

# 实验3 Hbase

---

朱彤轩 191840376

## 实验3 Hbase

### 1 Hbase的安装及配置

#### 1.1 单机安装

#### 1.2 伪分布

[过渡] 表设计

### 2 Shell实现

#### 2.1 创建合适的表

##### 2.1.1 创建表

##### 2.1.2 添加学生信息

##### 2.1.3 添加选课信息

#### 2.2 查询选修Computer Science的学生的成绩

#### 2.3 增加新的列族和新列Contact:Email，并添加数据

#### 2.4 删除学号为2015003的学生的选课记录

#### 2.5 删除所创建的表

### 3 Java实现

#### 3.1 创建表

#### 3.2 查询选修Computer Science的学生的成绩

#### 3.3 增加新的列族和新列Contact:Email，并添加数据

#### 3.4 删除学号为2015003的学生的选课记录

#### 3.5 删除所创建的表

### 4 问题总结及解决方案

#### 4.1 发生jar包冲突

#### 4.2 HMaster执行几秒之后就消失

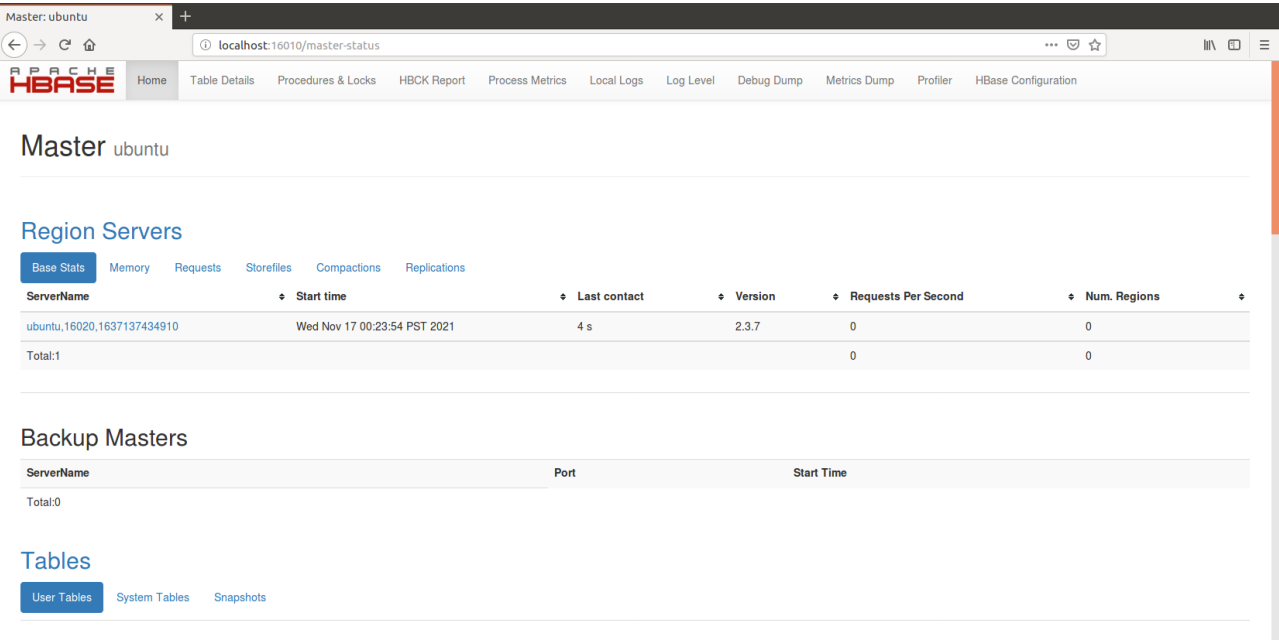
#### 4.3 windows下intellij maven用javaAPI写hbase程序编译出错

### 5 其它思考

### 6 文件说明

## 1 Hbase的安装及配置

# 1.1 单机安装



# 1.2 伪分布

如图，打开了如下的进程：

```
ztx@ubuntu:~$ start-hbase.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hadoop-2.10.1/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/hbase-2.3.7/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hadoop-2.10.1/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/hbase-2.3.7/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:HcnCTRA+CTT4JdH+UX4JRHZ0CnynB7IbCN7+TEQzYZA.
Are you sure you want to continue connecting (yes/no)? yes
127.0.0.1: Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
127.0.0.1: running zookeeper, logging to /opt/hbase/hbase-2.3.7/bin/../logs/hbase-ztx-zookeeper-ubuntu.out
running master, logging to /opt/hbase/hbase-2.3.7/logs/hbase-ztx-master-ubuntu.out
running regionserver, logging to /opt/hbase/hbase-2.3.7/logs/hbase-ztx-regionserver-ubuntu.out
ztx@ubuntu:~$ jps
15728 DataNode
18210 HMaster
18310 HRegionServer
18519 Jps
15911 SecondaryNameNode
16231 NodeManager
15608 NameNode
16109 ResourceManager
18110 HQuorumPeer
```

Master: ubuntu

localhost:16010/master-status

APACHEHBASEHomeTable DetailsProcedures & LocksHBCK ReportProcess MetricsLocal LogsLog LevelDebug DumpMetrics DumpProfilerHBase Configuration

Master

ubuntu

Region Servers

Base StatsMemoryRequestsStorefilesCompactionsReplications

ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
ubuntu,16020,1637138122473	Wed Nov 17 00:35:22 PST 2021	2 s	2.3.7	0	2
Total:1				0	2

Backup Masters

ServerName	Port	Start Time
Total:0		

Tables

User TablesSystem TablesSnapshots

Browsing HDFS

localhost:50070/explorer.html#/hbase

HadoopOverviewDatanodesDatanode Volume FailuresSnapshotStartup ProgressUtilities

Browse Directory

/hbase

Go!

