设计表保存通话记录，实现查询功能

> **功能需求：**
>
> - **查询某人当月通话记录**
> - **查询某人自定义时间通话记录**
> - **查询某人和某人当月通话记录**

一、题目分析

- 分析了三个需求得出该通话记录表存在四个最基本的列，分别是 **拨打者Call_Tel**、**接听者Ans_Tel**、**通话月份Month**、**通话日Day**
- 这四个基本列可以归纳为两大列族，分别是 **用户信息User_Info**、**通话信息Call_Info**
- 最后用一个表Call_Log封装起来，**以下是表内的通话记录：**

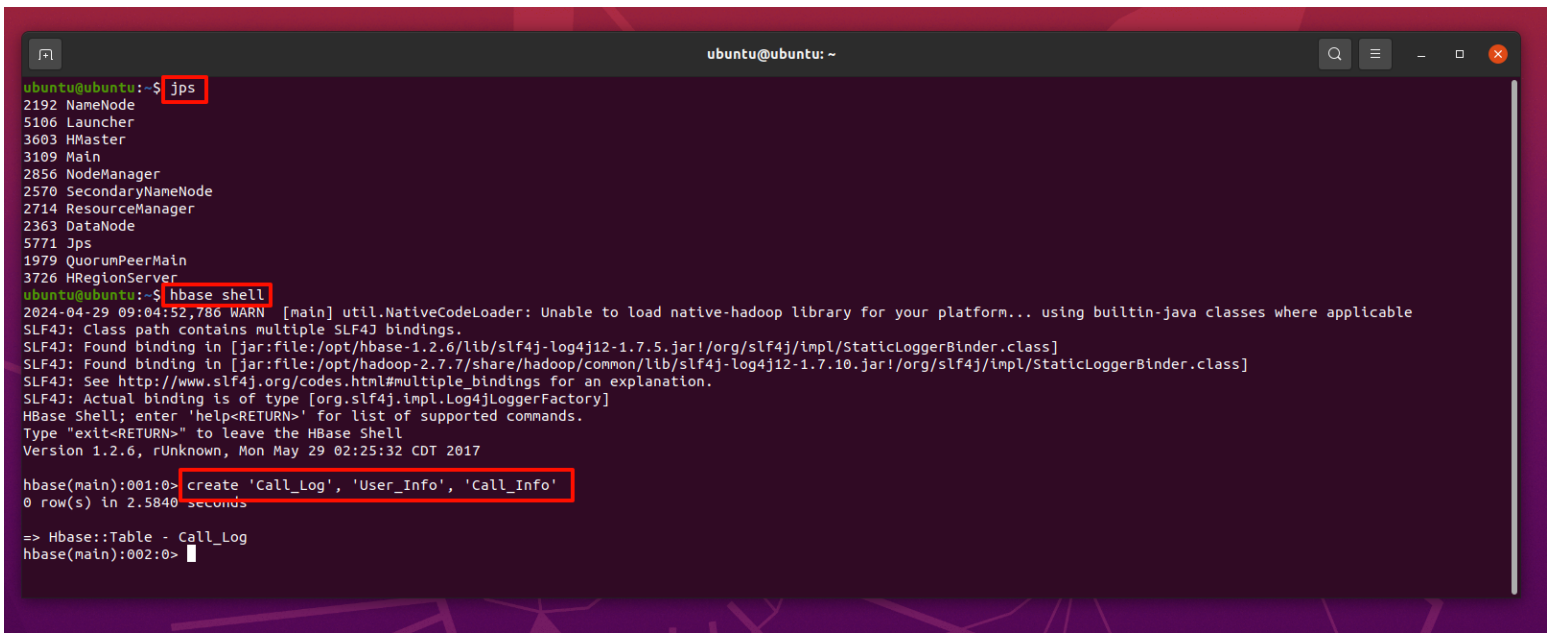
Call_Log					
RowKey	User_Info		Call_Info		时间戳
	Call_Tel	Ans_Tel	Month	Day	
Row1	191	150	4	23	
Row2	150	136	4	20	
Row3	136	180	4	14	
Row4	180	191	3	18	
Row5	150	191	2	30	
• • •					

