vinysiq@LAPTOP-27CLHU1L:~/treinamento-bigdata/docker-bigdata$ docker exec -it hbase-master

"docker exec" requires at least 2 arguments.

See 'docker exec --help'.

Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container

vinysiq@LAPTOP-27CLHU1L:~/treinamento-bigdata/docker-bigdata$ docker exec -it hbase-master bash

root@hbase-master:/# hbase help

Error: Could not find or load main class help

root@hbase-master:/# hbase shell

2022-06-02 22:22:44,755 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

HBase Shell; enter 'help<RETURN>' for list of supported commands.

Type "exit<RETURN>" to leave the HBase Shell

Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017

hbase(main):001:0>

hbase(main):001:0> create 'controle',{NAME=>'produto'},{NAME=>'fornecedor'}

0 row(s) in 1.7170 seconds

=> Hbase::Table - controle

hbase(main):002:0> put 'controle', '1','produto:nome','ram'

0 row(s) in 0.3370 seconds

hbase(main):003:0> put 'controle', '2','produto:nome','hd'

0 row(s) in 0.0150 seconds

hbase(main):004:0> put 'controle', '3','produto:nome','mouse'

0 row(s) in 0.0120 seconds

```
hbase(main):005:0> scan 'controle'
ROW                 COLUMN+CELL
 1                  column=produto:nome, timestamp=1654209323584, value=ram
 2                  column=produto:nome, timestamp=1654209357097, value=hd
 3                  column=produto:nome, timestamp=1654209367753, value=mouse
3 row(s) in 0.0390 seconds

hbase(main):006:0> put 'controle', '1','produto:qtd','100'
0 row(s) in 0.0090 seconds

hbase(main):007:0> put 'controle', '2','produto:qtd','50'
0 row(s) in 0.0070 seconds

hbase(main):008:0> put 'controle', '3','produto:qtd','150'
0 row(s) in 0.0100 seconds

hbase(main):009:0> scan 'controle'
ROW                 COLUMN+CELL
 1                  column=produto:nome, timestamp=1654209323584, value=ram
 1                  column=produto:qtd, timestamp=1654209429572, value=100
 2                  column=produto:nome, timestamp=1654209357097, value=hd
 2                  column=produto:qtd, timestamp=1654209439683, value=50
 3                  column=produto:nome, timestamp=1654209367753, value=mouse
 3                  column=produto:qtd, timestamp=1654209448861, value=150
3 row(s) in 0.0330 seconds

hbase(main):010:0> put 'controle', '1','fornecedor:nome','TI Comp'
0 row(s) in 0.0180 seconds

hbase(main):011:0> put 'controle', '2','fornecedor:nome','Peças PC'
```

```
0 row(s) in 0.0120 seconds

hbase(main):012:0> put 'controle', '3','fornecedor:nome','Inf Tec'
0 row(s) in 0.0130 seconds

hbase(main):013:0> scan 'controle'
ROW                 COLUMN+CELL
 1                  column=fornecedor:nome, timestamp=1654209489150, value=TI Com
                    p
 1                  column=produto:nome, timestamp=1654209323584, value=ram
 1                  column=produto:qtd, timestamp=1654209429572, value=100
 2                  column=fornecedor:nome, timestamp=1654209503094, value=Pe\xC3
                    \xA7as PC
 2                  column=produto:nome, timestamp=1654209357097, value=hd
 2                  column=produto:qtd, timestamp=1654209439683, value=50
 3                  column=fornecedor:nome, timestamp=1654209518301, value=Inf Te
                    c
 3                  column=produto:nome, timestamp=1654209367753, value=mouse
 3                  column=produto:qtd, timestamp=1654209448861, value=150
3 row(s) in 0.0540 seconds

hbase(main):014:0> put 'controle', '1','fornecedor:estado','SP'
0 row(s) in 0.2950 seconds

hbase(main):015:0> put 'controle', '2','fornecedor:estado','MG'
0 row(s) in 0.0120 seconds

hbase(main):016:0> put 'controle', '3','fornecedor:estado','SP'
0 row(s) in 0.0110 seconds

hbase(main):017:0> scan 'controle'
```

```
ROW                COLUMN+CELL
 1                 column=fornecedor:estado, timestamp=1654209560205, value=SP
 1                 column=fornecedor:nome, timestamp=1654209489150, value=TI Com
                   p
 1                 column=produto:nome, timestamp=1654209323584, value=ram
 1                 column=produto:qtd, timestamp=1654209429572, value=100
 2                 column=fornecedor:estado, timestamp=1654209568988, value=MG
 2                 column=fornecedor:nome, timestamp=1654209503094, value=Pe\xC3
                   \xA7as PC
 2                 column=produto:nome, timestamp=1654209357097, value=hd
 2                 column=produto:qtd, timestamp=1654209439683, value=50
 3                 column=fornecedor:estado, timestamp=1654209575607, value=SP
 3                 column=fornecedor:nome, timestamp=1654209518301, value=Inf Te
                   c
 3                 column=produto:nome, timestamp=1654209367753, value=mouse
 3                 column=produto:qtd, timestamp=1654209448861, value=150
3 row(s) in 0.0670 seconds

hbase(main):018:0> list
TABLE
controle
1 row(s) in 0.0850 seconds

=> ["controle"]
hbase(main):019:0> describe 'controle'
Table controle is ENABLED
controle
COLUMN FAMILIES DESCRIPTION
{NAME => 'fornecedor', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false',
 KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', CO
MPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536
```

