

中期实验：图书销售系统

姓名：段光杰

学号：22307130301

日期：2024 年 5 月 9 日

实验思路：

根据题目要求，我们需要完成一个图书销售系统，来实现对图书的进货、销售、财务等方面进行统一管理，那么，我们就可以构思出大致的思路：

1. 初步设计数据库，画出 E-R 图
2. 设计前端与后端，图形化界面将用户操作与数据库连接起来
3. 完善数据库中的逻辑，包括函数、触发器等

一、数据库的初步设计：

先根据题目要求设计出数据库，画出 E-R 图

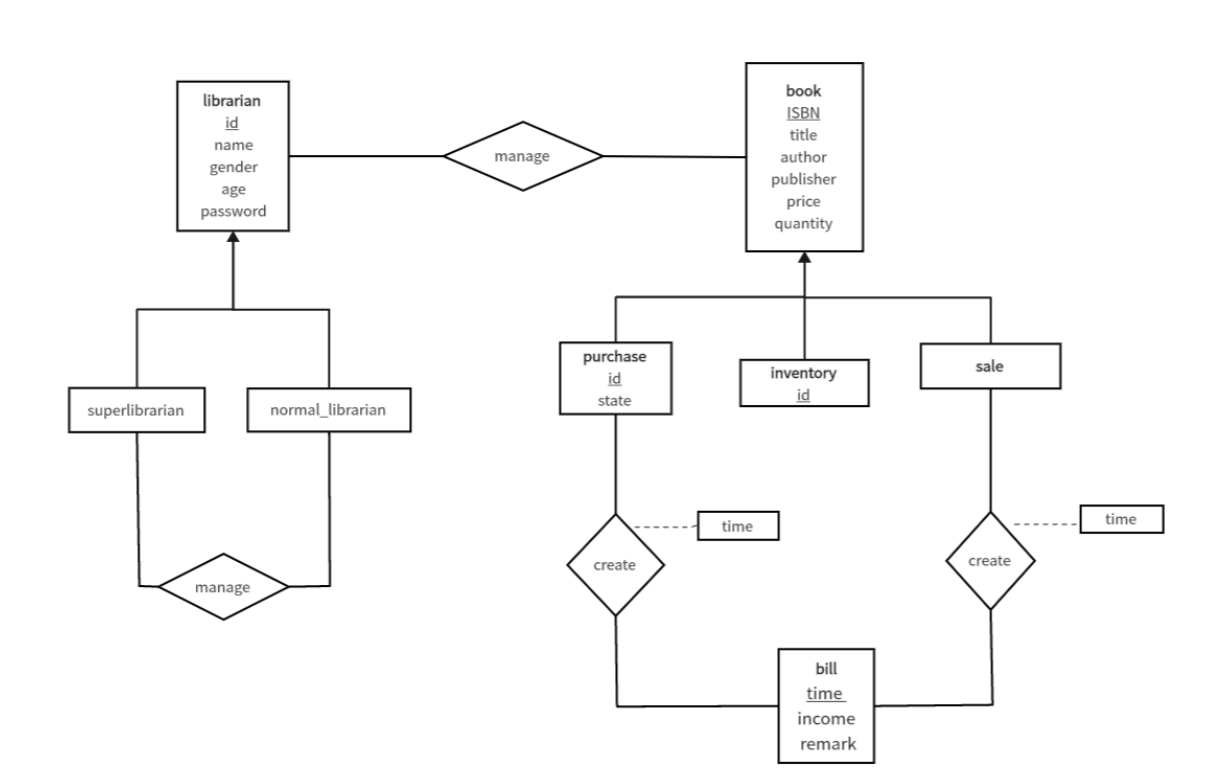


图 1 E-R 图

然后，我们设计的几个数据库大致如下：

```
CREATE TABLE IF NOT EXISTS public.librarian
(
  id character varying COLLATE pg_catalog."default" NOT NULL,
  name character varying COLLATE pg_catalog."default" NOT NULL,
  gender character varying COLLATE pg_catalog."default" NOT NULL,
  age integer NOT NULL,
  password character varying COLLATE pg_catalog."default" NOT NULL,
  CONSTRAINT librarian_pkey PRIMARY KEY (id)
)
```

```
CREATE TABLE IF NOT EXISTS public.superlibrarian
(
  id character varying COLLATE pg_catalog."default" NOT NULL,
  name character varying COLLATE pg_catalog."default" NOT NULL,
  gender character varying COLLATE pg_catalog."default" NOT NULL,
  age integer NOT NULL,
  password character varying COLLATE pg_catalog."default" NOT NULL,
  CONSTRAINT superlibrarian_pkey PRIMARY KEY (id)
)
```

```
CREATE TABLE IF NOT EXISTS public.inventory
(
  id character varying COLLATE pg_catalog."default" NOT NULL,
  "ISBN" character varying COLLATE pg_catalog."default" NOT NULL,
  title character varying COLLATE pg_catalog."default" NOT NULL,
  author character varying COLLATE pg_catalog."default",
  publisher character varying COLLATE pg_catalog."default",
  price money,
  quantity integer,
  CONSTRAINT inventory_pkey PRIMARY KEY ("ISBN")
)
```

```
CREATE TABLE IF NOT EXISTS public.purchase
(
