=>*

*['cf:qualifier1:toInt','cf:qualifier2:c(org.apache.hadoop.hbase.util.Bytes).toInt'] }*

Note that you can specify a FORMATTER by column only (cf:qualifer). You cannot specify a FORMATTER for all columns of a column family.The same commands also can be run on a reference to a table (obtained via get\_table or create\_table). Suppose you had a reference t to table ‘t1′, the corresponding commands would be:

*hbase> t.get 'r1'*

*hbase> t.get 'r1', {TIMERANGE => [ts1, ts2]}*

*hbase> t.get 'r1', {COLUMN => 'c1'}*

*hbase> t.get 'r1', {COLUMN => ['c1', 'c2', 'c3']}*

*hbase> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}*

*hbase> t.get 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts2], VERSIONS =>4}*

*hbase> t.get 'r1', {COLUMN => 'c1', TIMESTAMP => ts1, VERSIONS => 4}*

*hbase> t.get 'r1', {FILTER => "ValueFilter(=, 'binary:abc')"}*

*hbase> t.get 'r1', 'c1'*

*hbase> t.get 'r1', 'c1', 'c2'*

*hbase> t.get 'r1', ['c1', 'c2']*

get\_counter

Return a counter cell value at specified table/row/column coordinates. A cell cell should be managed with atomic increment function oh HBase and the data should be binary encoded. Example:

*hbase> get\_counter 't1', 'r1', 'c1'*

The same commands also can be run on a table reference. Suppose you had a reference t to table ‘t1′, the corresponding command would be:

*hbase> t.get\_counter 'r1', 'c1'*

incr

Increments a cell ‘value’ at specified table/row/column coordinates. To increment a cell value in table ‘t1′ at row ‘r1′ under column ‘c1′ by 1 (can be omitted) or 10 do:

*hbase> incr 't1', 'r1', 'c1'*

*hbase> incr 't1', 'r1', 'c1', 1*

*hbase> incr 't1', 'r1', 'c1', 10*

The same commands also can be run on a table reference. Suppose you had a reference t to table ‘t1′, the corresponding command would be:hbase> t.incr ‘r1′, ‘c1′

*hbase> t.incr 'r1', 'c1', 1*

*hbase> t.incr 'r1', 'c1', 10*

put

Put a cell ‘value’ at specified table/row/column and optionally timestamp coordinates. To put a cell value into table ‘t1′ at row ‘r1′ under column ‘c1′ marked with the time ‘ts1′, do:

*hbase> put 't1', 'r1', 'c1', 'value', ts1*

The same commands also can be run on a table reference. Suppose you had a reference t to table ‘t1′, the corresponding command would be:

*hbase> t.put 'r1', 'c1', 'value', ts1*

scan

Scan a table; pass table name and optionally a dictionary of scanner specifications. Scanner specifications may include one or more of: TIMERANGE, FILTER, LIMIT, STARTROW, STOPROW, TIMESTAMP, MAXLENGTH, or COLUMNS, CACHEIf no columns are specified, all columns will be scanned. To scan all members of a column family, leave the qualifier empty as in ‘col\_family:’.The filter can be specified in two ways:

1. Using a filterString – more information on this is available in the Filter Language document attached to the HBASE-4176 JIRA

2. Using the entire package name of the filter.Some examples:

*hbase> scan '.META.'*

*hbase> scan '.META.', {COLUMNS => 'info:regioninfo'}*

*hbase> scan 't1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}*

*hbase> scan 't1', {COLUMNS => 'c1', TIMERANGE => [1303668804,1303668904]}*

*hbase> scan 't1', {FILTER => "(PrefixFilter ('row2') AND(QualifierFilter (>=, 'binary:xyz'))) AND (TimestampsFilter ( 123, 456))"}*

*hbase> scan 't1', {FILTER =>org.apache.hadoop.hbase.filter.ColumnPaginationFilter.NEW(1, 0)}*

For experts, there is an additional option — CACHE\_BLOCKS — which switches block caching for the scanner on (true) or off (false). By default it is enabled. Examples:

*hbase> scan 't1', {COLUMNS => ['c1', 'c2'],CACHE\_BLOCKS => FALSE}*

Also for experts, there is an advanced option — RAW — which instructs the scanner to return all cells (including delete markers and uncollected deleted cells). This option cannot be combined with requesting specific COLUMNS.

