HBase

概述

HBase是Google的开源项目。

HBase是一个具备高可靠性/高性能/面向列/可伸缩的分布式存储系统。可以利用HBase技术在老旧的或廉价的PC Server 端搭建起大规模结构化存储集群

HBase的目标是存储并处理大型的数据，更具体的来说，就是仅需使用普通的硬件拍照，就可以处理成千山万的行和列所组成的大型数据。

HBase特点

1.海量存储

Hbase适合存储PB级的海量数据，在PB级别的数据以及采用廉价PC存储的情况下，能在几十到几百毫秒内返回数据。

2.列式存储

也叫列族存储，HBase适合根据列族来存储数据的。列族下面可以有非常多的列，列族在创建表的时候就必须指定

3.极易扩展

HBase的扩展性主要体现在两方面

一个是基于上层处理能力的扩展

另一个是基于存储的扩展

4.高并发

由于大部分使用HBase的架构，都是采用廉价的PC，因此单个IO的延迟其实并不大，一半在几十到上百毫秒之间。最终起到一个 高并发，低延迟的效果。

5.稀疏

稀疏主要是针对HBase列的灵活性，在列族中，可以指定任意多的列，在列数据为空的情况下，不会占用存储的空间。

HBase中的角色

HMaster

功能：

```
1. 监控RegionServer
1. 处理RegionServer故障转移
1. 处理元数据的变更
1. 处理region的分配和转移
1. 在空闲时间进行数据的负载均衡
1. 通过Zookeeper发布自己的位置给客户端
```

RegionServer

功能：

1. 负责存储HBase的实际数据
2. 处理分配给它的Region
3. 刷新缓存到HDFS
4. 维护Hlog
5. 执行压缩
6. 负责处理Region分片

Region （分片）

分片，HBase表回根据RowKey值被切分成不同的Region存储到RegionServer中，在一个RegionServer中可以有多个不同的Region

Store

HFile存储在Store中，一个Store对应HBase表中的一个列族

HFile

在磁盘上保存原始数据的实际物理文件，是实际存储的文件。StoreFile是以Hfile的形式存储在HDFS中

MemStore

内存存储，用来保存当前数据操作

HBase的部署

前提：

Zookeeper集群是正常部署的

Hadoop集群正常部署

1. 下载HBase的安装包 hbase-2.2.0-bin.tar.gz 到 /opt/module/ 目录

```
-- 将宿主主机中的 /opt/moudle/hbase-2.2.0-bin.tar.gz 复制到 docker容器 （hadoop101容器）中
docker cp /opt/module/hbase-2.2.0-bin.tar.gz  hadoop101:/opt/module/
```

2. 进入容器

```
docker exec -it hadoop101 /bin/bash
```

3. 解压 hbase的安装包 到 /opt/software 中

```
tar -xvf /opt/module/hbase-2.2.0-bin.tar.gz  -C /opt/software/
```

4. HBase的配置

    1. 配置 /opt/software/hbase-2.2.0/conf/hbase-env.sh 文件

```
-- 文件最后添加内容
export JAVA_HOME=/opt/software/jdk1.8.0_212
export HBASE_MANAGES_ZK=false
```

    2. 配置 /opt/software/hbase-2.2.0/conf/hbase-site.xml 文件

```
<configuration>
        <property>
                <name>hbase.rootdir</name>
                <value>hdfs://hadoop101:9000/hbase</value>
        </property>
        <property>
                <name>hbase.zookeeper.quorum</name>
                <value>hadoop101:2181,hadoop102:2181,hadoop103:2181</value>
        </property>
        <property>
                <name>hbase.zookeeper.property.dataDir</name>
                <value>/opt/software/apache-zookeeper-3.8.4-bin/zkData</value>
        </property>
        <property>
                <name>hbase.cluster.distributed</name>
                <value>true</value>
        </property>
        <property>
                <name>hbase.master.port</name>
                <value>16000</value>
        </property>
</configuration>
```

    3. 配置 /opt/software/hbase-2.2.0/conf/regionservers 文件

```
hadoop101
hadoop102
hadoop103
```

