

第 4 章

◀ 数据表的基本操作 ▶



学习目标 | Objective

在数据库中，数据表是数据库中最重要、最基本的操作对象，是数据存储的基本单位。数据表被定义为列的集合，数据在表中是按照行和列的格式来存储的。每一行代表一条唯一的记录，每一列代表记录中的一个域。

本章将详细介绍数据表的基本操作，主要内容包括：创建数据表、查看数据表结构、修改数据表、删除数据表。通过本章的学习，读者能够熟练掌握数据表的基本概念，理解约束、默认和规则的含义并且学会运用；能够在图形界面模式和命令行模式下熟练地完成有关数据表的常用操作。



内容导航 | Navigation

- 掌握如何创建数据表
- 掌握查看数据表结构的方法
- 掌握如何修改数据表
- 熟悉删除数据表的方法
- 熟练操作综合案例数据表的基本操作

4.1 创建数据表

在创建完数据库之后，接下来的工作就是创建数据表。所谓创建数据表，指的是在已经创建好的数据库中建立新表。创建数据表的过程是规定数据列的属性的过程，同时也是实施数据完整性（包括实体完整性、引用完整性和域完整性等）约束的过程。本节将介绍创建数据表的语法形式、如何添加主键约束、外键约束、非空约束等。

4.1.1 创建表的语法形式

数据表属于数据库，在创建数据表之前，应该使用语句“USE <数据库名>”指定操作是

在哪个数据库中进行, 如果没有选择数据库, 会抛出 “No database selected” 的错误。

创建数据表的语句为 CREATE TABLE, 语法规则如下:

```
CREATE TABLE <表名>
(
    字段名1, 数据类型 [列级别约束条件] [默认值],
    字段名2, 数据类型 [列级别约束条件] [默认值],
    .....
    [表级别约束条件]
);
```

使用 CREATE TABLE 创建表时, 必须指定以下信息:

(1) 要创建的表的名称, 不区分大小写, 不能使用 SQL 语言中的关键字, 如 DROP、ALTER、INSERT 等。

(2) 数据表中每一个列 (字段) 的名称和数据类型, 如果创建多个列, 要用逗号隔开。

【例 4.1】创建员工表 tb_emp1, 结构如表 4.1 所示。

表 4.1 tb_emp1 表结构

字段名称	数据类型	备注
id	INT(11)	员工编号
name	VARCHAR(25)	员工名称
deptId	INT(11)	所在部门编号
salary	FLOAT	工资

首先创建数据库, SQL 语句如下:

```
CREATE DATABASE test_db;
```

选择创建表的数据库, SQL 语句如下:

```
USE test_db;
```

创建 tb_emp1 表, SQL 语句为:

```
CREATE TABLE tb_emp1
(
    id      INT(11),
    name    VARCHAR(25),
    deptId  INT(11),
    salary  FLOAT
);
```

语句执行后, 便创建了一个名称为 tb_emp1 的数据表, 使用 SHOW TABLES; 语句查看数据表是否创建成功, SQL 语句如下:

```
mysql> SHOW TABLES;
+-----+
| Tables_in_test_db |
+-----+
| tb_emp1           |
+-----+
1 row in set (0.00 sec)
```

可以看到，test_db 数据库中已经有了数据表 tb_tmp1，数据表创建成功。

4.1.2 使用主键约束

主键，又称主码，是表中一列或多列的组合。主键约束（Primary Key Constraint）要求主键列的数据唯一，并且不允许为空。主键能够唯一地标识表中的一条记录，可以结合外键来定义不同数据表之间的关系，并且可以加快数据库查询的速度。主键和记录之间的关系如同身份证和人之间的关系，它们之间是一一对应的。主键分为两种类型：单字段主键和多字段联合主键。

1. 单字段主键

单字段主键由一个字段组成，SQL 语句格式分为以下两种情况。

(1) 在定义列的同时指定主键，语法规则如下：

字段名 数据类型 PRIMARY KEY [默认值]

【例 4.2】定义数据表 tb_emp 2，其主键为 id，SQL 语句如下：

```
CREATE TABLE tb_emp2
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(25),
  deptId  INT(11),
  salary  FLOAT
);
```

(2) 在定义完所有列之后指定主键。

[CONSTRAINT <约束名>] PRIMARY KEY [字段名]

【例 4.3】定义数据表 tb_emp 3，其主键为 id，SQL 语句如下：

```
CREATE TABLE tb_emp3
(
  id INT(11),
  name VARCHAR(25),
  deptId INT(11),
  salary FLOAT,
```

```
PRIMARY KEY(id)
);
```

上述两个例子执行后的结果是一样的，都会在 id 字段上设置主键约束。

2. 多字段联合主键

主键由多个字段联合组成，语法规则如下：

```
PRIMARY KEY [字段1, 字段2, . . . , 字段 n]
```

【例 4.4】定义数据表 tb_emp4，假设表中没有主键 id，为了唯一确定一个员工，可以把 name、deptId 联合起来做为主键，SQL 语句如下：

```
CREATE TABLE tb_emp4
(
  name VARCHAR(25),
  deptId INT(11),
  salary FLOAT,
  PRIMARY KEY(name, deptId)
);
```

语句执行后，便创建了一个名称为 tb_emp4 的数据表，name 字段和 deptId 字段组合在一起成为 tb_emp4 的多字段联合主键。

4.1.3 使用外键约束

外键用来在两个表的数据之间建立链接，它可以是一列或者多列。一个表可以有一个或多个外键。外键对应的是参照完整性，一个表的外键可以为空值，若不为空值，则每一个外键值必须等于另一个表中主键的某个值。

外键：首先它是表中的一个字段，它可以不是本表的主键，但对应另外一个表的主键。外键主要作用是保证数据引用的完整性，定义外键后，不允许删除在另一个表中具有关联关系的行。外键的作用是保持数据的一致性、完整性。例如，部门表 tb_dept 的主键是 id，在员工表 tb_emp5 中有一个键 deptId 与这个 id 关联。

主表（父表）：对于两个具有关联关系的表而言，相关联字段中主键所在的那个表即是主表。

从表（子表）：对于两个具有关联关系的表而言，相关联字段中外键所在的那个表即是从表。

创建外键的语法规则如下：

```
[CONSTRAINT <外键名>] FOREIGN KEY 字段名1 [ , 字段名2, ...]
REFERENCES <主表名> 主键列1 [ , 主键列2, ...]
```

“外键名”为定义的外键约束的名称，一个表中不能有相同名称的外键；“字段名”表示子表需要添加外键约束的字段列；“主表名”即被子表外键所依赖的表的名称；“主键列”表

示主表中定义的主键列，或者列组合。

【例 4.5】定义数据表 tb_emp5，并在 tb_emp5 表上创建外键约束。

创建一个部门表 tb_dept1，表结构如表 4.2 所示，SQL 语句如下：

```
CREATE TABLE tb_dept1
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(22) NOT NULL,
  location VARCHAR(50)
);
```

