信息工程系

教

案

课程名称： 数据库系统

教　　师： 陈旭文

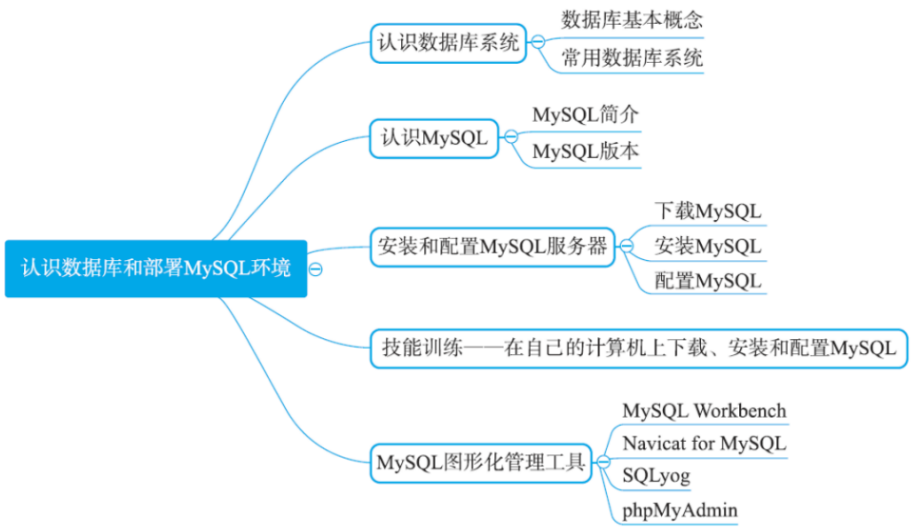
总 学 时 ： 54

理论学时： 18

实训学时： 36

上课班级： 电子 241、电子（自主）241

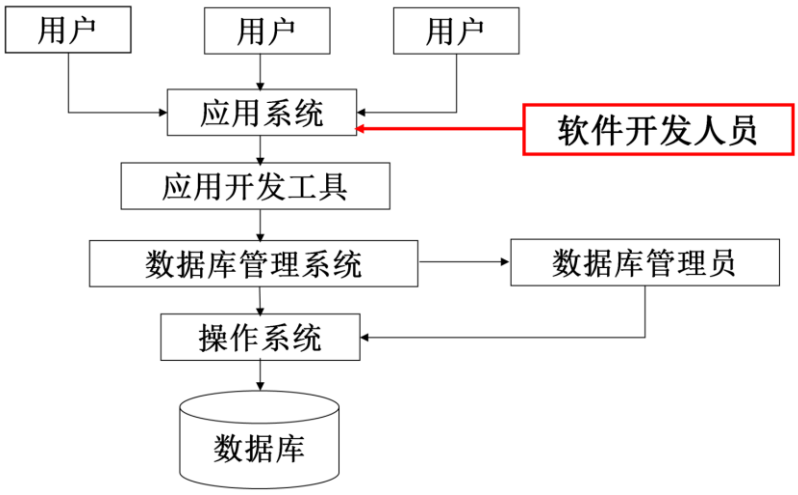
授课学期： 2025-2026 学年第一学期

<table>
<tr><td>课题</td><td colspan="3">第 1 章　认识数据库和部署 MySQL 环境</td></tr>
<tr><td>教学目标</td><td colspan="3">➢ 理解数据库的相关概念
➢ 了解常用数据库以及数据库的发展历史
➢ 了解 MySQL 数据库及其版本
➢ 熟练完成 MySQL 数据库的下载，并保存在自己计算机上
➢ 熟练完成在自己计算机上安装和配置 MySQL 数据库
➢ 灵活处理安装中遇到的一般问题
➢ 了解 4 种常见的 MySQL 图形化管理工具</td></tr>
<tr><td>重点难点</td><td colspan="3">➢ 配置 MySQL 数据库
➢ 处理安装 MySQL 数据库中遇到的一般问题</td></tr>
<tr><td>建议学时</td><td>3</td><td>教学教具</td><td>一体化教学实训室</td></tr>
<tr><td>主要教法</td><td colspan="3">讲授演示法</td></tr>
<tr><td>思政融入</td><td colspan="3">介绍数据库技术的发展
引入国产数据库的各种结构化数据和非结构化数据的厂商和产品介绍。国产替代拉动了国产数据库需求。国产自主可控信创最初是２０１６年提出，２０２０年随着电子公文的推进经历两年，在２０２２年上半年整体收官，下一步进入政府下沉部门和行业国产化替代的阶段。数据库作为国产替代核心产品，华为高斯、达梦、金仓、星环等产品已经从起步的“能用”阶段发展到逐步“好用”的阶段。国产数据库产品在信创推动的政府和其他行业得到广泛地应用，是自身迭代升级的机会，拉进和国外厂商之间的差距。目前党政、金融、电信、石油、电力、交通等行业出台的相关政策，将全面带动其他行业国产替代。
引导学生讨论我国信创产业的发展，提升国产数据使用的民族自豪感。</td></tr>
<tr><td colspan="4">教　学　过　程
 
</td></tr>
</table>

一、认识数据库系统

- 数据库技术的重要性
  - 90%以上的应用软件都需要使用到数据库系统
- 数据库技术的作用
  - 按照一定的模型进行组织和存储数据，方便检索和访问
  - 可以保证数据的完整性
  - 可以满足多用户使用数据的安全性
  - 数据库技术可以进行“大数据分析”
- 数据库
  - DataBase，简称：DB
  - 指长期存储在计算机内有组织的、可共享的数据集合
  - 数据库中数据的特点
    - ✓ 按一定的数据模型组织、描述和存储
    - ✓ 较小的冗余度
    - ✓ 较高的数据独立性和易扩展性
    - ✓ 为各种用户共享
- 数据库管理系统
  - DataBase Management System，简称：DBMS
  - 位于用户与操作系统之间的一层数据管理软件
  - 为了建立、使用和维护数据库而配置的**系统软件**

    例如：MySQL、Access、Microsoft SQL Server、Oracle 等
- 数据库系统
  - DataBase System，简称：DBS
  - 指在计算机系统中引入数据库后的系统
  - 组成：数据库、数据库管理系统（及其开发工具）、应用系统、数据库管理员（DataBase Administrator，简称 DBA）、用户
- 数据库系统结构图



- DBMS 是 DBS 的**核心**

- 数据库管理系统统一管理和统一控制
  - ✓ 数据库的建立
  - ✓ 数据库的运用
  - ✓ 数据库的维护

■ 应用程序
- 作用：响应操作并显示结果、向数据库请求数据
- 要求：美观、操作简单方便

■ 数据库
- 作用：存储数据、检索数据、生成新的数据
- 要求：统一、安全、性能等

■ **时下主流的数据库**
- MySQL 数据库：关系型数据库、开放源代码、适合于中小型网站和应用系统

■ SQL Server 数据库
- 大型关系型数据库
- 微软公司产品
- 功能全面，效率高
- 大中型企业或单位数据库平台
- 具有 Microsoft 产品共有的易用性，用户广泛

■ Oracle 数据库
- 关系型数据库
- 大型数据库
- Oracle 公司的产品
- 产品免费、服务收费

■ DB2 数据库
- 关系型数据库
- IBM 公司的产品
- 网络支持能力强，每个子系统可以连接十几万个分布式用户
- 可同时激活上千个活动线程
- 适合大型分布式应用系统

二、认识 MySQL

- MySQL 数据库是一个关系数据库管理系统
- 瑞典 MySQL AB 公司开发
- 2008 年被 Sun 公司收购
- Sun 公司又在 2010 年被 Oracle 公司收购

■ MySQL 数据库特点
- 可移植性好
- 支持跨平台
- 为多种编程语言提供了 API
- 核心程序采用完全多线程服务，可高效地利用多 CPU 资源
- 优化的 SQL 查询算法，查询速度得到更好的提升
- 既能够作为一个单独的应用程序应用于客户端服务器网络环境，也能够作为一个库而嵌入到其他软件中提供多语言支持
- 提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接途径
- 提供用于管理、检查、优化数据库操作的管理工具

- 可以处理拥有千万条记录的大型数据库

■ MySQL 的主要版本

- Oracle MySQL Cloud Service（企业版）
- MySQL Enterprise Edition（企业版）
- MySQL Cluster CGE（企业版）
- **MySQL Community Edition（社区版）**

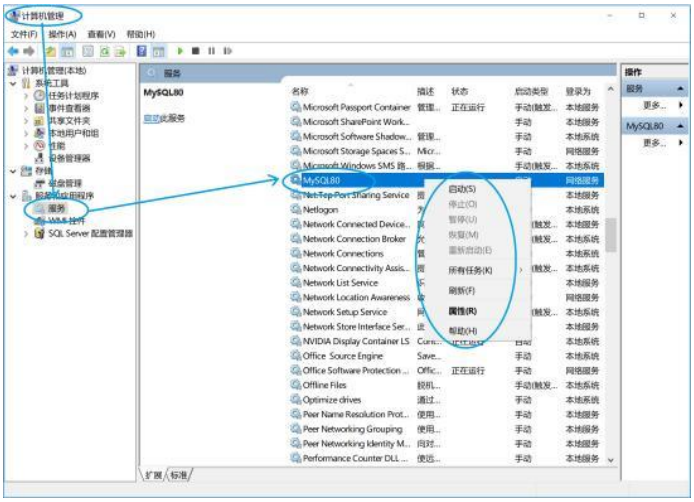
三、安装和配置 MySQL 服务器

下载地址：https://dev.mysql.com/downloads/installer/

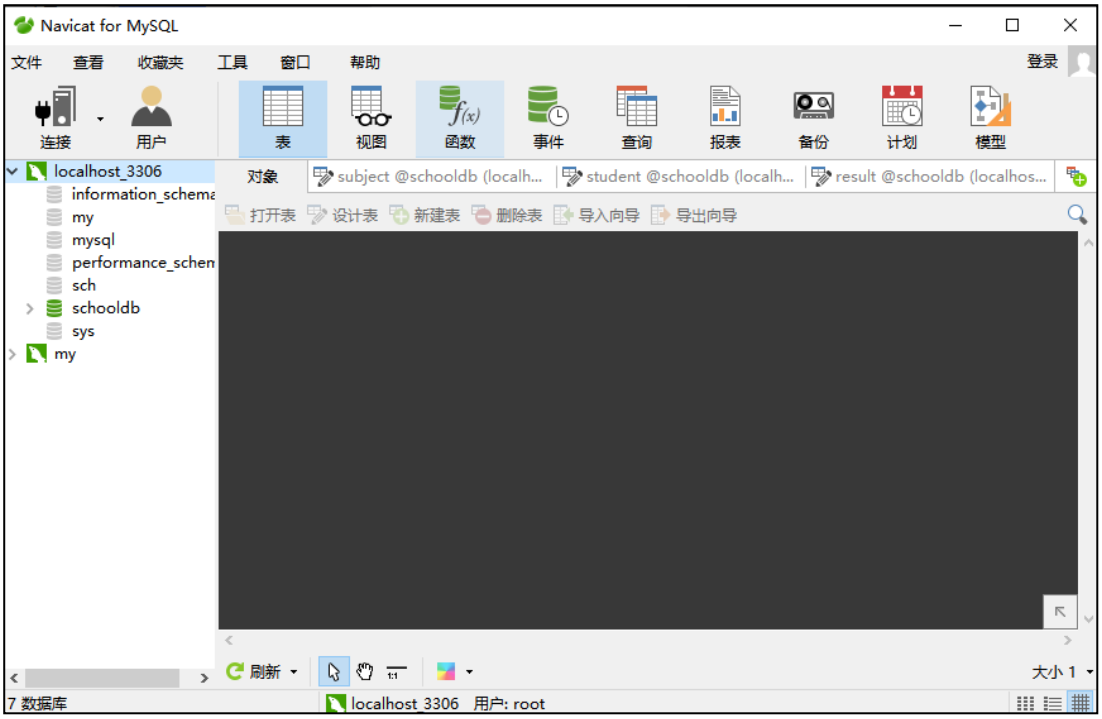
1、使用 phpStudy_32.zip **安装 phpStudy 应用程序**。

2、使用 navicat11.zip **安装 navicat11 应用程序**。

查看 MySQL 服务情况



3、使用“Navicat for MySQL”连接 MySQL。



4、使用 cmd 连接 MySQL。

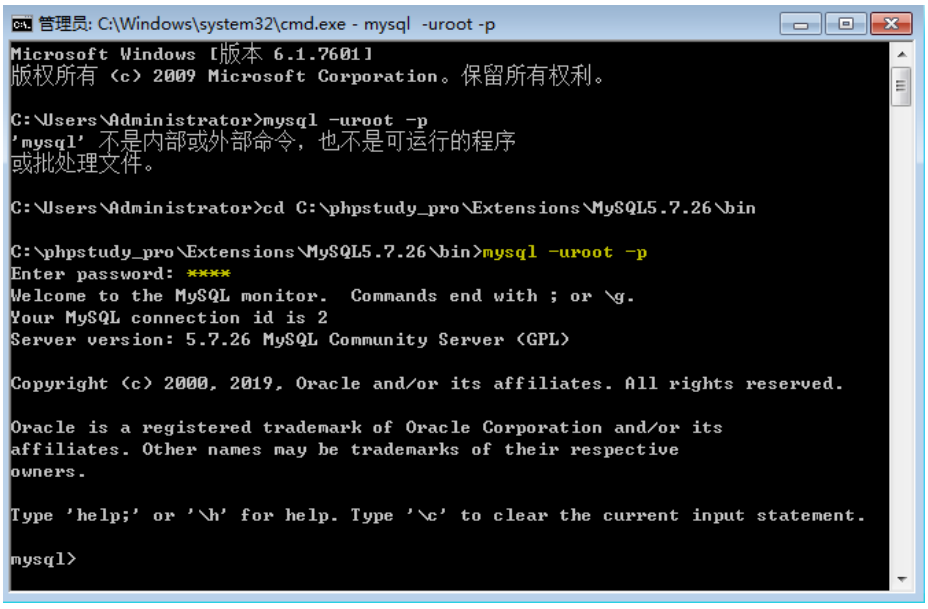
（1）定位 phpstudy 应用程序位置；

（2）进入 MySQL——bin 目录，复制地址栏目录信息；

（3）打开 cmd 软件；

（4）使用 cd 命令切换到 bin 目录；

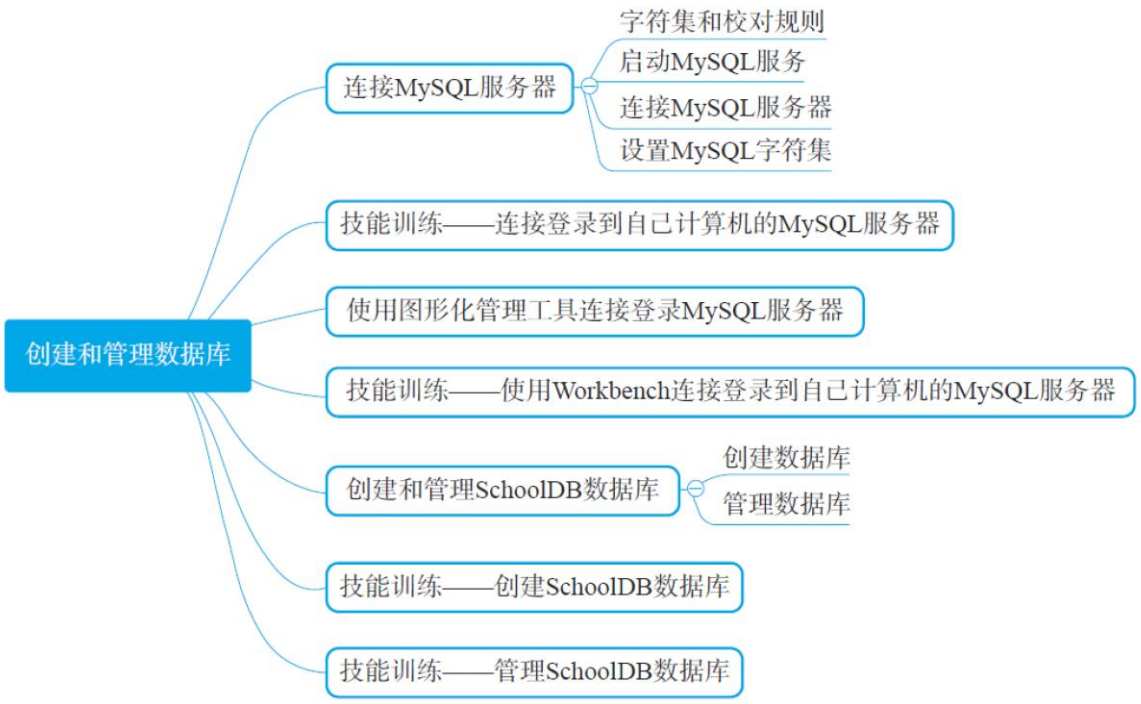
（5）运行 mysql -uroot -p，然后输入密码完成登录。



课后反思	

课题	第 02 章 创建和管理数据库		
教学目标	➢ 理解字符集和校对规则的相关概念 ➢ 熟练掌握使用命令行的方式创建数据库 ➢ 熟练掌握使用 Workbench 客户端创建数据库 ➢ 熟练掌握查看显示和打开数据库的操作 ➢ 熟练掌握修改数据库的方法和步骤 ➢ 掌握删除数据库的操作		
重点难点	➢ 在命令行模式下对数据库的操作 ➢ 数据库创建和修改命令的使用方法和步骤		
建议学时	3	教学教具	一体化教学实训室
主要教法	讲授演示法		
思政融入	**强调数据的严谨性** 强调 “IF NOT EXISTS” 子句的使用体现了编程中的 “严谨性”。在数据管理工作中，严谨是基本职业素养，就像财务人员核算账目需反复核对一样，数据操作者每一个代码的编写都要考虑潜在风险，避免因疏忽造成数据冲突或系统异常。这种严谨不仅是对工作负责，更是对数据使用者、企业的负责。		

教　学　过　程



一、连接 MySQL 服务器

■ 字符集和校验规则

字符（Character）是指人类语言中最小的表义符号，如 A、B 等对给定每个字符赋予一个数值，用数值来代表对应的字符，这一数值就是字符的编码。给定一系列字符并赋予对应的编码后，所有这些字符和编码对组成的集合，即字符集。

校验规则：指在同一字符集内字符之间的比较规则，确定字符集的校对规则后，才能在一个字符集上定义什么是等价的字符，以及字符之间的大小关系。

字符校对规则名称遵从命名惯例，以字符校对规则对应的字符集名称开头，以“_ci”、“_cs”或“_bin”结尾。

- “_ci”表示大小写不敏感，即不区分大小写
- “_cs”表示大小写敏感，即区分大小写
- “_bin”表示按编码值比较

例如:

在字符校对规则“utf8_general_ci”下

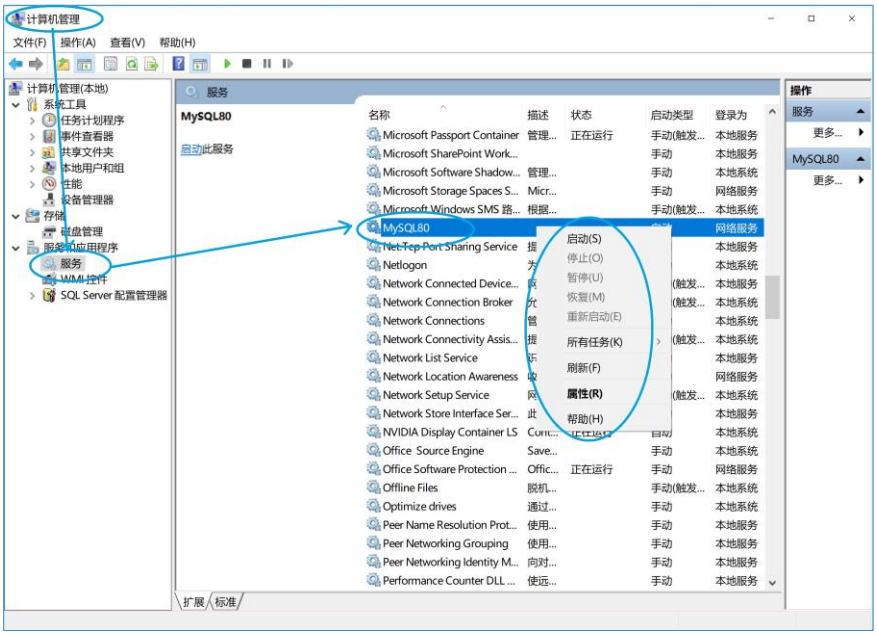
字符“a”和“A”是等价的，即不区分大小写

■ MySQL 支持中文的常用字符集主要有 3 种 DataBase，简称：DB

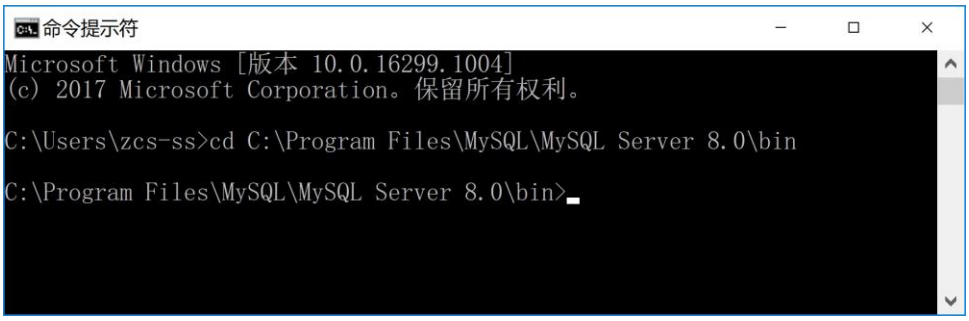
- UTF-8 字符集。互联网广泛支持的 Unicode 字符集，长度为 3 字节
- GBK 字符集。主要用于显示汉字，长度为 2 字节
- GB2312 按一定的数据模型组织、描述和存储

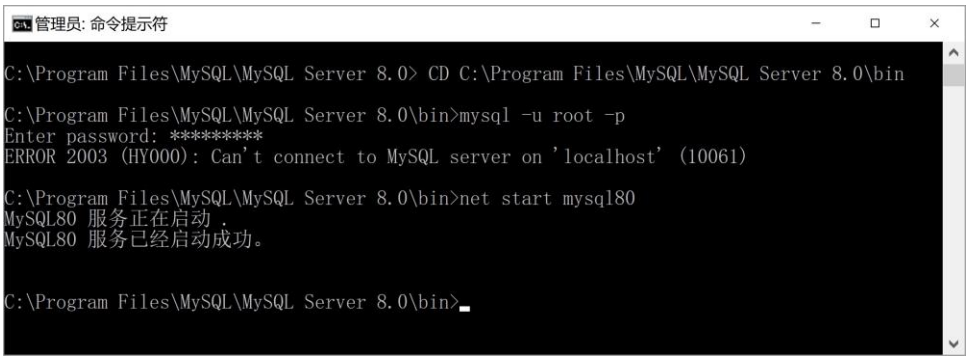
■ 启动和停止 MySQL 服务

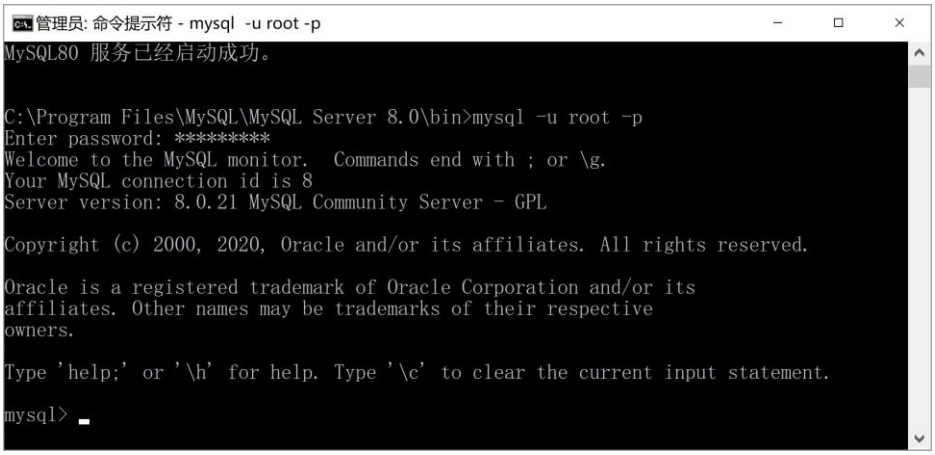
- 途径一：通过“计算机管理”窗口


计算机管理
文件(F) 操作(A) 查看(V) 帮助(H)
计算机管理(本地)
系统工具
任务计划程序
事件查看器
共享文件夹
本地用户和组
性能
设备管理器
存储
磁盘管理
服务和应用程序
服务
WMI 控件
SQL Server 配置管理器
服务
MySQL80
启动此服务
名称 描述 状态 启动类型 登录为
Microsoft Passport Container 管理... 正在运行 手动(触发... 本地服务
Microsoft SharePoint Work... 手动 本地服务
Microsoft Software Shadow... 管理... 手动 本地系统
Microsoft Storage Spaces S... Micr... 手动 网络服务
Microsoft Windows SMS 路... 根据... 手动(触发... 本地系统
MySQL80 网络服务
Net.Tcp Port Sharing Service 提... 本地服务
Netlogon 为... 本地系统
Network Connected Device... 网... (触发... 本地服务
Network Connection Broker 允... (触发... 本地系统
Network Connections 管... 本地系统
Network Connectivity Assis... 提... (触发... 本地系统
Network List Service 识... 本地服务
Network Location Awareness 收... 网络服务
Network Setup Service 网... (触发... 本地系统
Network Store Interface Ser... 此... 本地服务
NVIDIA Display Container LS Cont... 本地系统
Office Source Engine Save... 手动 本地系统
Office Software Protection ... Offic... 正在运行 手动 网络服务
Offline Files 脱机... 手动(触发... 本地系统
Optimize drives 通过... 手动 本地系统
Peer Name Resolution Prot... 使用... 手动 本地服务
Peer Networking Grouping 使用... 手动 本地服务
Peer Networking Identity M... 向对... 手动 本地服务
Performance Counter DLL ... 使远... 手动 本地服务
启动(S)
停止(O)
暂停(U)
恢复(M)
重新启动(E)
所有任务(K)
刷新(F)
属性(R)
帮助(H)
操作
服务
更多...
MySQL80
更多...
扩展 标准

- 途径二：命令提示符


命令提示符
Microsoft Windows [版本 10.0.16299.1004]
(c) 2017 Microsoft Corporation。保留所有权利。
C:\Users\zcs-ss>cd C:\Program Files\MySQL\MySQL Server 8.0\bin
C:\Program Files\MySQL\MySQL Server 8.0\bin>


管理员: 命令提示符
C:\Program Files\MySQL\MySQL Server 8.0> CD C:\Program Files\MySQL\MySQL Server 8.0\bin
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: *********
ERROR 2003 (HY000): Can't connect to MySQL server on 'localhost' (10061)
C:\Program Files\MySQL\MySQL Server 8.0\bin>net start mysql80
MySQL80 服务正在启动 .
MySQL80 服务已经启动成功。
C:\Program Files\MySQL\MySQL Server 8.0\bin>


管理员: 命令提示符 - mysql -u root -p
MySQL80 服务已经启动成功。
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: *********
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.21 MySQL Community Server - GPL
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>

■ 连接 MySQL 服务器

➢ 使用“Navicat for MySQL”连接 MySQL

➢ 使用 cmd 连接 MySQL

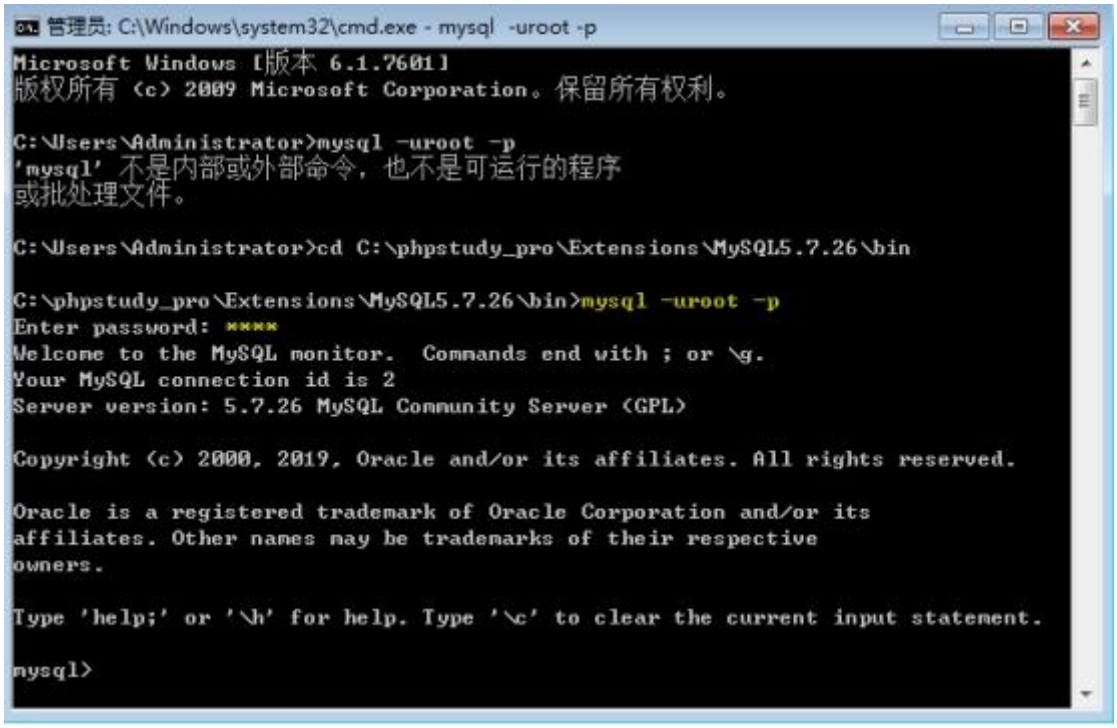
（1）定位 phpstudy 应用程序位置；

（2）进入 MySQL——bin 目录，复制地址栏目录信息；

（3）打开 cmd 软件；

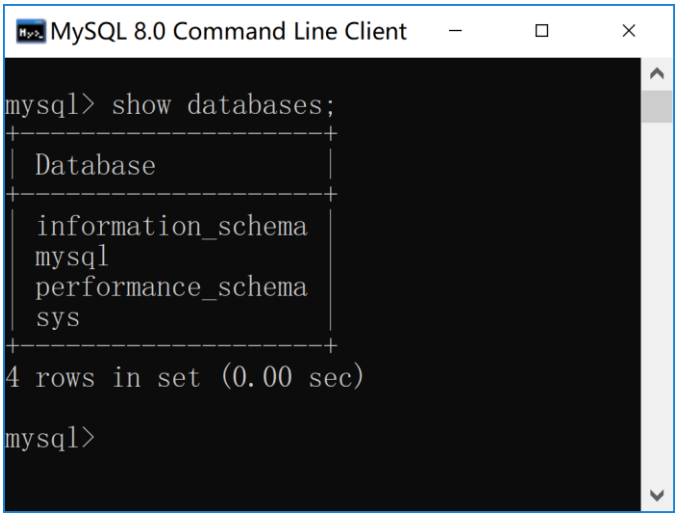
（4）使用 cd 命令切换到 bin 目录；

（5）运行 mysql -uroot -p，然后输入密码完成登录。



二、创建和管理 SchoolDB 数据库

- MySQL 自带 4 个数据库
  - information_schema：保存 MySQL 服务器维护的所有其他数据库的信息
  - Mysql：主要负责存储数据库的用户、权限设置、关键字
  - Sys：所有的数据都来自 performance_schema 数据库
  - performance_schema：主要用于收集数据库服务器性能参数
- 查看系统数据库，命令：show databases;



■ 使用命令行模式创建数据库

■基本语法格式介绍

CREATE {DATABASE|SCHEMA} [IF NOT EXISTS]数据库名

[DEFAULT CHARACTER SET [=] 字符集名

|[DEFAULT] COLLATE [=] 校对规则名];

◆{ }表示必选项、|表示任选其一、[]表示可选项

◆IF NOT EXISTS：在创建数据库前判断该数据库是否存在

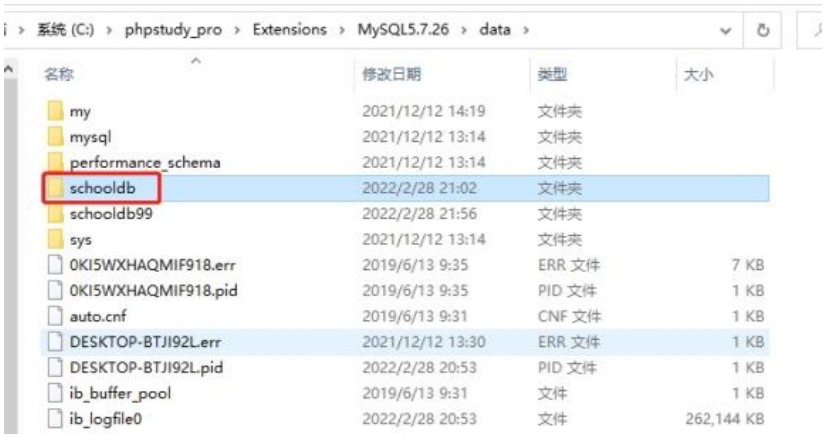
◆DEFAULT：采用默认值

◆CHARACTER SET：指定数据库的字符集

◆COLLATE：指定字符集校对规则

【演示示例 2-1】创建数据库 SchoolDB

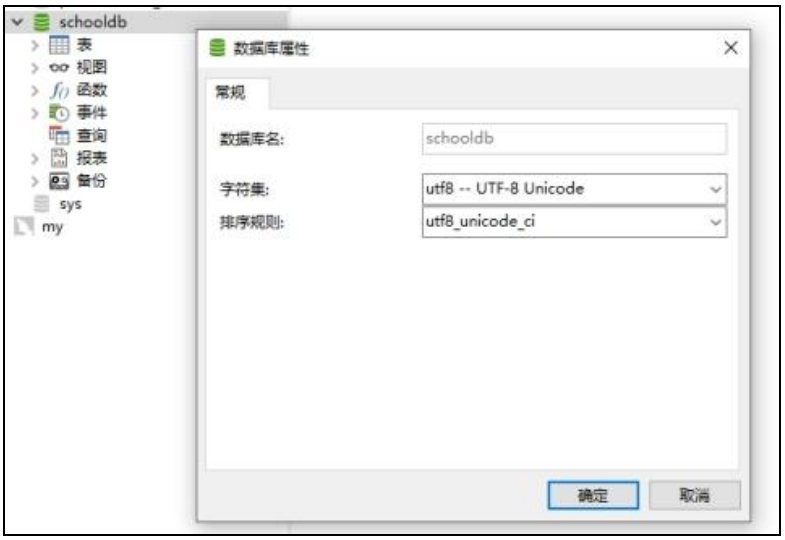
```
CREATE DATABASE IF NOT EXISTS SchoolDB
    DEFAULT CHARACTER SET gb2312
    DEFAULT COLLATE gb2312_chinese_ci;
```



