

第 3 章

结构化查询语言

CHAPTER 3



结

结构化查询语言(SQL)是使用关系模型的数据库应用语言。“不以规矩,不能成方圆。”作为程序员,必须要遵守规则与规范。遵守一定的规则与规范可以使程序更容易阅读和调试。运用运算符则可以让程序员灵活地使用表中的数据,常见的运算符类型有算术运算符、比较运算符、逻辑运算符和位运算符。本章将通过丰富的案例和 7 个课业任务分别演示算术运算符、比较运算符、逻辑运算符和位运算符的使用,并通过 MySQL Workbench 工具和 Navicat Premium 工具进行一些运算。

【教学目标】

- 掌握 SQL 的规则与规范;
- 熟练掌握 MySQL 运算符的使用;
- 掌握运算符的优先级;
- 能够使用 MySQL Workbench 工具进行比较运算;
- 能够使用 Navicat Premium 工具进行综合运算。

【课业任务】

王小明想利用 MySQL+Java 开发一个数据库学习系统,在掌握了 MySQL 的安装与配置以后,需要熟悉 SQL 的规则与规范,并能够灵活地运用运算符处理数据,现通过 7 个课业任务来完成。

课业任务 3-1 算术运算符的应用

课业任务 3-2 比较运算符的应用

课业任务 3-3 逻辑运算符的应用

课业任务 3-4 位运算符的应用

课业任务 3-5 运算符的综合应用

课业任务 3-6 使用 MySQL Workbench 工具进行比较运算

课业任务 3-7 使用 Navicat Premium 工具进行综合运算

3.1 SQL 的概述

3.1.1 SQL 的背景

1974 年,IBM 公司的研究员发布了一篇关于数据库技术的论文——《SEQUEL: 一门结构化的英语查询语言》,从而引出 SQL,直到目前为止,这种结构化查询语言并没有太大的变化,与其他语言相比,SQL 的半衰期可以说是非常长了。

SQL 是使用关系模型的数据库应用语言,可以应用到所有关系数据库中,如 MySQL、Oracle、SQL Server、达梦等,它同时也是与数据直接“打交道”的一种语言。SQL 由 IBM 公司在 20 世纪 70 年代开发出来,由美国国家标准学会(American National Standards Institute, ANSI)着手制定标准,先后有 SQL-86、SQL-89、SQL-92、SQL-99 等标准。其中最重要的两个标准是 SQL-92 和 SQL-99,分别代表了 1992 年和 1999 年颁布的 SQL 标准,现在的 SQL 依然遵循这些标准。

自从 SQL 加入了 TIOBE 编程语言排行榜,就一直保持在前 10,如图 3-1 所示。

Jan 2023	Jan 2022	Change	Programming Language	Ratings	Change
1	1		 Python	16.36%	+2.78%
2	2		 C	16.26%	+3.82%
3	4	^	 C++	12.91%	+4.62%
4	3	v	 Java	12.21%	+1.55%
5	5		 C#	5.73%	+0.05%
6	6		 Visual Basic	4.64%	-0.10%
7	7		 JavaScript	2.87%	+0.78%
8	9	^	 SQL	2.50%	+0.70%
9	8	v	 Assembly language	1.60%	-0.25%
10	11	^	 PHP	1.39%	-0.00%
11	10	v	 Swift	1.20%	-0.21%
12	13	^	 Go	1.14%	+0.10%
13	12	v	 R	1.04%	-0.21%
14	15	^	 Classic Visual Basic	0.98%	+0.01%
15	16	^	 MATLAB	0.91%	-0.05%

图 3-1 TIOBE 编程语言的排行榜

不同的数据库生产厂商都支持 SQL,但很多数据库有自己特有的内容,如 MySQL 中的 LIMIT 语句就是 MySQL 特有的语句,其他数据库都不支持。当然,Oracle 和 SQL Server 也都有自己特有的语句,数据库的特有内容图解如图 3-2 所示。

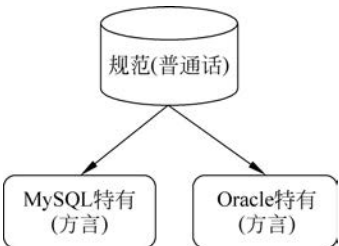


图 3-2 数据库的特有内容图解

3.1.2 SQL 的分类

SQL 在功能上主要分为以下 3 类。

(1) 数据定义语言(Data Definition Language, DDL): 主要用于定义数据库、数据表、索引、视图等数据库对象,还可以用来创建、删除、修改数据库和数据表的结构。其关键字包括 CREATE、DROP、ALTER 等。

(2) 数据操作语言(Data Manipulation Language,DML): 主要用于添加、删除、更新和查询数据库记录,并检查数据的完整性。其关键字包括 INSERT、DELETE、UPDATE、SELECT 等,其中 SELECT 是 SQL 的基础,最为重要。

(3) 数据控制语言(Data Control Language,DCL): 主要用于定义数据库、数据表、字段、用户的访问权限和安全级别。其关键字包括 GRANT、REVOKE、COMMIT、ROLLBACK、SAVEPOINT 等。

因为查询语句的使用非常频繁,所以可以把查询语句单独作为一类,另外将 COMMIT、ROLLBACK 单独作为一类,具体如下。

(1) 数据查询语言(Data Query Language,DQL): 主要用于查询记录(数据),执行 SELECT 操作。数据库执行 DQL 语句不会对数据进行改变,而是让数据库发送结果集给客户端。

(2) 事务控制语言(Transaction Control Language,TCL): 主要用于事务管理,如果需要在事务中包含多条 SQL 语句,那么需要开启事务和结束事务。其关键字包括 COMMIT(提交)、ROLLBACK(回滚)、SAVEPOINT(保存点)等。

3.2 SQL 的规则与规范

3.2.1 SQL 的基本规则

作为程序员,规则是一定要遵守的,如果不遵守规则,程序就会出错,无法执行。以下是使用 SQL 必须遵守的基本规则。

(1) SQL 语句可以单行或多行书写。为了提高可读性,各子句可分行书写,必要时使用缩进。

(2) 每条命令以分号(;)、\g 或 \G 结尾。

(3) 关键字不能缩写,也不能分行。

(4) 必须保证所有括号、单引号、双引号是成对结束的。

(5) 必须使用英文状态下的半角输入方式。

(6) 字符串和日期时间类型的数据可以使用单引号表示。

(7) 列的别名尽量使用双引号,而且不建议省略 AS。

3.2.2 SQL 大小写的规范

规范是建议程序员去遵守的,遵守一定的规范可以使程序更容易阅读和调试,提高复写性。以下是 SQL 大小写的规范。

(1) MySQL 在 Windows 环境下是大小写不敏感的。

(2) MySQL 在 Linux 环境下是大小写敏感的: 数据库名、表名、表的别名、变量名是严格区分大小写的; 关键字、函数名、列名(字段名)、列的别名(字段的别名)是忽略大小写的。

