湖南大學
HUNAN UNIVERSITY

实验报告

大 数 据 处 理 技 术

报 告 人 姓 名	刘飞鸿
学　　　　号	S2010W0748
学 科 专 业	电子信息
报告提交日期	2020 年 11 月 3 日

实验 5 NoSQL 和关系数据库的操作比较

一、实验目的

- 理解四种数据库(MySQL、HBase、Redis 和MongoDB)的概念以及不同点；
- 熟练使用四种数据库操作常用的 Shell 命令；
- 熟悉四种数据库操作常用的 Java API。

二、实验平台

- 操作系统：Linux（建议Ubuntu16.04）；
- Hadoop 版本：2.7.1；
- MySQL 版本：5.6；
- HBase 版本：1.1.2；
- Redis 版本：3.0.6；
- MongoDB 版本：3.2.6；
- JDK 版本：1.7 或以上版本；
- Java IDE：Eclipse；

三、实验步骤

（一） MySQL 数据库操作

学生表 Student

Name	English	Math	Computer
zhangsan	69	86	77
lisi	55	100	88

1. 根据上面给出的 Student 表，在 MySQL 数据库中完成如下操作：

（1） 在MySQL 中创建 Student 表，并录入数据；

（2） 用SQL 语句输出 Student 表中的所有记录；

（3） 查询 zhangsan 的Computer 成绩；

（4） 修改 lisi 的Math 成绩，改为 95。

2.根据上面已经设计出的 Student 表，使用 MySQL 的JAVA 客户端编程实现以下操作：

（1） 向Student 表中添加如下所示的一条记录：

scofield	45	89	100

（2） 获取 scofield 的English 成绩信息

（二）HBase 数据库操作

学生表 Student

name	score		
	English	Math	Computer
zhangsan	69	86	77
lisi	55	100	88

1. 根据上面给出的学生表 Student 的信息，执行如下操作：

（1） 用Hbase Shell 命令创建学生表 Student；

（2） 用scan 命令浏览 Student 表的相关信息；

（3） 查询 zhangsan 的Computer 成绩；

（4） 修改 lisi 的Math 成绩，改为 95。

2.根据上面已经设计出的 Student 表，用 HBase API 编程实现以下操作：

（1） 添加数据：English:45 Math:89 Computer:100

scofield	45	89	100

（2） 获取 scofield 的English 成绩信息。

（三）Redis 数据库操作

Student 键值对如下：

```
zhangsan: {
        English: 69
        Math: 86
        Computer: 77
}
lisi: {
        English: 55
        Math: 100
        Computer: 88
}
```

1. 根据上面给出的键值对，完成如下操作：

（1） 用 Redis 的哈希结构设计出学生表 Student（键值可以用 student.zhangsan 和 student.lisi 来表示两个键值属于同一个表）；

（2） 用hgetall 命令分别输出 zhangsan 和lisi 的成绩信息；

（3） 用hget 命令查询 zhangsan 的Computer 成绩；

（4） 修改 lisi 的Math 成绩，改为 95。

2.根据上面已经设计出的学生表Student，用Redis 的 JAVA 客户端编程(jedis)，实现如下操作：

（1） 添加数据：English:45 Math:89 Computer:100

该数据对应的键值对形式如下：

```
scofield: {
                                        English: 45
                                        Math: 89
                                        Computer: 100
}
```

（2） 获取 scofield 的English 成绩信息

（四）MongoDB 数据库操作

Student 文档如下:

```
                {
                        "name": "zhangsan",
                        "score": {
                              "English": 69,
                              "Math": 86,
                              "Computer": 77
                        }
                }
                {
                        "name": "lisi",
                        "score": {
                              "English": 55,
                              "Math": 100,
                              "Computer": 88
                        }
}
```

1.根据上面给出的文档，完成如下操作：

（1） 用MongoDB Shell 设计出 student 集合；

（2） 用find()方法输出两个学生的信息；

（3） 用find()方法查询 zhangsan 的所有成绩(只显示 score 列)；

（4） 修改 lisi 的Math 成绩，改为 95。

2. 根据上面已经设计出的 Student 集合，用 MongoDB 的Java 客户端编程，实现如下操作：

（1） 添加数据：English:45　　Math:89　Computer:100

　　与上述数据对应的文档形式如下：

```
{
                    “name”: “scofield”,
                    “score”: {
                        “English”: 45,
                        “Math”: 89,
                        “Computer”: 100
                    }
}
```

（2） 获取 scofield 的所有成绩成绩信息(只显示 score 列)

四、实验报告

- **HBase搭建环境、安装部署**

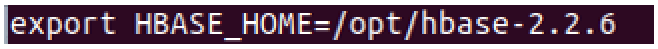
（1） 下载HBase发布包，通过上传软件上传至/usr/local/src目录

（2） 解压软件包并移动至/opt/目录

```
cd /usr/local/src
tar -xzvf hbase-2.2.6-bin.tar.gz
mv ./hbase-2.2.6 /opt/hbase-2.2.6
```

（3） 配置环境变量

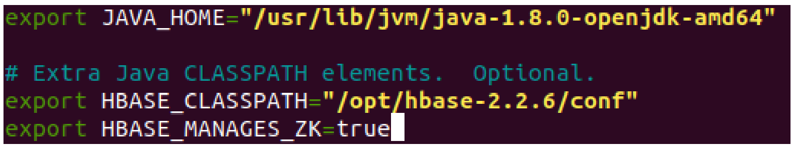
打开etc/profile:



```
export HBASE_HOME=/opt/hbase-2.2.6
```

（4） 配置相关的文件

1. 配置hbase-env.sh



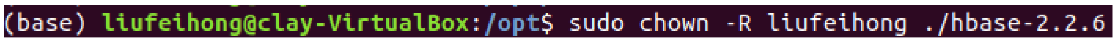
```
export JAVA_HOME="/usr/lib/jvm/java-1.8.0-openjdk-amd64"

# Extra Java CLASSPATH elements.  Optional.
export HBASE_CLASSPATH="/opt/hbase-2.2.6/conf"
export HBASE_MANAGES_ZK=true
```

2. 配置hbase-site.xml

```
<property>
    <name>hbase.rootdir</name>
    <value>hdfs://localhost:9000/hbase</value>
</property>
<property>
    <name>hbase.cluster.distributed</name>
    <value>true</value>
</property>
```

