数据库系统课程实验报告

实验名称:	数据库的完整性
实验日期:	2022.12.9
实验地点:	四号楼
提交日期:	2022.12.10

学号:	22920202202877
姓名:	陈鑫蕾
专业年级:	数媒 2020 级
学年学期:	2022-2023 学年第一学期

1. 实验目的

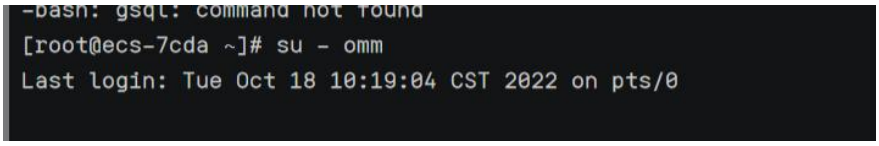
- 理解并掌握关系数据库完整性的运行机制
  - 完整性约束定义>完整性约束检查>违约处理
- 理解并掌握关系数据库完整性主要约束类型及其含义和作用
  - PRIMARY KEY，FOREIGN KEY，NOT NULL，UNIQUE，CHECK
- 理解并掌握关系数据库完整性定义、修改、删除和重命名的方法
  - CREATE TABLE，ALTER TABLE
- 熟练掌握 openGauss 下通过系统表 pg_constraint 查看完整性信息的方法
- 熟练掌握 openGauss 下通过查看表结构来查看主外码信息的方法
- 熟练掌握 openGauss 下通过查看完整性约束定义的方法

2. 实验内容和步骤

（1）连接到数据库

步骤如下:

1.在数据库主节点服务器上，切换至 omm 操作系统用户环境。



2.查看服务是否启动。

```
[omm@ecs-7cda ~]$  gs_om -t status
-----------------------------------------------------------------
--------

cluster_name     : dbCluster
cluster_state    : Normal
redistributing   : No

-----------------------------------------------------------------
--------
```

3.启动数据库服务

```
[omm@ecs-7cda ~]$ gs_om -t start
Starting cluster.
=========================================
[SUCCESS] ecs-7cda:
[2022-10-18 12:26:33.753][14929][][gs_ctl]: gs_ctl started,data
dir is /gaussdb/data/db1
[2022-10-18 12:26:33.756][14929][][gs_ctl]:  another server mig
ht be running; Please use the restart command
=========================================
Successfully started.
```

4.连接数据库

```
[omm@ecs-7cda ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 2.0.0 build 78689da9) compiled at 2021-03-31 2
1:03:52 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requirin
g high-security)
Type "help" for help.
```

（2）数据库安全性

1.创建两张表：雇员表 Emp 和工作表 Work，它们的表结构如下：

```
postgres=# CREATE TABLE Emp(
postgres(#     Eid varchar(5) NOT NULL,
postgres(#     Ename varchar(10),
postgres(#     WorkID char(3),
postgres(#     Salary numeric(8,2),
postgres(#     Phone char(11)
postgres(# );
CREATE TABLE
postgres=#
```

```
postgres=# CREATE TABLE Work(
postgres(#     WorkID char(3),
postgres(#     LowerSalary numeric(8,2),
postgres(#     UpperSalary numeric(8,2)
postgres(# );
CREATE TABLE
```

2.分别为两张表插入如下数据，查看插入操作是否成功。

```
postgres=# INSERT INTO Emp
postgres-# VALUES('10001', 'Smith', '001', 200
0, '13800010001'),('10001', 'Jonny', '001', 30
00,'13600010002'),('10002', 'Mary', '002', 250
0,  '13800020002');
INSERT 0 3
```

```
postgres=# INSERT INTO Work
postgres-# VALUES('001', 1000, 5000),('002', 2
000, 8000);
INSERT 0 2
```

3.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。若不成功，请说明原因并思考如何处理才能使得添加约束成功。

```
postgres=# ALTER TABLE Emp
postgres-# ADD CONSTRAINT eid_pk PRIMARY KEY(E
id);
NOTICE:  ALTER TABLE / ADD PRIMARY KEY will cr
eate implicit index "eid_pk" for table "emp"
ERROR:  could not create unique index "eid_pk"
DETAIL:  Key (eid)=(10001) is duplicated.
```

