

《金融大数据处理技术》实验3报告

嵇泽同 191870068

《金融大数据处理技术》实验3报告

- 一、环境设置
- 二、HBase的安装和配置
- 三、HBase表格设计
- 四、JAVA编程实现
 - ①创建表
 - ②插入数据
 - ③查询选修Computer Science的学生的成绩
 - ④增加新的列族Contact和新列S_Email
 - ⑤删除学号为2015003的学生的选课记录
 - ⑥删除所创建的表
- 五、HBase Shell实现
 - ①创建表
 - ②插入学生信息
 - ③插入课程及选课成绩信息
 - ④查询选修Computer Science的学生的成绩
 - ⑤增加新的列族和新列Contact:Email, 并添加数据
 - ⑥删除学号为2015003的学生的选课记录
 - ⑦删除所创建的表
- 六、遇到的主要困难
 - ①在shell和IDEA中执行指令/代码报错
 - ②在已有表新建列族时报错"read only"

一、环境设置

Ubuntu16.04 + Hadoop3.2.2 + HBase2.4.8 + maven + IDEA

pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>hbase</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
```

```

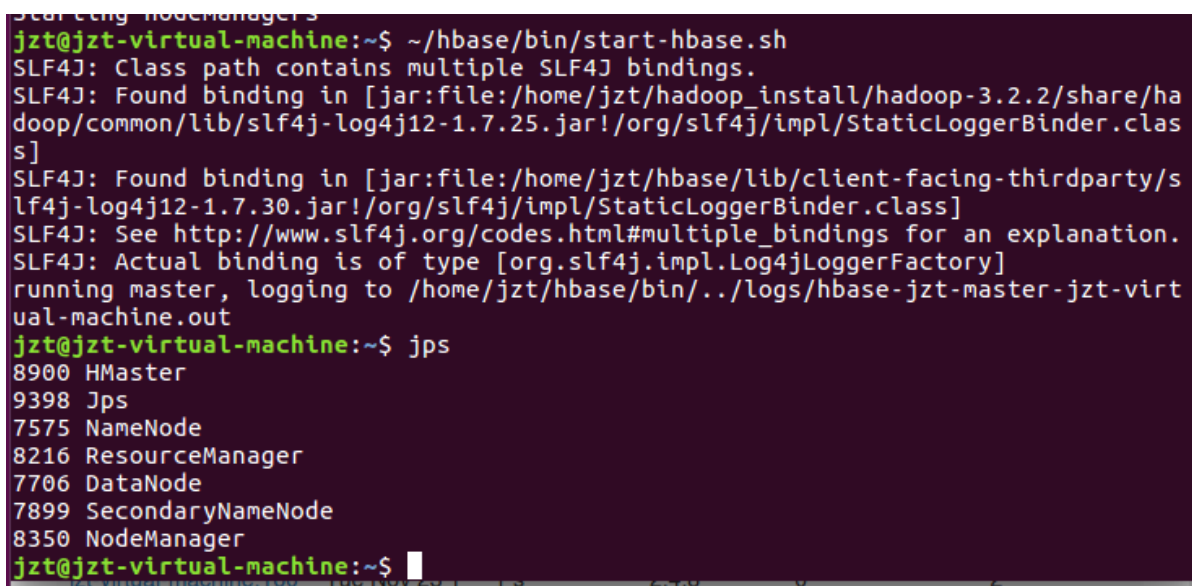
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-client</artifactId>
        <version>3.2.2</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>3.2.2</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-hdfs</artifactId>
        <version>3.2.2</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hbase</groupId>
        <artifactId>hbase-it</artifactId>
        <version>2.4.8</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hbase</groupId>
        <artifactId>hbase-client</artifactId>
        <version>2.4.8</version>
    </dependency>
</dependencies>
</project>

```

二、HBase的安装和配置

HBase的安装和配置按照教程上的步骤执行，较为顺利。

单机模式：



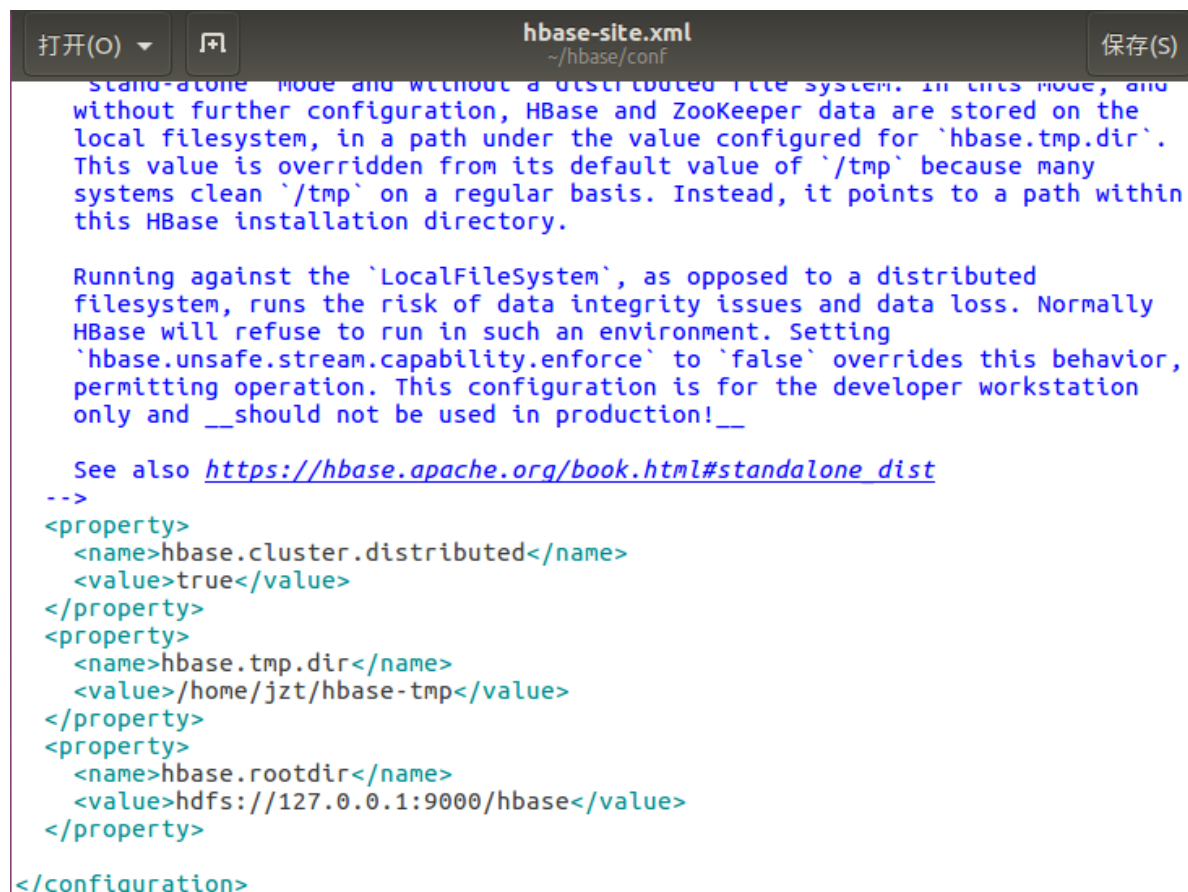
```

