项目3 MySQL 8数据库语言与编程

3.1 项目描述

当面对一个陌生的数据库时，常需要一种方式与它进行交互，以完成用户各种需求，这时，就要用到 SQL 语言。本项目旨在通过实际操作，让读者学会 MySQL 8 中 SQL 语言结构以及数据库与表基本语句的使用，具体任务包括以下三方面。

（1）学会 MySQL 8 中常量、变量、运算符、函数的使用。

（2）学会使用 SQL 语句创建数据库和表。

（3）学会使用 SQL 语句对数据表进行插入、修改和删除操作。

3.2 任务解析

编写程序需要写出正确的代码，这就需要了解这种编程语言的语法结构，要详细掌握构造命令语句的基本元素，一个操作所需的代码越多，就越容易出现 Bug，也越难发现它们，任何一个小的语法错误都有可能导致 Bug。因此，本项目详细介绍了 MySQL 中数据类型、常量、变量、运算符、函数、表达式的使用。MySQL 安装完成以后，首先需要创建数据库，这是使用 MySQL 各种功能的前提，数据表是数据库中最基本也是最重要的操作对象，是数据存储的基本单位，对数据库中的表操作才能完成数据的插入、删除和修改。因此，本项目将详细介绍 MySQL 语言中的数据库创建、查看、删除等语句，数据表的创建、修改、删除等语句，表记录的插入、更新、删除语句的使用。

3.3 相关知识与实例

3.3.1 数据库语言概述

1. 数据库语言及其分类

数据库语言以记录集合作为操作对象，所有 SQL 语句接受集合作为输入，返回集合作为输出，这种集合特性允许一条 SQL 语句的输出作为另一条 SQL 语句的输入。因此，SQL 语句可以嵌套，这使它具有极大的灵活性和强大的功能。在多数情况下，在其他语言中需要一大段程序实现的功能，在 SQL 中只需要一条 SQL 语句就可以达到目的，这也意味着用 SQL 语言可以写出非常复杂的语句。

一般说来，数据库语言可分为数据定义语言(Data Definition Language，DDL)、数据操作语言(Data Manipulation Language，DML)、数据查询语言(Data Query Language，DQL)、数据控制语言(Data Control Language，DCL)、事务控制语言(Transaction Control Language，TCL)。

DDL是数据库高级操作，主要用于数据库项目的首次上线，由DBA统一执行初始化脚本，主要包括CREATE、DROP、ALTER等语句，用于创建、删除、修改数据库、创建、删除、修改数据表、创建、删除、修改视图和索引等操作。从安全角度考虑，一般开发人员没有权限执行上述这些操作。

DML用于对数据库的普通操作，主要包括INSERT、UPDATE、DELETE等语句，满足用户根据不同的业务逻辑执行增加、修改、删除、查询等操作。另外，数据库的运维人员也常使用DML对数据库进行查询操作。

DQL的基本结构是由SELECT子句、FROM子句、WHERE子句组成的查询块。

DCL用于授予或回收访问数据库的某种特权，并控制数据库操纵事务发生的时间及效果，对数据库实行监视等。如在数据库的插入、删除和修改操作时，只有当事务提交到数据库时才算完成。在事务提交前，只有操作数据库的人才能有权看到所做的事情，别人只有在提交完成后才可以看到。

TCL主要包括COMMIT、ROLLBACK等语句。

2. SQL简介

20世纪80年代初，美国国家标准局(ANSI)开始着手制定SQL标准，最早的ANSI标准于1986年完成，叫作SQL-86。随后，SQL标准几经修改，更趋完善。由于SQL语言的标准化，所以大多数关系数据库系统都支持SQL语言，它已经发展成为多种平台进行交互操作的底层会话语言。

与其他计算机语言相似，MySQL数据库语言包含若干MySQL语句、常量、变量、函数、运算符和表达式。有关MySQL数据库语言的说明如下。

- MySQL语句以分号结束，并且SQL处理器忽略空格、制表符和回车符。
- 箭头(→)代表MySQL语句没有输入完。
- 取消MySQL语句使用\C。
- Windows操作系统下MySQL语句关键字和函数名不区分大小写，但Linux操作系统下区分。
- 使用函数时，函数名与其后的括号之间不能有空格。

3.3.2　MySQL数据库操作


视频讲解

1. 创建数据库

命令格式如下：

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
[Create_Specification [, Create_Specification] …];
```

其中，Create_Specification的格式如下：

```
[DEFAULT] CHARACTER SET charset_name | [DEFAULT] COLLATE collation_name
```

提醒：语句中“[ ]”内的内容为可选项，其他参数的含义如下。

- db_name：数据库名称（不区分大小写）。
- IF NOT EXISTS：创建数据库前先判断，只有该数据库目前尚不存在时才执行CREATE DATABASE操作。如果数据库已经存在，出现错误信息提示。
- DEFAULT：指定默认值。
- CHARACTER SET：指定数据库字符集，charset_name为字符集名称。
- COLLATE：指定字符集的校对规则，collation_name为校对规则名称。

【例3.1】 创建数据库jwgl。

```
mysql > CREATE DATABASE jwgl;
Query OK, 1 row affected
```

MySQL不允许两个数据库使用相同的名字，使用IF NOT EXISTS从句可以不显示错误信息（假设test已经存在），例如：

```
mysql > CREATE DATABASE IF NOT EXISTS test;
Query OK, 1 row affected
```

2. 查看数据库

一个MySQL实例可以同时承载多个数据库，查看当前MySQL实例上所有数据库的语句格式如下：

```
SHOW DATABASES [LIKE Wild];
```

其中：LIKE Wild中的Wild字符串可使用SQL的“%”和“_”通配符。

【例3.2】 查看当前MySQL实例上所有数据库。

```
mysql > SHOW DATABASES;
```

3. 选择数据库

在进行数据库操作前，必须选定被操作的数据库，可用USE命令完成，其格式为：USE db_name;

提醒：该语句也可以用来从一个数据库“跳转”到另一个数据库，用CREATE DATABASE语句创建的数据库不会自动成为当前数据库，需要用USE语句指定当前数据库。

【例3.3】 指定当前数据库为jwgl。

```
mysql > USE jwgl;
Database changed
```

4. 删除数据库

删除数据库是将已存在的数据库从磁盘上清除，该数据库的所有表（包括其中的数据）也将永久删除，所以要慎用。格式如下：

```
DROP DATABASE [IF EXISTS] db_name;
```

其中，IF EXISTS 子句可避免在删除不存在的数据库时提示 MySQL 错误信息。

【例 3.4】 删除数据库 test。

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

5. 修改数据库

命令格式如下：

```
ALTER{DATABASE | SCHEMA} [db_name] alter_specification;
```

其中，alter_specification 的格式如下：