Show 25 entries

Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	ztx	supergroup	0 B	Nov 17 00:35	0	0 B	.hbck
drwxr-xr-x	ztx	supergroup	0 B	Nov 17 00:36	0	0 B	.tmp
drwxr-xr-x	ztx	supergroup	0 B	Nov 17 00:35	0	0 B	MasterData
drwxr-xr-x	ztx	supergroup	0 B	Nov 17 00:35	0	0 B	WALS
drwxr-xr-x	ztx	supergroup	0 B	Nov 17 00:35	0	0 B	archive
drwxr-xr-x	ztx	supergroup	0 B	Nov 17 00:35	0	0 B	corrupt
drwxr-xr-x	ztx	supergroup	0 B	Nov 17 00:36	0	0 B	data
-rw-r--r--	ztx	supergroup	42 B	Nov 17 00:35	1	128 MB	hbase.id
-rw-r--r--	ztx	supergroup	7 B	Nov 17 00:35	1	128 MB	hbase.version
drwxr-xr-x	ztx	supergroup	0 B	Nov 17 00:35	0	0 B	mobdir

## [过渡] 表设计

如图所示，**学号**作为行键，有**s\_info**，**c\_1**，**c\_2**，**c\_3**四个列族。**s\_Name**，**s\_Sex**，**s\_Age**作为**S\_info**的列族限定，**sc\_Sno**，**c\_Credit**，**sc\_Score**为**C\_1**，**C\_2**，**C\_3**的列族限定。

S No	S info		C 1		C 2		C 3	
2015001	S_info: S_Name S_info: S_Sex S_info: S_Age	Li Lei Male 23	C_1: SC_Cno C_1: C_Name C_1: C_Credit C_1: SC_Score	123001 Math 2 86			C_3: SC_Cno C_3: C_Name C_3: C_Credit C_3: SC_Score	123003 English 3 69
2015002	S_info: S_Name S_info: S_Sex S_info: S_Age	Han Meimei female 22			C_2: SC_Cno C_2: C_Name C_2: C_Credit C_2: SC_Score	123002 Computer Science 5 77	C_3: SC_Cno C_3: C_Name C_3: C_Credit C_3: SC_Score	123003 English 3 99
2015003	S_info: S_Name S_info: S_Sex S_info: S_Age	Zhang San male 24	C_1: SC_Cno C_1: C_Name C_1: C_Credit C_1: SC_Score	123001 Math 2 98	C_2: SC_Cno C_2: C_Name C_2: C_Credit C_2: SC_Score	123002 Computer Science 5 95		

## 2 Shell实现

### 2.1 创建合适的表

#### 2.1.1 创建表

表名: studentInfoDB; 三个列族: S\_info, C\_1, C\_2, C\_3

```
create 'studentInfoDB', 'S_info', 'C_1', 'C_2', 'C_3'
```

```
ztx@ubuntu:~$ start-hbase.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hadoop-2.10.1/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/hbase-2.3.7/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hadoop-2.10.1/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/hbase-2.3.7/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
127.0.0.1: zookeeper running as process 18110. Stop it first.
master running as process 18210. Stop it first.
; regionserver running as process 18310. Stop it first.
ztx@ubuntu:~$ hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop/hadoop-2.10.1/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/hbase-2.3.7/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
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 0.0247 seconds
hbase(main):001:0> create 'studentInfoDB', 'S_info', 'C_1', 'C_2', 'C_3'
Created table studentInfoDB
Took 4.9297 seconds
=> Hbase::Table - studentInfoDB
hbase(main):002:0>
```

如上图, 先用 **hbase shell** 命令打开Shell, 然后新建表。

#### 2.1.2 添加学生信息

```

put 'studentInfoDB','2015001','S_info:S_Name','Li Lei'
put 'studentInfoDB','2015001','S_info:S_Sex','male'
put 'studentInfoDB','2015001','S_info:S_Age','23'

put 'studentInfoDB','2015002','S_info:S_Name','Han Meimei'
put 'studentInfoDB','2015002','S_info:S_Sex','female'
put 'studentInfoDB','2015002','S_info:S_Age','22'

put 'studentInfoDB','2015003','S_info:S_Name','Zhang san'
put 'studentInfoDB','2015003','S_info:S_Sex','male'
put 'studentInfoDB','2015003','S_info:S_Age','24'

```

```

hbase(main):002:0> put 'studentInfoDB','2015001','S_info:S_Name','Li Lei'
Took 0.8371 seconds
hbase(main):003:0> put 'studentInfoDB','2015001','S_info:S_Sex','male'
Took 0.0304 seconds
hbase(main):004:0> put 'studentInfoDB','2015001','S_info:S_Age','23'
Took 0.0241 seconds
hbase(main):005:0>
hbase(main):006:0* put 'studentInfoDB','2015002','S_info:S_Name','Han Meimei'
Took 0.0318 seconds
hbase(main):007:0> put 'studentInfoDB','2015002','S_info:S_Sex','female'
Took 0.0229 seconds
hbase(main):008:0> put 'studentInfoDB','2015002','S_info:S_Age','22'
Took 0.0254 seconds
hbase(main):009:0>
hbase(main):010:0* put 'studentInfoDB','2015003','S_info:S_Name','Zhang san'
Took 0.0248 seconds
hbase(main):011:0> put 'studentInfoDB','2015003','S_info:S_Sex','male'
Took 0.0221 seconds
hbase(main):012:0> put 'studentInfoDB','2015003','S_info:S_Age','24'
Took 0.0220 seconds
hbase(main):013:0> 

```

### 2.1.3 添加选课信息

```

put 'studentInfoDB','2015001','C_1:SC_Cno','123001'
put 'studentInfoDB','2015001','C_1:C_Name','Math'
put 'studentInfoDB','2015001','C_1:C_Credit','2.0'
put 'studentInfoDB','2015001','C_1:SC_Score','86'
put 'studentInfoDB','2015001','C_3:SC_Cno','123003'
put 'studentInfoDB','2015001','C_3:C_Name','English'
put 'studentInfoDB','2015001','C_3:C_Credit','3.0'
put 'studentInfoDB','2015001','C_3:SC_Score','69'

put 'studentInfoDB','2015002','C_2:SC_Cno','123002'
put 'studentInfoDB','2015002','C_2:C_Name','Computer Science'
put 'studentInfoDB','2015002','C_2:C_Credit','5.0'
put 'studentInfoDB','2015002','C_2:SC_Score','77'