Disabled by default. Example:

*hbase> scan 't1', {RAW => TRUE, VERSIONS => 10}*

Besides the default ‘toStringBinary’ format, ‘scan’ supports custom formatting by column. A user can define a FORMATTER by adding it to the column name in the scan specification. The FORMATTER can be stipulated:

1. either as a org.apache.hadoop.hbase.util.Bytes method name (e.g, toInt, toString)

2. or as a custom class followed by method name: e.g. ‘c(MyFormatterClass).format’.

Example formatting cf:qualifier1 and cf:qualifier2 both as Integers:

*hbase> scan 't1', {COLUMNS => ['cf:qualifier1:toInt','cf:qualifier2:c(org.apache.hadoop.hbase.util.Bytes).toInt'] }*

Note that you can specify a FORMATTER by column only (cf:qualifer). You cannot specify a FORMATTER for all columns of a column family.

Scan can also be used directly from a table, by first getting a reference to a table, like such:

*hbase> t = get\_table 't'*

*hbase> t.scan*

Note in the above situation, you can still provide all the filtering, columns, options, etc as described above.

truncate

Disables, drops and recreates the specified table.

Examples:

*hbase>truncate 't1'*

1. **HBase Surgery tools**

assign

Assign a region. Use with caution. If region already assigned, this command will do a force reassign. For experts only. Examples:

*hbase> assign 'REGION\_NAME'*

balancer Trigger the cluster balancer. Returns true if balancer ran and was able to tell the region servers to unassign all the regions to balance (the re-assignment itself is async). Otherwise false (Will not run if regions in transition). Examples:

*hbase> balancer*

Enable/Disable balancer. Returns previous balancer state. Examples:

*hbase> balance\_switch TRUE*

*balance\_switch*

*hbase> balance\_switch FALSE*

close\_region

Close a single region. Ask the master to close a region out on the cluster or if ‘SERVER\_NAME’ is supplied, ask the designated hosting regionserver to close the region directly. Closing a region, the master expects’REGIONNAME’ to be a fully qualified region name. When asking the hosting regionserver to directly close a region,you pass the regions’ encoded name only.

A regionnamelookslikethis:TestTable,0094429456,1289497600452.527db22f95c8a9e0116f0cc13c680396.The

trailing period is part of the regionserver name. A region’s encoded name is the hash at the end of a region name; e.g. 527db22f95c8a9e0116f0cc13c680396 (without the period). A ‘SERVER\_NAME’ is its host, port plus startcode. For example: host187.example.com,60020,1289493121758 (find servername in master ui or when you do detailed status in shell). This command will end up running close on the region hosting regionserver. The close is done without the master’s involvement (It will not know of the close). Once closed, region will stay closed. Use assign to reopen/reassign. Use unassign or move to assign the region elsewhere on cluster. Use

with caution. For experts only. Examples:hbase> close\_region ‘REGIONNAME’

*hbase> close\_region 'REGIONNAME', 'SERVER\_NAME'*

Compact all regions in passed table or pass a region row to compact an individual region. You can also compact a single column family within a region.Examples: Compact all regions in a table:

*hbase> compact 't1'*

*compact*

*Compact an entire region:*

*hbase> compact 'r1'*

*Compact only a COLUMN family within a region:*

*hbase> compact 'r1', 'c1'*

*Compact a COLUMN family within a TABLE:*

*hbase> compact 't1', 'c1'*

flush

Flush all regions in passed table or pass a region row to flush an individual region. For

example:

*hbase> flush 'TABLENAME'*

*hbase> flush 'REGIONNAME'*

major\_compact Run major compaction on passed table or pass a region row to major compact an individual region. To compact a single column family within a region specify the region name followed by the column family name. Examples:

Compact all regions in a table:

*hbase> major\_compact 't1'*

*Compact an entire region:*

*hbase> major\_compact 'r1'*

*Compact a single COLUMN family within a region:*

*hbase> major\_compact 'r1', 'c1'*

*Compact a single COLUMN family within a TABLE:*

*hbase> major\_compact 't1', 'c1'*

move

split

unassign

hlog\_roll

zk\_dump

Move a region. Optionally specify target regionserver else we choose one at random.

NOTE: You pass the encoded region name, not the region name so this command is a little different to the others. The encoded region name is the hash suffix on region names: e.g. if the region name were TestTable,0094429456,1289497600452.527db22f95c8a9e0116f0cc13c680396. then the encoded region name portion is 527db22f95c8a9e0116f0cc13c680396

A server name is its host, port plus startcode.

For example:

host187.example.com,60020,1289493121758

Examples:

*hbase> move 'ENCODED\_REGIONNAME'*

*hbase> move 'ENCODED\_REGIONNAME', 'SERVER\_NAME'*

Split entire table or pass a region to split individual region. With the second parameter, you can specify an explicit split key for the region.

Examples:

*split 'tableName'*

*split 'regionName' # format: 'tableName,startKey,id'*

*split 'tableName', 'splitKey'*

*split 'regionName', 'splitKey'*

Unassign a region. Unassign will close region in current location and then reopen it again. Pass ‘true’ to force the unassignment (‘force’ will clear all in-memory state in master before the reassign. If results in double assignment use hbck -fix to resolve. To be used by experts).

Use with caution. For expert use only. Examples:

*hbase> unassign 'REGIONNAME'*

*hbase> unassign 'REGIONNAME', TRUE*

Roll the log writer. That is, start writing log messages to a new file. The name of the regionserver should be given as the parameter. A ‘server\_name’ is the host, port plus startcode of a regionserver. For example: host187.example.com,60020,1289493121758 (find servername in master ui or when you do detailed status in shell)

*hbase>hlog\_roll*

Dump status of HBase cluster as seen by ZooKeeper. Example:

*hbase>zk\_dump*

1. **Cluster replication tools**

add\_peer

Add a peer cluster to replicate to, the id must be a short and the cluster key is composed like this:

hbase.zookeeper.quorum:hbase.zookeeper.property.clientPort:zookeeper.znode.parent

This gives a full path for HBase to connect to another cluster.

Examples:

*hbase> add\_peer '1', "server1.cie.com:2181:/hbase"*

*hbase> add\_peer '2', "zk1,zk2,zk3:2182:/hbase-prod"*

*[codesyntax lang="oracle11"]*

*]*

*hbase> remove\_peer '1'*

[*List*](http://www.php.net/list) *all replication peer clusters.*

*hbase> list\_peers*

[/codesyntax]

remove\_peer

list\_peers

enable\_peer

disable\_peer

Stops the specified replication stream and deletes all the meta information kept about it. Examples:

*hbase> remove\_peer '1'*

*List ALL replication peer clusters.*

*hbase> list\_peers*

Restarts the replication to the specified peer cluster,continuing from where it was disabled.Examples:

*hbase> enable\_peer '1'*

Stops the replication stream to the specified cluster, but still keeps track of new edits to replicate.Examples:

*hbase> disable\_peer '1'*

Restarts all the replication features. The state in which each stream starts in is undetermined.

start\_replication

WARNING: start/stop replication is only meant to be used in critical load situations.

Examples:

*hbase> start\_replication*

Stops all the replication features. The state in which each stream stops in is undetermined.

stop\_replication

WARNING: start/stop replication is only meant to be used in critical load situations.

Examples:

*hbase> stop\_replication*

1. **Security tools**

grant

revoke

Grant users specific rights.

