openGauss 数据库开发查询实验

姓名：蒋薇　　学号：2110957

实验步骤：

- 创建和管理用户、表空间和数据库
- 创建和管理表
- 创建和管理其他数据库对象
- 学校数据模型创建及表操作

实验报告

实验步骤截图：

截图 1：指导手册第 8 页，查询表空间当前使用情况截图

步骤 1　查询表空间的当前使用情况。（截图）

```
postgres=# SELECT PG_TABLESPACE_SIZE('fastspace');
```

返回如下信息：

```
pg_tablespace_size
--------------------
                4096
(1 row)
```

其中 4096 表示表空间的大小，单位为字节。

```
postgres=# \db
             List of tablespaces
    Name    | Owner |        Location
------------+-------+------------------------
 fastpace   | omm   | tablespace/tablespace_1
 pg_default | omm   |
 pg_global  | omm   |
(3 rows)

postgres=# select pg_tablespace_size('fastpace');
 pg_tablespace_size
--------------------
               4096
(1 row)

postgres=#
```

截图 2：指导手册第 10 页，创建表截图

2.1 创建表

表是建立在数据库中的，在不同的数据库中可以存放相同的表。甚至可以通过
个数据库中创建相同名称的表。

创建表。（截图）（说明，请将红色代码一次性拷贝执行）

```
postgres=# CREATE TABLE customer_t1
(
    c_customer_sk             integer,
    c_customer_id             char(5),
    c_first_name              char(6),
    c_last_name               char(8)
);
```

当结果显示为如下信息，则表示创建成功。

```
CREATE TABLE
```

```
postgres=#  CREATE TABLE customer_t1
postgres-# (
postgres(#     c_customer_sk             integer,
postgres(#     c_customer_id             char(5),
postgres(#     c_first_name              char(6),
postgres(#     c_last_name               char(8)
postgres(# );
CREATE TABLE
```

截图 3：指导手册第 16 页，向分区表中插入数据后查看分区表中所有数据并截图（该命令需自行撰写）

插入数据返回如下。（查看表中数据并截图，命令未提供，请自行完成）

```
INSERT 0 4
```

```
postgres=# select * from tpcds.web_returns_p2 order by ca_addr
ess_sk;
 ca_address_sk |  ca_address_id   | ca_street_number | ca_stre
et_name | ca_street_type  | ca_suite_number | ca_city | ca_cou
nty | ca_state |   ca_zip   | ca_country | ca_gmt_offset |   c
a_location_type
---------------+------------------+------------------+--------
--------+-----------------+-----------------+---------+-------
----+----------+------------+------------+---------------+----
------------------
             1 | a                | 1                |  a
       | a               | a               | a       |  a
    | a         | a          | a          |          1.00 | a

             2 | b                | 2                |  b
       | b               | b               | b       |  b
    | b         | b          | b          |          1.10 | b

          5050 | c                | 300              |  c
       | c               | c               | c       |  c
    | c         | c          | c          |          1.20 | c

         14888 | d                | 400              |  d
       | d               | d               | d       |  d
    | d         | d          | d          |          1.50 | d

(4 rows)

postgres=#
```

