openGauss AI特性创新实践课



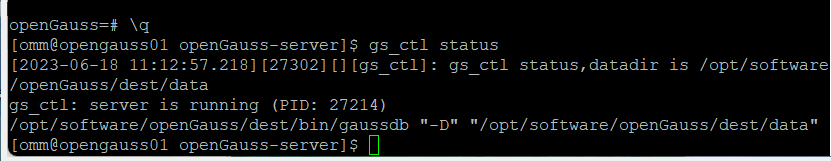
华为技术有限公司

# 关卡一、openGauss数据安装及基本操作

openGauss数据安装及基本操作, 作业提交任务如下：

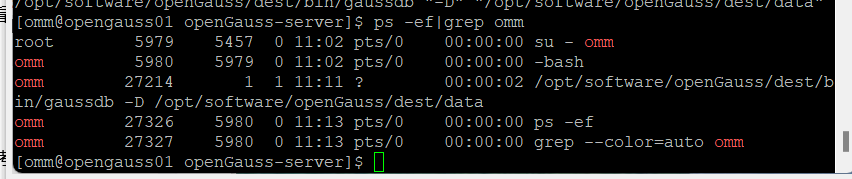
任务一：数据库状态验证

1. 查询数据库状态成功截图



任务二：数据库服务进程验证

1. 查看数据库服务进程截图（包含数据库服务器的主机名）



实验思考题：为什么需要通过源码编译，安装数据库？

为了使用某些数据库，需要将其源代码下载下来，并将其编译成可执行文件，然后

安装在本地的计算机上。因为有些数据库需要根据我们的特定计算机环境进行

编译，以使它们能够最好地利用我们的机器的资源。此外，编译和安装这些数据库还允

许我们选择特定的配置选项，以满足我们的具体需求，例如选择不同的存储引擎或启用

特定的功能。总之，通过源码编译，安装数据库可以确保我们能够轻松地定制和配置我

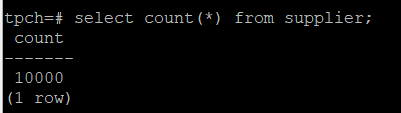
们的数据库，并实现最佳性能。

# 关卡二、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：

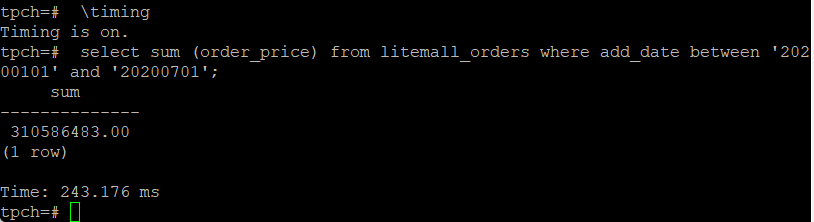
select count(\*) from supplier;;



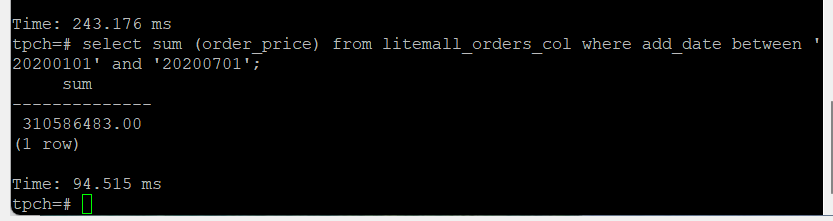
任务二：行存表与列存表执行效率对比

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';

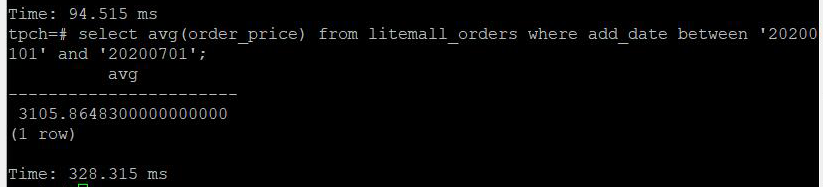


select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

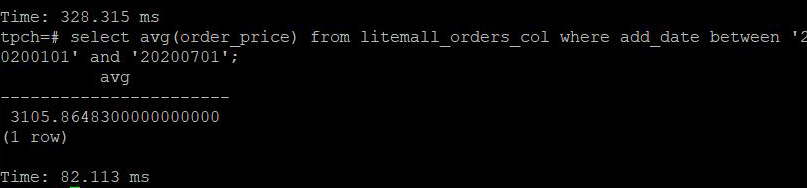


2. 2020年上半年litemall\_orders行存表与litemall\_orders\_col列存表中的order\_price的平均值查询，并对比执行效率截图

select avg (order\_price) from litemall\_orders where add\_date between '20200101' and '20200701';

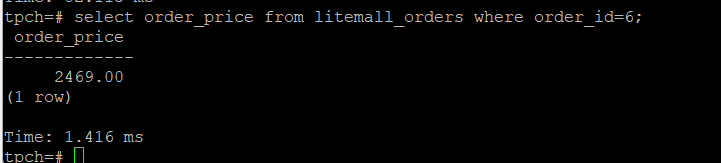


select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';