```

```
put 'studentInfoDB', '2015002', 'C_3:SC_Cno', '123003'
put 'studentInfoDB', '2015002', 'C_3:C_Name', 'English'
put 'studentInfoDB', '2015002', 'C_3:C_Credit', '3.0'
put 'studentInfoDB', '2015002', 'C_3:SC_Score', '99'

put 'studentInfoDB', '2015003', 'C_1:SC_Cno', '123001'
put 'studentInfoDB', '2015003', 'C_1:C_Name', 'Math'
put 'studentInfoDB', '2015003', 'C_1:C_Credit', '2.0'
put 'studentInfoDB', '2015003', 'C_1:SC_Score', '98'
put 'studentInfoDB', '2015003', 'C_2:SC_Cno', '123002'
put 'studentInfoDB', '2015003', 'C_2:C_Name', 'Computer Science'
put 'studentInfoDB', '2015003', 'C_2:C_Credit', '5.0'
put 'studentInfoDB', '2015003', 'C_2:SC_Score', '95'
```

```

Took 0.0134 seconds
hbase(main):015:0> put 'studentInfoDB','2015001','C_1:C_Credit','2.0'
Took 0.0487 seconds
hbase(main):016:0> put 'studentInfoDB','2015001','C_1:SC_Score','86'
Took 0.0182 seconds
hbase(main):017:0> put 'studentInfoDB','2015001','C_3:SC_Cno','123003'
Took 0.0179 seconds
hbase(main):018:0> put 'studentInfoDB','2015001','C_3:C_Name','English'
Took 0.0269 seconds
hbase(main):019:0> put 'studentInfoDB','2015001','C_3:C_Credit','3.0'
Took 0.0159 seconds
hbase(main):020:0> put 'studentInfoDB','2015001','C_3:SC_Score','69'
Took 0.0143 seconds
hbase(main):021:0>
hbase(main):022:0* put 'studentInfoDB','2015002','C_2:SC_Cno','123002'
Took 0.0641 seconds
hbase(main):023:0> put 'studentInfoDB','2015002','C_2:C_Name','Computer Science'
Took 0.0335 seconds
hbase(main):024:0> put 'studentInfoDB','2015002','C_2:C_Credit','5.0'
Took 0.0172 seconds
hbase(main):025:0> put 'studentInfoDB','2015002','C_2:SC_Score','77'
Took 0.0190 seconds
hbase(main):026:0> put 'studentInfoDB','2015002','C_3:SC_Cno','123003'
Took 0.0358 seconds
hbase(main):027:0> put 'studentInfoDB','2015002','C_3:C_Name','English'
Took 0.0265 seconds
hbase(main):028:0> put 'studentInfoDB','2015002','C_3:C_Credit','3.0'
Took 0.0375 seconds
hbase(main):029:0> put 'studentInfoDB','2015002','C_3:SC_Score','99'
Took 0.0193 seconds
hbase(main):030:0>
hbase(main):031:0* put 'studentInfoDB','2015003','C_1:SC_Cno','123001'
Took 0.0270 seconds
hbase(main):032:0> put 'studentInfoDB','2015003','C_1:C_Name','Math'
Took 0.0166 seconds
hbase(main):033:0> put 'studentInfoDB','2015003','C_1:C_Credit','2.0'
Took 0.0283 seconds
hbase(main):034:0> put 'studentInfoDB','2015003','C_1:SC_Score','98'
Took 0.0391 seconds
hbase(main):035:0> put 'studentInfoDB','2015003','C_2:SC_Cno','123003'
Took 0.0435 seconds
hbase(main):036:0> put 'studentInfoDB','2015003','C_2:C_Name','Computer Science'
Took 0.0188 seconds
hbase(main):037:0> put 'studentInfoDB','2015003','C_2:C_Credit','5.0'
Took 0.0190 seconds
hbase(main):038:0> put 'studentInfoDB','2015003','C_2:SC_Score','95'
Took 0.0339 seconds
hbase(main):039:0>

```

## 2.2 查询选修Computer Science的学生的成绩

Computer Science成绩在C\_2:SC\_Score中

```
scan 'studentInfoDB', {COLUMN=>'C_2:SC_Score'}
```

```

hbase(main):039:0> scan 'studentInfoDB', {COLUMN=>'C_2:SC_Score'}
ROW                                COLUMN+CELL
 2015002                           column=C_2:SC_Score, timestamp=2021-11-17T06:53:31.214, value=77
 2015003                           column=C_2:SC_Score, timestamp=2021-11-17T06:53:35.176, value=95
2 row(s)
Took 0.2569 seconds
hbase(main):040:0>

```

如上，运用scan查找出学号为2015002计算机成绩为77，学号为2015003的同学成绩为95。



## 2.3 增加新的列族和新列Contact:Email, 并添加数据

```
alter 'studentInfoDB', 'Contact'

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

```
hbase(main):040:0> alter 'studentInfoDB', 'Contact'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 3.8690 seconds
hbase(main):041:0>
hbase(main):042:0> put 'studentInfoDB', '2015001', 'Contact:Email', 'lilei@qq.com'
Took 0.0114 seconds
hbase(main):043:0> put 'studentInfoDB', '2015002', 'Contact:Email', 'hmm@qq.com'
Took 0.0135 seconds
hbase(main):044:0> put 'studentInfoDB', '2015003', 'Contact:Email', 'zs@qq.com'
Took 0.0296 seconds
hbase(main):045:0>
```

