**《数据库系统原理》实验报告**

### **实验题目：**实验16 数据库监视与性能优化实验

**姓名：**

**实验日期：**

**实验内容及完成情况：**（可续页）

### 实验16 数据库监视与性能优化实验

**实验学时：1学时**

**实验类型：验证**

**实验要求： 选做**

**一、实验目的**

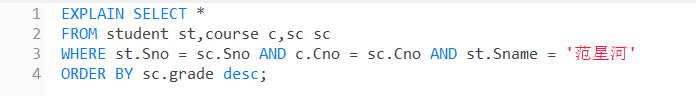
本实验的目的是使学生理解和掌握数据库性能调优的基本原理和方法，了解所使用的DBMS提供的数据库性能监视功能，学习数据库查询性能监视的基本原理和方法，了解数据库系统级参数和连接级参数的配置和调优的基本原理和方法。了解用户可以通过修改这些参数设置来调整系统运行的配置，以优化系统性能。

**二、实验内容**

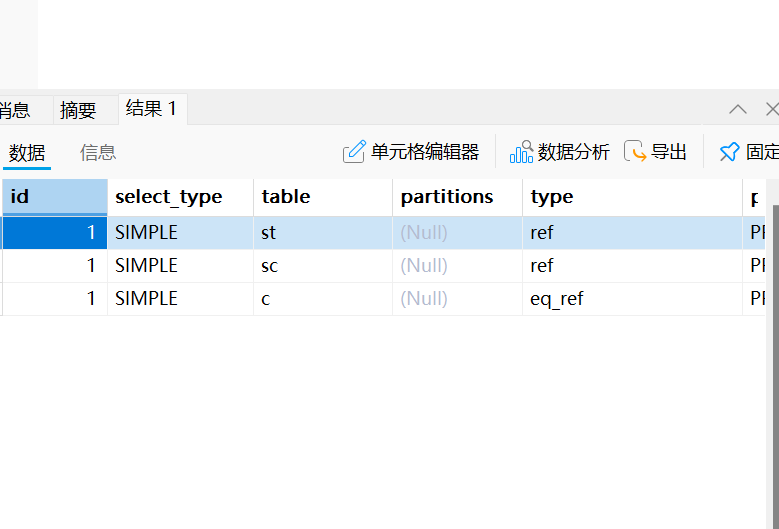
1.数据库查询性能调优实验：学会使用EXPLAIN命令分析查询执行计划、利用索引优化查询性能、优化SQL语句，以及理解和掌握数据库模式规范化设计对查询性能的影响。能针对给定的数据库模式，设计不同的示例验证查询性能优化效果。

（1）使用EXPLAIN命令查看查询执行计划

查看student、course、sc三个表连接查询的查询执行计划。



该SQL语句时要查询学生名字为’范星河’的学生相关信息、选修课程相关信息和课程成绩，并按照课程成绩（降序），排序输出结果。



EXPLAIN

SELECT

student.sno,

student.sname,

course.cname,

sc.grade

FROM

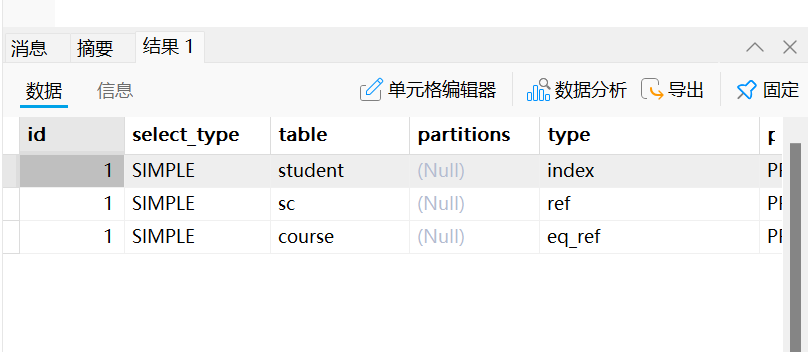
student

JOIN

sc ON student.sno = sc.sno

JOIN

course ON sc.cno = course.cno;



1. 利用索引优化查询性能

利用索引实验创建的索引进行SQL查询，优化SQL查询性能。比较在student表的Sname上有索引和无索引时，两种执行计划有何异同，并实际执行该查询，验证有索引和无索引时此查询语句的执行性能。

1. 初始查询语句

SELECT \*

FROM student

WHERE sname = '范星河';



查看无索引时的查询执行计划:

EXPLAIN

SELECT \*

FROM student

WHERE sname = '范星河';



-- 无索引查询

SELECT \*

FROM student

WHERE sname = '范星河';

-- 创建索引后查询

CREATE INDEX idx\_sname ON student(sname);

SELECT \*

FROM student

WHERE sname = '范星河';



（3）优化SQL语句

①IN与EXISTS查询：一般地使用EXISTS查询效率要高于IN查询。分别利用IN和EXISTS进行SQL查询，比较两种执行计划，并实际测试执行性能哪种情况好。

