关卡 2-2

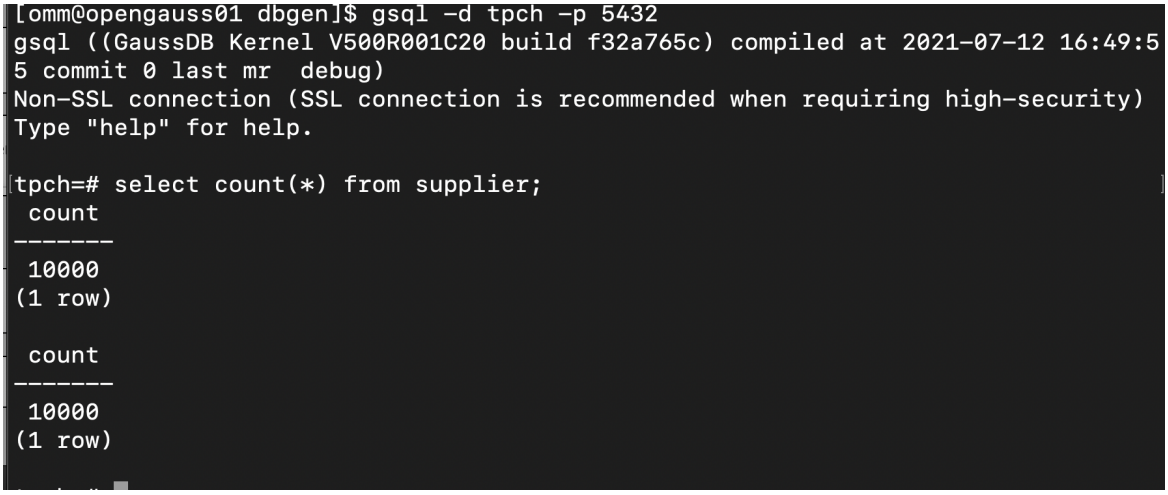
openGauss 数据导入及基本操作

openGauss 数据导入及基本操作

任务一：数据初始化验证

1. 查询 supplier 表的行数，并将结果进行图：

```
select count(*) from supplier;;
```



任务二：行存表与列存表执行效率对比

1. 2020 年上半年 litemall_orders 行存表与 litemall_orders_col 列存表中的 order_price 的总和查询，并对比执行效率截图

```
select sum (order_price) from litemall_orders where add_date between '20200101' and '20200701';
select sum (order_price) from litemall_orders_col where add_date between '20200101' and '20200701';
```

```
tpch=# select sum (order_price) from litemall_orders where add_date between '20200101' and '20200701';
     sum
-------------
 310586483.00
(1 row)

Time: 264.683 ms
tpch=# select sum (order_price) from litemall_orders_col where add_date between '20200101' and '20200701';
     sum
-------------
 310586483.00
(1 row)

Time: 24.089 ms
```

2. 2020 年上半年 litemall_orders 行存表与 litemall_orders_col 列存表中的 order_price 的平均值查询，并对比执行效率截图

```
select avg (order_price) from litemall_orders where add_date between '20200101' and '20200701';
select avg (order_price) from litemall_orders_col where add_date between '20200101' and '20200701';
```

```
tpch=# select avg(order_price) from litemall_orders where add_date between '20200101' and '20200701';
          avg
----------------------
 3105.8648300000000000
(1 row)

Time: 368.772 ms
tpch=# select avg(order_price) from litemall_orders_col where add_date between '20200101' and '20200701';
          avg
----------------------
 3105.8648300000000000
(1 row)

Time: 15.898 ms
```

3. 查询 litemall_orders 行存表与 litemall_orders_col 列存表中 order_id 为 6 的 order_price 的值，并对比执行效率截图。

```
select order_price from litemall_orders where order_id=6;
select order_price from litemall_orders_col where order_id=6;
```

```
tpch=# select order_price from litemall_orders where order_id=6;
 order_price
-------------
     2468.00
(1 row)

Time: 1.614 ms
tpch=# select order_price from litemall_orders_col where order_id=6;
 order_price
-------------
     2468.00
(1 row)

Time: 6.822 ms
```

4. 将 litemall_orders 行存表与 litemall_orders_col 列存表中 order_id 为 6 的 order_price 修改为 2468，并对比执行效率截图。

```
update litemall_orders set order_price=2468 where order_id=6;
update litemall_orders_col set order_price=2468 where order_id=6;
```

```
tpch=# update litemall_orders set order_price=2468 where order_id=6;
UPDATE 1
Time: 3.833 ms
tpch=# update litemall_orders_col set order_price=2468 where order_id=6;
UPDATE 1
Time: 107.553 ms
```

