

电子科技大学

实验报告

一、实验一：图书销售管理系统数据库 SQL 应用编程

二、实验室名称：

实验时间：2020-05-25

三、实验目的

针对图书销售管理数据库开发，了解 SQL 语言 DDL、DML、DQL 类型语句在数据库操作访问中的应用方法，培养数据库 SQL 编程访问能力。同时也掌握基本的数据库触发器、存储过程 SQL 编程方法，培养数据库后端编程能力。本实验完成图书销售管理系统数据库的 SQL 数据操作访问和后端数据处理功能。

四、实验原理

1、创建数据库。PostgreSQL 创建数据库可以用以下三种方式：使用 CREATE DATABASE SQL 语句来创建；使用 createdb 命令来创建；使用 pgAdmin 工具。其中 createdb 是命令 CREATE DATABASE 的封装，可以添加可选参数对数据库进行简单配置。

2、编写存储过程。存储过程（Stored Procedure）是在大型数据库系统中，一组为了完成特定功能的 SQL 语句集，存储在数据库中，首次编译后再次调用不需要再次编译，用户通过指定存储过程的名字并给出参数（如果有）来执行它。它是数据库中的一个重要对象，任何一个设计良好的数据库应用程序都应该用到存储过程。PostgreSQL 对存储过程的描述是：存储过程和用户自定义函数（UDF）是 SQL 和过程语句的集合，它存储于数据库服务器并能被 SQL 接口调用。例如：

```
/* 存储过程函数 */
CREATE OR REPLACE FUNCTION add(a INTEGER, b NUMERIC)
RETURNS NUMERIC
AS $$
    SELECT a+b;
$$ LANGUAGE SQL;
/* 调用存储过程 */
SELECT add(1,2);
```

```
SELECT * FROM add(1,2);

/* 调用结果 */
    add
-----
     3
(1 row)
```

3、编写视图。视图是从一个或者多个表中导出的，它的行为与表非常相似，但是视图是一个虚拟表。PostgreSQL 视图是只读的，因此可能无法在视图上执行 DELETE、INSERT 或 UPDATE 语句。但是可以在视图上创建一个触发器，当尝试 DELETE、INSERT 或 UPDATE 视图时触发，需要做的动作在触发器内容中定义。例如：

```
/*假设天气记录和城市为止的组合列表对我们的应用有用，但我们又不想每次需要使用它时都敲入整个
查询。我们可以在该查询上创建一个视图，这会为该查询一个名字，我们可以像使用一个普通表一样来
使用它*/
CREATE VIEW myview AS
SELECT city, temp_lo, temp_hi, prcp, date, location
FROM weather, cities
WHERE city = name;

/* 调用视图 */
SELECT * FROM myview;
```

4、编写触发器。一个触发器是一个声明，他不需要手动开启，只需要在一个预定义事件发生时，PostgreSQL 就会自动调用。在创建触发器之前，首先需要创建一个触发器函数。它是指一个没有参数并且返回 trigger 类型的函数。例如：

```
/* 简单的触发器函数模板 */
CREATE FUNCTION fun_name() RETURNS TRIGGER AS $$
    BEGIN
        函数体
    END;
$$ LANGUAGE plpgsql

/* 创建触发器 */
CREATE TRIGGER 触发器名 BEFORE | AFTER 触发事件
```

ON 表名 FOR EACH ROW EXECUTE PROCEDURE 触发器函数;

五、实验内容

使用 pgAdmin4 数据库管理工具对图书销售管理系统数据库进行 SQL 编程操作，并完成触发器、存储过程后端编程，具体实验内容如下：

- 1.在数据库服务器中，执行 SQL 创建图书销售管理系统数据库 BookSale。
- 2.在数据库 BookSale 中，执行 SQL 创建数据库表、视图、索引等对象。
- 3.在数据库 BookSale 中，执行 SQL 进行数据增、删、查、改访问操作。
- 4.在数据库 BookSale 中，采用 PL/pgSQL 语言编写存储过程函数 Pro_CurrentSale，实现当日图书销售量及销售金额汇总统计。
- 5.在数据库 BookSale 中，采用 PL/pgSQL 语言编写过程语句块，实现对存储过程函数 Pro_CurrentSale 的调用，并输出统计结果。
- 6.在数据库 BookSale 中，采用 PL/pgSQL 语言编写编写图书销售表 Insert 触发器 Tri_InsertSale，实现图书库存数据同步修改处理。
- 7.在数据库 BookSale 中，对图书销售表 Insert 触发器 Tri_InsertSale 程序进行功能验证。
- 8.在数据库 BookSale 中，创建存储过程函数实现图书销售数量和金额统计。

在实验计算机上，利用 pgAdmin4 数据库管理工具及 SQL、PL/pgSQL 语言，完成图书销售管理系统数据库应用编程操作，同时记录实验过程的步骤、操作、运行结果界面等数据，为撰写实验报告提供素材。