```
hbase(main):047:0> scan 'studentInfoDB'
ROW COLUMN+CELL
2015001 column=C_1:C_Credit, timestamp=2021-11-17T06:53:29.273, value=2.0
2015001 column=C_1:C_Name, timestamp=2021-11-17T06:53:29.009, value=Math
2015001 column=C_1:SC_Cno, timestamp=2021-11-17T06:53:28.848, value=123001
2015001 column=C_1:SC_Score, timestamp=2021-11-17T06:53:29.452, value=86
2015001 column=C_3:C_Credit, timestamp=2021-11-17T06:53:30.439, value=3.0
2015001 column=C_3:C_Name, timestamp=2021-11-17T06:53:29.889, value=English
2015001 column=C_3:SC_Cno, timestamp=2021-11-17T06:53:29.633, value=123003
2015001 column=C_3:SC_Score, timestamp=2021-11-17T06:53:30.581, value=69
2015001 column=Contact:Email, timestamp=2021-11-17T06:54:29.369, value=lilei@qq.com
2015001 column=S_info:S_Age, timestamp=2021-11-17T06:52:36.779, value=23
2015001 column=S_info:S_Name, timestamp=2021-11-17T06:52:36.411, value=Li Lei
2015001 column=S_info:S_Sex, timestamp=2021-11-17T06:52:36.595, value=male
2015002 column=C_2:C_Credit, timestamp=2021-11-17T06:53:31.082, value=5.0
2015002 column=C_2:C_Name, timestamp=2021-11-17T06:53:30.952, value=Computer Science
2015002 column=C_2:SC_Cno, timestamp=2021-11-17T06:53:30.797, value=123002
2015002 column=C_2:SC_Score, timestamp=2021-11-17T06:53:31.214, value=77
2015002 column=C_3:C_Credit, timestamp=2021-11-17T06:53:31.602, value=3.0
2015002 column=C_3:C_Name, timestamp=2021-11-17T06:53:31.464, value=English
2015002 column=C_3:SC_Cno, timestamp=2021-11-17T06:53:31.328, value=123003
2015002 column=C_3:SC_Score, timestamp=2021-11-17T06:53:31.751, value=99
2015002 column=Contact:Email, timestamp=2021-11-17T06:54:29.580, value=hmm@qq.com
2015002 column=S_info:S_Age, timestamp=2021-11-17T06:52:37.486, value=22
2015002 column=S_info:S_Name, timestamp=2021-11-17T06:52:37.101, value=Han Meimei
2015002 column=S_info:S_Sex, timestamp=2021-11-17T06:52:37.314, value=female
2015003 column=C_1:C_Credit, timestamp=2021-11-17T06:53:32.158, value=2.0
2015003 column=C_1:C_Name, timestamp=2021-11-17T06:53:32.038, value=Math
2015003 column=C_1:SC_Cno, timestamp=2021-11-17T06:53:31.916, value=123001
2015003 column=C_1:SC_Score, timestamp=2021-11-17T06:53:32.274, value=98
2015003 column=C_2:C_Credit, timestamp=2021-11-17T06:53:32.711, value=5.0
2015003 column=C_2:C_Name, timestamp=2021-11-17T06:53:32.575, value=Computer Science
2015003 column=C_2:SC_Cno, timestamp=2021-11-17T06:53:32.439, value=123003
2015003 column=C_2:SC_Score, timestamp=2021-11-17T06:53:35.176, value=95
2015003 column=Contact:Email, timestamp=2021-11-17T06:54:36.354, value=zs@qq.com
2015003 column=S_info:S_Age, timestamp=2021-11-17T06:52:42.472, value=24
2015003 column=S_info:S_Name, timestamp=2021-11-17T06:52:37.721, value=Zhang san
2015003 column=S_info:S_Sex, timestamp=2021-11-17T06:52:37.884, value=male
3 row(s)
Took 0.5774 seconds
hbase(main):048:0>
```

运用scan扫描全表, 可以看出加了Contact:Email

## 2.4 删除学号为2015003的学生的选课记录

Hbase不提供根据row key范围删除指定列的操作, 只能一次性删除一行或者删除一个单元格, 所以一个格子一个格子删除:



```

delete 'studentInfoDB','2015003','C_1:SC_Cno'
delete 'studentInfoDB','2015003','C_1:C_Name'
delete 'studentInfoDB','2015003','C_1:C_Credit'
delete 'studentInfoDB','2015003','C_1:SC_Score'
delete 'studentInfoDB','2015003','C_2:SC_Cno'
delete 'studentInfoDB','2015003','C_2:C_Name'
delete 'studentInfoDB','2015003','C_2:C_Credit'
delete 'studentInfoDB','2015003','C_2:SC_Score'

```

```

hbase(main):048:0> delete 'studentInfoDB','2015003','C_1:SC_Cno'
Took 0.0673 seconds
hbase(main):049:0> delete 'studentInfoDB','2015003','C_1:C_Name'
Took 0.0198 seconds
hbase(main):050:0> delete 'studentInfoDB','2015003','C_1:C_Credit'
Took 0.0104 seconds
hbase(main):051:0> delete 'studentInfoDB','2015003','C_1:SC_Score'
Took 0.0230 seconds
hbase(main):052:0> delete 'studentInfoDB','2015003','C_2:SC_Cno'
Took 0.0495 seconds
hbase(main):053:0> delete 'studentInfoDB','2015003','C_2:C_Name'
Took 0.0847 seconds
hbase(main):054:0> delete 'studentInfoDB','2015003','C_2:C_Credit'
Took 0.0415 seconds
hbase(main):055:0> delete 'studentInfoDB','2015003','C_2:SC_Score'
Took 0.0361 seconds
hbase(main):056:0> 

```

```

hbase(main):056:0> scan 'studentInfoDB'
ROW                                COLUMN+CELL
2015001                            column=C_1:C_Credit, timestamp=2021-11-17T06:53:29.273, value=2.0
2015001                            column=C_1:C_Name, timestamp=2021-11-17T06:53:29.009, value=Math
2015001                            column=C_1:SC_Cno, timestamp=2021-11-17T06:53:28.848, value=123001
2015001                            column=C_1:SC_Score, timestamp=2021-11-17T06:53:29.452, value=86
2015001                            column=C_3:C_Credit, timestamp=2021-11-17T06:53:30.439, value=3.0
2015001                            column=C_3:C_Name, timestamp=2021-11-17T06:53:29.889, value=English
2015001                            column=C_3:SC_Cno, timestamp=2021-11-17T06:53:29.633, value=123003
2015001                            column=C_3:SC_Score, timestamp=2021-11-17T06:53:30.581, value=69
2015001                            column=Contact:Email, timestamp=2021-11-17T06:54:29.369, value=lilei@qq.com
2015001                            column=S_info:S_Age, timestamp=2021-11-17T06:52:36.779, value=23
2015001                            column=S_info:S_Name, timestamp=2021-11-17T06:52:36.411, value=Li Lei
2015001                            column=S_info:S_Sex, timestamp=2021-11-17T06:52:36.595, value=male
2015002                            column=C_2:C_Credit, timestamp=2021-11-17T06:53:31.082, value=5.0
2015002                            column=C_2:C_Name, timestamp=2021-11-17T06:53:30.952, value=Computer Science
2015002                            column=C_2:SC_Cno, timestamp=2021-11-17T06:53:30.797, value=123002
2015002                            column=C_2:SC_Score, timestamp=2021-11-17T06:53:31.214, value=77
2015002                            column=C_3:C_Credit, timestamp=2021-11-17T06:53:31.602, value=3.0
2015002                            column=C_3:C_Name, timestamp=2021-11-17T06:53:31.464, value=English
2015002                            column=C_3:SC_Cno, timestamp=2021-11-17T06:53:31.328, value=123003
2015002                            column=C_3:SC_Score, timestamp=2021-11-17T06:53:31.751, value=99
2015002                            column=Contact:Email, timestamp=2021-11-17T06:54:29.580, value=hmm@qq.com
2015002                            column=S_info:S_Age, timestamp=2021-11-17T06:52:37.486, value=22
2015002                            column=S_info:S_Name, timestamp=2021-11-17T06:52:37.101, value=Han Meimei
2015002                            column=S_info:S_Sex, timestamp=2021-11-17T06:52:37.314, value=female
2015003                            column=Contact:Email, timestamp=2021-11-17T06:54:36.354, value=zs@qq.com
2015003                            column=S_info:S_Age, timestamp=2021-11-17T06:52:42.472, value=24
2015003                            column=S_info:S_Name, timestamp=2021-11-17T06:52:37.721, value=Zhang san
2015003                            column=S_info:S_Sex, timestamp=2021-11-17T06:52:37.884, value=male
3 row(s)
Took 0.6022 seconds

```

## 2.5 删除所创建的表

drop的表必须是disable的