3. 拷贝Hadoop的hdfs-site.xml文件至${HBASE_HOME}/conf 目录

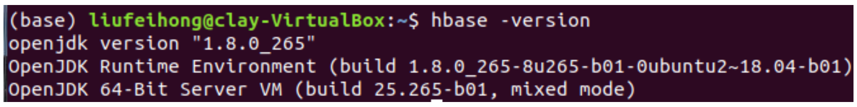
（5） 添加HBase权限：



```
(base) liufeihong@clay-VirtualBox:/opt$ sudo chown -R liufeihong ./hbase-2.2.6
```

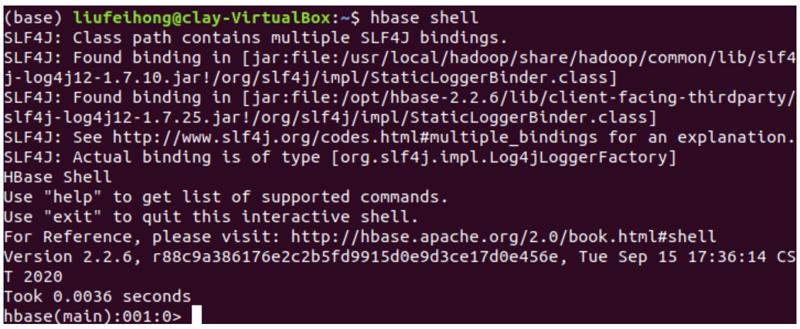
（6） 验证是否安装成功：

1. 输入 hbase -version:



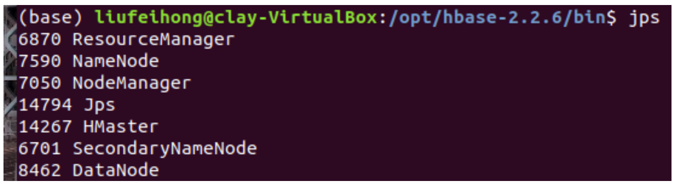
```
(base) liufeihong@clay-VirtualBox:~$ hbase -version
openjdk version "1.8.0_265"
OpenJDK Runtime Environment (build 1.8.0_265-8u265-b01-0ubuntu2~18.04-b01)
OpenJDK 64-Bit Server VM (build 25.265-b01, mixed mode)
```

2. 输入 hbase shell



```
(base) liufeihong@clay-VirtualBox:~$ hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hbase-2.2.6/lib/client-facing-thirdparty/
slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.2.6, r88c9a386176e2c2b5fd9915d0e9d3ce17d0e456e, Tue Sep 15 17:36:14 CS
T 2020
Took 0.0036 seconds
hbase(main):001:0>
```

3. 输入jps看到

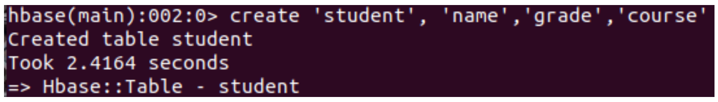


```
(base) liufeihong@clay-VirtualBox:/opt/hbase-2.2.6/bin$ jps
6870 ResourceManager
7590 NameNode
7050 NodeManager
14794 Jps
14267 HMaster
6701 SecondaryNameNode
8462 DataNode
```

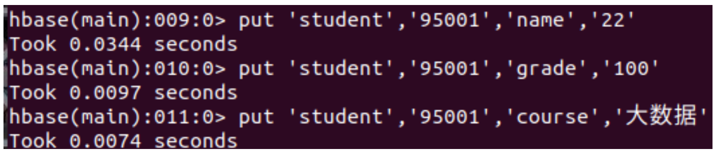
- **利用编程实现以下指定功能，并用 Hadoop 提供的 HBase Shell 命令完成相同任务：**

（1） 列出 HBase 所有的表的相关信息，例如表名；

1) 创建一个表并添加数据：

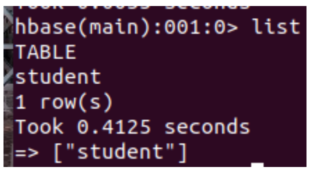


```
hbase(main):002:0> create 'student', 'name','grade','course'
Created table student
Took 2.4164 seconds
=> Hbase::Table - student
```



```
hbase(main):009:0> put 'student','95001','name','22'
Took 0.0344 seconds
hbase(main):010:0> put 'student','95001','grade','100'
Took 0.0097 seconds
hbase(main):011:0> put 'student','95001','course','大数据'
Took 0.0074 seconds
```

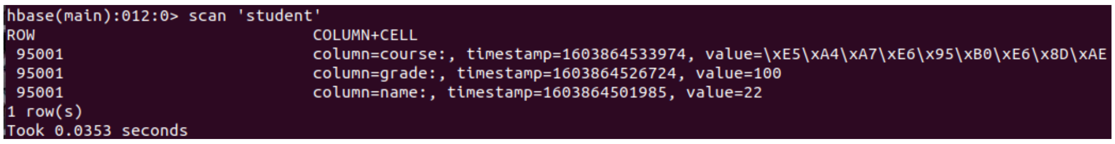
2) 查看所有表：



```
hbase(main):001:0> list
TABLE
student
1 row(s)
Took 0.4125 seconds
=> ["student"]
```

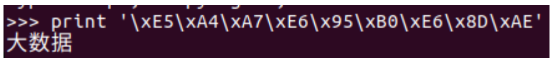
（2） 在终端打印出指定的表的所有记录数据；

将 Student 的所有记录打印出来



```
hbase(main):012:0> scan 'student'
ROW                              COLUMN+CELL
 95001                           column=course:, timestamp=1603864533974, value=\xE5\xA4\xA7\xE6\x95\xB0\xE6\x8D\xAE
 95001                           column=grade:, timestamp=1603864526724, value=100
 95001                           column=name:, timestamp=1603864501985, value=22
1 row(s)
Took 0.0353 seconds
```

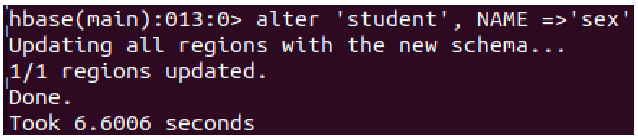
可以将 course 十六进制对应的中文打印出来：



```
>>> print '\xE5\xA4\xA7\xE6\x95\xB0\xE6\x8D\xAE'
大数据
```