六、实验设备及环境

“数据库原理及应用”实验所涉及的机房硬件设备为 pc 计算机、服务器以及网络环境，pc 计算机与服务器在同一局域网络。

操作系统： Windows 10

管理工具： pgAdmin4

DBMS 系统： PostgreSQL 11

七、实验步骤

1. 使用 PowerDesigner 建模工具对 Book_Sale 数据库进行建模设计；
2. 利用 PowerDesigner 建模工具导出建模设计的 SQL，在 pgadmin4 管理界面导入 SQL 以创建数据库；
3. 利用 INSERT、UPDATE 等功能在数据库中插入或修改预设数据；
4. 编写存储过程函数 Pro_CurrentSale，实现当日图书销售量及销售金额汇总统计；
5. 编写视图封装存储过程函数，调用视图输出统计结果；

6. 编写编写图书销售表 Insert 触发器函数和触发器 Tri_InsertSale，实现图书库存数据同步修改处理；
7. 在图书销售表 INSERT 数据以触发 Tri_InsertSale，对触发器功能进行验证；
8. 编写存储过程函数，实现对所有图书的销售量及销售金额汇总统计；
9. 编写视图封装该存储过程函数，调用视图输出统计结果。

八、实验数据及结果分析

使用 PowerDesigner 建模工具对本系统进行分析建模，如图 1-1 所示。初步设计系统应包含四个表：图书类别表、图书表、图书库存表和图书销售表。图书类别表的主键分类编号作为图书表的外键；图书表的主键图书编号是图书库存表的主键和外键，也是图书销售表的外键；图书销售表的主键为订单编号。

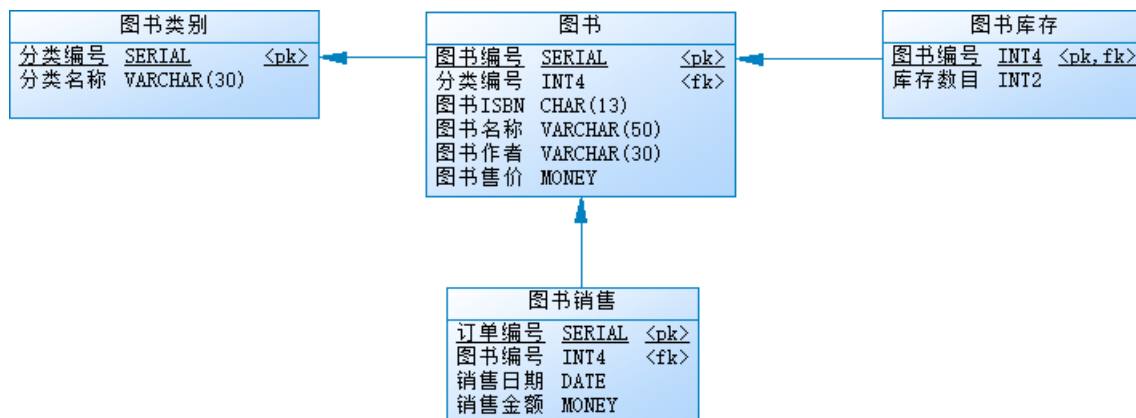


图 1-1 BookSale 数据库物理模型

在 pgadmin4 管理系统创建数据库 BookSale，设置所有者为 postgres，编码为 UTF8，不设置连接数上限，如代码 1-1 所示。

代码 1-1 创建 BookSale 数据库

```

CREATE DATABASE "BookSale"

WITH

OWNER = postgres

ENCODING = 'UTF8'

CONNECTION LIMIT = -1;
  
```

将 BookSale 的物理模型转换为数据库代码，如代码 1-2 所示，并导入 BookSale 数据库中，结果如图 1-2 所示。

代码 1-2 BookSale 数据库架构代码

```

/*=====*/
/* DBMS name:      PostgreSQL 9.x
/*=====*/

drop table BOOK;
  
```

```
drop table BOOK_CATEGORY;
```

```
drop table BOOK_INVENTORY;
```

```
drop table BOOK_SALES;
```

```
/*=====*/
```

```
/* Table: BOOK */
```

```
/*=====*/
```

```
create table BOOK (  
    Book_ID          SERIAL          not null,  
    Category_ID      INT4            not null,  
    Book_ISBN        CHAR(13)        not null,  
    Book_Name        VARCHAR(50)      not null,  
    Book_Author      VARCHAR(30)      not null,  
    Book_Price       MONEY            not null,  
    constraint PK_BOOK primary key (Book_ID)  
);
```

```
/*=====*/
```

```
/* Table: BOOK_CATEGORY */
```

```
/*=====*/
```

```
create table BOOK_CATEGORY (  
    Category_ID      SERIAL          not null,  
    Category_Name    VARCHAR(30)     not null,  
    constraint PK_BOOK_CATEGORY primary key (Category_ID)  
);
```

```
/*=====*/
```

```
/* Table: BOOK_INVENTORY */
```

```
/*=====*/
```

```
create table BOOK_INVENTORY (  
    Book_ID          INT4            not null,  
    Book_Inventory   INT2            not null,  
    constraint PK_BOOK_INVENTORY primary key (Book_ID)  
);
```

```
/*=====*/
/* Table: BOOK_SALES */
/*=====*/

create table BOOK_SALES (
    Sales_ID          SERIAL          not null,
    Book_ID           INT4            not null,
    Sales_Date        DATE            not null,
    Sales_Money       MONEY           not null,
    constraint PK_BOOK_SALES primary key (Sales_ID)
);

alter table BOOK
    add constraint FK_BOOK_REFERENCE_BOOK_CAT foreign key (Category_ID)
        references BOOK_CATEGORY (Category_ID)
        on delete restrict on update restrict;

alter table BOOK_INVENTORY
    add constraint FK_BOOK_INV_REFERENCE_BOOK foreign key (Book_ID)
        references BOOK (Book_ID)
        on delete restrict on update restrict;

alter table BOOK_SALES
    add constraint FK_BOOK_SAL_REFERENCE_BOOK foreign key (Book_ID)
        references BOOK (Book_ID)
        on delete restrict on update restrict;
```