```
disable 'studentInfoDB'
drop 'studentInfoDB'
```

```
hbase(main):057:0> disable 'studentInfoDB'
Took 1.3605 seconds
hbase(main):058:0> drop 'studentInfoDB'
Took 0.8048 seconds
hbase(main):059:0> scan 'studentInfoDB'
ROW                                COLUMN+CELL

ERROR: Unknown table studentInfoDB!

For usage try 'help "scan"'

Took 0.0462 seconds
hbase(main):060:0>
```

可以看出确实表已经被删除了。

## 3 Java实现

### 3.1 创建表

函数createTable:

```
public void createTable(String myTableName, String[] colFamily) throws IOException {
    TableName tableName = TableName.valueOf(myTableName);
    if(admin.tableExists(tableName)){
        System.out.println("table "+ myTableName + " exists!");
    } else {
        HTableDescriptor hTableDescriptor = new HTableDescriptor(tableName);
        for(String str:colFamily){
            HColumnDescriptor hColumnDescriptor = new HColumnDescriptor(str);
            hTableDescriptor.addFamily(hColumnDescriptor);
        }
        admin.createTable(hTableDescriptor);
        System.out.println("Successfully create table: "+ myTableName + "!");
    }
}
```

main函数中执行:

```

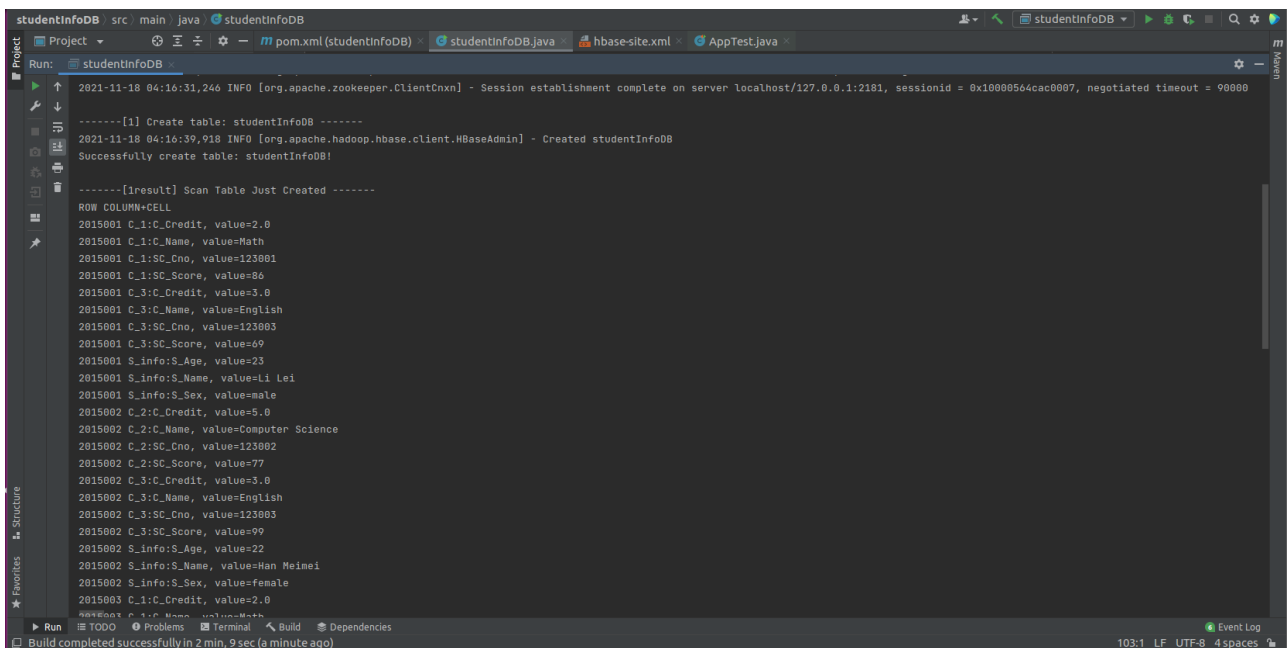
System.out.println("\n-----[1] Create table: studentInfoDB -----");
operation.createTable(myTableName, families);//建表

//添加学生信息
operation.insertData(myTableName, rowKey: "2015001", colFamily: "S_info", col: "S_Name", val: "Li Lei");
operation.insertData(myTableName, rowKey: "2015001", colFamily: "S_info", col: "S_Sex", val: "male");
operation.insertData(myTableName, rowKey: "2015001", colFamily: "S_info", col: "S_Age", val: "23");
operation.insertData(myTableName, rowKey: "2015002", colFamily: "S_info", col: "S_Name", val: "Han Meimei");
operation.insertData(myTableName, rowKey: "2015002", colFamily: "S_info", col: "S_Sex", val: "female");
operation.insertData(myTableName, rowKey: "2015002", colFamily: "S_info", col: "S_Age", val: "22");
operation.insertData(myTableName, rowKey: "2015003", colFamily: "S_info", col: "S_Name", val: "Li Lei");
operation.insertData(myTableName, rowKey: "2015003", colFamily: "S_info", col: "S_Sex", val: "male");
operation.insertData(myTableName, rowKey: "2015003", colFamily: "S_info", col: "S_Age", val: "24");

