

电子科技大学信息与软件工程学院

实 验 报 告

学 号 2018091618008

姓 名 袁昊男

(实验) 课程名称 数据库原理及应用

理论教师 张凤荔

实验教师 文淑华

电子科技大学

实验报告

学生姓名：袁昊男 学号：2018091618008 指导教师：张凤荔

实验地点：在线实验 实验时间：2020.06.08

一、实验室名称：信息与软件工程学院实验中心

二、实验名称：图书销售管理系统数据库 SQL 应用编程

三、实验学时：5 学时

四、实验原理：

1、数据库操作

在数据库系统中，最大的数据库对象就是数据库本身。SQL 提供了数据库对象的创建与维护语句，包括数据库创建、数据库属性修改、数据库删除等。以下分别对这些语句进行说明。

1.1 数据库创建 SQL 语句

基本语句格式为：

```
CREATE DATABASE <数据库名>
[ [ WITH ] [ OWNER [=] user_name ]      指定数据库用户
[ TEMPLATE [=] template ]              指定数据库模板
[ ENCODING [=] encoding ]              指定数据库使用的字符集编码
[ LC_COLLATE [=] lc_collate ]          指定数据库的字符排序规则
[ LC_CTYPE [=] lc_ctype ]              指定数据库的字符分类规则
[ TABLESPACE [=] tablespace_name ]    指定数据库使用的表空间
[ CONNECTION LIMIT [=] connlimit ] ] 指定数据库的并发连接数
```

其中，CREATE DATABASE 为创建数据库语句的关键词；<数据库名>为被创建数据库的标识符名称。

1.2 数据库修改 SQL 语句

使用 ALTER DATABASE 语句，可以修改数据库的属性。数据库修改语句包括更改数据库设置参数、数据库名称、数据库所有者、数据库默认表空间等。数据库属性修改的 SQL 语句格式为：

```
ALTER DATABASE <数据库名> CONNECTION LIMIT connlimit;
ALTER DATABASE <数据库名> RENAME TO <新数据库名>;
ALTER DATABASE <数据库名> OWNER TO <新所有者>;
ALTER DATABASE <数据库名> SET TABLESPACE <新表空间名>;
```

```
ALTER DATABASE <数据库名> SET 配置参数 {TO|=} {value|DEFAULT };
ALTER DATABASE <数据库名> SET 配置参数 FROM CURRENT;
ALTER DATABASE <数据库名> RESET 配置参数;
ALTER DATABASE <数据库名> RESET ALL;
```

1.3 数据库删除 SQL 语句

基本语句格式为：

```
DROP DATABASE <数据库名>;
```

其中，DROP DATABASE 为语句命令关键词；<数据库名>为数据库名称。该语句执行后，该数据库从数据库服务器中被删除。

2、数据库表操作

数据库表是数据库中最基本的操作对象。在 SQL 中，使用数据定义语言语句来完成数据表的创建、表结构修改、表删除等操作。

2.1 数据库表创建 SQL 语句

基本语句格式为：

```
CREATE TABLE <表名>
(
    <列名 1> <数据类型> [列完整性约束],
    <列名 2> <数据类型> [列完整性约束],
    <列名 3> <数据类型> [列完整性约束],
    ...
);
```

其中，CREATE TABLE 为创建表语句的关键词；<表名>为将被创建的数据库表名称。一个数据库不允许有两个表同名。在一个表中，可以定义多个列，但不允许有两个属性列同名。针对表中每个属性列，都需要指定其取值的数据类型。在进行属性列定义时，有时还需要给出该列的完整性约束。

2.2 数据库表修改 SQL 语句

基本语句格式为：

```
ALTER TABLE <表名> <修改方式>;
ALTER TABLE <表名> ADD <新列名称><数据类型>[完整性约束];
ALTER TABLE <表名> DROP COLUMN <列名>;
ALTER TABLE <表名> DROP CONSTRAINT<完整性约束名>;
ALTER TABLE <表名> RENAME TO <新表名>;
ALTER TABLE <表名> RENAME <原列名> TO <新列名>;
ALTER TABLE <表名> ALTER COLUMN <列名> TYPE<新的数据类型>;
```

2.3 数据库表删除 SQL 语句

基本语句格式为：

```
DROP TABLE <表名>;
```

其中，DROP TABLE 为数据库表删除语句的关键词；<表名>为将被删除的数据库表名称。该语句执行后，将删除指定的数据表，包括表结构和表中数

据。

3、视图操作

3.1 视图对象创建 SQL 语句

视图由一个或几个基础表（或其他视图）的 **SELECT** 查询结果创建生成。当它被创建后，被作为一种数据库对象存放在数据库中，其语句格式为：

```
CREATE VIEW <视图名>[(列名 1),(列名 2),...] AS <SELECT 查询>;
```

其中，**CREATE VIEW** 为创建视图语句的关键词；<视图名>为将被创建的视图名称。一个数据库不允许有两个视图同名。在视图名称后，可以定义组成视图的各个列名。若没有指定列名，则默认采用基础表查询结果集的所有列作为视图列。**AS** 关键词后为基础表的 **SELECT** 查询语句，其结果集为视图的数据。

3.2 视图对象删除 SQL 语句

当数据库不再需要某视图时，可以在数据库中删除该视图，其视图的删除语句格式为：

```
DROP VIEW <视图名>;
```

其中，**DROP VIEW** 为删除视图语句的关键词；<视图名>为将被删除的视图名称。

4、索引操作

4.1 索引对象创建 SQL 语句

基本语句格式为：

```
CREATE INDEX <索引名> ON <表名><(列名[,...])>;
```

其中，**CREATE INDEX** 为创建索引语句的关键词；<索引名>为在指定表中针对某列创建索引的名称。该语句执行后，系统在表中为指定列创建其列值的索引，使索引可实现数据表的快速查询。

4.2 索引对象修改 SQL 语句

使用 SQL 语句可以对索引对象进行修改操作，其中索引换名修改语句格式为：

```
ALTER INDEX <索引名> RENAME TO <新索引名>;
```

其中，**ALTER INDEX** 为索引对象修改语句的关键词；<索引名>为在数据库表中创建索引的名称；**RENAME TO** 为索引换名关键词。当该语句执行后，原有索引被换名为新名称。

4.3 索引对象删除 SQL 语句

基本语句格式为：

```
DROP INDEX <索引名>;
```

其中，**DROP INDEX** 为删除索引语句的关键词；<索引名>为被指定的索引名称。该语句执行后，系统将从表中删除该索引。

5、数据操作

5.1 数据插入 SQL 语句

每执行一个 **INSERT INTO** 语句，就会在表中插入一个行数据，其语句基本格式为：

```
INSERT INTO <基本表>[<列名表>] VALUES(列值表);
```

其中，**INSERT INTO** 为插入语句的关键词；<基本表>为被插入数据的数据库表；<列名表>给出在表中插入哪些列，若没有给出列名表，则为数据库表插入所有列；**VALUES** 关键词后括号中给出被插入的各个列值。

5.2 数据更新 SQL 语句

数据更新语句 **UPDATE** 是依给定条件，对数据库表中的指定数据进行更新处理，其语句基本格式为：

```
UPDATE <基本表>  
SET <列名 1>=<表达式 1> [, <列名 2>=<表达式 2>...]  
[WHERE <条件表达式>];
```

其中，**UPDATE** 为数据更新语句的关键词；<基本表>为被更新数据的数据库表；**SET** 关键词指定对哪些列设定新值；**WHERE** 关键词给出需要满足的条件表达式。

