基于 Android 和 MySQL 的邮件检索

JAVA 作业 2-1613415-潘巧巧

目录

一、整体实现基本情况

1、 问题描述

通过 JAVA 和 MySQL 实现检索安然公司的 51 万封邮件。

2、 实验环境

IDE：IntelliJ IDEA, Android Studio

工具：MySQL, Navicat

服务器：腾讯云

3、 实现简述

本次实验主要是在本地建好 MySQL 表，用 java 程序读取 51 万封邮件并做相应切割后存入数据库中，然后购买了一台腾讯云服务器，在服务器上部署 mysql，使用工具 Navicat 将本地数据库部署上云。再在 Android Studio 上实现 UI 界面，并用客户端直连服务器数据库进行查询操作。

ID/实例名	监控	状态	可用区	主机类型	配置	主IP地址	实例计费模式	网络计费模式
ins-24j5wru0 ubuntu-1GB-gz -3230		运行中	广州四区	标准型S2	1 核 1 GB 2 Mbps 系统盘：高性能云硬盘 网络： Default-VPC	106.52.165.12 (公) 172.16.0.2 (内)	包年包月 2019-11-16 22:19到期	按带宽包年包月计费

实现主要包含 2 个主要 java 文件。

（1） 【IDEA】SaveDataToMysql.java：读取所有文件并进行关键信息的切割，连接 mysql 数据库，将路径（Path）、标题（Subject）、发件人（Author）、收件人

（Addressee）、邮件内容（Content）存入 mysql 中。将 517401 封邮件的信息全部存入 mysql 中。

```
mysql> select count(*) from email;
+----------+
| count(*) |
+----------+
|   517401 |
+----------+
1 row in set (14.20 sec)

mysql> desc email;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| Path      | varchar(500) | NO   | PRI | NULL    |       |
| Subject   | varchar(500) | YES  |     | NULL    |       |
| Author    | varchar(500) | YES  |     | NULL    |       |
| Addressee | varchar(500) | YES  |     | NULL    |       |
| Content   | longtext     | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

MySQL 建表：

```
CREATE TABLE `Email`(
`Path`          VARCHAR(500) NOT NULL,
`Subject`       VARCHAR(500) ,
`Author`        VARCHAR(500),
`Addressee`     VARCHAR(500),
`Content`       LONGTEXT ,
PRIMARY KEY(`Path`));
```

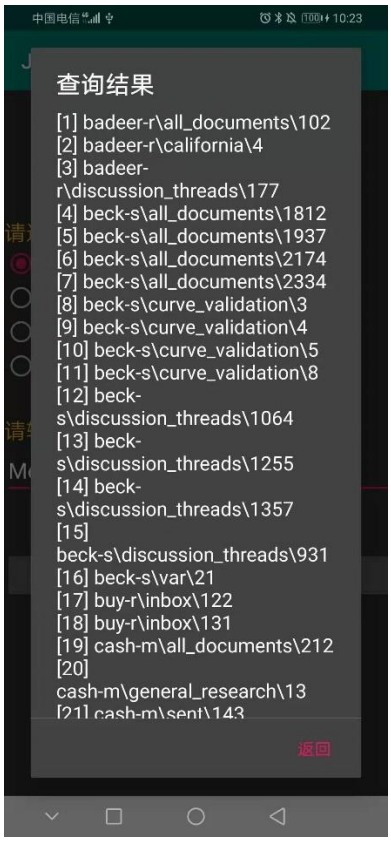
（2） 【Android Studio】MainActivity.java：实现交互界面，用户输入检索类型和检索内容，有未输入的则报错。输入合法则连接服务器上的 mysql 后进行检索，并将结果返回给用户。

4、 效果展示


JavaHomework2
邮件检索
JavaHomework2 by 1613415
请选择检索类型
按邮件标题检索
按邮件发件人检索
按邮件收件人检索
按邮件内容检索
请输入检索内容
点此输入...
确定
检索失败：没有选择检索内容


JavaHomework2
邮件检索
JavaHomework2 by 1613415
请选择检索类型
按邮件标题检索
按邮件发件人检索
按邮件收件人检索
按邮件内容检索
请输入检索内容
点此输入...
确定
检索失败：没有选择检索内容


JavaHomework2
邮件检索
JavaHomework2 by 1613415
请选择检索类型
按邮件标题检索
按邮件发件人检索
按邮件收件人检索
按邮件内容检索
请输入检索内容
Memorandum
确定


查询结果
[1] badeer-r\all_documents\102
[2] badeer-r\california\4
[3] badeer-r\discussion_threads\177
[4] beck-s\all_documents\1812
[5] beck-s\all_documents\1937
[6] beck-s\all_documents\2174
[7] beck-s\all_documents\2334
[8] beck-s\curve_validation\3
[9] beck-s\curve_validation\4
[10] beck-s\curve_validation\5
[11] beck-s\curve_validation\8
[12] beck-s\discussion_threads\1064
[13] beck-s\discussion_threads\1255
[14] beck-s\discussion_threads\1357
[15] beck-s\discussion_threads\931
[16] beck-s\var\21
[17] buy-r\inbox\122
[18] buy-r\inbox\131
[19] cash-m\all_documents\212
[20] cash-m\general_research\13
返回

二、【IDEA】SaveDataToMysql.java

1、 基本情况

包含 3 个步骤

（1） getFiles()：获取所有邮件的绝对路径

（2） getData()：遍历所有邮件，截取关键信息（标题、发件人、邮件内容）。

（3） InsertMysql()：数据插入 mysql 中。

