实验名称

一、 实验目的

1. 掌握关系数据库语言 SQL 的使用。
2. 使所有的 SQL 作业都能上机通过。

二、 实验环境

1. 数据库管理系统：MySQL

三、 实验流程

1. 数据库的建立

为了方便，本次实验我在 Workbench 上进行，而不是采用命令行进行实验。根据实验指导，建立数据库可以点击如图 3.1.1 中圆框圈起的图标。

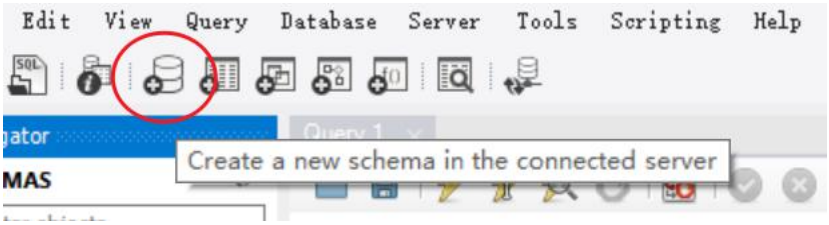

图 3.1.1 新建数据库图表

点击后，Workbench 会提示输入数据库的名字，如图 3.1.2 所示。

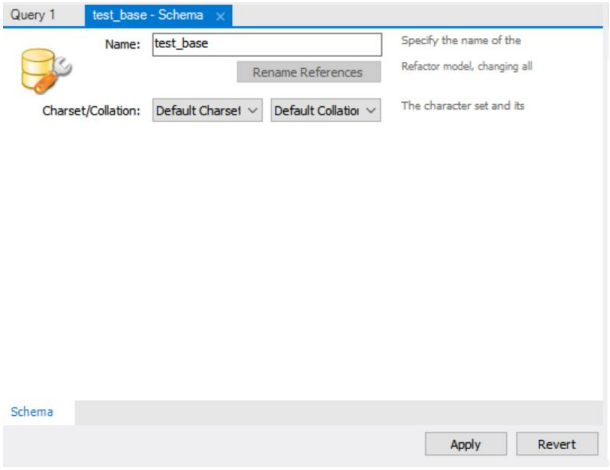

图 3.1.2 输入数据库名字

输入完毕后，可以选择性地输入一些初始化指令，这里不需要输入任何指令，直接点击 Apply 后即可，如图 3.1.3 所示。

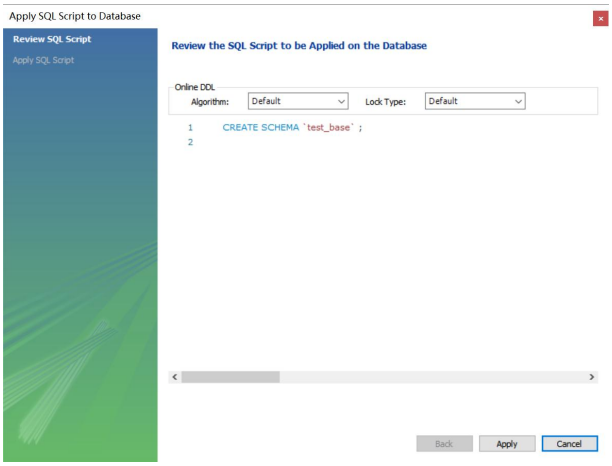

图 3.1.3 输入数据库初始化指令

刷新后检查左侧 Navigator，可见数据库 test_case 已经创建完成，如图 3.1.4 所示。

图 3.1.4 数据库创建完毕

当然，还有更为简单的方法可以创建数据库，直接打开一个写 SQL 语句的文本，输入指令即可创建数据库，如图 3.1.5 所示。第一条指令用来查看当前存在的数据库，第二条指令即为创建数据库。执行后可见左侧数据库增加了。

