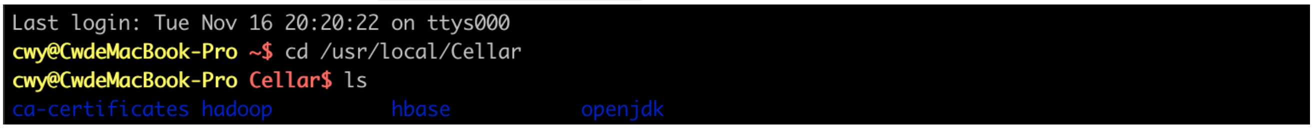
实验三：HBase 安装、操作和编程

191820231　严楚雯

一、安装并运行 HBase

安装HBase

下载HBase 2.3.7，将其解压到`/usr/local/Cellar`目录下

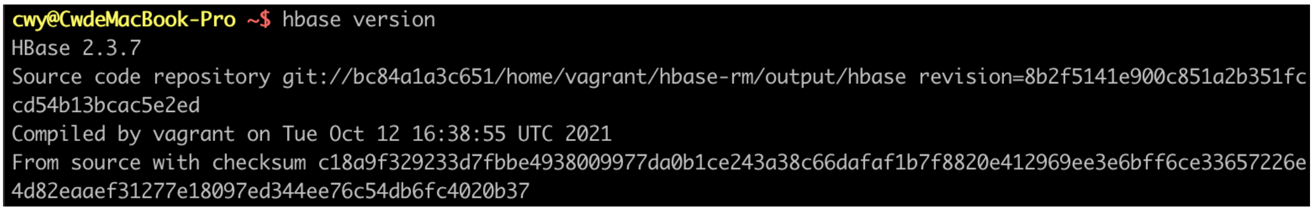


```
Last login: Tue Nov 16 20:20:22 on ttys000
cwy@CwdeMacBook-Pro ~$ cd /usr/local/Cellar
cwy@CwdeMacBook-Pro Cellar$ ls
ca-certificates hadoop          hbase           openjdk
```

配置环境变量`/Users/cwy/.bash_profile`，添加以下内容

```
export HBASE_HOME=/usr/local/Cellar/hbase
export HBASE_CONF_DIR=$HBASE_HOME/conf
export PATH=$PATH:$HBASE_HOME/bin
```

执行`hbase version`查看版本，检测是否安装成功

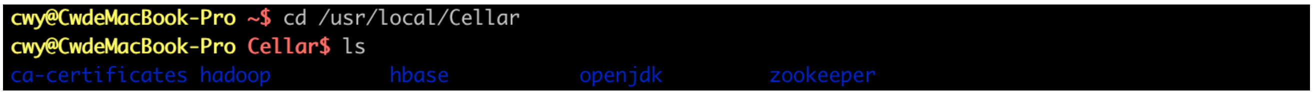


```
cwy@CwdeMacBook-Pro ~$ hbase version
HBase 2.3.7
Source code repository git://bc84a1a3c651/home/vagrant/hbase-rm/output/hbase revision=8b2f5141e900c851a2b351fc
cd54b13bcac5e2ed
Compiled by vagrant on Tue Oct 12 16:38:55 UTC 2021
From source with checksum c18a9f329233d7fbbe4938009977da0b1ce243a38c66dafaf1b7f8820e412969ee3e6bff6ce33657226e
4d82eaaef31277e18097ed344ee76c54db6fc4020b37
```

安装并配置zookeeper

因为使用Hbase自带的zookeeper不太稳定，所以选择安装外置的zookeeper。

下载zookeeper 3.5.7，将其解压到`/usr/local/Cellar`目录下



```
cwy@CwdeMacBook-Pro ~$ cd /usr/local/Cellar
cwy@CwdeMacBook-Pro Cellar$ ls
ca-certificates hadoop          hbase           openjdk         zookeeper
```

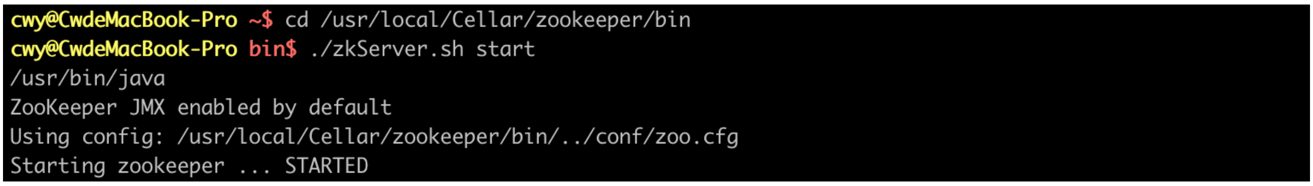
修改`zoo.cfg`文件里面内容

```
dataDir=/usr/local/Cellar/zookeeper/data
dataLogDir=/usr/local/Cellar/zookeeper/logs
clientPort=2181
```

zookeeper常用命令

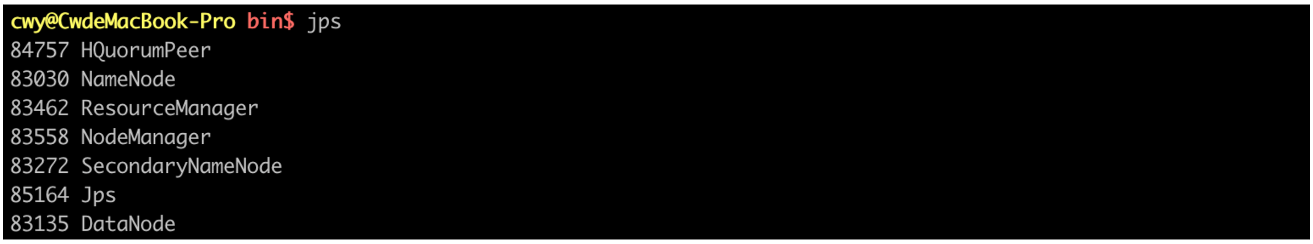
```
$ ./zkServer.sh start   #启动命令
$ ./zkServer.sh status  #查看状态
$ ./zkServer.sh stop    #停止命令
```

在`/usr/local/Cellar/zookeeper/bin`目录下，执行`./zkServer.sh start`启动zookeeper服务端



```
cwy@CwdeMacBook-Pro ~$ cd /usr/local/Cellar/zookeeper/bin
cwy@CwdeMacBook-Pro bin$ ./zkServer.sh start
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /usr/local/Cellar/zookeeper/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
```

`jps`查看是否启动成功

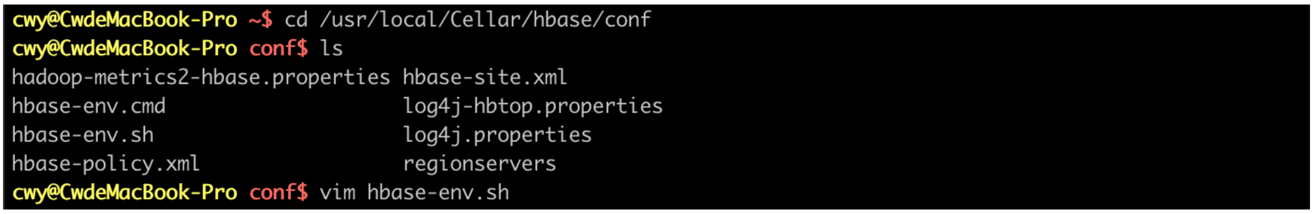


```
cwy@CwdeMacBook-Pro bin$ jps
84757 HQuorumPeer
83030 NameNode
83462 ResourceManager
83558 NodeManager
83272 SecondaryNameNode
85164 Jps
83135 DataNode
```

伪分布式运行

在`/usr/local/Cellar/hbase/conf/hbase-env.sh`设置 JAVA_HOME ，修改`HBASE_MANAGES_ZK`参数为 false，表示使用外置zookeeper。

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_271.jdk/Contents/Home
export HBASE_MANAGES_ZK=false
```



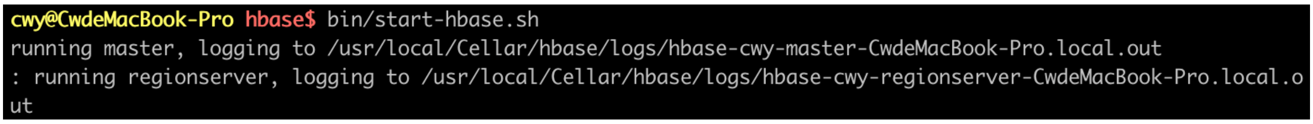
```
cwy@CwdeMacBook-Pro ~$ cd /usr/local/Cellar/hbase/conf
cwy@CwdeMacBook-Pro conf$ ls
hadoop-metrics2-hbase.properties hbase-site.xml
hbase-env.cmd                    log4j-hbtop.properties
hbase-env.sh                     log4j.properties
hbase-policy.xml                 regionservers
cwy@CwdeMacBook-Pro conf$ vim hbase-env.sh
```

配置`/usr/local/Cellar/hbase/conf/hbase-site.xml`

```
<configuration>
  <property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
  </property>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://localhost:9000/hbase</value>
  </property>
  <property>
    <name>hbase.tmp.dir</name>
    <value>/usr/local/Cellar/hbase/hbase-tmp</value>
  </property>
</configuration>
```

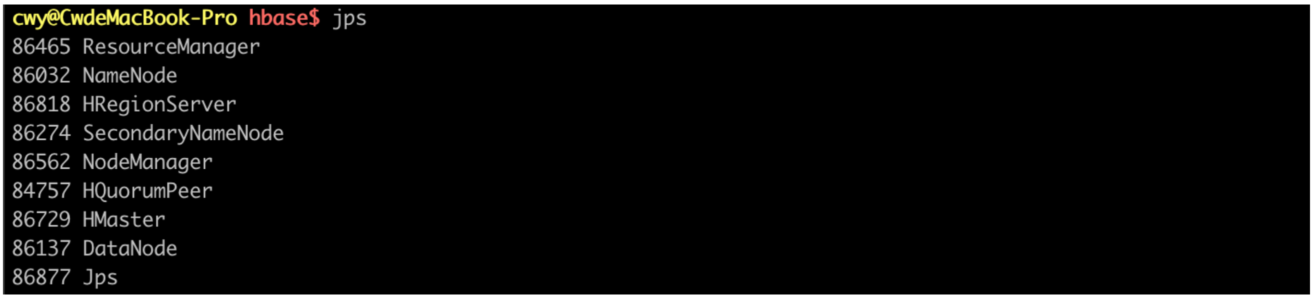
首先启动 Hadoop和zookeeper，然后在 `/usr/local/Cellar/hbase`目录下执行

```
$ bin/start-hbase.sh
```



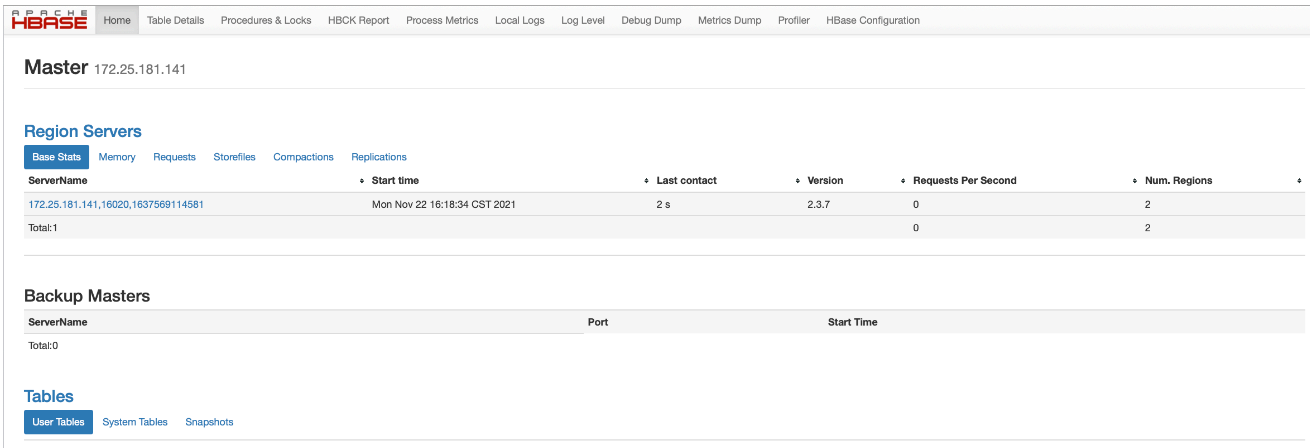
```
cwy@CwdeMacBook-Pro hbase$ bin/start-hbase.sh
running master, logging to /usr/local/Cellar/hbase/logs/hbase-cwy-master-CwdeMacBook-Pro.local.out
: running regionserver, logging to /usr/local/Cellar/hbase/logs/hbase-cwy-regionserver-CwdeMacBook-Pro.local.o
ut
```

启动 Hbase，执行`jps`命令查看各端口状态，检验HBase是否启动成功