    4. 软连接 hadoop 中的 core-site.xml 配置文件 到 hbase 配置目录中

```
ln -s /opt/software/hadoop-3.1.3/etc/hadoop/core-site.xml  /opt/software/hbase-2.2.0/conf/core-site.xml

ln -s /opt/software/hadoop-3.1.3/etc/hadoop/hdfs-site.xml  /opt/software/hbase-2.2.0/conf/hdfs-site.xml
```

```
[root@hadoop101 conf]# ll
total 40
lrwxrwxrwx. 1 root root   51 Mar  3 13:12 core-site.xml -> /opt/software/hadoop-3.1.3/etc/hadoop/core-site.xml
-rw-r--r--. 1 1001 1001 1811 Jun 11  2019 hadoop-metrics2-hbase.properties
-rw-r--r--. 1 1001 1001 4271 Jun 11  2019 hbase-env.cmd
-rw-r--r--. 1 1001 1001 7612 Mar  3 12:56 hbase-env.sh
-rw-r--r--. 1 1001 1001 2257 Jun 11  2019 hbase-policy.xml
-rw-r--r--. 1 1001 1001 1465 Mar  3 13:07 hbase-site.xml
lrwxrwxrwx. 1 root root   51 Mar  3 13:13 hdfs-site.xml -> /opt/software/hadoop-3.1.3/etc/hadoop/hdfs-site.xml
-rw-r--r--. 1 1001 1001 4977 Jun 11  2019 log4j.properties
-rw-r--r--. 1 1001 1001   30 Mar  3 13:11 regionservers
```

5. 复制 hbase的安装目录到其他的容器中

```
scp -r /opt/software/hbase-2.2.0/  hadoop102:/opt/software/
scp -r /opt/software/hbase-2.2.0/  hadoop103:/opt/software/
```

6. 添加环境变量

```
-- 配置 /etc/profile
vi /etc/profile
```

文件最后添加内容

```
export HBASE_HOME=/opt/software/hbase-2.2.0
export PATH=$PATH:$HBASE_HOME/bin
```

保存退出，并重新加载配置文件

```
source /etc/profile
```

复制当前的环境变量配置文件到 其他容器

```
scp /etc/profile  hadoop102:/etc/
scp /etc/profile  hadoop103:/etc/
```

7. 启动hbase

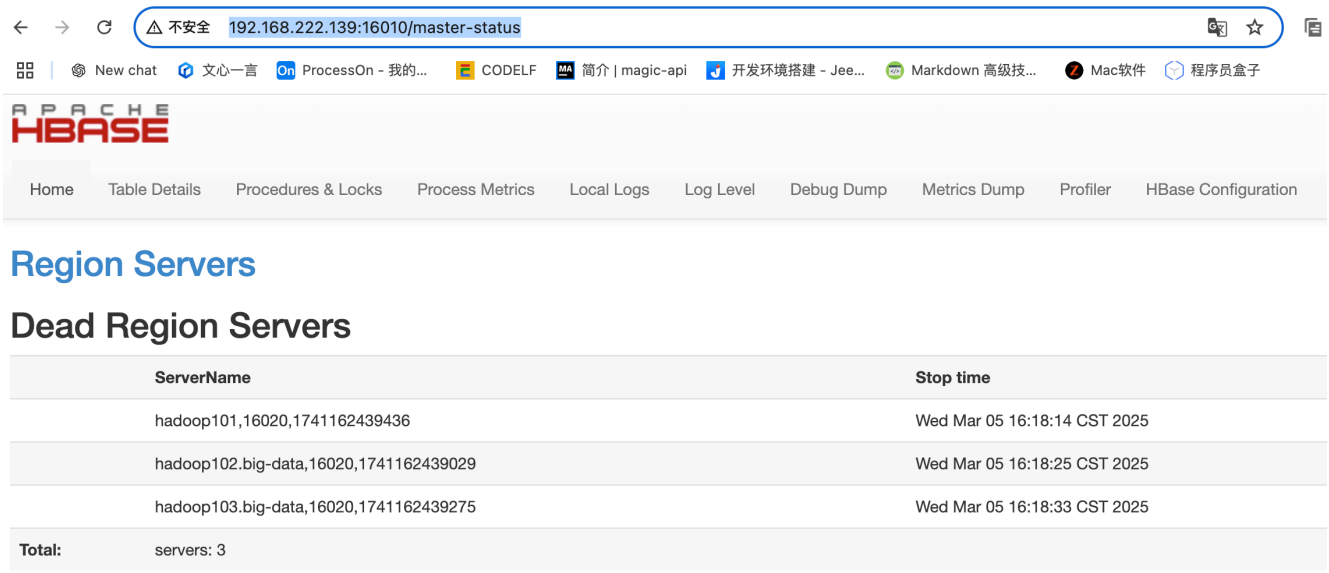
```
start-hbase.sh
```

8. 查看是否启动成功

```
jsp
```

```
[root@hadoop101 /]# jps
528 NameNode
720 DataNode
4020 HRegionServer
4821 ZooKeeperMain
1142 ResourceManager
2935 QuorumPeerMain
3547 HMaster
5278 Jps
1311 NodeManager
```

