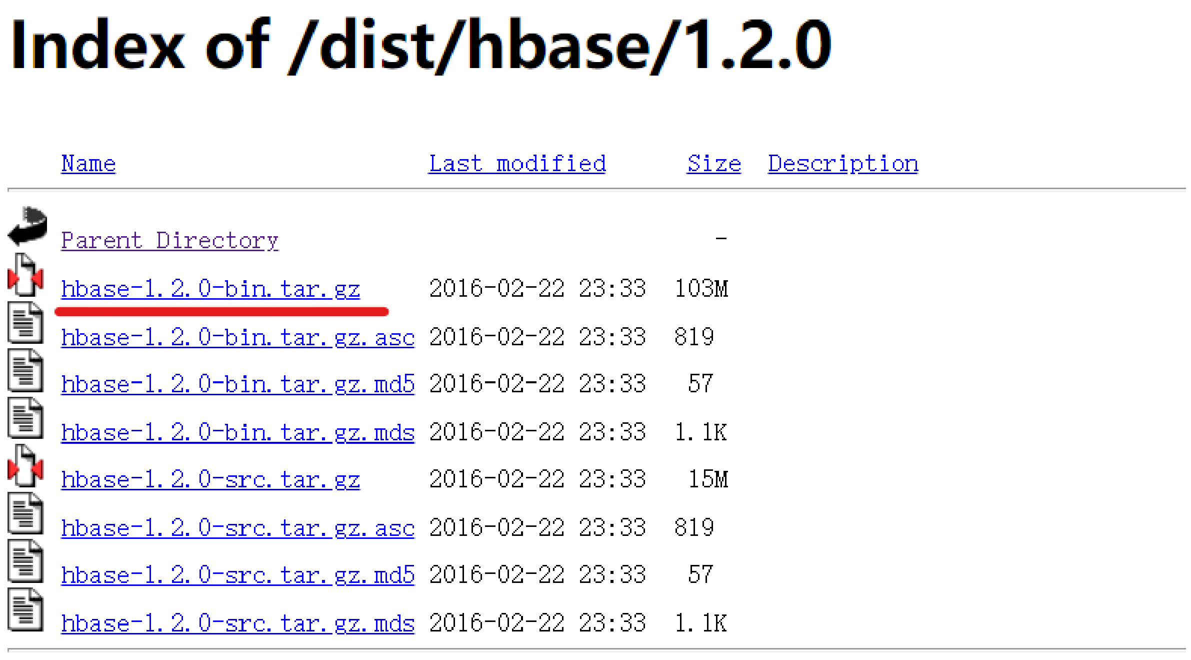
金融大数据-实验3

施宇 191250119

Windows下搭建伪分布式HBase环境

安装HBase

到官网下载HBase，这里选用的是1.2.0版本：



Index of /dist/hbase/1.2.0

Name	Last modified	Size	Description
Parent Directory		-	
hbase-1.2.0-bin.tar.gz	2016-02-22 23:33	103M	
hbase-1.2.0-bin.tar.gz.asc	2016-02-22 23:33	819	
hbase-1.2.0-bin.tar.gz.md5	2016-02-22 23:33	57	
hbase-1.2.0-bin.tar.gz.mds	2016-02-22 23:33	1.1K	
hbase-1.2.0-src.tar.gz	2016-02-22 23:33	15M	
hbase-1.2.0-src.tar.gz.asc	2016-02-22 23:33	819	
hbase-1.2.0-src.tar.gz.md5	2016-02-22 23:33	57	
hbase-1.2.0-src.tar.gz.mds	2016-02-22 23:33	1.1K	

下载完成后解压。打开hbase-1.2.0/conf文件夹，找到hbase-env.cmd文件，右键选择编辑，添加以下内容：

```
set HBASE_MANAGES_ZK=false
set JAVA_HOME=D:\Hadoop\Java7
set HBASE_CLASSPATH=D:\Hadoop\HBase\hbase-1.2.0\conf
```

其中`JAVA_HOME`和`HBASE_CLASSPATH`根据自己计算机上的Java和HBase路径进行配置。

找到hbase-site.xml文件，将内容替换为如下：

```
<configuration>
    <property>
        <name>hbase.rootdir</name>
        <value>file:///D:/Hadoop/HBase/hbase-1.2.0/root</value>
    </property>
    <property>
        <name>hbase.tmp.dir</name>
        <value>D:/Hadoop/HBase/hbase-1.2.0/tem</value>
    </property>
    <property>
        <name>hbase.zookeeper.quorum</name>
        <value>127.0.0.1</value>
    </property>
    <property>
        <name>hbase.zookeeper.property.dataDir</name>
        <value>D:/Hadoop/HBase/hbase-1.2.0/zoo</value>
    </property>
    <property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
     </property>
</configuration>
```

其中的路径也根据本机HBase路径进行配置。

启动HBase

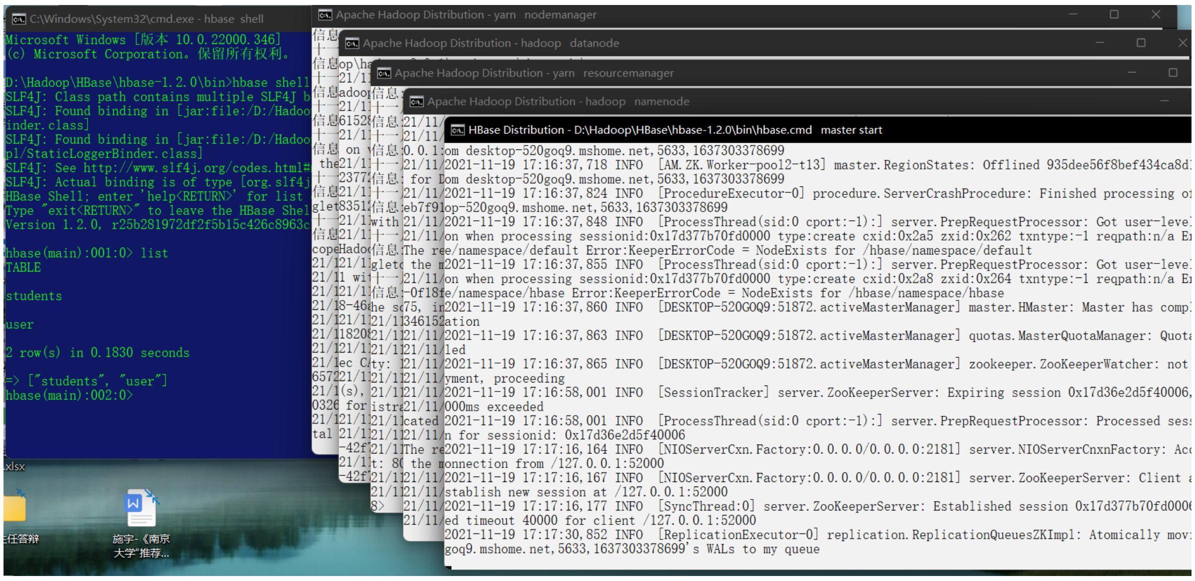
启动Hadoop后，进入hbase-1.2.0/bin目录下，启动start-hbase.cmd。在cmd中到hbase-1.2.0/bin目录下，启动hbase shell：

```
hbase shell
```

