**《数据库原理及应用》实验报告**

题目: 视图与索引

院系：计算机科学与工程学院

班级：210402

姓名：谢鸿

学号：20212345

**实验报告（四）**

**一、实验题目**

实验四 视图与索引

**二、实验目的**

1. 掌握 使⽤SQL语⾔进⾏视图的创建、修改与删除操作。

2. 掌握 视图的使⽤⽅法。

3. 掌握 使⽤SQL语⾔创建索引。

4. 理解 索引对查询优化的影响，并能根据实际情况决定 索引的使⽤。

**三、实验内容与实现**

**【实验要求一及实现】**

1、创建视图，并通过视图查询**200401班学生的学号、姓名，选修的课程名称及其正考成绩**。

**【答】**

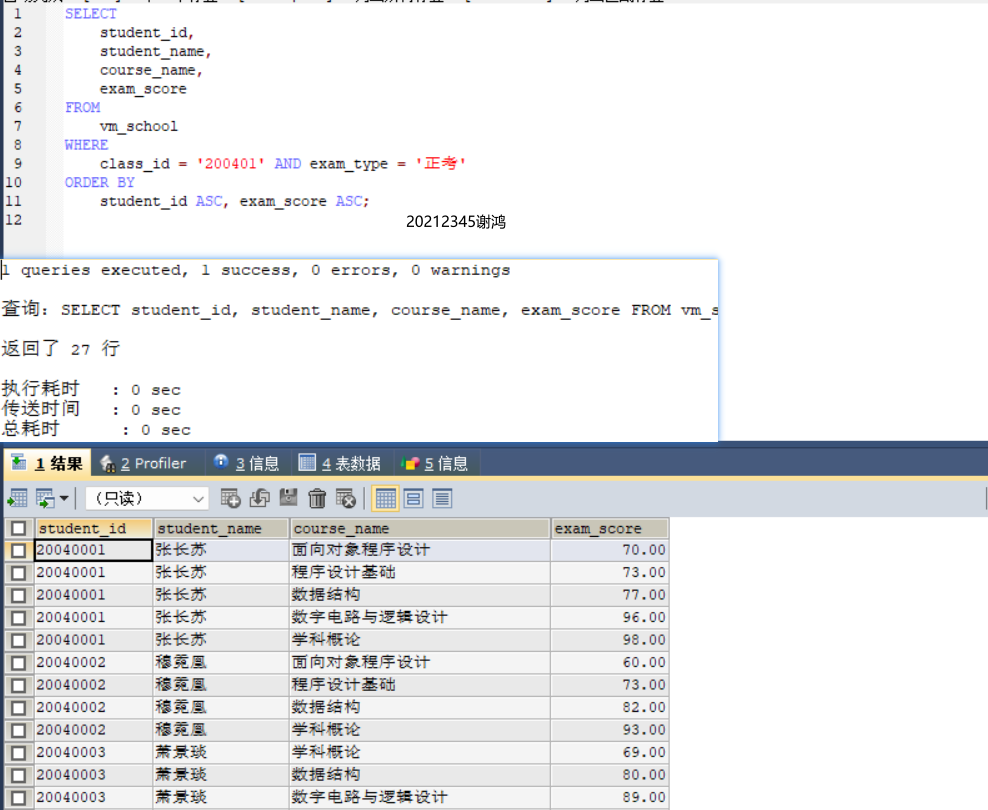
创建视图语句：

1. CREATE VIEW vm\_1 AS
2. SELECT
3. st.id AS student\_id,
4. st.name AS student\_name,
5. co.name AS course\_name,
6. sc.score AS exam\_score,
7. cl.id AS class\_id,
8. sc.type AS exam\_type
9. FROM
10. students st
11. JOIN
12. classes cl ON st.class\_id = cl.id
13. JOIN
14. scores sc ON st.id = sc.student\_id
15. JOIN
16. course\_info ci ON sc.cs\_id = ci.id
17. JOIN
18. courses co ON ci.course\_id = co.id

查询SQL语句：

1. SELECT
2. student\_id,
3. student\_name,
4. course\_name,
5. exam\_score
6. FROM
7. vm\_school
8. WHERE
9. class\_id = '200401' AND exam\_type = '正考'
10. ORDER BY
11. student\_id ASC, exam\_score ASC;