```
cwy@CwdeMacBook-Pro hbase$ jps
86465 ResourceManager
86032 NameNode
86818 HRegionServer
86274 SecondaryNameNode
86562 NodeManager
84757 HQuorumPeer
86729 HMaster
86137 DataNode
86877 Jps
```

浏览**HBase的Web接口**：http://localhost:16010



Master 172.25.181.141

Region Servers

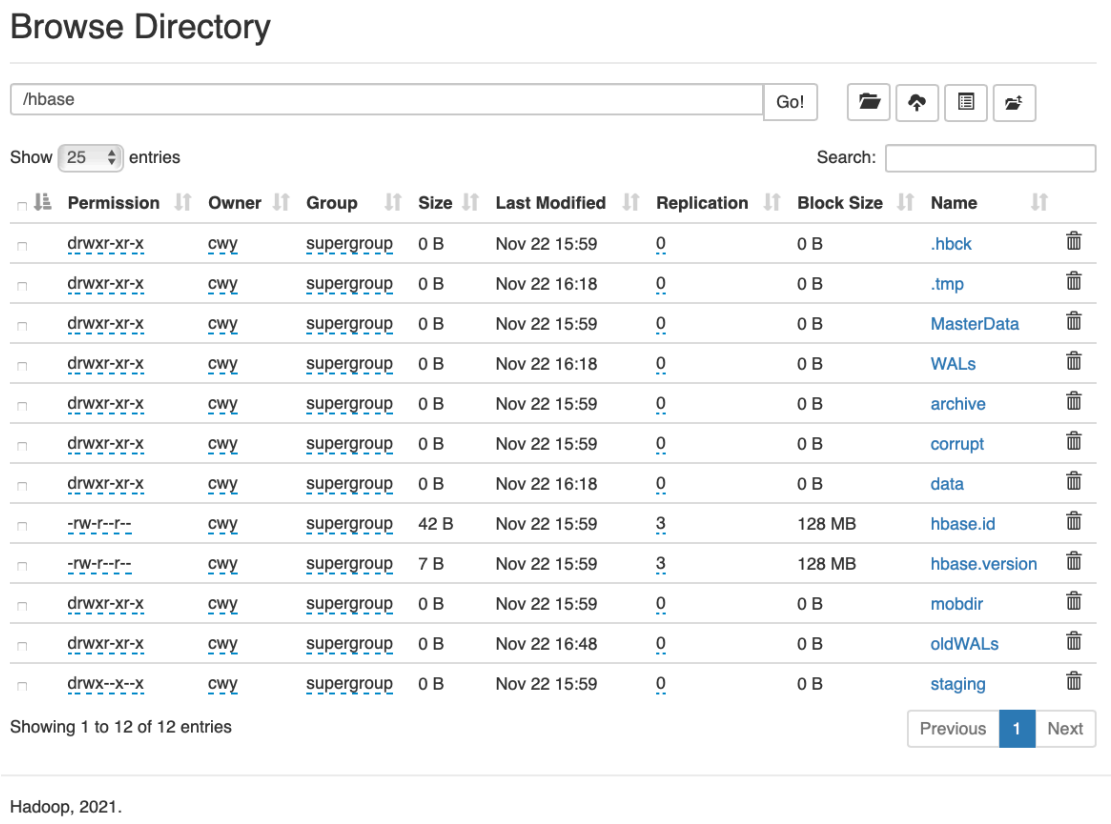
ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
172.25.181.141,16020,1637569114581	Mon Nov 22 16:18:34 CST 2021	2 s	2.3.7	0	2
Total:1				0	2

Backup Masters

ServerName	Port	Start Time
Total:0		

Tables

启动成功HBase会在HDFS下创建`/hbase`目录，浏览 **NameNode 的Web接口**：http://localhost:9870，检查`/hbase`目录是否存在。



Browse Directory

□	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 15:59	0	0 B	.hbck	🗑
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 16:18	0	0 B	.tmp	🗑
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 15:59	0	0 B	MasterData	🗑
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 16:18	0	0 B	WALs	🗑
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 15:59	0	0 B	archive	🗑
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 15:59	0	0 B	corrupt	🗑
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 16:18	0	0 B	data	🗑
□	-rw-r--r--	cwy	supergroup	42 B	Nov 22 15:59	3	128 MB	hbase.id	🗑
□	-rw-r--r--	cwy	supergroup	7 B	Nov 22 15:59	3	128 MB	hbase.version	🗑
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 15:59	0	0 B	mobdir	🗑
□	drwxr-xr-x	cwy	supergroup	0 B	Nov 22 16:48	0	0 B	oldWALs	🗑
□	drwx--x--x	cwy	supergroup	0 B	Nov 22 15:59	0	0 B	staging	🗑

Showing 1 to 12 of 12 entries

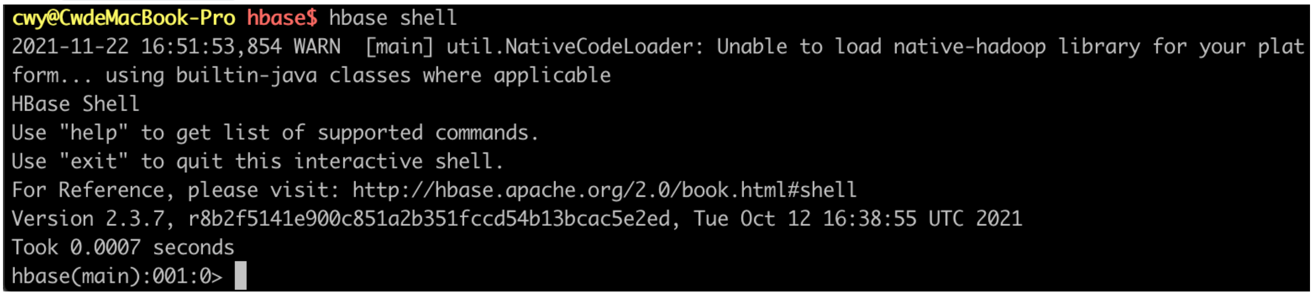
Hadoop, 2021.

二、HBase 操作及编程

分别用Hbase shell和 java编程实现HBase数据库中的一系列操作。

Hbase shell

执行`hbase shell`命令，进入Hbase shell



```
cwy@CwdeMacBook-Pro hbase$ hbase shell
2021-11-22 16:51:53,854 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-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 2.3.7, r8b2f5141e900c851a2b351fccd54b13bcac5e2ed, Tue Oct 12 16:38:55 UTC 2021
Took 0.0007 seconds
hbase(main):001:0> 
```

1. 设计并创建合适的表

根据要求，设计合适的表，将'student'、'course'、'sc'三表给出的信息整合到一张表中（如下图），将新表命名为'studentInfo'。

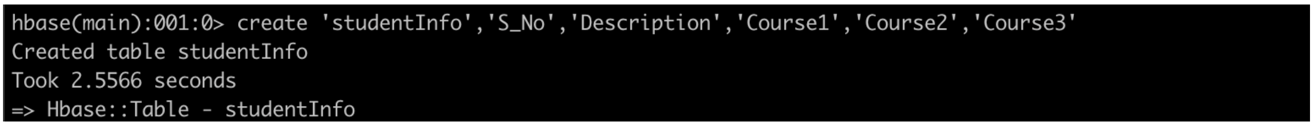
studentInfo 设计

S_No	Description		Course1		Course2		Course3	
2015001	S_Name S_Sex S_Age	Li Lei Male 23	C_No C_Name C_Credit C_Score	123001 Math 2.0 86			C_No C_Name C_Credit C_Score	123003 English 3.0 69
2015002	S_Name S_Sex S_Age	Han Meimei Female 22			C_No C_Name C_Credit C_Score	123002 Computer Science 5.0 77	C_No C_Name C_Credit C_Score	123003 English 3.0 99
2015003	S_Name S_Sex S_Age	Zhang San Male 24	C_No C_Name C_Credit C_Score	123001 Math 2.0 98	C_No C_Name C_Credit C_Score	123002 Computer Science 5.0 95		

详细设计如下：以学生的学号（S_No）为行键，设置Description, Course1, Course2, Course3这四个列族。其中，Description列族包含学生的姓名、性别、年龄信息（S_Name, S_Sex, S_Age），Course1, Course2, Course3则分别指代三门课程，包含课程编号、课程名、学分数和得分（C_No、C_Name、C_Credit、C_Score），若学生选择课程，则相应的列族中就会记录对应信息，否则为空。

创建表 'studentInfo'，并添加信息

```
create 'studentInfo','S_No','Description','Course1','Course2','Course3'
```

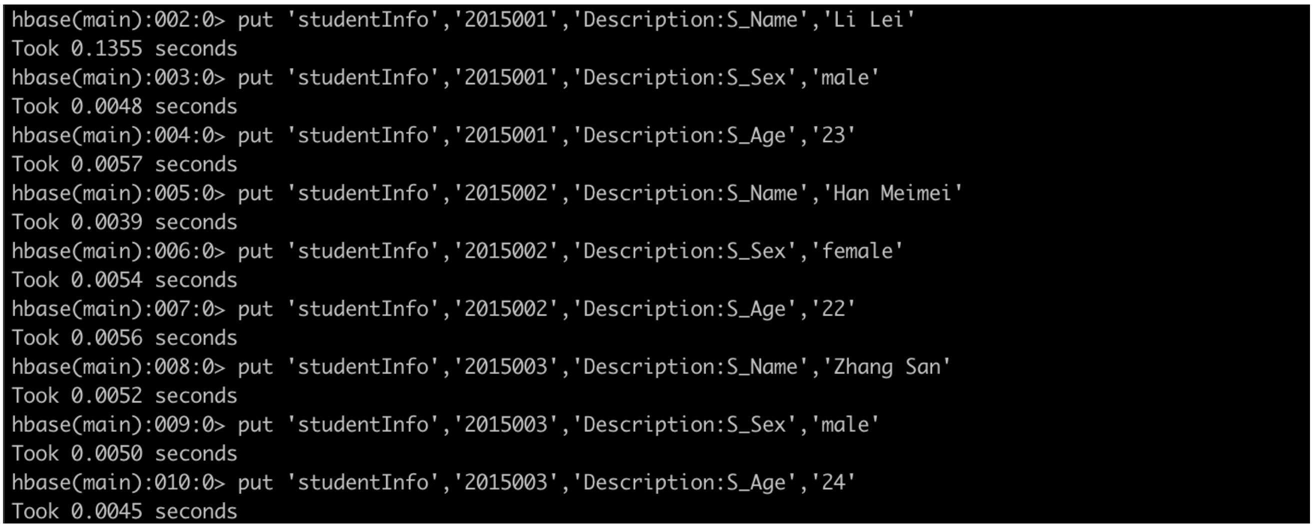


```
hbase(main):001:0> create 'studentInfo','S_No','Description','Course1','Course2','Course3'
Created table studentInfo
Took 2.5566 seconds
=> Hbase::Table - studentInfo
```

添加学生信息

