**HBase（二） HBase Shell**

【摘要】 Hbase Shell...

**Hbase Shell**

**Shell命令规则**

ü 表名、列名用单引号或双引号包裹   
ü 支持二进制、八进制、十六进制输入输出，但是必须用双引号，否则认为是文本   
ü 参数用逗号分隔   
ü 键值属性用Ruby散列的形式，键值间用”=>”分隔，整体被包在大括号中，键一般不需要双引号，是关键字，如 {COLUMN =>   
‘c1′, TIMERANGE => [ts1, ts2], VERSIONS => 4}   
ü 限制输出长度，MAXLENGTH，如get ‘t1’,’r1’, MAXLENGTH=>60   
ü Shell中删除字符，ctrl+backspace （和sqlplus一样)

**交互式与命令式:**

**交互式**

进入hbase shell，可以执行单行命令，也可以利用JRuby语法写一些简单的脚本：   
hbase(main):005:0> java.text.SimpleDateFormat.new("yyyy/MM/dd HH:mm:ss").parse( "2011/05/30 20:56:29").getTime();   
=> 1306760189000;   
hbase(main):006:0> for i in 'a'..'z' do for j in 'a'..'z' do put 't1', "row-#{i}#{j}", "f1:#{j}", "#{j}" end end;   
0 row(s) in 0.0030 seconds;   
……;   
=> "a".."z"   
require 'date';   
import java.lang.Long;   
import org.apache.hadoop.hbase.util.Bytes;   
(Date.new(2015, 01, 01)..Date.today).each {\;   
|x| put "t1", "daily", "f1:" + x.strftime("%Y%m%d"),\   
Bytes.toBytes(Long.new(rand \* \   
4000).longValue).to\_a.pack("CCCCCCCC") }

**命令式**

可以使用管道：   
pmapp2:/opt/netwatcher/pm4h2/app/opt/hbase-0.94.9/conf> echo "status" | hbase shell   
HBase Shell; enter 'help' for list of supported commands.   
Type "exit" to leave the HBase Shell   
Version 0.94.9, r1496217, Mon Jun 24 20:57:30 UTC 2013   
status   
3 servers, 0 dead, 26.0000 average load   
也可以使用文件

**命令全集**

COMMAND GROUPS:   
Group name: general   
Commands: status, version, whoami   
Group name: ddl   
Commands: alter, alter\_async, alter\_status, create, describe, disable, disable\_all, drop, drop\_all, enable, enable\_all, exists, is\_disabled,   
is\_enabled, list, show\_filters   
Group name: dml   
Commands: count, delete, deleteall, get, get\_counter, incr, put, scan, truncate   
Group name: tools   
Commands: assign, balance\_switch, balancer, close\_region, compact, flush, hlog\_roll, major\_compact, move, split, unassign, zk\_dump   
Group name: replication   
Commands: add\_peer, disable\_peer, enable\_peer, list\_peers, remove\_peer, start\_replication, stop\_replication   
Group name: snapshot   
Commands: clone\_snapshot, delete\_snapshot, list\_snapshots, restore\_snapshot, snapshot   
Group name: security   
Commands: grant, revoke, user\_permission

**General命令**

**help**

进入shell，输入help命令，会得到帮助信息   
help ‘’获取单独命令的详细帮助，如help ‘status’

**debug**

默认shell的日志级别为ERROR，通过这个命令切换调试模式   
hbase(main):002:0> debug   
Debug mode is ON   
hbase(main):003:0> debug   
Debug mode is OFF

