第10章_创建和管理表

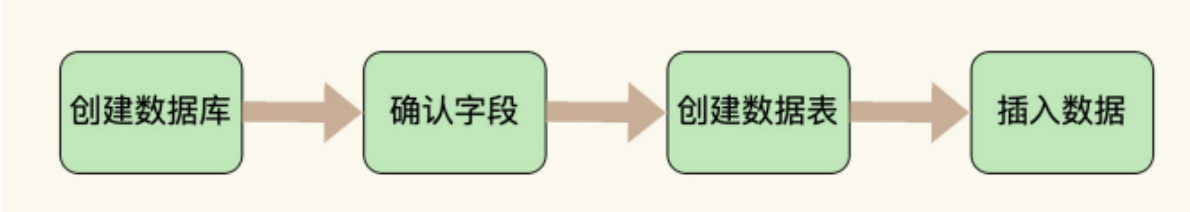
1. 基础知识

1.1 一条数据存储的过程

`存储数据是处理数据的第一步`。只有正确地把数据存储起来，我们才能进行有效的处理和分析。否则，只能是一团乱麻，无从下手。

那么，怎样才能把用户各种经营相关的、纷繁复杂的数据，有序、高效地存储起来呢？

在 MySQL 中，一个完整的数据存储过程总共有 4 步，分别是创建数据库、确认字段、创建数据表、插入数据。



我们要先创建一个数据库，而不是直接创建数据表呢？

因为从系统架构的层次上看，MySQL 数据库系统从大到小依次是`数据库服务器`、`数据库`、`数据表`、数据表的`行与列`。

MySQL 数据库服务器之前已经安装。所以，我们就从创建数据库开始

1.2 标识符命名规则

- 数据库名、表名不得超过30个字符，变量名限制为29个
- 必须只能包含 A–Z, a–z, 0–9, _共63个字符 **（数字字母下划线）**
- 数据库名、表名、字段名等对象名中间**不要包含空格**
- **同一个MySQL软件中，数据库不能同名；同一个库中，表不能重名；同一个表中，字段不能重名**
- 必须保证**你的字段**没有和**保留字、数据库系统或常用方法**冲突。如果坚持使用，请在SQL语句中使用``（着重号）引起来
- 保持字段名和类型的一致性：在命名字段并为其指定数据类型的时候一定要保证一致性，**假如数据类型在一个表里是整数，那在另一个表里可就别变成字符型了 （否则涉及隐式转换，效率低）**

2. 创建和管理数据库

2.1 创建数据库

- 方式1：创建数据库

> CREATE DATABASE 数据库名;

创建数据库

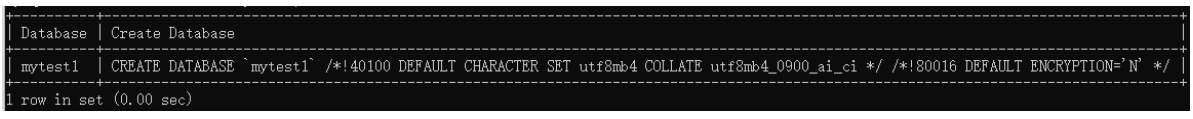
```
CREATE DATABASE mytest1; # 这里的数据库使用的是默认的字符集

Query OK, 1 row affected (0.01 sec)
```

查看所有的数据库

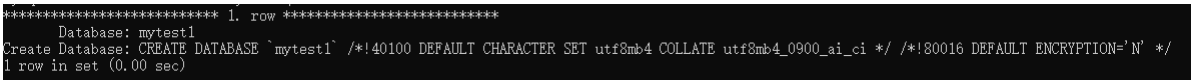
```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| atguigudb          |
| information_schema |
| mysql              |
| mytest1            |
| performance_schema |
| sys                |
+--------------------+
10 rows in set (0.01 sec)
```

查看创表语句

```
SHOW CREATE DATABASE mytest1;
```




```
SHOW CREATE DATABASE mytest1\G  -- 注意：大写
```



查看字符集

```
SHOW VARIABLES LIKE 'character_%';

+--------------------------+----------------------------------------------+
| Variable_name            | Value                                        |
+--------------------------+----------------------------------------------+
| character_set_client     | gbk                                          |
| character_set_connection | gbk                                          |
| character_set_database   | utf8mb4                                      | #
数据库默认的字符集
| character_set_filesystem | binary                                       |
| character_set_results    | gbk                                          |
| character_set_server     | utf8mb4                                      |
| character_set_system     | utf8mb3                                      |
| character_sets_dir       | D:\sofeware\Idea\Mysql8.0.26\share\charsets\ |
+--------------------------+----------------------------------------------+
8 rows in set, 1 warning (0.01 sec)
```

