**《数据库系统原理》**

**课程实验报告**



**学院：计算机（国家示范性软件）学院**

**班级：** 2021211304

**姓名：** **张梓良 杨晨 苗雨**

**学号：**2021212484 2021212171 2021212492

目录

[1实验目的 2](#_Toc154746504)

[2实验环境 2](#_Toc154746505)

[3实验内容 2](#_Toc154746506)

[4实验步骤及结果 2](#_Toc154746507)

[4.1表空间 2](#_Toc154746508)

[4.2分区表 6](#_Toc154746509)

[4.3索引 15](#_Toc154746510)

[5 实验总结 21](#_Toc154746511)

# 实验目的

1.1学会创建多个表空间，并在不同的表空间上创建对象；对表空间进行查询，删除等管理操作

1.2学会创建分区表，并向表中插入数据，进行观察；对分区表进行重命名，删除等管理操作

1.3学会在普通表上创建管理索引，学会在分区表上创建管理索引

# 2实验环境

VMware虚拟机openEuler20.03 系统上的openGauss数据库

# 3实验内容

3.1在openGauss数据库中创建和管理表空间

3.2在openGauss数据库中创建分区表，插入数据以及对分区表进行管理

3.3在openGauss数据库中创建和管理索引

# 4实验步骤及结果

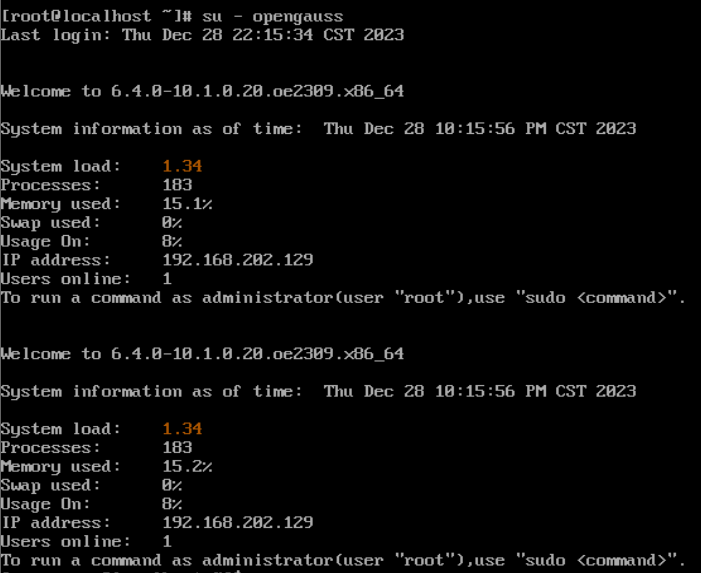
## 4.1表空间

**4.1.1创建表空间**

* 创建用户

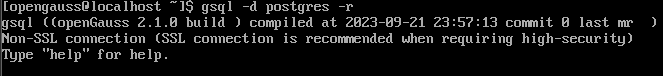
以操作系统用户opengauss登录数据库主节点，启动openGauss数据库服务：

su - opengauss



连接数据库：

gsql -d postgres -r



创建用户zzl

create user jack identified by ‘openeuler\_zzl123!;

用户创建成功截图:



* 创建表空间

create tablespace fastspace relative location ‘‘tablespace/tablespace\_1’’

创建成功截图：



* 访问权限授予

omm用户将“fastspace”表空间的访问权限赋予数据用户 jack，命令：

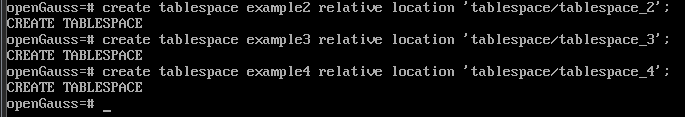
grant create on tablespace fastspace to jack；