```
put 'studentInfo','2015001','Description:S_Name','Li Lei'
put 'studentInfo','2015001','Description:S_Sex','male'
put 'studentInfo','2015001','Description:S_Age','23'
put 'studentInfo','2015002','Description:S_Name','Han Meimei'
put 'studentInfo','2015002','Description:S_Sex','female'
put 'studentInfo','2015002','Description:S_Age','22'
```

```
put 'studentInfo','2015003','Description:S_Name','Zhang San'
put 'studentInfo','2015003','Description:S_Sex','male'
put 'studentInfo','2015003','Description:S_Age','24'
```



```
hbase(main):002:0> put 'studentInfo','2015001','Description:S_Name','Li Lei'
Took 0.1355 seconds
hbase(main):003:0> put 'studentInfo','2015001','Description:S_Sex','male'
Took 0.0048 seconds
hbase(main):004:0> put 'studentInfo','2015001','Description:S_Age','23'
Took 0.0057 seconds
hbase(main):005:0> put 'studentInfo','2015002','Description:S_Name','Han Meimei'
Took 0.0039 seconds
hbase(main):006:0> put 'studentInfo','2015002','Description:S_Sex','female'
Took 0.0054 seconds
hbase(main):007:0> put 'studentInfo','2015002','Description:S_Age','22'
Took 0.0056 seconds
hbase(main):008:0> put 'studentInfo','2015003','Description:S_Name','Zhang San'
Took 0.0052 seconds
hbase(main):009:0> put 'studentInfo','2015003','Description:S_Sex','male'
Took 0.0050 seconds
hbase(main):010:0> put 'studentInfo','2015003','Description:S_Age','24'
Took 0.0045 seconds
```

按列族添加学生课程信息，Course1

```
put 'studentInfo','2015001','Course1:C_No','123001'
put 'studentInfo','2015001','Course1:C_Name','Math'
put 'studentInfo','2015001','Course1:C_Credit','2.0'
put 'studentInfo','2015001','Course1:C_Score','86'
```

```
put 'studentInfo','2015003','Course1:C_No','123001'
put 'studentInfo','2015003','Course1:C_Name','Math'
put 'studentInfo','2015003','Course1:C_Credit','2.0'
put 'studentInfo','2015003','Course1:C_Score','98'
```

Course2

```
put 'studentInfo','2015002','Course2:C_No','123002'
put 'studentInfo','2015002','Course2:C_Name','Computer Science'
put 'studentInfo','2015002','Course2:C_Credit','5.0'
put 'studentInfo','2015002','Course2:C_Score','77'
```

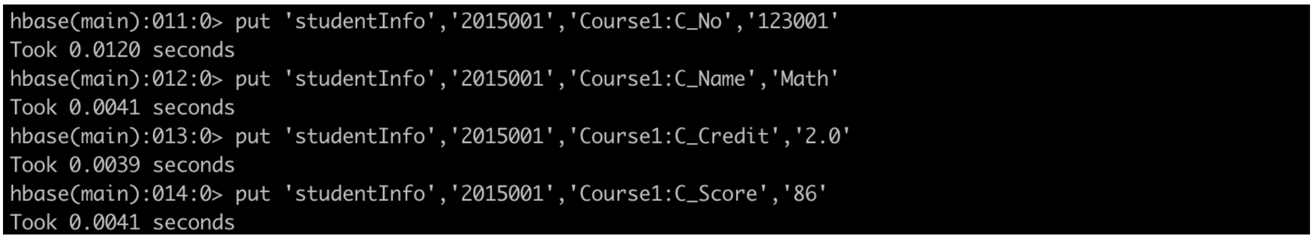
```
put 'studentInfo','2015003','Course2:C_No','123002'
put 'studentInfo','2015003','Course2:C_Name','Computer Science'
put 'studentInfo','2015003','Course2:C_Credit','5.0'
put 'studentInfo','2015003','Course2:C_Score','95'
```

Course3

```
put 'studentInfo','2015001','Course3:C_No','123003'
put 'studentInfo','2015001','Course3:C_Name','English'
put 'studentInfo','2015001','Course3:C_Credit','3.0'
put 'studentInfo','2015001','Course3:C_Score','69'
```

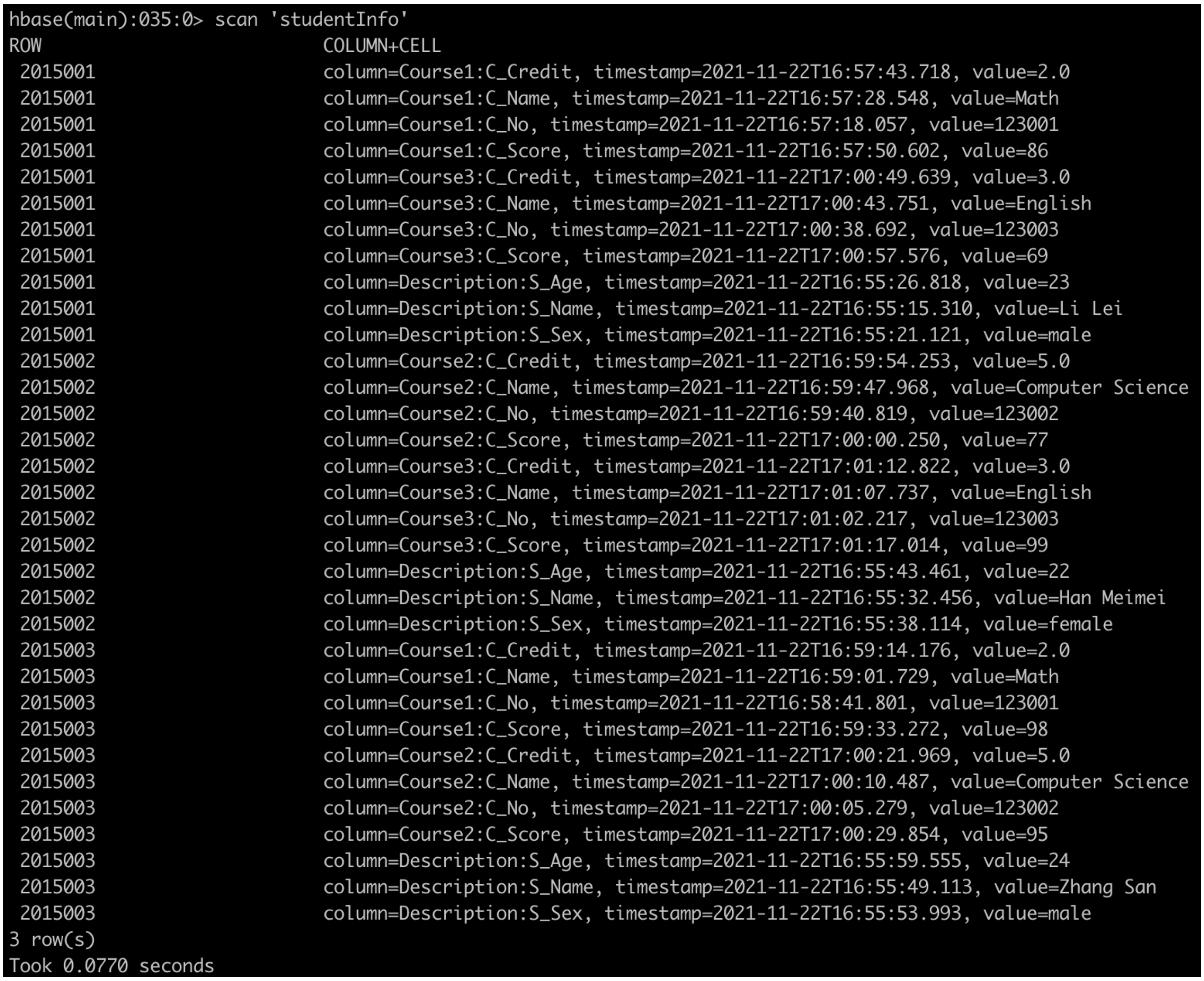
```
put 'studentInfo','2015002','Course3:C_No','123003'
put 'studentInfo','2015002','Course3:C_Name','English'
put 'studentInfo','2015002','Course3:C_Credit','3.0'
put 'studentInfo','2015002','Course3:C_Score','99'
```

局部截图



```
hbase(main):011:0> put 'studentInfo','2015001','Course1:C_No','123001'
Took 0.0120 seconds
hbase(main):012:0> put 'studentInfo','2015001','Course1:C_Name','Math'
Took 0.0041 seconds
hbase(main):013:0> put 'studentInfo','2015001','Course1:C_Credit','2.0'
Took 0.0039 seconds
hbase(main):014:0> put 'studentInfo','2015001','Course1:C_Score','86'
Took 0.0041 seconds
```

扫描全表，确认信息正确

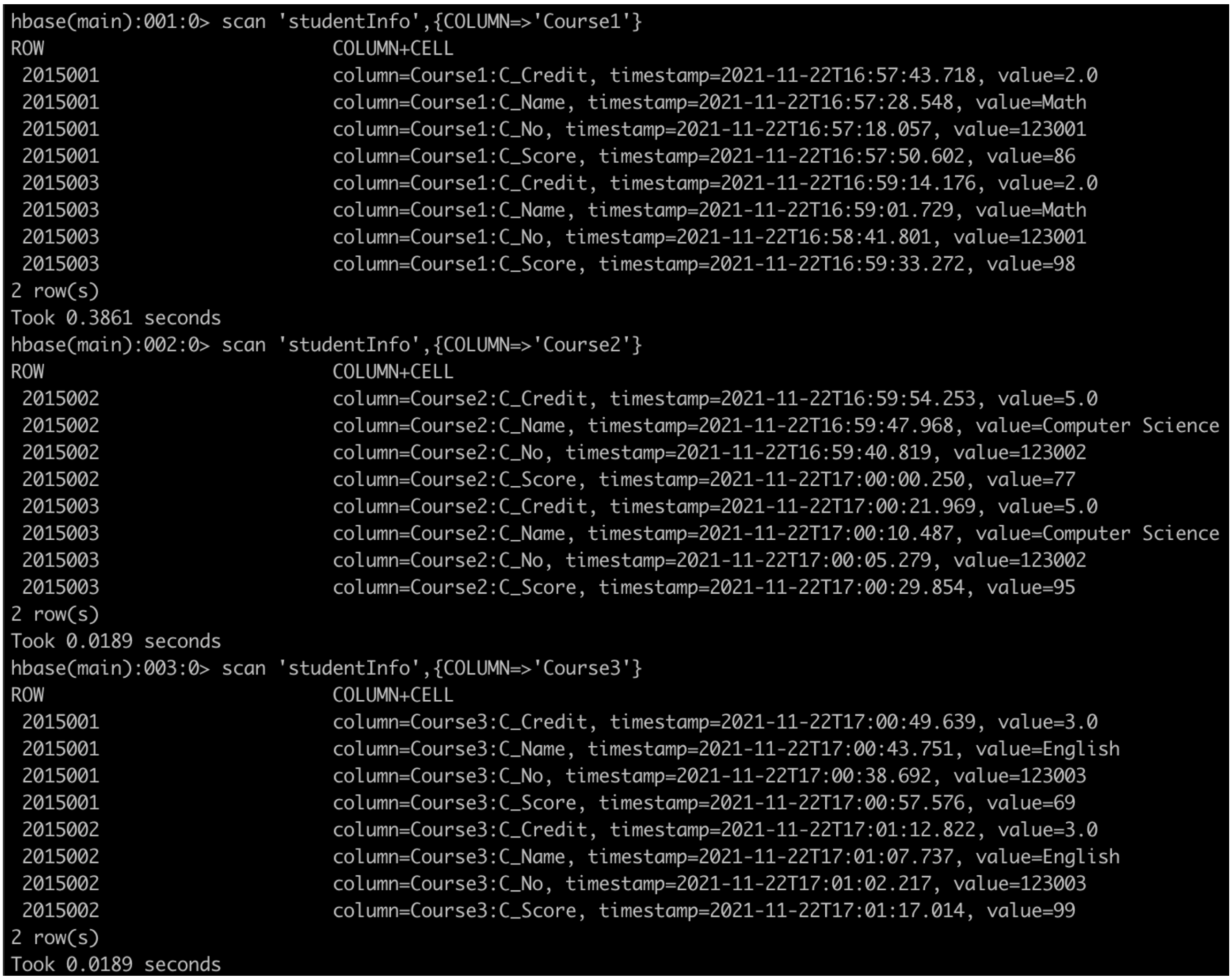