- 方式2：创建数据库并指定字符集

> CREATE DATABASE 数据库名 CHARACTER SET 字符集;

```
CREATE DATABASE mytest2 CHARACTER SET 'gbk'; #显式指明数据库的字符集

Query OK, 1 row affected (0.00 sec)
```

```
SHOW CREATE DATABASE mytest2;
```

```
+----------+-----------------------------------------------------------------------------------------------------------+
| Database | Create Database                                                                                           |
+----------+-----------------------------------------------------------------------------------------------------------+
| mytest2  | CREATE DATABASE `mytest2` /*!40100 DEFAULT CHARACTER SET gbk */ /*!80016 DEFAULT ENCRYPTION='N' */ |
+----------+-----------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

- 方式3：判断数据库是否已经存在，不存在则创建数据库（推荐）

> CREATE DATABASE IF NOT EXISTS 数据库名;

```
-- 查看所有的数据库
SHOW DATABASES;

+--------------------+
```

```
| Database           |
+--------------------+
| atguigudb          |
| information_schema |
| mysql              |
| mytest1            |
| mytest2            |
| performance_schema |
| sys                |
+--------------------+
11 rows in set (0.00 sec)



-- 执行建表语句（该表已经存在）
CREATE DATABASE mytest2 CHARACTER SET 'gbk';

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
'•CREATE DATABASE mytest2 CHARACTER SET 'gbk'' at line 1



-- 建表语句 如果数据库已经存在，则不会执行该建表语句
CREATE DATABASE IF NOT EXISTS mytest2 CHARACTER SET 'utf8';

Query OK, 1 row affected, 2 warnings (0.00 sec)


-- 查看建表语句

SHOW  CREATE DATABASE mytest2;
+----------+-----------------------------------------------------------------
----------------------------------+
| Database | Create Database
                                  |
+----------+-----------------------------------------------------------------
----------------------------------+
| mytest2  | CREATE DATABASE `mytest2` /*!40100 DEFAULT CHARACTER SET gbk */
/*!80016 DEFAULT ENCRYPTION='N' */ |
+----------+-----------------------------------------------------------------
----------------------------------+
1 row in set (0.00 sec)
```

如果MySQL中已经存在相关的数据库，则忽略创建语句，不再创建数据库。

2.2 使用数据库

查看当前所有的数据库

> SHOW DATABASES; #有一个S，代表多个数据库

查看当前正在使用的数据库

> SELECT DATABASE(); #使用的一个 mysql 中的全局函数

查看指定库下所有的表

> SHOW TABLES FROM 数据库名;

查看数据库的创建信息

> SHOW CREATE DATABASE 数据库名;
> 或者：
> SHOW CREATE DATABASE 数据库名\G

使用/切换数据库

> USE 数据库名;

> 注意：要操作表格和数据之前必须先说明是对哪个数据库进行操作，否则就要对所有对象加上“数据库名.”

2.3 修改数据库

一般是更改数据库字符集

> ALTER DATABASE 数据库名 CHARACTER SET 字符集; #比如：gbk、utf8等

更改字符集

```
ALTER DATABASE mytest2 CHARACTER SET 'utf8';

Query OK, 1 row affected, 1 warning (0.01 sec)
```

查看建表语句

```
SHOW CREATE DATABASE mytest2;
```

```
+----------+-----------------------------------------------------------------------------------------------------------------+
| Database | Create Database                                                                                                 |
+----------+-----------------------------------------------------------------------------------------------------------------+
| mytest2  | CREATE DATABASE `mytest2` /*!40100 DEFAULT CHARACTER SET utf8 */ /*!80016 DEFAULT ENCRYPTION='N' */ |
+----------+-----------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

注意：

DATABASE 不能改名。一些可视化工具可以改名，它是建新库，把所有表复制到新库，再删旧库完成的

2.4 删除数据库

- 方式1：删除指定的数据库

> DROP DATABASE 数据库名;

删除数据库

```
DROP DATABASE mytest1;

Query OK, 0 rows affected (0.01 sec)
```

查看所有数据库

```
SHOW DATABASES;

+--------------------+
| Database           |
+--------------------+
| atguigudb          |
| blog               |
| information_schema |
| javaweb            |
| mysql              |
| mytest2            |
| performance_schema |
| ssm                |
| ssmuser            |
| sys                |
+--------------------+
10 rows in set (0.00 sec)
```

删除数据库

```
DROP DATABASE mytest1;

ERROR 1008 (HY000): Can't drop database 'mytest1'; database doesn't exist
```

查看

- 方式2：删除指定的数据库（推荐）

> DROP DATABASE IF EXISTS 数据库名;

推荐的删除库的方式：

```
DROP DATABASE IF EXISTS mytest2;

Query OK, 0 rows affected (0.01 sec)
```

查看数据库

```
SHOW DATABASES;

+--------------------+
| Database           |
+--------------------+
| atguigudb          |
| blog               |
| information_schema |
| javaweb            |
| mysql              |
| performance_schema |
| ssm                |
| ssmuser            |
| sys                |
+--------------------+
9 rows in set (0.00 sec)
```

3. 创建表

3.1 创建方式1（白手起家）

- **必须具备：**
  - CREATE TABLE权限
  - 存储空间

- **语法格式：**

```
CREATE TABLE [IF NOT EXISTS] 表名(
    字段1, 数据类型 [约束条件] [默认值],
    字段2, 数据类型 [约束条件] [默认值],
    字段3, 数据类型 [约束条件] [默认值],
    ……
    [表约束条件]
);
```