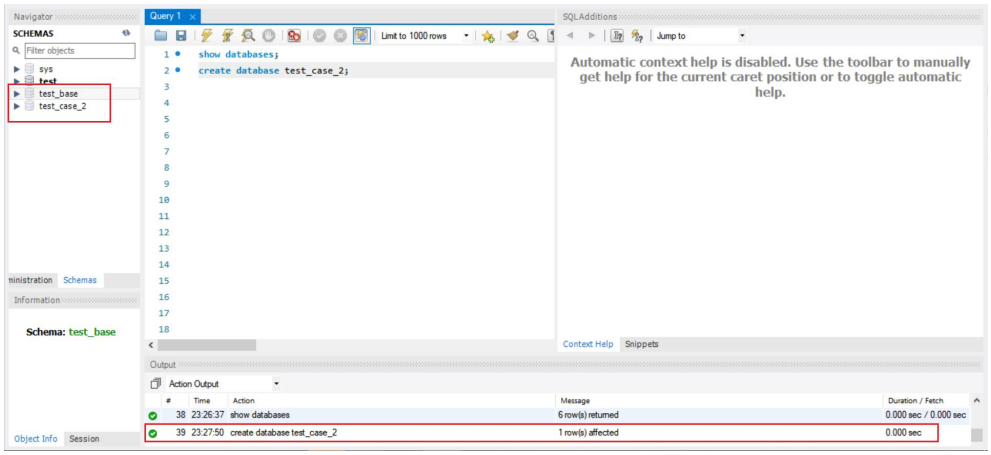

图 3.1.5 用指令创建数据库

此前的使用中，我已经建立了名为 test 的数据库，此后的实验步骤即在此数据库下进行。

2. 数据定义

首先查看数据库中的数据表，选择数据库 test 后查看其包含的数据表。如图 3.2.1 所示，依次执行第三第四行指令。可以看到 Result Grid 中列出了该数据库下所有的数据表名字。

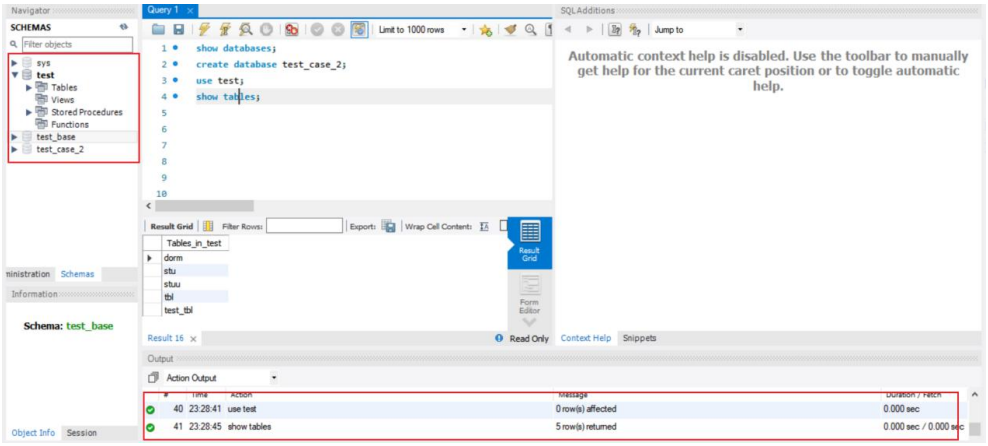

图 3.2.1 查看数据表表名

这里，先进行**数据表删除**，建立如图 3.2.2 所示的 procedure 用于删除，调用后可见左侧 navigator 显示的数据表减少了，数据表成功删除。

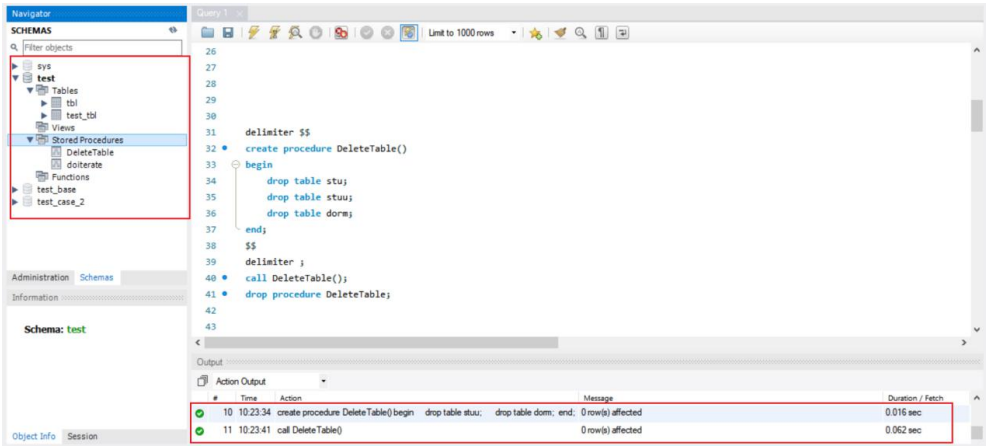

图 3.2.2 删除数据表

接下来，重新**创建数据表**，建立如图 3.2.3 所示的 procedure 用于创建，调用后可见左侧 navigator 显示的数据表增加，数据表成功创建。其中约束了主键与外键。

