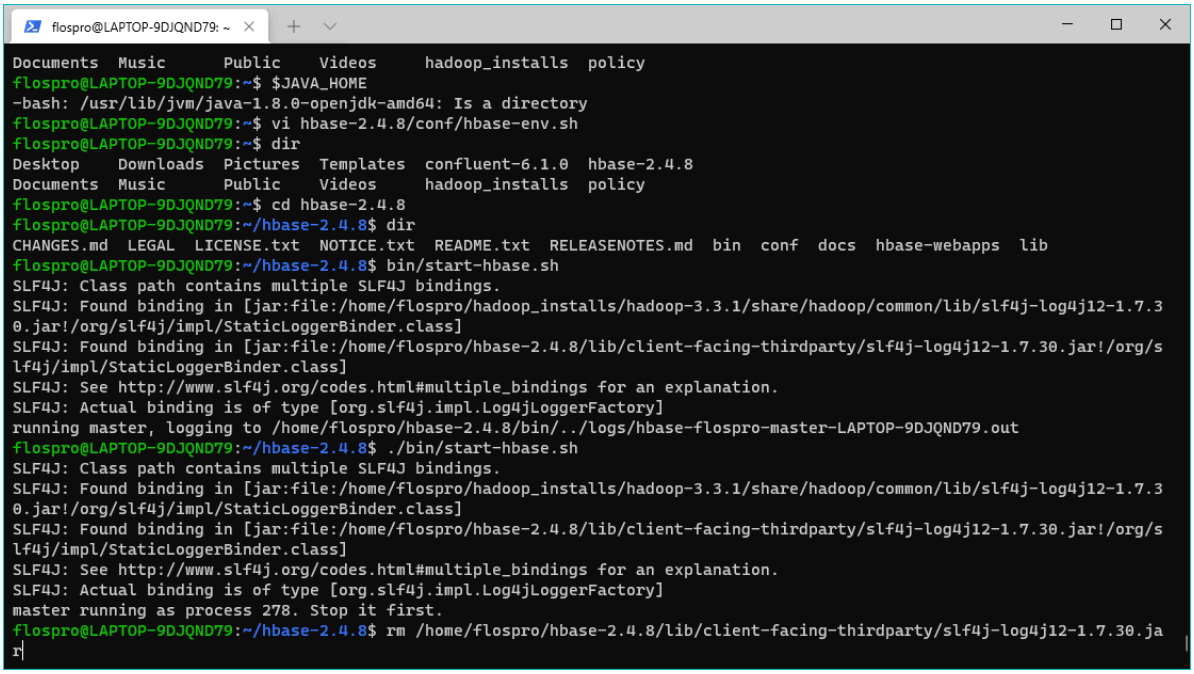
实验三

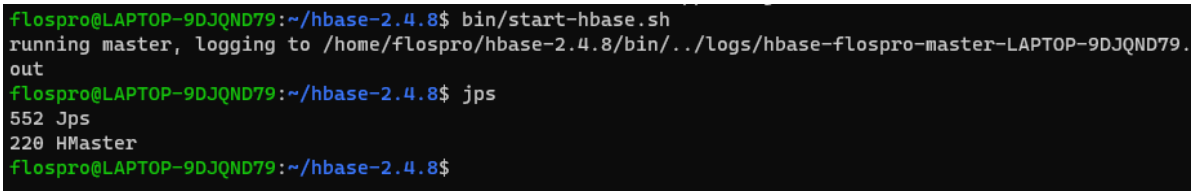
实验环境：win10 idea+wsl

首先下载安装HBase，在hbase-env.sh中配置JAVA_HOME。

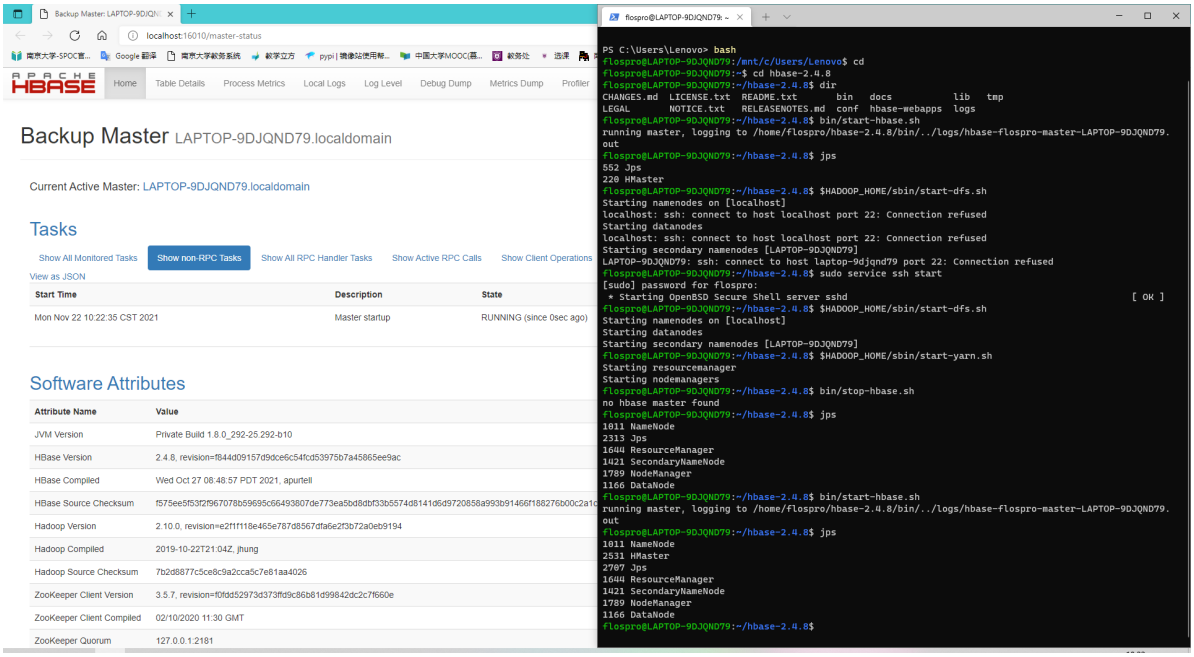
HBase单机运行



启动后发现多了一个slf4j，但是实际上应该不影响运行，实际操作时直接删除了HBase的slf4j。



启动HBase，发现启动失败，并且无法打开WebUI，这是因为没有事先启动HDFS。如果此时尝试直接启动HDFS，HBase仍然无法打开，这是因为违背了启动逻辑，会把HBase的部分进程顶掉。按照正常顺序可以启动。



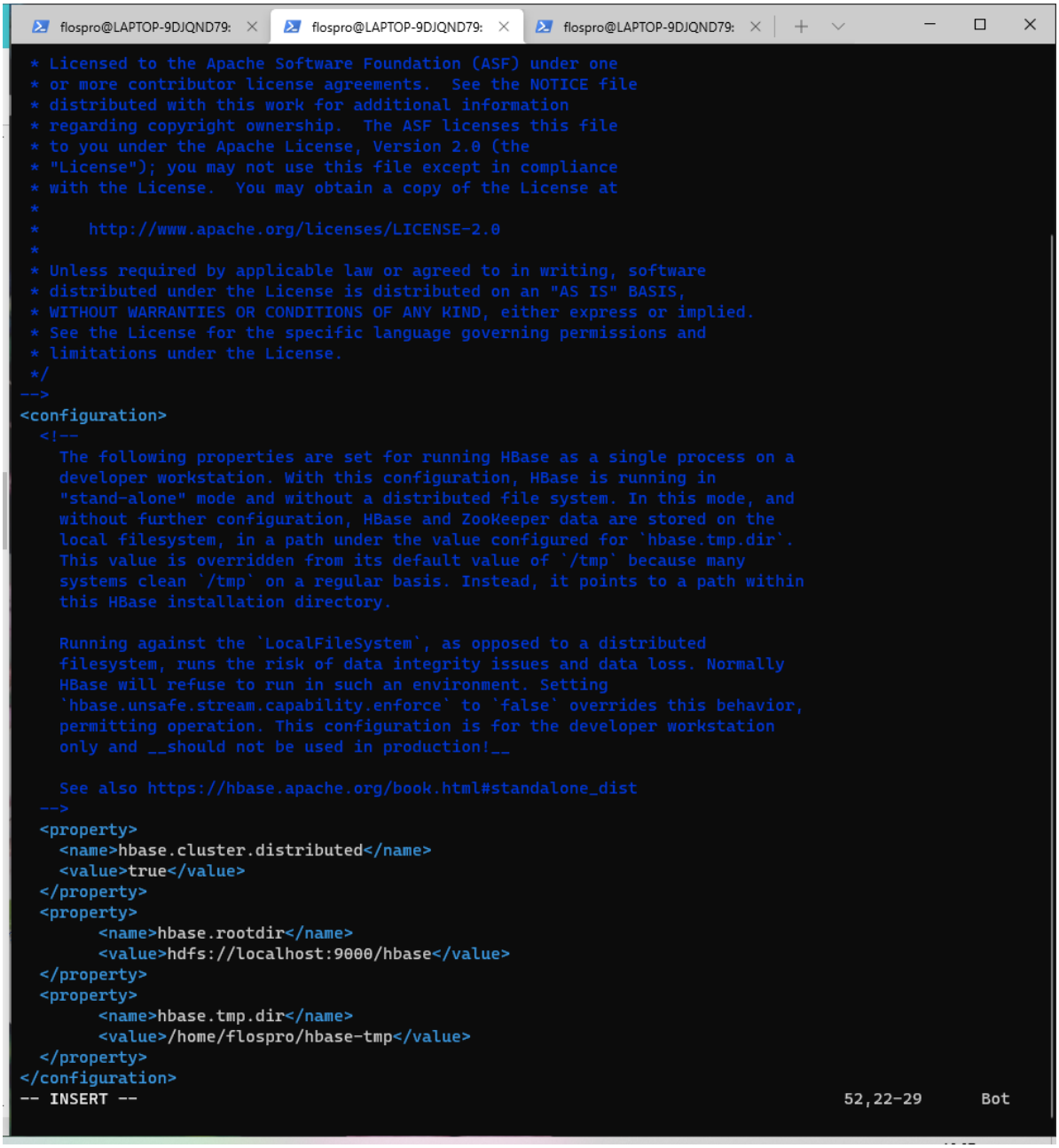
关闭单机HBase，会发现关闭非常的慢。可以改为下面两个命令加速stop进程。

```
bin/hbase-daemon.sh stop master
bin/hbase-daemon.sh stop regionserver
```