```
hbase(main):035:0> scan 'studentInfo'
ROW                            COLUMN+CELL
 2015001                       column=Course1:C_Credit, timestamp=2021-11-22T16:57:43.718, value=2.0
 2015001                       column=Course1:C_Name, timestamp=2021-11-22T16:57:28.548, value=Math
 2015001                       column=Course1:C_No, timestamp=2021-11-22T16:57:18.057, value=123001
 2015001                       column=Course1:C_Score, timestamp=2021-11-22T16:57:50.602, value=86
 2015001                       column=Course3:C_Credit, timestamp=2021-11-22T17:00:49.639, value=3.0
 2015001                       column=Course3:C_Name, timestamp=2021-11-22T17:00:43.751, value=English
 2015001                       column=Course3:C_No, timestamp=2021-11-22T17:00:38.692, value=123003
 2015001                       column=Course3:C_Score, timestamp=2021-11-22T17:00:57.576, value=69
 2015001                       column=Description:S_Age, timestamp=2021-11-22T16:55:26.818, value=23
 2015001                       column=Description:S_Name, timestamp=2021-11-22T16:55:15.310, value=Li Lei
 2015001                       column=Description:S_Sex, timestamp=2021-11-22T16:55:21.121, value=male
 2015002                       column=Course2:C_Credit, timestamp=2021-11-22T16:59:54.253, value=5.0
 2015002                       column=Course2:C_Name, timestamp=2021-11-22T16:59:47.968, value=Computer Science
 2015002                       column=Course2:C_No, timestamp=2021-11-22T16:59:40.819, value=123002
 2015002                       column=Course2:C_Score, timestamp=2021-11-22T17:00:00.250, value=77
 2015002                       column=Course3:C_Credit, timestamp=2021-11-22T17:01:12.822, value=3.0
 2015002                       column=Course3:C_Name, timestamp=2021-11-22T17:01:07.737, value=English
 2015002                       column=Course3:C_No, timestamp=2021-11-22T17:01:02.217, value=123003
 2015002                       column=Course3:C_Score, timestamp=2021-11-22T17:01:17.014, value=99
 2015002                       column=Description:S_Age, timestamp=2021-11-22T16:55:43.461, value=22
 2015002                       column=Description:S_Name, timestamp=2021-11-22T16:55:32.456, value=Han Meimei
 2015002                       column=Description:S_Sex, timestamp=2021-11-22T16:55:38.114, value=female
 2015003                       column=Course1:C_Credit, timestamp=2021-11-22T16:59:14.176, value=2.0
 2015003                       column=Course1:C_Name, timestamp=2021-11-22T16:59:01.729, value=Math
 2015003                       column=Course1:C_No, timestamp=2021-11-22T16:58:41.801, value=123001
 2015003                       column=Course1:C_Score, timestamp=2021-11-22T16:59:33.272, value=98
 2015003                       column=Course2:C_Credit, timestamp=2021-11-22T17:00:21.969, value=5.0
 2015003                       column=Course2:C_Name, timestamp=2021-11-22T17:00:10.487, value=Computer Science
 2015003                       column=Course2:C_No, timestamp=2021-11-22T17:00:05.279, value=123002
 2015003                       column=Course2:C_Score, timestamp=2021-11-22T17:00:29.854, value=95
 2015003                       column=Description:S_Age, timestamp=2021-11-22T16:55:59.555, value=24
 2015003                       column=Description:S_Name, timestamp=2021-11-22T16:55:49.113, value=Zhang San
 2015003                       column=Description:S_Sex, timestamp=2021-11-22T16:55:53.993, value=male
3 row(s)
Took 0.0770 seconds
```

2. 查询选修Computer Science的学生的成绩

首先用scan查找Course1、Course2、Course3分别指代的课程

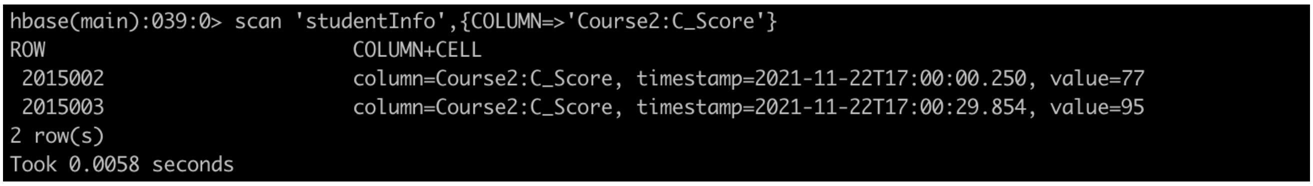
```
scan 'studentInfo',{COLUMN=>'Course1'}
scan 'studentInfo',{COLUMN=>'Course2'}
scan 'studentInfo',{COLUMN=>'Course3'}
```



```
hbase(main):001:0> scan 'studentInfo',{COLUMN=>'Course1'}
ROW                           COLUMN+CELL
 2015001                      column=Course1:C_Credit, timestamp=2021-11-22T16:57:43.718, value=2.0
 2015001                      column=Course1:C_Name, timestamp=2021-11-22T16:57:28.548, value=Math
 2015001                      column=Course1:C_No, timestamp=2021-11-22T16:57:18.057, value=123001
 2015001                      column=Course1:C_Score, timestamp=2021-11-22T16:57:50.602, value=86
 2015003                      column=Course1:C_Credit, timestamp=2021-11-22T16:59:14.176, value=2.0
 2015003                      column=Course1:C_Name, timestamp=2021-11-22T16:59:01.729, value=Math
 2015003                      column=Course1:C_No, timestamp=2021-11-22T16:58:41.801, value=123001
 2015003                      column=Course1:C_Score, timestamp=2021-11-22T16:59:33.272, value=98
2 row(s)
Took 0.3861 seconds
hbase(main):002:0> scan 'studentInfo',{COLUMN=>'Course2'}
ROW                           COLUMN+CELL
 2015002                      column=Course2:C_Credit, timestamp=2021-11-22T16:59:54.253, value=5.0
 2015002                      column=Course2:C_Name, timestamp=2021-11-22T16:59:47.968, value=Computer Science
 2015002                      column=Course2:C_No, timestamp=2021-11-22T16:59:40.819, value=123002
 2015002                      column=Course2:C_Score, timestamp=2021-11-22T17:00:00.250, value=77
 2015003                      column=Course2:C_Credit, timestamp=2021-11-22T17:00:21.969, value=5.0
 2015003                      column=Course2:C_Name, timestamp=2021-11-22T17:00:10.487, value=Computer Science
 2015003                      column=Course2:C_No, timestamp=2021-11-22T17:00:05.279, value=123002
 2015003                      column=Course2:C_Score, timestamp=2021-11-22T17:00:29.854, value=95
2 row(s)
Took 0.0189 seconds
hbase(main):003:0> scan 'studentInfo',{COLUMN=>'Course3'}
ROW                           COLUMN+CELL
 2015001                      column=Course3:C_Credit, timestamp=2021-11-22T17:00:49.639, value=3.0
 2015001                      column=Course3:C_Name, timestamp=2021-11-22T17:00:43.751, value=English
 2015001                      column=Course3:C_No, timestamp=2021-11-22T17:00:38.692, value=123003
 2015001                      column=Course3:C_Score, timestamp=2021-11-22T17:00:57.576, value=69
 2015002                      column=Course3:C_Credit, timestamp=2021-11-22T17:01:12.822, value=3.0
 2015002                      column=Course3:C_Name, timestamp=2021-11-22T17:01:07.737, value=English
 2015002                      column=Course3:C_No, timestamp=2021-11-22T17:01:02.217, value=123003
 2015002                      column=Course3:C_Score, timestamp=2021-11-22T17:01:17.014, value=99
2 row(s)
Took 0.0189 seconds
```

由查找结果可知，Computer Science课程对应的列族为 Course2，再次使用scan操作查找Course2课程对应的得分情况C_Score。

```
scan 'studentInfo',{COLUMN=>'Course2:C_Score'}
```



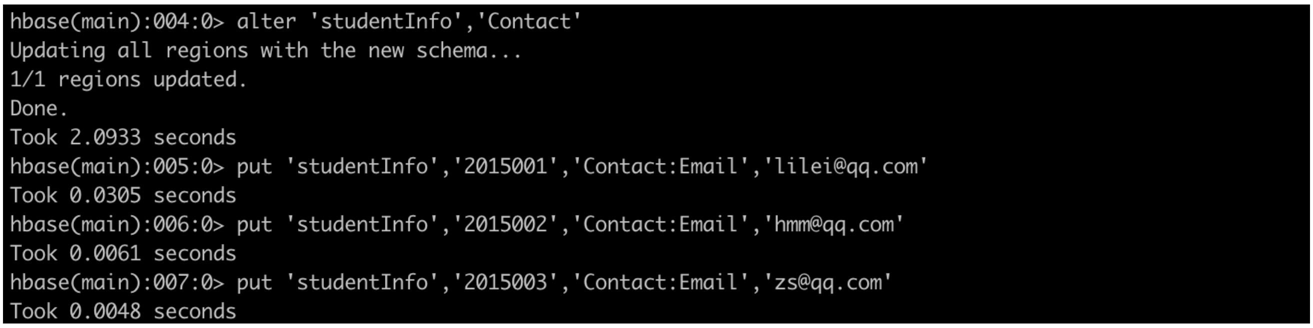
```
hbase(main):039:0> scan 'studentInfo',{COLUMN=>'Course2:C_Score'}
ROW                           COLUMN+CELL
 2015002                      column=Course2:C_Score, timestamp=2021-11-22T17:00:00.250, value=77
 2015003                      column=Course2:C_Score, timestamp=2021-11-22T17:00:29.854, value=95
2 row(s)
Took 0.0058 seconds
```

查询结果：学号为2015002的学生，Computer Science课程得分77，学号为2015003的学生，得分95。

3. 增加新的列族和新列Contact:Email，并添加数据

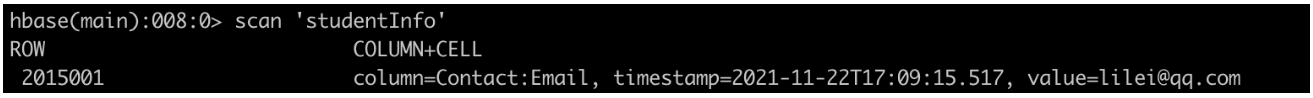
使用alter操作增加新的列族，使用put操作增加新列和数据

```
alter 'studentInfo','Contact'
put 'studentInfo','2015001','Contact:Email','lilei@qq.com'
put 'studentInfo','2015002','Contact:Email','hmm@qq.com'
put 'studentInfo','2015003','Contact:Email','zs@qq.com'
```