Starting nodeManagers
jzt@jzt-virtual-machine:~$ ~/hbase/bin/start-hbase.sh
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jzt/hadoop_install/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:/home/jzt/hbase/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 /home/jzt/hbase/bin/../logs/hbase-jzt-master-jzt-virtual-machine.out
jzt@jzt-virtual-machine:~$ jps
8900 HMaster
9398 Jps
7575 NameNode
8216 ResourceManager
7706 DataNode
7899 SecondaryNameNode
8350 NodeManager
jzt@jzt-virtual-machine:~$

```

伪分布模式：

hbase-site.xml文件配置:



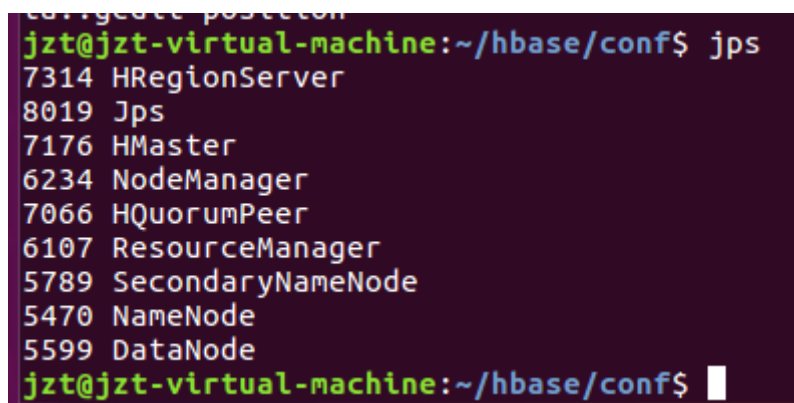
The screenshot shows a text editor window titled 'hbase-site.xml' with the path '~/.hbase/conf'. The file contains XML configuration for HBase. It includes a comment about standalone mode and a warning about running against the LocalFileSystem. It also provides a link to the HBase book for standalone distribution. The XML configuration includes properties for hbase.cluster.distributed, hbase.tmp.dir, hbase.rootdir, and hdfs://127.0.0.1:9000/hbase.

```
<!-- Stand-alone mode and without a distributed file system. In this mode, and without further configuration, HBase and ZooKeeper data are stored on the local filesystem, in a path under the value configured for 'hbase.tmp.dir'. This value is overridden from its default value of '/tmp' because many systems clean '/tmp' on a regular basis. Instead, it points to a path within this HBase installation directory.

Running against the 'LocalFileSystem', as opposed to a distributed filesystem, runs the risk of data integrity issues and data loss. Normally HBase will refuse to run in such an environment. Setting 'hbase.unsafe.stream.capability.enforce' to 'false' overrides this behavior, permitting operation. This configuration is for the developer workstation only and __should not be used in production!__

See also https://hbase.apache.org/book.html#standalone_dist -->
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.tmp.dir</name>
  <value>/home/jzt/hbase-tmp</value>
</property>
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://127.0.0.1:9000/hbase</value>
</property>
</configuration>
```

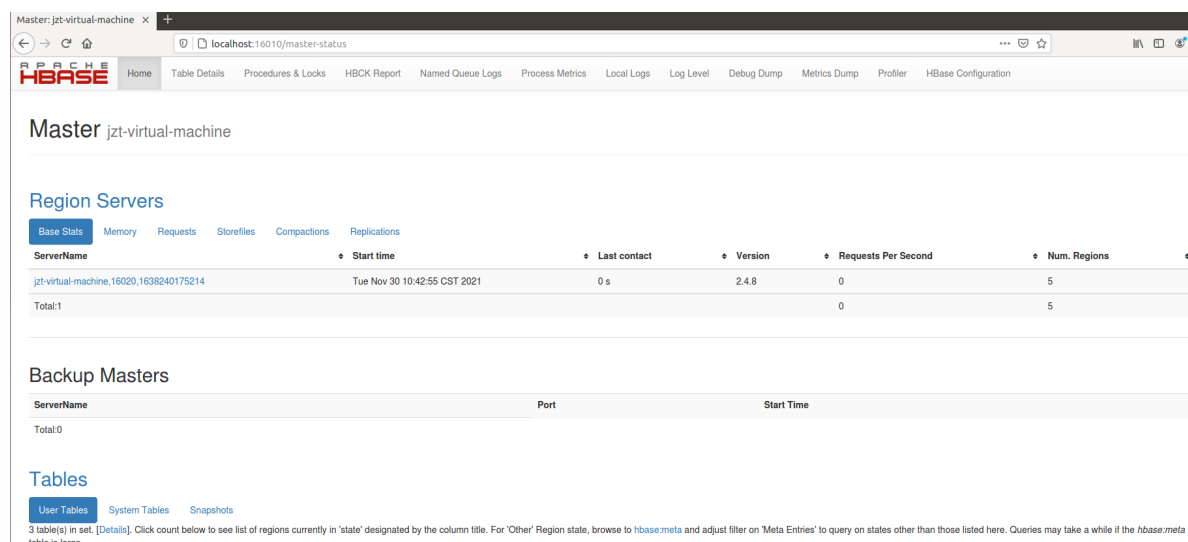
jps:



The screenshot shows a terminal window with the command 'jps' executed. The output lists several Java processes running on the machine, including HRegionServer, Jps, HMaster, NodeManager, HQuorumPeer, ResourceManager, SecondaryNameNode, NameNode, and DataNode.