数据视图：



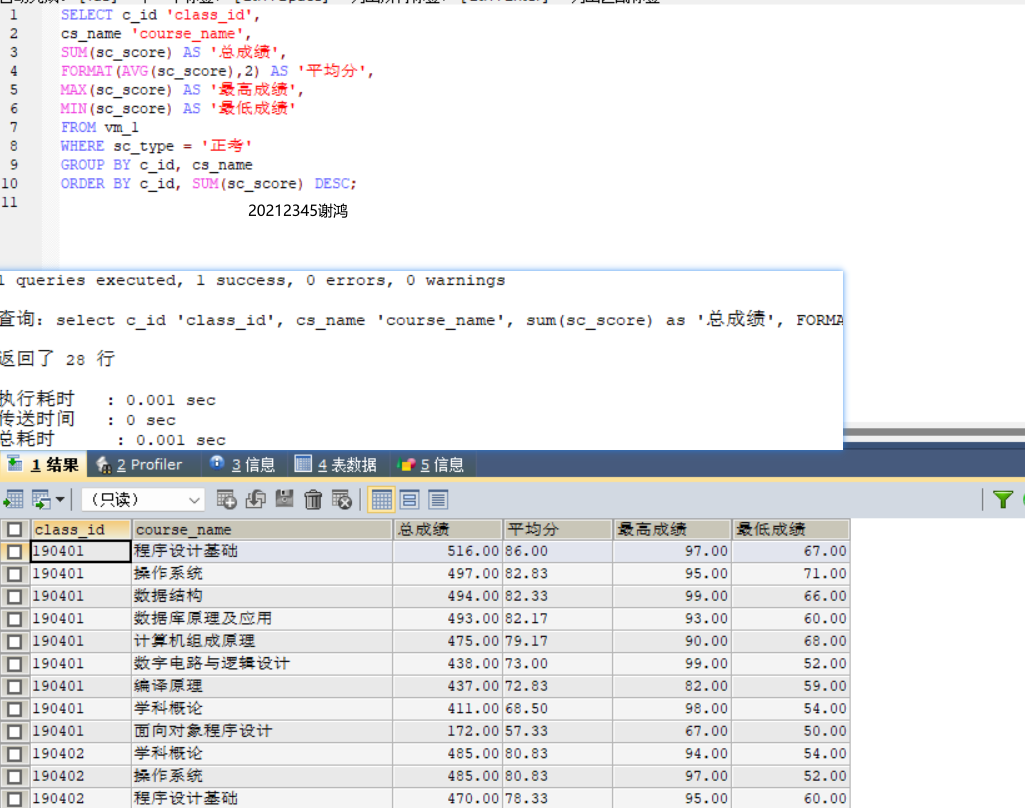
2、使用第1步的视图，**统计每个班每门课程的总成绩、平均分、最高成绩与最低成绩**。

**【答】**

SQL语句：

1. SELECT c\_id 'class\_id',
2. cs\_name 'course\_name',
3. SUM(sc\_score) AS '总成绩',
4. FORMAT(AVG(sc\_score),2) AS '平均分',
5. MAX(sc\_score) AS '最高成绩',
6. MIN(sc\_score) AS '最低成绩'
7. FROM vm\_1
8. WHERE sc\_type = '正考'
9. GROUP BY c\_id, cs\_name
10. ORDER BY c\_id, SUM(sc\_score) DESC;