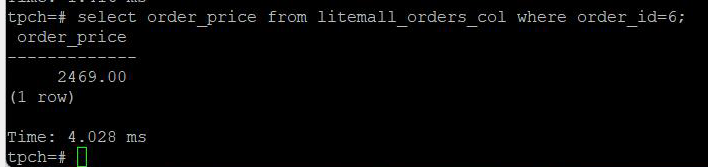


3. 查询litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price的值，并对比执行效率截图。

select order\_price from litemall\_orders where order\_id=6;

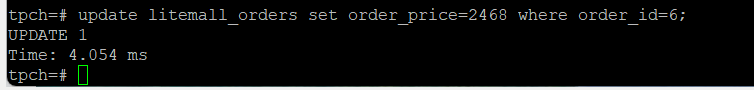


select order\_price from litemall\_orders\_col where order\_id=6;

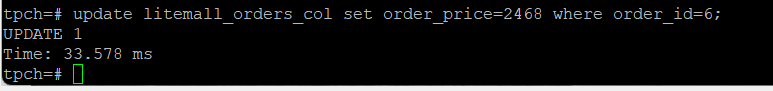


4. 将litemall\_orders行存表与litemall\_orders\_col列存表中order\_id为6的order\_price修改为2468，并对比执行效率截图。

update litemall\_orders set order\_price=2468 where order\_id=6;



update litemall\_orders\_col set order\_price=2468 where order\_id=6;



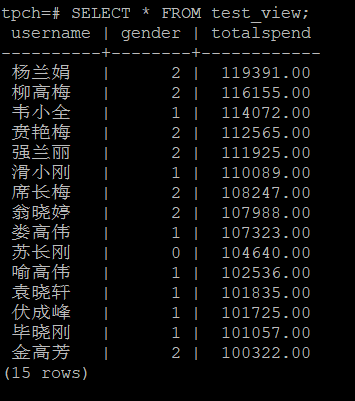
任务三：物化视图的使用

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;

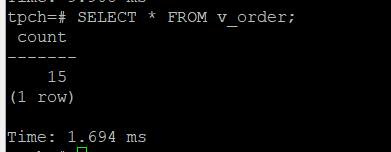
2. 使用物化视图统计人数，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



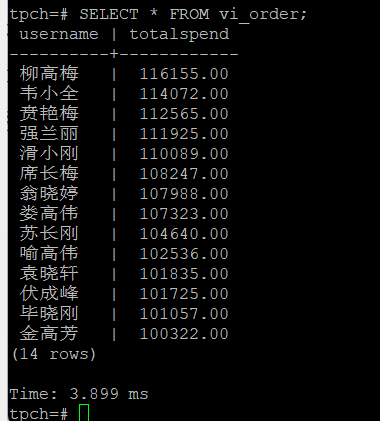
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



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;



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;



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？

行存表和列存表的执行效率取决于具体执行的 SQL 语句。由于行存表和列存表的数据结构和访问方式不同，它们在处理不同类型的查询时具有不同的性能优势。

行存表的存储结构和访问方式与传统的关系型数据库相同，数据按行存储，具有相同的列。行存表在处理事务型应用时比较高效，例如多次对相同行的修改、插入、删除等操作。

列存表的存储结构和访问方式不同于行存表，它将列按照顺序存储到不同的物理块中。列存表在处理大型数据集时比较高效，例如对大型表进行聚合操作（如 SUM 和 COUNT）以及数据分析型应用。这是因为列存表可以只读取需要的列，而不像行存表一样需要读取完整的行。

因此，在执行聚合操作、数据分析和 OLAP （On-Line Analytical Processing）查询时，列存表比较高效。而在执行事务型应用（如基于客户端的 OLTP）时，行存表比较高效。

实践思考题2：全量物化视图与增量物化视图有哪些差别？

物化视图是一种预先计算和存储的查询结果。全量物化视图和增量物化视图的区别在于它们是如何刷新和维护的：

1. 全量物化视图：全量物化视图是在创建时完全计算出来并存储在磁盘上的。当底层表中的数据发生变化时，全量物化视图需要完全重构，即重新执行计算并替换旧的物化视图。由于需要完全重构，因此全量物化视图刷新的代价相对较高。但是，查询时性能较好，因为数据已经预先计算和存储了。

2. 增量物化视图：对于增量物化视图，只有底层表中的变化部分被计算和更新到物化视图中，它们不需要完全重构。增量视图通常需要比全量视图更频繁地刷新，但是更新代价相对较小。因此，增量物化视图适用于需要频繁查询的基于 OLAP 的应用程序，并且这些应用程序的数据变化相对较频繁，但是单次刷新的代价需要尽可能地小。