  "ISBN" character varying COLLATE pg_catalog."default" NOT NULL,
  title character varying COLLATE pg_catalog."default" NOT NULL,
  author character varying COLLATE pg_catalog."default",
  publisher character varying COLLATE pg_catalog."default",
  price money,
  quantity integer,
  state character varying COLLATE pg_catalog."default",
  id integer NOT NULL DEFAULT nextval('purchase_id_seq'::regclass),
  CONSTRAINT purchase_pkey PRIMARY KEY (id)
)
```

```
CREATE TABLE IF NOT EXISTS public.bill
(
    "time" timestamp with time zone NOT NULL,
    income numeric,
    remark character varying COLLATE pg_catalog."default",
    CONSTRAINT bill_pkey PRIMARY KEY ("time")
)
```

到这里，我们已经初步完成了数据库的设计，但还没有完成其中的与用户交互操作的部分以及数据库的逻辑，接下来，我们完成前后端。

二、前后端以及数据库设计：

这里，我们使用 Qt 来设计一个 windows 桌面图像化交互界面。

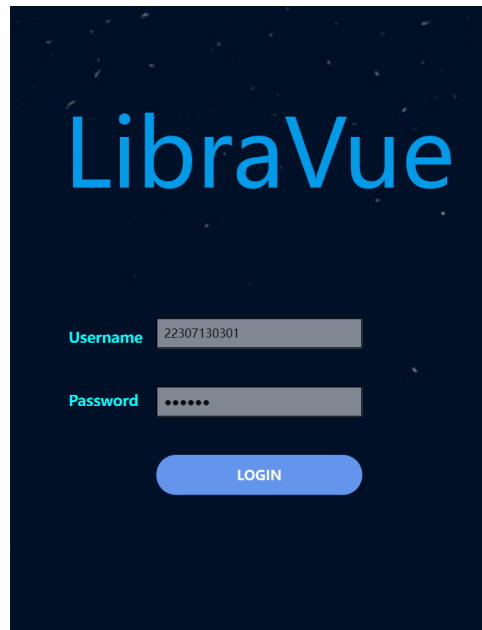
首先，使用 Qt 连接本地数据库，我们将这个过程写成函数来使用：

```
void MainWindow::createConnectionByName(const QString &connectionName){
    QSqlDatabase db = QSqlDatabase::addDatabase("QPSQL", connectionName);
    // 数据库连接需要设置的信息
    db.setHostName("127.0.0.1"); // 数据库服务器IP，我用的是本地电脑
    db.setDatabaseName("library");// 数据库名
    db.setUserName("postgres");// 用户名
    db.setPassword("lumilion");// 密码
    db.setPort(5432);// 端口号
    // 连接数据库
    bool ok = db.open();

    if (ok)
    {
        qDebug() << "database connect is ok";
    }
    else
    {
        qDebug() << "database connect is fail";
    }
}

// 使用自定义 connectionName 获取连接
QSqlDatabase MainWindow::getConnectionByName(const QString &connectionName) {
    // 获取数据库连接
    return QSqlDatabase::database(connectionName);
}
```

