第 3 章

表　管　理

从本章开始将介绍一些标准的 SQL 语句，首先介绍数据定义语言中的表管理。


微课视频

3.1　管理数据库

在数据库中还有**模式(Schema)**概念，模式是数据库中所有对象的集合，包括表、字段、视图、索引和存储过程等。

在 MySQL 中模式就是数据库，即 Database，为了防止混淆，本书统一将模式称为数据库。

3.1.1　创建数据库

在 2.3.3 节介绍了通过使用 MySQL Workbench 工具管理数据库(即 SCHEMA)，这里不再赘述，本节重点介绍通过 SQL 代码创建数据库。其基本语法结构如下：

```
CREATE DATABASE db_name
```

创建数据库时 DATABASE 可以换成 SCHEMA。

例如，创建一个学校数据库(school_db)的示例代码如下：

```
-- 创建 school_db 数据库
CREATE DATABASE school_db ;
```

执行代码后会创建 school_db 数据库，SQL 代码中“--”是注释符，它与注释内容之间会保留一个空格。

执行 SQL 代码过程可以参考 2.3.5 节，这里不再赘述。

3.1.2　删除数据库

有时还需要删除数据库，删除数据库的基本语法结构如下：

```
DROP DATABASE db_name
```

删除学校数据库(school_db)的示例代码如下：

```
-- 删除数据库 school_db
DROP DATABASE school_db ;
```

上述代码执行后会删除 school_db 数据库，但是如果数据库不存在，则会发生如下错误。

```
Error Code: 1008. Can't drop database 'school_db'; database doesn't exist
```

使用 MySQL Workbench 工具执行上述 SQL 代码，结果如图 3-1 所示。

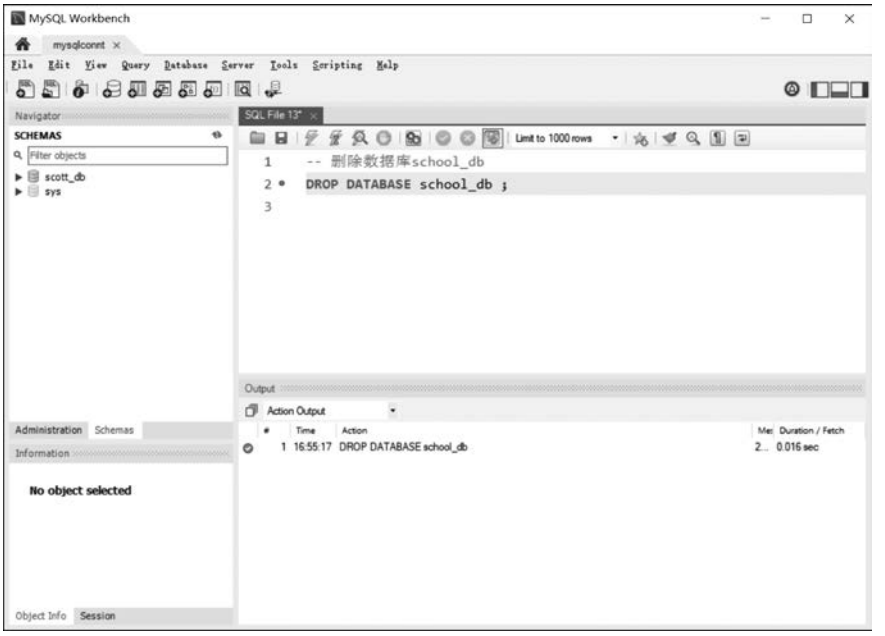

图 3-1 SQL 代码结果

为了防止试图删除不存在的数据库而引发的错误，可以使用 IF EXISTS 子句判断数据库是否存在，修改代码如下：

```
-- 删除数据库 school_db
DROP DATABASE IF EXISTS school_db;
```