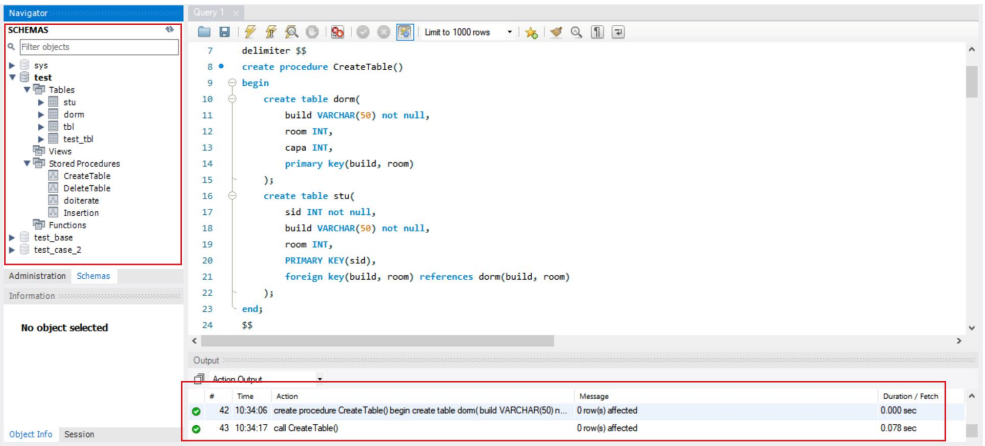

图 3.2.3 创建数据表

这里先进行一次**数据的插入**（数据更新），建立如图 3.2.4 所示的 procedure 用于插入数据，调用后查看数据表，可见 Result Grid 内显示数据成功插入，如图 3.2.5。

