Python A 语言程序设计

课程设计报告

题目：基于 Python 的知网爬虫系统

专　　业　信息管理与信息系统

班　　级　信管 201

学　　生　范一晨

学　　号　3200561015

指导教师　谢天保

西安理工大学

2022 年 春季 学期

目 录

1 项目背景

人类的知识浩如烟海，每年、每月乃至每天都有大量的知识信息出现。从图书馆中的藏书，到知识文献网站的数据库，知识的数量无穷无尽，而管理它们、利用它们就成了当前的一个重要任务。

文献的数据检索和下载作为数据整理的必经步骤，自然是该被严肃对待的重要一步。然而，想要进行文献期刊等知识资源的检索和获取却比较复杂：需要首先打开浏览器并进入期刊网站，键入所需要的查询关键字后进行搜索，对于得到的结果也很难进行整理、计数和统计，需要将所有搜索结果逐条输入之后再进行文献信息的整理和划分。

此外，传统的文献数据收集方法在搜集时常受到文章所属网站因素的影响, 如网站仅收录某一期刊或来源于某一机构的文章, 这样就难以保证文献数据的代表性, 并且如果要使用文献类型作统计分析，受限于数据获取的手段，能用来做分析处理的样本容量非常少。相比较而言, 使用爬虫对互联网上的海量文献数据进行收集、分析, 即所谓的“样本等于全体”的全数据模式,能在更短的时间内搜集更多的文献信息，之后将其化为可被计算机处理的数据形式，方便进行数据整理和统计分析。

2 项目简介

2.1 项目内容

本课程设计将实现对“知网”网站中的知识文献进行搜索和整理，选择文献的六个属性，记录并将其存入计算机的数据库之中。另外，本设计还实现了对已经获取的文献信息在数据库中进行查找、增加和删除。主要功能如下：

1）爬取所有目标文献的六类属性，并获取文献细节页面的 url；

2）与本地的数据库进行连接，以将爬取到的文献属性和 url 存入数据库中；

3）以友好、直观的方式向用户展示爬取到的文献数据；

4）对用户提供对数据库内容进行增加、查询、删除的功能接口。

2.2 爬取说明

本设计将知网中文献的六类属性：标题、作者、来源、发表日期、引用数和下载数爬取到内存中，并在进行一定的操作之后得到文献细节页面的 url，最后将以上的七类数据存入数据库之中。


数据一览

	标题	作者	来源	日期	被引数	下载数	详情页url
1	介入与创新:农...	廖金萍	湖州职业技术学...	2018-03-05 ...	0	340	https://...
2	张爱玲小说自绘...	姜雯滢	山东青年政治学...	2020-11-17 ...	0	0	https://...
3	康爱保生丸治疗...	王莉,和丽生,杨...	辽宁中医杂志	2021-04-27 ...	0	127	https://...
4	筏式养殖夹苗密...	闫令东,孙利芹,...	水产学杂志	2021-05-19 ...	0	189	https://...
5	日本爱知县：从...	张安迎;童昕;谷...	国际城市规划	2021-02-05 ...	0	440	https://...
6	加州大学系统图...	田晓迪	图书馆杂志	2022-02-18 ...	0	330	https://...
7	爱德万甜中间体...	方聪,刘怡雪,黎...	化工进展	2022-04-01 ...	0	156	https://...
8	情感结构与时代...	魏巍,李静	西华师范大学学...	2022-04-22 ...	0	153	https://...
9	广义观测相对...	阮晓钢	北京工业大学学...	2022-04-25 ...	0	757	https://...
10	巴黎-爱丁堡压...	杨功章,谢雷,陈...	物理学报	2022-04-28 ...	0	18	https://...
11	形式与思想的审...	李珍珍	西南交通大学学...	2022-05-31 ...	0	22	https://...

图 2-1 爬取内容概览

2.3 技术工具

本项目使用 Python 语言，借助 PyQt5 构造可视化界面，使用 Fiddler 工具进行 http 数据包的截获和解析，requests、selenium、lxml 等包实现数据的爬取，依托于 pandas、MySQL 数据库和相关的 SQL 命令实现已爬取数据信息内容的检索、增添和删除。

3 相关工具介绍

3.1 Python 语言

Python 是一种解释型、面向对象、动态语义、语法优美的脚本语言,自从 1989 年由 Guido Van Rossum 设计出来后,经过十余年的发展,已经同 Tcl、Perl 一起,成为目前应用最广的三种跨平台脚本语言。Python 支持现有的各种主流操作

系统,如 Microsoft Windows、Solaris、Mac OS、Linux 等,甚至包括 Palm OS 这样的嵌入式环境。它的源程序和二进制代码可以免费获得。由于其强大灵活的功能,简洁优美的语法和源代码免费开放,Python 被著名国际自由软件项目 KDE 计划选定为标准系统脚本语言,微软公司也宣布将在.NET 环境中提供对 Python 语言的支持。

与同为脚本语言的 Tcl、Perl 相比,Python 的特点有:

（1） 面向对象。Python 提供类,类的继承,类的私有和公有属性,例外处理等完善的对面向对象方法的支持。

（2） 虚拟机。像 Java 一样,Python 程序在执行前要先编译成字节码,再通过一个虚拟机解释执行。

（3） 高级数据结构 Python 内置了对列表,关联数组等常用数据结构的支持。

（4） 语法简洁优美 Python 的语法非常简单易学,并且采用缩进来表示程序的块层次结构。这样做不仅仅减少了不必要的块符号,更重要的是强制程序员用一种清晰统一的风格书写程序,增加了程序的可读性,降低了维护开销。

易于扩展和嵌入 Python 语言本身只提供了一个编程语言所需功能的最小内核,其它许多丰富的功能都由扩展模块实现。由于在设计时就考虑到了扩展性,可以很方便地用 C 或者 C++编写 Python 的扩展模块以添加新的功能，或者把解释器自身嵌入到其他程序内部。[1]

3.2 数据挖掘和网络爬虫

Web 数据挖掘是通过模拟用户正常的浏览器行为, 并设置一定的规则, 从而获取 Web 页面指定信息。Web 数据挖掘的最终目的是将非结构化的信息从海量的信息提取并以统一的方式进行存储 (以 CSV、JSON、XML 等模式存储) 。在此过程中, 将涉及网络爬虫、数据结构化与正则表达式等多种关键技术。

随着互联网技术的应用与普及, 网络中信息资源极为丰富, 但大多数信息以无结构的文笔形式存在, 导致信息采集与归类变得极为困难。在数据挖掘技术出现之前, 用户要将有效的信息进行采集和归类须采用手动复制粘贴的方式, 不仅耗时耗力, 而且数据质量较差, 难以实现数据采集与分析的自动化。而基于 Python 语言的网络爬虫技术, 具有速度快、准确性高等特点, 能够有效提升数据

采集与分析效率, 提高数据采集质量。同时人工操作难免存在数据错误、遗漏等问题, 在统计较大数据时, 纠错难度极大, 而借助网络爬虫技术, 数据准确性较高, 即使存在问题, 用户可通过规则、程序调整即可完成纠错, 具有无可比拟的应用优势。

网络爬虫是根据制定的规则对 Web 页面进行遍历查询, 从而自动抓取有效信息的脚本。网络爬虫的主要原理是通过互联网指定的子集合中读取 URL, 访问相应的 Web 内容, 并遍历 Web 页面所包含的链接, 并遍历链接继续爬取包含的子页面内容, 从而完成数据的收集、分类和整理。

当前网络爬虫核心算法主要包括广度优先、深度优先、Partial PageRank 及 Opic 爬虫算法。不同爬虫算法各有优劣, 应结合实际应用场景进行合理选择。[2]

3.3 Xpath 网页解析[3]

XML 路径语言(XML Path Language) ，被称为 Xpath，一般用于定位一个数据结构中的某个节点。由于一般的网页源码呈树状结构，Xpath 针对其进行逐层展开和定位，寻找到一个独立的标签节点。

XML 路径语言是一种用于确定某个节点在 XML 文档中的位置的计算机语言，其通过对树型文档的遍历操作实现查找出目标数据的功能。后来这种语言被广泛适用于 HTML 文档的搜索，在网络爬虫的工作中常用于网页信息的解析和抽取。

Xpath 具有简洁直观的路径表达式和强大的选择功能，可以提供超过 100 个内建函数用于数值、字符串的匹配以及节点、序列、逻辑值的处理。

Xpath 的常用规则示例如表格 1：

表格 1

表达式	示例	描述
nodename	newstitle	选取 newstitle 元素的所有子节点
/	/newstitle	选取根元素 newstitle
//	//newstitle	选取所有 newstitle 子元素，忽略位置
@	//@newstitle	选取名为 newstitle 的所有属性

在 Chrome 浏览器的开发者工具中，可以在浏览网页源码的同时直接复制

选中元素标签的 Xpath 地址，能有效地提高开发效率、解决大部分的解析困难。

3.4 PyQt[3]

PyQt 是一款应用于 GUI 应用程序创建的工具包，其融合了 Python 编程语言与 Qt 库是 Qt 特意为 Python 提供的 GUI 扩展工具包，与 Python 其他控件集，PyQt 具有相当优秀的特性：

（1） 基于 Qt 高性能的 GUI 控件集。Qt 是经挪威 Trolltech 公司开发的应用在 C++上的 GUI 控件集，具有相当优秀的跨平台性，能够在多个系统平台如 Linux、Windows、Unix 以及 Mac 等之间轻松移植；

（2） 使用信号（signal）和槽（slot）机制进行灵活通信，取代凌乱的函数指针，使控件和事件之间的通信变得简洁明了，同时也是区别于其他 GUI 控件集的重要特征；

（3） 可以将 Python 中的类直接声明为 Qt 中某个类的子类；

（4） 为开发者提供一个较为完整的控件库。

与此搭配使用的 QtDesigner 是一款将界面编辑过程可视化的界面设计工具，其界面与业务逻辑相互分离，符合 MVC 架构，在设计完成后支持生成可被 Python 调用的.py 文件，减少了设计开发人员的工作量，并能实施查看界面效果，方便对未设计完整或有漏洞的程序界面进行及时修改。

4 项目实施步骤

项目的实施步骤大体分为四步：

（1） 首先通过抓包工具明确要爬取的知网的信息存储与显示的机制和 http 报文请求发送情况，针对要爬取的文献属性进行规划，并确定爬取方案；同时通过 Qtdesigner 设计可视化窗口并生成对应代码，并完成操作逻辑的设计；

（2） 编写爬虫程序，在此同时需要破解一些加密算法。完成之后对爬取到的数据进行打包封装传出备用；

（3） 支持用户自行设置界面与本机数据库系统的连接参数，以便将爬取获得的数据存入数据库中；编写存入、读取数据库的功能代码并将之显

示在可视化窗口中；

（4） 设计对数据库内容进行增、删、查功能的代码，同样将结果显示在可视化界面上，支持用户对数据库中的内容进行直接、简便的操作；

4.1 方案确定

4.1.1 爬虫方案确定

要进行知网数据的爬取，首先需要获得在知网中进行搜索操作时访问的 url 地址和发送的 POST 报文内容。使用“航天”为关键词，“主题”为搜索参考选项，通过 Fiddler 抓包后结果如图 4-1 和图 4-2：