总的来说，全量物化视图更适用于查询较频繁、数据变化不频繁的场景，而增量物化视图更适用于查询频繁、数据变化较频繁的场景。

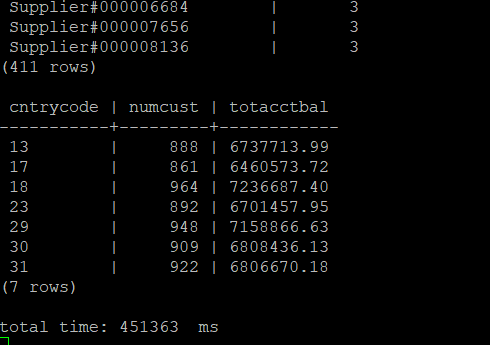
# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

1. 执行TPCH脚本，获得测试时间，将执行结果截图：

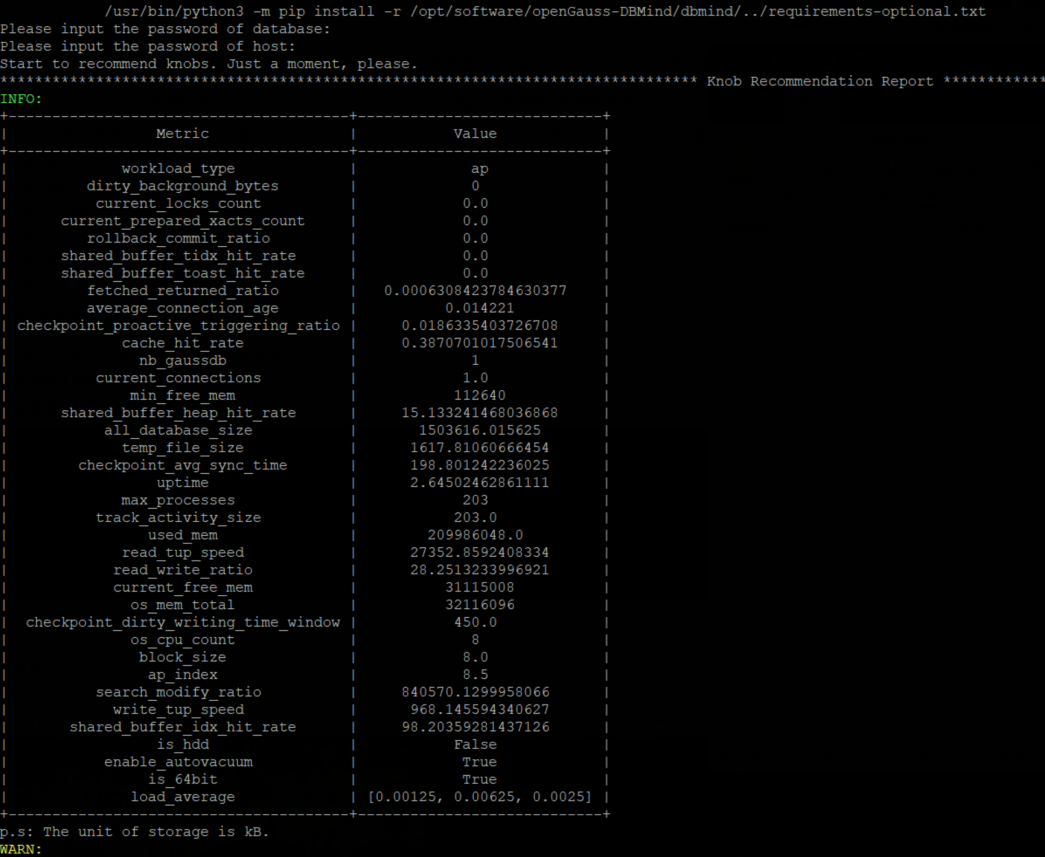
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries01.log

