实验三报告

关卡一：openGauss 数据库的编译和安装

1. 关卡验证

步骤 1 首先需要对数据库状态进行验证。

```
[omm@opengauss01 openGauss-server]$ gs_ctl status
```

（截图语句和执行结果）

```
[omm@opengauss01 ~]$ gs_ctl status
[2022-12-06 14:50:31.096][225310][][gs_ctl]: gs_ctl status,datadir is /opt/software/openGauss/data
gs_ctl: server is running (PID: 225266)
/opt/software/openGauss/bin/gaussdb "-D" "/opt/software/openGauss/data"
[omm@opengauss01 ~]$
```

步骤 2 对数据库进程进行截图验证，需包含数据库服务器的主机名。

```
[omm@opengauss01 openGauss-server]$ ps -ef|grep omm
```

（截图语句和执行结果）

```
[omm@opengauss01 ~]$ ps -ef|grep omm
root      225072    5175  0 14:47 pts/0    00:00:00 su - omm
omm       225073  225072  0 14:47 pts/0    00:00:00 -bash
omm       225266       1  1 14:48 pts/0    00:00:01 /opt/software/openGauss/bin/gaussdb -D /opt/software/openGauss/data
omm       225319  225073  0 14:51 pts/0    00:00:00 ps -ef
omm       225320  225073  0 14:51 pts/0    00:00:00 grep --color=auto omm
[omm@opengauss01 ~]$
```

关卡二：openGauss 数据导入及基本操作

1. 关卡验证

步骤 12 登录数据库验证

```
[omm@opengauss01 dbgen]$ gsql -d tpch -p 5432 -r
tpch=# select count(*) from supplier;
```

（截图语句和执行结果）

```
[omm@opengauss01 dbgen]$ gsql -d tpch -p 5432 -r
gsql ((GaussDB Kernel V500R002C00 build b2ff10be) compiled at 2022-12-06 14:40:30 commit 0 last mr  debug)
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

tpch=# select count(*) from supplier;
 count
-------
 10000
(1 row)

tpch=#
```

步骤 21 登录数据库进行验证

```
[omm@opengauss01 ~]$ gsql -d tpch -p 5432 -r
tpch=# \dt
```

（截图语句和执行结果）

```
tpch=# \dt
                        List of relations
 Schema |        Name         | Type  | Owner |             Storage
--------+---------------------+-------+-------+-----------------------------------
 public | address_dimension   | table | omm   | {orientation=row,compression=no}
 public | customer            | table | omm   | {orientation=row,compression=no}
 public | date_dimension      | table | omm   | {orientation=row,compression=no}
 public | lineitem            | table | omm   | {orientation=row,compression=no}
 public | litemall_orders     | table | omm   | {orientation=row,compression=no}
 public | nation              | table | omm   | {orientation=row,compression=no}
 public | orders              | table | omm   | {orientation=row,compression=no}
 public | part                | table | omm   | {orientation=row,compression=no}
 public | partsupp            | table | omm   | {orientation=row,compression=no}
 public | region              | table | omm   | {orientation=row,compression=no}
 public | supplier            | table | omm   | {orientation=row,compression=no}
 public | user_dimension      | table | omm   | {orientation=row,compression=no}
(12 rows)
```

步骤 22 查询 customer 表的数据

```
tpch=# select * from customer limit 10;
```

（截图语句和执行结果）

```
tpch=# select * from customer limit 10;
 c_custkey |       c_name       |                c_address                | c_nationkey |     c_phone     | c_acctbal | c_mktsegment |
                                                c_comment
-----------+--------------------+-----------------------------------------+-------------+-----------------+-----------+--------------+----------------
-----------------------------------------------------------------------------------------------------------------------
         1 | Customer#000000001 | IVhzIApeRb ot,c,E                       |          15 | 25-989-741-2988 |    711.56 | BUILDING     | to the even, reg
ular platelets. regular, ironic epitaphs nag e
         2 | Customer#000000002 | XSTf4,NCwDVaWNe6tEgvwfmRchLXak          |          13 | 23-768-687-3665 |    121.65 | AUTOMOBILE   | l accounts. blit
hely ironic theodolites integrate boldly: caref
         3 | Customer#000000003 | MG9kdTD2WBHm                            |           1 | 11-719-748-3364 |   7498.12 | AUTOMOBILE   |  deposits eat sl
yly ironic, even instructions. express foxes detect slyly. blithely even accounts abov
         4 | Customer#000000004 | XxVSJsLAGtn                             |           4 | 14-128-190-5944 |   2866.83 | MACHINERY    |  requests. final
, regular ideas sleep final accou
         5 | Customer#000000005 | KvpyuHCplrB84WgAiGV6sYpZq7Tj            |           3 | 13-750-942-6364 |    794.47 | HOUSEHOLD    | n accounts will
have to unwind. foxes cajole accor
         6 | Customer#000000006 | sKZz0CsnMD7mp4Xd0YrBvx,LREYKUWAh yVn     |          20 | 30-114-968-4951 |   7638.57 | AUTOMOBILE   | tions. even depo
sits boost according to the slyly bold packages. final accounts cajole requests. furious
         7 | Customer#000000007 | TcGe5gaZNgVePxU5kRrvXBfkasDTea          |          18 | 28-190-982-9759 |   9561.95 | AUTOMOBILE   | ainst the ironic
, express theodolites. express, even pinto beans among the exp
         8 | Customer#000000008 | I0B10bB0AymmC, 0PrRYBCPlyGJ8xcBPmWhl5   |          17 | 27-147-574-9335 |   6819.74 | BUILDING     | among the slyly
regular theodolites kindle blithely courts. carefully even theodolites haggle slyly along the ide
         9 | Customer#000000009 | xKiAFTjUsCuxfeleNqefumTrjS              |           8 | 18-338-906-3675 |   8324.07 | FURNITURE    | r theodolites ac
cording to the requests wake thinly excuses: pending requests haggle furiousl
        10 | Customer#000000010 | 6LrEaV6KR6PLVcgl2ArL Q3rqzLzcT1 v2      |           5 | 15-741-346-9870 |   2753.54 | HOUSEHOLD    | es regular depos
its haggle. fur
(10 rows)
```

2. 思考题

数据初始化中出现了 TPC-H，这是什么?