9. 访问网页端口16010



10. 进入客户端命令行模式

```
hbase shell
```

11. 查看集群运行状态

```
status 'detailed'
```

```
status
```

注意： 如果说 HMaster启动失败，则可以通过以下两个操作解决

a. 删除zk中的/hbase

进入 zk客户端 ， 执行 deleteall /hbash

b. 删除hdfs中的 /hbase

执行 hdfs dfs -rm -r /hbase

容器构建

```
docker network create --subnet 192.168.10.0/24 --gateway 192.168.10.254  big-data
```

```
docker run -id --name hadoop101 --net big-data --ip 192.168.10.101 --hostname hadoop101 -p 8088:8088  -p 16010:16010 -p 16030:16030 --privileged vcit/hadoop101:v2.0 /usr/sbin/init
```

```
docker run -id --name hadoop102 --net big-data --ip 192.168.10.102 --hostname hadoop102  --privileged vcit/hadoop102:v2.0 /usr/sbin/init
```

```
docker run -id --name hadoop103 --net big-data --ip 192.168.10.103 --hostname hadoop103  --privileged vcit/hadoop103:v2.0 /usr/sbin/init
```

HBase的使用

1. 进入客户端

```
hbase shell
```

2. 查看当前数据库中有哪些表

```
list
```

```
hbase(main):003:0> list
TABLE
0 row(s)
Took 0.0340 seconds
=> []
```

当前数据库中有没有表

3. 新建表

```
create 'student','name','sex','age','dept','course'
```

```
hbase(main):016:0> create 'student','name','sex','age','dept','course'
Created table student
Took 2.2524 seconds
=> Hbase::Table - student
```

4. 查看表结构

```
describ 'student'
```

```
hbase(main):017:0> describe 'student'
Table student is ENABLED
student
COLUMN FAMILIES DESCRIPTION
{NAME => 'age', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FA
LSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
 => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFE
TCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'course', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SC
OPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PR
EFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'dept', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'F
ALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOP
E => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREF
ETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'name', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'F
ALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOP
E => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREF
ETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'sex', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'FA
LSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
 => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFE
TCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
```

5. 插入数据到表

```
put 'student','name:xiaobai','sex:male','age:16'
put 'student','name:xiaohei','sex:female','age:17'
put 'student','name:dalan','sex:female','age:19'
put 'student','name:dabai','sex:male','age:21'
```

6. 查看表数据

```
scan 'student'
```

```
hbase(main):022:0> scan 'student'
ROW                             COLUMN+CELL
 name:dabai                     column=sex:male, timestamp=1741224747490, value=age:21
 name:dalan                     column=sex:female, timestamp=1741224736092, value=age:19
 name:xiaobai                   column=sex:male, timestamp=1741224665341, value=age:16
 name:xiaohei                   column=sex:female, timestamp=1741224725368, value=age:17
4 row(s)
Took 0.0760 seconds
```

7. 统计数据

```
count  'student'
```

```
hbase(main):023:0> count 'student'
4 row(s)
Took 0.0440 seconds
=> 4
```

8. 删除数据

    1. 删除所有表数据

```
truncate 'student'
```

2. 删除表

```
-- 将表的状态改为 disable
disable 'student'
-- 删除表
drop 'student'
```

HBase应用常用名词

常用名词解释

NameSpace 命名空间，相当于关系型数据库（MySQL）中的数据库（database）的概念。每个命名空间下有多个表。HBase默认自带的命名空间 hbase 和 default， 其中hbase中存放的是Hbase内置的表，default是用户默认使用的命名空间

Row 表中的每行数据被称为 行（Row），由一个RowKey 和 多个 Column 组成，数据是按照RowKey的字典顺序存储的，并且查询是只能根据RowKey进行检索，因此 RowKey的设计非常关键。