(3) 推荐采用统一的书写规范: 数据库名、表名、字段名、变量名、视图名、索引名、存储名、触发器名等都用小写; SQL 关键字、函数、数据类型等都大写。

3.2.3 MySQL 注释符

注释用于说明 SQL 语句的作用或存储过程中代码块的逻辑,在解析 SQL 代码时,MySQL

会忽略注释部分,只执行除注释以外的 SQL 部分。MySQL 可以使用以下 3 种注释结构。

- (1) 单行注释: # 注释文字。
- (2) 单行注释: -- 注释文字(--后面必须有一个空格)。
- (3) 多行注释: /* 注释文字 */。

3.2.4 MySQL 数据库对象的命名规范

用户在使用 MySQL 时需要遵循一定的命名规范,扬长避短。以下是 MySQL 数据库对象的命名规范。

- (1) 数据库名、表名不能超过 30 个字符,变量名的长度限制为 29。
- (2) 名称中只能包含 A~Z、a~z、0~9,共 62 个字符。
- (3) 数据库名、表名、字段名等对象名中不能有空格。
- (4) 在同一个 MySQL 软件中,数据库不能重名;在同一个数据库中,表不能重名;在同一个表中,字段不能重名。
- (5) 必须保证字段没有与保留字、数据库系统或常用方法冲突。如果一定要使用,则必须在 SQL 语句中使用着重号(`)引起来。
- (6) 在命名字段并为其指定数据类型时一定要保证一致性。

 3.3 运算符

3.3.1 算术运算符

算术运算符主要用于数学运算,其可以连接运算符前后的两个数值或表达式,对数值或表达式进行加(+)、减(-)、乘(*)、除(/)和求模(或称为求余,%)运算。MySQL 中的算术运算符如表 3-1 所示。

表 3-1 算术运算符

运 算 符	名 称	作 用
+	加法运算符	计算两个值或表达式的和
-	减法运算符	计算两个值或表达式的差
*	乘法运算符	计算两个值或表达式的乘积
/或 DIV	除法运算符	计算两个值或表达式的商
%或 MOD	求模(求余)运算符	计算两个值或表达式的余数

1. 加法运算符与减法运算符

加法运算符(+)用于计算一个或多个值的和;减法运算符(-)用于从一个值中减去另外一个值。

【案例 3-1】 加法运算符和减法运算符的应用。

在 MySQL 中,数字 6 加上一个数字或减去一个数字的 SQL 语句如下。

```
SELECT 6, 6 + 0, 6 - 0, 6 + 10, 6 - 10, 6 + 10 - 2, 6 + 4.5, 6 - 4.5
FROM dual;
```

执行上述 SQL 语句,运行结果如图 3-3 所示。

说明:

- (1) 一个整数类型的值对整数(浮点数)进行加法和减法操作,运算结果还是一个整数(浮

```
mysql> SELECT 6, 6+0, 6-0, 6+10, 6-10, 6+10-2, 6+4.5, 6-4.5
-> FROM dual;
```

6	6+0	6-0	6+10	6-10	6+10-2	6+4.5	6-4.5
6	6	6	16	-4	14	10.5	1.5

1 row in set (0.00 sec)

图 3-3 加法运算符和减法运算符的应用

点数)。

(2) 加法运算符和减法运算符的优先级相同,进行先加后减操作与进行先减后加操作的结果是一样的。

(3) 在 MySQL 中不需要虚拟表(dual)也可以进行加法、减法运算。

(4) 在早期的 MySQL 5.1 版本之前是没有虚拟表(dual)的,现在大多使用 MySQL 5.5 以上的版本,所以在 MySQL 中存在和 Oracle 中类似的虚拟表(dual),官方声明是为了满足 SELECT...FROM...结构,在 MySQL 中会忽略对该表的引用。

在 MySQL(SQL 语句需要以分号结尾)和 SQL Server(SQL 语句不需要以分号结尾)中可以直接用 SELECT 1,不需要加“FROM 表名”就可以执行,但在 Oracle(SQL 语句需要以分号结尾)中需要满足 SELECT...FROM...结构,Oracle 不能直接执行“SELECT 1;”语句,需要引入虚拟表(dual)满足结构。以下 SQL 语句是在 MySQL 中不用虚拟表(dual)执行加法和减法运算。

```
SELECT 6, 6 + 0, 6 - 0, 6 + 10, 6 - 10, 6 + 10 - 2, 6 + 4.5, 6 - 4.5;
```

执行上述 SQL 语句,运行结果如图 3-4 所示。

```
mysql> SELECT 6, 6+0, 6-0, 6+10, 6-10, 6+10-2, 6+4.5, 6-4.5;
```

6	6+0	6-0	6+10	6-10	6+10-2	6+4.5	6-4.5
6	6	6	16	-4	14	10.5	1.5

1 row in set (0.00 sec)

图 3-4 不用虚拟表(dual)执行加法和减法运算的结果

2. 乘法运算符与除法运算符

乘法运算符由星号(*)表示,用于两数相乘;除法运算符由斜杠(/)或 DIV 表示,用一个值除以另外一个值得到商。

【案例 3-2】 乘法运算符和除法运算符的应用。

在 MySQL 中,数字 20 乘以一个数字和除以一个数字的 SQL 语句如下。

```
SELECT 20, 20 * 1, 20/1, 20 * 0.5, 20/0.5, 20 * 3, 20/3, 20 * 2/5, 20/5 * 2, 20 DIV 0;
```

执行上述 SQL 语句,运行结果如图 3-5 所示。

```
mysql> SELECT 20, 20*1, 20/1, 20*0.5, 20/0.5, 20*3, 20/3, 20*2/5, 20/5*2, 20 DIV 0;
```

20	20*1	20/1	20*0.5	20/0.5	20*3	20/3	20*2/5	20/5*2	20 DIV 0
20	20	20.0000	10.0	40.0000	60	6.6667	8.0000	8.0000	NULL

1 row in set, 1 warning (0.01 sec)

图 3-5 乘法运算符和除法运算符的应用

说明:

- (1) 一个数乘以或除以浮点数后都变成浮点数,数值与原数相等。
- (2) 一个数除以整数后,不管是否能除尽,结果都是浮点数,并且结果都保留到小数点后

4 位。

(3) 因为乘法运算符和除法运算符的优先级相同,所以进行先乘后除操作与进行先除后乘操作的结果相同。

(4) 在数学运算中,0 不能用作除数。但是在 MySQL 中,一个数除以 0 的返回值为 NULL。

3. 求模(求余)运算符

求模(求余)运算符由百分号(%)表示,执行常规的除法运算,返回经过除法运算后得到的余数。在 MySQL 中%和 MOD 的作用相同,返回值一样。

3.3.2 比较运算符

比较运算符是在查询数据时最常用的一类运算符,用来对表达式左、右两边的操作数进行比较,如果比较的结果为真,返回 1;如果比较的结果为假,返回 0;如果是其他情况,返回 NULL。比较运算符经常在 SELECT 的查询条件子句中使用,用来查询满足指定条件的记录。MySQL 中的比较运算符又分为常见的算术比较运算符和非符号类型的比较运算符。

1. 常见的算术比较运算符

MySQL 中常见的算术比较运算符如表 3-2 所示。

表 3-2 常见的算术比较运算符

运 算 符	名 称	作 用
=	等于运算符	判断两个值、字符串或表达式是否相等
<=>	安全等于运算符	安全地判断两个值、字符串或表达式是否相等
<>、!=	不等于运算符	判断两个值、字符串或表达式是否不相等
<	小于运算符	判断前面的值、字符串或表达式是否小于后面的
<=	小于或等于运算符	判断前面的值、字符串或表达式是否小于或等于后面的
>	大于运算符	判断前面的值、字符串或表达式是否大于后面的
>=	大于或等于运算符	判断前面的值、字符串或表达式是否大于或等于后面的

1) 等于运算符

等于运算符(=)用于判断等号两边的值、字符串或表达式是否相等。如果相等,返回 1;如果不相等,返回 0。

【案例 3-3】 等于运算符的应用。

在 MySQL 中,执行等于运算符的运算的 SQL 语句如下。

```
SELECT 3=1, 3=3, '3'=2, 'm'='m', (3+0)=(1+2), 3=NULL, NULL=NULL;
```

执行上述 SQL 语句,运行结果如图 3-6 所示。

mysql> SELECT 3=1, 3=3, '3'=2, 'm'='m', (3+0)=(1+2), 3=NULL, NULL=NULL;						
3=1	3=3	'3'=2	'm'='m'	(3+0)=(1+2)	3=NULL	NUL=NULL
0	1	0	1	1	NULL	NULL
1 row in set (0.01 sec)						

图 3-6 等于运算符的应用

说明:

- (1) 等号两边的操作数都是整数,则 MySQL 会按照整数值比较两个操作数的大小。
- (2) 等号两边的操作数一个是整数,另一个是字符串,则 MySQL 会将字符串转换为数字

进行比较。

(3) 等号两边的操作数都是字符串,则 MySQL 会按照字符串进行比较。

(4) 等号两边的值、字符串或表达式中有一个为 NULL,则比较结果为 NULL。

2) 安全等于运算符

安全等于运算符(<=>)和等于运算符(=)的作用相似,当操作数相等时值都为 1,唯一的区别是<=>运算符在即使操作数是 NULL 时也可以正确比较。

说明:

(1) <=>运算符与=运算符的操作相同。

(2) 当两个操作数均为 NULL 时,返回值为 1,而不是 NULL;当一个操作数为 NULL 时,返回值为 0,而不是 NULL。

3) 不等于运算符

不等于运算符(<>和!=)用于判断运算符两边的数字、字符串或表达式是否不相等,如果不相等,返回 1;如果相等,返回 0。

【案例 3-4】 不等于运算符的应用。

在 MySQL 中,执行不等于运算符的运算的 SQL 语句如下。

```
SELECT 2 <> 2, 2!=2, 'mysql'<>'sql', 'm'<>NULL, NULL<>NULL;
```

执行上述 SQL 语句,运行结果如图 3-7 所示。

mysql> SELECT 2<>2, 2!=2, 'mysql'<>'sql', 'm'<>NULL, NULL<>NULL;				
2<>2	2!=2	'mysql'<>'sql'	'm'<>NULL	NULL<>NULL
0	0	1	NULL	NULL
1 row in set (0.00 sec)				

图 3-7 不等于运算符的应用

说明:

(1) <>和!=运算符两边的值不相等时返回 1,否则返回 0。

(2) 不等于运算符不能判断 NULL 值,若两边的值有一个为 NULL,或两边都为 NULL,结果都为 NULL。

4) 小于运算符和小于或等于运算符

小于运算符(<)和小于或等于运算符(<=)用来判断左侧的操作数是否小于、小于或等于右侧的操作数,如果小于、小于或等于,返回值为 1,否则返回值为 0。

【案例 3-5】 小于运算符和小于或等于运算符的应用。

在 MySQL 中,执行小于运算符、小于或等于运算符的运算的 SQL 语句如下。

```
SELECT 3<2, 2<=2, 'b'<'c', 'm'<NULL, NULL<NULL;
```

执行上述 SQL 语句,运行结果如图 3-8 所示。

mysql> SELECT 3<2, 2<=2, 'b'<'c', 'm'<NULL, NULL<NULL;				
3<2	2<=2	'b'<'c'	'm'<NULL	NULL<NULL
0	1	1	NULL	NULL
1 row in set (0.00 sec)				

图 3-8 小于运算符和小于或等于运算符的应用

说明:

(1) <=运算符具有左结合性。

(2) <运算符和<=运算符都不能用于判断空值(NULL),如果使用,结果都为 NULL。

5) 大于运算符和大于或等于运算符

大于运算符(>)和大于或等于运算符(>=)用来判断左侧的操作数是否大于、大于或等于右侧的操作数,如果大于、大于或等于,返回值为 1,否则返回值为 0。

说明:

(1) >和>=运算符与<和<=运算符的操作相同。

(2) >和>=运算符不能用于判断空值(NULL)。

2. 非符号类型的比较运算符

MySQL 中非符号类型的比较运算符如表 3-3 所示。

表 3-3 非符号类型的比较运算符

运 算 符	名 称	作 用
IS NULL(或 ISNULL)	为空运算符	判断值、字符串或表达式是否为空
IS NOT NULL	不为空运算符	判断值、字符串或表达式是否不为空
LEAST	最小值运算符	在多个值中返回最小值
GREATEST	最大值运算符	在多个值中返回最大值
BETWEEN AND	两值之间运算符	判断一个值是否在两个值之间
IN	属于运算符	判断一个值是否为列表中的任意值
NOT IN	不属于运算符	判断一个值是否不是列表中的任意值
LIKE	模糊匹配运算符	判断一个值是否符合模糊匹配规则
REGEXP	正则表达式运算符	判断一个值是否符合正则表达式的规则