出现

```
hbase(main):001.0>
```

说明搭建成功，Hadoop及HBase运行成功截图：


C:\Windows\System32\cmd.exe - hbase shell
Apache Hadoop Distribution - yarn nodemanager
Apache Hadoop Distribution - hadoop datanode
Apache Hadoop Distribution - yarn resourcemanager
Apache Hadoop Distribution - hadoop namenode
HBase Distribution - D:\Hadoop\HBase\hbase-1.2.0\bin\hbase.cmd master start
.xlsx
任答辩
施宇-《南京
大学"推荐...

```
Microsoft Windows [版本 10.0.22000.346]
(c) Microsoft Corporation。保留所有权利。

D:\Hadoop\HBase\hbase-1.2.0\bin>hbase shell
SLF4J: Class path contains multiple SLF4J b
SLF4J: Found binding in [jar:file:/D:/Hadoo
inder.class]
SLF4J: Found binding in [jar:file:/D:/Hadoo
pl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#
SLF4J: Actual binding is of type [org.slf4j
HBase Shell; enter 'help<RETURN>' for list
Type "exit<RETURN>" to leave the HBase Shel
Version 1.2.0, r25b281972df2f5b15c426c8963c

hbase(main):001:0> list
TABLE

students

user

2 row(s) in 0.1830 seconds

=> ["students", "user"]
hbase(main):002:0>
```

```
om desktop-520goq9.mshome.net,5633,1637303378699
2021-11-19 17:16:37,718 INFO  [AM.ZK.Worker-pool2-t13] master.RegionStates: Offlined 935dee56f8bef434ca8d
om desktop-520goq9.mshome.net,5633,1637303378699
2021-11-19 17:16:37,824 INFO  [ProcedureExecutor-0] procedure.ServerCrashProcedure: Finished processing of
op-520goq9.mshome.net,5633,1637303378699
2021-11-19 17:16:37,848 INFO  [ProcessThread(sid:0 cport:-1):] server.PrepRequestProcessor: Got user-leve
on when processing sessionid:0x17d377b70fd0000 type:create cxid:0x2a5 zxid:0x262 txntype:-1 reqpath:n/a E
e/namespace/default Error:KeeperErrorCode = NodeExists for /hbase/namespace/default
2021-11-19 17:16:37,855 INFO  [ProcessThread(sid:0 cport:-1):] server.PrepRequestProcessor: Got user-leve
on when processing sessionid:0x17d377b70fd0000 type:create cxid:0x2a8 zxid:0x264 txntype:-1 reqpath:n/a E
e/namespace/hbase Error:KeeperErrorCode = NodeExists for /hbase/namespace/hbase
2021-11-19 17:16:37,860 INFO  [DESKTOP-520GOQ9:51872.activeMasterManager] master.HMaster: Master has comp
ation
2021-11-19 17:16:37,863 INFO  [DESKTOP-520GOQ9:51872.activeMasterManager] quotas.MasterQuotaManager: Quot
led
2021-11-19 17:16:37,865 INFO  [DESKTOP-520GOQ9:51872.activeMasterManager] zookeeper.ZooKeeperWatcher: not
yment, proceeding
2021-11-19 17:16:58,001 INFO  [SessionTracker] server.ZooKeeperServer: Expiring session 0x17d36e2d5f40006,
000ms exceeded
2021-11-19 17:16:58,001 INFO  [ProcessThread(sid:0 cport:-1):] server.PrepRequestProcessor: Processed ses
n for sessionid: 0x17d36e2d5f40006
2021-11-19 17:17:16,164 INFO  [NIOServerCxn.Factory:0.0.0.0/0.0.0.0:2181] server.NIOServerCnxnFactory: Ac
onnection from /127.0.0.1:52000
2021-11-19 17:17:16,167 INFO  [NIOServerCxn.Factory:0.0.0.0/0.0.0.0:2181] server.ZooKeeperServer: Client
stablish new session at /127.0.0.1:52000
2021-11-19 17:17:16,177 INFO  [SyncThread:0] server.ZooKeeperServer: Established session 0x17d377b70fd000
ed timeout 40000 for client /127.0.0.1:52000
2021-11-19 17:17:30,852 INFO  [ReplicationExecutor-0] replication.ReplicationQueuesZKImpl: Atomically mov
goq9.mshome.net,5633,1637303378699's WALs to my queue
```

配置java操作HBase项目依赖

在原来Hadoop的java项目的基础上，修改pom.xml文件，添加如下依赖：

```xml
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-it</artifactId>
    <version>1.2.0</version>
</dependency>
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-client</artifactId>
    <version>1.2.0</version>
</dependency>
```

编写Java程序完成下列任务

设计并创建合适的表

原始数据如下：

学生(student)

学号(S_No)	姓名(S_Name)	性别(S_Sex)	年龄(S_Age)
2015001	Li Lei	male	23
2015002	Han Meimei	female	22
2015003	Zhang San	male	24

课程(course)

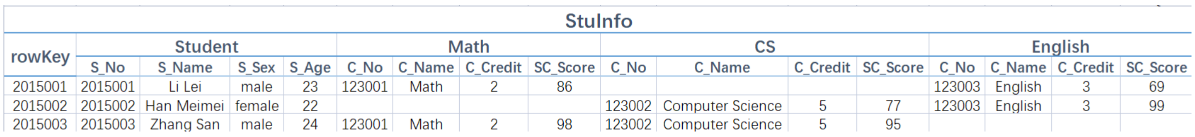
课程号(C_No)	课程名(C_Name)	学分(C_Credit)
123001	Math	2.0
123002	Computer Science	5.0
123003	English	3.0

选课(sc)

学号(SC_Sno)	课程号(SC_Cno)	成绩(SC_Score)
2015001	123001	86
2015001	123003	69
2015002	123002	77
2015002	123003	99
2015003	123001	98
2015003	123002	95