Syntax : grantpermissions is either zero or more letters from the set “RWXCA”.READ(‘R’), WRITE(‘W’), EXEC(‘X’), CREATE(‘C’), ADMIN(‘A’)For example:

*hbase>grant 'bobsmith', 'RWXCA'*

*hbase> GRANT 'bobsmith', 'RW', 't1', 'f1', 'col1'*

Revoke a user’s access rights.

Syntax : revoke

For example:

*hbase> REVOKE 'bobsmith', 't1', 'f1', 'col1'*

Show all permissions for the particular user.

user\_permission Syntax : user\_permission

For example:

*hbase> user\_permission*

*hbase> user\_permission 'table1'*

hbase Shell Commands to create table:

Hbase Shell Commands.Here below commands for create ‘airline’ table and also create column family and column qualifier and also insert values

here we are creating two rows only

**table name**                                    airline

**column family**

flightinfo                     **column qualifier**—flightno,airlines

 flightbetween          **column qualifier** —- source , destination

  time                         **column qualifier-**—arrival time,depature time,date

row key  —- row1 and row2

*CREATE 'airline', {NAME => 'flightinfo'}, {NAME => 'flightbetween'}, {NAME => 'time'}*

*put 'airline','row1','flightinfo:flightno','12345'*

*0 ROW(s) IN 0.0160 seconds*

*put 'airline','row1','flightinfo:airlines','indianairlines'*

*0 ROW(s) IN 0.0160 seconds*

*hbase(main):006:0> put 'airline','row1','flightbetween:source','hyd'*

*0 ROW(s) IN 0.0290 seconds*

*hbase(main):007:0> put 'airline','row1','flightbetween:destination','banglre'*

*0 ROW(s) IN 0.0120 seconds*

*hbase(main):008:0> put 'airline','row1','time:depaturetime','7am'*

*0 ROW(s) IN 0.0320 seconds*

*hbase(main):009:0> put 'airline','row1','time:arrivaltime','9am'*

*0 ROW(s) IN 0.0150 seconds*

*hbase(main):010:0> put 'airline','row1','time:date','20/05/2014'*

*0 ROW(s) IN 0.0440 seconds*

Hbase command for retrieve specify columns and row

*hbase(main):011:0> get 'airline','row1'*

*COLUMN CELL*

*flightbetween:desti TIMESTAMP=1411981750093, VALUE=banglre*

*nation*

*flightbetween:sourc TIMESTAMP=1411981724972, VALUE=hyd*

*e*

*flightinfo:airlines TIMESTAMP=1411981696986, VALUE=indianairlines*

*flightinfo:flightno TIMESTAMP=1411981668552, VALUE=12345*

*TIME:arrivaltime TIMESTAMP=1411981821497, VALUE=9am*

*TIME:DATE TIMESTAMP=1411981843455, VALUE=20/05/2014*

*TIME:depaturetime TIMESTAMP=1411981808445, VALUE=7am*

*7 ROW(s) IN 0.0940 seconds*

*hbase(main):027:0> get 'airline','row1','flightinfo:flightno','time:date'*

*COLUMN CELL*

*flightinfo:flightno TIMESTAMP=1411982109827, VALUE=12346*

*TIME:DATE TIMESTAMP=1411981843455, VALUE=20/05/2014*

*2 ROW(s) IN 0.0230 seconds*

Hbase command for to retrieve entire table

*hbase(main):028:0> scan 'airline'*

*ROW COLUMN+CELL*

*row1 COLUMN=flightbetween:destination, TIMESTAMP=1411981750093,*

*VALUE=banglre*

*row1 COLUMN=flightbetween:source, TIMESTAMP=1411981724972, valu*

