数据库系统课程实验报告

实验名称:	实验七：数据库的完整性
实验日期:	2022/5/5
实验地点:	厦门大学德旺图书馆
提交日期:	2022/5/5

学号:	20420192201952
姓名:	庾晓萍
专业年级:	软工 2020 级
学年学期:	2021-2022 学年第二学期

1. 实验目的

- 理解数据库系统用户（user）、权限（privilege）和角色（role）的概念和作用
- 熟练掌握用户的管理：创建、查看、删除和权限的授予与回收
- 熟练掌握通过数据字典查看用户权限、表和视图权限的方法
- 熟练掌握使用 Grant 命令给用户、角色授权的方法
- 熟练掌握使用 Revoke 命令回收已授权限的方法
- 熟练掌握角色定义、重命名和删除的方法
- 熟练掌握修改角色中权限的方法
- 理解视图的安全性作用

2. 实验内容和步骤

一、创建两张表：雇员表 Emp 和工作表 Work：

```
sale=> CREATE TABLE Emp(
sale(>      Eid CHAR(5) NOT NULL,
sale(>      Ename VARCHAR2( 10 ),
sale(>      WorkID  CHAR( 3 ) ,
sale(>      Salary  NUMBER( 8,2 ) ,
sale(>      Phone CHAR(11) NOT NULL
sale(> );
CREATE TABLE
sale=> CREATE TABLE Work(
sale(>      WorkID CHAR(3) NOT NULL,
sale(>      LowerSalary  NUMBER( 8,2 ) ,
sale(>      UpperSalary  NUMBER( 8,2 )
sale(> );
CREATE TABLE
```

```
CREATE TABLE Emp(
    Eid CHAR(5) NOT NULL,
    Ename VARCHAR2( 10 ),
    WorkID  CHAR( 3 ) ,
    Salary  NUMBER( 8,2 ) ,
    Phone CHAR(11) NOT NULL
);

CREATE TABLE Work(
    WorkID CHAR(3) NOT NULL,
    LowerSalary  NUMBER( 8,2 ) ,
    UpperSalary  NUMBER( 8,2 )
);
```

二、 分别为两张表插入如下数据，查看插入操作是否成功。（成功）

```
Insert into  Emp values ('10001', 'Smith', '001', 2000, '13800010001');
Insert into  Emp values ('10001', 'Jonny', '001', 3000,'13600010002');
Insert into  Emp values ('10002', 'Mary', '002', 2500, '13800020002');
Insert into  Work values  ('001', 1000, 5000);
Insert into  Work values ('002', 2000, 8000);
```

```
sale=> Insert into  Emp values ('10001', 'Smith', '001', 2000, '13800010001');
INSERT 0 1
sale=> Insert into  Emp values ('10001', 'Jonny', '001', 3000,'13600010002');
INSERT 0 1
sale=> Insert into  Emp values ('10002', 'Mary', '002', 2500, '13800020002');
INSERT 0 1
sale=> Insert into  Work values  ('001', 1000, 5000);
INSERT 0 1
sale=> Insert into  Work values ('002', 2000, 8000);
INSERT 0 1
```

三、修改雇员表的结构，设置 Eid 为主码，主码名称为 eid_pk，查看该操作是否成功。

（不成功，因为之前插入的数据会使得主键重复，于是将 jonny 的编号改为 10004，最后设置主键成功）

```
sale=> ALTER TABLE Emp ADD CONSTRAINT eid_pk PRIMARY KEY(Eid);
NOTICE:  ALTER TABLE / ADD PRIMARY KEY will create implicit index "eid_pk" for table "emp"
ERROR:  could not create unique index "eid_pk"
DETAIL:  Key (eid)=(10001) is duplicated.
sale=>
```

```
UPDATE Emp SET Eid = '10004' WHERE (Ename='Jonny');
ALTER TABLE Emp ADD CONSTRAINT eid_pk PRIMARY KEY(Eid);
```

```
sale=> ALTER TABLE Emp ADD CONSTRAINT eid_pk PRIMARY KEY(Eid);
NOTICE:  ALTER TABLE / ADD PRIMARY KEY will create implicit index "eid_pk" for table "emp"
ALTER TABLE
```

四、将 eid 为主码的约束名 eid_pk 改为 pk_eid。

五、设置雇员表中的 phone 字段取唯一值，查看该操作是否成功。

若不成功说明原因。

```
ALTER TABLE Emp RENAME CONSTRAINT eid_pk to pk_eid;
ALTER TABLE Emp ADD CONSTRAINT phoneunique UNIQUE(phone);
```

```
sale=> ALTER TABLE Emp RENAME CONSTRAINT eid_pk to pk_eid;
ALTER TABLE
sale=> ALTER TABLE Emp ADD CONSTRAINT phoneunique UNIQUE(phone);
NOTICE:  ALTER TABLE / ADD UNIQUE will create implicit index "phoneunique" for table "emp"
ALTER TABLE
```