表 4.2 tb_dept1 表结构

字段名称	数据类型	备注
id	INT(11)	部门编号
name	VARCHAR(22)	部门名称
location	VARCHAR(50)	部门位置

定义数据表 tb_emp5，让它的键 deptId 作为外键关联到 tb_dept1 的主键 id，SQL 语句为：

```
CREATE TABLE tb_emp5
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(25),
  deptId  INT(11),
  salary  FLOAT,
  CONSTRAINT fk_emp_dept1 FOREIGN KEY(deptId) REFERENCES tb_dept1(id)
);
```

以上语句执行成功之后，在表 tb_emp5 上添加了名称为 fk_emp_dept1 的外键约束，外键名称为 deptId，其依赖于表 tb_dept1 的主键 id。

提示

关联指的是在关系型数据库中，相关表之间的联系。它是通过相容或相同的属性或属性组来表示的。子表的外键必须关联父表的主键，且关联字段的数据类型必须匹配，如果类型不一样，则创建子表时，就会出现错误“ERROR 1005 (HY000): Can't create table 'database.tablename'(errno: 150)”。

4.1.4 使用非空约束

非空约束（Not Null Constraint）指字段的值不能为空。对于使用了非空约束的字段，如果用户在添加数据时没有指定值，数据库系统会报错。

非空约束的语法规则如下：

字段名 数据类型 not null

【例 4.6】定义数据表 tb_emp6，指定员工的名称不能为空，SQL 语句如下：

```
CREATE TABLE tb_emp6
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(25) NOT NULL,
  deptId  INT(11),
  salary  FLOAT
);
```

执行后，在 tb_emp6 中创建了一个 Name 字段，其插入值不能为空（NOT NULL）。

4.1.5 使用唯一性约束

唯一性约束（Unique Constraint）要求该列唯一，允许为空，但只能出现一个空值。唯一约束可以确保一列或者几列不出现重复值。

唯一性约束的语法规则如下：

（1）在定义完列之后直接指定唯一约束，语法规则如下：

字段名 数据类型 UNIQUE

【例 4.7】定义数据表 tb_dept2，指定部门的名称唯一，SQL 语句如下：

```
CREATE TABLE tb_dept2
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(22) UNIQUE,
  location VARCHAR(50)
);
```

（2）在定义完所有列之后指定唯一约束，语法规则如下：

[CONSTRAINT <约束名>] UNIQUE(<字段名>)

【例 4.8】定义数据表 tb_dept3，指定部门的名称唯一，SQL 语句如下：

```
CREATE TABLE tb_dept3
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(22),
  location VARCHAR(50),
  CONSTRAINT STH UNIQUE(name)
);
```

UNIQUE 和 PRIMARY KEY 的区别：一个表中可以有多个字段声明为 UNIQUE，但只能有一个 PRIMARY KEY 声明；声明为 PRIMARY KEY 的列不允许有空值，但是声明为 UNIQUE 的字段允许空值（NULL）的存在。

4.1.6 使用默认约束

默认约束 (Default Constraint) 指定某列的默认值。如男性同学较多, 性别就可以默认为‘男’。如果插入一条新的记录时没有为这个字段赋值, 那么系统会自动为这个字段赋值为‘男’。

默认约束的语法规则如下:

字段名 数据类型 DEFAULT 默认值

【例 4.9】定义数据表 tb_emp7, 指定员工的部门编号默认为 1111, SQL 语句如下:

```
CREATE TABLE tb_emp7
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(25) NOT NULL,
  deptId  INT(11) DEFAULT 1111,
  salary  FLOAT
);
```

以上语句执行成功之后, 表 tb_emp7 上的字段 deptId 拥有了一个默认的值 1111, 新插入的记录如果没有指定部门编号, 则默认都为 1111。

4.1.7 设置表的属性值自动增加

在数据库应用中, 经常希望在每次插入新记录时, 系统自动生成字段的主键值。可以通过为表主键添加 AUTO_INCREMENT 关键字来实现。默认的, 在 MySQL 中 AUTO_INCREMENT 的初始值是 1, 每新增一条记录, 字段值自动加 1。一个表只能有一个字段使用 AUTO_INCREMENT 约束, 且该字段必须为主键的一部分。AUTO_INCREMENT 约束的字段可以是任何整数类型 (TINYINT、SMALLINT、INT、BIGINT 等)。

设置表的属性值自动增加的语法规则如下:

字段名 数据类型 AUTO_INCREMENT

【例 4.10】定义数据表 tb_emp8, 指定员工的编号自动递增, SQL 语句如下:

```
CREATE TABLE tb_emp8
(
  id      INT(11) PRIMARY KEY AUTO_INCREMENT,
  name    VARCHAR(25) NOT NULL,
  deptId  INT(11),
  salary  FLOAT
);
```

上述例子执行后, 会创建名称为 tb_emp8 的数据表。表 tb_emp8 中的 id 字段的值在添加记录的时候会自动增加, 在插入记录的时候, 默认的自增字段 id 的值从 1 开始, 每次添加一条新记录, 该值自动加 1。

例如, 执行如下插入语句:

```
mysql> INSERT INTO tb_emp8 (name,salary)
-> VALUES ('Lucy',1000), ('Lura',1200), ('Kevin',1500);
```

语句执行完后, tb_emp8 表中增加了 3 条记录, 在这里并没有输入 id 的值, 但系统已经自动添加该值, 使用 SELECT 命令查看记录, 如下所示。

```
mysql> SELECT * FROM tb_emp8;
+----+-----+-----+-----+
| id | name | deptId | salary |
+----+-----+-----+-----+
| 1 | Lucy | NULL | 1000 |
| 2 | Lura | NULL | 1200 |
| 3 | Kevin | NULL | 1500 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)
```

提示

这里使用 INSERT 声明向表中插入记录的方法, 并不是 SQL 的标准语法, 这种语法不一定被其他的数据库支持, 只能在 MySQL 中使用。

4.2 查看数据表结构

使用 SQL 语句创建好数据表之后, 可以查看表结构的定义, 以确认表的定义是否正确。在 MySQL 中, 查看表结构可以使用 DESCRIBE 和 SHOW CREATE TABLE 语句。本节将针对这两个语句分别进行详细的讲解。

4.2.1 查看表基本结构语句 DESCRIBE

DESCRIBE/DESC 语句可以查看表的字段信息, 其中包括: 字段名、字段数据类型、是否为主键、是否有默认值等。语法规则如下:

```
DESCRIBE 表名;
```

或者简写为:

```
DESC 表名;
```

【例 4.11】 分别使用 DESCRIBE 和 DESC 查看表 tb_dept1 和表 tb_emp1 的表结构。

查看 tb_dept1 表结构, SQL 语句如下:

```
mysql> DESCRIBE tb_dept1;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
name	varchar(22)	NO		NULL	
location	varchar(50)	YES		NULL	

查看 tb_emp1 表结构, SQL 语句如下:

```
mysql> DESC tb_emp1;
```

Field	Type	Null	Key	Default	Extra
id	int (11)	YES		NULL	
name	varchar(25)	YES		NULL	
deptId	int (11)	YES		NULL	
salary	float	YES		NULL	

其中, 各个字段的含义分别解释如下:

- NULL: 表示该列是否可以存储 NULL 值。
- Key: 表示该列是否已编制索引。PRI 表示该列是表主键的一部分; UNI 表示该列是 UNIQUE 索引的一部分; MUL 表示在列中某个给定值允许出现多次。
- Default: 表示该列是否有默认值, 如果有的话值是多少。
- Extra: 表示可以获取的与给定列有关的附加信息, 例如 AUTO_INCREMENT 等。

4.2.2 查看表详细结构语句 SHOW CREATE TABLE

SHOW CREATE TABLE 语句可以用来显示创建表时的 CREATE TABLE 语句, 语法格式如下:

```
SHOW CREATE TABLE <表名\G>;
```

提示

使用 SHOW CREATE TABLE 语句, 不仅可以查看表创建时候的详细语句, 而且还可以查看存储引擎和字符编码。

如果不加 ‘G’ 参数, 显示的结果可能非常混乱, 加上参数 ‘G’ 之后, 可使显示结果更加直观, 易于查看。

【例 4.12】使用 SHOW CREATE TABLE 查看表 tb_emp1 的详细信息, SQL 语句如下:

```
mysql> SHOW CREATE TABLE tb_emp1;
```

```

-----+
| Table | Create Table
+-----+-----+
-----+

```

```

-----+
| fruits | CREATE TABLE `fruits` (
  `f_id` char(10) NOT NULL,
  `s_id` int(11) NOT NULL,
  `f_name` char(255) NOT NULL,
  `f_price` decimal(8,2) NOT NULL,
  PRIMARY KEY (`f_id`),
  KEY `index_name` (`f_name`),
  KEY `index_id_price` (`f_id`,`f_price`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312 |
+-----+-----+
-----+

```

使用参数 ‘\G’ 之后的结果如下：

```

mysql> SHOW CREATE TABLE tb_emp1\G
*** 1. row ***
      Table: tb_emp1
Create Table: CREATE TABLE `tb_emp1` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)

```

4.3 修改数据表

修改表指的是修改数据库中已经存在的数据表的结构。MySQL 使用 `ALTER TABLE` 语句修改表。常用的修改表的操作有：修改表名、修改字段数据类型或字段名、增加和删除字段、修改字段的排列位置、更改表的存储引擎、删除表的外键约束等。本节将对和修改表有关的操

作进行讲解。

4.3.1 修改表名

MySQL 是通过 ALTER TABLE 语句来实现表名的修改的，具体的语法规则如下：

```
ALTER TABLE <旧表名> RENAME [TO] <新表名>;
```

其中 TO 为可选参数，使用与否均不影响结果。

【例 4.13】将数据表 tb_dept3 改名为 tb_deptment3。

执行修改表名操作之前，使用 SHOW TABLES 查看数据库中所有的表。

```
mysql> SHOW TABLES;
+-----+
| Tables_in_test_db |
+-----+
| tb_dept1          |
| tb_dept2          |
| tb_dept3          |
```

使用 ALTER TABLE 将表 tb_dept3 改名为 tb_deptment3，SQL 语句如下：

```
ALTER TABLE tb_dept3 RENAME tb_deptment3;
```

语句执行之后，检验表 tb_dept3 是否改名成功。使用 SHOW TABLES 查看数据库中的表，结果如下：

```
mysql> SHOW TABLES;
+-----+
| Tables_in_test_db |
+-----+
| tb_dept1          |
| tb_dept2          |
| tb_deptment3      |
```

经过比较可以看到，数据表列表中已经有了名称为 tb_deptment3 的表。

提示

读者可以在修改表名称时使用 DESC 命令查看修改前后两个表的结构，修改表名并不修改表的结构，因此修改名称后的表和修改名称前的表的结构必然是相同的。

4.3.2 修改字段的数据类型

修改字段的数据类型，就是把字段的数据类型转换成另一种数据类型。在 MySQL 中修改字段数据类型的语法规则如下：

```
ALTER TABLE <表名> MODIFY <字段名> <数据类型>
```

其中“表名”指要修改数据类型的字段所在表的名称，“字段名”指需要修改的字段，“数据类型”指修改后字段的新数据类型。

【例 4.14】将数据表 tb_dept1 中 name 字段的数据类型由 VARCHAR(22) 修改为 VARCHAR(30)。

执行修改表名操作之前，使用 DESC 查看 tb_dept1 表结构，结果如下：

```
mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)   | NO   | PRI | NULL    |       |
| name  | varchar(22) | YES  |     | NULL    |       |
| location | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

可以看到现在 name 字段的数据类型为 VARCHAR(22)，下面修改其类型。输入如下 SQL 语句并执行：

```
ALTER TABLE tb_dept1 MODIFY name VARCHAR(30);
```

再次使用 DESC 查看表，结果如下：

```
mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)   | NO   | PRI | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
| location | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

语句执行之后，检验会发现表 tb_dept1 表中 name 字段的数据类型已经修改成 VARCHAR(30)，修改成功。

4.3.3 修改字段名

MySQL 中修改表字段名的语法规则如下：

```
ALTER TABLE <表名> CHANGE <旧字段名> <新字段名> <新数据类型>;
```

其中，“旧字段名”指修改前的字段名；“新字段名”指修改后的字段名；“新数据类型”指修改后的数据类型，如果不需要修改字段的数据类型，可以将新数据类型设置成与原来一样。

即可，但数据类型不能为空。

【例 4.15】将数据表 tb_dept1 中的 location 字段名称改为 loc，数据类型保持不变，SQL 语句如下：

```
ALTER TABLE tb_dept1 CHANGE location loc VARCHAR(50);
```

使用 DESC 查看表 tb_dept1，会发现字段的名称已经修改成功，结果如下：

```
mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null    | Key    | Default    | Extra    |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO      | PRI    | NULL       |          |
| name       | varchar(30)   | YES     |        | NULL       |          |
| loc        | varchar(50)   | YES     |        | NULL       |          |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

【例 4.16】将数据表 tb_dept1 中的 loc 字段名称改为 location，同时将数据类型变为 VARCHAR(60)，SQL 语句如下：

```
ALTER TABLE tb_dept1 CHANGE loc location VARCHAR(60);
```

使用 DESC 查看表 tb_dept1，会发现字段的名称和数据类型均已经修改成功，结果如下：

```
mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null    | Key    | Default    | Extra    |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO      | PRI    | NULL       |          |
| name       | varchar(30)   | YES     |        | NULL       |          |
| location   | varchar(60)   | YES     |        | NULL       |          |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

提示

CHANGE 也可以只修改数据类型，实现和 MODIFY 同样的效果，方法是将 SQL 语句中的“新字段名”和“旧字段名”设置为相同的名称，只改变“数据类型”。

由于不同类型的数据在机器中存储的方式及长度并不相同，修改数据类型可能会影响到数据表中已有的数据记录。因此，当数据库表中已经有数据时，不要轻易修改数据类型。

4.3.4 添加字段

随着业务需求的变化，可能需要在已经存在的表中添加新的字段。一个完整字段包括字段名、数据类型、完整性约束。添加字段的语法格式如下：

