实验报告	
题目	DBMS 软件安装及数据定义语句

实验环境（计算机配置，操作系统，编程语言，编程工具等）

服务器：华为弹性云服务器 ECS

配置：2vCPUs | 4GiB | kc1.large.2 | openEuler 20.03 64bit with ARM

操作系统：Linux

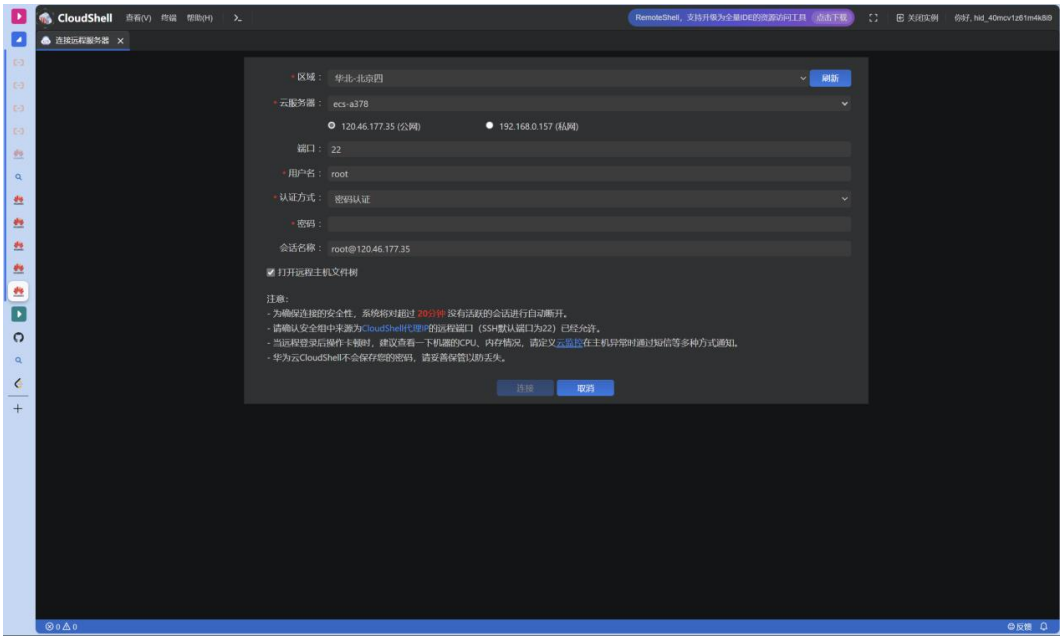
编程语言：shell, SQL

编程工具：华为 openGauss 数据库

实验结果及分析

1. 至少三个安装 DBMS 过程截图。

- 购买 ECS 服务器并登录：



- 下载、安装 openGauss 数据库：

```
[root@ecs-a378 bin]# mkdir -p /opt/software/openGauss
[root@ecs-a378 bin]# chmod 755 -R /opt/software
```

```
[root@ecs-a378 bin]# cd /opt/software/openGauss
[root@ecs-a378 openGauss]# wget https://opengauss.obs.cn-south-1.myhuaweicloud.com/5.0.1/arm/openGauss-5.0.1-openEuler-64bit-all.tar.gz
--2024-10-09 23:05:52--  https://opengauss.obs.cn-south-1.myhuaweicloud.com/5.0.1/arm/openGauss-5.0.1-openEuler-64bit-all.tar.gz
Resolving opengauss.obs.cn-south-1.myhuaweicloud.com (opengauss.obs.cn-south-1.myhuaweicloud.com)... 121.37.63.47, 122.9.127.2, 139.159.208.234, ...
Connecting to opengauss.obs.cn-south-1.myhuaweicloud.com (opengauss.obs.cn-south-1.myhuaweicloud.com)|121.37.63.47|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 128047213 (122M) [application/gzip]
Saving to: 'openGauss-5.0.1-openEuler-64bit-all.tar.gz'

openGauss-5.0.1-openEuler-64bit-all.tar.gz 100%[=========================================================================================>] 122.12M  1.35MB/s    in 62s

2024-10-09 23:06:54 (1.97 MB/s) - 'openGauss-5.0.1-openEuler-64bit-all.tar.gz' saved [128047213/128047213]

[root@ecs-a378 openGauss]#
```

```
[root@ecs-a378 openGauss]# tar -zxvf  openGauss-5.0.1-openEuler-64bit-all.tar.gz
openGauss-5.0.1-openEuler-64bit-cm.tar.gz
openGauss-5.0.1-openEuler-64bit-om.tar.gz
openGauss-5.0.1-openEuler-64bit.tar.bz2
openGauss-5.0.1-openEuler-64bit-cm.sha256
openGauss-5.0.1-openEuler-64bit-om.sha256
openGauss-5.0.1-openEuler-64bit.sha256
upgrade_sql.tar.gz
upgrade_sql.sha256
[root@ecs-a378 openGauss]# tar -zxvf  openGauss-5.0.1-openEuler-64bit-om.tar.gz
./lib/
./lib/pycparser/
./lib/pycparser/__pycache__/
./lib/pycparser/__pycache__/plyparser.cpython-37.pyc
./lib/pycparser/__pycache__/c_generator.cpython-37.pyc
./lib/pycparser/__pycache__/lextab.cpython-37.pyc
./lib/pycparser/__pycache__/ast_transforms.cpython-37.pyc
```