3.1.3 选择数据库

由于可以有多个数据库，不同的数据库中又有很多不同的对象，因此选择哪个数据库的 SQL 语句也是非常重要的，选择数据库可以使用 USE 语句实现。

选择使用 school_db 数据库，示例代码如下：

```
-- 选择使用 school_db 数据库
USE  school_db;
```

如图 3-2 所示，有三个数据库都没有被选中，使用 USE 选择数据库后，结果如图 3-3 所示，被选中的数据会以加粗字体显示。

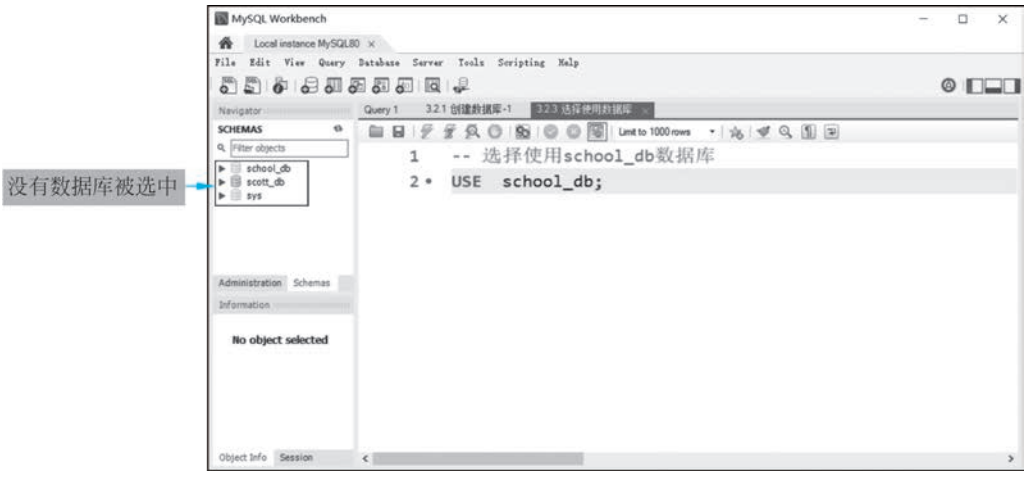

图 3-2　没有数据库被选中

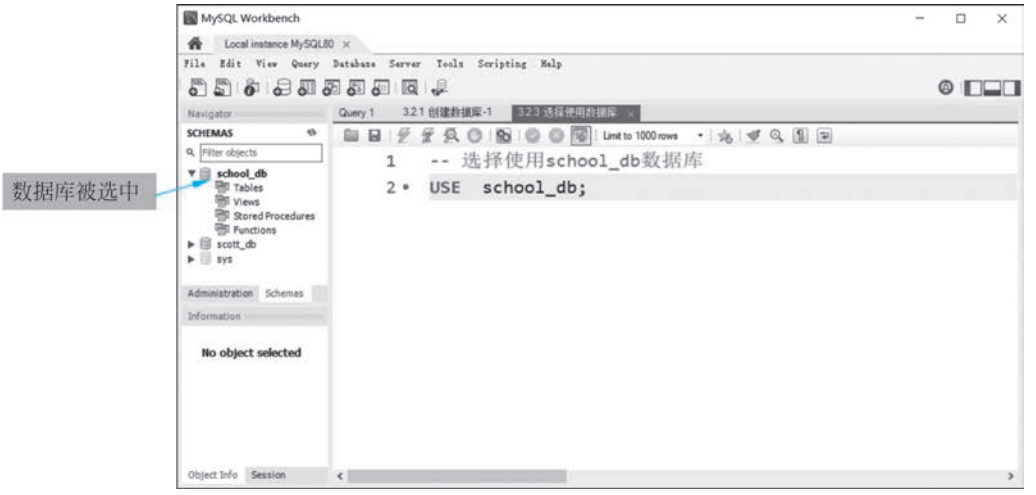

图 3-3　使用 USE 选择数据库


微课视频

3.2　创建表

表管理包括创建、修改和删除表操作，本节介绍创建表。

在数据库中创建表，可以使用 CREATE TABLE 语句。CREATE TABLE 语句的基本语法结构如下：

```
CREATE TABLE table_name (
table_field1 datatype[(size)],
table_field2 datatype[(size)],
    ...
)
```

其中，table_name 是表名；table_field1 和 table_field2 等是表中字段。表名和字段名是开发人员自定义的命名，但一般不推荐中文命名；如果有多个英文单词，推荐使用下画线分隔，如 s_id 和 s_name。

语法结构中 datatype 是字段的数据类型；size 指定数据类型所占用的内存空间。注意语法结构中“[]”括号中的内容可以省略，因此[(size)]表示 size 是可以省略的。如果要定义多个字段，则字段之间要用逗号“,”分隔，但是最后一个字段之后要省略逗号。

下面通过示例熟悉 CREATE TABLE 语句的使用，创建学生表，结构如表 3-1 所示。

表 3-1 学生表

字 段 名	数 据 类 型	长 度	备 注
s_id	INTEGER		学号
s_name	VARCHAR(20)	20	姓名
gender	CHAR(1)	1	性别，F 表示女，M 表示男
PIN	CHAR(18)	18	身份证号码

创建学生表的示例代码如下：

```
-- 3.2 创建表.sql
--   创建学生表的语句
CREATE TABLE student(
    s_id     INTEGER,              -- 学号                          ①
    s_name   VARCHAR(20),          -- 姓名                          ②
    gender   CHAR(1),              -- 性别,'F'表示女,'M'表示男      ③
    PIN      CHAR(18)              -- 身份证号码                    ④
)
```

由于创建表是属于 DDL 语句，因此该表的 SQL 可以命名为.ddl 或.sql。它是一个文本文件，可以通过任何文本编辑工具进行编辑。这种文件通常可以通过数据库管理工具执行，因此也称为脚本文件。

上述代码第①行定义 s_id 字段，其中 INTEGER 指定字段为整数类型。

代码第②行定义 s_name 字段，VARCHAR(20)表示可变长度，且最大长度为 20 位的字符串类型。

代码第③行定义 gender 字段，CHAR(1)表示固定长度为 1 位的字符串类型，其取值为'F'或'M'。

代码第④行定义 PIN 字段，目前身份证号码为 18 位字符串，因此该字段数据类型设置为 CHAR(18)，表示固定长度为 18 位的字符串类型。

上述代码运行会创建 student 表，但是如果没有使用 USE 选中数据库，而且也没有设置默认数据库，则会发生 Error Code：1046. No database selected 错误。


微课视频

3.3 字段数据类型

在创建表时，要求为每个字段指定具体的数据类型。关系数据共分为 4 种：字符串数据、数字数据、日期时间数据和大型对象。

3.3.1 字符串数据

大多数数据库都提供以下两种字符串类型。

(1) **固定长度(CHAR)**：固定长度字符串总是占据等量的内存空间，不管实际上在它们存储的数据量有多少。

(2) **可变长度(VARCHAR)**：可变长度的字符串只占据它们的内容所消耗的内存量。

例如，CHAR(2)表示固定两个字节长度的字符串，当只输入一个字节时，对于不足字符数据库会用空格补位，能够使之始终保持两个字节，这就是所谓的固定长度的字符串；VARCHAR(2)表示可变两个字节长度的字符串，当输入的字符串不足两个字节时，数据库不会补位。

如果不能确定字符串字段长度可以使用 TEXT 类型。它可以存储大量的文字数据。

3.3.2 数字数据

大多数数据库都提供至少两种数字数据类型：**整数(INTEGER)**、**浮点数(FLOAT 或 REAL)**。

整数和浮点数可以统一用 **numeric**[(p[,s])]类型表示。其中，numeric 表示十进制数字类型；p 为精度，即整数位数与小数位数之和；s 为小数位数。此外，还有一些数据库提供更加独特的数字类型。