```
ALTER TABLE <表名> ADD <新字段名> <数据类型>
```

[约束条件] [FIRST | AFTER 已存在字段名];

新字段名为需要添加的字段名称; “FIRST” 为可选参数, 其作用是将新添加的字段设置为表的第一个字段; “AFTER” 为可选参数, 其作用是将新添加的字段添加到指定的 “已存在字段名” 的后面。

提示

“FIRST” 或 “AFTER 已存在字段名” 用于指定新增字段在表中的位置, 如果 SQL 语句中没有这两个参数, 则默认将新添加的字段设置为数据表的最后列。

1. 添加无完整性约束条件的字段

【例 4.17】在数据表 tb_dept1 中添加一个没有完整性约束的 INT 类型的字段 managerId (部门经理编号), SQL 语句如下:

```
ALTER TABLE tb_dept1 ADD managerId INT(10);
```

使用 DESC 查看表 tb_dept1, 会发现在表的最后添加了一个名为 managerId 的 INT 类型的字段, 结果如下:

```
mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI  | NULL    |       |
| name       | varchar(30)   | YES  |      | NULL    |       |
| location   | varchar(60)   | YES  |      | NULL    |       |
| managerId  | int(10)       | YES  |      | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.03 sec)
```

2. 添加有完整性约束条件的字段

【例 4.18】在数据表 tb_dept1 中添加一个不能为空的 VARCHAR(12) 类型的字段 column1, SQL 语句如下:

```
ALTER TABLE tb_dept1 ADD column1 VARCHAR(12) not null;
```

使用 DESC 查看表 tb_dept1, 会发现在表的最后添加了一个名为 column1 的 VARCHAR(12) 类型且不为空的字段, 结果如下:

```
mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI  | NULL    |       |
| name       | varchar(30)   | YES  |      | NULL    |       |
| location   | varchar(60)   | YES  |      | NULL    |       |
| managerId  | int(10)       | YES  |      | NULL    |       |
| column1    | varchar(12)   | NO   |      |         |       |
+-----+-----+-----+-----+-----+-----+
```

```

| id      | int(11) | NO | PRI | NULL | |
| name    | varchar(30) | YES | | NULL | |
| location | varchar(60) | YES | | NULL | |
| managerId | int(10) | YES | | NULL | |
| column1 | varchar(12) | NO | | NULL | |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

3. 在表的第一列添加一个字段

【例 4.19】在数据表 tb_dept1 中添加一个 INT 类型的字段 column2, SQL 语句如下:

```
ALTER TABLE tb_dept1 ADD column2 INT(11) FIRST;
```

使用 DESC 查看表 tb_dept1, 会发现在表的第一列添加了一个名为 column2 的 INT(11) 类型字段, 结果如下:

```

mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| column2    | int(11)   | YES  |     | NULL    |       |
| id         | int(11)   | NO   | PRI | NULL    |       |
| name       | varchar(30) | YES  |     | NULL    |       |
| location   | varchar(60) | YES  |     | NULL    |       |
| managerId  | int(10)   | YES  |     | NULL    |       |
| column1    | varchar(12) | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

4. 在表的指定列之后添加一个字段

【例 4.20】在数据表 tb_dept1 中 name 列后添加一个 INT 类型的字段 column3, SQL 语句如下:

```
ALTER TABLE tb_dept1 ADD column3 INT(11) AFTER name;
```

使用 DESC 查看表 tb_dept1, 结果如下:

```

mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| column2    | int(11)   | YES  |     | NULL    |       |
| id         | int(11)   | NO   | PRI | NULL    |       |
| name       | varchar(30) | YES  |     | NULL    |       |
| column3    | int(11)   | YES  |     | NULL    |       |
| location   | varchar(60) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+

```

```
| managerId | int(10)      | YES | | NULL | |
| column1   | varchar(12)  | NO  | | NULL | |
+-----+-----+-----+-----+-----+
7 rows in set (0.03 sec)
```

可以看到, tb_dept1 表中增加了一个名称为 column3 的字段, 其位置在指定的 name 字段后面, 添加字段成功。

4.3.5 删除字段

删除字段是将数据表中的某个字段从表中移除, 语法格式如下:

```
ALTER TABLE <表名> DROP <字段名>;
```

“字段名”指需要从表中删除的字段名称。

【例 4.21】删除数据表 tb_dept1 表中的 column2 字段。

首先, 执行删除字段之前, 使用 DESC 查看 tb_dept1 表结构, 结果如下:

```
mysql> DESC tb_dept1;
```

```
+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extr |
+-----+-----+-----+-----+-----+
| column2    | int(11)       | YES  |     | NULL    |      |
| id         | int(11)       | NO   | PRI | NULL    |      |
| name       | varchar(30)   | YES  |     | NULL    |      |
| column3    | int(11)       | YES  |     | NULL    |      |
| location   | varchar(60)   | YES  |     | NULL    |      |
| managerId  | int(10)       | YES  |     | NULL    |      |
| column1    | varchar(12)   | NO   |     | NULL    |      |
+-----+-----+-----+-----+-----+
7 rows in set (0.03 sec)
```

删除 column2 字段, SQL 语句如下:

```
ALTER TABLE tb_dept1 DROP column2;
```

再次使用 DESC 查看表 tb_dept1, 结果如下:

```
mysql> DESC tb_dept1;
```

```
+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extr |
+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI | NULL    |      |
| name       | varchar(30)   | YES  |     | NULL    |      |
| column3    | int(11)       | YES  |     | NULL    |      |
| location   | varchar(60)   | YES  |     | NULL    |      |
| managerId  | int(10)       | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+
```

```
| column1 | varchar(12) | NO | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.03 sec)
```

可以看到, tb_dept1 表中已经不存在名称为 column2 的字段, 删除字段成功。

4.3.6 修改字段的排列位置

对于一个数据表来说, 在创建的时候, 字段在表中的排列顺序就已经确定了。但表的结构并不是完全不可以改变的, 可以通过 ALTER TABLE 来改变表中字段的相对位置。语法格式如下:

```
ALTER TABLE <表名> MODIFY <字段1> <数据类型> FIRST|AFTER <字段2>;
```

“字段 1”指要修改位置的字段, “数据类型”指“字段 1”的数据类型, “FIRST”为可选参数, 指将“字段 1”修改为表的第一个字段, “AFTER 字段 2”指将“字段 1”插入到“字段 2”后面。

1. 修改字段为表的第一个字段

【例 4.22】将数据表 tb_dept1 中的 column1 字段修改为表的第一个字段, SQL 语句如下:

```
ALTER TABLE tb_dept1 MODIFY column1 VARCHAR(12) FIRST;
```

使用 DESC 查看表 tb_dept1, 发现字段 column1 已经被移至表的第一列, 结果如下:

```
mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| column1    | varchar(12)   | NO   |     | NULL    |       |
| id         | int(11)       | NO   | PRI | NULL    |       |
| name       | varchar(30)   | YES  |     | NULL    |       |
| column3    | int(11)       | YES  |     | NULL    |       |
| location   | varchar(60)   | YES  |     | NULL    |       |
| managerId  | int(10)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.03 sec)
```

2. 修改字段到表的指定列之后

【例 4.23】将数据表 tb_dept1 中的 column1 字段插入到 location 字段后面, SQL 语句如下:

```
ALTER TABLE tb_dept1 MODIFY column1 VARCHAR(12) AFTER location;
```

使用 DESC 查看表 tb_dept1, 结果如下:

```
mysql> DESC tb_dept1;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI | NULL    |       |
| name       | varchar(30)   | YES  |     | NULL    |       |
| column3    | int(11)       | YES  |     | NULL    |       |
| location   | varchar(60)   | YES  |     | NULL    |       |
| column1    | varchar(12)   | NO   |     | NULL    |       |
| managerId  | int(10)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
column3	int(11)	YES		NULL	
location	varchar(60)	YES		NULL	
column1	varchar(12)	NO		NULL	
managerId	int(10)	YES		NULL	

6 rows in set (0.03 sec)

可以看到, tb_dept1 表中的字段 column1 已经被移至 location 字段之后。

4.3.7 更改表的存储引擎

通过前面章节的学习,知道存储引擎是 MySQL 中的数据存储文件或者内存中时采用的不同技术实现。可以根据自己的需要,选择不同的引擎,甚至可以为每一张表选择不同的存储引擎。MySQL 中主要存储引擎有: MyISAM、InnoDB、MEMORY (HEAP)、BDB、FEDERATED 等。可以使用 SHOW ENGINES; 语句查看系统支持的存储引擎。表 4.3 列出了 5.5.13 版本的 MySQL 所支持的存储引擎。

表 4.3 MySQL 支持的存储引擎

引擎名	是否支持
FEDERATED	否
MRG MYISAM	是
MyISAM	是
BLACKHOLE	是
CSV	是
MEMORY	是
ARCHIVE	是
InnoDB	默认
PERFORMANCE_SCHEMA	是

更改表的存储引擎的语法格式如下:

```
ALTER TABLE <表名> ENGINE=<更改后的存储引擎名>;
```

【例 4.24】将数据表 tb_deptment3 的存储引擎修改为 MyISAM。

在修改存储引擎之前,先使用 SHOW CREATE TABLE 查看表 tb_deptment3 当前的存储引擎,结果如下。

```
mysql> SHOW CREATE TABLE tb_deptment3 \G
*** 1. row ***
Table: tb_deptment3
```

```
Create Table: CREATE TABLE `tb_deptment3` (
  `id` int(11) NOT NULL,
  `name` varchar(22) DEFAULT NULL,
  `location` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `STH` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

可以看到，表 tb_deptment3 当前的存储引擎为 ENGINE=InnoDB，接下来修改存储引擎类型，输入如下 SQL 语句并执行：

```
mysql> ALTER TABLE tb_deptment3 ENGINE=MyISAM;
```

使用 SHOW CREATE TABLE 再次查看表 tb_deptment3 的存储引擎，发现表 tb_dept 的存储引擎变成了“MyISAM”，结果如下：

```
mysql> SHOW CREATE TABLE tb_deptment3 \G
*** 1. row ***
      Table: tb_deptment3
Create Table: CREATE TABLE `tb_deptment3` (
  `id` int(11) NOT NULL,
  `name` varchar(22) DEFAULT NULL,
  `location` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `STH` (`name`)
) ENGINE=MyISAM DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

4.3.8 删除表的外键约束

对于数据库中定义的外键，如果不再需要，可以将其删除。外键一旦删除，就会解除主表和从表间的关联关系，MySQL 中删除外键的语法格式如下：

```
ALTER TABLE <表名> DROP FOREIGN KEY <外键约束名>
```

“外键约束名”指在定义表时 CONSTRAINT 关键字后面的参数，详细内容可参考 4.1.3 节的“使用外键约束”。

【例 4.25】删除数据表 tb_emp9 中的外键约束。

首先创建表 tb_emp9，创建外键 deptId 关联 tb_dept1 表的主键 id，SQL 语句如下：

```
CREATE TABLE tb_emp9
(
  id      INT(11) PRIMARY KEY,
```

```

name VARCHAR(25),
deptId INT(11),
salary FLOAT,
CONSTRAINT fk_emp_dept FOREIGN KEY (deptId) REFERENCES tb_dept1(id)
);

```

使用 SHOW CREATE TABLE 查看表 tb_emp9 的结构, 结果如下:

```

mysql> SHOW CREATE TABLE tb_emp9 \G
*** 1. row ***
      Table: tb_emp9
Create Table: CREATE TABLE `tb_emp9` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_emp_dept` (`deptId`),
  CONSTRAINT `fk_emp_dept` FOREIGN KEY (`deptId`) REFERENCES `tb_dept1` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)

```

可以看到, 已经成功添加了表的外键, 下面删除外键约束, SQL 语句如下:

```
ALTER TABLE tb_emp9 DROP FOREIGN KEY fk_emp_dept;
```

执行完毕之后, 将删除表 tb_emp9 的外键约束, 使用 SHOW CREATE TABLE 再次查看表 tb_emp9 结构, 结果如下:

```

mysql> SHOW CREATE TABLE tb_emp9 \G
*** 1. row ***
      Table: tb_emp9
Create Table: CREATE TABLE `tb_emp9` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_emp_dept` (`deptId`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)

```

可以看到, tb_emp9 中已经不存在 FOREIGN KEY, 原有的名称为 fk_emp_dept 的外键约束删除成功。

4.4 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，在删除表的同时，表的定义和表中所有的数据均会被删除。因此，在进行删除操作前，最好对表中的数据做个备份，以免造成无法挽回的后果。本节将详细讲解数据库表的删除方法。

4.4.1 删除没有被关联的表

在 MySQL 中，使用 DROP TABLE 可以一次删除一个或多个没有被其他表关联的数据表。语法格式如下：

```
DROP TABLE [IF EXISTS] 表1, 表2, ... 表 n;
```

其中“表 n”指要删除的表的名称，后面可以同时删除多个表，只需将要删除的表名依次写在后面，相互之间用逗号隔开即可。如果要删除的数据表不存在，则 MySQL 会提示一条错误信息，“ERROR 1051 (42S02): Unknown table '表名’”。参数“IF EXISTS”用于在删除前判断删除的表是否存在，加上该参数后，再删除表的时候，如果表不存在，SQL 语句可以顺利执行，但是会发出警告（warning）。