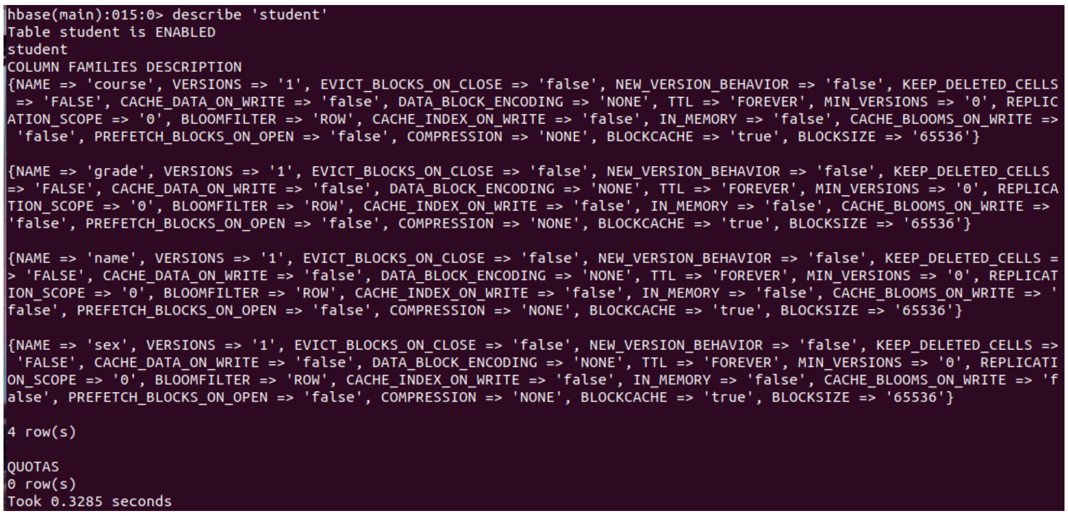
（3） 向已经创建好的表添加和删除指定的列族或列；

- 添加列sex：



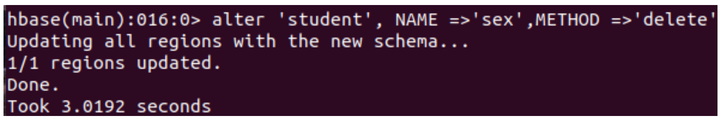
```
hbase(main):013:0> alter 'student', NAME =>'sex'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 6.6006 seconds
```

验证是否添加成功:



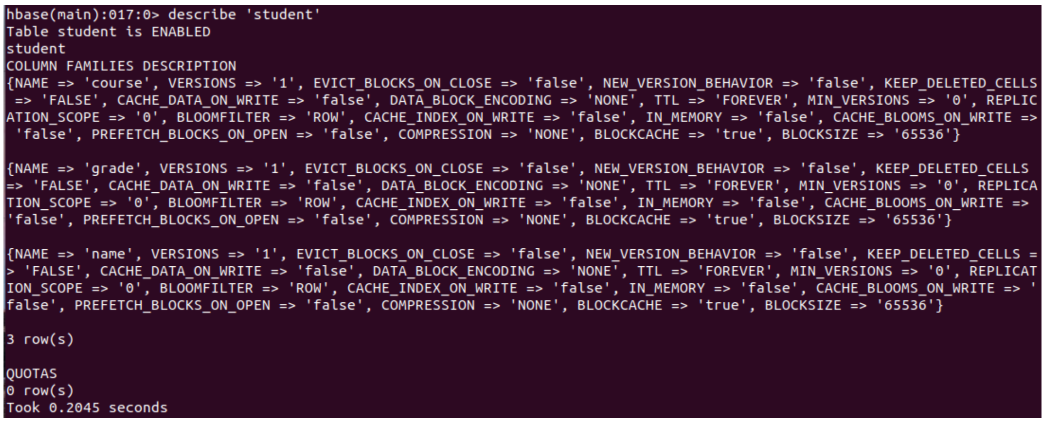
```
hbase(main):015:0> describe 'student'
Table student is ENABLED
student
COLUMN FAMILIES DESCRIPTION
{NAME => 'course', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS
 => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLIC
ATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE =>
 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'grade', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS
=> 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICA
TION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE =>
'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'name', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =
> 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICAT
ION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => '
false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'sex', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATI
ON_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'f
alse', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

4 row(s)

QUOTAS
0 row(s)
Took 0.3285 seconds
```

- 删除列sex:



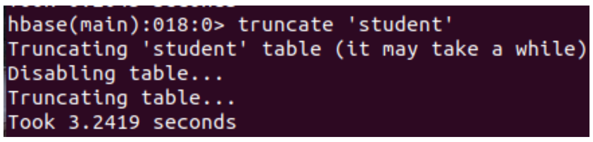
```
hbase(main):016:0> alter 'student', NAME =>'sex',METHOD =>'delete'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 3.0192 seconds
```

验证是否删除成功:



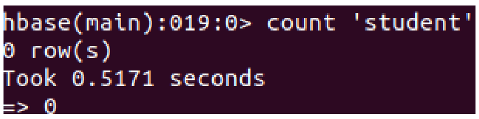
```
hbase(main):017:0> describe 'student'
Table student is ENABLED
student
COLUMN FAMILIES DESCRIPTION
{NAME => 'course', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS
 => 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLIC
ATION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE =>
 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'grade', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS
=> 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICA
TION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE =>
'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

{NAME => 'name', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =
> 'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICAT
ION_SCOPE => '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => '
false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}

3 row(s)

QUOTAS
0 row(s)
Took 0.2045 seconds
```

（4）清空指定的表的所有记录数据；



```
hbase(main):018:0> truncate 'student'
Truncating 'student' table (it may take a while):
Disabling table...
Truncating table...
Took 3.2419 seconds
```

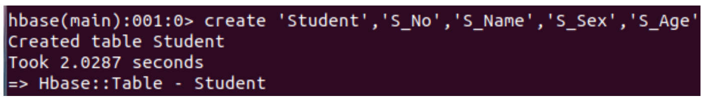
（5）统计表的行数。



```
hbase(main):019:0> count 'student'
0 row(s)
Took 0.5171 seconds
=> 0
```

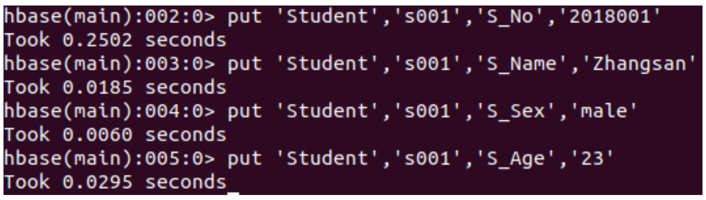
- **现有以下关系型数据库中的表和数据，要求将其转换为适合于 HBase 存储的表并插入数据：**