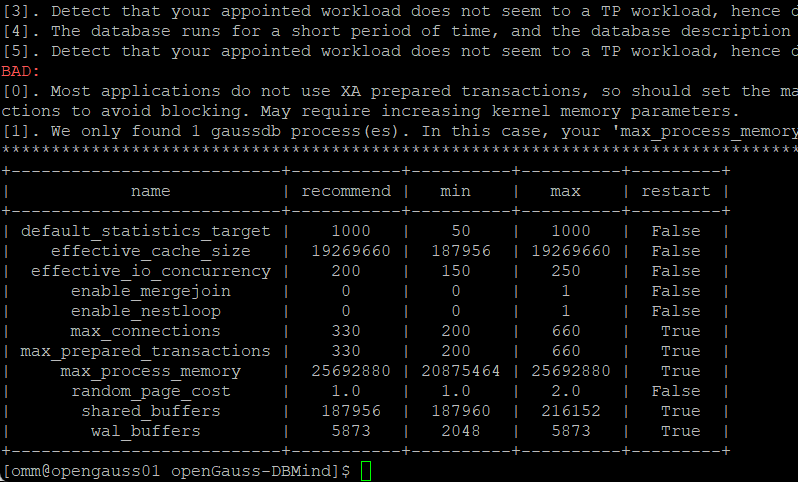




2. 使用root用户，执行X-Tuner进行参数建议优化，将执行结果截图

gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm

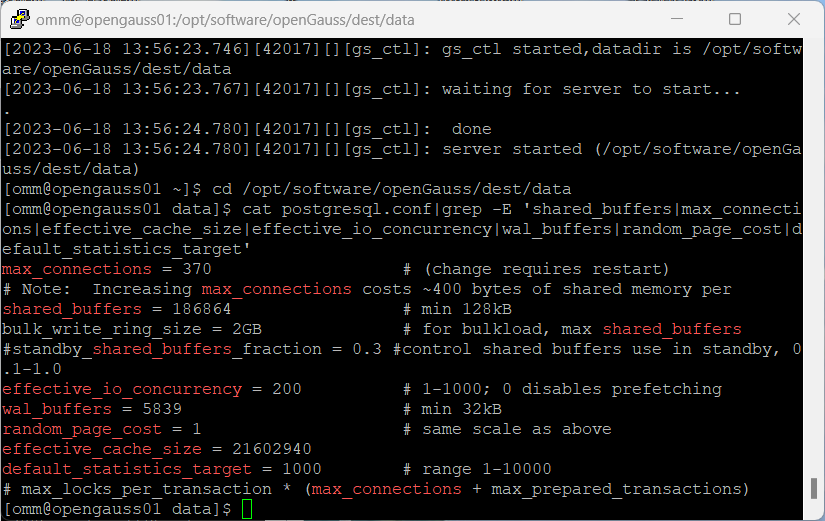




3.重启完成后，获取参数值：

cd /opt/software/openGauss/data

cat postgresql.conf|grep -E 'shared\_buffers|max\_connections|effective\_cache\_size|effective\_io\_concurrency|wal\_buffers|random\_page\_cost|default\_statistics\_target'



任务二：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

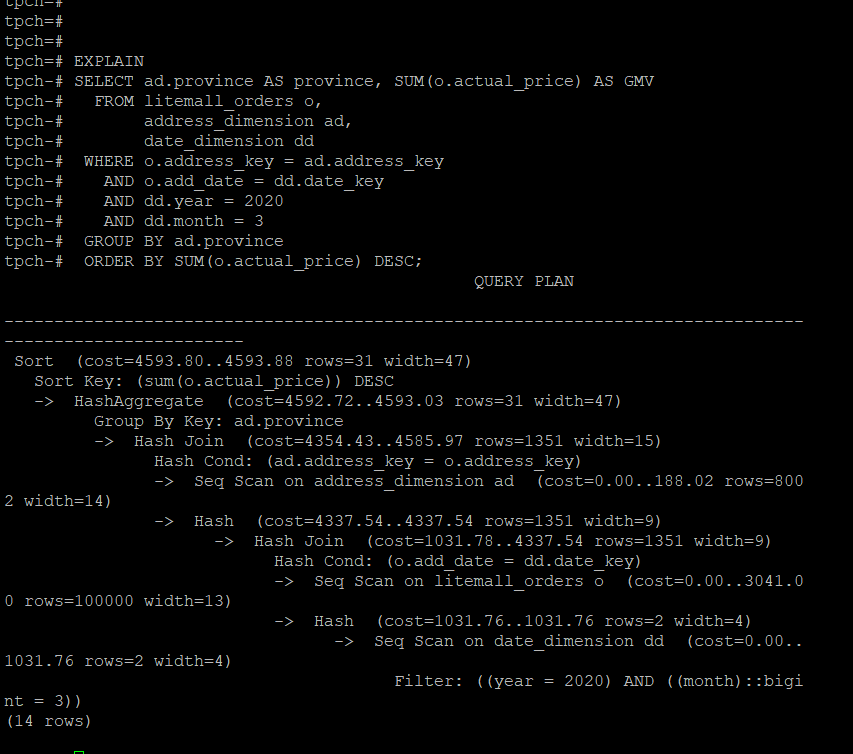
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



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

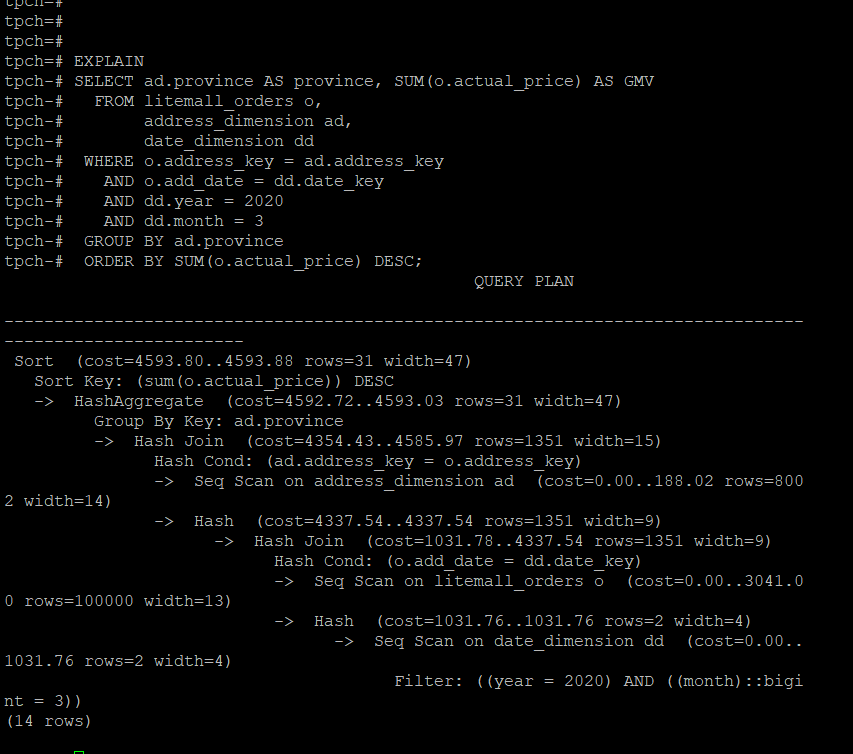
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