```
public static void main(String[] args) {
    List<File> files = getFiles( path: "data\\maildir");
    List<DATA> dates = getData(files);
    InsertMysql(dates);
```

```
// 获取path下所有文件夹
private static List<File> getFiles(String path) {...}
// 切割各个文件获取关键信息
private static List<DATA> getData(List<File> files) {...}
// 数据插入mysql中
private static void InsertMysql (List<DATA> dates){...}
```

2、 getFiles()

将 path 下所有文件的绝对路径保存在一个 List 中并返回。

```
// 获取path下所有文件夹
private static List<File> getFiles(String path) {
    File root = new File(path);
    List<File> files = new ArrayList<File>();
    if (!root.isDirectory()) {
        files.add(root);
    } else {
        File[] subFiles = root.listFiles();
        assert subFiles != null;
        for (File f : subFiles) {
            files.addAll(getFiles(f.getAbsolutePath()));
        }
    }
    return files;
}
```

3、 getData()

该函数遍历所有邮件，截取有用信息，存成一个 DATA 类的示例，最后返回一个 List<DATA>

```
class DATA{
    String Path;
    String Subject;
    String Author;
    String Addressee;
    String Content;
    DATA()
    {
        this.Path      = "";
        this.Subject   = "";
        this.Author    = "";
        this.Addressee = "";
        this.Content   = "";
    }
}
```

```
// 切割各个文件获取关键信息
private static List<DATA> getData(List<File> files) {
    List<DATA> all_data = new ArrayList<DATA>();
    for (File cur_file : files) { // 遍历所有文件
        String cur_file_path = cur_file.getAbsolutePath();
        try {
            DATA temp_data = new DATA();
            temp_data.Path = cur_file_path;
            FileReader f = new FileReader(cur_file_path);
            BufferedReader buf = new BufferedReader(f);
            String s;
            boolean if_content = false; // 当前是否是正文部分
            StringBuilder temp_content = new StringBuilder(); // 正文部分记录
            while ((s = buf.readLine()) != null) {
                if (!if_content) { // 当前还没当正文部分
                    if (s.contains(":")) { // 如果当前行中包含了冒号
                        String[] tempString = s.split(":"); //按冒号切割
                        if (tempString.length >= 2) { // 切割后至少有 2 个部分
                            switch (tempString[0]) {
                                case "Subject":
                                    StringBuilder temp_subject = new StringBuilder();
                                    for (int i = 1; i < tempString.length; ++i) {
                                        temp_subject.append(tempString[i]);
                                    }
```