```
[DEFAULT] CHARACTER SET charset_name | [DEFAULT] COLLATE collation_name
```

说明：ALTER DATABASE 语句用于修改数据库的全局特性，这些特性储存在数据库目录中的 db.opt 文件中。用户必须具有对数据库进行修改的权限，才可以使用该命令。如果语句中数据库名称忽略，则修改当前数据库。

【例 3.5】 修改数据库 jwgl(假设 jwgl 已经创建)的默认字符集和校对规则。

```
mysql > ALTER DATABASE jwgl
  DEFAULT CHARACTER SET gb2312
  DEFAULT COLLATE gb2312_chinese_ci;
Query OK, 1 row affected
```

3.3.3　MySQL 数据表操作


视频讲解

1. 创建数据表

命令格式如下：

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table_name
       [([column_definition], … |[index_definition])]
     [table_option] [select_statement];
```

参数说明如下。

- TEMPORARY：用 CREATE 命令创建临时表(不加 TEMPORARY 创建的表称为持久表，持久表一旦创建将一直存在，多个用户或者多个应用程序可以同时使用持久表)，临时表的生命周期短，而且只能对创建它的用户可见，当断开与该数据库的连接时，MySQL 会自动删除临时表。
- IF NOT EXISTS：创建表前先判断，用以避免出现表已存在无法再新建的错误。
- table_name：要创建的表的表名。该表名必须符合标志符规则，如果表名中包含 MySQL 保留字，则必须用单引号括起来。
- column_definition：列定义，包括列名、数据类型，可能还包括一个空值声明和一个完整性约束。

- index_definition：表索引项定义，主要定义表的索引、主键、外键等，具体定义将在项目 5 中讨论。
- table_option：用于描述表的选项。
- select_statement：可以在 CREATE TABLE 语句的末尾添加一个 SELECT 语句，在一个表的基础上创建表。
- column_definition：用于列定义，其格式如下。

```
col_name type [NOT NULL | NULL] [DEFAULT default_value]
[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
[COMMENT 'string'] [reference_definition]
```

参数说明如下。

- col_name：表中列的名字。列名必须符合标志符规则，长度不能超过 64 个字符，而且在表中要唯一。如果有 MySQL 保留字必须用单引号括起来。
- type：列的数据类型，有的数据类型需要指明长度 n，并用圆括号括起来。
- AUTO_INCREMENT：设置自增属性，只有整型列才能设置此属性。当插入 NULL 值或 0 到一个 AUTO_INCREMENT 列中时，列被设置为 value+1，在这里 value 是此前表中该列的最大值。AUTO_INCREMENT 顺序从 1 开始。每个表只能有一个 AUTO_INCREMENT 列，并且它必须被索引。
- NOT NULL | NULL：指定该列是否为空值。如果不指定，则默认为 NULL。
- DEFAULT default_value：为列指定默认值，默认值必须为一个常数。其中，BLOB 和 TEXT 列不能被赋予默认值。如果没有为列指定默认值，MySQL 自动地分配一个值。如果列可以取 NULL 值，默认值就是 NULL。如果列被声明为 NOT NULL，默认值取决于列类型：对于没有声明 AUTO_INCREMENT 属性的数字类型，默认值是 0；对于一个 AUTO_INCREMENT 列，默认值是在顺序中的下一个值；对于除 TIMESTAMP 以外的日期和时间类型，默认值是该类型适当的零值；对于表中第一个 TIMESTAMP 列，默认值是当前的日期和时间。对于除 ENUM 外的字符串类型，默认值是空字符串；对于 ENUM，默认值是第一个枚举值。
- UNIQUE KEY | PRIMARY KEY：PRIMARY KEY 和 UNIQUE KEY 都表示字段中的值是唯一的。PRIMARY KEY 表示设置为主键，一个表只能定义一个主键，主键一定要为 NOT NULL。
- COMMENT 'string'：对于列的描述，string 是描述的内容。
- reference_definition：指定参照的表和列。
- 数据表归属于数据库。在创建数据表之前，应用语句 USE DATABASE 指定当前数据库，否则会出现 no database selected 错误提示。

【例 3.6】 假设已经创建了数据库 jwgl，在该数据库中创建学生情况表 student。

```
mysql > USE jwgl;
mysql > CREATE TABLE student
(Sno char(14) PRIMARY KEY,
```

```
Sname varchar(10) NOT NULL,
Ssex char(2) NOT NULL,
Sbirth date NOT NULL,
Smajor varchar(22) NOT NULL,
Sphone char(11),
Spwd varchar(6) NOT NULL,
memo varchar(45) NULL) ENGINE = InnoDB;
```

在例 3.6 中，每个字段都包含附加约束或修饰符，这些可以用来增加对所输入数据的约束。PRIMARY KEY 表示将“学号”字段(Sno)定义为主键。ENGINE＝InnoDB 表示采用的存储引擎是 InnoDB，InnoDB 是 MySQL 在 Windows 平台默认的存储引擎，所以 ENGINE＝InnoDB 可以省略。

2. 修改数据表

用 ALTER TABLE 语句更改数据表的结构，如增加或删减列，创建或取消索引，更改原字段的类型，重新命名列或表，更改表的评注和表的类型。

命令格式如下：

```
ALTER [IGNORE] TABLE tbl_name alter_specification;
```

参数说明如下。

- tbl_name：表名。
- IGNORE：是 MySQL 相对于标准 SQL 语言的扩展。若在修改后的新表中存在重复关键字，如果没有指定 IGNORE，当重复关键字错误发生时操作失败。如果指定了 IGNORE，则对于有重复关键字的行只使用第一行，其他有冲突的行被删除。
- alter_specification：用于指定修改的内容，其格式如下。

```
ADD [COLUMN] column_definition [FIRST | AFTER col_name ]
| ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
| CHANGE [COLUMN] old_col_name column_definition
| MODIFY [COLUMN] column_definition [FIRST | AFTER col_name]
| DROP [COLUMN] col_name
| RENAME [TO] new_tbl_name
| ORDER BY col_name
| CONVERT TO CHARACTER SET charset_name [COLLATE collation_name]
| [DEFAULT] CHARACTER SET charset_name [COLLATE collation_name]
| table_options
```

其中的参数含义如下。

- ADD [COLUMN]子句：向表中增加新列。
- column_definition：定义列的数据类型和属性，具体内容在 CREATE TABLE 的语法中已做说明。
- col_name：指定的列名。
- FIRST | AFTER col_name：表示在某列的前或后添加新列，不指定则添加到最后。
- ALTER [COLUMN]子句：修改表中指定列的默认值。

- CHANGE [COLUMN]子句：修改列的名称。重命名时，需给定旧的和新的列名和列当前的类型，old_col_name 表示旧的列名。column_definition 中定义新的列名和当前数据类型。
- MODIFY [COLUMN]子句：修改指定列的类型。
- DROP 子句：从表中删除列或约束。
- RENAME 子句：修改该表的表名，new_tbl_name 是新表名。
- ORDER BY 子句：在创建新表时，让各行按一定的顺序排列。注意，在插入和删除后，表不会仍保持此顺序。在对表进行了大的改动后，通过使用此选项，可以提高查询效率。在有些情况下，如果表按列排序，对于 MySQL 来说，排序可能会更简单。
- table_options：修改表选项，具体定义与 CREATE TABLE 语句中一样。