赋予成功截图：



* 创建多个表空间

运行截图：



**4.1.2在表空间上创建对象**

omm用户具有以上表空间的CREATE权限，jack用户拥有fastspace表空间的CREATE权限。

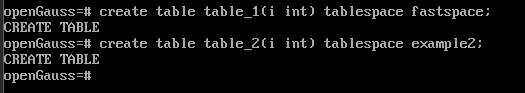
1. 在指定的表空间上创建表

命令：

CREATE table table\_1(i int) TABLESPACE fastspace；

CREATE TABLE table\_2(i int) TABLESPACE example2；

创建成功截图：



1. 在默认表空间上创建表

设置默认表空间

set default\_tablespace = ‘example3’；



再创建表,表创建在默认表空间

CREATE TABLE table\_3(i int)；

创建成功截图：



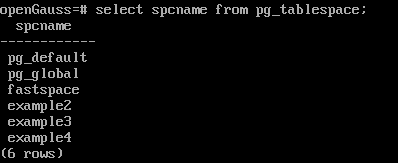
**4.1.3管理表空间**

（1）查询表空间

方法1，检查pg\_tablespace系统表，命令:

tpch=# SELECT spcname FROM pg\_tablespace;

查询结果截图：



方法2，使用gsql程序的元命令查询表空间，命令:

tpch=# \db

查询结果截图：

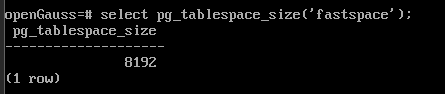


（2）查询表空间当前使用情况

使用PG\_TABLESPACE\_SIZE(‘表空间名字’)来查询，命令：

tpch=# SELECT PG\_TABLESPACE\_SIZE(‘fastspace’)；

查询结果截图：

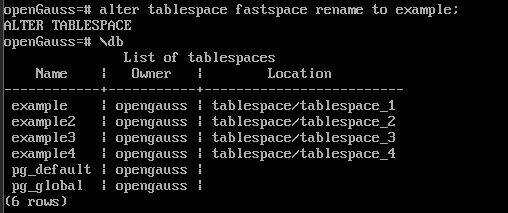


1. 重命名表空间

对表空间fastspace重命名为example：

tpch=# ALTER TABLESPACE fastspace RENAME TO example;

重命名成功截图：



1. 删除表空间

命令：

tpch=# DROP TABLESPACE example;

删除失败截图：



失败原因是表空间不为空的情况下无法删除表空间，避免误删里面的重要数据。

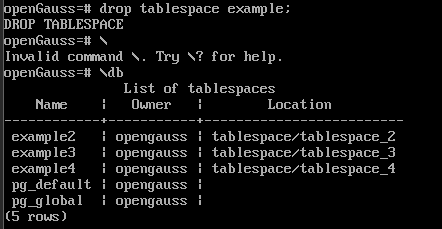
先清空表空间

tpch=# DROP TABLE table\_1;

清除成功



然后在删除表空间



删除成功

## 4.2分区表

**4.2.1 创建分区表**

（1）方法一：VALUES LESS THAN

create table partition\_orders\_1(

o\_orderkey integer,

o\_custkey integer,

o\_orderstatus char(1),

o\_totalprice decimal(15,2),

o\_orderdate date,

o\_orderpriority char(15),

o\_clerk char(15),

o\_shippriority integer,

o\_comment varchar(79),

PRIMARY KEY (o\_orderkey)

)

partition by range(o\_orderkey)

(

partition p1 values less than(100),

partition p2 values less than(200),

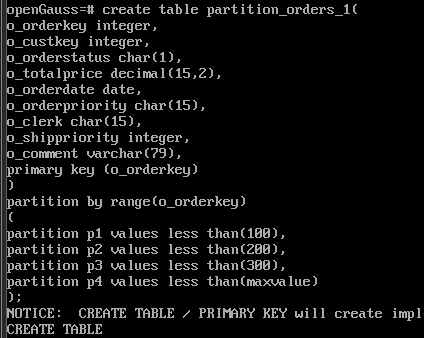
partition p3 values less than(300),

partition p4 values less than(maxvalue)

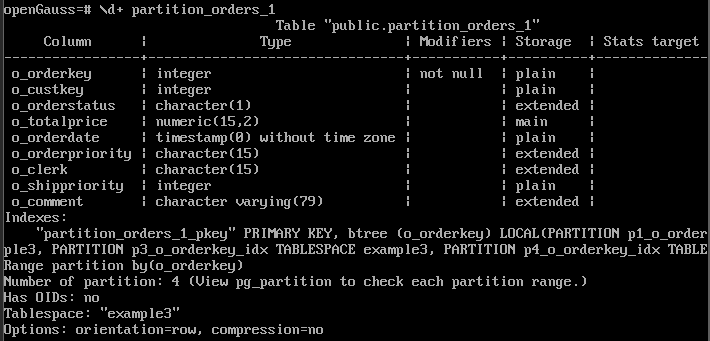
);