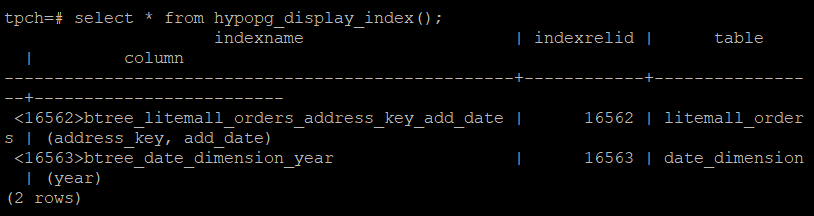
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC');



3. 查看创建的虚拟索引列，将执行结果截图。

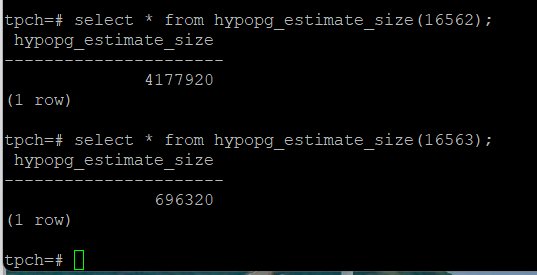
select \* from hypopg\_display\_index();



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);



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

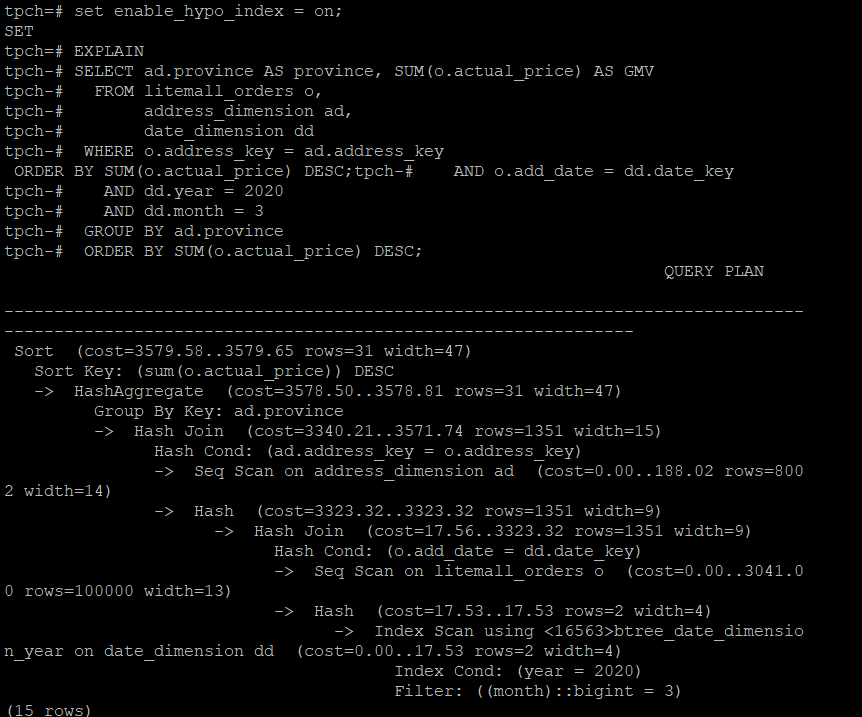
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

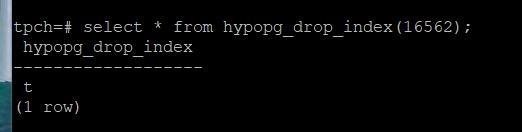
GROUP BY ad.province

ORDER BY SUM(o.actual\_price) DESC;



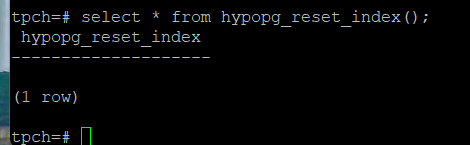
6. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_drop\_index(16715);



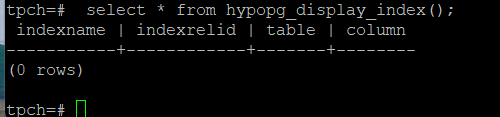
7. 删除某一个索引虚拟列，将执行结果截图。

select \* from hypopg\_reset\_index();