//添加选课信息
operation.insertData(myTableName, rowKey: "2015001", colFamily: "C_1", col: "C_Credit", val: "2.0");

```

执行结果:



```

2021-11-18 04:16:31,246 INFO [org.apache.zookeeper.ClientCnxn] - Session establishment complete on server localhost/127.0.0.1:2181, sessionId = 0x10000564cac0007, negotiated timeout = 90000
-----[1] Create table: studentInfoDB -----
2021-11-18 04:16:39,918 INFO [org.apache.hadoop.hbase.client.HBaseAdmin] - Created studentInfoDB
Successfully create table: studentInfoDB!
-----[result] Scan Table Just Created -----
ROW COLUMN+CELL
2015001 C_1:C_Credit, value=2.0
2015001 C_1:C_Name, value=Math
2015001 C_1:SC_Cno, value=123001
2015001 C_1:SC_Score, value=86
2015001 C_3:C_Credit, value=3.0
2015001 C_3:C_Name, value=English
2015001 C_3:SC_Cno, value=123003
2015001 C_3:SC_Score, value=69
2015001 S_info:S_Age, value=23
2015001 S_info:S_Name, value=Li Lei
2015001 S_info:S_Sex, value=male
2015002 C_2:C_Credit, value=5.0
2015002 C_2:C_Name, value=Computer Science
2015002 C_2:SC_Cno, value=123002
2015002 C_2:SC_Score, value=77
2015002 C_3:C_Credit, value=3.0
2015002 C_3:C_Name, value=English
2015002 C_3:SC_Cno, value=123003
2015002 C_3:SC_Score, value=99
2015002 S_info:S_Age, value=22
2015002 S_info:S_Name, value=Han Meimei
2015002 S_info:S_Sex, value=female
2015003 C_1:C_Credit, value=2.0
2015003 C_1:C_Name, value=Math
2015003 C_1:SC_Cno, value=123001
2015003 C_1:SC_Score, value=86
2015003 C_3:C_Credit, value=3.0
2015003 C_3:C_Name, value=English
2015003 C_3:SC_Cno, value=123003
2015003 C_3:SC_Score, value=69
2015003 S_info:S_Age, value=24
2015003 S_info:S_Name, value=Li Lei
2015003 S_info:S_Sex, value=male

```

## 3.2 查询选修Computer Science的学生的成绩

定义函数 `scanTableByColumn`，用于查询某一列下的信息：

```

//按照列来扫描
public void scanTableByColumn(String myTableName, String colFamily, String col) throws IOException {
    Table table=connection.getTable(TableName.valueOf(myTableName));

    ResultScanner scanResult = table.getScanner(colFamily.getBytes(), col.getBytes()); //getScanner方法中设
    System.out.println("ROW\tCOLUMN+CELL");
    for (Result result : scanResult) {
        String row = new String(result.getRow());
        List<Cell> cells = result.listCells();
        for (Cell c:cells) {
            System.out.println(row+"\t"+new String(CellUtil.cloneFamily(c))+":"+
                new String(CellUtil.cloneQualifier(c))+", value="+new String(CellUtil.cloneValue(c)))
        }
    }
}

```

main函数中:

```
System.out.println("\n-----[2] Query the score of students select course Computer Science -----");
operation.scanTableByColumn(myTableName, colFamily: "C_2", col: "SC_Score"); //scan C_2:SC_Score
```

执行结果:

```
-----[2] Query the score of students select course Computer Science -----
ROW COLUMN+CELL
2015002 C_2:SC_Score, value=77
2015003 C_2:SC_Score, value=95
```

### 3.3 增加新的列族和新列Contact:Email, 并添加数据

先用 `addFamily` 加入新的列族 `contact`, 然后插入 `email` 的 `value`。

```
//添加新的列族
public void addFamily(String myTableName, String colFamily) throws IOException {
    HTableDescriptor tableDescriptor = admin.getTableDescriptor(TableName.valueOf(myTableName)); //获得原来表的定义信息
    HColumnDescriptor nColumnDescriptor = new HColumnDescriptor(colFamily); //define a column family
    tableDescriptor.addFamily(nColumnDescriptor); //add new column family into table
    admin.modifyTable(TableName.valueOf(myTableName), tableDescriptor); //commit it to admin
    System.out.println("Add column family: "+colFamily+" successfully!");
}

/**
 * 插入一行数据
 * @param myTableName
 * @param rowKey
 * @param colFamily
 * @param col
 * @param val
 * @throws IOException
 */
public void insertData(String myTableName, String rowKey, String colFamily, String col, String val) throws IOException {
    Table table = connection.getTable(TableName.valueOf(myTableName));
    Put put = new Put(rowKey.getBytes());
    put.addColumn(colFamily.getBytes(), col.getBytes(), val.getBytes());
    table.put(put);
    table.close();
}
```

main函数中:

```
//添加Contact:Email列的信息
System.out.println("\n-----[3] Add Contact:Email -----");
operation.addFamily(myTableName, colFamily: "Contact");
operation.insertData(myTableName, rowKey: "2015001", colFamily: "Contact", col: "Email", val: "lilei@qq.com");
operation.insertData(myTableName, rowKey: "2015002", colFamily: "Contact", col: "Email", val: "hmm@qq.com");
operation.insertData(myTableName, rowKey: "2015003", colFamily: "Contact", col: "Email", val: "zs@qq.com");

System.out.println("\n-----[3result] Scan Contact:Email Just Modified -----");
operation.scanTableByColumn(myTableName, colFamily: "Contact", col: "Email"); //扫描Contact:Email列
```

执行结果:

```

-----[3] Add Contact:Email -----
Add column family: Contact successfully!