5.3 数据删除 SQL 语句

数据删除语句 **DELETE** 将从指定数据库表中删除满足条件的数据行，其语句基本格式为：

```
DELETE FROM <表名>  
[WHERE <条件表达式>];
```

其中，**DELETE FROM** 为数据删除语句的关键词；<表名>为被删除数据的数据库表；**WHERE** 关键词给出需要满足的条件表达式。

5.4 数据查询 SQL 语句

在 SQL 中，实现对数据库表进行数据查询处理的语句只有 **SELECT** 语句。虽然只有一种语句，但该类语句功能丰富、组合条件使用灵活。所有数据查询操作都可以通过 **SELECT** 语句实现，其基本语句格式为：

```
SELECT [ALL|DISTINCT] <目标列>[, <目标列>...]  
[ INTO <新表> ]  
FROM <表名>[, <表名>...]  
[ WHERE <条件表达式> ]  
[ GROUP BY <列名> [HAVING <条件表达式> ]  
[ ORDER BY <列名> [ ASC | DESC ] ];
```

6、存储过程编程

许多数据库为创建存储过程和函数提供不同命令，如 Oracle、MySQL、SQL Server 等数据库管理系统，使用 **CREATE PROCEDURE** 命令创建存储过程，使用 **CREATE FUNCTION** 命令创建函数。PostgreSQL 数据库没有为创建存储过程提供专用命令，而是通过创建数据库函数来实现存储过程的功能，后面将 PostgreSQL 的函数和存储过程统称为存储过程。

PostgreSQL 使用 **CREATE FUNCTION** 命令创建新的函数或存储过程，可以在许多语言中创建 PostgreSQL 函数，如 SQL、PL/pgSQL、C、Python 等语言。在 PostgreSQL 内置的过程控制语言 PL/pgSQL 中，创建存储过程的语句格式为：

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = }
default_expr ] [, ...] ] )
    [ RETURNS retype | RETURNS TABLE ( column_name column_type
[, ...] ) ]
    AS $$
    DECLARE
    -- 声明段
    BEGIN
    --函数体语句
    END;
    $$ LANGUAGE lang_name;
```

7、触发器编程

触发器是存储在数据库中的独立对象，它与存储过程不同，存储过程通过其他程序来启动运行或直接启动运行，而触发器由一个事件触发启动运行。也就是说，触发器是在某个事件发生时自动地隐式运行，所以启动触发器执行在某些文献里被称为触发或点火。数据库事件指对数据库的表进行 **INSERT**、**UPDATE** 及 **DELETE** 操作。

PostgreSQL 触发器可以在表、特殊的视图和外部表上定义。触发器经常用于定义逻辑比较复杂的完整性约束，或者某种业务规则的约束。其创建触发器的语法为：

