软件设计

这篇文档采用自顶向下的撰写思路，将本工具的设计思路层层展开，由粗到细，这也是我在开发过程中逐步细化得到的文档，并以此为基础开发：

概览

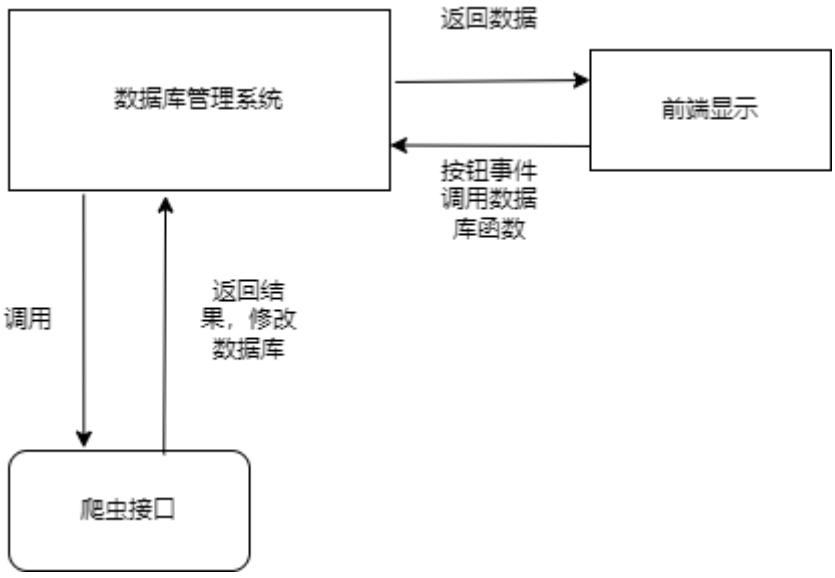
我们要实现的功能有：

- 用户应该可以对数据库中已有的文献信息进行增删查改；
- 用户应该可以在管理工具中记录对论文提出的问题、笔记等
- 用户应该可以指定年份和会议批量爬取论文信息并导入数据库；
- 用户应该可以根据本工具快速打开对应论文；

一些独特的需求有：

- 界面力求简洁，有学术的严谨性
- 要留给用户(研究者)一些自己标注、记录的空间，不应太过自动化

本软件大体上分为3部分，分别是论文数据库的管理、前端显示、爬虫的调用。几个模块关系可以如下所示：



技术路线

采用Python编写，运行在Windows10/11操作系统上：

- 前端：采用python自带的tkinter包进行UI开发
- 后端：采用sqlite数据库存储论文信息，调用清华大学cs论文爬虫PaperRobot来进行论文的自动爬取

文件列表：

```
|- \crawl_API            //爬虫的实现，这里调用了清华大学PaperRobot的开源库
|- database.db          // 数据库文件，自动生成
|- setting.conf          // 配置文件，自主编写
|
|- globalvar.py          // 全局变量文件，存储返回的错误类型，自主编写
|- database.py          // DataBase 的实现，自主编写
|- tkui.py              // TkUI 的实现，自主编写
|
|- main.py              // 主程序，自主编写
```