二、解决问题

1、创建表并插入数据

这里使用HBase shell命令创建表并插入数据，主要是方便一点

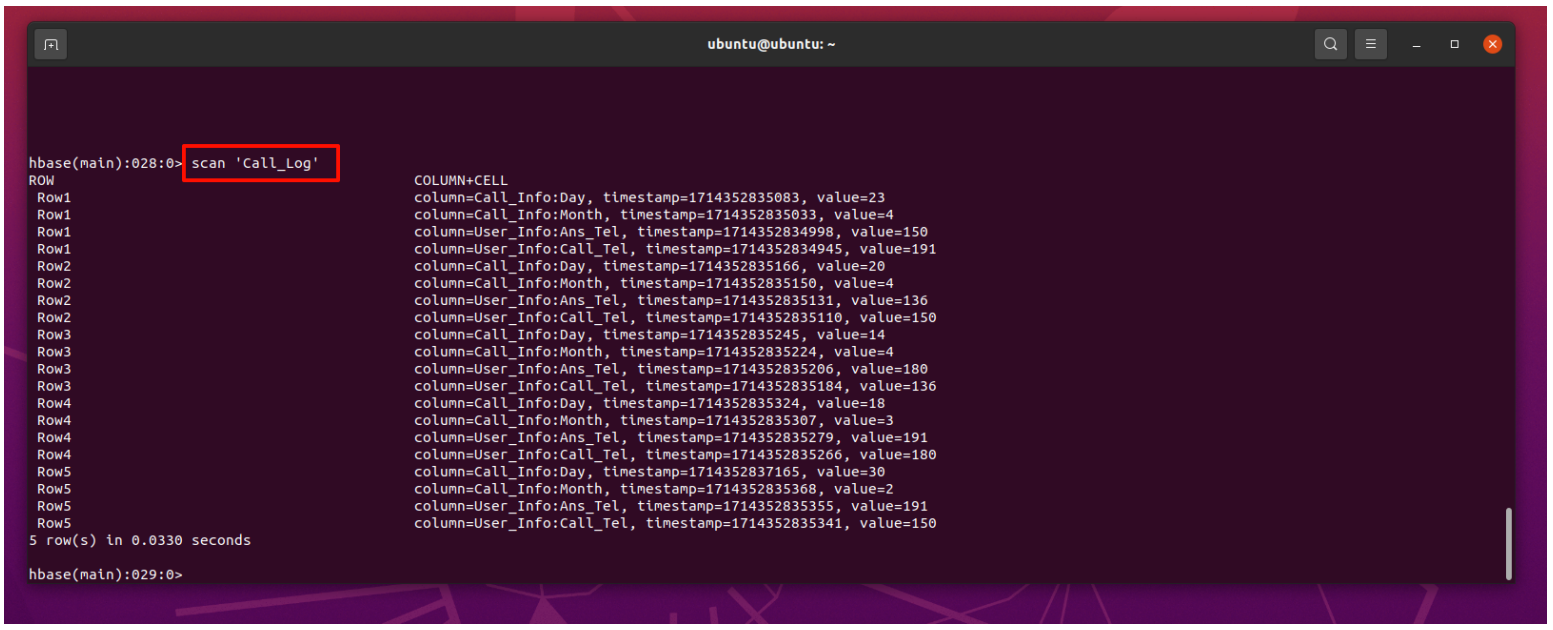
1. 首先启动HBase，然后通过 `hbase shell` 启动HBase的命令模式，并通过`create`和`put`创建表、插入数据操作：




```
# 创建表并指定列族
create 'Call_Log', 'User_Info', 'Call_Info'

# 插入数据
put 'Call_Log', 'Row1', 'User_Info:Call_Tel', '191'
put 'Call_Log', 'Row1', 'User_Info:Ans_Tel', '150'
put 'Call_Log', 'Row1', 'Call_Info:Month', '4'
put 'Call_Log', 'Row1', 'Call_Info:Day', '23'

put 'Call_Log', 'Row2', 'User_Info:Call_Tel', '150'
put 'Call_Log', 'Row2', 'User_Info:Ans_Tel', '136'
put 'Call_Log', 'Row2', 'Call_Info:Month', '4'
put 'Call_Log', 'Row2', 'Call_Info:Day', '20'
```

```
put 'Call_Log', 'Row3', 'User_Info:Call_Tel', '136'
put 'Call_Log', 'Row3', 'User_Info:Ans_Tel', '180'
put 'Call_Log', 'Row3', 'Call_Info:Month', '4'
put 'Call_Log', 'Row3', 'Call_Info:Day', '14'

put 'Call_Log', 'Row4', 'User_Info:Call_Tel', '180'
put 'Call_Log', 'Row4', 'User_Info:Ans_Tel', '191'
put 'Call_Log', 'Row4', 'Call_Info:Month', '3'
put 'Call_Log', 'Row4', 'Call_Info:Day', '18'

put 'Call_Log', 'Row5', 'User_Info:Call_Tel', '150'
put 'Call_Log', 'Row5', 'User_Info:Ans_Tel', '191'
put 'Call_Log', 'Row5', 'Call_Info:Month', '2'
put 'Call_Log', 'Row5', 'Call_Info:Day', '30'
```