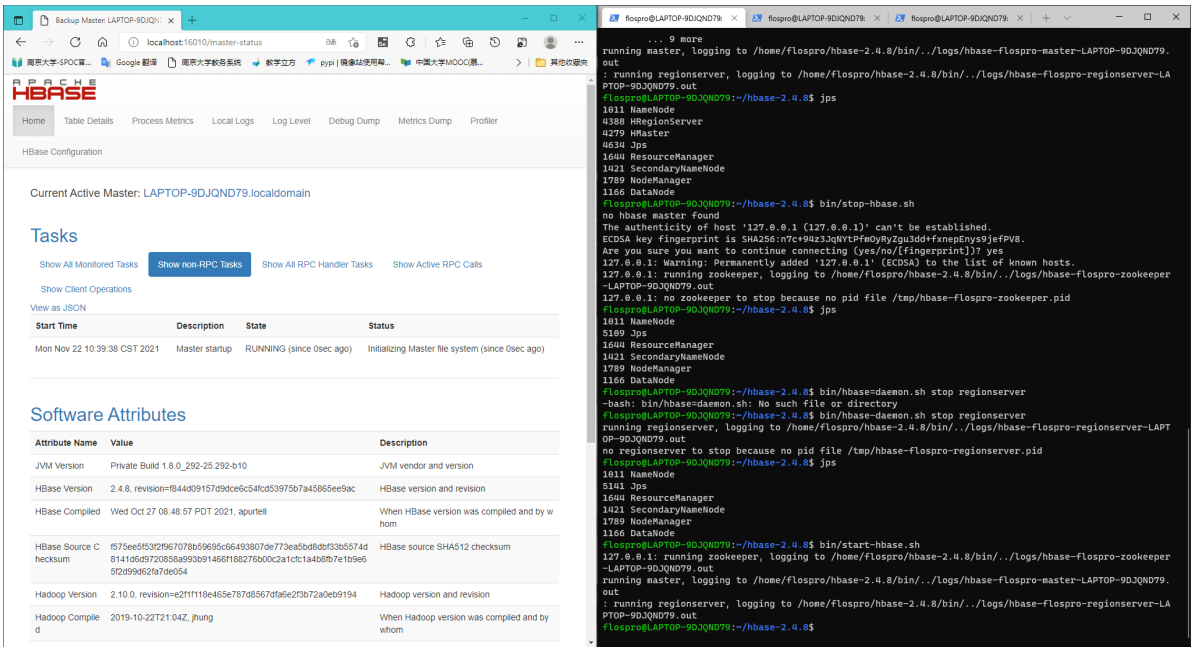
```
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ bin/stop-hbase.sh
stopping hbase.....................................................................................
.....................................................................
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ hbase-daemon.sh stop regionserver
hbase-daemon.sh: command not found
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ bin/hbase-daemon.sh stop regionserver
running regionserver, logging to /home/flospro/hbase-2.4.8/bin/../logs/hbase-flospro-regionserver-LAPTOP-9DJQND79.out
no regionserver to stop because no pid file /tmp/hbase-flospro-regionserver.pid
```

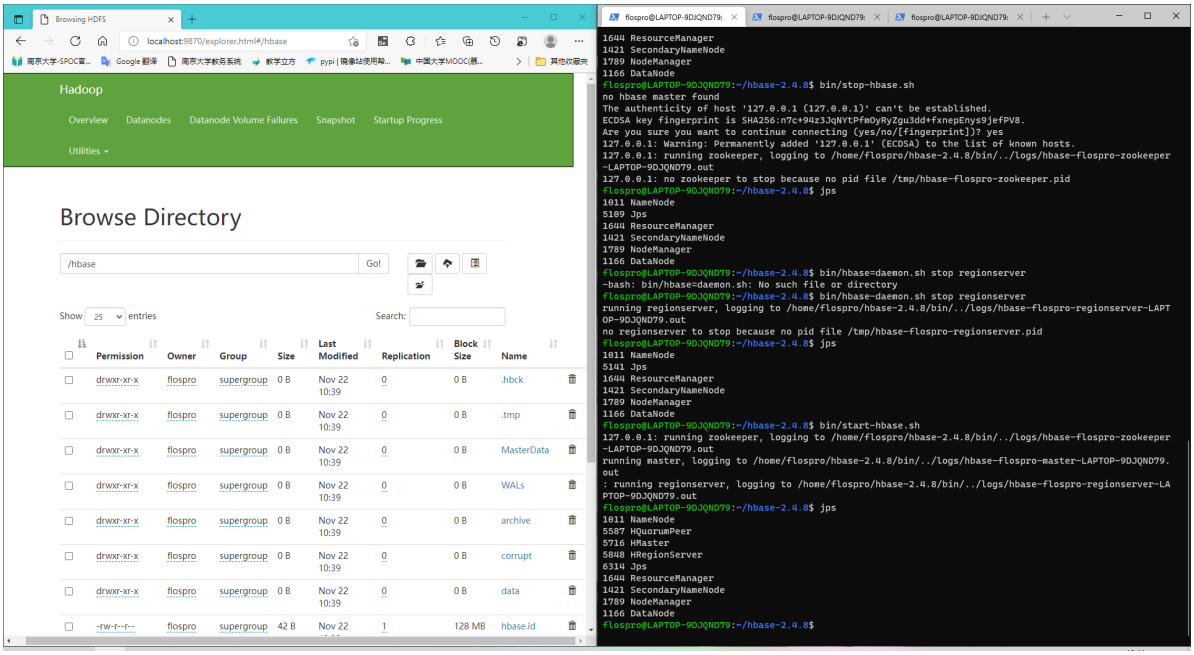
HBase伪分布式运行

首先修改配置文件，经检查hadoop下面的fs.defaultFS是hdfs://localhost:9000，修改hbase-site.xml。



可以发现WebUI和dfs集群文件正常。





使用bin/hbase shell成功进入shell：

```
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8/bin$ dir
chaos-daemon.sh       hbase-config.cmd        local-regionservers.sh    start-hbase.cmd
considerAsDead.sh     hbase-config.sh         master-backup.sh          start-hbase.sh
draining_servers.rb   hbase-daemon.sh         region_mover.rb           stop-hbase.cmd
get-active-master.rb  hbase-daemons.sh        region_status.rb          stop-hbase.sh
graceful_stop.sh      hbase-jruby             regionservers.sh          test
hbase                 hbase.cmd               replication               zookeepers.sh
hbase-cleanup.sh      hirb.rb                 rolling-restart.sh
hbase-common.sh       local-master-backup.sh  shutdown_regionserver.rb
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8/bin$ ./hbase shell
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 2.4.8, rf844d09157d9dce6c54fcd53975b7a45865ee9ac, Wed Oct 27 08:48:57 PDT 2021
Took 0.0022 seconds
hbase:001:0>
```

但是无法进行创建表格等基本操作：

```
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8/bin$ ./hbase shell
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 2.4.8, rf844d09157d9dce6c54fcd53975b7a45865ee9ac, Wed Oct 27 08:48:57 PDT 2021
Took 0.0014 seconds
hbase:001:0> create 'Student','StuInfo','Grades'

ERROR: org.apache.hadoop.hbase.ipc.ServerNotRunningYetException: Server is not running yet
        at org.apache.hadoop.hbase.master.HMaster.checkServiceStarted(HMaster.java:2817)
        at org.apache.hadoop.hbase.master.MasterRpcServices.isMasterRunning(MasterRpcServices.java:120
5)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlocking
Method(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:392)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:354)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:334)

For usage try 'help "create"'

Took 9.4718 seconds
hbase:002:0> create 'Student','StuInfo','Grades'

ERROR: org.apache.hadoop.hbase.ipc.ServerNotRunningYetException: Server is not running yet
        at org.apache.hadoop.hbase.master.HMaster.checkServiceStarted(HMaster.java:2817)
        at org.apache.hadoop.hbase.master.MasterRpcServices.isMasterRunning(MasterRpcServices.java:120
5)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlocking
Method(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:392)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:354)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:334)

For usage try 'help "create"'

Took 8.2278 seconds
```

这个报错并没有在网上直接找到好的解决方法，于是直接想到的就是先尝试改用外置ZooKeeper。首先下载外置ZooKeeper：

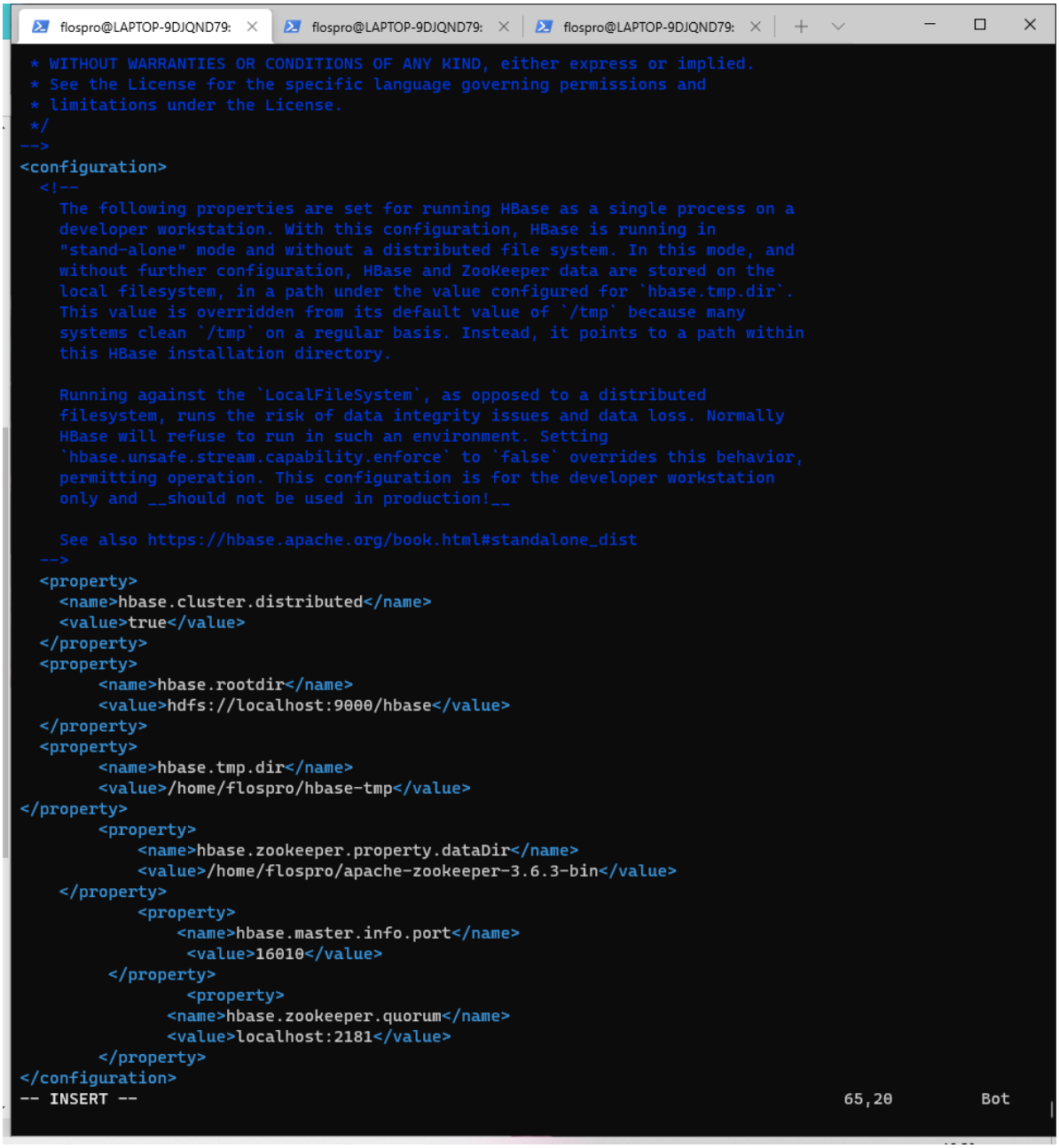
```
wget https://mirrors.nju.edu.cn/apache/zookeeper/zookeeper-3.6.3/apache-zookeeper-3.6.3-bin.tar.gz
```