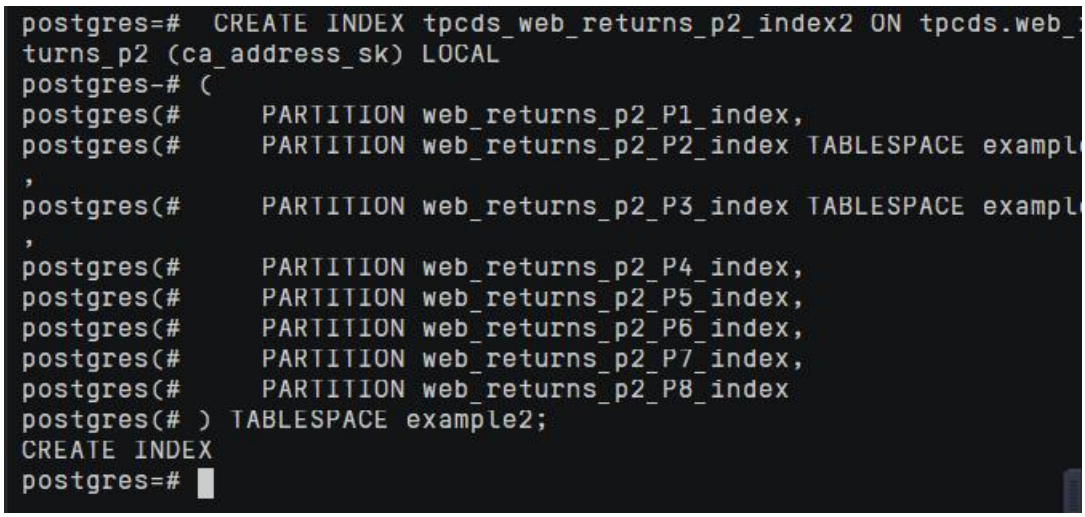
截图 4：指导手册第 19 页，创建分区索引截图。

创建分区索引 tpcds_web_returns_p2_index2，并指定索引分区的名称。（截图）

```
postgres=# CREATE INDEX tpcds_web_returns_p2_index2 ON tpcds.web_returns_p2 (ca_address_sk) LOCAL
(
    PARTITION web_returns_p2_P1_index,
    PARTITION web_returns_p2_P2_index TABLESPACE example3,
    PARTITION web_returns_p2_P3_index TABLESPACE example4,
    PARTITION web_returns_p2_P4_index,
    PARTITION web_returns_p2_P5_index,
    PARTITION web_returns_p2_P6_index,
    PARTITION web_returns_p2_P7_index,
    PARTITION web_returns_p2_P8_index
) TABLESPACE example2;
```

当结果显示为如下信息，则表示创建成功。

```
CREATE INDEX
```



截图 5：指导手册第 23 页，更新物化视图。

再查看物化视图 MV_MyView，发现多了两条记录。 （截图）

```
postgres=# SELECT * FROM MV_MyView;
 ca_address_sk |  ca_address_id   | ca_street_number | ca_street_name | ca_street_type | ca_suite_number |
ca_city | ca_county | ca_state |  ca_zip  | ca_country | ca_gm
t_offset |  ca_location_type
---------------+------------------+------------------+----------------+----------------+-----------------+---------
--+-----------+------
---------+---------------------
          5050 | c                | 300              | c              | c              | c
| c       | c         | c       | c         | c          |
    1.20 | c
          7050 | c                | 300              | c              | c              | c
| c       | c         | c       | c         | c          |
    1.20 | c
          8888 | d                | 400              | d              | d              | d
| d       | d         | d       | d         | d          |
```

```
postgres=# select * from mv_myview;
 ca_address_sk |  ca_address_id   | ca_street_number | ca_street_nam
e | ca_street_type  | ca_suite_number | ca_city | ca_county | ca_sta
te |   ca_zip    | ca_country | ca_gmt_offset |   ca_location_type
---------------+------------------+------------------+--------------
--+-----------------+-----------------+---------+-----------+-------
---+------------+------------+---------------+----------------------
          5050 | c                | 300              | c
  | c               | c               | c       | c         | c
   | c          | c          |          1.20 | c
          7050 | c                | 300              | c
  | c               | c               | c       | c         | c
   | c          | c          |          1.20 | c
          8888 | d                | 400              | d
  | d               | d               | d       | d         | d
   | d          | d          |          1.50 | d
         14888 | d                | 400              | d
  | d               | d               | d       | d         | d
   | d          | d          |          1.50 | d
(4 rows)

postgres=#
```

截图 6：指导手册第 26 页，管理存储过程

```
(2 rows)
```

2 管理存储过程

管理存储过程，命令如下：（截图）

```
postgres=# \sf insert_data
```

结果如下：

```
CREATE OR REPLACE FUNCTION public.insert_data()
 RETURNS void
 LANGUAGE plpgsql
 NOT FENCED NOT SHIPPABLE
AS $function$ DECLARE
a int;
b int;
begin
```

```
postgres=# \sf insert_data
CREATE OR REPLACE PROCEDURE public.insert_data()
AS  DECLARE
a int;
b int;
begin
a = 1;
b = 2;
insert into t_test values(a,b);
insert into t_test values(b,a);
end;
/
postgres=#
```