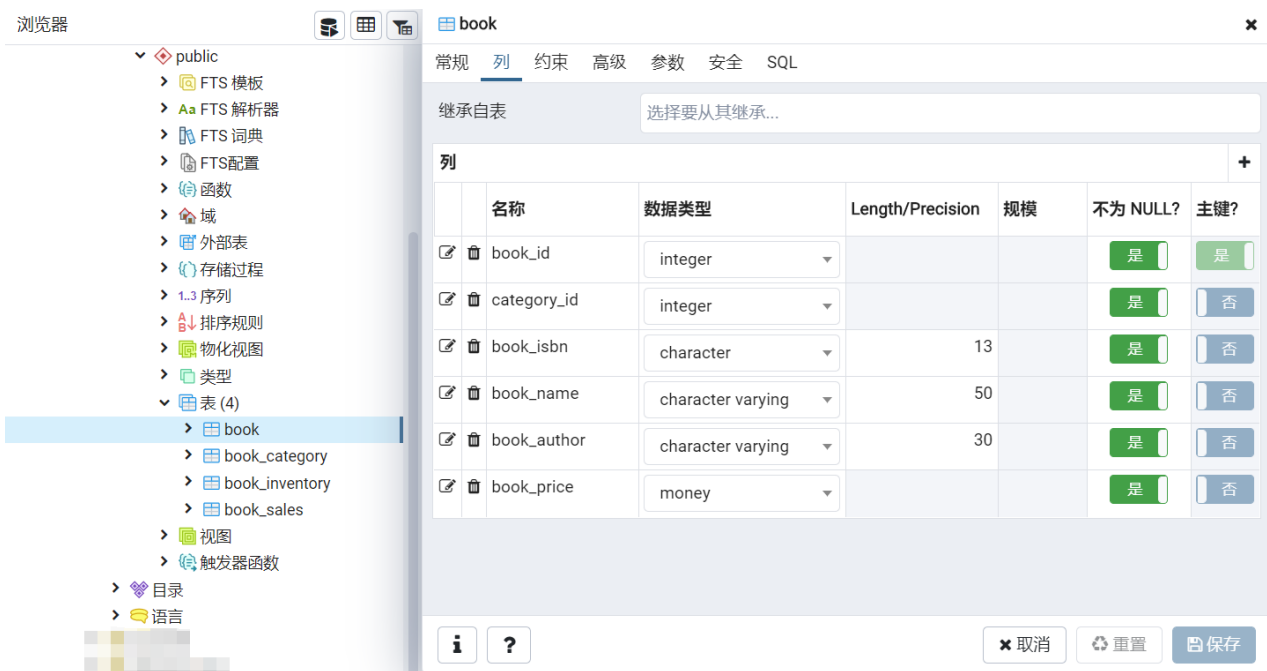


图 1-2 导入 SQL 结果

在 BookSale 数据库的四个表中插入测试数据，如代码 1-3 所示，结果如图 1-3 至 1-5 所示。

代码 1-3 插入测试数据

```
INSERT INTO public.book_category(
    category_id, category_name)
VALUES (1, '科幻');
```

```
INSERT INTO public.book_category(
    category_id, category_name)
VALUES (2, '哲学');
```

```
INSERT INTO public.book_category(
    category_id, category_name)
VALUES (3, '传记');
```

```
INSERT INTO public.book_category(
    category_id, category_name)
VALUES (4, '心理学');
```

```
INSERT INTO public.book_category(
    category_id, category_name)
```



```
VALUES (5, '小说');
```

```
INSERT INTO public.book(  
    category_id, book_isbn, book_name, book_author, book_price)  
VALUES (1, '9787544737579', '凡尔纳作品精选：从地球到月球', '儒尔·凡尔纳', '22');
```

```
INSERT INTO public.book(  
    category_id, book_isbn, book_name, book_author, book_price)  
VALUES (2, '9787547734315', '人间值得', '中村恒子', '24.5');
```

```
INSERT INTO public.book(  
    category_id, book_isbn, book_name, book_author, book_price)  
VALUES (3, '9787544276986', '你当像鸟飞往你的山', '塔拉·韦斯特弗', '29.5');
```

```
INSERT INTO public.book(  
    category_id, book_isbn, book_name, book_author, book_price)  
VALUES (4, '9787508095226', '非暴力沟通', '马歇尔·卢森堡', '29.4');
```

```
INSERT INTO public.book(  
    category_id, book_isbn, book_name, book_author, book_price)  
VALUES (5, '9787530219218', '人生海海', '麦家', '27.5');
```

```
INSERT INTO public.book(  
    category_id, book_isbn, book_name, book_author, book_price)  
VALUES (5, '9787530212837', '穆斯林的葬礼', '霍达', '34.4');
```

```
INSERT INTO public.book(  
    category_id, book_isbn, book_name, book_author, book_price)  
VALUES (1, '9787536692930', '三体1 地球往事', '刘慈欣', '14');
```

```
INSERT INTO public.book(  
    category_id, book_isbn, book_name, book_author, book_price)  
VALUES (1, '9787536484276', '球状闪电 典藏版', '刘慈欣', '16.8');
```

```
INSERT INTO public.book_inventory(  
    book_id, book_inventory)
```

```
VALUES (1, 64);
```

```
INSERT INTO public.book_inventory(  
    book_id, book_inventory)  
VALUES (2, 33);
```

```
INSERT INTO public.book_inventory(  
    book_id, book_inventory)  
VALUES (3, 2);
```

```
INSERT INTO public.book_inventory(  
    book_id, book_inventory)  
VALUES (4, 66);
```

```
INSERT INTO public.book_inventory(  
