CSDN 搜索引擎

详细设计说明书

作者：吴宇涛 李德宏

2021 年 12 月 1 日

華南師範大學
SOUTH CHINA NORMAL UNIVERSITY

目录

文档名称： 详细设计规格说明书

项目名称： CSDN 搜索引擎

项目负责人：吴宇涛 李德宏

编写 TEST _2021____年__12___月__1___日

校对 所有小组成员 _2021____年__12___月__11___日

审核 所有小组成员 _2021____年__12___月__11___日

批准 潘明老师 _2021____年__12___月__16___日

开发单位_____华南师范大学计算机学院_____

组员： 吴宇涛 李德宏

1 引言

1.1 编写目的

搜索引擎是指根据一定的策略、运用特定的计算机程序从互联网上采集信息，在对信息进行组织和处理后，为用户提供检索服务，将检索的相关信息展示给用户的系统。

网络中用于各种功能的搜索引擎非常多，对于计算机专业的同学，寻找博客自学，代码题解等方面，国内的博客园、CSDN比较优秀。但数据都过于集中，缺少重点。且界面不是很纯净，会有一些广告。对于计算机专业的大学生，要花费大量的时间寻找质量不一的博文，无疑增加了学生们的工作量。这次软件工程实验项目，我们想要实现的就是整合CSDN网站的有益信息，搭建起一个更为方便、便利的搜索引擎，让同学们在学习的时候能够获得更多的帮助。

1.2 项目背景

搜索引擎技术是互联网上面用的最普遍之一的技术。主要是以用户为中心。当客户输入查询的请求时候，同一个查询的请求关键词在用户的背后可能是不同查询要求。例如用户输入的是“苹果”，那么作为一个想要购买iPhone的用户和一个果农来说，那么要求就是大大

的不一样。甚至是同一个用户，所查询的关键词一样，也会因为所在的时间和所在的场合不同而返回的结果不同的所有主流搜索引擎，都在致力于解决同一个问题：怎样才能从用户所输入的一个简短的关键词来判断用户的真正查询请求。当代搜索引擎主要是以用户为中心。

1.3 定义

- **Django：**Django是一个由Python写成的Web应用框架。Django的主要目的是简便，快速的开发数据库驱动的网站。它强调代码复用，多个组件可以很方便的以“插件”形式服务于整个框架，Django有许多功能强大的第三方插件。

- **HTML：**HTML是超文本标记语言（Hyper Text Markup Language),HTML不是一种编程语言，而是一种标记语言。

- **Web crawler：**网络爬虫（又称为网页蜘蛛，网络机器人，在FOAF社区中间，更经常的称为网页追逐者），是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。

- **MySQL：**MySQL 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。

1.4 参考资料

[1] https://www.runoob.com/mysql/mysql-tutorial.html

[2] https://www.runoob.com/django/django-tutorial.html

[3] Python 测试驱动开发：使用 Django、Selenium 和 JavaScript 进行 Web 编程.第 2 版

[4] Python 网络爬虫权威指南（第 2 版） by 米切尔

2 总体设计

2.1 需求概述

按照需求分析文档中的规格要求，使用条形码扫描器进书、借书、还书，使得信息传递准确、流畅。同时，系统最大限度地实现易维护性，易操作性，运行稳定，安全可靠。

（1）易用性

要求操作简单，快捷，功能分类清晰并能最大限度满足需要，避免复杂的选择，简化数据的输入。

（2）可靠性

系统运行稳定可靠，考虑系统在平时和峰值的情况下，安全可靠地运行并记录数据，确保不死机，确保不丢失数据。

（3）可扩展性

在设计中不仅考虑当前的业务需求，更应该满足未来业务量和需求的增长。

软件应具备逐步升级的能里，采用模块化设计，能添加新功能以满足需求，或对各部分的功能灵活地进行升级，扩展。

（4）可管理性

软件应满足提供良好的应用操作维护界面，维护操作简单。管理员对数据库的运行进行监控以及维护。

（5） 灵活性

软件的设计和实现考虑到运行环境的变化，能够在运行环境变化的情况下正常使用。同时，软件兼容其他软件接口的变化，保证在不同运行环境，不同软件接口的情况下的正常使用。具体要求如下：

