openGauss AI特性创新实践课



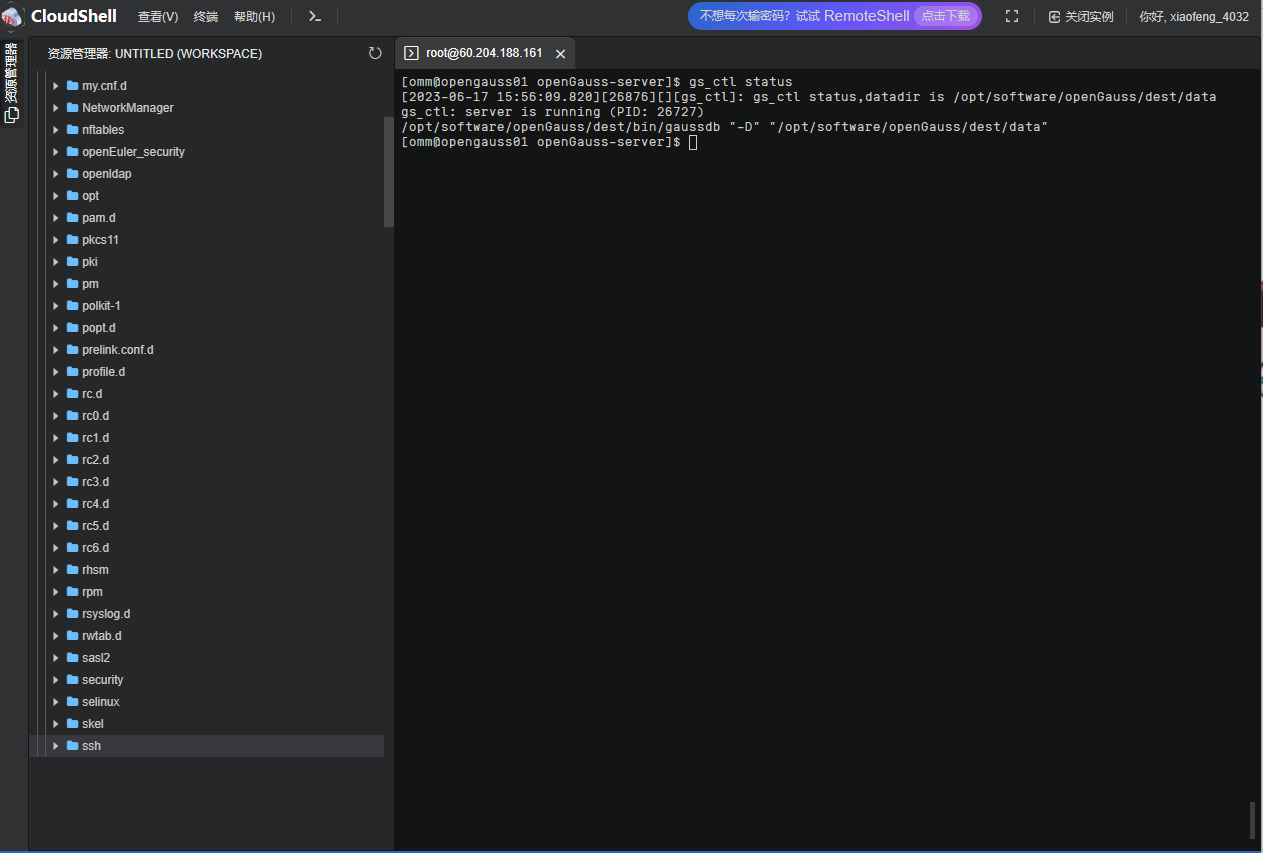
华为技术有限公司

# 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：

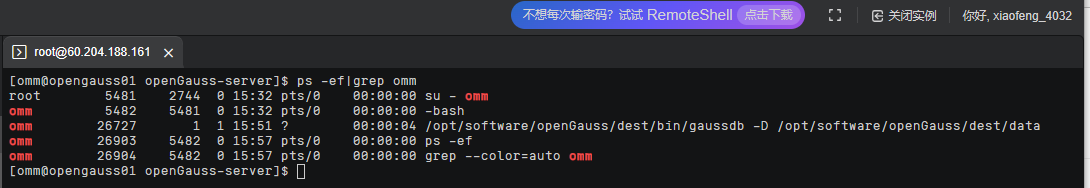
任务一：数据库状态验证

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？

Opengauss支持多种架构和操作系统，通过源码编译可以满足不同运行平台的需求。Opengauss存在多种可选特性，源码编译可以方便用户定制，满足不同的需求。源码编译可以根据本地CPU环境更快速的生成可执行文件。，