在 MySQL 中，可以在一个 ALTER TABLE 语句中写入多个 ADD、ALTER、DROP 和 CHANGE 子句，中间用逗号分开。

【例 3.7】 假设已经在数据库 jwgl 中创建了表 student，向表中增加新列“籍贯”。

```
mysql > ALTER TABLE student ADD COLUMN 籍贯 char(20) NULL ;
```

【例 3.8】 将数据库 jwgl 中 student 表的列 name 名称修改为“姓名”。

```
mysql > ALTER TABLE student CHANGE name 姓名 char(8);
```

【例 3.9】 将数据库 jwgl 中 student 表的列 Sbirth 的数据类型改为 INT。

```
mysql > ALTER TABLE student MODIFY Sbirth INT NOT NULL;
```

【例 3.10】 将数据库 jwgl 中 student 表的 Sbirth 列删除，并增加 age 列。

```
mysql > USE jwgl;
mysql > ALTER TABLE student ADD age Tinyint NULL, DROP COLUMN Sbirth;
```

【例 3.11】 将数据库 jwgl 中 student 表改名为 Stu。

```
mysql > ALTER TABLE student RENAME TO Stu ;
```

此外，还可以用 RENAME TABLE 语句来更改表的名字，其格式如下：

```
RENAME TABLE tbl_name TO new_tbl_name [, tbl_name2 TO new_tbl_name2];
```

3. 查看表结构

在创建好数据表之后，可用 DESCRIBE 和 SHOW CREATE TABLE 语句查看表结构。DESCRIBE 的命令格式如下：

```
DESCRIBE tbl_name [col_name];
```

其中：参数 col_name 可以是单列的名称，也可以是包含“%”和“_”通配符的字符串。

【例 3.12】 查看表 jwgl 数据库中学生表(表名：student)结构。

查询语句如下：

```
mysql > DESCRIBE student;
```

查询结果如图 3.1 所示。

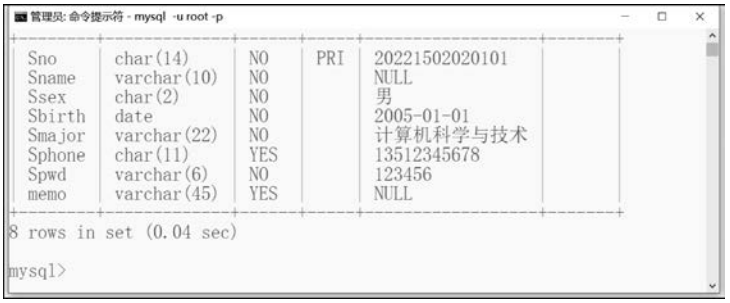

图 3.1　命令行模式的 MySQL 8 数据表结构查询过程和结果

此外，也可以用 SHOW COLUMNS FROM table_name FROM database_name 或 SHOW COLUMNS FROM database_name. table_name 来查看表中各列的信息。

4. 删除数据表

删除数据表就是将数据库中已经存在的表从数据库中删除。注意，在删除之前，最好对表中的数据做个备份，以免造成无法挽回的后果。删除数据表的命令格式如下：

```
DROP [TEMPORARY] TABLE [IF EXISTS] tbl_name [,tbl_name]…
```

其中，tbl_name 是要被删除的表名。IF EXISTS 子句可以避免要删除的表不存在时出现错误信息。这个命令将表的描述、表的完整性约束、索引、表相关的权限等全部删除。

【例 3.13】 删除 jwgl 数据库的表 test。

```
mysql > USE jwgl;
mysql > DROP TABLE IF EXISTS test;
```

5. 复制数据表

命令格式如下：

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
            [ ( ) LIKE old_tbl_name [ ] ]
            | [AS (select_statement)];
```

说明：使用 LIKE 关键字创建一个与 old_table_name 表相同结构的新表，列名、数据类型、空指定和索引也将复制，但是表的内容不会复制，因此创建的新表是一个空表。使用 AS 关键字可以复制原表的内容，但索引和完整性约束不会被复制。select_statement 是一个表达式，也可以是一条 SELECT 语句。

【例 3.14】 复制数据库 jwgl 的表 student，新表名为 student_1。

```
mysql > CREATE TABLE student_1 LIKE student ;
```

【例 3.15】 创建表 student 的一个名为 student_2 的副本，并且复制其内容。

```
mysql > CREATE TABLE student_2 AS (SELECT * FROM student);
```


视频讲解

3.3.4 MySQL 表记录操作

1. 插入新记录

在 MySQL 中，向表中添加新记录的命令是 INSERT，其格式如下：

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name [(col_name, …)] VALUES ({expr | DEFAULT}, …),(…), …
    | SET col_name = {expr | DEFAULT}, …
    [ ON DUPLICATE KEY UPDATE col_name = expr, … ];
```

参数说明如下。

- tbl_name：被操作的表名。
- col_name：需要插入数据的列名。如果要给全部列插入数据，列名可以省略。对于没有指出的列，它们的值根据列默认值或有关属性来确定。MySQL 处理的原则是：具有 IDENTITY 属性的列，系统生成序号来唯一标志列；具有默认值的列，其值为默认值；没有默认值的列，若允许为空值，则其值为空值，若不允许为空值，则出错；TIMESTAMP 列系统自动赋值。
- VALUES 子句：包含各列需要插入的数据清单，数据的顺序要与列的顺序相对应。若 tbl_name 后不给出列名，则在 VALUES 子句中要给出每一列（除 IDENTITY 和 TIMESTAMP 类型的列）的值，如果列值为空，则值必须置为 NULL，否则会出错。VALUES 子句中的值：expr 可以是一个常量、变量或一个表达式，也可以是空值，其值的数据类型要与列的数据类型一致。例如，列的数据类型为 char，插入的数据是 123 就会出错。当数据为字符型时，要用单引号括起。DEFAULT：指定为该列的默认值。前提是该列原先已经指定了默认值。如果列清单和 VALUES 清单都为空，则 INSERT 会创建一行，每个列都设置成默认值。
- LOW_PRIORITY：可以用在 INSERT、DELETE 和 UPDATE 等操作中，当原有客户端正在读取数据时，延迟操作的执行，直到没有其他客户端从表中有新的读取请求（仅适用于 MyISAM、MEMORY 和 ARCHIVE 表）为止。
- DELAYED：若使用此关键字，则服务器会把待插入的行放到一个缓冲器中，而发送 INSERT DELAYED 语句的客户端会继续运行。如果表正在被使用，则服务器会保留这些行。当表空闲时服务器开始插入行，并定期检查是否有新的读取请求（仅适用于 myisam、memory 和 archive 表）。
- HIGH_PRIORITY：可用在 SELECT 和 INSERT 操作中，使操作优先执行。
- IGNORE：使用此关键字，在执行语句时出现的错误就会被当作警告处理。
- ON DUPLICATE KEY UPDATE…：使用此选项插入行后，若导致 UNIQUE KEY 或 PRIMARY KEY 出现重复值，则根据 UPDATE 后的语句修改旧行（使用

