openGauss 数据库开发查询实验

姓名：张洋 学号：2111460

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

```
postgres=# SELECT PG_TABLESPACE_SIZE('fastspace');
 pg_tablespace_size
--------------------
               4096
(1 row)
```

截图 2：指导手册第 10 页，创建表截图

```
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）

命令：`postgres=# select*from tpcds.web_returns_p2;`

```
postgres=# select*from tpcds.web_returns_p2;
 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_gmt_offset |   ca_lo
cation_type
---------------+------------------+------------------+----------------+-----------------+-----------------+---------+-----------+----------+------------+------------+---------------+--------
-------------
             1 | a                | 1                | a              | a               | a               | a       | a         | a        | a          |            |          1.00 | a
             2 | b                | 2                | b              | b               | b               | b       | b         | b        | b          |            |          1.10 | b
          5050 | c                | 300              | c              | c               | c               | c       | c         | c        | c          |            |          1.20 | c
         14888 | d                | 400              | d              | d               | d               | d       | d         | d        | d          |            |          1.50 | d
(4 rows)
```

截图 4：指导手册第 19 页，创建分区索引截图。

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

截图 5：指导手册第 23 页，更新物化视图。

```
postgres=# SELECT * FROM MV_MyView;
 ca_address_sk |  ca_address_id   | ca_street_number | ca_street_name | ca_street_type  | c
a_suite_number | ca_city | ca_county | ca_state |   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)
```

截图 6：指导手册第 26 页，管理存储过程

```
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;
/
```

截图 7：指导手册第 39 页，删除数据后表中内容截图

```
postgres=#  SELECT * FROM school_department;
 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)
```

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）

命令 "DROP USER kim CASCADE" 的效果是删除数据库中的用户 "kim"，并且级联删除该用户所拥有的所有对象和权限。

具体来说，当使用 CASCADE 关键字时，该命令将删除用户 "kim" 拥有的所有对象，例如表、视图、函数等，并且也会撤销授予该用户的所有权限，包括对对象的 SELECT、INSERT、UPDATE、DELETE 等操作权限，以及对数据库的创建、删除、更改等操作权限。

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？

OpenGauss 数据库向表中插入数据时允许只对部分属性插入数值。这些属性称为可选属性。在某些情况下，只有一些属性需要被指定数值，其他属性可以为空或使用默认值。这可以通过省略 INSERT 语句中某些属性的值来实现。具体来说，INSERT 语句中可以指定要插入的属性列表和对应的值列表。如果省略了某些属性，则数据库将使用该属性的默认值或 NULL 值。

具体操作指令如下：

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;
```

在以下情况下，不允许省略某些属性的值：

1. 如果该属性在表中被定义为 NOT NULL，则必须为该属性指定一个非空值。

2. 如果该属性在表中没有默认值，则必须为该属性指定一个值。

在这些情况下，省略某个属性的值将导致插入失败并引发错误。

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？

OpenGauss 数据库可以向表中一次性插入多条数据。在 OpenGauss 中，可以使用 INSERT INTO 语句的多个值列表来一次性插入多条数据。具体来说，可以将多个值列表用逗号分隔，并将它们包含在一个 INSERT INTO 语句中，如下所示：

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

在这个语句中，需要指定要插入的表名和要插入的列名列表。然后，可以将多个值列表放在一个大括号内，并用逗号分隔它们。每个值列表必须与

列名列表中的列一一对应，并按照相同的顺序列出。

批量插入的效率通常比单行插入要高，因为批量插入可以减少通信开销和锁竞争。另外，可以通过使用 INSERT INTO ... SELECT 语句来进一步优化批量插入，它允许从另一个表或子查询中选择数据来插入。这种方法可以避免复制和转换大量数据，从而提高性能。

5. openGauss 中将表中所有元组删除的两种命令是?

可以使用 DELETE 语句和 TRUNCATE 语句来将表中所有元组删除。这两种命令的使用方式和效果有所不同，具体如下：

DELETE 语句：该语句用于从表中删除行。使用 DELETE 语句时，可以指定一个 WHERE 子句来限定要删除的行。如果没有 WHERE 子句，则将删除表中所有行。

TRUNCATE 语句：该语句用于截断表。使用 TRUNCATE 语句时，不需要指定 WHERE 子句，它会删除表中所有的行，并将表的计数器重置为零。

TRUNCATE 语句比 DELETE 语句更快，因为它不需要扫描表中的每一行，而是通过重置表的计数器来快速删除所有行。另外，TRUNCATE 语句还可以释放表所使用的空间，从而减少磁盘使用量。全表删除的场景下，建议使用 truncate，不建议使用 delete。但是，TRUNCATE 语句不支持 WHERE 子句和其他限制条件，因此必须小心使用。如果需要删除表中特定的行，应该使用 DELETE 语句。

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）

如果经常需要查询某字段值小于某一指定值的信息，可以考虑在该字段上创建索引。这样可以大大提高查询效率，并减少数据库扫描的开销。

在 OpenGauss 中，可以使用 CREATE INDEX 语句来创建索引。例如，假设需要在名为 table_name 的表中查询某一字段 field_name 的值小于指定值 value 的记录，则可以执行以下操作：

CREATE INDEX idx_name ON table_name (field_name);

这将在 field_name 字段上创建一个名为 idx_name 的 B 树索引。一旦创建了该索引，就可以使用以下查询语句来查找该字段值小于指定值 value 的记录：

SELECT * FROM table_name WHERE field_name < value;

这个查询语句将利用 B 树索引来查找所有满足条件的记录，从而提高查询效率。

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？

OpenGauss 数据库可以在以下场景下使用物化视图：

1. 数据仓库应用：在数据仓库应用中，经常需要进行复杂的查询和数据聚合操作，这些操作通常需要消耗大量的计算资源和时间。通过创建物化视图来缓存查询结果，可以大大提高查询性能和效率。
2. 数据报表：在数据报表应用中，经常需要对大量的数据进行统计和分析，

这些操作通常需要进行多次计算。通过创建物化视图来缓存计算结果，可以减少计算时间和计算资源的消耗。

物化视图和普通视图的区别主要在以下几个方面：

1. 存储方式：普通视图是虚拟的，不存储实际的数据，每次查询时都需要重新计算；而物化视图是实际存储了计算结果的表，可以提高查询效率。
2. 数据实时性：普通视图的数据是实时计算得到的，查询结果是动态的；而物化视图的数据是预先计算并存储的，查询结果是静态的。
3. 查询性能：由于物化视图存储了实际的数据，因此在查询时可以直接从物化视图中读取数据，而不需要每次都重新计算。这样可以大大提高查询效率。
4. 存储空间：由于物化视图存储了实际的数据，因此占用的存储空间相比普通视图会更多。

8. 学校模型 ER 图绘制