3.3.3 日期和时间数据

大多数关系数据库支持的另一种独特的数据类型是日期和时间数据。数据库处理时间数据的方式有很多种，日期的存储和显示方法都可以变化，有些数据库还支持更多类型的时间数据。本质上，关系数据库所支持的 3 种日期时间数据类型为日期、时间、日期＋时间组合。

3.3.4 大型对象

大多数数据库为字段提供大型对象类型数据，大型对象主要分为以下两种数据类型。

（1）字符串大型对象（CLOB）：字符串大型对象保存大量的文本数据。有的数据库中字符串大型对象可以容纳高达 4GB 的数据，有的数据库提供使用 TEXT 作为字符串大型对象数据类型。

（2）二进制大型对象（BLOB）：二进制大型对象保存大量的二进制数据，如图片、视频等二进制文件数据。

3.4 指定键

键是数据库的一种约束行为，它对于防止数据重复、保证数据的完整性是非常重要的，在定义表时可以指定键，这些键包括**候选键（CK）**、**主键（PK）**和**外键（FK）**。


微课视频

3.4.1 指定候选键

指定表的候选键使用 UNIQUE 关键字实现，语法有如下两种。

1. 在定义字段时指定

示例代码如下：

```
--   指定候选键
--   创建学生表语句
CREATE TABLE student(
   s_id     INTEGER,                    -- 学号
   s_name   VARCHAR(20),                -- 姓名
   gender   CHAR(1) ,                   -- 性别,'F'表示女,'M'表示男
   PIN      CHAR(18) UNIQUE             -- 身份证号码              ①
)
```

上述代码第①行定义 PIN 字段，可见在定义 PIN 字段后面使用 UNIQUE 关键字，这样就将 PIN 字段指定为候选键了。

2. 在 CREATE TABLE 语句结尾处添加 UNIQUE 子句指定

示例代码如下：

```
--   指定候选键
--   创建学生表语句
CREATE TABLE student(
   s_id       INTEGER,                  -- 学号
   s_name     VARCHAR(20),              -- 姓名
   gender     CHAR(1),                  -- 性别,'F'表示女,'M'表示男
   PIN        CHAR(18),                 -- 身份证号码
   UNIQUE     (PIN)                     -- 定义身份证号码为候选键      ①
)
```

上述代码第①行在 CREATE TABLE 语句结尾处添加 UNIQUE 子句(单独一行)，注意它与其他字段定义语句用逗号分隔。

学生表创建完成后，可以使用 MySQL Workbench 测试候选键，如图 3-4 所示。试图通过 INSERT 语句插入两条数据，注意它们的 PIN 字段数据(即 51 ********** 3)是重复的，则会引发违反候选键约束错误“Error Code：1062. Duplicate entry '51 ********** 3' for key 'student.PIN'”。

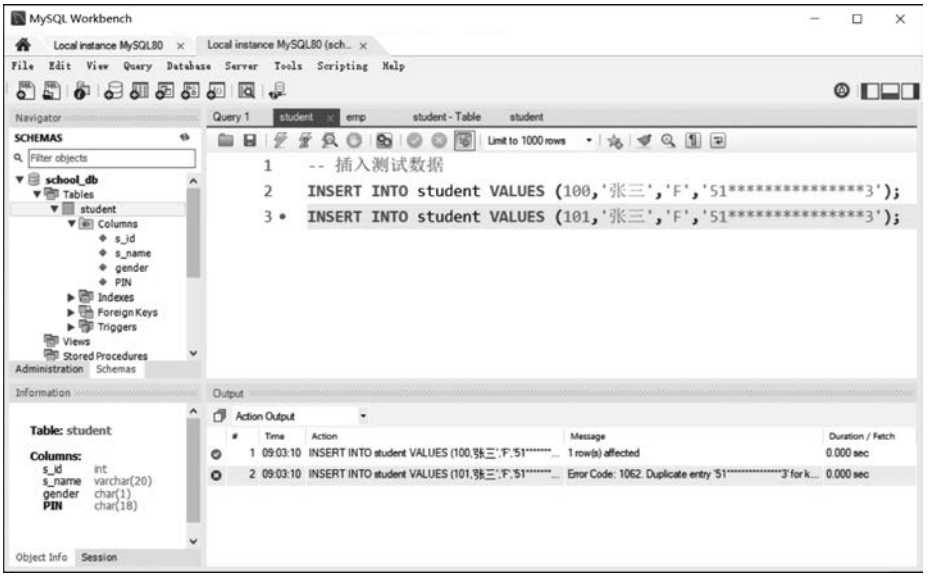

图 3-4　测试候选键

候选键可以是一个字段或多个字段的组合，上述示例介绍的是一个字段作为候选键的情况，下面再介绍多字段组合作为候选键的示例，该示例是创建一个学生成绩表(student_score)。学生成绩表的相关信息如表 3-2 所示。

表 3-2　学生成绩表的相关信息

字　段　名	数 据 类 型	长　　度	是否为候选键	备　　注
s_id	INTEGER		是	学号
c_id	INTEGER		是	课程编号
score	INTEGER		否	成绩

创建学生成绩表的示例代码如下：

```
--  指定多字段候选键
--  创建学生成绩表语句
CREATE TABLE student_score(
   s_id     INTEGER,                    -- 学号
   c_id     INTEGER,                    -- 课程编号
   score    INTEGER,                    -- 成绩
```

```
    UNIQUE  (s_id,c_id)                          -- 定义多字段组合候选键      ①
)
```

上述代码第①行指定 s_id 和 c_id 字段为组合候选键。

学生成绩表创建完成后，可以测试候选键，使用 MySQL Workbench 测试候选键，试图通过 INSERT 语句插入数据，如图 3-5 所示，如果候选键数据重复会引发错误“Error Code: 1062. Duplicate entry '100-2' for key 'student_score.s_id'”。

