

TURING

从0到1

SQL 即学即用

计算机通识精品课

莫振杰 著



绿叶学习网计算机系列教程
累计超**1000**万人次学习

读完就学会, 上手就能用

基于SQL标准编写, 涵盖MySQL、SQL Server、Oracle
三大主流DBMS语法差异

人民邮电出版社
北京

图书在版编目 (CIP) 数据

从0到1 : SQL即学即用 / 莫振杰著. — 北京 :
人民邮电出版社, 2023. 1
(图灵程序设计丛书)
ISBN 978-7-115-60886-4

I. ①从… II. ①莫… III. ①SQL语言 IV.
①TP311.132.3

中国版本图书馆CIP数据核字 (2022) 第250870号

内 容 提 要

本书主要分为4部分:第1部分主要介绍SQL的基础语法,包括查询操作、数据统计、高级查询、内置函数、数据修改、表的操作、列的属性等;第2部分主要介绍SQL的高级技术,包括多表查询、视图、索引、存储程序、游标、事务等。第3部分通过一个完整的案例,将前面所介绍的知识串连起来,帮助读者融汇贯通;第4部分提供了用于参考的各种常用操作。

为了让读者更好地掌握书中内容,作者基于实际工作以及面试经验,精心设置了大量高质量的练习题。此外,本书还配有课件PPT以及各种资源,以便各大高校的老师教学使用。

-
- ◆ 著 莫振杰
责任编辑 赵 轩
责任印制 彭志环
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <https://www.ptpress.com.cn>
北京 印刷
 - ◆ 开本: 787×1092 1/16
印张: 26 2023年1月第1版
字数: 609千字 2023年1月北京第1次印刷
-

定价: .00元

读者服务热线: (010)84084456-6009 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京东市监广登字 20170147 号

前言

一本好书就如一盏指路明灯，不仅可以让小伙伴们学得更轻松，还可以让小伙伴们少走很多弯路。如果你需要的并不是大而全的图书，而是恰到好处的图书，那么不妨看看“从 0 到 1”这个系列的图书。

“第一眼看到的美，只是全部美的八分之一。”实际上，这个系列的图书是我多年从事开发的经验总结，除了介绍技术，还注入了自己非常多的思考。虽然我是一名技术工程师，但我对文字非常敏感。对技术写作来说，我喜欢用最简单的语言把最丰富的知识呈现出来。

在接触任何一门技术时，我都会记录初学时遇到的各种问题，以及自己的各种思考。所以我还算比较了解初学者的心态，也知道怎样才能让大家快速而无阻碍地学习。对于这个系列的图书，我更多是站在初学者的角度，而不是已学会的人的角度来编写的。

“从 0 到 1”系列还包含前端开发、Python 开发等方面的图书，感兴趣的小伙伴们可以到我的个人网站（绿叶学习网）具体了解。

最后想要跟大家说的是，或许这个系列并非十全十美，但我相信，独树一帜的讲解方式能够让小伙伴们走得更快、更远。

本书对象

- ▶ 零基础的读者。
- ▶ 想要系统学习数据库的工程师。
- ▶ 大中专院校相关专业的老师和学生。

配套资源

绿叶学习网是我开发的一个开源技术网站，也是“从 0 到 1”系列图书的配套网站。本书的所有配套资源（包括源码、习题答案、PPT 等）都可以在该网站上找到。

此外，小伙伴们如果有任何技术问题，或者想要获取更多学习资源，抑或希望和更多技术人员进行交流，可以加入官方 QQ 群：280972684、387641216。

特别感谢

在写作本书的过程中，我得到了很多人的帮助。首先要感谢赵轩老师，他是一位非常专业而不拘一格的编辑，有他的帮忙本书才能顺利出版。

其次，感谢五叶草团队的一路陪伴，感谢韦雪芳、陈志东、莫振浩这几位小伙伴花费大量时间对本书进行细致的审阅，并且给出了诸多非常棒的建议。

最后，特别感谢我的妹妹莫秋兰，她一直在默默地支持和关心着我。有这样一个懂自己的人，是非常幸运的事情，她既是我的亲人也是我的朋友。

特别说明

本书中数据均为虚拟数据，仅供学习操作使用，并无实际用途。

由于个人水平有限，书中难免会有错漏之处，小伙伴们如果发现问题或有任何意见，可以到绿叶学习网或发邮件（lvystudy@qq.com）与我联系。

莫振杰
2023 年 1 月

目录

第1部分 基础语法

第1章 数据库.....2	
1.1 数据库是什么.....2	
1.1.1 数据库简介.....2	
1.1.2 DBMS 简介.....3	
1.1.3 MySQL 简介.....3	
1.2 安装 MySQL.....4	
1.3 安装 Navicat for MySQL.....9	
1.4 使用 Navicat for MySQL.....11	
1.4.1 连接 MySQL.....11	
1.4.2 创建数据库.....13	
1.4.3 创建表.....14	
1.4.4 运行代码.....17	
1.5 本书说明.....18	
1.6 本章练习.....19	
第2章 语法基础.....20	
2.1 SQL 是什么.....20	
2.1.1 SQL 简介.....20	
2.1.2 关键字.....21	
2.1.3 语法规则.....22	
2.1.4 命名规则.....23	
2.2 数据类型.....23	
2.2.1 数字.....24	
2.2.2 字符串.....25	
2.2.3 日期时间.....28	
2.2.4 二进制.....29	
2.3 注释.....30	
2.4 本章练习.....31	
第3章 查询操作.....32	
3.1 select 语句简介.....32	
3.1.1 select 语句.....33	
3.1.2 特殊列名.....38	
3.1.3 换行说明.....40	
3.2 使用别名: as.....41	
3.2.1 as 关键字.....41	
3.2.2 特殊别名.....44	
3.3 条件子句: where.....46	
3.3.1 比较运算符.....47	
3.3.2 逻辑运算符.....50	
3.3.3 其他运算符.....53	
3.3.4 运算符优先级.....58	
3.4 排序子句: order by.....60	
3.4.1 order by 子句.....60	
3.4.2 中文字符串字段排序.....64	
3.5 限制行数: limit.....68	
3.5.1 limit 子句.....68	
3.5.2 深入了解.....71	
3.6 去重处理: distinct.....77	
3.7 本章练习.....80	

第4章 数据统计83

4.1 算术运算..... 83

4.2 聚合函数..... 85

4.2.1 求和: sum()..... 85

4.2.2 求平均值: avg() 86

4.2.3 求最值: max() 和 min() 87

4.2.4 获取行数: count() 88

4.2.5 深入了解..... 90

4.2.6 特别注意..... 91

4.3 分组子句: group by..... 93

4.4 指定条件: having 97

4.5 子句顺序..... 99

4.6 本章练习..... 100

第5章 高级查询 102

5.1 模糊查询: like 102

5.1.1 通配符: % 103

5.1.2 通配符: _ 105

5.1.3 转义通配符 106

5.2 随机查询: rand()..... 108

5.3 子查询..... 111

5.3.1 单值子查询 111

5.3.2 多值子查询..... 114

5.3.3 关联子查询 118

5.4 本章练习..... 121

第6章 内置函数 123

6.1 内置函数简介 123

6.2 数学函数..... 123

6.2.1 求绝对值: abs() 124

6.2.2 求余: mod()..... 125

6.2.3 四舍五入: round()..... 127

6.2.4 截取小数: truncate() 127

6.2.5 获取符号: sign()..... 128

6.2.6 获取圆周率: pi() 129

6.2.7 获取随机数: rand()..... 129

6.2.8 向上取整: ceil()..... 130

6.2.9 向下取整: floor() 131

6.3 字符串函数 132

6.3.1 获取长度: length() 133

6.3.2 去除空格: trim() 134

6.3.3 反转字符串: reverse() 135

6.3.4 重复字符串: repeat() 135

6.3.5 替换字符串: replace() 136

6.3.6 截取字符串: substring() 137

6.3.7 截取开头结尾: left()、 right() 137

6.3.8 拼接字符串: concat() 138

6.3.9 大小写转换: lower()、 upper() 141

6.3.10 填充字符串: lpad()、 rpad() 142

6.4 时间函数..... 143

6.4.1 获取当前日期: curdate() 143

6.4.2 获取当前时间: curtime() 144

6.4.3 获取当前日期时间: now()..... 144

6.4.4 获取年份: year() 145

6.4.5 获取月份: month()、 monthname()..... 146

6.4.6 获取星期: dayofweek()、 dayname()..... 147

6.4.7 获取天数: dayofmonth()、 dayofyear()..... 148

6.4.8 获取季度: quarter().....	150	8.2.3 修改库.....	182
6.5 排名函数.....	150	8.2.4 删除库.....	182
6.5.1 rank().....	151	8.3 创建表.....	184
6.5.2 row_number().....	152	8.4 查看表.....	187
6.5.3 dense_rank().....	155	8.4.1 show tables 语句.....	187
6.6 加密函数.....	156	8.4.2 show create table 语句.....	188
6.6.1 md5().....	157	8.4.3 describe 语句.....	189
6.6.2 sha1().....	157	8.5 修改表.....	190
6.7 系统函数.....	158	8.5.1 修改表名.....	190
6.8 其他函数.....	159	8.5.2 修改字段.....	191
6.8.1 cast().....	159	8.6 复制表.....	196
6.8.2 if().....	160	8.6.1 只复制结构.....	196
6.8.3 ifnull().....	161	8.6.2 同时复制结构和数据.....	197
6.9 本章练习.....	162	8.7 删除表.....	199
第 7 章 数据修改.....	163	8.8 本章练习.....	200
7.1 数据修改简介.....	163	第 9 章 列的属性.....	202
7.2 插入数据: insert.....	163	9.1 列的属性简介.....	202
7.2.1 insert 语句.....	163	9.2 默认值.....	203
7.2.2 特殊情况.....	166	9.3 非空.....	206
7.2.3 replace 语句.....	168	9.4 自动递增.....	208
7.3 更新数据: update.....	170	9.5 条件检查.....	213
7.4 删除数据: delete.....	173	9.6 唯一键.....	214
7.4.1 delete 语句.....	173	9.7 主键.....	218
7.4.2 深入了解.....	176	9.8 外键.....	222
7.5 本章练习.....	177	9.9 注释.....	226
第 8 章 表的操作.....	179	9.10 操作已有表.....	229
8.1 表的操作简介.....	179	9.10.1 约束型属性.....	229
8.2 库操作.....	179	9.10.2 其他属性.....	233
8.2.1 创建库.....	180	9.11 本章练习.....	236
8.2.2 查看库.....	181		

第 2 部分 高级技术

第 10 章 多表查询.....	240	第 12 章 索引.....	290
10.1 多表查询简介	240	12.1 索引简介	290
10.2 集合运算	241	12.2 创建索引	291
10.3 内连接	245	12.3 查看索引	292
10.3.1 基本语法	246	12.4 删除索引	294
10.3.2 深入了解	251	12.5 本章练习	295
10.4 外连接	254	第 13 章 存储程序.....	296
10.4.1 外连接是什么	254	13.1 存储程序简介	296
10.4.2 左外连接	255	13.2 存储过程	297
10.4.3 右外连接	257	13.2.1 创建存储过程	297
10.4.4 完全外连接	258	13.2.2 查看存储过程	307
10.4.5 深入了解	259	13.2.3 修改存储过程	308
10.5 笛卡儿积连接	260	13.2.4 删除存储过程	309
10.6 自连接	261	13.3 存储函数	310
10.7 本章练习	267	13.3.1 创建存储函数	310
第 11 章 视图	268	13.3.2 查看存储函数	313
11.1 创建视图	268	13.3.3 修改存储函数	314
11.1.1 视图简介	268	13.3.4 删除存储函数	314
11.1.2 修改数据	271	13.3.5 变量的定义	315
11.2 查看视图	281	13.3.6 常用的语句	317
11.3 修改视图	282	13.4 触发器	323
11.3.1 alter view	282	13.4.1 创建触发器	324
11.3.2 create or replace view	284	13.4.2 查看触发器	327
11.4 删除视图	285	13.4.3 删除触发器	328
11.5 多表视图	287	13.5 事件	328
11.6 本章练习	288	13.5.1 创建事件	329


13.5.2 查看事件.....	331	16.2.3 删除用户.....	354
13.5.3 修改事件.....	332	16.3 权限管理.....	354
13.5.4 删除事件.....	335	16.3.1 授予权限.....	356
13.6 本章练习.....	336	16.3.2 查看权限.....	359
第14章 游标.....	337	16.3.3 撤销权限.....	359
14.1 创建游标.....	337	16.4 本章练习.....	360
14.2 本章练习.....	342	第17章 数据备份.....	361
第15章 事务.....	344	17.1 数据备份简介.....	361
15.1 事务是什么.....	344	17.2 库的备份与还原.....	361
15.1.1 事务简介.....	344	17.2.1 库的备份.....	361
15.1.2 使用事务.....	344	17.2.2 库的还原.....	364
15.1.3 自动提交.....	346	17.3 表的备份与还原.....	365
15.1.4 使用范围.....	346	17.3.1 表的备份.....	365
15.2 事务的属性.....	346	17.3.2 表的还原.....	368
15.3 本章练习.....	347	17.4 本章练习.....	371
第16章 安全管理.....	348	第18章 其他内容.....	372
16.1 安全管理简介.....	348	18.1 系统数据库.....	372
16.2 用户管理.....	348	18.2 分页查询.....	373
16.2.1 创建用户.....	350	18.3 表的设计.....	375
16.2.2 修改用户.....	353	18.4 本章练习.....	376

第3部分 实战案例

第19章 经典案例.....	378	19.2 基础问题.....	380
19.1 案例准备.....	378	19.3 高级问题.....	385

第 4 部分 附录

附录 A 查询子句.....	396	附录 F “表” 操作	402
附录 B 列的属性.....	397	附录 G “数据” 操作.....	403
附录 C 连接方式	398	附录 H “视图” 操作.....	404
附录 D 内置函数.....	399	附录 I “索引” 操作.....	405
附录 E “库” 操作	401		
后记.....	406		



第 1 部分 基础语法

第 1 章

数据库

1.1 数据库是什么

1.1.1 数据库简介

数据库，也就是“DataBase”，简称“DB”。数据库，简单来说就是将大量数据保存起来的一个数据集合。数据库的应用极其广泛，日常生活中经常使用，只是有些小伙伴不是很了解而已。

比如一所学校有几万名学生，学生入学时学校会登记每名学生的相关信息，包括姓名、年龄等。这些信息都会保存到一个数据库中，平常考试、进出校门等都需要核对学生的信息。再拿各银行来说，所有客户的信息包括账号、密码、余额等，都是存放在数据库中的。

如果没有数据库，我们的生活会怎么样呢？此时不仅仅是银行取钱无法正常进行，像平常的网上购物，以及其他方方面面，都可能无法正常进行（图 1-1）。



图 1-1

1.1.2 DBMS 简介

DBMS，也就是“DataBase Management System”（数据库管理系统）。简单来说，DBMS 就是一种用来管理数据库的软件。DBMS 是根据数据库的类型进行分类的。

对于数据库来说，它的种类非常多。不过随着互联网以及大数据的发展，现在常用的数据库可以分为两种：① 关系数据库；② 非关系数据库。

因此对于 DBMS 来说，它也可以分为“关系 DBMS”以及“非关系 DBMS”这两种，如表 1-1 所示。这些 DBMS 在实际开发中会大量运用，所以这里我们有必要了解一下。

表 1-1 常见的 DBMS

关系 DBMS	非关系 DBMS
MySQL	MongoDB
SQL Server	Redis
Oracle	
PostgreSQL	

现在大多数公司主要使用关系 DBMS。可能会有小伙伴问：“拿 MySQL 来说，它和 SQL 到底有什么关系呢？”大家可以这样去理解：**SQL 是“一门语言”，而 MySQL 是基于这门语言的“一个软件”**。实际上，MySQL、SQL Server、Oracle、PostgreSQL 这 4 个都是基于 SQL 的“软件”。

由于 MySQL、SQL Server、Oracle 和 PostgreSQL 来自不同的厂商，所以它们的部分语法会有一些的差别，不过大部分语法都是相同的。小伙伴们只需记住：**常用的关系 DBMS 有 MySQL、SQL Server、Oracle 和 PostgreSQL。不管哪一个，都是使用 SQL 来操作的。**

