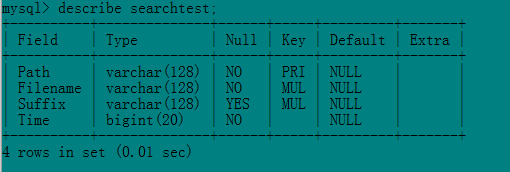
# JAVA大作业实验报告

## 数据库创建与配置

使用mysql建立数据库。源程序所建数据库的名称为filesearch，所建数据表名为searchtest。数据表详细描述如下：



其中Path存储文件的路径，类型为varchar(128)；Filename存储文件名，类型为varchar(128)；Suffix存储文件的后缀名，类型为varchar(128)，可为NULL；Time存储文件最后修改时间，类型为bigint(20)。Path设为主键，另外在Filename和Suffix建立索引。程序以root用户名、root密码连接数据库。

需要指出的是，Path存储的文件路径包含文件名以及后缀。

## 设计思路

1. 更新模块设计思路

由于程序运行时遍历系统文件较慢，若设计成每次查询前更新，将花费很多时间。因此每次程序启动时，将进行数据库的更新。

程序启动时，将遍历根目录下的文件，①如果文件不是文件夹类型，则判断其最后修改时间是否迟于上次程序结束运行时间，若是，则向数据库中添加该文件的信息，此时如果文件已存在于数据库中，由于文件路径被设置为主键，数据库将抛出异常，只需处理该异常即可。②如果文件是文件夹类型，则除了完成向数据库中添加其信息外，还将在数据库中查询该文件夹下的所有文件并判断文件是否还存在，不存在则删除数据库中的信息。遍历以递归的方式进行。需要特别指出的是，无论子文件夹修改时间是否迟于上次程序结束运行时间，都将处理该文件夹。因为我们在测试中发现，一个文件的最后修改时间变化时，其所在的文件夹最后修改时间将发生变化，但再往上的文件夹修改时间不会发生改变。因此仅通过最后修改时间无法判断文件夹中文件是否修改过。

1. 查询模块设计思路

程序查询方式严格按照大作业要求中的以下叙述：

“当提出一个文件查找请求时，将文件查找请求转化为一个SQL查询语句。文件查找请求中可以有通配符，例如“\*.doc”是查找所有扩展名为doc的文件，“D:\\\*.\*”是查找D盘根目录下的所有文件，“ABC”是查找文件系统中所有名为ABC的目录或文件。”

所谓“严格按照”要求，是指当输入的查询请求为“\*.doc”时，程序将查找所有扩展名为doc的文件，而当输入的查询请求为“doc”或“.doc”时，程序将查找所有名为doc的目录或文件，因此查询时需特别注意。

当程序获得查询请求时，①若请求中包含通配符“\*”，程序将在数据库中查询以“\*”之后字符串为后缀名或在“\*”之前字符串所指路径文件夹中的所有文件。②若请求中不包含通配符“\*”，程序将严格按照文件名查询。

1. 图形界面模块设计思路

先构建一个最开始的窗口，该窗口包含JTextField和JButton两个组件，JTextField用于获得将要搜索的文件名，JButton用于捆绑接下来的一系列动作。随后，将搜索得到的结果以列表的形式用另一窗口显示。然后，对于列表中选中的选项，捆绑上鼠标事件监听，当两次点击鼠标左键时进行打开文件，右键时进行弹出菜单。给菜单中的两个选项均捆绑上事件监听，当点击弹出菜单中的“打开”选项时，则进行打开文件；当点击“复制”选项时，先弹出一个文件选择器的窗口用来选取源文件将要备份到的目录，然后再将文件复制到该目录上。复制文件是采用了文件复制的方法，即先获取源文件的文件名，再在选中的目录下创建一个文件名与源文件文件名相同的文件，以此达到将文件从当前目录备份到目的目录的功能。

## 关键代码解释

1. **更新部分**

**public** **void** AddL(String ss,Connection con,Long Lasttime) **throws** SQLException{ //AddL()方法用于更新根目录

Statement stmt=**null**;

stmt=con.createStatement();

String pathh1=ss;

Set<String> depath=**new** HashSet<String>();

String tt=Pattern.*compile*("\\\\").matcher(pathh1).replaceAll("\\\\\\\\\\\\\\\\"); //数据库中正则式查询需进行转义字符的转化

**try**{

String sqlselect="select \* from searchtest where Path regexp '"+tt+"'";

ResultSet result=stmt.executeQuery(sqlselect);

**while**(result.next()){

String ppath=result.getString(1);

File file2=**new** File(ppath);

**if**(!file2.exists()){ //文件不存在则删除它的信息

String tt1=Pattern.*compile*("\\\\").matcher(ppath).replaceAll("\\\\\\\\");