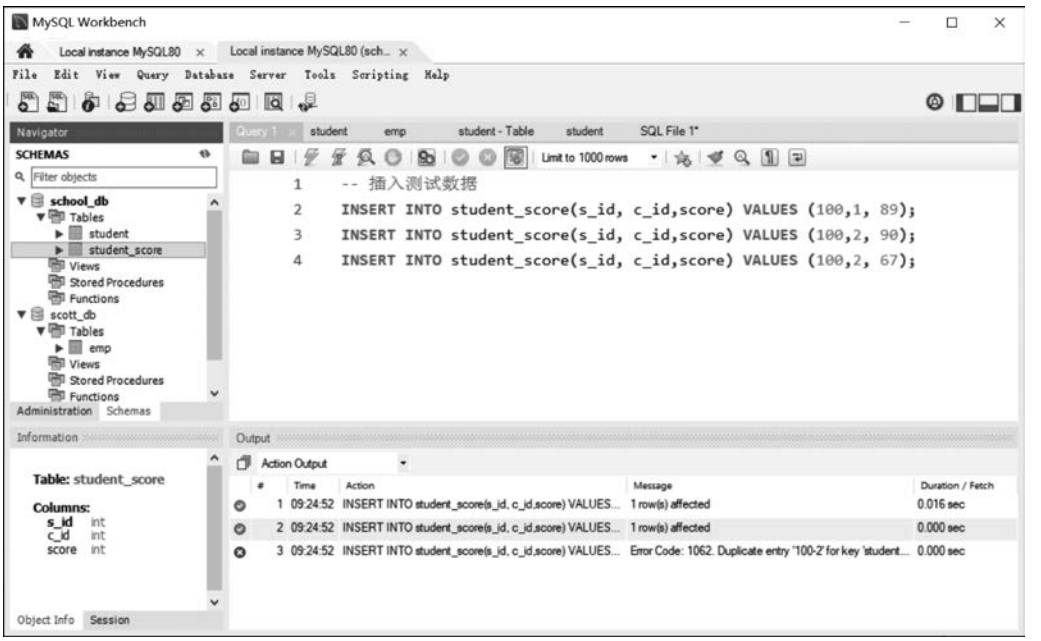

图 3-5 执行结果

3.4.2 指定主键

可以使用 **PRIMARY KEY** 关键字指定主键，它可以与 **UNIQUE** 关键字一起使用在 CREATE TABLE 语句中。指定主键的方法也有如下两种。

1. 定义字段时指定

示例代码如下：

```
--   指定主键
--   创建学生表语句
CREATE TABLE student(
   s_id     INTEGER PRIMARY KEY,              -- 学号          ①
   s_name   VARCHAR(20),

   s_name   VARCHAR(20),
   gender   CHAR(1),
   PIN      CHAR(18) UNIQUE
)
```

上述代码第①行定义 s_id 字段，可见在定义 s_id 字段后面使用 PRIMARY KEY 关键字，这样就将 s_id 字段指定为主键了。

2. 在 CREATE TABLE 语句结尾处添加 PRIMARY KEY 子句指定

示例代码如下：

```
--   指定主键
--   创建学生表语句
CREATE TABLE student (
   s_id     INTEGER ,              -- 学号
   s_name   VARCHAR(20),
   gender   CHAR(1),
   PIN   CHAR(18) UNIQUE,                              ①
   PRIMARY KEY(s_id)                                   ②
)
```

上述代码第①行指定候选键，代码第②行指定主键。主键和候选键都可以防止数据重复，读者可以参考候选键测试一下，这里不再赘述。

主键也可以是一个字段或多个字段的组合，修改学生成绩表，如表 3-3 所示，学生成绩表的主键是由 s_id 和 c_id 两个字段组合而成。

表 3-3　修改学生成绩表

字　段　名	数 据 类 型	长　　度	是否为主键	备　　注
s_id	INTEGER		是	学号
c_id	INTEGER		是	课程编号
score	INTEGER		否	成绩

创建学生成绩表代码如下：

```
--   指定主键
--   创建学生成绩表语句
CREATE TABLE student_score(
   s_id     INTEGER,                          -- 学号
   c_id     INTEGER,                          -- 课程编号
   score    INTEGER,                          -- 成绩
   PRIMARY KEY   (s_id,c_id)                  -- 定义多字段组合主键      ①
)
```

上述代码第①行指定 s_id 和 c_id 字段为主键。


微课视频

3.4.3　指定外键

指定外键使用 REFERENCES 关键字实现，将表 3-3 所示的学生成绩表中的学号字段(s_id)引用到表 3-1 所示的学生表中的学号字段(s_id)。学生成绩表称为子表，学生表称为父表。

提示 这种表之间的外键关联关系，通过文字描述不够形象，在数据库设计中这种关系可以通过 ER(实体关系)图描述。如图 3-6 所示，学生成绩表有两个外键(学号、课程编号)，学生成绩表通过学号关联到学生表。另外，学生成绩表通过课程编号关联到课程表。

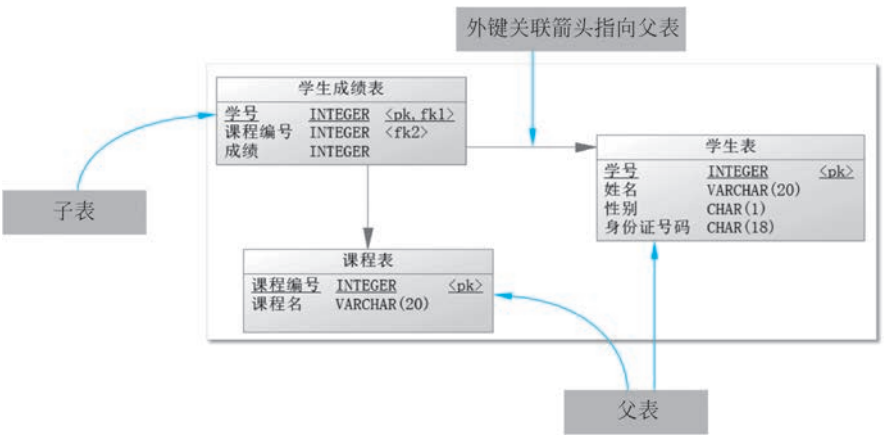

图 3-6 ER 图

指定外键的方法也有两种。

1. 在定义字段时通过 REFERENCES 关键字指定

示例代码如下：

```
--   指定外键
--   创建学生成绩表语句
CREATE TABLE student_score(
   s_id     INTEGER REFERENCES student(s_id),          -- 学号          ①
   c_id     INTEGER,                                   -- 课程编号
   score    INTEGER,                                   -- 成绩
   PRIMARY KEY  (s_id,c_id)
)
```

