图书管理系统实验报告

PB21000004 吴越

1 简介

本实验报告介绍了一个基于图书管理系统的应用程序。该应用程序使用Python编程语言和Tkinter库进行开发，使用SQLite数据库进行数据存储和管理，实现了一个简单而功能完善的图书管理系统。

2 需求分析

该图书管理系统应当具有以下功能特点：

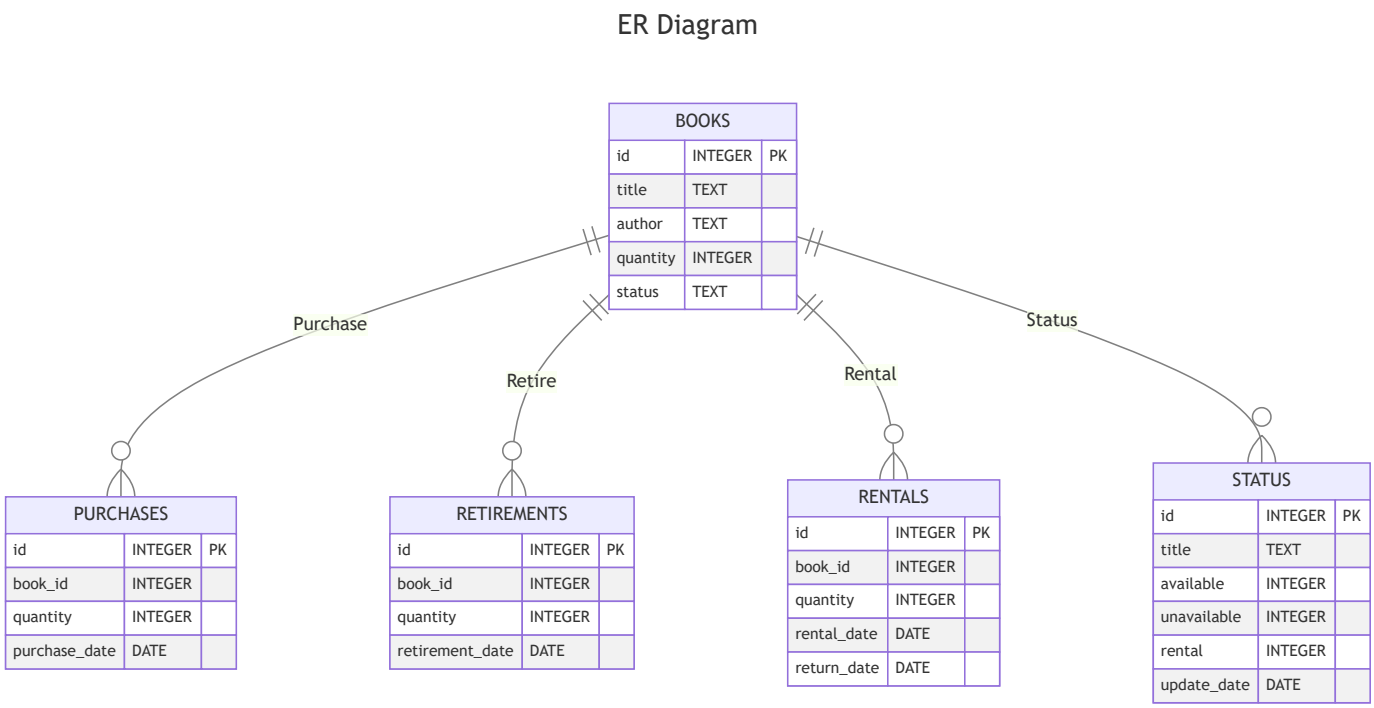
- 状态查询：用户可以查询图书的状态，包括可用数量、借出数量等。
- 图书管理：用户可以执行图书的增加、删除、借出和归还等操作。
- 数据库支持：系统使用SQLite数据库进行数据存储和管理，确保数据的持久性和安全性。
- 用户界面：系统提供了一个直观友好的用户界面，使用者可以轻松地浏览和操作图书信息。

3 实现思路

图书管理系统的实现思路如下：

1. 创建一个主窗口，使用Tkinter库进行界面设计。
2. 在主窗口中添加标签、按钮等控件，用于显示图书信息和执行相关操作。
3. 使用SQLite数据库存储图书信息和采购记录。
4. 实现图书采购相关的方法，包括增加图书、更新图书状态等。
5. 实现与SQLite数据库的交互，包括连接数据库、执行SQL查询和更新等操作。
6. 实现一些通用的辅助函数，如日期格式转换、样式设置等。