1.1.3 MySQL 简介

由于 SQL 必须依托于某一个软件，所以如果想要学习 SQL，我们必须要从 MySQL、SQL Server、Oracle 和 PostgreSQL 这几个软件中选一个出来学习和使用。对于本书来说，我们选择的是 MySQL。

MySQL（其 Logo 如图 1-2 所示）是一款开源的数据库软件，也是目前使用最多的 DBMS 之一。很多编程语言的相关项目都会使用 MySQL 作为主要的数据库，包括 PHP、Python、Go 等。特别是在 Web 应用方面，MySQL 可以说是最好用的关系 DBMS 之一。

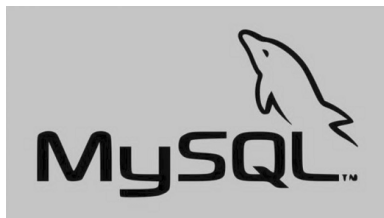


图 1-2

市面上主要使用的数据库是 MySQL、SQL Server、Oracle 这 3 种，而 PostgreSQL 较少用到。所以本书只重点关注 MySQL、SQL Server、Oracle 这 3 种数据库使用的 SQL 语法。

大多数情况下，这些数据库都有着相同的语法，然而偶尔也有所不同。所以本书会采取如下讲解方式：正文部分将会采用 MySQL 的语法，如果 SQL Server 或 Oracle 存在不同语法，就会以“数据库差异性”这样的板块给出。

1.2 安装 MySQL

在 Windows 系统中下载和安装 MySQL，只需要简单的 3 步就可以完成。但是在安装的过程中，小伙伴们不要一味追求快，一定要严格落实每一步。因为在安装 MySQL 时很容易出问题，重装也非常麻烦。

① **下载 MySQL**：首先打开 MySQL 官网下载页面，选择【MySQL Installer for Windows】，如图 1-3 所示。

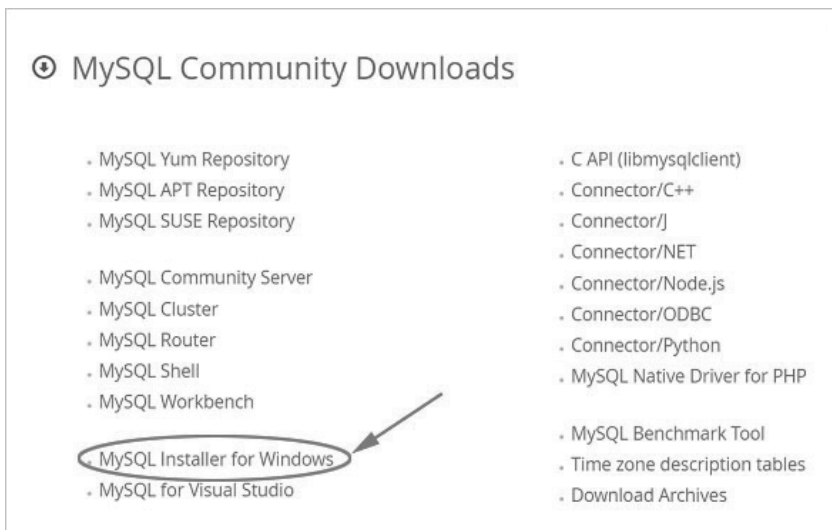


图 1-3

在如图 1-4 所示的下载页面中，这里我们选择下面文件体积较大的，单击【Download】按钮即可。

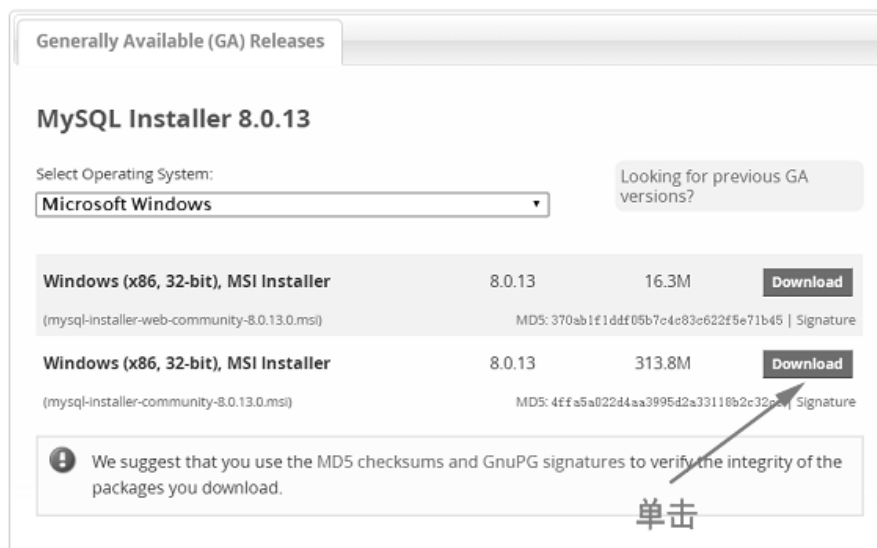


图 1-4

接下来会弹出一个登录页面，如图 1-5 所示。我们单击最下方的【No thanks, just start my download.】，就可以下载 MySQL。

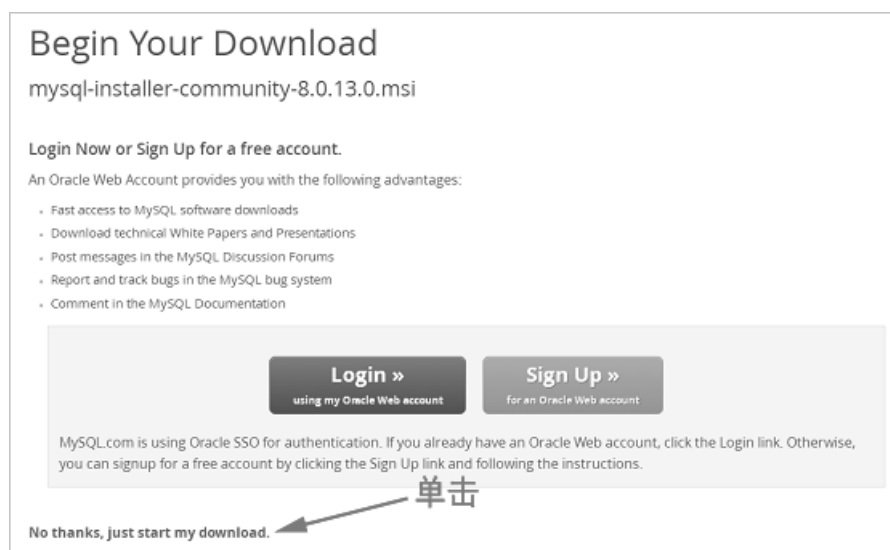


图 1-5

② 安装 MySQL：下载完成后，双击软件以进行安装，安装界面如图 1-6 所示，勾选【I accept the license terms】，然后单击【Next】按钮。

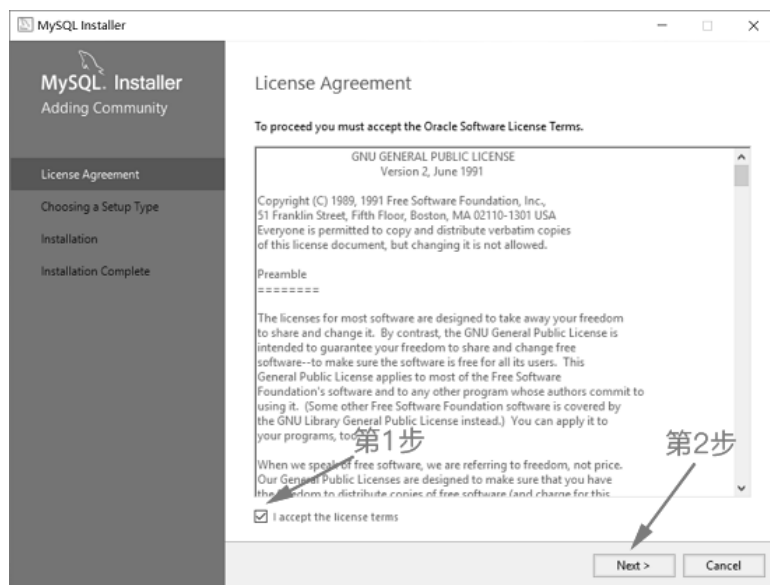


图 1-6

在如图 1-7 所示的界面中，我们选择【Server only】，然后单击【Next】按钮。

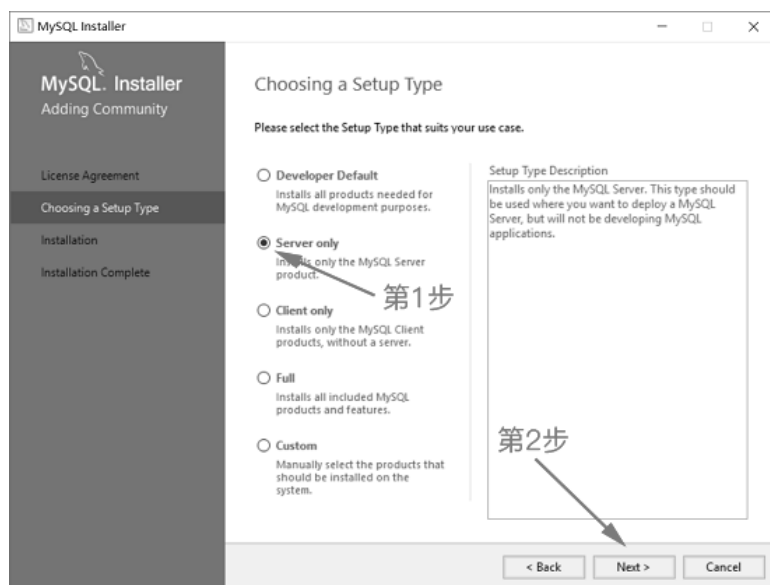


图 1-7

接下来，一路单击【Next】按钮即可，直到出现如图 1-8 所示的界面。在该界面中填写 root 用户的密码，长度最少为 4 位。界面下方还可以添加普通用户，一般情况下不需要再添加其他用户，直接使用 root 用户就可以了。填写密码之后，再单击【Next】按钮。

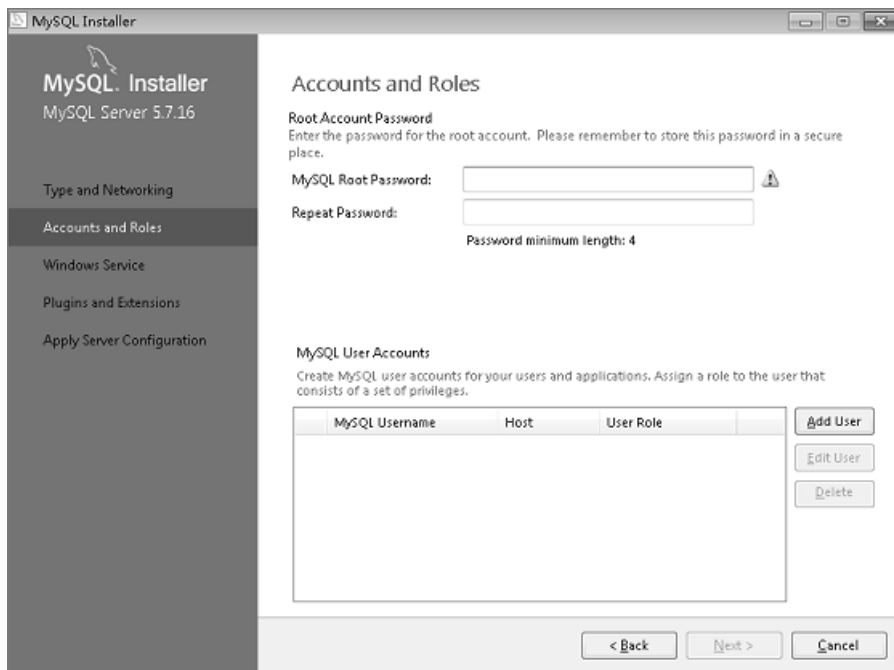


图 1-8

然后，一路单击【Next】按钮，最后单击【Finish】按钮即可完成安装。

③ 设置环境变量：安装完成后，还需要设置环境变量。设置环境变量是为了可以在任意的目录下使用 MySQL 命令。

鼠标右键单击【此电脑】，选择【属性】打开，【系统】窗口，选择【高级系统设置】，此时会打开如图 1-9 所示的对话框，再单击【环境变量】按钮。



图 1-9

在如图 1-10 所示的对话框中，我们选中【Path】变量（记得要先选中），并单击【编辑】按钮。

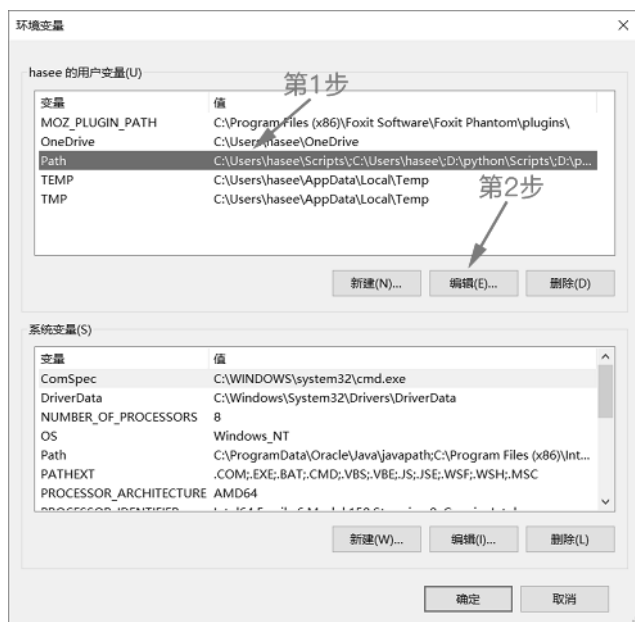


图 1-10

MySQL 安装完成之后，默认的安装路径是“C:\Program Files\MySQL\MySQL Server 8.0\bin”（安装路径有可能不一样，小伙伴们自行确认一下）。我们在【编辑环境变量】对话框（如图 1-11 所示）中，单击【新建】按钮，把 MySQL 安装路径写在变量值中。

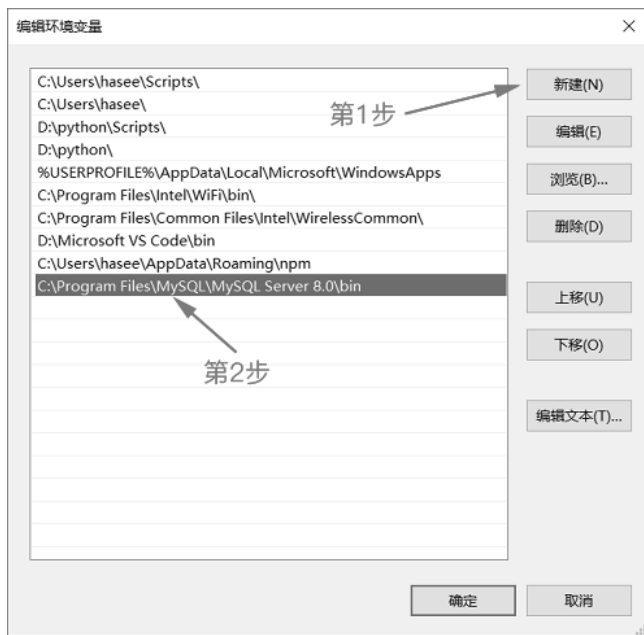


图 1-11

接下来我们一路单击【确定】按钮，然后把刚刚打开的窗口和对话框都关闭就可以了。最后需要说明一点，如果想要重装 MySQL，我们必须先把它卸载干净再去重装，不然就无法重装成功。至于如何卸载干净，小伙伴们可以自行搜索一下，非常简单。