    book_id, book_inventory)  
VALUES (5, 120);
```

```
INSERT INTO public.book_inventory(  
    book_id, book_inventory)  
VALUES (6, 61);
```

```
INSERT INTO public.book_inventory(  
    book_id, book_inventory)  
VALUES (7, 1);
```

```
INSERT INTO public.book_inventory(  
    book_id, book_inventory)  
VALUES (8, 99);
```

	category_id [PK] integer	category_name character varying (30)
1	1	科幻
2	2	哲学
3	3	传记
4	4	心理学
5	5	小说

图 1-3 插入图书类别结果

	book_id [PK] integer	category_id integer	book_isbn character (13)	book_name character varying (50)	book_author character varying (30)	book_price money
1	1	1	9787544737579	凡尔纳作品精选: 从地球到...	儒尔·凡尔纳	\$22.00
2	2	2	9787547734315	人间值得	中村恒子	\$24.50
3	3	3	9787544276986	你当像鸟飞往你的山	塔拉·韦斯特弗	\$29.50
4	4	4	9787508095226	非暴力沟通	马歇尔·卢森堡	\$29.40
5	5	5	9787530219218	人生海海	麦家	\$27.50
6	6	5	9787530212837	穆斯林的葬礼	霍达	\$34.40
7	7	1	9787536692930	三体1 地球往事	刘慈欣	\$14.00
8	8	1	9787536484276	球状闪电 典藏版	刘慈欣	\$16.80

图 1-4 插入图书结果

	book_id [PK] integer	book_inventory smallint
1	1	64
2	2	33
3	3	2
4	4	66
5	5	120
6	6	61
7	7	1
8	8	99

图 1-5 插入图书库存结果

采用 PL/pgSQL 语言编写存储过程函数 Pro_CurrentSale, 实现查询当日图书销售量及销售金额汇总统计, 如代码 1-4 所示, 返回值为存储有 Book_Sales 和 Sales_Money 的表, 其中 CURRENT_DATE 获取当前日期。

代码 1-4 存储过程函数 Pro_CurrentSale

```
CREATE OR REPLACE FUNCTION Pro_CurrentSale()
RETURNS TABLE (
    "Book_Sales" BIGINT,
    "Sales_Money" MONEY
) AS $BODY$
BEGIN
    RETURN QUERY
    SELECT
        COUNT(*),
        SUM(sales_money)
    FROM
        public.book_sales
    WHERE
        sales_date = CURRENT_DATE;
END;
$BODY$ LANGUAGE plpgsql
```

封装视图 Pro_CurrentSale_View 实现对存储过程函数 Pro_CurrentSale 的调用，如代码 1-5 所示。在图书销售表中插入数据如图 1-6 所示，查询当日图书销售量及销量金额汇总结果如图 1-7 所示，其中 Book_Sales 表示当日图书销售量，Sales_Money 表示销量金额。

代码 1-5 视图 Pro_CurrentSale_View

```
CREATE VIEW Pro_CurrentSale_View AS
SELECT * FROM Pro_CurrentSale();
```



	 sales_id [PK] integer	 book_id integer	 sales_date date	 sales_money money
1	1	1	2020-05-25	\$22.00
2	2	1	2020-05-25	\$22.00
3	3	1	2020-05-25	\$22.00
4	4	3	2020-05-25	\$29.50
5	5	2	2020-05-25	\$24.50
6	6	6	2020-05-25	\$34.40
7	7	6	2020-05-25	\$34.40
8	8	6	2020-05-26	\$34.40
9	9	6	2020-05-26	\$34.40
10	10	7	2020-05-26	\$14.00
11	11	7	2020-05-26	\$14.00
12	12	7	2020-05-26	\$14.00
13	13	7	2020-05-26	\$14.00
14	14	8	2020-05-26	\$16.80
15	15	8	2020-05-26	\$16.80