图 4-1 请求头相关信息

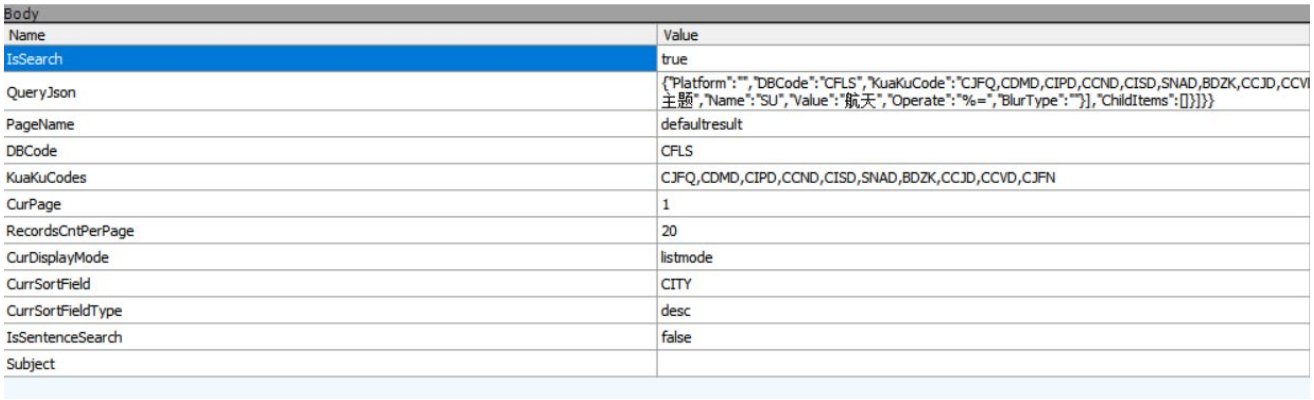

图 4-2 请求表单信息

可以得知搜索目标 url 为 https://kns.cnki.net/kns8/Brief/GetGridTableHtml，并且 POST 表单发送的形式是普通的 url 编码，而不是 json 格式。在发送的表单中的 QueryJson 项，其值是一个嵌套了多层字典的字典，在最内层的字典中存在我们搜索的关键词和搜索附加条件，考虑这可能就是搜索的参数。

所以，在将最内层的字典中的“航天”改为“航海”并将整个表单内容重新进行 url 编码后，使用 Composer 功能重新发包，得到返回结果后与正常搜索结果对比：

	题名	作者	来源	发表时间	数据库	被引	下载	操作
1	MEMS传感器现状及应用	王淑华	微纳电子技术	2011-08-15	期刊	341	13357	下载
2	惯性技术研究现状及发展趋势	王巍	自动化学报	2013-06-15	期刊	314	4597	下载
3	捷联惯性导航系统关键技术研究	杨艳娟	哈尔滨工程大学	2001-06-01	博士	183	8385	下载
4	基于四元数改进型互补滤波的MEMS姿态解算	陈孟元; 谢义建; 陈跃东	电子测量与仪器学报	2015-09-15	期刊	177	2652	下载
5	声呐波束形成与波束域高分辨方位估计技术研究	杨益新	西北工业大学	2002-04-01	博士	137	4543	下载
6	基于MEMS惯性传感器的机器人姿态检测系统的研究	秦勇; 臧希喆; 王晓宇; 赵杰; 蔡鹤皋	传感技术学报	2007-02-28	期刊	131	2186	下载
7	博物馆儿童教育研究	周婧景	复旦大学	2013-05-31	博士	129	22564	下载
8	MEMS微陀螺仪研究进展	成宇翔; 张卫平; 陈文元; 崔峰; 刘武; 吴校生	微纳电子技术	2011-05-15	期刊	127	4804	下载

	题名	作者	来源	发表时间	数据库	被引	下载	操作
1	MEMS传感器现状及应用	王淑华	微纳电子技术	2011-08-15	期刊	341	13357	
2	惯性技术研究现状及发展趋势	王巍	自动化学报	2013-06-15	期刊	314	4597	
3	捷联惯性导航系统关键技术研究	杨艳娟	哈尔滨工程大学	2001-06-01	博士	183	8385	
4	基于四元数改进型互补滤波的MEMS姿态解算	陈孟元; 谢义建; 陈跃东	电子测量与仪器学报	2015-09-15	期刊	177	2652	
5	声呐波束形成与波束域高分辨方位估计技术研究	杨益新	西北工业大学	2002-04-01	博士	137	4543	
6	基于MEMS惯性传感器的机器人姿态检测系统的研究	秦勇; 臧希喆; 王晓宇; 赵杰; 蔡鹤皋	传感技术学报	2007-02-28	期刊	131	2186	
7	博物馆儿童教育研究	周婧景	复旦大学	2013-05-31	博士	129	22564	
8	MEMS微陀螺仪研究进展	成宇翔;张卫平;陈文元;崔峰;刘武	微纳电子技术	2011-05-15	期刊	127	4804	

图 4-3 POST 搜索和真实搜索对比

由图 4-3 可知，猜测正确，最内层字典的确是用于控制搜索选项的。

由于本设计期望支持用户通过标题和作者两种不同的方式进行文献搜索和爬取，因此，以“艾伦”为关键词，“作者”为搜索条件进行搜索，使用 Fiddler 抓包结果如下：

Body

Name	Value
IsSearch	true
QueryJson	{"Platform":"","DBCode":"CFLS","KuaKuCode":"CJFQ,CDMD,CIPD,CCND,CISD,SNAD,BDZK,CCJD,CCVD 作者","Name":"AU","Value":"艾伦","Operate":"=","BlurType":""}],"ChildItems":[]}]}
SearchSql	0645419CC2F0B23BC604FFC82ADF67C6E920108EDAD48468E8156BA693E89F481391D6F5096D7FFI
PageName	defaultresult
HandlerId	1
DBCode	CFLS
KuaKuCodes	CJFQ,CDMD,CIPD,CCND,CISD,SNAD,BDZK,CCJD,CCVD,CJFN
CurPage	1
RecordsCntPerPage	20
CurDisplayMode	listmode
CurrSortField	CITY
CurrSortFieldType	desc
IsSortSearch	false
IsSentenceSearch	false
Subject	

图 4-4 “艾伦”请求表单信息

通过文本比对可知，图 4-2 与图 4-4 的不同点主要在于内层字典中的

“Title”、“Name”和“Value”项的值。因此，可以推断这三个键值共同表示用户搜索的选项。

最后，还需要确认要爬取的属性是否存在在网页中并可以通过 xpath 找到，而不是动态网页的某个请求返回的 Frame。以主题搜索“航海”为例，使用 Chrome 浏览器提供的开发者工具对网页框架进行解析：

图 4-5 文献属性框架分析

由图 4-5 可以得知，本页面所有的文献信息被划分为不同的<tr>标签，在<tr>标签内部还存在多个<td>标签来存储该文献的各项属性。因此，这些属性信息位置都可以被 xpath 表示，也可以被方便地捕获。

至此，爬虫方案已经设计完成：遵循用户设定的爬取选项，向目标 url 发送 url 编码格式的合适 POST 请求获取所有搜索结果文献，然后使用 xpath 定位每篇文献的属性信息（标题、作者、来源、发表时间、被引数、下载数、详情 url）并爬取。

4.1.2 界面设计与操作逻辑

进入界面后，用户首先需要配置连接数据库的参数：

图 4-6 数据库参数配置界面

然后即可进入应用主界面，用户可以在此输入要爬取的文献信息并调整搜索条件，在输入之后还可以进行预览，以确定是否真的要开始爬取。用户可以点击“开始爬取”按钮进入爬取设置界面，也可以点击“数据管理”按钮进入数据一览界面查看已经爬取保存的数据，并决定是否操作此数据库：

图 4-7 爬取系统主界面

在点击“爬取”按钮后，用户进入爬取设置界面，在此处可以设置爬取的页数和存储方式，按下“开始”按钮后才真正开始爬取。

存储方式包括存储至文件根目录（当前项目源代码所处目录），以 csv 逗号分隔形式存储，和存储至进入程序时指定的数据库两个选项。由于在程序的调试中发现 SQLServer 数据库的 ODBC 连接速度过慢导致软件卡死，所以此处的指定数据库必须是 MySQL，程序界面如图 4-8：

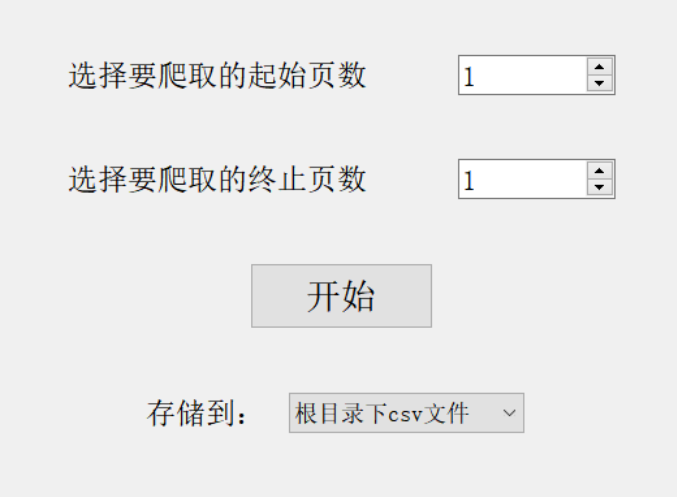

图 4-8 爬取设置界面

如果用户在主界面点击了“数据管理”按钮，则进入数据一览界面，可以在此对数据库内信息进行总览，并决定是否进行管理操作：

	标题	作者	来源	日期	被引数	下载数	详情页
1	介入与创新:农...	廖金萍	湖州职业技术学...	2018-03-05 ...	0	340	https://...
2	康爱保生丸治疗...	王莉,和丽生,杨...	辽宁中医杂志	2021-04-27 ...	0	127	https://...
3	筏式养殖夹苗密...	闫令东,孙利芹,...	水产学杂志	2021-05-19 ...	0	189	https://...
4	加州大学系统图...	田晓迪	图书馆杂志	2022-02-18 ...	0	330	https://...
5	爱德万甜中间体...	方聪,刘怡雪,黎...	化工进展	2022-04-01 ...	0	156	https://...
6	情感结构与时代...	魏巍,李静	西华师范大学学...	2022-04-22 ...	0	153	https://...
7	广义观测相对...	阮晓钢	北京工业大学学...	2022-04-25 ...	0	757	https://...
8	巴黎-爱丁堡压...	杨功章,谢雷,陈...	物理学报	2022-04-28 ...	0	18	https://...
9	形式与思想的审...	李珍珍	西南交通大学学...	2022-05-31 ...	0	22	https://...
10	“故事”内外的自...	关琳琳	文艺论坛	2022-06-09 ...	0	41	https://...