数据库差异性

对于其他 DBMS（如 Oracle 和 SQL Server），以及它们对应的开发工具的安装教程和使用教程，小伙伴们可以在本书的配套文件中找到。

1.3 安装 Navicat for MySQL

MySQL 本身并未提供可视化工具。对于初学者来说，如果想要更轻松地了解 MySQL，强烈推荐 Navicat for MySQL 软件（其 Logo 如图 1-12 所示）来辅助学习。

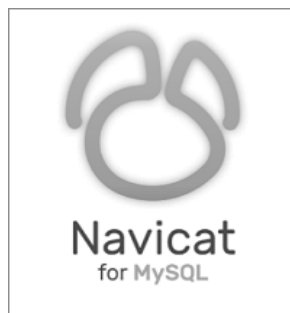


图 1-12

Navicat for MySQL 主要为 MySQL 提供图形化的界面操作，使得用户在使用 MySQL 时更加方便。如果不借助 Navicat for MySQL，就需要使用命令行的方式（如图 1-13 所示）。大家可能都知道，命令行这种方式有时是非常麻烦的。

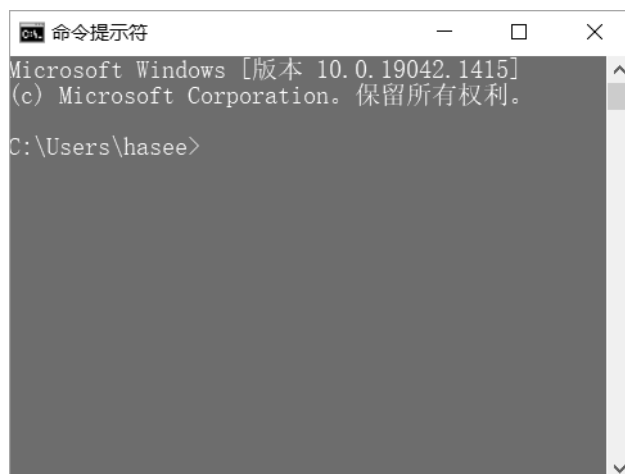


图 1-13

常见问题

除了 Navicat for MySQL，还有其他的可视化工具吗？

如果想要使用 MySQL 进行开发，除了 Navicat for MySQL，我们还可以使用 Workbench、phpMyAdmin 等。不过公认最简单、最好用的还是 Navicat for MySQL，其他的了解一下即可。

1.4 使用 Navicat for MySQL

1.4.1 连接 MySQL

① 连接 MySQL: 打开 Navicat for MySQL 后, 选择【连接】→【MySQL】, 界面如图 1-14 所示。

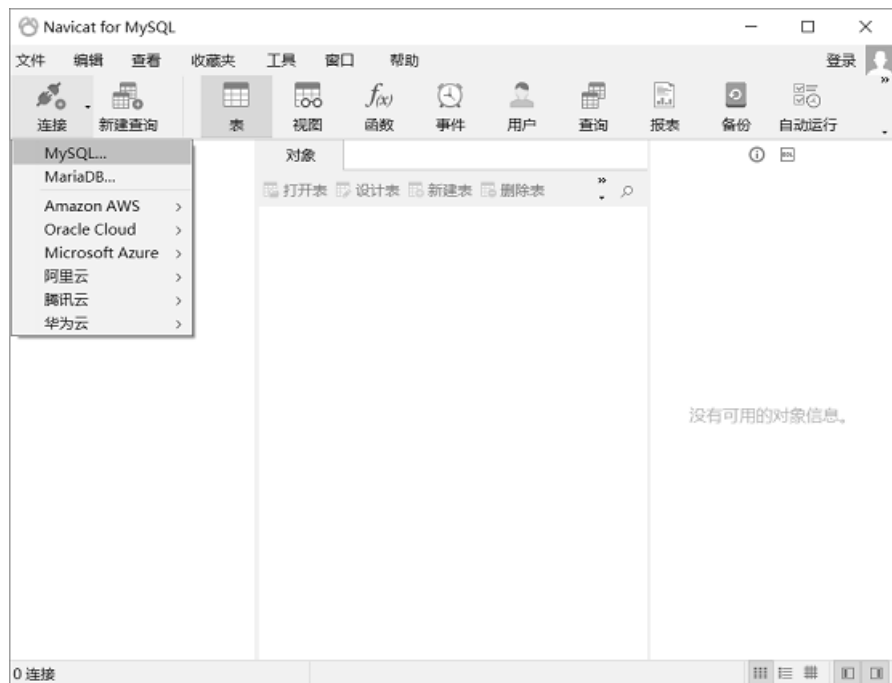


图 1-14

② 填写连接信息: 在弹出的对话框中填写连接信息, 连接名随便写即可, 密码就是 root 用户的密码, 如图 1-15 所示。填写完成后, 单击【确定】按钮。

为了方便学习, 小伙伴们可以将密码设置得简单一点儿, 比如设置成“123456”。不过在实际开发中, 考虑到安全性问题, 密码应尽可能设置得复杂一点儿。



图 1-15

③ 打开连接：在左侧选中【mysql】，单击鼠标右键并选择【打开连接】，如图 1-16 所示。之后就可以打开连接了。或者，直接双击【mysql】也可实现同样的效果。

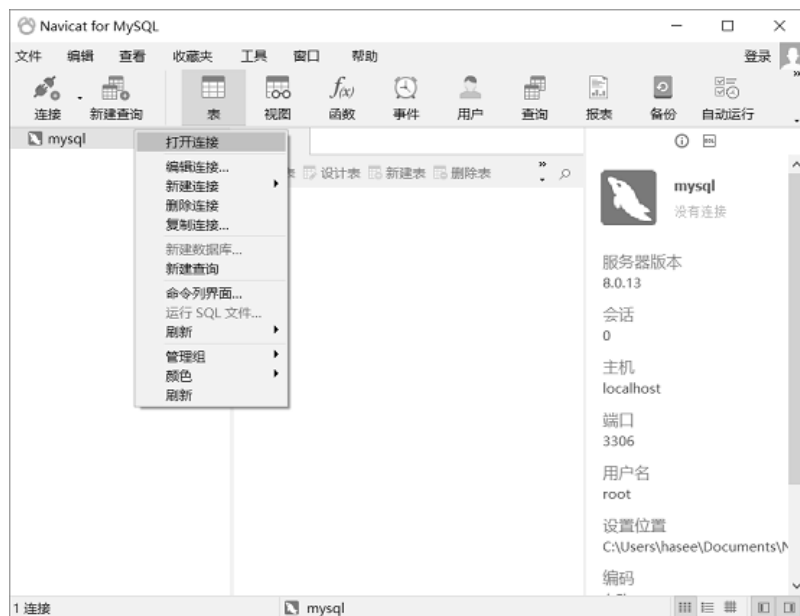


图 1-16

1.4.2 创建数据库

① 新建数据库：在左侧选中【mysql】，单击鼠标右键并选择【新建数据库】，如图 1-17 所示。



图 1-17

② 填写数据库名：在弹出的对话框中，填写数据库的基本信息，这里只需填写数据库名就可以了。数据库名是随便取的，这里填写的是“lvy”（即本书配套资源网站“绿叶学习网”的简称），然后单击【确定】按钮，如图 1-18 所示。

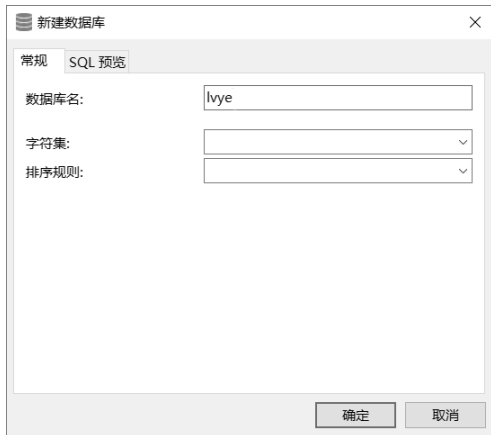


图 1-18

③ 打开数据库：在左侧选中【lvy】，然后单击鼠标右键并选择【打开数据库】，如图 1-19 所示。之后就可以打开该数据库了。或者，直接双击【lvy】也可实现同样的效果。



图 1-19

1.4.3 创建表

① 新建表：选择【lvy】下的【表】，单击鼠标右键并选择【新建表】，如图 1-20 所示。



图 1-20

② **填写表信息**：创建一个名为 product 的表，该表保存的是商品的基本信息，包括商品编号、商品名称、商品类型、来源城市、出售价格、入库时间等。其中，product 表的字段信息如图 1-21 所示。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int			<input type="checkbox"/>	<input type="checkbox"/>		商品编号
name	varchar	10		<input type="checkbox"/>	<input type="checkbox"/>		商品名称
type	varchar	10		<input type="checkbox"/>	<input type="checkbox"/>		商品类型
city	varchar	10		<input type="checkbox"/>	<input type="checkbox"/>		来源城市
price	decimal	5	1	<input type="checkbox"/>	<input type="checkbox"/>		出售价格
rdate	date			<input type="checkbox"/>	<input type="checkbox"/>		入库时间

图 1-21

我们还需要设置 id 为主键才行。首先选中 id 行，然后单击鼠标右键并选择【主键】，如图 1-22 所示。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id				<input type="checkbox"/>	<input type="checkbox"/>		商品编号
name		0		<input type="checkbox"/>	<input type="checkbox"/>		商品名称
type		0		<input type="checkbox"/>	<input type="checkbox"/>		商品类型
city		0		<input type="checkbox"/>	<input type="checkbox"/>		来源城市
price			1	<input type="checkbox"/>	<input type="checkbox"/>		出售价格
rdate				<input type="checkbox"/>	<input type="checkbox"/>		入库时间

第1步：选中列

第2步：选择【主键】

图 1-22

设置完主键之后，我们会看到 id 行中【不是 null】这一项前面打上了“√”，并且右边会显示钥匙图标，如图 1-23 所示。

名	类型	长度	小数点	不是 null	虚拟	键	注释
id	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>		商品编号
name	varchar	10		<input type="checkbox"/>	<input type="checkbox"/>		商品名称
type	varchar	10		<input type="checkbox"/>	<input type="checkbox"/>		商品类型
city	varchar	10		<input type="checkbox"/>	<input type="checkbox"/>		来源城市
price	decimal	5	1	<input type="checkbox"/>	<input type="checkbox"/>		出售价格
rdate	date			<input type="checkbox"/>	<input type="checkbox"/>		入库时间

图 1-23

③ **填写表名**：字段填写完成之后，我们使用“Ctrl+S”快捷键就可以保存了。这里会弹出【表名】对话框，这里填写的是“product”，如图 1-24 所示。



图 1-24

④ **打开表**：在左侧单击【表】左侧的“>”将其展开，然后选中【product】，单击鼠标右键并选择【打开表】，如图 1-25 所示。

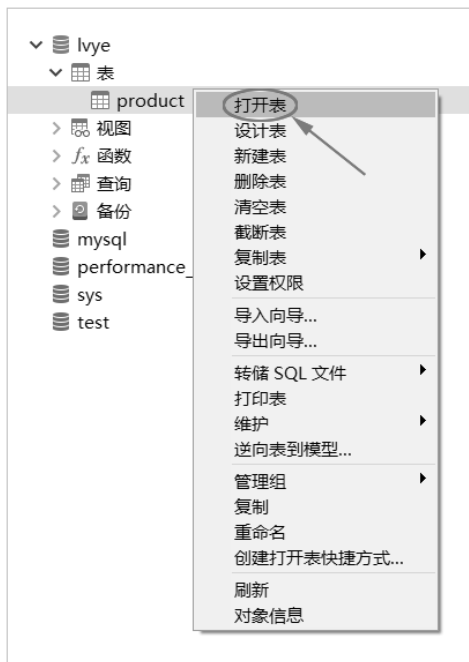


图 1-25

⑤ **添加数据**：打开表之后，我们可以通过单击左下角的“+”来添加一行数据，添加完成之后，再单击“√”即可，如图 1-26 所示。

由于 product 表在后面章节中会被反复用到，所以小伙伴们要严格对比图 1-26 并把所有数据都填写好才行。

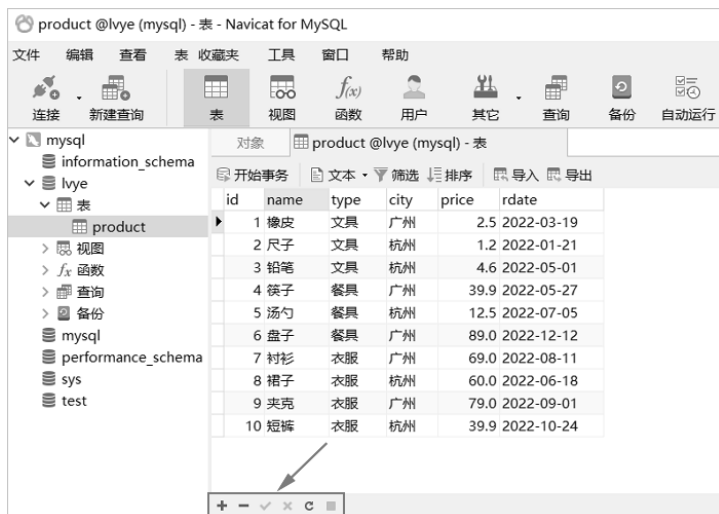


图 1-26

至此就成功创建了一个 product 表。在后面的章节中创建一个新表时，我们都是通过上面这几个步骤来完成的。

1.4.4 运行代码

① **新建查询**：在 Navicat for MySQL 上方单击【新建查询】，就会打开一个代码窗口；然后选择你想要使用的数据库，这里选择的是【lvye】，如图 1-27 所示。



图 1-27

② **运行代码**：在打开的代码窗口中，我们尝试输入一句简单的 SQL 代码 “select * from product;”，然后单击上方的【运行】按钮，如图 1-28 所示，就会自动执行并显示结果。

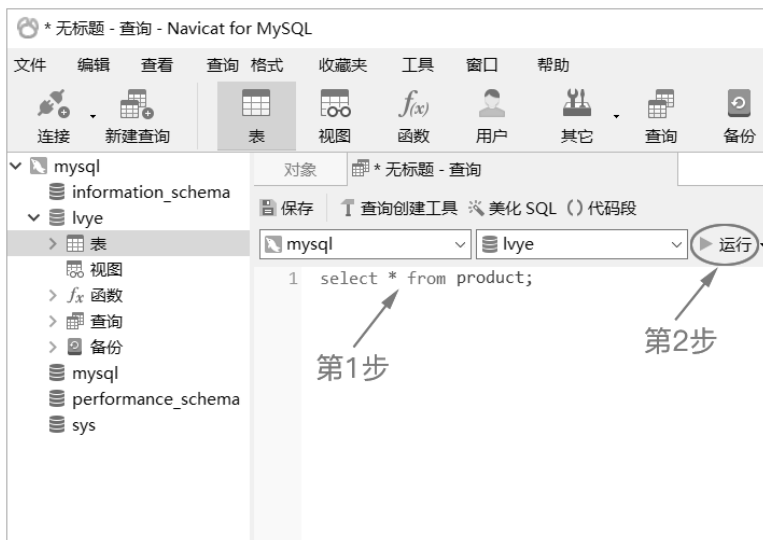


图 1-28

最后，对于 Navicat for MySQL 的使用，我们还需要特别注意以下两点（非常重要）。

- ▶ 在执行 SQL 语句之前，一定要确保选择正确的数据库，否则可能会报错。
- ▶ 所有的 SQL 语句（包括查询、插入、删除等），都是在【新建查询】窗口执行的，而不仅仅只有查询语句才可以。

1.5 本书说明

本书不仅适合零基础的初学者阅读，同样适合想要系统学习且有一定基础的小伙伴阅读。不过本书只会挑选核心的 SQL 语法进行介绍，并不会方方面面都涉及。有需要参考完整语法的小伙伴，一定要参考不同 DBMS 的官方文档。

很多小伙伴在学习技术的时候，都有点儿“眼高手低”，觉得看懂了就可以了。其实技术这东西，看懂了没太多意义，只有自己能够写出来才有意义。所以对于本书的每一个例子，小伙伴们一定要在 Navicat for MySQL 上面亲自操作一遍。