使用 IN 查询

EXPLAIN

SELECT \*

FROM student

WHERE sno IN (

SELECT sno

FROM sc

WHERE institute = '软件学院'

);



使用 EXISTS 查询

EXPLAIN

SELECT \*

FROM student s

WHERE EXISTS (

SELECT 1

FROM sc

WHERE sc.sno = s.sno

AND institute ='软件学院'

);



结论：

· IN 更适合子查询返回较少结果的场景。

· EXISTS 更适合子查询结果较大或涉及复杂逻辑的场景，性能通常优于 IN。

②尽可能使用不相关子查询，避免使用相关子查询。不相关子查询一般比相关子查询执行效率高，在可能的情况下，改写相关子查询为不相关子查询。比较两种执行计划，并实际测试执行性能哪种情况好。

相关子查询：

SELECT sname

FROM student

WHERE sno IN (

SELECT sno

FROM sc

WHERE grade > (

SELECT AVG(grade)

FROM sc

)

);



不相关子查询：

SET @avg\_grade = (SELECT AVG(grade) FROM sc);

SELECT sname

FROM student

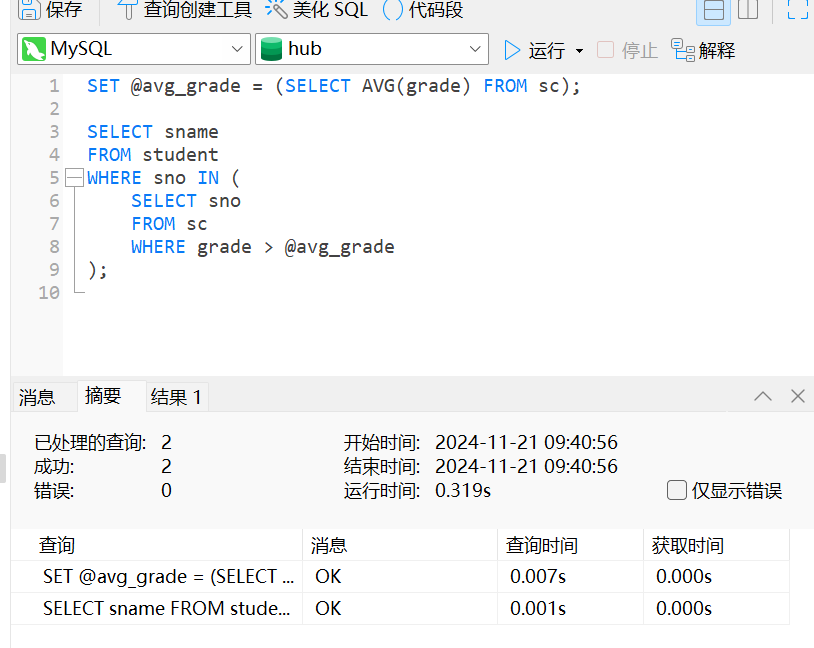
WHERE sno IN (

SELECT sno

FROM sc

WHERE grade > @avg\_grade

);



· **不相关子查询** 性能明显优于相关子查询，特别是在处理大数据量时。

· 通过优化，将相关子查询改写为不相关子查询，可以显著提高查询效率。

· 该实验验证了提前计算常用结果（如 @avg\_grade）是一种有效的性能优化手段。

（4）数据库模式规范化对查询性能的影响

分析该数据库模式中是否存在不规范的设计。该设计在海量数据的情况下查询效率咋样？如何在设计上进一步提高海量数据的查询效率？

第三范式在一定程度上减少了不必要的冗余，提高了数据库的查询效率，但是如果数据量大且需要大量联合查询的时候，第三范式设计又可能会影响查询效率。

2.（了解）数据库性能监视实验：使用MySQL的数据库性能监视工具，通过标准统计视图和统计访问函数查看数据库系统收集到的性能统计信息、ANALYZE更新数据库统计信息，通过专门工具监视系统性能。希望能够熟悉数据库系统有关性能统计信息的标准视图和统计访问函数，了解如何通过系统收集到的性能数据监视系统性能。

3.（了解）数据库系统配置参数调优实验：熟悉和了解数据库各级参数的作用以及配置，包括系统级参数配置和调优、数据库级参数配置和调优、会话（连接）级参数配置和调优。

**三、课后习题**

1.对表数据的高级查询实验中的查询，使用不同的SQL语句来表达，比较它们的查询效率，体会并总结查询优化的技巧和方法。

使用 IN 查询：

SELECT sname

FROM student

WHERE sno IN (

SELECT sno

FROM sc

WHERE cno = 1

);