4 ER图设计



5 代码分析

1）存储图书信息、采购和淘汰情况、租借情况

以下是文件中的一部分代码，用于创建记录表：

```
self.cursor.execute('''
    CREATE TABLE IF NOT EXISTS books (
        id INTEGER PRIMARY KEY,
        title TEXT,
        author TEXT,
        quantity INTEGER,
        status TEXT
    )
''')

# 创建采购记录表
self.cursor.execute('''
    CREATE TABLE IF NOT EXISTS purchases (
        id INTEGER PRIMARY KEY,
        book_id INTEGER,
        quantity INTEGER,
        purchase_date DATE,
        FOREIGN KEY (book_id) REFERENCES books(id)
    )
''')

# 创建淘汰记录表
self.cursor.execute('''
    CREATE TABLE IF NOT EXISTS retirements (
        id INTEGER PRIMARY KEY,
        book_id INTEGER,
        quantity INTEGER,
        retirement_date DATE,
        FOREIGN KEY (book_id) REFERENCES books(id)
    )
''')

# 创建租借记录表
self.cursor.execute('''
    CREATE TABLE IF NOT EXISTS rentals (
        id INTEGER PRIMARY KEY,
        book_id INTEGER,
        quantity INTEGER,
        rental_date DATE,
        return_date DATE,
        FOREIGN KEY (book_id) REFERENCES books(id)
    )
''')
# 创建图书状态表（图书名称，可操作（本），淘汰（本），外借（本），更新时间）
self.cursor.execute('''
    CREATE TABLE IF NOT EXISTS status (
        id INTEGER PRIMARY KEY,
        title TEXT,
        available INTEGER,
```

```
        unavailable INTEGER,
        rental INTEGER,
        update_date DATE
    )
''')
```

例如。在上述代码创建采购记录表。该方法首先连接到SQLite数据库，然后执行SQL语句创建名为`purchases`的表，包含`id`、`book_id`、`quantity`和`purchase_date`等字段。最后，关闭数据库连接。实现数据库的建立。

2）实现图书采购、淘汰、租借功能。

这里以采购功能为例：

```
# 更新图书数量和状态
self.cursor.execute('UPDATE books SET quantity = quantity + ? WHERE id = ?', (quantity, id))
self.cursor.execute('UPDATE books SET status = ? WHERE id = ?', ('Available', id))
self.conn.commit()

# 插入采购记录
self.cursor.execute('INSERT INTO purchases (book_id, quantity, purchase_date) VALUES (?, ?, ?)', (id, quantity, datetime.now().strftime('%Y-%m-%d %H:%M')))
self.conn.commit()
self.cursor.execute('SELECT * FROM books WHERE id = ?', (id,))
book = self.cursor.fetchone()
self.cursor.execute('UPDATE status SET available = ? WHERE title = ?', (book[3], book[1]))
self.cursor.execute('UPDATE status SET update_date = ? WHERE title = ?', (datetime.now().strftime('%Y-%m-%d %H:%M'), book[1]))
self.conn.commit()
```

第一个SQL语句使用了参数化查询，通过传入数量和图书ID来更新相应图书的数量。第二个SQL语句也是参数化查询，将相应图书的状态更新为"Available"。然后通过self.conn.commit()来提交这些更新，将其应用到数据库中。

接着，代码执行了一系列操作来插入一条采购记录。首先，使用参数化查询将图书ID、数量和当前日期时间插入到"purchases"表中。然后通过self.conn.commit()提交这次插入操作。接着，代码执行了一系列SQL语句来更新图书状态表中的可用数量和更新日期时间，以反映最新的图书状态，这样就实现了基本的采购功能。

3）实现图书信息、采购和淘汰、库存、和租借情况查询

仍然以采购为例：

```
# 展示采购记录
self.tree_purchases = ttk.Treeview(self.tab_purchase, columns=('ID', 'Book ID', 'Quantity', 'Purchase Date'), show='headings')
self.tree_purchases.heading('ID', text='ID')
self.tree_purchases.heading('Book ID', text='Book ID')
self.tree_purchases.heading('Quantity', text='Quantity')
self.tree_purchases.heading('Purchase Date', text='Purchase Date')

self.tree_purchases.column('ID', width=50, minwidth=50, anchor="center")
self.tree_purchases.column('Book ID', width=100, minwidth=100, anchor="center")
self.tree_purchases.column('Quantity', width=100, minwidth=100, anchor="center")
self.tree_purchases.column('Purchase Date', width=100, minwidth=100, anchor="center")
```

```
12
13  self.tree_purchases.place(x=100, y=0, width=600, height=400)
14
15  #self.tree_purchases.grid(row=0, column=3, padx=5, pady=5, rowspan=3)
16
17  # 查询采购记录并插入表格
18  self.cursor.execute('SELECT * FROM purchases')
19  purchases = self.cursor.fetchall()
20  for purchase in purchases:
21      self.tree_purchases.insert('', 'end', values=purchase)
```