- 管理数据库

（1）查看数据库：SHOW DATABASES;
（2）打开数据库：USE SCHOOLDB;
（3）修改数据库：ALTER DATABASE
（4）删除数据库：DROP DATABASE SCHOOLDB99;

- 在 navicat 中查看数据库属性：

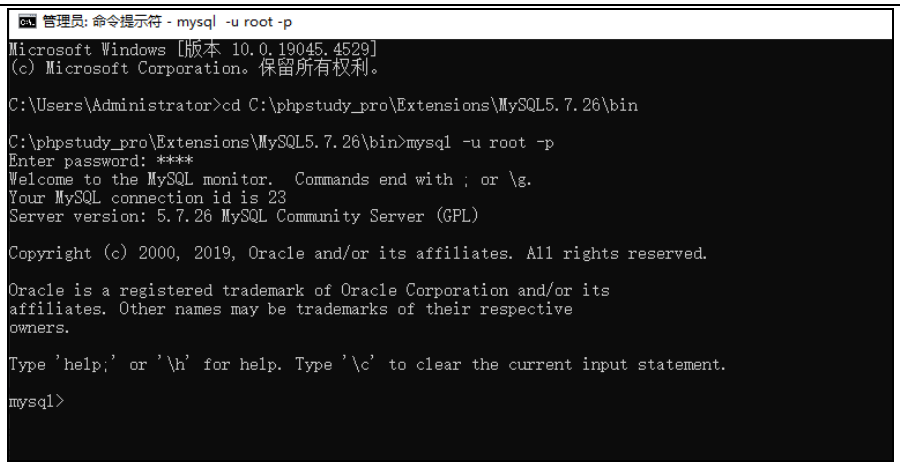


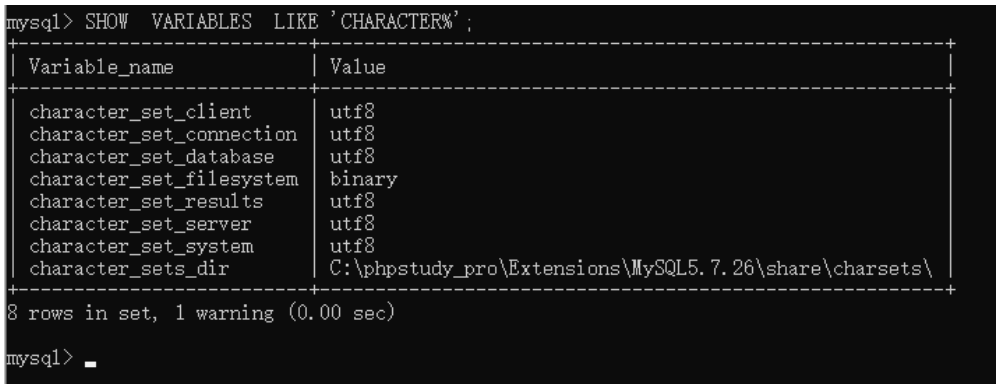
操作实训

1、查看了解当前系统字符集参数。

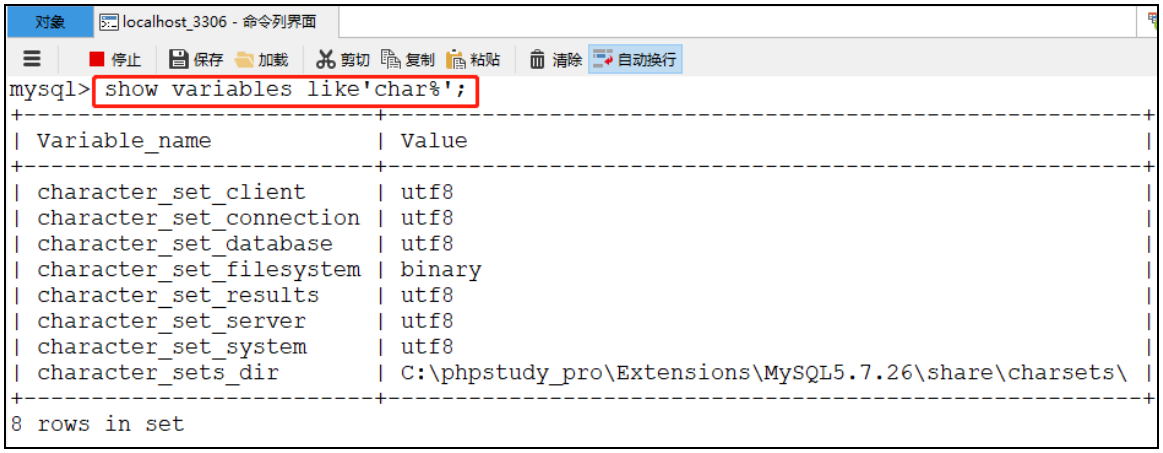
SHOW VARIABLES LIKE 'CHARACTER%';

（1）使用 cmd 连接 MySQL 数据库系统，输入以上命令：





（2）在 navicat 界面，选择【工具】——【命令列界面】，打开“命令列界面”，输入以上命令。



2、修改字符集参数。

（1）SET CHARACTER_SET_CLIENT = 'GB2312';

修改完后查看修改结果：SHOW VARIABLES LIKE 'CHARACTER%';

（2）SET NAMES UTF8;

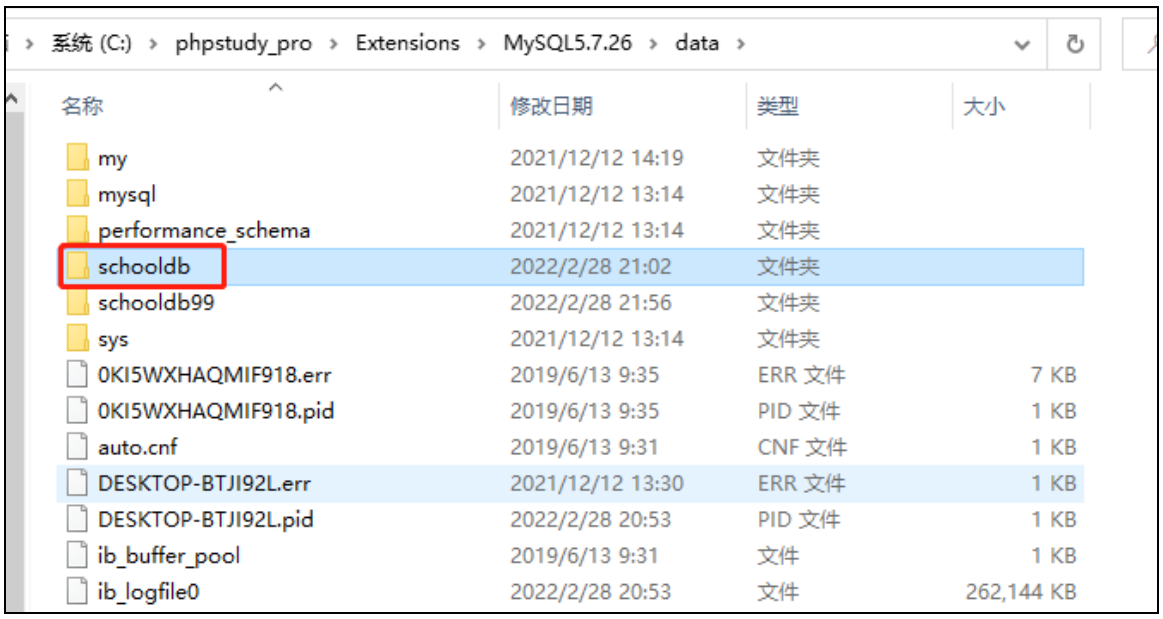
快速设置字符集为 UTF8，查看修改结果。

3、创建数据库

（1）创建数据库 SchoolDB

CREATE DATABASE IF NOT EXISTS SCHOOLDB;

打开数据库存放位置，查看数据库文件夹。



（2）创建数据库 abc

CREATE DATABASE IF NOT EXISTS abc;

4、管理数据库

（1）查看数据库：SHOW DATABASES;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| abc                |
| mysql              |
| performance_schema |
| sch                |
| schooldb           |
| sys                |
+--------------------+
7 rows in set
```

（2）打开数据库：USE SCHOOLDB;

```
mysql> use schooldb;
Database changed
```

```
mysql> show tables;
+--------------------+
| Tables_in_schooldb |
+--------------------+
| 大学英语成绩       |
| 王子洋             |
| grade              |
| result             |
| student            |
| subject            |
+--------------------+
6 rows in set
```

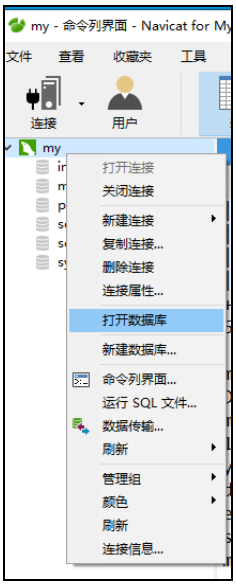
（3）修改数据库：ALTER DATABASE

（4）删除数据库：DROP DATABASE abc;

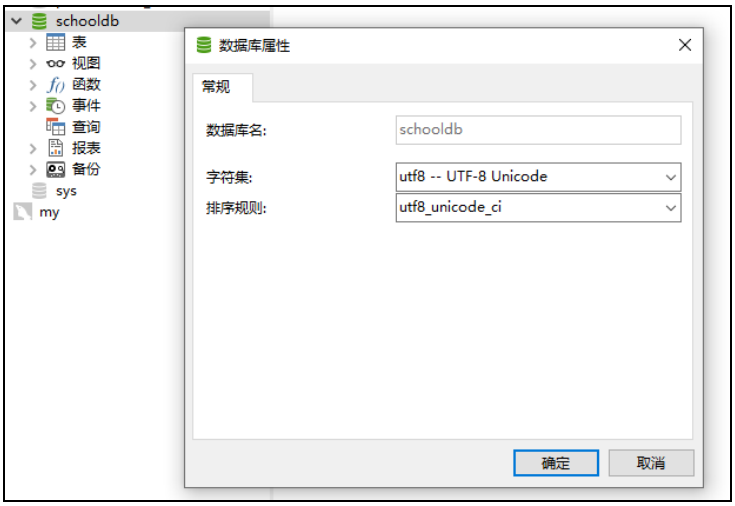
```
mysql> DROP DATABASE abc;
Query OK, 0 rows affected

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sch                |
| schooldb           |
| sys                |
+--------------------+
6 rows in set
```

5、在 navicat 中查看数据库属性：

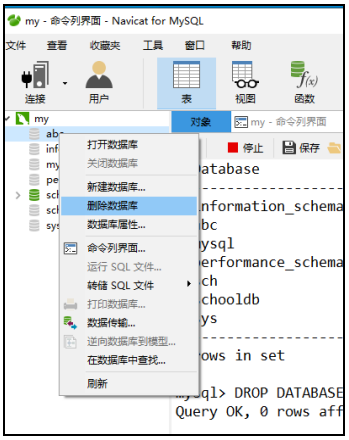
（1）新建数据库 SCHOOLDB99



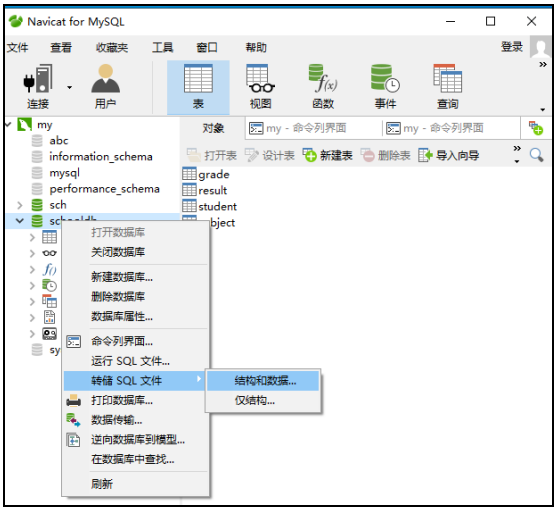
（2）查看数据库

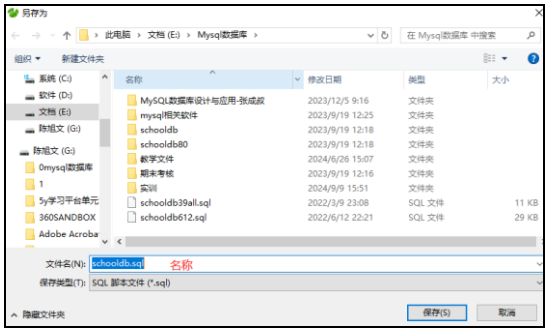


（3）删除数据库

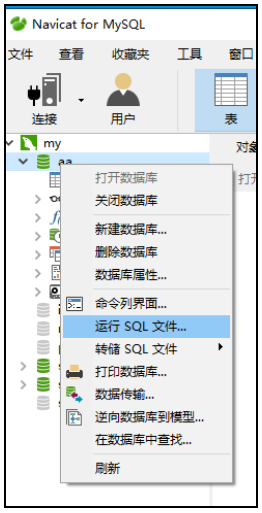


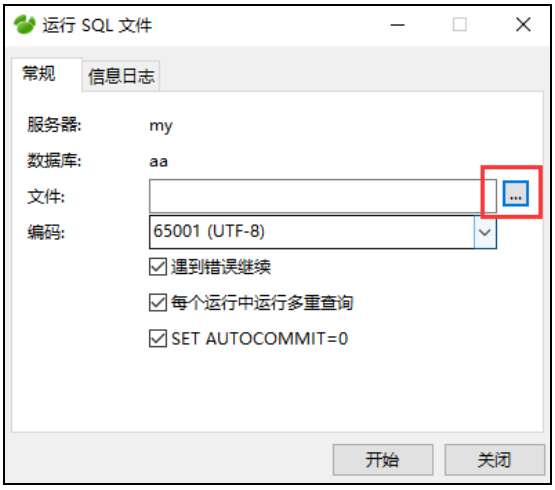
（4）数据库备份，非常重要！

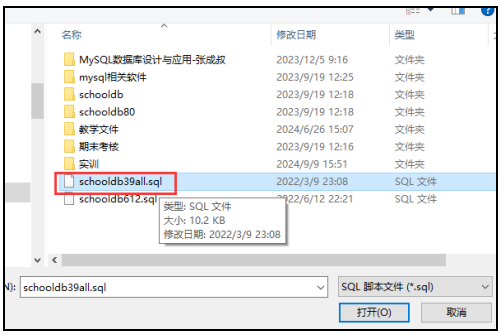


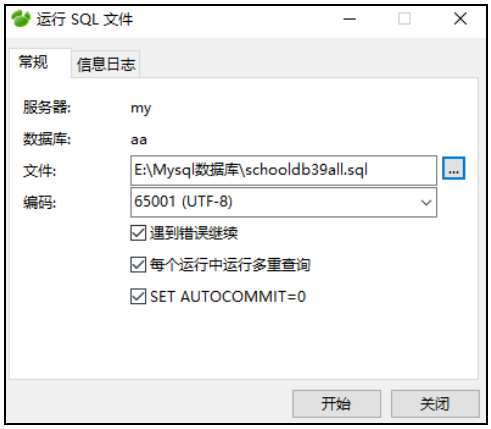


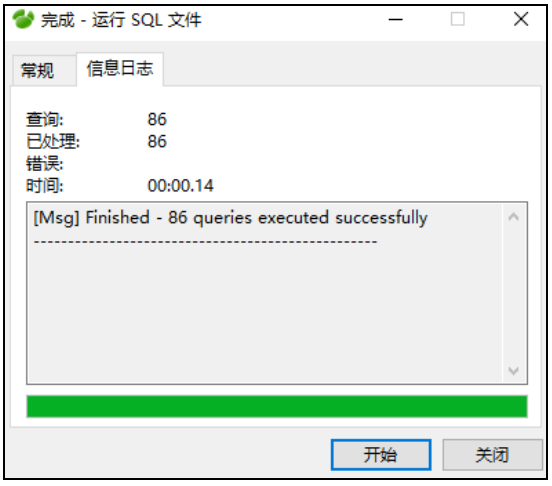
（5）数据库的恢复


Navicat for MySQL
文件
查看
收藏夹
工具
窗口
连接
用户
表
my
aa
打开数据库
关闭数据库
新建数据库...
删除数据库
数据库属性...
命令列界面...
运行 SQL 文件...
转储 SQL 文件
打印数据库...
数据传输...
逆向数据库到模型...
在数据库中查找...
刷新


运行 SQL 文件
常规
信息日志
服务器:
my
数据库:
aa
文件:
编码:
65001 (UTF-8)
遇到错误继续
每个运行中运行多重查询
SET AUTOCOMMIT=0
开始
关闭


名称
修改日期
类型
MySQL数据库设计与应用-张成叔
2023/12/5 9:16
文件夹
mysql相关软件
2023/9/19 12:25
文件夹
schooldb
2023/9/19 12:18
文件夹
schooldb80
2023/9/19 12:18
文件夹
教学文件
2024/6/26 15:07
文件夹
期末考核
2023/9/19 12:16
文件夹
实训
2024/9/9 15:51
文件夹
schooldb39all.sql
2022/3/9 23:08
SQL 文件
schooldb612.sql
SQL 文件
类型: SQL 文件
大小: 10.2 KB
修改日期: 2022/3/9 23:08
schooldb39all.sql
SQL 脚本文件 (*.sql)
打开(O)
取消

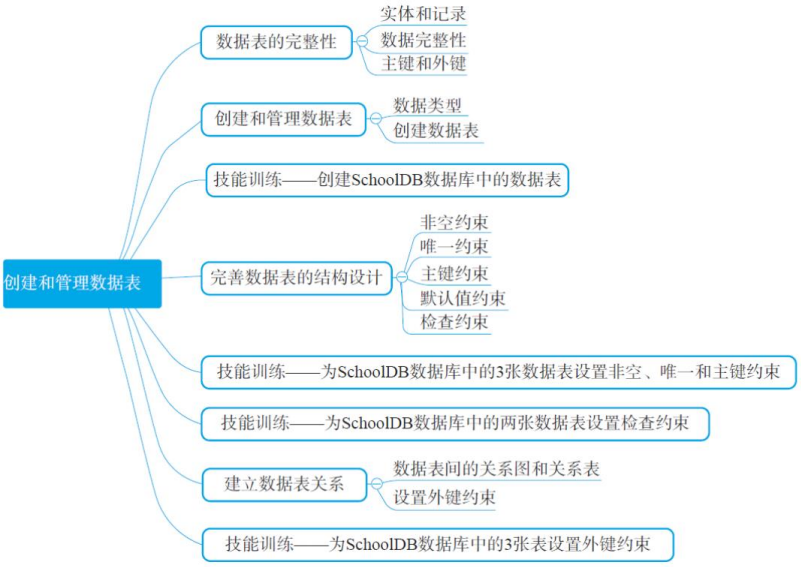

运行 SQL 文件
常规
信息日志
服务器:
my
数据库:
aa
文件:
E:\Mysql\数据库\schooldb39all.sql
编码:
65001 (UTF-8)
遇到错误继续
每个运行中运行多重查询
SET AUTOCOMMIT=0
开始
关闭


完成 - 运行 SQL 文件
常规
信息日志
查询:
86
已处理:
86
错误:
时间:
00:00.14
[Msg] Finished - 86 queries executed successfully
开始
关闭

课后 反思	

<table>
<tr><td>课题</td><td colspan="3">第 03 章 创建和管理数据表</td></tr>
<tr><td>教
学
目
标</td><td colspan="3">➢ 了解实体和记录的概念
➢ 理解数据表的结构
➢ 会为字段选择合适的数据类型
➢ 理解数据完整性和约束的作用
➢ 掌握表的创建
➢ 掌握常用约束的创建
➢ 掌握表的管理</td></tr>
<tr><td>重点难点</td><td colspan="3">➢ 数据完整性概念及每种约束在数据完整性中的作用
➢ 约束的创建
➢ 数据表之间关系的创建</td></tr>
<tr><td>建议学时</td><td>6</td><td>教学教具</td><td>一体化教学实训室</td></tr>
<tr><td>主要教法</td><td colspan="3">讲授演示法</td></tr>
<tr><td>思政融入</td><td colspan="3">强调“适配性”原则
以“数据类型选择”为例，强调“适配性”原则。如存储用户年龄选用 TINYINT 而非 INT，存储身份证号选用 CHAR (18) 而非 VARCHAR，这种精准选择不仅节省存储空间，更体现了 “资源集约” 的职业素养，如同工程设计中精准选用建材，既保证质量又避免浪费。以 “主键约束” 为例，说明主键作为数据唯一标识，就像公民身份证号一样，是数据有序管理的 “规则基石”，缺失主键会导致数据混乱，凸显 “规则先行” 的重要性。

设置约束时引导学生学生规范做事、严谨做人，做人做事都要讲规矩、守底线。
 
</td></tr>
<tr><td colspan="4">教 学 过 程</td></tr>
</table>



一、数据表的完整性

■ 数据表：存放数据的容器。

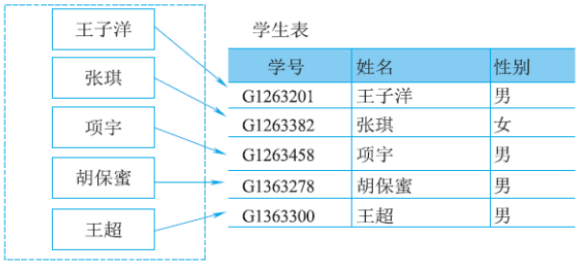
关系数据库中的数据表是二维表格，由行和列组成，每一行称为一条记录，每一列称为一个字段，描述记录的某一特征。

根据项目需求设计数据表

- 一个数据库中要包含多少张数据表
- 一个表应该包含几列
- 各个列要存放什么类型的数据
- 列值是否允许为空

■ 实体和记录

学生实体



学生表

学号	姓名	性别
G1263201	王子洋	男
G1263382	张琪	女
G1263458	项宇	男
G1363278	胡保蜜	男
G1363300	王超	男

数据库中用数据表来存储这种相同类型和格式的实体

- 记录：每一行对应一个实体，通常也叫做一条记录
- 字段：表中的每一列，如学号、姓名等，通常也称为“字段”

■ 数据完整性：指数据的准确性，通过数据库表的约束来实现的。

MySQL 中数据完整性包含四种类型

- 实体完整性：表中的每一条记录反映不同的实体，不能存在相同的记录。通过主键约束、标识列属性、唯一约束或索引实现。
- 域完整性：指表中字段输入值的有效性
- 参照完整性：在输入或删除记录时，保证了两张表中相关联字段的值的一致性
- 用户自定义的完整性：用户自定义完整性用来定义特定的规则

- 主键和外键
  - 主键
    - 简写：Primary Key，PK
    - 主键约束可以实现数据的实体完整性
    - 规范化的数据库中的每张表都必须设置主键约束
    - 主键的字段值必须是唯一的，不允许重复，也不能为空
    - 一张表只能定义一个主键，主键可以是单一字段，也可以是多个字段组合
  - 例如：
    - 在学生表中，设置“学号”为主键，因为在一所学校内部学号是唯一的
    - 在学生表中，不能设置“姓名”为主键，因为同名情况比较普遍
  - 外键
    - 简写：Foreign Key，FK
    - 外键约束可以使一个数据库的多张表之间建立关联
    - 外键约束可以保证数据的参照完整性
    - 例如：在成绩表的学号字段上建立外键约束，关联到学生表的学号字段
      - ✓ 学生表称为“主表”
      - ✓ 成绩表称为“从表”（或称“相关表”）
    - 一个表可以有多个外键
    - 设置了外键约束后，外键的值只能取主表中主键的值或空值

学生表

学号	姓名	性别	…
G1263201	王子洋	男	
G1263382	张琪	女	
G1263458	项宇	男	
G1363278	胡保蜜	男	
G1363300	王超	男	

成绩表

学号	课程	成绩	…
G1263201	大学英语	76	
G1263201	高等数学	88	
G1263382	大学英语	99	
G1263382	高等数学	80	
G1263458	大学英语	91	
G1263458	高等数学	62	

二、创建和管理数据表

- **数据类型：数据的一种特征，决定数据的存储格式**

■ 创建数据表

➢ 命令行模式

CREATE [TEMPORARY] TABLE [IF NOT EXISTS]表名
(
列名 1 数据类型 [约束][索引][注释],
列名 2 数据类型 [约束][索引][注释],
列名 3 数据类型 [约束][索引][注释],
…
列名 n 数据类型 [约束][索引][注释]
)ENGINE=存储引擎;

1、创建 年级表 Grade

列 名	数据类型	长度	字段说明	属性	备注
gradeId	int	4	年级编号	非空	主键
gradeName	varchar(50)	50	年级名称	非空	

（1）命令方式

```
mysql> create table Grade(
  -> gradeId int(4) comment '年级编号',
  -> gradeName varchar(50) comment '年级名称'
  -> );
```

或者 命令合并成一行

```
create table Grade(gradeId int(4) comment '年级编号',gradeName varchar(50) comment '年级名称');
```

（2）查看表结构：DESCRIBE GRADE;