此选项时 DELAYED 被忽略)。

- SET 子句：SET 子句用于给列指定值，使用 SET 子句时表名的后面省略列名。要插入数据的列名在 SET 子句中指定，col_name 为指定列名，等号后面为指定数据，未指定的列，列值指定为默认值。

【例 3.16】 向 jwgl 数据库的表 student(表中列包括 Sno，Sname，Ssex，Sbirth，Smajor，Sphone，Spwd，memo)中插入如下的一行：

20221502020101，张红，女，2004-04-10，计算机科学与技术，18138665177，123456，NULL。

使用下列语句：

```
mysql > USE jwgl;
mysql > INSERT INTO student VALUES('20221502020101','张红','女','2004 - 04 - 10','计算机科学与
技术','18138665177','123456',NULL);
Query OK, 1 row affected
```

也可以使用如下 SET 子句来实现：

```
mysql > INSERT INTO student
SET Sno = '20221502020101',Sname = '张红', Ssex = '女', Sbirth = '2004 - 04 - 10', Smajor = '计算机
科学与技术', Sphone = '18138665177';
Query OK, 1 row affected
```

另外，MySQL 还支持图片的存储，图片一般以路径的形式来存储，即插入图片可以采用直接插入图片的存储路径。当然也可以直接插入图片本身，只要用 LOAD_FILE 函数即可。

【例 3.17】 向 student 表中插入一行数据：20221502020103，李亮，男，2005-02-09，计算机科学与技术，18503715677，123456，picture.jpg。照片路径为 D:\IMAGE\picture.jpg。

使用如下语句：

```
mysql > INSERT INTO student VALUES('20221502020103','李亮','男','2005 - 02 - 09','计算机科学与
技术','18503715677','123456','D:\IMAGE\picture.jpg');
```

下列语句是直接存储图片本身：

```
mysql > INSERT INTO student VALUES('20221502020103','李亮','男','2005 - 02 - 09','计算机科学与
技术','18503715677','123456',LOAD_FILE('D:\IMAGE\picture.jpg'));
```

2. 更新表记录

MySQL 使用 UPDATE 命令更新表记录，命令的格式如下：

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1 = expr1 [, col_name2 = expr2 … ]
    [WHERE where_definition] [ORDER BY … ] [LIMIT row_count];
```

说明：SET 子句为根据 WHERE 子句中指定的条件对符合条件的数据行进行更新。若语句中不设定 WHERE 子句，则更新所有行。col_name1、col_name2…为要更新列值的列名，expr1、expr2…可以是常量、变量或表达式。可以同时更新所在数据行的多个列值，中间用逗号隔开。

【例 3.18】 将 jwgl 数据库的 student 表中姓名为“李亮”的同学的专业改为“市场营销”，电话号码改为 18603775600。

```
mysql > UPDATE student SET Smajor = '市场营销', Sphone = '18603775600'
        WHERE Sname = '李刚';
Query OK, 1 row affected (0.08 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

更新多个表，基本 SQL 语法格式如下：

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
SET col_name1 = expr1 [, col_name2 = expr2 … ] [WHERE where_definition];
```

说明：table_references 中包含了多个表的联合，各表之间用逗号隔开。

【例 3.19】 表 course 和表 select_course 中有共同字段 course_no，并且在表 course 中 course_no 为主键。当表 course 和表 select_course 中字段 course_no 的值相同并且 course_no 值为 01 时，将表 select_course 中对应的 score 值更新为 90，将表 course 中对应的 course_nature 值更新为“选修”。

```
mysql > UPDATE course, select_course SET select_course.score = '90', course.course_nature = '选
修'
      WHERE course.course_no = select_course.course_no and course.course_no = '01';
Query OK, 0 rows affected
Rows matched: 5 Changed: 0 Warnings: 0
```

3. 删除表记录

从数据表中删除记录通常用 DELETE 命令，其格式如下：

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
    [WHERE where_definition] [ORDER BY ...] [LIMIT row_count];
```

参数说明如下。

- QUICK：快速删除。
- FROM：用于说明从何处删除数据，tbl_name 为要删除数据的表名。
- WHERE：where_definition 中的内容为指定的删除条件。如果省略 WHERE 子句则删除该表的所有行。
- ORDER BY：各行按照子句中指定的顺序进行删除，此子句只在与 LIMIT 联用时才起作用。ORDER BY 子句和 LIMIT 子句的具体定义将在 SELECT 语句中介绍。
- LIMIT：被删除行的最大值。

【例 3.20】 将数据库 jwgl 的 student 表中姓名为“李刚”的记录删除。

```
mysql > USE jwgl;
mysql > DELETE FROM student WHERE name = '李刚';
```

【例 3.21】 将 jwgl 数据库的 student 表中专业为“市场营销”的所有行删除。

```
mysql > USE jwgl;
mysql > DELETE FROM student WHERE Smajor = '市场营销';
Query OK, 2 rows affected (0.30 sec)
```

如果想删除表中的所有记录，还可用 TRUNCATE TABLE 语句。由于 TRUNCATE TABLE 语句将删除表中的所有数据，且无法恢复，因此使用时必须十分小心。

TRUNCATE TABLE 语句的格式：TRUNCATE TABLE table－name;

说明：TRUNCATE TABLE 语句在功能上与不加 WHERE 子句的 DELETE 语句(如 DELETE FROM student)相同，二者均删除表中的全部行。但 TRUNCATE TABLE 语句比 DELETE 语句速度快，且使用的系统和事务日志资源少。DELETE 语句每删除一行，会在事务日志中为所删除的每行记录一项。而 TRUNCATE TABLE 语句通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。使用 TRUNCATE TABLE 语句，AUTO_INCREMENT 计数器被重新设置为该列的初始值。此外，参与索引和视图的表不能用 TRUNCATE TABLE 语句删除记录，而应使用 DELETE 语句。

3.3.5 MySQL 常量

常量是指在程序运行过程中值不变的量。按照数据类型，可以将常量分为字符串常量、数值常量、时间日期常量、布尔常量、二进制常量、十六进制常量和 NULL。

1. 字符串常量

字符串常量是用单引号或双引号括起来的字符序列。大多数编程语言(如 Java、C)使用双引号表示字符串，为了便于区别，在 MySQL 数据库中推荐使用单引号表示字符串。字符串常量分为 ASCII 字符串常量和 Unicode 字符串常量。ASCII 字符串常量是用单引号括起来的，由 ASCII 字符构成的符号串；Unicode 字符串常量与 ASCII 字符串常量类似，但它前面有一个 N 标志符[N 代表 SQL-92 标准中的国际语言(National Language)]，只能用单引号括起字符串，且 Unicode 数据中的每个字符用两个字节存储，而每个 ASCII 字符用一个字节存储。在字符串中不仅可以使用普通字符，也可使用转义符(如表 3.1 所示)。