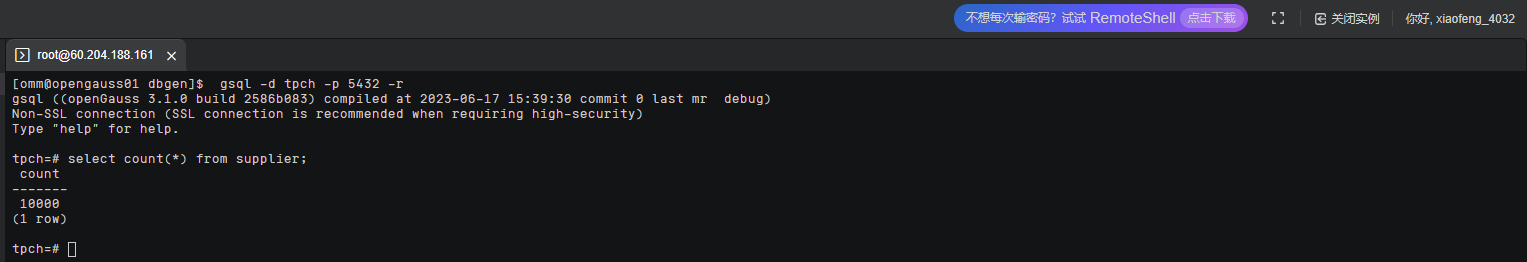
安装数据库大致需要，下载第三方编译库、下载openGauss源码、安装依赖包、编译和安装五个步骤

# 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：

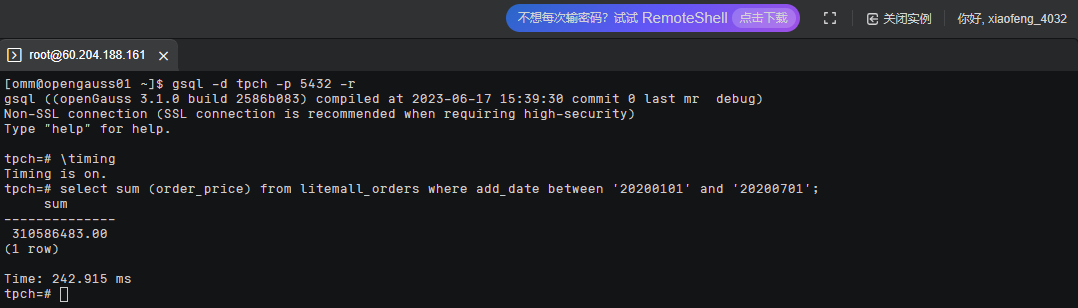
select count(\*) from supplier;;



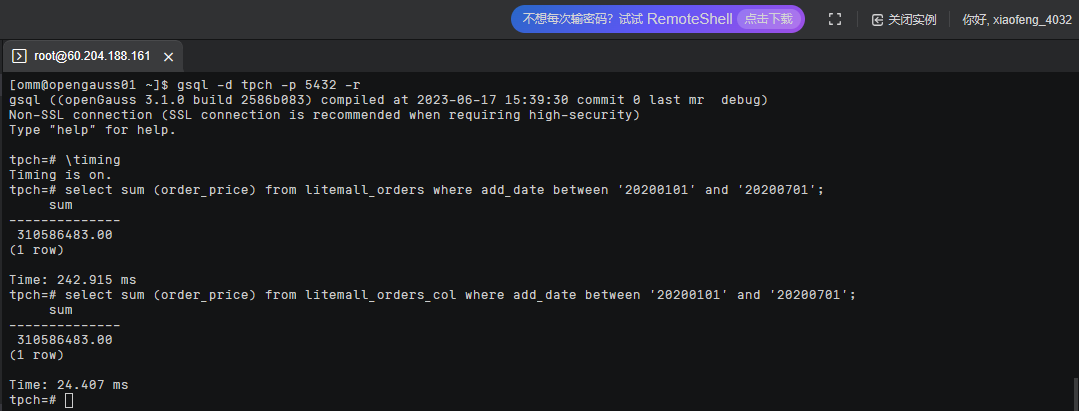
任务二：行存表与列存表执行效率对比

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';



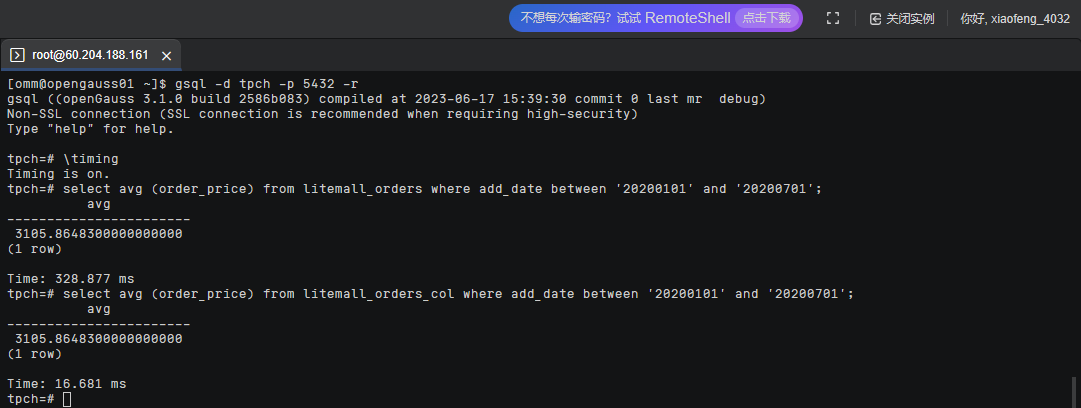
select sum (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';



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';