存储至HBase中：原始数据的三张表可以转化为HBase中的一张表进行存储。

StuInfo表：



StuInfo																
rowKey	Student				Math				CS				English			
	S_No	S_Name	S_Sex	S_Age	C_No	C_Name	C_Credit	SC_Score	C_No	C_Name	C_Credit	SC_Score	C_No	C_Name	C_Credit	SC_Score
2015001	2015001	Li Lei	male	23	123001	Math	2	86					123003	English	3	69
2015002	2015002	Han Meimei	female	22					123002	Computer Science	5	77	123003	English	3	99
2015003	2015003	Zhang San	male	24	123001	Math	2	98	123002	Computer Science	5	95				

其中学生信息、数学课信息、计算机课信息和英语课信息分别存储在Student、Math、CS和English三个列族中，使用学号作为rowKey。这样可以从一张表中查询到学生信息、课程信息和学生选课及成绩信息。

通过Java程序创建表

为了提高代码的复用性，创建HBaseOperator类，在该类中实现常用的HBase操作的函数，再通过调用这些函数完成后续任务。HBaseOperator类的所有方法都为静态方法，刚开始先静态初始化，完成HBase的连接操作。

```java
public class HBaseOperator {

    private static Configuration conf = null;
    private static Connection conn = null;
    private static Admin admin = null;
    public static AtomicInteger count = new AtomicInteger();
    static {
        conf = HBaseConfiguration.create();
        conf.set("hbase.zookeeper.quorum", "10.148.137.143");
        conf.set("hbase.zookeeper.property.clientPort", "2181");
    }

    static {
        try {
            conn = ConnectionFactory.createConnection();
            admin = conn.getAdmin();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

新建表格方法：

在确认要新建的表格不存在后，通过HTableDescriptor配置表格信息，增加列族，然后通过createTable新建表格：

```java
public static void createTable(String tablename, String[] cfs){
    try{
        if(admin.tableExists(TableName.valueOf(tablename))){
            System.out.println("Table already exists!");
        }else{
            HTableDescriptor tableDesc = new
HTableDescriptor(TableName.valueOf(tablename));
            for(int i=0; i<cfs.length; i++){
                HColumnDescriptor desc = new HColumnDescriptor(cfs[i]);
                tableDesc.addFamily(desc);
            }
            admin.createTable(tableDesc);
            System.out.println("Create table: " + tablename + " ... Done.");
        }
    }catch (IOException e){
        e.printStackTrace();
    }
}
```

插入数据方法:

先通过getTable方法得到要插入的表格的Table对象，然后通过Put对象配置要插入的数据的rowKey，列族和列标签及值的信息，最后通过put方法插入数据。

```
public static void addData(String tableName, String rowKey, String family,
String qualifier, String value){
    try{
        Table table = conn.getTable(TableName.valueOf(tableName));
        Put put = new Put(Bytes.toBytes(rowKey));
        put.add(Bytes.toBytes(family), Bytes.toBytes(qualifier),
Bytes.toBytes(value));
        table.put(put);
        System.out.println("Intert record ... Done.");
    }catch (IOException e){
        e.printStackTrace();
    }
}
```

通过以上两个方法，可以完成表格的建立:

```
// 创建表
HBaseOperator.createTable("StuInfo", new String[]{"Student", "Math", "CS",
"English"});
// 添加学生信息
HBaseOperator.addData("StuInfo", "2015001", "Student", "S_No", "2015001");
HBaseOperator.addData("StuInfo", "2015001", "Student", "S_Name", "Li Lei");
HBaseOperator.addData("StuInfo", "2015001", "Student", "S_Sex", "male");
HBaseOperator.addData("StuInfo", "2015001", "Student", "S_Age", "23");

HBaseOperator.addData("StuInfo", "2015002", "Student", "S_No", "2015002");
HBaseOperator.addData("StuInfo", "2015002", "Student", "S_Name", "Han Meimei");
HBaseOperator.addData("StuInfo", "2015002", "Student", "S_Sex", "female");
HBaseOperator.addData("StuInfo", "2015002", "Student", "S_Age", "22");

HBaseOperator.addData("StuInfo", "2015003", "Student", "S_No", "2015003");
HBaseOperator.addData("StuInfo", "2015003", "Student", "S_Name", "Zhang San");
HBaseOperator.addData("StuInfo", "2015003", "Student", "S_Sex", "male");
HBaseOperator.addData("StuInfo", "2015003", "Student", "S_Age", "24");

// 添加课程信息
HBaseOperator.addData("StuInfo", "2015001", "Math", "C_No", "123001");
HBaseOperator.addData("StuInfo", "2015001", "Math", "C_Name", "Math");
HBaseOperator.addData("StuInfo", "2015001", "Math", "C_Credit", "2");
HBaseOperator.addData("StuInfo", "2015001", "English", "C_No", "123003");
HBaseOperator.addData("StuInfo", "2015001", "English", "C_Name", "English");
HBaseOperator.addData("StuInfo", "2015001", "English", "C_Credit", "3");

HBaseOperator.addData("StuInfo", "2015002", "CS", "C_No", "123002");
HBaseOperator.addData("StuInfo", "2015002", "CS", "C_Name", "Computer Science");
HBaseOperator.addData("StuInfo", "2015002", "CS", "C_Credit", "5");
HBaseOperator.addData("StuInfo", "2015002", "English", "C_No", "123003");
HBaseOperator.addData("StuInfo", "2015002", "English", "C_Name", "English");
HBaseOperator.addData("StuInfo", "2015002", "English", "C_Credit", "3");

HBaseOperator.addData("StuInfo", "2015003", "Math", "C_No", "123001");
HBaseOperator.addData("StuInfo", "2015003", "Math", "C_Name", "Math");
HBaseOperator.addData("StuInfo", "2015003", "Math", "C_Credit", "2");
HBaseOperator.addData("StuInfo", "2015003", "CS", "C_No", "123002");
HBaseOperator.addData("StuInfo", "2015003", "CS", "C_Name", "Computer Science");
HBaseOperator.addData("StuInfo", "2015003", "CS", "C_Credit", "5");

// 添加成绩信息
HBaseOperator.addData("StuInfo", "2015001", "Math", "SC_Score", "86");
HBaseOperator.addData("StuInfo", "2015001", "English", "SC_Score", "69");

HBaseOperator.addData("StuInfo", "2015002", "CS", "SC_Score", "77");
HBaseOperator.addData("StuInfo", "2015002", "English", "SC_Score", "99");