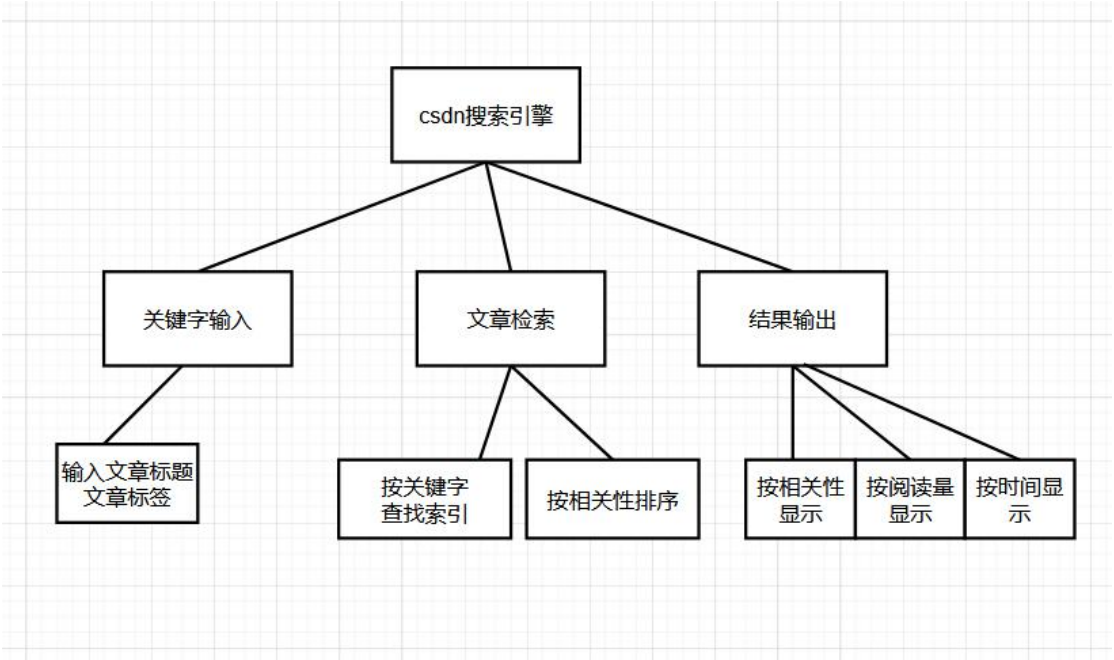
运行环境的变化：软件支持在 Windows Server 2003/Windows XP SP3 及以上 Windows 系统部署运行。

同其他软件接口的变化：当其他软件的接口发生变化时，该软件应能够适应接口的变化。

精度和有效时限的变化：软件能够方便的适应精度和有效时限的变化。

计划的变化或改进：软件具有足够的灵活性，可以支持将来有可能会出现的需求更改或增加。

2.2 软件结构



序号	名称
1，	登陆模块
2，	爬虫模块
3，	搜索模块
4，	排序模块
5，	热词搜索模块

3 程序描述

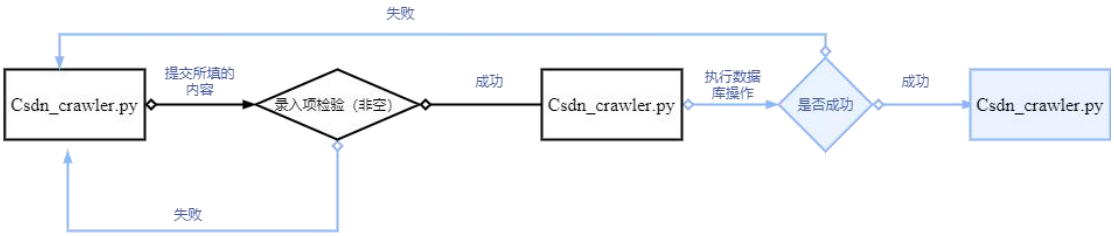
3.1 登陆模块

具体格式见下表：

功能编号	01	功能名称	登陆模块	内容	功能流程图
所属业务	CSDN 搜索引擎	模拟登录	所属项目	CSDN 博客	
编写人	吴宇涛	完成时间	2021-12-8	页码	第 7 页