使用 EXISTS 查询：

SELECT sname

FROM student s

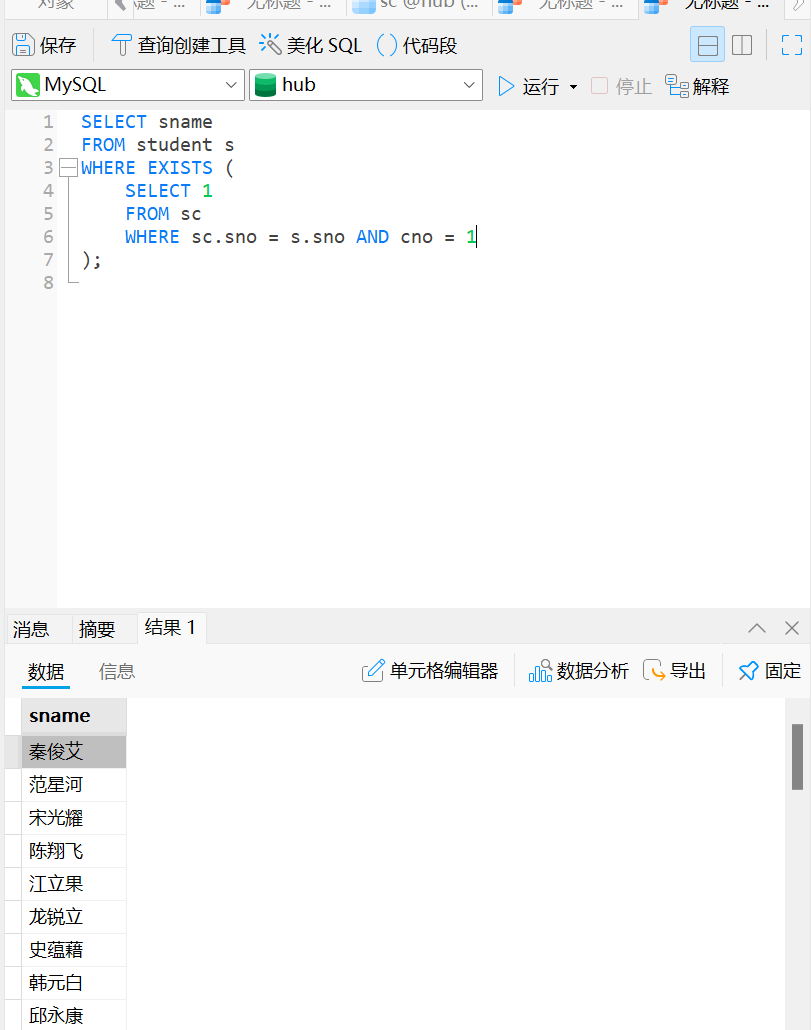
WHERE EXISTS (

SELECT 1

FROM sc

WHERE sc.sno = s.sno AND cno = 1

);



使用内连接：

SELECT s.sname

FROM student s

JOIN sc

ON s.sno = sc.sno

WHERE sc.cno = 1;



使用半连接（WITH 子句）：

WITH Temp AS (

SELECT sno

FROM sc

WHERE cno = 1

)

SELECT s.sname

FROM student s

JOIN Temp

ON s.sno = Temp.sno;



查询效率比较

#### ****性能对比****

1. IN **查询：** 当子查询返回的数据量较大时，性能下降显著。
2. EXISTS **查询：** 性能优于 IN，因为子查询匹配到第一个结果后会停止。
3. **内连接：** 在有适当索引时，通常是性能最优的选择。
4. **半连接：** 对于复杂查询，WITH 子句能简化逻辑，但性能略低于内连接。

查询优化技巧和总结

#### ****优化技巧****

· **使用索引：**

* 对查询条件中的字段（如 sno 和 cno）创建索引。
* 避免在索引列上使用函数或表达式（如 WHERE YEAR(date\_col) = 2024）。

· **选择高效的查询方式：**

* 优先使用内连接或 EXISTS。
* 避免 IN 查询，当子查询结果集较大时性能较差。

**简化子查询：**

* 使用 WITH 子句将复杂的子查询拆分为临时表，便于调试和优化。

**减少数据扫描：**

* 通过索引、条件过滤（WHERE）或限制查询范围（LIMIT）减少扫描行数。

**实验总结：**

通过本次实验，深入理解并掌握了数据库性能调优的基本原理和方法，体会到优化数据库性能的重要性及实际应用价值。总结如下：

**查询性能优化**

* 通过实验比较了不同的查询写法（如 IN、EXISTS、JOIN 等）的性能差异，验证了索引的有效使用对查询效率的显著提升。
* 学会使用数据库的查询性能监视工具（如 EXPLAIN）分析执行计划，识别性能瓶颈并优化查询。

**参数调优**

* 了解了数据库系统级参数（如缓存大小、连接池配置）和连接级参数（如事务隔离级别、超时设置）的作用。
* 掌握了通过修改参数配置优化系统性能的方法，并认识到不同场景下参数调整的灵活性和重要性。

**性能监视**

* 学习了数据库性能监视的基本原理，能够通过工具（如慢查询日志、性能视图等）监控和分析系统的运行状态，为后续的优化提供依据。

**实践意义**

* 实验强化了对索引、查询优化、参数配置的理解，并认识到数据库性能调优是一个持续的、数据驱动的过程，需要根据实际应用场景做出针对性优化。

通过本次实验，具备了基础的数据库性能调优能力，为提升数据库运行效率和系统稳定性打下了良好的基础。

**教师评语及成绩**：