HBaseOperator.addData("StuInfo", "2015003", "Math", "SC_Score", "98");
HBaseOperator.addData("StuInfo", "2015003", "CS", "SC_Score", "95");
```

查询选修Computer Science的学生的成绩

查询某一列为某个值的数据可以通过Filter过滤器完成，因此实现根据Filter的表格查询方法：

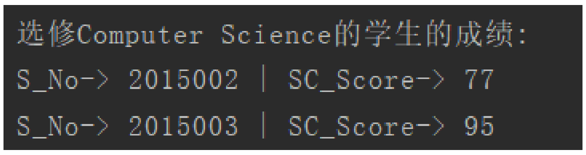
通过SingleColumnValueFilter配置单列按值扫描，然后将配置好的Scan类传入Table中，扫描得到ResulterScanner，提取出其中的KeyValue作为函数的返回值返回即可。

```
public static List<KeyValue> getByFilter(String tableName, List<String> arr){
    List<KeyValue> res = new ArrayList<KeyValue>();
    try{
        Table table = conn.getTable(TableName.valueOf(tableName));
        FilterList filterList = new FilterList();
        Scan s1 = new Scan();
        for(String v: arr){
            String[] s = v.split(",");
            filterList.addFilter(new
SingleColumnValueFilter(Bytes.toBytes(s[0]), Bytes.toBytes(s[1]),

CompareFilter.CompareOp.EQUAL, Bytes.toBytes(s[2])));
            //                s1.addColumn(Bytes.toBytes(s[0]),
Bytes.toBytes(s[1]));
        }
        s1.setFilter(filterList);
        ResultScanner ResultScannerFilterList = table.getScanner(s1);
        for(Result rr = ResultScannerFilterList.next(); rr != null; rr =
ResultScannerFilterList.next()){
            for(KeyValue kv: rr.list()){
                res.add(kv);
                //                System.out.println("row-> " + new
String(kv.getRow()));
                //                System.out.println("family:column-> " + new
String(kv.getFamily()) + " : " + new String(kv.getQualifier()));
                //                System.out.println("value-> " + new
String(kv.getValue()));
            }
        }
    }catch (IOException e){
        e.printStackTrace();
    }
    return res;
}
```

通过该函数得到所有CS:C_Name列的值为Computer Science的行以后，还需要通过训练提取出其中CS:SC_Score列的值，即考试成绩：

```
// 查询选修Computer Science的学生的成绩
System.out.println("选修Computer Science的学生的成绩:");
List<String> arr = new ArrayList<String>();
arr.add("CS,C_Name,Computer Science");
List<KeyValue> res = HBaseOperator.getByFilter("StuInfo", arr);
for(KeyValue kv: res){
    if(new String(kv.getFamily()).equals("CS") && new
String(kv.getQualifier()).equals("SC_Score")){
        System.out.print("S_No-> " + new String(kv.getRow()));
        System.out.println(" | SC_Score-> " + new String(kv.getValue()));
    }
}
```

查询结果为：



```
选修Computer Science的学生的成绩:
S_No-> 2015002 | SC_Score-> 77
S_No-> 2015003 | SC_Score-> 95
```

增加新的列族和新列Contact:Email，并添加数据

增加新列可以直接通过addData，即插入数据的方法增加，而增加新的列族需要修改表的结构，这里实现增加新列族的方法：

通过HColumnDescriptor设定列族名称，然后通过addColumn方法添加列族。

```
public static void addColumn(String tableName, String columnName){
    try{
        admin.disableTable(TableName.valueOf(tableName));
        HTableDescriptor desc =
admin.getTableDescriptor(TableName.valueOf(tableName));
        HColumnDescriptor cdesc = new HColumnDescriptor(columnName);
        desc.addFamily(cdesc);
        admin.addColumn(TableName.valueOf(tableName), cdesc);
        admin.enableTableAsync(TableName.valueOf(tableName));
    }catch (IOException e){
        e.printStackTrace();
    }
}
```

添加列族和新列，并加添数据的过程如下：

```
// 增加新的列族和新列Contact:Email，并添加数据
HBaseOperator.addColumn("StuInfo", "Contact");
HBaseOperator.addData("StuInfo", "2015001", "Contact", "Email", "lilie@qq.com");
HBaseOperator.addData("StuInfo", "2015002", "Contact", "Email", "hmm@qq.com");
HBaseOperator.addData("StuInfo", "2015003", "Contact", "Email", "zs@qq.com");
```

删除学号为2015003的学生的选课记录

删除指定行的指定列族可以通过deleteFamily方法实现：

```
public static void delOneRecordFamily(String tableName, String rowKey, String
family){
    try{
        Table table = conn.getTable(TableName.valueOf(tableName));
        List<Delete> list = new ArrayList<Delete>();
        Delete del = new Delete(rowKey.getBytes());
        del.deleteFamily(Bytes.toBytes(family));
        list.add(del);
        table.delete(list);
        System.out.println("Del record:" + rowKey + "-" + family + " ...
Done.");
    }catch (IOException e){
        e.printStackTrace();
    }
}
```

删除学号为2015003的学生的选课记录即删除rowKey=2015003的行的Math、CS、English三个列族：

```
// 删除学号为2015003的学生的选课记录
HBaseOperator.delOneRecordFamily("StuInfo", "2015003", "Math");
HBaseOperator.delOneRecordFamily("StuInfo", "2015003", "CS");
HBaseOperator.delOneRecordFamily("StuInfo", "2015003", "English");
```

删除所创建的表

删除表需要先通过disableTable方法将该表状态置为disable，然后通过deleteTable方法删除该表：

```
public static void deleteTable(String tablename){
    try{
        admin.disableTable(TableName.valueOf(tablename));
        admin.deleteTable(TableName.valueOf(tablename));
        System.out.println("Delete table:" + tablename + "... Done.");
    }catch (IOException e){
        e.printStackTrace();
    }
}
```

删除创建的表StuInfo的过程：