```
CREATE [ CONSTRAINT ] TRIGGER name
{ BEFORE | AFTER | INSTEAD OF } { event [ OR ...] }
ON table_name
[ FROM referenced_table_name ]
[ FOR [ EACH ] { ROW | STATEMENT } ]
[ WHEN (condition) ]
EXECUTE PROCEDURE function_name ( arguments )
```

CREATE TRIGGER 创建一个新触发器。该触发器将被关联到指定的表、

视图或者外部表，并且在特定事件发生时将执行指定的函数。创建触发器时，必须指定引发触发器的事件。在 PostgreSQL 数据库中，触发器事件可以指定为 INSERT、UPDATE、DELETE 或者 TRUNCATE 之一。

五、实验目的：

针对图书借阅管理数据库开发，掌握 DDL、DML、DQL 类型 SQL 语句在数据库操作访问中的应用方法，培养数据库 SQL 应用编程能力。同时也掌握基本的数据库触发器、存储过程编程方法，培养数据库后端编程能力。本实验还需要培养数据库 SQL 应用编程的复杂工程问题解决能力。

六、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。

七、实验器材（设备、元器件）：

“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：Windows7 / Windows XP

管理工具：pgAdmin4

DBMS 系统：PostgreSQL 11

八、实验步骤：

- 1、根据实验要求与内容进行数据需求分析，并根据分析结果进行系统概念数据模型设计、逻辑数据模型设计、物理数据模型设计。
- 2、创建图书销售管理系统数据库 BookSale。
- 3、在 BookSale 数据库中创建 Book（图书表）、Type（图书类别表）、Author（作者表）、Author_Rank（编著排名表）、Zone（地区表）、Publisher（出版社表）、Inventory（库存表）、Sale（销售表）、Store（商店表）、Discount（折扣表），以及为各表创建索引。
- 4、为创建的各表插入样本数据，并在各表上进行数据的增、删、查、改。
- 5、从 Book、Author、Publisher 表创建视图 Detail_View，并通过视图进行数据查询。
- 6、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。编写过程语句块，调用存储过程函数 Pro_CurrentSale，输出统计结果。
- 7、编写 Sale 表 Insert 触发器 Tri_InsertSale，实现图书库存数据同步修改处理，并进行功能验证。
- 8、编写存储过程函数 Pro_CountSale 实现图书销售数量和金额统计，并进行功能验证。

九、实验数据及结果分析

1、数据库设计

1.1 数据库建模设计

数据库建模设计分为概念数据模型设计、逻辑数据模型设计和物理数据模型设计。

概念数据模型是一种将现实世界数据及其关系映射到信息世界数据实体及其关系的顶层抽象，同时也是数据库设计人员与用户之间进行交流的数据模型载体。

逻辑数据模型是概念数据模型在系统设计角度的延伸，它使系统的 E-R 模型图体现数据库模型的针对性（如针对关系数据库设计），同时又不依赖于具体的 DBMS 产品。

在系统物理数据模型设计中，不再使用 E-R 图来描述数据模型结构，而需要考虑将实体如何转换为数据库表、实体联系如何转换为参照完整性约束，并进行数据库索引定义、视图定义、触发器定义、存储过程定义等设计。

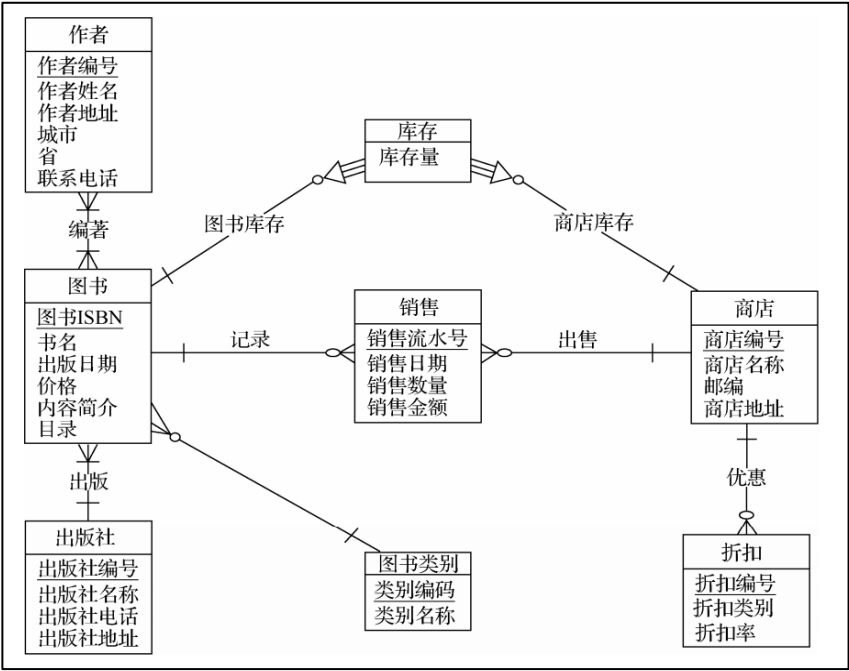


图 9.1.1 概念数据模型

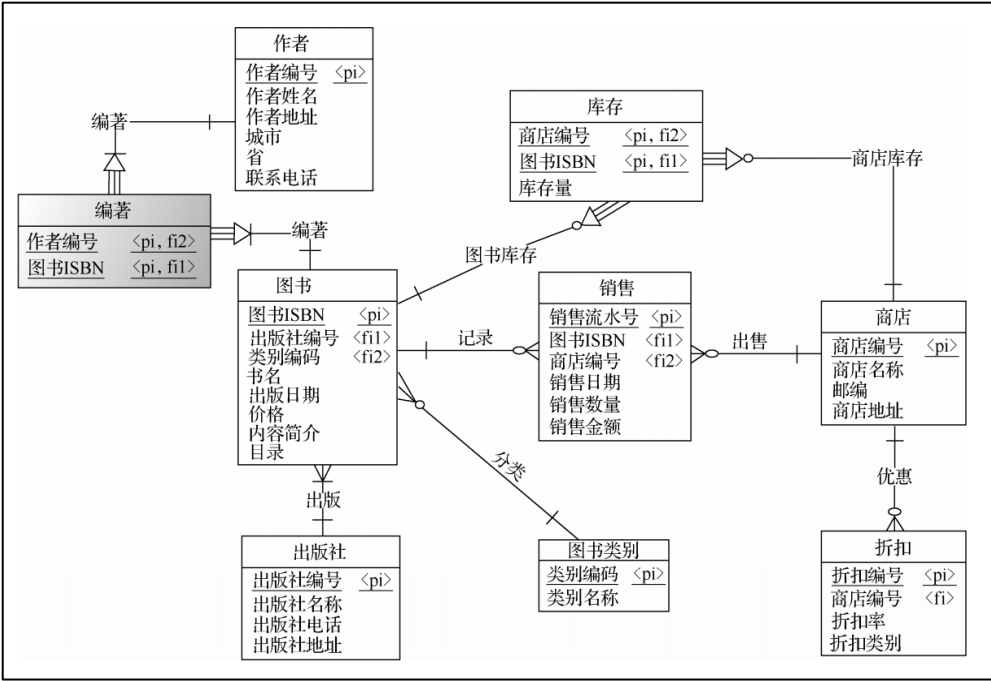


图 9.1.2 逻辑数据模型

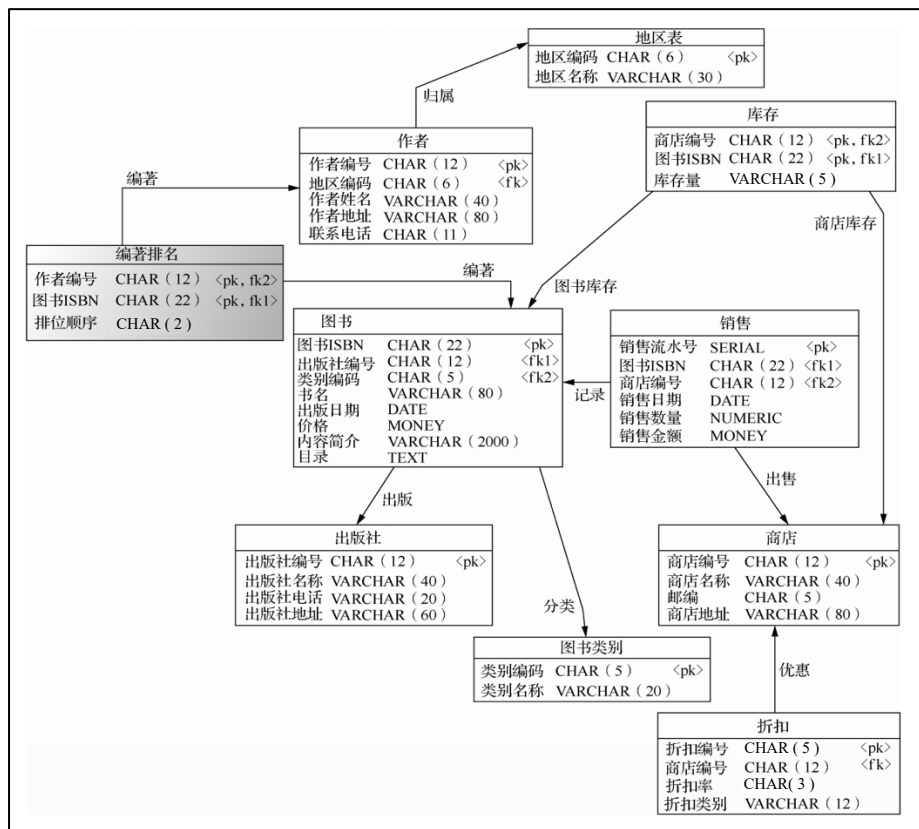


图 9.1.3 物理数据模型

1.2 数据表字段结构定义

表 9.2.1 Book (图书表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
图书 ISBN	ISBN	Char	22	是	主键
出版社编号	PublisherID	Char	12	是	外键
类别编码	TypeID	Char	5	是	外键
书名	BookName	Varchar	80	是	
出版日期	PubDate	Date	/	是	
价格	BookPrice	Money	/	是	
内容简介	BookIntro	Varchar	2000	是	
目录	BookCatalogue	Text	/	是	

表 9.2.2 Type (图书类别表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
类别编码	TypeID	Char	5	是	主键
类别名称	TypeName	Varchar	20	是	

表 9.2.3 Author (作者表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
作者编号	AuthorID	Char	12	是	主键
地区编码	ZoneID	Char	6	是	外键
作者姓名	AuthorName	Varchar	40	是	
作者地址	AuthorAddr	Varchar	80	是	
联系电话	AuthorTel	Char	11	是	

表 9.2.4 Author_Rank (编著排名表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
作者编号	AuthorID	Char	12	是	主键, 外键
图书 ISBN	ISBN	Char	22	是	主键, 外键
排位顺序	AuthorRank	Char	2	是	

表 9.2.5 Zone (地区表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
地区编码	ZoneID	Char	6	是	主键
地区名称	ZoneName	Varchar	30	是	

表 9.2.6 Publisher (出版社表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
出版社编号	PublisherID	Char	12	是	主键
出版社名称	PublisherName	Varchar	40	是	
出版社电话	PublisherTel	Varchar	20	是	
出版社地址	PublisherAddr	Varchar	60	是	

表 9.2.7 Inventory (库存表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
商店编号	StoreID	Char	12	是	主键, 外键
图书 ISBN	ISBN	Char	22	是	主键, 外键
库存量	InventoryAmount	Varchar	5	是	

表 9.2.8 Sale (销售表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
销售流水号	SaleID	Serial	/	是	主键
图书 ISBN	ISBN	Char	22	是	外键
商店编号	StoreID	Char	12	是	外键
销售日期	SaleDate	Date	/	是	
销售数量	SaleAmount	Numeric	(500, 0)	是	
销售金额	SalePrice	Money	/	是	

表 9.2.9 Store (商店表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
商店编号	StoreID	Char	12	是	主键
商店名称	StoreName	Varchar	40	是	
邮编	Postcode	Char	5	是	
商店地址	StoreAddr	Varchar	80	是	

表 9.2.10 Discount (折扣表)

字段名称	字段编码	数据类型	字段大小	必填字段	备注
折扣编号	DiscountID	Char	5	是	主键
商店编号	StoreID	Char	12	是	外键
折扣率	DiscountRate	Char	3	是	
折扣类别	DiscountType	Varchar	20	是	

2、创建数据库 BookSale

2.1 数据库创建 SQL 语句