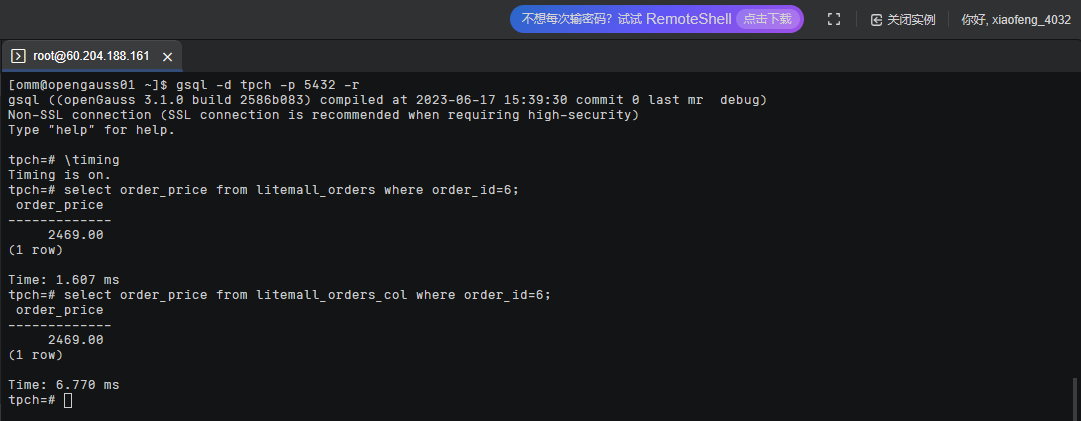
select avg (order\_price) from litemall\_orders\_col where add\_date between '20200101' and '20200701';



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;

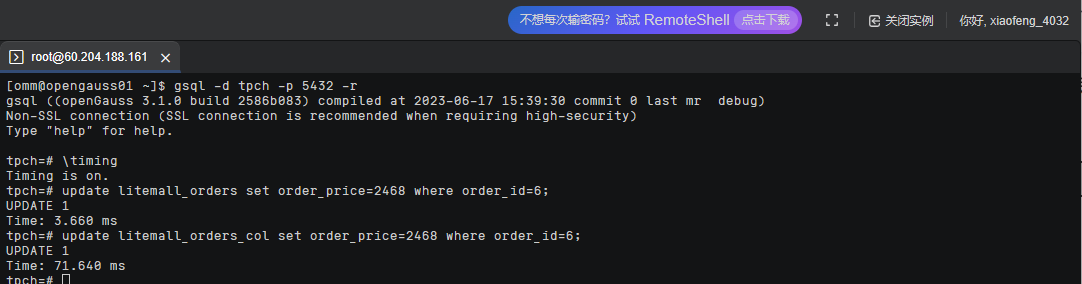
select order\_price from litemall\_orders\_col where order\_id=6;



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;

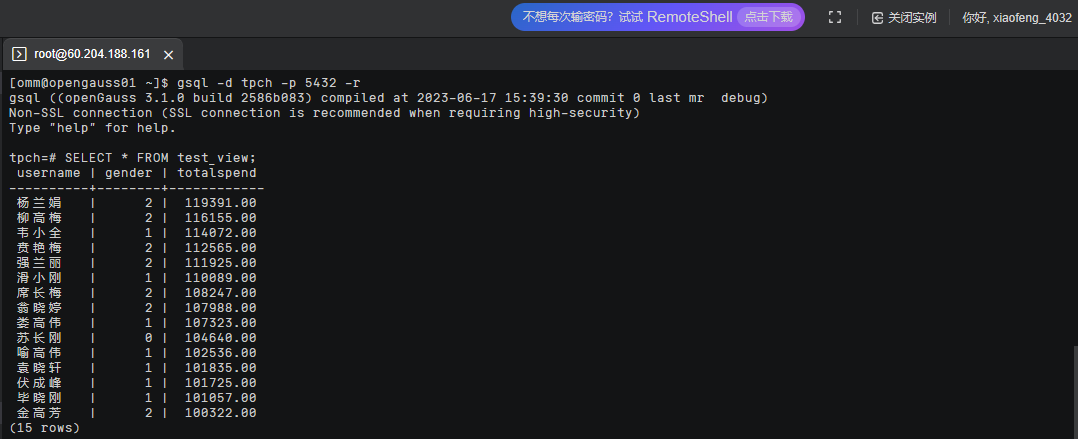
update litemall\_orders\_col set order\_price=2468 where order\_id=6;



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;



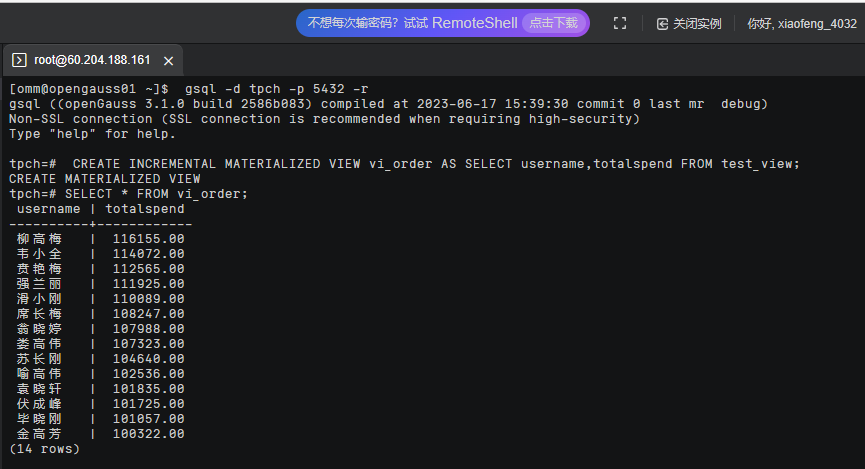
3. 对表进行操作后，刷新物化视图，查询物化视图结果，将执行结果截图。

SELECT \* FROM v\_order;