表 3.1　字符串转义序列表

序　列	含　义
\0	一个 ASCII 0 (NULL)字符
\N	一个换行符
\R	一个回车符(Windows 中使用\R\N 作为新行标志)
\T	一个定位符
\B	一个退格符
\Z	一个 ASCII 26 字符(CTRL＋Z)
\'	一个单引号(“'”)

续表

序　列	含　义
\"	一个双引号(“"”)
\\	一个反斜线(“\”)
\%	一个“%”符。它用于在正文中搜索“%”的文字实例
_	一个“_”符。它用于在正文中搜索“_”的文字实例

2. 数值常量

数值常量可以分为整数常量和小数常量。整数常量是不带小数点的十进制数，如2015，+14，-214。浮点数常量是使用小数点的数值常量，如99.5，10.5E3，0.4E-5。

3. 时间日期常量

时间日期常量是一个符合特殊格式的字符串。用单引号将表示日期时间的字符串括起来构成。日期型常量包括年、月、日，数据类型为DATE，表示为2015-06-17这样的值。时间型常量包括小时数、分钟数、秒数及微秒数，数据类型为TIME，表示为12:30:43.00013这样的值。MySQL还支持日期、时间的组合，数据类型为DATETIME或TIMESTAMP，如2015-06-03 12:27:43。需要特别注意的是，MySQL是按年-月-日的顺序表示日期的。中间的间隔符“-”也可以使用如“\”、“@”或“%”等特殊符号。

4. 布尔常量

布尔常量只包含：TURE和FALSE。FALSE代表0，TRUE代表1。

5. 二进制常量

二进制常量由数字0和1组成。二进制常量表示方法：前缀为B，后面紧跟着一个二进制字符串。使用SELECT语句显示二进制数时，会将其自动转换为字符串再进行显示。直接显示B'value'的值可能是一系列特殊的符号。例如，B'0'显示为空白，B'111101'显示为等号，B'11'显示为“心形”符号。

6. 十六进制常量

十六进制常量由数字0～9及字母A～F组成(字母不分大小写)。十六进制常量有两种表示方法。第一种：前缀为大写字母X或小写字母x，后面紧跟一个十六进制字符串。第二种：前缀为0x，后面紧跟一个十六进制数(不用引号)。

7. NULL常量

NULL值可适用于各种字段类型，它通常用来表示“值不确定”“没有值”等含义，并且不同于数字类型的0或字符串类型的空字符串。NULL参与算数运算、比较运算和逻辑运算时，结果依然是NULL。

3.3.6 MySQL变量

1. 系统变量

在MySQL中，系统变量(以@@开头)用于定义当前MySQL实例的属性、特征。系统变量分为全局系统变量(Global)与会话系统变量(Session)。MySQL服务成功启动后，如果没有MySQL客户机连接MySQL服务器，那么MySQL服务器内存中的系统变量全部是全局系统变量(有393个)；每一个MySQL客户机成功连接MySQL服务器后，都会产生

与之对应的会话，会话期间，MySQL 实例会在 MySQL 服务器内存中生成与该会话对应的会话系统变量，这些会话系统变量的初始值是全局系统变量值的复制。

查看 MySQL 服务器内存中所有的全局系统变量信息，可以使用 SHOW GLOBAL VARIABLES 语句。查看与当前会话相关的所有会话系统变量，可以使用 SHOW SESSION VARIABLES 语句。可以使用 SHOW VARIABLES 语句得到系统变量清单。

鉴于 MySQL 系统变量与用户编程不存在太大关系，因此本项目不再详述。

2. 用户自定义变量

与其他编程语言相似，在用 MySQL 语言进行编程时，使用用户自定义变量存储“临时结果”。在 MySQL 中，用户自定义变量分为用户会话变量（以@开头）以及局部变量（不以@开头）。用户自定义变量可以由当前字符集的字符、“.”、“_”和“$”组成，当变量名中需要包含一些特殊符号（如空格、#等）时，可以使用双引号或单引号将整个变量括起来。

用户会话变量与局部变量的区别与联系如下。

- 用户会话变量名以@开头，而局部变量名前面没有@符号。
- 局部变量使用 DECLARE 命令定义（存储过程参数、函数参数除外），定义时必须指定局部变量的数据类型。局部变量定义后，才可以使用 SET 命令或者 SELECT 语句为其赋值。用户会话变量使用 SET 命令或者 SELECT 语句定义并赋值，定义用户会话变量时无须指定数据类型。例如“Declare@Student_Num Int;”语句是错误语句，用户会话变量不能使用 DECLARE 命令定义。
- 用户会话变量的作用范围与生存周期大于局部变量。局部变量如果作为存储过程或者函数的参数使用，则在整个的存储过程或函数内有效；如果定义在存储程序的 BEGIN-END 语句块中，则仅在当前的 BEGIN-END 语句块中有效。用户会话变量在本次会话期间一直有效，直至关闭服务器连接。
- 如果局部变量嵌入到 SQL 语句中，由于局部变量名前没有@符号，这就要求局部变量名不能与表字段名同名，否则将出现无法预期的结果。用户会话变量前面存在@符号，因此用户会话变量没有该限制。

3.3.7　MySQL 函数

1. 数学函数

绝对值函数：ABS(X)，返回 X 的绝对值。

取整函数：ROUND(X)/ROUND(X,D)，返回最接近于 X 的整数。在有两个参数的情况下，返回 X，其值保留到小数点后 D 位，而第 D 位的保留方式为四舍五入。若要保留 X 值小数点左边的 D 位，可将 D 设为负值。

求平方根函数：SQRT(X)，返回非负 X 的二次方根。

随机数函数：RAND()/RAND(N)，返回一个随机浮点值 V，范围为 0～1；若指定整数参数 N，则它被用作种子值，用来产生重复序列。

取最大整数函数：FLOOR(X)，返回不大于 X 的最大整数值。

圆周率函数：PI()，返回圆周率 π 的值，默认显示小数位数是 7 位。

四舍五入函数：TRUNCATE(X,D)，返回被舍去至小数点后 D 位的数字 X。

最大值函数：GREATEST(X1,X2,X3…)，返回参数中的最大值。

最小值函数：LEAST(X1,X2,X3…)，返回参数中的最小值。

求二进制值函数：BIN(X)，返回参数 X 的二进制值。

求八进制值函数：OTC(X)，返回参数 X 的八进制值。

求十六进制值函数：HEX(X)，返回参数 X 的十六进制值。

2. 聚合函数

聚合函数是把数据聚合起来的函数，例如，对工资数据库中所有员工的实发工资求和、求平均数等，MySQL 的主要聚合函数及其功能说明如下。

求和函数：SUM([DISTINCT] Expr)，返回 Expr 的总和。

求平均值函数：AVG([DISTINCT] Expr)，返回 Expr 的平均值，DISTINCT 选项可用于返回 Expr 的不同值的平均值。

算数量函数：COUNT(Expr)，用来计算表中满足条件的记录条数。

求最大值函数：MAX([DISTINCT] Expr)，用来计算表中满足条件的数的最大值。

求最小值函数：MIN([DISTINCT] Expr)，用来计算表中满足条件的数的最小值。

3. 字符串函数

字符串长度函数：CHAR_LENGTH(str)函数的返回值为字符串 str 所包含的字符个数，LENGTH(str)函数的返回值为字符串 str 的字节长度。例如，CHAR_LENGTH('你是')=2，但 LENGTH('你是')=6。因为在使用 UTF8 编码字符集时，一个汉字是 3 字节，一个数字或字母是 1 字节。

