第二次 "数据库原理及应用"课程作业

在一个汽车租赁管理系统中,假定其数据库 CarRentDB 包括客户表(CLIENT)、汽车信息表(CAR)、租赁价目表(RENT_PRICE)、租赁登记表(RENT_REG)、租赁费用表(RENT_FEE)。系统用户角色有客户、业务员、经理、系统管理员。

在 PostgreSQL 数据库中,完成角色管理、权限管理、用户管理,以及数据库备份与恢复管理等操作,具体要求如下:

- 1) 在数据库 CarRentDB 中,创建 R_Client (客户)、R_SalesMan (业务员)、R_Manager (经理)、R_Adminstrator (系统管理员) 角色。
 - 2) 在数据库 CarRentDB 中,分别定义各个角色对数据库表对象的访问权限。
- 3) 创建用户 ClientUser 为客户角色用户,用户 SalesManUser 为业务员角色用户,用户 ManagerUser 为经理角色用户,用户 AdminstratorUser 为系统管理员角色用户。
 - 4) 分别以不同用户登录访问数据库,尝试进行不同类型访问操作。
- 5)以管理员身份进行 CarRentDB 数据库备份处理。分别创建数据库备份、schema 备份、数据库表备份。
 - 6) 当破坏数据库 CarRentDB 后,使用备份文件进行数据库恢复处理。

作业要求:在 PostgreSQL 数据库中创建 CarRentDB 数据库及其数据库表,插入样本数据,然后按照以上数据库系统管理要求进行 SQL 访问编程操作。给出每个问题解决的步骤、SQL 语句操作、执行结果界面,并对结果进行说明。

作业文件格式: 作业 2_学号_姓名.doc

作业成绩评价标准:

正确完成情况	优	优	优	良	良	良	良	良	中	中
作业过程情况	优	优	良	优	良	良	良	中	中	中
文档规范性	优	良	良	优	优	良	中	中	中	差
作业评分	100-98	97-95	94-92	91-89	88-85	84-82	81-79	78-76	75-73	72-70

1) 在数据库 CarRentDB 中,创建 R_Client(客户)、R_SalesMan(业务员)、R_Manager(经理)、R Adminstrator(系统管理员)角色。

首先根据题意创建数据库 CarRentDB, 和数据库中的客户表 (CLIENT)、汽车信息表 (CAR)、租赁价目表(RENT_PRICE)、租赁登记表(RENT_REG)与租赁费用表(RENT_FEE)。 SQL 代码如 SQL 代码 1-6 所示。

接下来在数据库中创建四种不同的角色,如图1所示。

SQL 代码 1 创建数据库 CarRentDB

```
CREATE DATABASE "CarRentDB"

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC_COLLATE = 'C'

LC_CTYPE = 'C'

TABLESPACE = pg_default

CONNECTION LIMIT = -1;
```

SQL 代码 2 创建表 CAR

```
CREATE TABLE public."CAR"

(
    "carId" integer NOT NULL DEFAULT nextval("CAR_carId_seq"::regclass),
    "carName" text COLLATE pg_catalog."default" NOT NULL,
    "carInfo" text COLLATE pg_catalog."default",
    CONSTRAINT "CAR_pkey" PRIMARY KEY ("carId")

)

WITH (
    OIDS = FALSE
)

TABLESPACE pg_default;

ALTER TABLE public."CAR"

OWNER to postgres;
```

SQL 代码 3 创建表 CLIENT

```
CREATE TABLE public."CLIENT"

(

"cId" integer NOT NULL DEFAULT nextval("CLIENT_cId_seq"::regclass),

"cName" text COLLATE pg_catalog."default" NOT NULL,

"cGender" character(2) COLLATE pg_catalog."default",

CONSTRAINT "CLIENT_pkey" PRIMARY KEY ("cId")

)

WITH (

OIDS = FALSE

)

TABLESPACE pg_default;
```

```
ALTER TABLE public."CLIENT"

OWNER to postgres;
```