3.1.1 功能流程图

功能流程图如下图所示。



需要说明的问题：

（1） 录入项检测使用 javascript 实现（各项必须非空）

（2） Csdn_crawler.py 页面也包含查询按钮，在此的 Csdn_crawler.py 提交的数据只是用户名和密码。

3.1.2 功能描述

（1） 功能类型：自动登录

（2） 功能描述：提高系统的安全性和效率

（3） 前提业务：无

（4） 后继业务：02 （管理模块）

（5） 功能约束：权限约束

（6） 约束描述：

（7） 操作权限：网站管理员

3.1.3 界面设计

（1） 基础信息处理

动作说明：

动作编号	动作名称	动作描述
A01	注册	点击注册按钮 提交数据到 csdn_crawler.py 页面
A02	登录	点击登录按钮 提交数据到 csdn_crawler.py 页面

（2） 数据要求

(1) 功能类型：数据查询

(2) 数据描述：

页面显示录入字段如下：

字段名称	长度	录入方式	是否非空项	数据检验	默认显示
管理员 ID	10	文本框	Y	N	
管理员密码	15	password	Y	N	

3.1.4 csdn_crawler.py 的内部逻辑

关键点两点：1，浏览器驱动配置；2，记录登陆信息及信息处理；

1，浏览器驱动配置

```
1.  url = "https://passport.csdn.net/account/login"
2.  driver = webdriver.Chrome(executable_path='C:/Users/PC_SKY_WYT/AppData/Local/Programs/Python/Python37/chromedriver')
3.          # 要加 chromedriver 绝对路径 or 把 chromedriver 加到系统 PATH 里
4.  driver.get(url)
```

```
    switch = driver.find_element_by_xpath(
        '//a[@class="login-code__open js_login_trigger login-user__active"]')
    if switch.text == '账号登录':
        switch.click()
        time.sleep(1)

```

2，记录登陆信息及信息处理

找到对应的控件按钮，自动输入账户和密码登录 CSDN 账号

```
    username = driver.find_element_by_id('username')
    password = driver.find_element_by_id('password')
    username.send_keys("username")
    password.send_keys("password")
    click = driver.find_element_by_class_name("logging")
    click.click()
    time.sleep(1)

    cookies = driver.get_cookies()

    # driver.page_source ->get html
    # 将 selenium 形式的 cookies 转换为 requests 可用的 cookies!!
    for cookie in cookies:
        session.cookies.set(cookie['name'], cookie['value'])

    return cookies
```

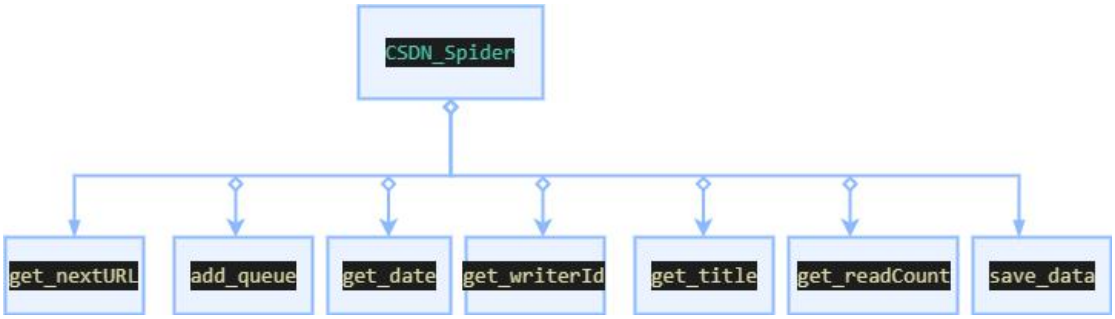
3.2 爬虫模块

具体格式见下表

功能编号	02	功能名称	管理模块	内容	功能流程图
所属业务	图书馆管理		所属项目	图书馆管理系统	
编写人	吴宇涛	完成时间	2021-12-08	页码	第 9 页