在hbase-env.sh中禁用内部zk：

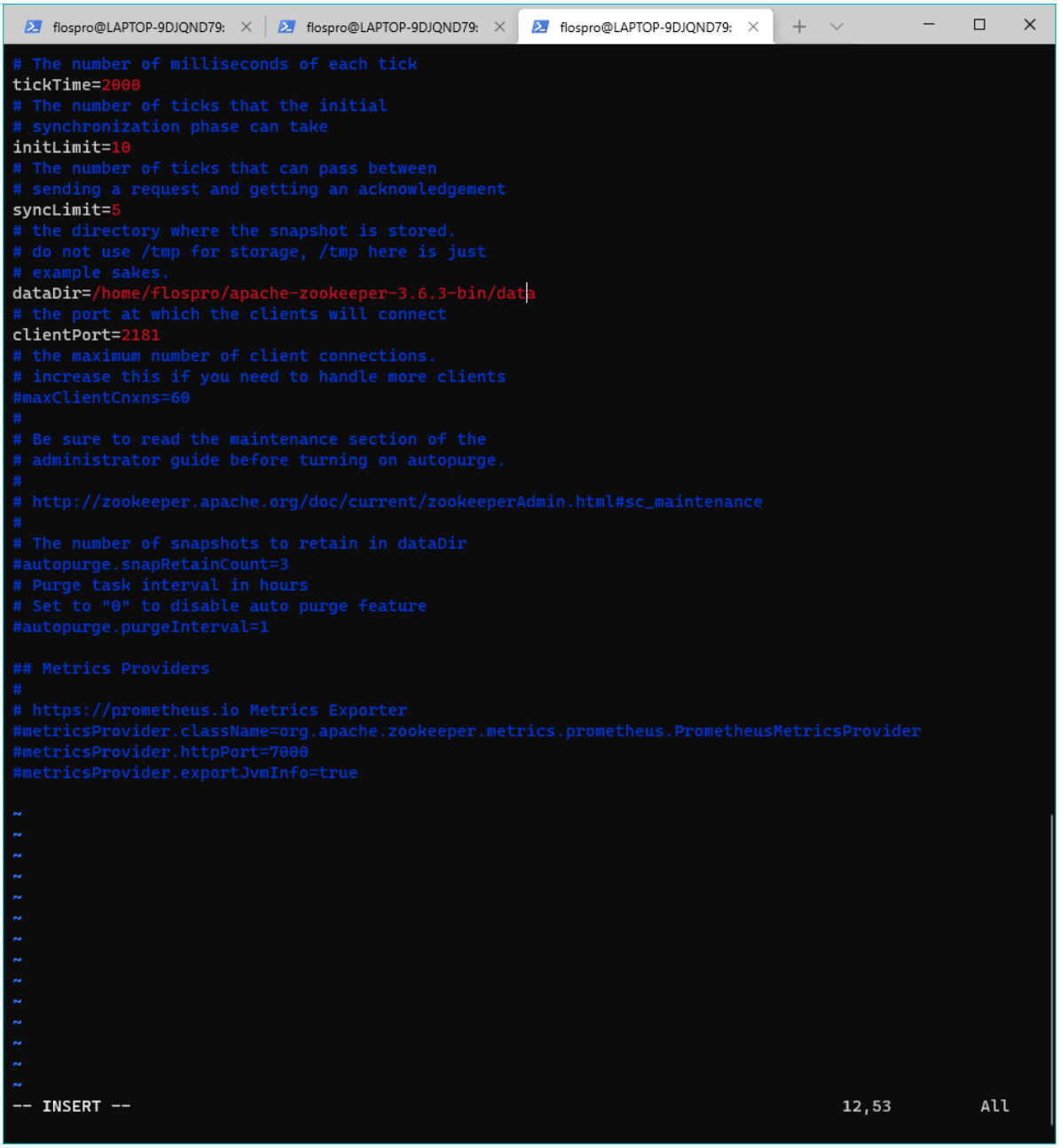
```
# The java implementation to use.  Java 1.8+ required.
# export JAVA_HOME=/usr/java/jdk1.8.0/
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
export HBASE_MANAGES_ZK=false
# Extra Java CLASSPATH elements.  Optional.
# export HBASE_CLASSPATH=

# The maximum amount of heap to use. Default is left to JVM
# export HBASE_HEAPSIZE=1G
```

修改hbase-site.xml:



复制配置文件样例，编辑zoo.cfg，将dataDir设置在zk目录下的data文件夹中：



启动HBase前，首先启动zk：

```
flospro@LAPTOP-9DJQND79:~/apache-zookeeper-3.6.3-bin/bin$ dir
README.txt    zkCli.sh   zkServer-initialize.sh  zkSnapShotToolkit.cmd  zkTxnLogToolkit.sh
zkCleanup.sh  zkEnv.cmd  zkServer.cmd            zkSnapShotToolkit.sh
zkCli.cmd     zkEnv.sh   zkServer.sh             zkTxnLogToolkit.cmd
flospro@LAPTOP-9DJQND79:~/apache-zookeeper-3.6.3-bin/bin$ ./zkServer.sh
ZooKeeper JMX enabled by default
Using config: /home/flospro/apache-zookeeper-3.6.3-bin/bin/../conf/zoo.cfg
Usage: ./zkServer.sh [--config <conf-dir>] {start|start-foreground|stop|version|restart|status|print-cmd}
flospro@LAPTOP-9DJQND79:~/apache-zookeeper-3.6.3-bin/bin$ ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /home/flospro/apache-zookeeper-3.6.3-bin/bin/../conf/zoo.cfg
Starting zookeeper ... STARTED
flospro@LAPTOP-9DJQND79:~/apache-zookeeper-3.6.3-bin/bin$
```

外置zk配置成功，参考：

http://blog.iis7.com/article/39306.html
https://www.iteye.com/blog/qindongliang-1978485
https://www.cnblogs.com/rgever/p/10222167.html

但是对解决问题没有任何帮助。

再次搜索资料发现，在hbase-site.xml中似乎需要补充配置数据，WAL(Write-Ahead-Log)需要指定写入模式(WALFactory)，有defaultProvider、asyncfs、filesystem、multiwal四种选项。

```
  <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>/home/flospro/hbase-tmp</value>
</property>
        <property>
            <name>hbase.zookeeper.property.dataDir</name>
            <value>/home/flospro/apache-zookeeper-3.6.3-bin</value>
    </property>
            <property>
                <name>hbase.master.info.port</name>
                 <value>16010</value>
         </property>
                 <property>
                <name>hbase.zookeeper.quorum</name>
                <value>localhost:2181</value>
        </property>
                <property>
  <name>hbase.wal.provider</name>
  <value>filesystem</value>
</property>
</configuration>
-- INSERT --                                                    68,28         Bot
```

此时发现可以正常进行HBase的shell操作。参考https://www.cnblogs.com/live41/p/15497029.html。

```
2195 Jps
483 NameNode
899 SecondaryNameNode
1500 HRegionServer
1038 QuorumPeerMain
639 DataNode
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ vi conf/hbase-site.xml
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ bin/hbase-daemon.sh stop master
running master, logging to /home/flospro/hbase-2.4.8/bin/../logs/hbase-flospro-master-LAPTOP-9DJQND79.out
stopping master.
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ bin/hbase-daemon.sh stop regionserver
running regionserver, logging to /home/flospro/hbase-2.4.8/bin/../logs/hbase-flospro-regionserver-LAPTOP-9DJQND79.out
stopping regionserver.
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ bin/stop-hbase.sh
no hbase master found
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ bin/start-hbase.sh
running master, logging to /home/flospro/hbase-2.4.8/bin/../logs/hbase-flospro-master-LAPTOP-9DJQND79.out
: running regionserver, logging to /home/flospro/hbase-2.4.8/bin/../logs/hbase-flospro-regionserver-LAPTOP-9DJQND79.out
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ bin/hbase shell
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 2.4.8, rf844d09157d9dce6c54fcd53975b7a45865ee9ac, Wed Oct 27 08:48:57 PDT 2021
Took 0.0015 seconds
hbase:001:0> create 'Student','StuInfo','Grades'
Created table Student
Took 1.1804 seconds
=> Hbase::Table - Student
hbase:002:0>
```

顺手把HBase和zk加入环境变量：

```
export CONFLUENT_HOME=~/confluent-6.1.0
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
export HADOOP_HOME=/home/flospro/hadoop_installs/hadoop-3.3.1
export HBASE_HOME=/home/flospro/hbase-2.4.8
export ZK_HOME=/home/flospro/apache-zookeeper-3.6.3-bin
export PATH=$JAVA_HOME/bin:$HBASE_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$CONFLUENT_HOME/bin:$HADOOP_HOME/bin:$PATH
export DISPLAY=:0.0
export LIBGL_ALWAYS_INDIRECT=1
export XMODIFIERS=@im=fcitx
export GTK_IM_MODULE=fcitx
export QT_IM_MODULE=fcitx
```

Java代码逻辑

首先根据关系型数据表格，设计适合HBase的列族数据表格：

原表格共三张，为学生表、课程、选课表。课程表实际上是一张2NF的表，每一行的三个属性高度绑定，若出现其中一个必然绑定另外两个固有属性同时出现，因此如果把这张表的内容作为另一张大表的一部分，一定会出现数据冗余。因此在HBase中，针对原有数据内容这里设计了两张表，一张是学生表和选课表的join（SC表），一张是原有的课程表：