MySQL 是基于 SQL 的，其实我们在学习 MySQL 时，更多的是学习一门语言的各种语法。这里顺便提一点：如果你学过一门编程语言，再去学习其他编程语言，是有非常大的帮助的。比如 SQL 中的存储过程就相当于其他语言中的“函数”，SQL 中同样也有类似其他编程语言的循环语句。

对于没有任何基础的小伙伴来说，学习哪一门语言比较好呢？这里并不推荐 C++、Java 等，因为这些语言本身比较烦琐，也不利于初学者理解。更推荐选择 Python 作为首选入门语言，因为它不仅语法简单，而且应用非常广泛。

常见问题

对于本书每一章后面的练习，大家有必要去做吗？

本书中所有的习题，都是我精挑细选处理的，对于掌握该章的知识点是非常有用的。当然，如果想要真正提升技术能力，仅仅靠练习几道题是远远不够的。小伙伴们还是要通过真正的项目多加练习，才能做到游刃有余。

1.6 本章练习

【特别说明】由于本书使用的是 MySQL，正文内容也以 MySQL 的语法作为主体，所以本书所有习题都是针对 MySQL 而言的。

一、单选题

- 下面选项中，不属于 DBMS 的是（ ）。
A. SQL
B. MySQL
C. SQL Server
D. Oracle
- 下面选项中，属于非关系 DBMS 的是（ ）。
A. MySQL
B. SQL Server
C. Oracle
D. MongoDB
- 下面有关数据库的说法中，正确的是（ ）。
A. MySQL 是非关系 DBMS
B. MySQL 是一门语言
C. MySQL 是使用非常广泛的开源 DBMS
D. MySQL、SQL Server 和 Oracle 的语法是完全一样的
- 下面数据库系统中，应用相对最为广泛的是（ ）。
A. 分布型数据库
B. 逻辑型数据库
C. 关系数据库
D. 层次型数据库

二、简答题

常见的关系 DBMS 和非关系 DBMS 都有哪些？请分别列举。

第 2 章

语法基础

2.1 SQL 是什么

本书中使用的 MySQL 是一个 DBMS，也就是一个软件。MySQL 本身是需要借助 SQL 来实现的。实际上，MySQL、SQL Server、Oracle 等 DBMS 都需要使用 SQL，只不过不同 DBMS 的语法略有不同而已。

2.1.1 SQL 简介

SQL，也就是“**Structured Query Language**（结构化查询语言）”（其 Logo 如图 2-1 所示），它是数据库的标准语言。SQL 非常简洁，它有 6 个常用动词：insert（插入）、delete（删除）、select（查询）、update（更新）、create（创建）和 grant（授权）。

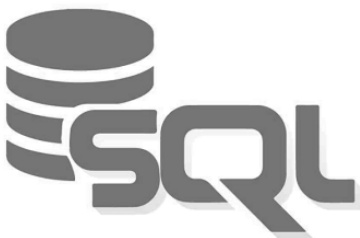


图 2-1

SQL 语言按照实现的功能不同，主要分为三大类：①数据定义语言；②数据操纵语言；③数据控制语言。

1. 数据定义语言

数据定义语言，也就是“DDL”(Data Definition Language)，主要用于对“数据表”进行创建、删除或修改操作。DDL 语句有 3 种，如表 2-1 所示。

表 2-1 DDL 语句

语 句	说 明
create table	创建表
drop table	删除表
alter table	修改表

2. 数据操纵语言

数据操纵语言，也就是“DML”(Data Manipulation Language)，主要用于对“数据”进行增、删、查、改操作。DML 语句有 4 种，如表 2-2 所示。

表 2-2 DML 语句

语 句	说 明
insert	增加数据
delete	删除数据
select	查询数据
update	更新数据

3. 数据控制语言

数据控制语言，也就是“DCL”(Data Control Language)，主要用于用户对数据库和数据表的权限管理。DCL 语句有两种，如表 2-3 所示。

表 2-3 DCL 语句

语 句	说 明
grant	赋予用户权限
revoke	取消用户权限

2.1.2 关键字

关键字，指的是 SQL 本身“已经在使用”的名字，因此我们在给库、表、列等命名时，是不能使用这些名字的（因为 SQL 自己要用）。

常见的关键字有: select、from、where、group by、order by、distinct、like、insert、delete、update、create、alter、drop、is not、inner join、left outer join、right outer join、procedure、function 等。

对于这些关键字,小伙伴们不需要刻意去记忆,等小伙伴们把这本书学完之后,自然而然就会认得了。

2.1.3 语法规则

SQL 本身也是一门编程语言,所以它也有属于自己的语法规则。不过 SQL 的语法规则非常简单,我们只需要清楚以下两点就可以了。

1. 不区分大小写

对于表名、列名、关键字等,SQL 是不区分大小写的。比如 select 这个关键字,写成 select、SELECT 或 Select 都是可以的。下面两种方式是等价的。

```
-- 方式1: 关键字小写
select * from product;

-- 方式2: 关键字大写
SELECT * FROM product;
```

对于大小写这点,这里要重点说明一下。很多书都推荐使用方式 2,也就是关键字一律使用大写的方式。其实对于使用中文环境的人来说,关键字大写这种方式,阅读起来是非常不直观的。因此,对于初学者来说,我们更推荐使用小写的方式。

可能有小伙伴就开始纠结了:“大多数书不都是使用关键字大写这种方式吗?”鲁迅先生说过一句话:“从来如此,便对么?”所以在做开发的时候,我们不必拘泥于别人定的规则,只要团队内部做好约定,不影响实际项目开发就可以了。

当然,上面只是本书的一个约定,并不是强制规定。在实际开发中,我们完全可以根据个人喜好来选择大写还是小写,甚至是大小写混合。

2. 应该以分号结尾

如果执行一条 SQL 语句,它的结尾加不加英文分号(;)都是可行的。但是如果同时执行多条 SQL 语句,每一条语句的后面都必须加上英文分号才行。

```
-- 方式1: 加分号
select * from product;

-- 方式2: 不加分号
select * from product
```


对于中文的一句话来说，我们需要用句号（。）来表示结束。一条 SQL 语句相当于 SQL 中的一句话，所以为了代码规范，不管是一条 SQL 语句，还是多条 SQL 语句，我们都建议一律加上英文分号。

2.1.4 命名规则

命名规则，主要是针对库名、表名、列名这 3 种而言的。对于库、表、列的命名，我们需要遵循以下两条规则。

1. 不能是 SQL 关键字

前面提过，我们在给库、表、列等命名时，是不能使用 SQL 本身已经在用的这些名字的，比如 select、delete、from 等都是 SQL 关键字。

2. 只能使用英文字母、数字、下划线

在给库、表、列等命名时，我们只能使用英文字母（大小写都可以）、数字、下划线（_），而不能使用其他符号，如中划线（-）、美元符号（\$）等。

```
-- 正确命名
product_name

-- 错误命名
product-name
```

本节只是对 SQL 做简单的介绍，小伙伴们暂时了解一下即可。我们建议大家把这本书学完之后，再回头看一遍，这样才会有更深的理解。

2.2 数据类型

如果小伙伴们接触过其他编程语言（如 C、Java、Python 等），应该知道每一门编程语言都有属于它自己的数据类型。SQL 本身也是一门编程语言，所以它也有自己的数据类型。

由于本书使用的是 MySQL，所以只会介绍 MySQL 中的数据类型。对于其他 DBMS（如 SQL Server、Oracle、PostgreSQL 等）来说，它们的数据类型也是大同小异的，小伙伴们可以根据需求自行去了解一下。

MySQL 的数据类型主要有以下四大类。

- ▶ 数字
- ▶ 字符串
- ▶ 日期时间
- ▶ 二进制

2.2.1 数字

数字，指的是数学上的数字（如图 2-2 所示），它是由 0 ~ 9、加号（+）、减号（-）和小数点（.）组成的，比如 10、-10、3.14 等。

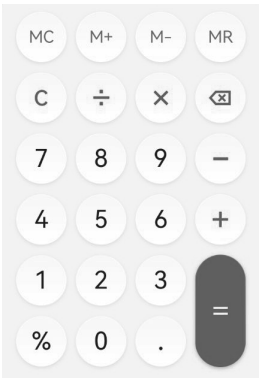


图 2-2

数字可以分为三大类：整数、浮点数和定点数。MySQL 中的整数类型有 5 种，如表 2-4 所示。在实际开发中，大多数情况下我们使用的是 int 类型。

表 2-4 整数类型

类 型	说 明	取值范围
tinyint	很小的整数	$-2^7 \sim 2^7-1$
smallint	小整数	$-2^{15} \sim 2^{15}-1$
mediumint	中等的整数	$-2^{23} \sim 2^{23}-1$
int（或 integer）	普通的整数	$-2^{31} \sim 2^{31}-1$
bigint	大整数	$-2^{63} \sim 2^{63}-1$

选择哪一种整数类型取决于列的范围，如果列中的最大值不超过 127，那么选择 tinyint 类型就足够了。选择范围过大的类型，需要占据更大的空间。

浮点数类型有两种，如表 2-5 所示。需要注意的是，浮点数类型存在精度损失，比如 float 类型的浮点数只保留 7 个有效位，然后对最后一位数四舍五入。

表 2-5 浮点数类型

类 型	说 明	有 效 位
float	单精度	7
double	双精度	15

定点数类型只有一种，如表 2-6 所示。和浮点数不一样，定点数不存在精度损失，所以大多数情况下我们都建议使用定点数来表示包含小数的数值。特别像银行存款这种数值，如果用浮点数来表示，就会非常麻烦。

表 2-6 定点数类型

类 型	说 明	有 效 位
decimal(m, d)	定点数	取决于 m 和 d

对于 decimal(m, d) 来说，m 表示该数值最多包含的有效数字的个数，d 表示有多少位小数。

举个简单的例子，decimal(10, 2) 中的“2”表示小数部分的位数为 2，如果插入的值没有小数部分或者小数部分不足 2 位，就会自动补全到 2 位小数（补 0）；如果插入的值的小数部分超过了 2 位，就会直接截断（不会四舍五入），最后保留 2 位小数。decimal(10, 2) 中的“10”指的是整数部分加小数部分的总位数，也就是整数部分不能超过 8 位（10-2），否则无法插入成功，并且会报错。

此外，decimal(m, d) 中的 m 和 d 都是可选的，m 的默认值是 10，d 的默认值是 0。因此我们可以得出下面的等式。

```
decimal = decimal(10, 0)
```

MySQL 中是不存在布尔型（即 Boolean）的。但是在实际开发中，经常需要用到“是”和“否”、“有”和“无”这种数据，此时应该怎么表示呢？我们可以使用 tinyint(1) 这种类型来存储，其中用“1”表示 true，“0”表示 false 即可。

2.2.2 字符串

字符串，其实就是一串字符，非常容易理解。在 MySQL 中，字符串都是使用英文单引号或英文双引号括起来的。对于 MySQL 来说，常用的字符串型有 7 种，如表 2-7 所示。

表 2-7 字符串型

类 型	说 明	字 节
char	定长字符串	$0 \sim 2^8-1$
varchar	变长字符串	$0 \sim 2^{16}-1$
tinytext	短文本	$0 \sim 2^8-1$
text	普通长度文本	$0 \sim 2^{16}-1$
mediumtext	中等长度文本	$0 \sim 2^{24}-1$
longtext	长文本	$0 \sim 2^{32}-1$
enum	枚举类型	取决于成员个数（最多 64 个）

在实际开发中，我们一般只会用到 char、varchar、text 这 3 种，所以接下来会重点介绍。

1. char

在 MySQL 中，我们可以使用 char 类型来表示一个“固定长度”的字符串。

语法：

char (n)

说明：

n 表示指定的长度，它是一个整数，取值范围为 0 ~ 255。比如 char(5) 表示长度为 5，也就是包含的字符个数最多为 5。如果字符串长度不足 5，那么在右边填充空格；如果字符串长度超过 5，就会报错而无法存入数据库，具体如表 2-8 所示。

表 2-8 char(5)

插 入 值	存 储 值	占用空间
"	' '	5 个字节
'a'	'a '	5 个字节
'ab'	'ab '	5 个字节
'abcde'	'abcde'	5 个字节
'abcdef'	无法存入	无法存入

2. varchar

在 MySQL 中，我们可以使用 varchar 类型来表示一个“可变长度”的字符串。

语法：

varchar (n)

说明：

n 表示指定的长度，它是一个整数，取值范围为 0 ~ 65535。和 char 不一样，varchar 的占用空间是由字符串实际长度来决定的。我们来看一下 varchar(5) 的存储情况，如表 2-9 所示。

表 2-9 varchar(5)

插 入 值	存 储 值	占用空间
"	"	1 个字节
'a'	'a'	2 个字节
'ab'	'ab'	3 个字节
'abcde'	'abcde'	6 个字节
'abcdef'	无法存入	无法存入

需要特别注意的是，varchar 实际的占用空间等于“字符串实际长度”再加上 1，因为它在存储字符串时会在尾部加上一个结束字符。

虽然 varchar 使用起来比较灵活，并且可以节省存储空间，但是从性能上来看，char 的处理速度更快，有时甚至可以超出 varchar 50%。因此在设计数据库时，应该综合考虑多方面的因素，以达到最佳的平衡。

其实从字面上也可以看出来，varchar 指的是“variable char”。下面总结一下 char 和 varchar 的区别。

- ▶ char 也叫作“定长字符串”，它的长度是固定的，存储占用空间大，但是性能稍微高一点儿。
- ▶ varchar 也叫作“变长字符串”，它的长度是可变的，存储占用空间小，但是性能稍微低一点儿。

3. text

如果你的数据是一个超长的字符串，比如文章内容，此时就更适合使用 text 类型。text 其实相当于 varchar(65535)，它本质上也是一个可变长度的字符串。

text 相关类型有 tinytext、mediumtext、longtext 等，如表 2-10 所示。它们都是可变长度的字符串，唯一的区别在于长度不同。

表 2-10 text 相关类型

类 型	说 明	字 节
tinytext	短文本	$0 \sim 2^8-1$
text	普通长度文本	$0 \sim 2^{16}-1$
mediumtext	中等长度文本	$0 \sim 2^{24}-1$
longtext	长文本	$0 \sim 2^{32}-1$

4. enum

在实际开发中，有些变量只有几种可能的取值。比如人的性别只有 2 种值——男和女，而星期只有 7 种值——1、2、3、4、5、6、7。

在 MySQL 中，我们可以定义一个字段为 enum 类型（也就是枚举类型），然后限定该字段在某个范围内取值。

比如有一个字段使用 enum 类型定义了一个枚举列表——first、second、third，那么该字段可以取的值和每个值的索引如表 2-11 所示。

表 2-11 enum 类型的取值范围

值	索 引
NULL	NULL
"	"

(续)

值	索引
first	1
second	2
third	3

如果 enum 类型加上 NOT NULL 属性，其默认值就是枚举列表的第一个元素。如果不加 NOT NULL 属性，enum 类型将允许插入 NULL，而且 NULL 为默认值。

2.2.3 日期时间

日期时间，主要用于表示“日期”(年、月、日)和“时间”(时、分、秒)。对于 MySQL 来说，日期时间型有 5 种，如表 2-12 所示。

表 2-12 日期时间型

类 型	格 式	说 明	举 例
date	YYYY-MM-DD	日期型	2022-01-01
time	HH:MM:SS	时间型	08:05:30
datetime	YYYY-MM-DD HH:MM:SS	日期时间型	2022-01-01 08:05:30
year	YYYY	年份型	2022
timestamp	YYYYMMDD HHMMSS	时间戳型	20220101 080530

我们在给 MySQL 输入日期时间数据时，必须符合上面的格式，才能正确输入。比如类型为 date，那么字段的值必须符合“YYYY-MM-DD”这种格式，而不能是其他格式。

每个类型都有特定的取值格式以及取值范围，当指定不合法的值得时候，系统就会将“0”插入数据库中。

如果使用的是 Navicat for MySQL，我们可以使用提示按钮来辅助输入，如图 2-3 所示。如果是在程序中插入数据，就需要特别注意格式。

1. 日期型

日期型 (date) 的数据格式为: YYYY-MM-DD。其中 YYYY 表示年份，MM 表示月份，DD 表示某一天。比如 2022 年 5 月 20 日，存储格式应为: 2022-05-20。