TPC-H 是事务处理性能委员会制定的基准程序之一，是针对随机查询/商业智能处理能力的决策支持基准，其主要目的是评价特定查询的决策支持能力，强调服务器在数据挖掘、分析处理方面的能力。该基准由一套面向业务的即时查询（ad-hoc）和并发数据修改组成，模拟了决策支持系统中的数据库操作，测试数据库系统复杂查询的响应时间，以每小时执行的查询数作为度量指标。

关卡三：openGauss 的 AI4DB 特性应用

1. 关卡验证

(1) 使用 X-Tuner 进行参数优化

步骤 2 在原来 CloudShell 连接窗口中查看 queries01.log。

```
[omm@opengauss01 ~]$ tail -10 /opt/software/tpch-kit/dbgen/queries/queries01.log
```

（截图执行语句和结果）

步骤 3 切换至 root 用户，执行 X-Tuner 进行参数建议优化

```
[omm@opengauss01 ~]$ exit
[root@opengauss01 xtuner]# gs_xtuner recommend --db-name tpch --db-user omm --port 5432
--host 127.0.0.1 --host-user omm
```

（截图执行语句和结果）

```
+---------------------------+-----------+----------+----------+---------+
|           name            | recommend |   min    |   max    | restart |
+---------------------------+-----------+----------+----------+---------+
| default_statistics_target |    1000   |   100    |   1000   |  False  |
|    effective_cache_size   |  21602334 |  186756  | 21602334 |  False  |
|  effective_io_concurrency |    200    |   150    |   250    |  False  |
|      enable_mergejoin     |    off    |    0     |    1     |  False  |
|      enable_nestloop      |    off    |    0     |    1     |  False  |
|      max_connections      |    370    |    50    |   741    |   True  |
| max_prepared_transactions |    370    |    50    |   741    |   True  |
|     max_process_memory    |  28803112 | 22402420 | 28803112 |   True  |
|      random_page_cost     |    1.0    |    1.0   |    2.0   |  False  |
|       shared_buffers      |   186756  |  186760  |  214772  |   True  |
|        wal_buffers        |    5836   |   2048   |   5836   |   True  |
+---------------------------+-----------+----------+----------+---------+
[root@opengauss01 xtuner]#
```

步骤 6 获取参数值

```
[omm@opengauss01 ~]$ cd /opt/software/openGauss/data
[omm@opengauss01 data]$ cat postgresql.conf|grep -E
'shared_buffers|max_connections|effective_cache_size|effective_io_concurrency|wal_buffers|rando
m_page_cost|default_statistics_target'
```

（截图执行语句和结果）

```
[omm@opengauss01 data]$  cat postgresql.conf|grep -E 'shared_buffers|max_connections|effective_cache_size|effective_io_concurrency|wal_buffers|random
_page_cost|default_statistics_target'
max_connections = 370                   # (change requires restart)
# Note:  Increasing max_connections costs ~400 bytes of shared memory per
shared_buffers = 187388                 # min 128kB
bulk_write_ring_size = 2GB              # for bulkload, max shared_buffers
#standby_shared_buffers_fraction = 0.3 #control shared buffers use in standby, 0.1-1.0
effective_io_concurrency = 200          # 1-1000; 0 disables prefetching
wal_buffers = 5855                      # min 32kB
random_page_cost = 1                    # same scale as above
effective_cache_size = 21602940
default_statistics_target = 1000        # range 1-10000
# max_locks_per_transaction * (max_connections + max_prepared_transactions)
```

步骤 7 再次执行步骤 2，对比优化前的执行时间。

（截图执行语句和结果）

```
[omm@opengauss01 data]$  tail -10 /opt/software/tpch-kit/dbgen/queries/queries01.log
 13        |      888 | 6737713.99
 17        |      861 | 6460573.72
 18        |      964 | 7236687.40
 23        |      892 | 6701457.95
 29        |      948 | 7158866.63
 30        |      909 | 6808436.13
 31        |      922 | 6806670.18
(7 rows)

total time: 1174541  ms
[omm@opengauss01 data]$
```

优化前的时间为 1207247ms，优化后的时间为 1174541ms，比优化前少了 32706ms。

步骤 8 【附加题】有兴趣的同学可以尝试并截图记录于此。

（截图执行语句和结果）

```
[root@opengauss01 xtuner]# su - omm
```

```
[omm@opengauss01 ~]$ gs_guc set -D /opt/software/openGauss/data/    -c
"default_statistics_target = 1000" -c "effective_cache_size = 21602334"    -c
"effective_io_concurrency = 200" -c "max_connections = 370" -c "max_prepared_transactions =
370" -c "max_process_memory=28803112" -c "random_page_cost = 1" -c "shared_buffers =
186756" -c "wal_buffers = 5836" -c "enable_mergejoin=off" -c "enable_nestloop=off"
```

```
[omm@opengauss01 data]$ cat postgresql.conf|grep -E 'shared_buffers|max_connections|effective_cache_size|effective_io_concurrency|wal_buffers|random_
page_cost|default_statistics_target'
max_connections = 370                   # (change requires restart)
# Note:  Increasing max_connections costs ~400 bytes of shared memory per
shared_buffers = 186756                 # min 128kB
bulk_write_ring_size = 2GB              # for bulkload, max shared_buffers
#standby_shared_buffers_fraction = 0.3 #control shared buffers use in standby, 0.1-1.0
effective_io_concurrency = 200          # 1-1000; 0 disables prefetching
wal_buffers = 5836                      # min 32kB
random_page_cost = 1                    # same scale as above
effective_cache_size = 21602334
default_statistics_target = 1000        # range 1-10000
# max_locks_per_transaction * (max_connections + max_prepared_transactions)
```

```
[omm@opengauss01 data]$ tail -10 /opt/software/tpch-kit/dbgen/queries/queries01.log
 13         |      888 | 6737713.99
 17         |      861 | 6460573.72
 18         |      964 | 7236687.40
 23         |      892 | 6701457.95
 29         |      948 | 7158866.63
 30         |      909 | 6808436.13
 31         |      922 | 6806670.18
(7 rows)

total time: 321901  ms
```

