



# 长安大学

## 二〇二〇届毕业设计

网络安全设备数据可视化及应用

学    院：信息工程学院  
专    业：计算机科学与技术  
姓    名：薛硕  
学    号：2016902804  
指导教师：王卫亚  
完成时间：2020年5月18日

二〇二〇年五月



## 摘 要

随着当代网络技术的高速发展，人类已经进入网络化社会。网络已经渗透到社会的方方面面，在教育，交通，医疗，文化交流等方面发挥着越来越重要的作用.网络安全问题愈来愈突出，每年很多企业因为网络安全问题造成的损失也越来越多。应对网络安全问题的方法很多，其中对数据做可视化是一种很好的分析安全问题的方法。此系统就是利用真实的企业日志数据，对数据进行筛选和处理，然后利用可视化技术对数据进行可视化，然后显示出来以达到帮助网络安全人分分析，解决网络安全问题的目的。具体研究内容如下：

筛选处理企业的日志数据，本系统是基于真实的企业网络数据，这些数据具有量大，结构混乱，价值密度低等特点。所以要先对数据进行筛选和处理

根据得到的数据，生成各种细节丰富，内容翔实的统计图，并展现到前端界面中

有效的图片资源更新机制，因为展示的是图片的静态资源，每当数据发生变换时，要有机制及时更新图片。

还要有信息查询机制，用户可以根据自己的需要精准查询自己想要的信息。因此解决自然语言处理等难点

**关键词：**网络安全 可视化技术 网络数据展示

## ABSTRACT

With the rapid development of contemporary network technology, mankind has entered a networked society. The network has penetrated into all aspects of society, playing an increasingly important role in education, transportation, medical care, cultural exchanges and other aspects. Network security issues are becoming more and more prominent, and many companies are losing more and more because of network security many. There are many ways to deal with network security issues, and visualizing data is a good way to analyze security issues. This system is to use real enterprise log data to filter and process the data, and then use visualization technology to visualize the data and then display it to help network security personnel analyze and solve network security problems. The specific research contents are as follows:

1. Filter and process the log data of the enterprise. This system is based on real enterprise network data. These data have the characteristics of large volume, chaotic structure and low value density. So we must first filter and process the data
2. According to the obtained data, generate various detailed and informative statistical graphs and display them in the front-end interface
3. Effective picture resource update mechanism, because the static resource of the picture is displayed, whenever the data changes, there must be a mechanism to update the picture in time.
4. There is also an information query mechanism, users can accurately query the information they want according to their needs. So solve the difficulties of natural language processing

**KEY WORDS:** network security, Visualization technology, Network data display

## 目 录

|                      |    |
|----------------------|----|
| 第一章 绪论.....          | 1  |
| 1.1 选题背景.....        | 1  |
| 1.2 国内外相关研究状态 .....  | 1  |
| 1.3 课题研究的目的与意义.....  | 1  |
| 1.4 本系统的研究方法.....    | 2  |
| 1.5 本文的主要工作 .....    | 2  |
| 第二章 系统分析.....        | 4  |
| 2.1 需求分析.....        | 4  |
| 2.1.1 用户需求 .....     | 4  |
| 2.1.2 系统需求 .....     | 4  |
| 2.2 系统可行性分析.....     | 5  |
| 2.2.1 技术可行性分析.....   | 5  |
| 2.2.2 运行可行性分析.....   | 6  |
| 2.3 系统功能需求分析.....    | 6  |
| 2.3.1 统计图生成模块.....   | 6  |
| 2.3.2 统计图显示模块.....   | 6  |
| 2.3.3 搜索查询模块 .....   | 7  |
| 2.3.4 管理员模块.....     | 7  |
| 2.3.5 系统资源更新模块 ..... | 7  |
| 2.3.6 数据审计模块 .....   | 7  |
| 第三章 系统设计.....        | 8  |
| 3.1 系统设计的目的与原则.....  | 8  |
| 3.2 结构设计.....        | 8  |
| 3.2.1 系统结构设计 .....   | 8  |
| 3.3 系统功能模块设计 .....   | 9  |
| 3.3.1 统计图生成模块.....   | 9  |
| 3.3.2 统计图显示模块.....   | 10 |
| 3.3.3 搜索查询模块 .....   | 10 |
| 3.3.4 管理员模块.....     | 13 |
| 3.3.5 系统资源更新模块 ..... | 13 |
| 3.4 数据库设计.....       | 14 |
| 第四章 系统的实现 .....      | 15 |
| 4.1 系统功能模块设计 .....   | 15 |

|                         |    |
|-------------------------|----|
| 4.1.1 系统界面的实现 .....     | 15 |
| 4.1.2 系统所用数据的获取 .....   | 15 |
| 4.2 面向对象类的实现.....       | 15 |
| 4.3 核心功能的实现.....        | 19 |
| 4.3.1 统计图生成模块 .....     | 19 |
| 4.3.2 统计图显示模块实现.....    | 20 |
| 4.3.3 搜索查询模块的实现 .....   | 21 |
| 4.3.4 管理员模块的实现.....     | 23 |
| 4.3.4 系统资源更新模块的实现 ..... | 25 |
| 4.3.6 数据库审计的实现.....     | 26 |
| 第五章 系统测试.....           | 27 |
| 5.1 系统测试方法.....         | 27 |
| 5.2 类的测试.....           | 27 |
| 5.3 交互测试.....           | 29 |
| 结论与展望 .....             | 35 |
| 1 结论 .....              | 35 |
| 2 展望.....               | 35 |
| 致 谢 .....               | 37 |
| 参考文献.....               | 39 |

# 第一章 绪论

## 1.1 选题背景

自上世纪 90 年代以来, 互联网一直在蓬勃发展, 已经对人类的生产和生活产生了巨大的影响。网络已经渗透到社会的方方面面, 在教育, 交通, 医疗, 文化交流等方面发挥着越来越重要的作用<sup>[1]</sup>。中国是一个互联网使用大国, 互联网在中国社会中的作用更是举足轻重, 教育, 电商, 娱乐等方面都有互联网的身影, 因此网络安全就显得极为重要。而对网络安全设备的数据可视化就是研究网络安全的中还要手段之一。

我们现在使用的网络安全产品, 都是在被动的应对所受到的网络安全问题, 网络安全被破坏, 造成一定的危害及损失。对抗网络安全威胁在形式上一直处于被动挨打的局面, 攻击者有足够的时间进行研究<sup>[3]</sup>。伴随着网络使用的越来越多, 就会导致更多的攻击类型出现, 并且更加复杂, 对网络安全造成了严重的影响。现有的一些网络安全产品防护较弱, 且防御范围小, 无法实现较强的防御功能, 在一些修复软件中, 只能对较为明显的漏洞进行扫描及修复, 无法做到彻底的防护, 在检测软件中, 防火墙适用于对内部网络的运行及外部网络的访问情况进行扫描<sup>[2]</sup>。针对以上分析, 为应对以往的安全产品的缺陷, 提出了网络安全可视化技术, 从而弥补安全防护软件中的不足<sup>[4]</sup>。

Python 是一种面向对象的设计语言, 在 web 后端, 爬虫, 科学计算, 人工智能等多方面有广泛的应用。Python 以其简洁明了的语法和功能丰富且强大的扩展库所著称。在此次课题中所用的主要技术, python 都提供了相应成熟的支持。使用 Python 语言可以完成本次的设计。

## 1.2 国内外相关研究状态

互联网在我们生活中发挥的作用越来越重要, 而当今的互联网安全形势依然严峻, 无论是个人信息的安全和各种对网络企业的安全问题, 都比较严重。因此使用网络安全可视化技术可以为网络安全提供一种更加直观和形象的表现方式<sup>[6]</sup>。在此系统中, 根据的是某个企业真是的日志文件, 因此可以真是反应当前网络安全企业的信息安全状况。企业网络安全工作者可以根据可视化数据的结果, 更加清楚攻击的类型, 攻击源的主要方向等相关网络安全信息, 以及相关的趋势变化。为此, 企业可以采取更加有目的和针对性的防护。

## 1.3 课题研究的目的与意义

计算机技术作为 21 世纪中的标志性技术, 在社会生产中的应用非常广泛, 直接影响人们的生活质量。近一段时期以来, 网络安全问题频繁出现, 给人们生产生活产生了巨大影响。随着信息技术不断发展, 网络安全问题越来越严重, 人们对网络安全问题也越来越重视。人们也对于网络安全问题提出了一系列的监测方法与解决措施, 例如使用防火墙、网关、杀毒软件等, 这些保证网络安

全的机制在很大程度上保证了网络安全。但随着网络病毒制造者的技术水平越来越高，很多网络病毒防御工作更加困难、病毒程序也更加难以破解，从而造成网络安全隐患<sup>[9]</sup>。因此，我么需要更多保障网络安全的方法，其中对网络数据进行可视化是一种很好的分析网络安全的方法，可视化后更加直观，更加便于分析。

在此项目中，使用的企业中真实的日志数据，在日志数据中，详细的记录的企业在某段时间内所受到的各种攻击，所采取的预妨手段，攻击地等。但是日志数据的特点是杂乱无章，数据价值密度低，因此很难直观的从这些数据中获取又价值的信息，因此对数据进行相应的筛选，处理，可视化等操作就显得很有必要，这样才能更好的了解企业目前的网络安全状况<sup>[7]</sup>，从而采取更加有针对性的措施防止网络攻击，这对降低企业的运行成本，提高有针对性的应多网络攻击很有帮助。

## 1.4 本系统的研究方法

在系统的实现中，我使用了 Python 语言。使用了现在主流的面向对象的编程思想。面向对象的程序设计（OOP）是当今主流的程序设计思想。在面向对象的观点中，一切皆对象，因此在此系统中，各个模块均是由对象组成<sup>[8]</sup>，用对象中的组合关系可以实现各个模块的交互。在此系统中用到的各种技术，如数据的筛选和处理，根据数据生成统计图，生成 GUI 界面等其他一些关键技术一些来自于 python 强大的库，一些来自于自己的编写。在此系统中，最重要的就是如何根据数据库中的数据生成相应的统计图，以及如何以合理的方式对统计图进行显示。在编写代码的初始阶段，就生成课各种统计图，保存在文件中，避免每次启动都要重新生成，降低程序的启动速度。

然后就是如何在 GUI 界面中显示这些图片，在此项目中，前端界面使用的是信号和槽技术。可以使用 Label 展现图片，但是可能有多幅图片要展示，所以要自己重新定义 Label，添加相应的信号。

还有一个问题是统计图片的更新问题。在此项目中，会有管理员模块，如果管理员更改了数据，管理员有权决定立即刷新图片，如果不立即刷新，项目会开启线程。30 分钟后跟新。并做相应标记。在开启程序前，会检查标志。如果发现为更新，会在展示前更新图片。用面向对象的技术整合这些模块，最终实现整个数据可视化系统。

## 1.5 本文的主要工作

（1）网络日志文件的处理，因为本项目使用的是真实的企业日志数据，所以要首先筛选日志文件，提取文件中的信息，然后将提取的信息保存到数据库中，为生成统计图做数据

（2）生成统计图，查询数据库中的信息，查询出合适的的数据，并对数据的格式等进行转换，然后根据要生成可视化对象的特征，生成合适的统计图，比如扇形图，条形图，散点图等，还有统计图要添加合适的标注，生成出尽可能清楚，漂亮的图片