3.2.1 功能流程图

功能流程图如下所示：



需要说明的问题：

在爬取网页的数据时，网页的数据是不定时更新的，所以如果出现一些小错误，则需要重新获取网页的 xpath 等数据。

3.2.2 功能描述

（1） 功能类型：获取数据

（2） 功能概述：取出数据库文章链接，每一个链接爬取获取文章内容

（3） 前提业务：登陆模块（01）

（4） 后续业务：03

（5） 功能约束：权限约束

（6） 约束描述：

（7） 操作权限：网页管理人员

3.2.3 界面设计

（1） 基本信息处理

动作说明如下：

动作编号	动作名称	动作描述
A01	读取数据	从数据库中取出文章 url
A02	爬取数据	以文章链接为爬虫 url 发

		送请求获取文章源码
A03	解析数据	文章源码，进行解析，保存到数据库

（2） 数据要求

功能类型：其他

3.2.4 csdn_crawler.py 的内部逻辑

取出数据库文章链接，每一个链接爬取获取文章内容。输入文章链接，以文章链接为爬虫url发送请求获取文章源码，进行解析，保存到数据库。文章解析后的信息，保存到数据库

```
class CSDN_Spider(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
          connection = pymysql.connect(host='127.0.0.1',user='root',password='Wyt666!',database='csdn_crawler',charset='utf8mb4')
        cur = connection.cursor()
        cur.execute("USE csdn_crawler")
        self.conn = connection
        self.cur = cur
        self.lock = threading.Lock()

    def get_nextURL(self):
        # self.lock.acquire()
        self.cur.execute("select url from url_queue")
        url = self.cur.fetchone()[0]
        return url

    def remove(self,url):
        # print('remove' + url)
        self.cur.execute("Delete from url_queue where url= %s",url)
        self.cur.execute("Insert into visited values (%s)",url)
        # print('remove ' + url)
        self.cur.connection.commit()

```

```
25.     def add_queue(self,url):
26.
27.         self.cur.execute("Select url from visited where url= %s",url)
28.         t = self.cur.fetchall()
29.         self.cur.connection.commit()
30.         self.cur.execute("Select url from url_queue where url= %s",url)
31.         n = self.cur.fetchall()
32.         self.cur.connection.commit()
33.         # print(t,len(t))
34.
35.         if(len(t) == 0 and len(n) == 0):
36.             # print('insert' + url)
37.             self.cur.execute("Insert into url_queue VALUES (%s)",url)
38.             self.cur.connection.commit()
39.
40.
41.     def save_data(self,data):
42.         url = data['url']
43.         title = data['title']
44.         writer = data['writer']
45.         writer_id = data['writer_id']
46.         read_count = data['read_count']
47.         date = data['date']
48.         content = 'python'
49.
50.         self.cur.execute("Insert into crawler_csdnblog(url, title, writer, " +
51.                          "read_count, content, date, writer_id) Values (%s,%s,%s,%s,%s,%s,%s)",
52.                          (url,title,writer,read_count,content,date,writer_id))
53.         self.cur.connection.commit()
54.
55.     def run(self):
56.         while(True):
57.             try:
58.                 self.lock.acquire()
59.                 url = self.get_nextURL()
60.                 page = Page(url)
61.                 print('正在抓取' + url)
62.                 data = page.get_data()
63.                 self.save_data(data)
64.                 self.remove(url)
65.                 urls = page.get_blogURLs()
66.                 # print(urls)
```

```
67.                 for url in urls:
68.                     self.add_queue(url)
69.
70.             except Exception as e:
71.                 if hasattr(e, 'reason'):
72.                     print('reason:', e.reason)
73.                 elif hasattr(e, 'code'):
74.                     print('error code:', e.code)
75.                 else:
76.                     print(e)
77.                 self.remove(self.get_nextURL())  # 暴力删除,可能会一直删完 queue.
78.                 continue
79.             finally:
80.                 self.lock.release()
```