添加失败，因为数据中有两个重复的 eid

所以将其中一个 eid 修改为‘10000’后

```
postgres=# UPDATE Emp
postgres-# SET Eid='10000'
postgres-# WHERE Ename='Smith';
UPDATE 1
```

再次添加约束，即可添加成功

```
postgres=# ALTER TABLE Emp
postgres-# ADD CONSTRAINT eid_pk PRIMARY KEY(E
id);
NOTICE:  ALTER TABLE / ADD PRIMARY KEY will cr
eate implicit index "eid_pk" for table "emp"
ALTER TABLE
```

4.将 eid 为主码的约束名 eid_pk 改为 pk_eid。

```
postgres=# ALTER TABLE Emp
postgres-# RENAME CONSTRAINT eid_pk to ok_eid;
ALTER TABLE
```

5.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。

```
postgres=# ALTER TABLE Emp
postgres-# ADD CONSTRAINT emp_u UNIQUE(Phone);
NOTICE:  ALTER TABLE / ADD UNIQUE will create
implicit index "emp_u" for table "emp"
ALTER TABLE
```

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

```
postgres=# INSERT INTO Emp
postgres-# VALUES('10003','Amy','002',3000,'13
800020003');
INSERT 0 1
```

7.设置工作表的 WorkID 为主码。

```
postgres=# ALTER TABLE Work
postgres-# ADD PRIMARY KEY(WorkID);
NOTICE:  ALTER TABLE / ADD PRIMARY KEY will cr
eate implicit index "work_pkey" for table "wor
k"
ALTER TABLE
```

8.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。

```
postgres=# ALTER TABLE Emp
postgres-# ADD CONSTRAINT emp_fk FOREIGN KEY (
WorkID) REFERENCES Work(WorkID);
ALTER TABLE
```

9.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。

```
postgres=# INSERT INTO Emp
postgres-# VALUES('10003','Amy','003',3000,'13
800020003');
ERROR:  duplicate key value violates unique co
nstraint "ok_eid"
DETAIL:  Key (eid)=(10003) already exists.
```

不成功，因为已经存在 eid=‘10003’的记录了，所以插入不成功

10.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。

```
postgres=# ALTER TABLE Emp
postgres-# ADD CONSTRAINT emp_salary_ck CHECK(
Salary>=1000);
ALTER TABLE
```

11.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。

先删除 eid=‘10003’的记录，避免因为主码重复插入不成功

```
postgres=# delete from emp where eid='10003';
DELETE 1
postgres=#
```

删除后发现仍然插入失败，原因是因为员工工资低于 1000，违反了完整性约束，所以插入失败

```
postgres=# INSERT INTO Emp
postgres-# VALUES('10003','Robert','002',500,'
13800020003');
ERROR:  new row for relation "emp" violates ch
eck constraint "emp_salary_ck"
DETAIL:  Failing row contains (10003, Robert,
002, 500.00, 13800020003).
```

12.在工作表中，设置其最低工资不超过最高工资。

```
postgres=# ALTER TABLE Work
postgres-# ADD CONSTRAINT work_salary_ck CHECK
 (LowerSalary<=UpperSalary);
ALTER TABLE
```

13.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。

```
postgres=# INSERT INTO Work
postgres-# VALUES('002',4000,3000);
ERROR:  new row for relation "work" violates c
heck constraint "work_salary_ck"
DETAIL:  Failing row contains (002, 4000.00, 3
000.00).
```