（3）在此系统中，搜索功能也是很重要的功能，用户可能需要查看一些具体的数据，所以就要使用搜索功能，在搜索功能中，最主要的功能就是使用分词算法，提取关键字，完成查询。

（4）统计图更新功能，因为在系统中有管理员模块，所以数据库中数据可能被更改，多线程



图片更新功能也是必须的。

(6)编写人机友好的界面

## 第二章 系统分析

### 2.1 需求分析

#### 2.1.1 用户需求

本系统在在界面方面要有简洁的界面，以简单美观的界面表现出尽可能多的信息，还要求程序有健壮性，不能出现程序崩溃等问题。能够满足基本的数据可视化和以及其他的主要功能。同时简洁操作界面可以减少系统的复杂程度，减小用户上手难度，节约时间，并且有利于该系统的推广。用户只需要点击一些按钮来实现界面的切换，获取到自己想要的数据的信息，如果用户想要查询一些具体信息，只需要在查询窗口输入所要查询的关键字即可。

还有就是数据一定要真是，这样才能真是的反应企业的网络安全状况，这样才能有可信度和价值。在系统中所使用的数据一定要足够，如果使用的数据不多，数据没有代表性和说服力。用户不能接受。所用的数据也不能过于巨大，如果使用过多的数据。会对数据的处理带来极大的负担，而且可能会引入过量的“脏数据”，影响结果的准确性。对数据的处理也要使用合理的方法，

不可漏过数据，丢失数据等。把处理好的数据要储存在数据库中，这样使用时直接查询数据库，比较方便。免去了重复筛选数据带来的系统性能的损失，可以提程序的响应速度。提高用户的使用体验。

对于系统的主要功能。显示统计图，要做好细节的处理，在统计图上，显示好各图例，能让用户看的明白。因为某一方面的统计图可能不只有一张，所以当用户点击图片时，就可以更换图片，用户由此可以看到连续几个月的变化情况。用户需求中，很重要的一点是时效性，所以用户看到的图片必须是最新的。因此在管理员更改数据后，管理员可以选择手动更新图片。如果不跟新，系统默认 30 分钟后自动更新，如果 30 分钟后更新失败，在下一启动时就会更新。这样设计就会基本保证用户看到的都是最新统计图。

#### 2.1.2 系统需求

本系统由 python 语言进行编写，python 已其丰富强大的扩展库著称。在此系统中使用的是强大流行的图形界面开发库,PYQT.他可以满足本系统所有的有关界面的要求。同时 Python 作为面向对象的语言，采取每个类单独封装的方式，可以更加简单与 容易的编写该仿真平台，同时也有利于项目的后期修改与维护。从下面三个方面对该仿真系统进行需求分析

##### （1）功能需求

功能需求是有关软件系统的最基本的需求表述，用于说明系统应该做什么，该系统主要是根据企业真是的日志数据进行筛选和处理，并利用可视化技术将他们展现出来。并能实现图片的切换，更新等。更新操作是利用多线程的方式更新，并在多线程中加入了线程同步机制，保证在多线程环

境下顺利运行。同时还加入了相关信息查询，数据库管理员等模块，该系统可以做查询，对数据库的处理。

此系统使用的是当前流行的 MVC 架构模式<sup>[9]</sup>。在此系统中分为三层，第一层名为 Dao 层，用于和数据库的交互。比如查询数据，增加数据，删除数据的代码都在这一层，这一层向下完成所有对数据库的操作，向上屏蔽所有的底层细节，只负责提供数据。第二层为 Service 层。这一层负责生成各种统计图，实现前端界面的一些逻辑等。这一层向下，需要 Dao 层提供数据，向上为前端界面提供给服务。第三层为 view 层，负责生成前端界面。View 层只需要和 service 层沟通。获取 service 提供的服务。在 service 获取数据中，有的统计图需要对数据的格式进行一些复杂的变换，在不修改 Dao 层代码的基础上，又为了保持 Service 代码的单一职责原则，因此在 service 和 Dao 层之间又加了一层 Modify。这一层主要是用于 数据格式的转换。这就是系统的基本架构

### (2) 数据需求

此系统对数据的要求较为严苛。因为被系统的实用性和真实性都取决于所用的数据。所以真是可靠的数据在此系统中绝对重要。此系统中的数据均为真实的企业数据，在经过 筛选后存入了数据库中。在此系统中只有管理员可以更改数据，但是每次更改，都会记录下修改人，修改时间等信息，尽最大可能保证数据的安全。

### (3) 其他需求

为保证界面的整洁性，因此可以选择将按钮设置为小方格图片的样子，鼠标悬浮后可以提示该按钮能够实现什么功能，显示区域尽可能宽广，同时变现形式简洁，易于用户直接抓住主要对象进行问题的分析。同时也需要信息交互窗口，可以完成对整个系统的观察。

## 2.2 系统可行性分析

当我们研究一个项目时，首先应该解释我们的目标和任务，然后我们应该仔细研究该项目，并将开发阶段进入项目开发的第一阶段。我们从以下几个部分进行可行性的研究。

### 2.2.1 技术可行性分析

面向对象具有维护简单、质量高、效率高、可扩充性等特点，其实质是以建立模型体现出来的抽象思维过程和面向对象的方法。模型可以描述客观事物。但是任何一个模型都不可能反映客观事物的一切具体特征，只能对事物的特征做出抽象，且在它所涉及的 范围内更普遍、更集中、更深刻地描述客体的特征。通过建立模型而达到的抽象是人们对客体认识的深化<sup>[10]</sup>。

同时 python 语言是纯面向对象的语言，并且提供了多种内置的 API，可以帮助我们更好的进行面向对象的开发，尤其是其跨平台的特点，可以让仿真平台在多种计算机平台上运行，增加了平台的兼容性。

Python 语言提供了多种做 UI 界面，如 python 自带的 Tkinter，还可以使用其他的优秀框架，如 PyQt。PyQt 是强大的 GUI 库，用他可以做出精美的界面。

因此此系统使用 python 完全可以实现所要求求得功能。因为 python 简洁的语法，可与让我们用更多的精力去思考架构，算法，内部的优化等。而不必拘泥于繁杂的像 c++那样的语法。

### 2.2.2 运行可行性分析

随着计算机网络技术已经渗透到社会的方方面面，网络数据也开始大量的增长，计算机网络技术的安全性也越来越重要。同时要最大限度地保证计算机网络数据的安全、抵制恶性网络攻击、毁坏信息的行为。也是当今网络技术的一大难题。网络安全数据可视化就是在这种社会背景下产生的，笔者将结合网络安全数据可视化的特点展开分析和论述，希望能够找到问题的突破口，起到一定的参考价值<sup>[11]</sup>。

## 2.3 系统功能需求分析

本系统主要完成的是数据可视化，所以生成各种统计图，并在图形化界面上显示是被系统最主要的功能。除此之外，还有基本信息显示功能，关键信息查询功能，数据管理功能等。本系统的总体结构功能图见图 2-1 所示。

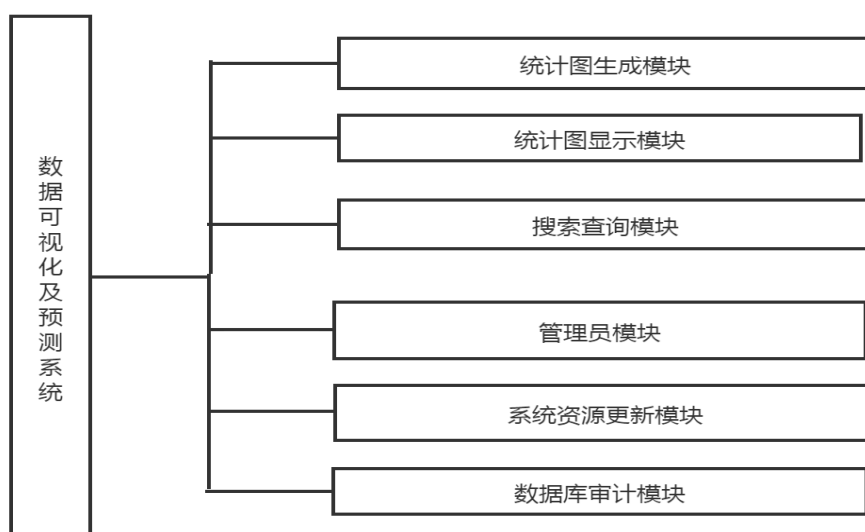


图 2-1 系统功能模块图

### 2.3.1 统计图生成模块

在这个模块中，主要负责本系统统计图的生成。针对这一“事务”，可以把他抽象成类。这个类中就负责统计图的生成。每生成一个统计图，就会存在资源文件中。为服务模块调用做准备。

### 2.3.2 统计图显示模块

此模块包括前端界面。在前端界面中，最主要的部分就是显示出生成模块生的各种统计图。还有一些关键信息的显示。因为每一个主题都不只一幅统计图，所以在用 label 标签显示图片时，不能直接使用原生的 label，需要自己重新设计，添加点击的信号，这个可以用面向对象技术中的继承来解决。这样，你点击图片，就会切换下一张图。便于观看。

### 2.3.3 搜索查询模块

在此模块中实现的最主要的功能就是根据用户的输入语句查询相关信息。在搜索查询模块中会用到一个关键的算法，就是分词算法。在此系统中，这个模块实现的原理是，利用分词算法对用户的输入语句进行分词。从句子中分出所有词汇，用所分出的词进行关键字的匹配。匹配出关键词就可以查询出相应的信息。然后就是查询出信息的显示。这里还用到了分页技术

### 2.3.4 管理员模块

在此系统中，所使用的数据是很重要的。因此有管理员对数据库的维护也是正常合理的需求。在此模块中会以树状图的形状显示出此数据库中的全部表，和表中的字段。因为可能有些表名过长，所以当点击某一表名和字段时，旁边会有完整的信息提示。管理员模块针对的是有专业技能的管理员，有专业技能，可以写 sql 语句。所以对数据库的操作是以 SQL 语句的形式，如果对数据库操作成功会提示成功，如果失败会提示失败。在写 SQL 语句的区域，只需要点击右键执行即可执行，点击清空就会清空。当前管理员所做的操作都会记录到数据库中，以防止数据被恶意篡改。

### 2.3.5 系统资源更新模块

因为数据库中的数据可能被刮管理员添加，删除或者修改。所以对应的前端界面统计图的显示一要更新。解决此问题使用的方法是标志法。当后台发现管理员已经更改了数据时，系统就会在资源文件中做上标志。例如写上 1，表示需要更新。在另一个函数中，就会开启定时线程，30 分钟后，就会调用刷新函数，去刷新静态资源。此处延长一段时间再去更新是基于这样的考量。如果系统发现管理员更新了树马上就更新资源，而此时的管理员的更新操作还未完成，在这样的情况下，系统会一直更新，会极大降低系统性能。如果这个系统更新完毕，会把文件中的标志位重新置为 0，表示系统需要更新，系统在每次启动前检查此标志位，如果需要更新则更新。还有一点就是线程同步问题。读写标志位时，只允许一个线程操作此标志位。所以要做线程同步。