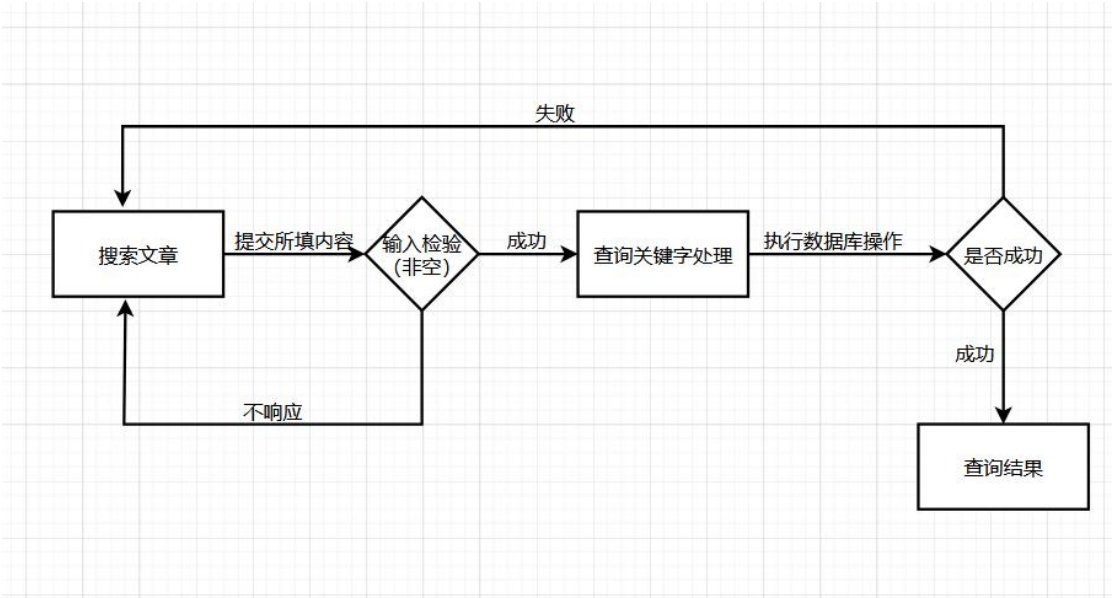
3.3 搜索模块

具体格式见下表：

功能编号	03	功能名称	搜索模块	内容	功能流程图
所属业务	CSDN 搜索引擎	搜索	所属项目	CSDN 博客	
编写人	李德宏	完成时间	2021-12-9	页码	第 15 页

3.3.1 功能流程图

功能流程图如下图：



需要说明的问题：

操作权限：面向所有用户

3.3.2 功能描述

（1） 功能类型：查询数据

（2） 功能概述：显示查询结果

（3） 前提业务：无

（4） 后继业务：

（5） 功能约束：没有约束；

（6） 约束描述：

（7） 操作权限：面向所有用户

3.3.3 界面设计

（1） 基础信息处理

下表是动作说明：

动作编号	动作名称	动作描述
A01	确定	点击按钮 提交数据到 searcher.py，执行索引查

		询

（2） 图书信息查询的输出项

文章标题
文章得分
作者
评论
发布时间
阅读量

3.3.4 模块内部逻辑

先执行 create_ix 函数对数据创建索引，保存到 index 文件里面，读取数据库每一条数据，创建索引，create_searcher 创建搜索对象，按照索引查找返回相应的结果。

```
# 定义索引模式
class whoosh_text():
    def __init__(self):
        self.schema= Schema(url=ID(stored=True),
title=TEXT(stored=True),
                            nickname=TEXT(stored=True),
readcount=TEXT(stored=True)
                            , text=TEXT(stored=True,
analyzer=ChineseAnalyzer()), time=TEXT(stored=True))


    # 创建索引
    def create_ix(self):
        analyzer = ChineseAnalyzer()
        schema = self.schema
        # 创建索引存储目录
```

```
        if not os.path.exists("index"):
            os.mkdir("index")
        # 创建新索引
        ix = create_in("index", schema, 'my_index')
        # 从数据库中取数据
        db = pymysql.connect(host="127.0.0.1",
user="root",password="7894152630",database="csdn_crawler",charset="ut
f8")
        content = get_dbtext(db)
        # 新建 writer
        writer = ix.writer()
        # 遍历数据库, 插入 doc
        count = 0
        for blog in content:
            writer.add_document(url=u'%s' % blog[0], title=u'%s' %
blog[1], nickname=u'%s' % blog[2],
                                readcount=u'%s' % blog[3], text=u'%s' %
blog[4], time=u'%s' % blog[5])
            count += 1
            print('第', count, '篇 blog 添加成功...')
        writer.commit()

    def create_searcher(self):
        # 建立搜索对象
        ixr =
open_dir("F:\\search_engine\\CSDN_SearchEngine-master\\index",
'my_index')
        # with ixr.searcher(weighting=scoring.BM25F()) as searcher:
        self.searcher = ixr.searcher(weighting=scoring.BM25F()) # 没有
close, 可能会内存泄露, 记得关服务器。
        # 建立解析器(使用多字段查询解析器)
        self.parser = MultifieldParser(['title', 'text'],
schema=self.schema)
```

```
# 按照索引查询
def search_document(self,query,limit_=10000):
    searcher = self.searcher
    parser = self.parser
    q = parser.parse(query)
    results = searcher.search(q,limit=limit_)
    return results
```