图 2-3

2. 时间型

时间型 (time) 的数据格式为: HH:MM:SS。其中 HH 表示小时, MM 表示分钟, SS 表示秒。
比如 13 时 14 分 20 秒, 存储格式应为: 13:14:20。

3. 日期时间型

日期时间型 (datetime) 的数据格式为: YYYY-MM-DD HH:MM:SS。其中 YYYY 表示年份, MM 表示月份, DD 表示某一天, HH 表示小时, MM 表示分钟, SS 表示秒。

比如 2022 年 5 月 20 日 13 时 14 分 20 秒, 存储格式应为: 2022-05-20 13:14:20。

4. 年份型

年份型 (year) 的数据格式为: YYYY。其中 YYYY 表示年份。

比如 2022 年, 存储格式应为: 2022。

5. 时间戳型

时间戳型 (timestamp) 的数据格式为: YYYYMMDD HHMMSS。其中 YYYY 表示年份, MM 表示月份, DD 表示某一天, HH 表示小时, MM 表示分钟, SS 表示秒。

比如 2022 年 5 月 20 日 13 时 14 分 20 秒, 存储格式应为: 20220520 131420。

datetime 和 timestamp 都可以用于表示 “YYYY-MM-DD HH:MM:SS” 类型的日期, 除了存储方式、存储大小以及表示范围有所不同之外, 它们没有太大的区别。一般情况下我们使用 datetime 较多, 对于跨时区的业务, 则使用 timestamp 更合适。

2.2.4 二进制

二进制型, 适用于存储图像、格式文本 (如 Word、Excel 文件等)、程序文件等数据。对于 MySQL 来说, 二进制型有 5 种, 如表 2-13 所示。

表 2-13 二进制型

类 型	说 明	字 节
bit	位	$0 \sim 2^8-1$
tinyblob	二进制型的短文本	$0 \sim 2^8-1$
blob	二进制型的普通文本	$0 \sim 2^{16}-1$
mediumblob	二进制型的中文本	$0 \sim 2^{24}-1$
longblob	二进制型的长文本	$0 \sim 2^{32}-1$

不过在实际开发中, 我们并不推荐在数据库中存储二进制数据, 主要是因为二进制数据往往非常大, 会占用过多的存储空间, 并且读写的性能非常差。

2.3 注释

在实际开发中，有时我们需要为 SQL 语句添加一些注释，以方便自己和别人理解代码。

方式 1:

```
-- 注释内容
```

方式 2:

```
/*  
    注释内容  
    注释内容  
    注释内容  
*/
```

所有主流 DBMS（包括 MySQL、SQL Server、Oracle 等）的注释方式都有上面两种。需要特别注意的是，对于方式 1，“--”与注释内容之间必须有一个空格，否则某些 DBMS（如 MySQL）会有问题。

数据库差异性

对于 MySQL 的单行注释来说，它还有一种独有的注释方式，语法如下。

```
# 注释内容
```

不过这种方式是 MySQL 独有的，像 SQL Server、Oracle 等就没有这种方式。为了统一语法，对于单行注释来说，我们还是推荐使用下面这种方式。

```
-- 注释内容
```

最后需要特别说明的是，小伙伴们在学习本书的内容时，大多数情况下关注正文主体内容就可以了。对于“数据库差异性”这个板块，你在学习的时候完全可以忽略，因为它更多是方便我们在使用其他 DBMS 时进行查询。当然，如果小伙伴们想对比一下不同 DBMS 的语法，认真看一下也是非常有用的。

2.4 本章练习

一、单选题

- SQL 又被称为 ()。
A. 结构化定义语言
B. 结构化控制语言
C. 结构化查询语言
D. 结构化操纵语言
- 每一条 SQL 语句的结束符是 ()。
A. 英文句号
B. 英文逗号
C. 英文分号
D. 英文问号
- 下面关于 SQL 的说法中, 不正确的是 ()。
A. SQL 语句中的所有关键字必须大写
B. 每一条 SQL 语句应该以英文分号结尾
C. 库、表和列的命名不能使用 SQL 关键字
D. 库、表和列的命名只能使用英文字母、数字和下划线
- 如果给某一列命名, 下面合法的命名是 ()。
A. product-name
B. product_name
C. product+name
D. \$productname
- 下面不属于数字型的是 ()。
A. decimal
B. enum
C. bigint
D. float
- 下面不属于字符串型的是 ()。
A. float
B. char
C. text
D. varchar
- 下面不属于日期时间型的是 ()。
A. date
B. year
C. decimal
D. timestamp
- 下列选项中, 可以用于注释多行内容的方式是 ()。
A. -- 注释内容
B. /* 注释内容 */
C. # 注释内容
D. / 注释内容 /
- 如果表某一列的取值是不固定长度的字符串, 最适合使用的类型是 ()。
A. char
B. varchar
C. nchar
D. int

二、简答题

请简述一下 MySQL 的数据类型都有哪些。

第 3 章

查询操作

3.1 select 语句简介

从这一章开始，我们正式学习 SQL 的各种语句。前面的 1.4 节已经创建了一个名为“product”的表，该表的结构如表 3-1 所示，该表的数据如表 3-2 所示。小伙伴们认真对比一下，看看是否已正确填写了。

表 3-1 product 表的结构

列 名	类 型	长 度	小 数 点	允许 NULL	是否主键	注 释
id	int			×	√	商品编号
name	varchar	10		√	×	商品名称
type	varchar	10		√	×	商品类型
city	varchar	10		√	×	来源城市
price	decimal	5	1	√	×	出售价格
rdate	date			√	×	入库时间

表 3-2 product 表的数据

id	name	type	city	price	rdate
1	橡皮	文具	广州	2.5	2022-03-19
2	尺子	文具	杭州	1.2	2022-01-21
3	铅笔	文具	杭州	4.6	2022-05-01
4	筷子	餐具	广州	39.9	2022-05-27
5	汤勺	餐具	杭州	12.5	2022-07-05
6	盘子	餐具	广州	89.0	2022-12-12
7	衬衫	衣服	广州	69.0	2022-08-11
8	裙子	衣服	杭州	60.0	2022-06-18
9	夹克	衣服	广州	79.0	2022-09-01
10	短裤	衣服	杭州	39.9	2022-10-24

这里提前给小伙伴们说明一下“主键”的作用。如果某一列被设置为主键，那么这一列的值具有两个特点：**不允许为空（NULL）**，以及**具有唯一性**。一般来说，一个表都需要有一个作为主键的列，这样可以保证每一行都有唯一标识。

举个简单的例子，如果一条记录包含身份证号、姓名、性别、年龄等，那么我们怎样对两个人进行区分呢？很明显，只有通过身份证号才可以，因为姓名、性别、年龄这些都是可能相同的。所以主键就相当于每一行数据的“身份证号”，可以对不同行数据进行区分。

此外，在实际开发中，对于 MySQL 中包含小数的列，建议使用 decimal 类型来表示，而不是使用 float 或 double 类型来表示。这主要是因为 decimal 类型不存在精度损失，而 float 或 double 类型可能会存在精度损失。

3.1.1 select 语句

在 SQL 中，我们可以使用 select 语句来对一个表进行查询操作。其中，select 是 SQL 中的关键字。select 语句是 SQL 所有语句中用得最多的一种语句，如果你能把 select 语句认真掌握好，那么说明离掌握 SQL 已经不远了。

语法：

```
select 列名 from 表名；
```

说明：

select 语句由“select 子句”和“from 子句”这两个部分组成。可能小伙伴们会觉得很奇怪：为什么这里除了“select 语句”这种叫法，还有“select 子句”这样的叫法呢？

实际上，select 语句是对“查询语句”的统称，它是由“子句”组合而成的。所谓的“子句”，指的是语句的一部分，不能单独使用。对于 select 语句来说，它包含的子句主要有 7 种，如表 3-3 所示。

表 3-3 select 语句子句

子 句	说 明
select	查询哪些列
from	从哪个表查询
where	查询条件
group by	分组
having	分组条件
order by	排序
limit	限制行数

从表 3-3 可以看出来，where、group by、order by 等其实都属于查询子句，它们都是配合 select 子句一起使用的。小伙伴们一定要深刻地理解这一点，这样在后续内容的学习中才会有清晰的学习思路。

■ 举例：查询一列

```
select name from product;
```

运行结果如图 3-1 所示。

name
橡皮
尺子
铅笔
筷子
汤勺
盘子
衬衫
裙子
夹克
短裤

图 3-1

■ 分析：

上面这条语句表示从 product 表中查询 name 这一列数据。其中“select name”是 select 子句，而“from product”是 from 子句，整个 select 语句是由 select 子句和 from 子句组成的。

对于 SQL 来说，它的关键字是不区分大小写的。所以对于这个例子来说，下面两种方式是等价的。不过对于初学者来说，我们更推荐使用方式 1，因为它更加直观。

```
-- 方式1
select name from product;

-- 方式2
SELECT name FROM product;
```

■ 举例：查询多列

```
select name, type, price from product;
```

运行结果如图 3-2 所示。

name	type	price
橡皮	文具	2.5
尺子	文具	1.2
铅笔	文具	4.6
筷子	餐具	39.9
汤勺	餐具	12.5
盘子	餐具	89.0
衬衫	衣服	69.0
裙子	衣服	60.0
夹克	衣服	79.0
短裤	衣服	39.9

图 3-2

分析：

如果想要查询多列数据，我们只需要在 select 子句中把多个列名列举出来就可以了。其中，列名之间使用英文逗号（,）隔开。我们把 select 后面跟的东西称为“查询列表”，然后查询结果中列的顺序，是按照 select 子句中列名的顺序（也就是查询列表）来显示的。

对于这个例子来说，就是从 product 表中查询（提取）name、type、price 这 3 列数据，如图 3-3 所示。

id	name	type	city	price	rdate
1	橡皮	文具	广州	2.5	2022-03-19
2	尺子	文具	杭州	1.2	2022-01-21
3	铅笔	文具	杭州	4.6	2022-05-01
4	筷子	餐具	广州	39.9	2022-05-27
5	汤勺	餐具	杭州	12.5	2022-07-05
6	盘子	餐具	广州	89.0	2022-12-12
7	衬衫	衣服	广州	69.0	2022-08-11
8	裙子	衣服	杭州	60.0	2022-06-18
9	夹克	衣服	广州	79.0	2022-09-01
10	短裤	衣服	杭州	39.9	2022-10-24

获取这3列

图 3-3

如果我们改变列的顺序，比如改为下面这条语句，此时运行结果如图 3-4 所示。

```
select type, name, price from product;
```

type	name	price
文具	橡皮	2.5
文具	尺子	1.2
文具	铅笔	4.6
餐具	筷子	39.9
餐具	汤勺	12.5
餐具	盘子	89.0
衣服	衬衫	69.0
衣服	裙子	60.0
衣服	夹克	79.0
衣服	短裤	39.9

图 3-4

当然，同一个列名可以在查询列表中重复出现，比如下面这条 SQL 语句（运行结果如图 3-5 所示）。像这种情况也是可行的，只不过在实际开发中，我们一般不会这样做。

```
select name, name, price from product;
```

name	name(1)	price
橡皮	橡皮	2.5
尺子	尺子	1.2
铅笔	铅笔	4.6
筷子	筷子	39.9
汤勺	汤勺	12.5
盘子	盘子	89.0
衬衫	衬衫	69.0
裙子	裙子	60.0
夹克	夹克	79.0
短裤	短裤	39.9

图 3-5

■ 举例：查询所有列

```
select * from product;
```

运行结果如图 3-6 所示。

id	name	type	city	price	rdate
1	橡皮	文具	广州	2.5	2022-03-19
2	尺子	文具	杭州	1.2	2022-01-21
3	铅笔	文具	杭州	4.6	2022-05-01
4	筷子	餐具	广州	39.9	2022-05-27
5	汤勺	餐具	杭州	12.5	2022-07-05
6	盘子	餐具	广州	89.0	2022-12-12
7	衬衫	衣服	广州	69.0	2022-08-11
8	裙子	衣服	杭州	60.0	2022-06-18
9	夹克	衣服	广州	79.0	2022-09-01
10	短裤	衣服	杭州	39.9	2022-10-24

图 3-6

分析：

如果想要查询所有列，我们可以使用 “*” 来代替所有列。对于这个例子来说，下面两种方式是等价的。

-- 方式1

```
select * from product;
```

-- 方式2

```
select id, name, type, city, price, rdate from product;
```

查询所有列有上面两种方式。很明显，方式 1 比方式 2 更为简单，那是不是意味着更推荐使用方式 1 呢？恰恰相反，我们更推荐使用方式 2，原因有以下两点。

► 使用 “*”，无法指定列的显示顺序。

如果使用 “*” 来表示所有列，我们就无法指定列的显示顺序。此时结果中的列顺序是按照数据表中的列顺序来显示的。如果想要查询所有列，并且想要指定顺序，就需要在 select 子句中把每一个列名都列举出来才行，比如下面这样。

```
select id, city, type, name, price, rdate from product;
```

► 使用 “*”，查询速度会变慢。

想查询“部分列”的数据时，很多小伙伴习惯使用 “*” 来先查询所有列，如果表的数据量比较大，就会导致查询速度变慢。并且数据是需要从服务端传输到客户端的，也会导致传输速度变慢，从而影响用户体验。

即使需要查询表的所有列，我们也更推荐使用方式 2。因为方式 2 的查询速度略高于方式 1。对于方式 1 来说，它需要先转换成方式 2，再去执行。当然，本书为了讲解方便，例子中可能会使用 “*”，但是在实际开发中，我们并不推荐这样做。

3.1.2 特殊列名

如果列名中包含空格，此时应该怎么办呢？比如姓名这一列是“product name”（两个单词中间有一个空格）。可能小伙伴们会写出下面的 SQL 语句。

```
select product name from product;
```

实际上，上面这条语句是无效的。对于包含空格的列名，我们需要使用反引号（`）来把列名括起来。其中反引号在键盘左上方数字 1 的左边，切换到英文状态下可以输入。

我们尝试在 Navicat for MySQL 中将 product 表中“name”这个列名改为“product name”，然后执行下面的 SQL 语句，此时结果如图 3-7 所示。

```
select `product name` from product;
```

product name
橡皮
尺子
铅笔
筷子
汤勺
盘子
衬衫
裙子
夹克
短裤

图 3-7

需要特别注意的是，对于特殊列名（比如包含空格或关键字），我们只能使用反引号括起来，而不能使用单引号或双引号。

```
-- 正确方式  
select `product name` from product;
```

```
-- 错误方式  
select 'product name' from product;
```

```
-- 错误方式  
select "product name" from product;
```

为了方便后面内容的学习，小伙伴们需要手动把“product name”改回原来的“name”。记得一定要改过来，不然会影响后面的学习。

数据库差异性

不同 DBMS 对特殊列名的处理方式是不同的。对于 MySQL 来说，我们需要使用反引号来把特殊列名括起来。

```
select `product name` from product;
```

对于 SQL Server 来说，我们需要在 SSMS（SQL Server Management Studio，SQL Server 管理平台）中使用 “[]” 把特殊列名括起来（如图 3-8 所示），否则会报错。

	列名	数据类型	允许 Null 值
▼	id	int	<input type="checkbox"/>
▶	[product name]	varchar(10)	<input checked="" type="checkbox"/>
	type	varchar(10)	<input checked="" type="checkbox"/>
	city	varchar(10)	<input checked="" type="checkbox"/>
	price	decimal(5, 1)	<input checked="" type="checkbox"/>
	rdate	date	<input checked="" type="checkbox"/>

图 3-8

比如商品名称这一列的名字是 “product name”，此时表结构中的列名应该写成 “[product name]”。SQL 语句中，该列名也应该写成 “[product name]”，而不能写成 “product name”。

-- 正确

```
select [product name] from product;
```

-- 错误

```
select product name from product;
```

和其他 DBMS（如 MySQL、SQL Server 等）不一样，Oracle 中的列名是不允许包含特殊字符的。如果 Oracle 中的列名包含特殊字符，比如我们尝试将列名 “name” 改为 “product name”，此时 Oracle 就会报错，如图 3-9 所示。