六、给雇员表添加一条新记录(‘10003’,’Amy’,’002’,3000,’13800020003’)，查看执行结果。

```
sale=> Insert into  Emp values ('10003','Amy','002', 3000,'13800020003');
INSERT 0 1
```

七、设置工作表的 WorkID 为主码。

八、修改雇员表，设置雇员表的 WorkID 字段为外码，它引用工作表中的 WorkID 字段，查看操作是否成功。若不成功说明原因。

九、给雇员表添加一条新记录(‘10003’,’Amy’, ‘003’,3000, ‘13800020003’)，查看操作是否成功。若不成功说明原因。

（插入数据不成功，因为之前已经插入过('10003','Amy','002', 3000,'13800020003');主键不可以重复）

```
ALTER TABLE Work ADD CONSTRAINT work_pk PRIMARY KEY(WorkID);
ALTER TABLE Emp ADD CONSTRAINT fk_emp_work  FOREIGN KEY(WorkID)
REFERENCES Work(WorkID);
Insert into  Emp values ('10003','Amy', '003', 3000, '13800020003');
```

```
sale=> ALTER TABLE Work ADD CONSTRAINT work_pk PRIMARY KEY(WorkID);
NOTICE:  ALTER TABLE / ADD PRIMARY KEY will create implicit index "work_pk" for table "work"
ALTER TABLE
sale=> ALTER TABLE Emp ADD CONSTRAINT fk_emp_work  FOREIGN KEY(WorkID)
sale-> REFERENCES Work(WorkID);
ALTER TABLE
```

```
sale=> Insert into  Emp values ('10003','Amy', '003', 3000, '13800020003');
ERROR:  duplicate key value violates unique constraint "pk_eid"
DETAIL:  Key (eid)=(10003) already exists.
```

十、在雇员表中，设置雇员工资必须大于或等于 1000。查看操作是否成功。若不成功说明原因。

十一、给雇员表添加一条新记录(‘10003’,’Robert’,‘002’,500,

‘13800020003’），查看执行操作是否成功。若不成功说明原因。

（不成功，因为插入的员工其薪水小于 1000，违反了约束所以无法插入）

```
ALTER TABLE Emp ADD CONSTRAINT checksalary CHECK (Salary>=1000);
Insert into  Emp values ('10003','Robert','002',500,'13800020003');
```

```
sale=> ALTER TABLE Emp ADD CONSTRAINT checksalary CHECK (Salary>=1000);
ALTER TABLE
sale=> Insert into  Emp values ('10003','Robert','002',500,'13800020003');
ERROR:  new row for relation "emp" violates check constraint "checksalary"
DETAIL:  Failing row contains (10003, Robert, 002, 500.00, 13800020003).
```

十二、在工作表中，设置其最低工资不超过最高工资。

十三、给工作表添加一条新记录(‘002’,4000,3000)，查看操作是否成功。若不成功说明原因。

（不成功，因为最低工资超过最高工资，违反了约束所以无法插入）

```
ALTER TABLE Work  ADD CONSTRAINT lowerupper CHECK (LowerSalary<=UpperSalary);
Insert into  Work values ('002',4000,3000);
```

```
sale=> ALTER TABLE Work  ADD CONSTRAINT lowerupper CHECK (LowerSalary<=UpperSalary);
ALTER TABLE
sale=> Insert into  Work values ('002',4000,3000);
ERROR:  new row for relation "work" violates check constraint "lowerupper"
DETAIL:  Failing row contains (002, 4000.00, 3000.00).
```