*e=hyd*

*row1 COLUMN=flightinfo:airlines, TIMESTAMP=1411982131699, VALUE*

*=americanairlines*

*row1 COLUMN=flightinfo:flightno, TIMESTAMP=1411982109827, VALUE*

*=12346*

*row1 COLUMN=TIME:arrivaltime, TIMESTAMP=1411981821497, VALUE=9a*

*m*

*row1 COLUMN=TIME:DATE, TIMESTAMP=1411981843455, VALUE=20/05/201*

*4*

*row1 COLUMN=TIME:depaturetime, TIMESTAMP=1411981808445, VALUE=7*

*am*

*row2 COLUMN=flightbetween:destination, TIMESTAMP=1411982226629,*

*VALUE=banglre*

*row2 COLUMN=flightbetween:source, TIMESTAMP=1411982209701, valu*

*e=hyd*

*row2 COLUMN=flightinfo:airlines, TIMESTAMP=1411982193228, VALUE*

*=americanairlines*

*row2 COLUMN=flightinfo:flightno, TIMESTAMP=1411982183561, VALUE*

*=12346*

*row2 COLUMN=TIME:arrivaltime, TIMESTAMP=1411982277561, VALUE=10*

*am*

*row2 COLUMN=TIME:DATE, TIMESTAMP=1411982261000, VALUE=21/05/201*

*4*

*row2 COLUMN=TIME:depaturetime, TIMESTAMP=1411982244265, VALUE=8*

*am*

*2 ROW(s) IN 0.1110 seconds*

*hbase(main):030:0> scan 'airline',{COLUMNS => 'flightinfo:flightno'}*

*ROW COLUMN+CELL*

*row1 COLUMN=flightinfo:flightno, TIMESTAMP=1411982109827, VALUE*

*=12346*

*row2 COLUMN=flightinfo:flightno, TIMESTAMP=1411982183561, VALUE*

*=12346*

*2 ROW(s) IN 0.0710 seconds*

*hbase(main):004:0> scan 'airline',{ COLUMNS => ['flightinfo:airlines','time:date']}*

*ROW COLUMN+CELL*

*row1 COLUMN=flightinfo:airlines, TIMESTAMP=1411982131699, VALUE*

*=americanairlines*

*row1 COLUMN=TIME:DATE, TIMESTAMP=1411981843455, VALUE=20/05/201*

*4*

*row2 COLUMN=flightinfo:airlines, TIMESTAMP=1411982193228, VALUE*

*=americanairlines*

*row2 COLUMN=TIME:DATE, TIMESTAMP=1411982261000, VALUE=21/05/201*

*4*

*2 ROW(s) IN 0.0610 seconds*

Hbase command for count no of rows in table and describe a table

*hbase(main):045:0> COUNT 'airline'*

*2 ROW(s) IN 0.1290 seconds*

*hbase(main):001:0> describe 'airline'*

*DESCRIPTION ENABLED*

*'airline', {NAME => 'flightbetween', DATA\_BLOCK\_ENCODING => 'NONE', BLOOMFILTER => 'NONE', R TRUE*

*EPLICATION\_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL =>*

*'2147483647', KEEP\_DELETED\_CELLS => 'false', BLOCKSIZE => '65536', IN\_MEMORY => 'false', EN*

*CODE\_ON\_DISK => 'true', BLOCKCACHE => 'true'}, {NAME => 'flightinfo', DATA\_BLOCK\_ENCODING =>*

*'NONE', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'N*

*ONE', MIN\_VERSIONS => '0', TTL => '2147483647', KEEP\_DELETED\_CELLS => 'false', BLOCKSIZE =>*

*'65536', IN\_MEMORY => 'false', ENCODE\_ON\_DISK => 'true', BLOCKCACHE => 'true'}, {NAME => 'ti*

*me', DATA\_BLOCK\_ENCODING => 'NONE', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', VERSION*

*S => '3', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL => '2147483647', KEEP\_DELETED\_CELL*

*S => 'false', BLOCKSIZE => '65536', IN\_MEMORY => 'false', ENCODE\_ON\_DISK => 'true', BLOCKCAC*

*HE => 'true'}*

*1 ROW(s) IN 1.3010 seconds*