登录界面的设计过程，可以选择登录或者退出，选择登录后可以输入用户名和密码，若输入正确，则成功登录，若输入错误，则会提示错误信息。



密码的检索在数据库中使用了摘要算法进行加密，所以此处使用摘要算法哈希计算来与数据库中的值进行比对：

Query

Query History

1

2

SELECT id, name, gender, age, password

FROM public.librarian;

Data Output

Messages

Notifications

	id [PK] character varying	name character varying	gender character varying	age integer	password character varying
1	001	James	男	19	a7fda0b61e2047f0f1057d1f5f064c272fd5d490961c531f4df64b0dd354683a
2	002	Marry	女	22	f9200e0934f3c98a0a6b7dcfb5753aa0d6e4a401a89db72cccd84e2845d0...
3	003	Mike	男	21	86e0fa410d1c284b13d7b123ff1484fac76b17c4087bad575b8471b7624e0d...
4	004	Jerry	女	23	77c588483d8834805c7a53c156f2c769cf041cd527d89c14b7ebdaeeb7e77...
5	005	Linda	女	21	da5fd72f957f98e2c456d54b4288e7bece4ca62114c17d866b4a4e1260df06...

连接数据库后，在 Qt 中使用数据库查询语句查询管理员是否存在，以及密码是否匹配，同时，使用全局变量来记录管理员的身份：

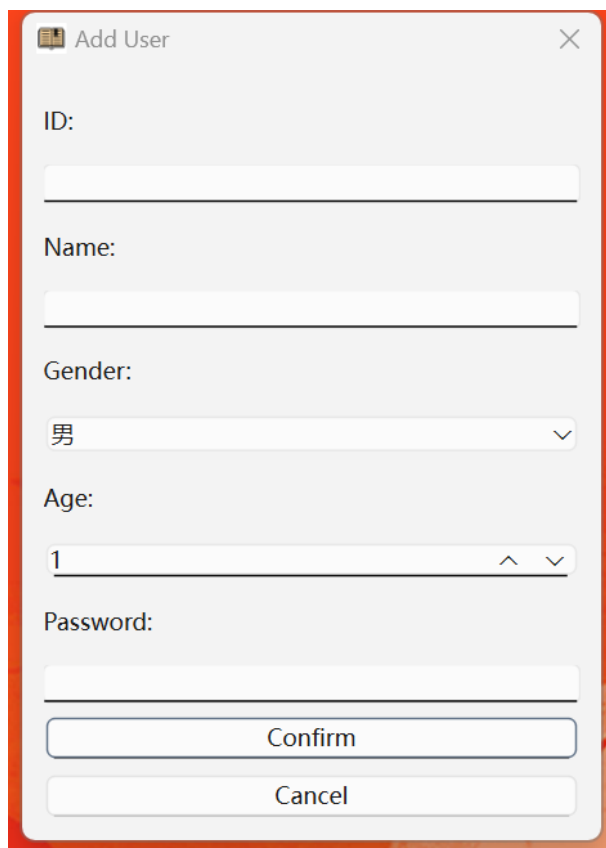
```

createConnectionByName("firstConnect");
QSqlDatabase db = getConnectionByName("firstConnect");
QSqlQuery query(db);
query.prepare("SELECT password FROM superlibrarian WHERE id = :username ");
query.bindValue(":username", username);
if (!query.exec()) {
    qWarning() << "Query failed:";
    return;
}
// 检查是否存在对应的用户名
if (!query.next()) {
    qWarning() << "Username not found";
    Privilege = false;
    ui->error_prompt->show();
    query.prepare("SELECT password FROM librarian WHERE id = :username ");
    query.bindValue(":username", username);
    if (!query.exec()) {
        qWarning() << "Query failed:";
        return;
    }
    // 检查是否存在对应的用户名
    if (!query.next()) {
        qWarning() << "Username not found";
        ui->error_prompt->show();
        return;
    }
}
}

```

然后，我们按照要求来完成每一项功能

2.1 添加人员

A screenshot of a web application dialog box titled "Add User". The dialog box has a light gray background and a red border. It contains several input fields: "ID:" with a text input, "Name:" with a text input, "Gender:" with a dropdown menu showing "男" (Male), "Age:" with a spinner box showing "1", and "Password:" with a text input. At the bottom, there are two buttons: "Confirm" and "Cancel".