1) 为空运算符和不为空运算符

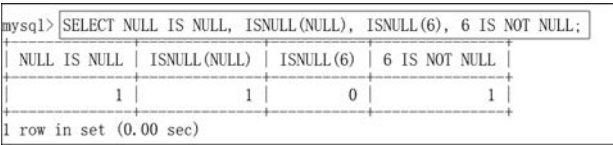
为空运算符(IS NULL 或 ISNULL)用于判断一个值是否为 NULL,如果为 NULL,返回 1,否则返回 0;不为空运算符(IS NOT NULL)用于判断一个值是否不为 NULL,如果不为 NULL,返回 1,否则返回 0。

【案例 3-6】 为空运算符和不为空运算符的应用。

在 MySQL 中,执行为空运算符和不为空运算符的运算的 SQL 语句如下。

```
SELECT NULL IS NULL, ISNULL(NULL), ISNULL(6), 6 IS NOT NULL;
```

执行上述 SQL 语句,运行结果如图 3-9 所示。



mysql>	SELECT NULL IS NULL, ISNULL(NULL), ISNULL(6), 6 IS NOT NULL;
	NULL IS NULL ISNULL(NULL) ISNULL(6) 6 IS NOT NULL
	1 1 0 1
	1 row in set (0.00 sec)

图 3-9 为空运算符和不为空运算符的应用

说明:

(1) NULL IS NULL 的返回值为 1。

(2) ISNULL(表达式)函数求表达式的值是否为 NULL,当表达式的值为 NULL 时,返回值为 1;当表达式的值不为 NULL 时,返回值为 0。

(3) IS NULL 和 ISNULL 的作用相同,ISNULL 和 IS NOT NULL 的返回值正好相反。

(4) IS NULL 运算符和 ISNULL()函数的格式不同,ISNULL()函数将表达式作为参数,而 IS NULL 运算符将表达式放在其左侧。

(5) IS NULL 运算符的效率比 ISNULL() 函数高,因为它不需要扫描,只搜索符合条件的记录,通常比扫描更快。

2) 最小值运算符

最小值运算符(LEAST)在有两个或多个参数比较的情况下返回最小值。基本的语法格式如下。

```
LEAST(值 1, 值 2, ..., 值 n)    # 值 n 表明参数列表中有 n 个值
```

【案例 3-7】 最小值运算符的应用。

在 MySQL 中执行最小值运算符的运算的 SQL 语句如下。

```
SELECT LEAST(2,5,1.3), LEAST('q','w','e'), LEAST(3,NULL,7);
```

执行上述 SQL 语句,运行结果如图 3-10 所示。

mysql> SELECT LEAST(2,5,1.3), LEAST('q','w','e'), LEAST(3,NULL,7);		
LEAST(2,5,1.3)	LEAST('q','w','e')	LEAST(3,NULL,7)
1.3	e	NULL
1 row in set (0.00 sec)		

图 3-10 最小值运算符的应用

说明:

(1) 当参数列表中的值是整数(浮点数)时,LEAST 运算符将返回其中最小的整数(浮点数)。

(2) 当参数列表中的值为字符串时,返回字母表中顺序最靠前的字符。

(3) 当参数列表中的值有 NULL 时,不能判断大小,其返回值为 NULL。

3) 最大值运算符

最大值运算符(GREATEST)在有两个或多个参数比较的情况下返回最大值。基本的语法格式如下。

```
GREATEST(值 1, 值 2, ..., 值 n)    # 值 n 表示参数列表中有 n 个值
```

说明:

(1) 最大值运算符和最小值运算符的操作相同。

(2) 当参数列表中的值是整数(浮点数)时,GREATEST 运算符将返回其中最大的整数(浮点数)。

(3) 当参数列表中的值为字符串时,返回字母表中顺序最靠后的字符。

(4) 当参数列表中的值有 NULL 时,不能判断大小,其返回值为 NULL。

4) 两值之间运算符

两值之间运算符(BETWEEN AND)用于判断一个值(c)是否在两个值(a 和 b)之间,当 $c \geq a$, 并且 $c \leq b$ 时,返回 1,否则返回 0。基本的语法格式如下。

```
SELECT d FROM TABLE WHERE c BETWEEN a AND b;
```

【案例 3-8】 两值之间运算符的应用。

在 MySQL 中,执行两值之间运算符的运算的 SQL 语句如下。

```
SELECT 'c' BETWEEN 'a' AND 'b', 1 BETWEEN 3 AND 4, 7 BETWEEN 5 AND 8;
```

执行上述 SQL 语句,运行结果如图 3-11 所示。

mysql> SELECT 'c' BETWEEN 'a' AND 'b', 1 BETWEEN 3 AND 4, 7 BETWEEN 5 AND 8;		
'c' BETWEEN 'a' AND 'b'	1 BETWEEN 3 AND 4	7 BETWEEN 5 AND 8
0	0	1
1 row in set (0.00 sec)		

图 3-11 两值之间运算符的应用

说明：当比较字符串类型时,按照字母表的顺序进行比较。

5) 属于运算符和不属于运算符

属于运算符(IN)用于判断给定的值是否为列表中的一个值,如果是,返回 1,否则返回 0;不属于运算符(NOT IN)用于判断给定的值是否不是列表中的一个值,如果不是,返回 1,否则返回 0。

【案例 3-9】 属于运算符和不属于运算符的应用。

在 MySQL 中,执行属于运算符和不属于运算符的运算的 SQL 语句如下。

```
SELECT 1 IN (2,3,4), 1 NOT IN (2,3,4), NULL NOT IN (2,3,4);
```

执行上述 SQL 语句,运行结果如图 3-12 所示。

mysql> SELECT 1 IN (2,3,4), 1 NOT IN (2,3,4), NULL NOT IN (2,3,4);		
1 IN (2,3,4)	1 NOT IN (2,3,4)	NULL NOT IN (2,3,4)
0	1	NULL
1 row in set (0.00 sec)		

图 3-12 属于运算符和不属于运算符的应用

说明：

- (1) IN 运算符和 NOT IN 运算符的运行结果相反。
- (2) 当在列表中找到给定的值相匹配时,或者给定的值是 NULL,又或者列表中存在 NULL 值,结果均为 NULL。

6) 模糊匹配运算符

模糊匹配运算符(LIKE)主要用来匹配字符串,通常用于模糊匹配,如果满足条件,返回 1,否则返回 0;如果给定的值或匹配条件为 NULL,则返回结果为 NULL。