（分区键为id，分为4个区：p1<100,100<=p2<200,200<=p3<300,300<=p4）

截图：



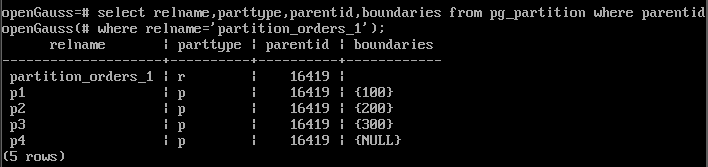
执行\d+ 命令显示该分区表的详细信息：



执行select命令列出各分区情况：

tpch=# select relname,parttype,parentid,boundaries from pg\_partition where parentid in(select oid from pg\_class where relname=’partition\_orders\_1’);

运行截图:



（2）方法二：START END

create table partition\_orders\_2(

o\_orderkey integer,

o\_custkey integer,

o\_orderstatus char(1),

o\_totalprice decimal(15,2),

o\_orderdate date,

o\_orderpriority char(15),

o\_clerk char(15),

o\_shippriority integer,

o\_comment varchar(79),

PRIMARY KEY (o\_orderkey)

)

partition by range(o\_orderkey)

(

partition p1 start(2) end(100) every(10),

partition p2 end(200),

partition p3 end(300),

partition p4 start(300),

partition p5 start(400),

partition p6 start(500) end(600)

);

第一个分区定义含start，范围（minvalue，2）自动作为第一个分区p1\_0,p1\_0<2

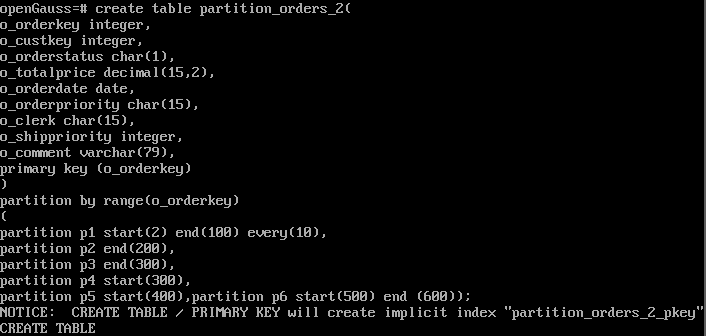
由于every（10），则p1分区进行间隔分区，间隔为10 :

p1\_1 [2,12),p1\_2 [12,22),p1\_3 [22,32),p1\_4 [32,42),p1\_5 [42,52),p1\_6 [52,62),p1\_7 [62,72) p1\_8 [72,82),p1\_9 [82,92),p1\_10 [92,100)

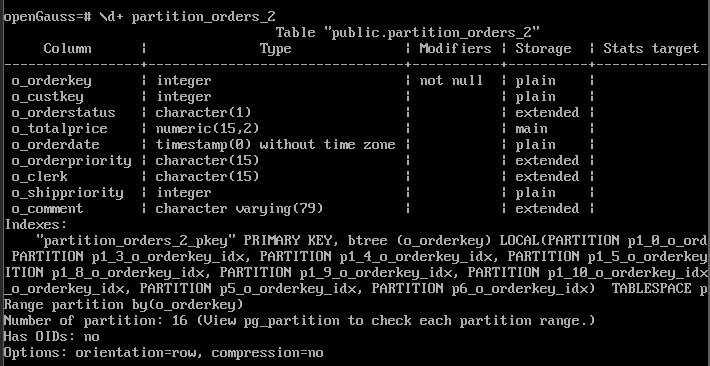
之后5个分区：

p2 [100,200),p3 [200,300),p4 [300,400),p5 [400,500),p6 [500,600)