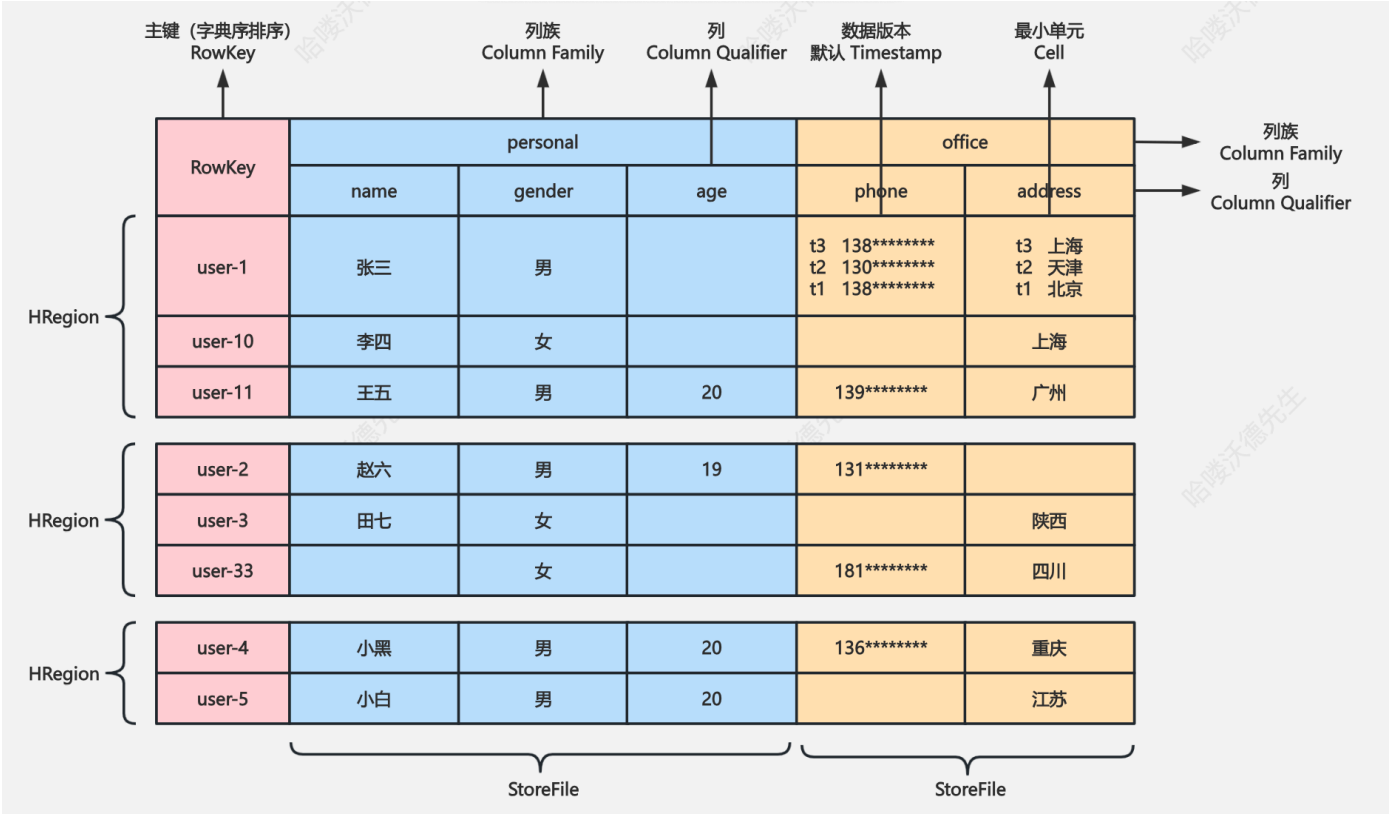
Column 是由列族（Column Family） 和 列限定符 （Column Qualifier）进行限定。 建表的时候只需要定义列族，而列限定符无需定义

Cell 某行中的某一列被称为Cell （单元格），由 rowkey（确定行），column family：column qualifier（确定列），timestamp（确定时间）最终确定具体单元。 Cell中没有具体的类型，全部都是字节码的形式（字节数组）存储

Timestamp 用于标识数据的不同版本（version），每条数据写入的时候，如果不指定时间戳，系统会自动为其加上该字段，值就是HBase的时间

Region 分片，HBase表会根据RowKey值被切分成不同的Region存储到RegionServer中，在一个RegionServer中可以有多个不同的Region

HBase的数据模型



逻辑上，HBase的数据模型同关系型数据库很类似，数据存储在一张表中，有行有列。但从底层物理存储结构来看，其实就是一个MAP

操作命令

namespace的操作

```
-- Group name: namespace
-- Commands: alter_namespace, create_namespace, describe_namespace, drop_namespace, --
list_namespace, list_namespace_tables

-- 新建namespace
create_namespace 'test'

-- 删除namespace
drop_namespace 'test'

-- 查看某个namespace中有哪些表
list_namespace_tables 'hbase'
```

table的操作

```
-- Group name: ddl
-- Commands: alter, alter_async, alter_status, clone_table_schema, create, describe,
disable, disable_all, drop, drop_all, enable, enable_all, exists, get_table, is_disabled,
is_enabled, list, list_regions, locate_region, show_filters

-- 新建表语法
```

```
create 'namespace:tablename', {NAME => '列族名'}
说明:
namespace 如果是default，可以省略

-- 新建表实操
create 'student',{NAME => 'stuinfo'}
说明:
student  表名
stuinfo  列族名，如果是多个列族，使用逗号分隔

--查看当前namespace中所有表
list
```