行键	列族 StuInfo			列族 Grades		
	S_Name	S_Sex	S_Age	123001	123002	123003
2015001	Li Lei	male	23	86		69
2015002	Han Meimei	female	22		77	99
2015003	Zhang San	male	24	98	95	

因为空cell不占用存储，所以Grades下课程多少都没关系。

行键	列族 C_Info	
	C_Name	C_Credit
123001	Math	2.0
123002	Computer Science	5.0
123003	English	3.0

具体实现

在idea的run configuration中选择run on在wsl上，同时也已启动wsl上的HBase后，首先连接HBase：

```
public static void getConnect() throws IOException {
        conf=HBaseConfiguration.create();
        try{
            connection=ConnectionFactory.createConnection(conf);
            System.out.println("Connect to HBASE successfully");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

1. **设计并创建合适的表；**

    设计见上。

    创建表格使用函数：

```
public static void createTable(String table, String[] familyNames) throws
IOException{
        TableName tableName = TableName.valueOf(table);
        Admin admin = connection.getAdmin();
        if(admin.tableExists(tableName)) {
            admin.disableTable(tableName);
            admin.deleteTable(tableName);
```

```
            System.out.println("Table '"+tableName.toString() + "' is exist,
it will be deleted and recreated");
        }
            System.out.println("start create table '"+table+"'");
            HTableDescriptor tdesc = new HTableDescriptor(tableName);
            for (String familyName : familyNames) {
                tdesc.addFamily(new HColumnDescriptor(familyName));
            }
            admin.createTable(tdesc);
            System.out.println("create table '"+table+"' successfully");
            admin.close();
    }
```

为了方便反复试验，首先判断将要创建的表格是否存在，若存在则删除重建。函数通过传入表名和列族名（字符数组）执行。

插入数据使用函数：

```
public static void addData(String tableName, String rowKey, String family,
String column, String value) throws IOException {
        Table table = connection.getTable(TableName.valueOf(tableName));
        Put put = new Put(Bytes.toBytes(rowKey));
        put.addColumn(Bytes.toBytes(family), Bytes.toBytes(column),
Bytes.toBytes(value));
        table.put(put);
        System.out.println("put " + rowKey + " | " + family + " => "+column
+ " : " + value + " to table " + tableName + " successfully");
    }
```

传入表名、行键、列族、列、值作为参数。

为了检查是否成功创建表格并插入数据，构造scan函数查看表格，其中cellDecoder函数为格式化输出函数，参数为表名：

```
  public static void scan(String tableName) throws IOException {
        System.out.println("show table '"+tableName +"' status:");
        Table table = connection.getTable(TableName.valueOf(tableName));
        ResultScanner scanner = table.getScanner(new Scan());
        for (Result result : scanner) {
            byte[] row = result.getRow();
            System.out.println("row key is: " + new String(row));
            cellDecoder(result);
        }
        System.out.println("that's the case & continue next step\n");
        scanner.close();
    }
```

2. **查询选修Computer Science的学生的成绩；**

由于HBase中是两张表，SC表中课程数据是列限定符且为课程号，所以需要进行一次嵌套查询。首先在课程表中查询CS课程对应的课程号，结果作为字符数组返回，再在SC表中查询数组中所有元素对应的列限定符所在的行，返回这些行的结果。具体来说：

```
public static String[] colFilter(String tableName, String colFamily, String
col, String value) throws IOException {
```

```
        System.out.println("start filtering table '"+tableName +"' for
specific "+colFamily+"~"+col+":"+value+":");
        Table table = connection.getTable(TableName.valueOf(tableName));
        Scan scan = new Scan();
        SingleColumnValueFilter filter = new
SingleColumnValueFilter(Bytes.toBytes(colFamily),Bytes.toBytes(col),
CompareFilter.CompareOp.EQUAL, Bytes.toBytes(value));
        scan.setFilter(filter);
        ResultScanner scanner = table.getScanner(scan);
        List<String> ansRowKey = new ArrayList<String>();
        for (Result result : scanner) {
            for (Cell cell : result.rawCells())
                ansRowKey.add(new String(CellUtil.cloneRow(cell)));
        }
        Set set = new HashSet();
        set.addAll(ansRowKey);
        ansRowKey.clear();
        ansRowKey.addAll(set);
        String ans[] = ansRowKey.toArray(new String[ansRowKey.size()]);
        for(int i = 0; i < ans.length; i++)
            System.out.println(ans[i]+" satisfies the filter");
        scanner.close();
        return ans;
    }
```

首先查询课程号，函数以表名、列族、列、值作为参数，对scan操作施加过滤器**SingleColumnValueFilter**，比较符为“=”，作用是在指定的列族和列中进行值相等的比较，即查找C_Info中的C_Name等于Computer Science的行。

需要注意的是，这里的对查询结果的cell进行遍历时，由于一个逻辑行拥有多个cell，对result遍历记录行键时结果会多次记录，需要去重，比如将数族转化为集合再转化回来。这样得到结果以字符数组形式传回。

```
public static void qualFilter(String tableName, String family, String
colName) throws IOException {
        System.out.println("start filtering table '" + tableName + "' for
specific colName:" + colName + ":");
        Table table = connection.getTable(TableName.valueOf(tableName));
        Scan scan = new Scan();
        scan.setFilter(new QualifierFilter(CompareOperator.EQUAL, new
BinaryComparator(Bytes.toBytes(colName))));
        ResultScanner scanner = table.getScanner(scan);
        for (Result result : scanner) {
            for (Cell cell : result.rawCells()) {
                System.out.println("filtered student " + new
String(CellUtil.cloneRow(cell)) + " get "+new
String(CellUtil.cloneValue(cell))+" in "+ colName);
            }
        }
        scanner.close();
    }
```

接下来通过for循环遍历上一步查询结果的数组，传入表名、列族和得到的结果执行父查询，虽然实际上我们了解数组中其实只有一个元素但我们仍需标准一点使用数组和循环。

由于表格设计中课程号的信息存储在列限定符中，使用**QualifierFilter**过滤器筛选查询结果，比较符为“=”，比较器为**BinaryComparator**比较完整字节数组，此时我们得到的结果只有一个个列限定符为123002（CS课程号）的cell，非常干净没有其他数据，cell的行键即为学号，cell内的值即为成绩，可以在遍历时直接格式化输出。

3. **增加新的列族和新列Contact:Email，并添加数据；**

HBase中列族不能直接添加，需要借助admin的接口添加。据悉，较新版本的HBase增加列族不需要事先diable表格：

```
HColumnDescriptor newFamliy = new HColumnDescriptor("Contact");
Admin admin = connection.getAdmin();
admin.addColumn(TableName.valueOf("Students"), newFamliy);
admin.close();
```

之后调用addData函数即可。

4. **删除学号为2015003的学生的选课记录；**

删除选课记录，在原关系型数据库表格中不考虑将选课表拆分成更高范式的情况下相当于直接删除选课表中对应学号的所有数据（若可拆分，则是可以保留成绩的，否则成绩也会一并抹去，两种情况都是合理的需求情况）。

在HBase中，选课表已经和学生表绑定且所有数据存在Grades列族中，故删除选课记录对应的就是删除特定行键对应的Grades列族下的所有数据。实现函数：

```
public static void deleteRow(String tableName, String rowKey, String
colFami) throws IOException {
        System.out.println("start filtering table '" + tableName + "' to
find specific student " + rowKey + ":");
        Table table = connection.getTable(TableName.valueOf(tableName));
        Scan scan = new Scan();
        FilterList allFilters = new
FilterList(FilterList.Operator.MUST_PASS_ALL);
        allFilters.addFilter(new FamilyFilter(CompareOperator.EQUAL, new
BinaryComparator(Bytes.toBytes(colFami))));
        allFilters.addFilter(new RowFilter(CompareFilter.CompareOp.EQUAL,
new RegexStringComparator(rowKey)));
        scan.setFilter(allFilters);
        ResultScanner scanner = table.getScanner(scan);
        for (Result result : scanner){
            for (Cell cell : result.rawCells()) {
                Delete d = new Delete(result.getRow());
                d.addFamily(Bytes.toBytes(colFami));
                table.delete(d);
            }
            System.out.println("Row " + rowKey +" column family " + colFami
+" from table " + tableName + " deleted");
        }
        scanner.close();
    }
```

传入表名、行键、列族进入函数。删除的方法就是找到所有满足条件的cell，然后调用delete方法删除。有两点需要注意：

1. 要找到所有满足条件的cell，需要满足行键正确，列族名正确两个过滤条件，此时scan的过滤器应该使用继承于抽象类Filter的Filterlist类，设置这个综合过滤器的过滤条件为构成其的子过滤器的交，即**MUST_PASS_ALL**（对应MUST_PASS_ONE则是并），构成它的两个过滤器一个过滤行键、一个过滤列族。

   过滤行键的过滤器为**RowFilter**，针对列族进行过滤的过滤器为**FamilyFilter**，这里两个比较符都为“=”，比较器为**BinaryComparator**比较完整字节数组或者**RegexStringComparator**匹配正则表达式等在这里都能得到正确结果，以前者更符合实际意义。

   参考：java - HBase: How to specify multiple prefix filters in a single scan operation - Stack Overflow

   Hbase FilterList使用总结 - Syn良子 - 博客园 (cnblogs.com)

2. delete方法接受delete对象作为参数，delete对象通过行键实例化，如果此时直接删除，则会连带该逻辑行其它列族也被删除，需要对delete对象限制删除列族。

改进：但当想到这里的时候，其实发现自己已经绕了个弯子，既然可以直接根据行键和列族执行删除，其实就不需要前面两个过滤器了，代码也会简洁许多。

5. **删除所创建的表。**

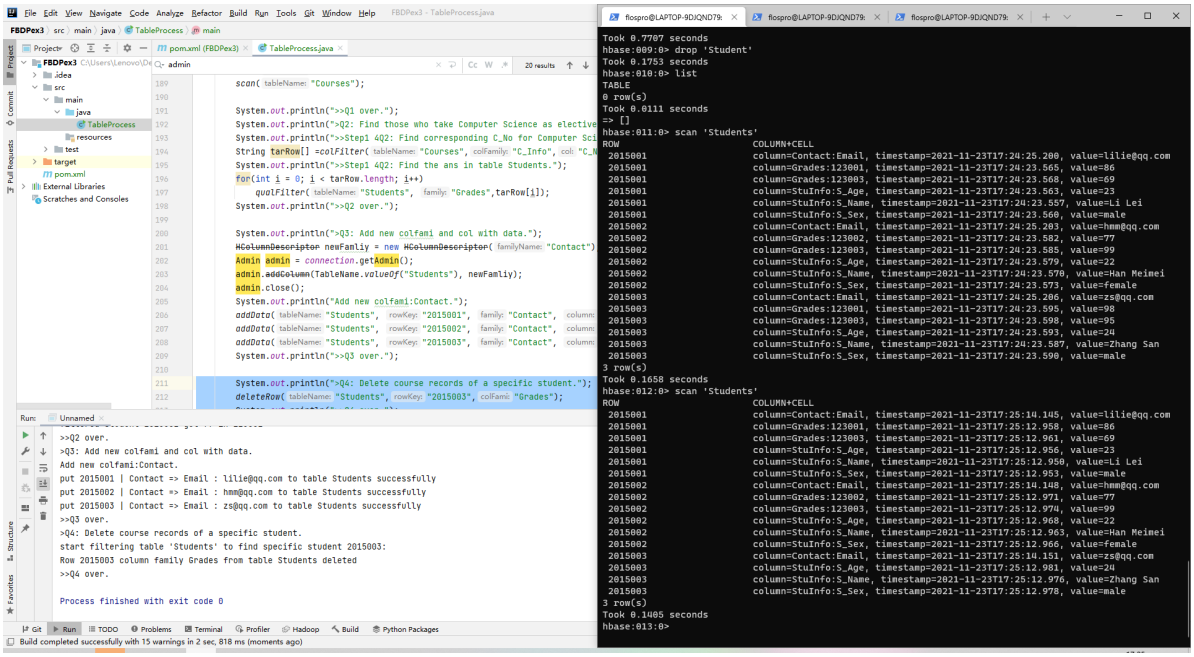
   创建dropTable函数，传入表名，若表不存在则跳过，存在则先disable再delete。