2. 数据插入完成后通过 `scan 'Call_Log'` 命令查看插入完成后的表，判断数据无误



2、问题一：查询某人当月通话记录

（1）首先要理清思路

- 函数的传参至少包含查询对象，即某人。同时**通话记录不仅包含拨打电话，也包含接听电话**
- 当月时间获取可以通过`Java`的`java.time.LocalDate.now().getMonthValue()`来获取。
- 查询需要全表扫描，通过引入`scan`类可以进行全表扫描。
- 查询需要过滤操作，通过引入`Filter`类并加以`scan`类即可实现**在扫描数据时进行过滤操作。**
- 在`filter`类中，有一个构造函数为`SingleColumnValueFilter`，参数分别为列族、列名、比较操作符和比较器。
  定义：`public SingleColumnValueFilter(byte[] family, byte[] qualifier, CompareOp compareOp, ByteArrayComparable comparator)`。
- 然后设置扫描器（Scan）的过滤器（Filter）。
- `Table`类中存在一个`getScanner(Scan scan)`方法检索Scan的结果，但是它返回的是`ResultScanner`类的数据，所以需要定义一个`ResultScanner`对象去获取检索结果。 但是注意：当完成对`ResultScanner`对象的使用时，应该调用`close()`方法释放资源并关闭扫描器。
- 最后遍历`ResultScanner`对象的值就可以实现查询某人当月通话记录。

（2）代码实现

```
public void thisMonth(String call_tel) throws IOException {
    Table table = connection.getTable(TableName.valueOf("Call_Log"));
    Scan scan = new Scan();
    int Month = java.time.LocalDate.now().getMonthValue();
    Filter filter = new SingleColumnValueFilter(Bytes.toBytes("Call_Info"), Bytes.toBytes("Month"),
            CompareFilter.CompareOp.EQUAL, Bytes.toBytes("" + Month));
    scan.setFilter(filter);
    ResultScanner scanner = table.getScanner(scan);
    callLog(call_tel,scanner);
    //自定义函数callLog()，用于从给定的ResultScanner中提取通话记录，并输出与指定电话号码相关的记录
    scanner.close();
}
```

（3）运行结果

3、问题二：查询某人自定义时间通话记录

（1）理清思路

- 数据的查找与问题一类似，但本题的侧重点在于对时间进行比价，比如自定义时间范围是2~4，则表示月份都需要在这之间，即2、3、4月。
- 所以就要创建两个过滤器，分别是**大于等于开始月份**和**小于等于结束月份**，并且这两个条件必须同时满足。
- 多个过滤器结合使用就需要**过滤器组合类**`FilterList`，用于将多个过滤器组合成一个逻辑条件来过滤出符合所有条件的数据行。定义：`FilterList(FilterList.Operator operator, List<? extends Filter> filters)`，其中参数`operator`存在两个枚举值：`MUST_PASS_ALL`和`MUST_PASS_ONE`，相当于与和或的逻辑，后面的参数则为过滤器。
- 其余内容是和问题一相同的思路。