```
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| gradeId   | int(4)       | YES  |     | NULL    |       |
| gradeName | varchar(50)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
```

（3）向 年级表 Grade 添加数据

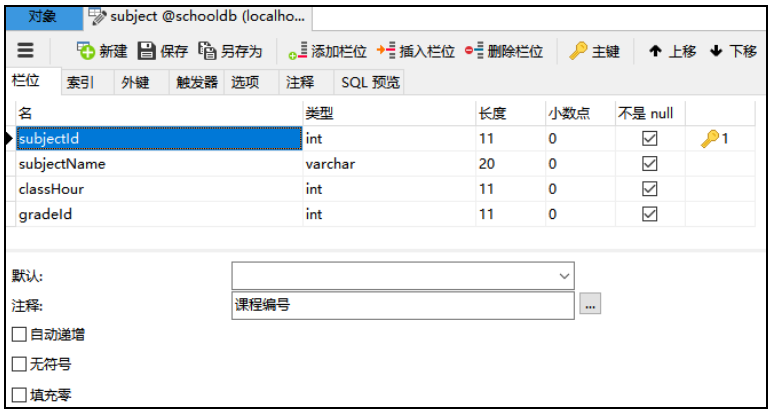
mysql> insert into grade (gradeid,gradename) values(1,'S1');

2、使用 navicat 创建 年级表 Grade1，结构与 Grade 相同，掌握可视化建表的方法。

3、创建课程表 Subject

（1）使用命令行和 navicat 两种方式创建课程表。

列 名	数据类型	长度	字段说明	属性	备注
subjectId	int	4	课程编号	非空	主键
subjectName	nvarchar(20)	20	课程名称		
classHour	int	4	学时		>=0
gradeId	int	4	所属年级		

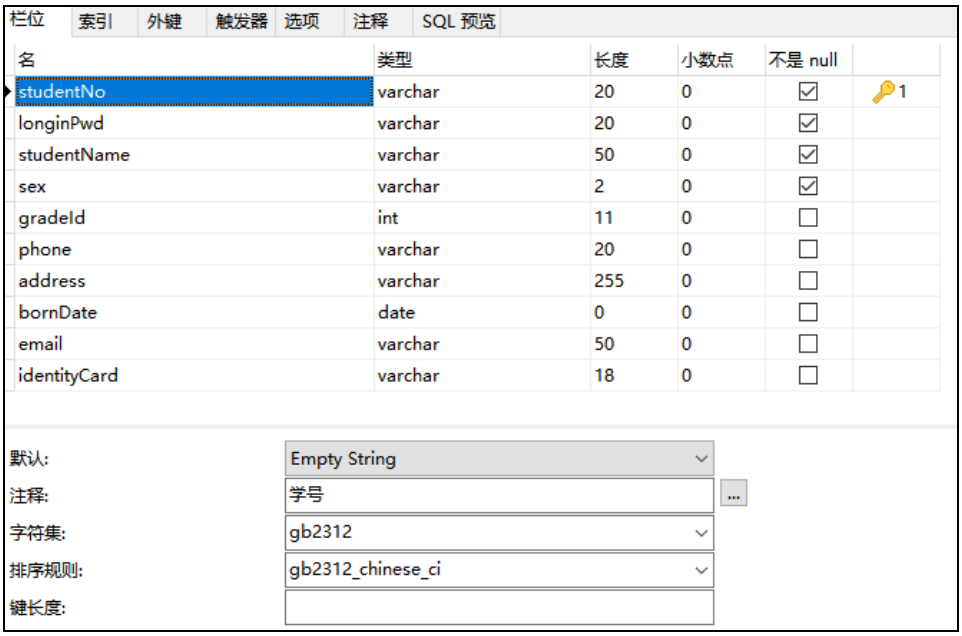


（2）添加数据内容

subjectId	subjectName	classHour	gradeId
1	C 语言程序设计	64	1
2	大学英语	96	1
3	图形图像处理	64	1
4	网页设计	64	2
5	C# 面向对象设计	64	2
6	数据库设计与应用	96	2
7	Android 应用开发	64	3
8	Java 面向对象设计	64	3
9	Web 客户端编程	64	3
10	数据结构与算法	64	4
11	JavaWeb 应用开发	64	4
12	计算机网络基础	64	4

4、创建 学生信息表 Student

列名	数据类型	长度	字段说明	属性	备注
studentNo	varchar	20	学号	非空	主键
loginPwd	varchar	20	密码	非空	
studentName	varchar	50	姓名	非空	
sex	char	2	性别	非空,默认“男”	“男”或“女”
gradeId	int	4	年级编号		
phone	varchar	20	电话		
address	varchar	255	地址	默认值“地址不详”	
bornDate	datetime		出生日期		
email	varchar	50	邮件账号		
identityCard	varchar	18	身份证号	唯一	全国唯一



（1）使用命令方式或 navicat 可视化模式完成数据表创建

（2）设置学号（非空）、密码（非空）、姓名（非空）、性别（非空，默认“男”）、

地址（默认“地址不详”）、身份证号（唯一）属性。

（3）查看表结构：DESCRIBE Student;

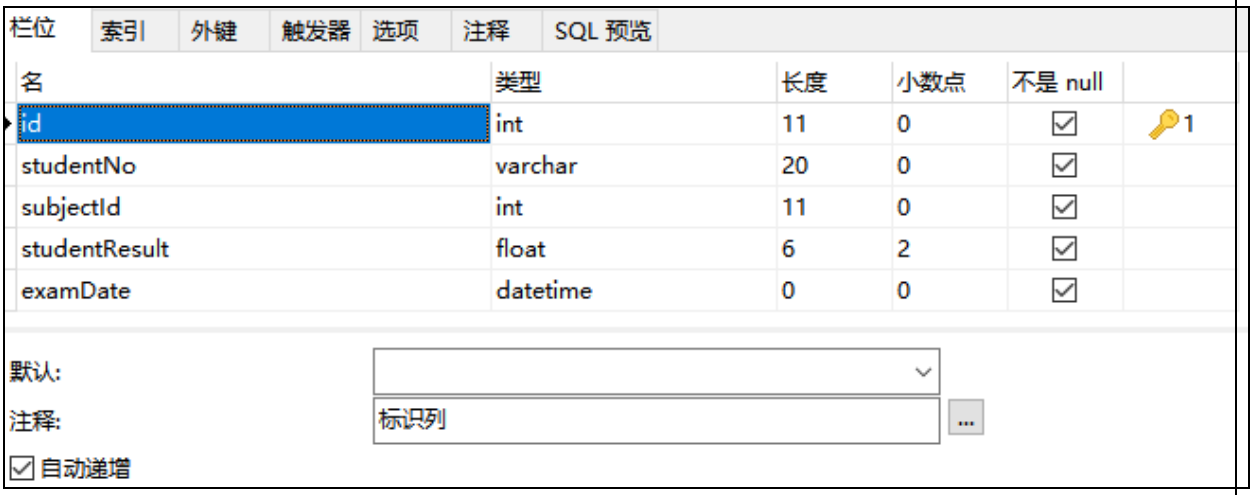
5、创建 成绩表 Result

列名	数据类型	长度	字段说明	属性	备注
id	int	4	标识列	非空	主键
studentNo	varchar	20	学号	非空	
subjectId	int	4	课程编号	非空	
studentResult	float	(6,2)	考试成绩	非空	0~100
examDate	datetime		考试日期	非空	

（1）使用命令行和 navicat 两种方式创建成绩表。

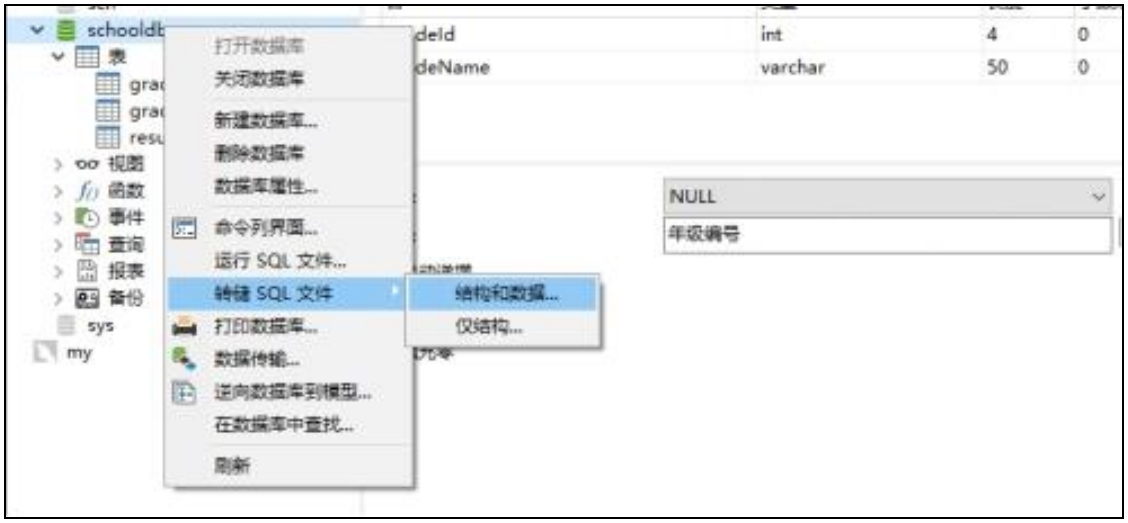
（2）设置学号为主键，所有字段都为（非空）属性。

（3）查看表结构：DESCRIBE Result;

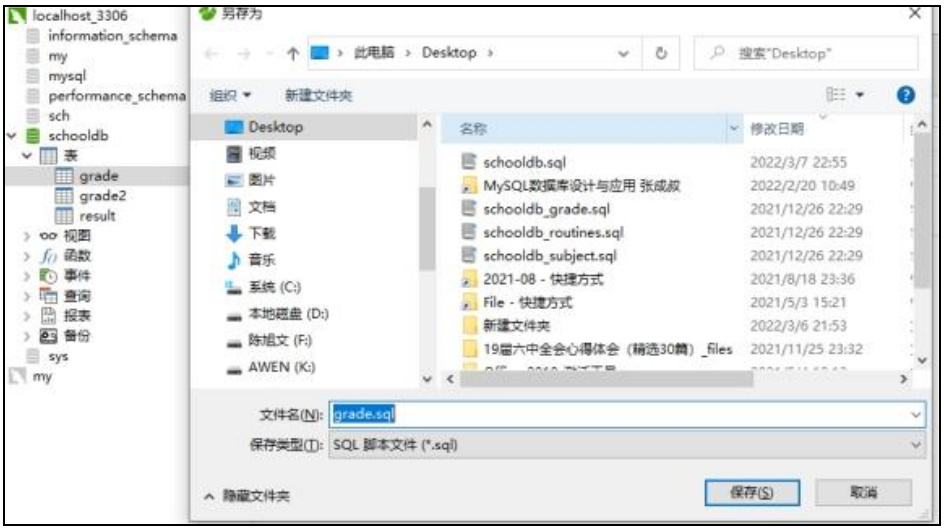


4、数据库和数据表的导出

（1）导出数据库 schooldb：schooldb.sql



（2）导出各数据表：grade：grade.sql 、Subject.sql、Student.sql、Result.sql

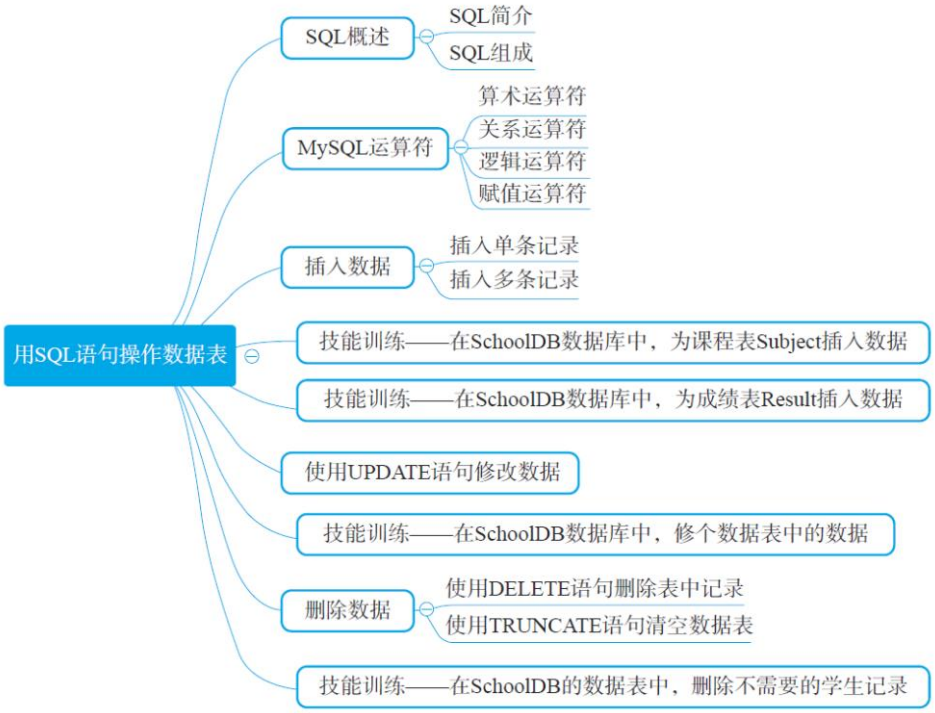


注意：

将导出的数据库和数据表文件使用 U 盘拷贝，后续实验将采用现有数据进行操作。

课后反思	

<table>
<tr><td>课题</td><td colspan="3">第 04 章 用 SQL 语句操作数据表</td></tr>
<tr><td>教
学
目
标</td><td colspan="3">➢ 了解 SQL 的概念
➢ 掌握 SQL 中运算符的规则和使用
➢ 使用 MySQL 命令向表中插入数据
➢ 使用 MySQL 命令更新表中数据
➢ 使用 MySQL 命令删除表中数据</td></tr>
<tr><td>重点难点</td><td colspan="3">➢ MySQL 系列命令的规则和应用
➢ 理解数据表的约束和数据表之间的关系
➢ 通过数据来验证约束是否有效</td></tr>
<tr><td>建议学时</td><td>6</td><td>教学教具</td><td>一体化教学实训室</td></tr>
<tr><td>主要教法</td><td colspan="3">讲授演示法</td></tr>
<tr><td>思政融入</td><td colspan="3">坚守 “真实性” 底线
以 “数据录入真实性” 为核心，结合某高校学籍管理员录入学生信息时伪造学历数据被追责的案例，说明数据新增是 “数据生命周期的源头”，源头造假会引发连锁错误，如同建筑地基不实会导致楼宇坍塌。作为数据录入者，需秉持 “诚信履职” 的态度，对每一个录入的值负责，这是遵守《数据安全法》中 “数据真实、准确” 要求的直接体现。
敬畏数据、守护价值
以某互联网公司因员工误执行 DELETE FROM user;（无 WHERE 条件）导致用户数据全部丢失，公司面临巨额赔偿的案例，敲响 “操作审慎” 的警钟。延伸讲解 “删除审批流程”：在关键业务系统中，数据删除需经过申请、审核、备份、执行、验证五个环节，如同销毁重要档案需履行审批手续，这是 “敬畏数据、守护价值” 的体现。</td></tr>
<tr><td colspan="4">教 学 过 程</td></tr>
</table>



一、SQL 概述

一门针对数据库而言的语言，可以创建数据库、创建数据表、添加约束等，可以针对数据库的数据进行增、删、改、查等操作，可以创建视图、存储过程等，可以赋予用户权限等。

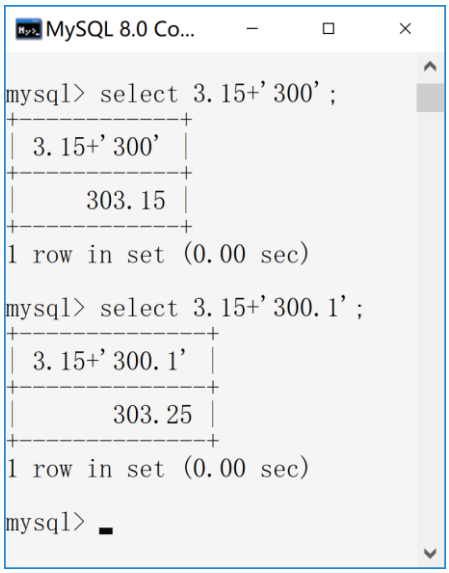
- **SQL 语言简介：结构化查询语言（Structured Query Language）**

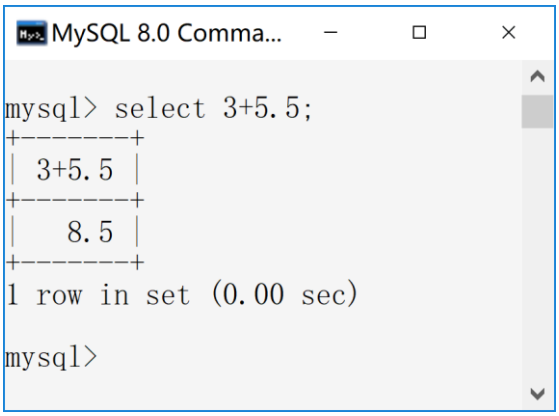
SQL 语言是一种数据库查询和程序设计语言，用于存取、查询、更新数据，管理关系数据库系统。SQL 结构简洁、功能强大、应用广泛：Oracle、DB2、Microsoft SQL Server、MySQL、Access 等数据库支持。

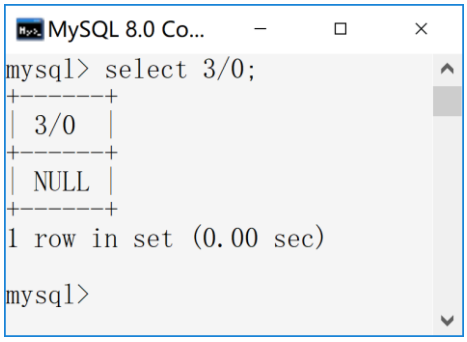
- **SQL 组成**
  - 数据定义语言（DDL，Data Definition Language）：用于对数据库及各种对象进行创建（CREATE）、删除（DROP）、修改（ALTER）等操作，数据库对象包括：表、默认约束、规则、视图、触发器、存储过程等。
  - 数据操纵语言（DML，Data Manipulation Language）：用于对数据库中的数据进行操作，包括：查询（SEIECT）、插入（INSERT）、修改（UPDATE）、删除（DEIETE）等语句。
  - 数据控制语言（DCL，Data Control Language）：包括：授权（GRANT）、收回（REVOKE）等语句。
  - MySQL 增加的语言元素：为了方便用户编程而增加的语言元素，包括：常量、变量、运算符、函数、流程控制语句和注释等。

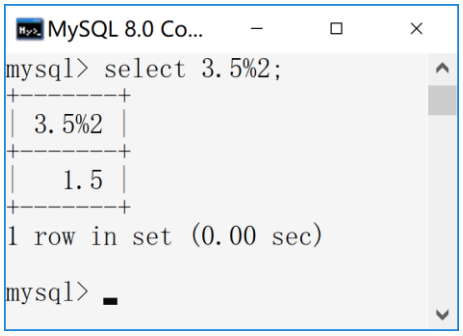
- **MySQL 运算符**
  - 算术运算符：加、减、乘、除、求余数

运算符	说　明
+	加运算,求两个数或表达式相加的和
-	减运算,求两个数或表达式相减的差
*	乘运算,求两个数或表达式相乘的积
/	除运算,求两个数或表达式相除的商
DIV	除运算,求两个数或表达式相除的商
%	取余数运算,求两个数或表达式相除的余数
MOD	取余数运算,求两个数或表达式相除的余数









- 关系运算符：等于、大于、小于、大于或等于、小于或等于、不等于，是否为空、在某个区间

运算符	说　　明
=	等于,例如,age=23
>	大于,例如,price>100
<	小于
>=	大于或等于
<=	小于或等于
<>	不等于
!=	不等于(非 SQL 92 标准)
is null	字段是否为空
is not null	字段是否不为空
between…and	在某个区间

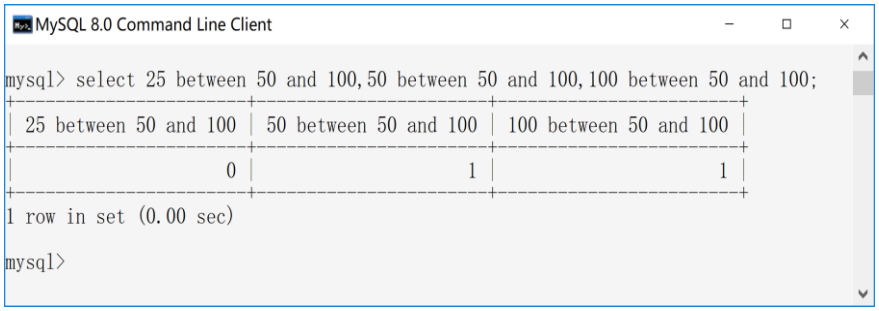
关于空的表达：

判断一条记录的某个字段是否为空，不能使用逻辑运算符等于“=”，应该使用“is null”或者“is not null”，例如：判断学生信息表中某条记录的字段 email 的值是否为空：

正确表达式：email is null　　错误表达式：email = null

between…and 的应用

between…and 表示在某个范围内。如果在范围内结果为 1，否则结果为 0，在计算时含等于，如 between 50 and 100，包含 50 和 100



■ 逻辑运算符

（1）与运算规则：“全真为真，有假即假”。真—非 0 的值，假—0. 例如：表达式“1 and 5”的值为 1.“有假即假”，两边的两个运算符只要有一个运算数是假，与运算的结果就假，例如：表达式“0 and 5”的值为 0.

- **插入数据**

插入数据是对数据表进行后续操作的基础，插入数据以行为单位。

插入数据的方式：

- 可以一次插入一行记录

- 可以一次插入多行记录
- 可以将 SELECT 语句的查询结果批量插入到数据表中

语法：

INSERT [INTO] 表名 [(列名 1,列名 2,…)]
VALUES({表达式|DEFAULT}，…),(. . .),…,;

- 表名：即插入数据表的名称，表名前的“INTO”可以省略不写
- 列名：如果要给所有列都插入数据，列名可以省略
- VALUES 子句：包含各列需要插入的数据清单

按照以上顺序输入数据。

1、数据输入

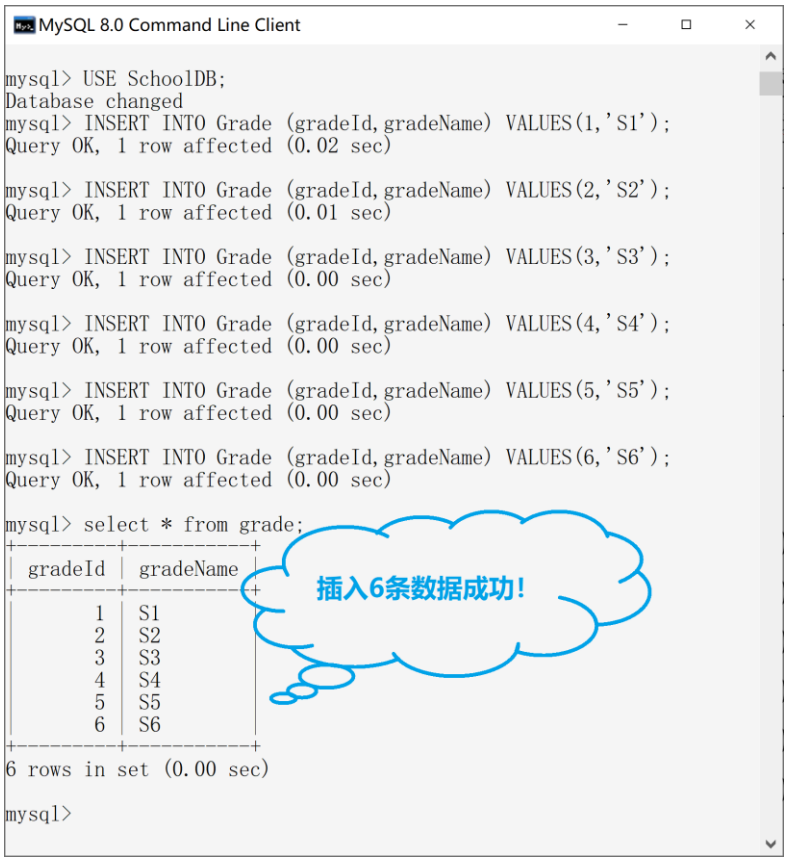
（1）在 navicat 中直接输入。

（2）命令方式：

先选择数据库（将以下命令在 navicat 中新建“查询”然后执行）

年级表：

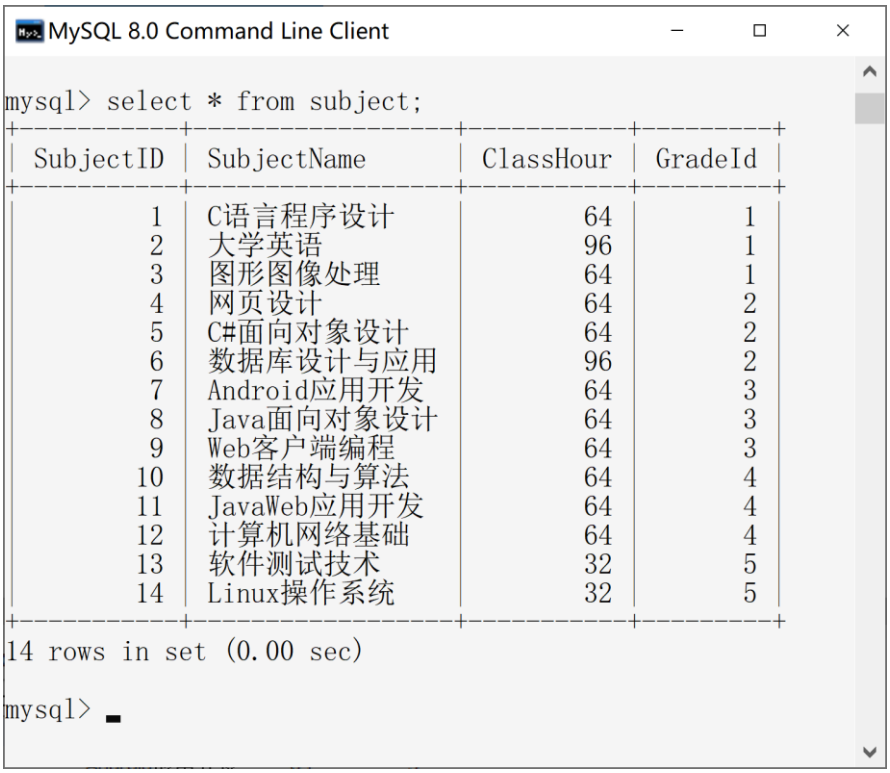
```
use schooldb;
INSERT INTO `grade` VALUES ('1', 'S1');
INSERT INTO `grade` VALUES ('2', 'S2');
INSERT INTO `grade` VALUES ('3', 'S3');
INSERT INTO `grade` VALUES ('4', 'S4');
INSERT INTO `grade` VALUES ('5', 'S5');
INSERT INTO `grade` VALUES ('6', 'S6');
```



课程表：

use schooldb；

INSERT INTO `subject` VALUES ('1', 'C 语言程序设计', '64', '1');
INSERT INTO `subject` VALUES ('2', '大学英语', '96', '1');
INSERT INTO `subject` VALUES ('3', '图形图像设计', '64', '1');
INSERT INTO `subject` VALUES ('4', '网页设计', '64', '2');
INSERT INTO `subject` VALUES ('5', 'C#面向对象设计', '64', '2');
INSERT INTO `subject` VALUES ('6', '数据库设计与应用', '96', '2');
INSERT INTO `subject` VALUES ('7', 'Android 应用开发', '64', '3');
INSERT INTO `subject` VALUES ('8', 'Java 面向对象设计', '64', '3');
INSERT INTO `subject` VALUES ('9', 'Web 客户端编程', '64', '3');
INSERT INTO `subject` VALUES ('10', '数据结构与算法', '64', '3');
INSERT INTO `subject` VALUES ('11', 'JavaWeb 应用开发', '64', '4');
INSERT INTO `subject` VALUES ('12', '计算机网络基础', '64', '4');
INSERT INTO `subject` VALUES ('13', '软件测试技术', '32', '5');
INSERT INTO `subject` VALUES ('14', 'Linux 操作系统', '32', '5');



学生表：

use schooldb；

INSERT INTO `student` VALUES ('G1263201', '1', '王子洋', '男', '5', '18655290000', '安徽省蚌埠市', '1993-08-07', 'wzy@163.com', '340423199308070000');

INSERT INTO `student` VALUES ('G1263382', '1', '张琪', '女', '5', '15678090000', '安徽省合肥市', '1993-05-07', 'zhangqi@126.com', '340104199305070000');

INSERT INTO `student` VALUES ('G1263458', '1', '项宇', '男', '5', '18298000000', '地址不详', '1992-12-10', 'xiangyu@163.com', '340881199212100000');

INSERT INTO `student` VALUES ('G1363278', '1', '胡保蜜', '男', '3', '18965000000', '北京市通州区', '1993-06-29', null, '346542199306290000');

INSERT INTO `student` VALUES ('G1363300', '1', '王超', '男', '3', '18123560000', '上海市闵行区', '1993-04-30', 'wangchao@126.com', '340409199304300000');

INSERT INTO `student` VALUES ('G1363301', '1', '党志鹏', '男', '3', '15876550000', '安徽省芜湖市', '1994-12-20', 'dzp@sohu.com', '456765199412200000');

INSERT INTO `student` VALUES ('G1363302', '1', '胡仲友', '男', '3', '15032450000', '安徽省淮南市', '1994-06-13', '12454344@qq.com', '340043199406130000');

INSERT INTO `student` VALUES ('G1363303', '1', '朱晓燕', '女', '3', '15155670000', '安徽省安庆市', '1994-04-18', 'yanyan@163.com', null);

INSERT INTO `student` VALUES ('G1463337', '1', '高伟', '男', '1', '18390870000', '山东省济南市', '1995-06-07', null, '450504199506070000');

INSERT INTO `student` VALUES ('G1463342', '1', '胡俊文', '男', '1', '13976870000', '河南省郑州市', '1995-04-20', null, '340408199504200000');

INSERT INTO `student` VALUES ('G1463354', '1', '陈大伟', '男', '1', '15067340000', '四川省成都市', '1995-08-23', 'wangkuan@163.com', '340422199508230000');

INSERT INTO `student` VALUES ('G1463358', '1', '温海南', '男', '1', '18028760000', '地址不详', '1995-01-30', null, '450560199501300000');

INSERT INTO `student` VALUES ('G1463383', '1', '钱嫣然', '女', '1', '18656430000', '安徽省潜山市',

'1994-01-14', 'yanran@126.com', '340408199401140000');

INSERT INTO `student` VALUES ('G1463388', '1', '卫丹丹', '女', '1', '15134870000', '广东省深圳市', '1995-04-17', null, '340411199504170000');

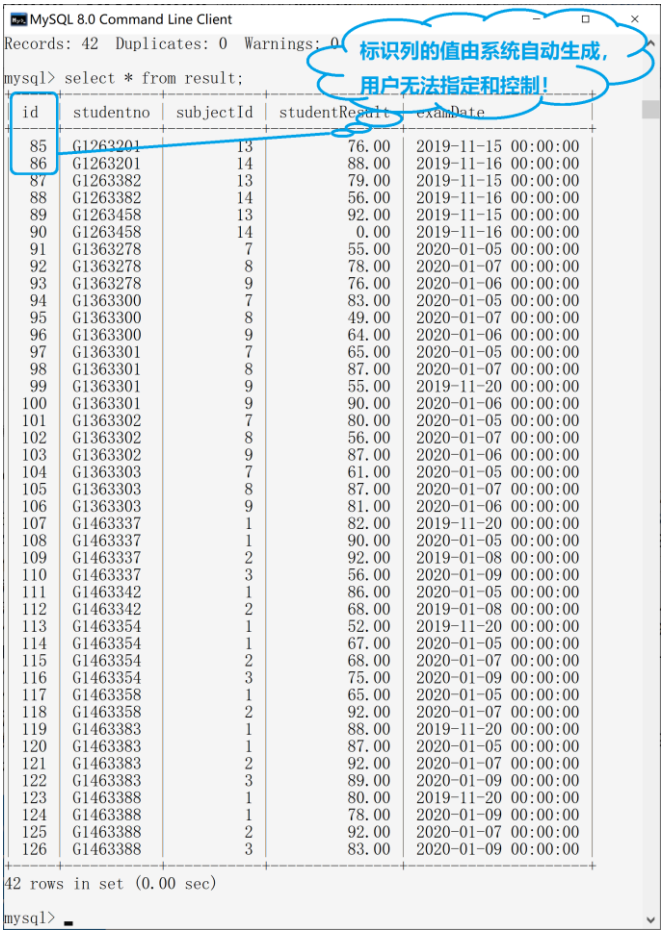
学号	密码	姓名	性别	年级	电话	地址	出生日期	邮箱	身份证号
G1263201	1	王子洋	男	5	18655290000	安徽省蚌埠市	1993/8/7	wzy@163.com	340423199308070000
G1263382	1	张琪	女	5	15678090000	安徽省合肥市	1993/5/7	zhangqi@126.com	340104199305070000
G1263458	1	项宇	男	5	18298000000		1992/12/10	xiangyu@163.com	340881199212100000
G1363278	1	胡保蜜	男	3	18965000000	北京市通州区	1993/6/29		346542199306290000
G1363300	1	王超	男	3	18123560000	上海市闵行区	1993/4/30	wangchao@126.com	340409199304300000
G1363301	1	党志鹏	男	3	15876550000	安徽省芜湖市	1994/12/20	dzp@suho.com	456765199412200000
G1363302	1	胡仲友	男	3	15032450000	安徽省淮南市	1994/6/13	12454344@qq.com	340043199406130000
G1363303	1	朱晓燕	女	3	15155670000	安徽省安庆市	1994/4/18	yanyan@163.com	
G1463337	1	高伟	男	1	18390870000	山东省济南市	1995/6/7		450504199506070000
G1463342	1	胡俊文	男	1	13976870000	河南省郑州市	1995/4/20		340408199504200000
G1463354	1	陈大伟	男	1	15067340000	四川省成都市	1995/8/23	wangkuan@163.com	340422199508230000
G1463358	1	温海南	男	1	18028760000		1995/1/30		450560199501300000
G1463383	1	钱嫣然	女	1	18656430000	安徽省潜山市	1994/1/14	yanran@126.com	340408199401140000
G1463388	1	卫丹丹	女	1	15134870000	广东省深圳市	1995/4/17		340411199504170000

成绩表：

use schooldb；

INSERT INTO `result` VALUES ('85', 'G1263201', '13', '76.00', '2019-11-15 00:00:00');
INSERT INTO `result` VALUES ('86', 'G1263201', '14', '88.00', '2019-11-16 00:00:00');
INSERT INTO `result` VALUES ('87', 'G1263382', '13', '79.00', '2019-11-15 00:00:00');
INSERT INTO `result` VALUES ('88', 'G1263382', '14', '56.00', '2019-11-16 00:00:00');
INSERT INTO `result` VALUES ('89', 'G1263458', '13', '92.00', '2019-11-15 00:00:00');
INSERT INTO `result` VALUES ('90', 'G1263458', '14', '0.00', '2019-11-16 00:00:00');
INSERT INTO `result` VALUES ('91', 'G1363278', '7', '55.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('92', 'G1363278', '8', '78.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('93', 'G1363278', '9', '76.00', '2020-01-06 00:00:00');
INSERT INTO `result` VALUES ('94', 'G1363300', '7', '83.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('95', 'G1363300', '8', '49.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('96', 'G1363300', '9', '64.00', '2020-01-06 00:00:00');
INSERT INTO `result` VALUES ('97', 'G1363301', '7', '65.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('98', 'G1363301', '8', '87.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('99', 'G1363301', '9', '55.00', '2019-11-20 00:00:00');
INSERT INTO `result` VALUES ('100', 'G1363301', '9', '90.00', '2020-01-06 00:00:00');
INSERT INTO `result` VALUES ('101', 'G1363302', '7', '80.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('102', 'G1363302', '8', '56.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('103', 'G1363302', '9', '87.00', '2020-01-06 00:00:00');
INSERT INTO `result` VALUES ('104', 'G1363303', '7', '61.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('105', 'G1363303', '8', '87.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('106', 'G1363303', '9', '81.00', '2020-01-06 00:00:00');
INSERT INTO `result` VALUES ('107', 'G1463337', '1', '82.00', '2019-11-20 00:00:00');
INSERT INTO `result` VALUES ('108', 'G1463337', '1', '90.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('109', 'G1463337', '2', '92.00', '2019-01-08 00:00:00');
INSERT INTO `result` VALUES ('110', 'G1463337', '3', '56.00', '2020-01-09 00:00:00');

```
INSERT INTO `result` VALUES ('111', 'G1463342', '1', '86.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('112', 'G1463342', '2', '68.00', '2019-01-08 00:00:00');
INSERT INTO `result` VALUES ('113', 'G1463354', '1', '52.00', '2019-11-20 00:00:00');
INSERT INTO `result` VALUES ('114', 'G1463354', '1', '67.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('115', 'G1463354', '2', '68.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('116', 'G1463354', '3', '75.00', '2020-01-09 00:00:00');
INSERT INTO `result` VALUES ('117', 'G1463358', '1', '65.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('118', 'G1463358', '2', '92.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('119', 'G1463383', '1', '88.00', '2019-11-20 00:00:00');
INSERT INTO `result` VALUES ('120', 'G1463383', '1', '87.00', '2020-01-05 00:00:00');
INSERT INTO `result` VALUES ('121', 'G1463383', '2', '92.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('122', 'G1463383', '3', '89.00', '2020-01-09 00:00:00');
INSERT INTO `result` VALUES ('123', 'G1463388', '1', '80.00', '2019-11-20 00:00:00');
INSERT INTO `result` VALUES ('124', 'G1463388', '1', '78.00', '2020-01-09 00:00:00');
INSERT INTO `result` VALUES ('125', 'G1463388', '2', '92.00', '2020-01-07 00:00:00');
INSERT INTO `result` VALUES ('126', 'G1463388', '3', '83.00', '2020-01-09 00:00:00');
```



注意：完成以上表格数据插入后，使用 navicat 的数据导出功能，将各个表格独立存放为一个 sql 文件。

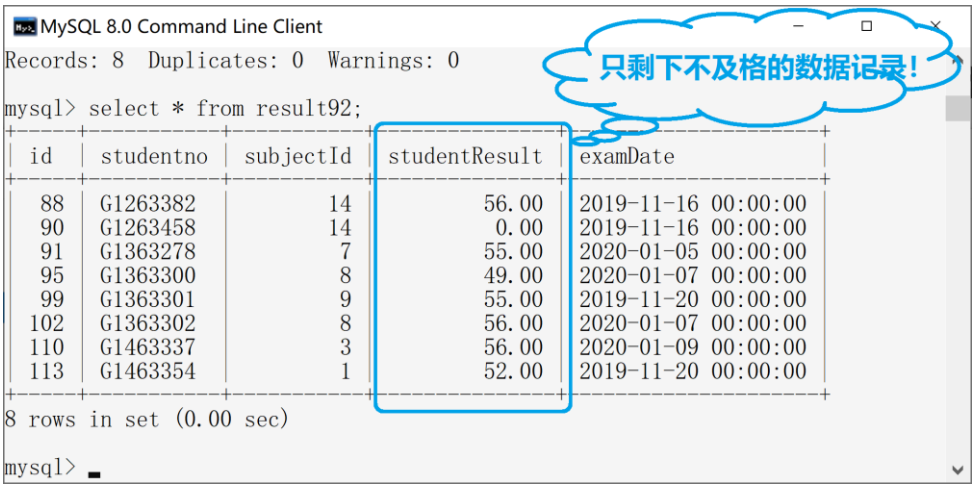
【演示示例4-3】将查询结果插入到数据表中

需求分析

- 数据库SchoolDB中
- 新建一张新表Result92，其表结构与Result相同
- 用来存储Result表中成绩不及格的信息

代码分析

```
USE SchoolDB;
CREATE TABLE Result92 LIKE Result;
INSERT INTO Result92
   SELECT * FROM Result WHERE studentResult<60;
SELECT * FROM Result92;
```




```
Records: 8  Duplicates: 0  Warnings: 0

mysql> select * from result92;
+-----+-----------+-----------+---------------+---------------------+
| id  | studentno | subjectId | studentResult | examDate            |
+-----+-----------+-----------+---------------+---------------------+
|  88 | G1263382  |        14 |         56.00 | 2019-11-16 00:00:00 |
|  90 | G1263458  |        14 |          0.00 | 2019-11-16 00:00:00 |
|  91 | G1363278  |         7 |         55.00 | 2020-01-05 00:00:00 |
|  95 | G1363300  |         8 |         49.00 | 2020-01-07 00:00:00 |
|  99 | G1363301  |         9 |         55.00 | 2019-11-20 00:00:00 |
| 102 | G1363302  |         8 |         56.00 | 2020-01-07 00:00:00 |
| 110 | G1463337  |         3 |         56.00 | 2020-01-09 00:00:00 |
| 113 | G1463354  |         1 |         52.00 | 2019-11-20 00:00:00 |
+-----+-----------+-----------+---------------+---------------------+
8 rows in set (0.00 sec)

mysql>
```

四、使用 UPDATE 语句修改数据

语法

```
UPDATE 表名
  SET 列名 1=表达式 1,列名 2=表达式 2,…
  [WHERE 更新条件]
```

分析

- 表名
  - 即需要修改数据的数据表名称

- SET子句
  - 根据WHERE子句中指定的条件对符合条件的数据进行修改
  - 若语句中不设定WHERE子句，则更新表中所有行
  - 指定数据修改的内容
  - 可以多个数据列的更新值，多个数据列之间以逗号“,”分隔开
- 列名和表达式列表
  - 列名1、列名2...为要修改列值的列名
  - 表达式1、表达式2...可以是常量、变量或表达式
  - 可以同时修改所在数据行的多个列值，中间用逗号隔开

【演示示例4-4】更新表中数据

需求分析

- 数据库SchoolDB中
- 复制学生信息Student的结构和数据得到新表Student93
- 将Student93中所有邮箱地址为空的学生的邮箱都改为“未注册邮箱@163.com”

代码分析