```
public static void dropTable(String table) throws IOException {
        Admin admin = connection.getAdmin();
        TableName tableName = TableName.valueOf(table);
        if (!admin.tableExists(tableName)) {
            System.out.println("Table '"+tableName + "' does not exist");
            return;
        }
        admin.disableTable(tableName);
        admin.deleteTable(tableName);
        System.out.println("Table '"+tableName.toString() + "' has been
deleted");
        admin.close();
    }
```

最后如果没有手动关闭与HBase的连接，等待片刻后程序会自动结束，及时关闭连接就不需要等待了。

```
connection.close();
```

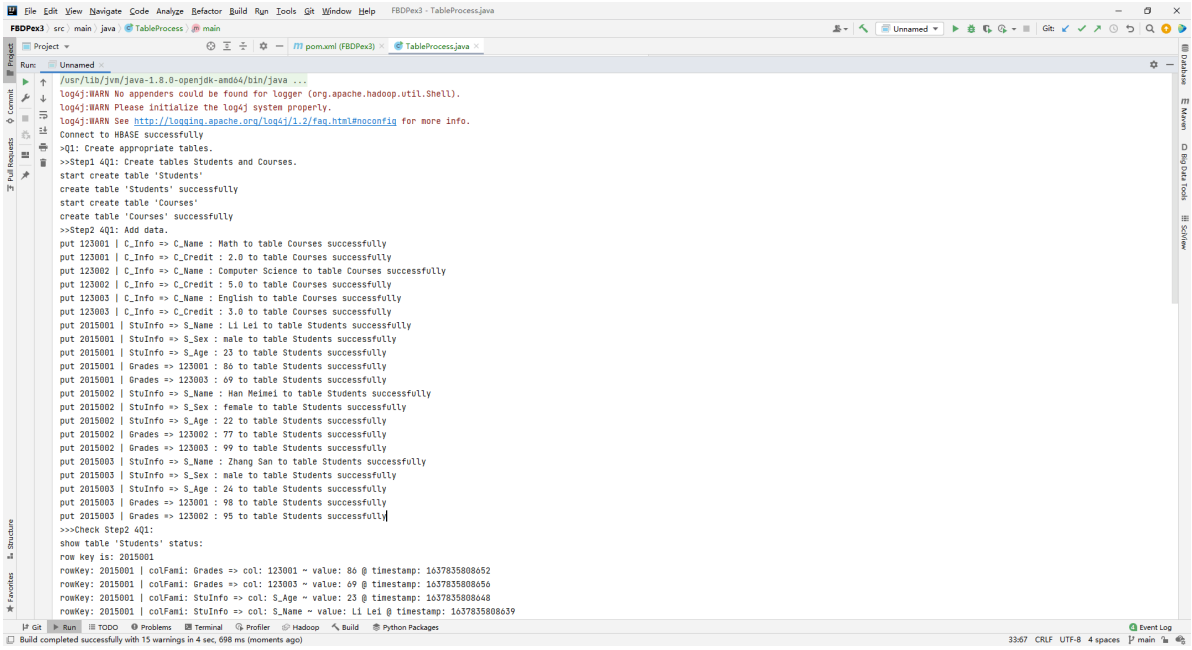
以下是部分运行截图：



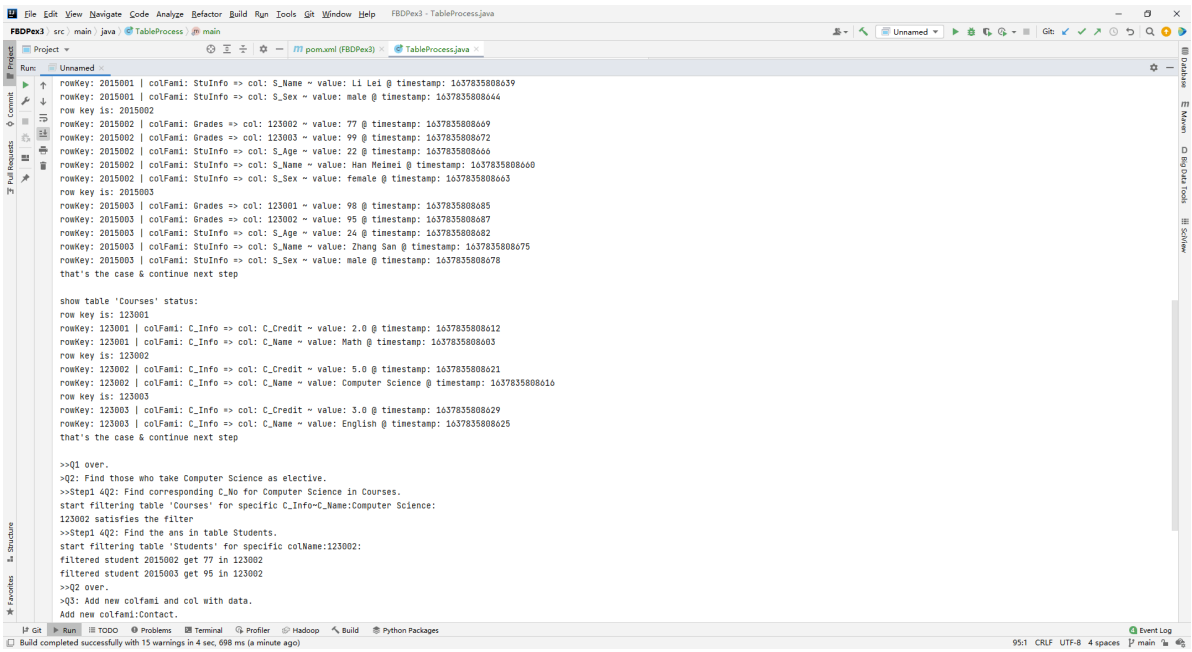
```
Took 0.7707 seconds
hbase:009:0> drop 'Student'
Took 0.1753 seconds
hbase:010:0> list
TABLE
0 row(s)
Took 0.0111 seconds
=> []
hbase:011:0> scan 'Students'
ROW                     COLUMN+CELL
 2015001                column=Contact:Email, timestamp=2021-11-23T17:24:25.200, value=lilie@qq.com
 2015001                column=Grades:123001, timestamp=2021-11-23T17:24:23.565, value=86
 2015001                column=Grades:123003, timestamp=2021-11-23T17:24:23.568, value=69
 2015001                column=StuInfo:S_Age, timestamp=2021-11-23T17:24:23.563, value=23
 2015001                column=StuInfo:S_Name, timestamp=2021-11-23T17:24:23.557, value=Li Lei
 2015001                column=StuInfo:S_Sex, timestamp=2021-11-23T17:24:23.560, value=male
 2015002                column=Contact:Email, timestamp=2021-11-23T17:24:25.203, value=hmm@qq.com
 2015002                column=Grades:123002, timestamp=2021-11-23T17:24:23.582, value=77
 2015002                column=Grades:123003, timestamp=2021-11-23T17:24:23.585, value=99
 2015002                column=StuInfo:S_Age, timestamp=2021-11-23T17:24:23.579, value=22
 2015002                column=StuInfo:S_Name, timestamp=2021-11-23T17:24:23.570, value=Han Meimei
 2015002                column=StuInfo:S_Sex, timestamp=2021-11-23T17:24:23.573, value=female
 2015003                column=Contact:Email, timestamp=2021-11-23T17:24:25.206, value=zs@qq.com
 2015003                column=Grades:123001, timestamp=2021-11-23T17:24:23.595, value=98
 2015003                column=Grades:123003, timestamp=2021-11-23T17:24:23.598, value=95
 2015003                column=StuInfo:S_Age, timestamp=2021-11-23T17:24:23.593, value=24
 2015003                column=StuInfo:S_Name, timestamp=2021-11-23T17:24:23.587, value=Zhang San
 2015003                column=StuInfo:S_Sex, timestamp=2021-11-23T17:24:23.590, value=male
3 row(s)
Took 0.1658 seconds
hbase:012:0> scan 'Students'
ROW                     COLUMN+CELL
 2015001                column=Contact:Email, timestamp=2021-11-23T17:25:14.145, value=lilie@qq.com
 2015001                column=Grades:123001, timestamp=2021-11-23T17:25:12.958, value=86
 2015001                column=Grades:123003, timestamp=2021-11-23T17:25:12.961, value=69
 2015001                column=StuInfo:S_Age, timestamp=2021-11-23T17:25:12.956, value=23
 2015001                column=StuInfo:S_Name, timestamp=2021-11-23T17:25:12.950, value=Li Lei
 2015001                column=StuInfo:S_Sex, timestamp=2021-11-23T17:25:12.953, value=male
 2015002                column=Contact:Email, timestamp=2021-11-23T17:25:14.148, value=hmm@qq.com
 2015002                column=Grades:123002, timestamp=2021-11-23T17:25:12.971, value=77
 2015002                column=Grades:123003, timestamp=2021-11-23T17:25:12.974, value=99
 2015002                column=StuInfo:S_Age, timestamp=2021-11-23T17:25:12.968, value=22
 2015002                column=StuInfo:S_Name, timestamp=2021-11-23T17:25:12.963, value=Han Meimei
 2015002                column=StuInfo:S_Sex, timestamp=2021-11-23T17:25:12.966, value=female
 2015003                column=Contact:Email, timestamp=2021-11-23T17:25:14.151, value=zs@qq.com
 2015003                column=StuInfo:S_Age, timestamp=2021-11-23T17:25:12.981, value=24
 2015003                column=StuInfo:S_Name, timestamp=2021-11-23T17:25:12.976, value=Zhang San
 2015003                column=StuInfo:S_Sex, timestamp=2021-11-23T17:25:12.978, value=male
3 row(s)
Took 0.1485 seconds
hbase:013:0>
```

```
>>Q2 over.
>Q3: Add new colfami and col with data.
Add new colfami:Contact.
put 2015001 | Contact => Email : lilie@qq.com to table Students successfully
put 2015002 | Contact => Email : hmm@qq.com to table Students successfully
put 2015003 | Contact => Email : zs@qq.com to table Students successfully
>>Q3 over.
>Q4: Delete course records of a specific student.
start filtering table 'Students' to find specific student 2015003:
Row 2015003 column family Grades from table Students deleted
>>Q4 over.