图 1-6 插入图书销售结果



	 Book_Sales bigint	 Sales_Money money
1	8	\$158.40

图 1-7 查询当日销售结果

采用 PL/pgSQL 语言编写图书销售表 Insert 触发器函数 Tri_InsertSale_Function() 和触发器 Tri_InsertSale，实现图书库存表数据同步修改处理，如代码 1-6 和代码 1-7 所示。

代码 1-6 触发器函数 Tri_InsertSale_Function()

```
CREATE OR REPLACE FUNCTION Tri_InsertSale_Function()
RETURNS TRIGGER AS $BODY$
BEGIN
    IF (TG_OP=' INSERT' ) THEN
        UPDATE public.book_inventory
```

```

        SET book_inventory=book_inventory-1
        WHERE book_id=NEW.book_id;
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$BODY$ LANGUAGE plpgsql

```

代码 1-7 触发器 Tri_InsertSale

```

CREATE TRIGGER Tri_InsertSale
BEFORE INSERT ON public.book_sales
FOR EACH ROW EXECUTE PROCEDURE Tri_InsertSale_Function();

```

接下来在图书销售表中插入新的销售数据，如代码 1-8 所示。查询图书库存表结果，如图 1-8 所示，book_id（图书编号）为 1 的图书的 book_inventory（图书库存）从原先的 64（本）变成了现在的 63（本），实现了 Insert 触发器 Tri_InsertSale 数据同步修改功能。

代码 1-8 插入销售数据

```

INSERT INTO public.book_sales(
    book_id, sales_date, sales_money)
VALUES (1, CURRENT_DATE, 22);

```

	book_id [PK] integer	book_inventory smallint
1	2	33
2	3	2
3	4	66
4	5	120
5	6	61
6	7	1
7	8	99
8	1	63

图 1-8 测试触发器结果

创建存储过程函数 BookSale_Statistics() 实现对图书销售数量和金额统计，封装视图

BookSale_Statistics_View 实现对该存储过程函数的调用，如代码 1-9 和 1-10 所示。查询视图得到的结果如图 1-9 所示。至此，本实验全部要求均已基本实现。

代码 1-9 存储过程函数 BookSale_Statistics()

```
CREATE OR REPLACE FUNCTION BookSale_Statistics()  
RETURNS TABLE(  
    "图书编号" INT,  
    "图书名称" VARCHAR(50),  
    "销售数量" BIGINT,  
    "销售金额" MONEY  
) AS $BODY$  
BEGIN  
    RETURN QUERY  
    SELECT  
        s.book_id, b.book_name, count(s.book_id), sum(s.sales_money)  
    FROM  
        public.book_sales AS s, public.book AS b  
    WHERE  
        s.book_id=b.book_id  
    GROUP BY s.book_id, b.book_name  
    ORDER BY s.book_id;  
END;  
$BODY$ LANGUAGE plpgsql
```

代码 1-10 视图 BookSale_Statistics_View

```
CREATE VIEW BookSale_Statistics_View AS  
SELECT * FROM BookSale_Statistics();
```

	图书编号 integer	图书名称 character varying	销售数量 bigint	销售金额 money
1	1	凡尔纳作品精选：从地...	4	\$88.00
2	2	人间值得	1	\$24.50
3	3	你当像鸟飞往你的山	1	\$29.50
4	6	穆斯林的葬礼	4	\$137.60
5	7	三体1 地球往事	4	\$56.00
6	8	球状闪电 典藏版	2	\$33.60

图 1-9 查询图书销售结果

九、总结及心得体会

本实验的关键技术包括但不限于 PostgreSQL 中的存储过程、触发器、视图等内容。

存储过程类似于高级编程语言中的函数，可以有参数。在存储过程中利用 `plpgsql`（当然也可以使用其它 `LANGUAGE`）提供的基本指令对数据表进行相关处理操作，最终返回一个结果。利用存储过程，可以减少应用与数据库服务器的通信开销，提高整体性能；一次编译，多次调用，提高了执行性能；使一套业务逻辑可以共用，减少应用程序开发复杂度等。

触发器函数是一种特殊的存储过程，其返回值为 `TRIGGER`，是创建触发器必不可少的一部分。触发器不由用户直接调用，它在指定的表中的数据发生变化时自动生效，唤醒调用触发器以响应 `INSERT`、`UPDATE` 或 `DELETE` 语句。利用触发器，可通过数据库中的相关表实现级联更改；通过级联引用完整性约束可以更有效地执行这些更改；实现更复杂的数据约束等。

视图是一种虚拟表，本质是通过相关的名称存储在数据库中的一个 PostgreSQL 语句。利用视图，用户或用户组可以更自然或直观查找结构数据；可以限制数据访问，用户只能看到有限的数数据，而不是完整的表；还可以汇总各种表中的数据，用于生成报告。