```
jzt@jzt-virtual-machine:~/hbase/conf$ jps
7314 HRegionServer
8019 Jps
7176 HMaster
6234 NodeManager
7066 HQuorumPeer
6107 ResourceManager
5789 SecondaryNameNode
5470 NameNode
5599 DataNode
jzt@jzt-virtual-machine:~/hbase/conf$
```

localhost:16010:



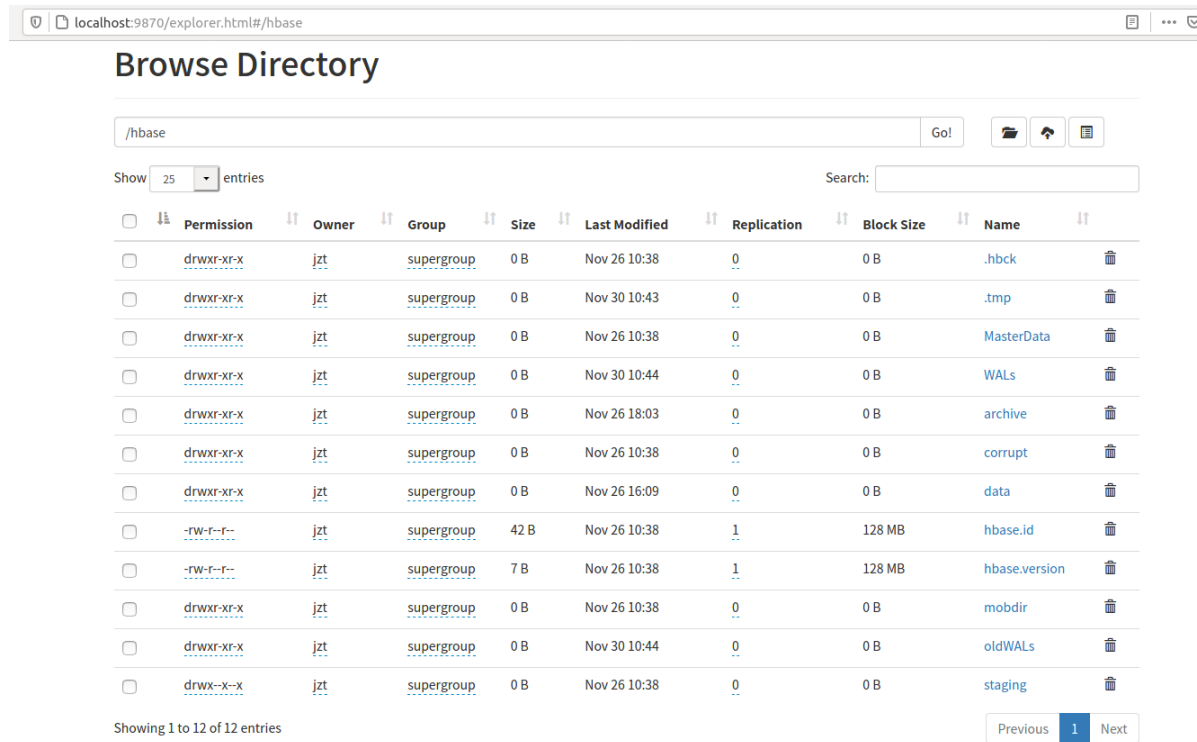
The screenshot shows the HBase web interface for the Master node. The page displays the status of the Master and the Region Servers. The Region Servers table shows one server running on localhost:16020. The Backup Masters table is empty. The Tables section shows a list of tables in the 'state' designated by the column title.

ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
jzt-virtual-machine,16020,1638240175214	Tue Nov 30 10:42:55 CST 2021	0 s	2.4.8	0	5
Total:1				0	5

ServerName	Port	Start Time
Total:0		

3 table(s) in set. [Details]. Click count below to see list of regions currently in 'state' designated by the column title. For 'Other' Region state, browse to [hbase:meta](#) and adjust filter on 'Meta Entries' to query on states other than those listed here. Queries may take a while if the [hbase:meta](#) table is large.

hdfs:



三.HBase表格设计

本次试验中，我将所给关系数据库中的三张表合成为HBase中的一张表，格式如下：

Row Key	Basic Info				Contact	Math				Computer Science				English			
	S_No	S_Name	S_Sex	S_Age	S_Email	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	lilei@qq.com	123001	Math	2.0	86					123003	English	3.0	69
2015002	2015002	Han Meimei	female	22	hmmi@qq.com					123002	Computer Science	5.0	77	123003	English	3.0	99
2015003	2015003	Zhan San	male	24	zs@qa.com	123001	Math	2.0	98	123002	Computer Science	5.0	95				

关于行键（Row Key）的设计：

由于本次实验中数据较少，而且以学号作为行键本身也不怎么会出现数据热点问题，因此本次实验中我没有对行键采取特殊的处理（如加盐、哈希、颠倒等），直接以学号作为行键。

关于列族的设计：

本次实验中我将学生的基础个人信息放在列族“Basic Info”中，该列族后续还可以添加其他基础信息列；

将学生的联系方式放在列族“Contact”中，该列族除了目前已有的邮箱列，后续还可以添加如电话号码列、地址列等等；

将每门课程单独设置为一个列族，该列族存储该课程的基础信息和该学生与该课程间的关联，如果该学生没有选修该课程，那么该行对应的该列族数据为空，比较易于理解。

其实当完成实验后，回过头来审视这张表时，我发觉关于课程的列族设计其实似乎欠缺合理性：

目前的课程只有三门，将每门课程都设置为一个列族可以轻易实现；但实际中，一个学校的所有课程门数是非常大的，如果将每门课程都设置为一个列族，那么将造成非常多的列族和空列族，而HBase的列族是不建议过多的，因此当数据量增大时，该表结构的设计是不合理的。可以考虑转为“开辟一个“Score”列族、下面存储该学生选修的所有课程的分数的设计，或者设计多个表，不要将所有数据都放在一个表里。由于时间紧张且本人缺乏此类设计经验，因此在本次实验中没有较好地设计出这类表格，也是本次实验的一个不足和缺憾。

四、JAVA编程实现

在JAVA编程实现前，我先上网查找了一些常用类和常用方法，如下所示：

HBaseAdmin类的主要方法：

返回值	方法
void	addColumn(String tableName, HColumnDescriptor column) 向一个已存在的表添加列
static void	checkHBaseAvailable(HBaseConfiguration conf) 检查 HBase 是否处于运行状态
void	closeRegion(String regionName, String serverName) 关闭 Region
void	createTable(HTableDescriptor desc) 创建表
void	deleteTable(String tableName) 删除表
void	disableTable(String tableName) 使表无效
void	enableTable(String tableName) 使表有效
boolean	tableExists(String tableName) 检查表是否存在
HTableDescriptor	listTables() 列出所有的表项
void	abort(String why, Throwable e) 终止服务器或客户端
boolean	balancer() 调用 balancer 进行负载均衡

HTable类的主要方法：

Void	delete(Delete delete) 删除指定的单元格或行
boolean	exists(Get get) 检查 Get 对象指定的列是否存在
Result	get(Get get) 从指定行的某些单元格中取出相应的值
void	put (Put put) 向表中添加值
ResultScanner	getScanner(byte[] family) getScanner(byte[] family, byte[] qualifier) getScanner(Scan scan) 获得 ResultScanner 实例
HTableDescriptor	getTableDescriptor() 获得当前表格的 HTableDescriptor 实例
byte[]	getTableName() 获得当前表格的名字

HTableDescriptor类的主要方法：

返回值	方法
void	addFamily(HColumnDescriptor family) 添加列族
Set<byte[]>	getFamilies() 返回表中所有列族的名字
byte[]	getName() 返回表的名字
void	setName(byte[] name) 设置表的名字
byte[]	getValue(byte[] key) 获得某个属性的值
HColumnDescriptor	removeFamily(byte[] column) 删除某个列族
void	setValue(byte[] key, byte[] value) 设置属性的值

HColumnDescriptor类的主要方法：

返回值	方法
byte[]	getName() 获取列族的名字
byte[]	getValue(byte[] key) 获得某列单元格的值
void	setValue(byte[] key, byte[] value) 设置某列单元格的值

Put类的主要方法：

返回值	方法
Put	add(byte[] family, byte[] qualifier, byte[] value) 将指定的列族、列限定符、对应的值添加到 Put 实例中
List<KeyValue>	get(byte[] family, byte[] qualifier) 获取列族和列限定符指定列中所有行的值
boolean	has(byte[] family, byte[] qualifier) 检查列族和列限定符指定列是否存在
boolean	has(byte[] family, byte[] qualifier, byte[] value) 检查列族和列限定符指定列中是否存在指定的 value

Get类的主要方法：

返回值	方法
Get	addColumn(byte[] family, byte[] qualifier) 根据列族和列限定符获得对应的列
Get	addColumn(byte[] family) 根据列族获得所有的列
Get	setFilter(Filter filter) 为获得具体的列，设置相应的过滤器

Result类的主要方法：

返回值	方法
boolean	containsColumn(byte[] family, byte[] qualifier) 检查是否包含列族和列限定符指定的列
List<KeyValue>	getColumn(byte[] family, byte[] qualifier) 获得列族和列限定符指定的列所有行的值
NavigableMap<byte[], byte[]>	getFamilyMap(byte[] family) 根据列族获得包含列限定符和值的所有行的键值对
byte[]	getValue(byte[] family, byte[] qualifier) 获得列族和列限定符指定的单元格的最新值

ResultScanner类的主要方法：

返回值	方法
void	close() 关闭 scanner 并释放相应的资源
Result	next() 获得下一个 Result 实例

Scan类的主要方法：

返回值	方法
Scan	addFamily(byte[] family) 限定需要查找的列族
Scan	addColumn(byte[] family, byte[] qualifier) 限定列族和列限定符指定的列
Scan	setMaxVersions() setMaxVersions(int maxVersions) 限定最大的版本个数。如果不带任何参数调用 setMaxVersions，表示取得所有的版本。如果不调用 setMaxVersions，只会取得最新的版本
Scan	setTimeRange(long minStamp, long maxStamp) 限定最大的时间戳和最小的时间戳，只有在此范围内的单元格才能被获取
Scan	setFilter(Filter filter) 指定 Filter 过滤掉不需要的数据
Scan	setStartRow(byte[] startRow) 限定开始的行，否则从表头开始
Scan	setStopRow(byte[] stopRow) 限定结束的行（不含此行）
void	setBatch(int batch) 限定最多返回的单元格数目。用于防止返回过多的数据，导致 OutOfMemory 错误

本次试验中，我将对于HBase的操作封装在一个类Hbase中：