图 4-9 数据总览界面

若用户选择“进行操作”，则进入“管理数据”页面，在此用户可以对数据库中内容进行增、删、查和总览操作：

图 4-10 数据管理界面

如图 4-10，数据管理界面除总览功能外，还拥有三个直接执行 SQL 命令的功能：

查询：用户根据需要在一个或多个输入框中输入查找条件。程序设定标题、作者、来源为模糊搜索；日期、被引数、下载数为精确搜索。搜索结束后将结果在上方的表框中展示；

增加：程序要求用户输入所有的输入框，然后将内容加入数据库中；

删除：按照用户输入的文献属性信息将指定的一条数据删除，其搜索逻辑与查询功能相同。

4.2 编写爬虫程序

4.2.1 请求构造

爬虫程序的请求构造需要使用 requests 和 urllib 库中的方法。

首先根据 Fiddler 工具抓取到的搜索 POST 请求构建爬虫的请求头，需要注意的是请求头中需要带有本次访问的 cookie，否则服务器将返回错误信息。

接下来将由 Fiddler 抓取的目标 url 写入程序之中，作为每次爬取信息发送请求的固定 url 站点。

最后，构造 POST 请求发送的表单数据。因为表单数据中存在多个字典嵌套的情况，因此需要手动进行字典的嵌套，并且由于用户对爬取模式设置的不同，需要注意最内层字典的封装：

```
dic1={"Title":"搜索方式","Name":"方式","Value":"搜索内容","Operate":"=","BlurType":""}
dic1['Value']=searchItem
if getMethod==0:
    dic1['Title']='作者'
    dic1['Name']='AU'
else:
    dic1['Title']='主题'
    dic1['Name']='SU'
    dic1['Operate']="%="

dic2={"Key":"Subject","Title":"","Logic":1,"Items":[dic1],"ChildItems":[]}
dic3={"QGroup":[dic2]}
dic4={"Platform":"","DBCode":"CFLS","KuaKuCode":"CJFQ,CDMD,CIPD,CCND,CISD,SNAD,BDZK,CCJD,CCVD,CJFN","QNode":dic3}
dic={'IsSearch':'false',
'QueryJson':dic4,
'PageName': 'defaultresult',
'DBCode': 'CFLS',
'KuaKuCodes': 'CJFQ,CDMD,CIPD,CCND,CISD,SNAD,BDZK,CCJD,CCVD,CJFN',
'CurPage': 1,
'RecordsCntPerPage': 20,
'CurDisplayMode': 'listmode',
'CurrSortField': 'CITY',
'CurrSortFieldType': 'desc',
'IsSentenceSearch': False,
'Subject': ''}
dic['CurPage']=str(page)
```

图 4-11 POST 表单数据构造

构造表单完成后，还需要借助 urllib.parse()方法对构造完成的表单进行 url 编码，以契合服务器对 POST 表单编码的要求。

在以上三个必要元素具备后，调用 requests.post()方法向服务器发送 POST 请求，将返回的内容使用.text()方法转换后获得预期的响应。

4.2.2 信息解析与 Debug

信息的解析借助 lxml 库中的 etree 类，可以精确地通过 xpath 定位网页中元素的标签信息；使用 pandas 的方法和数据结构来将解析的结果临时保存在内存中，并在之后按照用户的选择将其存入 csv 或数据库中。

首先将 POST 请求的响应使用 lxml.etree 中的方法读入，以便进行解析。

观察发现知网的一页搜索结果中包含 20 条文献信息，因此考虑使用循环次数为 20 的 for 循环实现对每一个<tr>标签的解析获取，在循环内部依次解析<td>标签。每一个<td>标签都表示某个文献的某一项属性，将解析出的相同属性存储至一个列表之中，如此在 for 循环结束之后即获得 7 个长度为 20 的列表，每个列表都只存储某一项属性。

在测试中，发现 etree 无法找到由 Chrome 生成的元素 xpath 路径，最终的程序返回空列表，如图 4-12：

图 4-12 xpath 路径不可用

开始排除问题，考虑逐级降低路径等级，观察在降低到何种程度时 etree 才能获取到指定标签。当路径降低至<table>时，etree 报告找到元素：

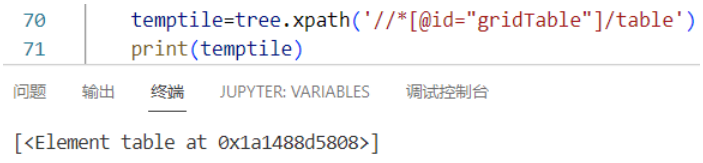

图 4-13 找到 table 标签

通过对比图 4-12 和图 4-13 可以得知，错误出现在找不到 <tbody>标签。通过在 Chrome 中查看网页源代码可知，网页中的<tbody>标签应是被动态加载出现的，而 requests.POST 的响应中并不包含此种结构，所以 etree 不能在响应中找到此标签。在路径中去除<table>标签后，爬取正常。如图 4-14：

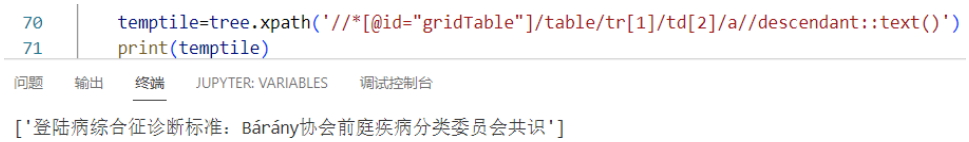

图 4-14 路径正确 爬取正常

爬取完成后，在检查得到的数据时，发现直接从文献标题标签中获取的 href 超链接在点击进入后提示“访问方式错误”无法进入。考虑是访问到了已经弃用的网页链接或是网站对真实的 url 进行了隐藏，因此需要寻找真实的详情页 url。

首先观察访问失败的 url，内容如下：

https://kns.cnki.net/kns8/Detail?sfield=fn&QueryID=0&CurRec=1&recid=&FileName=FUHE200701000&DbName=CJFD2007&DbCode=CJFD&yx=&pr=&URLID=

打开 Chrome 中的抓包工具，对打开详情页时的数据包信息进行抓取：

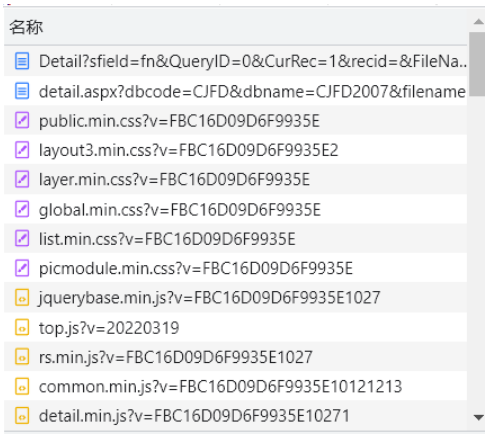

图 4-15 Chrome 抓详情页包

发现对页面的请求只有两个，这其中的第一个请求就是被 302 转移访问的请求，只剩余一个能够请求页面的请求，推测这就是真实地址，将其在浏览器打开后发现可以看到正常的页面，证实了猜测。

获知一篇文章的详情页真实 url 并不代表能够访问所有爬取到文章的详情页真实 url，因此需要寻找该真实 url 的生成算法。

观察真实 url，内容如下：

https://kns.cnki.net/kcms/detail/detail.aspx?dbcode=CJFD&dbname=CJFD2007&filename=FUHE200701000&uniplatform=NZKPT&v=1Eln23L5USKJA6BV5jKwGKMZdklSpgllEoTSts8jhpGya-8FbHVIkisNi_OSqwKc

推测载荷中的 dbname、filename 两个参数区分了不同文献的详情页。为了验证猜测，再次抓取一个真实地址：

https://kns.cnki.net/kcms/detail/detail.aspx?dbcode=CJFD&dbname=CJFD2014&filename=DLXB201401002&uniplatform=NZKPT&v=pjgzYe6F0FJ6PkRTsiJWqg-onV6iSZ-LRJMQFns34x_9mRu5ebt1-gUCcLzDKLE8

在进行文本比对后发现，二者载荷中的 dbname、filename、v 值确实存在差异，尤其以 v 的差异最大。但考虑到 v 的值内容杂乱无序，认为此项并不能作为区分不同文献详情页的因素。

通过将同一篇文献的无效 url 与真实 url 对比发现，dbname 项和 filename 项在容易获取的无效 url 中同样存在，因此，存在一套算法，可以由无效 url 推导出真实 url，考虑使用字符串的拼接，算法如下：

真实 url=f”

https://kns.cnki.net/kcms/detail/detail.aspx?dbcode=CJFD&dbname={dbname}&filename={filename}&uniplatform=NZKPT&v=pjgzYe6F0FJ6PkRTsiJWqg-onV6iSZ-LRJMQFns34x_9mRu5ebt1-gUCcLzDKLE8”

在程序中实现时，只需要如图 4-16，截取无效 url 中的 dbname 和 filename 两项，之后再使用 f 表达式将字符串拼接为真实 url 即可：

```
tempurl=tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[2]/a/@href')
tempurl="".join(tempurl)
tempurl=tempurl.split("&")
filename_url=tempurl[4][9:]
dbname_url=tempurl[5][7:]
```

图 4-16 截取两项 name

最后，将在每次循环中解析出的各类文献属性存入各自的大列表中，之后手动构造一个字典，每一个键值对都是一个字符串对各类列表的描述；调用 pandas 的 DataFrame 方法将字典转换成 DataFrame 数据形式并从函数中返回，作为一次爬取的结果。

4.3 数据存储和读取

在本步骤中需要实现用户自定义数据库连接参数的功能，实现将爬取数据存入数据库的功能、从数据库读取数据的功能以及将读取数据显示在可视化窗口中的功能。

4.3.1 自定义连接参数

考虑使用 sqlalchemy 库实现对 MySQL 数据库的连接，并安装 pysql 库以支持本次连接。根据 **4.1.2 界面设计与操作逻辑**一节中确定的数据库连接界面，连接参数具有“数据库名称”、“用户名”、“密码”、“表名”四项。其中前三项是用于引擎建立连接的，而最后一项是用于确定在对数据库进行操作时的被操作表。

首先程序需要读入用户在此界面输入的所有信息并存储为全局变量，以供其他类方法直接调用替换。为所有的数据库操作单独创建一个 py 文件，在其中定义各类操作函数，在函数每次被调用时都首先将 sqlalchemy.create_engine 类通过带参构造实例化为 engine 对象。如此一来，之后在进行各类数据库操作时都可以直接使用该 engine 对象作为一些读取、写入方法的参数。

另外，如果用户输入了错误的参数信息导致连接数据库失败，则抛出错误

提示框，指明发生错误的可能原因。

在 engine 类实例化时使用的带参构造函数的参数需要使用先前所定义的全局变量：使用 f 表达式将连接参数字符串进行占位替换。对于只需要执行 SQL 命令而不需要读取或存储的函数，要另外对 engine 对象使用.connect()方法，查看源码后得知，此方法返回值是一个新的类对象：

```
def connect(self, close_with_result=False):
    """Return a new :class:`_engine.Connection` object.

    The :class:`_engine.Connection` object is a facade that uses a DBAPI
    connection internally in order to communicate with the database.  This
    connection is procured from the connection-holding :class:`_pool.Pool`
    referenced by this :class:`_engine.Engine`. When the
    :meth:`_engine.Connection.close` method of the
    :class:`_engine.Connection` object
    is called, the underlying DBAPI connection is then returned to the
    connection pool, where it may be used again in a subsequent call to
    :meth:`_engine.Engine.connect`.

    """

    return self._connection_cls(self, close_with_result=close_with_result)
```