（2）代码实现

```
public void whatTime(String call_tel,String startMonth,String endMonth)throws IOException{
    Table table = connection.getTable(TableName.valueOf("Call_Log"));
    Scan scan = new Scan();
    Filter startFilter = new SingleColumnValueFilter(Bytes.toBytes("Call_Info"), Bytes.toBytes("Month"),
            CompareFilter.CompareOp.GREATER_OR_EQUAL, Bytes.toBytes(startMonth));
    Filter endFilter = new SingleColumnValueFilter(Bytes.toBytes("Call_Info"), Bytes.toBytes("Month"),
            CompareFilter.CompareOp.LESS_OR_EQUAL, Bytes.toBytes(endMonth));
    Filter filter = new FilterList(FilterList.Operator.MUST_PASS_ALL, startFilter, endFilter);
    scan.setFilter(filter);
    ResultScanner scanner = table.getScanner(scan);
    callLog(call_tel,scanner);
    scanner.close();
}
```

（3）运行结果

4、问题三：查询某人和某人当月通话记录

(1) 理清思路

- 整体思路其实是和问题一类似的，但是问题一是某人的，所以只需要查同一个人的拨打记录和接听记录，但是问题三是某人和某人，所以双方的身份是相反的，并且要考虑到**可能存在当月两人各拨打过电话**，所以月份过滤之后还要改变用户的筛选条件。
- 前两个问题通过`if (call.equals(call_tel) || ans.equals(call_tel)){...}`的条件即可筛选出同一个人拨打或接听的记录
- **现在需要满足（A拨打，B接听）或者（B拨打，A接听）条件，而每个条件内的两个小条件是必须同时满足的**
- 所以得到`if ( (call.equals(call_tel) && ans.equals(ans_tel)) || (call.equals(ans_tel) && ans.equals(call_tel))){...}`

(2) 代码实现

```
public void callToans(String call_tel, String ans_tel) throws IOException {
    Table table = connection.getTable(TableName.valueOf("Call_Log"));
    Scan scan = new Scan();
    int Month = java.time.LocalDate.now().getMonthValue();
    Filter filter = new SingleColumnValueFilter(Bytes.toBytes("Call_Info"), Bytes.toBytes("Month"),
            CompareFilter.CompareOp.EQUAL, Bytes.toBytes("" + Month));
    scan.setFilter(filter);
    ResultScanner scanner = table.getScanner(scan);
    for (Result result : scanner) {
        String call = Bytes.toString(result.getValue(Bytes.toBytes("User_Info"), Bytes.toBytes("Call_Tel")));
        String ans = Bytes.toString(result.getValue(Bytes.toBytes("User_Info"), Bytes.toBytes("Ans_Tel")));
        if ( (call.equals(call_tel) && ans.equals(ans_tel)) || (call.equals(ans_tel) && ans.equals(call_tel)))
{
            outPut(result);
        }
    }
    scanner.close();
}
```

(3) 运行结果

```
96      public static void main(String[] args) throws IOException {
97          ConnectHBase connectHbase = new ConnectHBase();
98          connectHbase.connect();
99          connectHbase.callToans( call_tel: "150", ans_tel: "191");
100     }
101 }
```

```
运行  ConnectHBase
/opt/jdk1.8.0_301/bin/java ...
========开始连接=======
log4j:WARN No appenders could be found for logger (org.apache.hadoop.security.Groups).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
========连接成功=======
4月20日: 150 --> 191

进程已结束，退出代码为 0
```

5、完整代码

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.*;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.filter.*;
import org.apache.hadoop.hbase.util.Bytes;

public class ConnectHBase {
    public static Configuration conf;
    public static Connection connection;

    public void connect() {
        conf = HBaseConfiguration.create();
        conf.set("hbase.zookeeper.quorum", "127.0.0.1");
        conf.set("hbase.zookeeper.property.clientPort", "2181");
        try {
            System.out.println("========开始连接=======");
            connection = ConnectionFactory.createConnection(conf);
            System.out.println("========连接成功=======");
        } catch (IOException e) {
```