```
hbase(main):004:0> alter 'studentInfo','Contact'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.0933 seconds
hbase(main):005:0> put 'studentInfo','2015001','Contact:Email','lilei@qq.com'
Took 0.0305 seconds
hbase(main):006:0> put 'studentInfo','2015002','Contact:Email','hmm@qq.com'
Took 0.0061 seconds
hbase(main):007:0> put 'studentInfo','2015003','Contact:Email','zs@qq.com'
Took 0.0048 seconds
```

使用scan操作查看，添加成功



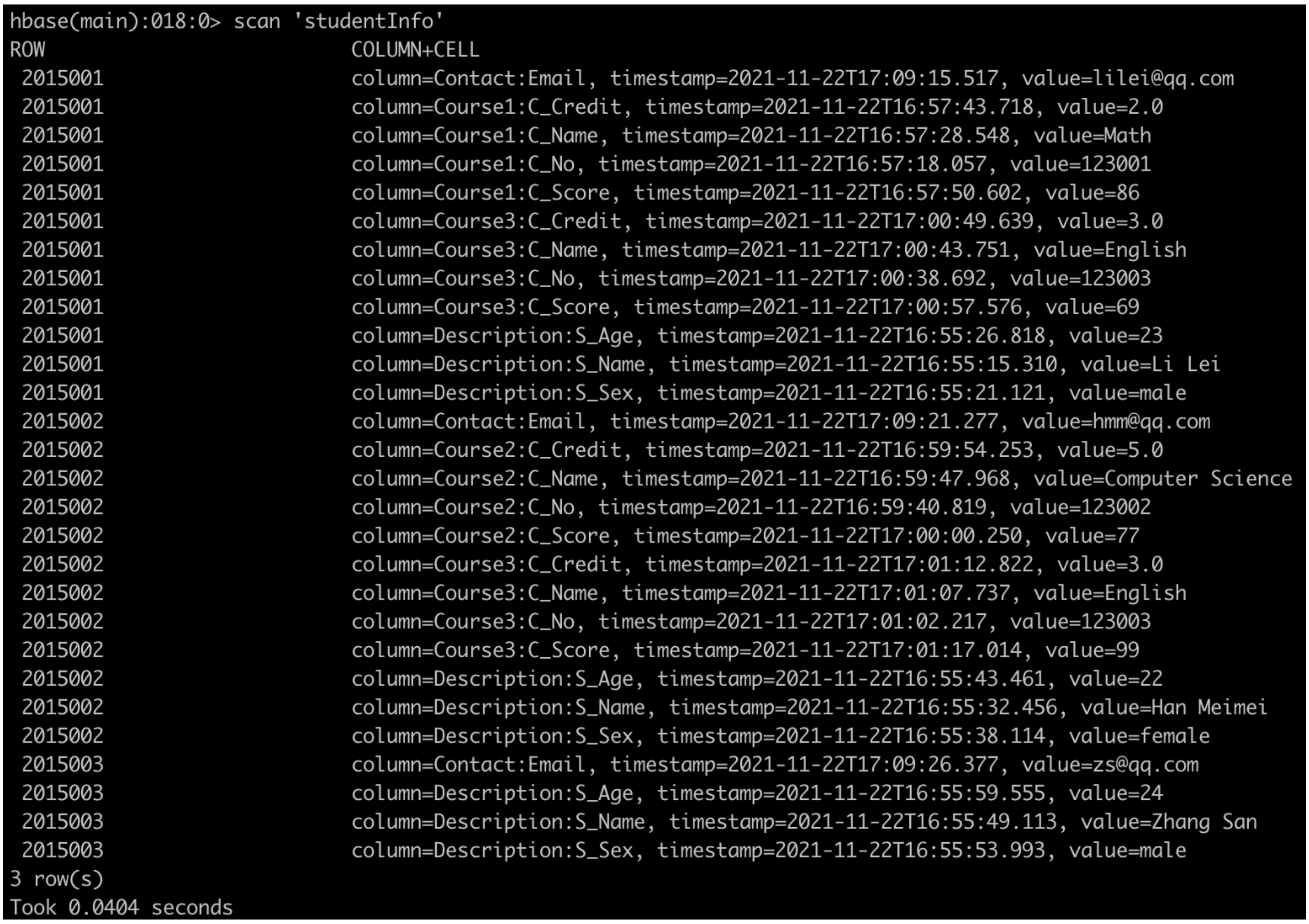
```
hbase(main):008:0> scan 'studentInfo'
ROW                            COLUMN+CELL
 2015001                       column=Contact:Email, timestamp=2021-11-22T17:09:15.517, value=lilei@qq.com
```

4. 删除学号为2015003的学生的选课记录

因为 Hbase shell 没有删除"指定行键的指定列族"的操作，只能删除一整行或者一个单元格，所以需要将选课记录对应的单元格信息逐个删除。

```
delete 'studentInfo','2015003','Course1:C_No'
delete 'studentInfo','2015003','Course1:C_Name'
delete 'studentInfo','2015003','Course1:C_Credit'
delete 'studentInfo','2015003','Course1:C_Score'
delete 'studentInfo','2015003','Course2:C_No'
delete 'studentInfo','2015003','Course2:C_Name'
delete 'studentInfo','2015003','Course2:C_Credit'
delete 'studentInfo','2015003','Course2:C_Score'
```

执行scan操作扫描全表验证，删除成功

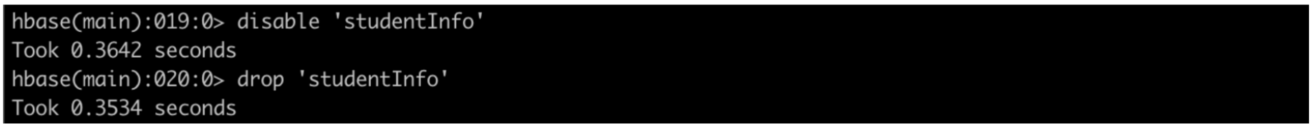


```
hbase(main):018:0> scan 'studentInfo'
ROW                          COLUMN+CELL
 2015001                     column=Contact:Email, timestamp=2021-11-22T17:09:15.517, value=lilei@qq.com
 2015001                     column=Course1:C_Credit, timestamp=2021-11-22T16:57:43.718, value=2.0
 2015001                     column=Course1:C_Name, timestamp=2021-11-22T16:57:28.548, value=Math
 2015001                     column=Course1:C_No, timestamp=2021-11-22T16:57:18.057, value=123001
 2015001                     column=Course1:C_Score, timestamp=2021-11-22T16:57:50.602, value=86
 2015001                     column=Course3:C_Credit, timestamp=2021-11-22T17:00:49.639, value=3.0
 2015001                     column=Course3:C_Name, timestamp=2021-11-22T17:00:43.751, value=English
 2015001                     column=Course3:C_No, timestamp=2021-11-22T17:00:38.692, value=123003
 2015001                     column=Course3:C_Score, timestamp=2021-11-22T17:00:57.576, value=69
 2015001                     column=Description:S_Age, timestamp=2021-11-22T16:55:26.818, value=23
 2015001                     column=Description:S_Name, timestamp=2021-11-22T16:55:15.310, value=Li Lei
 2015001                     column=Description:S_Sex, timestamp=2021-11-22T16:55:21.121, value=male
 2015002                     column=Contact:Email, timestamp=2021-11-22T17:09:21.277, value=hmm@qq.com
 2015002                     column=Course2:C_Credit, timestamp=2021-11-22T16:59:54.253, value=5.0
 2015002                     column=Course2:C_Name, timestamp=2021-11-22T16:59:47.968, value=Computer Science
 2015002                     column=Course2:C_No, timestamp=2021-11-22T16:59:40.819, value=123002
 2015002                     column=Course2:C_Score, timestamp=2021-11-22T17:00:00.250, value=77
 2015002                     column=Course3:C_Credit, timestamp=2021-11-22T17:01:12.822, value=3.0
 2015002                     column=Course3:C_Name, timestamp=2021-11-22T17:01:07.737, value=English
 2015002                     column=Course3:C_No, timestamp=2021-11-22T17:01:02.217, value=123003
 2015002                     column=Course3:C_Score, timestamp=2021-11-22T17:01:17.014, value=99
 2015002                     column=Description:S_Age, timestamp=2021-11-22T16:55:43.461, value=22
 2015002                     column=Description:S_Name, timestamp=2021-11-22T16:55:32.456, value=Han Meimei
 2015002                     column=Description:S_Sex, timestamp=2021-11-22T16:55:38.114, value=female
 2015003                     column=Contact:Email, timestamp=2021-11-22T17:09:26.377, value=zs@qq.com
 2015003                     column=Description:S_Age, timestamp=2021-11-22T16:55:59.555, value=24
 2015003                     column=Description:S_Name, timestamp=2021-11-22T16:55:49.113, value=Zhang San
 2015003                     column=Description:S_Sex, timestamp=2021-11-22T16:55:53.993, value=male
3 row(s)
Took 0.0404 seconds
```

5. 删除所创建的表

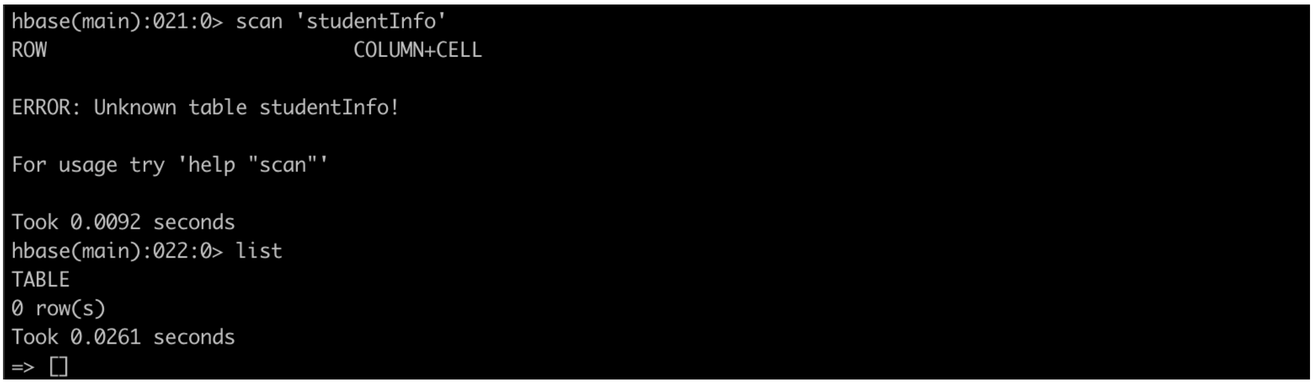
要删除（drop）的表必须是 disable 的，所以现将'studentInfo' disable，在执行drop命令

```
disable 'studentInfo'
drop 'studentInfo'
```



```
hbase(main):019:0> disable 'studentInfo'
Took 0.3642 seconds
hbase(main):020:0> drop 'studentInfo'
Took 0.3534 seconds
```

执行scan和list操作验证，删除成功



```
hbase(main):021:0> scan 'studentInfo'
ROW                          COLUMN+CELL

ERROR: Unknown table studentInfo!