```
                                temp_data.Subject = temp_subject.toString();
                                break;
                            case "From":
                                StringBuilder temp_from = new StringBuilder();
                                for (int i = 1; i < tempString.length; ++i) {
                                    temp_from.append(tempString[i]);
                                }
                                temp_data.Author = temp_from.toString();
                                break;
                            case "To":
                                StringBuilder temp_to = new StringBuilder();
                                for (int i = 1; i < tempString.length; ++i) {
                                    temp_to.append(tempString[i]);
                                }
                                temp_data.Addressee = temp_to.toString();
                                break;
                            case "X-FileName":
                                if_content = true; //后面开始是正文部分
                                break;
                            default:
                        }
                    }
                }
            } else  // 正文部分
                temp_content.append(s);
        }
        temp_data.Content = temp_content.toString();
        f.close();
        buf.close();
        all_data.add(temp_data);
    } catch (IOException e) {
        System.out.println("error: " + cur_file_path);
    }
  }
  return all_data;
}
```

4、 InsertMysql()

将 List<DATA>中的数据插入 mysql 中。

```
// 数据插入 mysql 中
private static void InsertMysql (List<DATA> dates){
```

```
    Connection conn = null;
    PreparedStatement  stmt  = null;
    try {
        // 注册 JDBC 驱动
        Class.forName(JDBC_DRIVER);
        // 打开链接
        conn = DriverManager.getConnection(DB_URL, USER, PASS);
        // 执行 sql 语句
        for (DATA data : dates) {
            String sql = "INSERT INTO `Email` VALUES (?,?,?,?,?)";
            stmt = conn.prepareStatement(sql);
            stmt.setString(1, data.Path);
            stmt.setString(2, data.Subject);
            stmt.setString(3, data.Author);
            stmt.setString(4, data.Addressee);
            stmt.setString(5, data.Content);
            stmt.executeUpdate();
            System.out.println("InsertMysql: " + data.Path);
        }
        // 完成后关闭
        stmt.close();
        conn.close();
    } catch(Exception se) {
        se.printStackTrace();// 处理 JDBC 错误
    }
    finally{
            // 关闭资源
            try {
                if (stmt != null) stmt.close();
            } catch (SQLException ignored) {
            }// 什么都不做
            try {
                if (conn != null) conn.close();
            } catch (SQLException se) {
                se.printStackTrace();
            }
        }
}
```

三、【Android】MainActivity.java

1、基本情况

主界面做用户的输入合法性判断，用户选择需要检索的类型（主题、寄件人、收件人、内容）和输入检索的内容后点击“确定”按钮。主线程启动子线程用于查询数据库，子线程连接腾讯云服务器上部署的 mysql 数据库，并发送相应的 sql 语句，服务器返回查询结果后，子线程通过 Handler 给 UI 界面传递信息，并在 UI 上将结果展示给用户。

2、Handler

全局 Handler 用户等待子线程的返回，子线程返回的 msg.what = 1 是正常返回，msg.obj 中存放着检索结果。调用 ShowResult 函数展示给用户。若返回的 msg.what = 2 是不正常返回，调用 Toast 告知检索失败。

```
private Handler uiHandler = new Handler(){
    public void handleMessage(Message msg) {
        switch (msg.what){
            case 1:
                ShowResult((Vector<String>) msg.obj);
                break;
            case 2:
                Toast toast = Toast.makeText(getApplicationContext(), "检索失败", Toast.LENGTH_SHORT);
                toast.show();
                break;
        }
    }
};
```

3、ShowResult()

在控件 AlertDialog 上展示子线程查询服务器上 mysql 查询的结果。

```
public void ShowResult(Vector<String> Info){
    AlertDialog.Builder ResultDialog = new AlertDialog.Builder( context: MainActivity.this);
    ResultDialog.setTitle("查询结果");
    String content = "";
    for(int i = 0;i < Info.size(); ++i){
        content += "["+ (i + 1) + "] " + Info.get(i) + "\n";
    }
    ResultDialog.setMessage(content);
    ResultDialog.setNegativeButton( text: "返回",  listener: null);
    ResultDialog.show();
}
```

4、SearchEmail()

对用户的输入和合法性进行判断，若用户输入不合法，则弹出 toast 进行提示；若用户输入均合法，则拼接 sql 语句并启动子线程进行数据库查询。