插入失败，因为插入的数据中最低工资高于最高工资，违反了完整性约束，所以插入失败

14.通过查看 openGauss 的系统表 pg_constraints 了解表上的约束。

```
postgres=# \d pg_constraint;
     Table "pg_catalog.pg_constraint"
    Column     |     Type     | Modifiers
---------------+--------------+-----------
 conname       | name         | not null
 connamespace  | oid          | not null
 contype       | "char"       | not null
 condeferrable | boolean      | not null
 condeferred   | boolean      | not null
 convalidated  | boolean      | not null
 conrelid      | oid          | not null
 contypid      | oid          | not null
 conindid      | oid          | not null
 confrelid     | oid          | not null
 confupdtype   | "char"       | not null
 confdeltype   | "char"       | not null
 confmatchtype | "char"       | not null
 conislocal    | boolean      | not null
 coninhcount   | integer      | not null
 connoinherit  | boolean      | not null
 consoft       | boolean      | not null
 conopt        | boolean      | not null
 conkey        | smallint[]   |
 confkey       | smallint[]   |
 conpfeqop     | oid[]        |
```

```
 conpfeqop     | oid[]        |
 conppeqop     | oid[]        |
 conffeqop     | oid[]        |
 conexclop     | oid[]        |
 conbin        | pg_node_tree |
 consrc        | text         |
 conincluding  | smallint[]   |
Indexes:
    "pg_constraint_oid_index" UNIQUE, btree (oid) TABLESPACE pg_default
    "pg_constraint_conname_nsp_index" btree (conname, connamespace) TABLESPACE pg_default
    "pg_constraint_conrelid_index" btree (conrelid) TABLESPACE pg_default
    "pg_constraint_contypid_index" btree (contypid) TABLESPACE pg_default
Replica Identity: NOTHING
```

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

```
postgres=# \d Emp;
            Table "public.emp"
 Column |         Type          | Modifiers
--------+-----------------------+-----------
 eid    | character varying(5)  | not null
 ename  | character varying(10) |
 workid | character(3)          |
 salary | numeric(8,2)          |
 phone  | character(11)         |
Indexes:
    "ok_eid" PRIMARY KEY, btree (eid) TABLESPACE pg_default
    "emp_u" UNIQUE CONSTRAINT, btree (phone) TABLESPACE pg_default
Check constraints:
    "emp_salary_ck" CHECK (salary >= 1000::numeric)
Foreign-key constraints:
    "emp_fk" FOREIGN KEY (workid) REFERENCES work(workid)
```

```
ALTER TABLE
postgres=# \d Work;
           Table "public.work"
    Column     |     Type      | Modifiers
---------------+---------------+-----------
 workid        | character(3) | not null
 lowersalary | numeric(8,2) |
 uppersalary | numeric(8,2) |
Indexes:
    "work_pkey" PRIMARY KEY, btree (workid) TA
BLESPACE pg_default
Check constraints:
    "work_salary_ck" CHECK (lowersalary <= upp
ersalary)
```

16.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。

```
postgres=# ALTER TABLE Emp
postgres-# DROP CONSTRAINT ok_eid;
ALTER TABLE
postgres=# ALTER TABLE Emp
postgres-# DROP CONSTRAINT emp_u;
ALTER TABLE
postgres=# ALTER TABLE Emp
postgres-# DROP CONSTRAINT emp_fk;
ALTER TABLE
postgres=# ALTER TABLE Emp
postgres-# DROP CONSTRAINT emp_salary_ck;
ALTER TABLE
```

17.删除工作表所有约束，包括主码约束。

```
postgres=# ALTER TABLE Work
postgres-# DROP CONSTRAINT work_pkey;
ALTER TABLE
postgres=# ALTER TABLE Work
postgres-# DROP CONSTRAINT work_salary_ck;
ALTER TABLE
```

（3）思考题：

· openGauss 实现完整性规则的机制是什么？

通过设置完整性约束来是实现完整性约束；

·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？

有三种常见约束：实体完整性约束，参照完整性约束，用户定义的完整性约束；

实体完整性适用于需要主键，或某值需取唯一值时；

参照完整性适用于某值需要参照另外一个表中的主码的时候，通常用于表达表之间的关系；

用户定义的完整性约束适用于用户的特定需求；

3.实验总结

3.1 完成的工作

创建两张表；

为表插入数据；

为表添加主键；

修改主键的约束名；

为表添加 unique 约束；

为表添加外键;

为表定义 check 语句;

删除表的约束;

3.2 对实验的认识

掌握了表的约束的添加方法;

修改表的约束名的方法;

当不满足完整性约束时，数据库会拒绝操作;

删除表的约束的方法;

3.3 遇到的困难及解决方法

无