（1）**学生表创建及插入:**



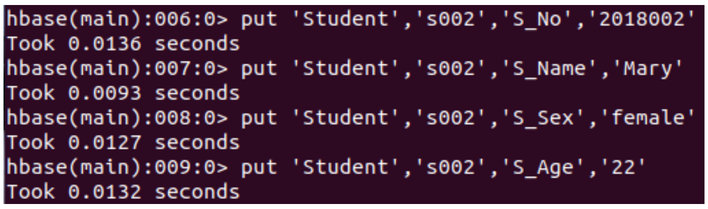
```
hbase(main):001:0> create 'Student','S_No','S_Name','S_Sex','S_Age'
Created table Student
Took 2.0287 seconds
=> Hbase::Table - Student
```

插入第一行数据:



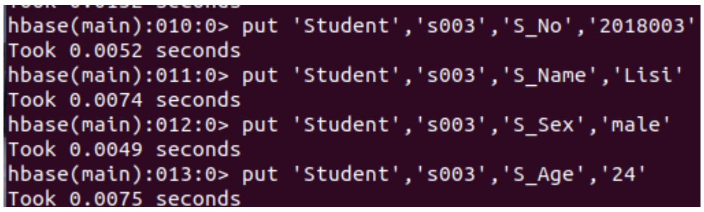
```
hbase(main):002:0> put 'Student','s001','S_No','2018001'
Took 0.2502 seconds
hbase(main):003:0> put 'Student','s001','S_Name','Zhangsan'
Took 0.0185 seconds
hbase(main):004:0> put 'Student','s001','S_Sex','male'
Took 0.0060 seconds
hbase(main):005:0> put 'Student','s001','S_Age','23'
Took 0.0295 seconds
```

插入第二行数据:



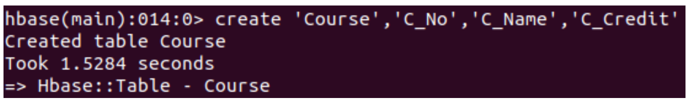
```
hbase(main):006:0> put 'Student','s002','S_No','2018002'
Took 0.0136 seconds
hbase(main):007:0> put 'Student','s002','S_Name','Mary'
Took 0.0093 seconds
hbase(main):008:0> put 'Student','s002','S_Sex','female'
Took 0.0127 seconds
hbase(main):009:0> put 'Student','s002','S_Age','22'
Took 0.0132 seconds
```

插入第三行数据:



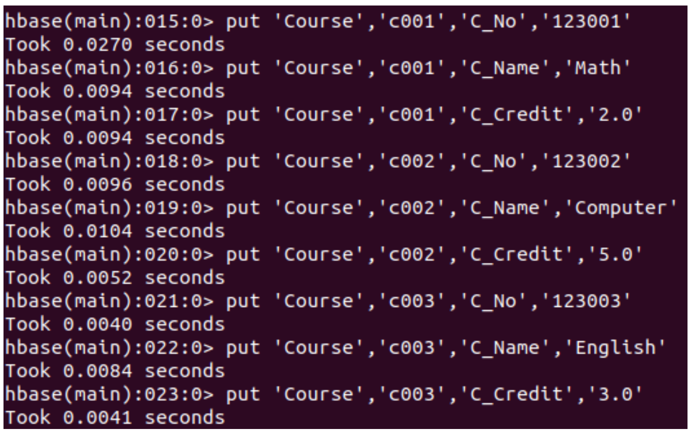
```
hbase(main):010:0> put 'Student','s003','S_No','2018003'
Took 0.0052 seconds
hbase(main):011:0> put 'Student','s003','S_Name','Lisi'
Took 0.0074 seconds
hbase(main):012:0> put 'Student','s003','S_Sex','male'
Took 0.0049 seconds
hbase(main):013:0> put 'Student','s003','S_Age','24'
Took 0.0075 seconds
```

（2）课程表创建及插入:



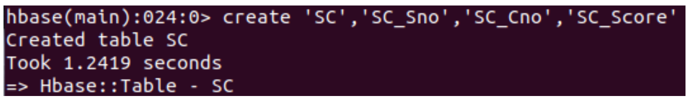
```
hbase(main):014:0> create 'Course','C_No','C_Name','C_Credit'
Created table Course
Took 1.5284 seconds
=> Hbase::Table - Course
```

插入数据:



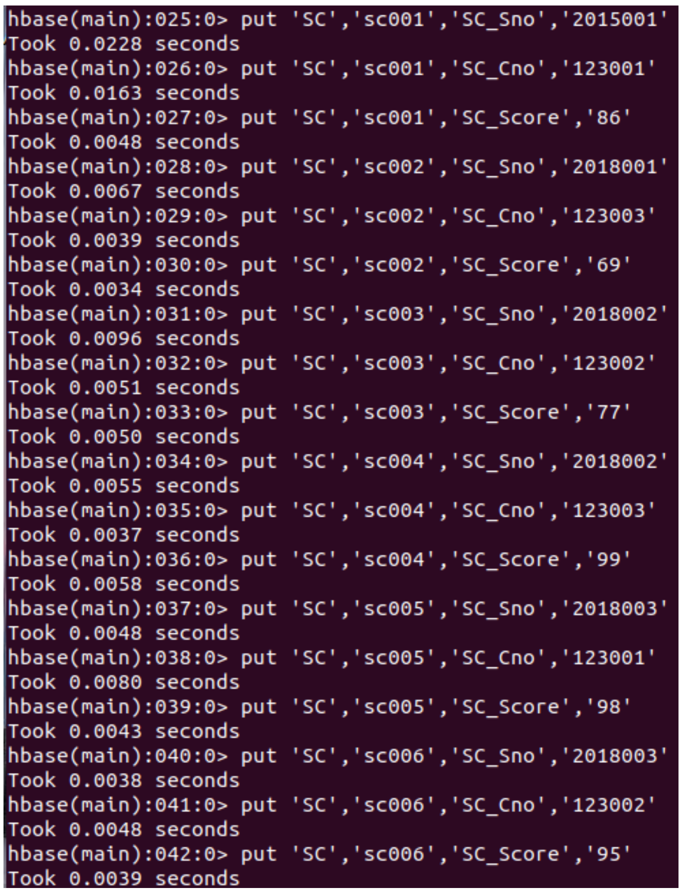
```
hbase(main):015:0> put 'Course','c001','C_No','123001'
Took 0.0270 seconds
hbase(main):016:0> put 'Course','c001','C_Name','Math'
Took 0.0094 seconds
hbase(main):017:0> put 'Course','c001','C_Credit','2.0'
Took 0.0094 seconds
hbase(main):018:0> put 'Course','c002','C_No','123002'
Took 0.0096 seconds
hbase(main):019:0> put 'Course','c002','C_Name','Computer'
Took 0.0104 seconds
hbase(main):020:0> put 'Course','c002','C_Credit','5.0'
Took 0.0052 seconds
hbase(main):021:0> put 'Course','c003','C_No','123003'
Took 0.0040 seconds
hbase(main):022:0> put 'Course','c003','C_Name','English'
Took 0.0084 seconds
hbase(main):023:0> put 'Course','c003','C_Credit','3.0'
Took 0.0041 seconds
```