```
            System.out.println("========连接失败=======");
        }
    }

    public void outPut(Result result) {
        List<String> message = new ArrayList<>();
        for (Cell cell : result.rawCells()) {
            message.add(Bytes.toString(CellUtil.cloneValue(cell)));
        }
        System.out.printf("%s月%s日: %s --> %s\n", message.get(1), message.get(0), message.get(3),
message.get(2));
    }

    public void callLog(String call_tel,ResultScanner scanner){
        for (Result result : scanner) {
            String call = Bytes.toString(result.getValue(Bytes.toBytes("User_Info"),
Bytes.toBytes("Call_Tel")));
            String ans = Bytes.toString(result.getValue(Bytes.toBytes("User_Info"),
Bytes.toBytes("Ans_Tel")));
            if (call.equals(call_tel) || ans.equals(call_tel)) {
                outPut(result);
            }
        }
    }

    public void thisMonth(String call_tel) throws IOException {
        Table table = connection.getTable(TableName.valueOf("Call_Log"));
        Scan scan = new Scan();
        int Month = java.time.LocalDate.now().getMonthValue();
        Filter filter = new SingleColumnValueFilter(Bytes.toBytes("Call_Info"), Bytes.toBytes("Month"),
                CompareFilter.CompareOp.EQUAL, Bytes.toBytes("" + Month));
        scan.setFilter(filter);
        ResultScanner scanner = table.getScanner(scan);
        callLog(call_tel,scanner);
        scanner.close();
    }

    public void whatTime(String call_tel,String startMonth,String endMonth)throws IOException{
        Table table = connection.getTable(TableName.valueOf("Call_Log"));
        Scan scan = new Scan();
        Filter startFilter = new SingleColumnValueFilter(Bytes.toBytes("Call_Info"), Bytes.toBytes("Month"),
                CompareFilter.CompareOp.GREATER_OR_EQUAL, Bytes.toBytes(startMonth));
        Filter endFilter = new SingleColumnValueFilter(Bytes.toBytes("Call_Info"), Bytes.toBytes("Month"),
                CompareFilter.CompareOp.LESS_OR_EQUAL, Bytes.toBytes(endMonth));
        Filter filter = new FilterList(FilterList.Operator.MUST_PASS_ALL, startFilter, endFilter);
        scan.setFilter(filter);
        ResultScanner scanner = table.getScanner(scan);
        callLog(call_tel,scanner);
        scanner.close();
    }

    public void callToans(String call_tel, String ans_tel) throws IOException {
        Table table = connection.getTable(TableName.valueOf("Call_Log"));
        Scan scan = new Scan();
        int Month = java.time.LocalDate.now().getMonthValue();
        Filter filter = new SingleColumnValueFilter(Bytes.toBytes("Call_Info"), Bytes.toBytes("Month"),
                CompareFilter.CompareOp.EQUAL, Bytes.toBytes("" + Month));
        scan.setFilter(filter);
        ResultScanner scanner = table.getScanner(scan);
        for (Result result : scanner) {
            String call = Bytes.toString(result.getValue(Bytes.toBytes("User_Info"),
Bytes.toBytes("Call_Tel")));
            String ans = Bytes.toString(result.getValue(Bytes.toBytes("User_Info"),
Bytes.toBytes("Ans_Tel")));
            if ( (call.equals(call_tel) && ans.equals(ans_tel)) || (call.equals(ans_tel) &&
ans.equals(call_tel))) {
                outPut(result);
            }
        }
        scanner.close();
    }

    public static void main(String[] args) throws IOException {
        ConnectHBase connectHbase = new ConnectHBase();
        connectHbase.connect();
        //...
    }
}
```

解释

- `public void connect()`

  用于建立与 `HBase` 的连接。

- `public void outPut(Result result)`

  用于输出查询结果。它接收一个 `Result` 对象作为参数，然后遍历该结果的单元格（Cell），将单元格中的值提取出来，并格式化输出。

- `public void callLog(String call_tel, ResultScanner scanner)`

  用于查询指定电话的通话记录并输出。它接收一个电话号和一个结果扫描器作为参数，然后遍历扫描器中的结果，并提取出指定电话的拨打记录和接听记录。