```
// 删除所创建的表
HBaseOperator.deleteTable("StuInfo");
```

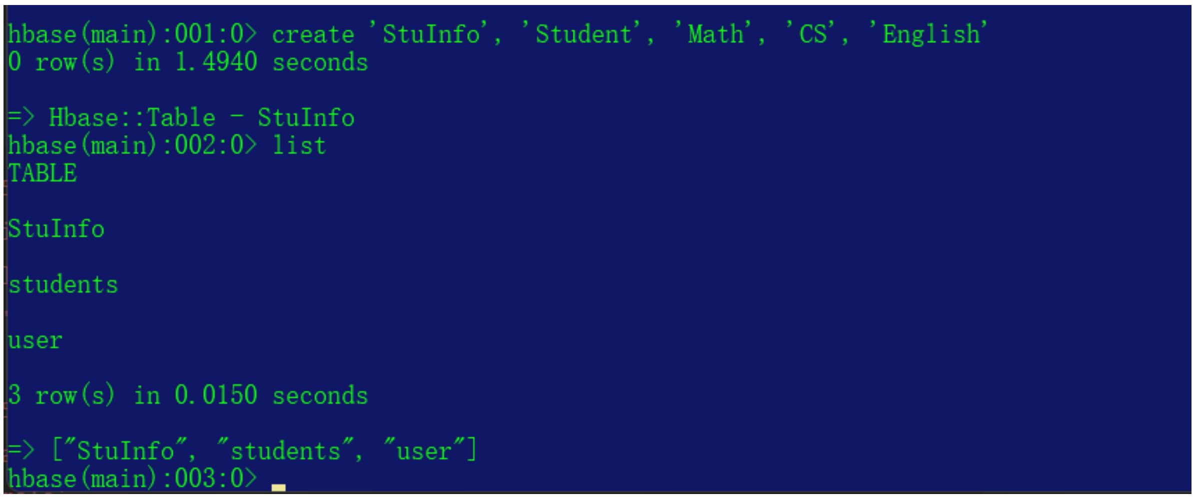
除此之外，还有一些实现了但是本次实验没有用到的方法，包括获取指定行的数据(getOneRecord)，删除指定行(delOneRecord)，查询指定rowkey和列簇下的所有数据(getByRawKeyColumn)。

使用Shell完成上述Java程序的任务

创建表并插入数据

创建表：

```
create 'StuInfo', 'Student', 'Math', 'CS', 'English'
```

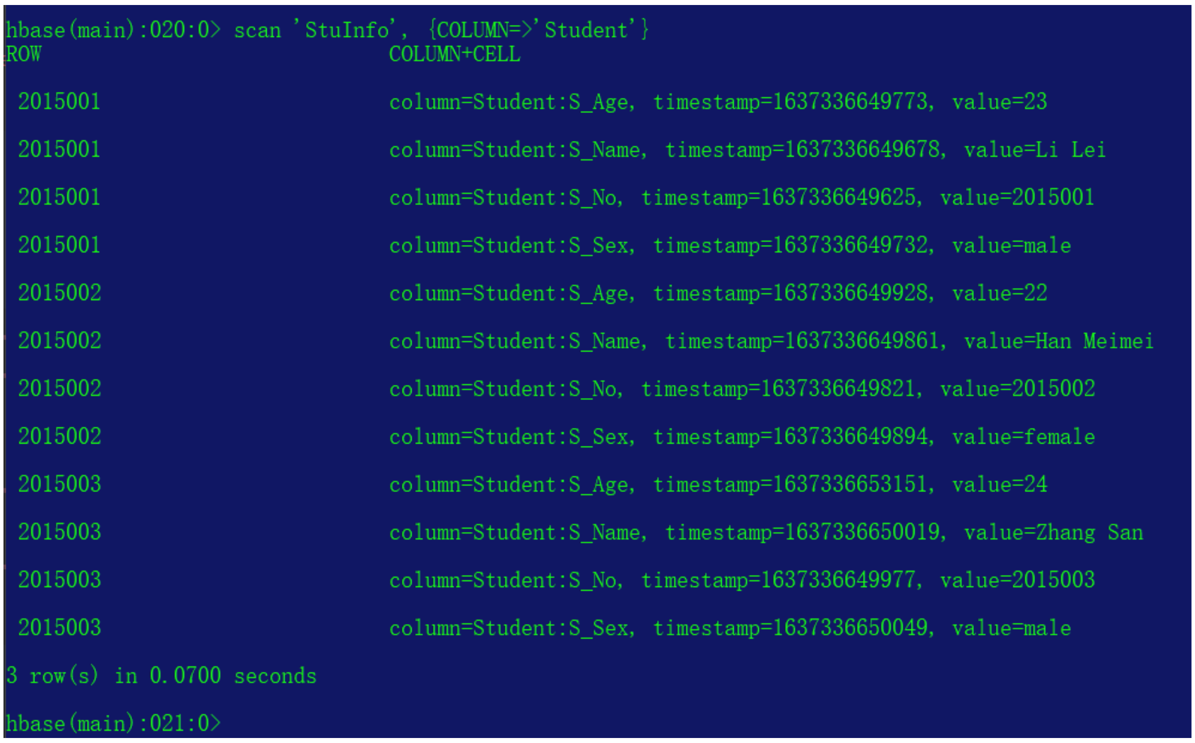


```
hbase(main):001:0> create 'StuInfo', 'Student', 'Math', 'CS', 'English'
0 row(s) in 1.4940 seconds

=> Hbase::Table - StuInfo
hbase(main):002:0> list
TABLE

StuInfo

students

user

3 row(s) in 0.0150 seconds

=> ["StuInfo", "students", "user"]
hbase(main):003:0> 
```

添加学生信息：

```
put 'StuInfo', '2015001', 'Student:S_No', '2015001'
put 'StuInfo', '2015001', 'Student:S_Name', 'Li Lei'
put 'StuInfo', '2015001', 'Student:S_Sex', 'male'
put 'StuInfo', '2015001', 'Student:S_Age', '23'

put 'StuInfo', '2015002', 'Student:S_No', '2015002'
put 'StuInfo', '2015002', 'Student:S_Name', 'Han Meimei'
put 'StuInfo', '2015002', 'Student:S_Sex', 'female'
put 'StuInfo', '2015002', 'Student:S_Age', '22'