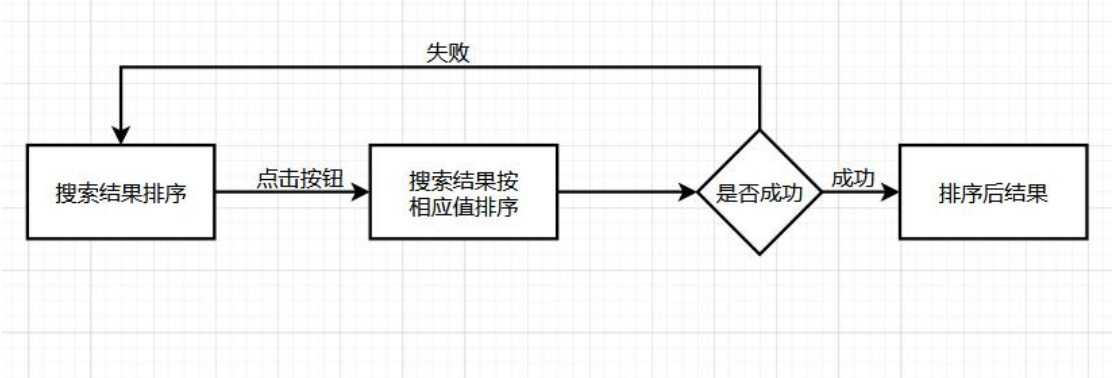
3.4 排序模块

具体格式如下：

功能编号	04	功能名称	排序模块	内容	功能流程图
所属业务	CSDN 搜索引擎	排序	所属项目	CSDN 博客	
编写人	李德宏	完成时间	2021-12-9	页码	第 19 页

3.4.1 功能流程图

功能流程图如下图：



需要说明的问题：

操作权限：面向所有用户

3.4.2 功能描述

（8） 功能类型：查询数据

（9） 功能概述：显示查询结果

（10） 前提业务：无

（11）后继业务：

（12） 功能约束：没有约束；

（13） 约束描述：

（14） 操作权限：面向所有用户

3.4.3 界面设计

（3） 基础信息处理

下表是动作说明：

动作编号	动作名称	动作描述
A01	按时间排序、按相关度、按阅读量排序	点击按钮 对数据进行排序，数据传输到 views.py

3.4.4 模块内部逻辑

对于已缓存的数据点击相应排序按钮，按照对应的属性值进行排序再把数据传输到 views.py 进行显示。

```
# 排序
    data['order'] = order
    order_num = 300
    if len(results) > order_num:
        results1 = results[:order_num]
```

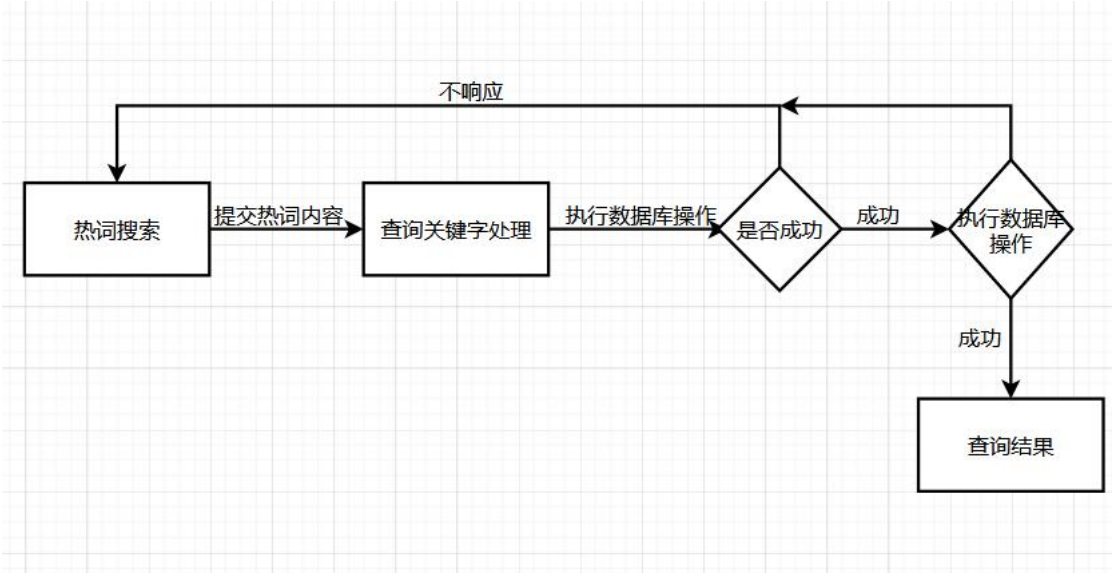
```
        results2 = results[order_num:]
    else:
        results1 = results
        results2 = []

    if (order == 'time'):
        results = sorted(results1, key=lambda x: x['time'],
reverse=True) + results2
    elif (order == 'readcount'):
        results = sorted(results1,
                         key=lambda x: (int)(x['readcount']), reverse=True)
+ results2
```