截图：



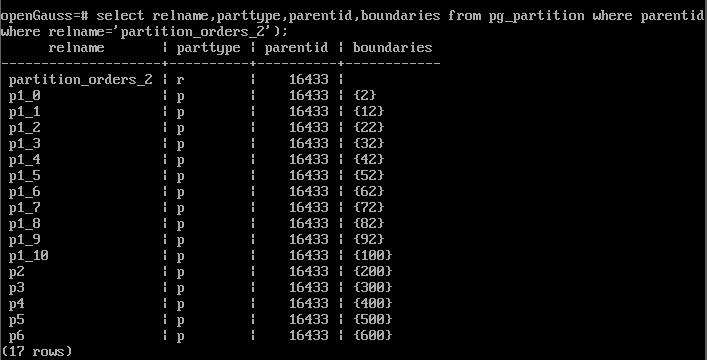
执行\d+ 命令显示该分区表的详细信息：



执行select命令列出各分区情况：

tpch=# select relname,parttype,parentid,boundaries from pg\_partition where parentid in(select oid from pg\_class where relname=’partition\_orders\_2’);

运行截图:



（3）方法三：INTERVAL

create table partition\_orders\_3(

o\_orderkey integer,

o\_custkey integer,

o\_orderstatus char(1),

o\_totalprice decimal(15,2),

o\_orderdate date,

o\_orderpriority char(15),

o\_clerk char(15),

o\_shippriority integer,

o\_comment varchar(79),

PRIMARY KEY (o\_orderkey)

)

partition by range(o\_orderdate)

interval('1 day')

(

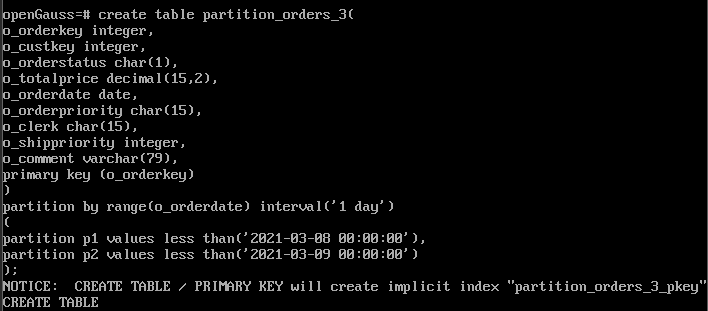
partition p1 values less than('2021-03-08 00:00:00'),

partition p2 values less than('2021-03-09 00:00:00')

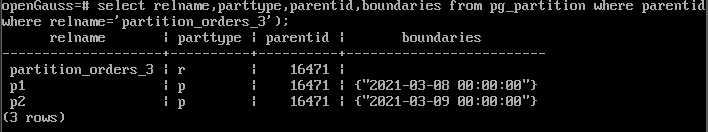
);

一开始创建的分区只有p1,p2。但如果向表中插入键值不在已有分区范围内的元组，比如：（1， '2021-03-11 00:00:00'），则会自动创建一个分区，其上界与插入的元组间隔1 day，该元组存入该分区。

截图：



执行select命令列出各分区情况，截图：



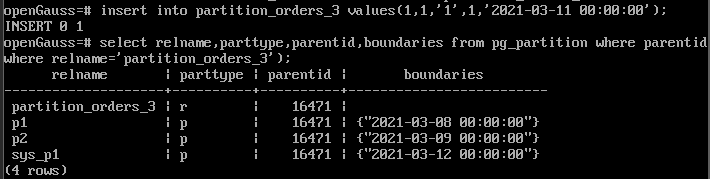
插入元组（1，'2021-03-11 00:00:00'），命令：

tpch=# insert into partition\_orders\_3 values（1，'2021-03-11 00:00:00'）;

再次通过select命令列出各分区情况:

tpch=# select relname,parttype,parentid,boundaries from pg\_partition where parentid in(select oid from pg\_class where relname=’partition\_orders\_3’);

截图：



自动生成了1个新的分区，上界为‘2021-03-12 00:00:00’，与插入元组间隔为1 day。

（4）设置分区所在的表空间

在创建分区表时，可以把分区表的不同分区设置在不同的表空间，从而提升整个系统的性能 通过在partition语句后面加上tablespace来指定创建的分区所在的表空间:

create table partition\_orders\_4(

o\_orderkey integer,

o\_custkey integer,

o\_orderstatus char(1),

o\_totalprice decimal(15,2),

o\_orderdate date,

o\_orderpriority char(15),

o\_clerk char(15),

o\_shippriority integer,

o\_comment varchar(79),

PRIMARY KEY (o\_orderkey)