```
USE SchoolDB;
CREATE TABLE Student93 LIKE Student;
INSERT INTO Student93 SELECT * FROM Student;
SELECT * FROM Student93 WHERE email is null;
UPDATE Student93 SET email='未注册邮箱@163.com'
  WHERE email is null;
SELECT * FROM Student93 WHERE email='未注册邮箱@163.com';
```

studentno	loginPwd	studentName	sex	gradeId	phone	address	bornDate	email	identityCard
G1363278	1	胡保蜜	男	3	18965000000	北京市通州区	1993-06-29	NULL	346542199306290000
G1463337	1	高伟	男	1	18390870000	安徽省灵璧县	1995-06-07	NULL	450504199506070000
G1463342	1	胡俊文	男	1	13976870000	安徽省定远县	1995-04-20	NULL	340408199504200000
G1463358	1	温海南	男	1	18028760000	地址不详	1995-01-30	NULL	450560199501300000
G1463388	1	卫丹丹	女	1	15134870000	安徽省凤阳县	1995-04-17	NULL	340411199504170000

studentno	loginPwd	studentName	sex	gradeId	phone	address	bornDate	email	identityCard
G1363278	1	胡保蜜	男	3	18965000000	北京市通州区	1993-06-29	未注册邮箱@163.com	346542199306290000
G1463337	1	高伟	男	1	18390870000	安徽省灵璧县	1995-06-07	未注册邮箱@163.com	450504199506070000
G1463342	1	胡俊文	男	1	13976870000	安徽省定远县	1995-04-20	未注册邮箱@163.com	340408199504200000
G1463358	1	温海南	男	1	18028760000	地址不详	1995-01-30	未注册邮箱@163.com	450560199501300000
G1463388	1	卫丹丹	女	1	15134870000	安徽省凤阳县	1995-04-17	未注册邮箱@163.com	340411199504170000

【技能训练4-3 】修改数据表中数据

需求

- 在SchoolDB数据库中
  - 修改学号为G1363300的学生地址为“山东省济南市文化路2号院”
  - 修改学号为G1363301的学生的所属年级为2
  - 修改“大学英语”课程的学时数为55
  - 将2020年1月7日考试的“Java面向对象设计”课程分数低于60分的学生全部提高5分
  - 将学号为G1363300的学生在2020年1月5日考试的“Android应用开发”课程的分数修改为80
  - 将邮件账号为空的学生的邮件账号统一修改为“未知@”

关键点分析

- 在修改数据之前，先打开数据库
- 修改学生的住址，参考语句

```
UPDATE Student
    SET Address='山东省济南市文化路 2 号院'
    WHERE StudentNo='G1363300';
```

- 修改G1363300的学生“Android应用开发”课程的分数
  - 通过“SELECT”语句查询课程表Subject
  - 查询得到课程“Java面向对象设计”的课程编号为8
  - 再使用“UPDATE”语句对Result表进行成绩的更新
  - 修改前后使用SELECT语句查询结果，对比分析

- 修改G1363300的学生“Android应用开发”课程的分数

```
SELECT * FROM Subject;
SELECT * FROM Result
   WHERE examDate='2020-1-7' AND subjectId=8 AND studentResult<60;
UPDATE Result SET studentResult=studentResult+5
   WHERE examDate='2020-1-7' AND subjectId=8 AND studentResult<60;
SELECT * FROM Result
   WHERE examDate='2020-1-7' AND subjectId=8 AND studentResult<65;
```

- 邮箱为空的表达方式

```
UPDATE Student SET email='未知@' WHERE email IS NULL
```

重点分析

- 多个条件的组合
  - 有学号、日期、课程的限制，又有分数的限制
  - 需要使用逻辑与运算符“AND”来连接多个条件

```
WHERE examDate='2020-1-7' AND subjectId=8 AND studentResult<60
```

五、删除数据

使用DELETE语句删除表中记录

语法

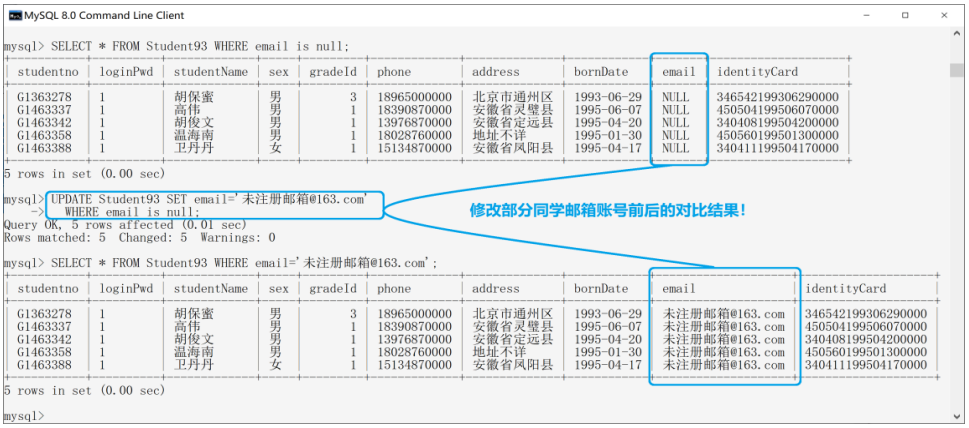
```
DELETE FROM 表名
[WHERE 条件]
```

分析

- FROM子句
  - 指定需要删除数据的数据表名称
- WHERE子句
  - 指定删除的条件
  - 若省略WHERE于句则表示删除该表的所有行

- 外键约束
  - 要确保要删除的数据记录没有被其它子表引用
  - 如：要删除学生信息表Student中的学生记录，要确保该学生在成绩表Reuslt中没有成绩

【演示示例4-5】
删除姓名为“项宇”的学生信息



使用TRUNCATE语句清空数据表

语法

```
TRUNCATE TABLE 表名;
```

分析

- 删除指定表中的所有数据，即清除表数据
- ANTO INCREAMENT计数器被重新设置为该列的初始值
- 对于参与了索引和视图的表，不能使用TRUNCATE TABLE语句，必须使用DELETE语句

示例

- 例如：清空成绩表Result92的数据

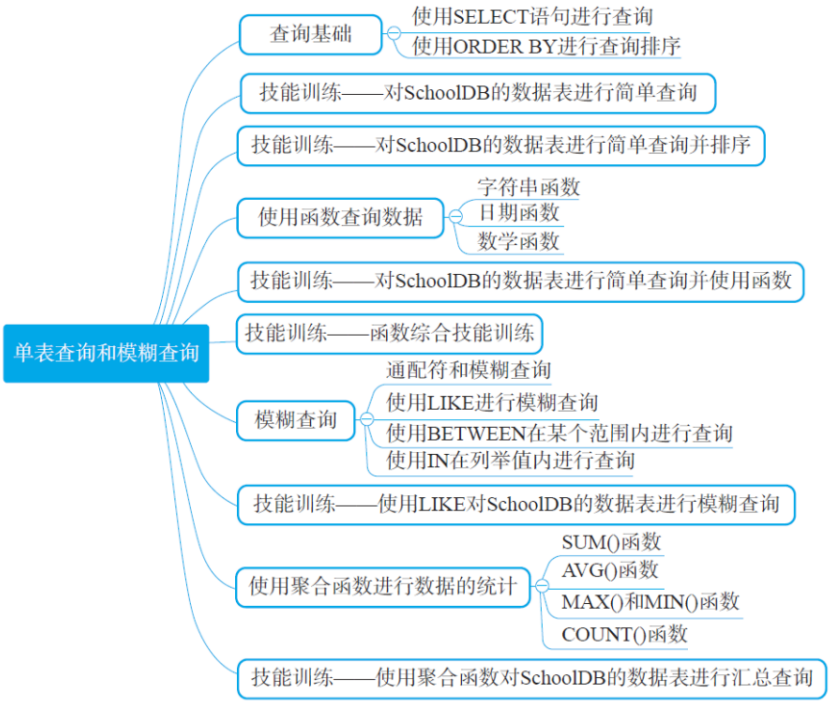
```
TRUNCATE TABLE Result92;
```

- 建议：清楚数据表前复制一份
  - 一旦被清楚，无法恢复数据

课后反思	

<table>
<tr><td>课题</td><td colspan="3">第 5 章 单表查询和模糊查询</td></tr>
<tr><td>教
学
目
标</td><td colspan="3">➢ 理解查询处理的机制
➢ 掌握常用的系统函数
➢ 熟练使用 SELECT 语句单表查询并实现排序
➢ 使用 LIKE、BETWEEN、IN 进行模糊查询
➢ 使用聚合函数统计和汇总查询信息</td></tr>
<tr><td>重点难点</td><td colspan="3">➢ 查询条件的构造
➢ 使用表达式、运算符和函数解决实际问题
➢ 灵活应用聚合函数对成绩信息进行汇总统计</td></tr>
<tr><td>建议学时</td><td>9</td><td>教学教具</td><td>一体化教学实训室</td></tr>
<tr><td>主要教法</td><td colspan="3">讲授演示法</td></tr>
<tr><td>思政融入</td><td colspan="3">聚焦 “模糊查询的隐私风险控制”
某企业 HR 因使用 LIKE '%138%'无限制查询员工手机号，导致敏感信息在报表中泄露的案例，强调 “模糊查询不是无边界查询”。讲解 “脱敏 + 范围约束” 的双重防护策略，演示合规的模糊查询。
以 “去重与排序的必要性” 为例，结合某统计局数据公示案例 —— 因未对重复统计的企业数据去重，导致行业产值虚高，引发公众质疑。这说明查询结果的规范处理是 “数据公信力” 的保障，如同新闻报道需核实信息准确性，数据呈现也需经过去重、校验等环节，体现 “对数据负责、对公众负责” 的职业操守。强调 “LIMIT 的合规应用”：在分页展示用户数据时，LIMIT 0,10 的分页查询可避免一次性暴露大量数据，降低隐私泄露风险。</td></tr>
</table>

教　学　过　程



5.1　查询基础

5.1.1　使用 SELECT 语句进行查询

```
SELECT *|字段列表
    FROM  表名
    [WHERE  查询条件]
    [ORDER BY  排序的列名  [ASC|DESC];
```

■ SELECT 子句

指定要查看的列，即字段，即列出查询结果中要显示的字段名

字段列表可以是多个列名或表达式，之间用逗号间隔

表达式可以是列名、函数或常数的列表

如果要查看所有列，使用“*”表示

■ FROM 子句

指定要查询的表，可以是表名或视图名

FROM 中可以是多个表名，它们之间用逗号间隔

- WHERE 子句

用于给出限制查询的条件或多个表的连接条件

根据具体的查询要求进行选择使用

可以为空，即查询表中所有记录

- ORDER BY 子句

用于对查询的结果进行排序

指明排序的列名，可以是多列，使用逗号分隔

排序时可以指定升序或者降序，默认为升序

- 查询课程表 Subject 中所有的课程信息

```
USE SchoolDB;
SELECT * FROM Subject;
```

- 查询 Student 表中所有学生的学号、姓名和邮箱账号信息

```
USE SchoolDB;
SELECT studentNo,studentName,email FROM Student;
```

- 要求使用中文汉字为列标题

```
USE SchoolDB;
SELECT studentNo AS '学号',studentName AS '姓名',email AS '邮箱账号'
  FROM Student;
```

- 查询 Student 表中的前 4 行以及第 6 行开始的 4 行记录，显示学号、姓名和邮箱账号信息，使用中文汉字为列标题

```
USE SchoolDB;
SELECT studentNo AS '学号',studentName AS '姓名',email AS '邮箱账号'
  FROM Student
  LIMIT 4;
SELECT studentNo AS '学号',studentName AS '姓名',email AS '邮箱账号'
  FROM Student
  LIMIT 5,4;
```

- 查询 Student 表中的年级编号，要求删除结果集中的重复记录

```
USE SchoolDB;
SELECT gradeId AS '年级编号' FROM Student;
SELECT DISTINCT gradeId AS '年级编号' FROM Student;
```

- 查询表 Subject 表中所有的课程信息，学校考虑为所有课程增加课时，增加比例的原课时的 10%，只需要显示课程名称、原课时和增加后课时信息

```
USE SchoolDB;
SELECT subjectName AS '课程名称',classHour AS '原课时',
  classHour*1.1 AS '增加 10%后课时'
  FROM Subject;
```

- 查询 Subject 表中学时超过 64 的课程信息，显示课程名称、学时和学期，使用中文汉字标题

```
USE SchoolDB;
SELECT subjectName AS '课程名称',classHour AS '学时',gradeId AS '学期'
  FROM Subject
  WHERE classHour>64;
```

- 查询 Student 表中邮箱账号为空的学生信息，显示学号、姓名和邮箱账号信息，要求使用中文汉字为列标题

```
USE SchoolDB;
SELECT studentNo AS '学号',studentName AS '姓名',email AS '邮箱账号'
  FROM Student
  WHERE email IS NULL;
```

【技能训练 5-1】对数据库 SchoolDB 的，数据表进行简单查询

5.1.2 使用 ORDER BY 子句进行

- 查询排序

```
SELECT *|字段列表
    FROM 表名
    [WHERE 查询条件]
    [ORDER BY 排序的列名 [ASC|DESC];
```

ORDER BY 次序表达式 1 [ASC|DESC][,次序表达式 2 [ASC|DESC]]…

- SELECT 子句

指定要查看的列，即字段，即列出查询结果中要显示的字段名

ORDER BY 次序表达式 1 [ASC|DESC][,次序表达式 2 [ASC|DESC]]…

- 次序表达式

为排序的表达式，可以是列名或者列的别名，也可以是一个表达式

关键字 ASC 表示升序排列，DESC 表示降序排列，默认值为 ASC

排序时，空值（NULL）被认为是最小值，参与排序

可以为多个表达式，且每个可以单独标明升序或降序，用逗号分隔

当有多个次序表达式时，按照从前往后的顺序执行，当表达式 1 的值相等的情况下，再按照表达式 2 的排序，依次类推

演示示例 5-1】查询应用

```
USE SchoolDB;
SELECT subjectId AS '课程编号', subjectName AS '课程名称',classHour AS '学时'
    FROM Subject
    ORDER BY classHour DESC, subjectName ASC;
```

- 有 2 个排序依据

先按照学时降序排列

学时相同再按照课程名称升序排列

- ORDER BY 子句后有 2 个表达式，且第一个表达式后的 **DESC** 不可少，否则就是按照学时升序
- 代码中最后的 **ASC 可以省略**，默认是按照升序排序

5.2 使用函数查询数据

- 函数是完成特定功能的一组 SQL 语句的集合
- 应用：在数据查询中经常会使用函数来实现一些复杂运算
- MySQL 提供了丰富的内置函数：字符串函数、日期和时间函数、聚合函数

5.2.1 字符串函数 109

- 功能

字符串函数主要针对字符型数据进行操作和运算

- 应用

为了实现某些复杂的查询功能，使用字符串函数对字符型数据进行处理，字符串函数中包

含的字符串必须要用单引号括起来。

SELECT ASCII('email');

函数	功能	示例和说明
ASCII(字符串表达式)	返回字符串表达式最左端字符的 ASCII 值。返回值为整数	SELECT ASCII('email'); 说明:返回字母 e 的 ASCH 码值 101
CHAR(整型表达式)	返回整型 ASCII 码转换的字符	SELECT CHAR(65); 说明:返回 ASCII 码值为 65 的字符 A
LENGTH(字符串表达式)	返回字符串表达式的长度	SELECT LENGTH('email'); 说明:返回 email 的长度值 5
LEFT(字符串表达式,长度)	返回从字符串表达式左边开始指定长度个字符	SELECT LEFT('telephone',3); 说明:返回 telephone 左边开始 3 个字符,返回 tel
RIGHT(字符串表达式,长度)	返回从字符串表达式右边开始指定长度个字符	SELECT RIGHT('telephone',3); 说明:返回 telephone 右边开始 3 个字符,返回 one
TRIM(字符串表达式)	返回删除字符串表达式首部和尾部的所有空格,返回值为字符串	SELECT TRIM('□□□I like MySQL!□□□'); 说明:删除 I like MySQL ! 前后的所有空格,返回 I like MySQL ! ,□代表空格,下同
LTRIM(字符串表达式)	删除字符串中前面的空格,返回值为字符串	SELECT LTRIM('□□□I like MySQL !□□□'); 说明:删除 I like MySQL ! 前面的所有空格,返回 I like MySQL ! □□□
RTRIM(字符串表达式)	删除字符串中尾部的空格,返回值为字符串	SELECT RTRIM('□□□I like MySQL!□□□'); 说明:删除 I like MySQL ! 后面的所有空格,返回□□□I like MySQL !

函数	功能	示例和说明
REPLACE(字符串1,字符串2,字符串 3)	用字符串 3 替换字符串 1 中出现的字符串 2,最后返回替换后的字符串	SELECT REPLACE ('Welcome tK BEIJING!','K','o'); 说明:将 Welcome tK BEIJING ! 中出现的 K 替换成 o,返回 Welcome to BEIJING !
SUBSTRING(字符串表达式,指定位置,长度)	返回字符串表达式从指定位置开始指定长度的子串	SELECT SUBSTRING('telephone',5,5); 说明:取字符串 telephone 中从第 5 个字符 p 开始连续 5 个字符构成的子串,返回 phone
CONCAT(字符串 1,字符串 2,…,字符串 n)	返回字符串 1,字符串 2,…,字符串 n 连接起来构成的字符串	SELECT CONCAT('中国','北京'); 说明:字符串中国和北京连接起来,返回中国北京
LOCATE(字符串 1,字符串 2)	返回字符串 1 的第 1 个字符在字符串 2 中的序号,从 1 开始计数	SELECT LOCATE('Happy','I am happy!'); 返回 6

演示示例 5-2】字符串函数的应用

```
USE SchoolDB;
SELECT LEFT(studentName,1) AS '姓',
   SUBSTRING(studentName,2,LENGTH(studentName)-1) AS '名'
   FROM Student
   ORDER BY studentName;
```

- **问题要求按照“姓名”排序，不是按照“姓”排序**
- **两者有一定的区别，按照“姓”排序的代码如下**

```
SELECT LEFT(studentName,1) AS '姓',
  SUBSTRING(studentName,2,LENGTH(studentName)-1) AS '名'
  FROM Student
  ORDER BY LEFT(studentName,1);
```

5.2.2 日期函数

■ 功能

日期和时间函数主要用来处理日期和时间值

日期函数帮助提取日期值中的年月和日，以便分别操作它们

■ 通常不能对日期直接运用数学运算

例如：如果执行一个诸如“当前日期+1”的语句，MySQL 无法理解要增加的是一日、一月还是一年

函数	功能	示例和说明
NOW( )	获得当前的日期和时间，它以 YYYY-MM-DD HH:MM:SS 的格式返回当前的日期和时间	SELECT NOW( ); 返回 2020-08-08 10 :26 :49
CURTIME( )	返回当前时间	SELECT CURTIME( ); 返回 10 :28 :40
CURDATE( )	返回当前日期	SELECT CURDATE( ); 返回 2018-08-25
YEAR( )	分析一个日期值并返回其中关于年的部分	SELECTYEAR(20200808142800),YEAR ('2021-11-11'); 返回 2020 和 2021
MONTH( )	以数值格式返回参数中月的部分	SELECT MONTH(20200808142800); 返回 8
MONTHNAME( )	以字符串的格式返回参数中月的部分	SELECT MONTHNAME('2021-11-22'); 返回 November
DAYNAME( )	以字符串形式返回星期名	SELECT DAYNAME('2021-11-22'); 返回 Monday

函数	功能	示例和说明
WEEK( )	返回指定的日期是一年的第几个星期	SELECT WEEK(20200808142800); 返回 31
YEARWEEK( )	返回指定的日期是哪一年的哪一个星期	SELECT YEARWEEK('2021-11-22'); 返回 202147,表示 2021 年第 47 周
HOUR( )	返回时间值的小时部分	SELECT HOUR(155300); 返回 15
MINUTE( )	返回时间值的分钟部分	SELECT MINUTE('13 :55 :00'); 返回 55
SECOND( )	返回时间值的秒部分	SELECT SECOND(132445); 返回 45
DATE_FORMAT (日期时间,格式符)	将日期时间按照指定的格式符输出，格式主要有 %Y、%y、%m、%d 等	SELECT DATE_FORMAT(now(),'%Y,%y,%m,%d'); 返回 2020,20,08,12

【演示示例 5-3】日期函数的应用

```
USE SchoolDB;
SELECT studentName AS '姓名',YEAR(NOW())-YEAR(bornDate) AS '年龄'
   FROM Student
   ORDER BY bornDate DESC;
```

- 按照“出生日期”降序和按照计算的“年龄”升序的效果相同

因为日期型数据比较时，靠后的日期更大一些

靠后的日期转换为年龄时会更小一点

- 使用 YEAR()函数提取出生日期中的年份

5.2.3 数学函数

- 功能

数学函数用于对数值型数据进行处理

- 常用的数学函数分析

函数	功能	示例和说明
RAND( )	返回从 0 到 1 之间的随机 float 值	SELECT RAND( ); 返回 0.767553509644696(该值随机产生)
ABS(数值表达式)	取数值表达式的绝对值	SELECT ABS(-26); 返回 26
CEILING(数值表达式)	向上取整,取大于或等于指定数值、表达式的最小整数	SELECT CEILING(28.5); 返回 29
FLOOR(数值表达式)	向下取整,取小于或等于指定表达式的最大整数	SELECT FLOOR(28.5); 返回 28
POWER(数值表达式 1,数值表达式 2)	取数值表达式的幂值	SELECT POWER(3,2); 返回 9
ROUND(数值表达式)	将数值表达式四舍五入为指定精度	SELECT ROUND(28.543,1); 返回 28.5

函数	功能	示例和说明
SIGN(数值表达式)	对于正数返回 +1,对于负数返回 −1,对于 0 则返回 0	SELECT SIGN(-28); 返回 −1
SQRT(数值表达式)	取浮点表达式的平方根	SELECT SQRT(4); 返回 2

【技能训练 5-3】对数据库 SchoolDB 的

数据表进行简单查询并使用函数

【技能训练 5-4】函数综合技能训练

5.3 模糊查询

5.3.1　通配符和模糊查询

- 通配符
  - 是一类字符
  - 它可以代替一个或多个真正的字符，查找信息时作为替代字符出现
- 模糊查询
  - 主要通过模式匹配来实现
  - 不需要精确的查询条件，而是某些列值的一部分，不要求与列值完全相等
  - 例如：要查找用户表中姓李的客户的相关信息
- 模糊查询会用到 LIKE 运算符
  - 用于指出一个字符串与指定字符串是否相匹配
  - 需要与通配符一起使用
- 常用通配符：“_”和“%”
  - “%”代表 0 个或多个字符
  - “_”代表单个字符

5.3.2　使用 LIKE 进行模糊查询 117

表达式 1 [NOT] LIKE 表达式 2

- 表达式 1 和表达式 2 一般要求是字符型表达式
- 表达式的值
  - 为逻辑真或者逻辑假
  - 一般用在 WHERE 子句中，作为查询、更新和删除的条件出现
- 由于 MySQL 默认不区分大小写，要区分大小写时需要更换字符集的校对规则
- 转义字符
  - 如果给出的查询条件本身包含特殊符号中的一个或全部（_或%）

- ◆ 必须使用转义字符来实现查询
- ◆ 转义字符为单个字符，没有默认值

演示示例 5-4】使用 LIKE 进行模糊查询

- 在数据库 SchoolDB 中
  - ◆ 查询所有所有姓王学生的邮箱信息
  - ◆ 使用中文汉字显示学号、姓名和邮箱账号

```
USE SchoolDB;
SELECT studentNo AS '学号',studentName AS '姓名',email AS '邮箱'
   FROM Student
   WHERE studentName LIKE '王%';
```

- 姓王和包含王要使用不同的表达式
  - ◆ 姓名中姓王的表达式：“studentName LIKE '王%'”
  - ◆ 姓名中包含王的表达式：“studentName LIKE '%王%'”

【技能训练 5-5】使用 LIKE 对 SchoolDB

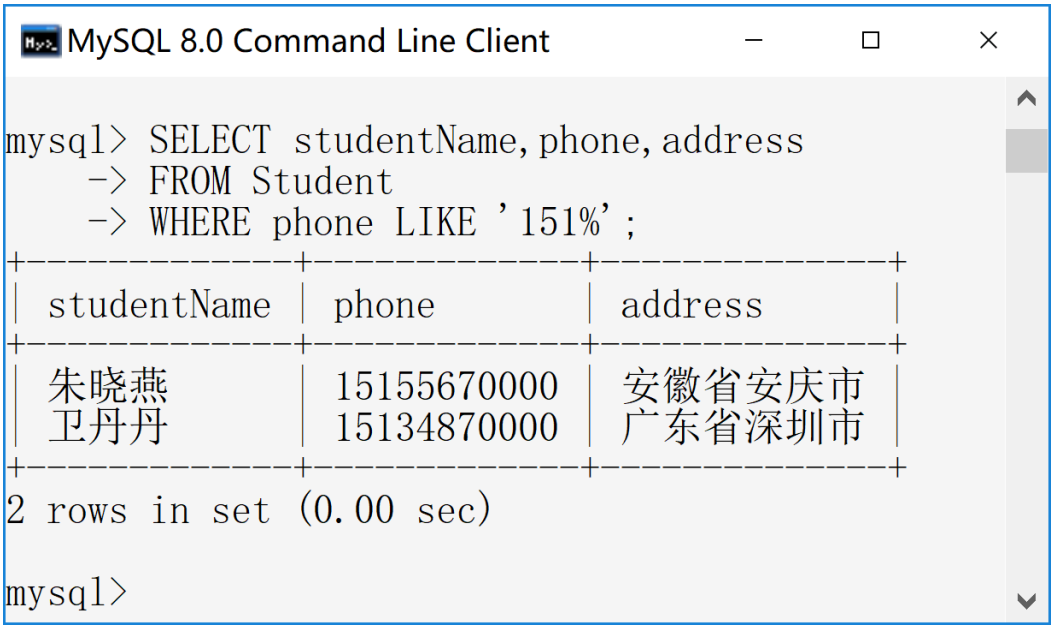
的数据表进行模糊查询 117

- 在学生信息表 Student 和课程表 Subject 中进行信息的模糊查询
  - ◆ 查询住址在“合肥”的学生姓名、电话、住址
  - ◆ 查询名称中含有“设计”字样的课程名称、学时及所属年级，并按年级由低到高显示
  - ◆ 查询电话号码以“151”开头的学生姓名、住址和电话
- 住址为合肥
- 并没有说明具体的地址，所以使用匹配多个字符的通配符

```
SELECT studentName,phone,address
    FROM Student
    WHERE address LIKE '%合肥%';
```

电话也是字符型

可以直接使用 LIKE 运算符住址为合肥



5.3.3　使用 BETWEEN 在某个范围内

进行查询 118

- 关键字 BETWEEN 可以查找那些介于两个已知值之间的一组值
  - 要实现这种查找，必须知道查找的初值和终值
  - 并且初值要小于等于终值，初值和终值用 AND 关键字分开
  - 查询结果包含初值和终值
- 例如：查询分数在 70（含）85（含）之间的信息

```
SELECT * FROM result
    WHERE studentResult BETWEEN 70 AND 85;
```

- 查询分数在 70（含）85（含）之间的信息
- 查询分数在 70（含）85（含）之间的信息

- 如果写成如下形式，执行结果为空
- 表示没有查询到任何信息，因为初值没有小于等于终值

```
SELECT * FROM Result
    WHERE StudentResult BETWEEN 85 AND 70;
```

- BETWEEN 在查询日期范围的时候使用得比较多
  - 例如：查询不在 2019 年 11 月 1 日到 2020 年 1 月 7 日之间考试的成绩信息

```
SELECT * FROM Result
    WHERE ExamDate NOT BETWEEN '2019-11-1' AND '2020-1-7';
```

5.3.4 使用 IN 在列举值内进行查询

- 应用场合
  - 查询的值是指定的某些值之一
  - 使用带列举值的 IN 关键字来进行查询
  - 将列举值放在圆括号里，用逗号分开
  - 列举值类型必须与匹配的列具有相同的数据类型
- 例如：查询第 2、第 3 和第 4 学年开设课程的详细信息

```
SELECT * FROM Subject
    WHERE gradeId IN(2,3,4) ORDER BY gradeId;
```

5.4 使用聚合函数进行数据的统计

- 聚合函数也称为统计函数
  - 它是对一组值进行计算并返回一个数值
- 主要的聚合函数
  - SUM()
  - AVG()
  - MAX()

- MIN()
- COUNT()

5.4.1 SUM( )函数

- SUM()函数返回表达式中所有数值的总和
  - 忽略其中的空值
  - SUM()函数只能用于数字类型的列
  - 不能够汇总字符、日期等其他数据类型

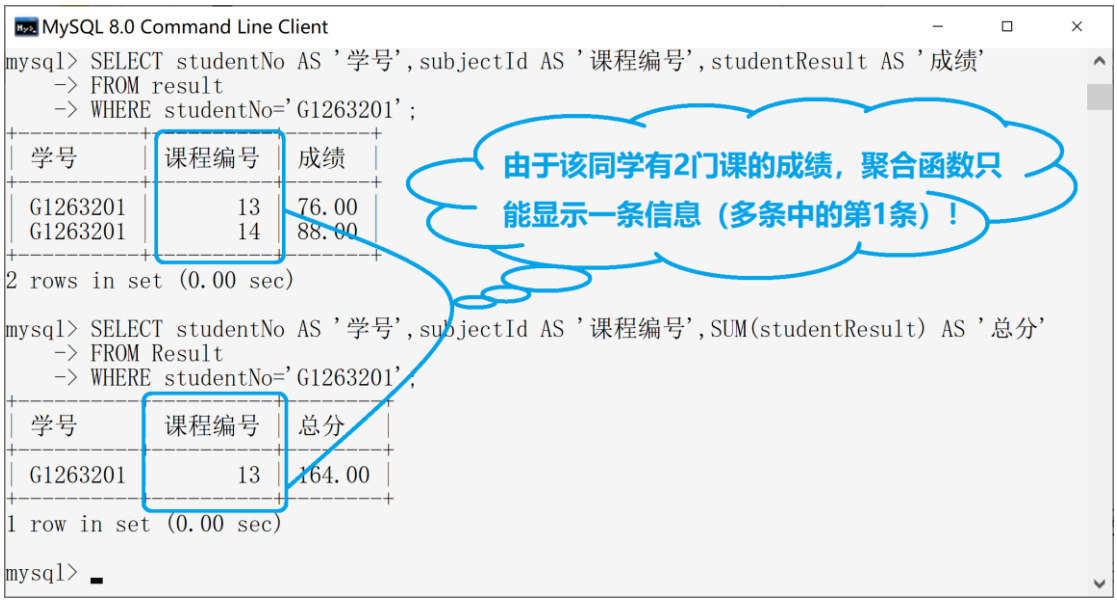
【演示示例 5-5】SUM()函数的应用

- 在数据库 SchoolDB 的成绩表 Result 中
- 查询学号为“G1263201”学生所有课程的考试总分

```
USE SchoolDB;
SELECT studentNo AS '学号',SUM(studentResult) AS '总分'
    FROM Result
    WHERE studentNo='G1263201';
```

- 聚合函数的查询一般只返回一个数值
  - 不建议与可能返回多行的列一起来查询
  - 本示例代码中显示了学号列，由于查询某个学号学生的成绩，返回的学号最多还有 1 个，符合要求
- 如果要求显示该学生所有的课程编号，则结果出现偏差
- 要求显示该学生所有的课程编号，则结果出现偏差

```
SELECT studentNo AS '学号',subjectId AS '课程编号',studentResult AS '成绩'
    FROM result
    WHERE studentNo='G1263201';
SELECT studentNo AS '学号',subjectId AS '课程编号',SUM(studentResult) AS '总分'
    FROM result
    WHERE studentNo='G1263201';
```

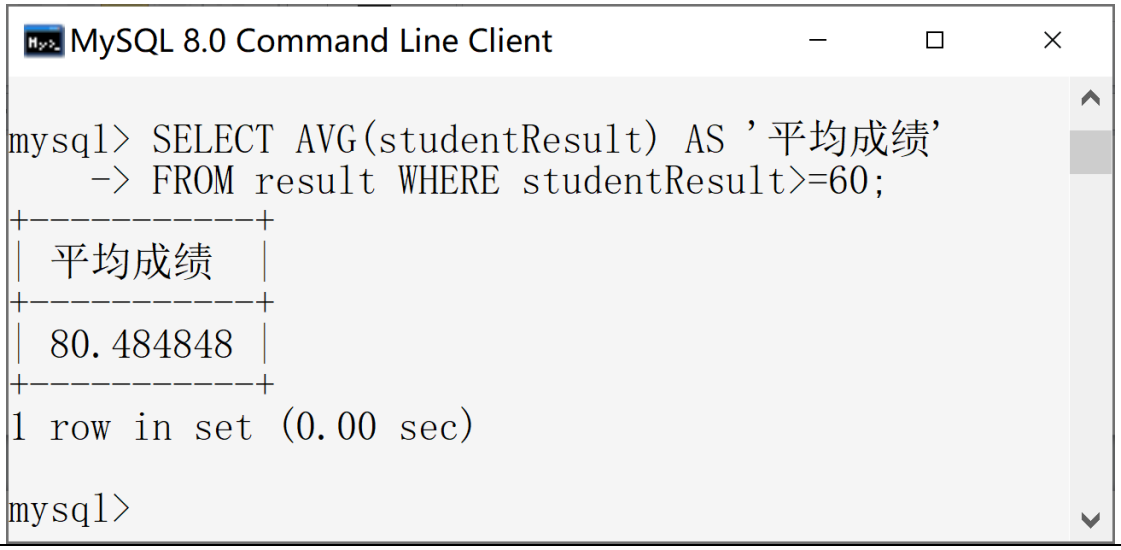


5.4.2 AVG( )函数

- AVG()函数返回表达式中所有数值的平均值
  - 空值将被忽略
  - 只能用于数字类型的列
  - 不能够汇总字符、日期等其他数据类型
- 例如：在成绩表 Result 中，查询所有及格学生的平均成绩