For usage try 'help "scan"'

Took 0.0092 seconds
hbase(main):022:0> list
TABLE
0 row(s)
Took 0.0261 seconds
=> []
```

Java编程实现

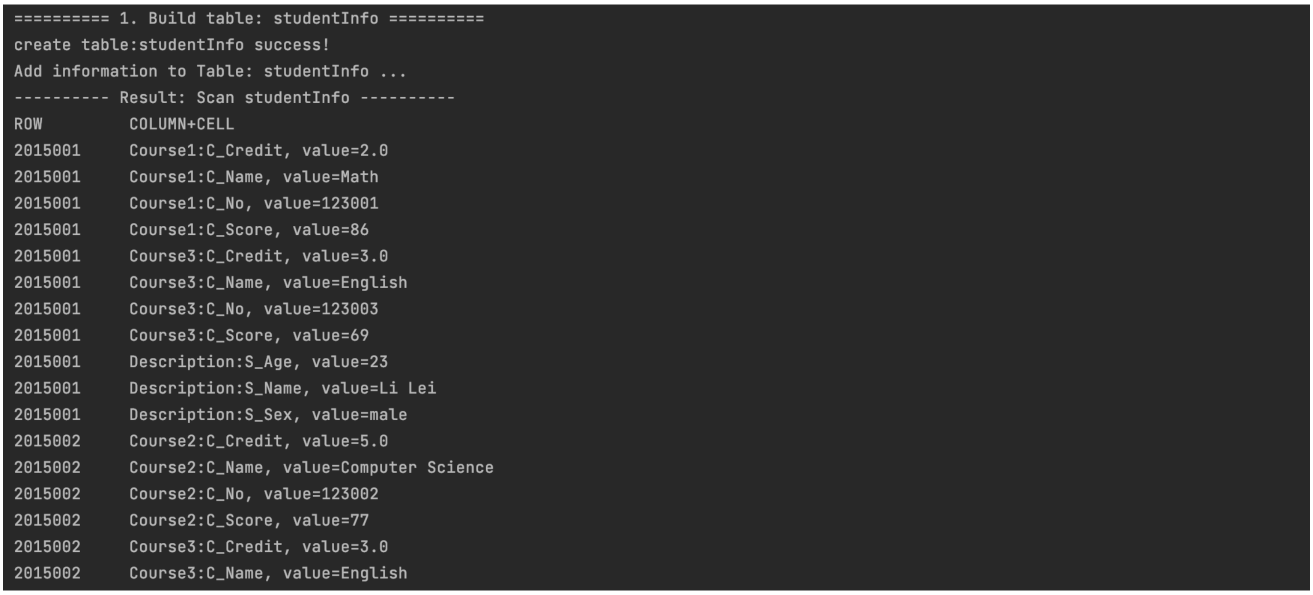
实现的 hbase java api

```
public class HbaseOpt {
  getConnection(){} //创建连接
  closeConnection(){} //关闭连接
  createTable(String tableName, String familyNames[]){} //创建表
  dropTable(String tableName){} //删除表
  insert(String tableName, String rowKey, String family,
         String column, String value){} //指定单元格中插入数据

  delete(String tableName, String rowKey, String family,
         String column){} //删除表中的指定单元格
  delete(String tableName, String rowKey, String family){}//删除表中指定行键的指定列族
  alter(String tableName, String family){} //添加新的列族

  scan(String tableName){} //显示全表
  scan(String tableName, String family, String column){} //按列查询
  scan(String tableName, String rowkey){} //按行键查询
  list(){} //列出现有表
}
```

1. Build table: studentInfo

```
System.out.println("========== 1. Build table: studentInfo ==========");
String[] familyNames= new String[] {"Description", "Course1", "Course2",
"Course3"};
createTable("studentInfo",familyNames);//创建表
System.out.println("Add information to Table: studentInfo ...");
//添加学生信息
//指定单元格中插入数据
insert(  "studentInfo","2015001","Description","S_Name", "Li Lei");
insert(  "studentInfo","2015001","Description","S_Sex", "male");
insert(  "studentInfo","2015001","Description","S_Age", "23");
....
//添加学生的课程信息
insert(  "studentInfo","2015001","Course1","C_No", "123001");
insert(  "studentInfo","2015001","Course1","C_Name", "Math");
insert(  "studentInfo","2015001","Course1","C_Credit", "2.0");
insert(  "studentInfo","2015001","Course1","C_Score", "86");
....
System.out.println("---------- Result: Scan studentInfo ----------");
scan("studentInfo");//显示全表
```

先创建表，再添加信息，使用函数：`createTable()，insert()，scan()`，运行结果如下（局部截图）：

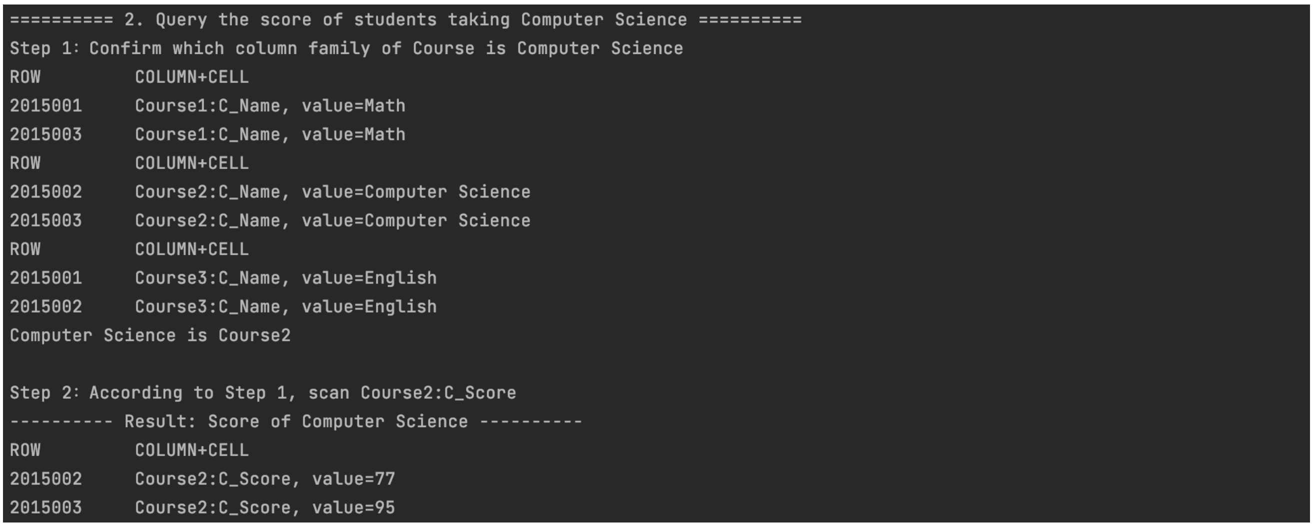


```
========== 1. Build table: studentInfo ==========
create table:studentInfo success!
Add information to Table: studentInfo ...
---------- Result: Scan studentInfo ----------
ROW         COLUMN+CELL
2015001     Course1:C_Credit, value=2.0
2015001     Course1:C_Name, value=Math
2015001     Course1:C_No, value=123001
2015001     Course1:C_Score, value=86
2015001     Course3:C_Credit, value=3.0
2015001     Course3:C_Name, value=English
2015001     Course3:C_No, value=123003
2015001     Course3:C_Score, value=69
2015001     Description:S_Age, value=23
2015001     Description:S_Name, value=Li Lei
2015001     Description:S_Sex, value=male
2015002     Course2:C_Credit, value=5.0
2015002     Course2:C_Name, value=Computer Science
2015002     Course2:C_No, value=123002
2015002     Course2:C_Score, value=77
2015002     Course3:C_Credit, value=3.0
2015002     Course3:C_Name, value=English
```

2. Query the score of students taking Computer Science

```
System.out.println("\n\n========== 2. Query the score of students taking Computer Science ==========");
System.out.println("Step 1: Confirm which column family of Course is Computer Science");
scan("studentInfo", "Course1", "C_Name");//按列查询
scan("studentInfo", "Course2", "C_Name");
scan("studentInfo", "Course3", "C_Name");
System.out.println("Computer Science is Course2");
System.out.println("\nStep 2: According to Step 1, scan Course2:C_Score");
System.out.println("---------- Result: Score of Computer Science ----------");
scan("studentInfo", "Course2", "C_Score");//按列查询
```

先找出Computer Science所属的列族，再查询对应列族的列，使用函数：`scan()`，运行结果如下：

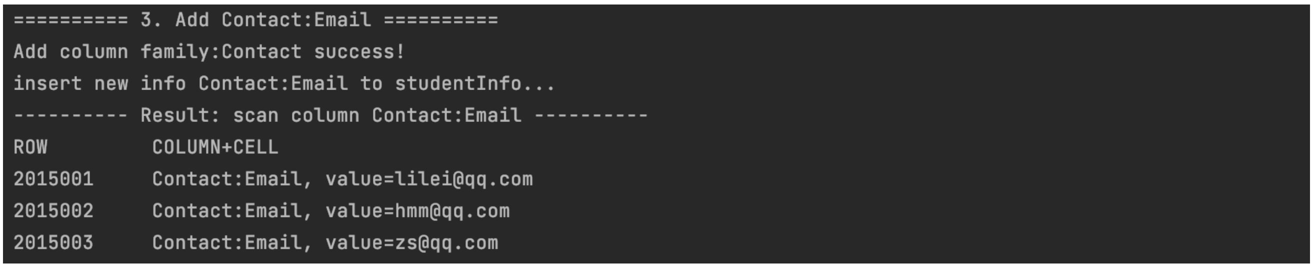


```
========== 2. Query the score of students taking Computer Science ==========
Step 1: Confirm which column family of Course is Computer Science
ROW         COLUMN+CELL
2015001     Course1:C_Name, value=Math
2015003     Course1:C_Name, value=Math
ROW         COLUMN+CELL
2015002     Course2:C_Name, value=Computer Science
2015003     Course2:C_Name, value=Computer Science
ROW         COLUMN+CELL
2015001     Course3:C_Name, value=English
2015002     Course3:C_Name, value=English
Computer Science is Course2

Step 2: According to Step 1, scan Course2:C_Score
---------- Result: Score of Computer Science ----------
ROW         COLUMN+CELL
2015002     Course2:C_Score, value=77
2015003     Course2:C_Score, value=95
```

3. Add Contact:Email

```
System.out.println("\n\n========== 3. Add Contact:Email ==========");
alter("studentInfo", "Contact");//添加新的列族
System.out.println("insert new info Contact:Email to studentInfo...");
insert("studentInfo", "2015001", "Contact", "Email", "lilei@qq.com");//指定单元格中插入数据
insert("studentInfo", "2015002", "Contact", "Email", "hmm@qq.com");
insert("studentInfo", "2015003", "Contact", "Email", "zs@qq.com");
System.out.println("---------- Result: scan column Contact:Email ----------");
scan("studentInfo", "Contact", "Email");//按列查询
```

先添加Contact列族，再输入Contact:Email信息，使用的函数：`alter(), insert(), scan()`，运行结果如下：



```
========== 3. Add Contact:Email ==========
Add column family:Contact success!
insert new info Contact:Email to studentInfo...
---------- Result: scan column Contact:Email ----------
ROW         COLUMN+CELL
2015001     Contact:Email, value=lilei@qq.com
2015002     Contact:Email, value=hmm@qq.com
2015003     Contact:Email, value=zs@qq.com
```

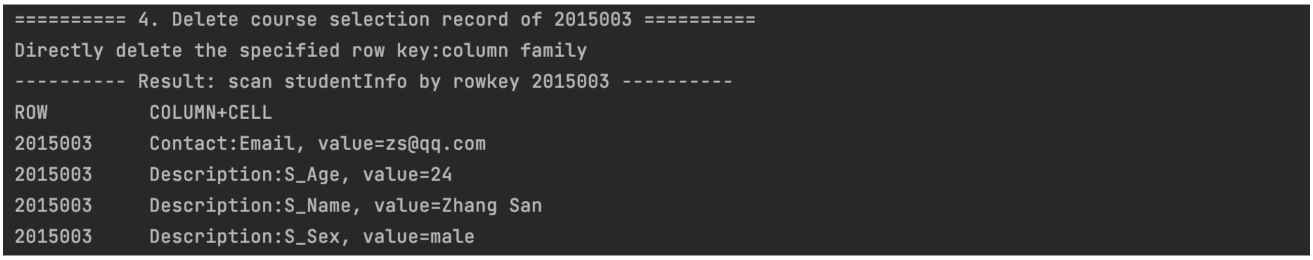
4. Delete course selection record of 2015003

采用两种删除方式：（1）直接删除表中指定行键的指定列族（2）逐个删除单元格

```
System.out.println("\n\n========== 4. Delete course selection record of 2015003 ==========");
//直接删除表中指定行键的指定列族
System.out.println("Directly delete the specified row key:column family");
delete("studentInfo", "2015003", "Course1");
delete("studentInfo", "2015003", "Course2");
delete("studentInfo", "2015003", "Course3");
////逐个删除单元格
//System.out.println("Delete related cells one by one");
//delete("studentInfo", "2015003","Course1", "C_No");
//delete("studentInfo", "2015003","Course1", "C_Name");
//delete("studentInfo", "2015003","Course1", "C_Credit");
//delete("studentInfo", "2015003","Course1", "C_Score");
//delete("studentInfo", "2015003","Course2", "C_No");
//delete("studentInfo", "2015003","Course2", "C_Name");
//delete("studentInfo", "2015003","Course2", "C_Credit");
//delete("studentInfo", "2015003","Course2", "C_Score");
//delete("studentInfo", "2015003","Course3", "C_No");
//delete("studentInfo", "2015003","Course3", "C_Name");
//delete("studentInfo", "2015003","Course3", "C_Credit");
//delete("studentInfo", "2015003","Course3", "C_Score");
```