主函数main.py：用于作为软件入口

数据库模块：

database.py用于包装SQL语句，实现增删查改功能，将database实现为一个类

database.db用于本地存储论文数据库

前端模块：

tkui.py：内含ui实现函数，与按钮对应事件，包装并调用database.py的函数

爬虫模块：

/crawl_API实现爬虫功能并指定文件存储位置

具体实现

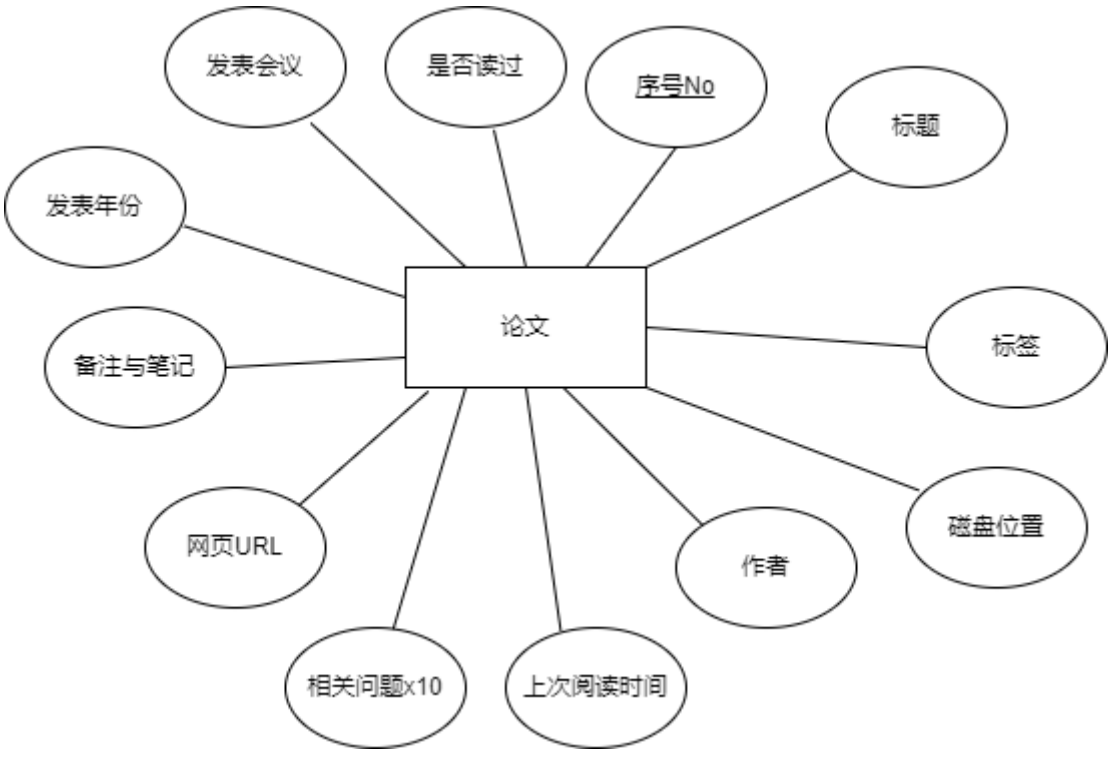
本节介绍前后端函数粒度的实现思路：

后端

database.db数据库只保存一张表，各字段如下：

属性中文名	属性英文名	类型	说明	
文献号	No	INTEGER	primary key AUTOINCREMENT	自动填写，对用户透明
阅读标记	ReadOrNot	INTEGER	仅允许 0 1	0表示未阅读完，1表示已经阅读完
发表日期	PublicationYear	INTEGER	YYYY	4位数字，表示年份，设置为INT类型以方便筛选
会议	Publisher	TEXT		
文献名称	PaperName	TEXT		
标签	Tags	TEXT	标签$标签$标签	使用$分割标签，标签内容不允许使用$
备注	Notes	TEXT		
网页链接	Url	TEXT		

属性中文名	属性英文名	类型	说明	
本地链接	Path	TEXT	要求使用绝对路径	
上一次阅读日期	LastReadDate	TEXT	YYYYMMDD	推荐使用年月日的格式，实际上不做要求
问题	Q0-Q9	TEXT	来自Readpaper的10个论文快读问题	
	Q1	TEXT		
	Q2	TEXT		
	Q3	TEXT		
	Q4	TEXT		
	Q5	TEXT		
	Q6	TEXT		
	Q7	TEXT		
	Q8	TEXT		
	Q9	TEXT		



database.py中实现了数据库类，其包括的内容如下：

使用组件 `sqlite3, os, copy`

database.py 全局变量

`DB_REL_PATH`：数据库文件的相对目录

DataBase 静态变量

`FIELD_LIST`：字符串 list，保存文献信息的全部字段名

`SIMPLE_FIELD_LIST`：字符串 list，保存显示文献条目时需要的信息，是 `FIELD_LIST` 的子集

DataBase 函数

函数列表

```
__init__(self)
add_paper(self, info:dict)
modi_paper(self, info:dict)
del_paper(self, info:dict)
show_paper(self, info:dict)
show_all_paper(self)
find_paper(self, info:dict)
get_all_pub(self)
get_all_tags(self)
```

__init__()

输入：

- 无输入

输出：

- None

主要功能：

- 构造函数
- 连接数据库文件（如没有数据库文件就生成数据库文件）

add_paper()

输入：

- info：字典，以 FIELD_LIST 中的字段为 key；PaperName字段不能为空；不读取 No、ReadOrNot 字段信息

输出：

- 成功添加返回 MY_SUCCESS

主要功能：

- 向文献库中添加文献
- No 字段自动生成
- ReadOrNot 字段默认为0

modi_paper()

输入：

- info：字典，可能包含 FIELD_LIST 中的全部字段，No字段不能为空

输出：

- 成功修改返回 MY_SUCCESS

主要功能：

- 以 No 查找，使用 info 提供的信息修改对应文献的信息

del_paper()

输入：

- info：字典，必须包含 No 字段

输出：

- 成功返回 MY_SUCCESS

主要功能：

- 从 info 中查找 No ，删除对应的文献条目
- 只能删除一条文献

show_all_paper()