```
public void SearchEmail(View v) {
    RadioGroup radio_group_type = findViewById(R.id.RadioGroup);
    int selected = radio_group_type.getCheckedRadioButtonId();
    String SendInfo = "SELECT Path from email where ";
    switch (selected) {
        case R.id.radioButton_title:
            SendInfo += "Subject LIKE \"%";
            break;
        case R.id.radioButton_writer:
            SendInfo += "Author LIKE \"%";
            break;
        case R.id.radioButton_Addressee:
            SendInfo += "Addressee LIKE \"%";
            break;
        case R.id.radioButton_content:
            SendInfo += "Content LIKE \"%";
            break;
        default:
            SendInfo += "NULL";
            break;
    }
    if (SendInfo.equals("SELECT Path from Email where ")) //用户没有选择检索类型
    {
        Toast toast = Toast.makeText(getApplicationContext(), "检索失败：没有选择检索类型", Toast.LENGTH_SHORT);
        toast.show();
        return;
```

```
    }
    EditText EditText_content = findViewById(R.id.editText);
    String content = EditText_content.getText().toString();
    if (content.equals("")) //用户没有输入检索内容
    {
        Toast toast = Toast.makeText(getApplicationContext(), "检索失败：没有选择检索内容", Toast.LENGTH_SHORT);
        toast.show();
        return;
    }
    SendInfo += content;
    SendInfo += "%\";";
    final String final_info = SendInfo;

    new Thread(new Runnable() {
        public void run() {
            DBConnection db = new DBConnection();
            db.linkMysql(final_info);
        }
    }).start();
}
```

5、DBconnection 类与 linkMysql()

类中的常量信息为数据库基本信息

```
private static final String DBDRIVER = "com.mysql.jdbc.Driver";
private static final String DBURL =
"jdbc:mysql://106.52.165.12:3306/test?useUnicode=true&characterEncoding=utf-8&useSSL=false";
private static final String DBUSER = "root";
private static final String DBPASSWORD = "";
```

主要函数 linkMysql()的功能为连接数据库，然后执行 sql 语句并获取查询结果，将结果(邮件的 Path)存放在一个 Vector<String>中,然后通过 sendMessage 返回给 Handler 并告知执行成功。

```
public boolean linkMysql(String SendInfo_sql) {
    Connection conn = null;
    Statement stmt;
    Vector<String> SearchResult = new Vector();
    try {
        Class.forName(DBDRIVER);
```

```
    }
    catch (Exception e){
        e.printStackTrace();
        return false;
    }
    try{
        conn = DriverManager.getConnection(DBURL,DBUSER,DBPASSWORD);
        stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(SendInfo_sql);

        while (rs.next()) {
            String temp_result1 = rs.getString("Path");
            String temp_result2[] = temp_result1.split("maildir"+"\\\\");
            SearchResult.add(temp_result2[1]);
        }

        rs.close();
        stmt.close();
        conn.close();

        Message msg = new Message();
        msg.what = 1;
        msg.obj = SearchResult;
        uiHandler.sendMessage(msg);

        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
    finally {
        if(conn!=null){
            try {
                conn.close();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }
}
```

四、遇到的问题

1、JAVA 导入 mysql 库参考：

https://blog.csdn.net/qq_38000422/article/details/79962314

2、Mysql 中存储邮件内容时，一开始使用 VARCHAR 类型，后来发现有的邮件过长无法存储，改成 TEXT，还是不行，后来改成 LONGTEXT 类型。

3、Java heap space：邮件太多，分 2-3 次录入 mysql。

4、Android Studio 导入 mysql 的 jar 不能导入太高的版本，原先导入 IDEA 的 8 以上版本根本运行不了，后来换成 5 的运行成功。

5、Android Studio 中连接 mysql 需要调用子线程，不能在主线程运行，涉及线程的同步互斥问题，采用了 Handler 的方式解决。一开始尝试忙等待发现不行，暂时还不知道为什么。

6、连接数据库时出现【Access denied for user 'root'@服务器 ip】的问题，应该权限出错，尝试解决未果，直接在服务器中修改配置文件，把 mysql 的拦截关掉了，所有用户直接无密码登录，有风险，暂未解决。

五、改进方案

本实验直接让客户端 APP 操作后端数据库，实际应用中十分危险，应该在服务器中开发后台 java 程序，让客户端和后台通过网络 Socket 编程进行交互，然后后端程序对客户端请求进行合法性判断，并操作 mysql。