图 4-17 connect()方法返回值说明

最后对返回值使用.execute()方法即可执行 SQL 命令。

4.3.2 存入数据库

考虑对爬取到的已临时存储为 DataFrame 格式的数据使用 pandas 库提供的.to_sql()方法，查看方法所需参数：

```
def to_sql(
    self,
    name: str,
    con,
    schema=None,
    if_exists: str = "fail",
    index: bool_t = True,
    index_label=None,
    chunksize=None,
    dtype=None,
    method=None,
) -> None:
```

图 4-18 .to_sql()参数

如图 4-18 所示，本方法包括 2 个必填参数，7 个可选默认参数。

继续查看源码中的参数说明可知，name 参数接收要存储至的表名。在本应用中表名已经被用户手动录入完成并存储为全局变量，所以在此处可以直接调用；观察 con 参数说明，如图 4-19：

```
con : sqlalchemy.engine.(Engine or Connection) or sqlite3.Connection
    Using SQLAlchemy makes it possible to use any DB supported by that
    library. Legacy support is provided for sqlite3.Connection objects. The user
    is responsible for engine disposal and connection closure for the SQLAlchemy
    connectable See `here \
        <https://docs.sqlalchemy.org/en/13/core/connections.html>`_.
```

图 4-19 con 参数说明

可以得知，con 参数接收一个 sqlalchemy.engine 对象，即在之前的步骤中已经实例化并设置完成的 engine 对象，所以在此可以直接调用传入。

接着设置 index=false，if_exists="append"即如果表已经存在则向表中添加数据。最后进行方法调用即可完成数据库的写入操作。

4.3.3 读取数据库并显示

考虑使用 pandas 库提供的.read_sql()方法，传入参数如下：

```
@overload
def read_sql(
    sql,
    con,
    index_col=None,
    coerce_float=True,
    params=None,
    parse_dates=None,
    columns=None,
    chunksize: None = None,
) -> DataFrame:
```

图 4-20 .read_sql()参数

通过查看如图 4-20，.read_sql()的其中一个方法重载可以得知，方法接收 2 个必填参数和 6 个选填默认参数，并在最后返回 DataFrame 数据类型，需要传入 SQL 命令和 engine 对象。

由于需要展示数据库内的所有内容，所以 SQL 命令写为 f"select * from {tablename}"，在执行方法结束后使用一个空的 DataFrame 数据接取其返回值。

为了将返回的 DataFrame 类型执行结果在可视化界面中的 TableWidget 控件上展示出来，需要一个函数接收 DataFrame 数据，并直接修改控件对象的内容。通过查阅资料，获得以下函数：

```
def display_dynamic_form(self, df, target_obj):

    # horizontalHeader().setVisible
    # .verticalHeader().setVisible
    input_table_rows = df.shape[0]
    input_table_columns = df.shape[1]
    input_table_header = df.columns.values.tolist()
    target_obj.setColumnCount(input_table_columns)
    target_obj.setRowCount(input_table_rows)
    target_obj.setHorizontalHeaderLabels(input_table_header)
    # print(input_table_header)
    for i in range(input_table_rows):
        for j in range(input_table_columns):
            new_item = QTableWidgetItem(str(df.iat[i, j]))
            target_obj.setItem(i, j, new_item)
```

图 4-21 展示 df 函数

在窗口类中调用此方法，传入读取结果 df 和目标控件对象名后，数据库读取结果成功显示在控件中。

4.4 数据库操作设计

本设计所爬取的信息属于知网的文献属性，这类数据在本地数据库中基本不需要修改操作，只需重新爬取正确内容即可。因此只需要实现用户对数据库的增、删、查功能。

由图 4-10 的设计，程序需要在一次操作中依次读取所有输入框中的内容，然后将这些内容整合为一条 SQL 命令并执行。对于查找功能，需要在用户点击按钮后立刻显示结果；而增加、删除功能则不需要直接将结果展示给用户。因此，查找功能考虑使用既接收 SQL 命令、又返回 DataFrame 数据的.read_sql()方法；增加和删除功能考虑使用 engine 对象的.execute()方法直接传入 SQL 命令并执行，无需返回值。

SQL 命令主要由三个部分构成：select 的元素、from 的表名和 where 的搜索条件。在三个操作当中，查询、删除操作的 where 条件应该都是相同的，所以只需要编写一个方法来进行 where 条件字符串的拼接。

对查询功能来说，用户只能输入查询选项即 where 条件，所以程序默认为用户显示搜索到内容的全部属性，即”select * from {TableName}”。

对于增加功能来说，系统不允许输入框留空，SQL 命令相对更简单，单独编写其 SQL 命令字符串的生成算法即可。

4.4.1 查询、删除功能

因为查询删除两项功能的设计是允许用户对某些编辑框留空的，因此在SQL 命令的生成算法中需要考虑到输入条件个数的不同：既不能在最后的条件后多出一个”and”，也不能在命令中间出现多个连续的”and”。

同时，还要考虑到进行模糊搜索条件的语法变化，从“=”变为“like %%”。

要处理模糊搜索的问题，计划定义一个函数专门生成”‘标题’={title}”类型的短字符串，其传入参数分别是属性名称、属性值和是否是模糊搜索：

```
def buildPartQuery(self,type:str,content:str,isLike:bool=True):
    if content=='':
        return ''
    if isLike:
        queryStr=f"{type} like '%{content}%'"
    else:
        queryStr=f"{type} = '{content}'"
    return queryStr
```

图 4-22 模糊搜索处理方法

接着，对于“and”的控制问题，计划先简单生成可能存在 bug 的字符串，接着以“and”为标志将字符串划分为列表，再使用循环将列表中所有的空元素去除，最后再使用.join()方法将“and”重新插入每个元素之间并返回最终的字符串。如图 4-23 为此算法的流程：

```
def buildWhereQuery(self,contents:tuple):
    title,author,source,date,use,download=contents
    query=f"{self.buildPartQuery('标题',title)}and{self.buildPartQuery('作者',author)}and{self.buildPartQuery('来源',source)}and{self.
    query=query.split("and")
    while '' in query:
        query.remove('')
    query=" and ".join(query)
    return query
```

图 4-23 “and”处理方法

4.4.2 增加功能

数据增加的 SQL 语句十分简单，并且由于先前已经使用过.to_sql()方法，直接调用即可。

5 实验总结

本次课程设计是一次令人难忘的经历。起初，根据以往编写爬虫代码、操

作数据库的经验，我以为这只是一项小工程，也许只需要调用几个库，在同一个文件里就能解决；后来使用了在制作中学习到了 PyQt 的使用，才明白原来制作一个软件的界面原来需要这么大的工作量：从界面布局、操作逻辑的规划，到非法输入、违规操作的处理，最后到界面按钮与代码算法的对接、各类方法的编写，都让我对 Python 这门语言，乃至面向对象编程这种思想都有了更深的理解。

在进行课程设计中，我遇到了很多过去从来没有遇见过的问题和困难：对 xpath 无法定位问题的解决和寻找文献详情页的真实 url 等等。不过，我认为 PyQt 界面之间的交互方式的学习和适应是这次课程设计中最难的一关，无论是类的相互继承、super()语句的使用，还是函数与方法的区分和编写，都是需要我们去认真学习的东西。关于数据库的配置也给我造成了很大的困扰，甚至让我最终抛弃了 SQLServer 转而使用 MySQL，这其中又发生了许多令人困惑的问题，但最终我都一一解决了，并搞明白了错误的原因。

在遇到大量纷繁复杂的困难的同时，我也学到了很多过去没听说过的、没学习过的技术知识和技巧。例如此次实验让我认识到了 Chrome 自带的抓包系统的优势：直观而方便，这在我之前的使用中是从来没有体会过的；在进行数据库的保存与读取时，我学习到了数据库是如何通过 ODBC 与软件进行对接的，并认识到了 pandas 库中更多强大的方法，让我对这一数据分析处理库有了更加深入的认识和理解。

在今后的学习生活中，我将始终秉持虚心学习、刻苦钻研的学习态度，不断增强编写代码的能力，多多实践，以实际行动代替“云学习”；提高算法编写能力，多做一些如力扣、牛客等算法题的训练，提高编程素养和数据思维；继续学习 Python、Java、C++等面向对象的编程语言，深入体会编程的乐趣和奥妙所在。

6 参考文献

[1] 吴爽. 基于 python 语言的 web 数据挖掘与分析研究[J/OL]. 电脑知识与技术, 2018, 14(27): 1-2. DOI:10.14004/j.cnki.ckt.2018.2895.

[2] 网络爬虫技术的研究 - 中国知网 [EB/OL]. [2022-07-02].

https://webvpn.xaut.edu.cn/https/77726476706e69737468656265737421fbf952d2243e635930068cb8/kcms/detail/detail.aspx?dbcode=CJFD&dbname=CJFD2010&filename=DNZS201015111&uniplatform=NZKPT&v=c0MY7xH7RRH6viQhixID3UTOlSUo0v0gWLDRpW7wikZU8rvj2hYpdrffvELyB1mN.

[3] 基于 Python 的网络新闻爬虫与检索 - 中国知网[EB/OL]. [2022-07-03]. https://kns.cnki.net/kcms/detail/detail.aspx?filename=RJDK201905040&dbcode=CJFD&dbname=CJFDTEMP&v=sTbm63uFVypoA0G5c4BwFjXZHRhP7vesj7iSNq2VxriCgiIsqEv93E-8kwEhrk15.

附录 1：程序代码

1. 主界面生成代码

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Main(object):
    def setupUi(self, Main):
        Main.setObjectName("Main")
        Main.resize(800, 500)
        self.Title = QtWidgets.QLabel(Main)
        self.Title.setGeometry(QtCore.QRect(200, 30, 400, 61))
        font = QtGui.QFont()
        font.setFamily("Adobe Devanagari")
        font.setPointSize(27)
        font.setBold(True)
        font.setWeight(75)
        self.Title.setFont(font)
        self.Title.setAlignment(QtCore.Qt.AlignCenter)
        self.Title.setObjectName("Title")
        self.search_Botton = QtWidgets.QPushButton(Main)
        self.search_Botton.setGeometry(QtCore.QRect(560, 140, 141, 51))
        font = QtGui.QFont()
        font.setPointSize(16)
        self.search_Botton.setFont(font)
        self.search_Botton.setObjectName("search_Botton")
        self.inputLine = QtWidgets.QLineEdit(Main)
        self.inputLine.setGeometry(QtCore.QRect(90, 140, 261, 51))
        font = QtGui.QFont()
        font.setFamily("Adobe Devanagari")
        font.setPointSize(16)
        self.inputLine.setFont(font)
```