```
SELECT AVG(studentResult) AS '平均成绩'
    FROM result WHERE studentResult>=60;
```

- 在成绩表 Result 中，查询所有及格学生的平均成绩

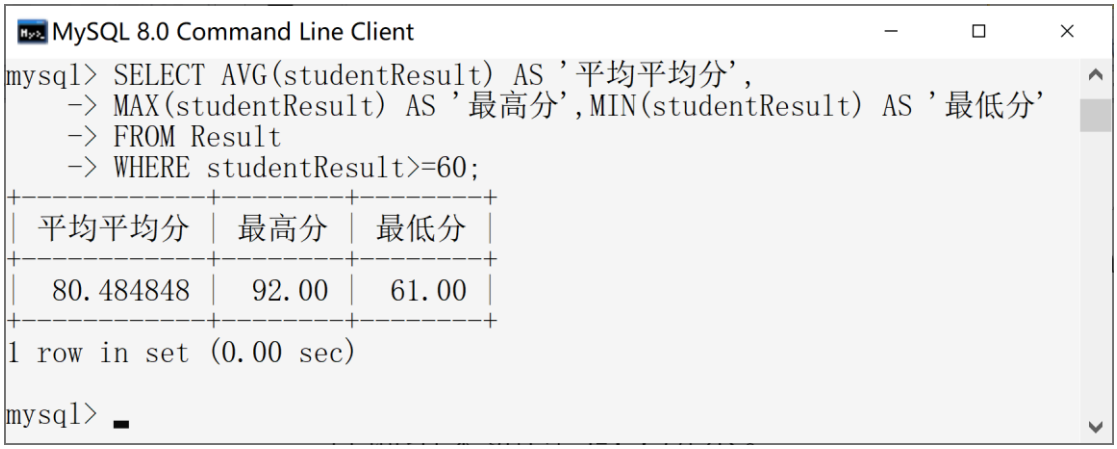


5.4.3 MAX( )函数和 MIN( )函数

- MAX()函数返回表达式中的最大值，MIN()函数返回最小值
  - 这两个函数同样都忽略任何空值
  - 都可以用于数字型、字符型及日期/时间类型的列
- 对于字符序列
  - MAX()函数查找排序序列的最大值
  - MIN()函数返回排序序列的最小值
- 例如：查询及格以上成绩的平均分、最高分、最低分

```
SELECT AVG(studentResult) AS '平均平均分',
  MAX(studentResult) AS '最高分',MIN(studentResult) AS '最低分'
  FROM Result
  WHERE studentResult>=60;
```

- 例如：查询及格以上成绩的平均分、最高分、最低分

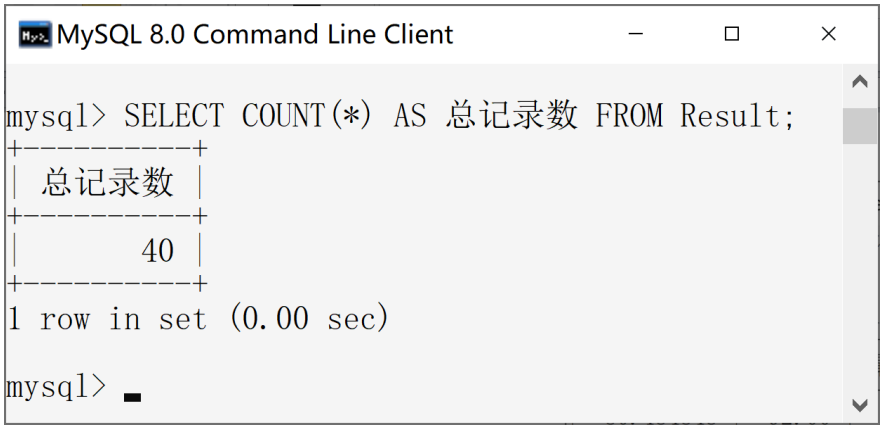


5.4.4 COUNT( )函数

- COUNT()函数返回记录集的条数
  - 一般使用星号(*)作为 COUNT 函数的参数
  - 使用星号可以不必指定特定的列而计算所有的行数
  - 当对所有的行进行计数时，则包括包含空值的行
- 例如：查询成绩表 Result 中总记录数

```
SELECT COUNT(*) AS 总记录数 FROM Result;
```

- 查询成绩表 Result 中总记录数



也可以对某列进行计数，此时忽略空值

```
SELECT COUNT(studentResult) AS 分数记录数 FROM Result;
```

【技能训练 5-6】使用聚合函数对 SchoolDB 的数据表进行汇总查询

统计学生信息表 Student 中学生总人数

使用 COUNT()函数，统计“S1”年级的总学时

使用 SUM()函数

使用 WHERE 子句

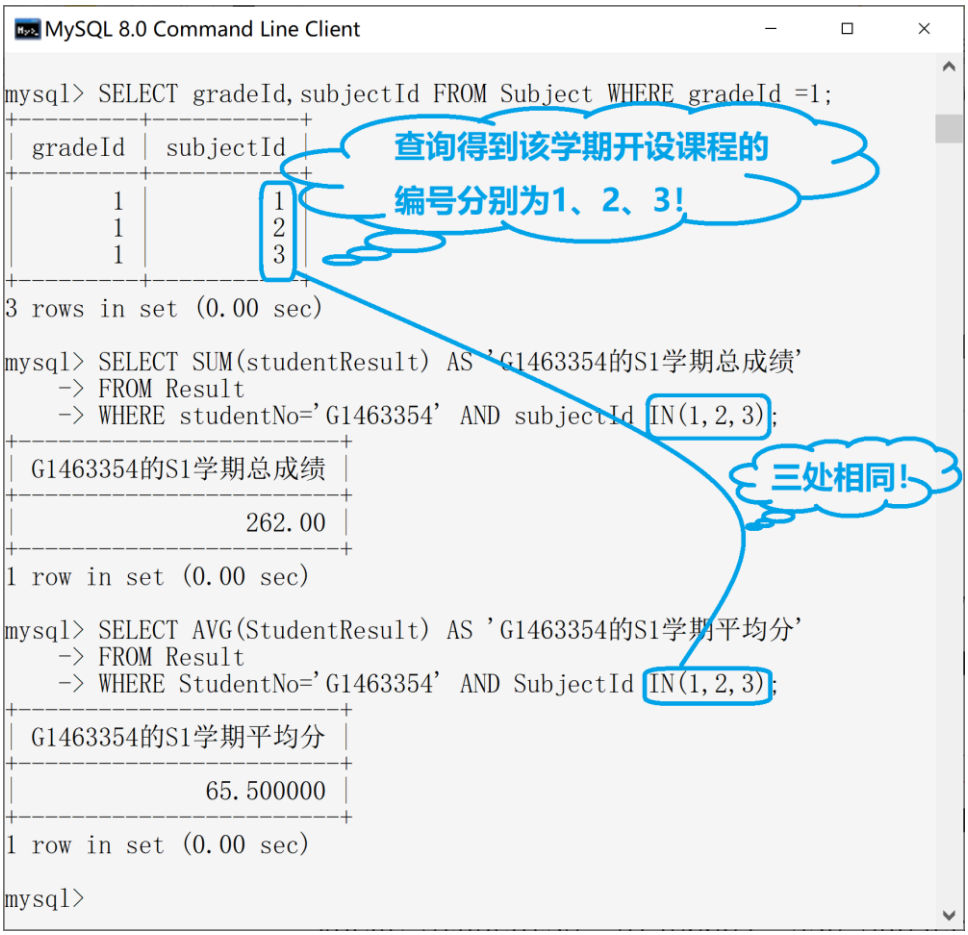
由于课程表 Subject 只有课程编号，无法直接以年级名称为条件，需要先在年级表 Grade 中查询“S1”的年级编号。

查询学号为“G1463354”的学生“S1”年级考试的总成绩

查询学号为“G1463354”的学生“S1”年级所有考试的平均分

分别使用 SUM()和 AVG()函数

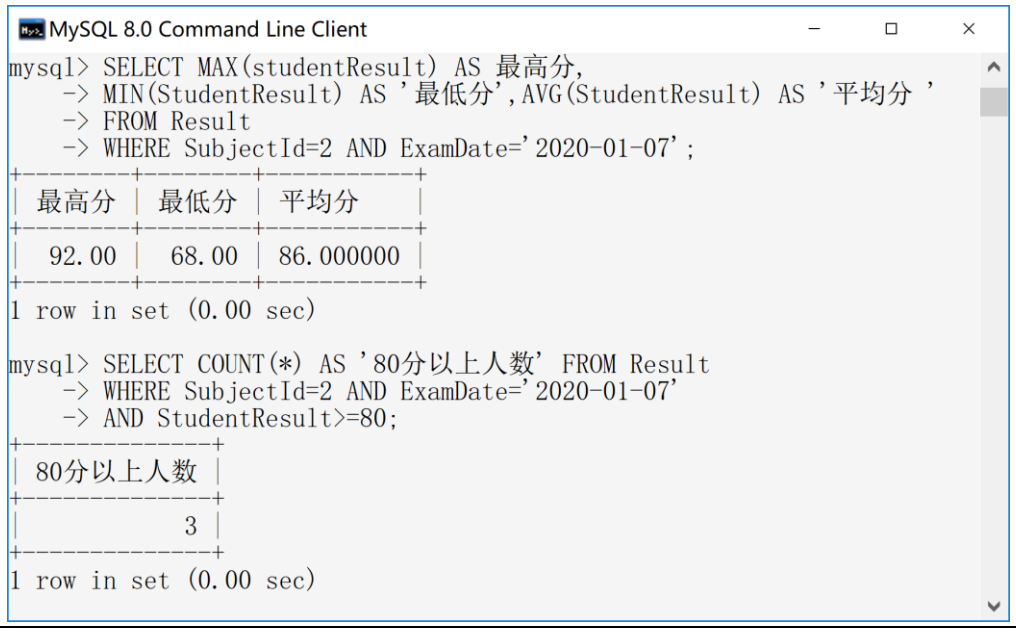
先在年级表 Grade 中查询“S1”的年级编号，使用 IN 进行查询



查询 2020 年 1 月 7 日课程“大学英语”的最高分、最低分、平均分

统计参加 2020 年 1 月 7 日“大学英语”课程考试中有多少人达到 80 分

先在课程表 Subject 中查询得到“大学英语”课程的课程编号，再在成绩表 Result 中进行统计。



查询所有参加“C 语言程序设计”课程考试的学生的平均分

先在课程表 Subject 中查询得到“C 语言程序设计”课程的课程编号，再在成绩表 Result 中进行统计，使用 AVG()函数。

课后反思	

<table>
<tr><td>课题</td><td colspan="3">第 6 章　分组查询和多表查询</td></tr>
<tr><td>教学目标</td><td colspan="3">➢ 理解和掌握分组查询和连接查询的机制
➢ 使用 GROUP BY 和 HAVING 子句实现分组和筛选查询
➢ 掌握多表连接查询及应用
➢ 掌握子查询及应用</td></tr>
<tr><td>重点难点</td><td colspan="3">➢ 外连接和内连接的区别及应用场景
➢ 灵活使用多表连接查询或者子查询解决实际问题</td></tr>
<tr><td>建议学时</td><td>9</td><td>教学教具</td><td>一体化教学实训室</td></tr>
<tr><td>主要教法</td><td colspan="3">讲授演示法</td></tr>
<tr><td>思政融入</td><td colspan="3">查询优化的必要性
某政务服务平台因多表查询未优化导致系统卡顿，影响群众办事体验的案例，说明 “高效操作也是责任”。如同窗口服务需提高效率减少群众等待，查询优化不仅能提升系统性能，更能保障数据服务的及时性与可靠性，体现 “以用户为中心” 的职业理念。</td></tr>
<tr><td colspan="4">教　学　过　程
 
</td></tr>
</table>

6.1 分组查询

- 在实际的信息处理中，经常需要进行分类统计
  - 例如：统计每个人的平均成绩
  - 首先需要对成绩表的记录按照学号来分组
  - 再对每组计算平均成绩
- 分类统计在实际应用场合比较多
  - 例如，网上书店销售系统，需要分类统计不同种类图书的总数、平均价格
  - 首先按照图书种类进行分类，然后在每组的基础上分别进行汇总和统计
  - 分组后的统计计算要利用聚合函数 COUNT()、AVG()等

分组查询

语法

```
GROUP BY  列名 1[,列名 2,...]
```

分析

- 使用该子句实现分组
- 只有一列表示单列分组查询
- 如果分多列进行分组查询，需要列出具体的列名，以分号分隔

6.1.1 使用 GROUP BY 进行分组查询

1.单列分组查询

- 成绩表中存储了学生参加考试的成绩
- 需要统计不同科目的平均成绩
- 首先需要对成绩表中的记录按照科目来分组
- 再针对每个组进行平均成绩计算
- 使用聚合函数 AVG()

【演示示例6-1】统计每门课的平均分

需求

- 数据库SchoolDB的成绩表Result中
  - 统计每门课程的平均分

分析

- 成绩记录了多门课程的学生成绩，统计不同科目的平均成绩
- 把相同的subjectId都分为一组，这样就将数据分成了多组
- 针对每一组使用聚合函数取平均分

（1）没有分组则求所有记录平均分

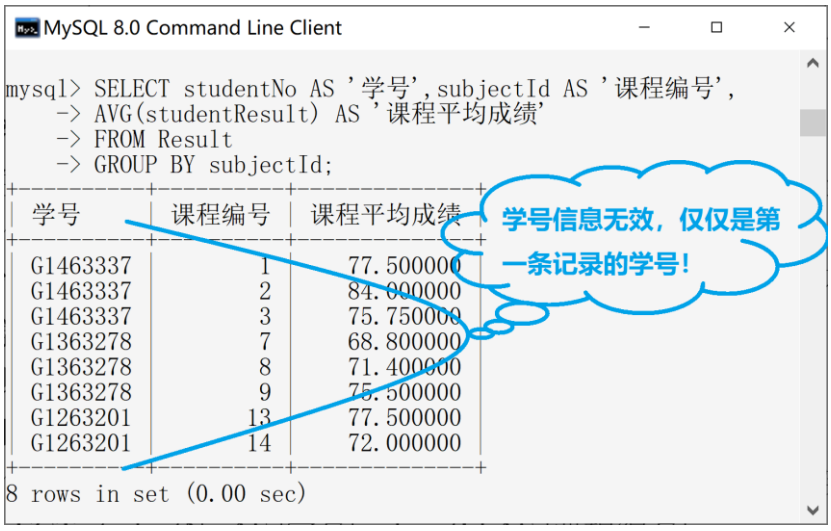
```
mysql> select subjectid 课程编号,avg(studentResult) 平均分
    -> from result;
    +----------+----------+
    | 课程编号 | 平均分   |
    +----------+----------+
    |       13 | 74.452381 |
    +----------+----------+
```

2.按分组字段求各组平均分

```
mysql> select subjectid 课程编号,avg(studentResult) 平均分
    -> from result
    -> group by subjectid;
    +----------+----------+
    | 课程编号 | 平均分   |
    +----------+----------+
    |        1 |     77.5 |
    |        2 |       84 |
    |        3 |    75.75 |
    |        7 |     68.8 |
    |        8 |     71.4 |
    |        9 |     75.5 |
    |       13 | 82.333333 |
    |       14 |       48 |
    +----------+----------+
    8 rows in set
```

能否输出显示学号信息

- 不可以
- 学生的学号与课程不是一对一的关系
- 因为科目已经被“分组”了，分组后的数量减少为 8 组
- 每组中有多位学生，即多个学号



【演示示例6-2】统计每位学生的平均分

需求

- 数据库SchoolDB的成绩表Result中
  - 统计每位学生的平均分
  - 按照平均分由高到低的顺序排列显示

分析

- 求每位同学的平均分
  - 需要前按照学号分组，再使用聚会函数AVG()
- 对显示结果排序
  - 排序的依据是平均分，即利用聚合函数AVG()作为排序的依据

MySQL 8.0 Command Line Client

```
mysql> SELECT studentNo AS '学号',AVG(studentResult) AS '平均成绩'
    -> FROM Result
    -> GROUP BY studentNo
    -> ORDER BY AVG(studentResult) DESC;
+----------+-----------+
| 学号     | 平均成绩  |
+----------+-----------+
| G1463383 | 89.000000 |
| G1463388 | 83.250000 |
| G1263201 | 82.000000 |
| G1463337 | 80.000000 |
| G1463358 | 78.500000 |
| G1463342 | 77.000000 |
| G1363303 | 76.333333 |
| G1363302 | 74.333333 |
| G1363301 | 74.250000 |
| G1363278 | 69.666667 |
| G1263382 | 67.500000 |
| G1463354 | 65.500000 |
| G1363300 | 65.333333 |
| G1263458 | 46.000000 |
+----------+-----------+
14 rows in set (0.00 sec)
```

2.多列分组查询

- 在实际应用中，分组查询时还可以按照多个列来进行分组
- 比如：按照年级统计男女生的人数
- 先按照年级分组，再按照性别分组进行统计
- 使用 COUNT()函数

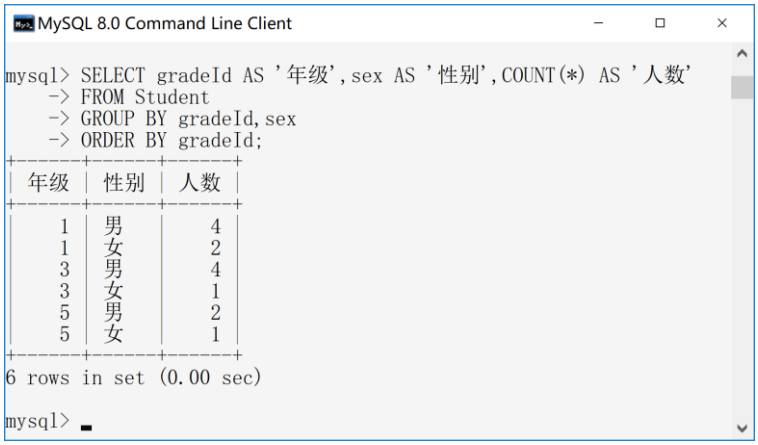
【演示示例6-3】统计每个年级的男女生人数

需求

- 数据库SchoolDB的成绩表Result中
  - 统计每个年级的男女生人数
  - 按照年级编号显示

分析

- 需要使用多列分组查询
  - GROUP BY后面需要2列，年级编号和性别
  - 年级编号在前面，使用逗号分隔
- 排序使用ORDER BY子句
  - 写在GROUP BY子句的后面，排序依据为年级编号



【技能训练6-1】分组查询应用

技能目标

- 使用GROUP BY进行分组查询
- 灵活应用GROUP BY和ORDER BY等子句解决实际问题

1.查询每个年级的总学时数，按总学时数降序排列

```
mysql> select gradeId 年级,sum(classHour) 总学时数
   -> from subject
   -> group by gradeId
   -> order by sum(classHour) desc;
      +------+----------+
      | 年级 | 总学时数 |
      +------+----------+
      |   1 | 224      |
      |   2 | 224      |
      |   3 | 192      |
      |   4 | 192      |
      |   5 | 64       |
      +------+----------+
      5 rows in set
```

2.查询参加每门课考试的人数，按照考试人数降序排列

```
mysql> select subjectId 课程编号,count(*) 考试人数
   -> from result
   -> group by subjectId
   -> order by count(*) desc;
      +----------+----------+
      | 课程编号 | 考试人数 |
      +----------+----------+
      |      1 |      10 |
      |      2 |       6 |
      |      9 |       6 |
      |      7 |       5 |
      |      8 |       5 |
      |      3 |       4 |
      |     13 |       3 |
      |     14 |       3 |
      +----------+----------+
      8 rows in set
```

3.查询每个年级的总人数

```
mysql> select gradeId 年级,count(*) 总人数
    -> from student
    -> group by gradeId;
    +------+--------+
    | 年级 | 总人数 |
    +------+--------+
    |    1 |      6 |
    |    3 |      5 |
    |    5 |      3 |
    +------+--------+
    3 rows in set
```

4.查询每个学生参加的考试的总分，并按总分降序排列

```
mysql> select studentNO 学号, sum(studentResult) 总分
    -> from result
    -> group by studentNO
    -> order by sum(studentResult) desc;
    +----------+--------+
    | 学号     | 总分   |
    +----------+--------+
    | G1463383 | 356.00 |
    | G1463388 | 333.00 |
    | G1463337 | 320.00 |
    | G1363301 | 297.00 |
    | G1463354 | 262.00 |
    | G1363303 | 229.00 |
    | G1363302 | 223.00 |
    | G1363278 | 209.00 |
    | G1363300 | 196.00 |
    | G1263201 | 164.00 |
    | G1463358 | 157.00 |
    | G1463342 | 154.00 |
    | G1263382 | 135.00 |
    | G1263458 |  92.00 |
    +----------+--------+
    14 rows in set
```

5.查询每门课的最高分、最低分，按照课程编号升序排列

```
mysql> select subjectId 课程编号,max(studentResult)最高分,min(studentResult)最低分
    -> from result
    -> group by subjectId
    -> order by subjectId;
    +----------+--------+--------+
    | 课程编号 | 最高分 | 最低分 |
    +----------+--------+--------+
    |        1 |  90.00 |  52.00 |
    |        2 |  92.00 |  68.00 |
    |        3 |  89.00 |  56.00 |
    |        7 |  83.00 |  55.00 |
    |        8 |  87.00 |  49.00 |
    |        9 |  90.00 |  55.00 |
    |       13 |  92.00 |  76.00 |
    |       14 |  88.00 |   0.00 |
    +----------+--------+--------+
    8 rows in set
```

6.查询每个学生所参加考试的最高分、最低分，按最高分降序排列

```
mysql> select studentNo 学号,max(studentResult)最高分,min(studentResult)最低分
    -> from result
    -> group by studentNo
    -> order by max(studentResult) desc;
    +----------+--------+--------+
    | 学号     | 最高分 | 最低分 |
    +----------+--------+--------+
    | G1463358 |     92 |     65 |
    | G1263458 |     92 |      0 |
    | G1463383 |     92 |     87 |
    | G1463337 |     92 |     56 |
    | G1463388 |     92 |     78 |
    | G1363301 |     90 |     55 |
    | G1263201 |     88 |     76 |
    | G1363302 |     87 |     56 |
    | G1363303 |     87 |     61 |
    | G1463342 |     86 |     68 |
    | G1363300 |     83 |     49 |
    | G1263382 |     79 |     56 |
    | G1363278 |     78 |     55 |
    | G1463354 |     75 |     52 |
    +----------+--------+--------+
    14 rows in set
```

6.1.2 使用 HAVING 子句进行分组筛选

使用HAVING子句进行分组筛选

- 在实际应用中，有的需求还需要在分组统计的基础上进行筛选
  - 对分组统计的结果进行过滤
  - 比如：统计总分达到200分的学生信息
- HAVING子句
  - 实现分组后的筛选功能
  - HAVING子句要用在GROUP BY子句后，用于过滤分组后的结果
- 与WHERE子句比较
  - WHERE子句是用来在FROM子句中选择行
  - HAVING子句是在GROUP BY子句后选择行

使用HAVING子句进行分组筛选

- HAVING和WHERE子句可以一起使用
- 使用顺序

示例

```
SELECT subjectId AS '课程编号',COUNT(*) AS '及格人数',
  AVG(studentResult) AS '及格学生的平均分'
  FROM Result
  WHERE studentResult>=60
  GROUP BY subjectId
  HAVING AVG(studentResult)>=80
  ORDER BY AVG(studentResult) DESC;
```

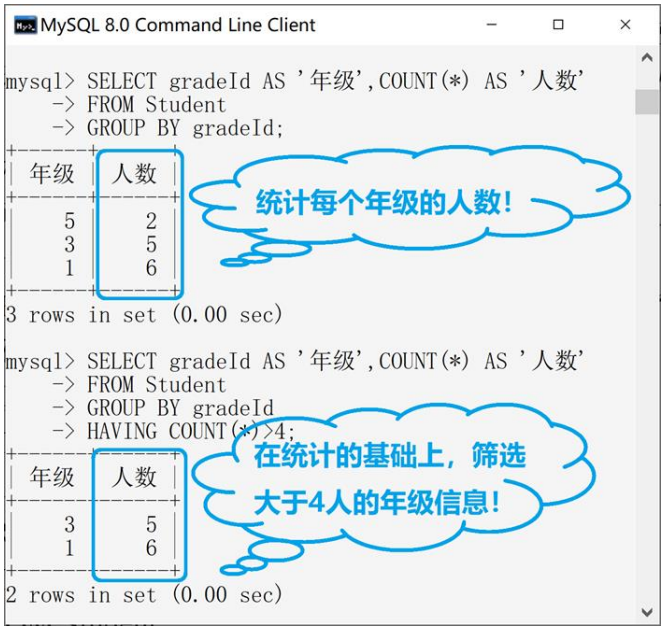
【演示示例6-4】分组筛选的应用

需求

- 在数据库SchoolDB的学生信息表Student中
  - 查询年级总人数超过4的年级编号

分析

- 通过分组查询获取每个年级的总人数
  - 使用GROUP BY子句，按照年级编号分组
  - 使用COUNT()函数统计
- 在GROUP BY后面使用HAVING子句，条件:" COUNT(*)>4"



- 使用 WHERE 子句是不能满足查询要求的
  - WHERE 子句只能对没有分组统计前的数据进行筛选
- 用 HAVING 子句来指定筛选的条件
  - 该子句中的条件通常是统计函数
  - 如：条件为"COUNT(*)>4"

【演示示例 6-5】分组筛选的综合应用

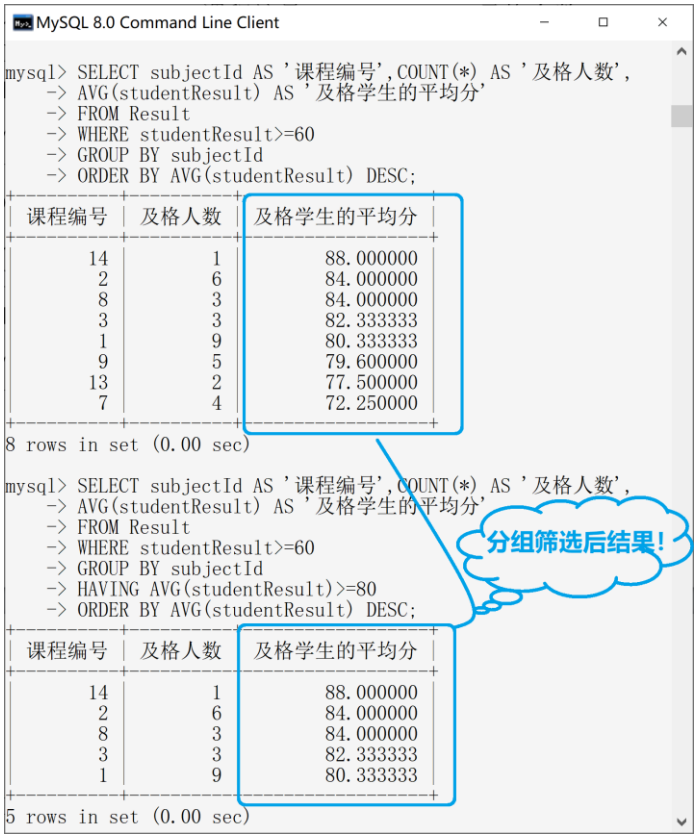
- 在数据库SchoolDB的成绩表Result中
  - 统计每门科目及格总人数
  - 统计及格学生的平均分达到80分的课程信息
  - 按照平均分降序显示

分析

- 所查询的信息都是及格的信息
  - 先从数据源中将不及格的学生信息进行滤除
  - 使用WHERE子句
- 对符合及格要求的数据再进行分组
  - 通过分组查询获取每门课及格学生的人数和平均分
  - 通过GROUP BY子句实现分组
- 筛选出平均分达到80分的课程信息
  - 在分组统计出的平均分的基础上，使用HAVING子句

关键步骤

- 步骤一：使用WHERE子句选择及格的信息
- 步骤二：使用GROUP BY对数据再进行分组
- 步骤三：使用HAVING筛选出平均分达到80分的课程信息



- 理解各子句的功能和使用顺序
  - WHERE 子句
  - GROUP BY 子句
  - HAVING 子句

- ◆ ORDER BY 子句
- ■ 在所有子句中，ORDER BY 子句一般都在最后
  - ◆ 用于对查询最后结果的显示排序

6.2 多表查询

- ■ 在实际应用中，很多查询需要的数据要来源于多张表
  - ◆ 比如：在查询成绩的结果中显示学生姓名和课程名称
  - ◆ 涉及到三张表
    - ✓ 成绩表Result
    - ✓ 学生信息表Student
    - ✓ 课程表Subject
- ■ 查询命令SELECT支持多表连接查询
  - ◆ 内连接查询
  - ◆ 外连接查询

6.2.1 内连接查询

- ■ 内连接查询是最典型、最常用的连接查询，它根据表中共同的列来进行匹配。特别是两张表存在主外键关系时通常会使用内连接查询。
- ■ MySQL数据库中，内连接查询可以通过两种方式实现，一种是在FROM子句中使用INNER JOIN...ON关键字，另一种是在WHERE子句中设置连接条件
- ■ 在实际应用中，很多查询需要的数据要来源于多张表
  - ◆ 比如：在查询成绩的结果中显示学生姓名和课程名称

1. 在 WHERE 子句中指定连接条件

语法

```
SELECT 显示列表
  FROM 表 1,表 2[,...]
  WHERE 连接条件;
```

分析

- ■ 将查询需要数据所在的数据表分别列在FROM子句后面
  - ◆ 表名之间用逗号分隔
- ■ 将表与表之间连接的条件作为WHERE子句的条件表达式

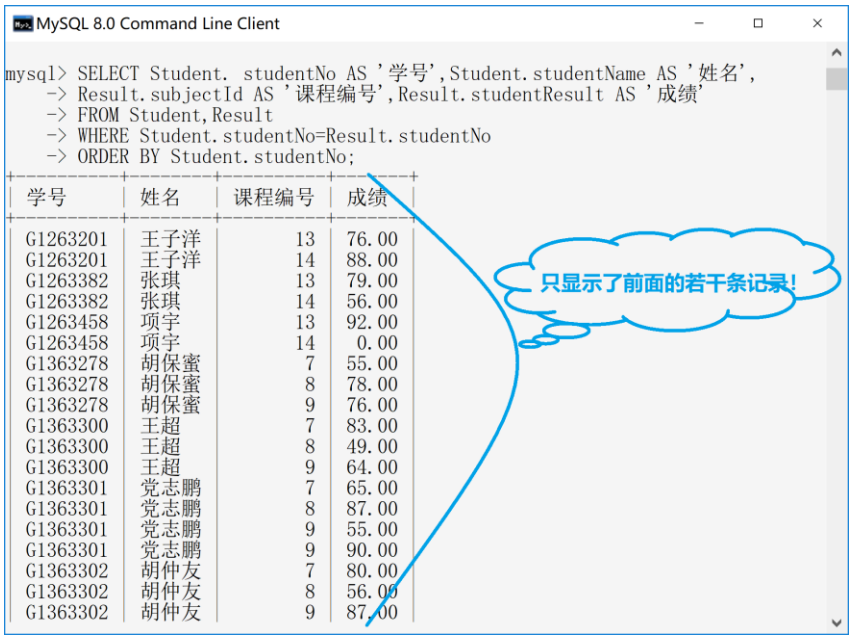
【演示示例 6-6】2 张表内连接查询的应用

需求

- ■ 数据库SchoolDB中，查询学生的成绩
  - ◆ 显示学号、姓名、课程编号和成绩，按照学号排列

分析

- ■ 需要查询的结果来源于2张表，需要2张表连接查询
- ■ 成绩表中的学号引用了学生信息表的学号，连接条件：学号相等
- ■ Student和Result中学号列的名称都是“studentNo”
  - ◆ 需要在列名的前面加上表的名称，并使用点“.”作为连接符
  - ◆ 如：“Student. studentNo”和“Result. studentNo”



拓展

- 多表列名相同问题的处理
  - 只有当多表中存在相同名称的列时，需要使用“表名.列名”来区别
  - 如果列名不相同时可以直接使用列名

```
SELECT Student. studentNo AS '学号',studentName AS '姓名',
   subjectId AS '课程编号',studentResult AS '成绩'
   FROM Student,Result
   WHERE Student.studentNo=Result.studentNo
   ORDER BY Student.studentNo;
```

2.在 FROM 子句中使用 INNER JOIN…ON

语法

```
SELECT 显示列表
  FROM 表 1 INNER JOIN 表 2
  ON (连接条件);
```

分析

- 通过FROM子句中使用INNER JOIN...ON关键字
  - 不需要在WHERE子句中设计表连接的条件表达式
  - 可以连接2张表，也可以连接3张表等

```
mysql> select s.studentNo 学号,studentName 姓名,subjectId 课程编号,studentResult 成绩
  -> from student AS s
  -> INNER JOIN result AS r ON (s.studentNo = r.studentNo)
  -> order by s.studentNo;
```

- AS指定表的“别名”
  - 即表的名称可以用简单的别名表示
  - 如：“Student AS S”，S表示Student表的别名
  - 使用“Student”和使用“S”等价
  - 如“S.studentNo”等同于“Student.studentNo”

【演示示例 6-7】查询某门课及格学生的信息

```
mysql> select subjectId 课程编号,studentName 姓名,studentResult 成绩
    -> from student as S inner join result as r
    -> on (s.studentNo = r.studentNo)
    -> where subjectId = 3;
      +----------+--------+------+
      | 课程编号 | 姓名   | 成绩 |
      +----------+--------+------+   ——没有及格条件限制
      |        3 | 高伟   |   56 |
      |        3 | 陈大伟 |   75 |
      |        3 | 钱嫣然 |   89 |
      |        3 | 卫丹丹 |   83 |
      +----------+--------+------+
      4 rows in set

mysql> select subjectId 课程编号,studentName 姓名,studentResult 成绩
    -> from student as S inner join result as r
    -> on (s.studentNo = r.studentNo)
    -> where subjectId = 3 and studentResult >= 60;
      +----------+--------+------+
      | 课程编号 | 姓名   | 成绩 |
      +----------+--------+------+   ——增加及格条件限制
      |        3 | 陈大伟 |   75 |
      |        3 | 钱嫣然 |   89 |
      |        3 | 卫丹丹 |   83 |
      +----------+--------+------+
      3 rows in set
```

需求

- 数据库SchoolDB中
- 查询课程编号为3的及格学生的姓名和分数

分析

- 需要查询的结果来源于2张表，需要2张表连接查询
- 采用组合条件
  - 统计及格的学生使用WHERE子句
  - 课程编号为“3”也是使用WHERE子句
  - 使用逻辑运算符“AND”连接2个条件

3. 3张表及更多表的查询

- 多表连接
  - 内连接查询可以连接两个表，还可以实现涉及三个表或者更多表
  - 查询更多需要的信息
- 实现的方式
  - 2张连接查询的方式相同
  - 在FROM子句增加需要的表名
  - 或者在WHERE中增加需要的连接条件

【演示示例 6-8】查询指定学生的信息

需求

- 数据库SchoolDB中，查询“胡保蜜”同学的成绩
  - 显示学号、姓名、课程名称和考试成绩
  - 按照课程编号升序排序

分析

- 所需查询的信息来源于3张表，需要3张表的连接查询
- 涉及3张表的查询，连接条件至少需要2个
  - 先将Result表和Student表连接，连接条件是学号相等
  - 再继续和Subject表连接，连接条件是课程编号相等