（3）选课表创建及插入:



```
hbase(main):024:0> create 'SC','SC_Sno','SC_Cno','SC_Score'
Created table SC
Took 1.2419 seconds
=> Hbase::Table - SC
```

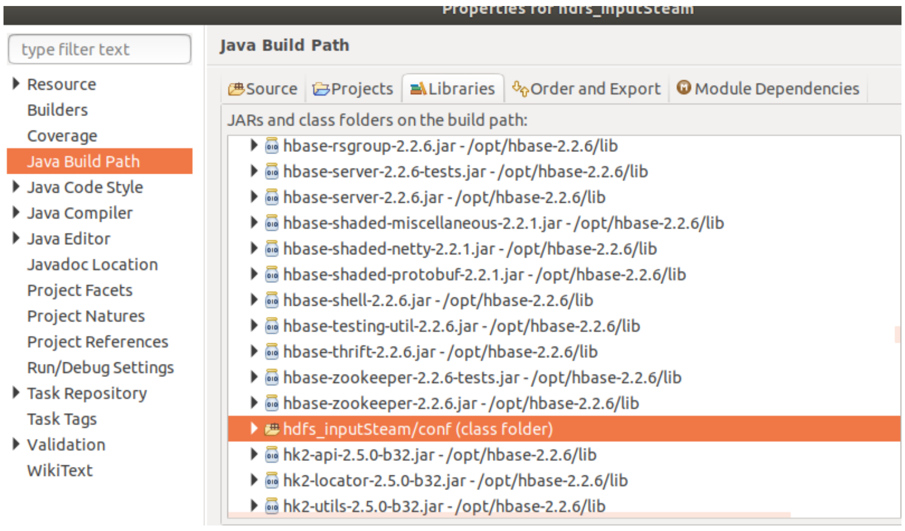
插入数据:



```
hbase(main):025:0> put 'SC','sc001','SC_Sno','2015001'
Took 0.0228 seconds
hbase(main):026:0> put 'SC','sc001','SC_Cno','123001'
Took 0.0163 seconds
hbase(main):027:0> put 'SC','sc001','SC_Score','86'
Took 0.0048 seconds
hbase(main):028:0> put 'SC','sc002','SC_Sno','2018001'
Took 0.0067 seconds
hbase(main):029:0> put 'SC','sc002','SC_Cno','123003'
Took 0.0039 seconds
hbase(main):030:0> put 'SC','sc002','SC_Score','69'
Took 0.0034 seconds
hbase(main):031:0> put 'SC','sc003','SC_Sno','2018002'
Took 0.0096 seconds
hbase(main):032:0> put 'SC','sc003','SC_Cno','123002'
Took 0.0051 seconds
hbase(main):033:0> put 'SC','sc003','SC_Score','77'
Took 0.0050 seconds
hbase(main):034:0> put 'SC','sc004','SC_Sno','2018002'
Took 0.0055 seconds
hbase(main):035:0> put 'SC','sc004','SC_Cno','123003'
Took 0.0037 seconds
hbase(main):036:0> put 'SC','sc004','SC_Score','99'
Took 0.0058 seconds
hbase(main):037:0> put 'SC','sc005','SC_Sno','2018003'
Took 0.0048 seconds
hbase(main):038:0> put 'SC','sc005','SC_Cno','123001'
Took 0.0080 seconds
hbase(main):039:0> put 'SC','sc005','SC_Score','98'
Took 0.0043 seconds
hbase(main):040:0> put 'SC','sc006','SC_Sno','2018003'
Took 0.0038 seconds
hbase(main):041:0> put 'SC','sc006','SC_Cno','123002'
Took 0.0048 seconds
hbase(main):042:0> put 'SC','sc006','SC_Score','95'
Took 0.0039 seconds
```

- 请编程实现以下功能：

1. createTable(String tableName, String[] fields)创建表，参数 tableName 为表的名称，字符串数组 fields 为存储记录各个字段名称的数组。要求当 HBase 已经存在名为 tableName 的表的时候，先删除原有的表，然后再创建新的表。
   a) 在eclipse下配置hbase开发环境：
      - ◆ 在工程中添加所需的jar包：我们需要的jar包在hbase的安装(解压缩)目录下的lib目录中
      - ◆ 指定HBase配置文件的位置：将HBase的配置文件复制一份到工程里。先在工程目录下创建一个名为conf的目录，再将HBase的配置文件 hbase-site.xml以及hadoop的配置文件core-site.xml、hdfs-site.xml、mapred-site.xml三个文件复制到该目录下。接着，还是右击项目工程，选择Properties->Java Build Path->Libraries->Add Class Folder, 将刚刚增加的conf目录选上：


Properties for hdfs_inputSteam

type filter text

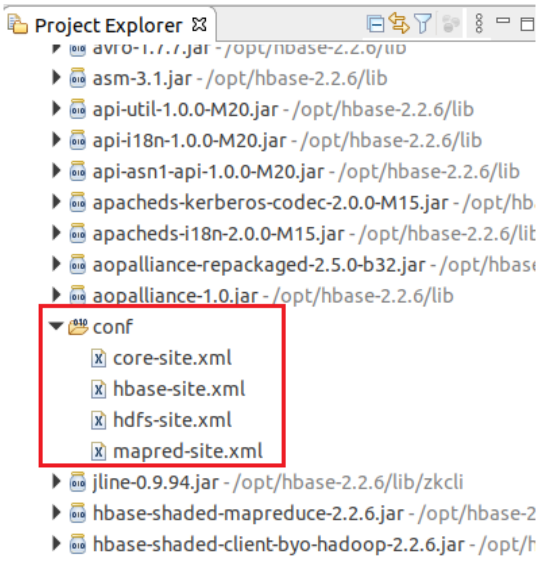
- Resource
- Builders
- Coverage
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project Facets
- Project Natures
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