put 'StuInfo', '2015003', 'Student:S_No', '2015003'
put 'StuInfo', '2015003', 'Student:S_Name', 'Zhang San'
put 'StuInfo', '2015003', 'Student:S_Sex', 'male'
put 'StuInfo', '2015003', 'Student:S_Age', '24'
```

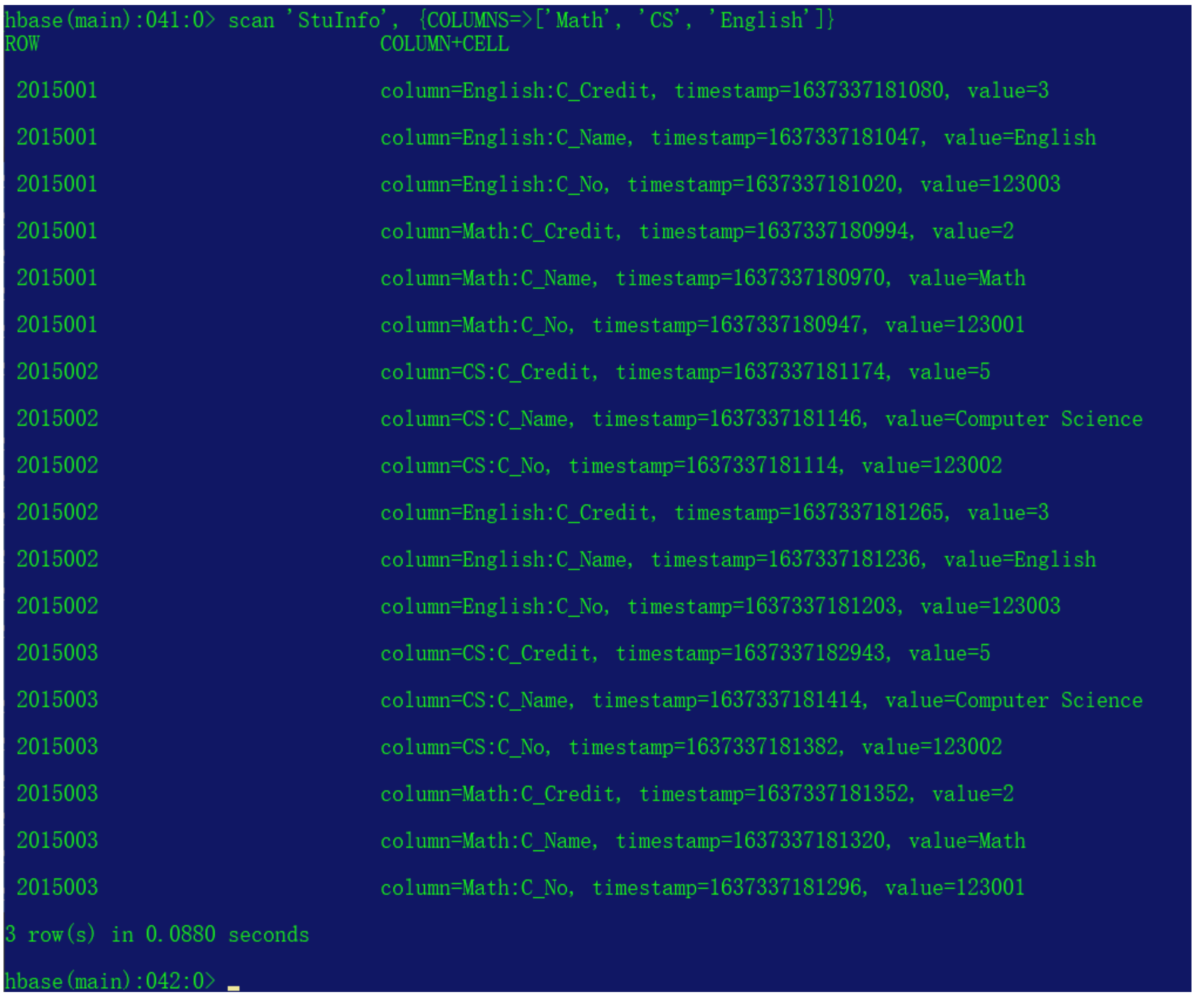
添加结果：


```
hbase(main):020:0> scan 'StuInfo', {COLUMN=>'Student'}
ROW                              COLUMN+CELL
 2015001                         column=Student:S_Age, timestamp=1637336649773, value=23
 2015001                         column=Student:S_Name, timestamp=1637336649678, value=Li Lei
 2015001                         column=Student:S_No, timestamp=1637336649625, value=2015001
 2015001                         column=Student:S_Sex, timestamp=1637336649732, value=male
 2015002                         column=Student:S_Age, timestamp=1637336649928, value=22
 2015002                         column=Student:S_Name, timestamp=1637336649861, value=Han Meimei
 2015002                         column=Student:S_No, timestamp=1637336649821, value=2015002
 2015002                         column=Student:S_Sex, timestamp=1637336649894, value=female
 2015003                         column=Student:S_Age, timestamp=1637336653151, value=24
 2015003                         column=Student:S_Name, timestamp=1637336650019, value=Zhang San
 2015003                         column=Student:S_No, timestamp=1637336649977, value=2015003
 2015003                         column=Student:S_Sex, timestamp=1637336650049, value=male
3 row(s) in 0.0700 seconds
hbase(main):021:0>
```

添加课程信息：

```
put 'StuInfo', '2015001', 'Math:C_No', '123001'
put 'StuInfo', '2015001', 'Math:C_Name', 'Math'
put 'StuInfo', '2015001', 'Math:C_Credit', '2'
put 'StuInfo', '2015001', 'English:C_No', '123003'
put 'StuInfo', '2015001', 'English:C_Name', 'English'
put 'StuInfo', '2015001', 'English:C_Credit', '3'

put 'StuInfo', '2015002', 'CS:C_No', '123002'
put 'StuInfo', '2015002', 'CS:C_Name', 'Computer Science'
put 'StuInfo', '2015002', 'CS:C_Credit', '5'
put 'StuInfo', '2015002', 'English:C_No', '123003'
put 'StuInfo', '2015002', 'English:C_Name', 'English'
put 'StuInfo', '2015002', 'English:C_Credit', '3'

put 'StuInfo', '2015003', 'Math:C_No', '123001'
put 'StuInfo', '2015003', 'Math:C_Name', 'Math'
put 'StuInfo', '2015003', 'Math:C_Credit', '2'
put 'StuInfo', '2015003', 'CS:C_No', '123002'
put 'StuInfo', '2015003', 'CS:C_Name', 'Computer Science'
put 'StuInfo', '2015003', 'CS:C_Credit', '5'
```

添加结果：

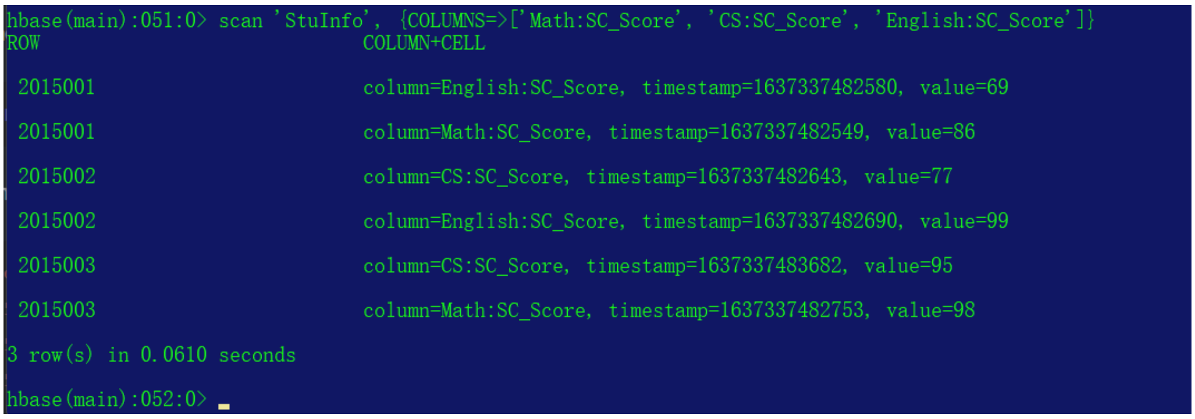