拼接函数：CONCAT(str1,str2,…)，返回结果为连接参数产生的字符串。如有任何一个参数为 NULL，则返回值为 NULL。

重复函数：REPEAT(str,count)，返回一个由重复的字符串 str 组成的字符串，字符串 str 的数目为 count。若 count≤0，则返回一个空字符串。若 str 或 count 为 NULL，则返回 NULL。

定位函数 1：FIND_IN_SET(str,strlist)，返回 str 在 strlist 中的位置值。如果 str 不在 strlist 或 strlist 为空字符串，则返回值为 0；如果任意一个参数为 NULL，则返回值为 NULL。

定位函数 2：LOCATE(substr,str)或 LOCATE(substr,str,pos)。LOCATE(substr,str)返回字符串 str 中子字符串 substr 第一次出现的位置；LOCATE(substr,str,pos)返回字符串 str 中子字符串 substr 从 pos 开始第一次出现的位置，若 substr 不在 str 中，则返回值为 0。

定位函数 3：INSTR(str,substr)，返回字符串 str 中子字符串 substr 第一次出现的位置，与 LOCATE(substr,str)函数的功能相同。

左截取函数：LEFT(str,len)，返回字符串 str 最左侧 len 个字符。

右截取函数：RIGHT(str,len)，返回字符串 str 最右侧 len 个字符。

中间截取函数：SUBSTRING(str,pos)、SUBSTRING(str FROM pos)、SUBSTRING(str,pos,len)、SUBSTRING(str FROM pos FOR len)。不带 len 参数的 SUBSTRING 函数从字符串 str 返回一个从 pos 开始的子字符串；带有 len 参数的 SUBSTRING 函数从字符串 str 返回一个长度为 len、从 pos 开始的子字符串；使用 FROM 的格式为标准 SQL 语法。也可能对 pos 使用一个负值。若这样，则子字符串的位置起始于字符串结尾的 pos 字

符，而不是字符串的开头位置。

大小写转换函数：LCASE(str)，将字符串 str 转换为小写。

字符串替换函数：REPLACE(str,from_str,to_str)，将字符串 str 中的子字符串 from_str 替换为子字符串 to_str。

反转函数：REVERSE(str)，返回字符串 str 的反序。

空格字符串：SPACE(N)，返回一个由 *N* 个空格组成的字符串。

4. 日期和时间函数

日期函数：CURDATE()、CURRENT_DATE()，按照年月日格式返回当前日期。

时间函数：CURTIME()、CURRENT_TIME()，按照时分秒格式返回当前时间。

日期时间函数：NOW()，返回当前日期和时间值。

求年份函数：YEAR(Expr)，返回 Expr 中年的数值。

求月份函数：MONTH(Expr)、MONTHNAME(Expr)，分别返回数值和字符串格式的月份。

求日数函数：DAYOFYEAR(Expr)、DAYOFWEEK(Expr)、DAYOFMONTH(Expr)函数分别返回这一天在一年、一星期及一个月中的序数。

时分秒函数：HOUR(Expr)、MINUTE(Expr)和 SECOND(Expr)分别返回时间值的小时、分钟和秒的部分。

5. 加密函数

AES 加密函数：AES_ENCRYPT(str,key_str)，基于 AES 对称加密算法，用第二个参数 key_str 作为加密密钥，对第一个参数 str 进行加密。

AES 解密函数：AES_DECRYPT(crypt_str,key_str)，基于 AES 对称加密算法，用第二个参数 key_str 作为解密密钥，对已加密的 crypt_str 进行解密。

密码字符串生成函数：PASSWORD(str)，创建一个经过加密的密码字符串。

加密字符串函数 1：ENCRYPT(str[, salt])，该函数通过使用 UNIX crypt()系统调用来加密 str，并返回一个二进制串。其中 salt 变量是一个包含多于两个字符的字符串。如果 salt 没有给定，则使用一个随机值。如果 crypt()系统调用在用户的操作系统上不可用(Windows 操作系统便如此)，该函数返回为 NULL。

加密字符串函数 2：ENCODE(str,pass_str)，使用 pass_str 作为密码加密 str。

解密字符串函数：DECODE (crypt_str,pass_str)，用 pass_str 解密 crypt_str。

MD5 数字摘要函数：MD5(str)，返回字符串 str 的 MD5 校验和(128 位)。

SHA 数字摘要函数：SHA(str)或 SHA1(str)，返回 str 的 SHA1 校验和(160 位)。

6. 格式化函数

数据内容格式化函数：FORMAT(X,D)，将数字 X 的格式写成'#,###,###.##'格式，即保留小数点后 D 位，而第 D 位的保留方式为四舍五入，然后将结果以字符串的形式返回。若 D 为 0，则返回不带有小数点的结果或不含小数部分的结果。

日期、时间格式化函数：DATE_FORMAT(date,format)、TIME_FORMAT(time,format)，根据 format 字符串格式化 date、time，format 的格式如下。

%S，%s：两位数字形式的秒(00,01,…,59)。

%I，%i：两位数字形式的分(00,01,…,59)。

%H：两位数字形式的小时，24 小时(00,01,…,23)。
%h：两位数字形式的小时，12 小时(01,02,…,12)。
%k：数字形式的小时，24 小时(0,1,…,23)。
%l：数字形式的小时，12 小时(1,2,…,12)。
%T：24 小时的时间形式(hh：mm：ss)。
%r：12 小时的时间形式(hh：mm：ss AM 或 hh：mm：ss PM)。
%p：AM 或 PM。
%W：一周中每一天的名称(Sunday,Monday,…,Saturday)。
%a：一周中每一天名称的缩写(Sun,Mon,…,Sat)。
%d：两位数字表示月中的天数(00,01,…,31)。
%e：数字形式表示月中的天数(1,2,…,31)。
%D：英文后缀表示月中的天数(1st,2nd,3rd,…)。
%w：以数字形式表示周中的天数(0＝Sunday,1＝Monday,…,6＝Saturday)。
%j：以三位数字表示年中的天数(001,002,…,366)。
%U：周(0,1,…,52)，其中 Sunday 为周中的第一天。
%u：周(0,1,…,52)，其中 Monday 为周中的第一天。
%M：月名(January,February,…,December)。
%b：缩写的月名(Jan,Feb,…,Dec)。
%m：两位数字表示的月份(01,02,…,12)。
%c：数字表示的月份(1,2,…,12)。
%Y：四位数字表示的年份。
%y：两位数字表示的年份。
%%：直接值“%”。