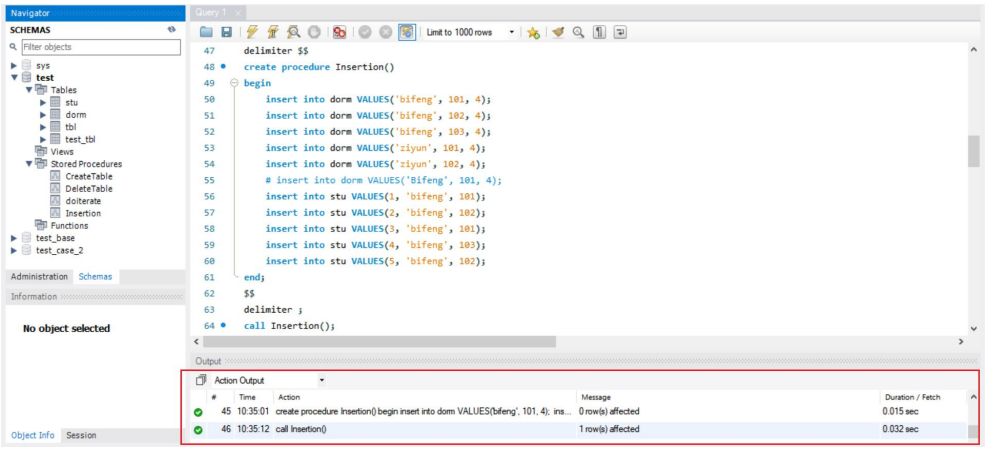

图 3.2.4 插入数据

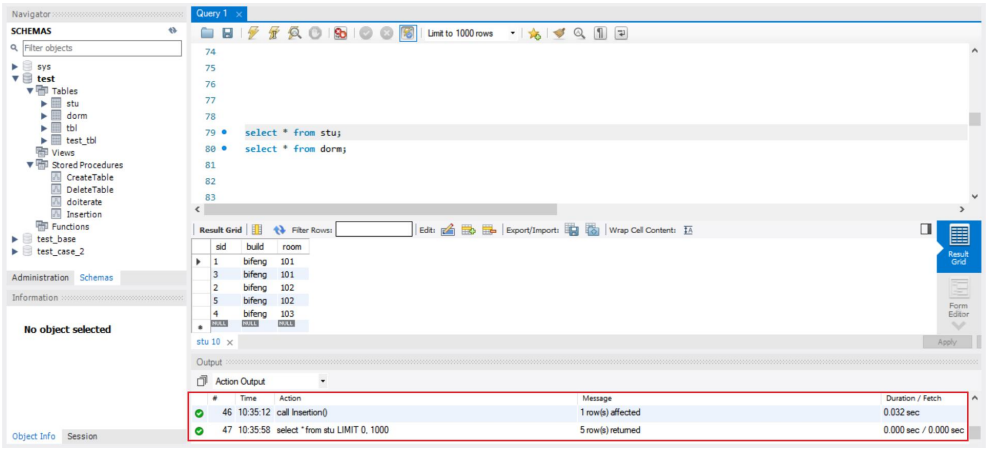

图 3.2.5 查看数据表

接下来对数据表进行**修改**，如图 3.2.6 所示，先给数据表增加一个属性，再对其 sid 为 1 的学生更新姓名。最后通过 select 语句查看结果，可见 sid 为 1 的学生已经有了姓名。

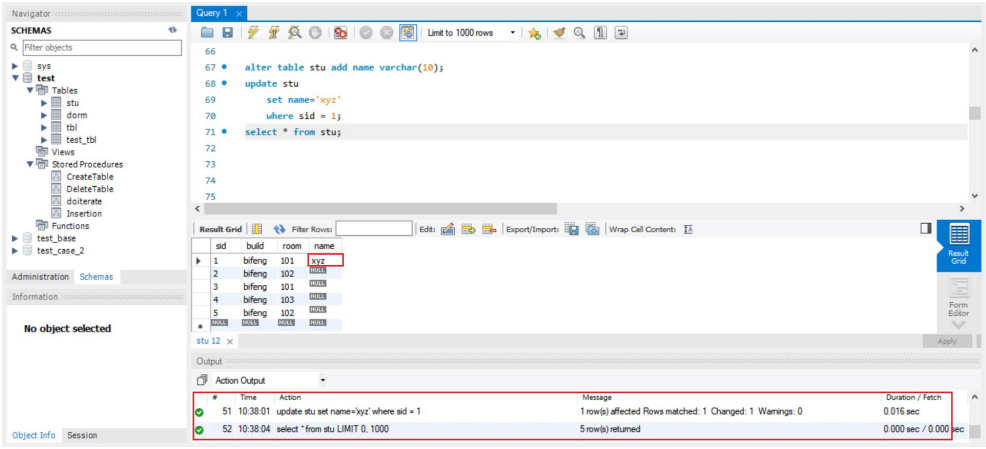

图 3.2.6 修改数据表

接下来**建立索引**，如图 3.2.7 所示，给数据表 stu 建立 student 索引，索引学生的 sid 与姓名。可见执行指令后，左侧的 indexes 出现了 student 索引。（dormitory 是外键索引，我给其重命名了）

