本次上机内容

- 安装 Docker 并拉取 openGauss 镜像
- openGauss 基本操作

提交要求

1. 使用 markdown 作答，导出为 PDF，命名为 “学号_姓名_实验11.pdf”。
2. 截止时间周日晚上 12 点，提交到软院云平台。
3. 补交邮箱：15652581355@163.com，hzy1721@qq.com。

Task 1

1. 在官网下载并安装 Docker：https://www.docker.com/products/docker-desktop
2. 拉取 enmotech/opengauss 镜像：https://hub.docker.com/r/enmotech/opengauss

```
docker pull enmotech/opengauss
```

3. 运行容器

```
docker run --name opengauss --privileged=true -d -e GS_PASSWORD=DBLab1921!
enmotech/opengauss:latest
```

4. 打开终端

```
docker exec -it opengauss /bin/bash
```

5. 使用 gsql 连接数据库 postgres

```
gsql -d postgres -U gaussdb
```

输入密码 `DBLab1921!`

（不需要提交任何内容）

Task 2

1. 按照如下模式创建基本表（都不需要定义主键，数据类型合理即可）：

STUDENT(sno, sname, ssex, sage)

COURSE(cno, cname, credit)

ELECTIVE(sno, cno, grade)

2. 编写 SQL 语句完成以下内容（语法跟 MySQL 基本一致，可以参考以往的上机内容）：

（1）查询学生编号为 10 的学生的姓名信息

（2）将 STUDENT 基本表中的学号设置为主键

（3）为 ELECTIVE 中的学生编号和课程编号创建 UNIQUE 索引

（4）创建一个视图，显示学生的姓名、课程名称以及获得的分数

3. 编写一个函数，返回某个学生的分数总和（语法参考下面的示例）
4. 为 STUDENT 表创建一个触发器，当删除学生信息时，同步删除 ELECTIVE 表中学生的选课信息

（提交使用的 SQL 语句）

openGauss 基本操作

创建一个包含仓库信息的基本表：

```
create table warehouse (
  w_id smallint,
  w_name varchar(10),
  w_street_1 varchar(20),
  w_street_2 varchar(20),
  w_city varchar(20),
  w_state char(2),
  w_zip char(9),
  w_tax dec(4, 2),
  w_ytd dec(12, 2)
);
```

查询 ID 为 1 的仓库名称：

```
select w_name from warehouse where w_id = 1;
```

在基本表 warehouse 上增加主键列：

```
alter table warehouse add primary key (w_id);
```

为 new_orders 基本表创建一个基于全部列的索引：

```
create table new_orders (
    no_o_id int not null,
    no_d_id smallint not null,
    no_w_id smallint not null
);

create unique index new_orders_index on new_orders (no_o_id, no_d_id, no_w_id);
```

创建一个与 warehouse 表相关的视图，只显示编号小于 10 的仓库的名称和地址：

```
create view warehouse_idlt10 as
    select w_name, w_street_1
    from warehouse
    where w_id < 10;
```

创建一个函数，向 new_orders 表中插入数据，并将 new_orders 中的元组数作为返回值：

```
create function new_orders_insert_func (
    in o_id int,
    in d_id int,
    in w_id int
)
returns int as $$
declare count int;
begin
  insert into new_orders values (o_id, d_id, w_id);
  select count(*) into count from new_orders;
  return count;
end;
$$ language plpgsql;
```

在 warehouse 表上创建一个完整的触发器，触发器的工作是在 wh_log 表中记录 DELETE/UPDATE/INSERT 操作的具体信息：

```
create table wh_log (
    event varchar(10),
    time_stamp timestamp,
    w_id smallint,
    w_name varchar(10)
);

create function record_warehouse_log ()
returns trigger as $warehouse_log$
begin
  if (tg_op = 'DELETE') then
    insert into wh_log select 'D', now(), old.w_id, old.w_name;
    return old;
  elseif (tg_op = 'UPDATE') then
    insert into wh_log select 'U', now(), new.w_id, new.w_name;
    return new;
  elseif (tg_op = 'INSERT') then
    insert into wh_log select 'I', now(), new.w_id, new.w_name;
    return new;
  end if;
  return null;
end;
$warehouse_log$ language plpgsql;

create trigger warehouse_log
after insert or update or delete on warehouse
for each row execute procedure record_warehouse_log();
```

退出 gsql：

```
postgres=# \q
```