```

1. CREATE DATABASE "BookSale"
2.     WITH
3.     OWNER = postgres
4.     ENCODING = 'UTF8'
5.     LC_COLLATE = 'C'

```

```

6. LC_CTYPE = 'C'
7. TABLESPACE = pg_default
8. CONNECTION LIMIT = -1;

```

2.2 数据库创建结果

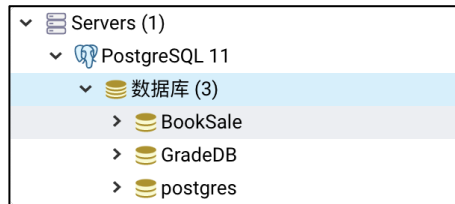


图 9.2.1 数据库创建结果

结果说明：成功创建 BookSale 数据库。

3、创建数据库表及索引

3.1 数据库表创建 SQL 语句

```

1. CREATE TABLE public."Book"
2. (
3.     "ISBN" character(22) NOT NULL,
4.     "PublisherID" character(12) NOT NULL,
5.     "TypeID" character(5) NOT NULL,
6.     "BookName" character varying(80) NOT NULL,
7.     "PubDate" date NOT NULL,
8.     "BookPrice" money NOT NULL,
9.     "BookIntro" character varying(2000) NOT NULL,
10.    "BookCatalogue" text NOT NULL,
11.    PRIMARY KEY ("ISBN"),
12.    CONSTRAINT "PublisherID_fk" FOREIGN KEY ("PublisherID")
13.        REFERENCES public."Publisher" ("PublisherID") MATCH SIMPLE
14.        ON UPDATE NO ACTION
15.        ON DELETE NO ACTION,
16.    CONSTRAINT "TypeID_fk" FOREIGN KEY ("TypeID")
17.        REFERENCES public."Type" ("TypeID") MATCH SIMPLE
18.        ON UPDATE NO ACTION
19.        ON DELETE NO ACTION
20. )
21.
22.
23.
24. CREATE TABLE public."Type"
25. (
26.     "TypeID" character(5) NOT NULL,
27.     "TypeName" character varying(20) NOT NULL,
28.     PRIMARY KEY ("TypeID")
29. )
30.
31.
32.
33. CREATE TABLE public."Author"
34. (
35.     "AuthorID" character(12) NOT NULL,
36.     "ZoneID" character(6) NOT NULL,
37.     "AuthorName" character varying(40) NOT NULL,
38.     "AuthorAddr" character varying(80) NOT NULL,

```

```

39.     "AuthorTel" character(11) NOT NULL,
40.     PRIMARY KEY ("AuthorID"),
41.     CONSTRAINT "ZoneID_fk" FOREIGN KEY ("ZoneID")
42.         REFERENCES public."Zone" ("ZoneID") MATCH SIMPLE
43.         ON UPDATE NO ACTION
44.         ON DELETE NO ACTION
45. )
46.
47.
48.
49. CREATE TABLE public."Author_Rank"
50. (
51.     "AuthorID" character(12) NOT NULL,
52.     "ISBN" character(22) NOT NULL,
53.     "AuthorRank" character(2) NOT NULL,
54.     PRIMARY KEY ("AuthorID", "ISBN"),
55.     CONSTRAINT "AuthorID_fk" FOREIGN KEY ("AuthorID")
56.         REFERENCES public."Author" ("AuthorID") MATCH SIMPLE
57.         ON UPDATE NO ACTION
58.         ON DELETE NO ACTION,
59.     CONSTRAINT "ISBN_fk" FOREIGN KEY ("ISBN")
60.         REFERENCES public."Book" ("ISBN") MATCH SIMPLE
61.         ON UPDATE NO ACTION
62.         ON DELETE NO ACTION
63. )
64.
65.
66.
67. CREATE TABLE public."Zone"
68. (
69.     "ZoneID" character(6) NOT NULL,
70.     "ZoneName" character varying(30) NOT NULL,
71.     PRIMARY KEY ("ZoneID")
72. )
73.
74.
75.
76. CREATE TABLE public."Publisher"
77. (
78.     "PublisherID" character(12) NOT NULL,
79.     "PublisherName" character varying(40) NOT NULL,
80.     "PublisherTel" character varying(20) NOT NULL,
81.     "PublisherAddr" character varying(60) NOT NULL,
82.     PRIMARY KEY ("PublisherID")
83. )
84.
85.
86.
87. CREATE TABLE public."Inventory"
88. (
89.     "StoreID" character(12) NOT NULL,
90.     "ISBN" character(22) NOT NULL,
91.     "InventoryAmount" character varying(5) NOT NULL,
92.     PRIMARY KEY ("StoreID", "ISBN"),
93.     CONSTRAINT "StoreID_fk" FOREIGN KEY ("StoreID")
94.         REFERENCES public."Store" ("StoreID") MATCH SIMPLE
95.         ON UPDATE NO ACTION
96.         ON DELETE NO ACTION,
97.     CONSTRAINT "ISBN_fk2" FOREIGN KEY ("ISBN")

```

```

98. REFERENCES public."Book" ("ISBN") MATCH SIMPLE
99. ON UPDATE NO ACTION
100. ON DELETE NO ACTION
101. )
102.
103.
104.
105. CREATE TABLE public."Sale"
106. (
107.     "SaleID" serial NOT NULL,
108.     "ISBN" character(22) NOT NULL,
109.     "StoreID" character(12) NOT NULL,
110.     "SaleDate" date NOT NULL,
111.     "SaleAmount" numeric(500, 0) NOT NULL,
112.     "SaleMoney" money NOT NULL,
113.     PRIMARY KEY ("SaleID"),
114.     CONSTRAINT "ISBN_fk2" FOREIGN KEY ("ISBN")
115.         REFERENCES public."Book" ("ISBN") MATCH SIMPLE
116.         ON UPDATE NO ACTION
117.         ON DELETE NO ACTION,
118.     CONSTRAINT "StoreID_fk" FOREIGN KEY ("StoreID")
119.         REFERENCES public."Store" ("StoreID") MATCH SIMPLE
120.         ON UPDATE NO ACTION
121.         ON DELETE NO ACTION
122. )
123.
124.
125.
126. CREATE TABLE public."Store"
127. (
128.     "StoreID" character(12) NOT NULL,
129.     "StoreName" character varying(40) NOT NULL,
130.     "Postcode" character(5) NOT NULL,
131.     "StoreAddr" character varying(80) NOT NULL,
132.     PRIMARY KEY ("StoreID")
133. )
134.
135.
136.
137. CREATE TABLE public."Discount"
138. (
139.     "DiscountID" character(5) NOT NULL,
140.     "StoreID" character(12) NOT NULL,
141.     "DiscountRate" character(3) NOT NULL,
142.     "DiscountType" character varying(20) NOT NULL,
143.     PRIMARY KEY ("DiscountID"),
144.     CONSTRAINT "StoreID_fk2" FOREIGN KEY ("StoreID")
145.         REFERENCES public."Store" ("StoreID") MATCH SIMPLE
146.         ON UPDATE NO ACTION
147.         ON DELETE NO ACTION
148. )

```

3.2 数据库表创建结果

结果说明：成功创建 Book（图书表）、Type（图书类别表）、Author（作者表）、Author_Rank（编著排名表）、Zone（地区表）、Publisher（出版社表）、Inventory（库存表）、Sale（销售表）、Store（商店表）、Discount（折扣表）。

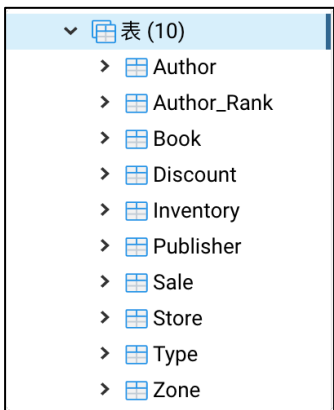


图 9.3.1 数据库表创建结果

3.3 索引创建 SQL 语句