Process finished with exit code 0
```



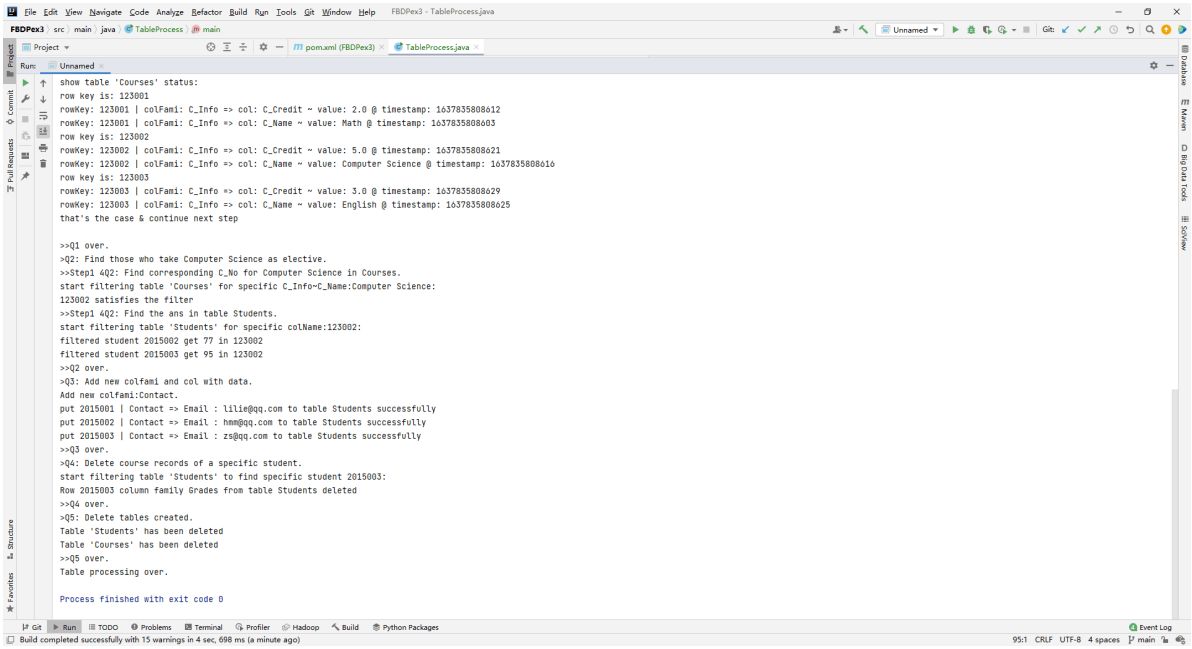
```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Connect to HBASE successfully
>Q1: Create appropriate tables.
>>Step1 4Q1: Create tables Students and Courses.
start create table 'Students'
create table 'Students' successfully
start create table 'Courses'
create table 'Courses' successfully
>>Step2 4Q1: Add data.
put 123001 | C_Info => C_Name : Math to table Courses successfully
put 123001 | C_Info => C_Credit : 2.0 to table Courses successfully
put 123002 | C_Info => C_Name : Computer Science to table Courses successfully
put 123002 | C_Info => C_Credit : 5.0 to table Courses successfully
put 123003 | C_Info => C_Name : English to table Courses successfully
put 123003 | C_Info => C_Credit : 3.0 to table Courses successfully
put 2015001 | StuInfo => S_Name : Li Lei to table Students successfully
put 2015001 | StuInfo => S_Sex : male to table Students successfully
put 2015001 | StuInfo => S_Age : 23 to table Students successfully
put 2015001 | Grades => 123001 : 86 to table Students successfully
put 2015001 | Grades => 123003 : 69 to table Students successfully
put 2015002 | StuInfo => S_Name : Han Meimei to table Students successfully
put 2015002 | StuInfo => S_Sex : female to table Students successfully
put 2015002 | StuInfo => S_Age : 22 to table Students successfully
put 2015002 | Grades => 123002 : 77 to table Students successfully
put 2015002 | Grades => 123003 : 99 to table Students successfully
put 2015003 | StuInfo => S_Name : Zhang San to table Students successfully
put 2015003 | StuInfo => S_Sex : male to table Students successfully
put 2015003 | StuInfo => S_Age : 24 to table Students successfully
put 2015003 | Grades => 123001 : 98 to table Students successfully
put 2015003 | Grades => 123002 : 95 to table Students successfully
>>>Check Step2 4Q1:
show table 'Students' status:
row key is: 2015001
rowKey: 2015001 | colFami: Grades => col: 123001 ~ value: 86 @ timestamp: 1637835808652
rowKey: 2015001 | colFami: Grades => col: 123003 ~ value: 69 @ timestamp: 1637835808656
rowKey: 2015001 | colFami: StuInfo => col: S_Age ~ value: 23 @ timestamp: 1637835808648
rowKey: 2015001 | colFami: StuInfo => col: S_Name ~ value: Li Lei @ timestamp: 1637835808639
```



```
rowKey: 2015001 | colFami: StuInfo => col: S_Name ~ value: Li Lei @ timestamp: 1637835808639
rowKey: 2015001 | colFami: StuInfo => col: S_Sex ~ value: male @ timestamp: 1637835808644
row key is: 2015002
rowKey: 2015002 | colFami: Grades => col: 123002 ~ value: 77 @ timestamp: 1637835808669
rowKey: 2015002 | colFami: Grades => col: 123003 ~ value: 99 @ timestamp: 1637835808672
rowKey: 2015002 | colFami: StuInfo => col: S_Age ~ value: 22 @ timestamp: 1637835808666
rowKey: 2015002 | colFami: StuInfo => col: S_Name ~ value: Han Meimei @ timestamp: 1637835808660
rowKey: 2015002 | colFami: StuInfo => col: S_Sex ~ value: female @ timestamp: 1637835808663
row key is: 2015003
rowKey: 2015003 | colFami: Grades => col: 123001 ~ value: 98 @ timestamp: 1637835808685
rowKey: 2015003 | colFami: Grades => col: 123002 ~ value: 95 @ timestamp: 1637835808687
rowKey: 2015003 | colFami: StuInfo => col: S_Age ~ value: 24 @ timestamp: 1637835808682
rowKey: 2015003 | colFami: StuInfo => col: S_Name ~ value: Zhang San @ timestamp: 1637835808675
rowKey: 2015003 | colFami: StuInfo => col: S_Sex ~ value: male @ timestamp: 1637835808678
that's the case & continue next step

show table 'Courses' status:
row key is: 123001
rowKey: 123001 | colFami: C_Info => col: C_Credit ~ value: 2.0 @ timestamp: 1637835808612
rowKey: 123001 | colFami: C_Info => col: C_Name ~ value: Math @ timestamp: 1637835808603
row key is: 123002
rowKey: 123002 | colFami: C_Info => col: C_Credit ~ value: 5.0 @ timestamp: 1637835808621
rowKey: 123002 | colFami: C_Info => col: C_Name ~ value: Computer Science @ timestamp: 1637835808616
row key is: 123003
rowKey: 123003 | colFami: C_Info => col: C_Credit ~ value: 3.0 @ timestamp: 1637835808629
rowKey: 123003 | colFami: C_Info => col: C_Name ~ value: English @ timestamp: 1637835808625
that's the case & continue next step

>>Q1 over.
>Q2: Find those who take Computer Science as elective.
>>Step1 4Q2: Find corresponding C_No for Computer Science in Courses.
start filtering table 'Courses' for specific C_Info~C_Name:Computer Science:
123002 satisfies the filter
>>Step1 4Q2: Find the ans in table Students.
start filtering table 'Students' for specific colName:123002:
filtered student 2015002 get 77 in 123002
filtered student 2015003 get 95 in 123002
>>Q2 over.
>Q3: Add new colfami and col with data.
Add new colfami:Contact.
```



命令行逻辑

参考教程http://c.biancheng.net/hbase/。命令行代码在hbase_bat.txt文件中。

1. **设计并创建合适的表；**

创建表格：

```
create 'Students','Stulnfo','Grades'
create 'Courses','C_Info'
```

插入数据示例：

```
put 'Courses','123003','C_Info:C_Name','English'
put 'Courses','123003','C_Info:C_Credit','3.0'
put 'Students', '2015001', 'Stulnfo:S_Name', 'Li Lei'
put 'Students', '2015001', 'Stulnfo:S_Sex', 'male'
put 'Students', '2015001', 'Stulnfo:S_Age', '23'
put 'Students', '2015001', 'Grades:123001', '86'
put 'Students', '2015001', 'Grades:123003', '69'
```

查看结果：

```
scan 'Students'
scan 'Courses'
```

2. **查询选修Computer Science的学生的成绩；**

格式：scan '表名', { Filter => "过滤器(比较运算符, '比较器') }。这里由于表格设计，和Java实现时一样，是不方便使用一行解决的，主要是HBase并不支持join或嵌套查询，且这里中间结果只有一项，分成两步、在第二步中显式表现中间结果并没有回避根本问题。在实际应用中确实遇到多个中间结果的，集成Hive使用SQL语言查询是比较合理的解决办法。

```
scan 'Courses', FILTER => "SingleColumnValueFilter('C_Info', 'C_Name', =,
'binary:Computer Science')"
scan 'Students', FILTER => "QualifierFilter(=,'binary:123002')"
```

3. **增加新的列族和新列Contact:Email，并添加数据；**

先添加新列族，不需要disable表：

```
alter 'Students','Contact'
```

插入数据：

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

4. **删除学号为2015003的学生的选课记录；**