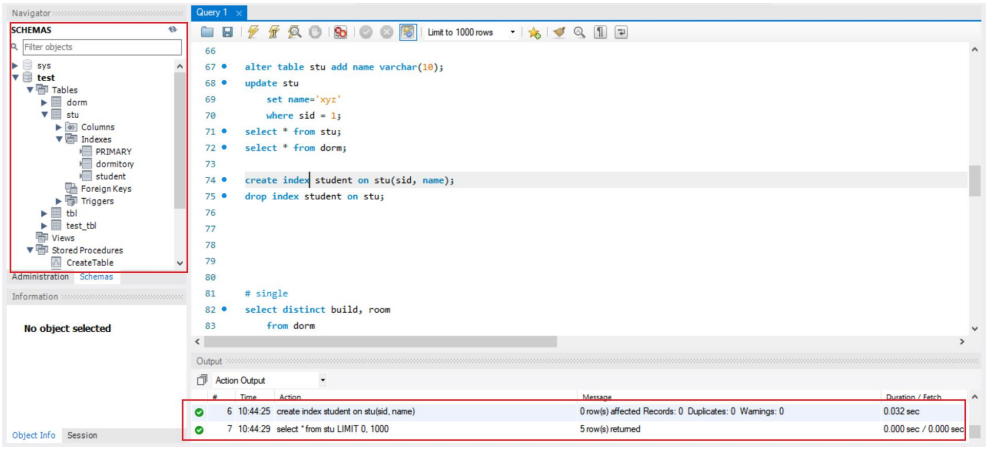

图 3.2.7 建立索引

删除索引很简单，如图 3.2.8 所示，执行 drop 语句后即可。执行后 Navigator 中的 student 索引消失。

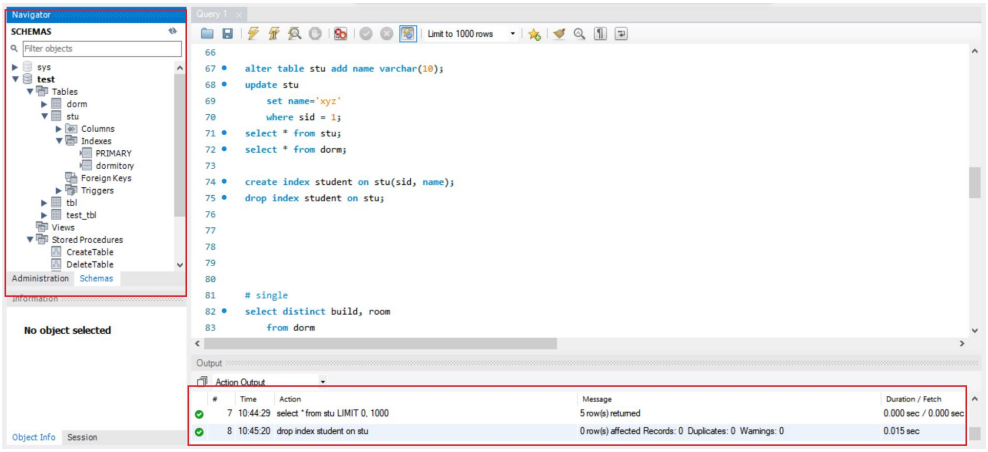

图 3.2.8 删除索引

建立视图过程如图 3.2.9，执行语句后可见 Navigator 下多出了名为 sid_101 的视图，其代表着所有住在 101 的学生的 sid。（此处暂不考虑是哪一建筑的 101）