Add User

ID:

Name:

Gender:

男

Age:

1

Password:

Confirm

Cancel

添加人员的实现实际上需要前端接受很多的变量，这一部分变量可以直接插入数据库中，因此为了简便，没有在数据库中再写一个函数，具体实现为：使用前端对话框让用户输入数据信息，再将信息存入数据库。（这个框架的实现在后续所有代码中都可以使用，可以套用代码来实现对所有数据库信息的查看。）

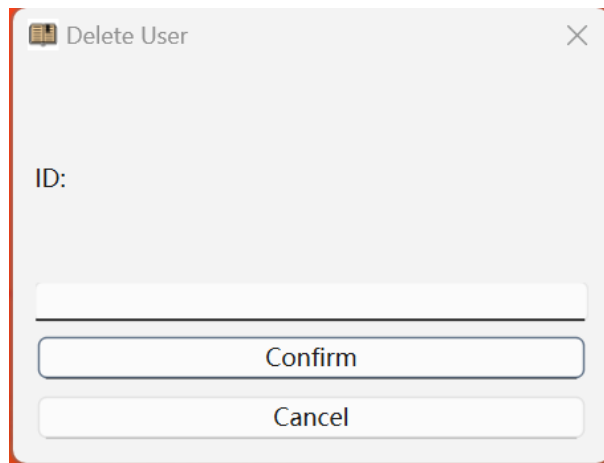
```

// 根据对话框的返回值执行相应操作
if (result == QDialog::Accepted) {
    // 用户点击了确认按钮
    QString id = idLineEdit->text();
    QString name = nameLineEdit->text();
    QString gender = genderComboBox->currentText();
    int age = ageSpinBox->value();
    QString password = passwordLineEdit->text();
    // 在这里执行添加用户的操作，可以将获取到的用户信息存入数据库或进行其他处理
    MainWindow mainWindow;
    mainWindow.createConnectionByName("Connect2");
    QSqlDatabase db = mainWindow.getConnectionByName("Connect2");
    QSqlQuery query(db);
    query.prepare("INSERT INTO librarian (id, name, gender, age, password) "
        | | | | "VALUES (:id, :name, :gender, :age, :password)");
    query.bindValue(":id", id);
    query.bindValue(":name", name);
    query.bindValue(":gender", gender);
    query.bindValue(":age", age);
    QByteArray inputHash = QCryptographicHash::hash(password.toUtf8(), QCryptographicHash::Sha256);
    QString inputHashStr = QString(inputHash.toHex());
    query.bindValue(":password", inputHashStr);

    if (!query.exec()) {
        | qWarning() << "Insert failed:" ;
    } else {
        | qDebug() << "User inserted successfully!";
    }
}
}

```

2.2 删除人员



同样的方式来执行数据库的删除操作，也是简单的删除操作，我们直接在后端实现。


```

// 根据对话框的返回值执行相应操作
if (result == QDialog::Accepted) {
    // 用户点击了确认按钮
    QString id = idLineEdit->text();
    // 执行删除操作
    MainWindow mainWindow;
    mainWindow.createConnectionByName("Connect2");
    QSqlDatabase db = mainWindow.getConnectionByName("Connect2");
    QSqlQuery query(db);
    query.prepare("DELETE FROM librarian WHERE id = :id");
    query.bindValue(":id", id);

    // 执行 SQL 删除语句
    if (!query.exec()) {
        qWarning() << "Delete query failed:" ;
        return;
    }

    // 显示删除成功的消息或执行其他操作
    qDebug() << "Record with ID" << id << "has been deleted successfully!";
}