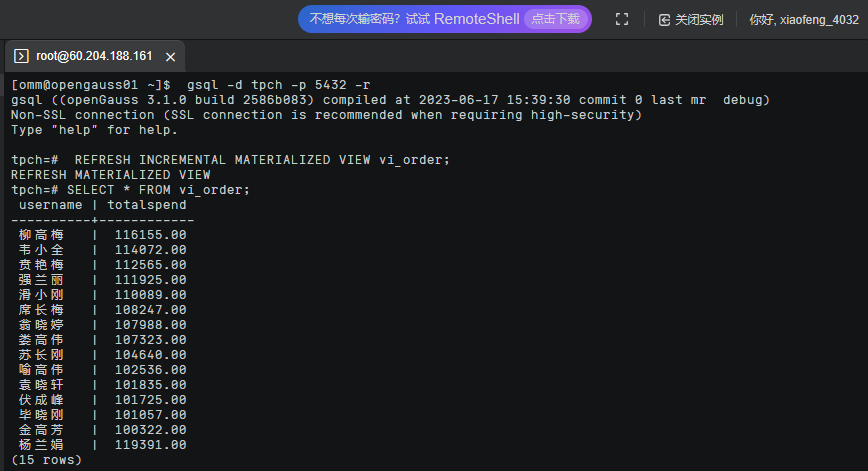
4. 创建增量物化视图，查询物化视图结果，将执行结果截图。

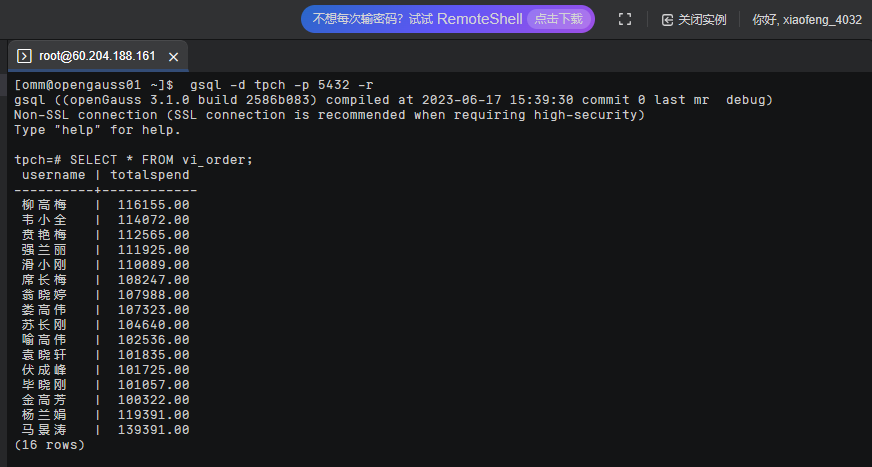
SELECT \* FROM vi\_order;



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？

行存表是将一行数据存储在一起，而列存表是将一列数据存储在一起。不同的数据布局会影响数据在磁盘上的读写效率和压缩率。

行存表在进行简单查询和进行数据修改操作时占有优势，列存表在数据的复杂查询中占有优势。可以得出结论，对于小规模，需要经常修改的数据，如个人信息等数据，使用行存表更好。对于大规模且不需要经常修改的数据，如气象数据，使用列存储更好

实践思考题2：全量物化视图与增量物化视图有哪些差别？

全量物化视图是每次刷新时，都会清空物化视图的数据，然后重新执行查询语句，将结果集全部插入到物化视图中。全量物化视图支持全部查询语句，可以用于复杂查询，且刷新代价较高。

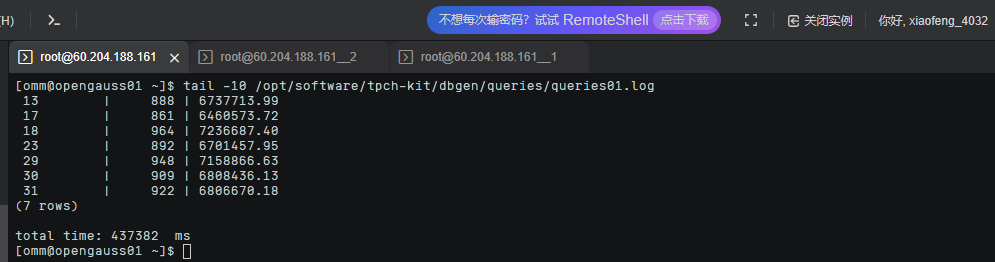
增量数据视图是指每次刷新时，只会获取基表的增量修改，然后计算并应用到物化视图上。增量数据视图只支持基表查询语句或UNION ALL语句，且数据刷新代价较低。