```

public class Hbase {
    public static Configuration conf;
    public static Connection connection;
    public static Admin admin;
    public void getconnect() throws IOException {
        conf=HBaseConfiguration.create();
        conf.set("hbase.zookeeper.quorum", "127.0.0.1");
        conf.set("hbase.zookeeper.property.clientPort","2181");
        try{
            connection=ConnectionFactory.createConnection(conf);
            admin = connection.getAdmin();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // 根据表名和列族名创建表
    public void createtable(String tableName, String... columnFamily) throws IOException {...}

    // 判断表是否存在
    public boolean existTable(String tableName) throws Exception {...}

    // disable表
    public void disableTable(String tableName) throws Exception {...}

    // drop表
    public void dropTable(String tableName) throws Exception {...}

    // 增加列族
    public void addColumnFamily(String tableName, String addColumn) throws Exception {...}

    /**添加数据 ...*/
    public void putData(String tableName, String rowKey, String familyName, String columnName, String value)
        //转化为表名

```

该类具有属性conf、connection和admin，用来绑定指定的hbase，当创建该类的实例后，通过getconnect()方法来获取并设置这些属性的值。之后该实例就与hbase绑定，通过该类的其他自定义操作方法即可对hbase中的表进行各类操作。

①创建表

```
test.createtable("student","BasicInfo", "Math", "Computer Science", "English");
```

```

// 根据表名和列族名创建表
public void createtable(String tableName, String... columnFamily) throws
IOException{
    TableName name = TableName.valueOf(tableName);
    if(admin.tableExists(name)) {
        admin.disableTable(name);
        admin.deleteTable(name);
        System.out.println(tableName.toString() + " is exist,delete it");
    }
    HTableDescriptor tb = new HTableDescriptor(name);
    for(int i = 0;i <columnFamily.length;i++) {
        HColumnDescriptor family = new HColumnDescriptor(columnFamily[i]);
        tb.addFamily(family);
    }
    admin.createTable(tb);
}

```

该方法接收表名和列族名，先判断该表是否存在，若存在则先删除该表，然后创建HTableDescriptor实例，并调用admin.createTable()方法实现表的创建，较为简单。

②插入数据

```
// 插入学生基础信息
test.putData("student", "2015001", "BasicInfo", "S_No", "2015001");
test.putData("student", "2015001", "BasicInfo", "S_Name", "Li Lei");
test.putData("student", "2015001", "BasicInfo", "S_Sex", "male");
test.putData("student", "2015001", "BasicInfo", "S_Age", "23");
.....

// 插入课程信息
test.putData("student", "2015001", "Math", "C_No", "123001");
test.putData("student", "2015001", "Math", "C_Name", "Math");
test.putData("student", "2015001", "Math", "C_Credit", "2.0");
test.putData("student", "2015001", "English", "C_No", "123003");
test.putData("student", "2015001", "English", "C_Name", "English");
test.putData("student", "2015001", "English", "C_Credit", "3.0");
.....

// 插入学生课程成绩
test.putData("student", "2015001", "Math", "SC_Score", "86");
test.putData("student", "2015001", "English", "SC_Score", "69");
test.putData("student", "2015002", "Computer Science", "SC_Score", "77");
test.putData("student", "2015002", "English", "SC_Score", "99");
test.putData("student", "2015003", "Math", "SC_Score", "98");
test.putData("student", "2015003", "Computer Science", "SC_Score", "95");
```

```
public void putData(String tableName, String rowKey, String familyName, String
columnName, String value) throws Exception {
    //转化为表名
    TableName name = TableName.valueOf(tableName);
    //添加数据之前先判断表是否存在，不存在的话先创建表
    if(admin.tableExists(name)){
    }else {
        //根据表名创建表结构
        HTableDescriptor tableDescriptor = new HTableDescriptor(name);
        //定义列簇的名字
        HColumnDescriptor columnFamilyName = new HColumnDescriptor(familyName);
        tableDescriptor.addFamily(columnFamilyName);
        admin.createTable(tableDescriptor);
    }

    Table table = connection.getTable(name);
    Put put = new Put(rowKey.getBytes());

    put.addColumn(familyName.getBytes(), columnName.getBytes(),
value.getBytes());
    table.put(put);
}
```

该方法接收要插入的表名、行键、列族名、列限定符和要插入的值，通过put实例的addColumn方法将这些参数传入，然后调用Table.put()方法将该值插入，也较为简单。

③查询选修Computer Science的学生的成绩

```
// 查询选修Computer Science的学生的成绩
List<String> filterStrList = new ArrayList<>();
filterStrList.add("Computer Science,C_Name,Computer Science");
ResultScanner results = test.getResultByFilter("student", filterStrList);
for (Result result=results.next(); result!=null; result=results.next()) {
    List<Cell> cells = result.listCells();
    for (Cell cell:cells){
        String family = new
String(cell.getFamilyArray(),cell.getFamilyOffset(),cell.getFamilyLength(),"UTF-
8");
        String qualifier = new
String(cell.getQualifierArray(),cell.getQualifierOffset(),cell.getQualifierLengt
h(),"UTF-8");
        String value = new
String(cell.getValueArray(),cell.getValueOffset(),cell.getValueLength(),"UTF-
8");
        if (family.equals("BasicInfo") && qualifier.equals("S_No")) {
            System.out.println("S_No: " + value);
        }
        else if (family.equals("BasicInfo") && qualifier.equals("S_Name")) {
            System.out.println("S_Name: " + value);
        }
        else if (family.equals("Computer Science") &&
qualifier.equals("SC_Score")){
            System.out.println("SC_Score: " + value);
        }
    }
}
```

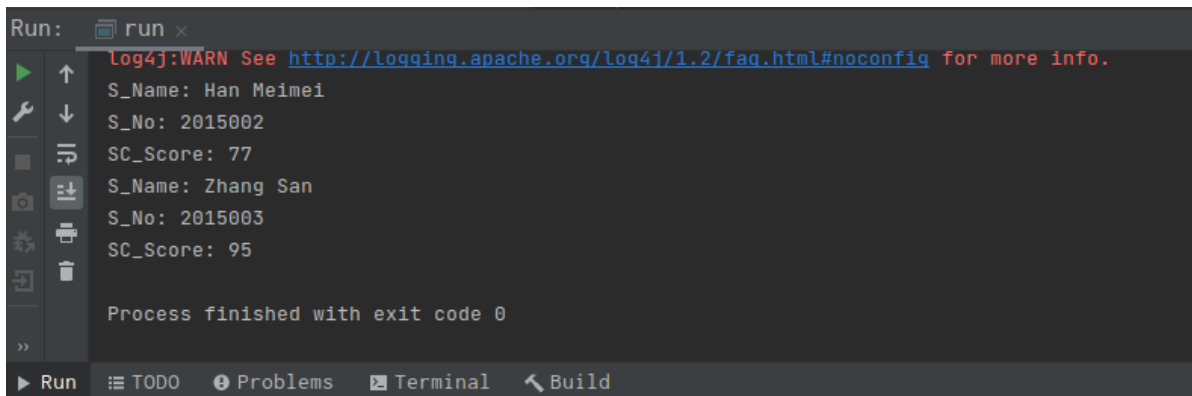
```
public ResultScanner getResultByFilter(String tableName, List<String>
filterStrList) throws Exception{
    ResultScanner result;
    TableName name = TableName.valueOf(tableName);
    if (admin.tableExists(name)) {
        Table table = connection.getTable(name);
        FilterList filterList = new FilterList();
        Scan scan = new Scan();
        for(String filterStr: filterStrList){
            String[] conditions = filterStr.split(",");
            SingleColumnValueFilter filter = new
SingleColumnValueFilter(conditions[0].getBytes(), conditions[1].getBytes(),
CompareFilter.CompareOp.EQUAL, conditions[2].getBytes());
            filter.setFilterIfMissing(true);
            filterList.addFilter(filter);
            s1.addColumn(Bytes.toBytes(s[0]), Bytes.toBytes(s[1]));
        }
        scan.setFilter(filterList);
        result = table.getScanner(scan);
    }
    else {
        result = null;
    }
}
```