任务三：物化视图的使用

1. 创建物化视图所需要的表后，对表内容进行查询，对查询结果截图：

```
SELECT * FROM test_view;
```

```
tpch=# SELECT * FROM test_view;
 username | gender | totalspend
----------+--------+------------
 杨兰娟   |      2 |  119391.00
 柳高梅   |      2 |  116155.00
 韦小全   |      1 |  114072.00
 贲艳梅   |      2 |  112565.00
 强兰丽   |      2 |  111925.00
 滑小刚   |      1 |  110089.00
 席长梅   |      2 |  108247.00
 翁晓婷   |      2 |  107988.00
 娄高伟   |      1 |  107323.00
 苏长刚   |      0 |  104640.00
 喻高伟   |      1 |  102536.00
 袁晓轩   |      1 |  101835.00
 伏成峰   |      1 |  101725.00
 毕晓刚   |      1 |  101057.00
 金高芳   |      2 |  100322.00
(15 rows)

Time: 2.339 ms
```

2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

```
SELECT * FROM v_order;
```

```
tpch=# SELECT * FROM v_order;
 count
-------
    15
(1 row)

Time: 2.275 ms
```

3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

```
SELECT * FROM v_order;
```

```
tpch=# SELECT * FROM v_order;
 count
-------
    14
(1 row)

Time: 2.242 ms
```

4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

```
SELECT * FROM vi_order;
```

```
tpch=# SELECT * FROM vi_order;
 username | totalspend
----------+------------
 柳高梅   |  116155.00
 韦小全   |  114072.00
 贲艳梅   |  112565.00
 强兰丽   |  111925.00
 滑小刚   |  110089.00
 席长梅   |  108247.00
 翁晓婷   |  107988.00
 娄高伟   |  107323.00
 苏长刚   |  104640.00
 喻高伟   |  102536.00
 袁晓轩   |  101835.00
 伏成峰   |  101725.00
 毕晓刚   |  101057.00
 金高芳   |  100322.00
(14 rows)

Time: 2.859 ms
```

5. 对表进行操作后，刷新增量物化视图，查询物化视图结果，将执行结果截图。

```
SELECT * FROM vi_order;
```

```
tpch=# SELECT * FROM vi_order;
 username | totalspend
----------+------------
 柳高梅   |  116155.00
 韦小全   |  114072.00
 贲艳梅   |  112565.00
 强兰丽   |  111925.00
 滑小刚   |  110089.00
 席长梅   |  108247.00
 翁晓婷   |  107988.00
 娄高伟   |  107323.00
 苏长刚   |  104640.00
 喻高伟   |  102536.00
 袁晓轩   |  101835.00
 伏成峰   |  101725.00
 毕晓刚   |  101057.00
 金高芳   |  100322.00
 杨兰娟   |  119391.00
 杨兰娟   |  119391.00
(16 rows)

Time: 3.880 ms
```

```
tpch=# SELECT * FROM vi_order;
 username | totalspend
----------+------------
 柳高梅   |  116155.00
 韦小全   |  114072.00
 贲艳梅   |  112565.00
 强兰丽   |  111925.00
 滑小刚   |  110089.00
 席长梅   |  108247.00
 翁晓婷   |  107988.00
 娄高伟   |  107323.00
 苏长刚   |  104640.00
 喻高伟   |  102536.00
 袁晓轩   |  101835.00
 伏成峰   |  101725.00
 毕晓刚   |  101057.00
 金高芳   |  100322.00
 杨兰娟   |  119391.00
 杨兰娟   |  119391.00
 马景涛   |  139391.00
(17 rows)

Time: 3.878 ms
```

实践思考题 1：行存表与列存表在执行相同的 SQL 语句时，为何执行的时间不同？在执行哪些类型 SQL 时，行存表效率更高？在执行哪些类型 SQL 时，列存表效率更高？

1.不同的存储方法在不同的搜索场景下性能有较大差别，需要遍历的量不同。当你的核心业务是 OLTP 时采用行式存储，当你的核心业务是 OLAP 时，采用列式存储。

实践思考题 2：全量物化视图与增量物化视图有哪些差别？

全量：全量刷新机制是首先物化视图对应表中的数据采用 delete 全部删除，然后再从原表中使用 insert 把数据重新插入。

增量：主表上每插入或删除一条数据，对应物化视图日志中同样会插入一条数据，物化视图刷新后主表上物化视图日志记录信息会被清空，重新开始记录后面的更新。