优化前的时间为 1207247ms，优化后的时间为 321901ms，比优化前少了 885346ms。

(2) Index-advisor：索引推荐

步骤 4 使用 explain，对该 SQL 加以分析

```
tpch=# EXPLAIN
SELECT ad.province AS province, SUM(o.actual_price) AS GMV
  FROM litemall_orders o,
       address_dimension ad,
       date_dimension dd
 WHERE o.address_key = ad.address_key
   AND o.add_date = dd.date_key
   AND dd.year = 2020
   AND dd.month = 3
 GROUP BY ad.province
 ORDER BY SUM(o.actual_price) DESC;
```

（截图执行语句和结果）

```
tpch=# EXPLAIN
tpch-# SELECT ad.province AS province, SUM(o.actual_price) AS GMV
tpch-#   FROM litemall_orders o,
tpch-#        address_dimension ad,
tpch-#        date_dimension dd
tpch-#  WHERE o.address_key = ad.address_key
tpch-#    AND o.add_date = dd.date_key
tpch-#    AND dd.year = 2020
tpch-#    AND dd.month = 3
tpch-#  GROUP BY ad.province
tpch-#  ORDER BY SUM(o.actual_price) DESC;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Sort  (cost=4593.80..4593.88 rows=31 width=47)
   Sort Key: (sum(o.actual_price)) DESC
   ->  HashAggregate  (cost=4592.72..4593.03 rows=31 width=47)
         Group By Key: ad.province
         ->  Hash Join  (cost=4354.43..4585.97 rows=1351 width=15)
               Hash Cond: (ad.address_key = o.address_key)
               ->  Seq Scan on address_dimension ad  (cost=0.00..188.02 rows=8002 width=14)
               ->  Hash  (cost=4337.54..4337.54 rows=1351 width=9)
                     ->  Hash Join  (cost=1031.78..4337.54 rows=1351 width=9)
                           Hash Cond: (o.add_date = dd.date_key)
                           ->  Seq Scan on litemall_orders o  (cost=0.00..3041.00 rows=100000 width=13)
                           ->  Hash  (cost=1031.76..1031.76 rows=2 width=4)
                                 ->  Seq Scan on date_dimension dd  (cost=0.00..1031.76 rows=2 width=4)
                                       Filter: ((year = 2020) AND ((month)::bigint = 3))
(14 rows)

tpch=#
```

步骤 10 使用 explain，对该 SQL 加以分析

```
tpch=# EXPLAIN
SELECT ad.province AS province, SUM(o.actual_price) AS GMV
  FROM litemall_orders o,
       address_dimension ad,
       date_dimension dd
 WHERE o.address_key = ad.address_key
   AND o.add_date = dd.date_key
   AND dd.year = 2020
   AND dd.month = 3
 GROUP BY ad.province
 ORDER BY SUM(o.actual_price) DESC;
```

（截图执行语句和结果）

```
tpch=# set enable_hypo_index = on;
SET
tpch=#  EXPLAIN
tpch-# SELECT ad.province AS province, SUM(o.actual_price) AS GMV
tpch-#   FROM litemall_orders o,
tpch-#        address_dimension ad,
tpch-#        date_dimension dd
tpch-#  WHERE o.address_key = ad.address_key
tpch-#    AND o.add_date = dd.date_key
tpch-#    AND dd.year = 2020
tpch-#    AND dd.month = 3
tpch-#  GROUP BY ad.province
tpch-#  ORDER BY SUM(o.actual_price) DESC;
                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Sort  (cost=3579.58..3579.65 rows=31 width=47)
   Sort Key: (sum(o.actual_price)) DESC
   ->  HashAggregate  (cost=3578.50..3578.81 rows=31 width=47)
         Group By Key: ad.province
         ->  Hash Join  (cost=3340.21..3571.74 rows=1351 width=15)
               Hash Cond: (ad.address_key = o.address_key)
               ->  Seq Scan on address_dimension ad  (cost=0.00..188.02 rows=8002 width=14)
               ->  Hash  (cost=3323.32..3323.32 rows=1351 width=9)
                     ->  Hash Join  (cost=17.56..3323.32 rows=1351 width=9)
                           Hash Cond: (o.add_date = dd.date_key)
                           ->  Seq Scan on litemall_orders o  (cost=0.00..3041.00 rows=100000 width=13)
                           ->  Hash  (cost=17.53..17.53 rows=2 width=4)
                                 ->  Index Scan using <16508>btree_date_dimension_year on date_dimension dd  (cost=0.00..17.53 rows=2 width=4)
                                       Index Cond: (year = 2020)
                                       Filter: ((month)::bigint = 3)
(15 rows)

tpch=#
```

步骤 11 【附加题】有兴趣的同学可以尝试并截图记录于此。仅需要从 queries.sql 文件里选择一条或多条进行索引优化即可。

（截图执行语句和结果）

选择 queries.sql 中的如下语句：

```
tpch=# select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= date '1998-12-01' - interval '90' day
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
```

耗时较长，执行结果如下：

```
tpch=# select
tpch-# l_returnflag,
tpch-# l_linestatus,
tpch-# sum(l_quantity) as sum_qty,
tpch-# sum(l_extendedprice) as sum_base_price,
tpch-# sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
tpch-# sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
tpch-# avg(l_quantity) as avg_qty,
tpch-# avg(l_extendedprice) as avg_price,
tpch-# avg(l_discount) as avg_disc,
tpch-# count(*) as count_order
tpch-# from
tpch-# lineitem
tpch-# where
tpch-# l_shipdate <= date '1998-12-01' - interval '90' day
tpch-# group by
tpch-# l_returnflag,
tpch-# l_linestatus
tpch-# order by
tpch-# l_returnflag,
tpch-# l_linestatus;
 l_returnflag | l_linestatus |   sum_qty   | sum_base_price |  sum_disc_price   |     sum_charge      |       avg_qty       |     avg_price      |
    avg_disc        | count_order
--------------+--------------+-------------+----------------+-------------------+---------------------+---------------------+--------------------+--
---------------------+-------------
 A            | F            | 37734107.00 |  56586554400.73 |  53758257134.8700 |  55909065222.827692 | 25.5220058532573370 | 38273.129734621672 | .
04998529583859761162 |     1478493
 N            | F            |   991417.00 |   1487504710.38 |   1413082168.0541 |   1469649223.194375 | 25.5164719205229835 | 38284.467760848304 | .
05009342667421629691 |       38854
 N            | O            | 74476040.00 | 111701729697.74 | 106118230307.6056 | 110367043872.497010 | 25.5022267695849915 | 38249.117988908270 | .
04999658605370408037 |     2920374
 R            | F            | 37719753.00 |  56568041380.90 |  53741292684.6040 |  55889619119.831932 | 25.5057936126907707 | 38250.854626099657 | .
05000940583012705647 |     1478870
(4 rows)

tpch=#
```