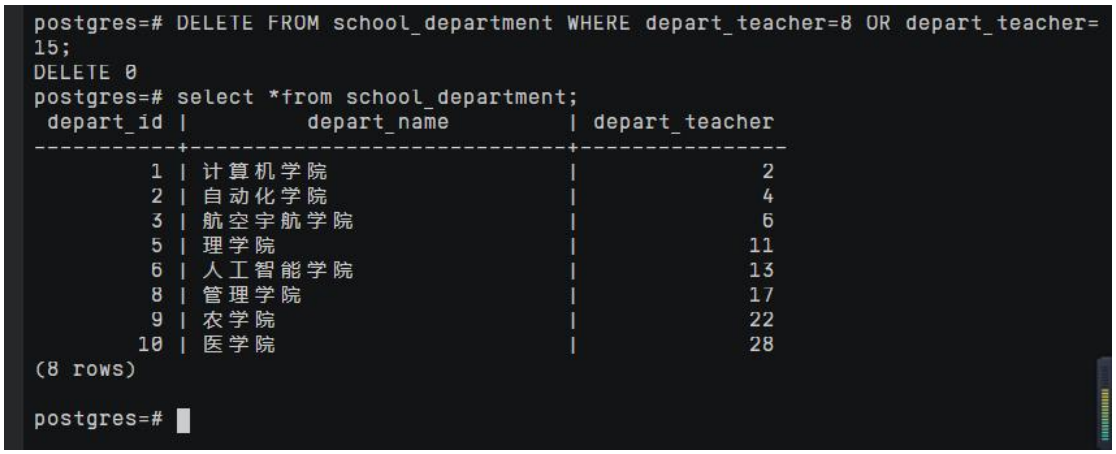
截图 7：指导手册第 39 页，删除数据后表中内容截图

.4.2 删除指定数据

在 B 校中删除教师编号 8 和 15 所管理的院系。

```
postgres=# DELETE FROM school_department WHERE depart_teacher=8 OR depart_teacher=15;
DELETE 0
postgres=# SELECT * FROM school_department;    (截图)
 depart_id |        depart_name        | depart_teacher
-----------+---------------------------+----------------
         1 | 计算机学院                |              2
         2 | 自动化学院                |              4
         3 | 航空宇航学院              |              6
         5 | 理学院                    |             11
         6 | 人工智能学院              |             13
         8 | 管理学院                  |             17
         9 | 农学院                    |             22
        10 | 医学院                    |             28
(8 rows)
```

本实验结束。



实验思考题：

1. 在 openGauss 中，创建具有“创建数据库”权限的用户 Alice，并设置其初始密码为”openGauss@0331”,应使用的语句是：

create user Alice createdb password ‘openGauss@0331’;

2. 命令 “DROP USER kim CASCADE”的效果是？（可以预习参考第八周主讲课内容，权限和授权）

删除名为 “kim” 的用户，并删除其拥有的所有对象（如表、视图等）。这个命令中的 “CASCADE” 参数指示系统删除与用户相关的所有对象。

ALTER TABLESPACE

.2.4 删除表空间

删除用户 jack。

```
postgres=# DROP USER jack CASCADE;
DROP ROLE
```

3. 向表中插入数据时，是否允许只对部分属性插入数值？在何种情况下允许，应如何书写语句？何种情况下不允许？

是

1.2.2.1 向表 customer_t1 中插入一行数据

数据值是按照这些字段在表中出现的顺序列出的，并且用逗号分隔。通常数据值是文本（常量），但也允许使用标量表达式。

```
postgres=# INSERT INTO customer_t1(c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', 'Grace');
```

如果用户已经知道表中字段的顺序，也可无需列出表中的字段。例如以下命令与上面的命令效果相同。

```
postgres=# INSERT INTO customer_t1 VALUES (3769, 'hello', 'Grace');
```

如果用户不知道所有字段的数值，可以忽略其中的一些。没有数值的字段将被填充为字段的缺省值。例如：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk, c_first_name) VALUES (3769, 'Grace');
```

或

```
postgres=# INSERT INTO customer_t1 VALUES (3769, 'hello');
```

用户也可以对独立的字段或者整个行明确缺省值：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk, c_customer_id, c_first_name) VALUES (3769, 'hello', DEFAULT);
```

或

```
postgres=# INSERT INTO customer_t1 DEFAULT VALUES;
```

不允许：在一些情况下，可能不允许只对部分属性插入数据。例如，如果某个属性被定义为表格的主键或非空约束，那么在插入新记录时必须为该属性指定一个值。如果 INSERT 语句中省略了该属性的值，数据库会报错并拒绝插入数据。

4. 是否可以向表中一次性插入多条数据？何种插入效率较高？

```
postgres=# INSERT INTO customer_t1 DEFAULT VALUES;
```

1.2.2.2 向表中插入多行数据

命令如下：

```
postgres=# INSERT INTO customer_t1 (c_customer_sk, c_customer_id, c_first_name) VALUES
    (6885, 'maps', 'Joes'),
    (4321, 'tpcds', 'Lily'),
(9527, 'world', 'James');
```

如果需要向表中插入多条数据，除此命令外，也可以多次执行插入一行数据命令实现。但是建议使用此命令可以提升效率。

5. openGauss 中将表中所有元组删除的两种命令是?