### 2.3.6 数据审计模块

在此模块中，主要是显示对数据库的所有操作，包括系统和管理员。因为此系统数据 安全很重要，所以有必要记录每一次对数据库的操作。

## 第三章 系统设计

### 3.1 系统设计的目的与原则

网络安全数据可视化系统主要的目的是在网络安全越来越重要的大背景下，使用真是的企业日志数据，对真是的企业数据进行筛选，可视化以达到合理的评估目前企业的网络安全状况，达到有针对性预防攻击的目的。本系统会尽可能向用户显示形式多样，及时准确的信息。依托 Python 的语言的面向对象的设计，可以合适的完成对这套系统的模拟，在使用 Python 面向对象设计时，应精确把握面向对象的概念，把握应当设计的每个对象的功能与意义，做好类与类之间松耦合，尽量减少全局变量的使用，增加每个类之间的独立性。同时对每个类内部的尽可能的进行保护，在没有访问特定的方式时，减少对于本类中变量的访问。设计的每个类尽可能实现功能的独立，满足单一职责原则。

交互界面设计的友好、简洁、易懂，基于用户尽可能多的提示，防止用户的操作不当而造成的仿真结果的错误，达不到用户预期的要求。

### 3.2 结构设计

#### 3.2.1 系统结构设计

在此系统的结构设计中，参考的 web 项目中 MVC 的三层架构，此项目也是采用的类似 MVC 的三层架构。

MVC 的英文是 Model\_View\_Controller,这是 web 项目中常见的架构方式，将 web 应用的流程分为输入，处理，输出三步，这样应用也被分成三层——模型层、视图层、控制层。

视图（View）是前端界面，负责和用户的交互，在 Web 应用来说，可以概括为 HTML,或者 jsp 技术展示的页面<sup>[12]</sup>。随着网站规模越来越大，复杂性越来越高，对界面的处理也变得越来越复杂。同一个应用会有不同的视图。MVC 设计模式中 View 层主要作用是数据的展示和处理，还要处理用户的请求。但是 View 层并不包括业务上的处理，业务的处理主要交给下层 Model 处理<sup>[13]</sup>。比如，我们有一个业务，View 层展示 Model 层的数据，并接受用户的操作和数据，将其传递给 Model 层。

模型(Model)：业务层，也叫持久层，主要作用是处理应用中的各种业务逻辑。业务流程对于其他来说并不可见，他只需要为其他层提供服务即可。Model 层接受 View 层传入的数据，对数据进行处理，并返回最终的处理结果。Model 层是整个 MVC 设计模式的核心，和 EJB 模型类似。MVC 模型从技术实现角度对此模型进行了进一步划分，这样可以充分利用现有的组件。对于开发者来说，减少了开发难度，可以使其更加专注于业务模型的设计。MVC 是很优先的组织架构方式。把应用的模型按一定的规则抽取出来，抽取的层次很重要，这也是判断开发人员是否优秀的设计依据<sup>[14]</sup>。抽象与具体的距离要适中。MVC 设计模式没有提供模型的设计方法，但是告诉了你应该如何组织管理

这种模型，这样便于模型的重构以及提高重用性。

控制层（Controller）可以理解为从用户接受请求，将 Model 和 View 连接在一起，共同组成整个应用。划分控制层的目的就是清楚的表明，他它就是一个分发器，选择什么样的模型，选择什么样的视图，可以完成什么样的用户请求<sup>[15]</sup>。控制层不做树的处理，他只接受请求。比如，当用户发出一个请求时，控制层接受到这个请求，但是他并不处理，只把信息传递给 Model 层，即持久层。告诉此层要做什么，选择符合要求的视图返回给用户。因此，在这种设计模式下，一个 Model 可能对应多个 View，一个 View 也可能对应多个 Model。

在经典 MVC 三层架构的基础上，结合项目基本情况，又增加了一个数据处理层。此层主要负责对持久层提供的数据进行格式上的转化，以便可以直接为服务层提供可用的数据。

本系统的架构图如图 3-1 所示。

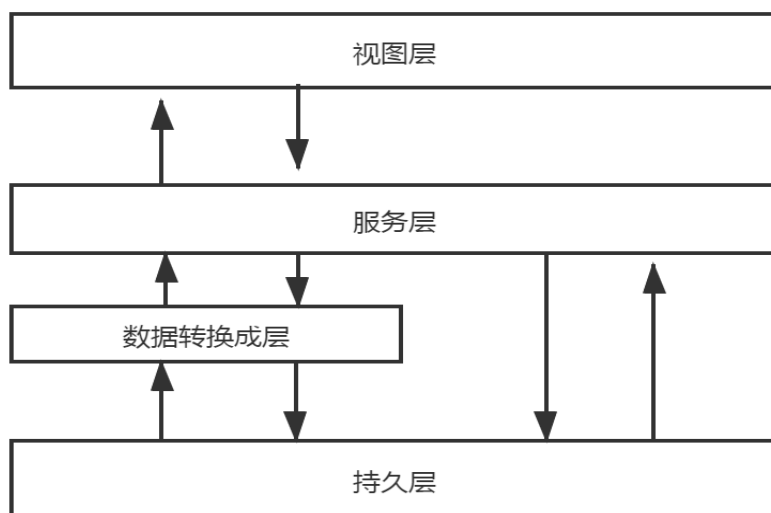


图 3-1 系统架构图

在参考 web 项目 MVC 架构的基础上，此系统主要分成四层。第一层是视图层，主要负责和用户的交互，以及一些静态资源的展示。第二层是服务层，此层主要处理此系统的业务逻辑，主要的功能包括各种统计图的生成，登陆，数据库管理员等方面的逻辑，以及系统资源更新策略等。第三层是数据转换层，此层获取持久层的数据，对数据提供形式上的转换，为服务层提供合适的数。第四层是持久层，此层主要负责和数据库的交互。

### 3.3 系统功能模块设计

#### 3.3.1 统计图生成模块

统计图生成模块是本系统的核心模块。其包括的类主要有持久层中的获取数据的类，以及服务层中的主服务类，此类负责生成上层图片，在此类中生成图片后就会存入资源文件中，这样就可以保证主界面以最快的速度加载图片。在前端界面中，使用 label 标签显示图片，并且还在 label 中连接了他的槽函数，没到点击图片时，相应主题的图片的就会循环显示。此系统启动时，会检查标志

位，是否应该更新图片资源。此模块的功能设计如图 3-2 所示。

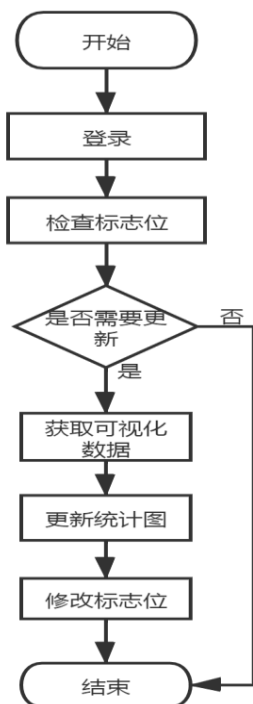
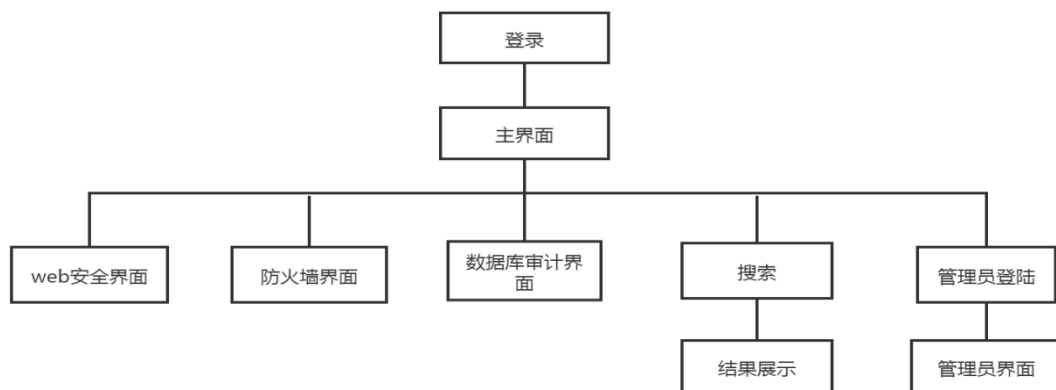


图 3-2 统计图生成界面流程图

### 3.3.2 统计图显示模块

在此模块中，主要就是前端界面的展示模块，在此模块中包含众多的界面，以及各种信号和函数，此模块的关系图如图 3-3 所示。



3-3 系统界面关系图，展示此系统的界面关系

在此系统界面中，各个界面不是孤立的，可以互相跳转

### 3.3.3 搜索查询模块

在此模块中，主要是用户可以根据自己的意愿选择相关信息，所以需要用到查询技术。在查询技术中所用的关键技术为分词技术。在此系统中使用的是一个名叫 JieBa 的扩展库，使用这个库很



容易实现分词的功能。JieBa 库中使用的分词算法为：

在介绍此系统使用的算法之前，先介绍一种要使用数据结构，Trie 树。Trie 字典树的主要功能是存储字符串，Trie 树的存储结构是一个一个的节点 Node，每个 Node 保存一个字符。然后用链表的形式保存整个词，每个 Node 都保存了他的所有子节点。

例如我们往字典树中插入 see、pain、paint 三个单词，Trie 字典树如图 3-4 所示

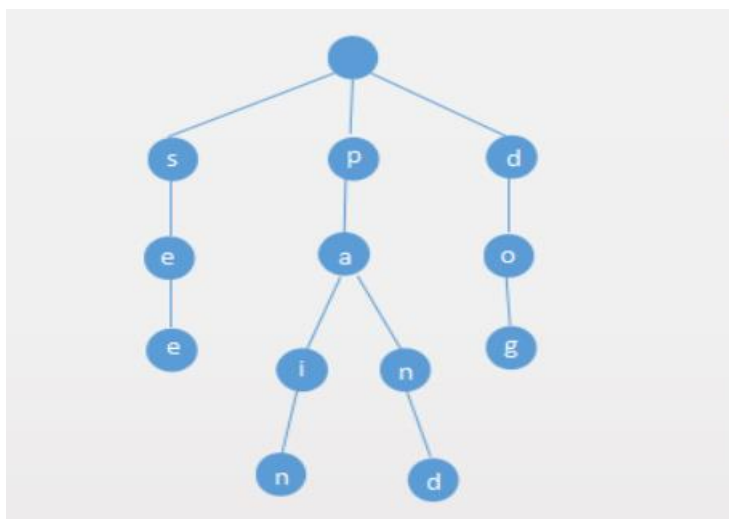


图 3-4 Trie 树

然后接着说分词算法

基于 Trie 树结构实现高效的词图扫描<sup>[4]</sup>，这样做的目的是实现对自然语言的分词。JieBa 分词库自使用了一个叫 dict.txt 的词典，这个词典包含了众多的汉语词汇，在词典中还保存了每个词出现的次数以及他是什么词，名词，动词或者形容词等。这个词典就是把作者收集的 2 万多个此组成一个 trie 树，trie 树是一个前缀树，所谓前缀就是一些词的前几个字相同，在 trie 中存储时，前缀相同的汉字使用相同的节点，因此这样会有更快的检索速度以及更少的空间。DAG 的意思是有向无环图，在分词时，对给定的句子，让其生成有向无环图，这样可以避免了重复的分词。然后说一下分词的步骤：