-----[3result] Scan Contact:Email Just Modified -----
ROW COLUMN+CELL
2015001 Contact:Email, value=lilei@qq.com
2015002 Contact:Email, value=hmm@qq.com
2015003 Contact:Email, value=zs@qq.com

```

### 3.4 删除学号为2015003的学生的选课记录

写了删除cell的函数：

```

public void deleteByCell(String myTableName, String rowKey, String colFamily, String col) throws IOException {
    Table table = connection.getTable(TableName.valueOf(myTableName));
    Delete delete = new Delete(Bytes.toBytes(rowKey));
    //删除指定列
    delete.addColumn(Bytes.toBytes(colFamily), Bytes.toBytes(col));
    table.delete(delete);
}

```

在main函数中调用：

```

System.out.println("\n-----[4] Delete Student 2015003 Course Select Information -----");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_1", col: "SC_Cno");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_1", col: "C_Name");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_1", col: "C_Credit");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_1", col: "SC_Score");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_2", col: "SC_Cno");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_2", col: "C_Name");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_2", col: "C_Credit");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_2", col: "SC_Score");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_3", col: "SC_Cno");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_3", col: "C_Name");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_3", col: "C_Credit");
operation.deleteByCell(myTableName, rowKey: "2015003", colFamily: "C_3", col: "SC_Score");

System.out.println("\n-----[4result] Scan Table Just Modified -----");
operation.scanTable(myTableName);

```

可以看到，关于2015003的选课信息确实已经删除，但是2015003同学的基本信息还在：

```
-----[4] Delete Student 2015003 Course Select Information -----  
  
-----[4result] Scan Table Just Modified -----  
ROW COLUMN+CELL  
2015001 C_1:C_Credit, value=2.0  
2015001 C_1:C_Name, value=Math  
2015001 C_1:SC_Cno, value=123001  
2015001 C_1:SC_Score, value=86  
2015001 C_3:C_Credit, value=3.0  
2015001 C_3:C_Name, value=English  
2015001 C_3:SC_Cno, value=123003  
2015001 C_3:SC_Score, value=69  
2015001 Contact:Email, value=lilei@qq.com  
2015001 S_info:S_Age, value=23  
2015001 S_info:S_Name, value=Li Lei  
2015001 S_info:S_Sex, value=male  
2015002 C_2:C_Credit, value=5.0  
2015002 C_2:C_Name, value=Computer Science  
2015002 C_2:SC_Cno, value=123002  
2015002 C_2:SC_Score, value=77  
2015002 C_3:C_Credit, value=3.0  
2015002 C_3:C_Name, value=English  
2015002 C_3:SC_Cno, value=123003  
2015002 C_3:SC_Score, value=99  
2015002 Contact:Email, value=hmm@qq.com  
2015002 S_info:S_Age, value=22  
2015002 S_info:S_Name, value=Han Meimei  
2015002 S_info:S_Sex, value=female  
2015003 Contact:Email, value=zs@qq.com  
2015003 S_info:S_Age, value=24  
2015003 S_info:S_Name, value=Li Lei
```

### 3.5 删除所创建的表

先看看目前有的表，然后删除，再看看还有什么：

```
//删除表  
public void dropTable(String myTableName) throws IOException {  
    if (admin.tableExists(tableName.valueOf(myTableName))) {  
        //如果表存在，则先disable表，然后才能删除表  
        admin.disableTable(tableName.valueOf(myTableName));  
        admin.deleteTable(tableName.valueOf(myTableName));  
        System.out.println("Drop table "+myTableName+" successfully!");  
    } else {  
        //如果表不存在，输出提示  
        System.out.println("There is no table "+myTableName);  
    }  
}
```



```

System.out.println("\n-----[5] Drop table StudentInfoDB -----");
operation.listTables(); //查询现在所有的表
operation.dropTable(myTableName); //删除表
operation.listTables(); //查询所有表，验证删除是否成功

//关闭连接
operation.close();

```

执行结果：删除成功，正常退出程序！

```

-----[5] Drop table studentInfoDB -----
Tables:
studentInfoDB
2021-11-18 04:42:10,995 INFO [org.apache.hadoop.hbase.client.HBaseAdmin] - Started disable of studentInfoDB
2021-11-18 04:42:11,647 INFO [org.apache.hadoop.hbase.client.HBaseAdmin] - Disabled studentInfoDB
2021-11-18 04:42:12,316 INFO [org.apache.hadoop.hbase.client.HBaseAdmin] - Deleted studentInfoDB
Drop table studentInfoDB successfully!
no table.
2021-11-18 04:42:12,326 INFO [org.apache.hadoop.hbase.client.ConnectionManager$HConnectionImplementation] - Closing master protocol: MasterService
2021-11-18 04:42:12,347 INFO [org.apache.hadoop.hbase.client.ConnectionManager$HConnectionImplementation] - Closing zookeeper sessionId=0x10000564cac000a
2021-11-18 04:42:12,355 INFO [org.apache.zookeeper.ClientCnxn] - EventThread shut down
2021-11-18 04:42:12,355 INFO [org.apache.zookeeper.ZooKeeper] - Session: 0x10000564cac000a closed

Process finished with exit code 0

```

## 4 问题总结及解决方案

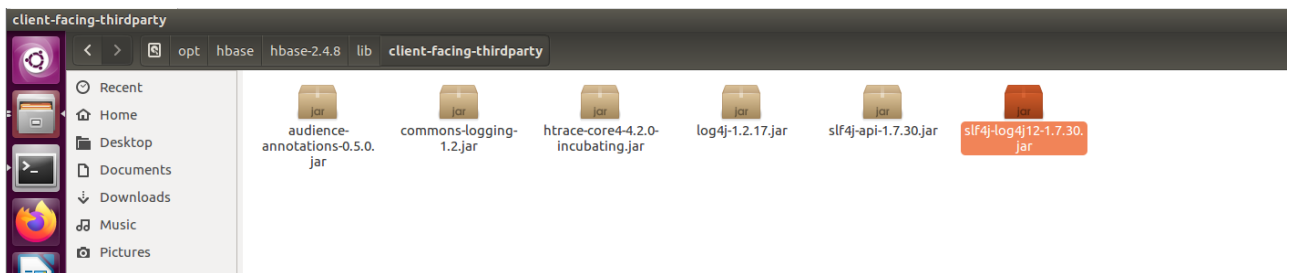
### 4.1 发生jar包冲突

```

ztx@191840376:~$ start-hbase.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hadoop-installs/hadoop-3.2.2/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase/hbase-2.4.8/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
running master, logging to /opt/hbase/hbase-2.4.8/logs/hbase-ztx-master-191840376.out
ztx@191840376:~$ jps
43857 DataNode
44311 ResourceManager
43720 NameNode
45176 Jps
44442 NodeManager
44063 SecondaryNameNode
ztx@191840376:~$

```

发生jar包冲突了，分别为图中的两个路径，移除其中一个jar包即可。



正确执行结果如下：