```
mysql> select studentName 姓名,s.studentNo 学号,subjectName 课程名称,studentResult 成绩
    -> from student as S
    -> JOIN result R on  (S.studentNo = R.studentNo)
    -> JOIN subject SJ on (SJ.subjectId = R.subjectId)
    -> where studentName = '胡保蜜'
    -> order by R.subjectId;
    +--------+----------+------------------+------+
    | 姓名   | 学号     | 课程名称         | 成绩 |
    +--------+----------+------------------+------+
    | 胡保蜜 | G1363278 | Android应用开发  |   55 |
    | 胡保蜜 | G1363278 | Java面向对象设计 |   78 |
    | 胡保蜜 | G1363278 | Web客户端编程    |   76 |
    +--------+----------+------------------+------+
    3 rows in set

mysql> select studentName 姓名,s.studentNo 学号,subjectName 课程名称,studentResult 成绩
    -> from student S,result R,subject SJ
    -> where S.studentNo = r.studentNo and r.subjectId = sj.subjectId and studentName = '胡保蜜'
    -> order by R.subjectId;
    +--------+----------+------------------+------+
    | 姓名   | 学号     | 课程名称         | 成绩 |
    +--------+----------+------------------+------+
    | 胡保蜜 | G1363278 | Android应用开发  |   55 |
    | 胡保蜜 | G1363278 | Java面向对象设计 |   78 |
    | 胡保蜜 | G1363278 | Web客户端编程    |   76 |
    +--------+----------+------------------+------+
    3 rows in set
```

6.2.2 外连接查询

- 应用场合
  - 使用内连接查询会出现查询信息不完整情况
  - 比如：查询所有课程的考试成绩，如果某门课没有被学生选修
  - 则该门课程不会出现在查询结果结果中
  - 如果需要查询未被选修的课程信息，可以通过外连接查询实现
- 外连接查询与内连接查询最大区别
  - 外连接查询中参与连接的表有主从之分
  - 以主表的每行数据匹配从表的数据，将符合条件的数据直接返回到结果集中
  - 对于不符合连接条件的列，在结果集中被填上NULL值（空值）
- 外连接分类
  - 左外连接（LEFT OUTER JOIN）
  - 右外连接（RIGHT OUTER JOIN）
  - 其中OUTER关键字可以省略

1.左外连接查询

左外连接查询

- 左外连接（LEFT OUTER JOIN）
  - 结果表中除了匹配行外（即与内连接查询结果相同的部分）
  - 还包括左表中有的但右表中不匹配的行，从右表选择列的列值为NULL
  - 查询结果包括
    - ✓ 第一个命名表（“左”表，出现在JOIN子句的最左边）中的所有行
    - ✓ 不包括右表中的不匹配行
  - 连接条件和方式与内连接相同，主要体现在查询的结果集不同上

【演示示例 6-9】统计所有课程选修情况

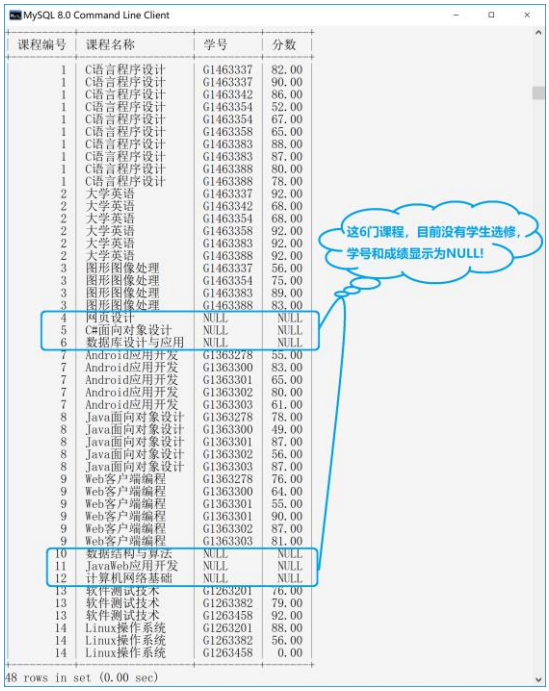
需求

- 数据库SchoolDB中，查询统计所有课程的选修及成绩信息
- 显示学号、姓名、课程名称和考试成绩，按照课程编号升序排序
  - 如果在Result表中有成绩，则表示该同学选修了该门课
  - 如果某门课Result表中没有一条成绩记录，则表示无人选修该课程，对应课程的学号和成绩显示为NULL

分析

- 要查询包括没有被选修的课程信息，使用外链接
- 使用左外连接，左表为课程表Subject，右表为成绩表Result

```
mysql> select SJ.subjectId 课程编号,SJ.subjectName 课程名称,
    -> studentNo 学号,studentResult 成绩
    -> from subject as SJ
    -> left join result as R on SJ.subjectId = R.subjectId;
```

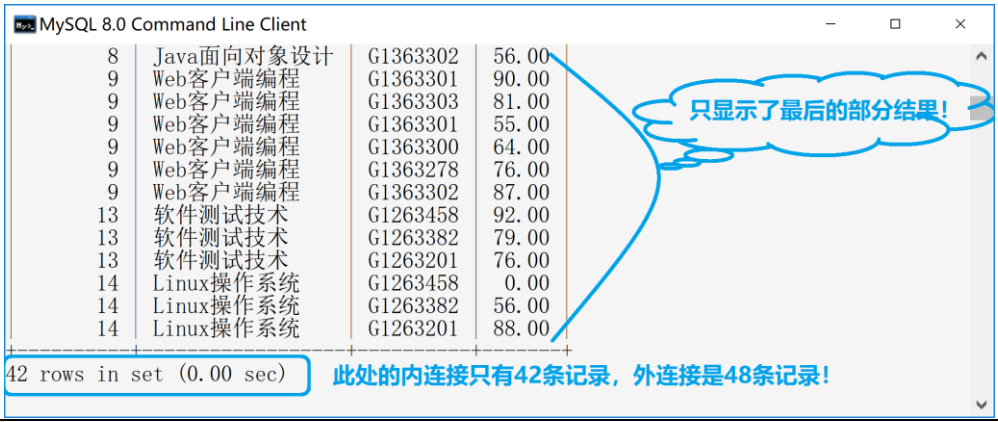


课程编号	课程名称	学号	分数
1	C语言程序设计	G1463337	82.00
1	C语言程序设计	G1463337	90.00
1	C语言程序设计	G1463342	86.00
1	C语言程序设计	G1463354	52.00
1	C语言程序设计	G1463354	67.00
1	C语言程序设计	G1463358	65.00
1	C语言程序设计	G1463383	88.00
1	C语言程序设计	G1463383	87.00
1	C语言程序设计	G1463388	80.00
1	C语言程序设计	G1463388	78.00
2	大学英语	G1463337	92.00
2	大学英语	G1463342	68.00
2	大学英语	G1463354	68.00
2	大学英语	G1463358	92.00
2	大学英语	G1463383	92.00
2	大学英语	G1463388	92.00
3	图形图像处理	G1463337	56.00
3	图形图像处理	G1463354	75.00
3	图形图像处理	G1463383	89.00
3	图形图像处理	G1463388	83.00
4	网页设计	NULL	NULL
5	C#面向对象设计	NULL	NULL
6	数据库设计与应用	NULL	NULL
7	Android应用开发	G1363278	55.00
7	Android应用开发	G1363300	83.00
7	Android应用开发	G1363301	65.00
7	Android应用开发	G1363302	80.00
7	Android应用开发	G1363303	61.00
8	Java面向对象设计	G1363278	78.00
8	Java面向对象设计	G1363300	49.00
8	Java面向对象设计	G1363301	87.00
8	Java面向对象设计	G1363302	56.00
8	Java面向对象设计	G1363303	87.00
9	Web客户端编程	G1363278	76.00
9	Web客户端编程	G1363300	64.00
9	Web客户端编程	G1363301	55.00
9	Web客户端编程	G1363301	90.00
9	Web客户端编程	G1363302	87.00
9	Web客户端编程	G1363303	81.00
10	数据结构与算法	NULL	NULL
11	JavaWeb应用开发	NULL	NULL
12	计算机网络基础	NULL	NULL
13	软件测试技术	G1263201	76.00
13	软件测试技术	G1263382	79.00
13	软件测试技术	G1263458	92.00
14	Linux操作系统	G1263201	88.00
14	Linux操作系统	G1263382	56.00
14	Linux操作系统	G1263458	0.00

48 rows in set (0.00 sec)

拓展

- 查询结果分析
  - 查询结果共48条
  - 包括了有匹配值的42行
  - 还包含了成绩表中无匹配的行，即表示成绩表中没有该课程的成绩信息，学号和成绩都显示为“NULL”



8	Java面向对象设计	G1363302	56.00
9	Web客户端编程	G1363301	90.00
9	Web客户端编程	G1363303	81.00
9	Web客户端编程	G1363301	55.00
9	Web客户端编程	G1363300	64.00
9	Web客户端编程	G1363278	76.00
9	Web客户端编程	G1363302	87.00
13	软件测试技术	G1263458	92.00
13	软件测试技术	G1263382	79.00
13	软件测试技术	G1263201	76.00
14	Linux操作系统	G1263458	0.00
14	Linux操作系统	G1263382	56.00
14	Linux操作系统	G1263201	88.00

2.右外连接查询

右外连接查询

- 右外连接（RIGHT OUTER JOIN）
  - 结果表中除了匹配行外（即与内连接查询结果相同的部分）
  - 还包括右表中有的但左表中不匹配的行，从左表选择列的列值为NULL
  - 右外连接查询结果
    - ✓ 包括第二个命名表（“右”表，出现在JOIN子句的最右边）中的所有行
    - ✓ 不包括左表中的不匹配行
- 使用RIGHT JOIN...ON或RIGHT OUTER JOIN...ON
- 右外连接查询可以通过左外连接来实现
  - 将左外连接中的2张表位置互换就实现了右外连接的效果

6. 3 子查询

- 子查询
  - 将查询的结果直接用于WHERE子句
- 子查询也是一个SELECT查询
  - 有返回值且嵌套在SELECT、INSERT、UPDATE、DELETE语句中
  - 任何允许使用表达式的地方都可以使用子查询
- 子查询也称为内部查询或内部选择
  - 包含子查询的语句也称为外部查询或外部选择
- 子查询能够将比较复杂的查询分解为几个简单的查询
- 子查询可以嵌套
  - 首先执行内部查询
  - 它查询出来的数据并不被显示出来，而是传递给外层语句
  - 并作为外层语句的查询条件来使用

6.3.1 简单子查询

语法

```
SELECT...FROM 表 1
  WHERE 字段名 比较运算符 (子查询)
```

分析

- 子查询语句必须放置在一对圆括号内
- 在字段名后面的运算符除
  - 可以是“>”、“<”等关系比较运算符
  - 也可以使用其他运算符号
- 习惯上
  - 外面的查询称为父查询
  - 圆括号中嵌入的查询称为子查询
  - 先执行子查询部分，求出子查询部分的值，不显示
  - 再执行整个父查询，返回最后的结果
- 子查询作为WHERE条件的一部分
  - 可以和UPDATE、INSERT、DELETE一起使用
  - 语法类似于SELECT语句
- 如果将子查询和比较运算符联合使用
  - 必须保证子查询返回的值不能多于一个
  - 可以为空

【演示示例 6-10】查询比某同学小的学生信息

【演示示例6-10】查询比“党志鹏”小的学生，显示学号、姓名、性别和出生日期

1.分步实现

（1）查询'党志鹏'的出生日期等信息

```
mysql> select studentNo 学号,studentName 姓名,sex 性别,bornDate 出生日期
    -> from student
    -> where studentName = '党志鹏';
    +----------+--------+------+------------+
    | 学号     | 姓名   | 性别 | 出生日期   |
    +----------+--------+------+------------+
    | G1363301 | 党志鹏 | 男   | 1994-12-20 |
    +----------+--------+------+------------+
    1 row in set
```

（2）利用'党志鹏'的出生日期查询比他小的结果

```
mysql> select studentNo 学号,studentName 姓名,sex 性别,bornDate 出生日期
    -> from student
    -> where bornDate > '1994-12-20';
    +----------+--------+------+------------+
    | 学号     | 姓名   | 性别 | 出生日期   |
    +----------+--------+------+------------+
    | G1463337 | 高伟   | 男   | 1995-06-07 |
    | G1463342 | 胡俊文 | 男   | 1995-04-20 |
    | G1463354 | 陈大伟 | 男   | 1995-08-23 |
    | G1463358 | 温海南 | 男   | 1995-01-30 |
    | G1463388 | 卫丹丹 | 女   | 1995-04-17 |
    +----------+--------+------+------------+
    5 rows in set
```

2.合并实现

```
mysql> select studentNo 学号,studentName 姓名,sex 性别,bornDate 出生日期
    -> from student
    -> where bornDate > (select bornDate from student where studentName = '党志鹏') ;
    +----------+--------+------+------------+
    | 学号     | 姓名   | 性别 | 出生日期   |
    +----------+--------+------+------------+
    | G1463337 | 高伟   | 男   | 1995-06-07 |
    | G1463342 | 胡俊文 | 男   | 1995-04-20 |
    | G1463354 | 陈大伟 | 男   | 1995-08-23 |
    | G1463358 | 温海南 | 男   | 1995-01-30 |
    | G1463388 | 卫丹丹 | 女   | 1995-04-17 |
    +----------+--------+------+------------+
    5 rows in set

mysql> select bornDate from student where studentName = '党志鹏';
    +------------+
    | bornDate   |
    +------------+
    | 1994-12-20 |
    +------------+
    1 row in set
```

【演示示例 6-11】查询“大学英语”课程的考试情况，显示课程编号、学号、成绩和考试日期

```
mysql> select subjectId 课程编号,studentNo 学号,studentResult 成绩,examDate 考试日期
    -> from result
    -> where subjectId = (select subjectId from subject where subjectName = '大学英语');
    +----------+----------+------+---------------------+
    | 课程编号 | 学号     | 成绩 | 考试日期            |
    +----------+----------+------+---------------------+
    |        2 | G1463337 |   92 | 2019-01-08 00:00:00 |
    |        2 | G1463342 |   68 | 2019-01-08 00:00:00 |
    |        2 | G1463354 |   68 | 2020-01-07 00:00:00 |
    |        2 | G1463358 |   92 | 2020-01-07 00:00:00 |
    |        2 | G1463383 |   92 | 2020-01-07 00:00:00 |
    |        2 | G1463388 |   92 | 2020-01-07 00:00:00 |
    +----------+----------+------+---------------------+
    6 rows in set
```

使用内连接进行查询

```
mysql> select R.subjectId 课程编号,studentNo 学号,studentResult 成绩,examDate 考试日期,
    -> subjectName 课程名称
    -> from result as R
    -> join subject as SJ on R.subjectId = SJ.subjectId
    -> where subjectName = '大学英语';
    +----------+----------+------+---------------------+----------+
    | 课程编号 | 学号     | 成绩 | 考试日期            | 课程名称 |
    +----------+----------+------+---------------------+----------+
    |        2 | G1463337 |   92 | 2019-01-08 00:00:00 | 大学英语 |
    |        2 | G1463342 |   68 | 2019-01-08 00:00:00 | 大学英语 |
    |        2 | G1463354 |   68 | 2020-01-07 00:00:00 | 大学英语 |
    |        2 | G1463358 |   92 | 2020-01-07 00:00:00 | 大学英语 |
    |        2 | G1463383 |   92 | 2020-01-07 00:00:00 | 大学英语 |
    |        2 | G1463388 |   92 | 2020-01-07 00:00:00 | 大学英语 |
    +----------+----------+------+---------------------+----------+
    6 rows in set
```

6.3.2 IN 子查询

- 在实际应用中
  - 使用“=、>、<”等比较运算符时，要求子查询只能返回一条或空的记录
  - 有的要求子查询返回多条记录值
  - 并依赖多条记录继续处理后续问题
- 使用IN关键字可以使父查询匹配子查询返回的多个单列值

【演示示例 6-12】查询学习了“大学英语”课程的学生信息，显示学号、姓名

```
mysql> select studentNo 学号,studentName 姓名
    -> from student
    -> where studentNo in (
    -> select studentNo from result
    -> where subjectId = (
    -> select subjectId from subject
    -> where subjectName = '大学英语'));
    +----------+--------+
    | 学号     | 姓名   |
    +----------+--------+
    | G1463337 | 高伟   |
    | G1463342 | 胡俊文 |
    | G1463354 | 陈大伟 |
    | G1463358 | 温海南 |
    | G1463383 | 钱嫣然 |
    | G1463388 | 卫丹丹 |
    +----------+--------+
    6 rows in set
```

分步操作：

1.查询“大学英语”课程编号

```
mysql> select subjectId from subject where subjectName = '大学英语';
    +-----------+
    | subjectId |
    +-----------+
    |         2 |
    +-----------+
```

2.查询有“大学英语”成绩的同学

```
mysql> select studentNo from result where subjectId = 2;
    +-----------+
    | studentNo |
    +-----------+
    | G1463337  |
    | G1463342  |
    | G1463354  |
    | G1463358  |
    | G1463383  |
    | G1463388  |
    +-----------+
    6 rows in set
```

3.根据学号查询对应的学生信息

```
mysql> select studentNo 学号,studentName 姓名
    -> from student
    -> where studentNo in('G1463337','G1463342',
    -> 'G1463354','G1463358','G1463383','G1463388');
    +----------+--------+
    | 学号     | 姓名   |
    +----------+--------+
    | G1463337 | 高伟   |
    | G1463342 | 胡俊文 |
    | G1463354 | 陈大伟 |
    | G1463358 | 温海南 |
    | G1463383 | 钱嫣然 |
    | G1463388 | 卫丹丹 |
    +----------+--------+
    6 rows in set
```

6.3.3 EXISTS 子查询

- EXISTS关键字用于检测数据是否存在
- EXISTS语句
  - 在学习创建数据库和创建表的语句时已经使用
  - 是一个检测是否存在的子查询语句
- EXISTS（子查询）取值
  - 如果子查询的结果非空，则EXISTS（子查询）将返回真（TRUE）
  - 如果子查询的结果为空，则返回假（FALSE）

【演示示例 6-13】处理某门课程的成绩

【演示示例6-13】根据“C语言程序设计”课程成绩进行处理：若不及格，所有人每人加5分，

若加分后超过100分的不得加分。

1.复制成绩表为 rs9

```
mysql> create table rs9 as(select * from result);
    Query OK, 42 rows affected
    Records: 42  Duplicates: 0  Warnings: 0
```

2.查询课程成绩情况

```
mysql> select studentNo 学号,subjectId 课程编号,studentresult 成绩
    -> ,examdate 考试日期
    -> from result
    -> where subjectId = (select subjectId from subject where subjectName = 'C语言程序设计');
    +----------+----------+------+---------------------+
    | 学号     | 课程编号 | 成绩 | 考试日期            |
    +----------+----------+------+---------------------+
    | G1463337 |        1 |   82 | 2019-11-20 00:00:00 |
    | G1463337 |        1 |   98 | 2020-01-05 00:00:00 |   ——修改成绩为98 原来是90
    | G1463342 |        1 |   86 | 2020-01-05 00:00:00 |
    | G1463354 |        1 |   52 | 2019-11-20 00:00:00 |
    | G1463354 |        1 |   67 | 2020-01-05 00:00:00 |
    | G1463358 |        1 |   65 | 2020-01-05 00:00:00 |
    | G1463383 |        1 |   88 | 2019-11-20 00:00:00 |
    | G1463383 |        1 |   87 | 2020-01-05 00:00:00 |
    | G1463388 |        1 |   80 | 2019-11-20 00:00:00 |
    | G1463388 |        1 |   78 | 2020-01-09 00:00:00 |
    +----------+----------+------+---------------------+
    10 rows in set
```

3.更新成绩，大于95的不加分

```
mysql> select studentNo 学号,subjectId 课程编号,studentresult 成绩,examdate 考试日期
    -> from result
    -> where subjectId = 1 AND EXISTS (
    -> SELECT * FROM Result91 WHERE studentResult<60 AND subjectId = 1)
    -> AND StudentResult<=95;
      +----------+----------+------+---------------------+
      | 学号     | 课程编号 | 成绩 | 考试日期            |
      +----------+----------+------+---------------------+
      | G1463337 |        1 |   82 | 2019-11-20 00:00:00 |
      | G1463342 |        1 |   86 | 2020-01-05 00:00:00 |
      | G1463354 |        1 |   52 | 2019-11-20 00:00:00 |   ——成绩为98的不显示
      | G1463354 |        1 |   67 | 2020-01-05 00:00:00 |
      | G1463358 |        1 |   65 | 2020-01-05 00:00:00 |
      | G1463383 |        1 |   88 | 2019-11-20 00:00:00 |
      | G1463383 |        1 |   87 | 2020-01-05 00:00:00 |
      | G1463388 |        1 |   80 | 2019-11-20 00:00:00 |
      | G1463388 |        1 |   78 | 2020-01-09 00:00:00 |
      +----------+----------+------+---------------------+
      9 rows in set
```

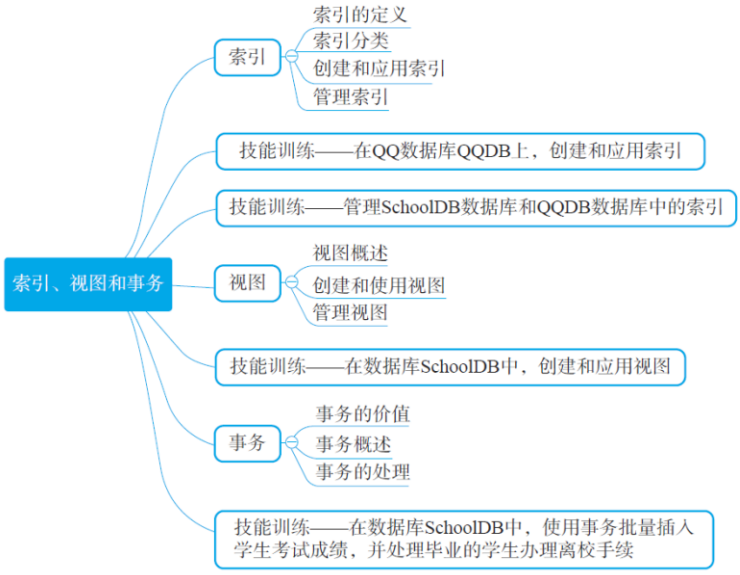
4.将52分修改为60，则显示结果为空集

```
mysql> select studentNo 学号,subjectId 课程编号,studentresult 成绩,examdate 考试日期
    -> from result
    -> where subjectId = 1 AND EXISTS (
    -> SELECT * FROM Result91 WHERE studentResult<60 AND subjectId = 1);
    Empty set

mysql> select exists(SELECT * FROM Result91 WHERE studentResult<60 AND subjectId = 1) 结果;
      +------+
      | 结果 |
      +------+
      |    0 |
      +------+
      1 row in set
```

课后反思	

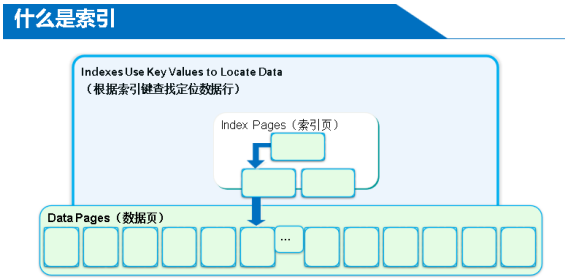
课题	第 07 章 索引、视图和事务		
教学目标	➢ 理解索引、视图和事务的概念和价值 ➢ 掌握索引的创建并管理和应用索引 ➢ 掌握视图的创建并管理和应用视图 ➢ 掌握事务的创建并应用事务解决实际问题		
重点难点	➢ 灵活应用视图和索引解决实际问题 ➢ 事务的创建、提交和回滚机制的理解与应用		
建议学时	6	教学教具	一体化教学实训室
主要教法	讲授演示法		
思政融入	➢ 培养数据安全意识：通过事务的 ACID 特性学习，理解数据一致性和完整性对系统可靠性的重要性，树立“数据安全无小事”的理念，呼应企业级应用中对数据质量的严格要求。 ➢ 强化规范开发理念：在索引和视图的创建操作中，强调语法规范和命名规范，培养学生严谨、细致的开发习惯，理解“规范是效率和协作的基础”。 ➢ 树立责任担当意识：结合事务在金融、电商等关键领域的应用案例，让学生认识到数据库操作对业务安全的重要影响，培养“精益求精、对结果负责”的职业素养。		

教　学　过　程



8.1 索引

1. 索引的概念与作用：类比书籍目录，说明索引是提高查询效率的“目录”，减少数据扫描范围。



2. 索引的分类：

① 按存储结构：B+树索引（默认）、哈希索引等；② 按功能：主键索引、唯一索引、普通索引、复合索引。重点讲解 B+树索引的工作原理（简化图示）。

3. 索引的操作：① 创建：CREATE INDEX、ALTER TABLE；② 查看：SHOW INDEX；③ 删除：DROP INDEX。

4. 索引的应用

- 创建索引后，可以像在新华字典中查找字词一样
  - 可以选择拼音查找方式或笔画查找方式
- 索引主要是为查询服务
  - 如果查询的条件字段上未建立索引
    - ✓ 则使用 WHERE 条件，在表中对所有的记录就进行比较匹配
  - 如果在查询的条件字段上创建了索引
    - ✓ 则查询就会自动依据索引来搜索
    - ✓ 而不需要对表中所有的记录进行比较匹配，从而提高查询的效率

【演示示例8-1】为学生表姓名列创建索引，并查找姓王的学生信息

1.没有加索引的情况

```
mysql> select * from student where studentName like '王%';
+-----------+-----------+-------------+-----+---------+-------------+-----------------+------------+-----------------+--------------------+
| studentNo | loginPwd  | studentName | sex | gradeId | phone       | address         | bornDate   | email           | identityCard       |
+-----------+-----------+-------------+-----+---------+-------------+-----------------+------------+-----------------+--------------------+
| G1263201  | 1         | 王子洋      | 男  |       5 | 18655290000 | 安徽省蚌埠市    | 1993-08-07 | wzy@163.com     | 340423199308070000 |
| G1363300  | 1         | 王超        | 男  |       3 | 18123560000 | 上海市闵行区    | 1993-04-30 | wangchao@126.com | 340409199304300000 |
+-----------+-----------+-------------+-----+---------+-------------+-----------------+------------+-----------------+--------------------+
2 rows in set

mysql> explain select * from student where studentName like '王%';
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   14 |    11.11 | Using where |
+----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set

rows ——14
```

2.添加索引

```
mysql> ALTER TABLE student
    -> ADD INDEX IX_studentName(studentName);    ——修改表格，添加姓名索引
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain select * from student where studentName like '王%';
+----+-------------+---------+------------+-------+---------------+---------------+---------+------+------+----------+-----------------------+
| id | select_type | table   | partitions | type  | possible_keys | key           | key_len | ref  | rows | filtered | Extra                 |
+----+-------------+---------+------------+-------+---------------+---------------+---------+------+------+----------+-----------------------+
|  1 | SIMPLE      | student | NULL       | range | IX_studentName | IX_studentName | 102     | NULL |    2 |      100 | Using index condition |
+----+-------------+---------+------------+-------+---------------+---------------+---------+------+------+----------+-----------------------+
1 row in set

rows —— 2
```

【技能训练8-1】

1、查询昵称以“笨”开头的用户信息

（1）未加索引

```
mysql> select * from baseinfo where NickName like '笨%';
+-----------+----------+-----+-----+----------+------+-----------------------+-----------------+
| QQID      | NickName | Sex | Age | Province | City | Address               | Phone           |
+-----------+----------+-----+-----+----------+------+-----------------------+-----------------+
| 622009495 | 笨丫头   |   1 |  90 | 甘肃省   | 兰州 | 兰州市城关区庆阳路 | 841800178810000 |
+-----------+----------+-----+-----+----------+------+-----------------------+-----------------+
1 row in set

mysql> explain select * from baseinfo where NickName like '笨%';
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | baseinfo | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  402 |    11.11 | Using where |
+----+-------------+----------+------------+------+---------------+------+---------+------+------+----------+-------------+
1 row in set

rows —— 402
```

（2）添加昵称索引

```
mysql> ALTER TABLE BaseInfo
    -> ADD INDEX IX_nkname(NickName);    ——修改表格，添加昵称索引
Query OK, 0 rows affected
Records: 0  Duplicates: 0  Warnings: 0

mysql> explain select * from baseinfo where NickName like '笨%';
+----+-------------+----------+------------+-------+---------------+-----------+---------+------+------+----------+-----------------------+
| id | select_type | table    | partitions | type  | possible_keys | key       | key_len | ref  | rows | filtered | Extra                 |
+----+-------------+----------+------------+-------+---------------+-----------+---------+------+------+----------+-----------------------+
|  1 | SIMPLE      | baseinfo | NULL       | range | IX_nkname     | IX_nkname | 767     | NULL |    1 |      100 | Using index condition |
+----+-------------+----------+------------+-------+---------------+-----------+---------+------+------+----------+-----------------------+
1 row in set

rows —— 1
```

【技能训练8-2】

1、查看 shchooldb 中 4 张表的所有索引

```
mysql> show index from student;
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table   | Non_unique | Key_name      | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| student |          0 | PRIMARY       |            1 | studentNo   | A         |          14 | NULL     | NULL   |      | BTREE      |         |               |
| student |          0 | UN_ic         |            1 | identityCard| A         |          14 | NULL     | NULL   | YES  | BTREE      |         |               |
| student |          1 | FK_stugid     |            1 | gradeId     | A         |           3 | NULL     | NULL   | YES  | BTREE      |         |               |
| student |          1 | IX_studentName|            1 | studentName | A         |          14 | NULL     | NULL   |      | BTREE      |         |               |
+---------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
4 rows in set

mysql> show index from grade;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| grade |          0 | PRIMARY  |            1 | gradeId     | A         |           6 | NULL     | NULL   |      | BTREE      |         |               |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
1 row in set

mysql> show index from subject;
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table   | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| subject |          0 | PRIMARY  |            1 | subjectId   | A         |          14 | NULL     | NULL   |      | BTREE      |         |               |
| subject |          1 | Fk_sg    |            1 | gradeId     | A         |           5 | NULL     | NULL   |      | BTREE      |         |               |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
2 rows in set

mysql> show index from result;
+--------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table  | Non_unique | Key_name      | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+--------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| result |          0 | PRIMARY       |            1 | id          | A         |          42 | NULL     | NULL   |      | BTREE      |         |               |
| result |          1 | FK_subjectId  |            1 | subjectId   | A         |           8 | NULL     | NULL   |      | BTREE      |         |               |
| result |          1 | FK_studentNo  |            1 | studentNo   | A         |          14 | NULL     | NULL   |      | BTREE      |         |               |
+--------+------------+---------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
3 rows in set
```

2、查看 QQDB 中 3 张表的所有索引

```
mysql> show index from baseinfo;
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| baseinfo |          0 | PRIMARY  |            1 | QQID        | A         |         402 | NULL     | NULL   |      | BTREE      |         |               |
| baseinfo |          1 | IX_nkname|            1 | NickName    | A         |         397 | NULL     | NULL   |      | BTREE      |         |               |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
2 rows in set

mysql> show index from qquser;
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| qquser |          0 | PRIMARY  |            1 | QQID        | A         |         403 | NULL     | NULL   |      | BTREE      |         |               |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
1 row in set

mysql> show index from relation;
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| Table    | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
| relation |          0 | PRIMARY  |            1 | QQID        | A         |         286 | NULL     | NULL   |      | BTREE      |         |               |
| relation |          0 | PRIMARY  |            2 | RelationQQID| A         |         421 | NULL     | NULL   |      | BTREE      |         |               |
| relation |          1 | fk_qqnoB |            1 | RelationQQID| A         |         400 | NULL     | NULL   |      | BTREE      |         |               |
+----------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
3 rows in set
```

3、删除 SchoolDB 中表 student 的索引 IX_address

```
mysql> alter table student
    -> add index IX_address(address);   添加 地址 索引
       Query OK, 0 rows affected
       Records: 0  Duplicates: 0  Warnings: 0

mysql> explain select * from student where address like '%地址%';
       +----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
       | id | select_type | table   | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra       |
       +----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
       |  1 | SIMPLE      | student | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   14 |    11.11 | Using where |
       +----+-------------+---------+------------+------+---------------+------+---------+------+------+----------+-------------+
       1 row in set

       rows —— 14 所有记录都查询  like '%地址%'

mysql> explain select * from student where address like '地址%';
       +----+-------------+---------+------------+-------+---------------+------------+---------+------+------+----------+-----------------------+
       | id | select_type | table   | partitions | type  | possible_keys | key        | key_len | ref  | rows | filtered | Extra                 |
       +----+-------------+---------+------------+-------+---------------+------------+---------+------+------+----------+-----------------------+
       |  1 | SIMPLE      | student | NULL       | range | IX_address    | IX_address | 513     | NULL |    2 |      100 | Using index condition |
       +----+-------------+---------+------------+-------+---------------+------------+---------+------+------+----------+-----------------------+
       1 row in set

       rows —— 2   like '地址%'


mysql> show index from student;   ——删除 地址 索引前情况
       +---------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
       | Table   | Non_unique | Key_name     | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
       +---------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
       | student |          0 | PRIMARY      |            1 | studentNo   | A         |          14 | NULL     | NULL   |      | BTREE      |         |               |
       | student |          0 | UN_ic        |            1 | identityCard| A         |          14 | NULL     | NULL   | YES  | BTREE      |         |               |
       | student |          1 | FK_stugid    |            1 | gradeId     | A         |           3 | NULL     | NULL   | YES  | BTREE      |         |               |
       | student |          1 | IX_studentName|           1 | studentName | A         |          14 | NULL     | NULL   |      | BTREE      |         |               |
       | student |          1 | IX_address   |            1 | address     | A         |          13 | NULL     | NULL   | YES  | BTREE      |         |               |
       +---------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
       5 rows in set

mysql> drop index IX_address on student;       删除 地址 索引
       Query OK, 0 rows affected
       Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from student;          ——删除 地址 索引后情况
       +---------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
       | Table   | Non_unique | Key_name     | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment |
       +---------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
       | student |          0 | PRIMARY      |            1 | studentNo   | A         |          14 | NULL     | NULL   |      | BTREE      |         |               |
       | student |          0 | UN_ic        |            1 | identityCard| A         |          14 | NULL     | NULL   | YES  | BTREE      |         |               |
       | student |          1 | FK_stugid    |            1 | gradeId     | A         |           3 | NULL     | NULL   | YES  | BTREE      |         |               |
       | student |          1 | IX_studentName|           1 | studentName | A         |          14 | NULL     | NULL   |      | BTREE      |         |               |
       +---------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+
       4 rows in set
```

8.2 视图

1. 视图的定义与特点：虚拟表，基于基表的查询结果，不存储实际数据，随基表变化而变化。

2. 视图的作用：

① 简化查询（隐藏复杂关联）；② 权限控制（隐藏敏感字段）；③ 数据一致性（统一查询逻辑）。

3. 视图的操作：

① 创建：CREATE VIEW；② 查询：SELECT * FROM 视图名；③ 修改：CREATE OR REPLACE VIEW、ALTER VIEW；④ 删除：DROP VIEW。

4.创建视图