```
1. CREATE INDEX BookName_Idx ON Public."Book"("BookName");
2. CREATE INDEX TypeName_Idx ON Public."Type"("TypeName");
3. CREATE INDEX AuthorName_Idx ON Public."Author"("AuthorName");
4. CREATE INDEX AuthorRank_Idx ON Public."Author_Rank"("AuthorRank");
5. CREATE INDEX ZoneName_Idx ON Public."Zone"("ZoneName");
6. CREATE INDEX PublisherName_Idx ON Public."Publisher"("PublisherName");
7. CREATE INDEX InventoryAmount_Idx ON Public."Inventory"("InventoryAmount");
8. CREATE INDEX SaleDate_Idx ON Public."Sale"("SaleDate");
9. CREATE INDEX StoreName_Idx ON Public."Store"("StoreName");
10. CREATE INDEX DiscountType_Idx ON Public."Discount"("DiscountType");
```

3.4 索引创建结果

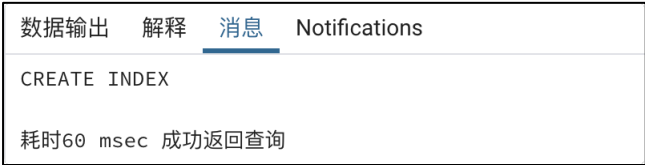


图 9.3.2 索引创建 SQL 语句执行结果

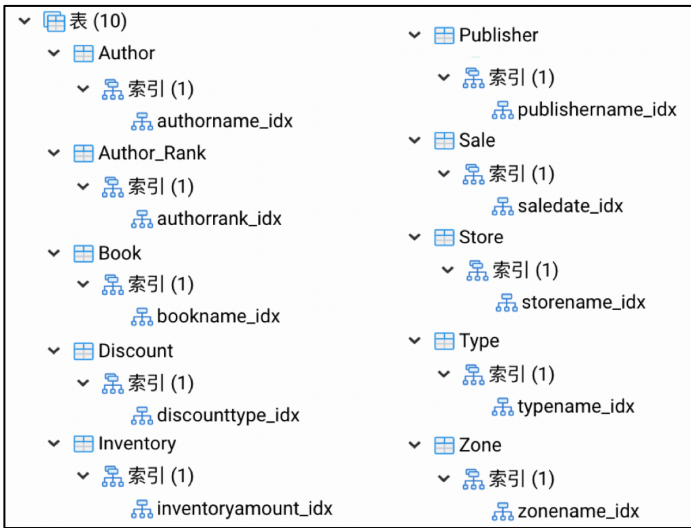


图 9.3.3 索引创建结果

结果说明：成功执行索引创建 SQL 语句，并成功创建 AuthorName_Idx、

AuthorRank_Idx、BookName_Idx、DiscountType_Idx、InventoryAmount_Idx、PublisherName_Idx、SaleDate_Idx、StoreName_Idx、TypeName_Idx、ZoneName_Idx 索引。

4、数据库表数据的插入与操作

4.1 向 Type（图书类别表）中插入数据

1.	INSERT INTO Type VALUES('G80','体育理论');
2.	INSERT INTO Type VALUES('TP311','自动化技术、计算机技术');
3.	INSERT INTO Type VALUES('H313','常用外国语');
4.	INSERT INTO Type VALUES('I253','文学、艺术');
5.	INSERT INTO Type VALUES('G633','外语');

4.2 查询 Type（图书类别表）中的所有数据

数据输出	解释	消息	Notifications
	TypeID [PK] character (5)	TypeName character varying (20)	
1	G80	体育理论	
2	H313	常用外国语	
3	I253	文学、艺术	
4	G633	外语	
5	TP311	自动化技术、计算机技术	

图 9.4.1 Type（图书类别表）数据查询结果

结果说明：成功向 Type（图书类别表）中插入数据。

4.3 向 Zone（地区表）中插入数据

1.	INSERT INTO Zone VALUES('510108','四川省成都市锦江区');
2.	INSERT INTO Zone VALUES('110108','北京市海淀区');
3.	INSERT INTO Zone VALUES('310106','上海市静安区');
4.	INSERT INTO Zone VALUES('440305','深圳市南山区');
5.	INSERT INTO Zone VALUES('710001','台湾省台北市');

4.4 查询 Zone（地区表）中的所有数据

数据输出	解释	消息	Notifications
	ZoneID [PK] character (6)	ZoneName character varying (30)	
1	110108	北京市海淀区	
2	310106	上海市静安区	
3	440305	深圳市南山区	
4	510108	四川省成都市成华区	
5	710001	台湾省台北市信义区	

图 9.4.2 Zone（地区表）数据查询结果

结果说明：成功向 Zone（地区表）中插入数据。

4.5 向 Store（商店表）中插入数据

1. **INSERT INTO** Store **VALUES**('st5289825801','教辅书店','62088','江苏省南京市江宁区迎宾大道 2 号');
2. **INSERT INTO** Store **VALUES**('st5289825802','文学书店','61066','上海市长宁区长宁路 1057 号');
3. **INSERT INTO** Store **VALUES**('st5289825803','新华文轩','16877','四川省成都市武侯区武侯祠大街 266 号附 2-3 号');
4. **INSERT INTO** Store **VALUES**('st5289825804','布克书店','91255','四川省成都市金牛区蜀汉路 355-10 号');
5. **INSERT INTO** Store **VALUES**('st5289825805','知博书店','61173','四川省成都市高新西区西源大道 2006 号');

4.6 查询 Store（商店表）中的所有数据

数据输出		解释	消息	Notifications
	StoreID [PK] character (12)	StoreName character varying (40)	Postcode character (5)	StoreAddr character varying (80)
1	st5289825801	教辅书店	62088	江苏省南京市江宁区迎宾...
2	st5289825802	文学书店	61066	上海市长宁区长宁路1057号
3	st5289825803	新华文轩	16877	四川省成都市武侯区武侯...
4	st5289825804	布克书店	91255	四川省成都市金牛区蜀汉...
5	st5289825805	知博书店	61173	四川省成都市高新西区西...

图 9.4.3 Store（商店表）数据查询结果

结果说明：成功向 Store（商店表）中插入数据。

4.7 向 Publisher（出版社表）中插入数据

1. **INSERT INTO** Publisher **VALUES**('pub862782601','人民邮电出版社','010-81055256','北京市丰台区成寿寺路 11 号');
2. **INSERT INTO** Publisher **VALUES**('pub862782602','电子工业出版社','010-88254888','北京市万寿路南口金家村 288 号华信大厦');
3. **INSERT INTO** Publisher **VALUES**('pub862782603','电子科技大学出版社','028-83202439','成都市一环路东一段 159 号电子信息产业大厦 9 层 901');
4. **INSERT INTO** Publisher **VALUES**('pub862782604','北京联合出版公司','010-82065168','北京市西城区德外大街 83 号楼 901 室(德胜园区)');
5. **INSERT INTO** Publisher **VALUES**('pub862782605','机械工业出版社','010-68326294','北京市西城区百万庄大街 22 号');

4.8 查询 Publisher（出版社表）中的所有数据

数据输出

解释

消息

Notifications

	<div>PublisherID</div> <div>[PK] character (12)</div>	<div>PublisherName</div> <div>character varying (40)</div>	<div>PublisherTel</div> <div>character varying (20)</div>	<div>PublisherAddr</div> <div>character varying (60)</div>
1	pub862782601	人民邮电出版社	010-81055256	北京市丰台区成寿寺路11号
2	pub862782602	电子工业出版社	010-88254888	北京市万寿路南口金家村2...
3	pub862782603	电子科技大学出版社	028-83202439	成都市一环路东一段159号...
4	pub862782604	北京联合出版公司	010-82065168	北京市西城区德外大街83...
5	pub862782605	机械工业出版社	010-68326294	北京市西城区百万庄大街2...

图 9.4.4 Publisher（出版社表）数据查询结果

结果说明：成功向 Publisher（出版社表）中插入数据。

4.9 向 Book（图书表）中插入数据

1.	<code>INSERT INTO Book VALUES('9787115502742','pub862782601','TP31','数据库系统—原理、设计与编程','2019-03-04','\$59.80','介绍关系数据库 PostgreSQL','数据库目录');</code>
2.	<code>INSERT INTO Book VALUES('9787111599715','pub862782605','TP31','计算机网络自顶向方法','2019-09-01','\$90.00','介绍计算机网络系统','计算机网络系统目录');</code>
3.	<code>INSERT INTO Book VALUES('9787559626813','pub862782604','G633','雅思词汇','2018-11-11','\$68.00','介绍雅思词汇','雅思词汇目录');</code>
4.	<code>INSERT INTO Book VALUES('9787560633503','pub862782603','TP31','计算机操作系统','2016-06-08','\$53','介绍计算机操作系统','计算机操作系统目录');</code>
5.	<code>INSERT INTO Book VALUES('9787121365010','pub862782602','TP31','密码学','2020-02-18','\$48','介绍密码学基础','密码学目录');</code>

4.10 查询 Book（图书表）中的所有数据

数据输出		解释	消息	Notifications				
	ISBN [PK] character	PublisherID character (12)	TypeID character	BookName character varying (128)	PubDate date	BookPri money	BookIntro character varying (255)	BookCatalogue text
1	97871155...	pub862782...	TP311	数据库系统——...	2019-03...	\$59.80	介绍关系数据库P...	数据库目录
2	97871115...	pub862782...	TP311	计算机网络自顶...	2019-09...	\$90.00	介绍计算机网络...	计算机网络系...
3	97875596...	pub862782...	G633	雅思词汇	2018-11...	\$68.00	介绍雅思词汇	雅思词汇目录
4	97875606...	pub862782...	TP311	计算机操作系统	2016-06...	\$53.00	介绍计算机操作...	计算机操作系...
5	97871213...	pub862782...	TP311	密码学	2020-02...	\$48.00	介绍密码学基础	密码学目录

图 9.4.5 Book（图书表）数据查询结果

结果说明：成功向 Book（图书表）中插入数据。

4.11 向 Author（作者表）中插入数据

1.	<code>INSERT INTO Author VALUES('aut263618601','510108','刘一','四川省成都市锦江区','18235472947');</code>
2.	<code>INSERT INTO Author VALUES('aut263618602','110108','吴二','北京市海淀区','18925480284');</code>
3.	<code>INSERT INTO Author VALUES('aut263618603','310106','张三','上海市静安区','13320980170');</code>
4.	<code>INSERT INTO Author VALUES('aut263618604','440305','李四','深圳市南山区','18990350627');</code>
5.	<code>INSERT INTO Author VALUES('aut263618605','710001','王二麻子','台湾省台北市','16238593746');</code>

4.12 查询 Author（作者表）中的所有数据

数据输出						解释	消息	Notifications
	AuthorID [PK] character (12)	ZoneID character (6)	AuthorName character varying (40)	AuthorAddr character varying (80)	AuthorTel character (11)			
1	aut263618601	510108	刘一	四川省成都市锦江区	18235472947			
2	aut263618602	110108	吴二	北京市海淀区	18925480284			
3	aut263618603	310106	张三	上海市静安区	13320980170			
4	aut263618604	440305	李四	深圳市南山区	18990350627			
5	aut263618605	710001	王二麻子	台湾省台北市	16238593746			

图 9.4.6 Author（作者表）数据查询结果

结果说明：成功向 Author（作者表）中插入数据。

4.13 向 Author_Rank（编者排名表）中插入数据