Java Build Path

Source | Projects | Libraries | Order and Export | Module Dependencies

JARs and class folders on the build path:

```
hbase-rsgroup-2.2.6.jar - /opt/hbase-2.2.6/lib
hbase-server-2.2.6-tests.jar - /opt/hbase-2.2.6/lib
hbase-server-2.2.6.jar - /opt/hbase-2.2.6/lib
hbase-shaded-miscellaneous-2.2.1.jar - /opt/hbase-2.2.6/lib
hbase-shaded-netty-2.2.1.jar - /opt/hbase-2.2.6/lib
hbase-shaded-protobuf-2.2.1.jar - /opt/hbase-2.2.6/lib
hbase-shell-2.2.6.jar - /opt/hbase-2.2.6/lib
hbase-testing-util-2.2.6.jar - /opt/hbase-2.2.6/lib
hbase-thrift-2.2.6.jar - /opt/hbase-2.2.6/lib
hbase-zookeeper-2.2.6-tests.jar - /opt/hbase-2.2.6/lib
hbase-zookeeper-2.2.6.jar - /opt/hbase-2.2.6/lib
hdfs_inputSteam/conf (class folder)
hk2-api-2.5.0-b32.jar - /opt/hbase-2.2.6/lib
hk2-locator-2.5.0-b32.jar - /opt/hbase-2.2.6/lib
hk2-utils-2.5.0-b32.jar - /opt/hbase-2.2.6/lib
```


Project Explorer

```
avro-1.7.7.jar - /opt/hbase-2.2.6/lib
asm-3.1.jar - /opt/hbase-2.2.6/lib
api-util-1.0.0-M20.jar - /opt/hbase-2.2.6/lib
api-i18n-1.0.0-M20.jar - /opt/hbase-2.2.6/lib
api-asn1-api-1.0.0-M20.jar - /opt/hbase-2.2.6/lib
apacheds-kerberos-codec-2.0.0-M15.jar - /opt/hb
apacheds-i18n-2.0.0-M15.jar - /opt/hbase-2.2.6/lit
aopalliance-repackaged-2.5.0-b32.jar - /opt/hbas
aopalliance-1.0.jar - /opt/hbase-2.2.6/lib
conf
    core-site.xml
    hbase-site.xml
    hdfs-site.xml
    mapred-site.xml
jline-0.9.94.jar - /opt/hbase-2.2.6/lib/zkcli
hbase-shaded-mapreduce-2.2.6.jar - /opt/hbase-2
hbase-shaded-client-byo-hadoop-2.2.6.jar - /opt/h
```

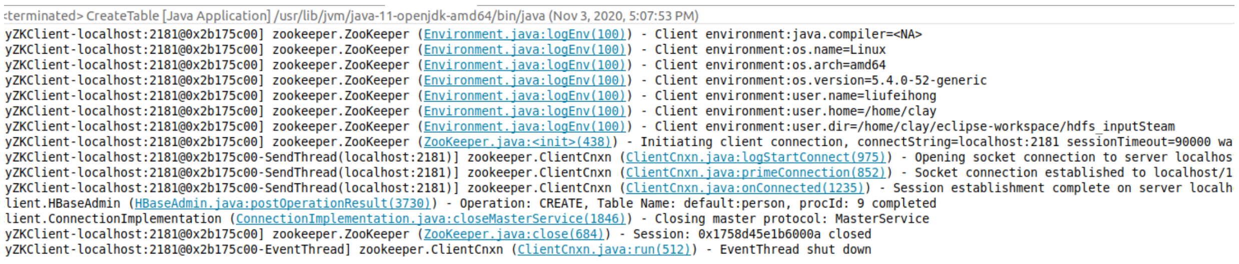
b） 运行的代码如下：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Admin;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;

import java.io.IOException;

public class CreateTable {
    public static Configuration configuration;
    public static Connection connection;
    public static Admin admin;
```

```
	public static void createTable(String tableName, String[] fields) throws
IOException {
		init();
		TableName tablename = TableName.valueOf(tableName);
		if (admin.tableExists(tablename)) {
			System.out.println("table is exists!");
			admin.disableTable(tablename);
			admin.deleteTable(tablename);
		}
		HTableDescriptor hTableDescriptor = new
HTableDescriptor(tablename);
		for (String str : fields) {
			HColumnDescriptor hColumnDescriptor = new
HColumnDescriptor(str);
			hTableDescriptor.addFamily(hColumnDescriptor);
		}
		admin.createTable(hTableDescriptor);
		close();
	}

	public static void init() {
		configuration = HBaseConfiguration.create();
		configuration.set("hbase.rootdir", "hdfs://localhost:9000/hbase");
		try {
			connection =
ConnectionFactory.createConnection(configuration);
			admin = connection.getAdmin();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public static void close() {
		try {
			if (admin != null) {
				admin.close();
			}
			if (null != connection) {
				connection.close();
			}
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
```

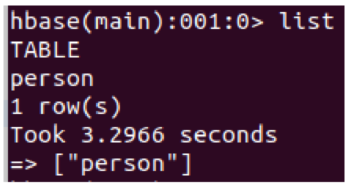
```
    public static void main(String[] args) {
        String[] fields = {"Score"};
        try {
            createTable("person", fields);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

c)　运行的结果如下：



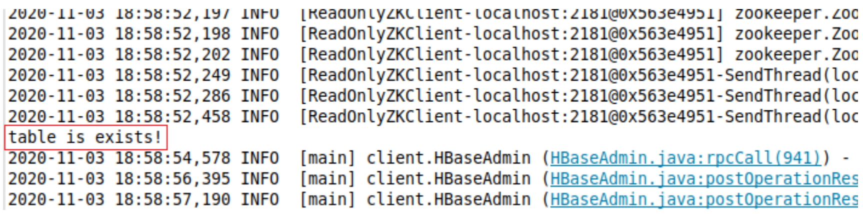
```
<terminated> CreateTable [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Nov 3, 2020, 5:07:53 PM)
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (Environment.java:logEnv(100)) - Client environment:java.compiler=<NA>
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (Environment.java:logEnv(100)) - Client environment:os.name=Linux
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (Environment.java:logEnv(100)) - Client environment:os.arch=amd64
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (Environment.java:logEnv(100)) - Client environment:os.version=5.4.0-52-generic
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (Environment.java:logEnv(100)) - Client environment:user.name=liufeihong
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (Environment.java:logEnv(100)) - Client environment:user.home=/home/clay
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (Environment.java:logEnv(100)) - Client environment:user.dir=/home/clay/eclipse-workspace/hdfs_inputSteam
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (ZooKeeper.java:<init>(438)) - Initiating client connection, connectString=localhost:2181 sessionTimeout=90000 wa
yZKClient-localhost:2181@0x2b175c00-SendThread(localhost:2181)] zookeeper.ClientCnxn (ClientCnxn.java:logStartConnect(975)) - Opening socket connection to server localhos
yZKClient-localhost:2181@0x2b175c00-SendThread(localhost:2181)] zookeeper.ClientCnxn (ClientCnxn.java:primeConnection(852)) - Socket connection established to localhost/1
yZKClient-localhost:2181@0x2b175c00-SendThread(localhost:2181)] zookeeper.ClientCnxn (ClientCnxn.java:onConnected(1235)) - Session establishment complete on server localh
lient.HBaseAdmin (HBaseAdmin.java:postOperationResult(3730)) - Operation: CREATE, Table Name: default:person, procId: 9 completed
lient.ConnectionImplementation (ConnectionImplementation.java:closeMasterService(1846)) - Closing master protocol: MasterService
yZKClient-localhost:2181@0x2b175c00] zookeeper.ZooKeeper (ZooKeeper.java:close(684)) - Session: 0x1758d45e1b6000a closed
yZKClient-localhost:2181@0x2b175c00-EventThread] zookeeper.ClientCnxn (ClientCnxn.java:run(512)) - EventThread shut down
```