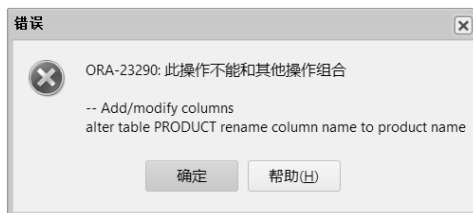


图 3-9

总结：在实际开发中，我们并不推荐使用包含特殊字符的列名，因为这样会导致各种意想不到的问题。

3.1.3 换行说明

在实际开发中，如果一条 SQL 语句过长，我们可以使用换行的方式来分割它。一般情况下，我们根据这样一个规则来进行分割：**一个子句占据一行**。

比如 `select name, type, price from product;` 这条 SQL 语句，如果使用换行的方式，我们可以写成下面这样。

```
select name, type, price
from product;
```

需要注意的是，行与行之间不允许出现空行，否则会报错。比如写成下面这样就是错误的。

```
select name, type, price

from product;
```

总而言之，我们记住这么一点就可以了：如果 SQL 语句比较短，只在一行写就可以了；如果 SQL 语句比较长，则可以使用以“子句”为单位进行换行的方式来书写。

常见问题

1. 对于很多书中提到的“字段”“记录”这些概念，我们应该怎么理解呢？

在 SQL 中，列也叫作“字段”，一列也叫作“一个字段”，列名就叫作“字段名”；行也叫作“记录”，一行数据也叫作“一条记录”，有多少行数据就叫作“多少条记录”，如图 3-10 所示。

id	name	type	city	price	rdate
1	橡皮	文具	广州	2.5	2022-03-19
2	尺子	文具	杭州	1.2	2022-01-21
3	铅笔	文具	杭州	4.6	2022-05-01
4	筷子	餐具	广州	39.9	2022-05-27
5	汤勺	餐具	杭州	12.5	2022-07-05
6	盘子	餐具	广州	89.0	2022-12-12
7	衬衫	衣服	广州	69.0	2022-08-11
8	裙子	衣服	杭州	60.0	2022-06-18
9	夹克	衣服	广州	79.0	2022-09-01
10	短裤	衣服	杭州	39.9	2022-10-24

一条记录

一个字段

图 3-10

字段和记录这两个概念非常重要，在很多地方都会运用，小伙伴们一定要搞清楚它们指的是什么。

2. 对于表的命名，为什么使用单数，而不是复数呢？

在实际开发中，我们应该遵守这样一个规则：**表名应该使用单数，而不是复数**。比如商品信息表应该命名为“product”，而不是“products”；而学生信息表应该命名为“student”，而不是“students”。

可能小伙伴们会觉得很奇怪，商品信息表一般包含多种商品，应该命名为 products 才对啊！实际上，这样理解是不正确的。

如果大家接触过其他编程语言（如 C++、Java、Python 等），肯定了解过类的定义。实际上，一个数据表就相当于一个类，而一个列就相当于类的一个属性。对于类来说，它是一个抽象的概念，我们在定义的时候，使用的是单数，而不是复数。

这样去对比理解，其实就很容易清楚为什么表名应该使用单数。因为表名就相当于类名，列名就相当于属性名。

3.2 使用别名: as

3.2.1 as 关键字

在使用 SQL 查询数据时，我们可以使用 as 关键字来给列起一个别名。别名的作用是增强代码和查询结果的可读性。

语法：

```
select 列名 as 别名  
from 表名;
```

说明：

在实际开发中，一般建议在以下几种情况中使用别名。对于内置函数、多表查询，我们在后续章节中会详细介绍。

- ▶ 列名比较长或可读性差。
- ▶ 使用内置函数。
- ▶ 用于多表查询。
- ▶ 需要把两个或更多的列放在一起。

■ 举例：英文别名

```
select name as product_name  
from product;
```

运行结果如图 3-11 所示。

product_name
橡皮
尺子
铅笔
筷子
汤勺
盘子
衬衫
裙子
夹克
短裤

图 3-11

■ 分析：

使用 as 来指定别名之后，查询结果中列名就从 name 变成了 product_name。需要清楚的是，as 关键字是可以省略的。下面两种方式是等价的。

```
-- 方式1  
select name as product_name  
from product;  
  
-- 方式2  
select name product_name  
from product;
```

不过在实际开发中，我们更推荐把 as 关键字加上，这样可以使代码的可读性更高。

■ 举例：中文别名

```
select name as 名称  
from product;
```

运行结果如图 3-12 所示。

名称
橡皮
尺子
铅笔
筷子
汤勺
盘子
衬衫
裙子
夹克
短裤

图 3-12

分析:

除了可以指定英文别名, 还可以指定中文别名。不过需要注意的是, 别名只在本次查询的结果中显示, 并不会改变真实表中的列名。我们在 Navicat for MySQL 中打开 product 表, 会发现列名并未改变, 如图 3-13 所示。

id	name	type	city	price	rdate
1	橡皮	文具	广州	2.5	2022-03-19
2	尺子	文具	杭州	1.2	2022-01-21
3	铅笔	文具	杭州	4.6	2022-05-01
4	筷子	餐具	广州	39.9	2022-05-27
5	汤勺	餐具	杭州	12.5	2022-07-05
6	盘子	餐具	广州	89.0	2022-12-12
7	衬衫	衣服	广州	69.0	2022-08-11
8	裙子	衣服	杭州	60.0	2022-06-18
9	夹克	衣服	广州	79.0	2022-09-01
10	短裤	衣服	杭州	39.9	2022-10-24

图 3-13

举例: 为多个列指定别名

```
select name as 名称, type as 类型, price as 售价
from product;
```

运行结果如图 3-14 所示。

名称	类型	售价
橡皮	文具	2.5
尺子	文具	1.2
铅笔	文具	4.6
筷子	餐具	39.9
汤勺	餐具	12.5
盘子	餐具	89.0
衬衫	衣服	69.0
裙子	衣服	60.0
夹克	衣服	79.0
短裤	衣服	39.9

图 3-14

分析：

如果需要指定别名的列比较多，我们可以分行来写。对于这个例子来说，可以写成下面这样。

```
select name as 名称,
       type as 类型,
       price as 售价
from product;
```

3.2.2 特殊别名

在使用 as 关键字起别名时，如果别名中包含关键字或者特殊字符，比如空格、加号、减号等，那么该别名必须使用英文双引号括起来。

MySQL、SQL Server、Oracle 都是这样处理的：英文单引号用于表示字符串，英文双引号用于表示列名，而别名本身就相当于列名。

举例：包含空格

```
select name as "商品 名称"
from product;
```

运行结果如图 3-15 所示。

商品 名称
橡皮
尺子
铅笔
筷子
汤勺
盘子
衬衫
裙子
夹克
短裤

图 3-15

分析:

在这个例子中, 由于别名包含空格, 所以我们必须使用英文双引号括起来, 否则会有问题。下面这种写法是错误的。

```
-- 错误写法
select name as 商品 名称
from product;
```

需要注意的是, 这里的引号必须是双引号, 而不允许是单引号或者反引号。由于列名的别名本身还充当列的名字, 所以我们应该使用英文双引号。

```
-- 正确
select name as "商品 名称"
from product;
```

```
-- 错误
select name as '商品 名称'
from product;
```

```
-- 错误
select name as `商品 名称`
from product;
```

举例: 包含 “-”

```
select name as "product-name"
from product;
```

运行结果如图 3-16 所示。

product-name
橡皮
尺子
铅笔
筷子
汤勺
盘子
衬衫
裙子
夹克
短裤

图 3-16

分析：

使用 as 关键字来指定别名，在实际开发中非常有用。比如当对多个列进行计算时，计算之后会产生一个新列，此时我们可以使用 as 关键字来为这个新列指定一个名字。如果不指定别名，那么默认列名就是 SQL 自己处理的名字，阅读体验并不是很好。

不同数据表中可能存在相同的列名（如 id、name 等），当我们同时对多个表进行查询操作时，结果中可能会出现相同的列名，这种情况很容易给人造成困惑。所以，此时使用 as 关键字来指定别名就非常有用。

常见问题

1. 对于中文别名来说，我们是否有必要使用引号括起来呢？

对于 SQL 来说，如果想要起一个中文别名，我们其实是没有必要使用引号括起来的。当然，使用引号括起来也没有问题。

2. as 关键字是不是只能给列起一个别名？我们能否给表也起一个别名呢？

很多小伙伴只知道 as 关键字可以给列起一个别名，实际上它还可以给表起一个别名。只不过对于单表查询，一般我们不会这样做。

不过对于多表查询，如果表名比较复杂，此时给表起一个别名就非常有用，这样可以让我们们的代码更加直观。

3.3 条件子句：where

在 SQL 中，我们可以使用 where 子句来指定查询的条件。其中，where 子句都是配合 select 子句使用的。

语法：

```
select 列名
from 表名
where 条件;
```

说明：

对于 where 子句来说，它一般都是需要结合运算符来使用的，主要包括以下 3 种。

- ▶ 比较运算符
- ▶ 逻辑运算符
- ▶ 其他运算符

3.3.1 比较运算符

在 where 子句中，我们可以使用比较运算符来指定查询的条件。其中，常用的比较运算符如表 3-4 所示。

表 3-4 常用的比较运算符

运 算 符	说 明
>	大于
<	小于
=	等于
>=	大于或等于
<=	小于或等于
!>	不大于（相当于 <=）
!<	不小于（相当于 >=）
!= 或 <>	不等于

对于 SQL 中的运算符，我们需要清楚以下两点。

- ▶ 对于“等于”来说，SQL 使用的是“=”而不是“==”，这一点和其他编程语言不同。
- ▶ 对于“不等于”来说，SQL 有两种表示方式：“!=”和“<>”。

举例：等于（数字）

```
select name, price
from product
where price = 2.5;
```

运行结果如图 3-17 所示。

name	price
橡皮	2.5

图 3-17

■ 分析：

对于这个例子来说，它其实是获取 price 等于 2.5 的所有记录。对于等号来说，它不仅可以用于对数字的判断，也可以用于对字符串的判断，请看下面的例子。

■ 举例：等于（字符串）

```
select name, price
from product
where name = '尺子';
```

运行结果如图 3-18 所示。

name	price
尺子	1.2

图 3-18

■ 分析：

需要注意的是，name 这个列的值类型是一个字符串，所以 '尺子' 两边的英文引号是不能去掉的。

```
-- 正确方式
select name, price
from product
where name = '尺子';
```

```
-- 错误方式
select name, price
from product
where name = 尺子;
```

■ 举例：大于或小于

```
select name, price
from product
where price > 20;
```

运行结果如图 3-19 所示。

name	price
筷子	39.9
盘子	89.0
衬衫	69.0
裙子	60.0
夹克	79.0
短裤	39.9

图 3-19

分析:

price>20 表示查询 price 大于 20 的所有记录。当我们把 price>20 改为 price<=20 之后, 运行结果如图 3-20 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
汤勺	12.5

图 3-20

举例: 日期时间

```
select name, rdate
from product
where rdate <= '2022-06-01';
```

运行结果如图 3-21 所示。

name	rdate
橡皮	2022-03-19
尺子	2022-01-21
铅笔	2022-05-01
筷子	2022-05-27

图 3-21

分析:

比较运算符同样可以用于日期时间型数据, rdate<='2022-06-01' 表示查询 rdate 小于或等于 '2022-06-01' 的所有记录。

当比较运算符用于日期时间型数据时, 我们需要清楚以下 3 点。

- ▶ 小于某个日期时间，指的是在该日期时间之前。
- ▶ 大于某个日期时间，指的是在该日期时间之后。
- ▶ 等于某个日期时间，指的是处于该日期时间。

数据库差异性

对于 MySQL 和 SQL Server 来说，它们的日期时间型数据可以和字符串直接比较。但是对于 Oracle 来说，我们需要先使用 Oracle 内置的 `to_date()` 函数将字符串转换为日期时间型数据，然后才能进行比较。

```
-- MySQL和SQL Server
select name, rdate
from product
where rdate <= '2022-06-01';

-- Oracle
select name, rdate
from product
where rdate <= to_date('2022-06-01', 'YYYY-MM-DD');
```

3.3.2 逻辑运算符

在 where 子句中，如果需要同时指定多个查询条件，此时就需要使用逻辑运算符。在 SQL 中，常见的逻辑运算符有 3 种，如表 3-5 所示。

表 3-5 常见的逻辑运算符

运 算 符	说 明
and	与
or	或
not	非

对于“与运算”来说，我们使用 `and` 关键字来表示。如果执行的是 `where A and B`，那么要求 A 和 B 这两个条件同时为真（true），where 子句才会返回真。

对于“或运算”来说，我们使用 `or` 关键字来表示。如果执行的是 `where A or B`，那么只要 A 和 B 这两个条件有一个为真，where 子句就会返回真。

对于“非运算”来说，我们使用 `not` 关键字来表示。如果执行的是 `where not price>20`，那么相当于执行 `where price<=20`。

举例：与运算

```
select name, price
from product
where price > 20 and price < 40;
```

运行结果如图 3-22 所示。

name	price
筷子	39.9
短裤	39.9

图 3-22

分析：

上面这条 SQL 语句表示查询 price 大于 20 且小于 40 的所有记录。也就是说，需要同时满足 price>20 和 price<40 这两个条件，该条记录（也就是该行数据）才会被查询出来。

如果想要指定更多的条件，我们使用更多的 and 即可。比如执行下面这条语句，运行结果如图 3-23 所示。

```
select name, price
from product
where price > 20 and price < 40 and type = '餐具';
```

name	price
筷子	39.9

图 3-23

举例：或运算

```
select name, price
from product
where price < 20 or price > 40;
```

运行结果如图 3-24 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
汤勺	12.5
盘子	89.0
衬衫	69.0
裙子	60.0
夹克	79.0

图 3-24

分析：

上面这条 SQL 语句表示查询 price 小于 20 或者大于 40 的所有记录。也就是说，只要满足 price<20 和 price>40 这两个条件的任意一个，该条记录就会被查询出来。

举例：非运算

```
select name, price
from product
where not price > 20;
```

运行结果如图 3-25 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
汤勺	12.5

图 3-25

分析：

上面这条 SQL 语句表示查询 price 不大于 20 的所有记录。对于这个例子来说，它可以等价于下面这条 SQL 语句。

```
select name, price
from product
where price <= 20;
```

not 运算符用于否定某一个条件，但很多时候不使用 not 运算符的查询条件可读性更好，比如 where price<=20 就比 where not price>20 更容易理解。即便如此，我们也不能完全否定 not 运算符的作用。实际上，在编写复杂的 SQL 语句时，not 运算符还是非常有用的。

数据库差异性

对于 MySQL 来说，它的逻辑运算符有两种写法：一种是“关键字”，如表 3-6 所示；另一种是“符号”，如表 3-7 所示。

表 3-6 关键字

运 算 符	说 明
and	与
or	或
not	非

表 3-7 符号

运 算 符	说 明
&&	与
	或
!	非

“符号”这种写法只适用于 MySQL，而不适用于 SQL Server 和 Oracle。所以为了统一规范，对于逻辑运算符，我们推荐使用“关键字”这种写法。

3.3.3 其他运算符

除了比较运算符和逻辑运算符之外，SQL 还有一些其他的运算符，如表 3-8 所示。这些运算符也是非常重要的，小伙伴们也要认真掌握。

表 3-8 其他的运算符

运 算 符	说 明
is null 或 isnull	是否值为 NULL
is not null	是否值不为 NULL
in	是否为列表中的值
not in	是否不为列表中的值
between A and B	是否处于 A 和 B 之间
not between A and B	是否不处于 A 和 B 之间

1. is null 和 is not null

当某一个字段（即某一行）没有录入数据（也就是数据为空）时，该字段的值就是 NULL。特别注意一点，NULL 代表该字段没有值，而不是代表该字段的值为 0 或 "（空字符串）。

接下来我们创建一个名为“product_miss”的表，专门用于测试 is null 和 is not null 这两种运算符。product_miss 表的结构如表 3-9 所示，其数据如表 3-10 所示。