上述代码第①行定义 s_id 字段，可见在定义 s_id 字段时，后面使用 REFERENCES 关键字指定外键关联的父表以及字段，这里的 s_id 字段就是外键。

2. 在 CREATE TABLE 语句结尾处添加 FOREIGN KEY 子句指定

示例代码如下：

```
--   指定外键
--   创建学生成绩表语句
CREATE TABLE student_score(
   s_id     INTEGER ,                                  -- 学号
   c_id     INTEGER,                                   -- 课程编号
   score    INTEGER,                                   -- 成绩
```

```
    PRIMARY KEY  (s_id,c_id),
    FOREIGN KEY (s_id) REFERENCES student(s_id)                    ①
)
```

上述代码第①行是 FOREIGN KEY 子句，FOREIGN KEY 关键字后面(s_id)是指定表外键。


微课视频

3.5 其他约束

除了指定键约束外，表管理时还可以**指定默认值**、**禁止空值**和设置 **CHECK 约束**等。

3.5.1 指定默认值

在定义表时可以为字段指定默认值，使用 DEFAULT 关键字实现。例如，在定义学生表时，可以为性别字段设置默认值为'F'。示例代码如下：

```
--   创建学生表
--   指定默认值
CREATE TABLE student(
   s_id     INTEGER,              -- 学号
   s_name   VARCHAR(20),          -- 姓名
   gender   CHAR(1) DEFAULT 'F',  -- 性别,'F'表示女,'M'表示男,默认值为'F'        ①
    PIN     CHAR(18) UNIQUE       -- 身份证号码
)
```

上述代码第①行是为性别(gender)字段设置默认值'F','F'表示是女性。当没有给性别字段提供数据时，数据库系统会为其提供默认值'F'。

3.5.2 禁止空值

有时输入空值会引起严重的程序错误，在定义字段时，可以使用 NOT NULL 关键字设置字段禁止输入空值。

示例代码如下：

```
--   创建学生表
--   禁止空值
CREATE TABLE student(
   s_id   INTEGER,                              -- 学号
   s_name   VARCHAR(20) NOT NULL,               -- 姓名          ①
   gender   CHAR(1) DEFAULT 'F',                -- 性别,'F'表示女,'M'表示男,默认值为'F'
   PIN     CHAR(18) UNIQUE                      -- 身份证号码
)
```

上述代码第①行是为姓名(s_name)字段设置禁止空值。插入数据时，如果没有为姓名(s_name)字段提供数据，则会引发错误“Error Code：1364. Field 's_name' doesn't have a default value”。

3.5.3 设置 CHECK 约束

CHECK 关键字用来限制字段所能接收的数据。例如，在学生成绩表中可以限制成绩(score)字段的值为 0～100。示例代码如下：

```
--  指定外键
--  创建学生成绩表语句
CREATE TABLE student_score(
   s_id     INTEGER REFERENCES student(s_id),                    -- 学号
   c_id     INTEGER,                                             -- 课程编号
   score    INTEGER  CHECK (score >= 0 AND  score <= 100),       -- 成绩     ①
    PRIMARY KEY  (s_id,c_id)
)
```

上述代码第①行定义 score 字段时设置对该字段的限制，CHECK 关键字后面的表达式“(score >= 0 AND score <= 100)”是限制条件，其中>= 和<= 为条件运算符，AND 为逻辑运算符，表示“逻辑与”，类似的还有 OR 表示“逻辑或”，NOT 表示“逻辑非”。有关添加运算符和逻辑运算符将在第 7 章详细介绍。

学生成绩表创建完成后，可以测试 CHECK 约束，如图 3-7 所示。通过 INSERT 语句插入数据时，试图为 score 字段输入－20，则会引发“Error Code：1136. Column count doesn't match value count at row 1”错误。