LIKE 运算符可以使用以下两个通配符。

- (1) %：可以匹配任何数目的字符以及零字符。
- (2) _：只能匹配一个字符。

【案例 3-10】 模糊匹配运算符的应用。

在 MySQL 中,执行模糊匹配运算符的运算的 SQL 语句如下。

```
SELECT 13579 LIKE '123%', 13579 LIKE '%35%', 2468 LIKE '24__';
```

执行上述 SQL 语句,运行结果如图 3-13 所示。

mysql> SELECT 13579 LIKE '123%', 13579 LIKE '%35%', 2468 LIKE '24__';		
13579 LIKE '123%'	13579 LIKE '%35%'	2468 LIKE '24__'
0	1	1
1 row in set (0.00 sec)		

图 3-13 模糊匹配运算符的应用

说明：_通配符匹配单个任意字符。

7) 正则表达式运算符

正则表达式运算符(REGEXP)用来匹配字符串,正则表达式(Regular Expression,通常简称为 REGEXP 或 RE)在搜索字符串时非常强大。正则表达式运算符的基本语法格式如下,若 expr 中包含相匹配的字符串,返回 1,否则返回 0。

```
SELECT expr REGEXP 匹配条件;
```

【案例 3-11】 正则表达式运算符的应用。

在 MySQL 中,执行正则表达式运算符的运算的 SQL 语句如下。

```
SELECT 'happy' REGEXP '^h', 'happy' REGEXP 'p$', 'happy' REGEXP 'py';
```

执行上述 SQL 语句,运行结果如图 3-14 所示。

```
mysql> SELECT 'happy' REGEXP '^h', 'happy' REGEXP 'p$', 'happy' REGEXP 'py';
```

'happy' REGEXP '^h'	'happy' REGEXP 'p\$'	'happy' REGEXP 'py'
1	0	1

```
1 row in set (0.00 sec)
```

图 3-14 正则表达式运算符的应用

说明:在 MySQL 中使用 REGEXP 关键字指定正则表达式的字符匹配模式,以下是 REGEXP 运算符常用的几个通配符。

- (1) ^: 匹配以该字符后面的字符开头的字符串。
- (2) \$: 匹配以该字符前面的字符结尾的字符串。
- (3) .: 匹配任意单字符。
- (4) *: 匹配 0 个或多个在它前面的字符。
- (5) [...]: 匹配在方括号内的任何字符。例如,[xyz] 匹配 x、y 或 z;[a-z] 匹配任何字母;[0-9] 匹配任何数字。

3.3.3 逻辑运算符

逻辑运算符主要用于判断表达式的真假,在 MySQL 中,逻辑运算符的返回结果为 1、0 或 NULL。在 MySQL 中支持 4 种逻辑运算符,如表 3-4 所示。

表 3-4 逻辑运算符

运 算 符	作 用	运 算 符	作 用
NOT 或 !	逻辑非	OR 或	逻辑或
AND 或 &.&	逻辑与	XOR	逻辑异或

1. 逻辑非运算符

逻辑非运算符(NOT 或!)当给定的值为 0 时返回 1;当给定的值为非 0 值时返回 0。但有一点除外,那就是当给定的值为 NULL 时返回 NULL。

【案例 3-12】 逻辑非运算符的应用。

在 MySQL 中,执行逻辑非运算符的运算的 SQL 语句如下。

```
SELECT NOT 0, NOT 1, NOT !1, NOT 1 + 2, ! 1 + 2, NOT NULL;
```

执行上述 SQL 语句,运行结果如图 3-15 所示。

mysql> SELECT NOT 0, NOT 1, NOT !1, NOT 1+2, ! 1+2, NOT NULL;					
NOT 0	NOT 1	NOT !1	NOT 1+2	! 1+2	NOT NULL
1	0	1	0	2	NULL
1 row in set, 2 warnings (0.00 sec)					

图 3-15 逻辑非运算符的应用

说明：

- (1) NOT 和!运算符的返回值相同。
- (2) NOT 1+2 和! 1+2 的运算结果不同,原因是 NOT 和!运算符的优先级不同,NOT 运算符的优先级低于+运算符,而!运算符的优先级高于+运算符,所以导致两者结果不同。

2. 逻辑与运算符

逻辑与运算符(AND 或 &&)当给定的所有值均为非零值且不为 NULL 时返回 1;当给定的一个值或多个值为 0 时返回 0,否则返回 NULL。

【案例 3-13】 逻辑与运算符的应用。

在 MySQL 中,执行逻辑与运算符的运算的 SQL 语句如下。

```
SELECT 0 AND 1, 0 AND NULL, 1 AND NULL, -1 AND 1, -1 && 1;
```

执行上述 SQL 语句,运行结果如图 3-16 所示。

mysql> SELECT 0 AND 1, 0 AND NULL, 1 AND NULL, -1 AND 1, -1 && 1;				
0 AND 1	0 AND NULL	1 AND NULL	-1 AND 1	-1 && 1
0	0	NULL	1	1
1 row in set, 1 warning (0.00 sec)				

图 3-16 逻辑与运算符的应用

说明：

- (1) AND 和 && 运算符的作用相同。
- (2) 在操作数中若有一个为 NULL,结果就返回 NULL。

3. 逻辑或运算符

逻辑或运算符(OR 或 ||)当给定的两个值都不为 NULL 且任何一个值为非零值时返回 1,否则返回 0;当一个值为 NULL 且另一个值为非零值时返回 1,否则返回 NULL;当两个值都为 NULL 时返回 NULL。

【案例 3-14】 逻辑或运算符的应用。

在 MySQL 中,执行逻辑或运算符的运算的 SQL 语句如下。

```
SELECT 0 OR 1, 0 || 1, 0 OR 0, 3 OR NULL, NULL || NULL;
```

执行上述 SQL 语句,运行结果如图 3-17 所示。

mysql> SELECT 0 OR 1, 0 1, 0 OR 0, 3 OR NULL, NULL NULL;				
0 OR 1	0 1	0 OR 0	3 OR NULL	NULL NULL
1	1	0	1	NULL
1 row in set, 2 warnings (0.00 sec)				