```

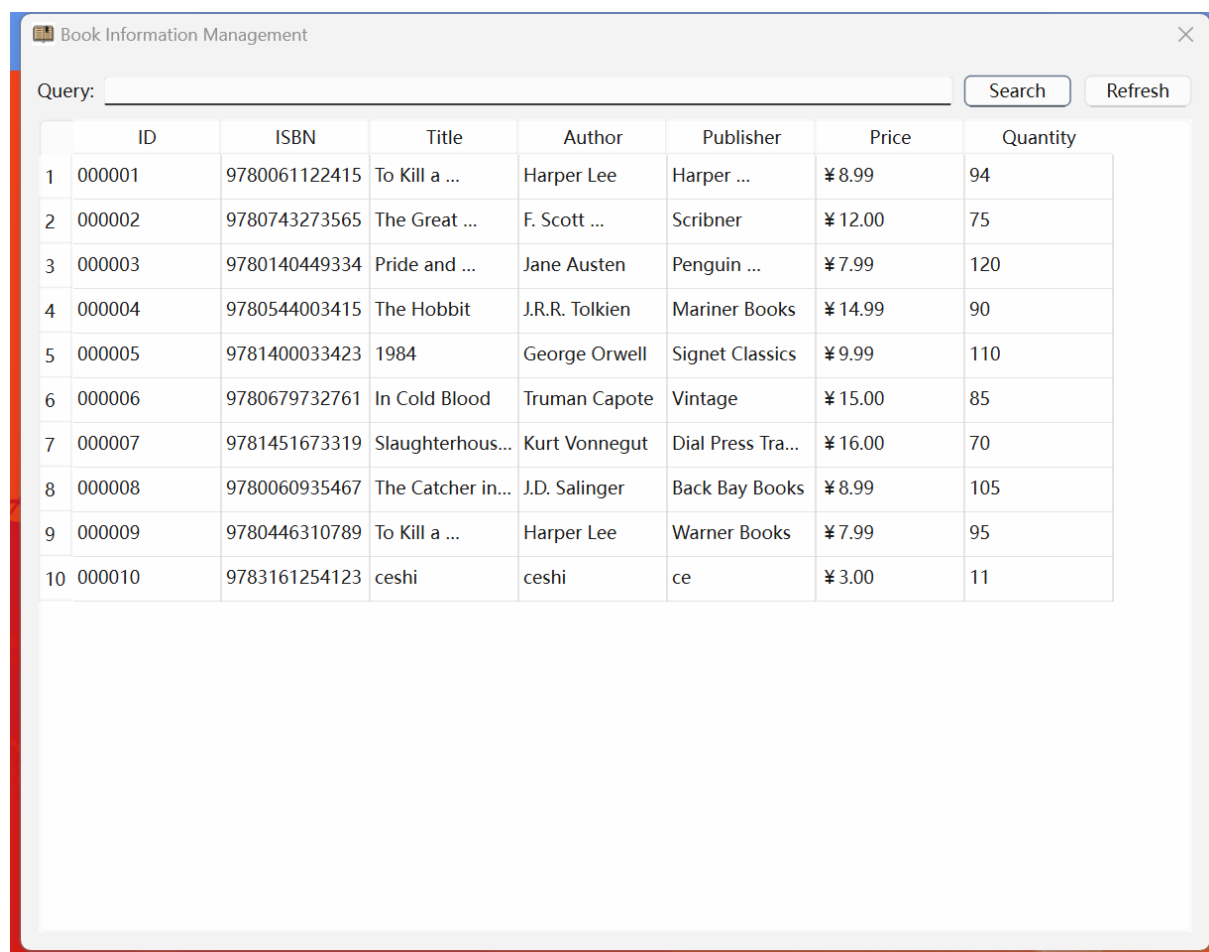
2.3 人员信息管理

User Information				
	ID	Name	Gender	Age
1	001	James	男	19
2	002	Marry	女	22
3	003	Mike	男	21
4	004	Jerry	女	23
5	005	Linda	女	21

为了方便以及高效，实现人员信息管理，因为涉及太多的槽函数，我们还是在后端进行操作，用户可以实时更新表格中的内容，更新完成后，连接槽函数到辅助函数，从后端完成对数据库信息的更新：

```
// 更新数据库中的数据
MainWindow mainWindow;
mainWindow.createConnectionByName("Connect2");
QSqlDatabase db = mainWindow.getConnectionByName("Connect2");
QSqlQuery query(db);
query.prepare("UPDATE librarian SET " + columnName + " = :newValue WHERE id = :id");
query.bindValue(":newValue", newValue);
query.bindValue(":id", userTableWidget->item(row, 0)->text()); // 根据 id 更新相应的行
qDebug() << columnName << " " << newValue;
if (!query.exec()) {
    |   qWarning() << "Update failed:" ;
} else {
    |   qDebug() << "User information updated successfully!";
}
}
```

