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:	2023/5/19
实验地点:	文宣楼 A402
提交日期:	2023/5/19

学号:	22920212204392
姓名:	黄勖
专业年级:	软工 2021 级
学年学期:	2022-2023 学年第二学期

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

（0）登录 ECS 服务器，以 omm 操作系统管理员身份登录数据库，使用 gsql 连接到数据库。

su - omm

gs_om -t start

gsql -d postgres -p 26000 -r

（gsql -d sales -p 26000 -U hx -W HX@123pass -r）

```
[root@ecs-hxnb ~]# su - omm
Last login: Wed May 17 22:43:46 CST 2023 on pts/0


Welcome to 4.19.90-2110.8.0.0119.oe1.aarch64

System information as of time:  Fri May 19 16:53:17 CST 2023

System load:    2.02
Processes:      150
Memory used:    10.4%
Swap used:      0.0%
Usage On:       14%
IP address:     192.168.0.99
Users online:   1



[omm@ecs-hxnb ~]$ gs_om -t start
Starting cluster.
=========================================
[SUCCESS] ecs-hxnb
2023-05-19 16:53:19.537 646738ff.1 [unknown] 281458419564560 [unk
=========================================
Successfully started.
```

(1)创建两张表:雇员表 Emp 和工作表 Work,它们的表结构如下:

Emp 表

字段	含义	数据类型	是否空
Eid	雇员编号	定长字符型，长度为 5	否
Ename	雇员姓名	变长字符型，长度为 10	/
WorkID	工作编号	定长字符，长度为 3	/
Salary	工资	数值型，总长度为 8，包括两位小数	/
Phone	电话号码	定长字符型，长度为 11	否

Work 表

字段	含义	数据类型	是否空
WorkID	工作编号	定长字符，长度为 3	否
LowerSalary	最低工资	数值型，总长度为 8，包括两位小数	/
UpperSalary	最高工资	数值型，总长度为 8，包括两位小数	/

CREATE TABLE Emp(Eid CHARACTER(5) NOT NULL, Ename VARCHAR(10), WorkID CHAR(3), Salary NUMERIC(8,2), Phone CHAR(11) NOT NULL);

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

```
sales=> CREATE TABLE Emp(Eid CHARACTER(5) NOT NULL, Ename VARCHAR(10), WorkID CHAR(3), Salary NUMERIC(8,2), Phone CHAR(11) NOT NULL);
CREATE TABLE
sales=> CREATE TABLE Work(WorkID CHAR(3) NOT NULL, LowerSalary NUMERIC(8,2), UpperSalary NUMERIC(8,2));
CREATE TABLE
```

创建成功。

（2）分别为两张表插入如下数据，查看插入操作是否成功。

雇员表数据：{(‘10001‘, ’Smith‘, ’001‘, 2000, ’13800010001‘),(’10001‘, ’Jonny‘, ’001‘, 3000,’13600010002‘),

(’10002‘, ’Mary‘, ’002‘, 2500, ’13800020002‘) }

工作表数据：{ (‘001‘, 1000, 5000)，(’002‘, 2000, 8000) }

INSERT INTO Emp VALUES ('10001', 'Smith', '001', 2000, '13800010001'), ('10001', 'Jonny', '001', 3000, '13600010002'), ('10002', 'Mary', '002', 2500, '13800020002');

INSERT INTO Work VALUES ('001', 1000, 5000), ('002', 2000, 8000);

```
sales=> INSERT INTO Emp VALUES ('10001', 'Smith', '001', 2000, '13800010001'),
INSERT 0 3
sales=> INSERT INTO Work VALUES ('001', 1000, 5000), ('002', 2000, 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成功添加约束。要求：所有约束都要显式给出约束名，不可由系统默认，因为删除约束时需要用到约束名。

ALTER TABLE Emp ADD CONSTRAINT eid_pk PRIMARY KEY(Eid);

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

此时不能添加约束，需要将数据中 Eid 重复的 10001 的数据进行修改，使其 Eid 不相同。

UPDATE Emp SET Eid='10010' WHERE Ename='Jonny';

ALTER TABLE Emp ADD CONSTRAINT eid_pk PRIMARY KEY(Eid);

```
sales=> UPDATE Emp SET Eid='10010' WHERE Ename='Jonny';
UPDATE 1
sales=> 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
```

（4）将 eid 为主码的约束名 eid_pk 改为 pk_eid。

ALTER TABLE Emp DROP CONSTRAINT eid_pk;

ALTER TABLE Emp ADD CONSTRAINT pk_eid PRIMARY KEY(Eid);

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

（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。

ALTER TABLE Emp ADD CONSTRAINT uni_phone UNIQUE(Phone);