- 安装完毕：

```
[root@ecs-a378 openGauss]# ls
clusterconfig.xml                         openGauss-5.0.1-openEuler-64bit-cm.sha256  openGauss-5.0.1-openEuler-64bit-om.tar
.gz  script             version.cfg
lib                                       openGauss-5.0.1-openEuler-64bit-cm.tar.gz  openGauss-5.0.1-openEuler-64bit.sha256
     upgrade_sql.sha256
openGauss-5.0.1-openEuler-64bit-all.tar.gz  openGauss-5.0.1-openEuler-64bit-om.sha256  openGauss-5.0.1-openEuler-64bit.tar.bz
2    upgrade_sql.tar.gz
[root@ecs-a378 openGauss]#
```

2. 执行数据定义语句的运行截图。

- 教材例 3.1

```
maka_test=> CREATE SCHEMA "S-C-SC" AUTHORIZATION makabaka;
CREATE SCHEMA
```

```
maka_test=> SET search_path TO "S-C-SC";
SET
```

- 教材例 3.5

创建 Student 表：

```
maka_test=> CREATE TABLE Student (
maka_test(>     Sno CHAR(8) PRIMARY KEY,
maka_test(>     Sname VARCHAR(20) UNIQUE,
maka_test(>     Ssex CHAR(6),
maka_test(>     Sbirthdate DATE,
maka_test(>     Smajor VARCHAR(40)
maka_test(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "student_pkey" for table "student"
NOTICE:  CREATE TABLE / UNIQUE will create implicit index "student_sname_key" for table "student"
CREATE TABLE
```

```
maka_test=> \d Student
                Table "S-C-SC.student"
   Column   |              Type              | Modifiers
------------+--------------------------------+-----------
 sno        | character(8)                   | not null
 sname      | character varying(20)          |
 ssex       | character(6)                   |
 sbirthdate | timestamp(0) without time zone |
 smajor     | character varying(40)          |
Indexes:
    "student_pkey" PRIMARY KEY, btree (sno) TABLESPACE pg_default
    "student_sname_key" UNIQUE CONSTRAINT, btree (sname) TABLESPACE pg_default
Referenced by:
    TABLE "sc" CONSTRAINT "sc_sno_fkey" FOREIGN KEY (sno) REFERENCES student(sno)
```

- 教材例 3.6

创建 Course 表：

```
maka_test=> CREATE TABLE Course (
maka_test(> Cno CHAR(5) PRIMARY KEY,
maka_test(> Cname VARCHAR(40) NOT NULL,
maka_test(> Ccredit SMALLINT,
maka_test(> Cpno CHAR(5),
maka_test(> FOREIGN KEY (Cpno) REFERENCES Course(Cno)
maka_test(> /* 表级完整性约束，Cpno是外码，被参照表是Course，被参照列是Cno */
maka_test(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "course_pkey" for table "course"
CREATE TABLE
```

```
maka_test=> \d Course
            Table "S-C-SC.course"
 Column  |         Type          | Modifiers
---------+-----------------------+-----------
 cno     | character(5)          | not null
 cname   | character varying(40) | not null
 ccredit | smallint              |
 cpno    | character(5)          |
Indexes:
    "course_pkey" PRIMARY KEY, btree (cno) TABLESPACE pg_default
Foreign-key constraints:
    "course_cpno_fkey" FOREIGN KEY (cpno) REFERENCES course(cno)
Referenced by:
    TABLE "course" CONSTRAINT "course_cpno_fkey" FOREIGN KEY (cpno) REFERENCES course(cno)
    TABLE "sc" CONSTRAINT "sc_cno_fkey" FOREIGN KEY (cno) REFERENCES course(cno)
```

- 教材例 3.7

创建 SC 表：

```
maka_test=> CREATE TABLE SC (
maka_test(> Sno CHAR(8),
maka_test(> Cno CHAR(5),
maka_test(> Grade SMALLINT, --成绩
maka_test(> Semester CHAR(5), --开课学期
maka_test(> Teachingclass CHAR(8), --学生选修一门课所在的教学班
maka_test(> PRIMARY KEY(Sno, Cno),
maka_test(> -- 主码由两个属性组成，必须作为表级完整性进行定义
maka_test(> FOREIGN KEY(Sno) REFERENCES Student(Sno),
maka_test(> -- 表级完整性约束，Sno是外码，被参照表是Student
maka_test(> FOREIGN KEY(Cno) REFERENCES Course(Cno)
maka_test(> -- 表级完整性约束，Cno是外码，被参照表是Course
maka_test(> );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "sc_pkey" for table "sc"
CREATE TABLE
```

```
maka_test=> \d SC
            Table "S-C-SC.sc"
    Column     |     Type     | Modifiers
---------------+--------------+-----------
 sno           | character(8) | not null
 cno           | character(5) | not null
 grade         | smallint     |
 semester      | character(5) |
 teachingclass | character(8) |
Indexes:
    "sc_pkey" PRIMARY KEY, btree (sno, cno) TABLESPACE pg_default
Foreign-key constraints:
    "sc_cno_fkey" FOREIGN KEY (cno) REFERENCES course(cno)
    "sc_sno_fkey" FOREIGN KEY (sno) REFERENCES student(sno)
```