2.4 库存书籍的管理



	ID	ISBN	Title	Author	Publisher	Price	Quantity
1	000001	9780061122415	To Kill a ...	Harper Lee	Harper ...	¥ 8.99	94
2	000002	9780743273565	The Great ...	F. Scott ...	Scribner	¥ 12.00	75
3	000003	9780140449334	Pride and ...	Jane Austen	Penguin ...	¥ 7.99	120
4	000004	9780544003415	The Hobbit	J.R.R. Tolkien	Mariner Books	¥ 14.99	90
5	000005	9781400033423	1984	George Orwell	Signet Classics	¥ 9.99	110
6	000006	9780679732761	In Cold Blood	Truman Capote	Vintage	¥ 15.00	85
7	000007	9781451673319	Slaughterhous...	Kurt Vonnegut	Dial Press Tra...	¥ 16.00	70
8	000008	9780060935467	The Catcher in...	J.D. Salinger	Back Bay Books	¥ 8.99	105
9	000009	9780446310789	To Kill a ...	Harper Lee	Warner Books	¥ 7.99	95
10	000010	9783161254123	ceshi	ceshi	ce	¥ 3.00	11

同样的，还是这个实时更新的表格，新增了一个关键词搜索的实现：

Book Information Management

Query: Harper Lee

Search

Refresh

	ID	ISBN	Title	Author	Publisher	Price	Quantity
1	000001	9780061122415	To Kill a ...	Harper Lee	Harper ...	¥8.99	94
2	000009	9780446310789	To Kill a ...	Harper Lee	Warner Books	¥7.99	95
3							
4							
5							
6							
7							
8							
9							
10							

```

// 连接查询按钮的槽函数
connect(searchButton, &QPushButton::clicked, [=]() {
    QString queryText = queryLineEdit->text();
    QString queryString = "SELECT * FROM public.inventory WHERE ";
    queryString += "id LIKE '%" + queryText + "%' OR ";
    queryString += "\"ISBN\" LIKE '%" + queryText + "%' OR ";
    queryString += "title LIKE '%" + queryText + "%' OR ";
    queryString += "author LIKE '%" + queryText + "%' OR ";
    queryString += "publisher LIKE '%" + queryText + "%'";

    // 执行查询操作
    QSqlQuery query(db);
    if (query.exec(queryString)) {
        // 清空表格内容
        bookTableWidget->clearContents();

        // 遍历查询结果，并更新表格内容
        int row = 0;
        while (query.next()) {
            for (int column = 0; column < bookTableWidget->columnCount(); ++column) {
                QTableWidgetItem *item = new QTableWidgetItem(query.value(column).toString());
                bookTableWidget->setItem(row, column, item);
            }
            ++row;
        }
    } else {
        qDebug() << "Query failed:";
    }
});

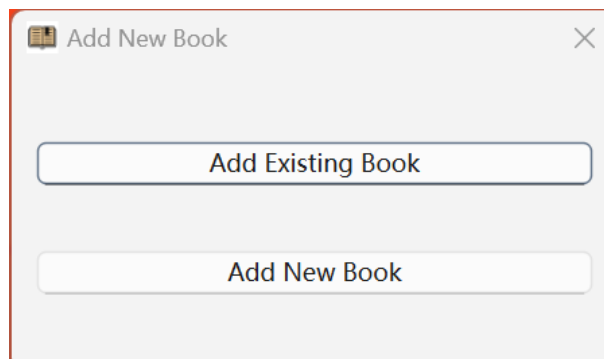
```

为了便捷与高效，实时修改表格更新到数据库的实现完全套用了上面人员信息实时更改的代码，是的，这个代码在后面所有功能实现中都可以使用。

2.5 进货，付款与退货

进货的要求有两个，

1. 添加现有书籍
2. 添加没有的书籍



给用户提供这两个选项，如果是现有书籍，用户只需输入 ISBN 码，需要的数量以

Existing Book

Book ID:

prime cost:

Quantity: 0

confirm

Add New Book

ISBN:

Title:

Author:

Publisher:

Price: 0.00

Quantity: 0

Confirm

Cancel

及进货的价格即可，如果添加新书，则需完善所有书籍的具体信息。添加完毕后，将书籍信息加入 `purchase` 表中，状态设置为未付款，等待后续操作然后，两个按钮，分别执行不同的 SQL 函数：付款与退货