```
mysql> CREATE VIEW vw_stuent_result
    -> as
    -> select studentName 姓名,S.studentNO 学号,studentResult 成绩,subjectName 课程编号,examDate 考试日期
    -> from student S join Result R on S.studentNO = R.studentNo
    -> join subject SJ on SJ.subjectId = R.subjectId
    -> where R.subjectId = 13 and examDate = '2019-11-15';
Query OK, 0 rows affected
```

5.查看视图

```
mysql> select * from vw_stuent_result;
+------+---------+------+--------------+---------------------+
| 姓名 | 学号    | 成绩 | 课程编号     | 考试日期            |
+------+---------+------+--------------+---------------------+
| 王子洋 | G1263201 |   76 | 软件测试技术 | 2019-11-15 00:00:00 |
| 张旗 | G1263382 |   79 | 软件测试技术 | 2019-11-15 00:00:00 |
| 项宇 | G1263458 |   92 | 软件测试技术 | 2019-11-15 00:00:00 |
+------+---------+------+--------------+---------------------+
3 rows in set
```

【演示示例8-2】为任课老师创建成绩视图：学生姓名、学号、课程名称、成绩、最后一次参加课程考试的日期

软件测试技术 课程为例

分步执行：

1.查询课程 软件测试技术 编号

```
mysql> select * from subject where subjectName = '软件测试技术';
+-----------+--------------+-----------+---------+
| subjectId | subjectName  | classHour | gradeId |
+-----------+--------------+-----------+---------+
|        13 | 软件测试技术 |        32 |       5 |
+-----------+--------------+-----------+---------+
```

2.查询课程编号为13（软件测试技术）的最近一次考试时间

```
mysql> select MAX(examDate) from result where subjectId = 13;
+---------------------+
| MAX(examDate)       |
+---------------------+
| 2019-11-15 00:00:00 |
+---------------------+
```

3.三表连接显示结果

```
mysql> select studentName 姓名,S.studentNO 学号,studentResult 成绩,subjectName 课程编号,examDate 考试日期
    -> from student S join Result R on S.studentNO = R.studentNo
    -> join subject SJ on SJ.subjectId = R.subjectId
    -> where R.subjectId = 13 and examDate = '2019-11-15';
+------+---------+------+--------------+---------------------+
| 姓名 | 学号    | 成绩 | 课程编号     | 考试日期            |
+------+---------+------+--------------+---------------------+
| 王子洋 | G1263201 |   76 | 软件测试技术 | 2019-11-15 00:00:00 |
| 张旗 | G1263382 |   79 | 软件测试技术 | 2019-11-15 00:00:00 |
| 项宇 | G1263458 |   92 | 软件测试技术 | 2019-11-15 00:00:00 |
+------+---------+------+--------------+---------------------+
3 rows in set
```

【技能训练8-3】创建视图，统计每个同学所有课程的总分：学号、姓名、总分

1.先查询结果

```
mysql> select R.studentNo 学号,studentName 姓名,sum(studentResult) 总分
    -> from Result R join student S on R.studentNo = S.studentNo
    -> group by R.studentNO;
+----------+--------+--------+
| 学号     | 姓名   | 总分   |
+----------+--------+--------+
| G1263201 | 王子洋 | 164.00 |
| G1263382 | 张旗   | 135.00 |
| G1263458 | 项宇   |  92.00 |
| G1363278 | 胡保蜜 | 209.00 |
| G1363300 | 王超   | 196.00 |
| G1363301 | 党志鹏 | 297.00 |
| G1363302 | 胡仲友 | 223.00 |
| G1363303 | 朱晓燕 | 229.00 |
| G1463337 | 高伟   | 320.00 |
| G1463342 | 胡俊文 | 154.00 |
| G1463354 | 陈大伟 | 262.00 |
| G1463358 | 温海南 | 157.00 |
| G1463383 | 钱嫣然 | 356.00 |
| G1463388 | 卫丹丹 | 333.00 |
+----------+--------+--------+
14 rows in set
```

2.创建视图

```
mysql> CREATE VIEW vw_SR as
    -> select R.studentNo 学号,studentName 姓名,sum(studentResult) 总分
    -> from Result R join student S on R.studentNo = S.studentNo
    -> group by R.studentNO;
Query OK, 0 rows affected
```

3.查看视图——无条件

```
mysql> select * from vw_sr;
+----------+--------+--------+
| 学号     | 姓名   | 总分   |
+----------+--------+--------+
| G1263201 | 王子洋 | 164.00 |
| G1263382 | 张旗   | 135.00 |
| G1263458 | 项宇   |  92.00 |
| G1363278 | 胡保蜜 | 209.00 |
| G1363300 | 王超   | 196.00 |
| G1363301 | 党志鹏 | 297.00 |
| G1363302 | 胡仲友 | 223.00 |
| G1363303 | 朱晓燕 | 229.00 |
| G1463337 | 高伟   | 320.00 |
| G1463342 | 胡俊文 | 154.00 |
| G1463354 | 陈大伟 | 262.00 |
| G1463358 | 温海南 | 157.00 |
| G1463383 | 钱嫣然 | 356.00 |
| G1463388 | 卫丹丹 | 333.00 |
+----------+--------+--------+
14 rows in set
```

4.查看 王超 的情况

```
mysql> select * from vw_sr where 姓名 = '王超';
+----------+------+--------+
| 学号     | 姓名 | 总分   |
+----------+------+--------+
| G1363300 | 王超 | 196.00 |
+----------+------+--------+
1 row in set
```

5. 查看 总分 > 200 的情况

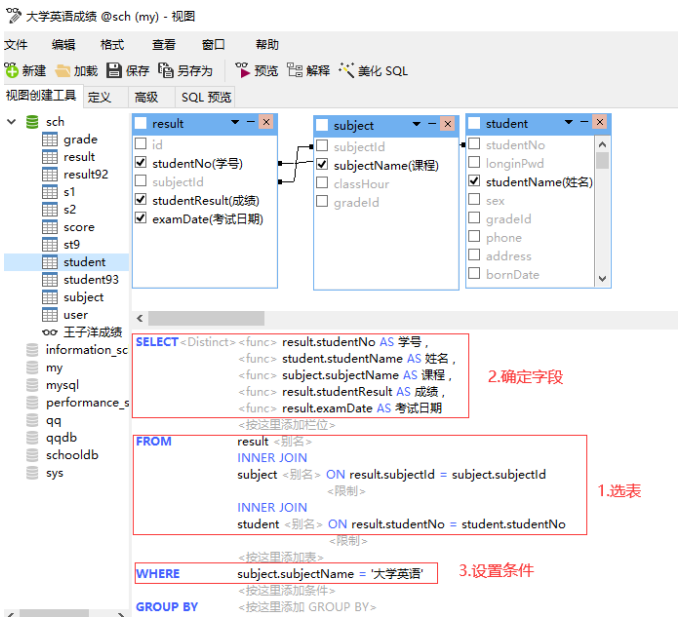
```
mysql> select * from vw_sr where 总分 > 200;
+----------+--------+--------+
| 学号     | 姓名   | 总分   |
+----------+--------+--------+
| G1363278 | 胡保蜜 | 209.00 |
| G1363301 | 党志鹏 | 297.00 |
| G1363302 | 胡仲友 | 223.00 |
| G1363303 | 朱晓燕 | 229.00 |
| G1463337 | 高伟   | 320.00 |
| G1463354 | 陈大伟 | 262.00 |
| G1463383 | 钱嫣然 | 356.00 |
| G1463388 | 卫丹丹 | 333.00 |
+----------+--------+--------+
8 rows in set
```

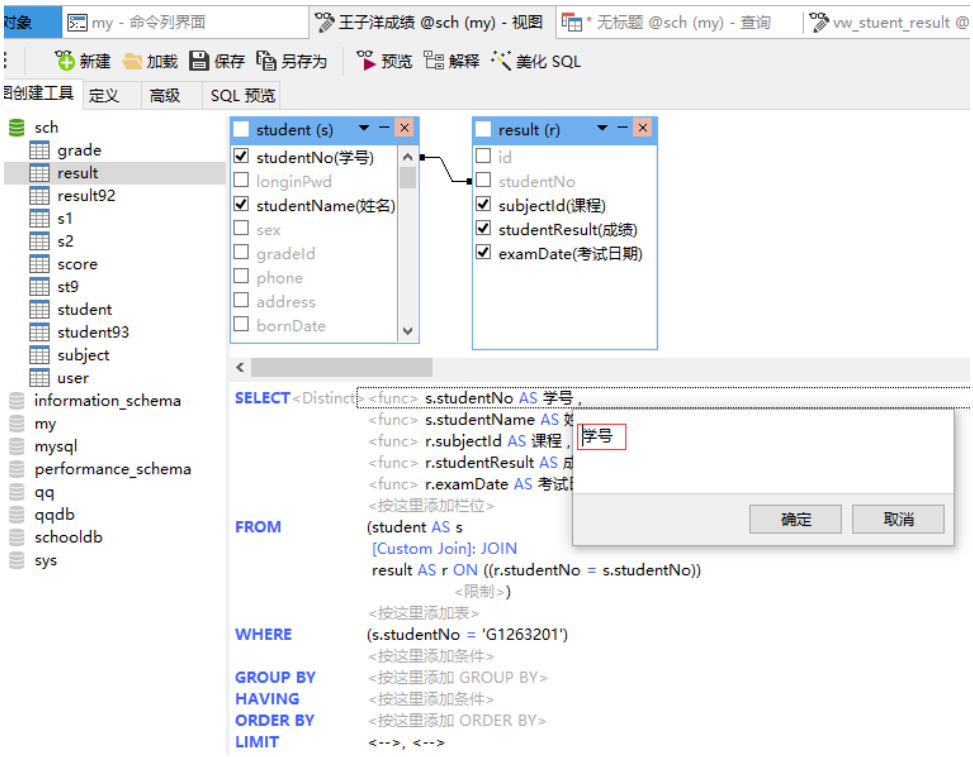
6. 将视图结果按总分降序

```
mysql> select * from vw_sr order by 总分 desc;
+----------+--------+--------+
| 学号     | 姓名   | 总分   |
+----------+--------+--------+
| G1463383 | 钱嫣然 | 356.00 |
| G1463388 | 卫丹丹 | 333.00 |
| G1463337 | 高伟   | 320.00 |
| G1363301 | 党志鹏 | 297.00 |
| G1463354 | 陈大伟 | 262.00 |
| G1363303 | 朱晓燕 | 229.00 |
| G1363302 | 胡仲友 | 223.00 |
| G1363278 | 胡保蜜 | 209.00 |
| G1363300 | 王超   | 196.00 |
| G1263201 | 王子洋 | 164.00 |
| G1463358 | 温海南 | 157.00 |
| G1463342 | 胡俊文 | 154.00 |
| G1263382 | 张旗   | 135.00 |
| G1263458 | 项宇   |  92.00 |
+----------+--------+--------+
14 rows in set
```

创建“C语言程序设计成绩”视图，显示 C语言程序设计 成绩情况

```
mysql> create view C语言程序设计成绩 as
    -> select S.studentNo 学号,studentName 姓名,subjectName 课程,studentResult 成绩
    -> from result R join student S on R.studentNo = S.studentNo
    -> join subject SJ on R.subjectId = SJ.subjectId
    -> where subjectName = 'C语言程序设计';
Query OK, 0 rows affected

mysql> select * from C语言程序设计成绩;
+----------+--------+--------------+------+
| 学号     | 姓名   | 课程         | 成绩 |
+----------+--------+--------------+------+
| G1463337 | 高伟   | C语言程序设计 |   82 |
| G1463337 | 高伟   | C语言程序设计 |   90 |
| G1463342 | 胡俊文 | C语言程序设计 |   86 |
| G1463354 | 陈大伟 | C语言程序设计 |   52 |
| G1463354 | 陈大伟 | C语言程序设计 |   67 |
| G1463358 | 温海南 | C语言程序设计 |   65 |
| G1463383 | 钱嫣然 | C语言程序设计 |   88 |
| G1463383 | 钱嫣然 | C语言程序设计 |   87 |
| G1463388 | 卫丹丹 | C语言程序设计 |   80 |
| G1463388 | 卫丹丹 | C语言程序设计 |   78 |
+----------+--------+--------------+------+
10 rows in set
```





8.3 事务

1. 事务的概念：一组不可分割的 SQL 操作集合，要么全部执行，要么全部不执行。

2. 事务的 ACID 特性：

① 原子性（Atomicity）：不可分割；② 一致性（Consistency）：执行前后数据一致；③ 隔离性（Isolation）：并发执行互不干扰；④ 持久性（Durability）：执行结果永久保存。重点讲解原子性和一致性。

3. 事务的操作：

① 开启：START TRANSACTION；② 提交：COMMIT；③ 回滚：ROLLBACK；④ 保存点：SAVEPOINT（可选）。

4. 实操演示：以银行转账案例（user 表：id, name, balance），演示正常转账（提交事务）和转账失败（回滚事务）的完整过程，直观展示 ACID 特性。5. 案例拓展：讲解某支付系统因事务未正确使用导致的资金异常案例，分析后果。

【演示示例 8-3】模拟银行转账业务

需求

- 数据库SchoolDB中
  - 创建测试用的账户表bank
  - 添加约束和测试数据
  - 实现从张三账户转账到李四账户

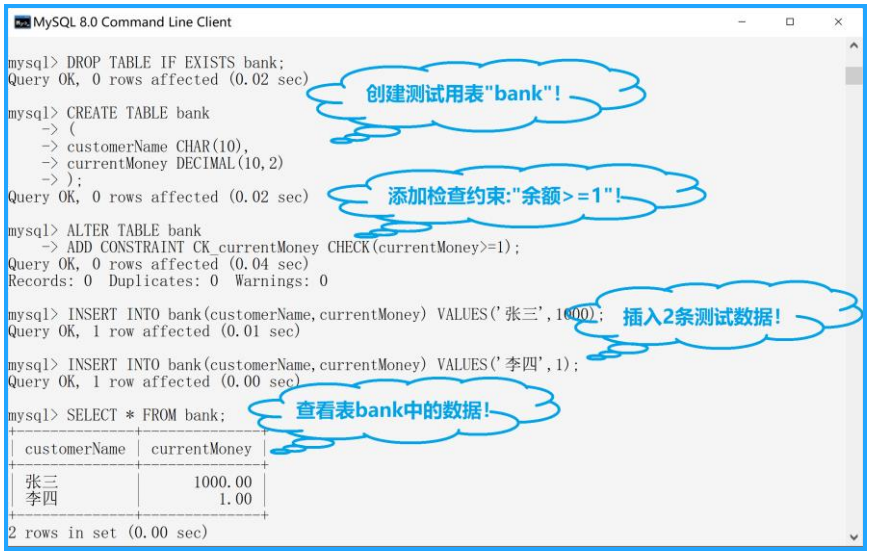
关键步骤

- 创建账户表bank
  - 存放用户张三和李四的账户信息
  - 只设计2个字段：姓名和余额
  - 添加约束： “帐户余额不能少于1元”
  - 插入2条测试数据： “('张三',1000)” 和 “('李四',1)”
- 使用UPDATE语句
  - 张三的账户减少1000元，李四的账户增加1000元
- 对比转账前后的余额，分析错误发生的原因

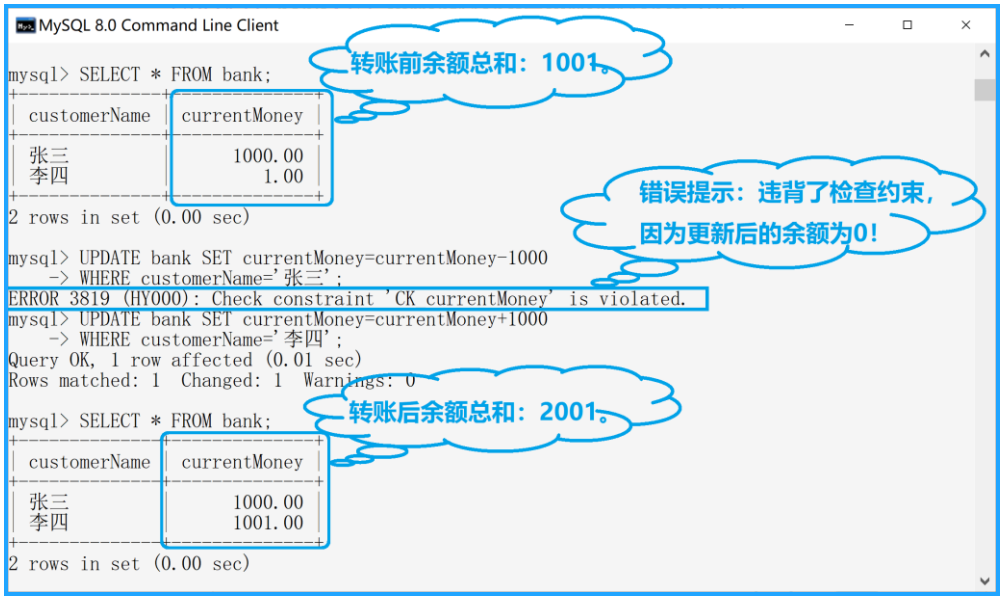
```
USE SchoolDB;
DROP TABLE IF EXISTS bank;
CREATE TABLE bank
(
      customerName CHAR(10),
      currentMoney DECIMAL(10,2));
ALTER TABLE bank
      ADD CONSTRAINT CK_currentMoney CHECK(currentMoney>=1);
INSERT INTO bank(customerName,currentMoney)
      VALUES('张三',1000);
INSERT INTO bank(customerName,currentMoney)
      VALUES('李四',1);
SELECT * FROM bank;
```

(2) 利用'党志鹏'的出生日期查询比他小的结果

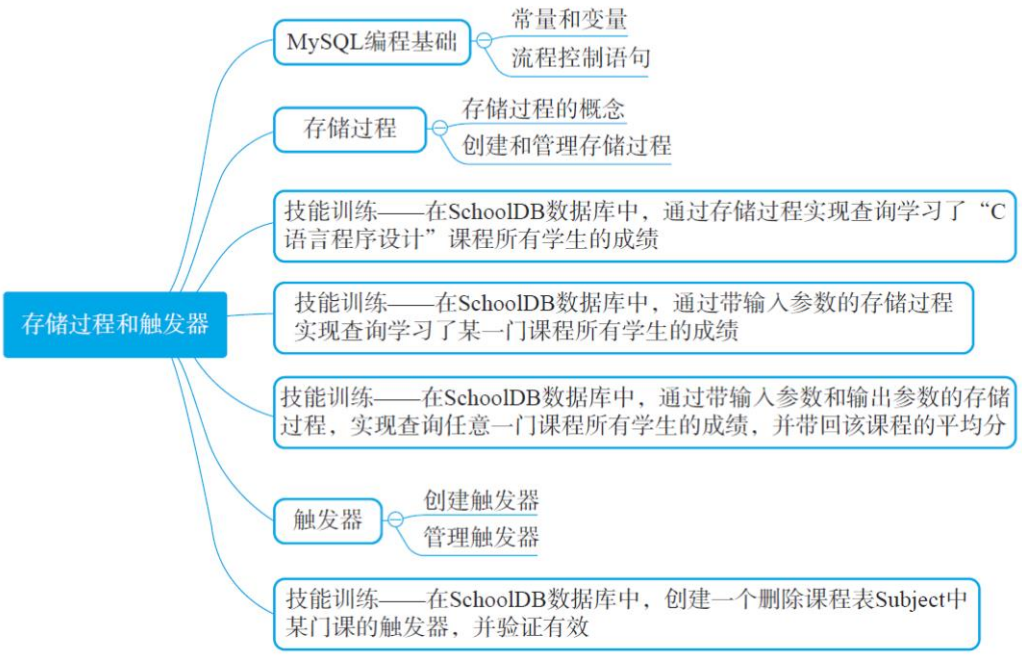
```
mysql> select studentNo 学号,studentName 姓名,sex 性别,bornDate 出生日期
   -> from student
   -> where bornDate > '1994-12-20';
     +----------+--------+------+------------+
     | 学号     | 姓名   | 性别 | 出生日期   |
     +----------+--------+------+------------+
     | G1463337 | 高伟   | 男   | 1995-06-07 |
     | G1463342 | 胡俊文 | 男   | 1995-04-20 |
     | G1463354 | 陈大伟 | 男   | 1995-08-23 |
     | G1463358 | 温海南 | 男   | 1995-01-30 |
     | G1463388 | 卫丹丹 | 女   | 1995-04-17 |
     +----------+--------+------+------------+
     5 rows in set
```



```
SELECT * FROM bank;
UPDATE bank SET currentMoney=currentMoney-1000
    WHERE customerName='张三';
UPDATE bank SET currentMoney=currentMoney+1000
    WHERE customerName='李四';
SELECT * FROM bank;
```

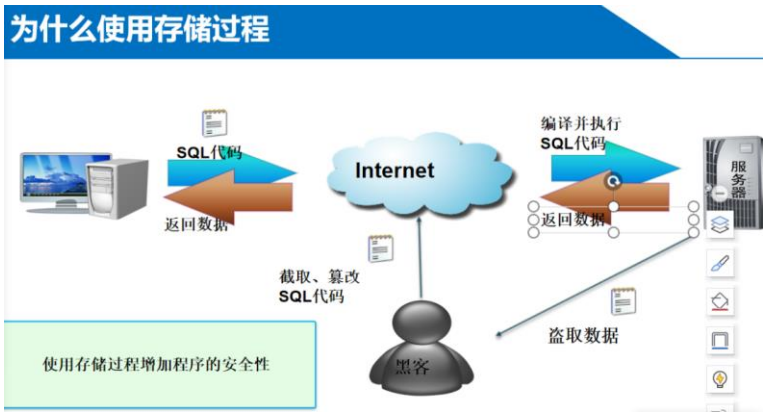


课后反思	

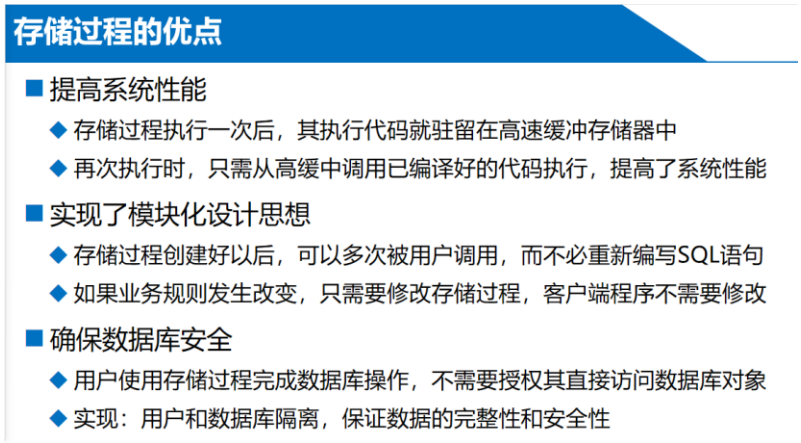
<table>
<tr><td>课题</td><td colspan="3">第 08 章 存储过程和触发器</td></tr>
<tr><td>教学目标</td><td colspan="3">➢ 理解编程访问数据库的概念和途径
➢ 理解存储过程的概念和工作过程
➢ 掌握存储过程的创建和管理
➢ 掌握触发器的创建和管理
➢ 灵活应用存储过程和触发器解决实际问题</td></tr>
<tr><td>重点难点</td><td colspan="3">➢ 通过编程来访问数据库
➢ 使用存储过程和触发器解决实际问题</td></tr>
<tr><td>建议学时</td><td>9</td><td>教学教具</td><td>一体化教学实训室</td></tr>
<tr><td>主要教法</td><td colspan="3">讲授演示法</td></tr>
<tr><td>思政融入</td><td colspan="3">➢ 强化代码复用与规范意识：通过存储过程的模块化设计学习，理解“复用是提升效率、降低冗余的关键”，培养规范编写注释、统一代码风格的开发习惯。
➢ 培养数据完整性守护意识：通过触发器实现数据校验、日志记录等功能的学习，认识到技术对数据质量的保障作用，树立“数据可信源于严谨设计”的理念。
➢ 树立责任与安全意识：结合存储过程权限控制、触发器异常处理的案例，让学生认识到数据库对象设计对系统安全的影响，培养“关注细节、防范风险”的职业素养。</td></tr>
<tr><td colspan="4">教 学 过 程
 
</td></tr>
</table>

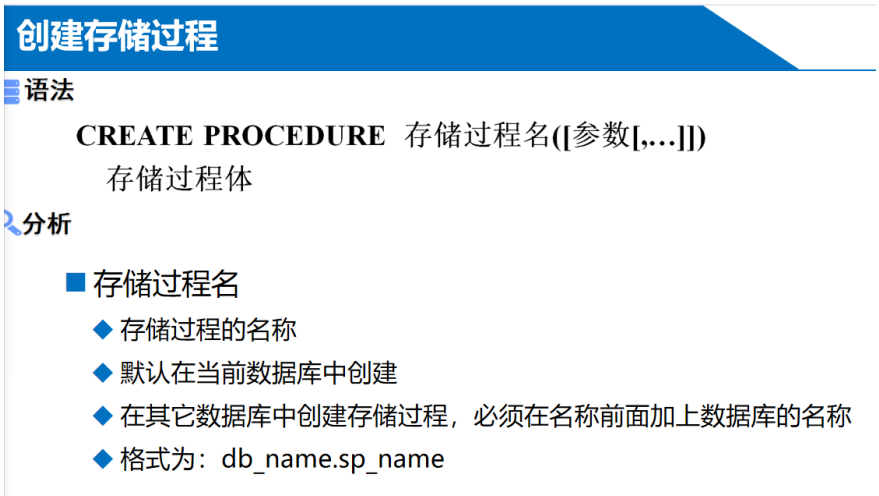
9.1 存储过程

1. 为什么需存储过程，什么是存储过程



思政点： 数据库中的数据是高度共享的，如果没有采取正确的确保数据库安全技术，必然会引起数据的泄露等问题。存储过程就是一种重要的保障机制。同理，人类社会也需要保障机制，从而引申学生尊重和遵守规则的品格。





思政点：存储过程名命名规范性，提高代码的可读性和可维护，从而引申到学生在社会中也应遵守行业规范，社会公序良俗，做任何事情都要”心中有度“。

调用存储过程

语法

```
CALL 存储过程名([参数[,...]]);
```

分析

- 存储过程名
  - 准备调用的存储过程的名称
  - 如果要调用其它数据库的存储过程，需要在前面加上该数据库的名称
- 参数：参数个数必须总是等于存储过程定义时的参数个数
- 括号：存储过程名后必须有一对括号，其中为参数列表

查看存储过程

查看当前数据库的存储过程

```
SHOW PROCEDURE STATUS;
```

查看存储过程的具体代码

```
SHOW CREATE PROCEDURE 存储过程名;
```

删除存储过程

语法

```
DROP PROCEDURE [IF EXISTS]存储过程名
```

分析

- 存储过程名：指要删除的存储过程名称
- IF EXISTS子句：如果存储过程不存在，防止发生错误
- 删除之前
  - 必须确认该存储过程没有任何依赖关系
  - 否则会导致其他与之关联的存储过程无法运行

9.2 存储过程

【演示示例9-1】查询 胡保蜜 同学的成绩：学号、姓名、课程名称、考试成绩，按课程编号升序

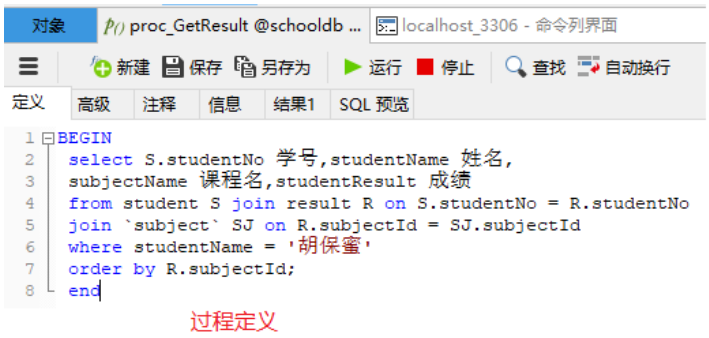
1. 创建存储过程

单击 navicat 的 查询——新建查询，在“查询编辑器”中输入以下内容：

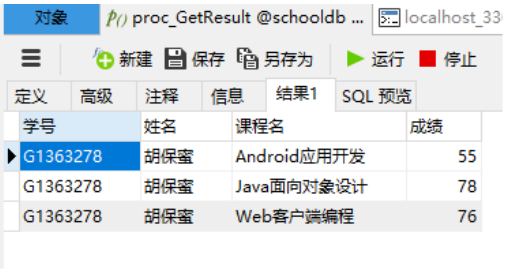
```
DROP PROCEDURE if EXISTS proc_GetResult;
delimiter //
create PROCEDURE proc_GetResult()
 BEGIN
 select S.studentNo 学号,studentName 姓名,subjectName 课程名,studentResult 成绩
 from student S join result R on S.studentNo = R.studentNo
 join `subject` SJ on R.subjectId = SJ.subjectId
 where studentName = '胡保蜜'
 order by R.subjectId;
 end //
```

2. 调用存储过程

```
mysql> call proc_getresult();
+----------+--------+--------------------+------+
| 学号     | 姓名   | 课程名             | 成绩 |
+----------+--------+--------------------+------+
| G1363278 | 胡保蜜 | Android应用开发    |   55 |
| G1363278 | 胡保蜜 | Java面向对象设计   |   78 |
| G1363278 | 胡保蜜 | Web客户端编程      |   76 |
+----------+--------+--------------------+------+
3 rows in set
```



过程定义



运行结果

【技能训练9-1】查询 C语言程序设计 的成绩：学号、姓名、课程名称、考试成绩，按课程编号升序

1.右击“函数”——新建函数——过程——完成，进入 函数定义界面

```
BEGIN
        #Routine body goes here...

END;
```

2.在中间输入函数功能代码：

```
BEGIN
 select S.studentNo 学号,studentName 姓名,subjectName 课程名,studentResult 成绩
 from student S join result R on S.studentNo = R.studentNo
 join `subject` SJ on R.subjectId = SJ.subjectId
 where subjectName = 'C语言程序设计'
 order by studentResult;

END
```

3.保存——输入过程名称：proc_GetSR

4.运行——单击“运行”，在“结果1”中查看结果

```
mysql> call proc_getsr;
+----------+--------+----------------+------+
| 学号     | 姓名   | 课程名         | 成绩 |
+----------+--------+----------------+------+
| G1463354 | 陈大伟 | C语言程序设计  |   52 |
| G1463358 | 温海南 | C语言程序设计  |   65 |
| G1463354 | 陈大伟 | C语言程序设计  |   67 |
| G1463388 | 卫丹丹 | C语言程序设计  |   78 |
| G1463337 | 高伟   | C语言程序设计  |   82 |
| G1463342 | 胡俊文 | C语言程序设计  |   86 |
| G1463383 | 钱嫣然 | C语言程序设计  |   87 |
| G1463383 | 钱嫣然 | C语言程序设计  |   88 |
| G1463337 | 高伟   | C语言程序设计  |   90 |
| G1463388 | 卫丹丹 | C语言程序设计  |   99 |
+----------+--------+----------------+------+
10 rows in set
```

【演示实例9-2】查询任意一位同学成绩，执行时由用户指定——带参数存储过程

1.右击“函数”——新建函数——过程——完成，进入 函数定义界面

```
BEGIN
        #Routine body goes here...