输入：

- 无

输出：

- 成功则返回：字典 list，包含当前数据库中全部文献的全部信息。

find_paper()

输入：

- info：字典，
    - key取值：pubyear_begin pubyear_end puber_list tag_list keyword keyword_flag
    - value对应取值：起始年份 终止年份 出版商列表 标签列表 关键词 关键词查询位置标志

输出：

- 文献信息字典列表

主要功能：

- 提供搜索功能
- 筛选 pubyear_begin 当年及之后的文献
- 筛选 pubyear_end 当年及之前的文献
- 筛选属于 puber_list 中的出版商的文献
- 筛选包含 tag_list 中任意标签的文献
- 由 keyword_flag 的不同，在文献名和文献备注中依据是否包含 keyword 筛选文献

get_all_pub()

输入：

- 无

输出：

- 包含数据库中全部出版商（不重复）的列表

主要功能：

- 返回数据库中全部的出版商的列表
- 为 TkUI的出版商选择列表提供信息

get_all_tags()

输入：

- 无

输出：

- 包含数据库中全部标签（不重复）的列表

主要功能：

- 返回数据库中全部的标签的列表
- 为 TkUI的标签选择列表提供信息

前端

前端实现为TkUI类，主要用来生成可视化界面，接收用户的输入，类的实现位于 tkui.py。主要使用组件 `tkinter` 实现界面，还使用 DataBase 类实例作为私有变量，使用组件 `configparser, os, subprocess, copy`

tkui.py 全部变量

`SETTING_PATH` 保存配置文件 `setting.conf` 的相对位置

TkUI 主要私有变量

m_db : DataBase 实例，用来进行数据库操作

m_conf : configparser.ConfigParser 实例，用来进行配置文件操作

all_tag_list : 全部标签列表

all_puber_list : 全部出版商列表

paper_list : 当前用于展示的paper列表，是一个字典的列表，每个字典包含一个文献的全部信息

filter : 字典，当前筛选条件，键值内容与 DataBase.find_paper() 函数的输入要求一致

TkUI 使用 tkinter 实现UI，有大量UI部件及其存储值的变量，大多数都被设置成了私有变量，数量过多，这里不全部写出

root : tkinter.Tk 实例，主窗口

info_var : tkinter.StringVar 实例，低端信息栏的信息内容

TkUI 函数

仅与 UI 操作相关的函数不在这里给出，详请参照源文件注释

由于使用类的私有函数，可以访问类实例的私有变量并修改私有变量内容，大部分的函数没有输入、输出，故部分函数没有给出输入/输出，而在主要功能中做相关说明

__init__(db:DataBase)

输入：

- db：DataBase 实例

输出：

- None

主要功能

- 生成主窗口，主要包含搜索栏与文献展示栏

table_renewer(self)

主要功能

- 调用 search()函数，使用 filter 的现有查询条件更新 paper_list
- 刷新 展示栏的显示
- 注：search 函数位于 tkui.py，不是TkUI的内部函数

search_renewer(self)

主要功能：

- 调用 `m_db` 的 `get_all_tag()` 与 `get_all_pub()` 函数更新 all_tag_list 与 all_puber_list

search_paper(self)

主要功能：

- 从搜索栏中的信息更新筛选条件，调用 self.table_renewer() 更新文献

resume_paper(self)

主要功能：

- 重置筛选条件，重置搜索结果

get_setting(self)

主要功能：

- 读取设置文件，更新 m_conf

tkui.py 中的其它函数（非TkUI类函数）

这一部分的函数也承担主要功能，但不属于TkUI类

search(ui:TkUI, info:dict={})

输入：

- ui： TkUI 类实例
- info：字典，查询条件，键值设置与 DataBase.find_paper() 中的info输入要求一致

输出：

- 文献字典列表

主要功能：

- 与数据库连接，使用 ui.m_db.find_paper(info) 返回查询结果

open_add_ui(ui:TkUI)

输入：

- ui： TkUI类实例

输出：

- 无

主要功能：

- 该函数用来响应主窗口设置栏的添加文献按钮，实现读入文献信息，向数据库中添加文献

open_edit_ui(event, ui:TkUI, paper_no:str)

输入：

- ui： TkUI类实例
- paper_no : 文献的No，用于确认文献条目

输出：

- 无

主要功能：

- 该函数用于建立文献的详请界面
- 处理文献的修改、删除功能
- 处理文献的打开功能

open_setting_ui(ui:TkUI)

输入：

- ui： TkUI类实例

输出：

- 无

主要功能：

- 主窗口设置栏设置按钮响应函数
- 用来修改设置、保存设置、重置设置