使用 explain，对该 SQL 加以分析

```
tpch=# EXPLAIN
select
    l_returnflag,
    l_linestatus,
```

```
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= date '1998-12-01' - interval '90' day
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
```

获得执行计划结果为：

```
tpch=#
tpch=# EXPLAIN
tpch-# select
tpch-# l_returnflag,
tpch-# l_linestatus,
tpch-# sum(l_quantity) as sum_qty,
tpch-# sum(l_extendedprice) as sum_base_price,
tpch-# sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
tpch-# sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
tpch-# avg(l_quantity) as avg_qty,
tpch-# avg(l_extendedprice) as avg_price,
tpch-# avg(l_discount) as avg_disc,
tpch-# count(*) as count_order
tpch-# from
tpch-# lineitem
tpch-# where
tpch-# l_shipdate <= date '1998-12-01' - interval '90' day
tpch-# group by
tpch-# l_returnflag,
tpch-# l_linestatus
tpch-# order by
tpch-# l_returnflag,
tpch-# l_linestatus;
                                     QUERY PLAN
-------------------------------------------------------------------------------------------------
 Sort  (cost=436259.85..436259.86 rows=6 width=257)
   Sort Key: l_returnflag, l_linestatus
   ->  HashAggregate  (cost=436259.66..436259.77 rows=6 width=257)
         Group By Key: l_returnflag, l_linestatus
         ->  Seq Scan on lineitem  (cost=0.00..199621.06 rows=5915965 width=25)
               Filter: (l_shipdate <= '1998-09-02 00:00:00'::timestamp without time zone)
(6 rows)

tpch=#
```

使用索引推荐功能，对查询语句进行推荐：

```
tpch=# select * from gs_index_advise('
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
```

```
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= date "1998-12-01" - interval "90" day
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus');
```

获得索引推荐结果如下：

```
tpch=# select * from gs_index_advise('
tpch'# select
tpch'# l_returnflag,
tpch'# l_linestatus,
tpch'# sum(l_quantity) as sum_qty,
tpch'# sum(l_extendedprice) as sum_base_price,
tpch'# sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
tpch'# sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
tpch'# avg(l_quantity) as avg_qty,
tpch'# avg(l_extendedprice) as avg_price,
tpch'# avg(l_discount) as avg_disc,
tpch'# count(*) as count_order
tpch'# from
tpch'# lineitem
tpch'# where
tpch'# l_shipdate <= date ''1998-12-01'' - interval ''90'' day
tpch'# group by
tpch'# l_returnflag,
tpch'# l_linestatus
tpch'# order by
tpch'# l_returnflag,
tpch'# l_linestatus');
 schema |  table   |           column
--------+----------+-----------------------------
 public | lineitem | (l_returnflag,l_linestatus)
(1 row)
```

在 lineitem 表上创建虚拟索引列。

```
tpch=# select * from hypopg_create_index('create index on lineitem(l_returnflag,l_linestatus)');
```

```
tpch=# select * from hypopg_create_index('create index on lineitem(l_returnflag,l_linestatus)');
 indexrelid |                  indexname
------------+---------------------------------------------
      16514 | <16514>btree_lineitem_l_returnflag_l_linestatus
(1 row)
```

查看创建的虚拟索引列。

```
tpch=# select * from hypopg_display_index();
```

```
tpch=# select * from hypopg_display_index();
                   indexname                    | indexrelid |  table   |           column
------------------------------------------------+------------+----------+----------------------------
 <16514>btree_lineitem_l_returnflag_l_linestatus |      16514 | lineitem | (l_returnflag, l_linestatus)
(1 row)
```

获取索引虚拟列大小结果

```
tpch=# select * from hypopg_estimate_size(16514);
```

返回结果为：

```
tpch=# select * from hypopg_estimate_size(16514);
 hypopg_estimate_size
----------------------
            219709440
(1 row)
```

开启 GUC 参数 enable_hypo_index。

```
tpch=# set enable_hypo_index = on;
```

再次使用 explain，对刚才执行的 SQL 加以分析

```
tpch=# set enable_hypo_index = on;
SET
tpch=# EXPLAIN
tpch-# select
tpch-# l_returnflag,
tpch-# l_linestatus,
tpch-# sum(l_quantity) as sum_qty,
tpch-# sum(l_extendedprice) as sum_base_price,
tpch-# sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
tpch-# sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
tpch-# avg(l_quantity) as avg_qty,
tpch-# avg(l_extendedprice) as avg_price,
tpch-# avg(l_discount) as avg_disc,
tpch-# count(*) as count_order
tpch-# from
tpch-# lineitem
tpch-# where
tpch-# l_shipdate <= date '1998-12-01' - interval '90' day
tpch-# group by
tpch-# l_returnflag,
tpch-# l_linestatus
tpch-# order by
tpch-# l_returnflag,
tpch-# l_linestatus;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------------
 Sort  (cost=436259.85..436259.86 rows=6 width=257)
   Sort Key: l_returnflag, l_linestatus
   ->  HashAggregate  (cost=436259.66..436259.77 rows=6 width=257)
         Group By Key: l_returnflag, l_linestatus
         ->  Seq Scan on lineitem  (cost=0.00..199621.06 rows=5915965 width=25)
               Filter: (l_shipdate <= '1998-09-02 00:00:00'::timestamp without time zone)
(6 rows)

tpch=#
```

可以看出该语句并未奏效，分析发现，这是由于该语句仅是单表操作。

清理创建的索引虚拟列：

```
tpch=# select * from hypopg_reset_index();
```

```
tpch=#  select * from hypopg_reset_index();
 hypopg_reset_index
--------------------

(1 row)
```