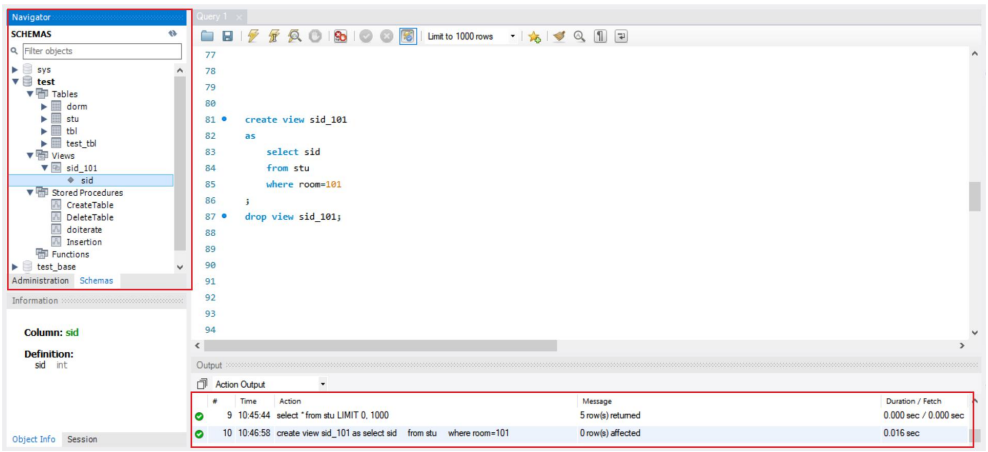

图 3.2.9 建立视图

删除视图则只要执行 drop 语句，如图 3.2.10，执行后视图 sid_101 消失。

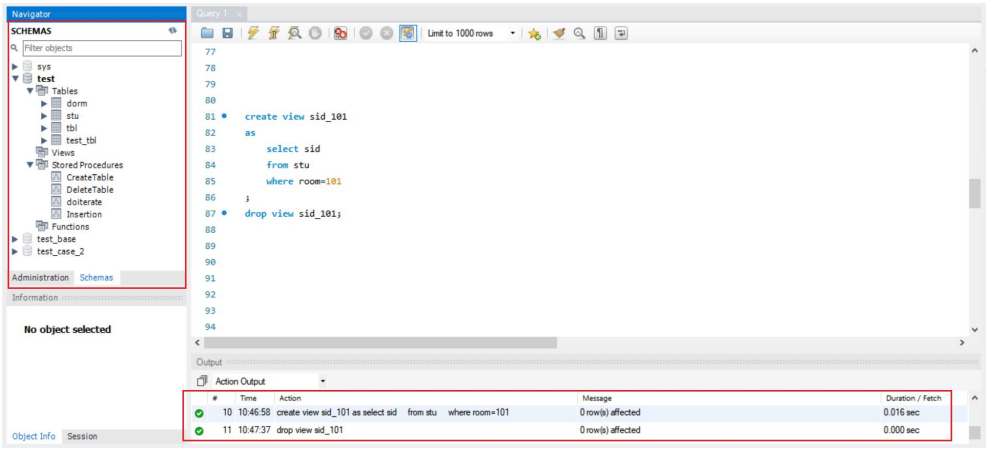

图 3.2.10 删除视图

3. 数据更新

数据的插入已经在数据定义中使用过了，不再复述。

删除表数据需要用到 delete 语句，如图 3.3.1，即为删除 stu 表中 sid 为 4 的学生数据。执行语句后，对比图 3.2.6，可见该学生信息已删除。