记录操作

```
-- Group name: dml
-- Commands: append, count, delete, deleteall, get, get_counter, get_splits, incr, put,
scan, truncate, truncate_preserve

-- 添加记录  rowkey是001的学生name是xiaobai， sex是female
put 'student', '001', 'stuinfo:name','xiaobai'
put 'student', '001', 'stuinfo:sex', 'female'
put 'student', '002', 'stuinfo:name','xiaohei'
put 'student', '002', 'stuinfo:sex', 'male'
put 'student', '003', 'stuinfo:name','xiaolan'
put 'student', '003', 'stuinfo:sex', 'male'
put 'student', '004', 'stuinfo:name','xiaohui'
put 'student', '004', 'stuinfo:sex', 'male'
put 'student', '005', 'stuinfo:name','xiaohong'
put 'student', '005', 'stuinfo:sex', 'female'


-- 查看某个rowkey下的信息
-- 语法  get 'namespace:tablename','rowkey'
get 'student','001'

-- 查看某个rowkey下某个列族的数据
-- 语法  get 'namespace:tablename','rowkey', 'column_family'
get 'student','001','stuinfo:name'

-- 查看表中的总记录数
count 'student'

-- 查看表中所有的数据
scan  'student'

-- 通过 rowkey 过滤查询
-- 过滤查询 rowkey 002(包含)-004(不包含) 的数据
scan 'student',{STARTROW => '002', STOPROW => '004'}
```

```
-- 过滤查询 rowkey 004(包含)之后的所有数据
scan 'student',{STARTROW => '004'}

-- 限制查询的行数
scan 'student',{LIMIT => 2}

-- 查询 rowkey 前缀匹配（可以完全匹配）的数据
scan 'student',{FILTER => "PrefixFilter('001')"}

-- 查询 rowkey 中包含某个字符串的结果
scan 'student',{FILTER => "RowFilter(=,'substring:04')"}

-- 查询 列名name 是xiaohong的列族数据   SingleColumnValueFilter
scan 'student',{FILTER => "SingleColumnValueFilter('stuinfo','name',=,'binary:xiaohong')"}

-- 查询列值中包含 lan 的列数据，这里指返回符合条件的列数据，不是列族数据  ValueFilter
scan 'student',{FILTER => "ValueFilter(=, 'substring:lan')"}

-- 查询结果返回键值对的个数限制  ColumnCountGetFilter
get 'student','002',{FILTER => 'ColumnCountGetFilter(1)'}
```

课堂练习

1. 进入hbase环境中

```
hbase shell
```

```
[root@hadoop103 /]# hbase shell
2025-03-06 15:11:40,029 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using b
uiltin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.2.0, rUnknown, Tue Jun 11 04:30:30 UTC 2019
Took 0.0044 seconds
hbase(main):001:0>
```

2. 罗列出所有的命名空间

```
list_namespace
```

3. 创建命名空间 vcit

```
 create_namespace 'vcit'
```

4. 创建表

```
create 'vcit:student','info','score'
```

```
hbase(main):007:0> create 'vcit:student','info','score'
Created table vcit:student
Took 2.3395 seconds
=> Hbase::Table - vcit:student
```

5. 查看 vcit 命名空间的表

```
list_namespace_tables 'vcit'
```

create_namespace 'vcit'

```
hbase(main):008:0>  list_namespace_tables 'vcit'
TABLE
student
1 row(s)
Took 0.0290 seconds
=> ["student"]
```

6. 判断 vcit:student 表是否存在

```
exists 'vcit:student'
```

```
hbase(main):009:0> exists 'vcit:student'
Table vcit:student does exist
Took 0.1300 seconds
=> true
```

7. 查看表结构

```
describe 'vcit:student'
```

```
hbase(main):010:0> describe 'vcit:student'
Table vcit:student is ENABLED
vcit:student
COLUMN FAMILIES DESCRIPTION
{NAME => 'info', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'F
ALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOP
E => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREF
ETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'score', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => '
FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCO
PE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PRE
FETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

2 row(s)

QUOTAS
0 row(s)
Took 0.1526 seconds
```