给定一个自然语言的句子，根据 JieBa 分词自带的词典，生成有向无环图。具体来说就是根据模块给定的词典，在句子中找出包含在词典中的所有的词，也就是对句子进行切分。在此算法的实现中，源码记录的是在句子中，某个词开始的位置，从 0 到 n-1(n 为句子的长度)。然后将每个位置记录下来，记录的格式是一个词典。Key 值是某个词开始的位置，value 值是一个列表。列表中存放的也是句子中某个字的位置，这些位置是可以和 key 值组成词的位置。例如：{0:[1,2,3]} 这样一个简单的 DAG，就是表示 0 位置开始，在 1,2,3 位置都是词，就是说 0~1, 0~2, 0~3 这三个起始位置之间的字符，在 dict.txt 中是词语。

结巴库中的词典不可能收录所有的词，所以对于未收录的词，在结巴库中使用了有生词能力的 HMM 模型，使用 Viterbi 算法生成未收录词。什么是 HMM 模型呢，所谓 HMM 模型，文词汇按照 BEMS 四个状态来标记，B 代表开始 begin 位置，E 代表 end，是结束位置，M 代表 middle，是中间位置，S 是 single，单独成词的位置，没有前，也没有后。也就是说，他采用了状态为(B,E,M,S)这四种状态来标记中文词语，比如上海可以标注为 BE，即 上/B 海/E，表示“上”是开始位置，“海”是结束位置，

长安大学可以标注为 BMME, 就是开始, 中间, 中间, 结束. 经过作者对大量语料的训练, 得到了 `finalseg` 目录下的三个文件(来自结巴项目的 `issues`): 要统计的主要有三个概率表: `prob_trans.py` 位置转换概率, 即 B (开头), M (中间), E (结尾), S (独立成词) 四种状态的转移概率;

```
{'B': {'E': 0.8518218565181658, 'M': 0.14817814348183422},
'E': {'B': 0.5544853051164425, 'S': 0.44551469488355755},
'M': {'E': 0.7164487459986911, 'M': 0.2835512540013088},
'S': {'B': 0.48617017333894563, 'S': 0.5138298266610544}}
```

$P(E|B) = 0.851$ ,  $P(M|B) = 0.149$ , 这表明当我们处于一个词的开头时, 下一个字是结尾的概率要远高于下一个字是中间字的概率, 这符合我们平时的经验, 因为二个字的词比多个字的词更常见。

综上, JieBa 分词的过程为

1. 加载字典, 生成 trie 树

2. 对于一个句子, 使用正则获取连续的中文字符和英文字符, 切分成短语列表, 对每个短语使用查字典和动态规划, 得到最大概率路径, 对 DAG 中那些没有在字典中查到的字, 组合成一个新的片段短语, 使用 HMM 模型进行分词, 也就是作者说的识别新词, 即识别字典外的新词<sup>[21]</sup>。

3. 使用 python 中的 `yield` 生成一个迭代器, 这样可以占用较少的内存空间, 把每个词逐个返回。

此模块主要完成的搜索功能, 在输入框中输入自然语言, 点击搜索, 搜索完成后, 相关信息就会显示出来。此内部的流程为, 点击搜索后, 后台获取到输入的自然语言, 然后利用分词算法对语句进行分词, 生成关键字列表。在系统后台, 根据数据库的信息已经存在一个关键字列表。此关键字列表会映射相应的 SQL 语句, 用前一个关键字列表和后一个关键字列表做比对, 如果有相同的, 取出相应的 SQL 语句, 交给持久层获取数据。当查询的数据返回后, 再经过服务层数据格式的转化, 最后显示到界面显示。系统流程图如图 3-5 所示。

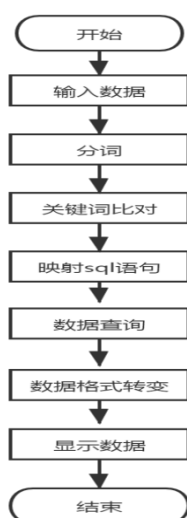


图 3-5 搜索查询流程

### 3.3.4 管理员模块

管理员模块主要负责对本系统所依赖的数据的管理。由于管理员模块针对的是专业技术人员，所以是用 SQL 语句的方式操作数据。在此模块中，主要有本数据库信息的展示。在图形界面的右边，这个数据库以树状图的形式展现数据库中的所有表结构，包括各个表的字段名，字段类型等。由于有的表名过长不方便观看，所以在这些控件上添加了信号以及对丁的槽，没每到点击它时，就会显示完整的字段信息。然后又一个输入框，在输入框中可以写相应的 SQL 语句，然后右键鼠标，点击执行就会把结果显示到输出栏中，对于像增加，删除，修改，只会显示操作是否成功。另外在输入框中，每一条 SQL 语句占用一行，点击执行都会执行最后一条。右键清楚开可以清理界面。本系统的流程图如图 3-6 所示

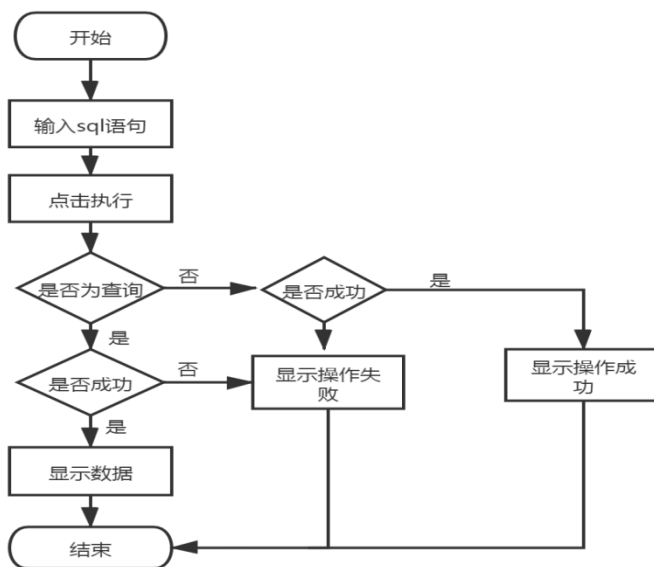


图 3-6 管理员模块流程图

### 3.3.5 系统资源更新模块

本系统是面向数据的，数据直接关系着统计图的真是性和时效性。在本系统中数据可能被增加，删除，或者修改，因此就需要在数据发生改变时，要更新本系统的资源，本系统使用了多种方法保证资源即使更新，第一种方法就是管理员主动更新。在管理员修改数据后，可以点击按钮更新数据。在管理员修改数据且没有更新资源，系统会记录是谁修改了数据，以及修改的时间，最重要的是会修改本系统是否要更新资源的标志位。注意这里资源标志位的更新需要做线程的同步，只允许一个线程读和写本标志位。如果管理员未更新，系统就会开启线程，在定时 30 分钟后自动刷新。并且修改相应的标志位。系统自动更新延迟 30 分钟的目的就是防止系统频繁的做更新操作，影响系统性能。如果在系统自动更新前关闭了系统，那么在此系统前会检查此标志位，如果此时标志位显示需要更新，那么就会更新图片。并且修改标志位。修改标志位要做同步。经过这三层的保障后，得到的图片资源就是最新的。逻辑图见图 3-7

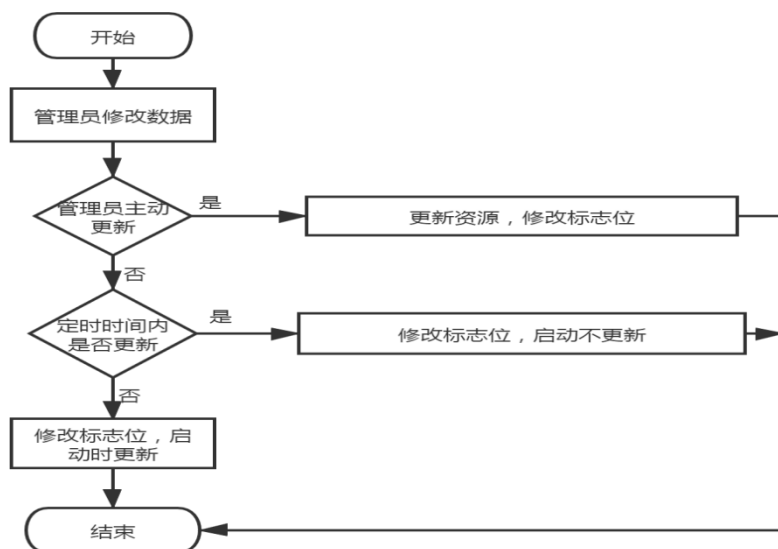


图 3-7 系统更新逻辑图,三层机制保障图片资源更新

### 3.3.6 数据库审计模块

在此系统中，数据的安全很重要，所以要记录下所有对数据操作，因此在此模块中会记录操作数据库的所有 SQL 语句，记录的信息包括 SQL 语句，操作者，操作时间，风险程度等。如果发生问题，在这里可以作为查询的依据。此模块的实现方式较为简单，主要是从数数据库中查询信息，经过 service 的数据处理，最后显示到前端界面上，这里用到了分页技术。

## 3.4 数据库设计

在此系统中需要用到较多的数据，此数据均是从数据库中查出的，数据的格式和形式比较简单，不会涉及到多表联合查询，此系统数据包括的表有如下：

Attack 表，此表记录了攻击类型的次数。

Attack\_degree，此表记录了攻击的严重程度

Attack\_method，此表记录了所有的攻击方式的次数

Attack\_type，此表记录了网络攻击类型的次数

Deal\_method，此表记录了应对攻击的处理方式

Illegal\_web，此表记录了所有的非法网站

Inner\_attack\_source，此表记录了国内的攻击源攻击次数

Outer\_attack\_source，此表记录了国外攻击源的攻击次数

Allsqls，此表记录了所有对此数据库操作的信息

User，用户信息表

Manager，管理员信息表

以上就是此系统用到的表，因为此数据库的目的主要是为统计图生成模块提供数据，所以表结构相对简单，基本都是某些信息的次数，在此数据库中，也不会设计到多表查询等操作。

## 第四章 系统的实现

### 4.1 系统功能模块设计

#### 4.1.1 系统界面的实现

在此系统中使用的图形界面库是 PyQt,PyQt 是一款优秀的图形界面库,在此库中使用了很多优秀的设计,比如信号和槽机制等。PyQt 在图形界面领域中使用的相当广泛。它有超过几百个类,将近几千个函数,可以满足我们开发的大部分要求,除此之外,他的另外一个优势是跨平台性,现在主流的操作系统都可以使用它,例如 windows,linux, Mac 等。

在此系统中,有众多的类,被分成了多个模块。图形组件和相关的类被保存到 QtGui 中,例如滚动条、位图、颜色、字体、按钮、窗体、状态栏、工具栏等。QtNetwork 主要负责网络编程,这些类可以编写基于 TCP 和 UD 的网络程序,他们使网络编程更简单,更方便。PyQtGUI 功能的核心库是 QtCore。该模块用于管理时间、线程或进程等。