**status**

status 命令返回ClusterStatus类中的各级别信息，分三种级别simple/summary/detailed   
hbase(main):008:0> status   
3 servers, 0 dead, 25.3333 average load   
hbase(main):009:0> status 'summary'   
3 servers, 0 dead, 25.3333 average load   
hbase(main):010:0> status 'simple'   
3 live servers   
pmapp3:15241 1437987453096   
requestsPerSecond=0, numberOfOnlineRegions=31, usedHeapMB=1404, maxHeapMB=4044   
pmapp2:15241 1437987453160   
requestsPerSecond=0, numberOfOnlineRegions=19, usedHeapMB=959, maxHeapMB=4044   
pmeam1:15241 1437987452789   
requestsPerSecond=0, numberOfOnlineRegions=26, usedHeapMB=589, maxHeapMB=4044   
0 dead servers Aggregate load: 0, regions: 76   
hbase(main):007:0> status 'detailed'   
version 0.94.9   
0 regionsInTransition   
master coprocessors: []   
3 live servers   
pmapp3:15241 1437987453096   
requestsPerSecond=0, numberOfOnlineRegions=31, usedHeapMB=1399, maxHeapMB=4044   
12H\_20150725\_0000,,1437797703299.31e4dc69db44be43274ab80f0b0355e5.   
numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=28, storefileSizeMB=28, compressionRatio=1.0000,   
memstoreSizeMB=0, storefileIndexSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=23, totalStaticIndexSizeKB=16,   
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN   
……

**version**

返回当前版本信息、仓库版本、编译信息，即ClusterStatus.getHBaseVersion()   
hbase(main):012:0> version   
0.94.9, r1496217, Mon Jun 24 20:57:30 UTC 2013

**exit**

退出shell

**DDL命令**

**create**

即createTable()方法，可以选择多个参数，但表名和列簇名是必须的参数，其它参数还包括版本数、TTL以及预分Region建表的   
key数组等，如：   
hbase> create 't1', {NAME => 'f1', VERSIONS => 5}   
hbase> create 't1', {NAME => 'f1'}, {NAME => 'f2'}, {NAME => 'f3'}   
hbase> # 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', 'f1', {SPLITS => ['10', '20', '30', '40']}   
hbase> create 't1', 'f1', {SPLITS\_FILE => 'splits.txt'}   
hbase> # Optionally pre-split the table into NUMREGIONS, using   
hbase> # SPLITALGO ("HexStringSplit", "UniformSplit" or classname)   
hbase> create 't1', 'f1', {NUMREGIONS => 15, SPLITALGO => 'HexStringSplit'}   
hbase(main):013:0> create 't1','r1'   
0 row(s) in 1.1480 seconds

**alter**

修改表结构，即modifyTable()，修改前需要disable表   
hbase(main):033:0> disable 't1'   
0 row(s) in 2.1280 seconds   
hbase(main):034:0> alter 't1', NAME => 'f1', VERSIONS => 5   
Updating all regions with the new schema...   
1/1 regions updated.   
Done.   
0 row(s) in 1.2150 seconds   
hbase(main):040:0> alter 't1', NAME => 'r1', METHOD => 'delete'   
Updating all regions with the new schema...   
1/1 regions updated.   
Done.   
0 row(s) in 1.1820 seconds   
hbase(main):042:0> describe 't1'   
DESCRIPTION ENABLED   
't1', {NAME => 'f1', DATA\_BLOCK\_ENCODING => 'NONE', BLOOMFILTER => 'NONE', REPLICATION\_SCOPE => '0', COMPRESSION => 'NONE',   
VERSIONS => '5', TTL => '21 false   
47483647', MIN\_VERSIONS => '0', KEEP\_DELETED\_CELLS => 'false', BLOCKSIZE => '65536', ENCODE\_ON\_DISK => 'true', IN\_MEMORY =>   
'false', BLOCKCACHE => 'tru   
e'}   
1 row(s) in 0.1510 seconds

**describe**

查看表机构：打印HTableDescriptor，类似oracle的desc。上面已有示例

**disable**

禁用表，调用disableTable()方法。上面已有示例

**enable**

启用表，调用enableTable()。上面已有示例

**exists**

查看表是否存在。tableExists()   
hbase(main):043:0> exists 't1'   
Table t1 does exist   
0 row(s) in 0.0240 seconds

**is\_disabled**

表是否启用isTableEnabled()   
hbase(main):046:0> is\_enabled 't1'   
false   
0 row(s) in 0.0180 seconds

**list**

列出所有表listTables()   
hbase(main):001:0> list   
TABLE   
12H\_20150722\_0000   
t1   
……   
55 row(s) in 0.7340 seconds