```
            self.inputLine.setText("")
            self.inputLine.setObjectName("inputLine")
            self.to_Getting = QtWidgets.QPushButton(Main)
            self.to_Getting.setGeometry(QtCore.QRect(160, 330, 141, 51))
            font = QtGui.QFont()
            font.setPointSize(16)
            self.to_Getting.setFont(font)
            self.to_Getting.setObjectName("to_Getting")
            self.to_dataControl = QtWidgets.QPushButton(Main)
            self.to_dataControl.setGeometry(QtCore.QRect(480, 330, 141,
    51))
            font = QtGui.QFont()
            font.setPointSize(16)
            self.to_dataControl.setFont(font)
            self.to_dataControl.setObjectName("to_dataControl")
            self.comboBox = QtWidgets.QComboBox(Main)
            self.comboBox.setGeometry(QtCore.QRect(410, 150, 101, 41))
            font = QtGui.QFont()
            font.setPointSize(13)
            self.comboBox.setFont(font)
            self.comboBox.setMaxVisibleItems(10)
            self.comboBox.setObjectName("comboBox")
            self.comboBox.addItem("")
            self.comboBox.addItem("")

            self.retranslateUi(Main)
            self.to_Getting.clicked.connect(Main.goToGettingPage)
            self.to_dataControl.clicked.connect(Main.goToDataPage)
            QtCore.QMetaObject.connectSlotsByName(Main)

        def retranslateUi(self, Main):
            _translate = QtCore.QCoreApplication.translate
            Main.setWindowTitle(_translate("Main", "Frame"))
            self.Title.setText(_translate("Main", "知网数据爬取系统"))
            self.search_Botton.setText(_translate("Main", "预览"))
            self.to_Getting.setText(_translate("Main", "爬取"))
            self.to_dataControl.setText(_translate("Main", "数据管理"))
            self.comboBox.setItemText(0, _translate("Main", "按作者"))
            self.comboBox.setItemText(1, _translate("Main", "按主题"))
```

2. 爬取选项界面生成代码

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Getting(object):
```

```
70        def setupUi(self, Getting):
71            Getting.setWindowFlags(QtCore.Qt.WindowStaysOnTopHint)
72            Getting.setWindowModality(QtCore.Qt.ApplicationModal)
73            Getting.setObjectName("Getting")
74            Getting.resize(532, 395)
75            self.Start = QtWidgets.QPushButton(Getting)
76            self.Start.setGeometry(QtCore.QRect(200, 210, 141, 51))
77            font = QtGui.QFont()
78            font.setPointSize(16)
79            self.Start.setFont(font)
80            self.Start.setObjectName("Start")
81            self.label = QtWidgets.QLabel(Getting)
82            self.label.setGeometry(QtCore.QRect(50, 50, 251, 31))
83            font = QtGui.QFont()
84            font.setPointSize(14)
85            self.label.setFont(font)
86            self.label.setAlignment(QtCore.Qt.AlignCenter)
87            self.label.setObjectName("label")
88            self.page_Start = QtWidgets.QSpinBox(Getting)
89            self.page_Start.setGeometry(QtCore.QRect(360, 50, 121, 31))
90            font = QtGui.QFont()
91            font.setPointSize(13)
92            self.page_Start.setFont(font)
93            self.page_Start.setMaximum(9)
94            self.page_Start.setMinimum(1)
95            self.page_Start.setObjectName("page_Start")
96            self.label_2 = QtWidgets.QLabel(Getting)
97            self.label_2.setGeometry(QtCore.QRect(50, 130, 251, 31))
98            font = QtGui.QFont()
99            font.setPointSize(14)
100           self.label_2.setFont(font)
101           self.label_2.setAlignment(QtCore.Qt.AlignCenter)
102           self.label_2.setObjectName("label_2")
103           self.page_End = QtWidgets.QSpinBox(Getting)
104           self.page_End.setGeometry(QtCore.QRect(360, 130, 121, 31))
105           font = QtGui.QFont()
106           font.setPointSize(13)
107           self.page_End.setFont(font)
108           self.page_End.setMaximum(9)
109           self.page_End.setMinimum(1)
110           self.page_End.setObjectName("page_End")
111           self.storeMethod = QtWidgets.QComboBox(Getting)
112           self.storeMethod.setGeometry(QtCore.QRect(230, 310, 181, 31)
    )
```

```
            font = QtGui.QFont()
            font.setPointSize(11)
            self.storeMethod.setFont(font)
            self.storeMethod.setObjectName("storeMethod")
            self.storeMethod.addItem("")
            self.storeMethod.addItem("")
            self.label_3 = QtWidgets.QLabel(Getting)
            self.label_3.setGeometry(QtCore.QRect(120, 310, 91, 31))
            font = QtGui.QFont()
            font.setPointSize(14)
            self.label_3.setFont(font)
            self.label_3.setObjectName("label_3")

            self.retranslateUi(Getting)
            QtCore.QMetaObject.connectSlotsByName(Getting)

        def retranslateUi(self, Getting):
            _translate = QtCore.QCoreApplication.translate
            Getting.setWindowTitle(_translate("Getting", "Form"))
            self.Start.setText(_translate("Getting", "开始"))
            self.label.setText(_translate("Getting", "选择要爬取的起始页数
"))
            self.label_2.setText(_translate("Getting", "选择要爬取的终止页
数"))
            self.storeMethod.setItemText(0, _translate("Getting", "根目录
下 csv 文件"))
            self.storeMethod.setItemText(1, _translate("Getting", "指定数
据库"))
            self.label_3.setText(_translate("Getting", "存储到："))
```

3）数据库连接配置界面生成代码

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_DataControl(object):
    def setupUi(self, DataControl):
        DataControl.setWindowFlags(QtCore.Qt.WindowStaysOnTopHint)
        DataControl.setWindowModality(QtCore.Qt.ApplicationModal)
        DataControl.setObjectName("DataControl")
        DataControl.resize(653, 554)
        self.formLayoutWidget = QtWidgets.QWidget(DataControl)
        self.formLayoutWidget.setGeometry(QtCore.QRect(100, 100, 443
, 271))
        self.formLayoutWidget.setSizeIncrement(QtCore.QSize(0, 0))
        self.formLayoutWidget.setBaseSize(QtCore.QSize(0, 0))
```

```
150             font = QtGui.QFont()
151             font.setPointSize(16)
152             self.formLayoutWidget.setFont(font)
153             self.formLayoutWidget.setObjectName("formLayoutWidget")
154             self.formLayout = QtWidgets.QFormLayout(self.formLayoutWidge
    t)
155             self.formLayout.setSizeConstraint(QtWidgets.QLayout.SetDefau
    ltConstraint)
156             self.formLayout.setLabelAlignment(QtCore.Qt.AlignRight|QtCor
    e.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
157             self.formLayout.setContentsMargins(0, 0, 0, 0)
158             self.formLayout.setHorizontalSpacing(33)
159             self.formLayout.setVerticalSpacing(41)
160             self.formLayout.setObjectName("formLayout")
161             self.label = QtWidgets.QLabel(self.formLayoutWidget)
162             self.label.setSizeIncrement(QtCore.QSize(0, 0))
163             self.label.setBaseSize(QtCore.QSize(0, 0))
164             font = QtGui.QFont()
165             font.setPointSize(16)
166             self.label.setFont(font)
167             self.label.setObjectName("label")
168             self.formLayout.setWidget(0, QtWidgets.QFormLayout.LabelRole
    , self.label)
169             self.DataBaseName = QtWidgets.QLineEdit(self.formLayoutWidge
    t)
170             self.DataBaseName.setSizeIncrement(QtCore.QSize(0, 0))
171             self.DataBaseName.setBaseSize(QtCore.QSize(0, 0))
172             font = QtGui.QFont()
173             font.setPointSize(16)
174             self.DataBaseName.setFont(font)
175             self.DataBaseName.setObjectName("DataBaseName")
176             self.formLayout.setWidget(0, QtWidgets.QFormLayout.FieldRole
    , self.DataBaseName)
177             self.label_2 = QtWidgets.QLabel(self.formLayoutWidget)
178             self.label_2.setSizeIncrement(QtCore.QSize(0, 0))
179             self.label_2.setBaseSize(QtCore.QSize(0, 0))
180             font = QtGui.QFont()
181             font.setPointSize(16)
182             self.label_2.setFont(font)
183             self.label_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.A
    lignLeft|QtCore.Qt.AlignVCenter)
184             self.label_2.setObjectName("label_2")
185             self.formLayout.setWidget(1, QtWidgets.QFormLayout.LabelRole
    , self.label_2)
```

```
            self.UserName = QtWidgets.QLineEdit(self.formLayoutWidget)
            self.UserName.setSizeIncrement(QtCore.QSize(0, 0))
            self.UserName.setBaseSize(QtCore.QSize(0, 0))
            font = QtGui.QFont()
            font.setPointSize(16)
            self.UserName.setFont(font)
            self.UserName.setObjectName("UserName")
            self.formLayout.setWidget(1, QtWidgets.QFormLayout.FieldRole
    , self.UserName)
            self.label_3 = QtWidgets.QLabel(self.formLayoutWidget)
            self.label_3.setSizeIncrement(QtCore.QSize(0, 0))
            self.label_3.setBaseSize(QtCore.QSize(0, 0))
            font = QtGui.QFont()
            font.setPointSize(16)
            self.label_3.setFont(font)
            self.label_3.setObjectName("label_3")
            self.formLayout.setWidget(2, QtWidgets.QFormLayout.LabelRole
    , self.label_3)
            self.PassWord = QtWidgets.QLineEdit(self.formLayoutWidget)
            self.PassWord.setSizeIncrement(QtCore.QSize(0, 0))
            self.PassWord.setBaseSize(QtCore.QSize(0, 0))
            font = QtGui.QFont()
            font.setPointSize(16)
            self.PassWord.setFont(font)
            self.PassWord.setObjectName("PassWord")
            self.formLayout.setWidget(2, QtWidgets.QFormLayout.FieldRole
    , self.PassWord)
            self.label_5 = QtWidgets.QLabel(self.formLayoutWidget)
            self.label_5.setSizeIncrement(QtCore.QSize(0, 0))
            self.label_5.setBaseSize(QtCore.QSize(0, 0))
            font = QtGui.QFont()
            font.setPointSize(16)
            self.label_5.setFont(font)
            self.label_5.setObjectName("label_5")
            self.formLayout.setWidget(3, QtWidgets.QFormLayout.LabelRole
    , self.label_5)
            self.TableName = QtWidgets.QLineEdit(self.formLayoutWidget)
            self.TableName.setSizeIncrement(QtCore.QSize(0, 0))
            self.TableName.setBaseSize(QtCore.QSize(0, 0))
            font = QtGui.QFont()
            font.setPointSize(16)
            self.TableName.setFont(font)
            self.TableName.setObjectName("TableName")
```