Qt 有独特的信号和槽机制。所谓信号槽(slot),就是一个函数。当事件发生之后,例如,用户点击了按钮一下,它就会发出一个信号(signal)。此时这个信号需要绑定一个槽(slot),即响应函数,它就会使用连接函数即 connect。这个槽函数这个时候就会执行。

#### 4.1.2 系统所用数据的获取

本系统所用的数据均为企业的日志数据,所以首先要对数据进行筛选和处理。并将获取的数据储存到数据库中。在此筛选的称程序中主要用到了两个模块,一个是读取 Excel 的功能,pandas 模块提供了这个功能。还有就是统计功能,Collections 中的 Counter 提供了这个功能。经筛选后的数据在储存在数据库中。

### 4.2 面向对象类的实现

#### 1.持久层

##### (1) 获取数据类

这个获取数据类主要的功能为查询数据库,为服务层提供数据。在此类中连接数据库采用的技术为 pymysql 模,此模块可以很方便的进行数据库的操作,首先建立 mysql 数据库的连接。建立连接的代码为

```
self.__conn = pymysql.connect(host='localhost', user='root', passwd='123', db='dataset',
port=3306,charset='utf8')
```

```
self.__cur_key = self.__conn.cursor(cursor=pymysql.cursors.DictCursor)
```

代码块的意思是使用本地连接来连接书库,数据库的用户名为 root,密码为 123,在 3306 端口

连接 dataset 数据库。建立连接的对象设置成类的私有属性。

第二行代码的意思是通过连接获取数据库的游标。并且设置查出的结果以键值对的形式显示出来。这两行代码为此类的关键代码。然后其他的一些函数主要就是使用此游标，查询数据，返回数据。例如

```
def getFangPic5(self):
    self.__cur_key.execute("select * from outer_attack_source")
    data = self.__cur_key.fetchall()
    return data
```

此类中还完成了一些其他有关数据库的操作。总之此类主要负责和数据库的交互，和数据库的交流都要通过 Getdata 类来完成。

## （2）执行管理员 sql 语句类

这个类主要是完成管理员和数据库的操作，和上一个类的功能稍有不同，把这两个类分开，满足面向对象设计中的单一职责原则。这个类和上边一样需要获取连接，获取游标。在此类中主要包括两个主要的函数，一是 executeSql()函数，一个是 insertSqls()函数。第一个函数的主要功能是执行管理员的 Sql 语句。代码如下。

```
def executeSql(self):
    try:
        if "select" in self.__sql:#判断是不是查询的语句
            self.__cur_key.execute(self.__sql) #如果是，就返回查询的结果
            data = self.__cur_key.fetchall()
            return data
        else:
            self.__cur_key.execute(self.__sql) #如果不是，执行对数据库的其他操作，返回受影响的函数
            data = self.__cur_key.fetchall()
            self.__conn.commit()
            self.insertSqls()
            return data
    except(Exception):
        return None #如果出现异常，返回 None
```

insertSqls()函数主要将 sql 语句插入数据库中。

## 2. 服务层

在此层中获取持久层中的数据，生成统计图，为显示层提供服务。

### （1）主服务类

在主服务类中主要负责生成各种统计图。在此类中要获取持久层获取数据类的对象，通过对象获取相应的数据。在此类中最主要的救赎绘图的工具，使用了强大的 matplotlib 模块。可以

画出各种好看的统计图。

### (2) 登陆逻辑类

此类主要负责处理登陆的逻辑，其主要的流程如下，在前端的界面中输入用户名，密码等，点击登录，此时就会调用此类中的方法，此类中有持久层中获取数据类的对象。持久层中会查询是否存在此用户，如果存在则返回 true,否则返回 false。如果返回 true，则登录成功。否则提示用户名或者密码错误。

### (3) 显示信息类

此类主要负责前端信息显示的处理，其主要流程为调用持久层获取数据，在此层中对数据格式做一定的转换，并将信息返回。

### (4) 更新图片类

此类主要负责更新图片的逻辑处理，包括四个函数，这些函数包括写入日志，写入标志，更新策略。

## 3. 表示层

表示层主要是一些 UI 界面，主要包括

### (1) 登录界面

登录界面很简单，就是输入用户名和密码的框，以及还有个判断是否管理员的复选框。登录界面的逻辑代码如下

```
def loginOthers(self):
    import mainWindow.mainWidget as mainWidget #倒入包
    import mainWindow.ManagerDialog as manager
    serviceobj=service.service() #获取处理登陆的对象
    isManager=self.radioButton.isChecked() #获取是否管理员
    username=self.edit_username.text() #获取用户名
    password=self.edit_password.text() #获取密码
    if isManager==False: #如果不是管理员
        if serviceobj.LoginUser(username,password): #如果密码正确
            self.hide()
            self.s = mainWidget.firstWidget() #弹出主界面
            self.s.show()
        else: #否则，弹出信息
            reply = QMessageBox.information(self, "提示", "用户名或密码错误",
            QMessageBox.Yes|QMessageBox.No, QMessageBox.Yes)
    else:
        if serviceobj.LoginManager(username,password): #如果管理员密码正确
            self.hide()
            self.s = manager.Manager()
```

```

self.s.show()

else:
    #不正确弹出提示信息
    reply = QMessageBox.information(self, "提示", "用户名或密码错误",
    QMessageBox.Yes | QMessageBox.No,
    QMessageBox.Yes)
    
```

登陆界面见图 4-1



图 4.1 登陆界面

## (2) 主界面

在此系统中，主界面是用户登录成功后显示的界面。主界面主要包显示统计图信息的空间，跳转按钮，查询搜索框等。器界面如图 4.2 所示。



图 4.2 主界面

实现此页面要继承 QWidget 类，此类可以实现界面。此界面使用的是绝对布局方式。首先在界



面上画出布局。使用 `def paintEvent(self, QPaintEvent):` 函数可以在界面上画出各种方框。然后使用 `def pasteLabel(self):` 在相应位置粘上显示图片或者信息的控件，例如

```
self.path = "../resources/mainPic/pic3/pic1.png" #图片的路径
self.label_P_LouDong = mySignal.myLabel(self) #创建经过自己定制的 label 对象
self.label_P_LouDong.resize(280, 190) #设置 label 的大小
self.label_P_LouDong.move(715, 130) #移动 label 的位置
self.label_P_LouDong.setPixmap(QPixmap(self.path)) #设置图片
self.label_P_LouDong.setScaledContents(True) # 使图片自适应标签大小
self.label_P_LouDong.setAutoFillBackground(True) #设置图片为背景
self.label_P_LouDong.clicked.connect(lambda :self.changPic3(int(filter(str.isdigit,
self.path.split("/").pop().split(".")[0]).__next__()-1)) #绑定图片所定影的槽函数
此模块还有一些其他界面，实现原理与此相似
```

### (3) 自定义 Label 控件

因为原生的 `label` 的空间不能使用信号和槽机制，所以需要重新自定义 `label` 控件。

```
class myLabel(QLabel): #自己定义的类，继承原生的 QLabel 控件
    def __init__(self, o):
        super(myLabel, self).__init__(o)
    clicked = pyqtSignal() #添加点击信号
    def mousePressEvent(self, QMouseEvent): #添加出发信号的事件
        if QMouseEvent.button() == Qt.LeftButton:
            self.clicked.emit()
```

在需要为 `label` 控件添加信号的地方，使用自己定义的 `myLabel` 控件就可以实现。

## 4.3 核心功能的实现

### 4.3.1 统计图生成模块

此模块的主要功能是生成统计图，是整个系统的核心。此模块需要底层查询数据库提供数据服务，生成统计图后，保存到文件中，为前端的显示做准备。关键代码如下

`self.__myData=getMyData.getMyData()` 这行代码是获取某个类的对象，此对象就是获取生成图片所要用的数据。生成图片的代码为：

```
def getMainPic4(self):
    plt.figure() #初始化画板
    data=self.__myData.getMainPic4() #获取数据,data 是 list,list 中的元素是 map
    for index,i in enumerate(data):
        labels = list(i.keys()) #获取每个 map 的 keys
        datause = list(i.values()) #获取每个 map 的 values
```

```

plt.pie(datause, startangle=90, shadow=True)    #画饼状图
plt.axis('equal')
plt.legend(labels, shadow=True, bbox_to_anchor=(0.85, 0.6)) #设置图例
plt.title("攻击次数") #设置图的标题
plt.savefig("../resources/mainPic/pic4/pic%s.png" % (index+1)) #保存图篇到某个位置

```

```

self.colors=["red","green","yellow","lightskyblue","blue","orange","cyan","purple","brown"]

```

```

.....

```

```

num = len(list(data[0].keys()))

```

```

myColor = random.sample(self.colors, num)

```

这是此类中对颜色的列举，在生成统计图时，会根据数据的种类随机使用颜色，更容易分辨。

将生成的图片保存到本工程资源目录下，当前端引用时，直接在本地查找就好。先在本地缓存再引用，而不是直接在使用时生成是基于以考虑。一是提高系统的启动速度，如果每次都是在启动时生成，会花费大量的时间，用户体验极差，使用这种机制，就是基于计算机中有名的空间换时间的思想。二是程序耦合度的角度，如果前端显示界面直接依赖统计图生成模块，那么程序的耦合度将大大增加，不利于程序的扩展和维护。但是使用此方法会带来另一个问题，就是图片的时效性问题。为解决此问题，在此系统中引入了资源更新模块。

#### 4.3.2 统计图显示模块实现

此模块主要是将统计图生成的图片显示出来。同一个主题可能存在好几张统计图，所以要先对资源中的统计图的路径信息做包装，此时在 service 层中有一个类，此类就是负责生成相应主题下所有图片的路径信息列表。在前端获取此列表，建立显示控件的信号和槽机制，每当点击此控件，就会轮流显示此列表中的图片。

```

def getPic1(self):

```

```

    pathMain=self.pathMain+"pic1" #pathMain 是此图片文件夹的位置

```

```

    path_list = os.listdir(pathMain) #获取此文件下的所有图片

```

```

    path_list_use = ["../resources/mainPic/pic1/" + i for i in path_list] #拼接所有图片的路径，

```

并且放在列表中

```

    return path_list_use #返回列表

```

这是封装图片路径信息类中的一个函数，其他函数类似。

```

self.path="../resources/secondPic/pic6/pic1.png" #初始图片的位置

```

```

self.label_P_LeiXing = mySignal.myLabel(self) #建立 label 的对象

```

```

self.label_P_LeiXing.resize(280, 190) #调整 label 的大小

```

```

self.label_P_LeiXing.move(715, 410) #移动 label

```

```

self.label_P_LeiXing.setPixmap(QPixmap(self.path)) # 显示图片

```

```

self.label_P_LeiXing.setScaledContents(True) # 使图片自适应标签大小

```

```

self.label_P_LeiXing.setAutoFillBackground(True)
self.label_P_LeiXing.clicked.connect(lambda: self.changPic6(int(filter(str.isdigit,
self.path.split("/").pop().split(".")[0]).__next__() - 1)) #连接信号