图 3-17 逻辑或运算符的应用

说明：

- (1) OR 和 || 运算符的作用相同。
- (2) 当两个值中有一个 0 时,返回 1;当两个值都为 0 时,返回 0;当两个值都为 NULL

时,返回 NULL。

4. 逻辑异或运算符

逻辑异或运算符(XOR)当给定的值中有一个为 NULL 时返回 NULL;如果是非 NULL 的操作数,两个的逻辑值相异返回 1,否则返回 0。

说明: 若一个操作数为 0,另一个不为 0,则结果返回 1。

3.3.4 位运算符

位运算符是先将操作数转换为二进制数,然后进行位运算,再将结果从二进制数转换为十进制数。MySQL 中支持的位运算符如表 3-5 所示。

表 3-5 位运算符

运 算 符	作 用	运 算 符	作 用
&	按位与(位 AND)	~	按位取反
	按位或(位 OR)	>>	按位右移
^	按位异或(位 XOR)	<<	按位左移

1. 按位与运算符

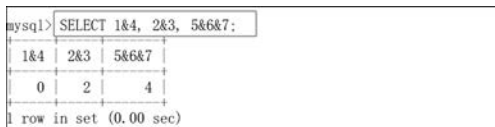
按位与运算符(&)将给定值对应的二进制数逐位进行逻辑与运算。先将十进制的操作数转换为二进制数,然后按对应的二进制数逐位进行逻辑与运算。当给定值对应的二进制位的数值都为 1 时,该位的运算结果为 1,否则该位的运算结果为 0。

【案例 3-15】 按位与运算符的应用。

在 MySQL 中,执行按位与运算符的运算的 SQL 语句如下。

```
SELECT 1&4, 2&3, 5&6&7;
```

执行上述 SQL 语句,运行结果如图 3-18 所示。



1&4	2&3	5&6&7
0	2	4

1 row in set (0.00 sec)

图 3-18 按位与运算符的应用

说明:

(1) 1 的二进制数为 0001,4 的二进制数为 0100,所以 1&4 的结果为 0000,对应的十进制数为 0。

(2) 2 的二进制数为 0010,3 的二进制数为 0011,所以 2&3 的结果为 0010,对应的十进制数为 2。

(3) 5 的二进制数为 0101,6 的二进制数为 0110,7 的二进制数为 0111,所以 5&6&7 的结果为 0100,对应的十进制数为 4。

2. 按位或运算符

按位或运算符(|)将给定值对应的二进制数逐位进行逻辑或运算。先将十进制的操作数转换为二进制数,然后逐位进行逻辑或运算。当给定值对应的二进制位的数值有一个或两个为 1 时,该位的运算结果为 1,否则该位的运算结果为 0。例如,计算 2|3,因为 2 的二进制数为 0010,3 的二进制数为 0011,所以 2|3 的结果为 0011,对应的十进制数为 3,即运算结果为 3。

3. 按位异或运算符

按位异或运算符(^)将给定值对应的二进制数逐位进行逻辑异或运算。当给定值对应的

二进制位的数值不同时,该位的运算结果为 1,否则该位的运算结果为 0。例如,计算 2^3 ,因为 2 的二进制数为 0010,3 的二进制数为 0011,所以 2^3 的结果为 0001,对应的十进制数为 1,即运算结果为 1。

4. 按位取反运算符

按位取反运算符($\sim$)将给定值的二进制数逐位按对应的补码进行反转,也就是进行取反操作,即1取反后变为0,0取反后变为1。

【案例 3-16】 按位取反运算符的应用。

在 MySQL 中,执行按位取反运算符的运算的 SQL 语句如下。

SELECT ~ 1 ;

执行上述 SQL 语句,运行结果如图 3-19 所示。

```
mysql> SELECT ~1;
```

~1
18446744073709551614

1 row in set (0.00 sec)

图 3-19 按位取反运算符的应用

在 MySQL 中常数是 8 字节,每字节是 8 位,则一个常数就是 64 位。常数 1 转换为二进制数后是由 64 位构成的,最后一位是 1,前面的 63 位是 0。在进行按位取反后,前 63 位的值是 1,最后一位是 0,最终将二进制数转换为十进制数 18446744073709551614。

用户可以使用 BIN() 函数查看常数 1 取反后的结果, BIN() 函数的作用是将一个十进制数转换为二进制数。

【案例 3-17】 BIN()函数的使用。

在 MySQL 中使用 BIN()函数的 SQL 语句如下。

```
SELECT BIN(  $\sim 1$  );
```

执行上述 SQL 语句,运行结果如图 3-20 所示。

[illegible]

图 3-20 BIN()函数的使用

常数 1 转换为二进制数后,最右边一位为 1,其他位均为 0。在进行按位取反操作后,除了最右边一位为 0 以外,其他位均为 1。

5. 按位右移运算符

按位右移运算符(>>)将给定值的二进制数的所有位右移指定的位数。在右移指定的位数之后,右边低位的数值被移出并丢弃,左边高位空出的位置用 0 补齐。例如, $m \gg n$ 表示将 m 的二进制数向右移 n 位,左边补 n 个 0。

【案例 3-18】 按位右移运算符的应用。

在 MySQL 中,执行按位右移运算符的运算的 SQL 语句如下。

```
SELECT 3 >> 1, 24 >> 2;
```

执行上述 SQL 语句,运行结果如图 3-21 所示。

mysql>	SELECT 3>>1, 24>>2;
3>>1	24>>2
1	6
1 row in set (0.00 sec)	

图 3-21 按位右移运算符的运行结果

说明：

(1) 3 的二进制数为 0000 0011, 右移一位为 0000 0001, 对应的十进制数为 1。

(2) 24 的二进制数为 0001 1000, 右移两位为 0000 0110, 对应的十进制数为 6。

6. 按位左移运算符

按位左移运算符(<<)将给定值的二进制数的所有位左移指定的位数。在左移指定的位数之后, 左边高位的数值被移出并丢弃, 右边低位空出的位置用 0 补齐。例如, $m \ll n$ 表示将 m 的二进制数向左移 n 位, 右边补 n 个 0。

说明: <<运算符与>>运算符的操作相同。

3.3.5 运算符的优先级

在 MySQL 中, 优先级高的运算符先进行计算; 赋值运算符的优先级最低, 使用括号括起来的表达式的优先级最高。MySQL 中运算符的优先级如表 3-6 所示。

表 3-6 MySQL 中运算符的优先级

优 先 级	运 算 符
1(低)	: =、=(赋值)
2	、OR、XOR
3	&.&、AND
4	NOT
5	BETWEEN、CASE、WHEN、THEN、ELSE
6	= (比较运算符)、<=>、>=、>、<=、<、<>、!=、IS、LIKE、REGEXP、IN
7	
8	&.
9	<<、>>
10	-、+
11	*、/、DIV、%、MOD
12	^
13	-(负号)、~(按位取反)
14	!
15(高)	()