3.3.8 MySQL 运算符

1. 算术运算符

算术运算符有＋(加)、－(减)、*(乘)、/(除)、%(求模)和 DIV(整除)6 种。注意就＋(加)、－(减)、*(乘)而言，若两个参数均为整数，则其计算结果的精确度为 BIGINT (64 比特)，若其中一个参数为无符号整数，而其他参数也是整数，则结果为无符号整数。

2. 比较运算符

比较运算符(又称关系运算符)，用于比较两个表达式的值，其运算结果为逻辑值，可以为三种之一：1(真)、0(假)和 NULL(不确定)。MySQL 的比较运算符以及含义如表 3.2 所示。

表 3.2 MySQL 的比较运算符以及含义

运 算 符	含 义	运 算 符	含 义
＝	等于	<＝	小于或等于
＞	大于	<＞、!＝	不等于
＜	小于	<＝＞	相等或都等于空时返回 True
＞＝	大于或等于		

3. 逻辑运算符

逻辑运算符用于对某个条件进行测试，运算结果为 TRUE(1)或 FALSE(0)。MySQL 提供的逻辑运算符及其作用如表 3.3 所示。

表 3.3　逻辑运算符及其作用

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

4. 位运算符

位运算符在两个表达式之间执行二进制位操作，这两个表达式的类型可为整型或与整型兼容的数据类型，如字符型(不能为 Image 类型)。位运算符及其运算规则如表 3.4 所示。

表 3.4　位运算符及其运算规则

运　算　符	运 算 规 则	运　算　符	运 算 规 则
&	按位与	~	按位取反
|	按位或	>>	按位右移
^	按位异或	<<	按位左移

3.3.9　MySQL 表达式

MySQL 表达式是由 MySQL 常量、变量、列名(字段名)、复杂计算、运算符和函数的组合。一个表达式通常可以得到一个值。与常量和变量一样，表达式的值也具有某种数据类型，可能的数据类型有字符类型、数值类型、日期时间类型。根据表达式的值的类型，表达式可分为字符型表达式、数值型表达式和日期型表达式。

在 MySQL 中，当表达式的结果只是一个值(如一个数值、一个单词或一个日期)，这种表达式叫作标量表达式，例如 1+2，'a'>'b'。当表达式的结果是由不同类型数据组成的一行值，这种表达式叫作行表达式，例如('081101','王林','计算机',500)。当表达式的结果为 0 个、1 个或多个行表达式的集合，那么这个表达式就叫作表的表达式。

表达式一般用在 SELECT 及 SELECT 语句的 WHERE 子句中，具体用法在后续项目中讲述。

3.4　任务小结

本项目主要介绍了 MySQL 中数据库级的 SQL 语句、数据表级的 SQL 语句、表记录的 SQL 语句的具体用法，详细介绍了 MySQL 在 SQL 语句中各种基本元素(如常量、变量、运算符、表达式、函数)的具体使用，掌握这些知识要点为下一步深入学习 MySQL 打下坚实的基础。

与本项目相关的经验如下：

MySQL 在 Windows 操作系统下不区分大小写。如果想在查询时区分搜索字段值的大小写，则字段值需要设置 BINARY 属性，设置办法如下。

- 创建时设置：CREATE TABLE T(A VARCHAR(10) BINARY)。
- 使用 ALTER 修改：ALTER TABLE `tablename` MODIFY COLUMN `cloname` VARCHAR(45) BINARY。
- MySQL TABLE EDITOR 中直接勾选 BINARY 项。

MySQL 在 Linux 操作系统下，数据库名、表名、列名、别名大小写规则如下。

- 数据库名与表名严格区分大小写。
- 表的别名是严格区分大小写的。
- 列名与列的别名在所有的情况下都忽略大小写。
- 变量名也是严格区分大小写的。

3.5 拓展提高

3.5.1 MySQL 复杂运算

1. 关系运算

关系的基本运算有两类：一类是传统的集合运算(并、差、交等)，另一类是专门的关系运算(选择、投影、连接等)。

传统的集合运算有以下几种。

- 并(UNION)：设有两个关系 R 和 S，它们具有相同的结构。R 和 S 的并是由属于 R 或属于 S 的元组组成的集合，运算符为∪。记为 T＝R∪S。
- 差(DIFFERENCE)：R 和 S 的差是由属于 R 但不属于 S 的元组组成的集合，运算符为－。记为 T＝R－S。
- 交(INTERSECTION)：R 和 S 的交是由既属于 R 又属于 S 的元组组成的集合，运算符为∩。记为 T＝R∩S 或 R∩S＝R－(R－S)。

MySQL 只支持 Union(并集)集合运算，而且是 MySQL 4.0 以后才有此功能；但是对于交集和差集未实现。

MySQL 的并集运算有 UNION 和 UNION ALL 两个，主要区别是 UNION 对两个结果集进行并集操作，重复数据只显示一次；而 UNION ALL，对两个结果集进行并集操作，重复数据全部显示。

2. 选择运算

从关系中找出满足给定条件的那些元组称为选择。其中的条件是以逻辑表达式给出的，值为真的元组将被选取。这种运算是从水平方向抽取元组。选择是单目运算，其运算对象是一个表。该运算按给定的条件，从表中选出满足条件的行形成一个新表作为运算结果。

可以记作：$s_F(R)=\{t|t\in R\wedge F(t)='真'\}$。

其中下标 F 表示选择条件，它是一个逻辑表达式，取逻辑值“真”或“假”。R 表示关系，也就是 MySQL 中的数据表。因此，选择运算是从关系 R 中选取使逻辑表达式 F 为真的元组。

3. 投影运算

从关系模式中挑选若干属性组成新的关系称为投影。这是从列的角度进行的运算，相

当于对关系进行垂直分解。投影是单目运算，该运算从表中选出指定的属性值组成一个新表，即关系 R 上的投影是从 R 中选择出若干属性列组成新的关系。记作：$\Pi A(R)=\{t[A]|t\in R\}$，其中 A 为 R 中的属性列(即列名)表，R 是表名即关系。

4. 连接运算

连接运算是根据给定的条件，从两个已知关系 R 和 S 的笛卡儿积中，选取满足连接条件(属性之间)的若干元组组成新的关系。

记作：$(R)\underset{F}{>\!\!<}(S)$。

其中 F 是选择条件。假设 A 和 B 分别为 R 和 S 上度数相等且可比的属性组。连接运算从 R 和 S 的笛卡儿积 R×S 中选取关系 R 在 A 属性组上的值与关系 S 在 B 属性组上值满足比较关系 F 的元组。