因此挑选多表操作的语句：

```
tpch=#  select
        s_acctbal,
        s_name,
        n_name,
        p_partkey,
        p_mfgr,
        s_address,
        s_phone,
        s_comment
from
        part,
        supplier,
        partsupp,
        nation,
        region
where
        p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and p_size = 15
        and p_type like '%BRASS'
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'EUROPE'
        and ps_supplycost = (
                select
                        min(ps_supplycost)
                from
                        partsupp,
                        supplier,
                        nation,
                        region
                where
p_partkey = ps_partkey
                        and s_suppkey = ps_suppkey
                        and s_nationkey = n_nationkey
                        and n_regionkey = r_regionkey
                        and r_name = 'EUROPE'
        )
order by
        s_acctbal desc,
        n_name,
        s_name,
        p_partkey;
;
```

```
tpch-#         and r_name = 'EUROPE'
tpch-#         and ps_supplycost = (
tpch(#                 select
tpch(#                         min(ps_supplycost)
tpch(#                 from
tpch(#                         partsupp,
tpch(#                         supplier,
tpch(#                         nation,
tpch(#                         region
tpch(#                 where
tpch(# p_partkey = ps_partkey
tpch(#                         and s_suppkey = ps_suppkey
tpch(#                         and s_nationkey = n_nationkey
tpch(#                         and n_regionkey = r_regionkey
tpch(#                         and r_name = 'EUROPE'
tpch(#         )
tpch-# order by
tpch-#         s_acctbal desc,
tpch-#         n_name,
tpch-#         s_name,
tpch-#         p_partkey;
 s_acctbal |          s_name          |         n_name          | p_partkey |         p_mfgr          |               s_address
 |     s_phone     |                                  s_comment
-----------+--------------------------+-------------------------+-----------+-------------------------+----------------------------------------
-+-----------------+-----------------------------------------------------------------------------------------------------
   9938.53 | Supplier#000005359       | UNITED KINGDOM          |    185358 | Manufacturer#4          | QKuHYh,vZGiwu2FWEJoLDx04
 | 33-429-790-6131 | uriously regular requests hag
   9937.84 | Supplier#000005969       | ROMANIA                 |    108438 | Manufacturer#1          | ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa
 | 29-520-692-3537 | efully express instructions. regular requests against the slyly fin
   9936.22 | Supplier#000005250       | UNITED KINGDOM          |       249 | Manufacturer#4          | B3rqp0xbSEim4Mpy2RH J
 | 33-320-228-2957 | etect about the furiously final accounts. slyly ironic pinto beans sleep inside the furiously
   9923.77 | Supplier#000002324       | GERMANY                 |     29821 | Manufacturer#4          | y3OD9UywSTOk
 | 17-779-299-1839 | ackages boost blithely. blithely regular deposits c
   9871.22 | Supplier#000006373       | GERMANY                 |     43868 | Manufacturer#5          | J8fcXWsTqM
 | 17-813-485-8637 | etect blithely bold asymptotes. fluffily ironic platelets wake furiously; blit
   9870.78 | Supplier#000001286       | GERMANY                 |     81285 | Manufacturer#2          | YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH
 | 17-516-924-4574 |  regular accounts. furiously unusual courts above the fi
   9870.78 | Supplier#000001286       | GERMANY                 |    181285 | Manufacturer#4          | YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH
 | 17-516-924-4574 |  regular accounts. furiously unusual courts above the fi
   9852.52 | Supplier#000008973       | RUSSIA                  |     18972 | Manufacturer#2          | t5L67YdBYYH6o,Vz24jpDyQ9
 | 32-188-594-7038 | rns wake final foxes. carefully unusual depende
   9847.83 | Supplier#000008097       | RUSSIA                  |    130557 | Manufacturer#2          | xMeQ7bpE69NzdwLoX
 | 32-375-640-3593 |  the special excuses. silent sentiments serve carefully final ac
   9847.57 | Supplier#000006345       | FRANCE                  |     86344 | Manufacturer#1          | VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag
 | 16-886-766-7945 | ges. slyly regular requests are. ruthless, express excuses cajole blithely across the unu
   9847.57 | Supplier#000006345       | FRANCE                  |    173827 | Manufacturer#2          | VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag
 | 16-886-766-7945 | ges. slyly regular requests are. ruthless, express excuses cajole blithely across the unu
   9836.93 | Supplier#000007342       | RUSSIA                  |      4841 | Manufacturer#4          | JOlK7C1,7xrEZSSOw
 | 32-599-414-5385 | blithely carefully bold theodolites. fur
   9817.10 | Supplier#000002352       | RUSSIA                  |    124815 | Manufacturer#2          | 4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw
 | 32-551-831-1437 | wake carefully alongside of the carefully final ex
   9817.10 | Supplier#000002352       | RUSSIA                  |    152351 | Manufacturer#3          | 4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw
 | 32-551-831-1437 | wake carefully alongside of the carefully final ex
   9739.86 | Supplier#000003384       | FRANCE                  |    138357 | Manufacturer#2          | o,Z3v4POifevE k9U1b 6J1ucX,I
 | 16-494-913-5925 | s after the furiously bold packages sleep fluffily idly final requests: quickly final
   9721.95 | Supplier#000008757       | UNITED KINGDOM          |    156241 | Manufacturer#3          | Atg6GnM4dT2
```

使用 explain，对该 SQL 加以分析

```
tpch=#  EXPLAIN
select
        s_acctbal,
        s_name,
        n_name,
        p_partkey,
        p_mfgr,
        s_address,
        s_phone,
        s_comment
from
        part,
        supplier,
        partsupp,
        nation,
        region
where
        p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and p_size = 15
        and p_type like '%BRASS'
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'EUROPE'
        and ps_supplycost = (
                select
                        min(ps_supplycost)
                from
                        partsupp,
```

```
                        supplier,
                        nation,
                        region
                    where
p_partkey = ps_partkey
                        and s_suppkey = ps_suppkey
                        and s_nationkey = n_nationkey
                        and n_regionkey = r_regionkey
                        and r_name = 'EUROPE'
            )
order by
            s_acctbal desc,
            n_name,
            s_name,
            p_partkey;
;
```