```
System.out.println("---------- Result: scan studentInfo by rowkey 2015003 --------
--");
scan("studentInfo","2015003");//按行键查询
```

程序中默认采用方法（1），方法（2）被注释掉了。使用的函数：`delete(), scan()`，运行结果如下：



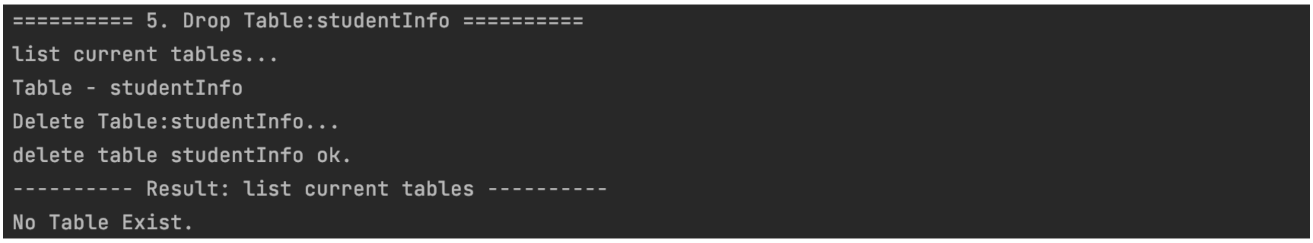
```
========== 4. Delete course selection record of 2015003 ==========
Directly delete the specified row key:column family
---------- Result: scan studentInfo by rowkey 2015003 ----------
ROW         COLUMN+CELL
2015003     Contact:Email, value=zs@qq.com
2015003     Description:S_Age, value=24
2015003     Description:S_Name, value=Zhang San
2015003     Description:S_Sex, value=male
```

按行键"2015003"查询可以看到，选课信息已经被删除了。

5. Drop Table:studentInfo

```
System.out.println("\n\n========== 5. Drop Table:studentInfo ==========");
System.out.println("list current tables:");
list(); //查询现在所有的表
System.out.println("Delete Table:studentInfo...");
dropTable("studentInfo"); //删除表
System.out.println("---------- Result: list current tables ----------");
list(); //查询所有表，验证删除是否成功
```

先查询所有的表，再执行函数删除表，删除后再次查询所有表验证是否删除成功，使用的函数：`list(), dropTable()`，运行结果如下：



```
========== 5. Drop Table:studentInfo ==========
list current tables...
Table - studentInfo
Delete Table:studentInfo...
delete table studentInfo ok.
---------- Result: list current tables ----------
No Table Exist.
```

三、问题总结及解决方案

HBase中遇到的问题

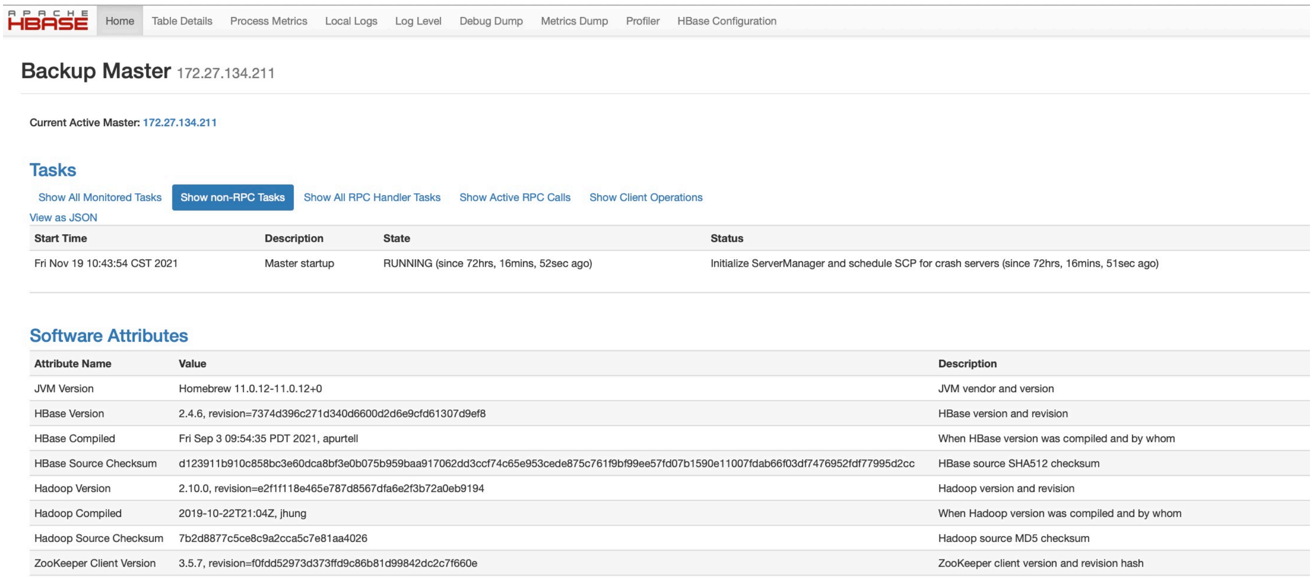
❌ HMaster、HRegionServer在开启后自动关闭的问题

在执行`start-hbase.sh`开启HBase后，没有任何操作的情况下，HMaster和HRegionServer几秒后自动关闭，导致Web界面加载失败。

原因是HBase自带的zookeeper不够稳定，在`hbase-env.sh`中修改`HBASE_MANAGES_ZK`参数为false，使用外置的zookeeper服务端，解决自动关闭问题。

❌WebUI不显示Region Sever问题

开启HBase后，使用`jps`命令查看各个端口发现一切正常，但是WebUI界面显示没有Region Sever，WebUI界面显示异常（如下图），与此同时，执行`stop-hbase.sh`时不能正常关闭，也无法连接HBase shell。



Backup Master 172.27.134.211

Current Active Master: 172.27.134.211

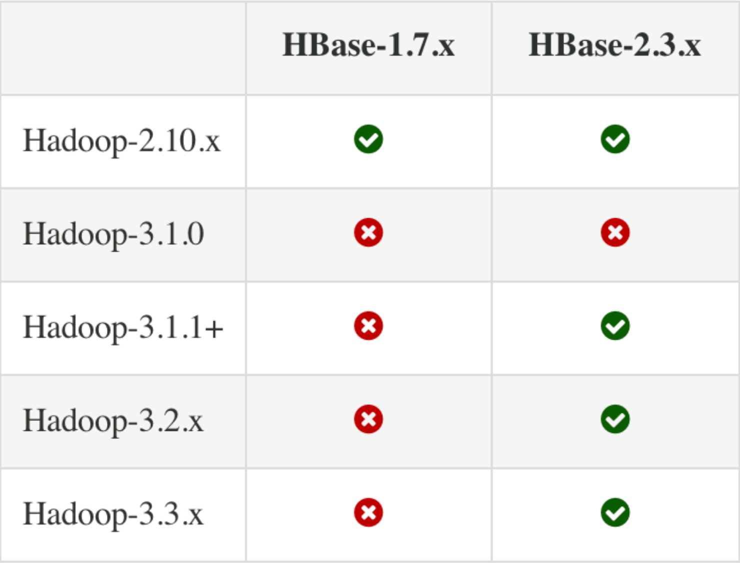
Tasks

Start Time	Description	State	Status
Fri Nov 19 10:43:54 CST 2021	Master startup	RUNNING (since 72hrs, 16mins, 52sec ago)	Initialize ServerManager and schedule SCP for crash servers (since 72hrs, 16mins, 51sec ago)

Software Attributes

Attribute Name	Value	Description
JVM Version	Homebrew 11.0.12-11.0.12+0	JVM vendor and version
HBase Version	2.4.6, revision=7374d396c271d340d6600d2d6e9cfd61307d9ef8	HBase version and revision
HBase Compiled	Fri Sep 3 09:54:35 PDT 2021, apurtell	When HBase version was compiled and by whom
HBase Source Checksum	d123911b910c858bc3e60dca8bf3e0b075b959baa917062dd3ccf74c65e953cede875c761f9bf99ee57fd07b1590e11007fdab66f03df7476952fdf77995d2cc	HBase source SHA512 checksum
Hadoop Version	2.10.0, revision=e2f1f118e465e787d8567dfa6e2f3b72a0eb9194	Hadoop version and revision
Hadoop Compiled	2019-10-22T21:04Z, jhung	When Hadoop version was compiled and by whom
Hadoop Source Checksum	7b2d8877c5ce8c9a2cca5c7e81aa4026	Hadoop source MD5 checksum
ZooKeeper Client Version	3.5.7, revision=f0fdd52973d373ffd9c86b81d99842dc2c7f660e	ZooKeeper client version and revision hash

尝试了很多方法都没有解决，甚至把自己的hadoop搞崩了，又重装了一次😭。后来，突然想到会不会是hadoop版本和hbase版本不兼容的问题，于是在官网上找到了这样一张表：

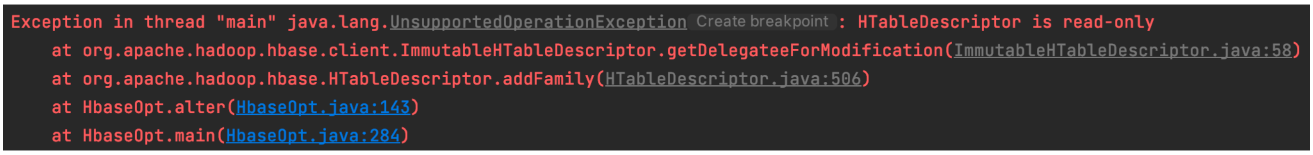


	HBase-1.7.x	HBase-2.3.x
Hadoop-2.10.x	✓	✓
Hadoop-3.1.0	✗	✗
Hadoop-3.1.1+	✗	✓
Hadoop-3.2.x	✗	✓
Hadoop-3.3.x	✗	✓

最后一行，可以确定的是，hadoop3.3.x和hbase2.3.x版本肯定可以适配，而我安装的、出现问题的版本是hadoop3.3.1和hbase2.4.8，确实可能有版本不兼容的问题。于是，我重新下载安装了低版本的hbase2.3.7，终于解决了WebUI不显示Region Sever的问题😭。

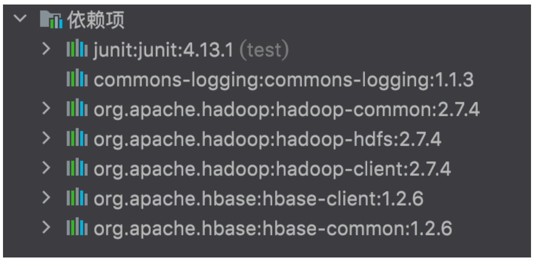
Java编程中遇到的问题

❌UnsupportedOperationException: HTableDescriptor is read-only



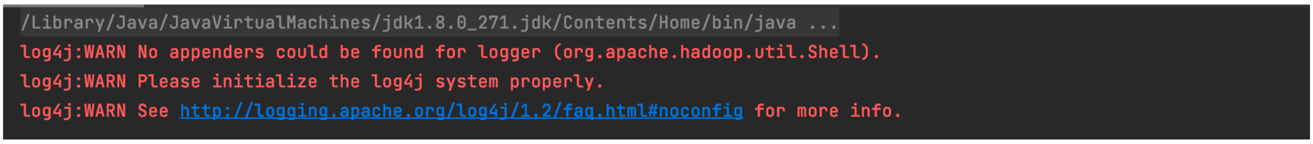
```
Exception in thread "main" java.lang.UnsupportedOperationException Create breakpoint : HTableDescriptor is read-only
	at org.apache.hadoop.hbase.client.ImmutableHTableDescriptor.getDelegateeForModification(ImmutableHTableDescriptor.java:58)
	at org.apache.hadoop.hbase.HTableDescriptor.addFamily(HTableDescriptor.java:506)
	at HbaseOpt.alter(HbaseOpt.java:143)
	at HbaseOpt.main(HbaseOpt.java:284)
```

运行程序，在添加列族Contact时出现报错提示：HTableDescriptor 是只读的。经检查发现，是maven中安装依赖包版本的问题，将依赖包版本进行调整后，错误消失。