十四、通过查看 openGauss 的系统表 pg_constraints 了解表上的约束。

十五、通过 gsql 命令\d+ table_name 查看改表上的约束定义。

```
select oid, conname from pg_constraint; -- conname是约束名 --
select pg_get_constraintdef(16856);
\d+ Emp
\d+ Work
```

```
  oid   |              conname
--------+-----------------------------------
 14343  | cardinal_number_domain_check
 14351  | yes_or_no_check
 16649  | pk
 16651  | ctpk
 16652  | fk_countries_regions
 16658  | loctpk
 16659  | fk_locations_countries
 16665  | employeepk
 16666  | fk_employees_manager
 16672  | warepk
 16673  | fk_warehouses_locations
 16679  | category_idpk
 16681  | product_idpk
 16682  | fk_products_categories
 16688  | customers_idpk
 16690  | contactspk
 16691  | fk_contacts_customers
 16697  | orderspk
 16698  | fk_orders_customers
 16703  | fk_orders_employees
 16709  | pk_order_items
 16710  | fk_order_items_products
 16715  | fk_order_items_orders
 16721  | pk_inventories
 16722  | fk_inventories_products
 16727  | fk_inventories_warehouses
 16743  | discounts_pkey
 16748  | palette_a_pkey
 16753  | palette_b_pkey
 16796  | student_pkey
--More--
```

```
sale=> select pg_get_constraintdef(16856);
          pg_get_constraintdef
---------------------------------------
 CHECK ((lowersalary <= uppersalary))
(1 row)
```

```
sale=> \d+ Emp
                                    Table "public.emp"
 Column |          Type          | Modifiers | Storage  | Stats target | Description
--------+------------------------+-----------+----------+--------------+-------------
 eid    | character(5)           | not null  | extended |              |
 ename  | character varying(10)  |           | extended |              |
 workid | character(3)           |           | extended |              |
 salary | numeric(8,2)           |           | main     |              |
 phone  | character(11)          | not null  | extended |              |
Indexes:
    "pk_eid" PRIMARY KEY, btree (eid) TABLESPACE pg_default
    "phoneunique" UNIQUE CONSTRAINT, btree (phone) TABLESPACE pg_default
Check constraints:
    "checksalary" CHECK (salary >= 1000::numeric)
Foreign-key constraints:
    "fk_emp_work" FOREIGN KEY (workid) REFERENCES work(workid)
Has OIDs: no
Options: orientation=row, compression=no
```

```
sale=> \d+ Work
                                   Table "public.work"
    Column     |      Type      | Modifiers | Storage  | Stats target | Description
---------------+----------------+-----------+----------+--------------+-------------
 workid        | character(3)   | not null  | extended |              |
 lowersalary   | numeric(8,2)   |           | main     |              |
 uppersalary   | numeric(8,2)   |           | main     |              |
Indexes:
    "work_pk" PRIMARY KEY, btree (workid) TABLESPACE pg_default
Check constraints:
    "lowerupper" CHECK (lowersalary <= uppersalary)
Referenced by:
    TABLE "emp" CONSTRAINT "fk_emp_work" FOREIGN KEY (workid) REFERENCES work(workid)
Has OIDs: no
Options: orientation=row, compression=no
```

十六、删除雇员表的所有约束，包括主码约束、外码约束和其他约束。

十七、删除工作表所有约束，包括主码约束。

```
ALTER TABLE Emp DROP CONSTRAINT pk_eid;
ALTER TABLE Emp DROP CONSTRAINT phoneunique;
ALTER TABLE Emp DROP CONSTRAINT fk_emp_work;
ALTER TABLE Emp DROP CONSTRAINT checksalary;
ALTER TABLE Work DROP CONSTRAINT lowerupper;
ALTER TABLE Work DROP CONSTRAINT work_pk;
```

```
sale=> ALTER TABLE Emp DROP CONSTRAINT pk_eid;
ALTER TABLE
sale=> ALTER TABLE Emp DROP CONSTRAINT phoneunique;
ALTER TABLE
sale=> ALTER TABLE Emp DROP CONSTRAINT fk_emp_work;
ALTER TABLE
sale=> ALTER TABLE Emp DROP CONSTRAINT checksalary;
ALTER TABLE
sale=> ALTER TABLE Work DROP CONSTRAINT lowerupper;
ALTER TABLE
sale=> ALTER TABLE Work DROP CONSTRAINT work_pk;
ALTER TABLE
```

3. 实验总结

3.1 实验思考

• openGauss 实现完整性规则的机制是什么？在 SQL 语句中实现完整性规则的常见约束有哪些？各自适应什么业务场景？

答：完整性实现的机制：①完整性约束定义机制（Primary key，Foreign key，Check，Not null，Unique）②完整性检查机制：违背完整性约束条件时关系数据库管理系统应采取的动作。其中触发器用于实现未被 SQL 约束机制指定的某些更复杂完整性约束。

3.2 对实验的认识

通过实验我对 openGauss 中的一些语句更熟悉了。如

如 SELECT * FROM customer_t1;可以用来查询表 customer_t1 的所有数据。gsql -d sale -p 26000 -U yuxiaoping -W yuxiaoping@123 -r 或者 gsql -d sale -p 26000 -U user1 -W user1@123 -r 可以用来将新用户连接到数据库。可以使用 gsql -d postgres -p 26000 -r 连接到 postgres。gs_om -t start 可以开启数据库。

3.3 遇到的困难及解决方法

要更改当前会话的默认 Schema，请使用 SET 命令。执行如下命令

SET SEARCH_PATH To icebear,public;将搜索路径设置为 myschema、public，首先搜索 myschema。

```
sale=> SET SEARCH_PATH TO icebear, public;
SET
```

高斯默认有 session 超时时间，若想要 session 一直保持，需要修改

配置项：ALTER DATABASE sale SET session_timeout TO 0;

```
postgres=# ALTER DATABASE postgres SET session_timeout TO 0;
ALTER DATABASE
```