SQL 代码 4 创建表 RENT_FEE

```
CREATE TABLE public."RENT_FEE"

(
    "cld" integer NOT NULL,
    "rentFee" money NOT NULL,
    "rentFeeId" integer NOT NULL DEFAULT nextval("RENT_FEE_rentFeeId_seq"::regclass),
    CONSTRAINT "RENT_FEE_pkey" PRIMARY KEY ("rentFeeId", "cld"),
    CONSTRAINT "RENT_FEE_fkey" FOREIGN KEY ("cld")
    REFERENCES public."CLIENT" ("cld") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
)

WITH (
OIDS = FALSE
)

TABLESPACE pg_default;

ALTER TABLE public."RENT_FEE"
OWNER to postgres;
```

SQL 代码 5 创建表 RENT_PRICE

```
CREATE TABLE public."RENT_PRICE"

(
    "carId" integer NOT NULL,
    "rentPrice" money NOT NULL,
    "rentPriceId" integer NOT NULL DEFAULT nextval(""RENT_PRICE_rentPriceId_seq"::regclass),
    CONSTRAINT "RENT_PRICE_pkey" PRIMARY KEY ("carId", "rentPriceId"),
    CONSTRAINT "RENT_PRICE_fkey" FOREIGN KEY ("carId")
    REFERENCES public."CAR" ("carId") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
)

WITH (
OIDS = FALSE
)

TABLESPACE pg_default;

ALTER TABLE public."RENT_PRICE"
OWNER to postgres;
```

SQL 代码 6 创建表 RENT_REG

```
CREATE TABLE public."RENT_REG"
    "regId" integer NOT NULL DEFAULT nextval(""RENT_REG_regId_seq"'::regclass),
    "cId" integer NOT NULL,
    "carId" integer NOT NULL,
    CONSTRAINT "RENT_REG_pkey" PRIMARY KEY ("regId", "cId", "carId"),
    CONSTRAINT "RENT_REG_fkey1" FOREIGN KEY ("cId")
       REFERENCES public."CLIENT" ("cId") MATCH SIMPLE
       ON UPDATE NO ACTION
       ON DELETE NO ACTION,
   CONSTRAINT "RENT_REG_fkey2" FOREIGN KEY ("carId")
       REFERENCES public."CAR" ("carId") MATCH SIMPLE
       ON UPDATE NO ACTION
       ON DELETE NO ACTION
WITH (
   OIDS = FALSE
TABLESPACE pg_default;
ALTER TABLE public."RENT_REG"
    OWNER to postgres;
```

▼ ♣ 登录/组角色 (13)

A R_Adminstrator



📤 R_Manager

A R_SalesMan

图 1 创建数据库角色

2) 在数据库 CarRentDB 中,分别定义各个角色对数据库表对象的访问权限。

分别定义各个角色对数据表对象的访问权限如表 1 所示,代码如 SQL 代码 7-11 所示,结果如图 2-6 所示。

表 1 角色权限表

Role	Table	INSERT	DELETE	UPDATE	SELECT	ELSE	WITH GRANT OPTION
R_Adminstrator	CAR	√	√	√	√	1	✓
	CLIENT	√	√	√	✓	√	✓
	RENT_FEE	✓	✓	√	✓	√	√
	RENT_PRICE	√	√	√	✓	√	√
	RENT_REG	✓	✓	✓	✓	√	✓
	CAR				✓		
	CLIENT		√	√	✓		
R_Client	RENT_FEE				✓		
	RENT_PRICE				√		
	RENT_REG				✓		
R_Manager	CAR	√	√	√	✓		✓
	CLIENT	√	✓	√	✓		√
	RENT_FEE	√	✓	√	✓		√
	RENT_PRICE	✓	✓	√	✓		√
	RENT_REG	✓	✓	√	\checkmark		✓
R_SalesMan	CAR	√	√	√	✓		
	CLIENT			√	✓		
	RENT_FEE	√	√	1	√		
	RENT_PRICE	√	√	√	√		
	RENT_REG	1	1	√	1		

SQL 代码 7 定义 CAR 角色权限

GRANT ALL ON TABLE public. "CAR" TO "R_Adminstrator" WITH GRANT OPTION;