获得执行计划结果为：

```
tpch-#         s_acctbal desc,
tpch-#         n_name,
tpch-#         s_name,
tpch-#         p_partkey;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------
 Sort  (cost=71159.24..71159.24 rows=1 width=270)
   Sort Key: public.supplier.s_acctbal DESC, public.nation.n_name, public.supplier.s_name, public.part.p_partkey
   ->  Hash Join  (cost=42631.34..71159.23 rows=1 width=270)
         Hash Cond: ((public.part.p_partkey = subquery."?column?") AND (public.partsupp.ps_supplycost = subquery.min))
         ->  Hash Join  (cost=7499.82..36027.56 rows=17 width=280)
               Hash Cond: (public.partsupp.ps_suppkey = public.supplier.s_suppkey)
               ->  Hash Join  (cost=7115.09..35631.60 rows=2951 width=44)
                     Hash Cond: (public.partsupp.ps_partkey = public.part.p_partkey)
                     ->  Seq Scan on partsupp  (cost=0.00..25487.00 rows=800000 width=14)
                     ->  Hash  (cost=7106.00..7106.00 rows=727 width=30)
                           ->  Seq Scan on part  (cost=0.00..7106.00 rows=727 width=30)
                                 Filter: (((p_type)::text ~~ '%BRASS'::text) AND (p_size = 15))
               ->  Hash  (cost=384.03..384.03 rows=56 width=244)
                     ->  Hash Join  (cost=24.46..384.03 rows=56 width=244)
                           Hash Cond: (public.supplier.s_nationkey = public.nation.n_nationkey)
                           ->  Seq Scan on supplier  (cost=0.00..323.00 rows=10000 width=144)
                           ->  Hash  (cost=24.45..24.45 rows=1 width=108)
                                 ->  Hash Join  (cost=12.24..24.45 rows=1 width=108)
                                       Hash Cond: (public.nation.n_regionkey = public.region.r_regionkey)
                                       ->  Seq Scan on nation  (cost=0.00..11.76 rows=176 width=112)
                                       ->  Hash  (cost=12.22..12.22 rows=1 width=4)
                                             ->  Seq Scan on region  (cost=0.00..12.22 rows=1 width=4)
                                                   Filter: (r_name = 'EUROPE'::bpchar)
         ->  Hash  (cost=35131.28..35131.28 rows=16 width=36)
               ->  Subquery Scan on subquery  (cost=35130.96..35131.28 rows=16 width=36)
                     ->  HashAggregate  (cost=35130.96..35131.12 rows=16 width=42)
                           Group By Key: public.partsupp.ps_partkey
                           ->  Hash Join  (cost=7499.82..35130.88 rows=16 width=10)
                                 Hash Cond: (public.partsupp.ps_suppkey = public.supplier.s_suppkey)
                                 ->  Hash Semi Join  (cost=7115.09..34734.92 rows=2951 width=14)
                                       Hash Cond: (public.partsupp.ps_partkey = public.part.p_partkey)
                                       ->  Seq Scan on partsupp  (cost=0.00..25487.00 rows=800000 width=14)
                                       ->  Hash  (cost=7106.00..7106.00 rows=727 width=4)
                                             ->  Seq Scan on part  (cost=0.00..7106.00 rows=727 width=4)
                                                   Filter: (((p_type)::text ~~ '%BRASS'::text) AND (p_size = 15))
                                 ->  Hash  (cost=384.03..384.03 rows=56 width=4)
                                       ->  Hash Join  (cost=24.46..384.03 rows=56 width=4)
                                             Hash Cond: (public.supplier.s_nationkey = public.nation.n_nationkey)
                                             ->  Seq Scan on supplier  (cost=0.00..323.00 rows=10000 width=8)
                                             ->  Hash  (cost=24.45..24.45 rows=1 width=4)
                                                   ->  Hash Join  (cost=12.24..24.45 rows=1 width=4)
                                                         Hash Cond: (public.nation.n_regionkey = public.region.r_regionkey)
                                                         ->  Seq Scan on nation  (cost=0.00..11.76 rows=176 width=8)
                                                         ->  Hash  (cost=12.22..12.22 rows=1 width=4)
                                                               ->  Seq Scan on region  (cost=0.00..12.22 rows=1 width=4)
                                                                     Filter: (r_name = 'EUROPE'::bpchar)
(46 rows)

tpch=#
```

使用索引推荐功能，对查询语句进行推荐：

```
tpch=# select * from gs_index_advise(' select
```

```
        s_acctbal,
        s_name,
        n_name,
        p_partkey,
        p_mfgr,
        s_address,
        s_phone,
        s_comment
from
        part,
        supplier,
        partsupp,
        nation,
        region
where
        p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and p_size = 15
        and p_type like "%BRASS"
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = "EUROPE"
        and ps_supplycost = (
                select
                        min(ps_supplycost)
                from
                        partsupp,
                        supplier,
                        nation,
                        region
                where
p_partkey = ps_partkey
                        and s_suppkey = ps_suppkey
                        and s_nationkey = n_nationkey
                        and n_regionkey = r_regionkey
                        and r_name = "EUROPE"
        )
order by
        s_acctbal desc,
        n_name,
        s_name,
        p_partkey;
');
```

获得索引推荐结果如下：

```
tpch'#         region
tpch'# where
tpch'#         p_partkey = ps_partkey
tpch'#         and s_suppkey = ps_suppkey
tpch'#         and p_size = 15
tpch'#         and p_type like ''%BRASS''
tpch'#         and s_nationkey = n_nationkey
tpch'#         and n_regionkey = r_regionkey
tpch'#         and r_name = ''EUROPE''
tpch'#         and ps_supplycost = (
tpch'#                 select
tpch'#                         min(ps_supplycost)
tpch'#                 from
tpch'#                         partsupp,
tpch'#                         supplier,
tpch'#                         nation,
tpch'#                         region
tpch'#                 where
tpch'# p_partkey = ps_partkey
tpch'#                         and s_suppkey = ps_suppkey
tpch'#                         and s_nationkey = n_nationkey
tpch'#                         and n_regionkey = r_regionkey
tpch'#                         and r_name = ''EUROPE''
tpch'#         )
tpch'# order by
tpch'#         s_acctbal desc,
tpch'#         n_name,
tpch'#         s_name,
tpch'#         p_partkey;
tpch'# ');
 schema |  table   |                   column
--------+----------+-------------------------------------------
 public | part     | (p_partkey,p_size)
 public | supplier | (s_suppkey,s_nationkey),(s_acctbal,s_name)
 public | partsupp | (ps_partkey,ps_suppkey)
 public | nation   | (n_name)
 public | region   | (r_regionkey)
(5 rows)

tpch=#
```