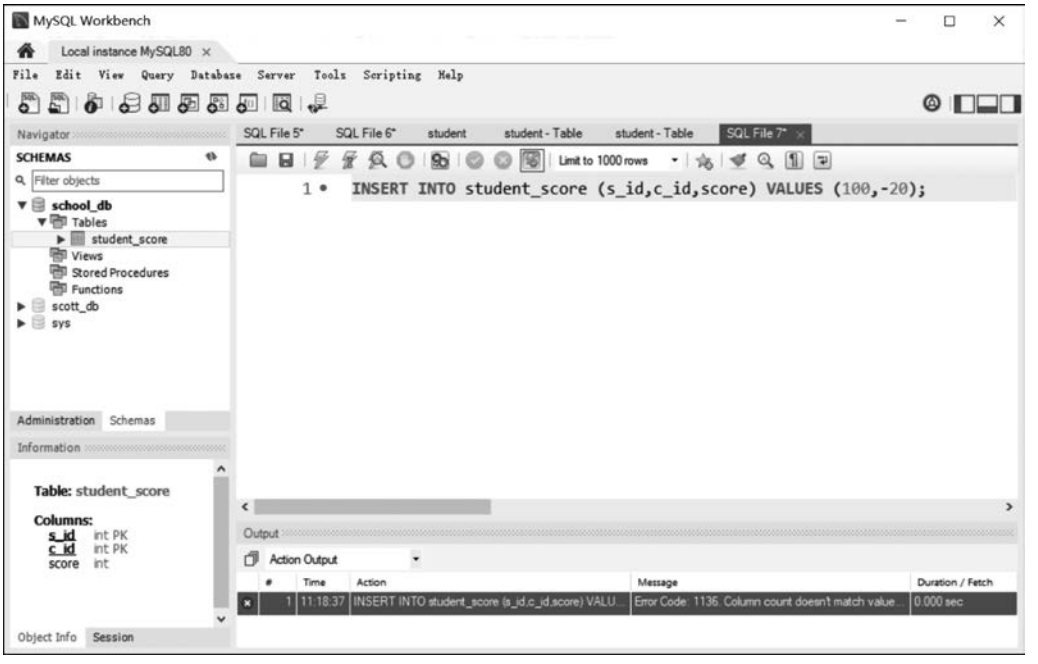

图 3-7 测试 CHECK 约束执行结果

微课视频

3.6 修改表

表建立后，由于某种原因需要修改表的结构或字段的定义，可以使用 ALTER TABLE 语句修改。下面介绍如何通过 ALTER TABLE 语句修改表名、添加字段和删除字段等。

3.6.1 修改表名

修改表名的 ALTER TABLE 语句，基本语法如下：

```
ALTER TABLE table_name
RENAME TO  new_table_name
```

其中，table_name 为要修改的表名；new_table_name 为修改后的表名。

注意 不同的数据库中 ALTER TABLE 语句有很大的不同，上述 ALTER TABLE 语句语法主要支持 Oracle 和 MySQL 数据库。

示例代码如下：

```
--  修改表名
--  将表名 student 修改为 stu_table
ALTER TABLE student RENAME TO stu_table;                ①
```

上述代码第①行将 student 表名修改为 stu_table，如图 3-8 所示。

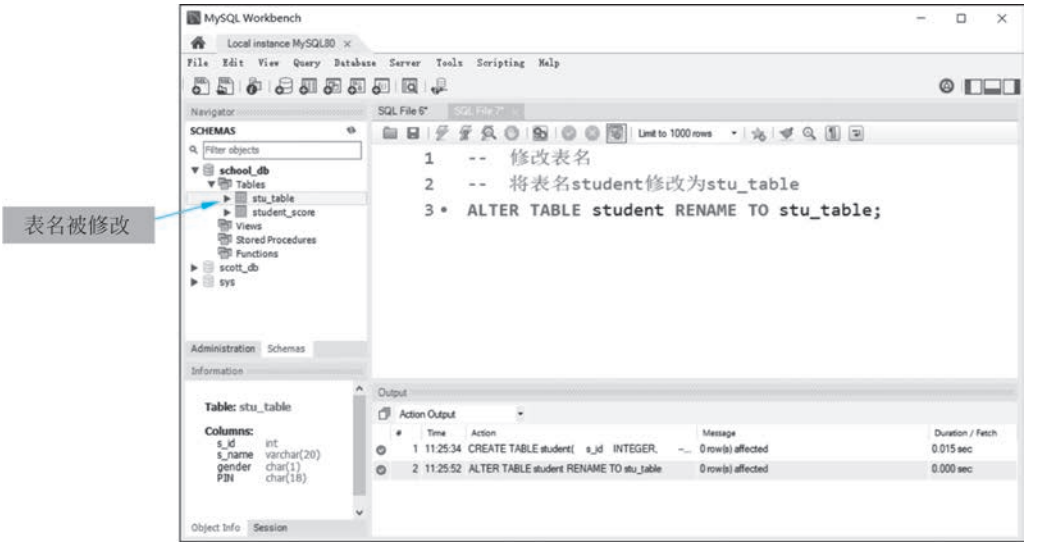

图 3-8 修改表名执行结果

3.6.2 添加字段

有时表已经创建好，甚至已经使用一段时间，并且表中已经有了一些数据，这时要在表中添加字段，如果删除表，再重新创建表代价很大。此时，可以使用 ALTER TABLE 中的 ADD 语句在现有表中添加字段，语法如下：

```
ALTER TABLE table_name ADD field_name datatype[(size)]
```

其中，table_name 为要修改的表名；field_name 为要添加的字段。

以下代码是在 student 表中添加两个字段。

```
--  在现有表中添加生日和电话字段
ALTER TABLE student ADD     birthday CHAR(10);          ①
ALTER TABLE student ADD     phone VARCHAR(20);          ②
```

上述代码第①行在 student 表中添加 birthday 字段，代码第②行在 student 表中添加 phone 字段。在 MySQL Workbench 中执行上述 SQL 语句，结果如图 3-9 所示。