)

tablespace example2

partition by range(o\_orderkey)

(

partition p1 values less than(100),

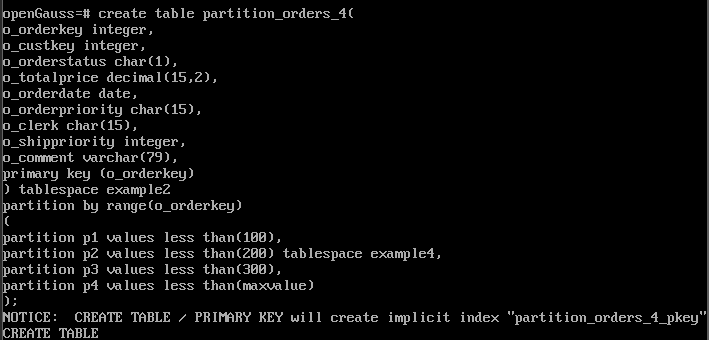
partition p2 values less than(200) tablespace example4,

partition p3 values less than(300),

partition p4 values less than(maxvalue)

);

创建成功截图：



分区p1，p3，p4都在表空间example2中，而分区p2在表空间example4中。

**4.2.2 管理分区表**

（1）删除，添加，重命名分区

删除分区p4

tpch=# ALTER TABLE partition\_orders\_4 DROP PARTITION p4;

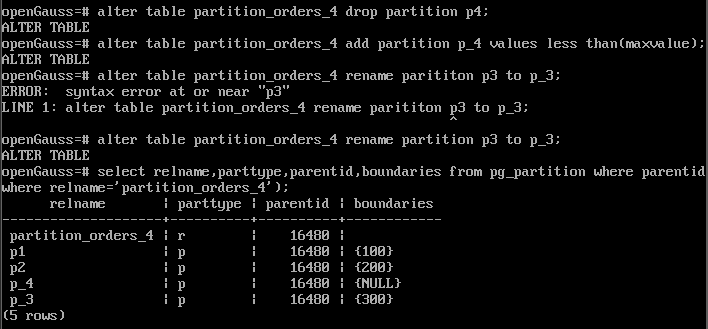
添加分区p\_4

tpch=# ALTER TABLE partition\_orders\_4 ADD PARTITION p\_4 VALUES LESS THAN(MAXVALUE);

重命名分区p3

tpch=# ALTER TABLE partition\_orders\_4 RENAME PARTITION p3 TO p\_3;

修改后结果：

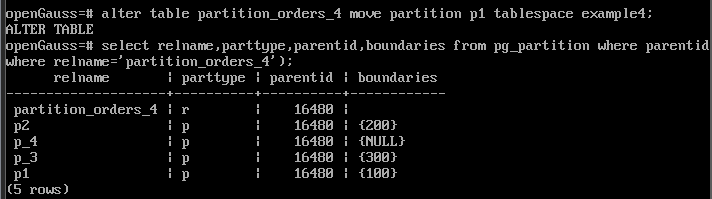


（2）修改分区的表空间

将分区p1由原来所在的表空间example2移动到example4

tpch=# ALTER TABLE partition\_orders\_4 MOVE PARTITION p1 TABLESPACE example4;

修改后结果：



（3）查询分区

查询单独分区内的数据：

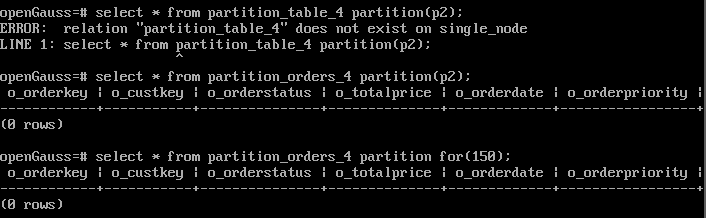
分区p2，直接使用partition(分区名称)：

tpch=# SELECT \* FROM partition\_orders\_4 PARTITION(p2);

或partition for（数值）：

tpch=# SELECT \* FROM partition\_orders\_4 PARTITION FOR(150);