表 3-9 product_miss 表的结构

列 名	类 型	长 度	小 数 点	允许 NULL	是否主键	注 释
id	int			×	√	商品编号
name	varchar	10		√	×	商品名称
type	varchar	10		√	×	商品类型

(续)

列 名	类 型	长 度	小 数 点	允许 NULL	是否主键	注 释
city	varchar	10		√	×	来源城市
price	decimal	5	1	√	×	出售价格
rdate	date			√	×	入库时间

表 3-10 product_miss 表的数据

id	name	type	city	price	rdate
1	橡皮	文具	广州	2.5	2022-03-19
2	尺子	文具	杭州	NULL	2022-01-21
3	铅笔	NULL	杭州	4.6	2022-05-01
4	筷子	NULL	广州	39.9	2022-05-27
5	汤勺	餐具	杭州	NULL	2022-07-05
6	盘子	餐具	广州	89.0	2022-12-12
7	衬衫	NULL	广州	69.0	2022-08-11
8	裙子	衣服	杭州	NULL	2022-06-18
9	夹克	衣服	广州	79.0	2022-09-01
10	短裤	衣服	杭州	39.9	2022-10-24

■ 举例：is null

```
select *
from product_miss
where price is null;
```

运行结果如图 3-26 所示。

id	name	type	city	price	rdate
2	尺子	文具	杭州	(Null)	2022-01-21
5	汤勺	餐具	杭州	(Null)	2022-07-05
8	裙子	衣服	杭州	(Null)	2022-06-18

图 3-26

■ 分析：

如果想要判断某一列的值是否为 NULL，不允许使用“=”或“!=”这样的比较运算符，而必须使用 is null 或 is not null 这种运算符。小伙伴们自行试一下 where price=null 这种方式就知道了。


```
-- 正确
select *
from product_miss
where price is null;

-- 错误
select *
from product_miss
where price = null;
```

对于这个例子来说, 如果将 is null 改为 is not null, 此时运行结果如图 3-27 所示。其中, is null 和 is not null 的操作是相反的。

id	name	type	city	price	rdate
1	橡皮	文具	广州	2.5	2022-03-19
3	铅笔	(Null)	杭州	4.6	2022-05-01
4	筷子	(Null)	广州	39.9	2022-05-27
6	盘子	餐具	广州	89.0	2022-12-12
7	衬衫	(Null)	广州	69.0	2022-08-11
9	夹克	衣服	广州	79.0	2022-09-01
10	短裤	衣服	杭州	39.9	2022-10-24

图 3-27

2. in 和 not in

在 SQL 中, 我们可以使用 in 运算符来判断列表中是否“存在”某个值, 也可以使用 not in 运算符来判断列表中是否“不存在”某个值。其中, in 和 not in 的操作是相反的。

语法:

```
where 列名 in (值1, 值2, ..., 值n)
```

说明:

对于值列表来说, 我们需要使用 “()” 括起来, 并且值与值之间使用 “,” 隔开。

举例:

```
select name, price
from product
where name in ('橡皮', '尺子', '铅笔');
```

运行结果如图 3-28 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6

图 3-28

■ 分析：

where name in ('橡皮', '尺子', '铅笔') 表示判断 name 的值是否为 '橡皮'、'尺子'、'铅笔' 中的任意一个。如果是，则满足条件。

上面例子等价于下面的 SQL 语句。很明显，in 运算符这种方式更简单。

```
select name, price
from product
where name = '橡皮' or name = '尺子' or name = '铅笔';
```

对于这个例子来说，如果我们把 in 改为 not in，此时运行结果如图 3-29 所示。

name	price
筷子	39.9
汤勺	12.5
盘子	89.0
衬衫	69.0
裙子	60.0
夹克	79.0
短裤	39.9

图 3-29

在实际开发中，如果想要查询某些值之外的值，not in 运算符就非常有用了，小伙伴们一定要学会灵活应用才行。

3. between...and... 和 not between...and...

在 MySQL 中，如果想要判断某一列的值是否在某个范围之间，我们可以使用 between...and... 运算符来实现。

■ 语法：

```
where 列名 between 值1 and 值2
```

■ 说明：

between A and B 的取值范围为 [A, B]，包含 A 也包含 B。

■ 举例：

```
select name, price
from product
where price between 20 and 40;
```

运行结果如图 3-30 所示。

name	price
筷子	39.9
短裤	39.9

图 3-30

分析:

上面这条 SQL 语句用于查询 price 为 20 ~ 40 的所有记录。对于这个例子来说, 下面两种方式是等价的。

```
-- 方式1
select name, price
from product
where price between 20 and 40;

-- 方式2
select name, price
from product
where price >= 20 and price <= 40;
```

其中, 方式 1 比方式 2 更加直观。在实际开发中, 我们也更推荐使用方式 1。

对于这个例子来说, 如果把 between...and... 改为 not between...and..., 此时运行结果如图 3-31 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
汤勺	12.5
盘子	89.0
衬衫	69.0
裙子	60.0
夹克	79.0

图 3-31

举例: 用于判断日期时间型数据

```
select name, rdate
from product
where rdate between '2022-01-01' and '2022-06-01';
```

运行结果如图 3-32 所示。

name	rdate
橡皮	2022-03-19
尺子	2022-01-21
铅笔	2022-05-01
筷子	2022-05-27

图 3-32

分析：

between...and... 同样可以用于判断日期时间型数据。where rdate between '2022-01-01' and '2022-06-01' 表示获取 rdate 处于“2022-01-01”和“2022-06-01”之间的数据。

3.3.4 运算符优先级

所谓的优先级，也就是执行顺序。我们知道，数学中的加、减、乘、除运算是有一定优先级的。比如先算括号里的，然后算“乘、除”，最后才算“加、减”。

在 SQL 中，运算符也是有优先级的。规则很简单：优先级高的先运算，优先级低的后运算，优先级相同的则从左到右进行运算。对于运算符优先级，我们只需要清楚以下两个规则就可以了。

- ▶ 对于算术运算来说，“乘除”比“加减”优先级要高。
- ▶ 对于逻辑运算来说，非（not）>与（and）>或（or）。

举例：

```
select name, city, price
from product
where city = '广州' and price < 20 or price > 40;
```

运行结果如图 3-33 所示。

name	city	price
橡皮	广州	2.5
盘子	广州	89.0
衬衫	广州	69.0
裙子	杭州	60.0
夹克	广州	79.0

图 3-33

分析：

由于与运算（and）的优先级比或运算（or）的高，所以上面这条 SQL 语句等价于下面的 SQL 语句。

```
select name, city, price
from product
where (city = '广州' and price < 20) or price > 40;
```

虽然不加“()”也没有关系，但是在实际开发中，我们建议加上一些必要的“()”，这样可以提高代码的可读性。

常见问题

1. 对于 SQL 中的字符串来说，我们应该使用英文单引号还是英文双引号呢？

对于 MySQL 来说，字符串既可以使用英文单引号来表示，也可以使用英文双引号来表示。但是在实际开发中，我们更建议使用英文单引号来表示。为什么要这样建议呢？

这是因为对于其他 DBMS（如 SQL Server 和 Oracle）来说，它们只能使用英文单引号来表示字符串，如果使用英文双引号来表示，就会报错。所以为了统一规范，在所有的 DBMS 中，我们都建议使用英文单引号来表示字符串。

英文双引号在 SQL 中用得比较少，小伙伴们不要搞那么多新花样，不然可能会导致各种意想不到的问题。

2. 对于 select 语句来说，from 子句是必要的吗？

从前文可知，select 语句一般是由“select 子句”和“from 子句”组成的。但实际上，对于 select 语句来说，from 子句并不是必要的，我们完全可以单独使用 select 子句。

如果只使用 select 子句而没有使用 from 子句，那么它代表的就不是从数据表中查询数据了，而是用于单独进行计算。计算的结果会使用表格来展示，其中列名就是该 select 子句的表达式。比如执行下面这条语句，此时运行结果如图 3-34 所示。

```
select 1000 + 2000;
```

1000+2000
3000

图 3-34

当然，我们也可以使用 as 关键字来改变默认的列名。比如执行下面这条语句，此时运行结果如图 3-35 所示。

```
select 1000 + 2000 as '计算结果';
```

计算结果
3000

图 3-35

这里我们只需要简单了解一下某些 DBMS 中的 from 子句并不是必要的就可以了。在实际开发中，我们很少会单独去使用 select 子句。

3.4 排序子句：order by

3.4.1 order by 子句

在 SQL 中，我们可以使用 order by 子句来对某一列按大小排序。其中，order 子句是作为 select 语句的一部分来使用的。

语法：

```
select 列名  
from 表名  
order by 列名 asc或desc;
```

分析：

asc 表示升序排列，也就是从小到大排列。desc 表示降序排列，也就是从大到小排列。其中，asc 是“ascend”（上升）的缩写，而 desc 是“descend”（下降）的缩写。

举例：升序排列

```
select name, price  
from product  
order by price asc;
```

运行结果如图 3-36 所示。

name	price
尺子	1.2
橡皮	2.5
铅笔	4.6
汤勺	12.5
筷子	39.9
短裤	39.9
裙子	60.0
衬衫	69.0
夹克	79.0
盘子	89.0

图 3-36

分析:

如果你想要使用升序排列, 那么后面的 asc 是可以省略的。对于这个例子来说, 下面两种方式是等价的。

```
-- 不省略 asc
select name, price
from product
order by price asc;

-- 省略 asc
select name, price
from product
order by price;
```

举例: 降序排列

```
select name, price
from product
order by price desc;
```

运行结果如图 3-37 所示。

name	price
盘子	89.0
夹克	79.0
衬衫	69.0
裙子	60.0
筷子	39.9
短裤	39.9
汤勺	12.5
铅笔	4.6
橡皮	2.5
尺子	1.2

图 3-37

分析:

对于升序排列来说, 后面的 asc 可以省略。但是对于降序排列来说, 后面的 desc 是不允许省略的。

举例: 对多列排序

```
select name, price, rdate
from product
order by price desc, rdate desc;
```

运行结果如图 3-38 所示。

name	price	rdate
盘子	89.0	2022-12-12
夹克	79.0	2022-09-01
衬衫	69.0	2022-08-11
裙子	60.0	2022-06-18
短裤	39.9	2022-10-24
筷子	39.9	2022-05-27
汤勺	12.5	2022-07-05
铅笔	4.6	2022-05-01
橡皮	2.5	2022-03-19
尺子	1.2	2022-01-21

图 3-38

分析：

上面这条 SQL 语句，表示同时对 price 和 rdate 这两列进行降序排列。由于这里执行的是多列排序，首先会对 price 这一列进行排序。排序好之后，如果两个商品的 price 相同的话，再对 rdate 进行排序。

由于 asc 和 desc 这两个关键字是以列为单位指定的，所以可以同时指定一个列为降序排列，而指定另一个列为升序排列，这样也是可行的，比如下面这样的写法。

```
select name, price, rdate
from product
order by price asc, rdate desc;
```

举例：使用别名

```
select name as 名称, price as 售价
from product
order by 售价 desc;
```

运行结果如图 3-39 所示。

名称	售价
盘子	89.0
夹克	79.0
衬衫	69.0
裙子	60.0
筷子	39.9
短裤	39.9
汤勺	12.5
铅笔	4.6
橡皮	2.5
尺子	1.2

图 3-39

分析:

如果我们在 select 子句中给列名起了一个别名,那么在 order by 子句中可以使用这个别名来代替原来的列名。当然,在 order by 子句中,不管是使用原名还是别名,得到的结果都是一样的。

对于这个例子来说,下面两种方式的查询结果是一样的。

```
-- 使用原名
select name as 名称, price as 售价
from product
order by price desc;

-- 使用别名
select name as 名称, price as 售价
from product
order by 售价 desc;
```

举例: 对日期时间排序

```
select name, rdate
from product
order by rdate desc;
```

运行结果如图 3-40 所示。

name	rdate
盘子	2022-12-12
短裤	2022-10-24
夹克	2022-09-01
衬衫	2022-08-11
汤勺	2022-07-05
裙子	2022-06-18
筷子	2022-05-27
铅笔	2022-05-01
橡皮	2022-03-19
尺子	2022-01-21

图 3-40

分析:

order by 同样可以对日期时间型数据进行排序。

举例: 结合 where 子句

```
select name, price
from product
where price < 10
order by price desc;
```

运行结果如图 3-41 所示。

name	price
铅笔	4.6
橡皮	2.5
尺子	1.2

图 3-41

分析：

order by 子句可以结合 where 子句来使用，其中 order by 子句必须放在 where 子句的后面。首先执行 where 子句来筛选数据，然后执行 order by 子句来排序。

3.4.2 中文字符串字段排序

默认情况下 MySQL 使用的是 UTF-8 字符集，此时对于中文字符串字段的排序，并不会按照中文拼音来进行排序。如果想要按照中文拼音来排序，我们需要借助 convert() 函数来实现。

语法：

```
order by convert(列名 using gbk);
```

说明：

convert(列名 using gbk) 表示强制该列使用 GBK 字符集。

举例：默认情况

```
select name, price  
from product  
order by name;
```

运行结果如图 3-42 所示。

name	price
夹克	79.0
尺子	1.2
橡皮	2.5
汤勺	12.5
盘子	89.0
短裤	39.9
筷子	39.9
衬衫	69.0
裙子	60.0
铅笔	4.6

图 3-42

分析:

从运行结果可以看出来, name 这一列并没有按照中文拼音进行排序。

举例: 使用 convert()

```
select name, price
from product
order by convert(name using gbk);
```

运行结果如图 3-43 所示。

name	price
衬衫	69.0
尺子	1.2
短裤	39.9
夹克	79.0
筷子	39.9
盘子	89.0
铅笔	4.6
裙子	60.0
汤勺	12.5
橡皮	2.5

图 3-43

分析:

使用 convert() 函数之后, name 这一列就会按照中文拼音进行排序。如果想要降序排列, 可以在后面加上 desc。

数据库差异性

对于中文字符字段的排序, 不同 DBMS 的语法是不一样的。下面补充说明一下 SQL Server 和 Oracle 的语法。

► SQL Server。

对于 SQL Server 来说, 对于中文字符字段的排序, 默认情况下会直接按照中文拼音来排序。

举例:

```
select name, price
from product
order by name;
```

运行结果如图 3-44 所示。

name	price
衬衫	69.0
尺子	1.2
短裤	39.9
夹克	79.0
筷子	39.9
盘子	89.0
铅笔	4.6
裙子	60.0
汤勺	12.5
橡皮	2.5

图 3-44

分析：

从运行结果可以看出来，name 这一列已经按照中文拼音进行排序了。

► Oracle。

对于 Oracle 来说，默认情况下并不能直接使用 order by 来对中文字符串字段进行排序。如果想要对中文字符串字段进行排序，我们需要结合 Oracle 自带的 nlssort() 函数来实现。

语法：

```
-- 按照拼音排序
order by nlssort(列名, 'nls_sort=schinese_pinyin_m')

-- 按照笔画排序
order by nlssort(列名, 'nls_sort=schinese_stroke_m')

-- 按照部首排序
order by nlssort(列名, 'nls_sort=schinese_radical_m')
```

说明：

在实际开发中，一般都是按照拼音排序，很少会按照笔画或部首排序。

举例：按照拼音排序

```
select name, price
from product
order by nlssort(name, 'nls_sort=schinese_pinyin_m');
```

运行结果如图 3-45 所示。

NAME	PRICE
衬衫	69.0
尺子	1.2
短裤	39.9
夹克	79.0
筷子	39.9
盘子	89.0
铅笔	4.6
裙子	60.0
汤勺	12.5
橡皮	2.5

图 3-45

举例：按照笔画排序

```
select name, price
from product
order by nlssort(name, 'nls_sort=schinese_stroke_m');
```

运行结果如图 3-46 所示。

NAME	PRICE
尺子	1.2
夹克	79.0
汤勺	12.5
衬衫	69.0
铅笔	4.6
盘子	89.0
短裤	39.9
裙子	60.0
筷子	39.9
橡皮	2.5

图 3-46

举例：按照部首排序

```
select name, price
from product
order by nlssort(name, 'nls_sort=schinese_radical_m');
```