8. 插入数据

```
put 'vcit:student', 'xiaobai', 'info:sid','20250001'
put 'vcit:student', 'xiaobai', 'info:class','202501'
```

```
put 'vcit:student', 'xiaobai', 'info:age','19'
put 'vcit:student', 'xiaobai', 'score:english','80'
put 'vcit:student', 'xiaobai', 'score:chainese','95'


put 'vcit:student', 'xiaohui', 'info:sid','20250002'
put 'vcit:student', 'xiaohui', 'info:class','202501'
put 'vcit:student', 'xiaohui', 'info:age','20'
put 'vcit:student', 'xiaohui', 'score:english','76'
put 'vcit:student', 'xiaohui', 'score:chainese','85'

put 'vcit:student', 'xiaohong', 'info:sid','20250003'
put 'vcit:student', 'xiaohong', 'info:class','202502'
put 'vcit:student', 'xiaohong', 'info:age','18'
put 'vcit:student', 'xiaohong', 'score:english','96'
put 'vcit:student', 'xiaohong', 'score:chainese','83'

put 'vcit:student', 'xiaolan', 'info:sid','20250004'
put 'vcit:student', 'xiaolan', 'info:class','202502'
put 'vcit:student', 'xiaolan', 'info:age','17'
put 'vcit:student', 'xiaolan', 'score:english','86'
put 'vcit:student', 'xiaolan', 'score:chainese','93'

put 'vcit:student', '小花', 'info:sid','20250005'
put 'vcit:student', '小花', 'info:class','202501'
put 'vcit:student', '小花', 'info:age','18'
put 'vcit:student', '小花', 'score:english','98'
put 'vcit:student', '小花', 'score:chainese','92'
```

9. 查询所有数据 vcit:student

```
scan 'vcit:student'
```

```
hbase(main):040:0> scan 'vcit:student',{FORMATTER => 'toString'}
ROW                                   COLUMN+CELL
 ??????                               column=info:age, timestamp=1741246698247, value=18
 ??????                               column=info:class, timestamp=1741246698223, value=202501
 ??????                               column=info:sid, timestamp=1741246698203, value=20250005
 ??????                               column=score:chainese, timestamp=1741246699301, value=92
 ??????                               column=score:english, timestamp=1741246698272, value=98
 xiaobai                              column=info:age, timestamp=1741246360217, value=19
 xiaobai                              column=info:class, timestamp=1741246360187, value=202501
 xiaobai                              column=info:sid, timestamp=1741246360140, value=20250001
 xiaobai                              column=score:chainese, timestamp=1741246361131, value=95
 xiaobai                              column=score:english, timestamp=1741246360251, value=80
 xiaohong                             column=info:age, timestamp=1741246467705, value=18
 xiaohong                             column=info:class, timestamp=1741246467670, value=202502
 xiaohong                             column=info:sid, timestamp=1741246467626, value=20250003
 xiaohong                             column=score:chainese, timestamp=1741246468176, value=83
 xiaohong                             column=score:english, timestamp=1741246467736, value=96
 xiaohui                              column=info:age, timestamp=1741246419027, value=20
 xiaohui                              column=info:class, timestamp=1741246418962, value=202501
 xiaohui                              column=info:sid, timestamp=1741246418921, value=20250002
 xiaohui                              column=score:chainese, timestamp=1741246419776, value=85
 xiaohui                              column=score:english, timestamp=1741246419056, value=76
 xiaolan                              column=info:age, timestamp=1741246515265, value=17
 xiaolan                              column=info:class, timestamp=1741246515239, value=202502
 xiaolan                              column=info:sid, timestamp=1741246515211, value=20250004
 xiaolan                              column=score:chainese, timestamp=1741246515646, value=93
 xiaolan                              column=score:english, timestamp=1741246515290, value=86
5 row(s)
```