', REPLICATION_SCOPE => '0'}

{NAME => 'produto', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KE

EP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPR

ESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536',

REPLICATION_SCOPE => '0'}

2 row(s) in 0.0920 seconds

hbase(main):020:0> count 'controle'

3 row(s) in 0.0990 seconds

=> 3

hbase(main):021:0> scan 'controle'

ROW COLUMN+CELL

1 column=fornecedor:estado, timestamp=1654209560205, value=SP

1 column=fornecedor:nome, timestamp=1654209489150, value=TI Com

p

1 column=produto:nome, timestamp=1654209323584, value=ram

1 column=produto:qtd, timestamp=1654209429572, value=100

2 column=fornecedor:estado, timestamp=1654209568988, value=MG

2 column=fornecedor:nome, timestamp=1654209503094, value=Pe\xC3

\xA7as PC

2 column=produto:nome, timestamp=1654209357097, value=hd

2 column=produto:qtd, timestamp=1654209439683, value=50

3 column=fornecedor:estado, timestamp=1654209575607, value=SP

3 column=fornecedor:nome, timestamp=1654209518301, value=Inf Te

c

3 column=produto:nome, timestamp=1654209367753, value=mouse

3 column=produto:qtd, timestamp=1654209448861, value=150

3 row(s) in 0.0910 seconds

hbase(main):022:0> alter 'controle', {NAME=>'produto', VERSIONS=>3}

```
Updating all regions with the new schema...
0/1 regions updated.
1/1 regions updated.
Done.
0 row(s) in 3.5610 seconds

hbase(main):023:0> DESCRIBE 'controle'
NoMethodError: undefined method `DESCRIBE' for #<Object:0xee8e7ff>

hbase(main):024:0> describe 'controle'
Table controle is ENABLED
controle
COLUMN FAMILIES DESCRIPTION
{NAME => 'fornecedor', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false',
 KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', CO
MPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536
', REPLICATION_SCOPE => '0'}
{NAME => 'produto', BLOOMFILTER => 'ROW', VERSIONS => '3', IN_MEMORY => 'false', KE
EP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPR
ESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE => '65536',
REPLICATION_SCOPE => '0'}
2 row(s) in 0.0490 seconds

hbase(main):025:0> put 'controle','produto:qtd','200'

ERROR: wrong number of arguments (3 for 4)

Here is some help for this command:
Put a cell 'value' at specified table/row/column and optionally
timestamp coordinates.  To put a cell value into table 'ns1:t1' or 't1'
at row 'r1' under column 'c1' marked with the time 'ts1', do:
```

```
hbase> put 'ns1:t1', 'r1', 'c1', 'value'
hbase> put 't1', 'r1', 'c1', 'value'
hbase> put 't1', 'r1', 'c1', 'value', ts1
hbase> put 't1', 'r1', 'c1', 'value', {ATTRIBUTES=>{'mykey'=>'myvalue'}}
hbase> put 't1', 'r1', 'c1', 'value', ts1, {ATTRIBUTES=>{'mykey'=>'myvalue'}}
hbase> put 't1', 'r1', 'c1', 'value', ts1, {VISIBILITY=>'PRIVATE|SECRET'}
```

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, {ATTRIBUTES=>{'mykey'=>'myvalue'}}
```