在前面的例子中，已经创建了名为 tb_dept2 的数据表。如果没有，读者可输入语句，创建该表，SQL 语句如例 4.8 所示。下面使用删除语句将该表删除。

【例 4.26】删除数据表 tb_dept2，SQL 语句如下：

```
DROP TABLE IF EXISTS tb_dept2;
```

语句执行完毕之后，使用 SHOW TABLES 命令查看当前数据库中所有的表，SQL 语句如下：

```
mysql> SHOW TABLES;
```

```
+-----+
| Tables_in_test_db |
+-----+
| tb_dept1          |
| tb_deptment3      |
```

执行结果可以看到，数据表列表中已经不存在名称为 tb_dept2 的表，删除操作成功。

4.4.2 删除被其他表关联的主表

数据表之间存在外键关联的情况下，如果直接删除父表，结果会显示失败。原因是直接删除，将破坏表的参照完整性。如果必须要删除，可以先删除与它关联的子表，再删除父表，只是这样同时删除了两个表中的数据。但有的情况下可能要保留子表，这时如要单独删除父表，

只需将关联的表的外键约束条件取消, 然后就可以删除父表, 下面讲解这种方法。

在数据库中创建两个关联表, 首先, 创建表 tb_dept2, SQL 语句如下:

```
CREATE TABLE tb_dept2
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(22),
  location VARCHAR(50)
);
```

接下来创建表 tb_emp, SQL 语句如下:

```
CREATE TABLE tb_emp
(
  id      INT(11) PRIMARY KEY,
  name    VARCHAR(25),
  deptId  INT(11),
  salary  FLOAT,
  CONSTRAINT fk_emp_dept FOREIGN KEY (deptId) REFERENCES tb_dept2(id)
);
```

使用 SHOW CREATE TABLE 命令查看表 tb_emp 的外键约束, 结果如下:

```
mysql> SHOW CREATE TABLE tb_emp\G
*** 1. row ***
      Table: tb_emp
Create Table: CREATE TABLE `tb_emp` (
  `id` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `deptId` int(11) DEFAULT NULL,
  `salary` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_emp_dept` (`deptId`),
  CONSTRAINT `fk_emp_dept` FOREIGN KEY (`deptId`) REFERENCES `tb_dept2` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=gb2312
1 row in set (0.00 sec)
```

可以看到, 以上执行结果创建了两个关联表 tb_dept2 和表 tb_emp, 其中 tb_emp 表为子表, 具有名称为 fk_emp_dept 的外键约束, tb_dept2 为父表, 其主键 id 被子表 tb_emp 所关联。

【例 4.27】 删除被数据表 tb_emp 关联的数据表 tb_dept2。

首先直接删除父表 tb_dept2, 输入删除语句如下:

```
mysql> DROP TABLE tb_dept2;
ERROR 1217 (23000): Cannot delete or update a parent row: a foreign key constraint
```

fails

可以看到,如前所述,在存在外键约束时,主表不能被直接删除。

接下来,解除关联子表 tb_emp 的外键约束,SQL 语句如下:

```
ALTER TABLE tb_emp DROP FOREIGN KEY fk_emp_dept;
```

语句成功执行后,将取消表 tb_emp 和表 tb_dept2 之间的关联关系,此时,可以输入删除语句,将原来的父表 tb_dept2 删除,SQL 语句如下:

```
DROP TABLE tb_dept2;
```

最后通过 SHOW TABLES;查看数据表列表,如下所示:

```
mysql> show tables;
+-----+
| Tables_in_test_db |
+-----+
| tb_dept1          |
| tb_deptment3      |
.....省略部分内容
```

可以看到,数据表列表中已经不存在名称为 tb_dept2 的表。

4.5 综合案例——数据表的基本操作

本章全面介绍了 MySQL 中数据表的各种操作,如创建表、添加各类约束、查看表结构,以及修改和删除表。读者应该掌握这些基本的操作,为以后的学习打下坚实的基础。在这里,给出一个综合案例,让读者全面回顾一下本章的知识要点,并通过这些操作来检验自己是否已经掌握了数据表的常用操作。

1. 案例目的

创建、修改和删除表,掌握数据表的基本操作。

创建数据库 company,按照表 4.4 和表 4.5 给出的表结构在 company 数据库中创建两个数据表 offices 和 employees,按照操作过程完成对数据表的基本操作。

表 4.4 offices 表结构

字段名	数据类型	主键	外键	非空	唯一	自增
officeCode	INT(10)	是	否	是	是	否
city	VARCHAR(50)	否	否	是	否	否
address	VARCHAR(50)	否	否	否	否	否
country	VARCHAR(50)	否	否	是	否	否
postalCode	VARCHAR(15)	否	否	否	是	否

表 4.5 employees 表结构

字段名	数据类型	主键	外键	非空	唯一	自增
employeeNumber	INT(11)	是	否	是	是	是
lastName	VARCHAR(50)	否	否	是	否	否
firstName	VARCHAR(50)	否	否	是	否	否
mobile	VARCHAR(25)	否	否	否	是	否
officeCode	INT(10)	否	是	是	否	否
jobTitle	VARCHAR(50)	否	否	是	否	否
birth	DATETIME	否	否	是	否	否
note	VARCHAR(255)	否	否	否	否	否
sex	VARCHAR(5)	否	否	否	否	否

2. 案例操作过程

步骤 01 登录 MySQL 数据库。

打开 Windows 命令行，输入登录用户名和密码：

```
C:\>mysql -h localhost -u root -p
Enter password: **
```

或者打开 MySQL 5.7 Command Line Client，只输入用户密码也可以登录。登录成功后显示如下信息：

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.10 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

登录成功，可以输入 SQL 语句进行操作。

步骤 02 创建数据库 company。

创建数据库 company 的语句如下：

```
mysql> CREATE DATABASE company;
Query OK, 1 row affected (0.00 sec)
```

结果显示创建成功, 在 company 数据库中创建表, 必须先选择该数据库, 输入语句如下:

```
mysql> USE company;
Database changed
```

结果显示选择数据库成功。

步骤 03 创建表 offices。

创建表 offices 的语句如下:

```
CREATE TABLE offices
(
officeCode INT(10) NOT NULL UNIQUE,
city       VARCHAR(50) NOT NULL,
address    VARCHAR(50)
country    VARCHAR(50) NOT NULL,
postalCode VARCHAR(15)
PRIMARY KEY (officeCode)
);
```

执行成功之后, 使用 SHOW TABLES;语句查看数据库中的表, 语句如下:

```
mysql> show tables;
+-----+
| Tables_in_company |
+-----+
| offices           |
+-----+
1 row in set (0.00 sec)
```

可以看到, 数据库中已经有了数据表 offices, 创建成功。

步骤 04 创建表 employees。

创建表 employees 的语句如下:

```
CREATE TABLE employees
(
employeeNumber INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
lastName       VARCHAR(50) NOT NULL,
firstName      VARCHAR(50) NOT NULL,
mobile         VARCHAR(25) UNIQUE,
officeCode     INT(10) NOT NULL,
jobTitle       VARCHAR(50) NOT NULL,
birth          DATETIME, NOT NULL
note           VARCHAR(255),
sex            VARCHAR(5),
```

```
CONSTRAINT office_fk FOREIGN KEY(officeCode) REFERENCES offices(officeCode)
);
```

