hbase Cheat Sheet

status

```
hbase(main):006:0> status
1 servers, 0 dead, 5.0000 average load
hbase(main):002:0> help 'status'
hbase> status
hbase> status 'simple'
hbase> status 'summary'
hbase> status 'detailed'
hbase> status 'replication'
hbase> status 'replication', 'source'
hbase> status 'replication', 'sink'
```

alter / alter_async

```
 # t1 t1 f1 f1  5.
 # t1 f1  f1 t1 f1   5.
hbase> alter 't1', NAME => 'f1', VERSIONS => 5
#
hbase> alter 't1', 'f1', {NAME => 'f2', IN_MEMORY => true}, {NAME =>
'f3', VERSIONS => 5}
#       ns1  t1  f1
hbase> alter 'ns1:t1', NAME => 'f1', METHOD => 'delete'
hbase> alter 'ns1:t1', 'delete' => 'f1' # t1MAX_FILESIZE

hbase> alter 't1', MAX_FILESIZE => '134217728'
#    t1 f2
hbase> alter 't1', CONFIGURATION =>
{'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}
hbase> alter 't1', {NAME => 'f2', CONFIGURATION =>
{'hbase.hstore.blockingStoreFiles' => '10'}} #

hbase> alter 't1', METHOD => 'tableatt_unset', NAME =>
'MAX_FILESIZE'
hbase> alter 't1', METHOD => 'tableatt_unset', NAME =>
'coprocessor$1'
#
hbase> alter 't1', { NAME => 'f1', VERSIONS => 3 },
{ MAX_FILESIZE => '134217728' }, { METHOD => 'delete', NAME =>
'f2' },
OWNER => 'johndoe', METADATA => { 'mykey' => 'myvalue' }
hbase(main):014:0>
```

append

```
#  t1 rowkey r1 c1   value
hbase> append 't1', 'r1', 'c1', 'value'
# t1 t11append
hbase> t11.append 'r1', 'c1', 'value'
```

count

```
#   t1
count 't1'
#   t1
# INTERVAL    rowkey 1000 CACHE     10
#       t1 10  1000
count 't1', INTERVAL => 10, CACHE => 1000
#
hbase> t.count
hbase> t.count INTERVAL => 100000
hbase> t.count CACHE => 1000
hbase> t.count INTERVAL => 10, CACHE => 1000
```

delete / deleteall

```
     #       ns1 t1 rowkeyr1c1 ts1
hbase> delete 'ns1:t1', 'r1', 'c1', ts1
#          t1 rowkey r1 c1 ts1
hbase> delete 't1', 'r1', 'c1', ts1
#
hbase> t.delete 'r1', 'c1', ts1
#     ns1 t1 rowkeyr1
hbase> deleteall 'ns1:t1', 'r1'
     #       t1 rowkey r1
 hbase> deleteall 't1', 'r1'
#     ns1 t1 rowkeyr1c1
hbase> deleteall 't1', 'r1', 'c1'
#        t1 rowkey r1 c1 ts1
hbase> deleteall 't1', 'r1', 'c1', ts1
#
hbase> t.deleteall 'r1' hbase>
t.deleteall 'r1', 'c1' hbase>
t.deleteall 'r1', 'c1', ts1
```

put

```
#      ns1 t1rowkey r1 c1
hbase> put 'ns1:t1', 'r1', 'c1', 'value'
#        t1 rowkeyr1c1 hbase>
put 't1', 'r1', 'c1', 'value'
# t1 rowkeyr1c1 ts1 hbase> put 't1',
'r1', 'c1', 'value', ts1
#        t1 rowkeyr1c1   ts1
hbase> put 't1', 'r1', 'c1', 'value', ts1, {ATTRIBUTES=>
{'mykey'=>'myvalue'}}
#
```

hbase cheatsheet Cheat Sheet

put (cont)

```
t .put 'r1', 'c1', 'value', ts1, {ATTRIBUTES=>{'mykey'=>'myvalue'}}
```