```
依赖项
  junit:junit:4.13.1 (test)
  commons-logging:commons-logging:1.1.3
  org.apache.hadoop:hadoop-common:2.7.4
  org.apache.hadoop:hadoop-hdfs:2.7.4
  org.apache.hadoop:hadoop-client:2.7.4
  org.apache.hbase:hbase-client:1.2.6
  org.apache.hbase:hbase-common:1.2.6
```

❌java 程序运行时没有任何输出



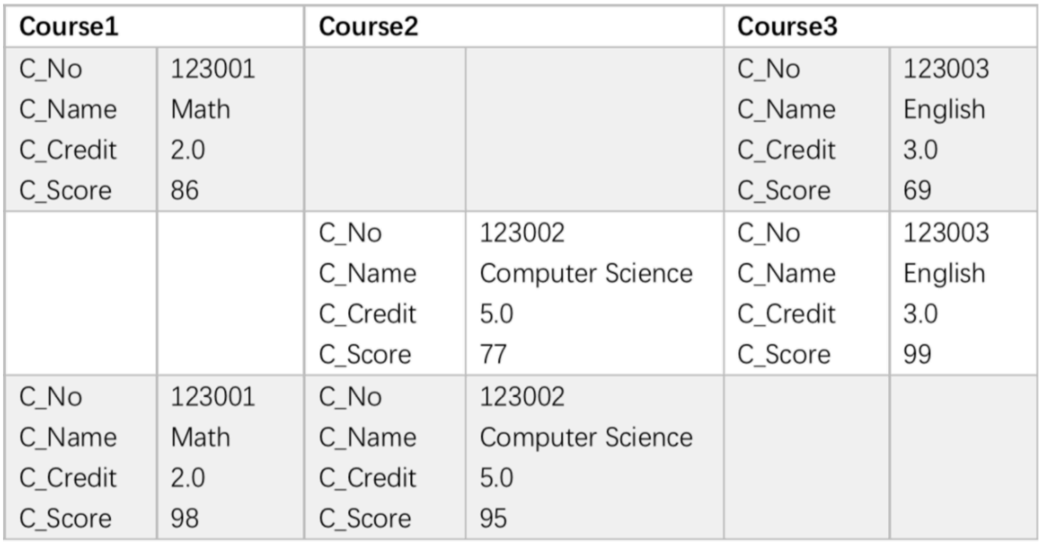
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_271.jdk/Contents/Home/bin/java ...
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

需要先在本地开启 hbase，再运行java程序。

四、其他思考

关于表格设计

本实验将三个表格'student'、'course'、'sc'的信息整合到一张表'studentInfo'中进行各种数据操作，虽然避免了跨表查询，但也导致了部分信息的重复存储，例如同一个Course的C_No、C_Name、C_Credit信息，存在可优化的空间。



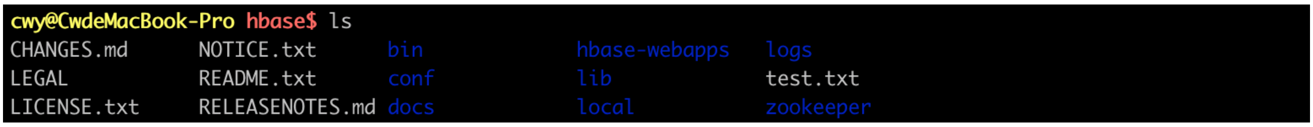
Course1		Course2		Course3	
C_No C_Name C_Credit C_Score	123001 Math 2.0 86			C_No C_Name C_Credit C_Score	123003 English 3.0 69
		C_No C_Name C_Credit C_Score	123002 Computer Science 5.0 77	C_No C_Name C_Credit C_Score	123003 English 3.0 99
C_No C_Name C_Credit C_Score	123001 Math 2.0 98	C_No C_Name C_Credit C_Score	123002 Computer Science 5.0 95		

HBase shell执行脚本

在HBase shell上进行操作时，觉得一条一条添加数据效率较低，在网上了解到可以通过执行脚本批量添加数据，于是尝试在HBase shell运行脚本完成（二）中的建表操作。

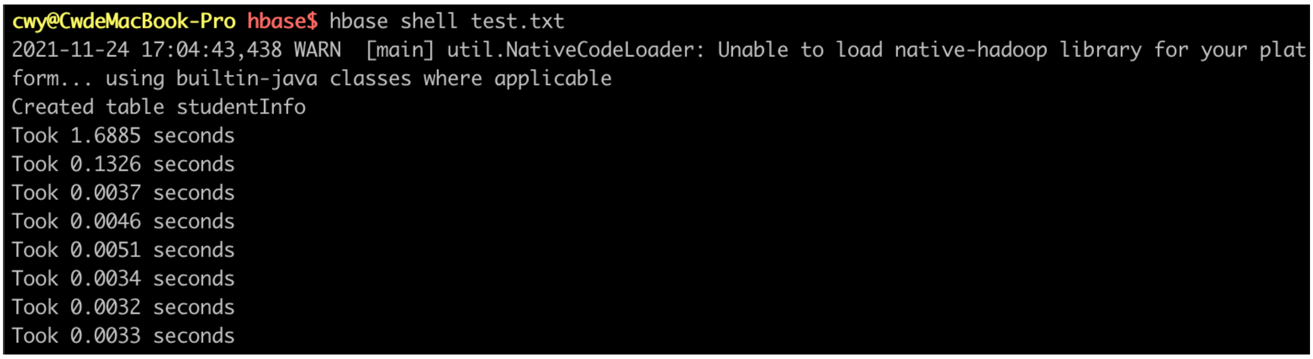
首先，编写一个文本文件`test.txt`，将其放在`/usr/local/Cellar/hbase/test.txt`目录下。

```
#test.txt
create 'studentInfo','S_No','Description','Course1','Course2','Course3'
put 'studentInfo','2015001','Description:S_Name','Li Lei'
put 'studentInfo','2015001','Description:S_Sex','male'
...
put 'studentInfo','2015002','Course3:C_Credit','3.0'
put 'studentInfo','2015002','Course3:C_Score','99'
scan 'studentInfo'
```



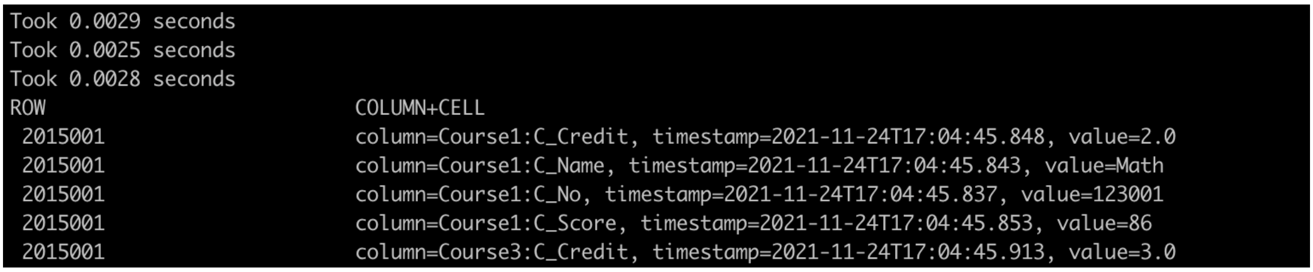
```
cwy@CwdeMacBook-Pro hbase$ ls
CHANGES.md      NOTICE.txt      bin             hbase-webapps   logs
LEGAL           README.txt      conf            lib             test.txt
LICENSE.txt     RELEASENOTES.md docs            local           zookeeper
```

在HBase 中执行`hbase shell test.txt`运行这个脚本：



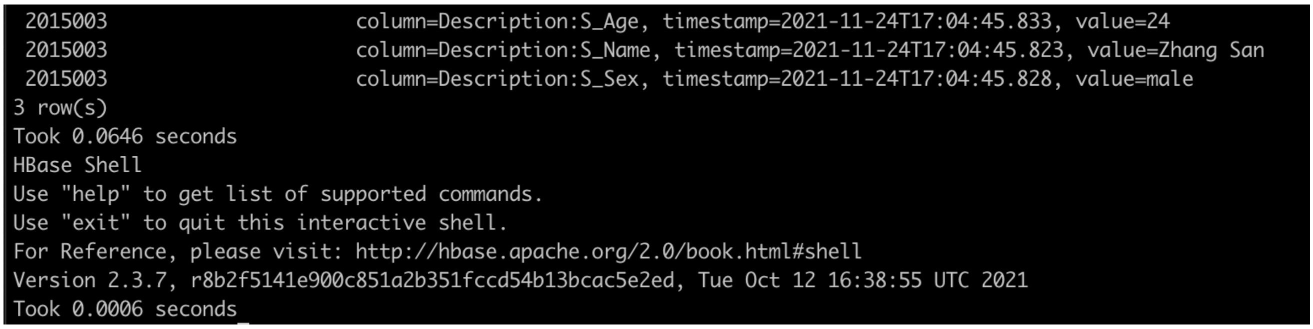
```
cwy@CwdeMacBook-Pro hbase$ hbase shell test.txt
2021-11-24 17:04:43,438 WARN  [main] util.NativeCodeLoader: Unable to load native-hadoop library for your plat
form... using builtin-java classes where applicable
Created table studentInfo
Took 1.6885 seconds
Took 0.1326 seconds
Took 0.0037 seconds
Took 0.0046 seconds
Took 0.0051 seconds
Took 0.0034 seconds
Took 0.0032 seconds
Took 0.0033 seconds
```

> …



```
Took 0.0029 seconds
Took 0.0025 seconds
Took 0.0028 seconds
ROW                          COLUMN+CELL
 2015001                     column=Course1:C_Credit, timestamp=2021-11-24T17:04:45.848, value=2.0
 2015001                     column=Course1:C_Name, timestamp=2021-11-24T17:04:45.843, value=Math
 2015001                     column=Course1:C_No, timestamp=2021-11-24T17:04:45.837, value=123001
 2015001                     column=Course1:C_Score, timestamp=2021-11-24T17:04:45.853, value=86
 2015001                     column=Course3:C_Credit, timestamp=2021-11-24T17:04:45.913, value=3.0
```

> …



```
 2015003                     column=Description:S_Age, timestamp=2021-11-24T17:04:45.833, value=24
 2015003                     column=Description:S_Name, timestamp=2021-11-24T17:04:45.823, value=Zhang San
 2015003                     column=Description:S_Sex, timestamp=2021-11-24T17:04:45.828, value=male
3 row(s)
Took 0.0646 seconds
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 2.3.7, r8b2f5141e900c851a2b351fccd54b13bcac5e2ed, Tue Oct 12 16:38:55 UTC 2021
Took 0.0006 seconds
```

可以看出，执行`test.txt`脚本自动完成了表studentInfo的创建并添加数据的操作，提升了执行效率。

Hbase shell 命令

在Hbase shell中操作时，发现Hbase shell没有提供删除指定行键指定列族的操作命令，导致在删除学号为2015003的学生的选课记录时，需要逐个删除相关的单元格；同时，Hbase shell也无法根据列值来反向查找对应的列族、行键等信息，推测这可能与Hbase 按列族存储的性质有关。

在编写的Java api中，实现了“删除表中指定行键的指定列族”命令，但“根据列值反向查找”暂时还没有找到合适的实现方式，究竟是否能够通过Java api实现还可以进行进一步的探索。