数据视图：



3、创建视图，并通过视图查询**计算机科学与工程学院教师编号、姓名、为2019级上过的课程名称及开课学期**。

**【答】**

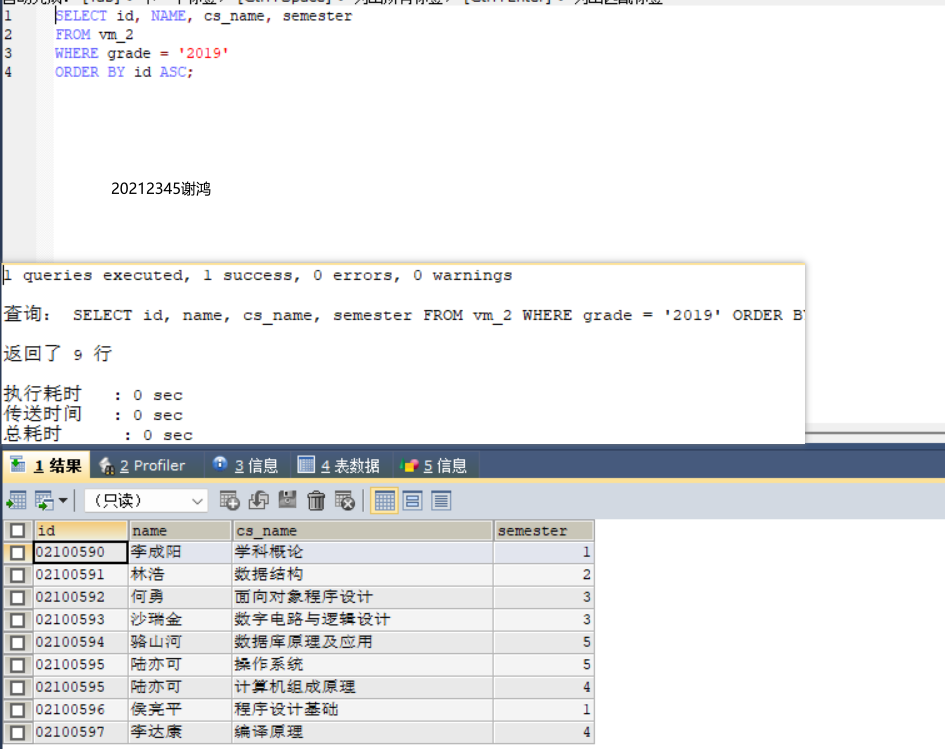
创建视图语句：

1. CREATE VIEW vm\_2(id, NAME, cs\_name, YEAR, grade, semester,course\_info\_id) AS
2. SELECT t.id, t.name, cs.name, ci.year, ci.grade, ci.semester,ci.id
3. FROM teachers t, courses cs, course\_info ci
4. WHERE t.id = ci.teacher\_id AND ci.course\_id = cs.id;

查询SQL语句：

1. SELECT id, NAME, cs\_name, semester
2. FROM vm\_2
3. WHERE grade = '2019'
4. ORDER BY id ASC;

数据视图：



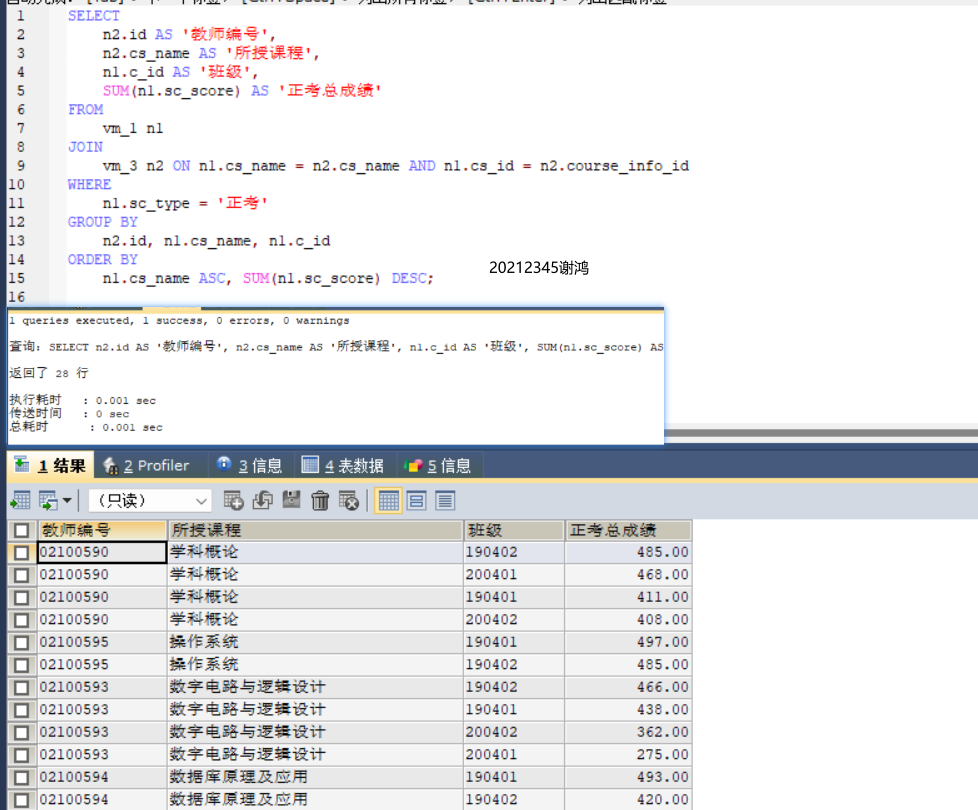
4、使用上述2个视图，查询**计算机与工程学院教师编号、所受课程名称、该课程每个班级的正考总成绩**，按课程名称升序总成绩降序排列。

**【答】**

SQL语句：

1. SELECT
2. n2.id AS '教师编号',
3. n2.cs\_name AS '所授课程',
4. n1.c\_id AS '班级',
5. SUM(n1.sc\_score) AS '正考总成绩'
6. FROM
7. vm\_1 n1
8. JOIN
9. vm\_3 n2 ON n1.cs\_name = n2.cs\_name AND n1.cs\_id = n2.course\_info\_id
10. WHERE
11. n1.sc\_type = '正考'
12. GROUP BY
13. n2.id, n1.cs\_name, n1.c\_id
14. ORDER BY
15. n1.cs\_name ASC, SUM(n1.sc\_score) DESC;