# 关卡三、openGauss的AI4DB特性应用

任务一：使用X-Tuner进行参数优化

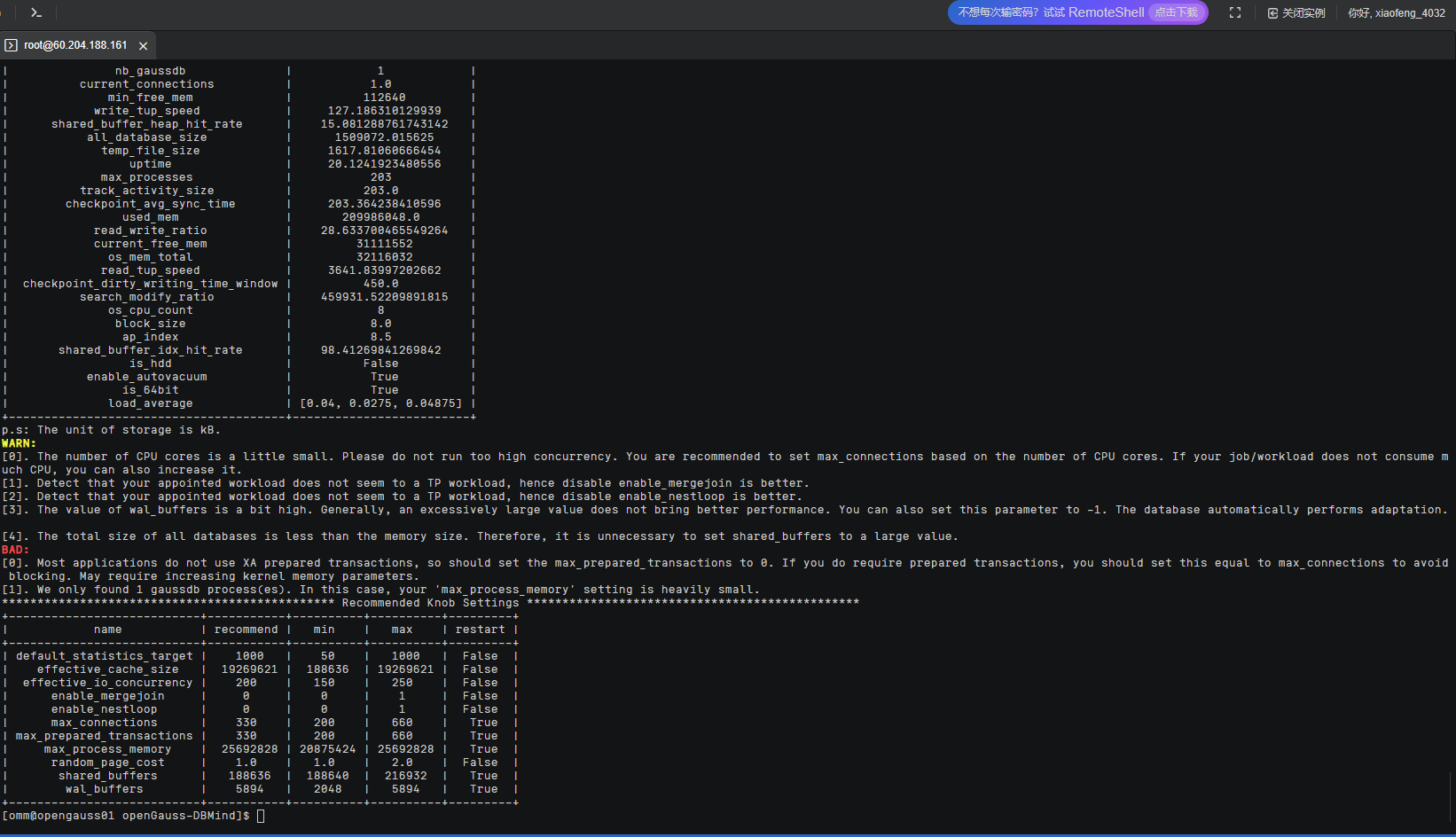
1. 执行TPCH脚本，获得测试时间，将执行结果截图：

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

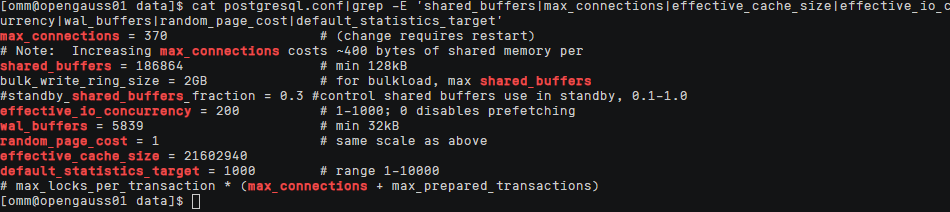
gs\_xtuner recommend --db-name tpch --db-user omm --port 5432 --host 127.0.0.1 --host-user omm