```
225             self.formLayout.setWidget(3, QtWidgets.QFormLayout.FieldRole
    , self.TableName)
226             self.titleConnect = QtWidgets.QLabel(DataControl)
227             self.titleConnect.setGeometry(QtCore.QRect(210, 20, 241, 51)
    )
228             font = QtGui.QFont()
229             font.setPointSize(20)
230             font.setBold(True)
231             font.setWeight(75)
232             self.titleConnect.setFont(font)
233             self.titleConnect.setObjectName("titleConnect")
234             self.savingButton = QtWidgets.QPushButton(DataControl)
235             self.savingButton.setGeometry(QtCore.QRect(190, 440, 241, 51
    ))
236             font = QtGui.QFont()
237             font.setPointSize(17)
238             self.savingButton.setFont(font)
239             self.savingButton.setObjectName("savingButton")
240             self.label_4 = QtWidgets.QLabel(DataControl)
241             self.label_4.setGeometry(QtCore.QRect(270, 380, 91, 17))
242             font = QtGui.QFont()
243             font.setPointSize(10)
244             font.setBold(True)
245             font.setWeight(75)
246             self.label_4.setFont(font)
247             self.label_4.setObjectName("label_4")
248 
249             self.retranslateUi(DataControl)
250             # self.savingButton.clicked.connect(DataControl.Saving)
251             QtCore.QMetaObject.connectSlotsByName(DataControl)
252 
253         def retranslateUi(self, DataControl):
254             _translate = QtCore.QCoreApplication.translate
255             DataControl.setWindowTitle(_translate("DataControl", "Form")
    )
256             self.label.setText(_translate("DataControl", "数据库名称"))
257             self.DataBaseName.setText(_translate("DataControl", "design"
    ))
258             self.label_2.setText(_translate("DataControl", "用户名"))
259             self.UserName.setText(_translate("DataControl", "root"))
260             self.label_3.setText(_translate("DataControl", "密码"))
261             self.PassWord.setText(_translate("DataControl", "123456"))
262             self.label_5.setText(_translate("DataControl", "表名"))
263             self.TableName.setText(_translate("DataControl", "result"))
```

```
264             self.titleConnect.setText(_translate("DataControl", "数据库连
    接设置"))
265             self.savingButton.setText(_translate("DataControl", "存储"))
266             self.label_4.setText(_translate("DataControl", "创建或补充"))
```

4）数据一览界面生成代码

```
267 from PyQt5 import QtCore, QtGui, QtWidgets
268 from PyQt5.QtWidgets import *
269 import database
270 
271 class Ui_DataView(object):
272     def setupUi(self, Form):
273         Form.setObjectName("Form")
274         Form.resize(940, 690)
275         self.title = QtWidgets.QLabel(Form)
276         self.title.setGeometry(QtCore.QRect(360, 20, 191, 61))
277         font = QtGui.QFont()
278         font.setPointSize(28)
279         font.setBold(True)
280         font.setWeight(75)
281         self.title.setFont(font)
282         self.title.setObjectName("title")
283         self.databaseWidget = QtWidgets.QTableWidget(Form)
284         self.databaseWidget.setGeometry(QtCore.QRect(30, 100, 881, 4
    31))
285         self.databaseWidget.setObjectName("databaseWidget")
286         self.databaseWidget.setColumnCount(0)
287         self.databaseWidget.setRowCount(0)
288         self.databaseWidget.setEditTriggers(QtWidgets.QAbstractItemV
    iew.NoEditTriggers)
289         self.pushButton = QtWidgets.QPushButton(Form)
290         self.pushButton.setGeometry(QtCore.QRect(350, 580, 221, 61))
291         font = QtGui.QFont()
292         font.setPointSize(15)
293         self.pushButton.setFont(font)
294         self.pushButton.setObjectName("pushButton")
295 
296         self.retranslateUi(Form)
297         QtCore.QMetaObject.connectSlotsByName(Form)
298 
299     def retranslateUi(self, Form):
300         _translate = QtCore.QCoreApplication.translate
301         Form.setWindowTitle(_translate("Form", "Form"))
302         self.title.setText(_translate("Form", "数据一览"))
```

```
303             self.pushButton.setText(_translate("Form", "进行操作"))
```

5）数据操作界面生成代码

```
304 from PyQt5 import QtCore, QtGui, QtWidgets
305 
306 class Manager(object):
307     def setupUi(self, Form):
308         Form.setObjectName("Form")
309         Form.resize(1124, 829)
310         self.label = QtWidgets.QLabel(Form)
311         self.label.setGeometry(QtCore.QRect(440, 20, 161, 61))
312         font = QtGui.QFont()
313         font.setPointSize(24)
314         font.setBold(True)
315         font.setWeight(75)
316         self.label.setFont(font)
317         self.label.setObjectName("label")
318         self.searchResult = QtWidgets.QTableWidget(Form)
319         self.searchResult.setGeometry(QtCore.QRect(25, 81, 1071, 351
    ))
320         self.searchResult.setObjectName("searchResult")
321         self.searchResult.setColumnCount(0)
322         self.searchResult.setRowCount(0)
323         self.searchResult.setEditTriggers(QtWidgets.QAbstractItemVie
    w.NoEditTriggers)
324         self.formLayoutWidget = QtWidgets.QWidget(Form)
325         self.formLayoutWidget.setGeometry(QtCore.QRect(30, 450, 511,
     211))
326         self.formLayoutWidget.setObjectName("formLayoutWidget")
327         self.formLayout = QtWidgets.QFormLayout(self.formLayoutWidge
    t)
328         self.formLayout.setLabelAlignment(QtCore.Qt.AlignRight|QtCor
    e.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
329         self.formLayout.setContentsMargins(0, 0, 0, 0)
330         self.formLayout.setHorizontalSpacing(40)
331         self.formLayout.setVerticalSpacing(28)
332         self.formLayout.setObjectName("formLayout")
333         self.label_2 = QtWidgets.QLabel(self.formLayoutWidget)
334         font = QtGui.QFont()
335         font.setPointSize(20)
336         self.label_2.setFont(font)
337         self.label_2.setObjectName("label_2")
338         self.formLayout.setWidget(0, QtWidgets.QFormLayout.LabelRole
    , self.label_2)
```

```
339             self.Title = QtWidgets.QLineEdit(self.formLayoutWidget)
340             font = QtGui.QFont()
341             font.setPointSize(20)
342             self.Title.setFont(font)
343             self.Title.setObjectName("Title")
344             self.formLayout.setWidget(0, QtWidgets.QFormLayout.FieldRole
    , self.Title)
345             self.label_3 = QtWidgets.QLabel(self.formLayoutWidget)
346             font = QtGui.QFont()
347             font.setPointSize(20)
348             self.label_3.setFont(font)
349             self.label_3.setObjectName("label_3")
350             self.formLayout.setWidget(1, QtWidgets.QFormLayout.LabelRole
    , self.label_3)
351             self.Author = QtWidgets.QLineEdit(self.formLayoutWidget)
352             font = QtGui.QFont()
353             font.setPointSize(20)
354             self.Author.setFont(font)
355             self.Author.setObjectName("Author")
356             self.formLayout.setWidget(1, QtWidgets.QFormLayout.FieldRole
    , self.Author)
357             self.label_4 = QtWidgets.QLabel(self.formLayoutWidget)
358             font = QtGui.QFont()
359             font.setPointSize(20)
360             self.label_4.setFont(font)
361             self.label_4.setObjectName("label_4")
362             self.formLayout.setWidget(2, QtWidgets.QFormLayout.LabelRole
    , self.label_4)
363             self.Source = QtWidgets.QLineEdit(self.formLayoutWidget)
364             font = QtGui.QFont()
365             font.setPointSize(20)
366             self.Source.setFont(font)
367             self.Source.setObjectName("Source")
368             self.formLayout.setWidget(2, QtWidgets.QFormLayout.FieldRole
    , self.Source)
369             self.formLayoutWidget_2 = QtWidgets.QWidget(Form)
370             self.formLayoutWidget_2.setGeometry(QtCore.QRect(560, 450, 5
    31, 211))
371             self.formLayoutWidget_2.setObjectName("formLayoutWidget_2")
372             self.formLayout_2 = QtWidgets.QFormLayout(self.formLayoutWid
    get_2)
373             self.formLayout_2.setLabelAlignment(QtCore.Qt.AlignRight|QtC
    ore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
```

```
374             self.formLayout_2.setFormAlignment(QtCore.Qt.AlignHCenter|Qt
    Core.Qt.AlignTop)
375             self.formLayout_2.setContentsMargins(0, 0, 0, 0)
376             self.formLayout_2.setHorizontalSpacing(40)
377             self.formLayout_2.setVerticalSpacing(28)
378             self.formLayout_2.setObjectName("formLayout_2")
379             self.label_5 = QtWidgets.QLabel(self.formLayoutWidget_2)
380             font = QtGui.QFont()
381             font.setPointSize(20)
382             self.label_5.setFont(font)
383             self.label_5.setObjectName("label_5")
384             self.formLayout_2.setWidget(0, QtWidgets.QFormLayout.LabelRo
    le, self.label_5)
385             self.Date = QtWidgets.QLineEdit(self.formLayoutWidget_2)
386             font = QtGui.QFont()
387             font.setPointSize(20)
388             self.Date.setFont(font)
389             self.Date.setObjectName("Date")
390             self.formLayout_2.setWidget(0, QtWidgets.QFormLayout.FieldRo
    le, self.Date)
391             self.label_6 = QtWidgets.QLabel(self.formLayoutWidget_2)
392             font = QtGui.QFont()
393             font.setPointSize(20)
394             self.label_6.setFont(font)
395             self.label_6.setObjectName("label_6")
396             self.formLayout_2.setWidget(1, QtWidgets.QFormLayout.LabelRo
    le, self.label_6)
397             self.Use = QtWidgets.QLineEdit(self.formLayoutWidget_2)
398             font = QtGui.QFont()
399             font.setPointSize(20)
400             self.Use.setFont(font)
401             self.Use.setObjectName("Use")
402             self.formLayout_2.setWidget(1, QtWidgets.QFormLayout.FieldRo
    le, self.Use)
403             self.label_7 = QtWidgets.QLabel(self.formLayoutWidget_2)
404             font = QtGui.QFont()
405             font.setPointSize(20)
406             self.label_7.setFont(font)
407             self.label_7.setObjectName("label_7")
408             self.formLayout_2.setWidget(2, QtWidgets.QFormLayout.LabelRo
    le, self.label_7)
409             self.Download = QtWidgets.QLineEdit(self.formLayoutWidget_2)
410             font = QtGui.QFont()
411             font.setPointSize(20)
```

```
412             self.Download.setFont(font)
413             self.Download.setObjectName("Download")
414             self.formLayout_2.setWidget(2, QtWidgets.QFormLayout.FieldRo
    le, self.Download)
415             self.widget = QtWidgets.QWidget(Form)
416             self.widget.setGeometry(QtCore.QRect(30, 680, 1061, 111))
417             self.widget.setObjectName("widget")
418             self.horizontalLayout = QtWidgets.QHBoxLayout(self.widget)
419             self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
420             self.horizontalLayout.setSpacing(32)
421             self.horizontalLayout.setObjectName("horizontalLayout")
422             self.Search = QtWidgets.QPushButton(self.widget)
423             self.Search.setSizeIncrement(QtCore.QSize(0, 0))
424             font = QtGui.QFont()
425             font.setPointSize(18)
426             self.Search.setFont(font)
427             self.Search.setObjectName("Search")
428             self.horizontalLayout.addWidget(self.Search)
429             self.Add = QtWidgets.QPushButton(self.widget)
430             self.Add.setSizeIncrement(QtCore.QSize(0, 0))
431             font = QtGui.QFont()
432             font.setPointSize(18)
433             self.Add.setFont(font)
434             self.Add.setObjectName("Add")
435             self.horizontalLayout.addWidget(self.Add)
436             self.Delete = QtWidgets.QPushButton(self.widget)
437             self.Delete.setSizeIncrement(QtCore.QSize(0, 0))
438             font = QtGui.QFont()
439             font.setPointSize(18)
440             self.Delete.setFont(font)
441             self.Delete.setObjectName("Delete")
442             self.horizontalLayout.addWidget(self.Delete)
443             self.revise = QtWidgets.QPushButton(self.widget)
444             self.revise.setSizeIncrement(QtCore.QSize(0, 0))
445             font = QtGui.QFont()
446             font.setPointSize(18)
447             self.revise.setFont(font)
448             self.revise.setObjectName("revise")
449             self.horizontalLayout.addWidget(self.revise)
450             self.label_2.setBuddy(self.Title)
451             self.label_3.setBuddy(self.Author)
452             self.label_4.setBuddy(self.Source)
453             self.label_5.setBuddy(self.Title)
454             self.label_6.setBuddy(self.Author)
```