```
NOTICE:  ALTER TABLE / ADD UNIQUE will create implicit index "uni_phone" for table "emp"
ALTER TABLE
```

成功

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

INSERT INTO Emp VALUES ('10003', 'Amy', '002', 3000, '13800020003');

```
sales=> INSERT INTO Emp VALUES ('10003', 'Amy', '002', 3000, '13800020003');
INSERT 0 1
```

成功

（7）设置工作表的 WorkID 为主码。

ALTER TABLE Work ADD CONSTRAINT pk_workid PRIMARY KEY(WorkID);

```
sales=> ALTER TABLE Work ADD CONSTRAINT pk_workid PRIMARY KEY(WorkID);
NOTICE:  ALTER TABLE / ADD PRIMARY KEY will create implicit index "pk_workid" for table "work"
ALTER TABLE
```

成功

（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。

ALTER TABLE Emp ADD CONSTRAINT fk_emp FOREIGN KEY(WorkID) REFERENCES Work(WorkID);

```
sales=> ALTER TABLE Emp ADD CONSTRAINT fk_emp FOREIGN KEY(WorkID) REFERENCES Work(WorkID)
ALTER TABLE
```

成功

（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。

INSERT INTO Emp VALUES ('10003', 'Amy', '003', 3000, '13800020003');

```
sales=> INSERT INTO Emp VALUES ('10003', 'Amy', '003', 3000, '13800020003');
ERROR:  duplicate key value violates unique constraint "pk_eid"
DETAIL:  Key (eid)=(10003) already exists.
```

失败了，因为表中有与该纪律 Eid 相同的 10003 数据了，不符合主码约束，无法插入。

同时，电话部分也会报错，因为不符合先前设定的 UNIQUE 约束

（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。

ALTER TABLE Emp ADD CONSTRAINT ck_emp CHECK(Salary>=1000);

```
sales=> ALTER TABLE Emp ADD CONSTRAINT ck_emp CHECK(Salary>=1000);
ALTER TABLE
```

成功

（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。

INSERT INTO Emp VALUES ('10003', 'Robert', '002', 500, '13800020003');

```
sales=> INSERT INTO Emp VALUES ('10003', 'Robert', '002', 500, '13800020003');
ERROR:  new row for relation "emp" violates check constraint "ck_emp"
DETAIL:  Failing row contains (10003, Robert, 002, 500.00, 13800020003).
```

操作失败了,因为其工资不大于1000,Eid重复不符合主码约束、Phone不符合UNIQUE约束。

(12) 在工作表中，设置其最低工资不超过最高工资。

ALTER TABLE Work ADD CONSTRAINT ck_work CHECK(LowerSalary<UpperSalary);

```
sales=> ALTER TABLE Work ADD CONSTRAINT ck_work 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。

INSERT INTO Work VALUES ('002', 4000, 3000);

```
sales=> INSERT INTO Work VALUES ('002', 4000, 3000);
ERROR:  new row for relation "work" violates check constraint "ck_work"
DETAIL:  Failing row contains (002, 4000.00, 3000.00).
```

操作不成功,因为其最高工资低于最低工资,不满足上面添加的约束。

(14)通过查看 openGauss 的系统表 pg_constraint 了解表上的约束。

SELECT conname, contype, consrc FROM pg_constraint;

```
sales=> SELECT conname, contype, consrc FROM pg_constraint;
            conname             | contype |                                     consrc
--------------------------------+---------+-----------------------------------------------------------------------------------------------
 cardinal_number_domain_check   | c       | (VALUE >= 0)
 yes_or_no_check                | c       | ((VALUE)::text = ANY ((ARRAY['YES'::character varying, 'NO'::character varying])::text[]))
 regions_pkey                   | p       |
 countries_pkey                 | p       |
 countries_region_id_fkey       | f       |
 locations_pkey                 | p       |
 locations_country_id_fkey      | f       |
 warehouses_pkey                | p       |
 warehouses_location_id_fkey    | f       |
 product_categories_pkey        | p       |
 products_pkey                  | p       |
 products_category_id_fkey      | f       |
 inventories_pkey               | p       |
 inventories_product_id_fkey    | f       |
 inventories_warehouse_id_fkey  | f       |
 customers_pkey                 | p       |
 contacts_pkey                  | p       |
 contacts_customer_id_fkey      | f       |
 employees_pkey                 | p       |
 orders_pkey                    | p       |
 orders_customer_id_fkey        | f       |
 orders_salesman_id_fkey        | f       |
 order_items_pkey               | p       |
 order_items_product_id_fkey    | f       |
 order_items_order_id_fkey      | f       |
 discounts_pkey                 | p       |
 palette_a_pkey                 | p       |
 palette_b_pkey                 | p       |
 pk_eid                         | p       |
 uni_phone                      | u       |
 pk_workid                      | p       |
 fk_emp                         | f       |
 ck_emp                         | c       | (salary >= (1000)::numeric)
 ck_work                        | c       | (lowersalary < uppersalary)
(34 rows)
```