```
hbase(main):026:0> put 'controle','2','produto:qtd','200'
0 row(s) in 0.0140 seconds

hbase(main):027:0> get 'controle','2', {COLUMNS=>'produto', VERSIONS=>2}
COLUMN              CELL
 produto:nome       timestamp=1654209357097, value=hd
 produto:qtd        timestamp=1654210034256, value=200
 produto:qtd        timestamp=1654209439683, value=50
3 row(s) in 0.0730 seconds

hbase(main):028:0> get 'controle','2', {COLUMNS=>{'produto:qtd','fornecedor:nome},
VERSIONS=>2}
hbase(main):029:2' get 'controle','2', {COLUMNS=>{'produto:qtd','fornecedor:nome},
VERSIONS=>2}
SyntaxError: (hbase):29: syntax error, unexpected tIDENTIFIER

get 'controle','2', {COLUMNS=>{'produto:qtd','fornecedor:nome}, VERSIONS=>2}
        ^
```

hbase(main):030:0> get 'controle','2', {COLUMNS=>{'produto:qtd','fornecedor:nome'}, VERSIONS=>2}

COLUMN CELL

ERROR: Failed parse column argument type {"COLUMNS"=>{"produto:qtd"=>"fornecedor:nome"}, "VERSIONS"=>2}, Hash

Here is some help for this command:

Get row or cell contents; pass table name, row, and optionally

a dictionary of column(s), timestamp, timerange and versions. Examples:

```
hbase> get 'ns1:t1', 'r1'
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']
hbase> get 't1', 'r1', {COLUMN => 'c1', ATTRIBUTES => {'mykey'=>'myvalue'}}
hbase> get 't1', 'r1', {COLUMN => 'c1', AUTHORIZATIONS => ['PRIVATE','SECRET']}
hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

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:qualifier). 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']
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

```
hbase(main):031:0> get 'controle','2', {COLUMNS=>{'produto:qtd','fornecedor:nome'},
VERSIONS=>2}
COLUMN                CELL

ERROR: Failed parse column argument type
{"COLUMNS"=>{"produto:qtd"=>"fornecedor:nome"}, "VERSIONS"=>2}, Hash
```

Here is some help for this command:

Get row or cell contents; pass table name, row, and optionally

a dictionary of column(s), timestamp, timerange and versions. Examples:

```
  hbase> get 'ns1:t1', 'r1'
  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']
  hbase> get 't1', 'r1', {COLUMN => 'c1', ATTRIBUTES => {'mykey'=>'myvalue'}}
  hbase> get 't1', 'r1', {COLUMN => 'c1', AUTHORIZATIONS => ['PRIVATE','SECRET']}
  hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE'}
  hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

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:qualifier). 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']
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

hbase(main):032:0> help 'get'

Get row or cell contents; pass table name, row, and optionally

a dictionary of column(s), timestamp, timerange and versions. Examples:

```
  hbase> get 'ns1:t1', 'r1'
  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']
  hbase> get 't1', 'r1', {COLUMN => 'c1', ATTRIBUTES => {'mykey'=>'myvalue'}}
  hbase> get 't1', 'r1', {COLUMN => 'c1', AUTHORIZATIONS => ['PRIVATE','SECRET']}
  hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE'}
  hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

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:qualifier). 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']
  hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE'}
  hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
hbase(main):033:0> get 'controle','2', {COLUMN=>{'produto:qtd','fornecedor:nome'},
VERSIONS=>2}
COLUMN              CELL

ERROR: Failed parse column argument type
{"COLUMN"=>{"produto:qtd"=>"fornecedor:nome"}, "VERSIONS"=>2}, Hash
```

Here is some help for this command:

Get row or cell contents; pass table name, row, and optionally

a dictionary of column(s), timestamp, timerange and versions. Examples:

```
  hbase> get 'ns1:t1', 'r1'
  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']
hbase> get 't1', 'r1', {COLUMN => 'c1', ATTRIBUTES => {'mykey'=>'myvalue'}}
hbase> get 't1', 'r1', {COLUMN => 'c1', AUTHORIZATIONS => ['PRIVATE','SECRET']}
hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

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:qualifier). 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']
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

```
hbase(main):034:0> get 'controle','2', {COLUMN=>{'produto:qtd','fornecedor:nome'}
VERSIONS=>2}
SyntaxError: (hbase):34: syntax error, unexpected tCONSTANT

get 'controle','2', {COLUMN=>{'produto:qtd','fornecedor:nome'} VERSIONS=>2}
                                                              ^

hbase(main):035:0> get 'controle','2', {COLUMN=>{'produto:qtd','fornecedor:nome'}
VERSION=>2}
SyntaxError: (hbase):35: syntax error, unexpected tCONSTANT

get 'controle','2', {COLUMN=>{'produto:qtd','fornecedor:nome'} VERSION=>2}
                                                              ^
```

hbase(main):036:0> get 'controle','2', {COLUMN=>{'produto:qtd','fornecedor:nome'}, VERSION=>2}

COLUMN CELL

ERROR: Failed parse column argument type {"COLUMN"=>{"produto:qtd"=>"fornecedor:nome"}, "1.8.7"=>2}, Hash