查询结果：

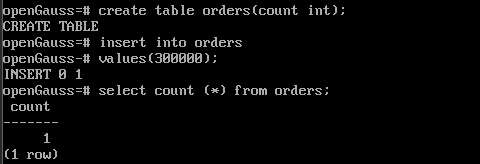


（4）数据转移

查看orders表中的数据量

select count（\*） from orders;

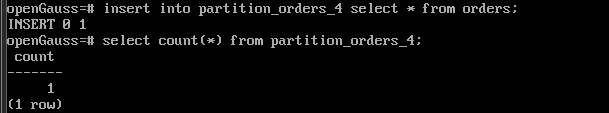
截图：



将orders表中的数据转移到分区表partition\_orders\_4中

insert into partition\_orders\_4 select \* from orders;

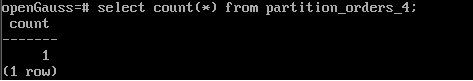
截图：



查看整个分区表情况，以及各分区情况：

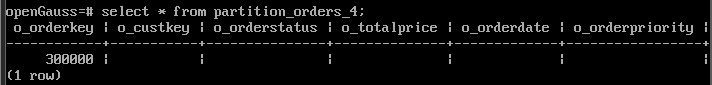
select count（\*） from partition\_orders\_4;

截图：



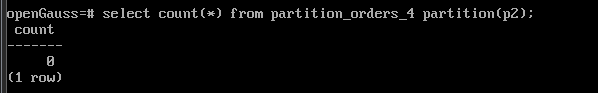
select \* from partition\_orders\_4;

截图：



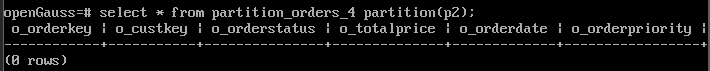
select count（\*） from partition\_orders\_4 partition(p2);

截图：



select \* from partition\_orders\_4 partition(p2);

截图：



## 4.3索引

**4.3.1 普通表上创建管理索引**

（1）创建索引

创建orders表的备份orderscopy2，并将数据导入：

create table orderscopy2(

o\_orderkey integer,

o\_custkey integer,

o\_orderstatus char(1),

o\_totalprice decimal(15,2),

o\_orderdate date,

o\_orderpriority char(15),

o\_clerk char(15),

o\_shippriority integer,

o\_comment varchar(79),

PRIMARY KEY (o\_orderkey)

)

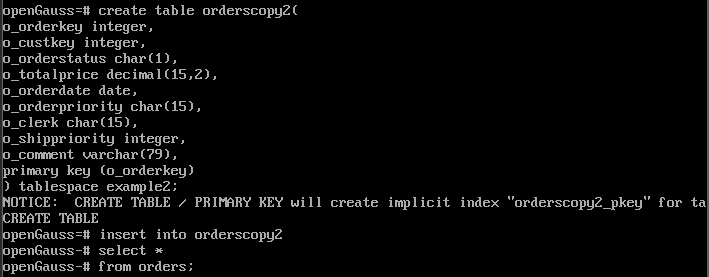
tablespace example2;

INSERT INTO orderscopy2

SELECT \*

FROM orders;

截图：



创建普通索引：

如果对于orderscopy2表，需要经常进行以下查询。:

tpch=# SELECT \* FROM orderscopy2 WHERE o\_totalprice>10000;

对于orderscopy2表，使用以下命令创建索引:

tpch=# CREATE INDEX orderscopy2\_index\_totalprice ON orderscopy2(o\_totalprice);

截图：



创建多字段索引:

假如用户需要经常查询表orderscopy2中o\_orderstatus是’0’，且o\_totalprice小于10000的记录，使用以下命令进行查询。

tpch=# SELECT \* FROM orderscopy2 WHERE o\_orderstatus=’O’ AND o\_totalprice<10000;

使用以下命令在字段o\_orderstatus和o\_totalprice上定义一个多字段索引。

tpch=# CREATE INDEX orderscopy2\_index\_more\_column ON orderscopy2(o\_orderstatus, o\_totalprice);

截图：



创建部分索引：

如果只需要查询o\_orderstatus为’O’的记录

tpch=# SELECT \* FROM orderscopy2 WHERE o\_orderstatus=’O’;