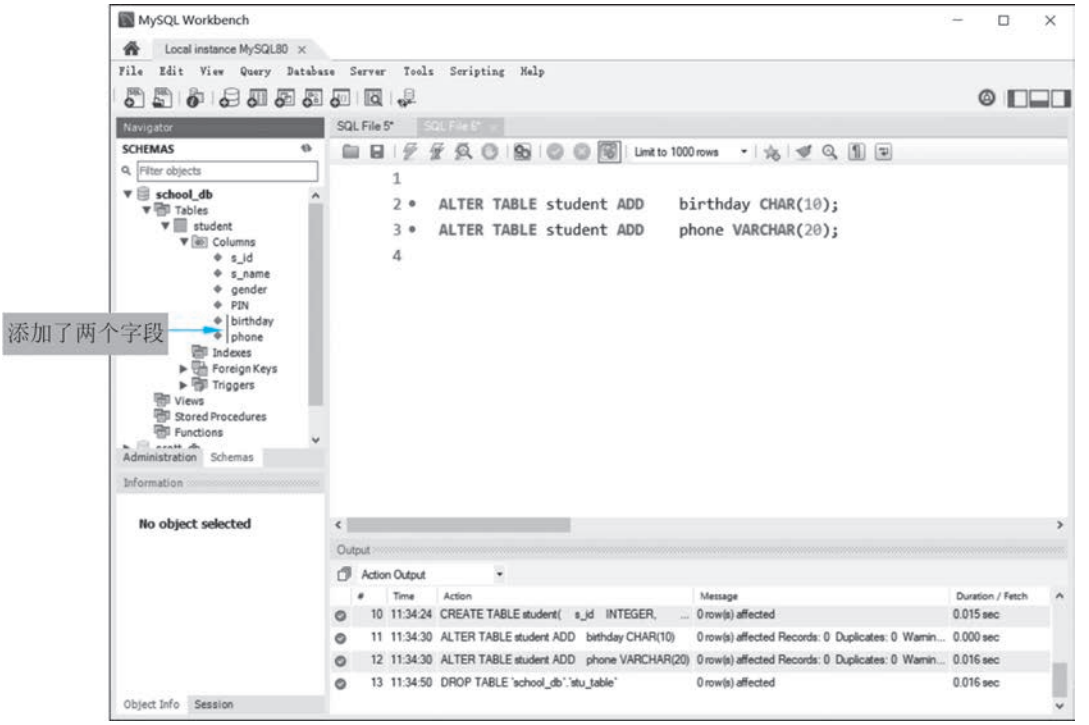

图 3-9 添加字段执行结果

3.6.3 删除字段

既然可以在现有表中添加字段，当然也可以在现有表中删除字段。可以使用 ALTER

TABLE 中的 DROP COLUMN 语句在现有表中删除字段，语法如下：

```
ALTER TABLE  table_name  DROP COLUMN field_name
```

以下代码是从 student 表中删除 birthday 字段。

```
--  在现有表中删除生日字段
ALTER TABLE student DROP COLUMN birthday;                    ①
```

上述代码第①行从 student 表中删除 birthday 字段，在 MySQL Workbench 中执行上述 SQL 语句，结果如图 3-10 所示。

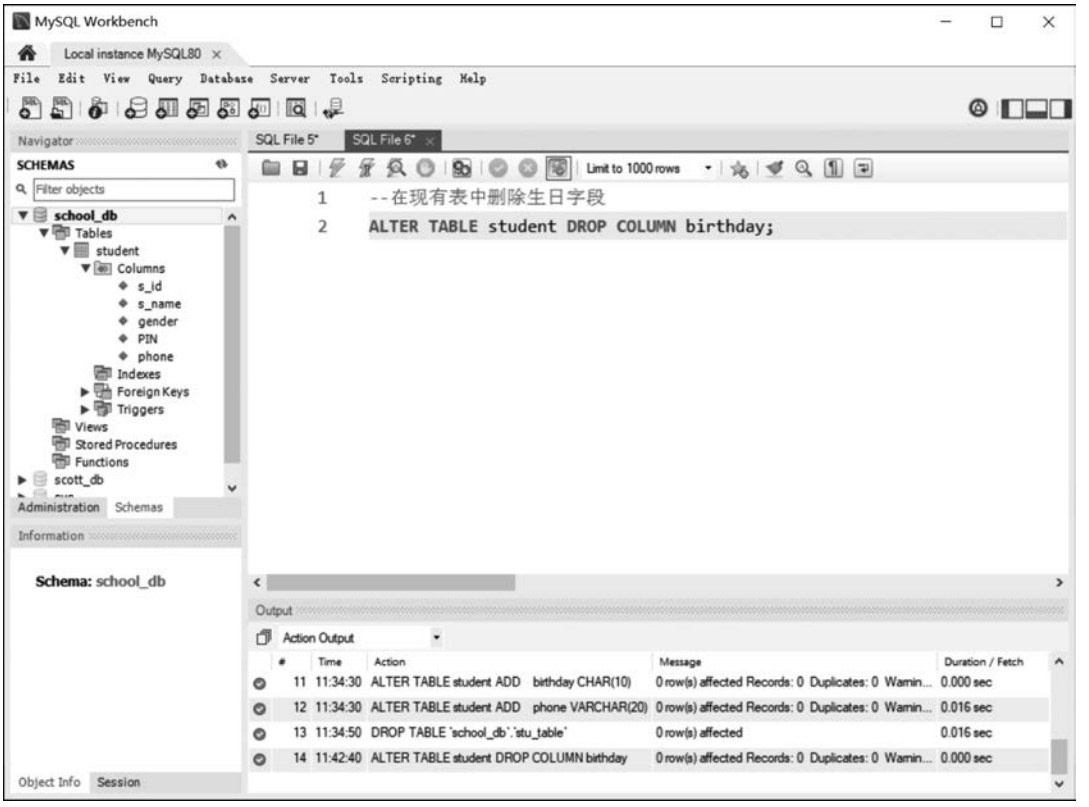

图 3-10　删除字段执行结果

3.7　删除表

通过 DROP TABLE 语句实现删除表，语法如下：

```
DROP TABLE [IF EXISTS] table_name
```

注意中括号[...]中的内容是可以省略的。

删除 student 表的示例代码如下：

```
--  删除学生表
DROP TABLE student;                                    ①
```