GRANT SELECT ON TABLE public. "CAR" TO "R_Client";

GRANT UPDATE, DELETE, INSERT, SELECT ON TABLE public. "CAR" TO "R_Manager" WITH GRANT OPTION;

GRANT UPDATE, DELETE, INSERT, SELECT ON TABLE public."CAR" TO "R_SalesMan";

SQL 代码 8 定义 CLIENT 角色权限

GRANT ALL ON TABLE public. "CLIENT" TO "R_Adminstrator" WITH GRANT OPTION;

GRANT DELETE, SELECT, UPDATE ON TABLE public. "CLIENT" TO "R_Client";

GRANT DELETE, INSERT, SELECT, UPDATE ON TABLE public. "CLIENT" TO "R_Manager" WITH GRANT OPTION;

GRANT SELECT, UPDATE ON TABLE public."CLIENT" TO "R_SalesMan";

SQL 代码 9 定义 RENT_FEE 角色权限

GRANT ALL ON TABLE public."RENT_FEE" TO "R_Adminstrator" WITH GRANT OPTION;

GRANT SELECT ON TABLE public. "RENT_FEE" TO "R_Client";

GRANT UPDATE, DELETE, INSERT, SELECT ON TABLE public. "RENT_FEE" TO "R_Manager" WITH GRANT OPTION;

GRANT UPDATE, DELETE, INSERT, SELECT ON TABLE public. "RENT_FEE" TO "R_SalesMan";

SQL 代码 10 定义 RENT_PRICE 角色权限

GRANT ALL ON TABLE public. "RENT_PRICE" TO "R_Adminstrator" WITH GRANT OPTION;

GRANT SELECT ON TABLE public."RENT_PRICE" TO "R_Client";

GRANT UPDATE, DELETE, INSERT, SELECT ON TABLE public. "RENT_PRICE" TO "R_Manager" WITH GRANT OPTION:

GRANT UPDATE, DELETE, INSERT, SELECT ON TABLE public. "RENT_PRICE" TO "R_SalesMan";

SQL 代码 11 定义 RENT_REG 角色权限

GRANT ALL ON TABLE public. "RENT_REG" TO "R_Adminstrator" WITH GRANT OPTION;

GRANT SELECT ON TABLE public. "RENT_REG" TO "R_Client";

GRANT UPDATE, DELETE, INSERT, SELECT ON TABLE public."RENT_REG" TO "R_Manager" WITH GRANT OPTION;

GRANT UPDATE, DELETE, INSERT, SELECT ON TABLE public. "RENT_REG" TO "R_SalesMan";



图 2 CAR 角色权限

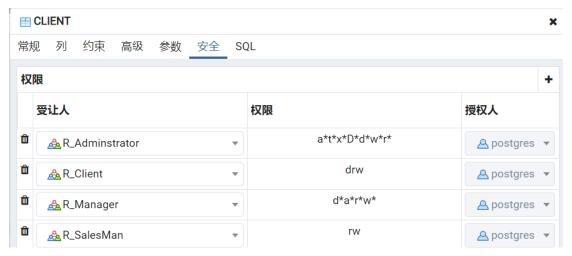


图 3 CLIENT 角色权限

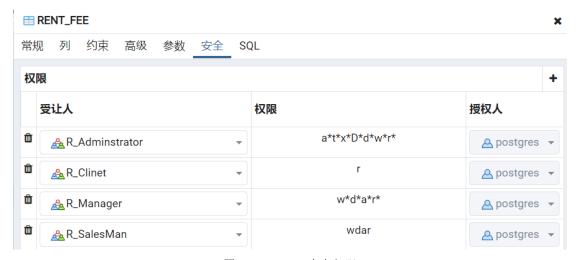


图 4 RENT_FEE 角色权限



图 5 RENT_PRICE 角色权限

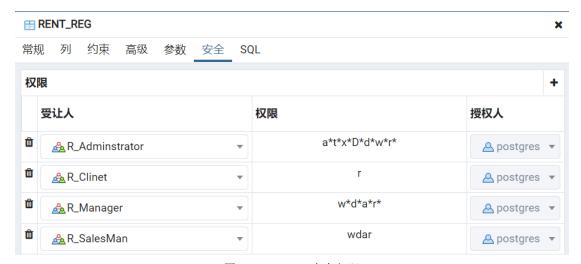


图 6 RENT_REG 角色权限

3) 创建用户 ClientUser 为客户角色用户,用户 SalesManUser 为业务员角色用户,用户 ManagerUser 为经理角色用户,用户 AdminstratorUser 为系统管理员角色用户。