```
            self.label_7.setBuddy(self.Source)

            self.retranslateUi(Form)
            QtCore.QMetaObject.connectSlotsByName(Form)

        def retranslateUi(self, Form):
            _translate = QtCore.QCoreApplication.translate
            Form.setWindowTitle(_translate("Form", "Form"))
            self.label.setText(_translate("Form", "管理数据"))
            self.label_2.setText(_translate("Form", "标题"))
            self.label_3.setText(_translate("Form", "作者"))
            self.label_4.setText(_translate("Form", "来源"))
            self.label_5.setText(_translate("Form", "日期"))
            self.label_6.setText(_translate("Form", "被引数"))
            self.label_7.setText(_translate("Form", "下载数"))
            self.Search.setText(_translate("Form", "查询"))
            self.Add.setText(_translate("Form", "增加"))
            self.Delete.setText(_translate("Form", "删除"))
            self.revise.setText(_translate("Form", "全览"))
```

6）爬虫与解析代码

```
import requests
from urllib import parse
from lxml import html
import pandas as pd

headers={'Host':'kns.cnki.net',
'Connection':'keep-alive',
'Content-Length':'770',
'sec-ch-ua':'" Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"',
'Accept':'text/html, */*; q=0.01',
'Content-Type':'application/x-www-form-urlencoded; charset=UTF-8',
'X-Requested-With':'XMLHttpRequest',
'sec-ch-ua-mobile':'?0',
'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36',
'sec-ch-ua-platform':'"Windows"',
'Origin':'https://kns.cnki.net',
'Sec-Fetch-Site':'same-origin',
'Sec-Fetch-Mode':'cors',
```

```
    'Sec-Fetch-Dest':'empty',
    'Referer':'https://kns.cnki.net/kns8/defaultresult/index',
    'Accept-Encoding':'gzip, deflate, br',
    'Accept-Language':'zh-CN,zh;q=0.9',
    'Cookie':'Ecp_ClientId=7211124224501223118; Ecp_loginuserbk=K10349;
    cnkiUserKey=8afb3c2f-e918-884f-b967-
    ad9b28a4f36e; sensorsdata2015jssdkcross=%7B%22distinct_id%22%3A%2217
    d5268713e40-075435a87d4bc8-978183a-1327104-
    17d5268713f7d%22%2C%22first_id%22%3A%22%22%2C%22props%22%3A%7B%7D%2C
    %22%24device_id%22%3A%2217d5268713e40-075435a87d4bc8-978183a-
    1327104-
    17d5268713f7d%22%7D; Ecp_loginuserjf=15991277627; knsLeftGroupSelect
    Item=1%3B2%3B; Ecp_ClientIp=219.144.235.53; Ecp_Userid=1089517534; c
    nkiUserKey=d60c6fe6-ba08-b0e6-0993-
    2199710c7a25; dsorder=cite; RsPerPage=20; UM_distinctid=1816fceacd33
    -033b740c9a121c-26021b51-144000-
    1816fceacd4bf; Hm_lvt_6e967eb120601ea41b9d312166416aa6=1655474920,16
    55479057,1655526956; SID_sug=126002; _pk_ref=%5B%22%22%2C%22%22%2C16
    56385931%2C%22https%3A%2F%2Fwww.baidu.com%2Flink%3Furl%3DqWZl4gYatn5
    WAOIF-
    uSx4Xf96UP9RsPmIgGWR9rF0KG%26wd%3D%26eqid%3De9b4e07d0000082200000006
    62ba7186%22%5D; _pk_ses=*; LID=WEEvREcwSlJHSldSdmVqMDh6cEFITnRlU3J2T
    VNpK3RlYnE0dlJzYXNRTT0=$9A4hF_YAuvQ5obgVAqNKPCYcEjKensW4IQMovwHtwkF4
    VYPoHbKxJw!!; Ecp_session=1; Ecp_LoginStuts={"IsAutoLogin":false,"Us
    erName":"K10349","ShowName":"%E8%A5%BF%E5%AE%89%E7%90%86%E5%B7%A5%E5
    %A4%A7%E5%AD%A6%E5%9B%BE%E4%B9%A6%E9%A6%86","UserType":"bk","BUserNa
    me":"","BShowName":"","BUserType":"","r":"krDVMO"}; c_m_LinID=LinID=
    WEEvREcwSlJHSldSdmVqMDh6cEFITnRlU3J2TVNpK3RlYnE0dlJzYXNRTT0=$9A4hF_Y
    AuvQ5obgVAqNKPCYcEjKensW4IQMovwHtwkF4VYPoHbKxJw!!&ot=06/28/2022 11:3
    2:11; c_m_expire=2022-06-
    28%2011%3A32%3A11; ASP.NET_SessionId=h4o0zmd40oo0uarhqnptpzgv; SID_k
    ns8=25123164; dblang=ch; _pk_id=04a88e1a-fa4c-49df-b450-
    f5eea7c25150.1637765109.13.1656385950.1656385931.',
    }

    url='https://kns.cnki.net/kns8/Brief/GetGridTableHtml'

    def getOnePage(getMethod,searchItem,page):

        dic1={"Title":"搜索方式","Name":"方式","Value":"搜索内容
    ","Operate":"=","BlurType":""}
        dic1['Value']=searchItem
        if getMethod==0:
            dic1['Title']='作者'
```

```
            dic1['Name']='AU'
        else:
            dic1['Title']='主题'
            dic1['Name']='SU'
            dic1['Operate']="%="

        dic2={"Key":"Subject","Title":"","Logic":1,"Items":[dic1],"Child
    Items":[]}
        dic3={"QGroup":[dic2]}
        dic4={"Platform":"","DBCode":"CFLS","KuaKuCode":"CJFQ,CDMD,CIPD,
    CCND,CISD,SNAD,BDZK,CCJD,CCVD,CJFN","QNode":dic3}
        dic={'IsSearch':'false',
        'QueryJson':dic4,
        'PageName': 'defaultresult',
        'DBCode': 'CFLS',
        'KuaKuCodes': 'CJFQ,CDMD,CIPD,CCND,CISD,SNAD,BDZK,CCJD,CCVD,CJFN
    ',
        'CurPage': 1,
        'RecordsCntPerPage': 20,
        'CurDisplayMode': 'listmode',
        'CurrSortField': 'CITY',
        'CurrSortFieldType': 'desc',
        'IsSentenceSearch': False,
        'Subject': ''}
        dic['CurPage']=str(page)
        dic = parse.urlencode(dic)

        req=requests.post(headers=headers,url=url,data=dic)
        # print(req.text)

        tree=html.etree
        tree = tree.HTML(req.text)

        titles=[]
        writers=[]
        sources=[]
        dates=[]
        uses=[]
        downloads=[]
        urls=[]
        for i in range(1,21):
            detailUrl="https://kns.cnki.net/kcms/detail/detail.aspx?"
            i=str(i)
            #group titles
```

```
548             temptile=tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[2]/a//descendant::text()')
549             tempurl=tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[2]/a/@href')
550             tempurl="".join(tempurl)
551             tempurl=tempurl.split("&")
552             filename_url=tempurl[4][9:]
553             dbname_url=tempurl[5][7:]
554 
555             titles.append("".join(temptile))
556 
557             #group authors
558             tempwriter=",".join(tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[3]/a//descendant::text()'))
559             if tempwriter=='':
560                 tempwriter=",".join(tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[3]//descendant::text()'))
561                 tempwriter=tempwriter.replace("\n","").replace("\r","").replace("\t","").replace(" ","")
562             tempwriter=tempwriter.strip(",")
563             writers.append(tempwriter)
564 
565             #group sources
566             tempsource=str(tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[4]/a/text()')[0])
567             sources.append(tempsource)
568 
569             #group dates
570             tempdate=tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[5]/text()')
571             tempdate="".join(tempdate)
572             dates.append(tempdate.strip())
573 
574             #group uses
575             tempuse=tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[7]/span/a/text()')
576             if not tempuse:
577                 tempuse=['0']
578             tempuse="".join(tempuse)
579             uses.append(tempuse)
580 
581             #group downloads
582             tempdown=tree.xpath('//*[@id="gridTable"]/table/tr['+ i +']/td[8]/a/text()')
```

```
            if not tempdown:
                tempdown=['0']
            downloads.append("".join(tempdown))

            #group urls
            detailUrl2=f"dbcode=CJFD&dbname={dbname_url}&filename={filename_url}&uniplatform=NZKPT&v=yVaiR-TqK3zCk5lBU-Dk57HxgTsFdwRkuI4MGemxETkhOnQw1U9mRM2Fv76YzAkR"
            detailUrl+=detailUrl2
            urls.append(detailUrl)

        res={'标题':titles,'作者':writers,'来源':sources,'日期':dates,'被引数':uses,'下载数':downloads,'详情页 url':urls}
        df=pd.DataFrame(res)
        return df
```

7）数据库操作代码

```
from PyQt5.QtSql import QSqlQuery, QSqlDatabase
from PyQt5.QtWidgets import *
import pandas as pd
from sqlalchemy import create_engine

def toDatabase(parent:QWidget,databaseName:str,userName:str,passWord:str,tableName:str,df:pd.DataFrame):
    try:
        engine = create_engine(f"mysql+pymysql://{userName}:{passWord}@localhost/{databaseName}",encoding='utf8')
        df.to_sql(tableName, engine, index=False,if_exists='append')
        return True
    except:
        throwErrors(parent,4)
        return False

def runSQL(parent:QWidget,databaseName:str,userName:str,passWord:str,sql:str):
    try:
        engine=create_engine(f"mysql+pymysql://{userName}:{passWord}@localhost/{databaseName}",encoding='utf8')
        conn=engine.connect()
        conn.execute(sql)
    except:
        throwErrors(parent,4)
        return False
```

```
617         return True
618
619     def searchDataBase(parent:QWidget,databaseName:str,userName:str,pass
    Word:str,sql:str):
620         df=pd.DataFrame()
621         try:
622             engine=create_engine(f"mysql+pymysql://{userName}:{passWord}
    @localhost/{databaseName}",encoding='utf8')
623             df=pd.read_sql(sql,engine)
624         except:
625             throwErrors(parent,4)
626             return df,False
627         return df,True
628
629     def readDatabase(parent:QWidget,databaseName:str,userName:str,passWo
    rd:str,tableName:str):
630         df=pd.DataFrame()
631         try:
632             engine=create_engine(f"mysql+pymysql://{userName}:{passWord}
    @localhost/{databaseName}",encoding='utf8')
633             sql=f"select * from {tableName}"
634             df=pd.read_sql(sql,engine)
635         except:
636             throwErrors(parent,4)
637             return df,False
638
639         return df,True
640
641     def throwErrors(parent:QWidget,flag:int):
642         errorDict={0:"数据库连接失败！检查 ODBC 配置和用户名与密码",
643                    1:"找不到已有的表！将会在默认数据库下新建...",
644                    2:"创建新表失败！",
645                    4:"数据库操作失败，确认 engine 连接",
646                    5:"请输入全部信息以插入！"}
647         QMessageBox.critical(parent,"Error",errorDict[flag],QMessageBox.
    Ok)
```

8）预览功能代码