执行成功之后, 使用 SHOW TABLES; 语句查看数据库中的表, 语句如下:

```
mysql> show tables;
+-----+
| Tables_in_company |
+-----+
| employees          |
| offices             |
+-----+
2 rows in set (0.00 sec)
```

可以看到, 现在数据库中已经创建好了 employees 和 offices 两个数据表。要检查表的结构是否按照要求创建, 可使用 DESC 分别查看两个表的结构, 如果语句正确, 则显示结果如下:

```
mysql> DESC offices;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| officeCode | int(10)   | NO   | PRI | NULL    |       |
| city       | varchar(50) | NO   |     | NULL    |       |
| address    | varchar(50) | NO   |     | NULL    |       |
| country    | varchar(50) | NO   |     | NULL    |       |
| postalCode | varchar(15) | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)
```

```
mysql> DESC employees;
```

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| employeeNumber | int(11)   | NO   | PRI | NULL    | auto_increment |
| lastName       | varchar(50) | NO   |     | NULL    |                 |
| firstName      | varchar(50) | NO   |     | NULL    |                 |
| mobile         | varchar(25) | NO   |     | NULL    |                 |
| officeCode     | int(10)   | NO   | MUL | NULL    |                 |
| jobTitle       | varchar(50) | NO   |     | NULL    |                 |
| birth          | datetime  | YES  |     | NULL    |                 |
| note           | varchar(255) | YES  |     | NULL    |                 |
| sex            | varchar(5) | YES  |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
```

```
-----+
9 rows in set (0.00 sec)
```

可以看到，两个表中字段分别满足表 4.2 和表 4.3 中要求的数据类型和约束类型。

步骤 05 将表 employees 的 mobile 字段修改到 officeCode 字段后面。

修改字段位置，需要用到 ALTER TABLE 语句，输入语句如下：

```
mysql> ALTER TABLE employees MODIFY mobile VARCHAR(25) AFTER officeCode;
Query OK, 0 rows affected (0.00 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

结果显示执行成功，使用 DESC 查看修改后的结果如下：

```
mysql>DESC employees;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| employeeNumber | int(11)       | NO   | PRI | NULL    | auto_increment |
| lastName       | varchar(50)   | NO   |     | NULL    |                |
| firstName      | varchar(50)   | NO   |     | NULL    |                |
| officeCode     | int(10)       | NO   | MUL | NULL    |                |
| mobile        | varchar(25) | NO  |     | NULL   |                |
| jobTitle       | varchar(50)   | NO   |     | NULL    |                |
| birth          | datetime      | YES  |     | NULL    |                |
| note           | varchar(255)  | YES  |     | NULL    |                |
| sex            | varchar(5)    | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

可以看到，mobile 字段已经插入到 officeCode 字段的后面。

步骤 06 将表 employees 的 birth 字段改名为 employee_birth。

修改字段名，需要用到 ALTER TABLE 语句，输入语句如下：

```
ALTER TABLE employees CHANGE birth employee_birth DATETIME;
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

结果显示执行成功，使用 DESC 查看修改后的结果如下：

```
mysql>DESC employees;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| employeeNumber | int(11)       | NO   | PRI | NULL    | auto_increment |
| lastName       | varchar(50)   | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
```

```

| firstName | varchar(50) | NO | | NULL |
| mobile    | varchar(25) | NO | | NULL |
| officeCode | int(10)      | NO | MUL | NULL |
| jobTitle  | varchar(50) | NO | | NULL |
| employee_birth | datetime | YES | | NULL |
| note      | varchar(255) | YES | | NULL |
| sex       | varchar(5)   | YES | | NULL |

```

```

+-----+
9 rows in set (0.00 sec)

```

可以看到,表中只有 employee_birth 字段,已经没有名称为 birth 的字段了,修改名称成功。

步骤 07 修改 sex 字段,数据类型为 CHAR(1),非空约束。

修改字段数据类型,需要用到 ALTER TABLE 语句,输入语句如下:

```

mysql>ALTER TABLE employees MODIFY sex CHAR(1) NOT NULL;
Query OK, 0 rows affected (0.00 sec)
Records: 0 Duplicates: 0 Warnings: 0

```

结果显示执行成功,使用 DESC 查看修改后的结果如下:

```

mysql>DESC employees;

```

```

+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+
| employeeNumber | int(11)       | NO   | PRI | NULL    | auto_increment |
| lastName       | varchar(50)   | NO   |     | NULL    |                |
| firstName      | varchar(50)   | NO   |     | NULL    |                |
| mobile         | varchar(25)   | NO   |     | NULL    |                |
| officeCode     | int(10)       | NO   | MUL | NULL    |                |
| jobTitle       | varchar(50)   | NO   |     | NULL    |                |
| employee_birth | datetime      | YES  |     | NULL    |                |
| note           | varchar(255)  | YES  |     | NULL    |                |
| sex           | char(1)     | NO  |     | NULL    |                |

```

```

+-----+
9 rows in set (0.00 sec)

```

执行结果可以看到,sex 字段的数据类型由前面的 VARCHAR(5)修改为 CHAR(1),且其 Null 列显示为 NO,表示该列不允许空值,修改成功。

步骤 08 删除字段 note。

删除字段,需要用到 ALTER TABLE 语句,输入语句如下:

```

mysql> ALTER TABLE employees DROP note;
Query OK, 0 rows affected (0.01 sec)

```

Records: 0 Duplicates: 0 Warnings: 0

结果显示执行语句成功, 使用 DESC employees; 查看语句执行后的结果:

```
mysql> desc employees;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| employeeNumber | int(11)       | NO   | PRI | NULL    | auto_increment |
| lastName       | varchar(50)   | NO   |     | NULL    |                |
| firstName      | varchar(50)   | NO   |     | NULL    |                |
| mobile         | varchar(25)   | NO   |     | NULL    |                |
| officeCode     | int(10)       | NO   | MUL | NULL    |                |
| jobTitle       | varchar(50)   | NO   |     | NULL    |                |
| employee_birth | datetime      | YES  |     | NULL    |                |
| sex            | char(1)       | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

可以看到, DESC 语句返回了 8 个列字段, note 字段已经不在表结构中, 删除字段成功。

步骤 09 增加字段名 favorite_activity, 数据类型为 VARCHAR(100)。

增加字段, 需要用到 ALTER TABLE 语句, 输入语句如下:

```
mysql> ALTER TABLE employees ADD favorite_activity VARCHAR(100);
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