```
get 'Students', '2015003', FILTER => "FamilyFilter(=, 'binary:Grades')"
delete 'Students', '2015003', 'Grades:123001'
delete 'Students', '2015003', 'Grades:123002'
```

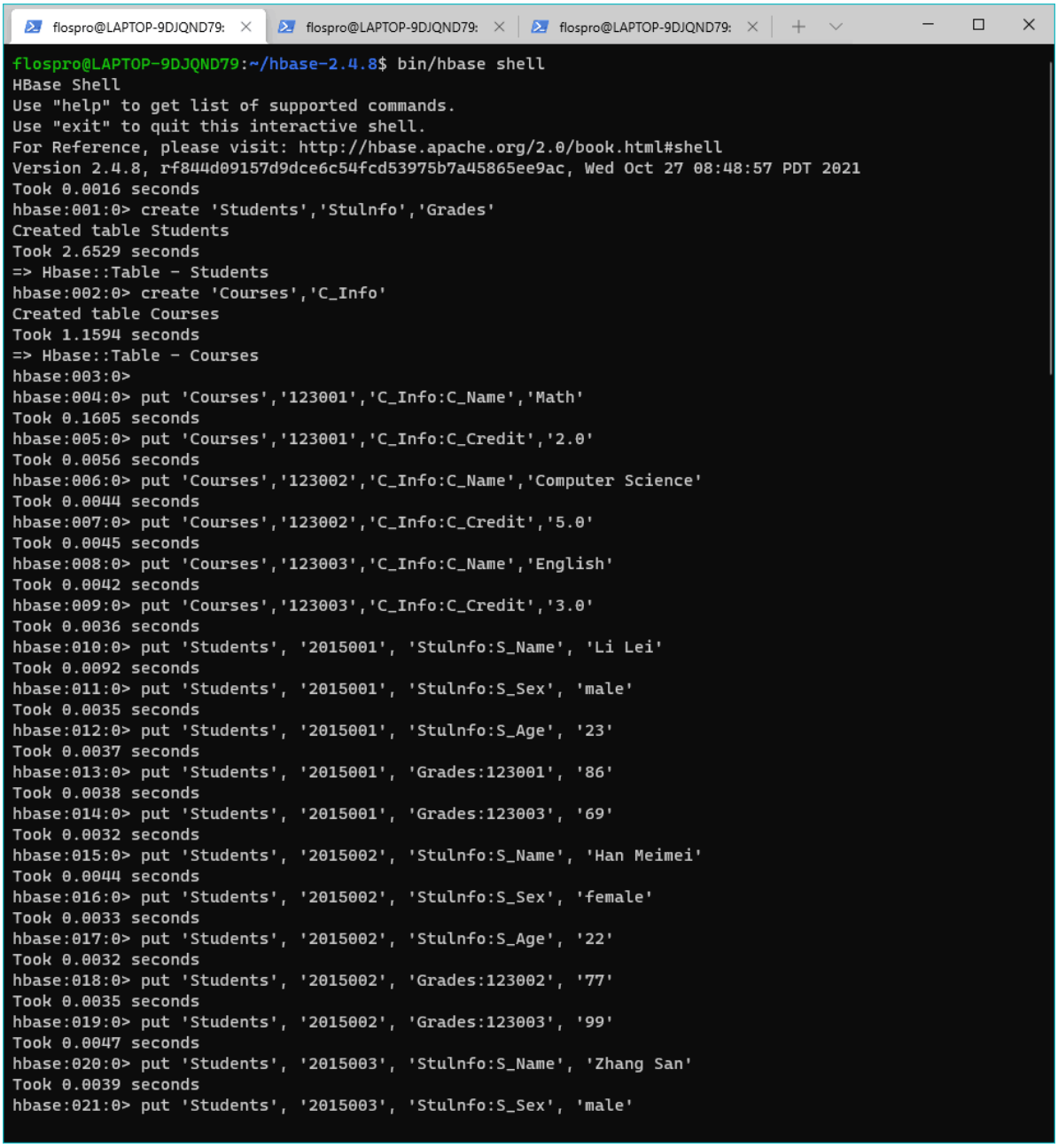
因为hbase中没有shell命令能直接删除指定行的列族信息，所以需要先获取指定行的列族的所有列限定符，然后使用delete一个一个作删除，比Java代码直接对delete对象限制行键和列族进行删除繁琐一些。可以使用scan检查结果是否执行。

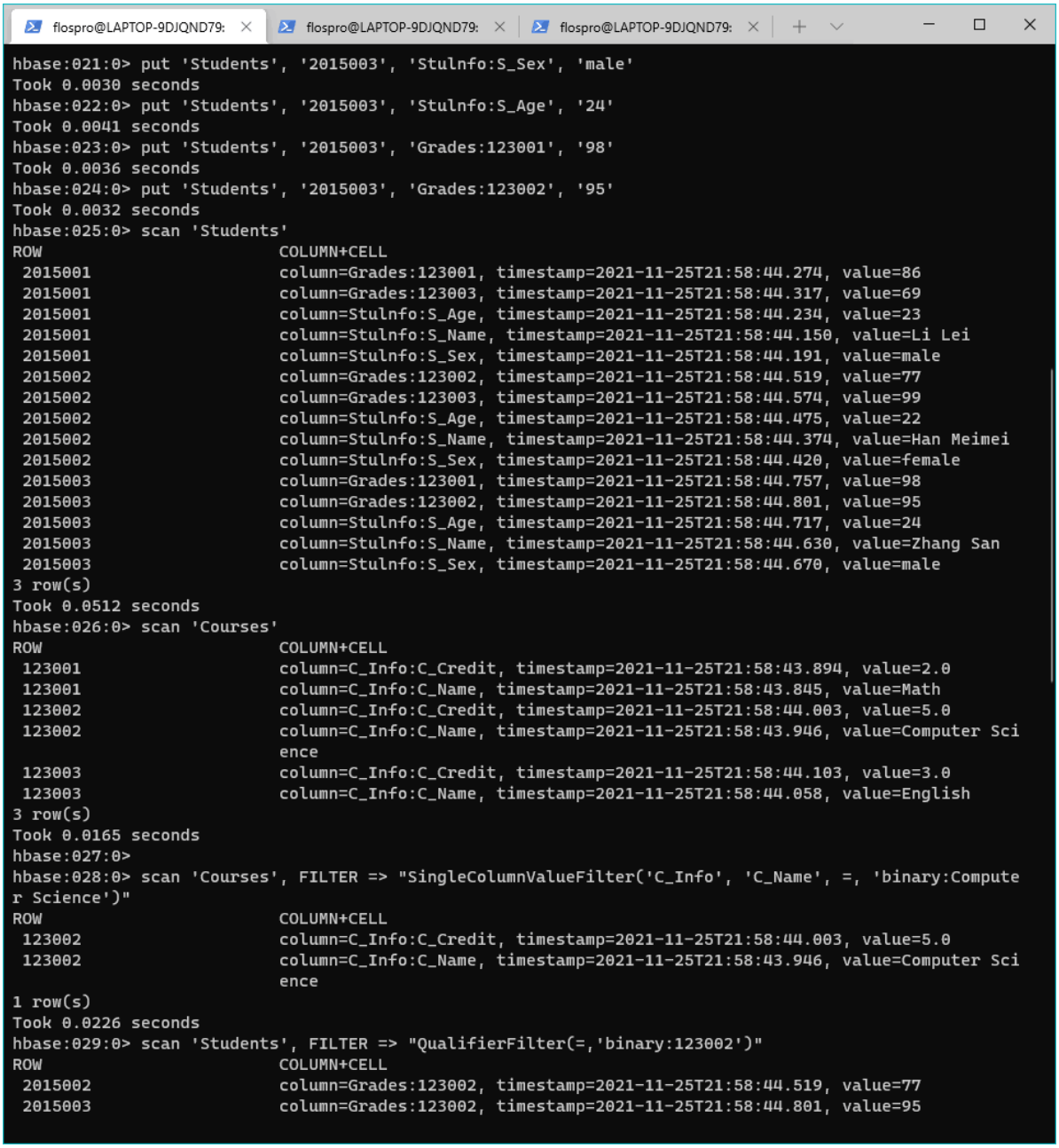
5. **删除所创建的表。**

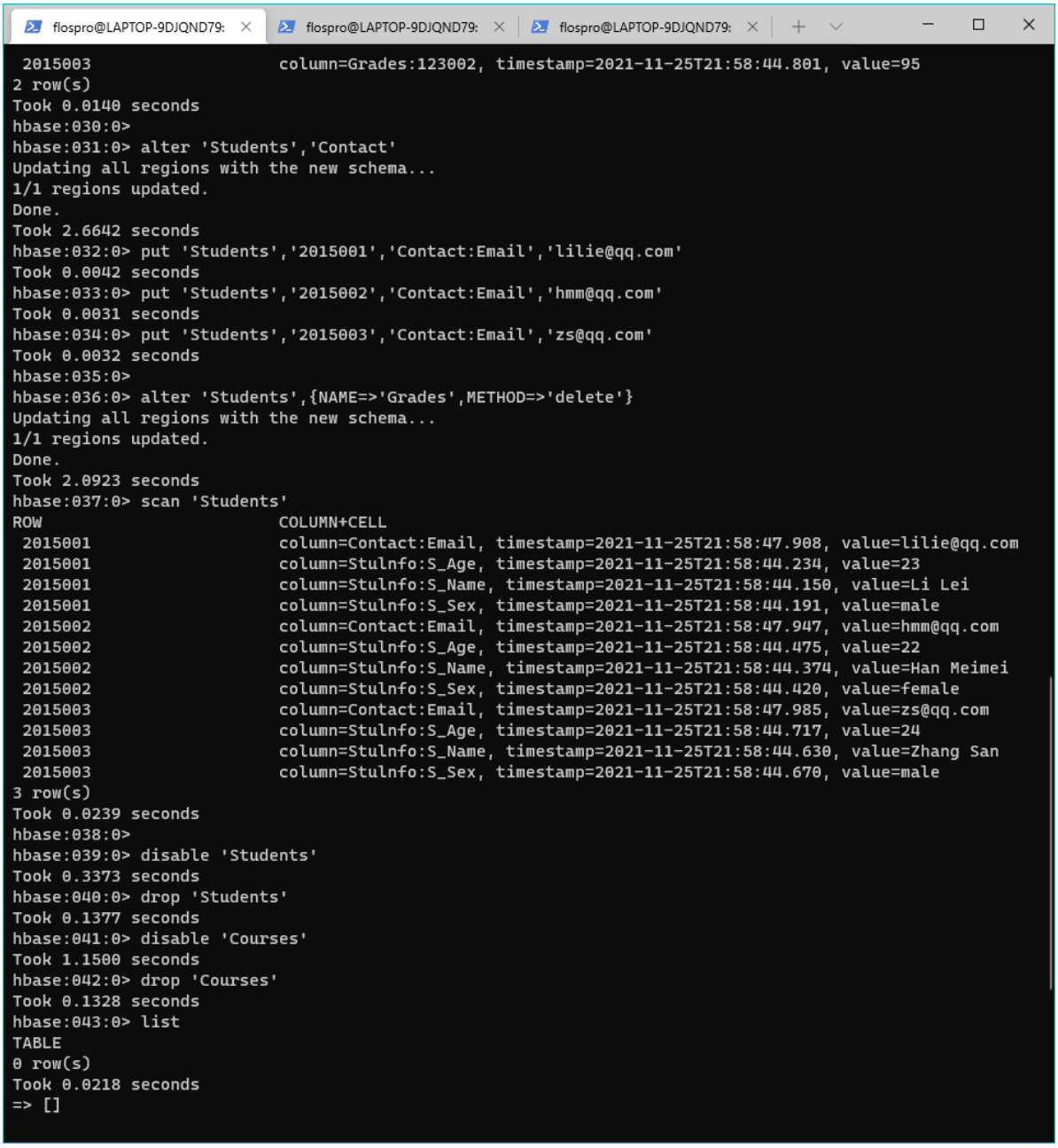
```
disable 'Students'
drop 'Students'
disable 'Courses'
drop 'Courses'
```

先禁用，再删除。可以使用list检查结果是否执行。

以下是运行截图（运行截图中第4问使用的是 alter 'Students',{NAME=>'Grades', METHOD=>'delete'}命令是删除了所有Grades列族，是和题意删除特定行列族不符的失误，这里懒得改了，以代码文件为准）：


flospro@LAPTOP-9DJQND79:
flospro@LAPTOP-9DJQND79:
flospro@LAPTOP-9DJQND79:
flospro@LAPTOP-9DJQND79:~/hbase-2.4.8$ bin/hbase shell
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 2.4.8, rf844d09157d9dce6c54fcd53975b7a45865ee9ac, Wed Oct 27 08:48:57 PDT 2021
Took 0.0016 seconds
hbase:001:0> create 'Students','Stulnfo','Grades'
Created table Students
Took 2.6529 seconds
=> Hbase::Table - Students
hbase:002:0> create 'Courses','C_Info'
Created table Courses
Took 1.1594 seconds
=> Hbase::Table - Courses
hbase:003:0>
hbase:004:0> put 'Courses','123001','C_Info:C_Name','Math'
Took 0.1605 seconds
hbase:005:0> put 'Courses','123001','C_Info:C_Credit','2.0'
Took 0.0056 seconds
hbase:006:0> put 'Courses','123002','C_Info:C_Name','Computer Science'
Took 0.0044 seconds
hbase:007:0> put 'Courses','123002','C_Info:C_Credit','5.0'
Took 0.0045 seconds
hbase:008:0> put 'Courses','123003','C_Info:C_Name','English'
Took 0.0042 seconds
hbase:009:0> put 'Courses','123003','C_Info:C_Credit','3.0'
Took 0.0036 seconds
hbase:010:0> put 'Students', '2015001', 'Stulnfo:S_Name', 'Li Lei'
Took 0.0092 seconds
hbase:011:0> put 'Students', '2015001', 'Stulnfo:S_Sex', 'male'
Took 0.0035 seconds
hbase:012:0> put 'Students', '2015001', 'Stulnfo:S_Age', '23'
Took 0.0037 seconds
hbase:013:0> put 'Students', '2015001', 'Grades:123001', '86'
Took 0.0038 seconds
hbase:014:0> put 'Students', '2015001', 'Grades:123003', '69'
Took 0.0032 seconds
hbase:015:0> put 'Students', '2015002', 'Stulnfo:S_Name', 'Han Meimei'
Took 0.0044 seconds
hbase:016:0> put 'Students', '2015002', 'Stulnfo:S_Sex', 'female'
Took 0.0033 seconds
hbase:017:0> put 'Students', '2015002', 'Stulnfo:S_Age', '22'
Took 0.0032 seconds
hbase:018:0> put 'Students', '2015002', 'Grades:123002', '77'
Took 0.0035 seconds
hbase:019:0> put 'Students', '2015002', 'Grades:123003', '99'
Took 0.0047 seconds
hbase:020:0> put 'Students', '2015003', 'Stulnfo:S_Name', 'Zhang San'
Took 0.0039 seconds
hbase:021:0> put 'Students', '2015003', 'Stulnfo:S_Sex', 'male'


flospro@LAPTOP-9DJQND79:
flospro@LAPTOP-9DJQND79:
flospro@LAPTOP-9DJQND79:
hbase:021:0> put 'Students', '2015003', 'Stulnfo:S_Sex', 'male'
Took 0.0030 seconds
hbase:022:0> put 'Students', '2015003', 'Stulnfo:S_Age', '24'
Took 0.0041 seconds
hbase:023:0> put 'Students', '2015003', 'Grades:123001', '98'
Took 0.0036 seconds
hbase:024:0> put 'Students', '2015003', 'Grades:123002', '95'
Took 0.0032 seconds
hbase:025:0> scan 'Students'
ROW COLUMN+CELL
2015001 column=Grades:123001, timestamp=2021-11-25T21:58:44.274, value=86
2015001 column=Grades:123003, timestamp=2021-11-25T21:58:44.317, value=69
2015001 column=Stulnfo:S_Age, timestamp=2021-11-25T21:58:44.234, value=23
2015001 column=Stulnfo:S_Name, timestamp=2021-11-25T21:58:44.150, value=Li Lei
2015001 column=Stulnfo:S_Sex, timestamp=2021-11-25T21:58:44.191, value=male
2015002 column=Grades:123002, timestamp=2021-11-25T21:58:44.519, value=77
2015002 column=Grades:123003, timestamp=2021-11-25T21:58:44.574, value=99
2015002 column=Stulnfo:S_Age, timestamp=2021-11-25T21:58:44.475, value=22