本实验主要考察了 SQL 编程技术，很好地巩固了 SQL 编程基础，为日后的数据库应用开发奠定了很好的基础。在本实验中还有很多可以改进的地方，比如新增销售数据前，触发器可以先判断该书本的库存是否为零，若为零，则拒绝添加销售数据，并返回错误报告等。

电子科技大学

实验报告

一、实验二：图书销售管理系统数据库安全管理

二、实验室名称：

实验时间：2020-05-26

三、实验目的

了解该 DBMS 系统对数据库管理的内容与方法，特别是理解数据库安全机制和作用，以及 PostgreSQL 数据库角色管理、用户管理、权限管理的基本方法，培养数据库管理能力。在图书销售管理系统数据库中，创建必要的角色和用户，并完成上述角色与用户的权限管理。

四、实验原理

1、为何需要设置权限。如果创建了一个数据库对象，那么你就成为它的所有者。缺省时代，一般只有数据库的所有者可以在数据库上做任何事情。为了允许其它用户使用它，必须赋予他们权限。不过拥有超级用户权限的用户总是可以访问任何数据库对象。

2、权限类型。在 PostgreSQL 数据库中有多种不同的权限：SELECT，INSERT，UPDATE，DELETE，RULE，REFERENCES，TRIGGER，CREATE，TEMPORARY，EXECUTE，USAGE。适用于特定对象的权限因对象类型（表、函数等）而异。而修改或者删除一个对象的权限永远是所有者独有的权限。

3、赋予权限。使用 GRANT 命令。如果 Sam 是一个现存的用户，而 accounts 是一个已经存在的表，更新表的权限可以用下面的命令赋予：

```
GRANT UPDATE ON accounts TO Sam;
```

4、撤销权限。使用 REVOKE 命令。撤销 Sam 对 accounts 表的所有权限可以使用下面的命令：

```
REVOKE ALL ON accounts FROM Sam;
```

5、授予权限的补充。只有对象所有者（或者超级用户）可以赋予或者撤销对象的权限。但是，我们可以赋予一个“with grant option”权限，这样就给接受权限的人以授予该权限给其它人的权限。如果授予选项后来被撤销，那么所有那些从这个接受者接受了权限的用户（直接或者通过级连的授权）都将失去该权限。

五、实验内容

使用 pgAdmin4 数据库管理工具对图书销售管理系统数据库进行数据库安全管理，具体实验内容如下：

- 1.针对图书销售管理系统数据库，设计数据存取权限控制模型。
- 2.在数据库中，创建客户（R_Customer）、商家（R_Seller）角色。
- 3.在数据库中，根据业务规则为客户（R_Customer）、商家（R_Seller）角色赋予数据库对象权限。
- 4.在数据库中，分别创建客户用户 U_Customer、商家用户 U_Seller。
- 5.分别为客户用户 U_Customer、商家用户 U_Seller 分派客户（R_Client）、商家（R_Seller）角色。
- 6.分别以客户用户 U_Customer、商家用户 U_Seller 身份访问图书销售管理数据库，验证所实现数据存取权限控制模型的正确性。

在实验计算机上，利用 pgAdmin4 数据库管理工具及 SQL 语句，完成图书销售管理系统数据库安全管理，同时记录实验过程的步骤、操作、运行结果界面等数据，为撰写实验报告提供素材。

六、实验设备及环境

“数据库原理及应用”实验所涉及的机房硬件设备为 pc 计算机、服务器以及网络环境，pc 计算机与服务器在同一局域网。

操作系统： Windows 10

管理工具： pgAdmin4

DBMS 系统： PostgreSQL 11

七、实验步骤

1. 分析设计数据库的用户访问数据表权限并建模；
2. 创建新的数据库角色；
3. 对每一个数据表设置角色访问操作权限；
4. 创建新的数据库用户；
5. 赋予新的数据库用户以角色的身份；
6. 利用 SQL CMD 工具，以不同的用户身份登入数据库；

7. 对数据表进行 SELECT、INSERT、DELETE、UPDATE 等操作，观察不同用户的操作结果。

八、实验数据及结果分析

针对图书销售管理系统数据库，设置数据存储权限控制模型，如表 2-1 所示。其中普通用户只能（受限地）查询图书表、图书分类表和图书库存表；管理员用户则拥有对本数据库的绝对管理权限。

表 2-1 BookSale 数据库存储权限控制模型

Role	Table	INSERT	DELETE	UPDATE	SELECT	ELSE	WITH GRANT OPTION
普通用户	book				√		
	book_category				√		
	book_inventory				√		
	book_sales						
管理员	book	√	√	√	√	√	√
	book_category	√	√	√	√	√	√
	book_inventory	√	√	√	√	√	√
	book_sales	√	√	√	√	√	√