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

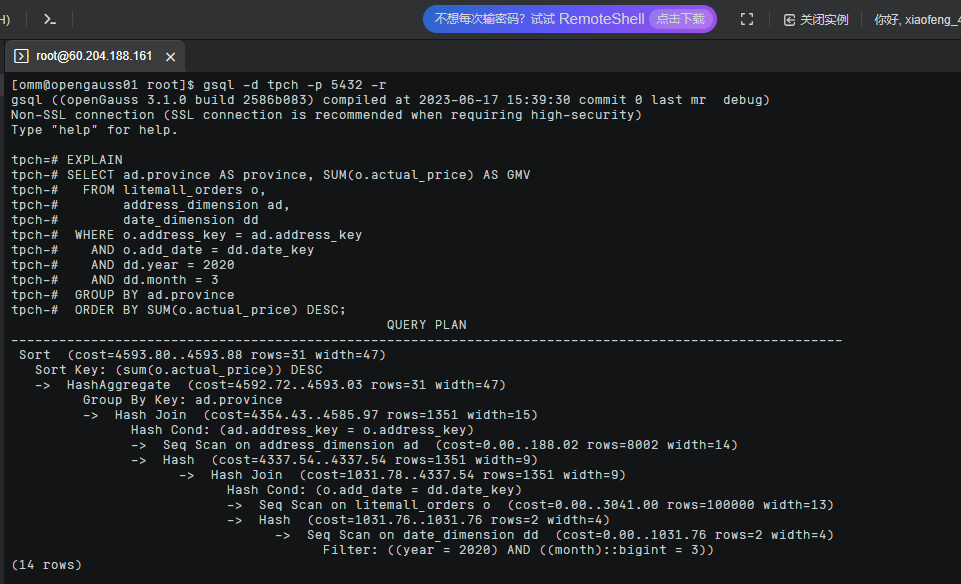
AND o.add\_date = dd.date\_key

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

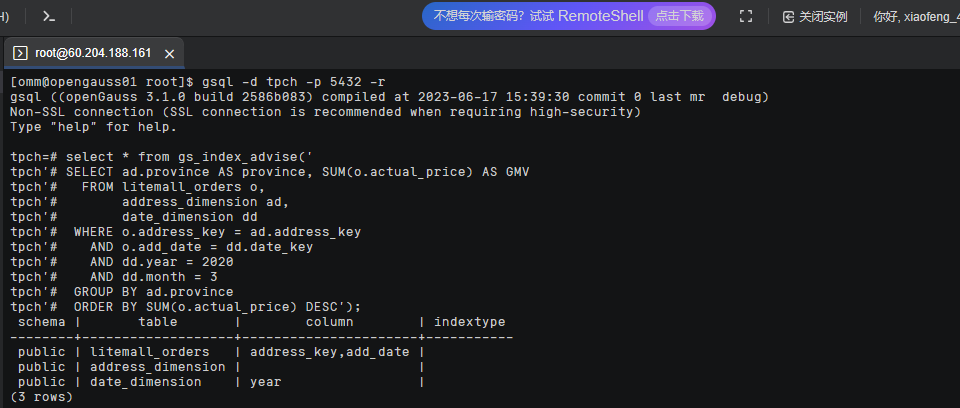
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

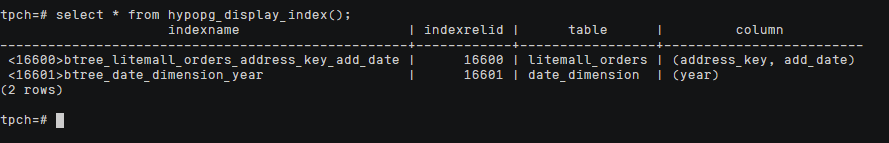
GROUP BY ad.province

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。

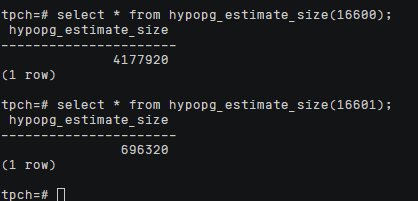
select \* from hypopg\_display\_index();