String sqldelete="delete from searchtest where Path="+"'"+tt1+"'";

depath.add(sqldelete);

}

}

**for**(String a:depath){

stmt.executeUpdate(a);

}

}**catch**(NullPointerException e){

}

}

**public** **void** Add(String ss,Connection con,Long Lasttime) **throws** SQLException{

Statement stmt=**null**;

**int** i=0;

Set<String> depath=**new** HashSet<String>();

File file1=**new** File(ss);

File[] templist1=file1.listFiles();

**for**( i=0;i<templist1.length;i++){

**if**(templist1[i].isFile() && !templist1[i].isHidden() &&templist1[i].lastModified()>=Lasttime){

//文件最后修改时间迟于程序上次结束运行时间，则向数据库中添加该文件信息，已存在则不会添加

**try**{

stmt=con.createStatement();

PreparedStatement updateSales;

String updateString="insert into searchtest set Path=?,Filename=?,Suffix=?,Time=?";

updateSales=con.prepareStatement(updateString);

updateSales.setString(1,templist1[i].getPath());

updateSales.setString(2,templist1[i].getName().substring(0,(templist1[i].getName()).lastIndexOf(".")));

updateSales.setString(3,(templist1[i].getName()).substring((templist1[i].getName()).lastIndexOf(".")+1));

updateSales.setLong(4,templist1[i].lastModified());

updateSales.executeUpdate();

}**catch**(SQLException ex){ //文件信息已存在于数据库中时将捕获异常

}

}

**else** **if**(templist1[i].isDirectory() && !templist1[i].isHidden()){ //如果是文件夹的话

**try**{

stmt=con.createStatement();

String pathh1=templist1[i].getPath();

String tt=Pattern.*compile*("\\\\").matcher(pathh1).replaceAll("\\\\\\\\\\\\\\\\");

**try**{

String sqlselect="select \* from searchtest where Path regexp '"+tt+"'"; //查询到数据库中此文件夹下的所有文件

ResultSet result=stmt.executeQuery(sqlselect);

**while**(result.next()){

String ppath=result.getString(1);

File file2=**new** File(ppath);

**if**(!file2.exists()){ //确定此文件夹下的文件是否存在

String tt1=Pattern.*compile*("\\\\").matcher(ppath).replaceAll("\\\\\\\\");

String sqldelete="delete from searchtest where Path="+"'"+tt1+"'";

depath.add(sqldelete);

}

}

**for**(String a:depath){

stmt.executeUpdate(a);

}

}**catch**(NullPointerException e){

}

stmt=con.createStatement();

PreparedStatement updateSales;

String updateString="insert into searchtest set Path=?,Filename=?,Suffix=null,Time=?";

//将此文件夹的信息添加到数据库中，已存在则不会添加

updateSales=con.prepareStatement(updateString);

updateSales.setString(1,templist1[i].getPath());

updateSales.setString(2,templist1[i].getName());

updateSales.setLong(3,templist1[i].lastModified());

updateSales.executeUpdate();

}**catch**(SQLException ex){

} //文件信息已存在于数据库中时将捕获异常

**try**{

Add(templist1[i].getPath(),con,Lasttime); //此处递归调用

}**catch**(Exception e){

}

}

}

}

1. **查询部分**

**class** Search{

**public** Set<String> search(String ss,Connection Scon) **throws** SQLException{

Statement stmt=Scon.createStatement();

Set<String> Sresult = **new** HashSet<String>();

String tt1=Pattern.*compile*("\\\*").matcher(ss).replaceAll("");

String tt2=Pattern.*compile*("\\\\").matcher(tt1).replaceAll("\\\\\\\\\\\\\\\\");

//数据库中正则式查询需进行转义字符的转化

String sqlselect;

**if**(ss.contains("\*")){ //含有通配符的查询

sqlselect="select \* from searchtest where Suffix regexp '("+tt2+"$)|(^"+tt2+")"+"'"+"or Filename regexp '("+tt2+"$)|(^"+tt2+")"+"'"+"or Path regexp '("+tt2+"$)|(^"+tt2+")"+"'";

}

//若请求中包含通配符“\*”，程序将在数据库中查询以“\*”之后字符串为后缀名或在“\*”之前字符串所指路径文件夹中的所有文件。

**else** sqlselect="select \* from searchtest where Filename = \""+tt2+"\"";

//不含有通配符的按文件名查询

System.***out***.println(sqlselect);

ResultSet result=stmt.executeQuery(sqlselect);

**while**(result.next()){

String ppath=result.getString(1);

Sresult.add(ppath);