在 pgadmin4 管理界面创建 R_Customer 和 R_Seller 角色，代码如 2-1 所示。

代码 2-1 创建角色

```
CREATE ROLE "R_Customer" WITH
    NOLOGIN
    NOSUPERUSER
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION;

CREATE ROLE "R_Seller" WITH
    NOLOGIN
    NOSUPERUSER
    INHERIT
    NOCREATEDB
    NOCREATEROLE
    NOREPLICATION;
```

参考前文的存储权限控制模型表 2-1，其中 R_Customer 角色作为普通用户，R_Seller 角色作为管理员用户，分别为数据库中的四个表设置角色的访问操作权限，结果如图 2-1 至 2-4 所示。

book

常规

列

约束

高级

参数

安全

SQL

权限

受让人

权限

授权人

R_Customer

r

postgres

R_Seller

a*r*w*d*D*x*t*

postgres

安全标签

提供者

安全标签

i

?

取消

重置

保存

图 2-1 book 用户权限设置

book_category

常规

列

约束

高级

参数

安全

SQL

权限

受让人

权限

授权人

R_Customer

r

postgres

R_Seller

a*r*w*d*D*x*t*

postgres

安全标签

提供者

安全标签

i

?

取消

重置

保存

图 2-2 book_category 用户权限设置

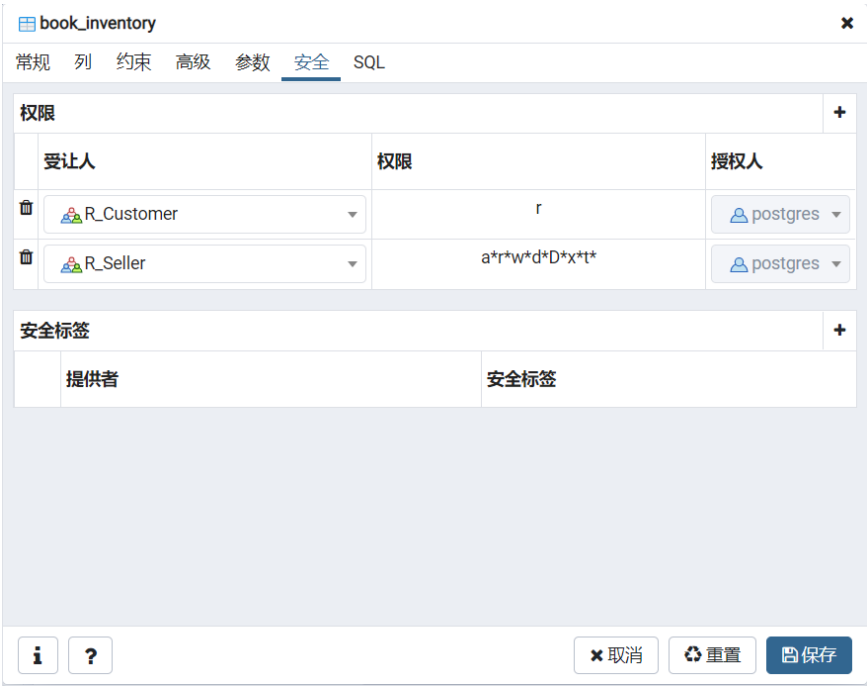


图 2-3 book_inventory 用户权限设置



图 2-4 book_sales 用户权限设置

在 pgadmin4 管理界面创建 U_Customer 和 U_Seller 用户，并分别分派 R_Client 和 R_Seller 角色，如代码 2-2 所示。当前的角色和用户如图 2-5 所示。

代码 2-2 创建用户

```
CREATE ROLE "U_Customer" WITH
  LOGIN
  NOSUPERUSER
```

```
INHERIT
NOCREATEDB
NOCREATEROLE
NOREPLICATION
ENCRYPTED PASSWORD 'md5b93bcac46fe1c19338e200b28e46acaf';
```

```
GRANT "R_Customer" TO "U_Customer";
```

```
CREATE ROLE "U_Seller" WITH
```

```
LOGIN
NOSUPERUSER
INHERIT
NOCREATEDB
NOCREATEROLE
NOREPLICATION
ENCRYPTED PASSWORD 'md5e522b55e46524f58e1776c7f9807e5c9';
```

```
GRANT "R_Seller" TO "U_Seller";
```



图 2-5 创建角色、用户结果

以 U_Customer 用户的身份登录 BookSale 数据库，对数据库中的表进行管理操作，结果如图 2-6 至 2-9 所示。因为 U_Customer 具有查询 book 表和 book_category 表的权限，如图 2-6 和图 2-9 所示。而 U_Customer 不具有删除 book 表或更新 book_inventory 表的权限，因此提示 “permission denied for table *”，如图 2-7 和图 2-8 所示。


```
BookSale=> SELECT * FROM book;
```

book_id	category_id	book_isbn	book_name	book_author	book_price
1	1	9787544737579	凡尔纳作品精选：从地球到月球	儒尔·凡尔纳	\$22.00
2	2	9787547734315	人间值得	中村恒子	\$24.50
3	3	9787544276986	你当像鸟飞往你的山	塔拉·韦斯特弗	\$29.50
4	4	9787508095226	非暴力沟通	马歇尔·卢森堡	\$29.40
5	5	9787530219218	人生海海	麦家	\$27.50
6	5	9787530212837	穆斯林的葬礼	霍达	\$34.40
7	1	9787536692930	三体1 地球往事	刘慈欣	\$14.00
8	1	9787536484276	球状闪电 典藏版	刘慈欣	\$16.80