8. 查看索引虚拟列，将执行结果截图。

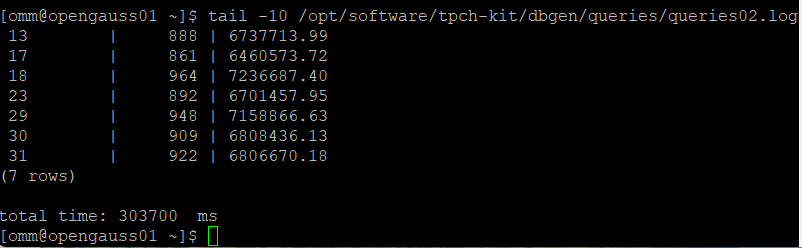
select \* from hypopg\_display\_index();



任务三：通过创建索引，对queries.sql中的SQL语句进行优化，并对比优化前后queries.sql执行的时间。

1. 重新执行queries.sql查询，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



挑战一：进一步优化queries.sql中的查询语句，使得前后执行时间出现倍数级的提升。

1. 重新执行queries.sql查询，将执行结果截图：

gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries03.log

不同情况下需要不同的优化策略和技巧。下面是一些实践中常用的 SQL 语句优化技巧，：

1. 避免 SELECT \* 操作：在大多数情况下，只选择需要的列，而不是选择整个表的所有列，可以提高查询执行的速度，并减少网络传输的数据量。2. 使用 WHERE 子句过滤数据：使用 WHERE 子句可以减少要检索的行的数量，从而提高查询效率。3. 优先考虑使用 INNER JOIN：在连接两个或多个表时，优先使用 INNER JOIN，因为 INNER JOIN 更常用，更高效。同时还可以考虑对 JOIN 表建立索引。4. 使用 EXISTS 和 NOT EXISTS 操作：存在子查询时，尽可能使用 EXISTS 或 NOT EXISTS，而避免使用 IN 或 NOT IN，因为前者速度更快。5. 对重复数据使用 GROUP BY 和 DISTINCT：在避免重复数据时，尽可能使用 GROUP BY 和 DISTINCT，而不是 ORDER BY 或 DISTINCT。6. 对 SELECT 语句进行优化：避免使用重复 SELECT，使用内置函数，避免使用 NULLS LAST 等，可以提高查询速度和效率。

总之，以上是SQL优化的几个常用技巧，具体的优化方法需要根据具体的查询语句和数据库模式进行调整和优化。

实践思考题1：根据X-Tuner给出的参数优化，在哪些参数上进行了优化，为何要对这些参数进行优化？

X-Tuner 是一种用于自动 SQL 参数优化的工具，可以使用机器学习和基于成本的优化器来优化 SQL 语句的执行计划。X-Tuner 在进行 SQL 优化时，会在以下几个方面进行参数优化：1. 查询语句的语法和结构：包括 SQL 查询语句的语法规范、查询语句的句法结构、查询中的过滤条件等。优化这些参数可以提高查询语句的执行效率。2. 数据库表的结构和索引：包括表结构设计、索引设计、行组织（ROW Format）和页组织（PAGE Format）等。优化这些参数可以提高查询的执行效率和响应时间。3. 数据库缓存和缓存机制：包括数据库缓存的大小、交换值的使用、缓存预热等。优化这些参数可以提高查询的响应时间和性能。4. SQL 优化器和执行计划：包括是否使用 Hint、执行计划的选择、SQL 优化器的配置等。优化这些参数可以提高查询的执行效率和响应时间。5. 数据库实例配置：包括服务器配置、线程配置、内存配置等。优化这些参数可以提高查询的执行效率和响应时间。

这些参数优化的目的在于提高 SQL 查询语句的执行效率、响应时间和性能，从而提高数据库系统的整体性能和可靠性。通过优化这些参数，可以使数据库系统在处理大量的数据和并发请求时，更加稳定和有效。

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？

索引是用于加速 SQL 查询的重要工具，它们可以提供快速访问表中的数据，以获得更好的查询性能和响应时间。使用索引可以减少扫描整个表的成本，并且可以在查询优化器中引入正确的执行计划。