首先，代码创建了一个ttk.Treeview部件，用来展示采购记录。它有四列，分别是ID、Book ID、Quantity和Purchase Date。然后设置了每列的标题和宽度，以及它们在Treeview中的位置和尺寸。接着将这个Treeview部件放置在界面上的指定位置，通过place方法设置了它的位置和尺寸。

最后，代码执行了一条SQL查询语句，从数据库中获取所有的采购记录，并将它们插入到创建的Treeview部件中，以便在界面上展示出来。这样就创建一个表格，用来展示数据库中的采购记录，并将这些记录插入到表格中以供用户查看。

4）实现图书的采购、库存、淘汰、租借情况等统计功能

统计功能实现如下：

```
1   # 查询统计信息
2   self.cursor.execute('SELECT * FROM status')
3   status = self.cursor.fetchall()
4
5   # 清空表格
6   for row in self.tree_statistics.get_children():
7       self.tree_statistics.delete(row)
8
9   # 插入新数据
10  for s in status:
11      self.tree_statistics.insert('', 'end', values=s)
12
13  # 更新统计信息
14  self.cursor.execute('SELECT SUM(quantity) FROM books')
15  total = self.cursor.fetchone()
16  self.label_statistics_total_num.config(text=total[0])
17
18  self.cursor.execute('SELECT SUM(available) FROM status')
19  available = self.cursor.fetchone()
20  self.label_statistics_available_num.config(text=available[0])
21
22  self.cursor.execute('SELECT SUM(rental) FROM status')
23  rental = self.cursor.fetchone()
24  self.label_statistics_rental_num.config(text=rental[0])
25
26  self.cursor.execute('SELECT SUM(unavailable) FROM status')
27  unavailable = self.cursor.fetchone()
28  self.label_statistics_unavailable_num.config(text=unavailable[0])
29
30  self.cursor.execute('SELECT MAX(update_date) FROM status')
31  update_date = self.cursor.fetchone()
```

```
32  self.label_statistics_update_date_num.config(text=update_date[0])
33
34  self.cursor.execute('SELECT COUNT(*) FROM books')
35  num = self.cursor.fetchone()
36  self.label_statistics_num_num.config(text=num[0])
```

首先，代码执行了一条SQL查询语句，从数据库中获取了"status"表中的所有数据，并将结果存储在变量status中。接着，代码清空了名为tree_statistics的表格部件中的所有行数据，以便插入新的统计信息。然后，代码遍历查询到的status数据，并将每条数据插入到tree_statistics表格中。接下来，代码执行了一系列SQL查询来计算图书总数量、可用数量、租借数量、不可用数量、最后更新日期和图书总数，并将这些统计信息更新到界面上相应的部件中，比如label_statistics_total_num、label_statistics_available_num等。

这样通过查询数据库中的统计信息，并将这些信息更新到界面上的相应部件中，以便用户可以看到最新的统计数据。

6　使用方法

要使用图书管理系统，按照以下步骤进行操作：

1. 安装Python和Tkinter库。
2. 下载并解压缩图书管理系统的源代码。
3. 打开终端或命令提示符，导航到源代码所在的目录。
4. 运行以下命令启动图书管理系统：