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

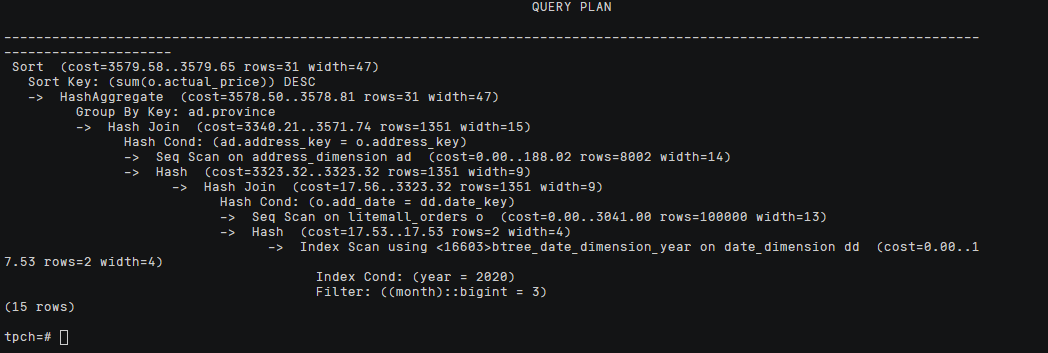
AND o.add\_date = dd.date\_key

AND dd.year = 2020

AND dd.month = 3

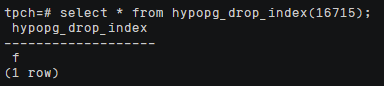
GROUP BY ad.province

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



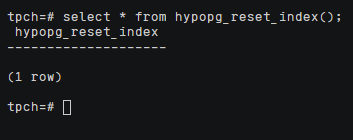
6. 删除某一个索引虚拟列，将执行结果截图。

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



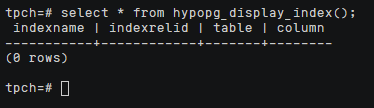
7. 删除某一个索引虚拟列，将执行结果截图。

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



8. 查看索引虚拟列，将执行结果截图。

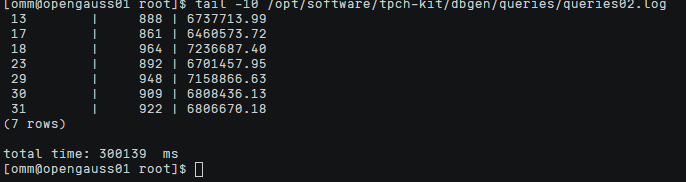
select \* from hypopg\_display\_index();



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：

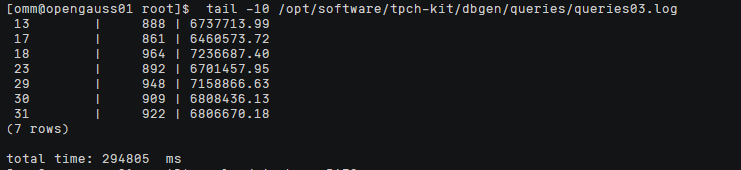
gsql -d tpch -p 5432 -r -f /opt/software/tpch-kit/dbgen/queries/queries.sql > /opt/software/tpch-kit/dbgen/queries/queries02.log



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



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？

在max\_connections和effective\_cache\_size进行了优化。修改max\_connections可以提高数据库的并发能力，更加充分的利用CPU的计算资源。增加effective\_cache\_size参数可以让优化器更倾向于使用索引扫描，可以减少I/O开销

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？

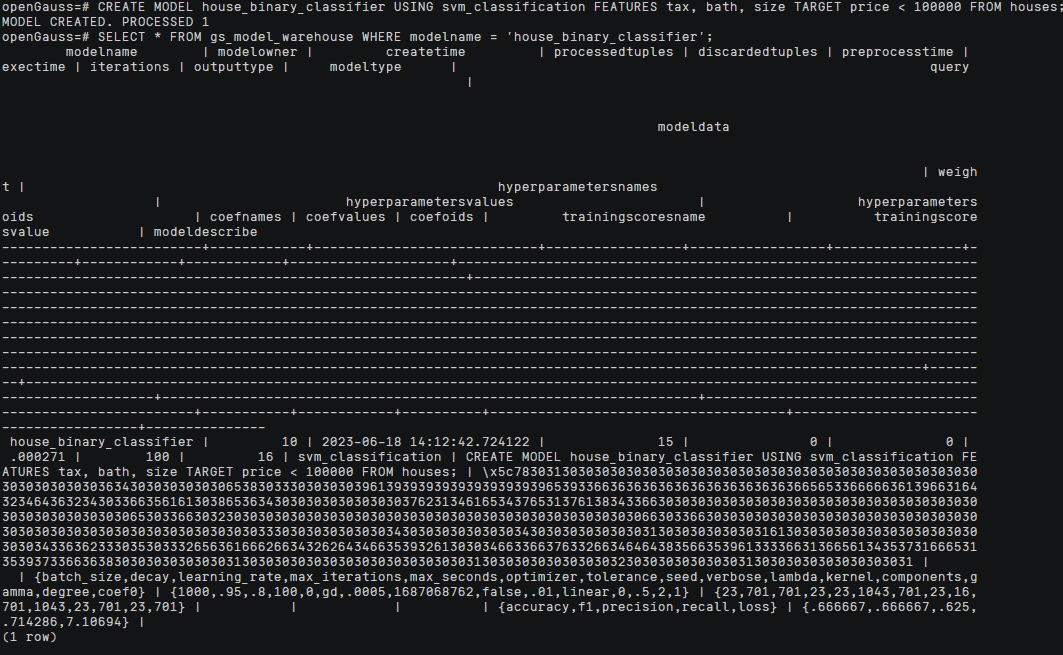
索引的使用可以加快数据的检索速度，减少I/O开销。可以加速表和表之间的连接，提高查询效率。可以在使用分组和排序子句进行数据检索时，利用检索排序，减少排序时间。