3.5 热词搜索模块

具体格式如下：

功能编号	05	功能名称	热词搜索模块	内容	功能流程图
所属业务	CSDN 搜索引擎	热词搜索	所属项目	CSDN 博客	
编写人	李德宏	完成时间	2021-12-9	页码	第 21 页

3.5.1 功能流程图



3.5.2 功能描述

（1） 功能类型：结果排序

（2） 功能描述：按照相应的属性进行排序

（3） 前提业务：搜索模块

（4） 后继业务：无

（5） 功能约束：权限约束

（6） 约束描述：无

（7） 操作权限：所有用户

3.5.3 界面设计

1， 基础信息处理

动作说明如下表：

动作编号	动作名称	动作描述
A01	热词内容	点击按钮 提交数据函数

处理

3.5.4 模块内部逻辑

数据库里面 search_engine_query 有一个 query 属性，是记录搜索关键字的，读取数据库，将关键字按数量排序，取前面 5 个作为热词显示出来。

```
# 把当前 query 存入数据库
    query1 = Query.objects.create(query=query,date=timezone.now())
    query1.save()

    data = dict()
    data['query'] = query
    pop_query = topQuery(pop_query_num)  # 热门搜索
    data['pop_query'] = pop_query
    results = searcher.search_document(query)
```

3.6 接口设计

（1） 用来查询一条数据的私有接口

```
def run(self):
    while(True):
        try:
            self.lock.acquire()
            url = self.get_nextURL()
            page = Page(url)
            print('正在抓取' + url)
            data = page.get_data()
            self.save_data(data)
            self.remove(url)
            urls = page.get_blogURLs()
            # print(urls)
            for url in urls:
```

```
                self.add_queue(url)

        except Exception as e:
            if hasattr(e, 'reason'):
                print('reason:', e.reason)
            elif hasattr(e, 'code'):
                print('error code:', e.code)
            else:
                print(e)
            self.remove(self.get_nextURL())  # 暴力删除,可能会一直删完 queue.
            continue
        finally:
            self.lock.release()
```

（2） 增加记录的公共接口

```
def add_queue(self,url):

    self.cur.execute("Select url from visited where url= %s",url)
    t = self.cur.fetchall()
    self.cur.connection.commit()
    self.cur.execute("Select url from url_queue where url= %s",url)
    n = self.cur.fetchall()
    self.cur.connection.commit()
    # print(t,len(t))

    if(len(t) == 0 and len(n) == 0):
        # print('insert' + url)
        self.cur.execute("Insert into url_queue VALUES (%s)",url)
        self.cur.connection.commit()
```

（3） 删除记录的公共接口

```
def remove(self,url):
    # print('remove' + url)
    self.cur.execute("Delete from url_queue where url= %s",url)
    self.cur.execute("Insert into visited values (%s)",url)
    # print('remove ' + url)
    self.cur.connection.commit()
```

（4） 查询记录的公共接口

```
def get_nextURL(self):
    # self.lock.acquire()
    self.cur.execute("select url from url_queue")
    url = self.cur.fetchone()[0]
    return url
```

3.7 测试要点

3.7.1 测试范围

测试范围	主要内容	简要说明
系统登陆验证	验证用户身份，进行权限控制	功能性测试
信息检索功能测试	测试数据库检索代码的健壮性	功能性测试

3.7.2 测试方法

功能性测试：黑盒测试