```

上面的代码就是显示图片的逻辑，在连接信号的代码中，`int(filter(str.isdigit, self.path.split("/").pop().split(".")[0]).__next__() - 1)`是对图片路径做解析，其返回值为图片后边的标号。并且将此数字以参数的形式传到槽函数中。

```

def changPic6(self, n):
    paths = self.picData.getFangPic6() #获取对应的主题的图片路径列表
    index = (n + 1) % paths.__len__() #以取余的方式遍历此列表，这样就会循环
    self.path = paths[index] #取对应的路径
    self.label_P_LeiXing.setPixmap(QPixmap(self.path)) #设置图片
    self.label_P_LeiXing.setScaledContents(True) #使图片自适应标签大小

```

上面的代码，就是标签控件的槽函数，此时点击此标签就会切换图片。

### 4.3.3 搜索查询模块的实现

此模块是搜索查询模块，用户可以在搜索框中输入自己想查询的信息，点击搜索查询，相关信息就会显示到结果显示模块。如图 4-3



图 4-3

在输入查询信息后，点击搜索，就会调用 `service` 中的处理查询类。在此类中，首先会对此自然语言进行分词，所使用的算法前边已经有介绍，实现此算法的在 `JieBa` 模块中实现了。

此算法对此语言的分词结果为，见图 4-4

```
{'我', '源', '攻击', '查询', '要'}
```

图 4-4 分词结果

代码如下

```

def getWords(self):
    result=set(jieba.lcut(self.dealStr,cut_all=True)) #分词
    sqls_list=list({"程度", "方法", "源", "国外", "国内"}&result) #求关键字的集合和分词结果
    的交集
    if sqls_list.__len__(>0): #剔除多余词
        if result.__contains__("攻击"):
            result.remove("攻击")
    return result&self.key_words_set #返回结果和系统关键字的交集

```

获取关键字后，映射成 `sql` 语句，并执行 `sql` 语句，将结果返回。

#获取要使用的 sql 语句

```
def get_use_sqls(self):
    use_sets=self.getWords()
    if use_sets is None:
        return None
    sql_list=[]
    for i in use_sets:
        sql_list.append(self.all_sqls[i])
    if use_sets.__contains__("源"):
        sql_list.append(self.all_sqls["国外"])
    Dao_level=myData.getData()
    result=Dao_level.get_Search_Result(sql_list)
    return result
```

返回结果后，将结果进行处理，返回给前端界面

```
def changge_format(self):
    re=[]
    data=self.get_use_sqls()
    for i in data:
        if i is not None:
            for j in i:
                for k, value in j.items():
                    re.append(k + "-----" + str(value))

    return re
```

查询结果的显示界面见图 4-5

| 查询结果 |             |
|------|-------------|
| 1    | 河北-----1181 |
| 2    | 江苏-----1066 |
| 3    | 河南-----1017 |
| 4    | 山东-----684  |
| 5    | 广东-----608  |
| 6    | 北京-----358  |
| 7    | 浙江-----357  |
| 8    | 香港-----271  |
| 9    | 陕西-----245  |
| 10   | 四川-----195  |

上一页      下一页

图 4-5 查询结果显示

### 4.3.4 管理员模块的实现

管理员模块主要负责对本系统数据的管理,在此模块中,管理员可以看到整个数据库的结构,管理员可以根据数据的结构通过 sql 语句去修改数据,查询数据等。操作结果会显示到本界面中。界面如图 4-6 所示



图 4.6 管理员界面

在此界面中,难点之一是数据库结构的显示,器代码如下

```
#10,70,200,620
```

```
self.tree_model=QTreeWidget(self) #定义树形控件的对象
self.tree_model.resize(249,589)    #调整大小
self.tree_model.move(11,101)    #移动位置
self.tree_model.setColumnCount(2) #设置行数
self.tree_model.setHeaderLabels(["表","类型"]) #设置标题
self.tree_model.setToolTip("hello")
root = QTreeWidgetItem(self.tree_model)
root.setText(0, 'dataset')    #设置数据库名称
root.setIcon(0, QIcon('../resources/icons/db.png')) #设置图片
#设置宽度
self.tree_model.setColumnWidth(0, 140)
self.tree_model.clicked.connect(self.onClicked) #给控件添加槽函数, 点击此函数会显示此控件设置的名字
QToolTip.setFont(QFont('SansSerif', 12))
for key,value in self.__msg.items(): #遍历获取的数据
```

```

child1 = QTreeWidgetItem()    #创建节点
child1.setText(0, key)         #给节点添加数据
child1.setIcon(0, QIcon('../resources/icons/table.png')) #给节点添加图片
root.addChild(child1)          #将节点添加到树形控件中
for i in value:
    child3 = QTreeWidgetItem(child1)    #给每个表名添加字段属性
    child3.setText(0, i)
    child3.setText(1, "int")

```

此槽函数的代码如下。

#点击左边控件的响应函数

```

def onCliked(self):
    item = self.tree_model.currentItem() #获取当前的空间
    key=item.text(0)    #获取此位置的信息
    value=item.text(1)
    if key == "dataset":    #如果点击根目录，就显示空
        showValue=""
    else:
        if value.__len__()==0:
            showValue="表名是:"+key    #否则就显示当前的表名
        else:
            showValue="字段名是:"+key+" "+"字段类型是:"+value
    self.showMsgEdit.setText(showValue)

```

当管理员写上 sql 语句时，右键鼠标，点击执行。即可执行最后一条 sql 语句。后态处理此 sql 语句的代码为

```

def executeSql(self):
    try:
        if "select" in self.__sql:    #判断 sql 语句是不是查询语句
            self.__cur_key.execute(self.__sql) #如果是执行查询
            data = self.__cur_key.fetchall() #获取结果
            return data    #返回数据
        else:    #不是查询语句
            self.__cur_key.execute(self.__sql)
            data = self.__cur_key.fetchall()
            self.__conn.commit()
            self.insertSqls()
            return data    #返回空元组，代码执行成功
    
```

```
except(Exception): #代码出错, 返回 None
```

```
return None
```

底层将数据返回后, 经过服务成数据格式的处理, 显示到前端界面中, 例如, 管理员输入“select \* from user”, 则显示的结果为图 4-7

```
id-----1
userName-----薛硕
password-----123
id-----2
userName-----li
password-----446
id-----19
userName-----ww
password-----12345
```

图 4-7 sql 语句查询结果

#### 4.3.4 系统资源更新模块的实现

在此系统中为保证程序的启动速度, 加载图片资源时是使用的已经生成好的, 这种方式虽然加快了程序的启动速度, 减少了程序的耦合, 但是也带来了图片资源的时效性无法保证的问题, 因此在管理员修改数据后, 图片资源也要相应的更新。在此系统中, 为保证图片资源及时更新, 采用了三种保障机制。

在介绍保障机制前, 先介绍图片更新的原理, 图片更新的原理是在资源图片生成类中添加了一个函数, 此函数调用了所有生成图片资源的函数。在更新时调用此函数就可以完成更新。

更新机制一, 管理员主动更新, 在管理员界面, 有一个更新按钮, 点击此按钮就可以完成更新。

更新机制二, 系统定时更新, 系统会检测管理员的 sql 语句, 如果发现修改数据库中数据的 sql 语句, 就会在文件中写入一个标志, 表示需要更新, 并且启另外一个线程, 默认 30 分钟后更新。需要注意的是, 在文件中写入标志, 只允许一个线程读写, 因此需要有同步机制。

```
def writeFlag(self):
```

```
    self.semaphore.acquire()    # p 操作
```

```
    with open("../resources/isUPdateFlag/flag.txt", "w") as f:
```

```
        f.write("1")    #将标志写入文件
```

```
    self.semaphore.release()    # v 操作
```

以上代码可以实现, 在文件中写入标志位。

```
def updateStrategy(self,sql):
```

```
    a=sql.split(" ")[0]    #获取 sql 语句中的第一个词
```

```
    isExist=a in self.key_words #判断是不是要更新
```

```
    if self.flag and isExist:    #更新条件
```

```
        self.flag = False
```

```
        t = threading.Timer(1800, self.updatePics)#开启线程更新, 30 分钟 后更新
```

```
t.start()
```

在系统自动更新机制中，默认 30 分钟后更新的目的是防止系统频繁的更新，而影响系统的性能，但是这种方法又引来了另外一个问题，就是系统还没有更新，就关闭了系统，这就用到了机制三。

更新机制三，此更新机制就是在系统启动前检查标志位是否需要跟更新，在用户输入用户名和密码正确准备登陆时，系统会读取文件中的标志位，读取操作也要进行同步，防止在读取的同时有线程写入。如果标志位显示需要更新，那么就立刻更新。

```
def aboutUpadePics(self):
    import Service.MainService as MainPic
    self.semaphore.acquire() #p 操作
    with open("../resources/isUPdateFlag/flag.txt", "r") as f: #读数据
        data = int(f.read())
    self.semaphore.release()
    if data == 0: #判断是否需要更新
        Pass #不更新
    else:
        MainPic.generatePic().updatePics() #更新图片
```

有上述三层更新机制的保障，基本可以保障图片资源的时效性。

#### 4.3.6 数据库审计的实现

此模块的主要功能是管理所有对数据的操作，显示的信息包括执行的 sql 语句，操作者，操作的时间，以及此操作的风险程度。此模块可以保证所有对数据库的操作都有张可循，尽可能保证此系统的安全，此模块的实现原理较为简单，在整个系统中所有对数据库的操作都会记录下来保存到数据库中，然后查询某个表中的信息，经过一些格式的转换显示到前端即可。其界面效果如图 4.8 所示。

|    | sql语句                             | 操作者    | 时间                  | 风险程度 |
|----|-----------------------------------|--------|---------------------|------|
| 1  | select * from attack_degree       | system | 2020-05-14 10:50... | safe |
| 2  | select * from outer_attack_sourc  | system | 2020-05-14 10:50... | safe |
| 3  | select * from deal_method         | system | 2020-05-14 10:50... | safe |
| 4  | select * from attack              | system | 2020-05-14 10:50... | safe |
| 5  | select * from attack_method       | system | 2020-05-14 10:50... | safe |
| 6  | select * from inner_atak_sourceb  | system | 2020-05-14 10:50... | safe |
| 7  | select * from attack_degree       | system | 2020-05-14 10:50... | safe |
| 8  | select * from attack              | system | 2020-05-14 10:50... | safe |
| 9  | select * from outer_attack_source | system | 2020-05-14 10:50... | safe |
| 10 | select * from user                | 薛硕     | 2020-04-14 09:26... | safe |

图 4.8 数据库审计模块，显示所有此系统对数据库的操作



## 第五章 系统测试

### 5.1 系统测试方法

本次系统测试采用面向对象软件的测试方法。由于面向对象软件开发引入了包括类与对象、继承、消息传递和动态链接等许多不同的特性。随着面向对象软件的开发方法的广泛使用。尽管例如白盒测试、黑盒测试、各种测试活动等软件测试技术对于面向对象软件同样适用的<sup>[15]</sup>，但针对面向对象软件的特征研究相应的测试方法，能更好的保证面向对象软件的质量。