```
648     from selenium import webdriver
649     def browserGet(item,Options):
650         Select={0:f"dbcode=SCDB&kw={item}&korder=AU",1:f"dbcode=SCDB&kw=
    {item}&korder=SU"}
651         url1=Select[Options]
```

```
652         browser=webdriver.Chrome()
653         browser.get("https://kns.cnki.net/kns8/defaultresult/index?"+url
    1)
```

9）主界面代码

```
654     from DataManager import Manager
655     from Main import Ui_Main
656     from Getting import Ui_Getting
657     from DataControl import Ui_DataControl
658     from DataView import Ui_DataView
659 
660     import test
661     import sele
662     import database
663 
664     import pandas as pd
665     from PyQt5 import QtCore, QtGui, QtWidgets
666     from PyQt5.QtWidgets import *
667 
668 
669     All_basename=0
670     All_user=0
671     All_password=0
672     All_table=0
673     All_Is_Option=False
674 
675 
676     class MainWindow(QWidget):
677         def __init__(self):
678             super().__init__()
679             self.ui=Ui_Main()
680             self.ui.setupUi(self)
681             self.ui.search_Botton.clicked.connect(self.sele)
682         def goToDataPage(self):
683             form4.show()
684             form4.reading()
685         def goToGettingPage(self):
686             if not self.ui.inputLine.text():
687                 QMessageBox.information(self,"error","请输入要爬取的内容！
    ",QMessageBox.Ok)
688             else:
689                 form2.show()
690         def sele(self):
```

```
691             Option=self.ui.comboBox.currentIndex()
692             item=self.ui.inputLine.text()
693             sele.browserGet(item,Option)
694 
695     class GettingWindow(QWidget):
696         startPage=0
697         endPage=0
698         getMethod=-1
699         searchItem=''
700         def __init__(self):
701             super().__init__()
702             self.ui=Ui_Getting()
703             self.ui.setupUi(self)
704             self.ui.Start.clicked.connect(self.Start)
705 
706         def Start(self):
707             GettingWindow.startPage=self.ui.page_Start.value()
708             GettingWindow.endPage=self.ui.page_End.value()
709             GettingWindow.getMethod=form1.ui.comboBox.currentIndex()
710             GettingWindow.searchItem=form1.ui.inputLine.text()
711             if GettingWindow.startPage>GettingWindow.endPage:
712                 QMessageBox.critical(self,"error","请正确设置起始和终止页
    数！",QMessageBox.Ok)
713                 return
714 
715             df=pd.DataFrame()
716             if self.ui.storeMethod.currentIndex()==1:
717                     df=pd.DataFrame()
718                     df,bool=GettingWindow.GettingPages(self,self,Getting
    Window.startPage,GettingWindow.endPage,GettingWindow.searchItem,Gett
    ingWindow.getMethod)
719                     bool=database.toDatabase(self,All_basename,All_user,
    All_password,All_table,df)
720                     if bool:
721                         QMessageBox.information(self,"成功","已将爬取的数
    据保存！",QMessageBox.Ok)
722                         return
723                     else:
724                         QMessageBox.critical(self,"失败","保存失败
    ",QMessageBox.Ok)
725                         return
726             else:
```

```
727                 df,bool=self.GettingPages(self,GettingWindow.startPage,GettingWindow.endPage,GettingWindow.searchItem,GettingWindow.getMethod)
728                 if bool:
729                     saveOrNot=QMessageBox.question(self,"爬取成功","已经成功爬取" + str((GettingWindow.endPage-GettingWindow.startPage+1)*20) + "条数据！是否将其存储？",QMessageBox.Save|QMessageBox.Cancel,QMessageBox.Save)
730                     if saveOrNot==QMessageBox.Save:
731                         self.saveToCsv(self,df)
732 
733         def GettingPages(self,parent:QWidget,start,end,searchItem,getMethod):
734             # print(start,end,getMethod,searchItem)
735             df=pd.DataFrame()
736             try:
737                 for i in range(start,end+1):
738                     df=pd.concat([df,test.getOnePage(getMethod,searchItem,i)],ignore_index=True)
739             except:
740                 QMessageBox.critical(parent,"error","失败！请检查爬取选项",QMessageBox.Ok)
741                 return df,False
742             return df,True
743 
744 
745         def saveToCsv(self,parent:QWidget,df:pd.DataFrame):
746             try:
747                 df.to_csv('result.csv',index=False,encoding='utf_8_sig')
748             except:
749                 QMessageBox.critical(parent,"失败","保存失败",QMessageBox.Ok)
750                 return
751             QMessageBox.information(parent,"成功","保存成功！",QMessageBox.Ok)
752             parent.close()
753 
754 
755 class DataOptionWindow(QWidget):
756     def __init__(self):
757         super().__init__()
758         self.ui=Ui_DataControl()
759         self.ui.setupUi(self)
760         self.ui.savingButton.clicked.connect(self.Saving)
```

```
761         def Saving(self):
762             global All_basename,All_user,All_password,All_table,All_Is_O
    ption
763             basename=self.ui.DataBaseName.text()
764             user=self.ui.UserName.text()
765             password=self.ui.PassWord.text()
766             table=self.ui.TableName.text()
767             pack=(basename,user,password,table)
768             if '' in pack:
769                 QMessageBox.information(self,"Caution","请填写完整数据
    ",QMessageBox.Ok)
770                 return
771             else:
772                 All_basename,All_user,All_password,All_table=pack
773                 All_Is_Option=True
774                 self.close()
775                 return
776 
777 
778     class DataViewWindow(QWidget):
779         def __init__(self):
780             super().__init__()
781             self.ui=Ui_DataView()
782             self.ui.setupUi(self)
783             self.ui.pushButton.clicked.connect(self.controlPage)
784 
785         def reading(self):
786             if All_Is_Option:
787                 df,bool=database.readDatabase(self,All_basename,All_user
    ,All_password,All_table)
788             else:
789                 return
790             if not bool:
791                 database.throwErrors(self,0)
792                 self.close()
793             self.display_dynamic_form(df,self.ui.databaseWidget)
794 
795 
796         def controlPage(self):
797             form5.show()
798 
799         def display_dynamic_form(self, df, target_obj):
800 
801             # horizontalHeader().setVisible
```

```
            # .verticalHeader().setVisible
            input_table_rows = df.shape[0]
            input_table_colunms = df.shape[1]
            input_table_header = df.columns.values.tolist()
            target_obj.setColumnCount(input_table_colunms)
            target_obj.setRowCount(input_table_rows)
            target_obj.setHorizontalHeaderLabels(input_table_header)
            # print(input_table_header)
            for i in range(input_table_rows):
                for j in range(input_table_colunms):
                    new_item = QTableWidgetItem(str(df.iat[i, j]))
                    target_obj.setItem(i, j, new_item)


class ManagerWindow(QWidget):
    def __init__(self):
            super().__init__()
            self.ui=Manager()
            self.ui.setupUi(self)
            self.ui.Search.clicked.connect(self.Search)
            self.ui.Add.clicked.connect(self.Add)
            self.ui.Delete.clicked.connect(self.Delete)
            self.ui.revise.clicked.connect(self.revise)
    def Search(self):
            df=pd.DataFrame()
            q=self.buildWhereQuery(self.getContents())
            query=f"select * from {All_table} where {q}"
            df,bool=database.searchDataBase(self,All_basename,All_user,All_password,query)
            if not bool:
                return
            DataViewWindow.display_dynamic_form(self,df,self.ui.searchResult)
    def Add(self):
            contents=self.getContents()
            if '' in contents:
                database.throwErrors(self,5)
                return
            title,author,source,date,use,download=contents
            dic={'标题':[title],'作者':[author],'来源':[source],'日期':[date],'被引数':[use],'下载数':[download]}
            df=pd.DataFrame(dic)
            # query=f"insert into test values('{title}','{author}','{source}','{date}','{use}','{download}')"
```

```
842             bool=database.toDatabase(self,All_basename,All_user,All_pass
    word,All_table,df)
843             if not bool:
844                 return
845             else:
846                 QMessageBox.information(self,"成功","成功添加数据!
    ",QMessageBox.Ok)
847         def Delete(self):
848             q=self.buildWhereQuery(self.getContents())
849             delOrNot=QMessageBox.question(self,"警告","确定要删除吗?
    " ,QMessageBox.Yes|QMessageBox.No,QMessageBox.No)
850             if delOrNot==QMessageBox.Yes:
851                 query=f"delete from {All_table} where {q}"
852                 print(query)
853                 bool=database.runSQL(self,All_basename,All_user,All_pass
    word,query)
854                 if not bool:
855                     return
856                 else:
857                     QMessageBox.information(self,"成功","成功删除数据!
    ",QMessageBox.Ok)
858         def revise(self):
859             if All_Is_Option:
860                 df,bool=database.readDatabase(self,All_basename,All_user
    ,All_password,All_table)
861             else:
862                 return
863             if not bool:
864                 database.throwErrors(self,0)
865                 self.close()
866             DataViewWindow.display_dynamic_form(self,df,self.ui.searchRe
    sult)
867
868
869         def buildWhereQuery(self,contents:tuple):
870             title,author,source,date,use,download=contents
871             query=f"{self.buildPartQuery('标题
    ',title)}and{self.buildPartQuery('作者
    ',author)}and{self.buildPartQuery('来源
    ',source)}and{self.buildPartQuery('日期
    ',date,False)}and{self.buildPartQuery('被引数
    ',use,False)}and{self.buildPartQuery('下载数',download,False)}"
872             query=query.split("and")
873             while '' in query:
```

```
874                 query.remove('')
875             query=" and ".join(query)
876             return query
877 
878 
879         def buildPartQuery(self,type:str,content:str,isLike:bool=True):
880             if content=='':
881                 return ''
882             if isLike:
883                 queryStr=f"{type} like '%{content}%'"
884             else:
885                 queryStr=f"{type} = '{content}'"
886             return queryStr
887         def getContents(self):
888             pack=[]
889             pack.append(self.ui.Title.text())
890             pack.append(self.ui.Author.text())
891             pack.append(self.ui.Source.text())
892             pack.append(self.ui.Date.text())
893             pack.append(self.ui.Use.text())
894             pack.append(self.ui.Download.text())
895             pack=tuple(pack)
896             return pack
897 
898 
899     app=QApplication([])
900     form1=MainWindow()
901     form2=GettingWindow()
902     form3=DataOptionWindow()
903     form4=DataViewWindow()
904     form5=ManagerWindow()
905 
906     form1.show()
907     form3.show()
908 
909     app.exec_()
```