```
1. INSERT INTO Author_Rank VALUES('aut263618601','9787115502742','1');
2. INSERT INTO Author_Rank VALUES('aut263618602','9787115502742','2');
3. INSERT INTO Author_Rank VALUES('aut263618603','9787115502742','3');
4. INSERT INTO Author_Rank VALUES('aut263618604','9787115502742','4');
5. INSERT INTO Author_Rank VALUES('aut263618605','9787115502742','5');
```

4.14 查询 Author_Rank（编者排名表）中的所有数据

数据输出	解释	消息	Notifications
	AuthorID [PK] character (12)	ISBN [PK] character (22)	AuthorRank character (2)
1	aut263618601	9787115502742	1
2	aut263618602	9787115502742	2
3	aut263618603	9787115502742	3
4	aut263618604	9787115502742	4
5	aut263618605	9787115502742	5

图 9.4.7 Author_Rank（编者排名表）数据查询结果

结果说明：成功向 Author_Rank（编者排名表）中插入数据。

4.15 向 Inventory（库存表）中插入数据

```
1. INSERT INTO Inventory VALUES('st5289825801','9787115502742','10000');
2. INSERT INTO Inventory VALUES('st5289825802','9787115502742','0');
3. INSERT INTO Inventory VALUES('st5289825803','9787115502742','8000');
4. INSERT INTO Inventory VALUES('st5289825804','9787115502742','6000');
5. INSERT INTO Inventory VALUES('st5289825805','9787115502742','627');
```

4.16 查询 Inventory（库存表）中的所有数据

数据输出	解释	消息	Notifications
	StoreID [PK] character (12)	ISBN [PK] character (22)	InventoryAmount character varying (5)
1	st5289825801	9787115502742	10000
2	st5289825802	9787115502742	0
3	st5289825803	9787115502742	8000
4	st5289825804	9787115502742	6000
5	st5289825805	9787115502742	627

图 9.4.8 Inventory（库存表）数据查询结果

结果说明：成功向 Inventory（库存表）中插入数据。

4.17 向 Sale（销售表）中插入数据