```
return result;
}
```

查询操作相对较为复杂，网上关于filter的介绍和资料也很多、五花八门、包含许多高级用法。针对此次实验，我选取了其中一种较为简单适用的通过SingleColumnValueFilter来查询的方法。

SingleColumnFilter实例通过传入指定列族、列限定符、比较运算符、比较值来构建，该filter能够对于每一行，检验该行对应列下cell的值是否满足筛选条件，若满足，则记录该行并返回。值得一提的是，若某一行对应的列下没有数据，则默认情况下该filter也会记录该行并返回，在本次实验的数据中，这导致查询结果会返回所有行（因为选修了该课程的学生自然会被返回，而没选修该课程的学生，由于该列数据为空，因此也会被返回）。通过SingleColumnFilter.setFilterIfMissing(true)方法可以对其进行设置，使得不返回该cell没有值的行，此次我就是采取了这种方法。

此外，值得一提的是，设置filter、scan并通过Table.getScanner(scan)获得结果后，得到的返回值result是一个ResultScanner类的实例，还需要对该实例进行一定的解析操作才能获得想要的字符串值。该解析方法在此次实验中我在网上没有找到能直接适用的，可能是由于Hbase版本不同导致接口变化较大。所以我就通过查看该实例存在的方法、查看源代码等方式一步步摸索，最终成功输出了想要的格式：



```
Run: run x
Log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
S_Name: Han Meimei
S_No: 2015002
SC_Score: 77
S_Name: Zhang San
S_No: 2015003
SC_Score: 95

Process finished with exit code 0
```

④增加新的列族Contact和新列S_Email

```
// 增加新的列族Contact和新列S_Email
test.addColumnFamily("student", "Contact");
test.putData("student", "2015001", "Contact", "S_Email", "lilei@qq.com");
test.putData("student", "2015002", "Contact", "S_Email", "hmm@qq.com");
test.putData("student", "2015003", "Contact", "S_Email", "zs@qq.com");
```

```
// 增加列族
public void addColumnFamily(String tableName, String addColumn) throws Exception
{
    //转化为表名
    TableName name = TableName.valueOf(tableName);
    //判断表是否存在
    if(admin.tableExists(name)) {
        //判断表是否可用状态
        boolean tableEnabled = admin.isTableEnabled(name);
        if(tableEnabled) {
            //使表变成不可用
            admin.disableTable(name);
        }
        //根据表名得到表
```

```

        ColumnFamilyDescriptor cfd =
ColumnFamilyDescriptorBuilder.newBuilder(addColumn.getBytes()).build();
        admin.addColumnFamily(name,cfd);
        admin.enableTable(name);
    }else {
        System.out.println("table不存在");
    }
}

```

该操作我是通过新建ColumnFamilyDescriptor实例，然后调用admin.addColumnFamily()方法实现列族的新建。数据插入方法则同上。看起来整体比较简单，但其中我也走了一些弯路，遇到了一些困难，这些在本报告的后续所遇到的困难部分给出。

⑤删除学号为2015003的学生的选课记录

```

// 删除学号为2015003的学生选课记录
test.deleteColumn("student","2015003","Math");
test.deleteColumn("student","2015003","Computer Science");
test.deleteColumn("student","2015003","English");

```

```

public void deleteColumn(String tableName, String rowKey, String familyName)
throws Exception {
    TableName name = TableName.valueOf(tableName);
    if(admin.tableExists(name)) {
        Table table = connection.getTable(name);
        Delete delete = new Delete(rowKey.getBytes());
        delete.addFamily(familyName.getBytes());
        table.delete(delete);
    }else {
        System.out.println("table不存在");
    }
}

```

该操作同样较为简单，先创建Delete实例，然后调用Table.delete()方法即可。

⑥删除所创建的表

```

// 删除所创建的表
test.dropTable("student");

```

```

public void dropTable(String tableName) throws Exception {
    //转化为表名
    TableName name = TableName.valueOf(tableName);
    //判断表是否存在
    if(admin.tableExists(name)) {
        //判断表是否处于可用状态
        boolean tableEnabled = admin.isTableEnabled(name);
        if(tableEnabled) {
            //使表变成不可用状态
            admin.disableTable(name);
        }
        //删除表
        admin.deleteTable(name);
    }
}

```

```
//判断表是否存在
if(admin.tableExists(name)) {
    System.out.println("删除失败");
}else {
    System.out.println("删除成功");
}
}else {
    System.out.println("table不存在");
}
```

该操作同样较为简单，先调用API disable要删除的表，再delete即可。细节是可以在删除后通过检验该表是否存在来判断删除是否成功。

五、HBase Shell实现

①创建表

```
create 'student','BasicInfo','Math','Computer Science','English'
```

```
hbase:001:0> create 'student','BasicInfo','Math','Computer Science','English'
Created table student
Took 6.4315 seconds
=> Hbase::Table - student
hbase:002:0> list
TABLE
boys
course
sc
student
4 row(s)
Took 0.0918 seconds
=> ["boys", "course", "sc", "student"]
hbase:003:0>
```

②插入学生信息

```
put 'student', '2015001', 'BasicInfo:S_No', '2015001'
put 'student', '2015001', 'BasicInfo:S_Name', 'Li Lei'
put 'student', '2015001', 'BasicInfo:S_Sex', 'male'
put 'student', '2015001', 'BasicInfo:S_Age', '23'

put 'student', '2015002', 'BasicInfo:S_No', '2015002'
put 'student', '2015002', 'BasicInfo:S_Name', 'Han Meimei'
put 'student', '2015002', 'BasicInfo:S_Sex', 'female'
put 'student', '2015002', 'BasicInfo:S_Age', '22'

put 'student', '2015003', 'BasicInfo:S_No', '2015003'
put 'student', '2015003', 'BasicInfo:S_Name', 'Zhang San'
put 'student', '2015003', 'BasicInfo:S_Sex', 'male'
put 'student', '2015003', 'BasicInfo:S_Age', '24'
```

插入结果:

```

hbase:019:0> scan 'student'
ROW COLUMN+CELL
2015001 column=BasicInfo:S_Age, timestamp=2021-11-30T00:49:09.495,
value=23
2015001 column=BasicInfo:S_Name, timestamp=2021-11-30T00:48:53.392,
value=Li Lei
2015001 column=BasicInfo:S_No, timestamp=2021-11-30T00:49:31.690,
value=2015001
2015001 column=BasicInfo:S_Sex, timestamp=2021-11-30T00:49:02.674,
value=male
2015002 column=BasicInfo:S_Age, timestamp=2021-11-30T00:50:18.446,
value=22
2015002 column=BasicInfo:S_Name, timestamp=2021-11-30T00:49:59.190,
value=Han Meimei
2015002 column=BasicInfo:S_No, timestamp=2021-11-30T00:49:47.725,
value=2015002
2015002 column=BasicInfo:S_Sex, timestamp=2021-11-30T00:50:09.775,
value=female
2015003 column=BasicInfo:S_Age, timestamp=2021-11-30T00:51:15.113,
value=24
2015003 column=BasicInfo:S_Name, timestamp=2021-11-30T00:50:54.614,
value=Zhang San
2015003 column=BasicInfo:S_No, timestamp=2021-11-30T00:50:38.871,
value=2015003
2015003 column=BasicInfo:S_Sex, timestamp=2021-11-30T00:51:05.683,
value=male
3 row(s)
Took 0.1967 seconds
hbase:020:0>

```

③插入课程及选课成绩信息