上述代码执行结果如图 3-11 所示，可见 student 表被删除了。

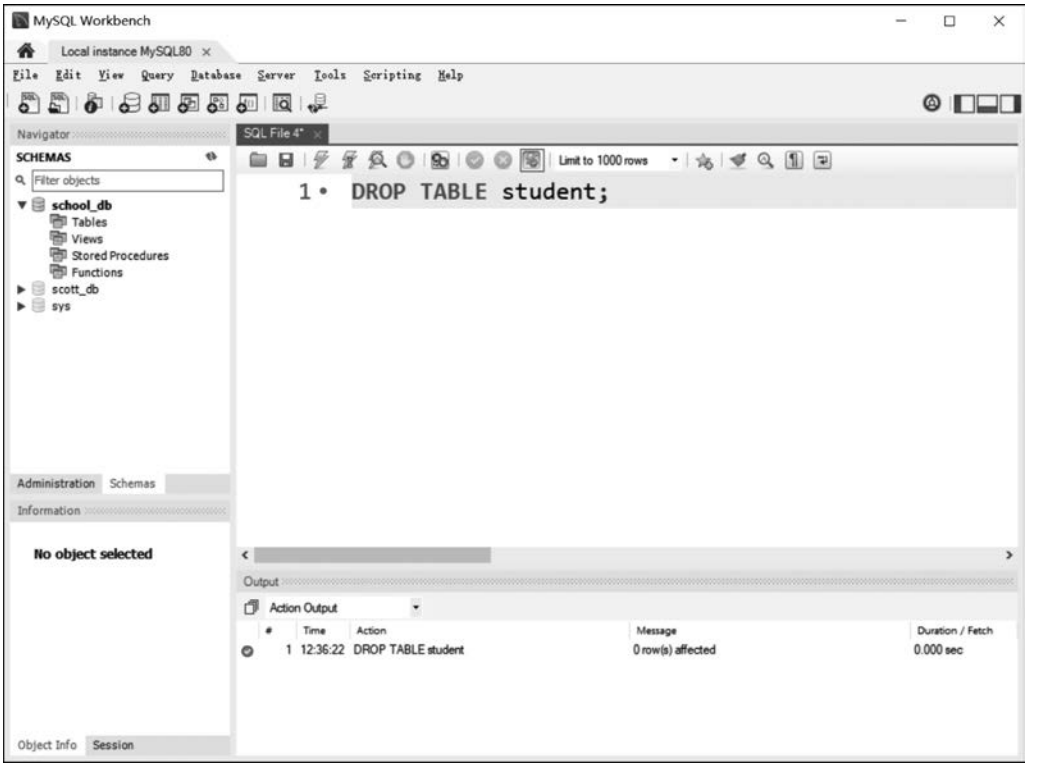

图 3-11 删除表执行结果

但是如果要删除的表不存在，则发生“Error Code：1051. Unknown table 'school_db.student'”错误。

为了防止错误发生，可以使用 IF EXISTS 子句判断表是否存在，示例代码如下：

```
DROP TABLE IF EXISTS student;
```

上述示例代码执行时，如果表不存在，则不会发生错误，但会发出警告：“1051 Unknown table 'school_db.student'”，如图 3-12 所示。

3.8 本章小结

本章重点介绍使用 SQL 创建表，其中包括为字段指定数据类型、指定键以及设置约束等，指定的键还可以细分为候选键、外键和主键。最后介绍修改表和删除表操作。

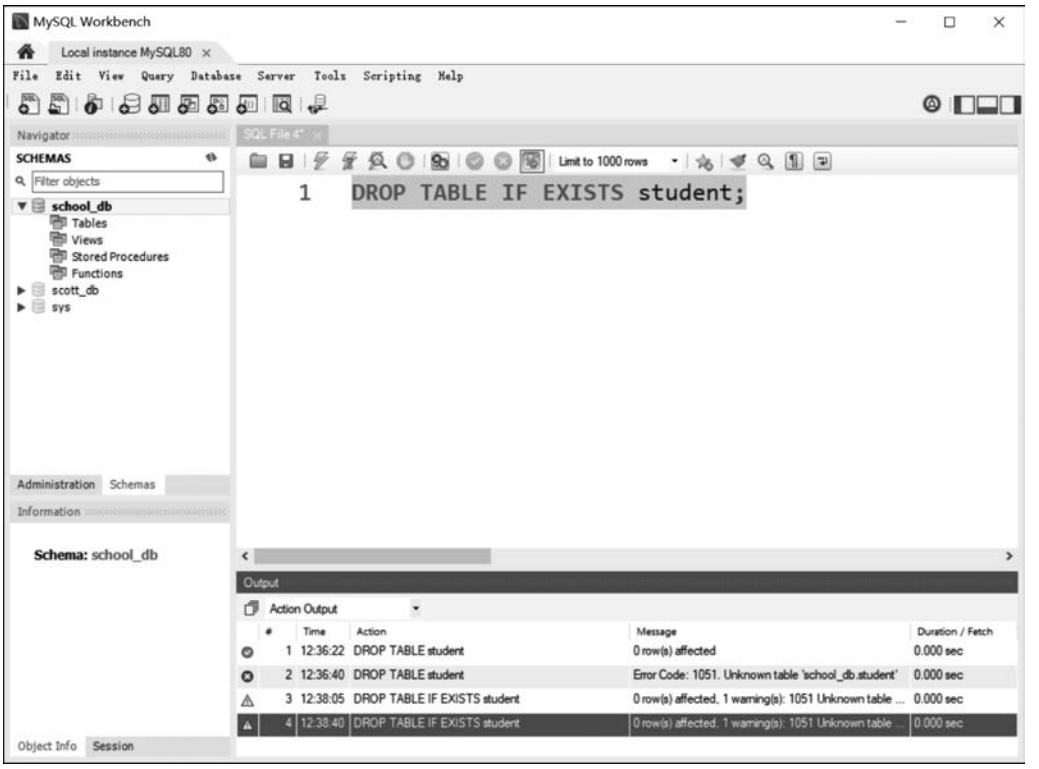

图 3-12　表不存在时的执行结果

3.9　同步练习

一、简述题

1. 简述什么是表记录和字段。

2. 简述主键、候选键和外键的区别。

二、操作题

1. 使用命令提示符工具登录 MySQL 数据库服务器，并创建 MyDB 数据库。

2. 使用命令提示符工具登录 MySQL 数据库服务器，并在 MyDB 数据库中创建 teacher 表。

三、选择题

下列哪些约束可以防止数据重复？（　　）

A. UNIQUE　　B. FOREIGN KEY　　C. PRIMARY KEY　　D. CHECK