分别创建登录用户代码如 SQL 代码 12-15 所示,结果如图 7 所示。

SQL 代码 12 创建 ClientUser 用户

CREATE ROLE "ClientUser" WITH

LOGIN

NOSUPERUSER

INHERIT

NOCREATEDB

NOCREATEROLE

NOREPLICATION

ENCRYPTED PASSWORD 'md5d4738e66c6e21ed727641a02b65a429c';

GRANT "R_Client" TO "ClientUser";

SQL 代码 13 创建 SalesManUser 用户

CREATE ROLE "SalesManUser" WITH

LOGIN

NOSUPERUSER

INHERIT

NOCREATEDB

NOCREATEROLE

NOREPLICATION

ENCRYPTED PASSWORD 'md50c0ab2f1111c8ff4ab4b90e8cf21aece';

GRANT "R_SalesMan" TO "SalesManUser";

SQL 代码 14 创建 ManagerUser 用户

CREATE ROLE "ManagerUser" WITH

LOGIN

NOSUPERUSER

INHERIT

NOCREATEDB

NOCREATEROLE

NOREPLICATION

 $ENCRYPTED\ PASSWORD\ 'md50f85ec9f219fb5fde38c0bd76a6b7fa5';$

GRANT "R_Manager" TO "ManagerUser";

SQL 代码 15 创建 AdminstratorUser 用户

CREATE ROLE "AdminstratorUser" WITH

LOGIN

NOSUPERUSER

INHERIT

NOCREATEDB

NOCREATEROLE

NOREPLICATION

ENCRYPTED PASSWORD 'md5efb68e8e2fbc184a9ddea123f6c6ed4e';

GRANT "R_Adminstrator" TO "AdminstratorUser";

▼ 4 登录/组角色 (17)

AdminstratorUser

ClientUser

A ManagerUser

A R_Adminstrator

A R_Client

📤 R_Manager

A R_SalesMan

SalesManUser

图 7 创建用户结果

4)分别以不同用户登录访问数据库,尝试进行不同类型访问操作。

首先以 ClientUser 的身份登录数据库,尝试查看数据表 CAR、删除数据表 CAR 中的一个数据和查看数据表 RENT_FEE 并更新数据。操作结果如图 8-11 所示。

```
Server [localhost]:
Database [postgres]: CarRentDB
Port [5432]:
Username [postgres]: ClientUser
用户 ClientUser 的口令:
psql (11.7)
输入 "help" 来获取帮助信息.
```

图 8 ClientUser 登录

CarRent carId		* FROM "CAR"; carInfo				
1 2 3 4 (4 行记	吉普 1 号 吉普 2 号 吉普 3 号 宝 宝 录)	尊贵的吉普 1 号,为尊贵的您 尊贵的吉普 2 号,为尊贵的您 尊贵的吉普 3 号,为尊贵的您 顶配的宝马 1 号,为追梦的您				

图 9 ClientUser 查看表 CAR

CarRentDB=> DELETE FROM "CAR" WHERE "carId"=1; ERROR: permission denied for table CAR

图 10 ClientUser 删除表 CAR 数据

图 11 ClientUser 查看表 RENT_FEE 并更新数据

由于 ClientUser 拥有查看数据表 CAR 和 RENT_FEE 的权限,所以可以看到数据,如图 9 和图 11 所示;没有删除表 CAR 或更新 RENT_FEE 的权限,所以会提示 ERROR,没有进行相关操作的权限,如图 10 和图 11 所示。