(8 行记录)

图 2-6 Customer 查询 book 表

```
BookSale=> DELETE FROM book WHERE book_id=1;
ERROR: permission denied for table book
```

图 2-7 Customer 删除 book 表数据

```
BookSale=> UPDATE book_inventory SET book_inventory=0 WHERE book_id=2;
ERROR: permission denied for table book_inventory
```

图 2-8 Customer 更新 book_inventory 表数据

```
BookSale=> SELECT * FROM book_category;
```

category_id	category_name
1	科幻
2	哲学
3	传记
4	心理学
5	小说

(5 行记录)

图 2-9 Customer 查询 book_category 表

以 U_Seller 用户的身份登录 BookSale 数据库，对数据库中的表进行管理操作，结果如图 2-10 至图 2-13 所示。因为 U_Seller 具有管理数据库的所有权限，因此能够查询 book_sales 表（图 2-10），删除 book_sales 表的数据（图 2-11），添加新数据到 book_sales 表（图 2-12），更新 book_inventory 表数据（图 2-13）。

至此，本实验内容已基本结束。

```
BookSale=> SELECT * FROM book_sales;
```

sales_id	book_id	sales_date	sales_money
1	1	2020-05-25	\$22.00
2	1	2020-05-25	\$22.00
3	1	2020-05-25	\$22.00
4	3	2020-05-25	\$29.50
5	2	2020-05-25	\$24.50
6	6	2020-05-25	\$34.40
7	6	2020-05-25	\$34.40
8	6	2020-05-26	\$34.40
9	6	2020-05-26	\$34.40
10	7	2020-05-26	\$14.00
11	7	2020-05-26	\$14.00
12	7	2020-05-26	\$14.00
13	7	2020-05-26	\$14.00
14	8	2020-05-26	\$16.80
15	8	2020-05-26	\$16.80
16	1	2020-05-26	\$22.00

(16 行记录)

图 2-10 Seller 查询 book_sales 表

```
BookSale=> DELETE FROM book_sales WHERE sales_id=1;
DELETE 1
BookSale=> SELECT * FROM book_sales;
```

sales_id	book_id	sales_date	sales_money
2	1	2020-05-25	\$22.00
3	1	2020-05-25	\$22.00
4	3	2020-05-25	\$29.50
5	2	2020-05-25	\$24.50
6	6	2020-05-25	\$34.40
7	6	2020-05-25	\$34.40
8	6	2020-05-26	\$34.40
9	6	2020-05-26	\$34.40
10	7	2020-05-26	\$14.00
11	7	2020-05-26	\$14.00
12	7	2020-05-26	\$14.00
13	7	2020-05-26	\$14.00
14	8	2020-05-26	\$16.80
15	8	2020-05-26	\$16.80
16	1	2020-05-26	\$22.00

(15 行记录)

图 2-11 Seller 删除 book_sales 表数据

```

BookSale=> INSERT INTO book_sales VALUES (1, 1, '2020-05-24', 22);
INSERT 0 1
BookSale=> SELECT * FROM book_sales;

```

sales_id	book_id	sales_date	sales_money
2	1	2020-05-25	\$22.00
3	1	2020-05-25	\$22.00
4	3	2020-05-25	\$29.50
5	2	2020-05-25	\$24.50
6	6	2020-05-25	\$34.40
7	6	2020-05-25	\$34.40
8	6	2020-05-26	\$34.40
9	6	2020-05-26	\$34.40
10	7	2020-05-26	\$14.00
11	7	2020-05-26	\$14.00
12	7	2020-05-26	\$14.00
13	7	2020-05-26	\$14.00
14	8	2020-05-26	\$16.80
15	8	2020-05-26	\$16.80
16	1	2020-05-26	\$22.00
1	1	2020-05-24	\$22.00

(16 行记录)

图 2-12 Seller 增加 book_sales 表数据

```

BookSale=> UPDATE book_inventory SET book_inventory=233 WHERE book_id=2;
UPDATE 1

```

图 2-13 Seller 更新 book_inventory 表

九、总结及心得体会

本实验运用的关键技术包括但不限于数据库用户权限管理。

为数据库中的用户授予合适的权限是数据库安全的重要组成部分之一。可以通过 **GRANT** 命令赋予用户对某数据表进行操作的权限，也可以通过 **REVOKE** 命令撤销已赋予的权限。

在实际运用当中，应当充分考虑用户在数据库担任的角色，赋予其相应的权限，提高数据库中数据的安全性，防止恶意访问数据库中的重要内容，关系着整个数据库的健壮性。

本实验中仅涉及两个用户主体，用户权限的分配也相对极端地分为两种，实现了对数据库数据的受限访问。