```
hbase(main):041:0> scan 'StuInfo', {COLUMNS=>['Math', 'CS', 'English']}
ROW                              COLUMN+CELL
 2015001                         column=English:C_Credit, timestamp=1637337181080, value=3
 2015001                         column=English:C_Name, timestamp=1637337181047, value=English
 2015001                         column=English:C_No, timestamp=1637337181020, value=123003
 2015001                         column=Math:C_Credit, timestamp=1637337180994, value=2
 2015001                         column=Math:C_Name, timestamp=1637337180970, value=Math
 2015001                         column=Math:C_No, timestamp=1637337180947, value=123001
 2015002                         column=CS:C_Credit, timestamp=1637337181174, value=5
 2015002                         column=CS:C_Name, timestamp=1637337181146, value=Computer Science
 2015002                         column=CS:C_No, timestamp=1637337181114, value=123002
 2015002                         column=English:C_Credit, timestamp=1637337181265, value=3
 2015002                         column=English:C_Name, timestamp=1637337181236, value=English
 2015002                         column=English:C_No, timestamp=1637337181203, value=123003
 2015003                         column=CS:C_Credit, timestamp=1637337182943, value=5
 2015003                         column=CS:C_Name, timestamp=1637337181414, value=Computer Science
 2015003                         column=CS:C_No, timestamp=1637337181382, value=123002
 2015003                         column=Math:C_Credit, timestamp=1637337181352, value=2
 2015003                         column=Math:C_Name, timestamp=1637337181320, value=Math
 2015003                         column=Math:C_No, timestamp=1637337181296, value=123001
3 row(s) in 0.0880 seconds
hbase(main):042:0> _
```

添加成绩信息：

```
put 'StuInfo', '2015001', 'Math:SC_Score', '86'
put 'StuInfo', '2015001', 'English:SC_Score', '69'

put 'StuInfo', '2015002', 'CS:SC_Score', '77'
put 'StuInfo', '2015002', 'English:SC_Score', '99'

put 'StuInfo', '2015003', 'Math:SC_Score', '98'
put 'StuInfo', '2015003', 'CS:SC_Score', '95'
```

添加结果：

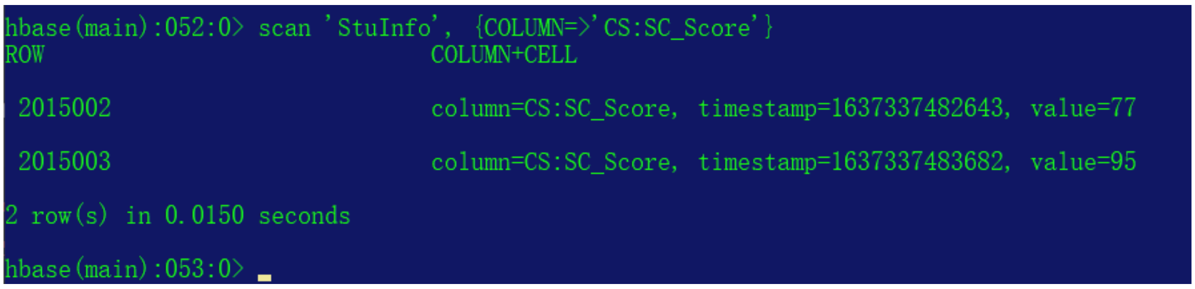


```
hbase(main):051:0> scan 'StuInfo', {COLUMNS=>['Math:SC_Score', 'CS:SC_Score', 'English:SC_Score']}
ROW                              COLUMN+CELL
 2015001                         column=English:SC_Score, timestamp=1637337482580, value=69
 2015001                         column=Math:SC_Score, timestamp=1637337482549, value=86
 2015002                         column=CS:SC_Score, timestamp=1637337482643, value=77
 2015002                         column=English:SC_Score, timestamp=1637337482690, value=99
 2015003                         column=CS:SC_Score, timestamp=1637337483682, value=95
 2015003                         column=Math:SC_Score, timestamp=1637337482753, value=98
3 row(s) in 0.0610 seconds
hbase(main):052:0> _
```

查询选修Computer Science的学生的成绩

```
scan 'StuInfo', {COLUMN=>'CS:SC_Score'}
```

查询结果：

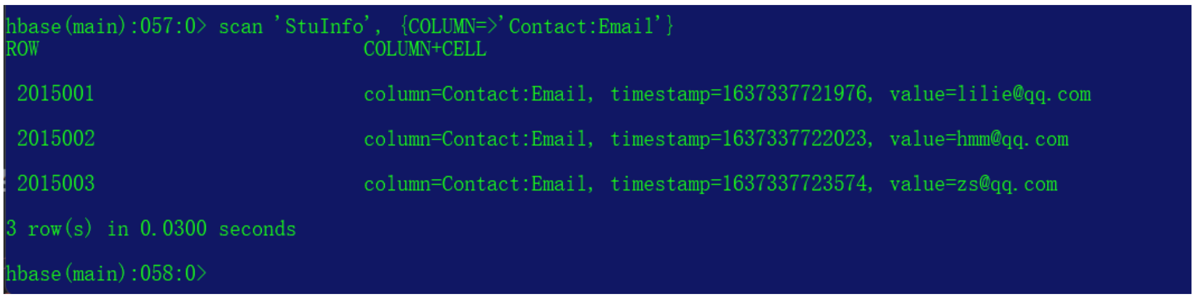


```
hbase(main):052:0> scan 'StuInfo', {COLUMN=>'CS:SC_Score'}
ROW                              COLUMN+CELL