}

**return** Sresult;

}

}

1. **GUI部分**

**class** GUI{

**private** **static** **void** copyFile(String source, String dest) **throws** IOException { //实现文件的备份，source为源文件路径，dest为目的路径（无文件名）

InputStream input = **null**;

OutputStream output = **null**;

File oldfile=**new** File(source);

String name=source.substring(source.lastIndexOf("\\")+1); //从源路径中取得文件名字

String target=dest+"\\"+name; //将文件名加到目的路径上

File newfile=**new** File(target);

**try** { //实现文件从源路径备份到目的路径

input = **new** FileInputStream(oldfile);

output = **new** FileOutputStream(newfile);

**byte**[] buf = **new** **byte**[1024];

**int** bytesRead;

**while** ((bytesRead = input.read(buf)) > 0) {

output.write(buf, 0, bytesRead);

}

} **finally** {

input.close();

output.close();

}

}

**static** **void** constructGUI(Search s,Connection con){ //构建GUI

JFrame.setDefaultLookAndFeelDecorated(**true**);

JFrame frame=**new** JFrame("File Seclect");

frame.setLayout(**new** FlowLayout());

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JTextField FileNameField=**new** JTextField(20);

JButton button=**new** JButton("Seclect File");

button.addActionListener(**new** ActionListener(){ //为"Select File"添加事件监听

**public** **void** actionPerformed(ActionEvent e){

String FileName =FileNameField.getText(); //获取需要搜索的文件的文件名

Set<String> result = **new** HashSet<String>();

**try** {

result=s.search(FileName, con); //将搜索得到的结果放入一个容器中

} **catch** (SQLException e1) {

// e1.printStackTrace();

}

JList list=**new** JList(result.toArray());

list.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION); //将列表设为单项选中

JPopupMenu popupMenu=**new** JPopupMenu(); //创建弹出菜单

JMenuItem open=**new** JMenuItem("打开"); //为菜单添加“打开”、“复制”选项

JMenuItem copy=**new** JMenuItem("复制");

popupMenu.add(open);

popupMenu.add(copy);

list.addMouseListener(**new** MouseAdapter(){ //给列表添加鼠标事件监听

@Override

**public** **void** mouseClicked(MouseEvent e) {

**if**(SwingUtilities.isLeftMouseButton(e)&&e.getClickCount()==2){ //当鼠标左键被按中且点击两次时，打开文件

File file=**new** File((String) list.getSelectedValue());

**try** {

Desktop.getDesktop().open(file);

} **catch** (IOException e1) {

// e1.printStackTrace();

}

}

}

@Override

**public** **void** mousePressed(MouseEvent e) {

list.setSelectedIndex(list.locationToIndex(e.getPoint())); //获取鼠标点击的项

maybeShowPopup(e);

}

@Override

**public** **void** mouseReleased(MouseEvent e) {

maybeShowPopup(e);

}

//弹出菜单

**private** **void** maybeShowPopup(MouseEvent e) {

**if** (e.isPopupTrigger()&&list.getSelectedIndex()!=-1) {

popupMenu.show(e.getComponent(),e.getX(), e.getY());

}

}

});

open.addActionListener(**new** ActionListener(){ //给右键中的“打开”选项添加事件监听，当鼠标点击时，将打开文件

**public** **void** actionPerformed(ActionEvent e){

File file=**new** File((String) list.getSelectedValue());

**try** {

Desktop.getDesktop().open(file);

} **catch** (IOException e1) {

// e1.printStackTrace();

}

}

});

copy.addActionListener(**new** ActionListener(){ //给右键中的“复制”选项添加事件监听，当鼠标点击时。将复制文件到指定目录

**public** **void** actionPerformed(ActionEvent e){

JFileChooser chooser = **new** JFileChooser(); //利用文件选择器来选取所需的目的路径

chooser.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

chooser.showOpenDialog(**null**);

**try** {

String path = chooser.getSelectedFile().getPath();

File file=**new** File((String) list.getSelectedValue());

File target=**new** File(path);

copyFile((String) list.getSelectedValue(),path); //复制文件

} **catch** (Exception e1) {

// e1.printStackTrace();

}

}

});

JScrollPane scrollpanel=**new** JScrollPane(list);

scrollpanel.setPreferredSize(**new** Dimension(400,250));

JFrame frame1=**new** JFrame("Result");

frame1.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame1.add(scrollpanel);

frame1.setLocation(800, 200);

frame1.setSize(400, 400);

frame1.setVisible(**true**);

}

});

frame.add(FileNameField);

frame.add(button);

frame.getContentPane().add(button);

frame.setLocation(400, 200);

frame.setSize(400, 120);

frame.setVisible(**true**);