说明: 由表 3-6 可以看出, 运算符的优先级与正常的运算符的规则很相似, 不同运算符的优先级是不同的。在一般情况下, 级别高的运算符优先进行计算, 如果级别相同, MySQL 按表达式的顺序从左到右依次计算。

🔑 课业任务

课业任务 3-1 算术运算符的应用

【能力测试点】

算术运算符的应用。

扫一扫



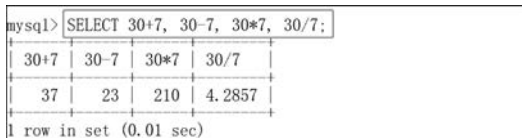
视频讲解

【任务实现步骤】

任务需求：在 MySQL 中使用算术运算符进行运算，计算 30 和 7 的和、差、积、商。
按要求进行程序的编写，SQL 语句如下。

```
SELECT 30 + 7, 30 - 7, 30 * 7, 30 / 7;
```

执行上述 SQL 语句，结果如图 3-22 所示。



```
mysql> SELECT 30+7, 30-7, 30*7, 30/7;
```

30+7	30-7	30*7	30/7
37	23	210	4.2857

1 row in set (0.01 sec)

图 3-22 课业任务 3-1 程序运行结果

课业任务 3-2 比较运算符的应用

【能力测试点】

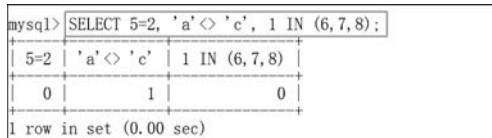
比较运算符的应用。

【任务实现步骤】

任务需求：在 MySQL 中使用比较运算符进行运算，计算 $5=2$ 、 $'a'<>'c'$ 、 $1 \text{ IN } (6,7,8)$ 。
按要求进行程序的编写，SQL 语句如下。

```
SELECT 5 = 2, 'a' <> 'c', 1 IN (6,7,8);
```

执行上述 SQL 语句，结果如图 3-23 所示。



```
mysql> SELECT 5=2, 'a'<>'c', 1 IN (6,7,8);
```

5=2	'a'<>'c'	1 IN (6,7,8)
0	1	0

1 row in set (0.00 sec)

图 3-23 课业任务 3-2 程序运行结果

课业任务 3-3 逻辑运算符的应用

【能力测试点】

逻辑运算符的应用。

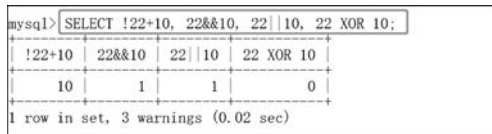
【任务实现步骤】

任务需求：在 MySQL 中使用逻辑运算符进行运算，计算 $!22+10$ 、 $22\&10$ 、 $22||10$ 、 $22 \text{ XOR } 10$ 。

按要求进行程序的编写，SQL 语句如下。

```
SELECT !22 + 10, 22 & 10, 22 || 10, 22 XOR 10;
```

执行上述 SQL 语句，结果如图 3-24 所示。



```
mysql> SELECT !22+10, 22&10, 22||10, 22 XOR 10;
```

!22+10	22&10	22 10	22 XOR 10
10	1	1	0

1 row in set, 3 warnings (0.02 sec)

图 3-24 课业任务 3-3 程序运行结果

课业任务 3-4 位运算符的应用

【能力测试点】

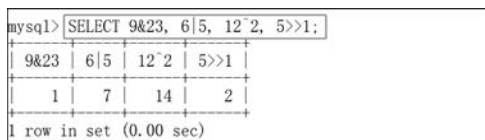
位运算符的应用。

【任务实现步骤】

任务需求：在 MySQL 中使用位运算符进行运算，计算 $9 \& 23$ 、 $6 | 5$ 、 $12 \wedge 2$ 、 $5 \gg 1$ 。
按要求进行程序的编写，SQL 语句如下。

```
SELECT 9&23, 6|5, 12^2, 5>>1;
```

执行上述 SQL 语句，结果如图 3-25 所示。



9&23	6 5	12^2	5>>1
1	7	14	2

1 row in set (0.00 sec)

图 3-25 课业任务 3-4 程序运行结果

扫一扫



视频讲解

课业任务 3-5 运算符的综合应用

【能力测试点】

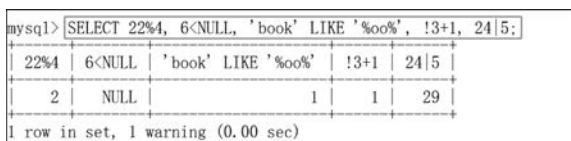
算术运算符、比较运算符、逻辑运算符和位运算符的综合应用。

【任务实现步骤】

任务需求：在 MySQL 中使用多种运算符进行运算，计算 $22 \% 4$ 、 $6 < \text{NULL}$ 、 $'\text{book}' \text{ LIKE } '\% \text{oo} \%'$ 、 $!3 + 1$ 、 $24 | 5$ 。
按要求进行程序的编写，SQL 语句如下。

```
SELECT 22 % 4, 6 < NULL, 'book' LIKE '%oo%', !3 + 1, 24 | 5;
```

执行上述 SQL 语句，结果如图 3-26 所示。



22%4	6<NULL	'book' LIKE '%oo%'	!3+1	24 5
2	NULL	1	1	29

1 row in set, 1 warning (0.00 sec)

图 3-26 课业任务 3-5 程序运行结果

扫一扫



视频讲解

课业任务 3-6 使用 MySQL Workbench 工具进行比较运算

【能力测试点】

使用 MySQL Workbench 工具进行比较运算。

【任务实现步骤】

任务需求：使用 MySQL Workbench 工具进行比较运算，计算 $'a' <=> 'b'$ 、 $5 < 32$ 、 $\text{LEAST}(4, 56, 7, 3)$ 。
(1) 按要求进行程序的编写，SQL 语句如下。

```
SELECT 'a' <=> 'b', 5 < 32, LEAST(4, 56, 7, 3);
```