以 ManagerUser 的身份登录数据库,尝试查看表 CLIENT、插入数据到表 CLIENT 和删除表 CLIENT 的某一条数据。操作结果如图 12-14 所示。

图 12 ManagerUser 查看表 CLIENT

图 13 ManagerUser 插入表 CLIENT

```
CarRentDB=> DELETE FROM "CLIENT" WHERE "cId"=3;
ERROR: update or delete on table "CLIENT" violates foreign key constraint "RENT_REG_fkey1" on table "RENT_REG"
描述: Key (cId)=(3) is still referenced from table "RENT_REG".
```

图 14 ManagerUser 删除表 CLIENT 数据

由于 ManagerUser 拥有在 CLIENT 表插入和删除的权限, 但是 CLIENT 中的 cId 为 3 的数据是 RENT_REG 等表的外键约束, 所以无法成功删除, 如图 14 所示。

以 SalesManUser 的身份登录数据库,尝试在表 CLIENT 插入新数据和更新表 CLIENT 数据。操作结果如图 15 和图 16 所示。

```
CarRentDB=> INSERT INTO "CLIENT" ("cId", "cName", "cGender") VALUES (5, '阿多', '未知');
ERROR: permission denied for table CLIENT
```

图 15 SalesManUser 插入表 CLIENT

图 16 SalesManUser 更新表 CLIENT

由于 SalesManUser 拥有在表 CLIENT 进行 UPDATE 操作的权限,但没有进行 INSERT 操作的权限,因此在 CLIENT 表中成功更新数据,如图 16 所示,而插入数据时提示没有权限,如图 15 所示。

5) 以管理员身份进行 CarRentDB 数据库备份处理。分别创建数据库备份、schema 备份、数据库表备份。

以 postgres 的超级用户身份对 CarRentDB 数据库进行备份处理,分别创建数据库备份、schema 备份和数据库表(CAR)备份如图 17-20 所示。

C:\Program Files\PostgreSQL\11\bin>pg_dump -U postgres -f /CRDB.sql CarRentDB口令:

图 17 备份数据库

C:\Program Files\PostgreSQL\11\bin>pg_dump -U postgres -f /CRDB_SCHEMA.sql -s CarRentDB 口令:

图 18 备份 schema

C:\Program Files\PostgreSQL\11\bin>pg_dump -U postgres -f /CRDB_TABLE_CAR.sql -t "public.\"CAR\"" CarRentDB 다송:

图 19 备份数据库表 CAR

名称	修改日期	类型	大小
CRDB_TABLE_CAR.sql	2020/5/9 13:56	SQL Text File	3 KB
CRDB.sql	2020/5/9 11:57	SQL Text File	11 KB
CRDB_SCHEMA.sql	2020/5/9 11:49	SQL Text File	9 KB

图 20 生成的备份文件

6) 当破坏数据库 CarRentDB 后,使用备份文件进行数据库恢复处理。

以 AdministratorUser 的身份登录数据库,使用简单的手段对数据库进行破坏处理,删除表 RENT_FEE 和 RENT_REG 中的数据,如图 21 所示。接下来用 postgres 的身份使用备份的数据库文件 CRDB.sql 恢复数据库,如图 22 所示。恢复后数据库的结果如图 23 所示。

```
CarRentDB=> DELETE FROM "RENT_FEE" WHERE "rentFeeId">0;
DELETE 3
CarRentDB=> DELETE FROM "RENT_REG" WHERE "regId">0;
DELETE 3
```

图 21 试图破坏数据库

C:\Program Files\PostgreSQL\11\bin>psql -U postgres -f /CRDB.sql CarRentDB 图 22 恢复数据库

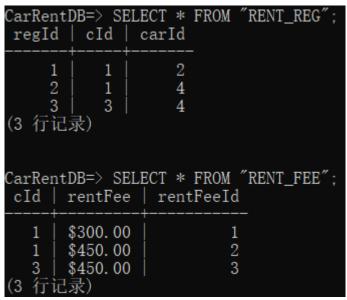


图 23 恢复数据库结果