- 修改表的定义

限定学生的性别只能是男或女：

```
maka_test=> ALTER TABLE Student
maka_test-> ADD CONSTRAINT check_ssex
maka_test-> CHECK (Ssex IN ('男', '女'));
ALTER TABLE
```

尝试插入错误性别，会报错：

```
maka_test=> INSERT INTO Student (Sno, Sname, Ssex, Sbirthdate, Smajor)
maka_test-> VALUES ('10001', '张三', '它', '2004-06-09', '数学');
ERROR:  new row for relation "student" violates check constraint "check_ssex"
DETAIL:  N/A
```

限定学生的出生年月必须是 2020 年之前：

```
maka_test=> ALTER TABLE Student
maka_test-> ADD CONSTRAINT check_sbirthdate
maka_test-> CHECK (Sbirthdate < '2020-01-01');
ALTER TABLE
```

尝试插入错误出生日期，会报错：

```
maka_test=> INSERT INTO Student (Sno, Sname, Ssex, Sbirthdate, Smajor)
maka_test-> VALUES ('10001', '张三', '男', '2021-06-09', '数学');
ERROR:  new row for relation "student" violates check constraint "check_sbirthdate"
DETAIL:  N/A
maka_test=>
```

限定学生学号必须以'20'开头（：

```
maka_test=> ALTER TABLE Student
maka_test-> ADD CONSTRAINT check_sno_start
maka_test-> CHECK (Sno ~ '^20');
ALTER TABLE
```

```
maka_test=> ALTER TABLE SC
maka_test-> ADD CONSTRAINT check_sno_start
maka_test-> CHECK (Sno ~ '^20');
ALTER TABLE
```

尝试插入错误学号，会报错：

```
maka_test=> INSERT INTO Student (Sno, Sname, Ssex, Sbirthdate, Smajor)
maka_test-> VALUES ('10001', '张三', '男', '2001-06-09', '数学');
ERROR:  new row for relation "student" violates check constraint "check_sno_start"
DETAIL:  N/A
```

限定课程开课学期必须是四个数字加字母 S 或 F，如‘2023F’,‘2023S’：

```
maka_test=> ALTER TABLE SC
maka_test-> ADD CONSTRAINT check_semester
maka_test-> CHECK (Semester ~ '^\d{4}[SF]$');
ALTER TABLE
```

尝试插入错误学期，会报错：

```
maka_test=> INSERT INTO SC (Sno, Cno, Grade, Semester, Teachingclass)
maka_test-> VALUES ('20221234', 'A2011', 90, '2023J', 'c22');
ERROR:  new row for relation "sc" violates check constraint "check_semester"
DETAIL:  N/A
```

在学生表增加一列，email，并根据 email 地址格式要求增加约束条件：

```
maka_test=> ALTER TABLE Student
maka_test-> ADD COLUMN email VARCHAR(50);
ALTER TABLE
```

```
maka_test=> ALTER TABLE Student
maka_test-> ADD CONSTRAINT check_email
maka_test-> CHECK (email ~ '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]+$');
ALTER TABLE
```

尝试插入错误的邮箱地址，会报错：

```
maka_test=> INSERT INTO Student (Sno, Sname, Ssex, Sbirthdate, Smajor, email)
maka_test-> VALUES ('10001', '张三', '男', '2001-06-09', '数学', '1234+qq.com');
ERROR:  new row for relation "student" violates check constraint "check_email"
DETAIL:  N/A
```

最后，按照正确约束尝试插入信息：

```
maka_test=> INSERT INTO Student (Sno, Sname, Ssex, Sbirthdate, Smajor, email)
VALUES ('20221234', '张三', '男', '2001-06-09', '数学', '1234666@qq.com');maka_test->
INSERT 0 1
```

```
maka_test=> INSERT INTO Student (Sno, Sname, Ssex, Sbirthdate, Smajor, email)
VALUES ('20221235', '李四', '女', '2001-07-09', '物理', '1234zs@qq.we.com');maka_test->
```

插入成功：

```
   sno    | sname | ssex |     sbirthdate      | smajor |      email
----------+-------+------+---------------------+--------+------------------
 20221234 | 张三  | 男   | 2001-06-09 00:00:00 | 数学   | 1234666@qq.com
 20221235 | 李四  | 女   | 2001-07-09 00:00:00 | 物理   | 1234zs@qq.we.com
(2 rows)
```

实验总结

（简要描述，不超过 200 字）

通过本次实验，我基本具备了在自己的机器上配置 DBMS 的能力，锻炼了建表、定义数据、数据增删等简单 SQL 语句的编写和使用能力，同时也进一步熟悉了 Linux Shell 开发环境。