图书销售管理系统

本实验实现的是一个图书销售管理系统的设计和实现。某书城需要一套图书管理系统对图书的进货、销售、财务等方面进行统一管理。

- 1. 用户管理
  - 1）系统用户分为超级管理员用户和普通管理员用户。普通管理员用 户只能对图书进货、销售等信息进行管理，只能查询和修改自己的 用户信息；而超级管理员除了可以对图书进货、销售等信息进行管 理，还能创建新的用户和查看所有用户的资料。
  - 2）超级管理员用户在系统完成时便已经存在（即其用户名和密码已 经存在于数据库中）。而普通管理员用户的用户名和密码需要由超 级管理员用户来创建。
  - 3）用户的密码不能以明文形式保存于数据库中，而必须先加密，一
    般采用MD5算法进行加密。
  - 4）每位用户除了用户名和密码信息外，还有真实姓名，工号，性别， 年龄等基本信息。
  - 5）系统所有功能只有用户登录了才能进行操作

- 2. 库存书籍管理
  - 系统中需要维护整个书城目前库存的所有书籍信息，包 括书籍ISBN号，书籍名称，出版社，作者，零售价格， 当前库存数量等。
- 3. 书籍查询
  - 可以使用书籍编号、书籍ISBN号、书名、作者、出版社 等方式查询库存的相关书籍。
- 4. 图书信息修改
  - 可以修改书籍名称、作者、出版社、零售价格等信息

- 8. 添加新书：
  - 对于已付款的书籍，当书籍到货后，可以将其添加到库 存中，此时需要添加上书籍的零售价格。
- 9. 书籍购买：
  - 使用标售零售价格购买书籍，这时书籍的库存数量需要 相应地减少。
- 10. 财务管理：
  - 当对书籍进货进行付款，或购买书籍时，系统的财务账 户都要添加一条账单记录，记录下财务账户的支出或收 入。
- 11. 查看账单
  - 查看某段时间内财务账户的收入或支出记录。

- 5. 图书进货：
  - 对于需要进货的书籍，如果库存中曾经有这本书的信息 的话，则直接将这本书的ID列入进货清单，否则需要输 入进货书籍的相关信息，包括ISBN号，书名，作者，出 版社等。此外，每种书都要指定其进货价格和购买数量。 对于刚列入进货清单的书籍给予未付款状态。
- 6. 进货付款：
  - 查询正在进货的书籍，并给予付款，付款后书籍状态为
    已付款。
- 7. 图书退货：
  - 对未付款的书籍可以进行退货，即将书籍状态改为已退 货。

系统功能

1. 用户管理

- 支持多角色用户系统：超级管理员(superadmin)、管理员(admin)和普通用户(user)
- 用户注册和登录功能
- 密码使用MD5加密存储
- 用户信息包含：用户名、密码、角色、真实姓名、工号、性别、年龄
- 超级管理员可以添加其他管理员账号

2. 图书管理

- 图书信息管理：添加、修改、查询图书信息
- 图书信息包含：ID、ISBN、书名、作者、出版社、出版年份、价格、总数量、可用数量
- 支持多条件组合查询图书
- 图书库存管理

3. 读者管理

- 读者信息管理：添加、查询读者信息
- 读者信息包含：ID、姓名、性别、电话

4. 借阅管理

- 图书借阅和归还功能
- 借阅记录管理
- 自动更新图书可用数量

5. 进货管理

- 进货单管理：添加、付款、退货
- 支持新书进货和已有图书补货
- 进货单状态管理：未付款、已付款、退货
- 新书入库功能

6. 账单管理

- 自动记录所有收支情况
- 支持按时间段查询账单
- 记录类型：收入（图书销售）、支出（图书进货）

设计思路

1. 系统架构

- 采用Python + SQLite实现
- 使用tkinter实现GUI图形用户界面

2. 安全设计

- 用户密码使用MD5加密存储，输入的时候会用"*"加密保护，如下所示
- 基于角色的访问控制(RBAC)
- 不同角色拥有不同的操作权限

```
def login_user(username, password):
    conn = sqlite3.connect('library.db')
    c = conn.cursor()
    c.execute('SELECT * FROM users WHERE username=? AND password=?', (username, md5_hash(passwor
    user = c.fetchone()
    conn.close()
    if user:
        print('登录成功! ')
        return user
    else:
        print('用户名或密码错误! ')
        return None
```