Book Information Management								
Query:							Search	Refresh
	ISBN	Title	Author	Publisher	Price	Quantity	State	
1	9780061122415	To Kill a ...	Harper Lee	Harper ...	¥ 8.99	1	unpaid-for	<div>8</div> <div>return</div> <div>purchase</div>
2	9783161484100	The Great ...	F. Scott ...	Scribner	¥ 12.99	5	unpaid-for	<div>8</div> <div>return</div> <div>purchase</div>
3	9783161254123	ceshi	ceshi	ce	¥ 3.00	11	paid	

```

// 连接按钮点击事件
connect(button1, &QPushButton::clicked, [=]() {
    // 根据第一列的值执行不同的操作
    // 例如, 输出第一列的值
    qDebug() << "Button1 clicked for row " << row << ", First column value: " << firstColumnValue;
    QSqlQuery query2(db);
    query2.prepare("SELECT returnBook(:id)");
    query2.bindValue(":id", row+1);
    query2.exec();
});

connect(button2, &QPushButton::clicked, [=]() {
    // 根据第一列的值执行不同的操作
    // 例如, 输出第一列的值
    qDebug() << "Button2 clicked for row " << row << ", First column value: " << firstColumnValue;
    QSqlQuery query2(db);
    query2.prepare("SELECT purchaseBook(:id)");
    query2.bindValue(":id", row+1);
    query2.exec();
});
// 添加按钮到表格的末尾
bookTableWidget->setCellWidget(row, 7, button1);
bookTableWidget->setCellWidget(row, 8, button2);

```

两个函数的定义如下:

```

CREATE OR REPLACE FUNCTION returnBook(id_value INT)
RETURNS VOID AS
$$
BEGIN
-- 更新 purchase 表中对应 id 的 state 列为 'returned'
UPDATE purchase
SET state = 'returned'
WHERE id = id_value;
END;
$$
LANGUAGE plpgsql;

```

```

CREATE OR REPLACE FUNCTION purchaseBook(id_value INT)
RETURNS VOID AS
$$
BEGIN
-- 更新 purchase 表中对应 id 的 state 列为 'returned'
UPDATE purchase
SET state = 'paid'
WHERE id = id_value;
END;
$$
LANGUAGE plpgsql;

```

然后，执行退款与付款操作之后，purchase 数据库会被修改，为了保证完备性，我们需要一个触发器来处理付款和退货产生的结果：这个触发器会在付款后将新书添加到库存里。当然，付款会产生账单，后续的图书售卖也同样会更新账单数据库，这一点我们之后实现。

```
CREATE OR REPLACE FUNCTION update_inventory()
RETURNS TRIGGER AS $$
DECLARE
    ct INTEGER;
    tep integer;
    ctt VARCHAR;
    cttt VARCHAR;
BEGIN
    IF NEW.state = 'paid' THEN
        -- 在 inventory 表中查找这一行的 ISBN
        SELECT COUNT(*) into ct FROM inventory;

        SELECT COUNT(*) into tep FROM inventory WHERE "ISBN" = NEW."ISBN";

        select trim(to_char(ct, '000000')) into cttt;
        -- SELECT LPAD(cttt, 6, '0') INTO ctt;
        -- 如果找到了
        IF tep > 0 THEN
            -- 将对应书籍的数量加上这个数量
            UPDATE inventory SET quantity = quantity + NEW.quantity WHERE "ISBN" = NEW."ISBN";
        ELSE
            -- 新增一条书籍记录
            INSERT INTO inventory (id, "ISBN", title, author, publisher, price, quantity)
            VALUES (cttt, NEW."ISBN", NEW.title, NEW.author, NEW.publisher, NEW.price, NEW.quantity);
        END IF;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_inventory_trigger
AFTER UPDATE ON purchase
FOR EACH ROW
EXECUTE FUNCTION update_inventory();
```

2.6 销售操作

直接调用对库存书籍的查询信息，再末尾添加“sale”按钮，连接槽函数，输入售卖的数量，然后执行数据库中的 sell_book 函数：

Book Information Management

Query:

	ID	ISBN	Title	Author	Publisher	Price	Quantity	Sale
1	000001	9780061122415	To Kill a ...	Harper Lee	Harper ...	¥ 8.99	94	<input type="button" value="sale"/>
2	000002	9780743273565	The Great ...	F. Scott ...	Scribner	¥ 12.00	75	<input type="button" value="sale"/>
3	000003	9780140449334	Pride and ...	Jane Austen	Penguin ...	¥ 7.99	120	<input type="button" value="sale"/>
4	000004	9780544003415	The Hobbit	J.R.R. Tolkien	Mariner Books	¥ 14.99	90	<input type="button" value="sale"/>
5	000005	9781400033423	1984	George Orwell	Signet Classics	¥ 9.99	110	<input type="button" value="sale"/>
6	000006	9780679732761	In Cold Blood	Truman Capote	Vintage	¥ 15.00	85	<input type="button" value="sale"/>
7	000007	9781451673319	Slaughterhous...	Kurt Vonnegut	Dial Press Tra...	¥ 16.00	70	<input type="button" value="sale"/>
8	000008	9780060935467	The Catcher in...	J.D. Salinger	Back Bay Books	¥ 8.99	105	<input type="button" value="sale"/>
9	000009	9780446310789	To Kill a ...	Harper Lee	Warner Books	¥ 7.99	95	<input type="button" value="sale"/>
10	000010	9783161254123	ceshi	ceshi	ce	¥ 3.00	11	<input type="button" value="sale"/>

```
// 连接确认按钮的点击信号到处理函数
connect(confirmButton, &QPushButton::clicked, this, [=]() {
    bool ok;
    int quantity = quantityLineEdit->text().toInt(&ok);
    if (ok) {
        // 在这里处理销售操作, 使用 quantity 参数
        qDebug() << "Sale confirmed with quantity:" << quantity<<ColumnValue;
        QSqlQuery query2(db);
        query2.prepare("SELECT sell_book(:quantity, :ColumnValue);");
        query2.bindValue(":quantity",quantity);
        query2.bindValue(":ColumnValue",ColumnValue);
        query2.exec();
        saleDialog->close();
    } else {
        qDebug() << "Invalid quantity input!";
    }
});
```

```
CREATE OR REPLACE FUNCTION sell_book(qt integer, isbn varchar(255)) RETURNS VOID AS $$
BEGIN
    -- 将 inventory 表中对应 ISBN 的书籍数量减去给定的数量
    UPDATE inventory SET quantity = quantity - qt WHERE "ISBN" = isbn;
END;
```



```
$$ LANGUAGE plpgsql;
```

2.7 产生以及查看账单

View Bill

Start Time:
2000-01-01 00:00:00

End Time:
2025-01-01 00:00:00

Query

	Time	Income	Remark
1	2024-05-09T19:22:23.999	-33	purchase ceshi * 11

每次售卖书籍或是购买书籍付款之后，都要在账单表里面产生一条记录，同时记录下此时的时间，我们使用触发器来实现：

```
CREATE OR REPLACE FUNCTION update_bill_trigger_function()
RETURNS TRIGGER AS $$
DECLARE
    cur_time TIMESTAMP;
    bill_income NUMERIC;
    bill_remark TEXT;
BEGIN
    -- 如果状态变为 'paid', 则插入记录到 bill 表中
    IF NEW.state = 'paid' THEN
        -- 获取当前时间
        cur_time := CURRENT_TIMESTAMP;
```

```

bill_income := OLD.quantity * OLD.price;
bill_remark := CONCAT('purchase ', OLD.title, ' * ', OLD.quantity);
INSERT INTO bill ("time", income, remark)
VALUES (cur_time, -bill_income, bill_remark);
END IF;
RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER purchase_bill_trigger
AFTER UPDATE ON purchase
FOR EACH ROW
EXECUTE FUNCTION update_bill_trigger_function();

```

然后，实现在时间区间内查询的操作，是在原有的查询框上面改进，输入两个日期时间格式的查询框，返回在这个值内的账单记录：

```

// 执行查询操作
MainWindow mainWindow;
mainWindow.createConnectionByName("Connect2");
QSqlDatabase db = mainWindow.getConnectionByName("Connect2");
QSqlQuery query(db);
query.prepare("SELECT * FROM bill WHERE time BETWEEN :startTime AND :endTime");
query.bindValue(":startTime", startTime);
query.bindValue(":endTime", endTime);

```

到此结束，我们已经完成图书销售系统的设计