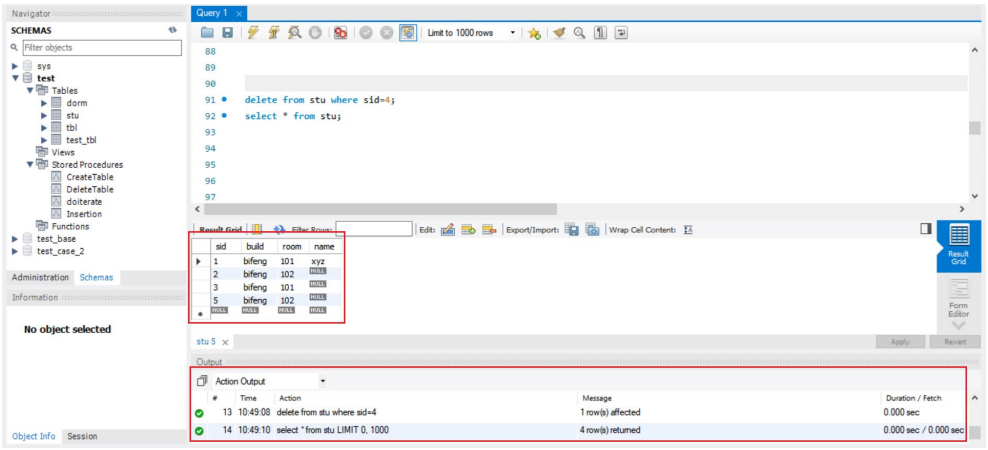

图 3.3.1 删除表数据

修改表数据需要用到 update 语句，也已经在数据定义中使用过了，不再复述。

4. 数据查询

首先进行**单表查询**，其实单表查询已经运用多次，此处展示附条件的查询。如图 3.4.1，即为查询所有的 102 房间，显示其建筑与房间号。

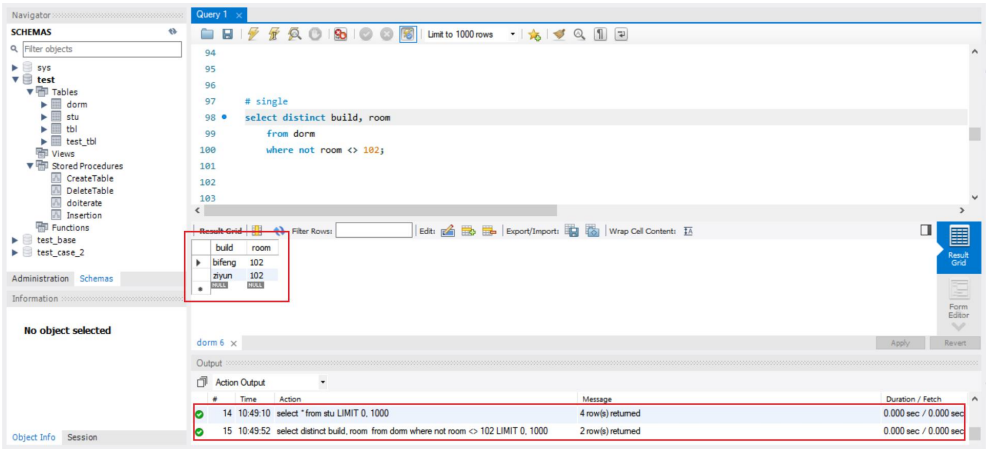

图 3.4.1 单表查询

多表查询如图 3.4.2 所示，查询 stu 与 dorm 自然连接后的表中，房间号不为 103 的所有信息。Result Grid 正确显示了结果。

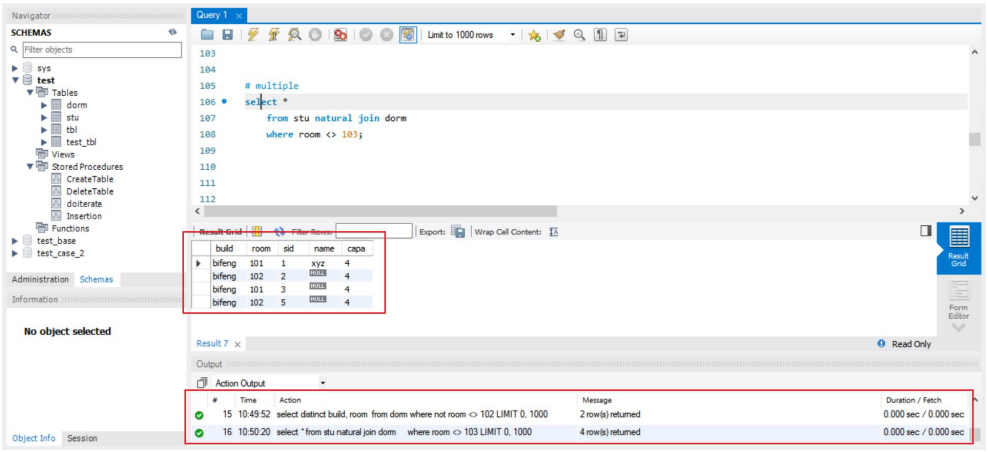

图 3.4.2 多表查询

嵌套子查询如图 3.4.3 所示，查询碧峰中有学生在住的房间有多少。

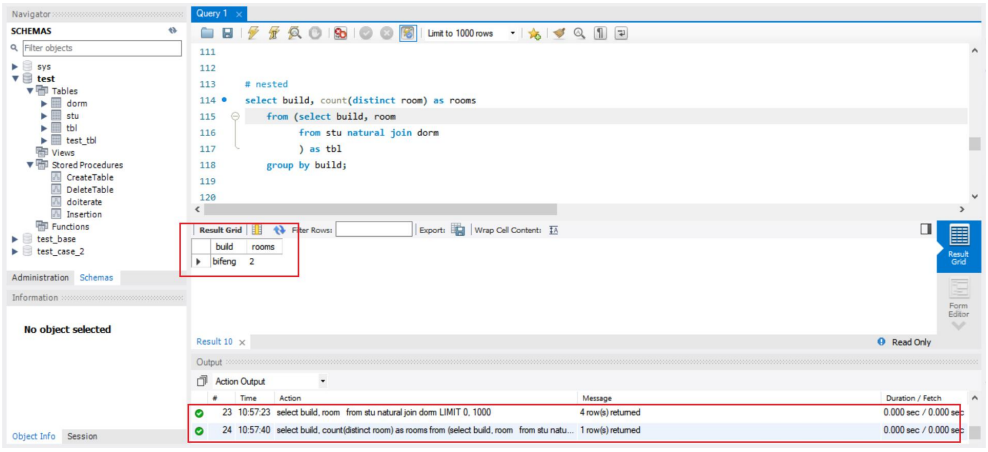

图 3.4.3 嵌套子查询

5. 视图操作

用视图进行数据查询如图 3.5.1 所示，利用视图 sid_101，统计住在 101 房间（不分建筑）的学生有多少人。

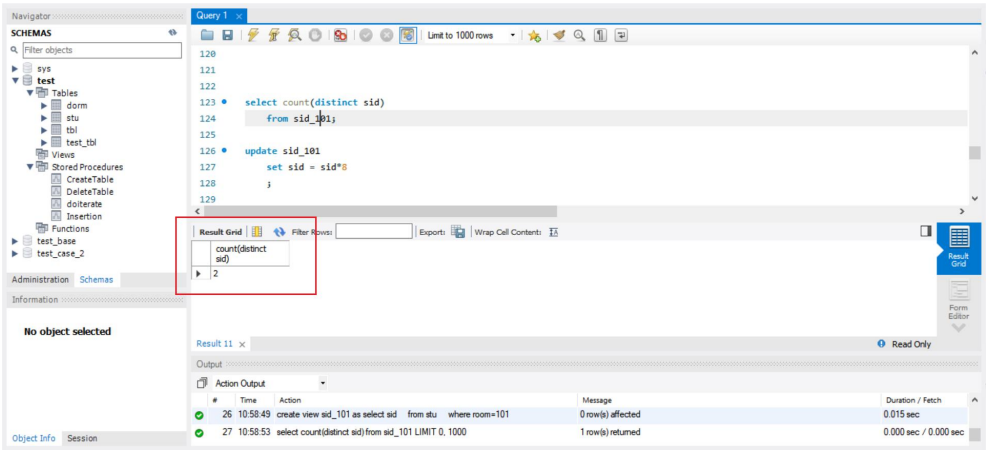

图 3.5.1 视图查询数据

利用视图进行数据修改如图 3.5.2 所示，利用视图 sid_101 将其中的学生 sid 乘八倍。执行后利用 select 语句查看结果，可见 sid 被成功修改。（此步由于 sid 为主键，需要关闭 safe update）