结果显示执行语句成功, 使用 DESC employees; 查看语句执行后的结果:

```
mysql> desc employees;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| employeeNumber | int(11)       | NO   | PRI | NULL    | auto_increment |
| lastName       | varchar(50)   | NO   |     | NULL    |                |
| firstName      | varchar(50)   | NO   |     | NULL    |                |
| mobile         | varchar(25)   | NO   |     | NULL    |                |
| officeCode     | int(10)       | NO   | MUL | NULL    |                |
| jobTitle       | varchar(50)   | NO   |     | NULL    |                |
| employee_birth | datetime      | YES  |     | NULL    |                |
| sex            | char(1)       | NO   |     | NULL    |                |
| favorite_activity | varchar(100) | YES |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

可以看到, 数据表 employees 中增加了一个新的列 favorite_activity, 数据类型为

VARCHAR(100), 允许空值, 添加新字段成功。

步骤 10 删除表 offices。

在创建表 employees 时, 设置了表的外键, 该表关联了其父表的 officeCode 主键。如前面所述, 删除关联表时, 要先删除子表 employees 的外键约束, 才能删除父表。因此, 必须先删除 employees 表的外键约束。

(1) 删除 employees 表的外键约束, 输入如下语句:

```
mysql>ALTER TABLE employees DROP FOREIGN KEY office_fk;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

其中 office_fk 为 employees 表的外键约束的名称, 即创建外键约束时 CONSTRAINT 关键字后面的参数, 结果显示语句执行成功, 现在可以删除 offices 父表。

(2) 删除表 offices, 输入如下语句:

```
mysql>DROP TABLE offices;
Query OK, 0 rows affected (0.00 sec)
```

结果显示执行删除操作成功, 使用 SHOW TABLES; 语句查看数据库中的表, 结果如下:

```
mysql> show tables;
+-----+
| Tables_in_company |
+-----+
| employees          |
+-----+
1 row in set (0.00 sec)
```

可以看到, 数据库中已经没有名称为 offices 的表了, 删除表成功。

步骤 11 修改表 employees 存储引擎为 MyISAM。

修改表存储引擎, 需要用到 ALTER TABLE 语句, 输入语句如下:

```
mysql>ALTER TABLE employees ENGINE=MyISAM;
Query OK, 0 rows affected (0.01 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

结果显示执行修改存储引擎操作成功, 使用 SHOW CREATE TABLE 语句查看表结构, 结果如下:

```
mysql> show CREATE TABLE employees\G
*** 1. row ***
      Table: employees
  Create Table: CREATE TABLE `employees` (
```

```

`employeeNumber` int(11) NOT NULL AUTO_INCREMENT,
`lastName` varchar(50) NOT NULL,
`firstName` varchar(50) NOT NULL,
`officeCode` int(10) NOT NULL,
`mobile` varchar(25) DEFAULT NULL,
`jobTitle` varchar(50) NOT NULL,
`employee_birth` datetime DEFAULT NULL,
`sex` char(1) NOT NULL,
`favoriate_activity` varchar(100) DEFAULT NULL,
PRIMARY KEY (`employeeNumber`),
KEY `office_fk` (`officeCode`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

```

可以看到，倒数第2行中的ENGINE后面的参数已经修改为MyISAM，修改成功。

步骤 12 将表 employees 名称修改为 employees_info。

修改数据表名，需要用到 ALTER TABLE 语句，输入语句如下：

```

mysql> ALTER TABLE employees RENAME employees_info;
Query OK, 0 rows affected (0.00 sec)

```

结果显示执行语句成功，使用 SHOW TABLES;语句查看执行结果：

```

mysql> show tables;
+-----+
| Tables_in_company |
+-----+
| employees_info    |
+-----+
1 rows in set (0.00 sec)

```

可以看到数据库中已经没有名称为 employees 的数据表。

4.6 专家解惑

疑问 1：表删除操作须谨慎。

表删除操作将把表的定义和表中的数据一起删除，并且 MySQL 在执行删除操作时，不会有任何的确认信息提示，因此执行删除操作时，应当慎重。在删除表前，最好对表中的数据进行备份，这样当操作失误时，可以对数据进行恢复，以免造成无法挽回的后果。

同样地，在使用 ALTER TABLE 进行表的基本修改操作时，在执行操作过程之前，也应该确保对数据进行完整的备份，因为数据库的改变是无法撤销的，如果添加了一个不需要的字

段, 可以将其删除; 相同的, 如果删除了一个需要的列, 该列下面的所有数据都将会丢失。

疑问 2: 每一个表中都要有一个主键吗?

并不是每一个表中都需要主键, 如果多个表之间进行连接操作时, 需要用到主键。因此并不需要为每个表建立主键, 而且有些情况最好不使用主键。

疑问 3: 并不是每个表都可以任意选择存储引擎。

外键约束 (FOREIGN KEY) 不能跨引擎使用。MySQL 支持多种存储引擎, 每一个表都可以指定一个不同的存储引擎, 但是要注意: 外键约束是用来保证数据的参照完整性, 如果表之间需要关联外键, 却指定了不同的存储引擎, 这些表之间是不能创建外键约束的。所以说, 存储引擎的选择也不完全是随意的。

疑问 4: 带 AUTO_INCREMENT 约束的字段值是从 1 开始的吗?

默认的, 在 MySQL 中, AUTO_INCREMENT 的初始值是 1, 每新增一条记录, 字段值自动加 1。设置自增属性 (AUTO_INCREMENT) 的时候, 还可以指定第一条插入记录的自增字段的值, 这样新插入的记录自增字段值从初始值开始递增, 如在 tb_emp8 中插入第一条记录, 同时指定 id 值为 5, 则以后插入的记录 id 值就会从 6 开始往上增加。添加唯一性的主键约束时, 往往需要设置字段自动增加属性。

4.7 经典习题

1. 创建数据库 Market, 在 Market 中创建数据表 customers, customers 表结构如表 4.6 所示, 按要求进行操作。

表 4.6 customers 表结构

字段名	数据类型	主键	外键	非空	唯一	自增
c_num	INT(11)	是	否	是	是	是
c_name	VARCHAR(50)	否	否	否	否	否
c_contact	VARCHAR(50)	否	否	否	否	否
c_city	VARCHAR(50)	否	否	否	否	否
c_birth	DATETIME	否	否	是	否	否

- (1) 创建数据库 Market。
- (2) 创建数据表 customers, 在 c_num 字段上添加主键约束和自增约束, 在 c_birth 字段上添加非空约束。
- (3) 将 c_contact 字段插入到 c_birth 字段后面。
- (4) 将 c_name 字段数据类型改为 VARCHAR(70)。

- (5) 将 c_contact 字段改名为 c_phone。
- (6) 增加 c_gender 字段, 数据类型为 CHAR(1)。
- (7) 将表名修改为 customers_info。
- (8) 删除字段 c_city。
- (9) 修改数据表的存储引擎为 MyISAM。

2. 在 Market 中创建数据表 orders, orders 表结构如表 4.7 所示, 按要求进行操作。

表 4.7 orders 表结构

字段名	数据类型	主键	外键	非空	唯一	自增
o_num	INT(11)	是	否	是	是	是
o_date	DATE	否	否	否	否	否
c_id	VARCHAR(50)	否	是	否	否	否

(1) 创建数据表 orders, 在 o_num 字段上添加主键约束和自增约束, 在 c_id 字段上添加外键约束, 关联 customers 表中的主键 c_num。

(2) 删除 orders 表的外键约束, 然后删除表 customers。