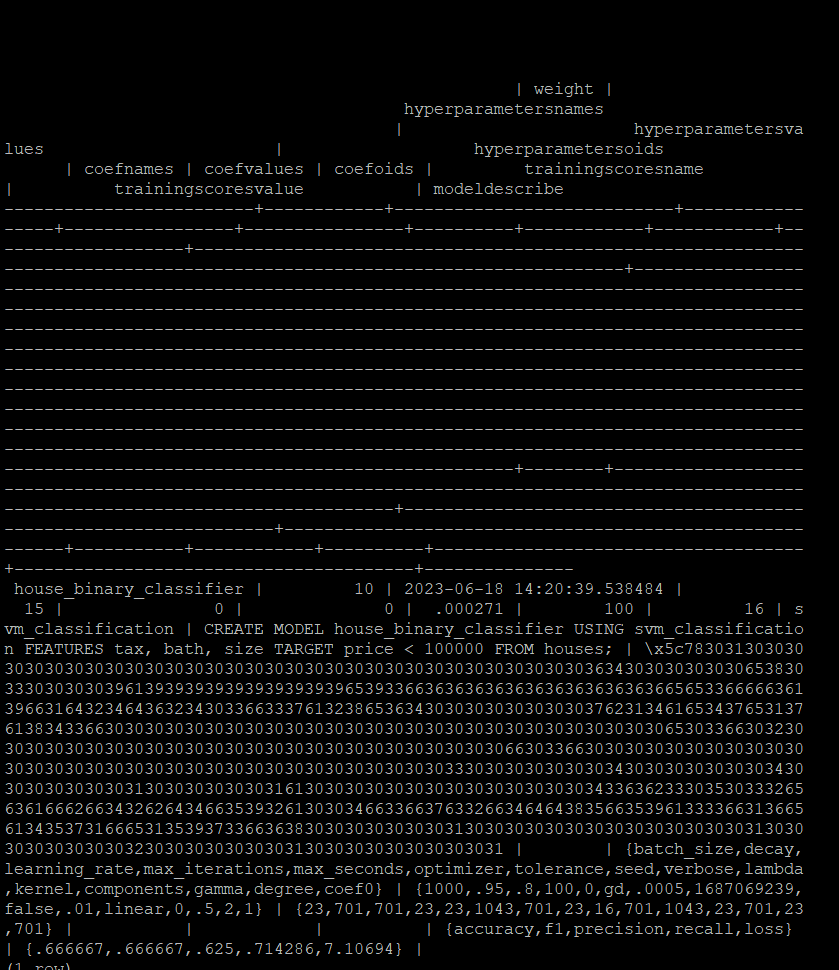
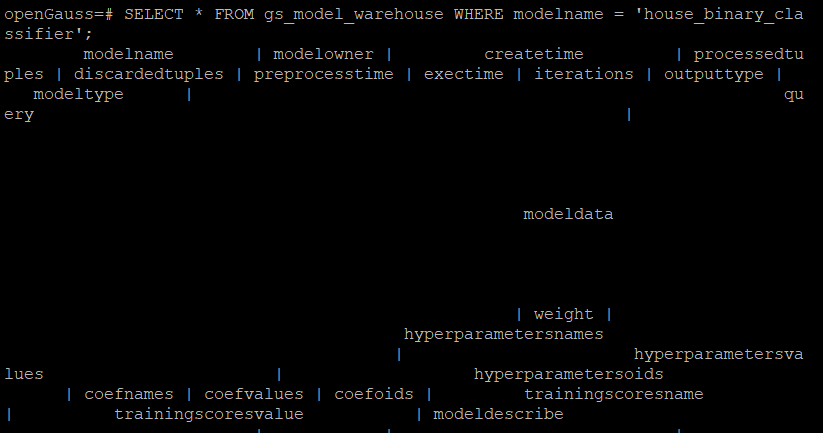
除了索引之外，数据库优化还可以从以下几个方面进行优化：1. 数据库范式设计：通过将数据分解成更小的单位并消除冗余，可以提高数据库的查询效率和响应速度。2. 把数据拆分成更小的库和表：通过在不同的数据存储层次中分隔数据，可以减少查询的联接，提高查询性能与响应时间。3. 配置合适的硬件和软件资源：包括数据库服务器、硬盘阵列、内存、网络等，可以提高数据库系统的整体性能。4. 正确配置数据库：包括保存空间、备份恢复、定期维护等，可以提高数据库的可用性和性能。5. 数据库性能监控和调优：通过监控数据库活动并及时识别和优化性能问题，可以提高数据库性能和可靠性。