(2) 启动 MySQL Workbench，首先单击工具栏中的“新建查询”按钮，然后按要求在 SQL

扫一扫



视频讲解

脚本编辑窗口中编写 SQL 语句,最后单击“运行”按钮或按 Ctrl+Enter 快捷键,如图 3-27 所示。



图 3-27 MySQL Workbench 工具的新建查询操作

(3) 在 MySQL Workbench 工具中执行上述 SQL 语句,结果如图 3-28 所示。

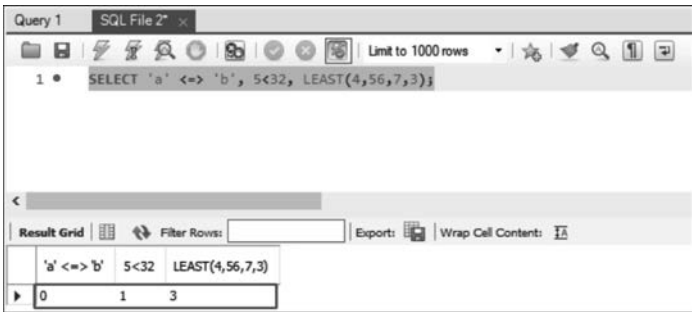


图 3-28 课业任务 3-6 程序运行结果



扫一扫

视频讲解

课业任务 3-7 使用 Navicat Premium 工具进行综合运算

【能力测试点】

使用 Navicat Premium 工具进行综合运算。

【任务实现步骤】

任务需求: 使用 Navicat Premium 工具进行综合运算,计算 434 BETWEEN 1 AND 500、-4&8&0。

按要求进行程序的编写,SQL 语句如下。

```
SELECT 434 BETWEEN 1 AND 500, -4&8&0;
```

启动 Navicat Premium,首先单击工具栏中的“查询”按钮,然后单击“新建查询”按钮,如图 3-29 所示,最后按要求在 SQL 脚本编辑窗口中编写 SQL 语句,如图 3-30 所示。

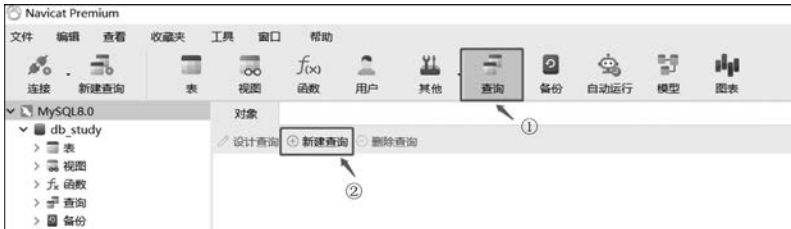


图 3-29 Navicat Premium 工具的新建查询操作

在 Navicat Premium 工具中执行上述 SQL 语句,单击“运行已选择的”按钮,执行成功后,结果如图 3-31 所示。



图 3-30 SQL 语句的编写操作

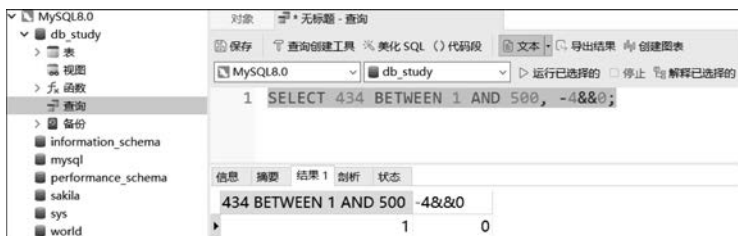


图 3-31 课业任务 3-7 程序运行结果

❓ 常见错误及解决方案

错误 3-1 关键字分行书写时报错

【问题描述】

在进行算术运算符的操作时,将 SELECT 关键字分行书写,出现如图 3-32 所示的错误提示,意为“SQL 语法有错误:检查与您的 MySQL 服务器版本对应的手册,了解需要在第 1 行 ‘SELECT 6+0’ 附近使用正确的语法”。

【解决方案】

根据 SQL 的基本规则,关键字不能分行书写,所以正确的书写方式如图 3-33 所示。

```
mysql> SE
-> LECT 6+0;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'SE
LECT 6+0' at line 1
```

图 3-32 关键字分行书写时报错

```
mysql> SELECT 6+0;
+-----+
| 6+0 |
+-----+
| 6   |
+-----+
1 row in set (0.00 sec)
```

图 3-33 SELECT 关键字的正确书写方式

错误 3-2 采用中文状态下的半角输入方式时报错

【问题描述】

在进行等于运算符的操作时,采用了中文状态下的半角输入方式,出现错误提示,如图 3-34 所示。

【解决方案】

根据 SQL 的基本规则,必须使用英文状态下的半角输入方式,正确的输入方式如图 3-35 所示。

```
mysql> SELECT 3=1 3=3;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near '3=3'
at line 1
mysql>
```

图 3-34 采用了中文状态下的半角输入方式时报错

```
mysql> SELECT 3=1 3=3;
+-----+-----+
| 3=1 | 3=3 |
+-----+-----+
| 0   | 1   |
+-----+-----+
1 row in set (0.00 sec)
```

图 3-35 正确的英文状态下的半角输入方式

扫一扫



自测题

习题

1. 选择题

- (1) “SELECT '2' = '2xy';”语句的执行结果为()。
A. 0 B. 1 C. NULL D. False
- (2) 下列选项中是 MySQL 的逻辑运算符的是()。
A. IN B. & C. AND D. <>
- (3) “SELECT (13=13) AND (4>3);”语句的执行结果为()。
A. 0 B. 1 C. NULL D. False
- (4) 在 MySQL 中,下列关于通配符的说法正确的是()。
A. _可以匹配多个字符,%可以匹配单个字符
B. _可以匹配单个字符,%可以匹配多个字符
C. _和%可以匹配多个字符
D. 以上说法都不正确
- (5) 下列选项中不属于 DML 操作的是()。
A. UPDATE B. INSERT C. DELETE D. CREATE
- (6) 在 MySQL 中,下列关于 BETWEEN AND 运算符的说法错误的是()。
A. 该运算符用在 WHERE 子句中
B. BETWEEN AND 运算符选取介于两个值之间的数据,值可以是数字和日期类型
C. 取值范围包括边界值
D. 以上说法都不正确

2. 填空题

- (1) SQL 在功能上主要分为 3 类,即_____、_____和_____。
- (2) “SELECT 10/4;”语句的运行结果为_____。
- (3) 用 SELECT 语句进行模糊查询,在条件值中应该使用_____或%通配符配合查询。
- (4) 在正则表达式中,匹配任意一个字符的符号是_____。
- (5) 在 SQL 中,数据操作语句包括 SELECT、INSERT、UPDATE、DELETE,其中最重要、使用最频繁的语句是_____。

3. 判断题

- (1) b BETWEEN a AND c 等同于 b>a && b<c。 ()
- (2) NULL 和 Null 都代表空值。 ()
- (3) != 和<>都表示不等于。 ()
- (4) 在子查询中使用 IN 运算符和使用=ANY 的效果是不一样的。 ()
- (5) 判断某个字段是否为空值可以用 IS NULL。 ()