2015002 column=Stulnfo:S_Name, timestamp=2021-11-25T21:58:44.374, value=Han Meimei
2015002 column=Stulnfo:S_Sex, timestamp=2021-11-25T21:58:44.420, value=female
2015003 column=Grades:123001, timestamp=2021-11-25T21:58:44.757, value=98
2015003 column=Grades:123002, timestamp=2021-11-25T21:58:44.801, value=95
2015003 column=Stulnfo:S_Age, timestamp=2021-11-25T21:58:44.717, value=24
2015003 column=Stulnfo:S_Name, timestamp=2021-11-25T21:58:44.630, value=Zhang San
2015003 column=Stulnfo:S_Sex, timestamp=2021-11-25T21:58:44.670, value=male
3 row(s)
Took 0.0512 seconds
hbase:026:0> scan 'Courses'
ROW COLUMN+CELL
123001 column=C_Info:C_Credit, timestamp=2021-11-25T21:58:43.894, value=2.0
123001 column=C_Info:C_Name, timestamp=2021-11-25T21:58:43.845, value=Math
123002 column=C_Info:C_Credit, timestamp=2021-11-25T21:58:44.003, value=5.0
123002 column=C_Info:C_Name, timestamp=2021-11-25T21:58:43.946, value=Computer Sci
ence
123003 column=C_Info:C_Credit, timestamp=2021-11-25T21:58:44.103, value=3.0
123003 column=C_Info:C_Name, timestamp=2021-11-25T21:58:44.058, value=English
3 row(s)
Took 0.0165 seconds
hbase:027:0>
hbase:028:0> scan 'Courses', FILTER => "SingleColumnValueFilter('C_Info', 'C_Name', =, 'binary:Compute
r Science')"
ROW COLUMN+CELL
123002 column=C_Info:C_Credit, timestamp=2021-11-25T21:58:44.003, value=5.0
123002 column=C_Info:C_Name, timestamp=2021-11-25T21:58:43.946, value=Computer Sci
ence
1 row(s)
Took 0.0226 seconds
hbase:029:0> scan 'Students', FILTER => "QualifierFilter(=,'binary:123002')"
ROW COLUMN+CELL
2015002 column=Grades:123002, timestamp=2021-11-25T21:58:44.519, value=77
2015003 column=Grades:123002, timestamp=2021-11-25T21:58:44.801, value=95


flospro@LAPTOP-9DJQND79:
flospro@LAPTOP-9DJQND79:
flospro@LAPTOP-9DJQND79:
2015003 column=Grades:123002, timestamp=2021-11-25T21:58:44.801, value=95
2 row(s)
Took 0.0140 seconds
hbase:030:0>
hbase:031:0> alter 'Students','Contact'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.6642 seconds
hbase:032:0> put 'Students','2015001','Contact:Email','lilie@qq.com'
Took 0.0042 seconds
hbase:033:0> put 'Students','2015002','Contact:Email','hmm@qq.com'
Took 0.0031 seconds
hbase:034:0> put 'Students','2015003','Contact:Email','zs@qq.com'
Took 0.0032 seconds
hbase:035:0>
hbase:036:0> alter 'Students',{NAME=>'Grades',METHOD=>'delete'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.0923 seconds
hbase:037:0> scan 'Students'
ROW COLUMN+CELL
2015001 column=Contact:Email, timestamp=2021-11-25T21:58:47.908, value=lilie@qq.com
2015001 column=Stulnfo:S_Age, timestamp=2021-11-25T21:58:44.234, value=23
2015001 column=Stulnfo:S_Name, timestamp=2021-11-25T21:58:44.150, value=Li Lei
2015001 column=Stulnfo:S_Sex, timestamp=2021-11-25T21:58:44.191, value=male
2015002 column=Contact:Email, timestamp=2021-11-25T21:58:47.947, value=hmm@qq.com
2015002 column=Stulnfo:S_Age, timestamp=2021-11-25T21:58:44.475, value=22
2015002 column=Stulnfo:S_Name, timestamp=2021-11-25T21:58:44.374, value=Han Meimei
2015002 column=Stulnfo:S_Sex, timestamp=2021-11-25T21:58:44.420, value=female
2015003 column=Contact:Email, timestamp=2021-11-25T21:58:47.985, value=zs@qq.com
2015003 column=Stulnfo:S_Age, timestamp=2021-11-25T21:58:44.717, value=24
2015003 column=Stulnfo:S_Name, timestamp=2021-11-25T21:58:44.630, value=Zhang San
2015003 column=Stulnfo:S_Sex, timestamp=2021-11-25T21:58:44.670, value=male
3 row(s)
Took 0.0239 seconds
hbase:038:0>
hbase:039:0> disable 'Students'
Took 0.3373 seconds
hbase:040:0> drop 'Students'
Took 0.1377 seconds
hbase:041:0> disable 'Courses'
Took 1.1500 seconds
hbase:042:0> drop 'Courses'
Took 0.1328 seconds
hbase:043:0> list
TABLE
0 row(s)
Took 0.0218 seconds
=> []