```
1. INSERT INTO Sale VALUES('','9787115502742','st5289825801','2020-02-06','1','$28.00');
2. INSERT INTO Sale VALUES('','9787111599715','st5289825802','2020-03-04','2','$34.00');
```

3. **INSERT INTO** Sale **VALUES**(' ', '9787559626813', 'st5289825803', '2020-01-02', '3', '\$76.45');
4. **INSERT INTO** Sale **VALUES**(' ', '9787560633503', 'st5289825804', '2020-05-06', '4', '\$23.87');
5. **INSERT INTO** Sale **VALUES**(' ', '9787121365010', 'st5289825805', '2020-06-27', '8', '\$66.66');

4.18 查询 Sale（销售表）中的所有数据

数据输出

解释

消息

Notifications

	SaleID [PK] integer	ISBN character (22)	StoreID character (12)	SaleDate date	SaleAmount numeric	SaleMoney money
1	1	9787115502742 ...	st5289825801	2020-02-06	1	\$28.00
2	2	9787111599715 ...	st5289825802	2020-03-04	2	\$34.00
3	3	9787559626813 ...	st5289825803	2020-01-02	3	\$76.45
4	4	9787560633503 ...	st5289825804	2020-05-06	4	\$23.87
5	5	9787121365010 ...	st5289825805	2020-06-27	8	\$66.66

图 9.4.9 Sale（销售表）数据查询结果

结果说明：成功向 Sale（销售表）中插入数据。

4.19 向 Discount（折扣表）中插入数据

1. **INSERT INTO** Discount **VALUES**('dct01', 'st5289825805', '0.6', '折扣类型 1');
2. **INSERT INTO** Discount **VALUES**('dct02', 'st5289825805', '0.9', '折扣类型 2');
3. **INSERT INTO** Discount **VALUES**('dct03', 'st5289825805', '0.7', '折扣类型 3');
4. **INSERT INTO** Discount **VALUES**('dct04', 'st5289825805', '0.2', '折扣类型 4');
5. **INSERT INTO** Discount **VALUES**('dct05', 'st5289825805', '0.1', '折扣类型 5');

4.20 查询 Discount（折扣表）中的所有数据


数据输出					解释	消息	Notifications
		DiscountID [PK] character (5)	StoreID character (12)	DiscountRate character (3)	DiscountType character varying (20)		
1		dct01	st5289825805	0.6	折扣类型1		
2		dct02	st5289825805	0.9	折扣类型2		
3		dct03	st5289825805	0.7	折扣类型3		
4		dct04	st5289825805	0.2	折扣类型4		
5		dct05	st5289825805	0.1	折扣类型5		

图 9.4.10 Discount（折扣表）数据查询结果

结果说明：成功向 Discount（折扣表）中插入数据。

4.21 从 Book（图书表）中删除数据

1. **DELETE FROM** Book **WHERE** ISBN='9787121365010';

4.22 查询 Book（图书表）执行删除操作后的所有数据

	数据输出	解释	消息	Notifications				
	ISBN [PK] character	PublisherID character (12)	TypeID charact	BookName character varying (PubDate date	BookPri money	BookIntro character varying	BookCatalogue text
1	97871155...	pub862782...	TP311	数据库系统——...	2019-0...	\$59.80	介绍关系数据...	数据库目录
2	97871115...	pub862782...	TP311	计算机网络自顶...	2019-0...	\$90.00	介绍计算机网...	计算机网络...
3	97875596...	pub862782...	G633	雅思词汇	2018-1...	\$68.00	介绍雅思词汇	雅思词汇目录
4	97875606...	pub862782...	TP311	计算机操作系统	2016-0...	\$53.00	介绍计算机操...	计算机操作...

图 9.4.11 Book（图书表）执行删除操作后查询结果

结果说明：成功从 Book（图书表）中删除数据。

4.23 从 Author（作者表）中修改数据

1. **UPDATE** Author
2. **SET** AuthorName='刘更改' **WHERE** AuthorID='aut263618601';

4.24 查询 Author（作者表）执行修改操作后的所有数据

	数据输出	解释	消息	Notifications				
	AuthorID [PK] character (12)	ZoneID character (6)	AuthorName character varying (40)	AuthorAddr character varying (80)	AuthorTel character (11)			
1	aut263618601	510108	刘更改	四川省成都市锦江区	18235472947			
2	aut263618602	110108	吴二	北京市海淀区	18925480284			
3	aut263618603	310106	张三	上海市静安区	13320980170			
4	aut263618604	440305	李四	深圳市南山区	18990350627			
5	aut263618605	710001	王二麻子	台湾省台北市	16238593746			

图 9.4.12 Author（作者表）执行修改操作后查询结果

结果说明：成功从 Author（作者表）中修改数据。

5、创建视图及从视图查询

5.1 视图创建与查询 SQL 语句

1. **CREATE VIEW** Detail_View **AS**
2. **SELECT** B.ISBN **AS** 图书 ISBN, B.BookName **AS** 书名, Aut.AuthorName **AS** 作者,
3. Pub.PublisherName **AS** 出版社, B.BookPrice **AS** 价格
4. **FROM** Book **AS** B **JOIN** Publisher **AS** Pub **ON** B.PublisherID = Pub.PublisherID
5. **JOIN** Author_Rank **AS** Aut_R **ON** B.ISBN = Aut_R.ISBN
6. **JOIN** Author **AS** Aut **ON** Aut_R.AuthorID = Aut.AuthorID;
- 7.
- 8.
9. **SELECT** * **FROM** public.detail_view;

5.2 视图查询结果

数据输出						解释	消息	Notifications
	图书ISBN character (22)	书名 character varying (80)	作者 character varying (40)	出版社 character varying (40)	价格 money			
1	9787115502742 ...	数据库系统——原理、设计与编程	刘更改	人民邮电出版社	\$59.80			
2	9787111599715 ...	计算机网络自顶向下方法	吴二	机械工业出版社	\$90.00			
3	9787559626813 ...	雅思词汇	张三	北京联合出版公司	\$68.00			
4	9787560633503 ...	计算机操作系统	李四	电子科技大学出版社	\$53.00			
5	9787121365010 ...	密码学	王二麻子	电子工业出版社	\$48.00			

图 9.5.1 Detai_View 视图查询结果

结果说明：成功创建 Detail_View 视图并正确查询到数据。

6、创建与调用存储过程实现统计当日销售数据

6.1 创建存储过程 Pro_CurrentSale() SQL 语句

```

1. CREATE OR REPLACE FUNCTION Pro_CurrentSale(IN Count_Date date, OUT Amount in
t, OUT Total_Money money) as $count$
2. BEGIN
3. SELECT SUM(SaleAmount) into Amount FROM Sale WHERE SaleDate=Count_Date;
4. SELECT SUM(SaleMoney) into Total_Money FROM Sale WHERE SaleDate=Count_Date;
5. END;
6. $count$ LANGUAGE plpgsql;

```

6.2 调用存储过程 Pro_CurrentSale()实现当日销售数量与金额统计

```

1. SELECT * FROM Pro_CurrentSale(cast ('2020-06-27' as date));

```

6.3 调用存储过程 Pro_CurrentSale()统计结果验证

	数据输出	解释	消息	Notifications
	amount integer	total_money money		
1	13	\$177.11		

图 9.6.1 调用存储过程 Pro_CurrentSale 结果


数据输出		解释	消息	Notifications		
	SaleID [PK] integer	ISBN character (22)	StoreID character (12)	SaleDate date	SaleAmount numeric (500)	SaleMoney money
1	1	9787115502742 ...	st5289825801	2020-02-06	1	\$28.00
2	2	9787111599715 ...	st5289825802	2020-06-27	2	\$34.00
3	3	9787559626813 ...	st5289825803	2020-06-27	3	\$76.45
4	4	9787560633503 ...	st5289825804	2020-05-06	4	\$23.87
5	5	9787121365010 ...	st5289825805	2020-06-27	8	\$66.66

图 9.6.2 查询 Sale（销售表）进行结果验证

结果说明：成功调用存储过程 Pro_CurrentSale()。当日（2020-06-27）共销售 13 本书，销售总额为\$177.11。

7、创建与使用触发器

7.1 创建触发器函数 Fun_InsertSale() SQL 语句