除了使用索引和优化参数外，还可以优化sql语句，避免使用子查询、全表扫描等低效操作。

# 关卡四、openGauss的DB4AI特性应用

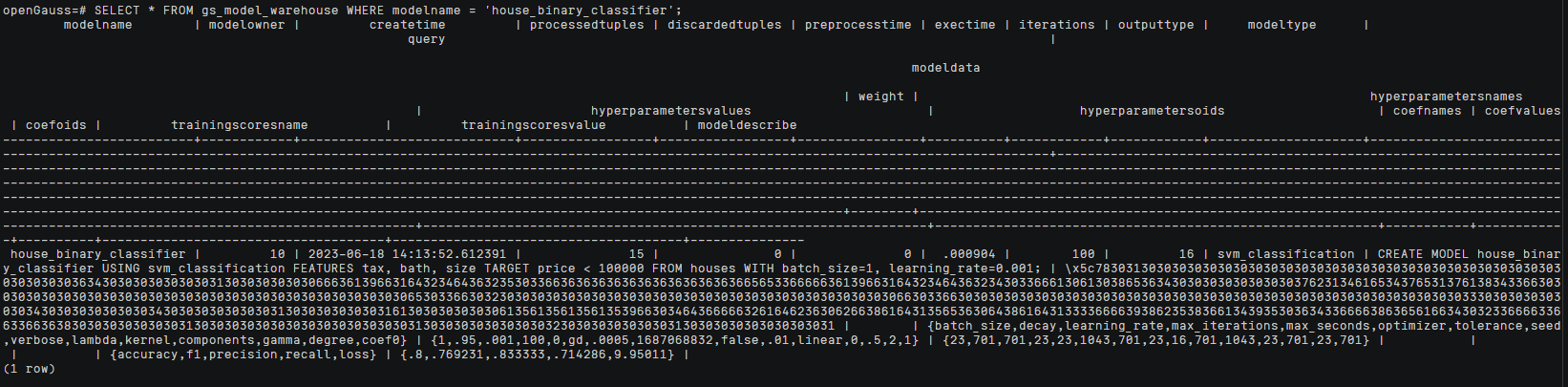
任务一：在gs\_model\_warehouse系统表中查看训练后的模型信息，将执行结果截图：

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



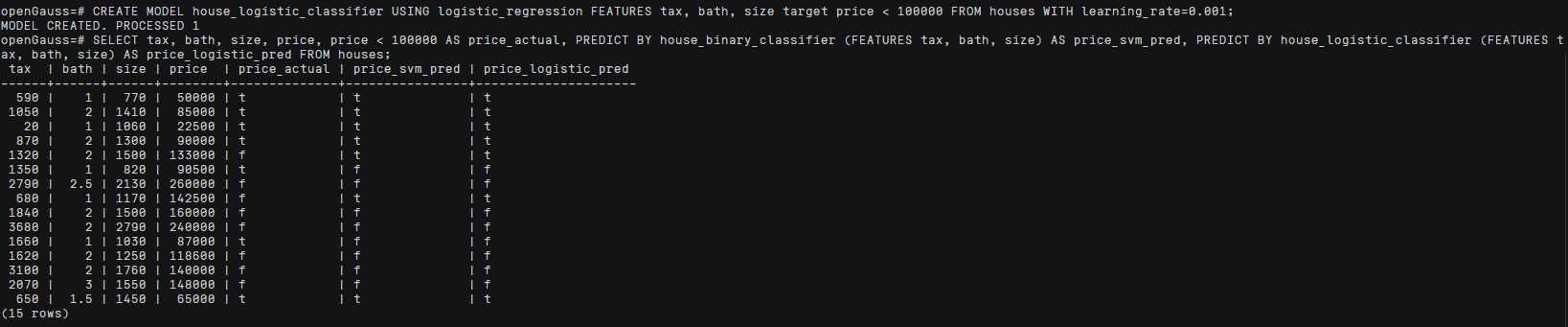
任务二：观察新模型的信息，将执行结果截图。

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？

1. 输出类型不同，分类模型的输出是离散值，而回归模型的输出一般是连续值。
2. 目的不同，分类模型的目的是寻找决策边界，将数据划分到不同的类别中；回归模型的目的是找到最优拟合，使得输出值与真实值之间的误差最小
3. 分类模型和回归模型的根本区别在于输出空间是否为一个度量空间。回归模型的输出空间是一个度量空间，可以定义误差函数来衡量预测值和真实值的距离。分类模型只有正确与否之分没有误差大小

实践思考题2：什么是SVM算法？

SVM算法是一种分类算法，其基本思想是将数据映射到高维空间中，并在该空间中找到一个超平面，使得各类数据点到该超平面的距离最大

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？

混淆矩阵：是一个表格，用来显示分类器对每个类别的预测结果和真实结果的对比，可以计算出其他指标的基础数据。

准确率：是预测正确的样本数量占总样本数量的比例，是一个整体的评价指标，不能反映不同类别的分类效果。

精确率：是预测为正类的样本中真正是正类的比例，是一个针对预测结果的评价指标，反映了分类器的查准能力。

召回率：是真正的正类样本中被预测为正类的比例，是一个针对原始样本的评价指标，反映了分类器的查全能力。

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？

均方误差（MSE，Mean Squared Error）：是预测值和真实值之差的平方的均值，反映了预测误差的大小，越小越好。

均方根误差（RMSE，Root Mean Squared Error）：是预测值和真实值之差的平方的均值的平方根，反映了预测误差的标准差，越小越好。

平均绝对误差（MAE，Mean Absolute Error）：是预测值和真实值之差的绝对值的均值，反映了预测误差的平均程度，越小越好。