10. 按照字典顺序过滤查询

```
scan 'vcit:student',{STARTROW => 'xiaobai', STOPROW => 'xiaohuj'}
```

```
hbase(main):043:0> scan 'vcit:student',{STARTROW => 'xiaobai', STOPROW => 'xiaohuj'}
ROW                                   COLUMN+CELL
 xiaobai                              column=info:age, timestamp=1741246360217, value=19
 xiaobai                              column=info:class, timestamp=1741246360187, value=202501
 xiaobai                              column=info:sid, timestamp=1741246360140, value=20250001
 xiaobai                              column=score:chainese, timestamp=1741246361131, value=95
 xiaobai                              column=score:english, timestamp=1741246360251, value=80
 xiaohong                             column=info:age, timestamp=1741246467705, value=18
 xiaohong                             column=info:class, timestamp=1741246467670, value=202502
 xiaohong                             column=info:sid, timestamp=1741246467626, value=20250003
 xiaohong                             column=score:chainese, timestamp=1741246468176, value=83
 xiaohong                             column=score:english, timestamp=1741246467736, value=96
 xiaohui                              column=info:age, timestamp=1741246419027, value=20
 xiaohui                              column=info:class, timestamp=1741246418962, value=202501
 xiaohui                              column=info:sid, timestamp=1741246418921, value=20250002
 xiaohui                              column=score:chainese, timestamp=1741246419776, value=85
 xiaohui                              column=score:english, timestamp=1741246419056, value=76
3 row(s)
Took 0.0201 seconds
```

11.获取单个rowkey的数据

```
get 'vcit:student','xiaohui'
```

```
hbase(main):044:0> get 'vcit:student','xiaohui'
COLUMN                          CELL
 info:age                       timestamp=1741246419027, value=20
 info:class                     timestamp=1741246418962, value=202501
 info:sid                       timestamp=1741246418921, value=20250002
 score:chainese                 timestamp=1741246419776, value=85
 score:english                  timestamp=1741246419056, value=76
1 row(s)
Took 0.0256 seconds
```

12. 获取单个rowkey中某个列族数据

```
get 'vcit:student','xiaohui','score'
```

```
hbase(main):045:0> get 'vcit:student','xiaohui','score'
COLUMN                          CELL
 score:chainese                 timestamp=1741246419776, value=85
 score:english                  timestamp=1741246419056, value=76
1 row(s)
Took 0.0152 seconds
```

13. 获取单个rowkey中某个列族里的列的数据

```
get 'vcit:student','xiaohui','score:english'
```

```
hbase(main):046:0> get 'vcit:student','xiaohui','score:english'
COLUMN                          CELL
 score:english                  timestamp=1741246419056, value=76
1 row(s)
Took 0.0327 seconds
```

14. 获取行数

```
count 'vcit:student'
```

```
hbase(main):047:0> count 'vcit:student'
5 row(s)
Took 0.0611 seconds
=> 5
```

15. 删除某个rowkey的数据

```
deleteall 'vcit:student', '小花'
```

```
hbase(main):050:0> scan 'vcit:student'
ROW                                   COLUMN+CELL
 xiaobai                              column=info:age, timestamp=1741246360217, value=19
 xiaobai                              column=info:class, timestamp=1741246360187, value=202501
 xiaobai                              column=info:sid, timestamp=1741246360140, value=20250001
 xiaobai                              column=score:chainese, timestamp=1741246361131, value=95
 xiaobai                              column=score:english, timestamp=1741246360251, value=80
 xiaohong                             column=info:age, timestamp=1741246467705, value=18
 xiaohong                             column=info:class, timestamp=1741246467670, value=202502
 xiaohong                             column=info:sid, timestamp=1741246467626, value=20250003
 xiaohong                             column=score:chainese, timestamp=1741246468176, value=83
 xiaohong                             column=score:english, timestamp=1741246467736, value=96
 xiaohui                              column=info:age, timestamp=1741246419027, value=20
 xiaohui                              column=info:class, timestamp=1741246418962, value=202501
 xiaohui                              column=info:sid, timestamp=1741246418921, value=20250002
 xiaohui                              column=score:chainese, timestamp=1741246419776, value=85
 xiaohui                              column=score:english, timestamp=1741246419056, value=76
 xiaolan                              column=info:age, timestamp=1741246515265, value=17
 xiaolan                              column=info:class, timestamp=1741246515239, value=202502
 xiaolan                              column=info:sid, timestamp=1741246515211, value=20250004
 xiaolan                              column=score:chainese, timestamp=1741246515646, value=93
 xiaolan                              column=score:english, timestamp=1741246515290, value=86
4 row(s)
Took 0.0202 seconds
```

16. 删除某个rowkey中某个字段数据

```
delete 'vcit:student','xiaobai','score:english'
-- 删除之后，再次查询
get 'vcit:student','xiaobai'
```

```
hbase(main):054:0> get 'vcit:student','xiaobai'
COLUMN                                CELL
 info:age                             timestamp=1741246360217, value=19
 info:class                           timestamp=1741246360187, value=202501
 info:sid                             timestamp=1741246360140, value=20250001
 score:chainese                       timestamp=1741246361131, value=95
1 row(s)
Took 0.0137 seconds
```