```
1. CREATE OR REPLACE FUNCTION Fun_InsertSale()
2. RETURNS TRIGGER AS $insert$
3. BEGIN
4.     IF(TG_OP = 'INSERT') THEN
5.         UPDATE Inventory SET InventoryAmount = cast(InventoryAmount as int) - c
6.         ast(NEW.SaleAmount as int)
7.         WHERE Inventory.ISBN = NEW.ISBN AND Inventory.StoreID = NEW.StoreID;
8.         RETURN NEW;
9.     END IF;
10.    RETURN NULL;
11. $insert$ LANGUAGE plpgsql;
```

7.2 创建触发器 Tri_InsertSale SQL 语句

```
1. CREATE TRIGGER Tri_InsertSale
2. AFTER INSERT ON Sale
3. FOR EACH ROW EXECUTE PROCEDURE Fun_InsertSale();
```

7.3 在 Sale（销售表）上插入销售数据 SQL 语句

```
1. INSERT INTO Sale VALUES(' ', '9787115502742', 'st5289825804', '2020-06-
27', '1', '66');
```

7.4 Sale（销售表）数据插入结果

数据输出		解释	消息	Notifications		
<div><div></div></div>	SaleID	ISBN	StoreID	SaleDate	SaleAmount	SaleMoney
	[PK] integer	character (22)	character (12)	date	numeric (500)	money
1	1	9787115502742 ...	st5289825801	2020-02-06	1	\$28.00
2	2	9787111599715 ...	st5289825802	2020-06-27	2	\$34.00
3	3	9787559626813 ...	st5289825803	2020-06-27	3	\$76.45
4	4	9787560633503 ...	st5289825804	2020-05-06	4	\$23.87
5	5	9787121365010 ...	st5289825805	2020-06-27	8	\$66.66
6	6	9787115502742 ...	st5289825804	2020-06-27	1	\$66.00

图 9.7.1 Sale（销售表）数据插入结果

7.5 验证触发器是否在 Inventory（库存表）上更新数据

数据输出	解释	消息	Notifications
	StoreID [PK] character (12)	ISBN [PK] character (22)	InventoryAmount character varying (5)
1	st5289825801	9787115502742	10000
2	st5289825802	9787115502742	0
3	st5289825803	9787115502742	8000
4	st5289825804	9787115502742	6000
5	st5289825805	9787115502742	627

图 9.7.2 触发器执行前 Inventory（库存表）数据

数据输出	解释	消息	Notifications
	StoreID [PK] character (12)	ISBN [PK] character (22)	InventoryAmount character varying (5)
1	st5289825801	9787115502742	10000
2	st5289825802	9787115502742	0
3	st5289825803	9787115502742	8000
4	st5289825805	9787115502742	627
5	st5289825804	9787115502742	5999

图 9.7.3 触发器执行后 Inventory（库存表）数据

结果说明：成功执行触发器 Tri_InsertSale。在 Sale（销售表）中插入销售数据后，Inventory（库存表）能实现同步更新库存数据。

8、创建与调用存储过程实现统计图书销售总数据

8.1 创建存储过程 Pro_CountSale() SQL 语句

```

1. CREATE OR REPLACE FUNCTION Pro_CountSale(IN Count_ISBN char, OUT Amount int,
   OUT Total_Money money) as $count$
2. BEGIN
3. SELECT SUM(SaleAmount) into Amount FROM public."Sale" WHERE ISBN=Count_ISBN;
4. SELECT SUM(SaleMoney) into Total_Money FROM public."Sale" WHERE ISBN=Count_I
   SBN;
5. END;
6. $count$ LANGUAGE plpgsql;

```

8.2 调用存储过程 Pro_CountSale()实现图书销售数量与金额统计

```

1. SELECT * FROM Pro_CountSale('9787115502742');
2. $count$ LANGUAGE plpgsql;

```

8.3 调用存储过程 Pro_CountSale()统计结果验证

数据输出	解释	消息	Notifications
	amount integer	total_money money	
1	3	\$160.00	

图 9.8.1 调用存储过程 Pro_CountSale 结果

数据输出		解释	消息	Notifications		
	SaleID [PK] integer	ISBN character (22)	StoreID character (12)	SaleDate date	SaleAmount numeric (500)	SaleMoney money
1	1	9787115502742	st5289825801	2020-02-06	1	\$28.00
2	4	9787560633503	st5289825804	2020-05-06	4	\$23.87
3	5	9787121365010	st5289825805	2020-06-27	8	\$66.66
4	2	9787111599715	st5289825802	2020-06-27	2	\$34.00
5	3	9787559626813	st5289825803	2020-06-27	3	\$76.45
6	11	9787115502742	st5289825804	2020-06-27	1	\$66.00
7	6	9787115502742	st5289825804	2020-06-27	1	\$66.00

图 9.8.2 查询 Sale（销售表）进行结果验证

结果说明: 成功调用存储过程 Pro_CountSale()。ISBN 号为 9787115502742 的图书共销售 3 本, 销售总额为\$160。

十、实验结论

- 1、根据实验要求与内容完成了系统概念数据模型设计、逻辑数据模型设计、物理数据模型设计。
- 2、成功创建图书销售管理系统数据库 BookSale。
- 3、在 BookSale 数据库中成功创建 Book(图书表)、Type(图书类别表)、Author (作者表)、Author_Rank (编著排名表)、Zone (地区表)、Publisher (出版社表)、Inventory (库存表)、Sale (销售表)、Store (商店表)、Discount (折扣表), 以及为各表正确创建索引, 与预期结果一致。
- 4、成功为创建的各表插入样本数据, 并在各表上正确进行数据的增、删、查、改, 与预期结果一致。
- 5、正确从 Book、Author、Publisher 表创建视图 Detail_View, 并通过视图进行数据查询。数据查询结果正确。
- 6、实现了编写存储过程函数 Pro_CurrentSale, 统计当日图书销售量及销售金额汇总。通过编写过程语句块、调用存储过程函数 Pro_CurrentSale, 正确输出统计结果。
- 7、实现了编写 Sale 表 Insert 触发器 Tri_InsertSale, 同步修改图书库存数据, 并进行功能验证, 验证结果一致。
- 8、实现了编写存储过程函数 Pro_CountSale, 统计某图书销售数量和金额, 并进行功能验证, 验证结果正确。

十一、总结及心得体会

本次实验使用 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。实验内容包括数据库、数据库表、索引、视图等数据库对象的创建操作; 定义数据库表的主、外键约束; 对数据进行增、删、查、改操作; 存储过程定义与调用以及触发器编程。

其中我对数据库对象的基本操作掌握较好, 对存储过程与触发器编程较为陌生, 还需要继续加强对数据库后端编程内容的理解与掌握。其重难点在于对数据库后端编程语句格式、框架、基本结构的掌握与灵活运用, 在编写相关语句时注意各数据库表中不同属性的相互关系。数据库编程能力的培养需要将实践与理论相结合, 总结典型问题与解决方案; 同时需要加强基础编程能力, 才能融会贯通, 达到较高的数据库编程水平。

十二、对本实验过程及方法、手段的改进建议

本实验既包含较为简单的数据库对象的基本操作，也有难度较大的存储过程与触发器编程，综合难度适中，实验内容丰富且有较强的现实意义，很好的考察了学生对数据库相关内容的掌握程度与上机实践能力。其中数据库表对象数量稍多，对数据的增、删、查、改重复操作部分较多。可以在实验内容中添加图书销售管理系统的概念模型、逻辑模型及物理模型的设计，是本实验更具综合性。

报告评分：

指导教师签字：