3. 数据更新

- 库存变动时自动更新相关数据
- 进货和销售时自动生成账单记录
- 自行设计了类似触发器的函数，borrow_book() return_book()等，如果未成功执行的话会自动interrupt异常中断，类似触发器当中rollback的作用，如下所示

```
def update_book(book_id, isbn=None, title=None, author=None, publisher=None, year=None, price=N
    conn = sqlite3.connect('library.db')
    c = conn.cursor()
    # 获取当前信息
    c.execute('SELECT * FROM books WHERE id=?', (book_id,))
    book = c.fetchone()
    if not book:
        print('未找到该图书')
        conn.close()
        return
    # 只更新有输入的字段
    new_isbn = isbn if isbn is not None else book[1]
    new_title = title if title is not None else book[2]
    new_author = author if author is not None else book[3]
    new_publisher = publisher if publisher is not None else book[4]
    new_year = year if year is not None else book[5]
    new_price = price if price is not None else book[6]
    new_total = total if total is not None else book[7]
    # 可用数量自动调整
    available = book[8] + (new_total - book[7])
    c.execute('''UPDATE books SET isbn=?, title=?, author=?, publisher=?, year=?, price=?, tota
              (new_isbn, new_title, new_author, new_publisher, new_year, new_price, new_total, a
    conn.commit()
    print('图书信息已更新')
    conn.close()
```

数据表设计

1. books（图书表）

```
CREATE TABLE books (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT,           -- 书名
    author TEXT,          -- 作者
    publisher TEXT,       -- 出版社
    year INTEGER,         -- 出版年份
    price REAL,           -- 价格
    total INTEGER,        -- 总数量
    available INTEGER     -- 可用数量
)
```

2. readers（读者表）

```
CREATE TABLE readers (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT,            -- 姓名
    gender TEXT,          -- 性别
    phone TEXT            -- 电话
)
```

3. borrows（借阅表）

```
CREATE TABLE borrows (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    book_id INTEGER,      -- 图书ID
    reader_id INTEGER,    -- 读者ID
    borrow_date TEXT,     -- 借阅日期
    return_date TEXT,     -- 归还日期
    returned INTEGER      -- 是否已归还
)
```

4. users（用户表）

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT UNIQUE, -- 用户名
    password TEXT,         -- 密码（MD5加密）
    role TEXT,            -- 角色
    realname TEXT,        -- 真实姓名
    job_id TEXT,          -- 工号
    gender TEXT,          -- 性别
    age INTEGER           -- 年龄
)
```

5. purchase_orders（进货表）

```
CREATE TABLE purchase_orders (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    book_id INTEGER,      -- 图书ID（可选）
    isbn TEXT,            -- ISBN
    title TEXT,           -- 书名
    author TEXT,          -- 作者
    publisher TEXT,       -- 出版社
    price REAL,           -- 进货价格
    quantity INTEGER,     -- 数量
    status TEXT           -- 状态（未付款/已付款/退货）
)
```

6. bills（账单表）

```
CREATE TABLE bills (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    type TEXT,            -- 类型（收入/支出）
    amount REAL,          -- 金额
    description TEXT,     -- 描述
    time TEXT            -- 时间
)
```

使用说明

1. 系统初始化时会自动创建超级管理员账号：
    - 用户名：root
    - 密码：root123
2. 超级管理员权限：
    - 添加管理员
    - 查看所有用户
    - 进行所有图书、读者、借阅管理
    - 进行进货和账单管理
3. 管理员权限：
    - 进行图书、读者、借阅管理
    - 进行进货和账单管理
4. 普通用户：
    - 无操作权限，需要联系管理员

运行环境

- Python 3.x
- SQLite3
- tkinter (GUI库)