总之，通过对数据库进行综合优化，可以使数据库在处理大量数据和并发请求时，更加高效、可靠和稳定，从而更好地支持业务应用程序的需求。

# 关卡四、openGauss的DB4AI特性应用

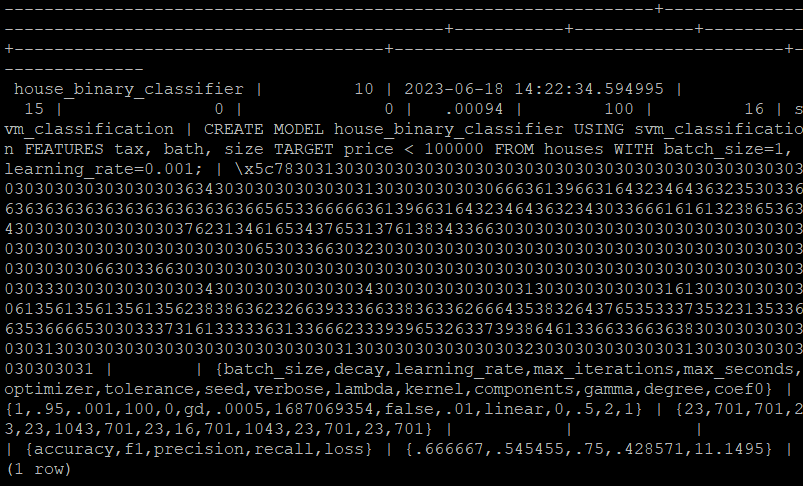
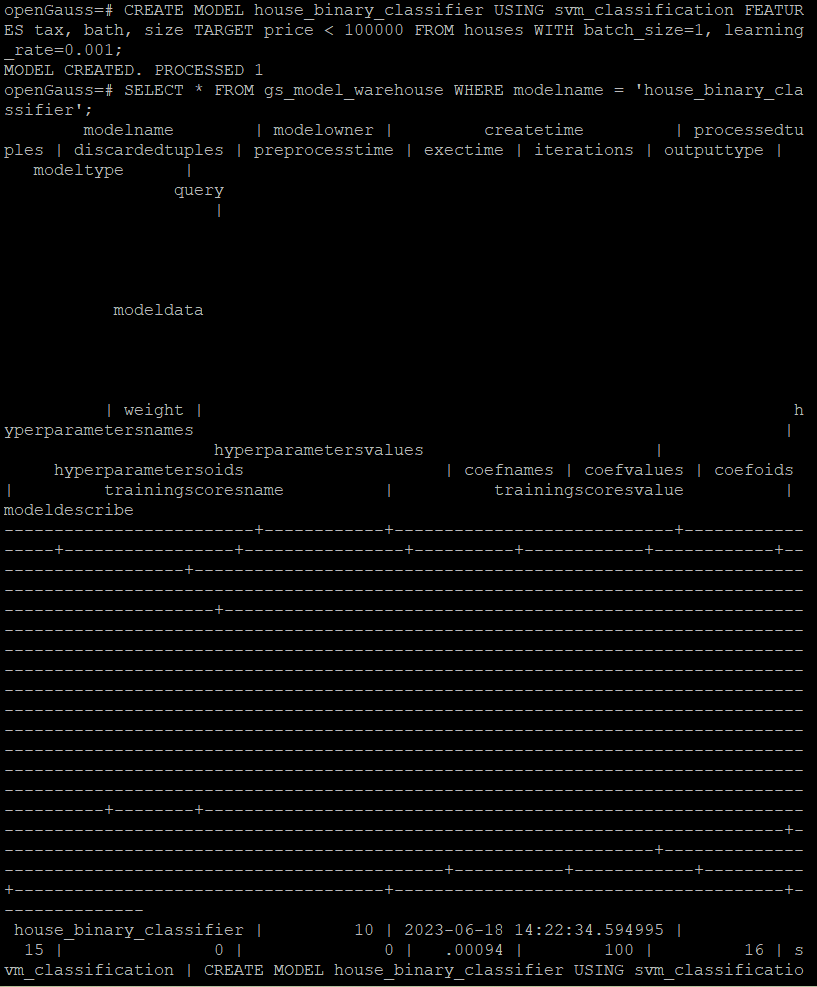
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



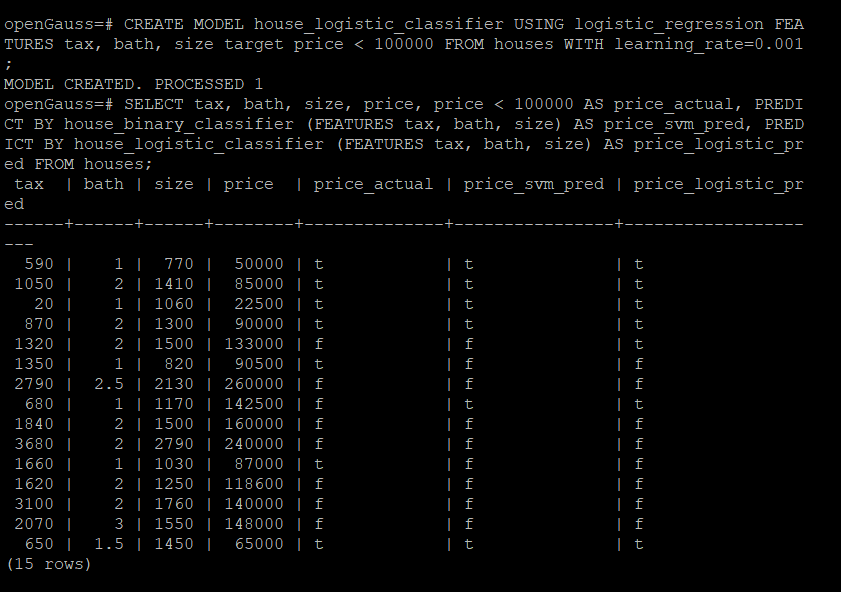
任务二：观察新模型的信息，将执行结果截图。

postgres=# SELECT \* FROM gs\_model\_warehouse WHERE modelname = 'house\_binary\_classifier';



任务三：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。

postgres=# 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;



实践思考题1：分类模型与回归模型有何不同？

实践思考题2：什么是SVM算法？

实践思考题3：分类问题有哪些评价指标，请分别说明他们的含义？

实践思考题4：回归问题有哪些评价指标，请分别说明他们的含义？