```

put 'student', '2015001', 'Math:C_No', '123001'
put 'student', '2015001', 'Math:C_Name', 'Math'
put 'student', '2015001', 'Math:C_Credit', '2'
put 'student', '2015001', 'Math:SC_Score', '86'
put 'student', '2015001', 'English:C_No', '123003'
put 'student', '2015001', 'English:C_Name', 'English'
put 'student', '2015001', 'English:C_Credit', '3'
put 'student', '2015001', 'English:SC_Score', '69'

put 'student', '2015002', 'CS:C_No', '123002'
put 'student', '2015002', 'CS:C_Name', 'Computer Science'
put 'student', '2015002', 'CS:C_Credit', '5'
put 'student', '2015002', 'CS:SC_Score', '77'
put 'student', '2015002', 'English:C_No', '123003'
put 'student', '2015002', 'English:C_Name', 'English'
put 'student', '2015002', 'English:C_Credit', '3'
put 'student', '2015002', 'English:SC_Score', '99'

put 'student', '2015003', 'Math:C_No', '123001'
put 'student', '2015003', 'Math:C_Name', 'Math'
put 'student', '2015003', 'Math:C_Credit', '2'
put 'student', '2015003', 'Math:SC_Score', '98'
put 'student', '2015003', 'CS:C_No', '123002'
put 'student', '2015003', 'CS:C_Name', 'Computer Science'
put 'student', '2015003', 'CS:C_Credit', '5'
put 'student', '2015003', 'CS:SC_Score', '95'

```

插入结果：

```

hbase:044:0> scan 'student'
ROW COLUMN+CELL
2015001 column=BasicInfo:S_Age, timestamp=2021-11-30T00:49:09.495,
value=23
2015001 column=BasicInfo:S_Name, timestamp=2021-11-30T00:48:53.392
, value=Li Lei
2015001 column=BasicInfo:S_No, timestamp=2021-11-30T00:49:31.690,
value=2015001
2015001 column=BasicInfo:S_Sex, timestamp=2021-11-30T00:49:02.674,
value=male
2015001 column=English:C_Credit, timestamp=2021-11-30T00:56:44.555
, value=3
2015001 column=English:C_Name, timestamp=2021-11-30T00:56:27.748,
value=English
2015001 column=English:C_No, timestamp=2021-11-30T00:56:19.070, va
lue=123003
2015001 column=English:SC_Score, timestamp=2021-11-30T00:58:05.484
, value=69
2015001 column=Math:C_Credit, timestamp=2021-11-30T00:55:53.427, v
alue=2
2015001 column=Math:C_Name, timestamp=2021-11-30T00:55:45.778, val
ue=Math
2015001 column=Math:C_No, timestamp=2021-11-30T00:55:37.248, value
=123001
2015001 column=Math:SC_Score, timestamp=2021-11-30T00:57:56.448, v
alue=86
2015002 column=BasicInfo:S_Age, timestamp=2021-11-30T00:50:18.446,
value=22
2015002 column=BasicInfo:S_Name, timestamp=2021-11-30T00:49:59.190

```

```

, value=Han Meimei
2015002 column=BasicInfo:S_No, timestamp=2021-11-30T00:49:47.725,
value=2015002
2015002 column=BasicInfo:S_Sex, timestamp=2021-11-30T00:50:09.775,
value=female
2015002 column=Computer Science:C_Credit, timestamp=2021-11-30T00:
59:11.039, value=5
2015002 column=Computer Science:C_Name, timestamp=2021-11-30T00:58
:59.550, value=Computer Science
2015002 column=Computer Science:C_No, timestamp=2021-11-30T00:58:4
7.940, value=123002
2015002 column=Computer Science:SC_Score, timestamp=2021-11-30T01:
00:57.330, value=77
2015002 column=English:C_Credit, timestamp=2021-11-30T01:00:34.091
, value=3
2015002 column=English:C_Name, timestamp=2021-11-30T00:59:52.459,
value=English
2015002 column=English:C_No, timestamp=2021-11-30T00:59:37.959, va
lue=123003
2015002 column=English:SC_Score, timestamp=2021-11-30T01:01:10.315
, value=99
2015003 column=BasicInfo:S_Age, timestamp=2021-11-30T00:51:15.113,
value=24
2015003 column=BasicInfo:S_Name, timestamp=2021-11-30T00:50:54.614
, value=Zhang San
2015003 column=BasicInfo:S_No, timestamp=2021-11-30T00:50:38.871,
value=2015003
2015003 column=BasicInfo:S_Sex, timestamp=2021-11-30T00:51:05.683,
value=male
2015003 column=Computer Science:C_Credit, timestamp=2021-11-30T01:
02:55.268, value=5
2015003 column=Computer Science:C_Name, timestamp=2021-11-30T01:02

```



```

2015003      column=BasicInfo:S_Name, timestamp=2021-11-30T00:50:54.614
, value=Zhang San
2015003      column=BasicInfo:S_No, timestamp=2021-11-30T00:50:38.871,
value=2015003
2015003      column=BasicInfo:S_Sex, timestamp=2021-11-30T00:51:05.683,
value=male
2015003      column=Computer Science:C_Credit, timestamp=2021-11-30T01:
02:55.268, value=5
2015003      column=Computer Science:C_Name, timestamp=2021-11-30T01:02
:46.090, value=Computer Science
2015003      column=Computer Science:C_No, timestamp=2021-11-30T01:02:2
7.175, value=123002
2015003      column=Computer Science:SC_Score, timestamp=2021-11-30T01:
03:05.737, value=95
2015003      column=Math:C_Credit, timestamp=2021-11-30T01:01:48.306, v
alue=2
2015003      column=Math:C_Name, timestamp=2021-11-30T01:01:41.763, val
ue=Math
2015003      column=Math:C_No, timestamp=2021-11-30T01:01:33.628, value
=123001
2015003      column=Math:SC_Score, timestamp=2021-11-30T01:01:59.704, v
alue=98
3 row(s)
Took 0.2335 seconds
hbase:045:0>

```

④查询选修Computer Science的学生的成绩

```
scan 'student', {COLUMN=>'Computer Science:SC_Score'}
```

查询结果:

```

hbase:045:0> scan 'student', {COLUMN=>'Computer Science:SC_Score'}
ROW          COLUMN+CELL
2015002      column=Computer Science:SC_Score, timestamp=2021-11-30T01:
00:57.330, value=77
2015003      column=Computer Science:SC_Score, timestamp=2021-11-30T01:
03:05.737, value=95
2 row(s)
Took 0.3756 seconds
hbase:045:0>

```

⑤增加新的列族和新列Contact:Email, 并添加数据

```

alter 'student', 'Contact'
put 'student', '2015001', 'Contact:S_Email', 'lilie@qq.com'
put 'student', '2015002', 'Contact:S_Email', 'hmm@qq.com'
put 'student', '2015003', 'Contact:S_Email', 'zs@qq.com'

```

添加并查询的结果:

```

hbase:052:0> scan 'student', {COLUMN=>'Contact:S_Email'}
ROW          COLUMN+CELL
2015001      column=Contact:S_Email, timestamp=2021-11-30T01:10:58.483,
value=lilie@qq.com
2015002      column=Contact:S_Email, timestamp=2021-11-30T01:11:06.330,
value=hmm@qq.com
2015003      column=Contact:S_Email, timestamp=2021-11-30T01:11:12.823,
value=zs@qq.com
3 row(s)
Took 0.1047 seconds
hbase:053:0>

```

⑥删除学号为2015003的学生的选课记录

```
delete 'student', '2015003', 'Math:C_No'
delete 'student', '2015003', 'Math:C_Name'
delete 'student', '2015003', 'Math:C_Credit'
delete 'student', '2015003', 'Math:SC_Score'
delete 'student', '2015003', 'Computer Science:C_No'
delete 'student', '2015003', 'Computer Science:C_Name'
delete 'student', '2015003', 'Computer Science:C_Credit'
delete 'student', '2015003', 'Computer Science:SC_Score'
```

删除结果:

```
hbase:067:0> scan 'student',{STARTROW=>'2015003',ENDROW=>'2015003'}
ROW COLUMN+CELL
2015003 column=BasicInfo:S_Age, timestamp=2021-11-30T00:51:15.113, value=24
2015003 column=BasicInfo:S_Name, timestamp=2021-11-30T00:50:54.614, value=Zhang San
2015003 column=BasicInfo:S_No, timestamp=2021-11-30T00:50:38.871, value=2015003
2015003 column=BasicInfo:S_Sex, timestamp=2021-11-30T00:51:05.683, value=male
2015003 column=Contact:S_Email, timestamp=2021-11-30T01:11:12.823, value=zs@qq.com
1 row(s)
Took 0.0586 seconds
hbase:068:0>
```

⑦删除所创建的表

```
disable 'student'
drop 'student'
```

删除结果:

```
hbase:068:0> disable 'student'
Took 4.4047 seconds
hbase:069:0> drop 'student'
Took 0.8544 seconds
hbase:070:0> list
TABLE
boys
course
sc
3 row(s)
Took 0.0099 seconds
=> ["boys", "course", "sc"]
hbase:071:0>
```

六、遇到的主要困难

①在shell和IDEA中执行指令/代码报错

一开始在shell和java代码中执行指令一切正常，但后来突然一下子，在java中执行代码无响应、在shell中执行命令报错（中间可能是由于我重启了虚拟机或者在程序的关闭问题上处理不小心）：

```

hbase:001:0> list
TABLE
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:1205)
    at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
    at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:392)
    at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:133)
    at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:354)
    at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:334)
For usage try 'help "list"'
Took 12.8323 seconds
hbase:002:0> exit

```

上网查找后发现是由于集群处于安全模式，于是退出安全模式并重启hbase：

```

jzt@jzt-virtual-machine:~/hadoop_install/hadoop-3.2.2/bin$ ./hdfs dfsadmin -safemode get
Safe mode is ON
jzt@jzt-virtual-machine:~/hadoop_install/hadoop-3.2.2/bin$ ./hdfs dfsadmin -safemode leave
Safe mode is OFF
jzt@jzt-virtual-machine:~/hadoop_install/hadoop-3.2.2/bin$

```

但是又报出新错误：

```

Took 0.0039 seconds
hbase:001:0> list
TABLE
ERROR: KeeperErrorCode = NoNode for /hbase/master
For usage try 'help "list"'
Took 0.3457 seconds
hbase:002:0>

```


在调试过程中，尝试调用stop-hbase.sh时弹出信息如下，我直接看到了最后一行的信息：

```

jzt@jzt-virtual-machine:~/hbase/bin$ ./stop-hbase.sh
no hbase master found
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/jzt/hadoop_install/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:/home/jzt/hbase/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:/home/jzt/hadoop_install/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:/home/jzt/hbase/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: running zookeeper, logging to /home/jzt/hbase/bin/../logs/hbase-jzt-zookeeper-jzt-virtual-machine.out
127.0.0.1: no zookeeper to stop because no pid file /tmp/hbase-jzt-zookeeper.pid
jzt@jzt-virtual-machine:~/hbase/bin$ ./start-hbase.sh

```

针对该信息，上网查找相关资料：

 大树168: 原因是，默认情况下pid文件保存在/tmp目录下，/tmp目录下的文件很容易丢失，解决办法：
在hbase-env.sh中修改pid文件的存放路径；在hbase-env.sh中下面的文字默认是注释掉的，放开即可，
也可以自己指定存放位置：# The directory where pid files are stored. /tmp by default. export HBASE_PID_DIR=/opt/hbase/pids 3年前 回复 ...

在hbase-env.sh中找到该行，并且取消注释、自定义了HBASE_PID_DIR路径：

```

# The directory where pid files are stored. /tmp by default.
# export HBASE_PID_DIR=/var/hadoop/pids

```

但结果还是不行。

在反复探索尝试过程中，我发现每次启动hadoop都会处于safemode状态，之前正常的时候则不会这样。上网查找之后发现是由于HDFS被异常关闭后造成数据库的不一致状态而形成的，解决方法之一是重新格式化hdfs，简单粗暴。

所以我按照网上的教程重新进行格式化，之后发现果然不再处于safemode状态了，hbase shell里也可以正常操作。

但是IDEA里还是报错：

```
org.apache.hadoop.hbase.NotServingRegionException: org.apache.hadoop.hbase.NotServingRegionException: hbase:meta,,1 is not online on jzt-virtual-machine,16020,163789426
```

在StackOverflow上找到的一点解决办法也不奏效。无奈之下，我重装了hbase，但结果问题依然存在。

在试图解决的过程中发现stop-hbase时提示no hbase master found，但是jps显示一切正常：

```
jzt@jzt-virtual-machine:~/hbase/bin$ ./stop-hbase.sh
no hbase master found

jzt@jzt-virtual-machine:~/hbase/bin$ jps
7808 SecondaryNameNode
22048 HQuorumPeer
7616 DataNode
7490 NameNode
29844 Jps
8100 ResourceManager
8229 NodeManager
13815 Main
22298 HRegionServer
14204 RemoteMavenServer36
22157 HMaster
25182 Launcher
jzt@jzt-virtual-machine:~/hbase/bin$
```

针对“no hbase master found”报错信息，我上网查找了相关资料：

报错原因

此时可以大体确定报错原因，系统找不到HBase的pid文件，pid文件里面是HBase的进程号，找不到进程号系统就没有办法去结束这个进程。
HBase的pid文件默认存放路径为 /tmp 路径，可以进去看一下有没有和HBase相关的文件。肯定没有，因为很有可能被操作系统删掉了。

(1) 修改pid文件存放路径

进入 /opt/hbase-2.0.0/conf 目录，找到 hbase-env.sh 进行修改

将文件中的对应行修改（大约119行左右）

```
export HBASE_PID_DIR=/var/hbase/pids
```

路径可以按照自己的习惯指定，但是一定要放在一个安全的地方，不要动不动就被linux系统抓了壮丁删掉。

(2) 暴力解决（测试环境使用）

直接kill掉HBase的相关进程，重启服务就行了，重启之后新的pid就会在指定目录下生成，相同的问题以后也就不会出现了。

如果是测试环境可以使用这种简单粗暴的方法，小几率会造成丢数据或者HBase无法启动的情况，但是解决问题简单、快速。

通过上述操作后问题成功解决，IDEA运行也不报错了！其实之前已经尝试过修改pid文件存放路径，但是相当于只执行了上图中的第（1）步，没有执行第（2）步，即修改后没有通过kill等手段关闭hbase相关进程，导致修改未生效。这个IDEA报错的问题在网上找了很久，大部分说法是数据损坏等，但是没想到最后以这种方法解决。这也提示我们要善于观察发现，如果没有发现stop-hbase时提示的no hbase master found，一味地按照网上的解决方案尝试，可能即使再花上很多时间也无法解决。

②在已有表新建列族时报错"read only"

一开始我新建列族采取的如下方法，通过admin.getTableDescriptor()方法得到HTableDescriptor实例，然后调用该实例的addFamily()方法增加列族，最后调用admin.modifyTable()方法使修改生效：