本次面向对象软件的测试分别采用类的测试和交互测试两种方法<sup>[16]</sup>。

类是面向对象的基本类型，对类实例化得到的对象才能运行。因此，对类进行测试首先需要开发驱动程序，负责创建类的实例，然后测试驱动程序根据测试用例的定义，向类的实例（对象）发送一个或多个消息（调用类的方法），根据响应值、实例对象的属性状态的变化判断测试用例的执行结果是否正确。测试驱动程序的主要功能时运行可执行的测试用例并记录运行的结果。对于面向对象软件驱动程序可用下面的方式编写。实现一个 `main()` 函数作为测试驱动程序，使得测试程序可以编译、运行。在 `main()` 函数中执行测试用例，独一类进行实例化，给类发送响应的测试消息，并输出测试执行的结果。

交互测试，面向对象软件的运行主要是通过各种对象之间的消息传递来完成。因此，即使单个类中的每个方法在独立测试中时是正确的，多个类的实例通过消息传递交互的完成某个任务时，如果交互过程不正确，软件仍然可能是错误的。在交互测试中，测试驱动程序负责对参与交互的多个类进行实例化，并向相应的对象发送启动交互的消息或交互所需的信息序列。在判断交互测试的结果是，一方面要看交互完成后的输出，另一方面要看各个对象在参与交互之后所处的状态是否正确<sup>[17]</sup>。

交互测试中不但要考虑信息的顺序，设计测试用例时，不仅要检查是否可以争取接受正常的消息序列，还要检查是否能够恰当处理不正确的消息序列<sup>[18]</sup>。

### 5.2 类的测试

在此系统中，主要包括的类有持久层的数据获取类，执行管理员 `sql` 语句类。在服务层主要有生成图片类，处理查询语句类，在显示层主要是前端 `GUI` 界面。下面就主要测试这几个类的功能是否完成。

1. 获取数据类，此类的主要作用是查询数据库，将数据返回到服务层，因此测试此类，只需要一次调用提供数据的函数，看他们是否发生异常以及输出的数据是否正确。测试此类就是生成此类的对象，然后依次调用每个方法，查看数据是否正确。代码如下，

```
if __name__ == '__main__': #main 函数
```

```
myData = getData()    #创建此类的对象
data=myData.getMainPic1()  #调用函数
print(data)           #打印数据
```

显示的结果为[[{'id': 1, '河北': 1181, '江苏': 1066, '河南': 1017, '山东': 684, '广东': 608, '北京': 358, '浙江': 357, '香港': 271, '陕西': 245, '四川': 195, '湖北': 168, '上海': 121, '云南': 40, '安徽': 36, '辽宁': 36, '吉林': 30, '黑龙江': 29, '台湾': 29, '山西': 27, '重庆': 20, '江西': 9, '甘肃': 9, '福建': 9, '天津': 9, '新疆': 7, '内蒙古': 7, '湖南': 7, '贵州': 4, '宁夏': 3, '广西': 2, '西藏': 2, '海南': 2, '青海': 1}, {'id': 2, '河北': 1761, '江苏': 56, '河南': 499, '山东': 27, '广东': 69, '北京': 279, '浙江': 75, '香港': 260, '陕西': 99, '四川': 48, '湖北': 34, '上海': 22, '云南': 31, '安徽': 45, '辽宁': 2, '吉林': 0, '黑龙江': 4, '台湾': 9, '山西': 1, '重庆': 3, '江西': 4, '甘肃': 5, '福建': 1, '天津': 25, '新疆': 1, '内蒙古': 112, '湖南': 7, '贵州': 2, '宁夏': 0, '广西': 0, '西藏': 0, '海南': 2, '青海': 4}]]

此函数是查询共计源攻击次数的排名，输出结果正确无异常。可用此方法测试所有函数

2. 执行管理员 sql 语句的类，在此系统中，管理员模块是个很重要的模块，在此模块中，管理员通过 sql 语句来管理数据库。因此要测试此模块是否能处理各种 sql 语句以及能否处理各种异常问题。

测试代码如下

```
if __name__ == '__main__': #main 函数
    execueManagersql = execueManagersql("aa") #生成对象
    sql="*****" #sql 语句
    execueManagersql.setSql(sql)
    data=execueManagersql.executeSql() #执行语句，返回结果
    print(data)
```

(1) 测试查询语句 sql="select \* from user"

返回的结果为[{'id': 1, 'userName': '薛硕', 'password': '123'}, {'id': 2, 'userName': 'li', 'password': '446'}, {'id': 19, 'userName': 'ww', 'password': '12345'}, {'id': 20, 'userName': 'z3', 'password': '123'}]

结果正确

(2) 测试插入语句 sql="insert into user(userName,password) values('z3','123')"

返回的结果为 "()" 空元组。代表插入成功。插入结果见图 5-1

|    |    |       |
|----|----|-------|
| 19 | ww | 12345 |
| 21 | z3 | 123   |

图 5-1

(3) 测试删除语句 sql="delete from user where id=19"

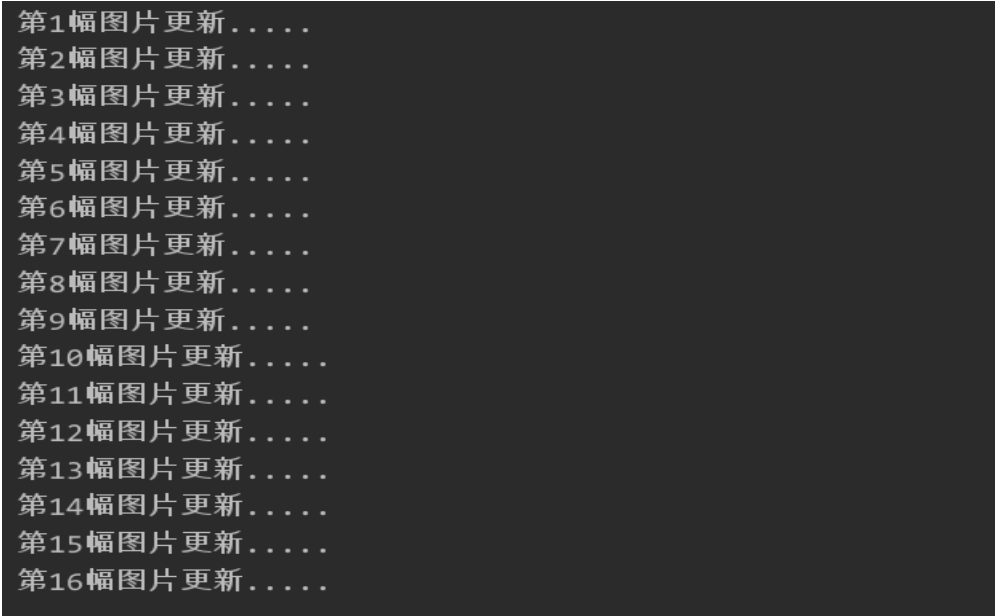
返回结果为 "()" 空元组，代表插入成功。

(4) 测试出现异常 sql="select \* from user2" user2 不存在

此时出现异常，在这里用了捕获异常机制，遇到异常后返回 None。  
测试的几种情况，均返回了正确的结果。

### 3. 图片生成类的测试。

在此类中，有一个函数，此函数会调用生成图片的所有函数，所以只要此函数运行成功，就可以证明此类没有问题。运行结果见图 5-2，可见程序正常退出，测试通过。

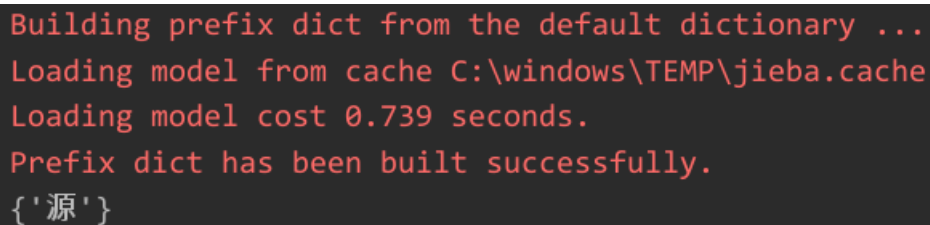


```
第1幅图片更新.....  
第2幅图片更新.....  
第3幅图片更新.....  
第4幅图片更新.....  
第5幅图片更新.....  
第6幅图片更新.....  
第7幅图片更新.....  
第8幅图片更新.....  
第9幅图片更新.....  
第10幅图片更新.....  
第11幅图片更新.....  
第12幅图片更新.....  
第13幅图片更新.....  
第14幅图片更新.....  
第15幅图片更新.....  
第16幅图片更新.....
```

图 5-2 测试图片生成类

### 4. 字符串处理类

此类的主要作用是对输入的自然语言进行分词，然后与系统给定的关键字进行求交集，再映射成 sql 语句。在此测试中，只要测试分词的结果和系统关键字的交集是否符合我们期望即可。例如输入“我想查询攻击源”，对于此系统来说，“源”就是关键字，所以只要能从此字符串中找出“源”即可。运行程序，结果见图 5-3 所示



```
Building prefix dict from the default dictionary ...  
Loading model from cache C:\windows\TEMP\jieba.cache  
Loading model cost 0.739 seconds.  
Prefix dict has been built successfully.  
{'源'}
```

图 5-3 字符串分析结果

## 5.3 交互测试

1. 用户登录模块，如果登陆成功，直接跳到主界面，否则提示用户名或者密码错误。。测试步骤见表 5-1

表 5-1 登陆模块测试

| 输入用例             | 预期结果     | 实际结果     | 原因分析          |
|------------------|----------|----------|---------------|
| 用户名: 薛硕, 密码: 123 | 登录成功     | 登录成功     | 数据库中的用户表存在此信息 |
| 用户名: 薛硕, 密码: 12  | 弹出登录失败的框 | 弹出登录失败的框 | 密码输入错误        |
| 用户名: 张三, 密码: 12  | 弹出登录失败的框 | 弹出登录失败的框 | 用户名输入错误       |

登陆失败的界面如图 5-4 所示。

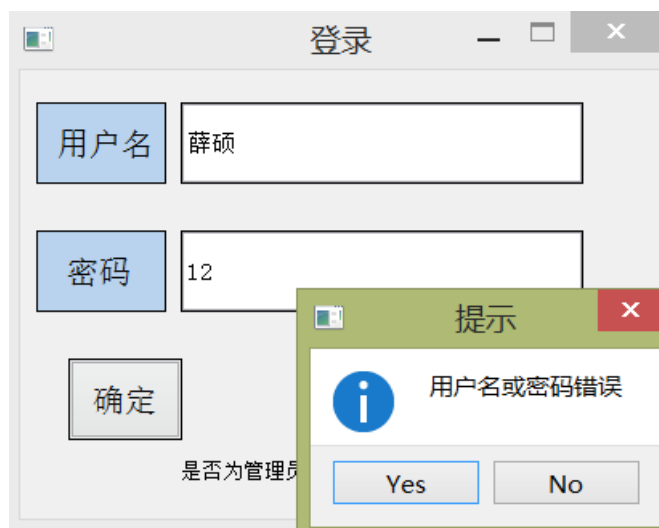


图 5-4 登陆失败的界面

登陆成功的界面会直接跳转到主界面，如图 5-5 所示



图 5-5，主界面

## 2. 界面跳转测试

经过测试，此项功能没有问题，界面间可以互相跳转

3. 搜索功能测试，主要测试在输入要查询的信息后，查询出的信息是否正确，显示是否正确，此模块的测试见表 5-2

表 5-2 查询模块测试

| 输入用例    | 预期结果     | 实际结果     | 原因分析   |
|---------|----------|----------|--------|
| 我要查询攻击源 | 显示出所有攻击源 | 显示出所有攻击源 | 结果正确   |
| 非法网站    | 显示所有非法网站 | 显示所有非法网站 | 结果正确   |
| Hello   | 显示空      | 显示空      | 没有相关信息 |