运行结果如图 3-47 所示。

NAME	PRICE
夹克	79.0
尺子	1.2
橡皮	2.5
汤勺	12.5
盘子	89.0
短裤	39.9
筷子	39.9
衬衫	69.0
裙子	60.0
铅笔	4.6

图 3-47

常见问题

1. SQL 能不能对字符串列进行排序呢，还是说只能对数字列进行排序？

数字是可以比较大小的，所以我们可以对数字列进行排序。实际上字符串也是可以比较大小的，所以 SQL 同样能对字符串列进行排序。

比较两个字符串的大小，其实就是依次比较每个字符的 ASCII。先比较两者的第 1 个字符，第 1 个字符较大的，就代表整个字符串大，后面就不用再比较了。如果两者的第 1 个字符相同，就接着比较第 2 个字符，第 2 个字符较大的，就代表整个字符串大，以此类推。

两个字符比较的是 ASCII 的大小。对于 ASCII，小伙伴们可以自行搜索一下，这里就不展开介绍了。注意，空格在字符串中也是被当成一个字符来处理的。

2. 当使用 order by 进行排序时，NULL 值是如何处理的呢？

如果存在 NULL 值，升序排列时会将有 NULL 值的行显示在最前面，降序排列时会将有 NULL 值的行显示在最后面。你可以这样理解：NULL 是该列的最小值。

3.5 限制行数：limit

3.5.1 limit 子句

默认情况下，select 语句会把符合条件的“所有行（即所有记录）”都查询出来。如果得到的结果有 100 条记录，而我们只需要获取前 10 条记录，此时应该怎么办呢？

对于 MySQL 来说, 我们可以使用 limit 这个关键字来获取前 n 行数据。

语法:

```
select 列名  
from 表名  
limit n;
```

说明:

如果 select 语句有 where 子句或 order by 子句, 则 limit n 需要放在最后才行。

举例: 获取前 n 行

```
select name, price  
from product  
limit 5;
```

运行结果如图 3-48 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
筷子	39.9
汤勺	12.5

图 3-48

分析:

limit 5 表示只获取查询结果的前 5 行数据。对于这个例子来说, 如果把 limit 5 去掉, 也就是没有了行数限制, 此时结果如图 3-49 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
筷子	39.9
汤勺	12.5
盘子	89.0
衬衫	69.0
裙子	60.0
夹克	79.0
短裤	39.9

图 3-49

■ 举例：结合 where 子句

```
select name, price
from product
where price > 10
limit 5;
```

运行结果如图 3-50 所示。

name	price
筷子	39.9
汤勺	12.5
盘子	89.0
衬衫	69.0
裙子	60.0

图 3-50

■ 分析：

where price>10 表示获取 price 大于 10 的所有记录。对于这个例子来说，如果把 limit 5 去掉，也就是没有了行数限制，此时结果如图 3-51 所示。

name	price
筷子	39.9
汤勺	12.5
盘子	89.0
衬衫	69.0
裙子	60.0
夹克	79.0
短裤	39.9

图 3-51

■ 举例：结合 order by 子句

```
select name, price
from product
order by price desc
limit 5;
```

运行结果如图 3-52 所示。

name	price
盘子	89.0
夹克	79.0
衬衫	69.0
裙子	60.0
筷子	39.9

图 3-52

分析:

对于这个例子来说, 首先使用 order by 子句来对 price 这一列进行降序排列, 然后使用 limit 5 来获取前 5 条记录, 此时得到的就是售价最高的前 5 条记录了。

如果想要获取售价最低的前 5 条记录, 只需要先对 price 进行升序排列, 再使用 limit 5 就可以了。实现代码如下, 其运行结果如图 3-53 所示。

```
select name, price
from product
order by price asc
limit 5;
```

name	price
尺子	1.2
橡皮	2.5
铅笔	4.6
汤勺	12.5
筷子	39.9

图 3-53

3.5.2 深入了解

想要获取售价最高的前 5 条记录, 使用 limit 关键字就可以很轻松地实现。但是如果让你获取售价最高的第 2 ~ 5 条记录, 此时又应该怎么实现呢?

其实像这种情况, 我们也是使用 limit 关键字来实现的, 只不过语法稍微有点儿不一样。

语法:

```
limit start, n
```

说明:

start 表示开始位置, 默认是 0。n 表示获取 n 条记录。

举例: 获取第 2 ~ 5 条记录

```
select name, price
from product
order by price desc
limit 1, 4;
```

运行结果如图 3-54 所示。

name	price
夹克	79.0
衬衫	69.0
裙子	60.0
筷子	39.9

图 3-54

分析：

执行了 `order by price desc` 之后，我们得到了一个对 `price` 进行降序排列的结果。然后 `limit 1, 4` 表示从查询结果中的第 1 条记录开始（不包括第 1 条记录），共截取 4 条记录，如图 3-55 所示。

name	price
盘子	89.0
夹克	79.0
衬衫	69.0
裙子	60.0
筷子	39.9
短裤	39.9
汤勺	12.5
铅笔	4.6
橡皮	2.5
尺子	1.2

图 3-55

实际上 `limit` 后面只有一个参数时，比如 `limit 5`，它等价于 `limit 0, 5`。也就是说 `limit` 后面的第 1 个参数是可选的，而第 2 个参数是必选的。

在实际开发中，使用 `limit` 关键字结合 `order by` 子句来获取前 `n` 条记录这种方式非常有用，比如获取热度最高的前 10 条新闻、获取浏览量最高的前 10 篇文章等。所以对于这种方式，我们应该重点掌握。

不过我们也应该清楚，`limit` 并不一定要结合 `order by` 子句来一起使用，这里小伙伴们不要误解了。

数据库差异性

对于获取前 `n` 条数据，MySQL、SQL Server 和 Oracle 的语法都是不一样的，下面来补充说明一下 SQL Server 和 Oracle 的语法。

► SQL Server。

在 SQL Server 中，我们可以使用 `top` 关键字来获取前 `n` 条数据。

语法:

```
select top n 列名  
from 表名;
```

说明:

top 关键字需要放在 select 的后面, 并且放在所有列名之前。n 可以是正整数, 也可以是百分数。

举例: 正整数

```
select top 5 name, price  
from product;
```

运行结果如图 3-56 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
筷子	39.9
汤勺	12.5

图 3-56

分析:

top 5 表示只获取查询结果的前 5 条数据。对于这个例子来说, 如果把 top 5 去掉, 也就是没有了行数限制, 此时结果如图 3-57 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
筷子	39.9
汤勺	12.5
盘子	89.0
衬衫	69.0
裙子	60.0
夹克	79.0
短裤	39.9

图 3-57

举例：百分数

```
select top 50 percent name, price
from product;
```

运行结果如图 3-58 所示。

name	price
橡皮	2.5
尺子	1.2
铅笔	4.6
筷子	39.9
汤勺	12.5

图 3-58

分析：

这个例子获取的是前 50% 的数据，由于 product 表共有 10 条数据，所以这里获取的是前 5 条数据。

► Oracle。

在 Oracle 中，如果想要获取前 n 条数据，我们可以借助 rownum 这个“伪列”来实现。所谓的伪列，也叫作“虚列”，指的是物理上是不存在的，只有在对表进行查询操作时才会构造出来的列。

对于 rownum 伪列来说，它有以下两个特点。

- rownum 不属于任何表，它一开始是不存在的，只有查询操作有结果时才可以使用。
- rownum 总是从 1 开始，并且一般只能与“<”或“<=”一起使用。

举例：

```
select product.*, rownum
from product;
```

运行结果如图 3-59 所示。

ID	NAME	TYPE	CITY	PRICE	RDATE	ROWNUM
1	橡皮	文具	广州	2.5	2022/3/19	1
2	尺子	文具	杭州	1.2	2022/1/21	2
3	铅笔	文具	杭州	4.6	2022/5/1	3
4	筷子	餐具	广州	39.9	2022/5/27	4
5	汤勺	餐具	杭州	12.5	2022/7/5	5
6	盘子	餐具	广州	89.0	2022/12/12	6
7	衬衫	衣服	广州	69.0	2022/8/11	7
8	裙子	衣服	杭州	60.0	2022/6/18	8
9	夹克	衣服	广州	79.0	2022/9/1	9
10	短裤	衣服	杭州	39.9	2022/10/24	10

图 3-59

分析:

`product.*` 表示获取 `product` 表下面的所有列, 这种写法我们在后面“第 10 章 多表查询”中会经常见到。由于 `rownum` 不属于任何表, 所以不能写成 `product.rownum`。下面两种方式都是错误的。

```
-- 方式1
select product.*, product.rownum
from product;

-- 方式2
select *, rownum
from product;
```

举例: 获取前 5 条数据

```
select *
from product
where rownum <= 5;
```

运行结果如图 3-60 所示。

ID	NAME	TYPE	CITY	PRICE	RDATE
1	橡皮	文具	广州	2.5	2022/3/19
2	尺子	文具	杭州	1.2	2022/1/21
3	铅笔	文具	杭州	4.6	2022/5/1
4	筷子	餐具	广州	39.9	2022/5/27
5	汤勺	餐具	杭州	12.5	2022/7/5

图 3-60

分析:

如果想要获取 `product` 表中的前 5 条数据, 我们可以使用上面这种方式。但如果想要获取售价最高的前 5 条数据, 此时应该怎么做呢? 很多小伙伴会写出下面这样的代码, 结果如图 3-61 所示。

```
select *
from product
where rownum <= 5
order by price;
```

ID	NAME	TYPE	CITY	PRICE	RDATE
2	尺子	文具	杭州	1.2	2022/1/21
1	橡皮	文具	广州	2.5	2022/3/19
3	铅笔	文具	杭州	4.6	2022/5/1
5	汤勺	餐具	杭州	12.5	2022/7/5
4	筷子	餐具	广州	39.9	2022/5/27

图 3-61

从结果可以看出来,上面这种方式是有问题的。原因很简单,如果 where 子句和 order by 子句同时存在,那么 Oracle 就会先执行 where 子句,此时得到的是原表中前 5 条数据,如图 3-62 所示。

ID	NAME	TYPE	CITY	PRICE	RDATE
1	橡皮	文具	广州	2.5	2022/3/19
2	尺子	文具	杭州	1.2	2022/1/21
3	铅笔	文具	杭州	4.6	2022/5/1
4	筷子	餐具	广州	39.9	2022/5/27
5	汤勺	餐具	杭州	12.5	2022/7/5

图 3-62

然后执行 order by 子句,也就是将上面得到的结果集再根据 price 进行排序,此时得到的结果如图 3-63 所示。

ID	NAME	TYPE	CITY	PRICE	RDATE
2	尺子	文具	杭州	1.2	2022/1/21
1	橡皮	文具	广州	2.5	2022/3/19
3	铅笔	文具	杭州	4.6	2022/5/1
5	汤勺	餐具	杭州	12.5	2022/7/5
4	筷子	餐具	广州	39.9	2022/5/27

图 3-63

如果想要获取售价最高的前 5 条数据,正确的做法应该是使用子查询来实现:首先在子查询中使用 order by 根据 price 进行降序排列,然后从得到的结果集中获取前 5 条数据就可以了。请看下面的例子。

■ 举例:获取售价最高的前 5 条数据

```
select * from (
    select * from product order by price desc
)
where rownum <= 5;
```

运行结果如图 3-64 所示。

ID	NAME	TYPE	CITY	PRICE	RDATE
6	盘子	餐具	广州	89.0	2022/12/12
9	夹克	衣服	广州	79.0	2022/9/1
7	衬衫	衣服	广州	69.0	2022/8/11
8	裙子	衣服	杭州	60.0	2022/6/18
4	筷子	餐具	广州	39.9	2022/5/27

图 3-64

3.6 去重处理: distinct

在 SQL 中,我们可以使用 distinct 关键字来实现数据去重。所谓的数据去重,指的是查询结果中如果包含多个重复行,结果只会保留其中一行。

语法:

```
select distinct 字段列表  
from 表名;
```

说明:

distinct 关键字用于 select 子句中,它总是紧跟在 select 关键字之后,并且放在第一个列名之前。此外,distinct 关键字作用于整个字段列表的所有列,而不是单独某一列。

举例:用于一列

```
select distinct type  
from product;
```

运行结果如图 3-65 所示。

type
文具
餐具
衣服

图 3-65

分析:

如果我们想要知道 type 这一列都有哪几种取值,就可以在 type 这个列名前面加上 distinct 关键字。对于这个例子来说,如果把 distinct 关键字去除,此时得到的结果如图 3-66 所示。

type
文具
文具
文具
餐具
餐具
餐具
衣服
衣服
衣服
衣服

图 3-66

■ 举例：

```
select distinct city  
from product;
```

运行结果如图 3-67 所示。

city
广州
杭州

图 3-67

■ 分析：

distinct city 表示对 city 这一列进行去重处理。如果把 distinct 关键字去除，此时得到的结果如图 3-68 所示。

city
广州
杭州
杭州
广州
杭州
广州
杭州
广州
杭州

图 3-68

■ 举例：NULL 数据

```
select distinct type  
from product_miss;
```

运行结果如图 3-69 所示。

type
文具
(Null)
餐具
衣服

图 3-69

分析:

需要清楚的是, NULL 被视为一类数据。如果列中存在多个 NULL 值, 则只会保留一个 NULL。
对于这个例子来说, 如果把 distinct 关键字去除, 此时得到的结果如图 3-70 所示。

type
文具
文具
(Null)
(Null)
餐具
餐具
(Null)
衣服
衣服
衣服

图 3-70

举例: 用于多列

```
select distinct type, city
from product;
```

运行结果如图 3-71 所示。

type	city
文具	广州
文具	杭州
餐具	广州
餐具	杭州
衣服	广州
衣服	杭州

图 3-71

分析:

对于这个例子来说, 如果没有使用 distinct 关键字, 此时得到的结果如图 3-72 所示。

type	city
文具	广州
文具	杭州
文具	杭州
餐具	广州
餐具	杭州
餐具	广州
衣服	广州
衣服	杭州
衣服	广州
衣服	杭州

图 3-72

使用 distinct 关键字之后，查询结果中重复的多条记录就只会保留其中一条。比如这里本来有 2 条“文具、杭州”记录，最后只会保留第 1 条记录。

需要注意的是，distinct 关键字只能放在第一个列名之前，然后它就会对后面所有的列进行去重处理。下面两种方式都是错误的。

```
-- 错误方式1
select type, distinct city
from product;

-- 错误方式2
select distinct type, distinct city
from product;
```

3.7 本章练习

一、单选题

1. 如果把一张表看成一个类，那么（ ）就相当于表的属性。
A. 行 B. 记录 C. 列 D. 数值
2. 在 MySQL 中，我们通常使用（ ）来表示字段没有值或缺值。
A. NULL B. EMPTY C. 0 D. ""
3. 在 MySQL 中，select 语句执行的结果是（ ）。
A. 数据库 B. 基本表 C. 临时表 D. 数据项
4. 在 MySQL 中，我们可以使用（ ）关键字来过滤查询结果中的重复行。
A. distinct B. limit C. like D. in

13. 下面有一段 SQL 代码，说法中正确的是（ ）。

```
select distinct type, city  
from product;
```

- A. distinct 只会作用于 type 这一列
- B. distinct 只会作用于 city 这一列
- C. distinct 同时作用于 type 和 city 这两列
- D. 语法有误，运行报错

二、简答题

请简述一下你对 NULL 值的理解。

三、编程题

下面有一个 student 表（如表 3-11 所示），请写出对应的 SQL 语句。

表 3-11 student 表

id	name	sex	grade	birthday	major
1	张欣欣	女	86	2000-03-02	计算机科学
2	刘伟达	男	92	2001-06-13	网络工程
3	杨璐璐	女	72	2000-05-01	软件工程
4	王明刚	男	80	2002-10-17	电子商务
5	张伟	男	65	2001-11-09	人工智能

- （1）查询成绩为 80 ~ 100 的学生基本信息。
- （2）查询所有学生基本信息，按照成绩从高到低排序。
- （3）查询成绩前 3 名的学生基本信息。
- （4）查询所有学生的 name、grade、major 这 3 列。
- （5）查询所有学生的 name、grade 这 2 列，并且给 name 起一个别名“姓名”，给 grade 也起一个别名“成绩”。