在 part、supplier、partsupp、nation 和 region 表上创建虚拟索引列：

```
tpch=# select * from hypopg_create_index('create index on part(p_partkey,p_size)');
tpch=# select * from hypopg_create_index('create index on supplier(s_suppkey,s_nationkey,
s_acctbal,s_name)');
tpch=# select * from hypopg_create_index('create index on
partsupp(ps_partkey,ps_suppkey)');
tpch=# select * from hypopg_create_index('create index on nation(n_name)');
tpch=# select * from hypopg_create_index('create index on region(r_regionkey)');
```

```
tpch=#  select * from hypopg_create_index('create index on part(p_partkey,p_size)');
 indexrelid |             indexname
------------+-----------------------------------
      16515 | <16515>btree_part_p_partkey_p_size
(1 row)

tpch=# select * from hypopg_create_index('create index on supplier(s_suppkey,s_nationkey, s_acctbal,s_name)');
 indexrelid |                          indexname
------------+-------------------------------------------------------------
      16516 | <16516>btree_supplier_s_suppkey_s_nationkey_s_acctbal_s_name
(1 row)

tpch=#  select * from hypopg_create_index('create index on partsupp(ps_partkey,ps_suppkey)');
 indexrelid |                  indexname
------------+-----------------------------------------------
      16517 | <16517>btree_partsupp_ps_partkey_ps_suppkey
(1 row)

tpch=# select * from hypopg_create_index('create index on nation(n_name)');
 indexrelid |         indexname
------------+----------------------------
      16518 | <16518>btree_nation_n_name
(1 row)

tpch=# select * from hypopg_create_index('create index on region(r_regionkey)');
 indexrelid |            indexname
------------+---------------------------------
      16519 | <16519>btree_region_r_regionkey
(1 row)
```

查看创建的虚拟索引列：

```
tpch=# select * from hypopg_display_index();
```

获得创建虚拟列的结果：

```
tpch=# select * from hypopg_display_index();
                        indexname                         | indexrelid |  table   |                  column
----------------------------------------------------------+------------+----------+-----------------------------------------
 <16515>btree_part_p_partkey_p_size                       |      16515 | part     | (p_partkey, p_size)
 <16516>btree_supplier_s_suppkey_s_nationkey_s_acctbal_s_name |      16516 | supplier | (s_suppkey, s_nationkey, s_acctbal, s_name)
 <16517>btree_partsupp_ps_partkey_ps_suppkey              |      16517 | partsupp | (ps_partkey, ps_suppkey)
 <16518>btree_nation_n_name                               |      16518 | nation   | (n_name)
 <16519>btree_region_r_regionkey                          |      16519 | region   | (r_regionkey)
(5 rows)
```

获取索引虚拟列大小结果

```
tpch=# select * from hypopg_estimate_size(16515);
tpch=# select * from hypopg_estimate_size(16516);
tpch=# select * from hypopg_estimate_size(16517);
tpch=# select * from hypopg_estimate_size(16518);
tpch=# select * from hypopg_estimate_size(16519);
```

返回结果为：

```
tpch=# select * from hypopg_estimate_size(16515);
 hypopg_estimate_size
----------------------
              8364032
(1 row)

tpch=# select * from hypopg_estimate_size(16516);
 hypopg_estimate_size
----------------------
              1146880
(1 row)

tpch=# select * from hypopg_estimate_size(16517);
 hypopg_estimate_size
----------------------
             33472512
(1 row)

tpch=# select * from hypopg_estimate_size(16518);
 hypopg_estimate_size
----------------------
                 8192
(1 row)

tpch=# select * from hypopg_estimate_size(16519);
 hypopg_estimate_size
----------------------
                 8192
(1 row)
```