```
public void modifyTable(String tableName, String[] addColumn) throws Exception {
    //转化为表名
    TableName name = TableName.valueOf(tableName);
    //判断表是否存在
    if(admin.tableExists(name)) {
        //判断表是否可用状态
        boolean tableEnabled = admin.isTableEnabled(name);
        if(tableEnabled) {
            //使表变成不可用
            admin.disableTable(name);
        }
        //根据表名得到表
        HTableDescriptor tableDescriptor = admin.getTableDescriptor(name);
        System.out.println("1"+tableDescriptor.isReadOnly());
        tableDescriptor.setReadOnly(false);
        System.out.println("2"+tableDescriptor.isReadOnly());
        //创建列簇结构对象，添加列
        for(String add : addColumn) {
            HColumnDescriptor addColumnDescriptor = new HColumnDescriptor(add);
            System.out.println("2.5"+tableDescriptor.isReadOnly());
            tableDescriptor.addFamily(addColumnDescriptor);
            System.out.println("2.7"+tableDescriptor.isReadOnly());
        }
        System.out.println("3"+tableDescriptor.isReadOnly());
        ...

        admin.modifyTable(tableDescriptor);
        System.out.println("4"+tableDescriptor.isReadOnly());
        admin.enableTable(name);
    }else {
        System.out.println("table不存在");
    }
}
```

但是报错说HTD是read-only的：

```
1false
2false
2.5false
java.lang.UnsupportedOperationException Create breakpoint : HTableDescriptor is read-only
    at org.apache.hadoop.hbase.client.ImmutableHTableDescriptor.getDelegateForModification(ImmutableHTableDescriptor.java:58)
    at org.apache.hadoop.hbase.HTableDescriptor.addFamily(HTableDescriptor.java:587)
    at Hbase.modifyTable(Hbase.java:108)
    at run.main(run.java:81)
```

在上图中可以看到我在代码中插入了若干行输出代码，输出该HTD的isReadOnly()属性，结果显示一直为false，即并非只读，那么为什么会报错说是只读的呢？看源代码也没有发现任何异常：

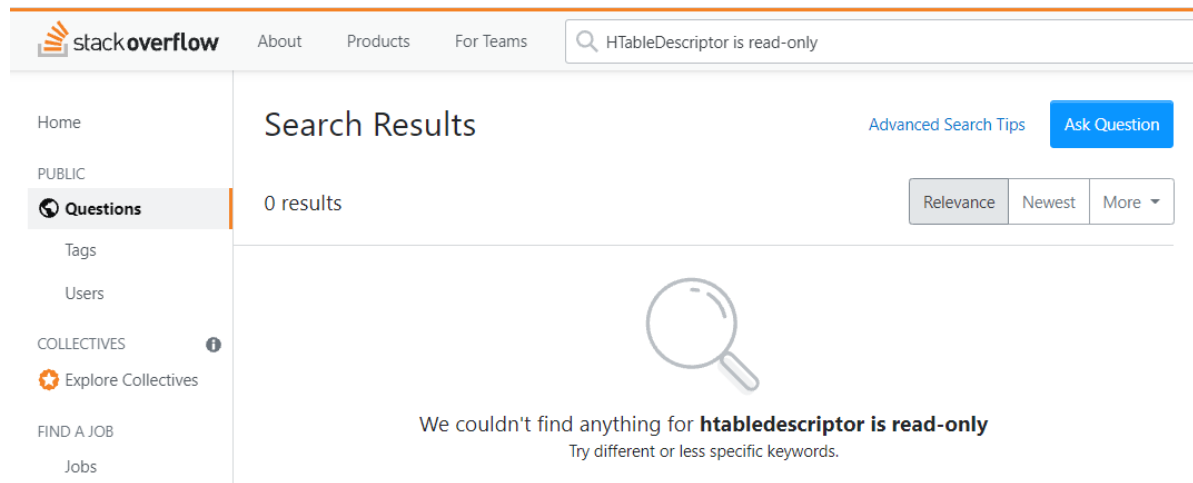

```
⚠️ @Deprecated
HTableDescriptor getTableDescriptor(TableName tableName)
    throws TableNotFoundException, IOException;

Get a table descriptor.
Params: tableName — as a TableName
Returns: the tableDescriptor
Throws: TableNotFoundException —
        IOException — if a remote or network exception occurs
```

也曾经考虑过是不是由于该方法为Deprecated，所以会有一些问题。但是尝试了使用不Deprecated的方法、获取到了TableDescriptor实例后发现该实例没有新增列族的方法：

```
HTableDescriptor tableDescriptor = admin.getTableDescriptor(name);
TableDescriptor tb = admin.getDescriptor(name);
tb.
  (m) getColumnFamilies() ColumnFamilyDescriptor[]
  (m) getColumnFamily(byte[] name) ColumnFamilyDescriptor
  (m) isReadOnly() boolean
  (m) getColumnFamilyCount() int
  (m) getTableName() TableName
  (m) getColumnFamilyNames() Set<byte[]>
  (m) getCoprocessorDescriptors() Collection<CoprocessorDescriptor>
  (m) getDurability() Durability
  (m) getFlushPolicyClassName() String
  (m) getMaxFileSize() long
  (m) getMemStoreFlushSize() long
  (m) getMemStoreFlushTargetDescriptorCount() int
  (m) getMemStoreFlushTargetDescriptorCount() int
Ctrl+向下箭头 and Ctrl+向上箭头 will move caret down and up in the editor Next Tip
System.out.println(" " + tableDescriptor.isReadOnly());
```

上StackOverflow查找直接显示0 results，百度也找不到任何相关信息：



万般无奈之下向助教请教，然后助教给我看了这个：

getTableDescriptor

```
public HTableDescriptor getTableDescriptor(TableName tableName)
    throws IOException
```

Description copied from interface: Admin

Get a table descriptor.

Specified by:

getTableDescriptor in interface Admin

Parameters:

tableName - as a TableName

Returns:

the **read-only** tableDescriptor

Throws:

TableNotFoundException

IOException - if a remote or network exception occurs

在HBase的官方API文档中写明了返回的是一个**read-only**的HTD！但是在源代码中并没有显示写出（之前我一直觉得官方文档密密麻麻的看着头疼，所以看的比较少，这次也是给了我一个教训）。

在官方文档中查找后发现，HBase2.4.8中，admin的方法中，所有返回HTD实例的方法，返回的都是read-only的HTD。因此网上的很多（基本可是说是所有）插入列族的方法在我目前的环境下是无效的。

为了解决这个问题，我在官方文档中进一步查找，发现了如下方法：

```
Future<Void> addColumnFamilyAsync(TableName tableName, ColumnFamilyDescriptor columnFamily)
Add a column family to an existing table.
```

该方法传入ColumnFamilyDescriptor实例，看起来挺靠谱。

所以我在代码中尝试了该方法：

```
admin.addColumnFamily(name, cfd);
```

结果发现能够成功执行！

不过，该方法**addColumnFamily**在官方API文档中并没有给出，官方文档中只有**addColumnFamilyAsync**方法，这一点我也不知道为什么。

这个问题让我学到的教训就是，随着版本的变化，很多接口可能都发生了巨大的变化，网上一些过去的教程可能不再有用，而新的教程可能也还未更新，这时候就需要我们自己去读源码、去读官方文档，自己发现问题所在并加以修正。