例如要输入“我要查询国内攻击源”，图见 5-6

|           |    |
|-----------|----|
| 我要查询国内攻击源 | 搜索 |
|-----------|----|

图 5-6 输入的查询信息

查询结果见图 5.7

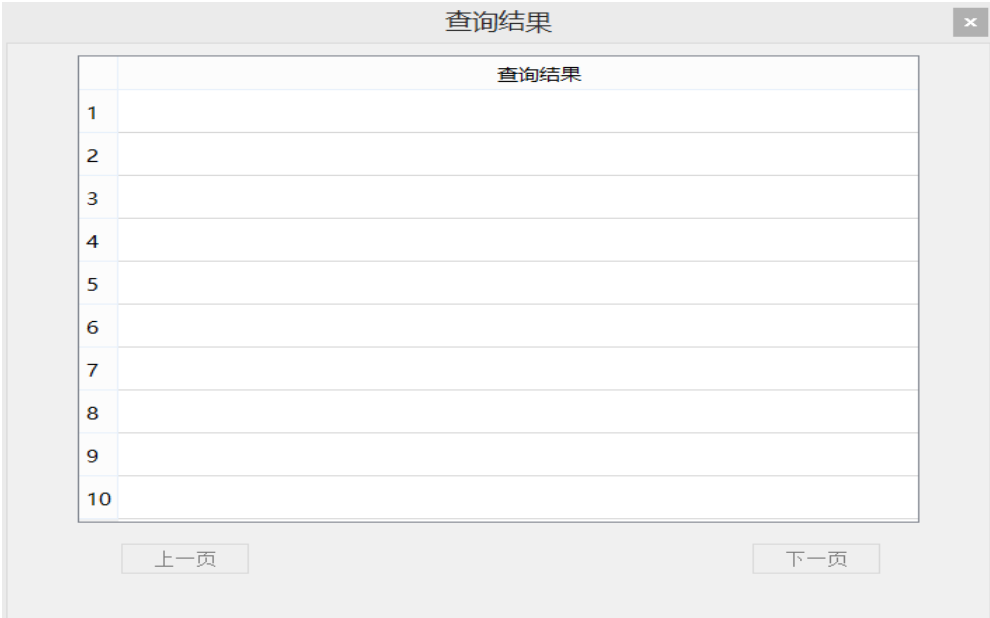
| 查询结果 |             |
|------|-------------|
| 查询结果 |             |
| 1    | 河北-----1181 |
| 2    | 江苏-----1066 |
| 3    | 河南-----1017 |
| 4    | 山东-----684  |
| 5    | 广东-----608  |
| 6    | 北京-----358  |
| 7    | 浙江-----357  |
| 8    | 香港-----271  |
| 9    | 陕西-----245  |
| 10   | 四川-----195  |

上一页 下一页

图 5.7 国内攻击源

查询中，所有的数据均来源于数据库中，根据用于所输入的信息，在数据库中配置相关的信息。

例如我要查询“hello”，结果如图 5-8 所示。



如图 5-8 ， 如果查询 hello,则结果为空

4. 管理员模块的测试

在管理员模块中，输入 sql 语句，右键查询，显示结果。以对 user 表的操作为例。测试步骤见表 5-3

表 5-3， 管理员的测试步骤

| 输入用例                         | 预期结果            | 实际结果            | 原因分析 |
|------------------------------|-----------------|-----------------|------|
| Select * from user           | 显示出 user 表的信息   | 显示出 user 表的信息   | 结果正确 |
| Delete from user where id =2 | 数据库删除，控制台显示操作成功 | 数据库删除，控制台显示操作成功 | 结果正确 |
| Select * from user2          | 显示结果出错          | 显示结果出错          | 没有此表 |

例，在控制台输入 “Select \* from user” ， 结果见图 5-9



图 5.9 查询成功

在控制台输入“delete from user where id=21”，见图 5-10



图 5-10，删除后显示的结果

在控制台输入“select \* from user2”，结果应该是出错，因为没有此表，结果见图 5-11



图 5-11，出错后显示的结果

以上就是我们经常遇到的情况，执行都没有问题。





# 结论与展望

## 1 结论

随着当代网络技术的高速发展，人类已经进入网络化社会。网络已经渗透到社会的方方面面，在教育，交通，医疗，文化交流等方面发挥着越来越重要的作用。中国已经是互联网大国，互联网在中国社会中的作用更是举足轻重，因此网络安全就显得极为重要。而对网络安全设备的数据可视化就是研究网络安全的中还要手段之一。

通过这段时间的设计与开发，总体上完成了以下几项工作：

1. 研究网络安全数据可视化，参考了一些相关的文献资料，并使用了一些相关系统，掌握了有关网络安全，数据可视化方面的知识。
2. 研究了 Python 技术对于实现此可视化系统的可行性，温习可能会在编程中用到的 Python 的函数接口和 Python 线程, 图界面等相关的知识，为系统原型的实现提供解决方案
3. 在对技术有了一定的掌握后，对系统的总体实现有一定的初步大纲，并在大纲的基础上尽可能考虑到各种现实情况和解决方案，完善设计说明。
4. 在研究设计的基础之上对系统进行实现，该系统实现了总体设计中的统计图像的生成，展示，图片资源的及时更新等。
5. 对系统进行测试，系统可以准确无误的运行，对系统每个功能模块也进行了测试，掌握了测试的方法。增长了自己软件开发的技能，提高了自己解决问题的能力，提高了自己的专业素养。

## 2 展望

网络安全数据可视化系统的设计与实现涉及了诸多方面的理论，方法和技术。此系统为应对愈来愈严峻的网络安全安全形势提供了一种新的可以使用的方法<sup>[19]</sup>。将来的社会必定是一个万物互联互通的时代，互联网将更加渗透到社会的方方面面，在教育，科技，文化，金融等领域发挥愈来愈重要的作用。网络安全也必定是人们关注的重点，而医用真实的企业网络日志去做可视化，分析此企业的网络安全状况，将成为有发展前景的技术<sup>[20]</sup>。

此系统使用的数据规模较小，实现的功能也较为简单，在实际的应用上还存在较大距离，

但是在互联网愈来愈普及的时代，网络安全感愈来愈重要的时代，企业的网络安全需要更多技术的保障，其中的可视化技术会扮演重要的角色。更复杂，更先进，更有使用价值的的可视化产品一定会越来越多<sup>[21]</sup>，此项技术也会发展的愈来愈好。



## 致 谢

在论文即将完成之际，非常感谢我的指导老师王卫亚老师对我工作的指导，在毕设的选题，开题，代码的编写，论文的完成过程中均给了我很大的帮助。王卫亚老师治学严谨，知识渊博，充满工作热情，这些优点值得我终身学习。在系统的开发和论文的撰写过程中，王卫亚老师不辞辛劳，不厌其烦的一遍又一遍的帮我修改并提出意见。对于我提出的问题，王卫亚老师总会非常及时的给予我解答，并提出修改意见。大学 4 年，王卫亚老师作为我的导师，不仅在学习上对我提供了许多帮助，这对于我的生活也给予我了一些关键性提议，再次对老师表示由衷的感谢，老师对于我的论文和毕设提出了许多有效的建议，让我的论文更加的严谨。王老师严肃的科学态度，一丝不苟的学术精神，求同存异的工作作风不断激励着我。同时感谢学长和其他同学在我系统开发中给我提出的建议，感谢他们的帮助。

感谢各位舍友与同学在毕业设计中对我的帮助与鼓舞。有了他们的陪伴，我的大学生活才多了一份充实、一份快乐、一份成长。



## 参考文献

- [1] 黄楚新, 王丹. “互联网+”意味着什么——对“互联网+”的深层认识[J]. 新闻与写作, 2015(5): 5-9
- [2] 陶广. 网络安全数据可视化研究综述[J]. 信息与电脑:理论版, 2015(8):75-76. 蔡跃洲. “互联网+”行动的创新创业机遇与挑战[J]. 求是学刊, 2016(3): 43—52.
- [3] 吴铭. 中国网络安全面临威胁[J]. 党政论坛, 2014(1).
- [4] 惠志斌. 中国国家网络空间安全战略的理论构建与实现路径[J]. 中国软科学, 2012(5).
- [5] 李城均, 郭家铭, 黄栋. 网络安全数据可视化分析[J]. 南方农机, 2017(6). 陈建军, 余志强. 数据可视化技术及其应用[J]. 红外与激光工程, 2011(30):239-243.
- [6] 钟明. 网络安全数据可视化探究[J]. 网络安全技术与应用, 2015(1):118.
- [7] Halstead M H. Elements of software science【M】. New York:Elsevier, 1977.
- [8] 王新. 基于 MVC 模式的通用 Web 软件系统开发框架设计与实现[D]. 成都: 电子科技大学, 2007..
- [9] 蔡希尧, 陈平, 面向对象技术. 西安: 西安电子科技大学出版社, 1993
- [10] 郝红岩. 基于 MVC 模式的 Web 框架的应用研究[D]. 武汉: 武汉理工大学, 2013
- [11] 耿永利. 网络安全数据可视化[J]. 信息与电脑:理论版, 2016, 000(009):P.203-204.
- [12] 易俗. 基于 Web 的4G 业务管理系统设计与实现[J]. 电脑知识与技术, 2016, 012(002):72-73,76. 朱文强. Trie 树和单字倒排相结合的汉英词典查找机制[J]. 哈尔滨工业大学学报: 自然科学版, 2008 (2): 182-185.
- [13] 陈志群. 基于 J2EE 架构的企业人力资源管理系统设计与实现[D]. 2008..
- [14] 陆荣幸, 郁洲, 阮永良, 等. J2EE 平台上 MVC 设计模式的研究与实现[J]. 计算机应用研究, 2003, 020(003):144-146.
- [15] 游学军. 基于 S2SH 的科研积分管理系统的设计与实现[D]. 南京邮电大学.
- [16] 路晓丽, 葛玮, 龚晓庆, 等. 软件测试技术[M]. 北京: 机械工业出版社, 2007.
- [17] 赵瑞莲. 软件测试方法研究[D]. 中国科学院计算技术研究所, 2001.
- [18] Jinfu Chen, Lili Zhu, Tsong Yueh Chen, Dave Towey, Fei-Ching Kuo, Rubing Huang, Yuchi Guo. Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering[J]. The Journal of Systems & Software, 2018, 135
- [19] 刘勘, 周洞汝. 大型数据库中的数据可视化技术 [A]. 第十八届全国数据库学术会议论文集 (技术报告篇) [C], 2011.
- [20] 任炜. 网络安全数据可视化综述 [J]. 信息通信, 2014(12):142-143.
- [21] 赵颖, 樊晓平, 周芳芳, 等. 网络安全数据可视化综述 [J]. 计算机辅助设计与图形学学报, 2014(5):687-697.
- [22] 王超. 基于社交关系的职位推荐系统的架构与实现[J]. 数字技术与应用, 2013, 000(011):123-125, 127.