d)　通过shell命令验证：



```
hbase(main):001:0> list
TABLE
person
1 row(s)
Took 3.2966 seconds
=> ["person"]
```

e)　如果再次运行代码，输出“table is exists!”：



```
2020-11-03 18:58:52,197 INFO  [ReadOnlyZKClient-localhost:2181@0x563e4951] zookeeper.Zoo
2020-11-03 18:58:52,198 INFO  [ReadOnlyZKClient-localhost:2181@0x563e4951] zookeeper.Zoo
2020-11-03 18:58:52,202 INFO  [ReadOnlyZKClient-localhost:2181@0x563e4951] zookeeper.Zoo
2020-11-03 18:58:52,249 INFO  [ReadOnlyZKClient-localhost:2181@0x563e4951-SendThread(loc
2020-11-03 18:58:52,286 INFO  [ReadOnlyZKClient-localhost:2181@0x563e4951-SendThread(loc
2020-11-03 18:58:52,458 INFO  [ReadOnlyZKClient-localhost:2181@0x563e4951-SendThread(loc
table is exists!
2020-11-03 18:58:54,578 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:rpcCall(941)) -
2020-11-03 18:58:56,395 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:postOperationRes
2020-11-03 18:58:57,190 INFO  [main] client.HBaseAdmin (HBaseAdmin.java:postOperationRes
```

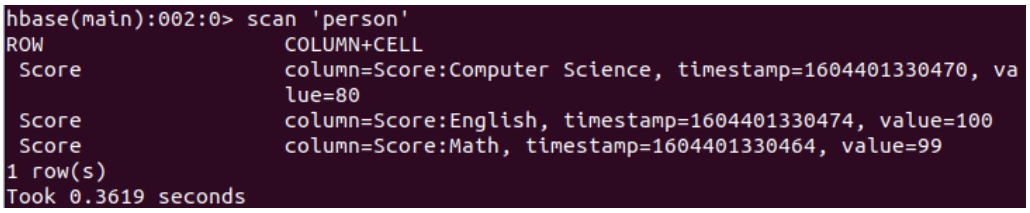
2. addRecord(String tableName, String row, String[] fields, String[] values)向表 tableName、行row（用S_Name 表示）和字符串数组 fields 指定的单元格中添加对应的数据 values。其中，fields 中每个元素如果对应的列族下还有相应的列限定符的话，用“columnFamily:column”表示。例如，同时向“Math”、“Computer Science”、“English”三列添加成绩时，字符串数组 fields 为 {“Score:Math”, ”Score:Computer Science”, ”Score:English”}， 数组 values 存储这三门课的成绩。

a)　运行的代码如下：

```
package hdfs_inputSteam;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;

import java.io.IOException;

public class AddRecord {
        public static Configuration configuration;
        public static Connection connection;
        public static Admin admin;

        public static void addRecord(String tableName, String row,
String[] fields, String[] values) throws IOException {
                init();
                Table                    table                    =
connection.getTable(TableName.valueOf(tableName));
                for (int i = 0; i != fields.length; i++) {
                        Put put = new Put(row.getBytes());
                        String[] cols = fields[i].split(":");
                        put.addColumn(cols[0].getBytes(),
cols[1].getBytes(), values[i].getBytes());
                        table.put(put);
                }
                table.close();
                close();
        }

        public static void init() {
                configuration = HBaseConfiguration.create();
                configuration.set("hbase.rootdir",
"hdfs://localhost:9000/hbase");
                try {
                        connection                                  =
ConnectionFactory.createConnection(configuration);
                        admin = connection.getAdmin();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

        public static void close() {
                try {
                        if (admin != null) {
                                admin.close();
                        }
                        if (null != connection) {
```

```
                        connection.close();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            public static void main(String[] args) {
                String[] fields = {"Score:Math", "Score:Computer Science", "Score:English"};
                String[] values = {"99", "80", "100"};
                try {
                    addRecord("person", "Score", fields, values);
                } catch (IOException e) {
                    e.printStackTrace();
                }

            }
        }
```

b） 对程序进行验证：



```
hbase(main):002:0> scan 'person'
ROW                    COLUMN+CELL
 Score                 column=Score:Computer Science, timestamp=1604401330470, va
                       lue=80
 Score                 column=Score:English, timestamp=1604401330474, value=100
 Score                 column=Score:Math, timestamp=1604401330464, value=99
1 row(s)
Took 0.3619 seconds
```

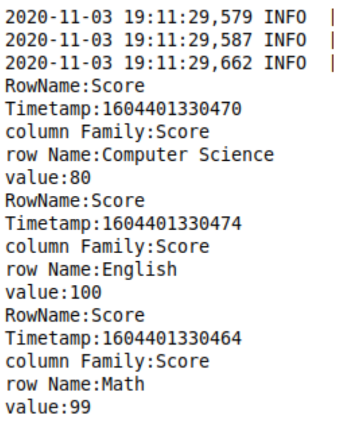
3. scanColumn(String tableName, String column)浏览表 tableName 某一列的数据，如果某一行记录中该列数据不存在，则返回 null。要求当参数 column 为某一列族名称时，如果底下有若干个列限定符，则要列出每个列限定符代表的列的数据；当参数 column 为某一列具体名称（例如“Score:Math”）时，只需要列出该列的数据。