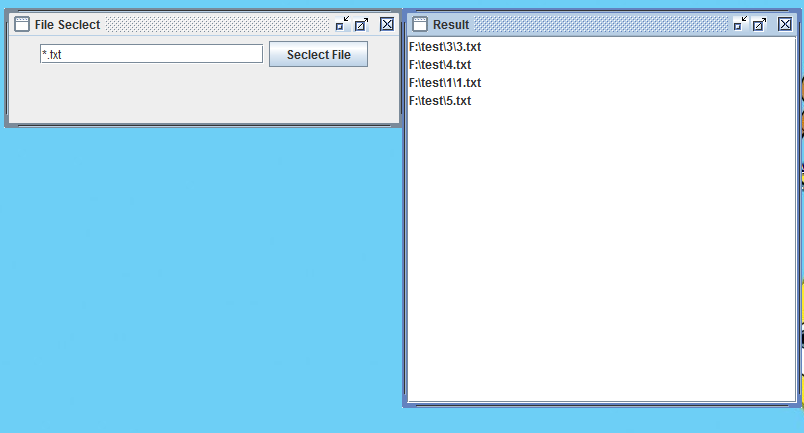
}

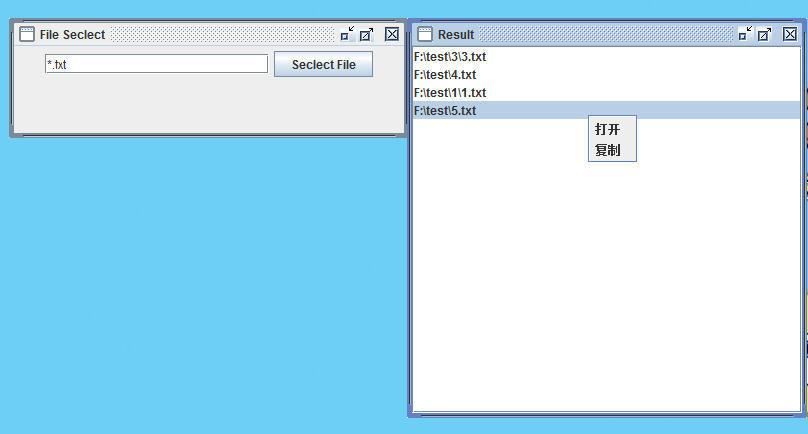
}

1. **其他代码注释见源程序代码**

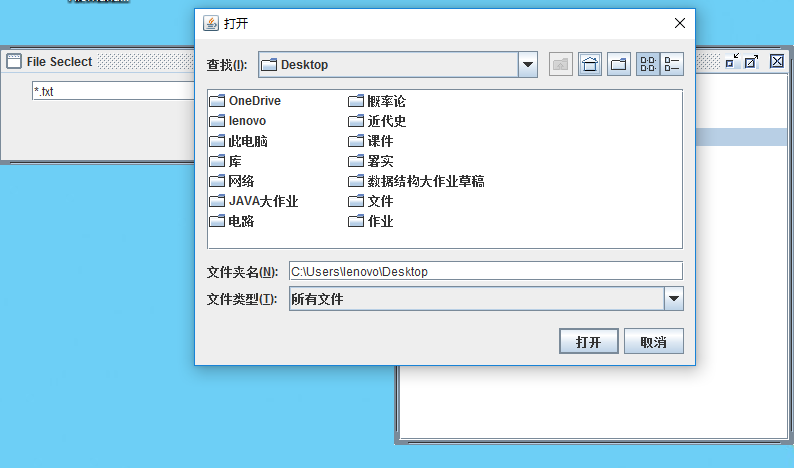
## 程序运行效果图

1、查询结果





2、右键选择复制后出现对话框，选择相应文件夹并单击“打开”即可将文件复制到该文件夹中。



## 使用说明

1. 由于程序更新时遍历系统文件较慢，建议使用一个小文件夹进行测试。被测试的文件夹路径在源程序中的main()方法内，位于程序第55行：String path="F:\\test"。
2. 程序使用date.txt文件存储每次结束运行的时间，注意，此文件由程序在首次运行时自己建立。建立的路径需在源程序中给出，位于main()方法内第42、45、49、60行：

DataOutputStream out = new DataOutputStream(new FileOutputStream("C:\\Users\\lenovo\\Documents\\workspace\\chp01\\date.txt"));

1. 需要修改数据库连接时的url、user、passward。位于connect类内，源程序第223行:

String url="jdbc:mysql://127.0.0.1:3306/filesearch?useUnicode=true&characterEncoding=utf-8";

1. 查询时，查找D盘根目录下的所有文件不是“D:\\\*.\*”，应该是“D:\\*.\*”。