如果执行如下命令之一，会删除表中所有的行。

```
postgres=# DELETE FROM customer_t1;
```

或：

```
postgres=# TRUNCATE TABLE customer_t1;
```

全表删除的场景下，建议使用 truncate，不建议使用 delete。

6. 如果经常需要查询某字段值小于某一指定值的信息，可以如何操作？（提示，从索引角度思考）

步骤 5 创建表达式索引

假如经常需要查询 ca_street_number 小于 1000 的信息，执行如下命令进行查询。

```
postgres=# SELECT * FROM tpcds.customer_address_bak WHERE trunc(ca_street_number) < 1000;
```

可以为上面的查询创建表达式索引：

```
postgres=# CREATE INDEX para_index ON tpcds.customer_address_bak (trunc(ca_street_number));
CREATE INDEX
```

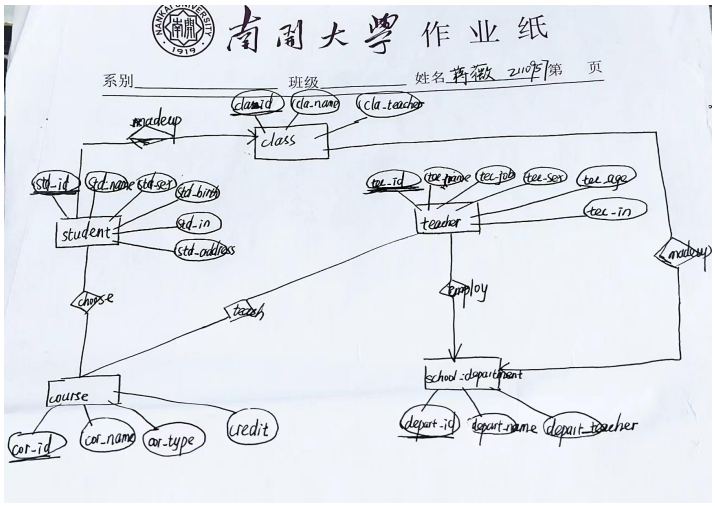
7. 在什么场景下可以使用物化视图？物化视图和普通视图的区别是？

①大型数据集的查询：当需要查询大量数据时，物化视图可以提高查询效率和性能，减少查询时间。②复杂查询的优化：当需要进行多个表的联合查询或聚合计算时，物化视图可以将复杂查询转化为一个简单的查询，提高查询效率。数据报表的生成：③当需要生成数据报表时，物化视图可以提供预计算的数据，减少报表生成时间。

物化视图是一种可缓存的视图，它将查询结果保存在磁盘上，以便在下次查询时可以直接使用缓存，而无需再次计算。物化视图通常用于需要频繁查询的大型数据集，可以提高查询效率和性能。

物化视图和普通视图的主要区别在于，普通视图只是一个虚拟表，它并不存储数据，每次查询都需要重新计算；而物化视图将查询结果保存在磁盘上，可以直接使用缓存，无需重新计算。因此，物化视图的查询效率和性能通常比普通视图更高，但也需要更多的存储空间和更新管理的开销。

8. 学校模型 ER 图绘制



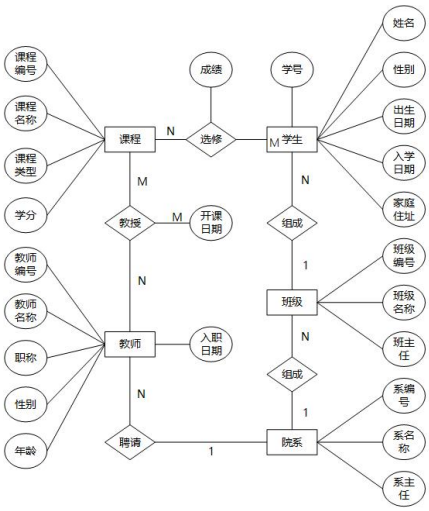


图 1-1 E-R 图