a） 运行的代码如下：

```
package hdfs_inputSteam;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.CellUtil;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;

public class ScanColumn {
	public static Configuration configuration;
	public static Connection connection;
	public static Admin admin;

	public static void scanColumn(String tableName, String column) throws IOException {
		init();
		Table table = connection.getTable(TableName.valueOf(tableName));
		Scan scan = new Scan();
		scan.addFamily(Bytes.toBytes(column));
		ResultScanner scanner = table.getScanner(scan);
		for (Result result = scanner.next(); result != null; result = scanner.next()) {
			showCell(result);
		}
		table.close();
		close();
	}

	public static void showCell(Result result) {
		Cell[] cells = result.rawCells();
		for (Cell cell : cells) {
			System.out.println("RowName:" + new String(CellUtil.cloneRow(cell)) + " ");
			System.out.println("Timetamp:" + cell.getTimestamp() + " ");
			System.out.println("column Family:" + new String(CellUtil.cloneFamily(cell)) + " ");
			System.out.println("row Name:" + new String(CellUtil.cloneQualifier(cell)) + " ");
			System.out.println("value:" + new String(CellUtil.cloneValue(cell)) + " ");
		}
	}

	public static void init() {
		configuration = HBaseConfiguration.create();
		configuration.set("hbase.rootdir", "hdfs://localhost:9000/hbase");
		try {
			connection = ConnectionFactory.createConnection(configuration);
			admin = connection.getAdmin();
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    // 关闭连接
    public static void close() {
        try {
            if (admin != null) {
                admin.close();
            }
            if (null != connection) {
                connection.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        try {
            scanColumn("person", "Score");
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

b) 运行的结果如下：



```
2020-11-03 19:11:29,579 INFO  |
2020-11-03 19:11:29,587 INFO  |
2020-11-03 19:11:29,662 INFO  |
RowName:Score
Timetamp:1604401330470
column Family:Score
row Name:Computer Science
value:80
RowName:Score
Timetamp:1604401330474
column Family:Score
row Name:English
value:100
RowName:Score
Timetamp:1604401330464
column Family:Score
row Name:Math
value:99
```

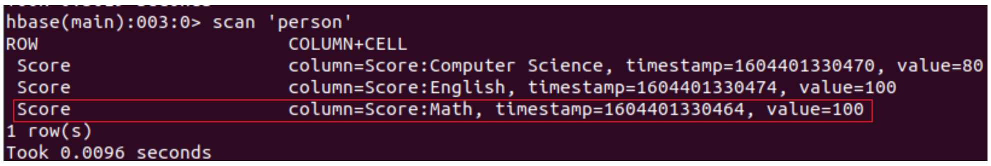
4. modifyData(String tableName, String row, String column)修改表 tableName，行 row（可以用学生姓名 S_Name 表示），列 column 指定的单元格的数据。

a) 代码如下：

```
package hdfs_inputSteam;
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;

import java.io.IOException;

public class ModifyData {

    public static long ts;
    public static Configuration configuration;
    public static Connection connection;
    public static Admin admin;

    public static void modifyData(String tableName,
    String row, String column, String val) throws
    IOException {
        init();
        Table table =
    connection.getTable(TableName.valueOf(tableName)
    );
        Put put = new Put(row.getBytes());
        Scan scan = new Scan();
        ResultScanner resultScanner =
    table.getScanner(scan);
        for (Result r : resultScanner) {
            for (Cell cell :
    r.getColumnCells(row.getBytes(),
    column.getBytes())) {
                ts = cell.getTimestamp();
            }
        }
        put.addColumn(row.getBytes(),
    column.getBytes(), ts, val.getBytes());
        table.put(put);
        table.close();
        close();
    }
    public static void init() {
        configuration =
    HBaseConfiguration.create();
        configuration.set("hbase.rootdir",
    "hdfs://localhost:9000/hbase");
        try {
            connection =
    ConnectionFactory.createConnection(configuration
    );
            admin = connection.getAdmin();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void close() {
        try {
            if (admin != null) {
                admin.close();
            }
            if (null != connection) {
                connection.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        try {
            modifyData("person", "Score", "Math",
    "100");
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

b) 验证如下：



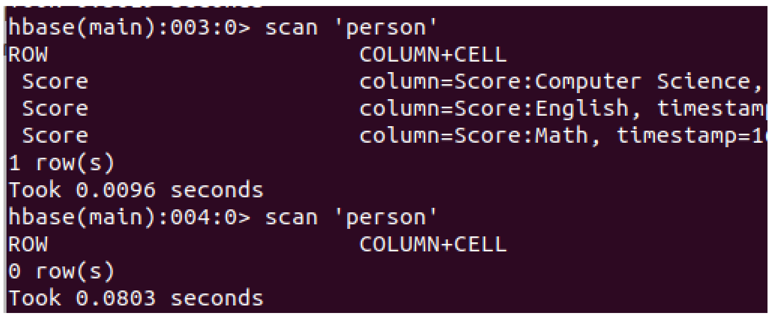
```
hbase(main):003:0> scan 'person'
ROW                         COLUMN+CELL
 Score                      column=Score:Computer Science, timestamp=1604401330470, value=80
 Score                      column=Score:English, timestamp=1604401330474, value=100
 Score                      column=Score:Math, timestamp=1604401330464, value=100
1 row(s)
Took 0.0096 seconds
```

5. deleteRow(String tableName, String row)删除表 tableName 中row 指定的行的记录。

   a) 运行的代码如下：

```
package hdfs_inputSteam;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;

import java.io.IOException;

public class DeleteRow {

    public static long ts;
    public static Configuration configuration;
    public static Connection connection;
    public static Admin admin;

    public static void deleteRow(String tableName, String
row) throws IOException {
        init();
        Table                  table                  =
connection.getTable(TableName.valueOf(tableName));
        Delete delete=new Delete(row.getBytes());
        table.delete(delete);
        table.close();
        close();
    }

    public static void init() {
        configuration = HBaseConfiguration.create();
        configuration.set("hbase.rootdir",
"hdfs://localhost:9000/hbase");
        try {
            connection                                =
		ConnectionFactory.createConnection(configuration);
			admin = connection.getAdmin();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public static void close() {
		try {
			if (admin != null) {
				admin.close();
			}
			if (null != connection) {
				connection.close();
			}
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	public static void main(String[] args) {
		try {
			deleteRow("person", "Score");
		} catch (IOException e) {
			e.printStackTrace();
		}

	}
}
```

b) 验证如下：



```
hbase(main):003:0> scan 'person'
ROW                          COLUMN+CELL
 Score                       column=Score:Computer Science,
 Score                       column=Score:English, timestam
 Score                       column=Score:Math, timestamp=1
1 row(s)
Took 0.0096 seconds
hbase(main):004:0> scan 'person'
ROW                          COLUMN+CELL
0 row(s)
Took 0.0803 seconds
```

对比运行前和运行后的结果，可以看出删除一行数据成功！