create

```
#     ns1 t1  f1 f1 5
hbase> create 'ns1:t1', {NAME => 'f1', VERSIONS => 5}
#       t1 f1,f2,f3
hbase> create 't1', {NAME => 'f1'}, {NAME => 'f2'}, {NAME => 'f3'}
#
hbase> create 't1', 'f1', 'f2', 'f3'
#   t1 f1 f1 1 TTL 2592000 BLOCKCACHE true
hbase> create 't1', {NAME => 'f1', VERSIONS => 1, TTL => 2592000,
BLOCKCACHE => true}
#    t1,f1 f1 hbase.hstore.blockingStoreFiles  10
hbase> create 't1', {NAME => 'f1', CONFIGURATION
=> {'hbase.hstore.blockingStoreFiles' => '10'}} #

hbase> create 'ns1:t1', 'f1', SPLITS => ['10', '20', '30', '40']
hbase> create 't1', 'f1', SPLITS => ['10', '20', '30', '40']
hbase> create 't1', 'f1', SPLITSFILE => 'splits.txt', OWNER =>
'johndoe' hbase> create 't1', {NAME => 'f1', VERSIONS => 5},
METADATA => { 'mykey' => 'myvalue' }
hbase> # Optionally pre-split the table into NUMREGIONS, using
hbase> # SPLITALGO ("HexStringSplit", "UniformSplit" or classname)
# Pre-splitting region
hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO =>
'HexStringSplit'}
hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO =>
'HexStringSplit', REGIONREPLICATION => 2, CONFIGURATION
=> {'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}}
# t2 t1 t1 t2 f1 hbase> t1
= create 't2', 'f1'
```

scan

```
#       hbase meta meta
hbase> scan 'hbase:meta'
#       hbase meta info regioninfo meta info regioninfo
hbase> scan 'hbase:meta', {COLUMNS => 'info:regioninfo'}
#       ns1t1 'c1' 'c2'   ns1t1 'c1' 'c2'
hbase> scan 'ns1:t1', {COLUMNS => ['c1', 'c2']}
#       ns1t1 'c1' 'c2'   ns1t1 'c1' 'c2'  10rowkey hbase> scan
'ns1:t1', {COLUMNS => ['c1', 'c2'], LIMIT => 10}
#       ns1t1 'c1' 'c2'   ns1t1 'c1' 'c2'  rowkey="xyz" 10 rowkey hbase> scan
'ns1:t1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}
```

scan (cont)

```
#       t1 c1 '1303668804''1303668904'
hbase> scan 't1', {COLUMNS => 'c1', TIMERANGE => [1303668804,
1303668904]}
#       t1
hbase> scan 't1', {REVERSED => true}
#       t1
hbase> scan 't1', {FILTER => "(PrefixFilter ('row2') AND (QualifierFilter
(>=, 'binary:xyz'))) AND (TimestampsFilter ( 123, 456))"} # RAW true t1

hbase> scan 't1', {RAW => true, VERSIONS => 10}
# t1
hbase> t11 = get_table 't1'
hbase> t11.scan
```

get

```
#     ns1    t1    rowkeyr1
hbase> get 'ns1:t1', 'r1'
#       t1 rowkey r1
hbase> get 't1', 'r1'
#       t1 rowkey r1  ts1ts2
hbase> get 't1', 'r1', {TIMERANGE => [ts1, ts2]}
#       t1 rowkey r1 c1
hbase> get 't1', 'r1', {COLUMN => 'c1'}
#       t1 rowkey r1 c1,c2,c3
hbase> get 't1', 'r1', {COLUMN => ['c1', 'c2', 'c3']}
#       t1 rowkey r1 c1 ts1
hbase> get 't1', 'r1', {COLUMN => 'c1', TIMESTAMP => ts1}
#       t1 rowkey r1 c1  ts1 ts2 4
hbase> get 't1', 'r1', {COLUMN => 'c1', TIMERANGE => [ts1, ts2],
VERSIONS => 4}
#
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}
```

other

```
truncate show_filters list drop_all drop disable_all disable whoami version
describe
```