创建部分索引来提升查询效率:

tpch=# CREATE INDEX orderscopy2\_part\_index ON orderscopy2(o\_orderstatus) WHERE o\_orderstatus=’O’;

截图：



创建表达式索引：

假如经常需要查询代号以8结尾的收银员的信息，执行如下命令进行查询。

tpch=# SELECT \* FROM orderscopy2 WHERE o\_clerk like ’%8’;

为上面的查询创建表达式索引：

tpch=# CREATE INDEX orderscopy2\_para\_index ON orderscopy2(reverse(o\_clerk) varchar\_pattern\_ops);

截图：



（2）管理索引

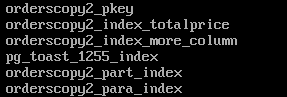
查询索引：

执行如下命令查询系统和用户定义的所有索引。

tpch=# SELECT RELNAME FROM PG\_CLASS WHERE RELKIND='i';

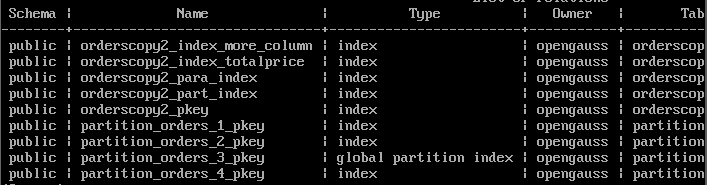
可以看到上面创建的索引：

截图：



\di+可以查询用户定义的所有索引：

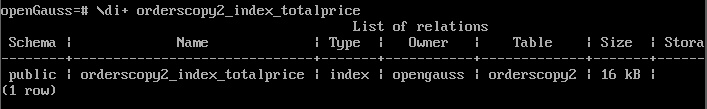
截图：



执行如下命令查询指定索引的信息：

tpch=# \di+ orderscopy2\_index\_totalprice

截图：



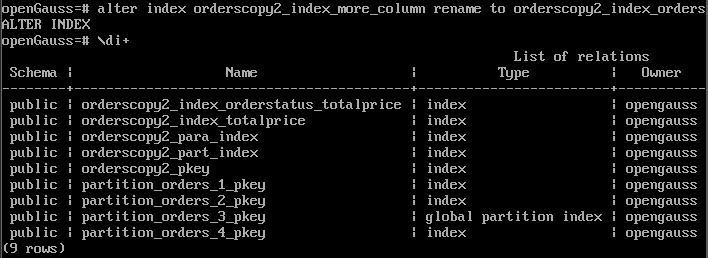
重命名索引：

执行如下命令对索引orderscopy2\_index\_more\_column重命名为orderscopy2\_index\_orderstatus\_totalprice。

tpch=# ALTER INDEX orderscopy2\_index\_more\_column RENAME TO orderscopy2\_index\_orderstatus\_totalprice;

用\di+查看，索引重命名成功

截图：



删除索引：

删除orderscopy2\_para\_index 索引

tpch=# DROP INDEX orderscopy2\_para\_index;

删除成功

截图：



**4.3.2 分区表上创建管理索引**

（1）创建索引

以orders表为例创建一个分区表：

create table partition\_orders\_0(

o\_orderkey integer,

o\_custkey integer,

o\_orderstatus char(1),

o\_totalprice decimal(15,2),

o\_orderdate date,

o\_orderpriority char(15),

o\_clerk char(15),

o\_shippriority integer,

o\_comment varchar(79),

PRIMARY KEY (o\_orderkey)

)

tablespace example2

partition by range(o\_orderkey)

(

partition p1 values less than(100),

partition p2 values less than(200),

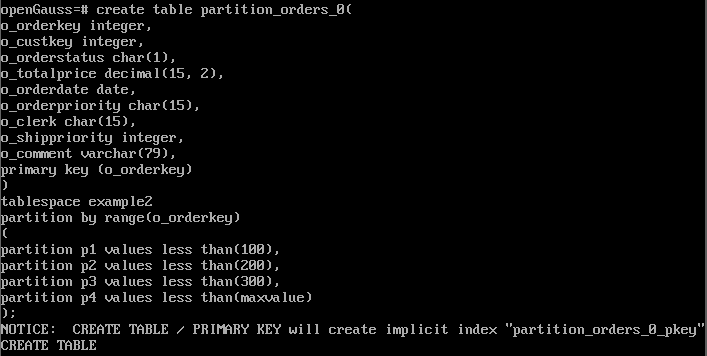
partition p3 values less than(300),

partition p4 values less than(maxvalue)

