关卡 3:
openGauss 的 AI4DB 特性应用

openGauss 的 AI4DB 特性应用

任务一：使用 Index-advisor 对 select 查询语句进行优化，并通过对比执行计划，得到优化前后的不同。

1. 使用 explain，对查询 2020 年 3 月订单表收入并进行排序的 SQL 加以分析，将结果截图。

```
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;
```

```
    province     |     gmv
-----------------+-------------
 上海市          | 13927163.00
 江苏省          | 13900815.00
 浙江省          | 13890720.00
 天津市          | 13750598.00
 北京市          | 13458554.00
 山东省          |  4381747.00
 重庆市          |  4172788.00
 福建省          |  4005384.00
 安徽省          |  3973989.00
 广东省          |  3842252.00
 河南省          |  1130277.00
 海南省          |  1113236.00
 新疆维吾尔自治区 |  1058210.00
 云南省          |  1041797.00
 西藏自治区      |  1040002.00
 贵州省          |  1003301.00
 吉林省          |  1002830.00
 辽宁省          |   982522.00
 宁夏回族自治区  |   885719.00
 陕西省          |   863598.00
 河北省          |   814762.00
 广西壮族自治区  |   812580.00
 黑龙江省        |   787436.00
 湖北省          |   775761.00
 湖南省          |   762020.00
 江西省          |   753550.00
 青海省          |   739701.00
 甘肃省          |   734090.00
 四川省          |   726917.00
 内蒙古自治区    |   667131.00
 山西省          |   643208.00
(31 rows)
```

2. 使用索引推荐功能，对查询语句进行推荐，将执行结果截图。

```
select * from gs_index_advise('
```

```
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');
```

```
tpch'# tpch'# tpch'# tpch'# tpch'# tpch'# tpch'# tpch'# tpch'# tpch'#        table       |
column
------------------+-----------------------
 litemall_orders   | (address_key,add_date)
 address_dimension |
 date_dimension    | (year)
(3 rows)
```

3. 查看创建的虚拟索引列，将执行结果截图。

```
select * from hypopg_display_index();
```

```
tpch'# tpch'# tpch'# tpch'# tpch'# tpch'# tpch'# tpch'# tpch'# tpch'#        table       |
column
-------------------+-----------------------
 litemall_orders   | (address_key,add_date)
 address_dimension |
 date_dimension    | (year)
(3 rows)

tpch=# select * from hypopg_display_index();
 indexname | indexrelid | table | column
-----------+------------+-------+--------
(0 rows)
```

4. 获取索引虚拟列大小结果（单位为：字节），将执行结果截图。

```
select * from hypopg_estimate_size(16715);
select * from hypopg_estimate_size(16716);
```

```
tpch=# select * from hypopg_estimate_size(16716);
 hypopg_estimate_size
----------------------
                    0
(1 row)

tpch=# select * from hypopg_estimate_size(16715);
 hypopg_estimate_size
----------------------
                    0
(1 row)
```

5.再次使用 explain，对该 SQL 加以分析，将执行结果截图。

```
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;
```

```
tpch-# tpch-# tpch-# tpch-# tpch-# tpch-# tpch-# tpch-# tpch-#                                                    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 widt
h=4)
                                       Filter: ((year = 2020) AND ((month)::bigint = 3))
(14 rows)
```

6. 删除某一个索引虚拟列，将执行结果截图。

```
select * from hypopg_drop_index(16715);
```

```
tpch=# select * from hypopg_drop_index(16715);
 hypopg_drop_index
-------------------
 f
(1 row)
```

7. 删除某一个索引虚拟列，将执行结果截图。

```
select * from hypopg_reset_index();
```

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

(1 row)
```

8. 查看索引虚拟列，将执行结果截图。

```
select * from hypopg_display_index();
```

```
tpch=# select * from hypopg_display_index();
 indexname | indexrelid | table | column
-----------+------------+-------+--------
(0 rows)
```

步骤 1　任务三：针对执行的 queries.sql 进行优化，为 customer、order、part 表增加主键索引，学员也可以针对 queries.sql 中的查询内容进行进一步优化。

步骤 2　在 order 表的 o_orderdate 列，以及 lineitem 表的 l_shipdate 列上创建索引，可以通过分析 queries.sql 查询，进行更深入的优化。

```
tpch=# ALTER TABLE "public"."customer" ADD CONSTRAINT "pk_customer" PRIMARY KEY
("c_custkey");
ALTER TABLE "public"."orders" ADD CONSTRAINT "pk_orders" PRIMARY KEY ("o_orderke
y");
ALTER TABLE "public"."part" ADD CONSTRAINT "pk_part" PRIMARY KEY ("p_partkey");
ERROR:  multiple primary keys for table "customer" are not allowed
tpch=# ERROR:  multiple primary keys for table "orders" are not allowed
tpch=# ERROR:  multiple primary keys for table "part" are not allowed
tpch=# CREATE INDEX idx_o_orderdate ON orders(o_orderdate);
CREATE INDEX idx_l_shipdate ON lineitem(l_shipdate);
CREATE INDEX
```

实践思考题 1：完成索引推荐功能的优化，使用索引推荐以及虚拟索引列的创建，对创建的索引虚拟列前后的执行时间进行对比截图；完成对 queries 查询的优化，对比优化前后的区别。

1. 避免检索大量不需要的数据
2. 更快速查询，更快速响应查询结果

实践思考题 2：索引的使用，对于执行 SQL 有什么好处？除了使用索引和参数外，还有哪些方面可以对数据库进行优化？

索引的好处：

3. 加快数据检索速度
4. 加快表的拼接
5. 减少分组和排序时间
6. 提高数据准确性和系统性能

数据库优化：

1. SQL 语句调优
2. 缓存调优