Here is some help for this command:

Get row or cell contents; pass table name, row, and optionally

a dictionary of column(s), timestamp, timerange and versions. Examples:

```
hbase> get 'ns1:t1', 'r1'
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']
hbase> get 't1', 'r1', {COLUMN => 'c1', ATTRIBUTES => {'mykey'=>'myvalue'}}
hbase> get 't1', 'r1', {COLUMN => 'c1', AUTHORIZATIONS => ['PRIVATE','SECRET']}
hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

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:qualifier). 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']
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

```
hbase(main):037:0> get 'controle'
```

ERROR: wrong number of arguments (1 for 2)

Here is some help for this command:
Get row or cell contents; pass table name, row, and optionally
a dictionary of column(s), timestamp, timerange and versions. Examples:

```
 hbase> get 'ns1:t1', 'r1'
 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']
 hbase> get 't1', 'r1', {COLUMN => 'c1', ATTRIBUTES => {'mykey'=>'myvalue'}}
 hbase> get 't1', 'r1', {COLUMN => 'c1', AUTHORIZATIONS => ['PRIVATE','SECRET']}
 hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE'}
 hbase> get 't1', 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

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:qualifier). 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']
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE'}
hbase> t.get 'r1', {CONSISTENCY => 'TIMELINE', REGION_REPLICA_ID => 1}
```

```
hbase(main):038:0> get 'controle', '1'
COLUMN              CELL
 fornecedor:estado    timestamp=1654209560205, value=SP
 fornecedor:nome      timestamp=1654209489150, value=TI Comp
 produto:nome         timestamp=1654209323584, value=ram
 produto:qtd          timestamp=1654209429572, value=100
```

```
4 row(s) in 0.0340 seconds

hbase(main):039:0> scan 'controle'
ROW                 COLUMN+CELL
 1                  column=fornecedor:estado, timestamp=1654209560205, value=SP
 1                  column=fornecedor:nome, timestamp=1654209489150, value=TI Com
                    p
 1                  column=produto:nome, timestamp=1654209323584, value=ram
 1                  column=produto:qtd, timestamp=1654209429572, value=100
 2                  column=fornecedor:estado, timestamp=1654209568988, value=MG
 2                  column=fornecedor:nome, timestamp=1654209503094, value=Pe\xC3
                    \xA7as PC
 2                  column=produto:nome, timestamp=1654209357097, value=hd
 2                  column=produto:qtd, timestamp=1654210034256, value=200
 3                  column=fornecedor:estado, timestamp=1654209575607, value=SP
 3                  column=fornecedor:nome, timestamp=1654209518301, value=Inf Te
                    c
 3                  column=produto:nome, timestamp=1654209367753, value=mouse
 3                  column=produto:qtd, timestamp=1654209448861, value=150
3 row(s) in 0.0690 seconds

hbase(main):040:0> scan 'controle', {COLUMNS=>'fornecedor:estado'}
ROW                 COLUMN+CELL
 1                  column=fornecedor:estado, timestamp=1654209560205, value=SP
 2                  column=fornecedor:estado, timestamp=1654209568988, value=MG
 3                  column=fornecedor:estado, timestamp=1654209575607, value=SP
3 row(s) in 0.0310 seconds

hbase(main):041:0> scan 'controle', {COLUMNS=>'fornecedor:estado', LIMIT => 5}
ROW                 COLUMN+CELL
 1                  column=fornecedor:estado, timestamp=1654209560205, value=SP
```

```
 2                column=fornecedor:estado, timestamp=1654209568988, value=MG
 3                column=fornecedor:estado, timestamp=1654209575607, value=SP
3 row(s) in 0.0160 seconds

hbase(main):042:0> deleteall 'controle','1'
0 row(s) in 0.1050 seconds

hbase(main):043:0> deleteall 'controle','3'
0 row(s) in 0.0090 seconds

hbase(main):044:0> count 'controle'
1 row(s) in 0.0250 seconds

=> 1
hbase(main):045:0> delete 'controle', '2', 'fornecedor:estado'
0 row(s) in 0.0290 seconds

hbase(main):046:0> count 'controle'
1 row(s) in 0.0200 seconds

=> 1
hbase(main):047:0> scan 'controle'
ROW                COLUMN+CELL
 2                column=fornecedor:nome, timestamp=1654209503094, value=Pe\xC3
                  \xA7as PC
 2                column=produto:nome, timestamp=1654209357097, value=hd
 2                column=produto:qtd, timestamp=1654210034256, value=200
1 row(s) in 0.0290 seconds

hbase(main):048:0>
```