);

步骤

截图：



创建GLOBAL索引：

在o\_custkey上创建分区表GLOBAL索引，存储在表空间example2上

create index global\_index\_custkey on partition\_orders\_0(o\_custkey) global tablespace example2;

创建成功后截图：



用WinSCP查看，只有一个索引文件（前面4个是上面创建partition\_orders\_0分区表，4个分区对应4个文件）

WinSCP截图四个文件：



创建不指定索引分区名称的LOCAL索引：

在o\_totalprice上创建分区表LOCAL索引，不指定索引分区的名称，存储在example3上（原表与索引可以分别存储在不同的表空间上）

create index local\_index\_totalprice on partition\_orders\_0(o\_totalprice) local tablespace example3;

创建成功后截图：



用WinSCP查看，有4个索引文件（因为有4个分区）

WinSCP截图四个文件：



创建指定索引分区名称的LOCAL索引：

在o\_shippriority 上创建分区表LOCAL索引，指定索引分区的名称，p1,p2分区索引存储在example3上，p3,p4f 分区索引存储在example4上（不同分区的索引可以分别存储在不同的表空间上）

create index local\_index\_shippriority on partition\_orders\_0(o\_shippriority) local

(

partition p1\_index,

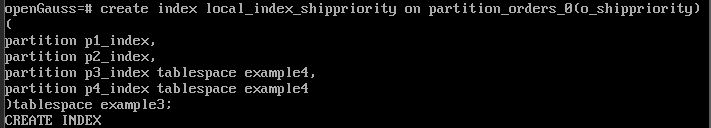
partition p2\_index,

partition p3\_index tablespace example4,

partition p4\_index tablespace example4

) tablespace example3;

创建成功后截图：



用WinSCP看到，分别在example3有2个索引文件和example4有2个索引文件



（2）管理索引

修改索引分区所在的表空间：

将分区索引p1\_index从example3移到example2

tpch=# ALTER INDEX local\_index\_shippriority MOVE PARTITION p1\_index TABLESPACE example2;

截图：



重命名索引分区：

将分区索引p2\_index重命名为p2\_index\_new

tpch=# ALTER INDEX local\_index\_shippriority RENAME PARTITION p2\_index TO p2\_index\_new;

截图：



删除索引 ：

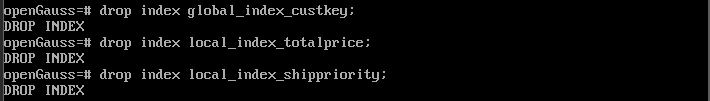
要删除索引只能删除整个索引，不能删除单独的分区索引

tpch=# drop index global\_index\_custkey;

tpch=# drop index local\_index\_totalprice;

tpch=# drop index local\_index\_shippriority;

截图：



# 5 实验总结

在本次数据库物理设计实验中，我们系统学习了在 openGauss 数据库中如何进行表空间、分区表和索引的创建与管理。

在实验的第一个阶段，我们深入学习了在 openGauss 数据库中如何创建和管理表空间。通过实际操作，我们成功创建了多个表空间，并了解了表空间的管理操作，包括查询和删除。这使我们对数据库结构的灵活管理有了更为深刻的认识。

在第二个阶段，我们着重学习了在 openGauss 数据库中创建分区表、插入数据以及对分区表进行管理的方法。通过实践，我们成功创建了分区表，插入了相应的数据，并学到了对分区表进行重命名和删除的操作。这一经验使我们更加理解了分区表在数据存储和查询优化中的应用。

最后，我们学习了在 openGauss 数据库中创建和管理索引的技术。通过实际操作，我们成功创建了索引，并理解了索引在数据库性能优化中的重要作用。同时，我们也学到了在普通表和分区表上创建和管理索引的方法。

本次实验让我们深入学习了 openGauss 数据库的表空间、分区表和索引的创建与管理。这些实践经验加强了我们对数据库管理技能的掌握，加深了对数据库物理设计原理的理解，为更高效地利用数据库提供了宝贵的经验。