**drop**

删除表deleteTable()   
hbase(main):002:0> drop 't1'   
0 row(s) in 1.2100 seconds   
hbase(main):003:0> exists 't1'   
Table t1 does not exist

**DML命令**

**put**

使用java api的Put类，插入数据。必选参数是表名、RowKey、列名（包括列簇和列名）和值，可选参数包括时间戳put 't1', 'r1',   
'c1', 'value', ts1   
hbase(main):016:0> put 't1','row1','f1:c2','value2'   
0 row(s) in 0.0060 seconds

**get**

Get类。单行查询，必选参数是表名和RowKey，可选参数包括列名（包括列簇和列名）、时间戳、版本数等语法：   
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(main):019:0> get 't1','row1'   
COLUMN CELL   
f1:c2 timestamp=1438152790698,   
value=value2   
f1:column1 timestamp=1438151985660,   
value=value1   
2 row(s) in 0.0140 seconds

**scan**

多行查询，依赖Scan类。必选参数是表名，可选参数包括列名（包括列簇和列名）、起止key、Filter，语法：   
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)}   
hbase(main):021:0> scan 't1'   
ROW COLUMN+CELL   
 row1 column=f1:c2, timestamp=1438152790698,   
value=value2   
row1 column=f1:column1, timestamp=1438151985660,   
value=value1   
row2 column=f1:c2, timestamp=1438152770375, value=value2

**delete**

单列删除，依赖Delete类。必选参数table/row/column，可选参数时间戳   
语法：delete 't1', 'r1', 'c1', ts1   
hbase(main):042:0> delete 't1','row1','f1:column1'   
0 row(s) in 0.0170 seconds

**deleteall**

可以删除一个列族或一列，也依赖Delete类。必选参数表、行，可选参数列、时间戳   
hbase> deleteall 't1', 'r1'   
hbase> deleteall 't1', 'r1', 'c1'   
hbase> deleteall 't1', 'r1', 'c1', ts1

**truncate**

清理表中数据，相当于disable drop create在同一个模式下的顺序使用   
hbase(main):045:0> truncate 't1'   
Truncating 't1' table (it may take a while):   
- Disabling table...   
- Dropping table...   
- Creating table...   
0 row(s) in 4.4000 seconds

**count**

统计一个表的行数，内部使用了Scan   
hbase(main):047:0> count 't1'   
0 row(s) in 0.0550 seconds

**incr**

操作计数器，默认加一，可以操作正数、负数、零。依赖Increment类。下面的命令会创建新的r1行和f1:1列，不用实现创建好。   
hbase(main):051:0> incr 't1', 'r1', 'f1:1'   
COUNTER VALUE = 1   
hbase(main):052:0> incr 't1', 'r1', 'f1:1'   
COUNTER VALUE = 2   
hbase(main):055:0> incr 't1', 'r1', 'f1:1' ,5   
COUNTER VALUE = 7

**get\_counter**

返回一个计数器的值，和get类似，但是把计数器的值转换为可读的形式，也依赖Get类   
hbase(main):004:0> get\_counter 't1','r1','f1:1'   
COUNTER VALUE = 7   
hbase(main):005:0> get 't1','r1','f1:1'   
COLUMN CELL   
f1:1 timestamp=1438153769249, value=\x00\x00\x00\x00\x00\x00\x00\x07   
1 row(s) in 0.0350 seconds

**Tool命令**

**assign**

java api 的assign() 方法，分配一个region到一台服务器中。

**balance\_switch**

balanceSwitch() 方法，切换负载均衡开关

**balancer**

balancer() ，启动负载均衡

**close\_region**

closeRegion() ，关闭一个region

**compact**

compact()，开启某个region或一张表的异步合并

**flush**

flush()，开启某个region或一张表的异步flush

**major\_compact**

majorCompact()，开启某个region或一张表的major合并

**move**

move()，移动一个region到不同的服务器中

**split**

split()，拆分一个region或一个表

**unassign**

unassign()，下线一个region

**zk\_dump**

把zookeeper的信息dump到hbase里，这是内部类提供的特殊功能，Hbase Master 的Web UI也有这个功能

作者 | 林钰鑫

转载请注明出处：华为云博客 https://portal.hwclouds.com/blogs