END;
```

注意：参数的类型必须指定，否则出错！！

2.在中间输入函数功能代码：在“定义”下面的“参数”中输入 IN name varchar(50)

```
BEGIN
  select S.studentNo 学号,studentName 姓名,subjectName 课程名,studentResult 成绩
  from student S join result R on S.studentNo = R.studentNo
  join subject SJ on SJ.subjectId = R.subjectId
  WHERE studentName = NAME
  order by R.subjectId;

END
```

3.保存过程：带参姓名

4.运行

```
mysql> call 带参姓名('王超');                                    输入的姓名加英文引号
+----------+------+-----------------+------+
| 学号     | 姓名 | 课程名          | 成绩 |
+----------+------+-----------------+------+
| G1363300 | 王超 | Android应用开发 |   83 |
| G1363300 | 王超 | Java面向对象设计|   49 |
| G1363300 | 王超 | Web客户端编程   |   64 |
+----------+------+-----------------+------+
3 rows in set

mysql> call 带参姓名('王子洋');
+----------+-------+--------------+------+
| 学号     | 姓名  | 课程名       | 成绩 |
+----------+-------+--------------+------+
| G1263201 | 王子洋| 软件测试技术 |   76 |
| G1263201 | 王子洋| Linux操作系统|   88 |
+----------+-------+--------------+------+
2 rows in set
```

使用命令行创建过程:

```
mysql> delimiter //

mysql> create procedure aa(in name varchar(50))
    -> begin
    -> select S.studentNo 学号,studentName 姓名,subjectName 课程名,studentResult 成绩
    -> from student S join result R on S.studentNo = R.studentNo
    -> join subject SJ on SJ.subjectId = R.subjectId
    -> WHERE studentName = NAME
    -> order by R.subjectId;
    -> end //
Query OK, 0 rows affected

mysql> delimiter ;
mysql> call aa('王子洋');
+----------+-------+--------------+------+
| 学号     | 姓名  | 课程名       | 成绩 |
+----------+-------+--------------+------+
| G1263201 | 王子洋| 软件测试技术 |   76 |
| G1263201 | 王子洋| Linux操作系统|   88 |
+----------+-------+--------------+------+
2 rows in set

Query OK, 0 rows affected
```

mysql> delimiter ; —— 修改命令结束符号，所以用 ;

```
mysql> call insjr('C语言程序设计');
+----------+--------+---------------+------+
| 学号     | 姓名   | 课程名        | 成绩 |
+----------+--------+---------------+------+
| G1463354 | 陈大伟 | C语言程序设计 |   52 |
| G1463358 | 温海南 | C语言程序设计 |   65 |
| G1463354 | 陈大伟 | C语言程序设计 |   67 |
| G1463388 | 卫丹丹 | C语言程序设计 |   78 |
| G1463337 | 高伟   | C语言程序设计 |   82 |
| G1463342 | 胡俊文 | C语言程序设计 |   86 |
| G1463383 | 钱嫣然 | C语言程序设计 |   87 |
| G1463383 | 钱嫣然 | C语言程序设计 |   88 |
| G1463337 | 高伟   | C语言程序设计 |   90 |
| G1463388 | 卫丹丹 | C语言程序设计 |   99 |
+----------+--------+---------------+------+
10 rows in set
```

【技能训练9-2】通过带参存储过程查询某门课程所有学生成绩

```
mysql> delimiter //
mysql> create procedure inSJR(in name varchar(50))
    -> begin
    -> select S.studentNo 学号,studentName 姓名,subjectName 课程名,studentResult 成绩
    -> from student S join result R on S.studentNo = R.studentNo
    -> join subject SJ on SJ.subjectId = R.subjectId
    -> WHERE sujectName = NAME
    -> order by studentResult;
    -> end //
Query OK, 0 rows affected

mysql> call insjr('大学英语')// ——sujectName 字段名错误，打开“函数”中的过程名进行修改

1054 - Unknown column 'sujectName' in 'where clause'

mysql> call insjr('大学英语')//   ——注意：没有修改命令结束符号，所以用 //
+----------+--------+----------+------+
| 学号     | 姓名   | 课程名   | 成绩 |
+----------+--------+----------+------+
| G1463342 | 胡俊文 | 大学英语 |   68 |
| G1463354 | 陈大伟 | 大学英语 |   68 |
| G1463337 | 高伟   | 大学英语 |   92 |
| G1463358 | 温海南 | 大学英语 |   92 |
| G1463388 | 卫丹丹 | 大学英语 |   92 |
| G1463383 | 钱嫣然 | 大学英语 |   92 |
+----------+--------+----------+------+
6 rows in set

Query OK, 0 rows affected
```

【演示示例9-3】带输入/输出参数的存储过程：

查询某同学（输入）考试成绩，并将其总分输出给一个变量（输出）

1.右击“函数”——新建函数——过程——完成，进入 函数定义界面

```
BEGIN
        #Routine body goes here...

END;
```

2.在中间输入函数功能代码：

在“定义”下面的“参数”中输入：IN `name` varchar(50),OUT `sum` float(6,2)

```
begin
select S.studentNo 学号,studentName 姓名,subjectName 课程名,studentResult 成绩
  from student S
        join result R on S.studentNo = R.studentNo
  join subject SJ on SJ.subjectId = R.subjectId
  WHERE studentName = NAME
  order by R.subjectId;
select sum(studentResult) into sum
        from student S
        join result R on S.studentNo = R.studentNo
  join subject SJ on SJ.subjectId = R.subjectId
  WHERE studentName = NAME
  order by R.subjectId;
end
```

3.保存过程：!OALL

4.运行

（1）——图形界面运行出错：

```
Procedure execution failed
1318 - Incorrect number of arguments for PROCEDURE schooldb.IOall; expected 2, got 1
```

出错原因：

在参数输入中只有 '王子洋'，缺少输出参数。——确保输入、输出参数个数一致！

改正的参数： '王子洋',@sum2

（2）运行结果：“信息”选项卡显示结果，内容如下

```
时间: 00:00.00

Procedure executed successfully
受影响的行: 0

受影响的行: 1

Parameters: IN `name` varchar(50),OUT `sum` float(6,2)
'王子洋',@sum2
Return values: 王子洋, 164
```

“结果1”选项卡中显示 王子洋 成绩信息

命令窗口运行情况

```
mysql> call IOALL('王超',@SUM1);      ——第1次运行
+----------+------+------------------+------+
| 学号     | 姓名 | 课程名           | 成绩 |
+----------+------+------------------+------+
| G1363300 | 王超 | Android应用开发  |   83 |
| G1363300 | 王超 | Java面向对象设计 |   49 |
| G1363300 | 王超 | Web客户端编程    |   64 |
+----------+------+------------------+------+
3 rows in set

mysql> select @sum1;
+-------+
| @sum1 |
+-------+
|   196 |
+-------+
1 row in set

mysql> call IO('胡保蜜',@SUM2);          ——第2次运行
+----------+--------+------------------+------+
| 学号     | 姓名   | 课程名           | 成绩 |
+----------+--------+------------------+------+
| G1363278 | 胡保蜜 | Android应用开发  |   55 |
| G1363278 | 胡保蜜 | Java面向对象设计 |   78 |
| G1363278 | 胡保蜜 | Web客户端编程    |   76 |
+----------+--------+------------------+------+
3 rows in set

Query OK, 1 row affected

mysql> select @sum2;
+-------+
| @sum2 |
+-------+
|   209 |
+-------+
1 row in set
```

【技能训练9-3】输入/输出存储过程，查询某门课程成绩，输出该课程平均分

1.右击“函数”——新建函数——过程——添加IN OUT参数——完成，进入 函数定义界面

```
BEGIN
        #Routine body goes here...

END;
```

2.在中间输入函数功能代码： ——数据类型后面的数据长度必须添加，否则报错！

在“定义”下面的“参数”中输入：IN `sjname` varchar(50),OUT `avgrc` float(6,2)

```
select S.studentNo 学号,studentName 姓名,subjectName 课程名,studentResult 成绩
  from student S
        join result R on S.studentNo = R.studentNo
  join subject SJ on SJ.subjectId = R.subjectId
  WHERE subjectName = sjNAME
  order by studentResult;
select avg(studentResult) into avgrc
        from student S
        join result R on S.studentNo = R.studentNo
  join subject SJ on SJ.subjectId = R.subjectId
  WHERE subjectName = sjNAME
  order by studentResult;
```

3.保存过程：IOavg

4.运行 |

（1）图形界面运行

参数： '大学英语',@avg1

（2）运行结果：“信息”选项卡显示结果，内容如下

```
Parameters: IN `sjname` varchar(50),OUT `avgrc` float(6,2)
'大学英语',@avg1
Return values: 大学英语, 84
```

“结果1”选项卡中显示 大学英语 成绩信息

命令窗口运行情况

```
mysql> call ioavg('Android应用开发',@avg3);
+---------+--------+-----------------+------+
| 学号    | 姓名   | 课程名          | 成绩 |
+---------+--------+-----------------+------+
| G1363278 | 胡保蜜 | Android应用开发 |   55 |
| G1363303 | 朱晓燕 | Android应用开发 |   61 |
| G1363301 | 党志鹏 | Android应用开发 |   65 |
| G1363302 | 胡仲友 | Android应用开发 |   80 |
| G1363300 | 王超   | Android应用开发 |   83 |
+---------+--------+-----------------+------+
5 rows in set

mysql> select @avg3;
+------------------+
| @avg3            |
+------------------+
| 68.80000305175781 |
+------------------+
1 row in set
```

9.2 触发器

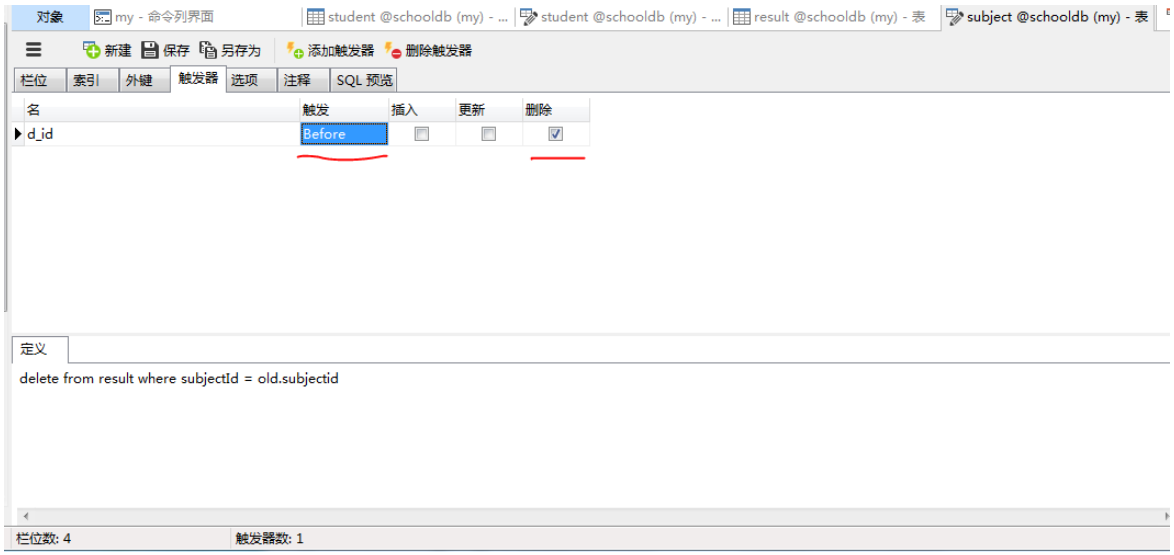
1. 触发器的概念与作用：定义（当满足触发条件时自动执行的 SQL 集合）、作用（数据校验、自动日志、关联更新）。

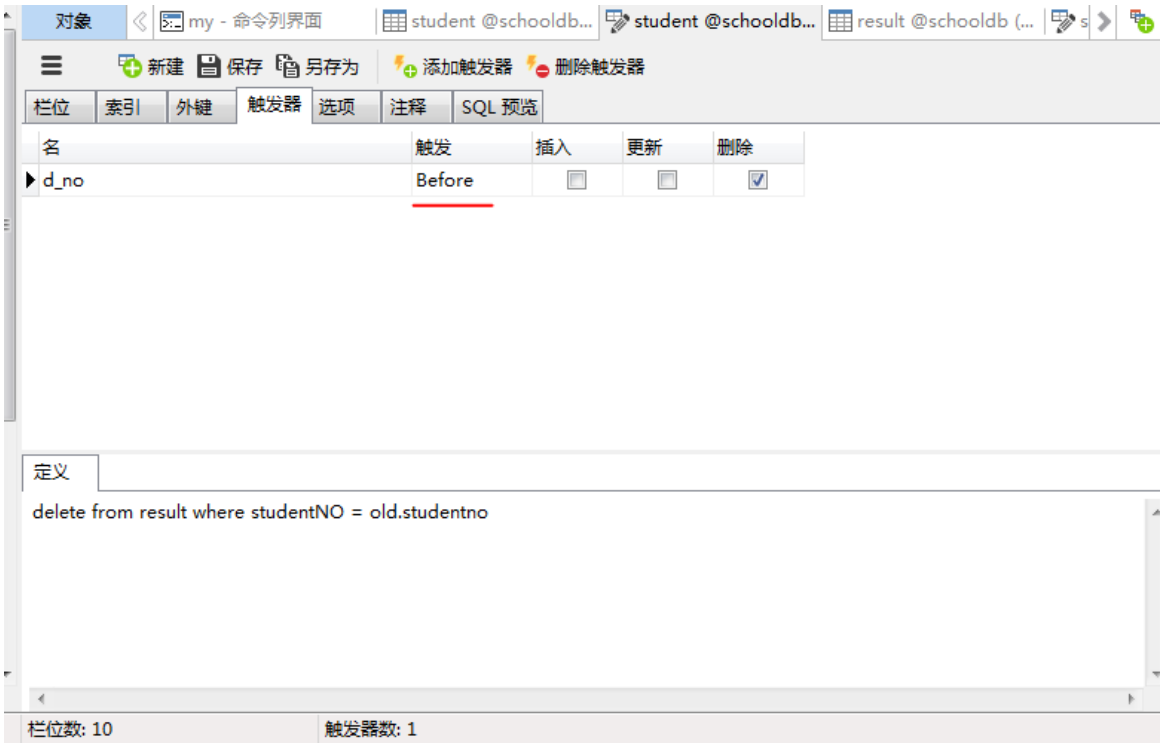
2. 触发器的核心要素：① 触发时机：BEFORE（执行前触发）、AFTER（执行后触发）；② 触发事件：INSERT、UPDATE、DELETE；③ 触发对象：表。

3. 触发器的语法：① 创建：CREATE TRIGGER；② 查看：SHOW TRIGGERS；③ 删除：DROP TRIGGER。

4. 实操演示 1：创建 BEFORE INSERT 触发器“校验员工年龄”，防止插入年龄<18 的数据，讲解 OLD/NEW 关键字（INSERT 时只有 NEW）。

5. 实操演示 2：创建 AFTER UPDATE 触发器“记录订单状态日志”，当订单表 status 字段修改时，自动插入日志到 order_log 表，讲解 OLD（修改前值）和 NEW（修改后值）的使用。6. 触发器的注意事项：避免递归触发、异常处理。

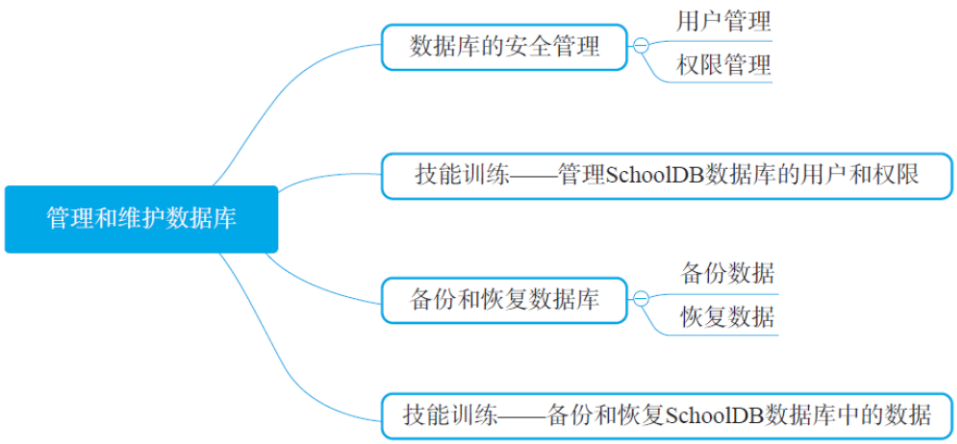

对象
my - 命令列界面
student @schooldb (my) - ...
student @schooldb (my) - ...
result @schooldb (my) - 表
subject @schooldb (my) - 表
新建
保存
另存为
添加触发器
删除触发器
栏位
索引
外键
触发器
选项
注释
SQL 预览
名
触发
插入
更新
删除
d_id
Before
定义
delete from result where subjectId = old.subjectid
栏位数: 4
触发器数: 1


对象
my - 命令列界面
student @schooldb...
student @schooldb...
result @schooldb (...
新建
保存
另存为
添加触发器
删除触发器
栏位
索引
外键
触发器
选项
注释
SQL 预览
名
触发
插入
更新
删除
d_no
Before
定义
delete from result where studentNO = old.studentno
栏位数: 10
触发器数: 1

课后反思	

课题	第 09 章　管理和维护数据库		
教学目标	➢ 能使用 SQL 语句创建和删除用户账户 ➢ 能使用 SQL 语句授予和回收权限 ➢ 能备份数据库数据 ➢ 能恢复数据库数据		
重点难点	➢ 权限的授予与回收 ➢ 在不同计算机之间迁移数据库和数据		
建议学时	3	教学教具	一体化教学实训室
主要教法	讲授演示法		
思政融入	➢ 强化数据安全责任意识：通过数据备份恢复的学习，结合数据丢失导致业务瘫痪的案例，让学生认识到“数据备份是最后防线”，树立“宁可备而不用，不可用而无备”的安全理念。 ➢ 培养严谨细致的工作作风：在性能监控、权限配置等操作中，强调流程规范和细节把控，让学生理解“细节失误可能引发系统故障”，养成严谨务实的职业习惯。 ➢ 树立服务与责任担当：结合数据库维护对业务系统稳定运行的支撑作用，让学生认识到技术维护工作的重要性，培养“保障系统稳定，服务业务发展”的责任担当。 ➢ 激发合规意识：通过权限管理和安全防护的学习，结合《数据安全法》《个人信息保护法》相关要求，培养“依法管理数据，合规操作数据库”的法治理念。		

教　学　过　程



任务一 数据库用户管理

用户管理

- 用户要访问数据库
  - 首先必须能连接数据库所在的MySQL服务器
  - 必须拥有登录MySQL服务器的用户名和密码
- MySQL的访问控制分为两个阶段
  - 第一个阶段：服务器验证是否允许连接
  - 第二个阶段：连接成功后，验证每个请求是否具有实施的权限
- 例如：要查看表中的数据
  - MySQL会检查是否具有对这个表的SELECT权限
  - 要执行某个存储过程，MySQL会检查是否具有该存储过程的执行权限

创建用户

- 由系统管理员创建登录账户
  - 给定用户名和登录密码
  - 登录的位置和默认连接的数据库
- 系统管理员是root用户
  - 拥有最高的权限
  - 可以完成所有的操作
  - 它的密码在安装MySQL服务器时设置

创建用户

语法

```
CREATE USER 用户名
    @主机名[IDENTIFIED BY [PASSWORD] '密码'];
```

分析

- 用户名：创建用户的名字
- 主机名
  - 指定创建用户所连接主机
  - 可以：某个IP地址主机名（如localhost）、某个IP段
  - 也可包含：通配符%（任意个字符）、_（任意单个字符）

为创建的用户创建不同的权限，每个用户允许做权利范围以内的事情，不允不可越界。

权限管理

- 用户执行SQL语句
  - MySQL将逐级进行权限检查
  - 看用户是否具有操作对象的SQL语句的执行权限
- MySQL的权限管理
  - 允许做权利范围以内的事情，不允许越界
  - 例如：只允许执行SELECT操作，那么就不能执行INSERT操作

权限管理

- MySQL中用户权限的五个层级
  - 全局层级
  - 数据库层级
  - 表层级
  - 列层级
  - 子程序层级

权限	权限级别	权限说明
CREATE	数据库、表或索引	创建数据库、表或索引权限
DROP	数据库或表	删除数据库或表权限
GRANT OPTION	数据库、表或保存的程序	赋予权限选项
REFERENCES	数据库或表	
ALTER	表	更改表，如添加字段、索引、约束
DELETE	表	删除数据权限
INDEX	表	索引权限
INSERT	表	插入权限
SELECT	表	查询权限
UPDATE	表	更新权限
CREATE VIEW	视图	创建视图权限
SHOW VIEW	视图	查看视图权限
CREATE ROUTINE	存储过程	创建存储过程权限

权限	权限级别	权限说明
ALTER ROUTINE	存储过程	修改存储过程权限
EXECUTE	存储过程	执行存储过程
FILE	服务器主机上的文件访问	文件访问权限
CREATE TEMPORARY	服务器管理	创建临时表权限
LOCK TABLES	服务器管理	锁表管理
CREATE USER	服务器管理	创建用户权限
PROCESS	服务器管理	查看进程权限
RELOAD	服务器管理	执行 FLUSH.REFRESH.RELOAD 等命令权限
REPLICATION CLIENT	服务器管理	复制权限
REPLICATION SLAVE	服务器管理	复制权限
SHOW DATABASES	服务器管理	查看数据库权限
SHUTDOWN	服务器管理	关闭数据库权限
SUPER	服务器管理	执行 KILL 线程权限

权限授予

语法

```
GRANT 权限 1[(列名列表 1)][，权限 2(列名列表 2)]]…
  ON[目标]{表名|*|*.*|库名.*}
  TO 用户 1[IDENTIFIED BY[PASSWORD] '密码 1']
      [,用户 2[IDENTIFIED BY[PASSWORD] '密码 2 ']]…
  [WITH 权限限制 1[权限限制 2]…]
```

分析

- 权限
  - 权限的名称，如SELECT、UPDATE等
  - 给不同的对象授予权限的值也不相同

权限收回

语法

```
REVOKE ALL PRIVILEGES，GRANT OPTION FROM 用户 1[,用户 2]...
```

分析

- 回收该用户的所有权限

任务二　数据库备份与恢复

如果系统管理员在对学生信息表进行管理，误删了重要的学生数据，为了挽回误操作造成的损失，数据库管理人员需要对数据库进行数据备份，在出现操作事故后可以将之前的数据还原。

备份数据库

- 目的：当数据库发生故障时可通过备份数据文件恢复数据
- 原因：
  - 存储介质故障：保存数据文件的磁盘设备损坏
  - 用户的错误操作：用户有意或无意删除了重要数据，甚至整个数据库
  - 服务器的瘫痪：数据库服务器因为软件漏洞彻底瘫痪
- 备份方式
  - 使用mysqldump备份
  - 使用SQL命令备份数据表
  - 使用mysql命令备份数据

使用mysqldump备份数据库或者指定表

语法

```
mysqldump -u user -h host -p password db [tb1,[tb2,....]]>filename
```

分析

- -u后的user表示用户名
- -h后的host表示主机名
- -p后的password表示用户密码
  - -p选项和密码之间不能有空格
  - 如果是本地MySQL服务器，则-h选项可以省略

使用mysqldump备份多个数据库

语法

```
mysqldump -u user -h host -p password
    --databases db1 [db2 [db3...]]>filename
```

分析

- --databases：要备份多个数据库
  - 后面至少要指定一个数据库名称
  - 多个数据库用空格隔开
- db1、db2、db3表示要备份的多个数据库的名称

使用SQL命令备份数据表

语法

```
SELECT columns FROM tablename [WHERE condition]
    INTO OUTFILE 'filename'[OPTION]
```

分析

- OPTION表示设置相应的选项
  - 决定数据行在文件中存放的格式
    - ✓ fields terminated by'string'：字段的分隔符为字符串对象，默认为“\t”
    - ✓ lines starting by'string'：每行开始的字符串符号，默认不使用任何字符
    - ✓ lines terminated by'string'：每行结束的字符串符号，默认为“\n”
    - ✓ fields escaped by 'char'：设置转义字符的字符符号，默认使用“\”

使用mysql命令恢复数据

语法

```
mysql -u root -p password [db]<filename
```

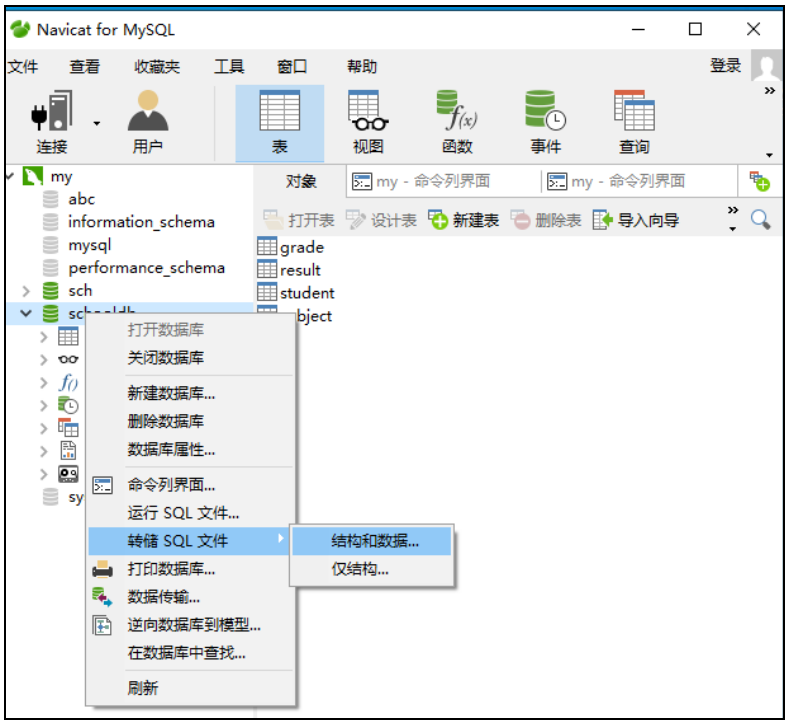
分析

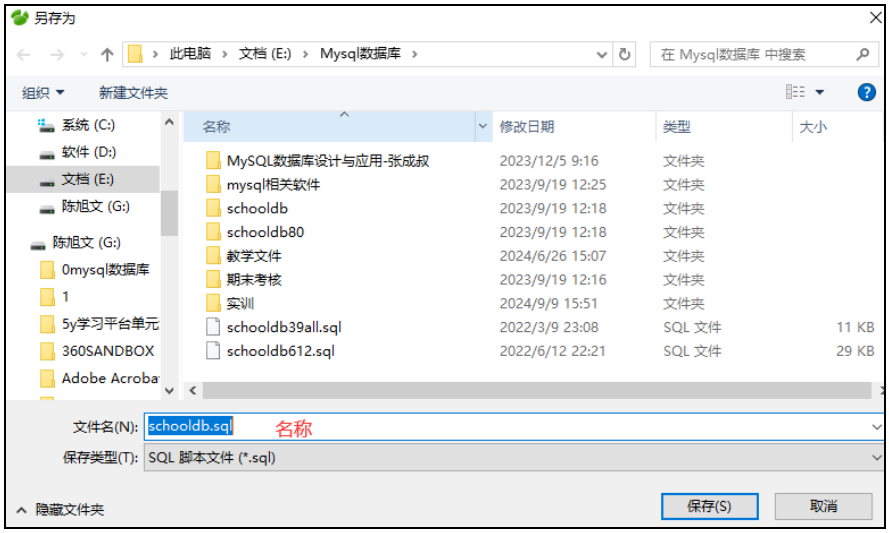
- Db：表示要还原的数据名称
  - 为可选项，可以指定数据库名，也可以不指定
  - 如果使用--all-databases备份所有数据库，则还原时无需指定数据库
  - 如果指定了数据库，则数据库要先创建

关键步骤

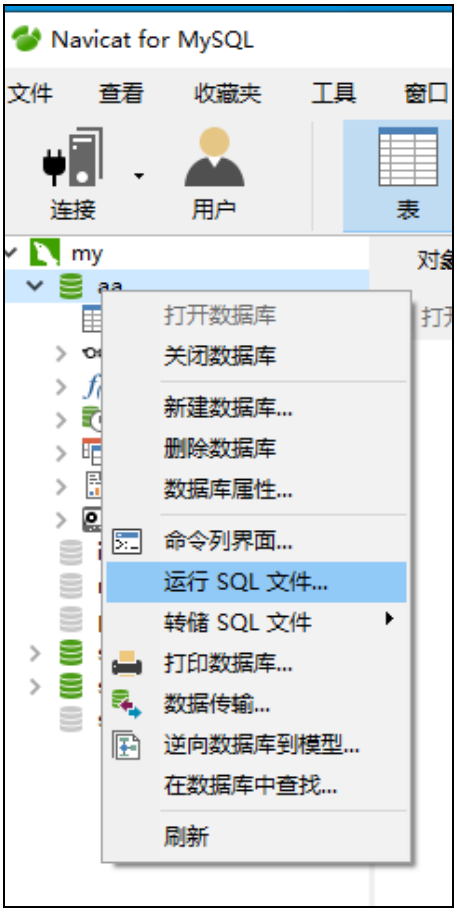
- 执行数据备份或恢复操作时
  - 当不使用默认目录，需要修改配置文件
  - 保存配置文件后，重新启动MySQL数据服务器
- 不同操作系统下的目录分隔符存在差异，注意区分
- 数据库备份在项目中极其重要
  - 要做到定时备份与即时故障恢复

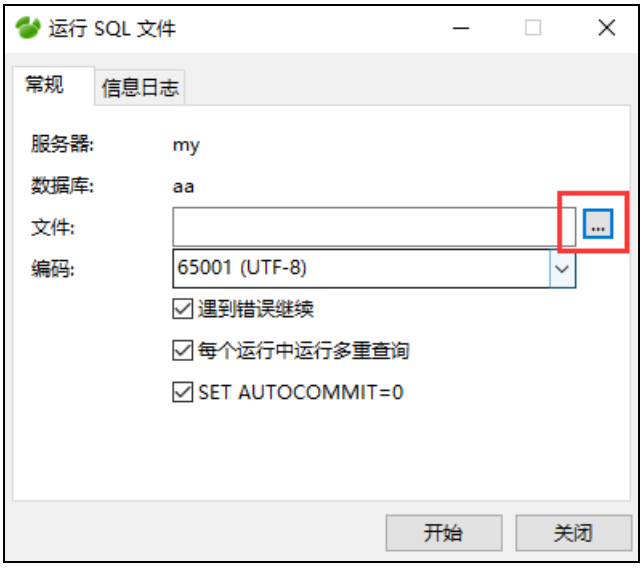
Navicat 中进行数据库备份

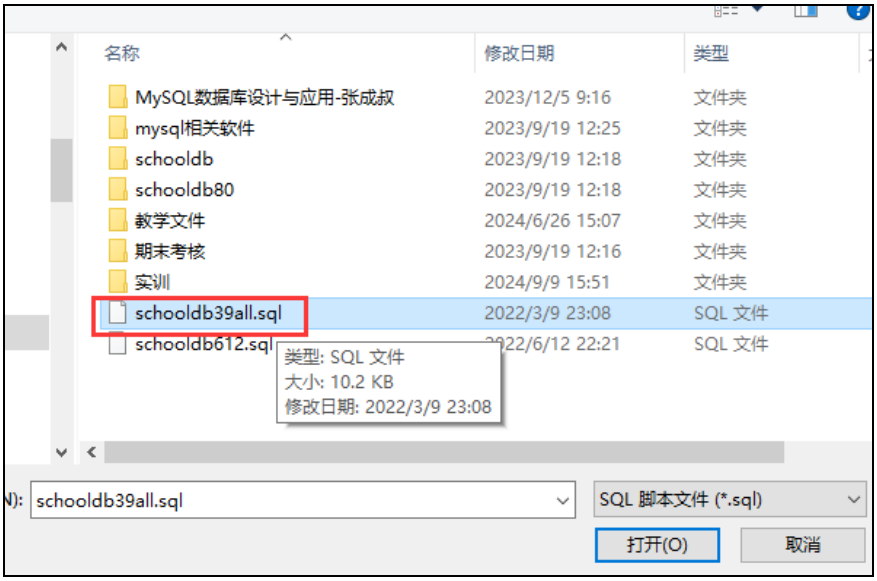


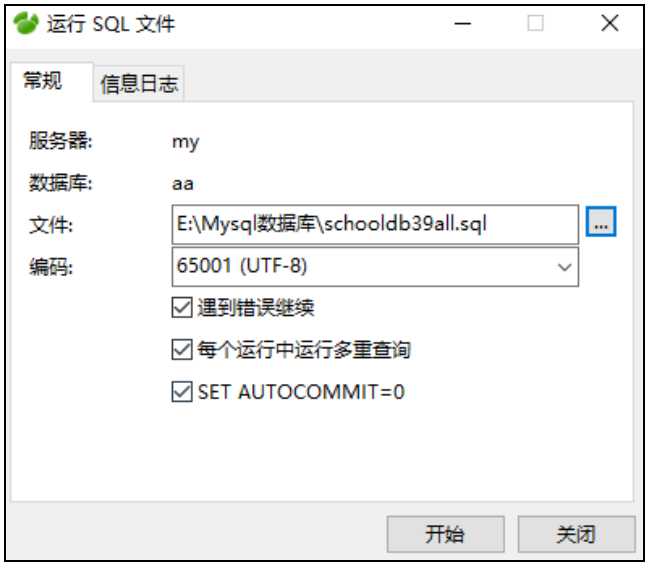


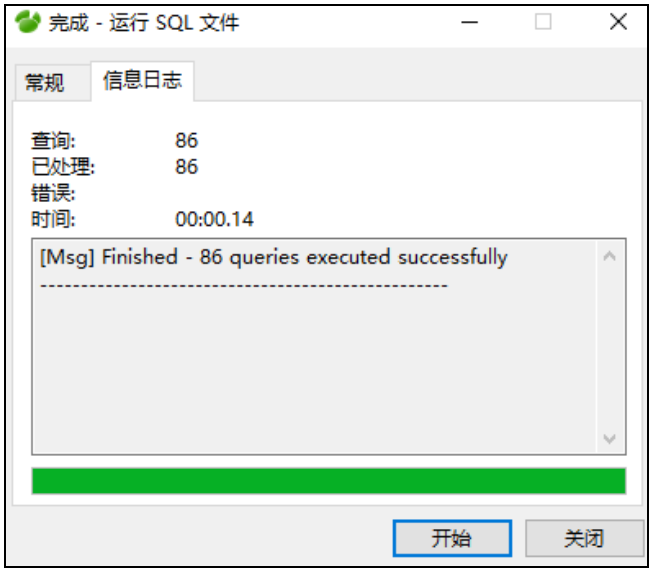
数据库的恢复


Navicat for MySQL
文件
查看
收藏夹
工具
窗口
连接
用户
表
my
aa
打开数据库
关闭数据库
新建数据库...
删除数据库
数据库属性...
命令列界面...
运行 SQL 文件...
转储 SQL 文件
打印数据库...
数据传输...
逆向数据库到模型...
在数据库中查找...
刷新


运行 SQL 文件
常规
信息日志
服务器: my
数据库: aa
文件:
编码: 65001 (UTF-8)
遇到错误继续
每个运行中运行多重查询
SET AUTOCOMMIT=0
开始
关闭


名称
修改日期
类型
MySQL数据库设计与应用-张成叔 2023/12/5 9:16 文件夹
mysql相关软件 2023/9/19 12:25 文件夹
schooldb 2023/9/19 12:18 文件夹
schooldb80 2023/9/19 12:18 文件夹
教学文件 2024/6/26 15:07 文件夹
期末考核 2023/9/19 12:16 文件夹
实训 2024/9/9 15:51 文件夹
schooldb39all.sql 2022/3/9 23:08 SQL 文件
schooldb612.sql 2022/6/12 22:21 SQL 文件
类型: SQL 文件
大小: 10.2 KB
修改日期: 2022/3/9 23:08
schooldb39all.sql
SQL 脚本文件 (*.sql)
打开(O)
取消


运行 SQL 文件
常规
信息日志
服务器:
my
数据库:
aa
文件:
E:\Mysql数据库\schooldb39all.sql
...
编码:
65001 (UTF-8)
遇到错误继续
每个运行中运行多重查询
SET AUTOCOMMIT=0
开始
关闭


完成 - 运行 SQL 文件
常规
信息日志
查询:
86
已处理:
86
错误:
时间:
00:00.14
[Msg] Finished - 86 queries executed successfully
开始
关闭

课后 反思	