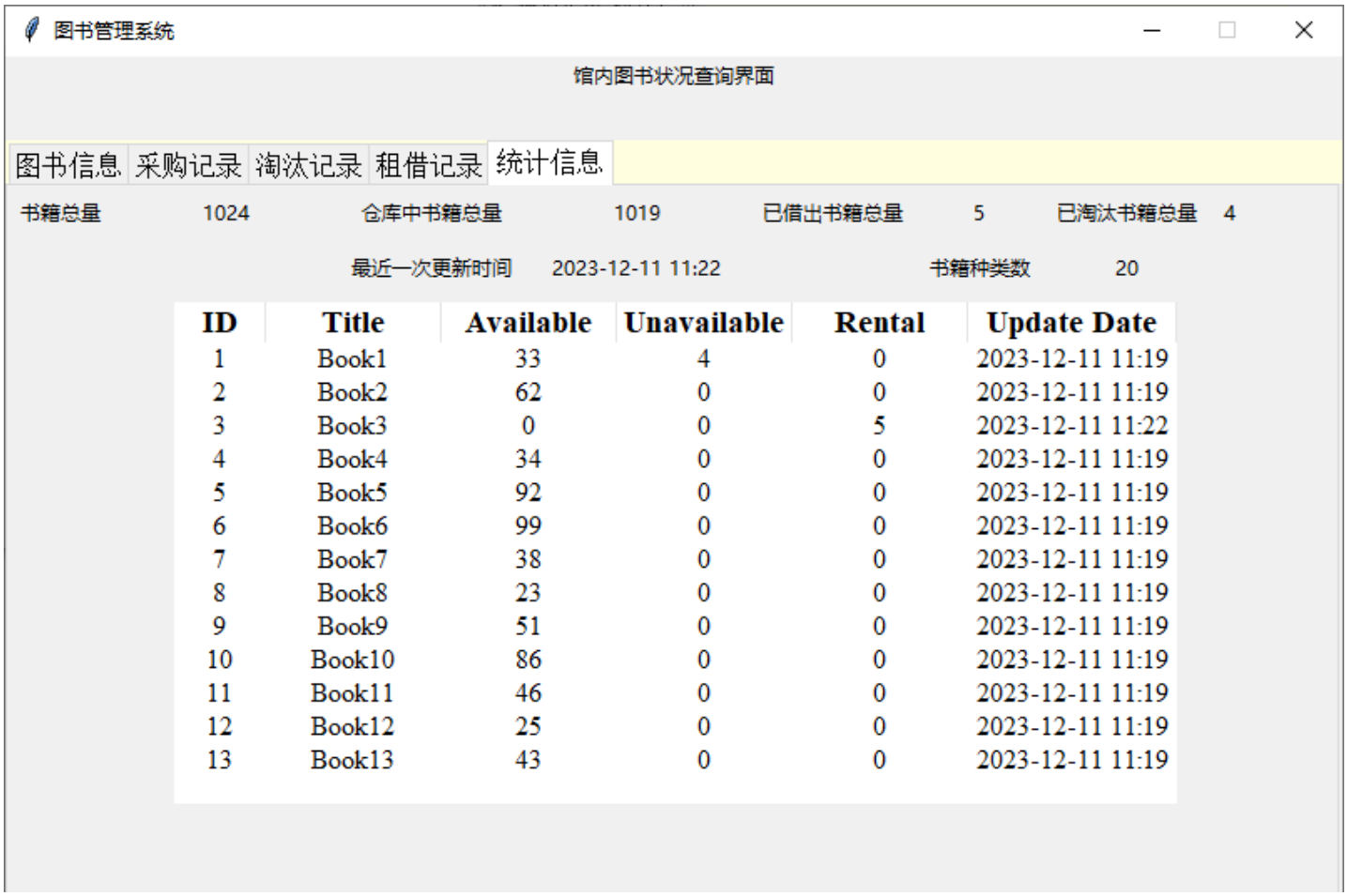
   python main.py
5. 图书管理系统的主界面将显示在屏幕上。
6. 在主界面上，您可以选择状态查询或图书管理等功能。
7. 根据需求，执行相应的操作，如查询图书状态、增加图书、借出图书等。

操作例如下：


图书管理系统
馆内图书状况管理界面
图书采购
图书淘汰
图书租借/归还
Book ID
2
Search
图书存在请输入购买的数量
3
购买


图书管理系统
馆内图书状况管理界面
图书采购
图书淘汰
图书租借/归还
Book ID
1
Search
图书1存在请输入淘汰的数量
4
淘汰


图书管理系统
馆内图书状况查询界面
图书信息
采购记录
淘汰记录
租借记录
统计信息
Refresh
ID
Book ID
Quantity
Purchase Date
1
2
3
2023-12-11 11:19



7 总结

通过本次实验，我们成功开发了一个图书管理系统应用程序。该系统具有直观的用户界面和丰富的功能，可以满足图书管理的基本需求。通过使用Python和Tkinter库，我们实现了系统的核心逻辑和界面设计。通过SQLite数据库的支持，我们实现了数据的持久化和安全性。图书管理系统能够为用户提供便利和效率。

8 参考资料

- Python官方文档：https://docs.python.org/
- Tkinter官方文档：https://docs.python.org/3/library/tkinter.html
- SQLite官方文档：https://www.sqlite.org/docs.html