开启 GUC 参数 enable_hypo_index：

```
tpch=# set enable_hypo_index = on;
```

再次使用 explain，对刚才执行的 SQL 加以分析

```
tpch=#  EXPLAIN
select
        s_acctbal,
        s_name,
        n_name,
        p_partkey,
        p_mfgr,
        s_address,
        s_phone,
        s_comment
from
        part,
        supplier,
        partsupp,
        nation,
        region
where
        p_partkey = ps_partkey
        and s_suppkey = ps_suppkey
        and p_size = 15
        and p_type like '%BRASS'
        and s_nationkey = n_nationkey
        and n_regionkey = r_regionkey
        and r_name = 'EUROPE'
        and ps_supplycost = (
                select
                        min(ps_supplycost)
                from
                        partsupp,
                        supplier,
                        nation,
                        region
                where
p_partkey = ps_partkey
                        and s_suppkey = ps_suppkey
                        and s_nationkey = n_nationkey
                        and n_regionkey = r_regionkey
                        and r_name = 'EUROPE'
        )
order by
        s_acctbal desc,
        n_name,
        s_name,
        p_partkey;
;
```

```
tpch-#         p_partkey;
                                                                                               QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------
 Sort  (cost=6461.31..6461.31 rows=1 width=270)
   Sort Key: public.supplier.s_acctbal DESC, public.nation.n_name, public.supplier.s_name, public.part.p_partkey
   ->  Nested Loop  (cost=6405.06..6461.30 rows=1 width=270)
         ->  Hash Join  (cost=6405.06..6460.95 rows=1 width=274)
               Hash Cond: (public.supplier.s_nationkey = public.nation.n_nationkey)
               ->  Nested Loop  (cost=6391.10..6446.98 rows=1 width=170)
                     ->  Nested Loop  (cost=6391.10..6446.70 rows=1 width=34)
                           Join Filter: ((min(public.partsupp.ps_supplycost)) = public.partsupp.ps_supplycost)
                           ->  Nested Loop  (cost=6391.10..6436.30 rows=3 width=66)
                                 ->  HashAggregate  (cost=6391.10..6391.26 rows=16 width=42)
                                       Group By Key: public.partsupp.ps_partkey
                                       ->  Hash Join  (cost=3859.74..6391.02 rows=16 width=10)
                                             Hash Cond: (public.partsupp.ps_suppkey = public.supplier.s_suppkey)
                                             ->  Nested Loop  (cost=3748.28..6268.32 rows=2951 width=14)
                                                   ->  HashAggregate  (cost=3748.28..3755.55 rows=727 width=4)
                                                         Group By Key: public.part.p_partkey
                                                         ->  Index Scan using <16515>btree_part_p_partkey_p_size on part  (cost=0.00..3746.46 rows=72
7 width=4)
                                                               Index Cond: (p_size = 15)
                                                               Filter: ((p_type)::text ~~ '%BRASS'::text)
                                                   ->  Index Scan using <16517>btree_partsupp_ps_partkey_ps_suppkey on partsupp  (cost=0.00..3.42 row
s=4 width=14)
                                                         Index Cond: (ps_partkey = public.part.p_partkey)
                                             ->  Hash  (cost=110.76..110.76 rows=56 width=4)
                                                   ->  Nested Loop  (cost=12.24..110.76 rows=56 width=4)
                                                         ->  Hash Join  (cost=12.24..24.45 rows=1 width=4)
                                                               Hash Cond: (public.nation.n_regionkey = public.region.r_regionkey)
                                                               ->  Seq Scan on nation  (cost=0.00..11.76 rows=176 width=8)
                                                               ->  Hash  (cost=12.22..12.22 rows=1 width=4)
                                                                     ->  Seq Scan on region  (cost=0.00..12.22 rows=1 width=4)
                                                                           Filter: (r_name = 'EUROPE'::bpchar)
                                                         ->  Index Only Scan using <16516>btree_supplier_s_suppkey_s_nationkey_s_acctbal_s_name on su
pplier  (cost=0.00..82.31 rows=400 width=8)
                                                               Index Cond: (s_nationkey = public.nation.n_nationkey)
                                 ->  Index Scan using <16515>btree_part_p_partkey_p_size on part  (cost=0.00..2.27 rows=53 width=30)
                                       Index Cond: ((p_partkey = public.partsupp.ps_partkey) AND (p_size = 15))
                                       Filter: ((p_type)::text ~~ '%BRASS'::text)
                           ->  Index Scan using <16517>btree_partsupp_ps_partkey_ps_suppkey on partsupp  (cost=0.00..3.42 rows=4 width=14)
                                 Index Cond: (ps_partkey = public.part.p_partkey)
                     ->  Index Scan using <16516>btree_supplier_s_suppkey_s_nationkey_s_acctbal_s_name on supplier  (cost=0.00..0.27 rows=1 width=144
)
                           Index Cond: (s_suppkey = public.partsupp.ps_suppkey)
               ->  Hash  (cost=11.76..11.76 rows=176 width=112)
                     ->  Seq Scan on nation  (cost=0.00..11.76 rows=176 width=112)
         ->  Index Scan using <16519>btree_region_r_regionkey on region  (cost=0.00..0.34 rows=1 width=4)
               Index Cond: (r_regionkey = public.nation.n_regionkey)
               Filter: (r_name = 'EUROPE'::bpchar)
(43 rows)
```

结果中出现：

```
->  Index Scan using <16517>btree_partsupp_ps_partkey_ps_suppkey on partsupp  (cost=0.00..3.42 rows=4 width=14)
      Index Cond: (ps_partkey = public.part.p_partkey)
```

```
->  Index Scan using <16519>btree_region_r_regionkey on region  (cost=0.00..0.34 rows=1 width=4)
      Index Cond: (r_regionkey = public.nation.n_regionkey)
```

说明索引推荐已经奏效。

清理创建的索引虚拟列：

```
tpch=# select * from hypopg_reset_index();
```

```
tpch=# select * from hypopg_reset_index();
 hypopg_reset_index
--------------------

(1 row)
```

关卡四【附加题】：openGauss 的 DB4AI 特性应用

*本关卡为附加题，有兴趣的同学可以尝试实验并记录于此。

1. 关卡验证

步骤 10 利用训练好的逻辑回归模型预测数据，并与 SVM 算法进行比较，将执行结果截图。

```
openGauss=# SELECT tax, bath, size, price, price < 100000 AS price_actual, PREDICT BY
house_binary_classifier (FEATURES tax, bath, size) AS price_svm_pred, PREDICT BY
house_logistic_classifier (FEATURES tax, bath, size) AS price_logistic_pred FROM houses;
```

（截图执行语句和结果）

```
openGauss=# SELECT tax, bath, size, price, price < 100000 AS price_actual, PREDICT BY house_binary_classif
_pred, PREDICT BY house_logistic_classifier (FEATURES tax, bath, size) AS price_logistic_pred FROM houses;
 tax  | bath | size | price  | price_actual | price_svm_pred | price_logistic_pred
------+------+------+--------+--------------+----------------+---------------------
  590 |    1 |  770 |  50000 | t            | t              | t
 1050 |    2 | 1410 |  85000 | t            | t              | t
   20 |    1 | 1060 |  22500 | t            | t              | t
  870 |    2 | 1300 |  90000 | t            | t              | t
 1320 |    2 | 1500 | 133000 | f            | t              | t
 1350 |    1 |  820 |  98500 | t            | f              | f
 2790 |  2.5 | 2130 | 260000 | f            | f              | f
  680 |    1 | 1170 | 142500 | f            | t              | t
 1840 |    2 | 1500 | 160000 | f            | f              | f
 3680 |    2 | 2790 | 240000 | f            | f              | f
 1660 |    1 | 1030 |  87000 | t            | f              | f
 1620 |    2 | 1250 | 118600 | f            | f              | f
 3100 |    2 | 1760 | 140000 | f            | f              | f
 2070 |    3 | 1550 | 148000 | f            | f              | f
  650 |  1.5 | 1450 |  65000 | t            | t              | t
(15 rows)
```

可以看出，逻辑回归模型与 SVM 算法的结果类似，说明某些样例较为困难，这两种方法都无法预测正确。

清理工作：资源释放

1. 关卡验证

步骤 3 查看到列表中已没有资源时，表示弹性云服务器已删除。

（截图执行语句和结果）