```
ztx@191840376:~$ start-hbase.sh
running master, logging to /opt/hbase/hbase-2.4.8/logs/hbase-ztx-master-191840376.out
ztx@191840376:~$ jps
43857 DataNode
45522 HMaster
45685 Jps
44311 ResourceManager
43720 NameNode
44442 NodeManager
44063 SecondaryNameNode
```

## 4.2 HMaster执行几秒之后就消失

先是在网上搜【HMaster出现几秒就消失】怎么回事，然后发现都没用。怀疑版本适配问题，甚至换了Hbase的版本。最后想起来能看日志，日志说是这个bug:

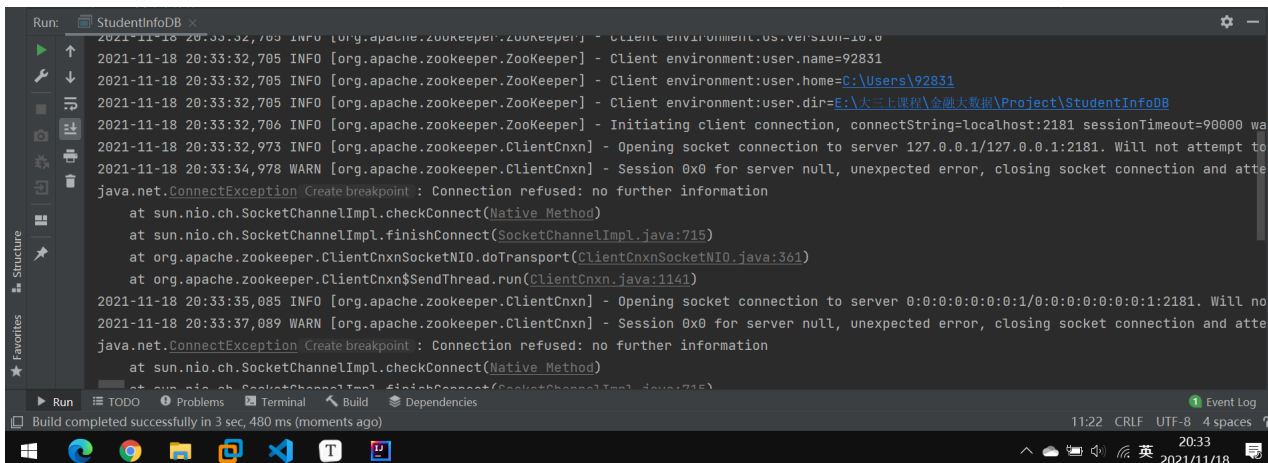
```
2021-11-17 12:12:53,188 ERROR [main] regionserver.HRegionServer: Failed construction RegionServer
java.lang.UnsupportedOperationException: Constructor threw an exception for org.apache.hadoop.hbase.ipc.NettyRpcServer
    at org.apache.hadoop.hbase.util.ReflectionUtils.instantiate(ReflectionUtils.java:66)
    at org.apache.hadoop.hbase.util.ReflectionUtils.instantiateWithCustomCtor(ReflectionUtils.java:45)
    at org.apache.hadoop.hbase.ipc.RpcServerFactory.createRpcServer(RpcServerFactory.java:66)
    at org.apache.hadoop.hbase.master.MasterRpcServices.createRpcServer(MasterRpcServices.java:416)
    at org.apache.hadoop.hbase.regionserver.RSRpcServices.<init>(RSRpcServices.java:1294)
    at org.apache.hadoop.hbase.regionserver.RSRpcServices.<init>(RSRpcServices.java:1245)
    at org.apache.hadoop.hbase.master.MasterRpcServices.<init>(MasterRpcServices.java:393)
    at org.apache.hadoop.hbase.master.HMaster.createRpcServices(HMaster.java:764)
    at org.apache.hadoop.hbase.regionserver.HRegionServer.<init>(HRegionServer.java:603)
    at org.apache.hadoop.hbase.master.HMaster.<init>(HMaster.java:528)
    at org.apache.hadoop.hbase.master.HMasterCommandLine$LocalHMaster.<init>(HMasterCommandLine.java:325)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
    at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
    at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
    at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
    at org.apache.hadoop.hbase.util.JVMClusterUtil.createMasterThread(JVMClusterUtil.java:132)
    at org.apache.hadoop.hbase.LocalHBaseCluster.addMaster(LocalHBaseCluster.java:239)
    at org.apache.hadoop.hbase.LocalHBaseCluster.<init>(LocalHBaseCluster.java:181)
    at org.apache.hadoop.hbase.LocalHBaseCluster.<init>(LocalHBaseCluster.java:128)
    at org.apache.hadoop.hbase.master.HMasterCommandLine.startMaster(HMasterCommandLine.java:246)
    at org.apache.hadoop.hbase.master.HMasterCommandLine.run(HMasterCommandLine.java:149)
```

反正在网上搜了一圈（包括Hbase官网，StackOverflow），没有在单机情况下就被打倒的情况...实在没办法了，最终选择👉

解决方案：全部重开，**remake!**

新建了一个虚拟机，从Hadoop开始配置（又走了一遍实验二的流程），重装过程中以前遇到的问题/这次遇到的问题，统统没遇到！有时候直接重开比死磕更好哈（落泪）

## 4.3 windows下intellij maven用javaAPI写hbase程序编译出错



报错，应该是zookeeper没有打开的缘故，但是在windows系统下真的不知道该怎么办了...

网上的教程又多是Windows下IntelliJ IDEA用maven布置真实集群，用javaAPI完成hbase调度。而集群又很麻烦，所以选择在Linux下安装IntelliJ完成实验。

在Linux下完成实验的时候也遇到了上述的问题，发现是重启虚拟机之后忘记用 `start-hbase.sh` 打开伪分布的hbase所致。还有用了老师bdkit上的demo完善了 `pom.xml` 的配置。

## 5 其它思考

(1) 表的设计感觉冗余比较大，列族C\_1,2,3下的课程名学分之类的全部都一样，还有改进的空间。

(2) 在做删除【按行的范围删除某几列值】这个操作的时候，发现hbase没有提供这种操作。在创建表的时候，**hbase**只关注**rowkey**，**column Family**，并没有说在创建表的时候指定列族限定有多少，这也是**hbase**列式存储的特点，所以在**hbase API**中是没有提供上述操作

## 6 文件说明

studentInfoDB文件夹完成用java完成hbase表的构建，操作。