17. 在已有的表中添加列族

```
alter 'vcit:student','test'
-- 添加列族之后，查询表结构
describe 'vcit:student'
```

```
hbase(main):065:0> describe 'vcit:student'
Table vcit:student is ENABLED
vcit:student
COLUMN FAMILIES DESCRIPTION
{NAME => 'info', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'F
ALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOP
E => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREF
ETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'score', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => '
FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCO
PE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PRE
FETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'test', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => 'F
ALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOP
E => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREF
ETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

3 row(s)

QUOTAS
0 row(s)
Took 0.0578 seconds
```

18. 在已有的表中删除列族

```
alter 'vcit:student',{NAME => 'test', METHOD => 'delete'}
```

19. 清空表，（删除表中所有的数据，但是保留表结构）

```
truncate 'vcit:student'
```

20. 删除表

```
disable 'vcit:student'
drop 'vcit:student'
```

21 . namespace的删除

```
drop_namespace 'vcit'
```

JavaAPI

HBase过滤器

HBase中提供了丰富的过滤器（filter），提高数据处理的效率。用户通过内置的或自定义的过滤器来对数据进行顾虑。所有的过滤器都在服务端生效，保证过滤掉的数据不会被传送到客户端。

过滤器的实现

Filter抽象类 和 FilterBase抽象类

Filter抽象类定义了过滤器中的基本方法， FilterBase抽象类继承了Filter抽象类，并且对方法进行的扩充。

因此，**所有的内置的过滤器都是直接或间接的继承于 FilerBase 抽象类**

用户只需要将定义好的过滤器对象 通过 setFilter 方法传递给 Scan 对象

比较过滤器

所有比较过滤器都是继承于 CompareFilter ， 创建一个比较过滤器需要有两个参数，分别**比较运算符**和**比较器对象**

比较运算符

比较运算符都是来自于 CompareFilter 中枚举 CompareOp

```
LESS  (<)
LESS_OR_EQUAL  (<=)
EQUAL (=)
NOT_EQUAL  (!=)
GREATER_OR_EQUAL  (>=)
GREATER  (>)
NO_OP (排除所有符合条件的值)
```

比较器

所有的比较器类都继承于 ByteArrayComparable 抽象类

常用的比较器

```
BinaryComparator  按照字典顺序比较指定的字节数据 （完全匹配）
```

```
SubstringComparator  给定的子字符串是否出现在目标字符串中 （包含匹配），使用的比较符有 EQUAL 和 NOT_EQUAL
```

```
NullComparator  判断给定的值是否为空
```

```
BitComparator 按位来进行比较
```

```
RegexStringComparator 使用给定的正则表达式和指定的字节数组进行比较。
```

```
BinaryPrefixComparator 按字典顺序和指定的字节数组进行比较，但只比较到这个字节数组的长度 （以比较值开头匹配）
```

比较过滤器的种类

```
ValueFilter 基于单元格（cell）的值来过滤数据
```

```
RowFilter  基于行键（rowkey）来过滤数据
```

```
FamilyFilter  基于 列族 来过滤数据
```

QualifierFilter 基于 列 来过滤数据

DependentColumnFilter 基于一个参考列来过滤其他列的过滤器，原则是基于参考列的时间戳来进行筛选

```
hbase(main):006:0> scan 'student', {FILTER => "FamilyFilter(=, 'substring:nfo')"}
ROW                                COLUMN+CELL

 xiaobai                           column=info:age, timestamp=1741708627869, value=20

 xiaobai                           column=info:sex, timestamp=1741708619735, value=male

 xiaobai                           column=info:sid, timestamp=1741708606928, value=20250011

 xiaohong                          column=info:age, timestamp=1741698040592, value=20

 xiaohong                          column=info:sex, timestamp=1741698040492, value=female

 xiaohong                          column=info:sid, timestamp=1741698040427, value=20250012

 xiaolan                           column=info:age, timestamp=1741738774841, value=18

 xiaolan                           column=info:sex, timestamp=1741738790220, value=male

 xiaolan                           column=info:sid, timestamp=1741738807778, value=20250014

3 row(s)
Took 0.0327 seconds

hbase(main):007:0> scan 'student', {FILTER => "QualifierFilter(=, 'substring:ex')"}
ROW                                COLUMN+CELL

 xiaobai                           column=info:sex, timestamp=1741708619735, value=male

 xiaohong                          column=info:sex, timestamp=1741698040492, value=female

 xiaolan                           column=info:sex, timestamp=1741738790220, value=male

3 row(s)
Took 0.0370 seconds
```