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

\d+ Emp;

```
sales=> \d+ Emp;
                                Table "sales.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
    "uni_phone" UNIQUE CONSTRAINT, btree (phone) TABLESPACE pg_default
Check constraints:
    "ck_emp" CHECK (salary >= 1000::numeric)
Foreign-key constraints:
    "fk_emp" FOREIGN KEY (workid) REFERENCES work(workid)
Has OIDs: no
Options: orientation=row, compression=no
```

\d+ Work;

```
sales=> \d+ Work;
                                Table "sales.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:
    "pk_workid" PRIMARY KEY, btree (workid) TABLESPACE pg_default
Check constraints:
    "ck_work" CHECK (lowersalary < uppersalary)
Referenced by:
    TABLE "emp" CONSTRAINT "fk_emp" FOREIGN KEY (workid) REFERENCES work(workid)
Has OIDs: no
Options: orientation=row, compression=no
```

(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。

ALTER TABLE Emp DROP CONSTRAINT pk_eid;

ALTER TABLE Emp DROP CONSTRAINT uni_phone;

ALTER TABLE Emp DROP CONSTRAINT fk_emp;

ALTER TABLE Emp DROP CONSTRAINT ck_emp;

```
sales=> ALTER TABLE Emp DROP CONSTRAINT pk_eid;
ALTER TABLE
sales=> ALTER TABLE Emp DROP CONSTRAINT uni_phone;
ALTER TABLE
sales=> ALTER TABLE Emp DROP CONSTRAINT fk_emp;
ALTER TABLE
sales=> ALTER TABLE Emp DROP CONSTRAINT ck_emp;
ALTER TABLE
```

(17) 删除工作表所有约束，包括主码约束。

ALTER TABLE Work DROP CONSTRAINT pk_workid;

ALTER TABLE Work DROP CONSTRAINT ck_work;

```
sales=> ALTER TABLE Work DROP CONSTRAINT pk_workid;
ALTER TABLE
sales=> ALTER TABLE Work DROP CONSTRAINT ck_work;
ALTER TABLE
```

3.实验总结

3.1 完成的工作

设计正确的 SQL 语句并完成了所有数据完整性操作修改等操作。

3.2 对实验的认识

(1)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？各自适用什么业务场景？

OpenGauss 中的完整性规则机制： OpenGauss 使用了传统的关系型数据库的完整性规则机制，它主要通过约束（constraints）来实现。约束是一组规则，用于限制数据库表中数据的合法性和完整性。当对表进行数据操作时，OpenGauss 会自动检查并执行这些约束，以确保数据的完整性。

常见的完整性约束及其适用场景：

1. 主键约束（Primary Key Constraint）：
   - 作用：保证表中的某个字段（或字段组合）唯一标识一条记录。
   - 适用场景：适用于需要确保数据唯一性并且能够快速通过主键查询的场景。
2. 唯一约束（Unique Constraint）：
   - 作用：确保表中的某个字段（或字段组合）的值唯一，但可以允许空值。
   - 适用场景：适用于需要保证数据唯一性，但允许空值的情况。
3. 外键约束（Foreign Key Constraint）：
   - 作用：创建表与表之间的关系，确保一个表中的某

个字段（外键）的值在另一个表的主键中存在。

•适用场景：适用于创建表与表之间的关联关系，并保证数据的完整性。

4. 检查约束（Check Constraint）：

•作用：定义表中字段的取值范围或满足某个条件的约束。

•适用场景：适用于需要对字段值进行条件限制的情况，如限制年龄大于等于 18、性别只能为男或女等。

这些完整性约束可以在 CREATE TABLE 语句中使用，或者使用 ALTER TABLE 语句添加到已存在的表中。根据具体业务需求，可以选择适合的约束来确保数据的完整性和一致性。

（2）收获

这次实验我通过实验掌握了关系数据库完整性的运行机制，理解了握关系数 据库完整性主要约束类型及其含义和作用以及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。

3.3 遇到的困难及解决方法

无。