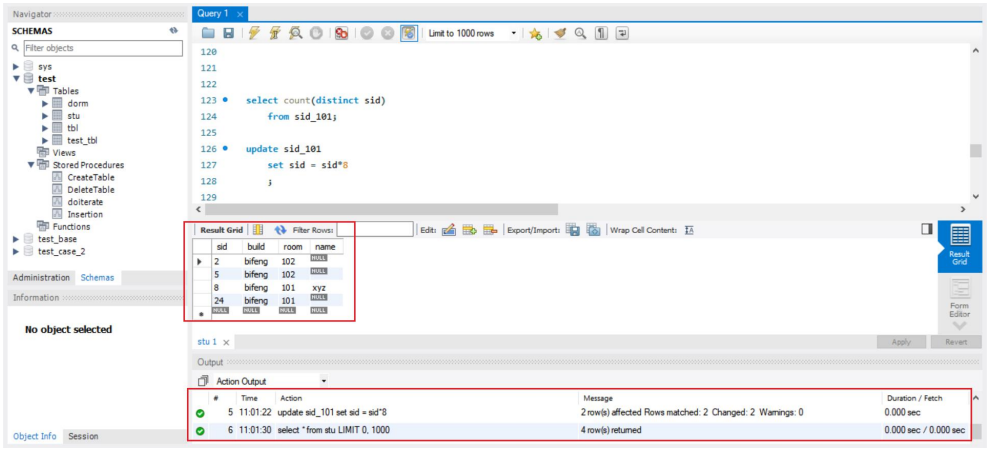

图 3.5.2 视图数据修改

四、 遇到的问题及解决方法

1. 嵌套子查询时报错 Every derived table must have its own alias

此报错需要在嵌套子查询时，给嵌套的 derived table 命名。如上实验过程，我给嵌套的 derived table 命名为 tbl 后，指令正常运行。

2. MySQL 不区分大小写

如图 3.2.4 插入数据时被我注释掉的指令，如果执行此语句，软件会提示该条信息已存在。

3. Delimiter 在创建 procedure 的作用

Delimiter 的作用是自定义指令结束符，创建 procedure 前修改结束符，被 procedure 包含的指令才能连贯地全部执行，否则则会中断，这一点在命令行下运行体现更为明显。而在创建完毕后，要记得把结束符改回来，否则影响其他指令的运行。

4. 外键的索引

如图 3.2.7 所示，数据表 stu 中有一名为 dormitory 的索引。其实该索引从创建表即存在，其索引即为声明的外键，我将其重命名为了 dormitory。

五、 总结

此次实验让我能够更加熟练地使用各种SQL指令,也更加了解Workbench的使用及设置，注意到了很多 MySQL 的语句特点和细节，如 MySQL 中的集合运算只支持 union，而不支持 except 等其他运算。