 2015002                         column=CS:SC_Score, timestamp=1637337482643, value=77

 2015003                         column=CS:SC_Score, timestamp=1637337483682, value=95

2 row(s) in 0.0150 seconds

hbase(main):053:0> _
```

增加新的列族和新列Contact:Email，并添加数据

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

结果：

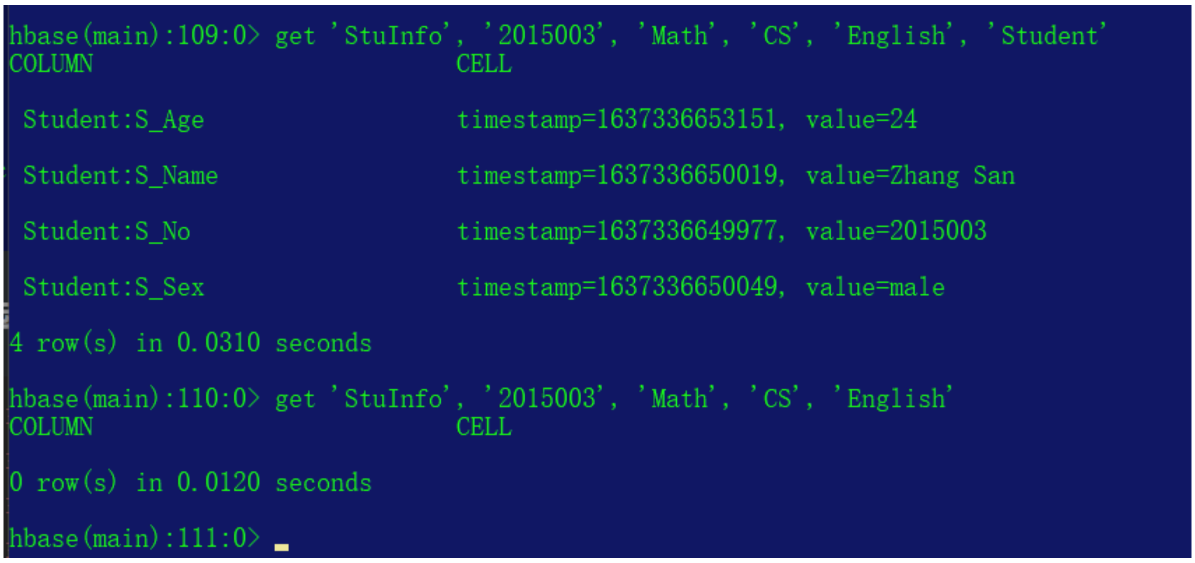


```
hbase(main):057:0> scan 'StuInfo', {COLUMN=>'Contact:Email'}
ROW                              COLUMN+CELL

 2015001                         column=Contact:Email, timestamp=1637337721976, value=lilie@qq.com

 2015002                         column=Contact:Email, timestamp=1637337722023, value=hmm@qq.com

 2015003                         column=Contact:Email, timestamp=1637337723574, value=zs@qq.com

3 row(s) in 0.0300 seconds

hbase(main):058:0>
```

删除学号为2015003的学生的选课记录

```
delete 'StuInfo', '2015003', 'Math:C_No'
delete 'StuInfo', '2015003', 'Math:C_Name'
delete 'StuInfo', '2015003', 'Math:C_Credit'
delete 'StuInfo', '2015003', 'Math:SC_Score'
delete 'StuInfo', '2015003', 'CS:C_No'
delete 'StuInfo', '2015003', 'CS:C_Name'
delete 'StuInfo', '2015003', 'CS:C_Credit'
delete 'StuInfo', '2015003', 'CS:SC_Score'
delete 'StuInfo', '2015003', 'English:C_No'
delete 'StuInfo', '2015003', 'English:C_Name'
delete 'StuInfo', '2015003', 'English:C_Credit'
delete 'StuInfo', '2015003', 'English:SC_Score'
```

删除结果：

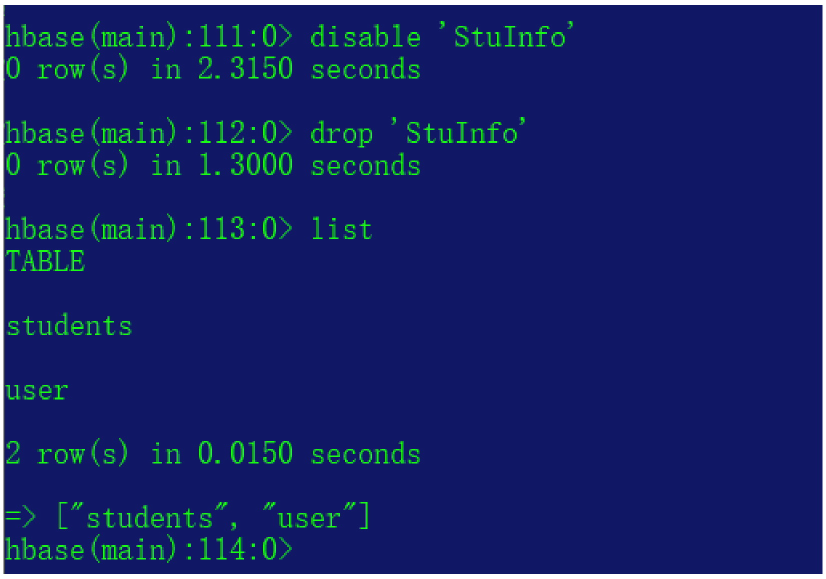


```
hbase(main):109:0> get 'StuInfo', '2015003', 'Math', 'CS', 'English', 'Student'
COLUMN                          CELL

 Student:S_Age                  timestamp=1637336653151, value=24

 Student:S_Name                 timestamp=1637336650019, value=Zhang San

 Student:S_No                   timestamp=1637336649977, value=2015003

 Student:S_Sex                  timestamp=1637336650049, value=male

4 row(s) in 0.0310 seconds

hbase(main):110:0> get 'StuInfo', '2015003', 'Math', 'CS', 'English'
COLUMN                          CELL

0 row(s) in 0.0120 seconds

hbase(main):111:0> _
```

删除所创建的表

```
disable 'StuInfo'
drop 'StuInfo'
```

删除结果:



```
hbase(main):111:0> disable 'StuInfo'
0 row(s) in 2.3150 seconds

hbase(main):112:0> drop 'StuInfo'
0 row(s) in 1.3000 seconds

hbase(main):113:0> list
TABLE

students

user

2 row(s) in 0.0150 seconds

=> ["students", "user"]
hbase(main):114:0>
```