连接运算有四种最为重要也是最为常用的连接，即条件连接、等值连接、自然连接和外连接，它们的含义分别如下。

- 条件连接：从两个关系的笛卡儿积中选取属性间满足一定条件的元组。
- 等值连接：从关系 R 与 S 的笛卡儿积中选取 A 和 B 属性值相等的那些元组。其查询结果中列出被连接关系(表)中的所有列，包括其中的重复列。
- 自然连接：也是等值连接，是数据库应用中最常用的连接。它从两个关系的笛卡儿积中，选取公共属性满足等值条件的元组，但新关系不包含重复的属性。一般的连接是从行的角度进行运算的，但自然连接还需要取消重复列，所以是同时从行和列的角度进行运算的。
- 外连接：内连接时，返回查询结果集合中的仅是符合查询条件(WHERE 搜索条件或 HAVING 条件)和连接条件的行。而采用外连接时，它返回到查询结果集合中的不仅包含符合连接条件的行，而且还包括左表(左外连接时)、右表(右外连接时)或两个表(全外连接)中的所有数据行。

3.5.2　数据类型选择

MySQL 提供大量的数据类型，为了优化存储，提高数据库性能，在任何情况下均应使用最精确的类型，即在所有可以表示该列值的类型中，该类型使用的存储最少。

1. 整数和浮点数

如果不需要小数部分，则使用整数来保存数据；如果需要表示小数部分，则用浮点数类型。对于浮点数据列，存入的数据会对该列定义的小数位进行四舍五入。例如，如果列的值的范围为 1～99999，若使用整数，则 MEDIUMINT UNSIGNED 是最好的类型；若需要存储小数，则使用 FLOAT 类型。

浮点类型包括 FLOAT 和 DOUBLE 类型。DOUBLE 类型精度比 FLOAT 类型高，因此，若要求存储精度较高时，应选择 DOUBLE 类型。

2. 浮点数和定点数

浮点数 FLOAT 和 DOUBLE 类型相对于定点数 DECIMAL 类型的优势是：在长度一定的情况下，浮点数能表示更大的数据范围。但是由于浮点数容易产生误差，因此对精度要求比较高时，建议使用 DECIMAL 类型来存储。DECIMAL 类型在 MySQL 中是以字符串存储的，用于定义货币等对精度要求较高的数据。在数据迁移中，FLOAT(M,D)是非标准

SQL 语言定义，数据库迁移可能会出现问题，最好不要这样使用。另外，两个浮点数进行减法和比较运算时也容易出问题，因此在进行计算时，一定要小心。如果进行数值比较，最好使用 DECIMAL 类型。

3. 日期与时间类型

MySQL 对于不同种类的日期和时间有很多的数据类型，例如 YEAR 类型和 TIME 类型。如果只需要记录年份，则使用 YEAR 类型即可；如果只记录时间，使用 TIME 类型。

如果同时需要记录日期和时间，则可以使用 TIMESTAMP 类型或者 DATETIME 类型。由于 TIMESTAMP 类型列的取值范围小于 DATETIME 类型的取值范围，因此存储范围较大的日期最好使用 DATETIME 类型。DATETIME 类型的年份可取 1000～9999，而 TIMESTAMP 类型的年份可取 1970～2037，还有就是 TIMESTAMP 类型在插入带微秒的日期时间时将微秒忽略。TIMESTAMP 类型还支持时区，即在不同时区转换为相应时间。

TIMESTAMP 类型还有一个 DATETIME 类型不具备的属性。默认的情况下，当插入一条记录但并没有指定 TIMESTAMP 类型这个列值时，MySQL 会把 TIMEATAMP 类型列设为当前的时间。因此当需要插入记录同时插入当前时间时，使用 TIMESTAMP 类型是方便的，另外，TIMESTAMP 类型在空间上比 DATETIME 类型更有效。

4. CHAR 类型与 VARCHAR 类型的特点与选择

CHAR 类型与 VARCHAR 类型的区别。

CHAR 类型是固定长度字符，VARCHAR 类型是可变长度字符；CHAR 类型会自动删除插入数据的尾部空格，VARCHAR 类型不会删除尾部空格。CHAR 类型是固定长度，所以它的处理速度比 VARCHAR 类型的速度要快，但是它的缺点是浪费存储空间。用户可根据实际需求选择使用。

存储引擎对于选择 CHAR 类型和 VARCHAR 类型的影响如下。

对于 MyISAM 存储引擎：最好使用固定长度的数据列代替可变长度的数据列。这样可以使整个表静态化，从而使数据检索更快，用空间换时间。

对于 InnoDB 存储引擎：使用可变长度的数据列，InnoDB 数据表的存储格式不分固定长度和可变长度，因此使用 CHAR 类型不一定比使用 VARCHAR 类型更好。但由于 VARCHAR 类型是按照实际的长度存储，比较节省空间，所以对磁盘 I/O 和数据存储总量比较好。

5. ENUM 类型和 SET 类型

ENUM 类型只能取单值，它的数据列表是一个枚举集合。它的合法取值列表最多允许有 65 535 个成员。因此，在需要从多个值中选取一个时，可以使用 ENUM 类型。例如，性别字段适合定义为 ENUM 类型，每次只能从“男”或“女”中取一个值。

SET 类型可取多值。它的合法取值列表最多允许有 64 个成员。空字符串也是一个合法的 SET 类型值。在需要取多个值时，适合使用 SET 类型。例如，要存储一个人的兴趣爱好，最好使用 SET 类型。

ENUM 类型和 SET 类型的值是以字符串形式出现的，但在内部，MySQL 以数值的形式存储它们。

6. BLOB 类型和 TEXT 类型

BLOB 类型是二进制字符串，主要存储图片、音频信息；TEXT 类型是非二进制字符串，只能存储纯文本文件。

自测与实验 3.1　教务管理数据库和数据表创建

1. 实验目的

(1) 验证用命令建立 MySQL 8 数据库。

(2) 验证用命令建立 MySQL 8 数据表。

2. 实验环境

(1) PC 一台。

(2) MySQL 8 数据库。

3. 实验内容

(1) 用本项目相关知识与实例所述的方法建立 MySQL 8 数据库。

(2) 用本项目相关知识与实例所述的方法建立 MySQL 8 数据表。

4. 实验步骤

参照相关知识与实例。

自测与实验 3.2　教务管理数据表信息的插入、删除、修改

1. 实验目的

(1) 验证用命令的方法在 MySQL 8 数据表中插入新记录。

(2) 验证用命令的方法删除 MySQL 8 数据表中的记录。

(3) 验证用命令的方法修改 MySQL 8 数据表中的记录。

2. 实验环境

(1) PC 一台。

(2) MySQL 8 数据库。

3. 实验内容

(1) 用本项目相关知识与实例所述的方法在 MySQL 8 数据表中插入新记录。

(2) 用本项目相关知识与实例所述的方法删除 MySQL 8 数据表中的记录。

(3) 用本项目相关知识与实例所述的方法修改 MySQL 8 数据表中的记录。

4. 实验步骤

参照相关知识与实例。