数据视图：



**【实验要求二及实现】**

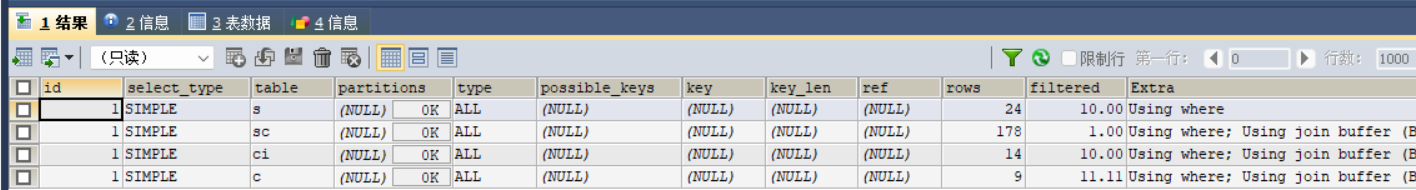
1、分析200401班学生的学号、姓名、选修的课程名称及其正考成绩查询语句的可执行计划。

**【答】**

SQL语句：

1. EXPLAIN
2. select s.id , s.name , c.name , sc.score
3. from students s, scores sc, courses c, course\_info ci
4. where s.class\_id = '200401'
5. and sc.student\_id = s.id
6. and sc.cs\_id = ci.id
7. and ci.course\_id = c.id
8. and sc.type = '正考';

数据视图：



2、给出对上述语句的优化方案。

**【答】**

查询优化：

确保没有不必要的列在SELECT中被提取。

如果可能，使用子查询或者临时表来减少重复的数据处理。

确保索引的使用：

对于过滤条件中经常使用的列（如 classes.id 和 scores.type），应确保有索引。

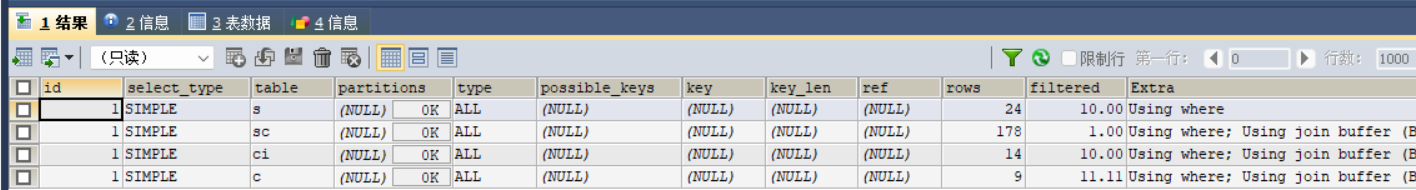
对于JOIN操作的外键（如 students.class\_id, scores.student\_id, course\_info.id 等），也应该有索引。

3、创建所需索引，并比对索引创建前后可执行计划发生的改变。

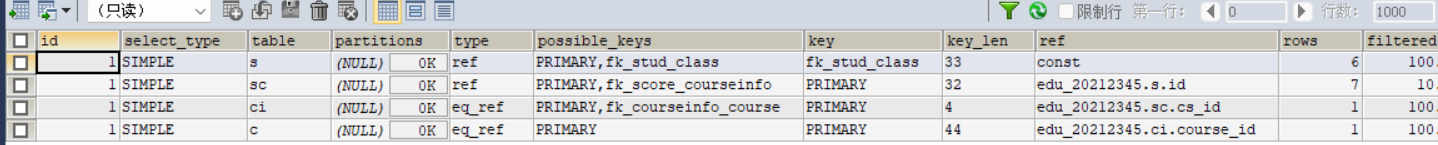
**【答】**

添加索引前后对比图：

前：



后：



**四、实验问题总结与心得**

在这次实验中，我深入学习了SQL语言在数据库视图操作中的应用。最初，视图的概念对我来说是一个挑战，但随着实践的深入，我逐渐理解了它在简化数据库操作和提高数据安全性方面的重要性。通过创建和修改视图，我学会了如何有效管理和呈现复杂的数据集。此外，实验中关于索引的部分让我领悟到数据库性能优化的重要性。我了解到，正确的索引可以显著提高查询速度，但也需要根据数据的特点和查询的频率来合理选择索引类型。通过比较不同索引对查询性能的影响，我更加理解了数据库设计和优化的复杂性。总的来说，这次实验不仅提升了我的SQL技能，还加深了我对数据库管理和优化的理解，这对我未来的学习和职业发展大有裨益。