> 加上了IF NOT EXISTS关键字，则表示：如果当前数据库中不存在要创建的数据表，则创建数据表；如果当前数据库中已经存在要创建的数据表，则忽略建表语句，不再创建数据表。

- **必须指定：**
  - 表名
  - 列名(或字段名)，数据类型，**长度**
- **可选指定：**
  - 约束条件
  - 默认值

MySQL中的数据类型

类型	类型举例
整数类型	TINYINT、SMALLINT、MEDIUMINT、**INT(或INTEGER)**、BIGINT
浮点类型	FLOAT、DOUBLE
定点数类型	**DECIMAL**（建议使用定点数）
位类型	BIT
日期时间类型	YEAR、TIME（时分秒）、**DATE（年月日）**、DATETIME（年月日+时分秒）、TIMESTAMP（年月日+时分秒）
文本字符串类型	CHAR、**VARCHAR**、TINYTEXT、TEXT、MEDIUMTEXT、LONGTEXT
枚举类型	ENUM
集合类型	SET
二进制字符串类型	（图片、音乐、视频） BINARY、VARBINARY、TINYBLOB、BLOB、MEDIUMBLOB、LONGBLOB
JSON类型	JSON对象、JSON数组
空间数据类型	单值：GEOMETRY、POINT、LINESTRING、POLYGON； 集合：MULTIPOINT、MULTILINESTRING、MULTIPOLYGON、GEOMETRYCOLLECTION

注意：

其中：

数值类型：整数类型、浮点类型、定点数类型、位类型

日期类型：日期时间类型

字符串类型：文本字符串类型、枚举类型、集合类型、二进制字符串类型、JSON类型

空间类型：空间数据类型

其中，常用的几类类型介绍如下：

数据类型	描述
INT	从-2^{31}到2^{31}-1的整型数据。存储大小为 4个字节
CHAR(size)	定长字符数据。若未指定，默认为1个字符，最大长度255
VARCHAR(size)	可变长字符数据，根据字符串实际长度保存，**必须指定长度**
FLOAT(M,D)	单精度，占用4个字节，M=整数位+小数位，D=小数位。 D<=M<=255,0<=D<=30，默认M+D<=6
DOUBLE(M,D)	双精度，占用8个字节，D<=M<=255,0<=D<=30，默认M+D<=15
DECIMAL(M,D)	高精度小数，占用M+2个字节，D<=M<=65，0<=D<=30，最大取值范围与DOUBLE相同。
DATE	日期型数据，格式'YYYY-MM-DD'
BLOB	二进制形式的长文本数据，最大可达4G
TEXT	长文本数据，最大可达4G

- 创建表举例1：

查看建库语句

```
USE atguigudb;

SHOW CREATE DATABASE atguigudb;

+----------+-----------------------------------------------------------------
-------------------------------------+
| Database  | Create Database
                                       |
+----------+-----------------------------------------------------------------
-------------------------------------+
| atguigudb | CREATE DATABASE `atguigudb` /*!40100 DEFAULT CHARACTER SET utf8
*/ /*!80016 DEFAULT ENCRYPTION='N' */ |
+----------+-----------------------------------------------------------------
-------------------------------------+
1 row in set (0.00 sec)
```

创表语句

```
-- 创建表
CREATE TABLE IF NOT EXISTS emp (
  -- int类型
  emp_id INT,
  -- 最多保存20个中英文字符
  emp_name VARCHAR(20),
  -- 总位数不超过15位
  salary DOUBLE,
  -- 日期类型
  birthday DATE
);

Query OK, 0 rows affected (0.02 sec)
```

查看表结构

```
DESC emp;

+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| emp_id   | int         | YES  |     | NULL    |       |
| emp_name | varchar(20) | YES  |     | NULL    |       |
| salary   | double      | YES  |     | NULL    |       |
| birthday | date        | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

查看建表语句

```
SHOW CREATE TABLE emp;  -- 如果创建表时，没有指定字符集，则使用数据库的字符集

+-------+-------------------------------------
| Table | Create Table                        |
+-------+-------------------------------------
| emp   | CREATE TABLE `emp` (
  `emp_id` int DEFAULT NULL,
  `emp_name` varchar(20) DEFAULT NULL,
  `salary` double DEFAULT NULL,
  `birthday` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3    |
+-------+-------------------------------------
1 row in set (0.00 sec)
```

MySQL在执行建表语句时，将id字段的类型设置为int(11)，这里的11实际上是int类型指定的显示宽度，默认的显示宽度为11

也可以在创建数据表的时候指定数据的显示宽度

- 创建表举例2：

```
CREATE TABLE dept(
    -- int类型，自增
    deptno INT(2) AUTO_INCREMENT,
    dname VARCHAR(14),
    loc VARCHAR(13),
    -- 主键
    PRIMARY KEY (deptno)
);
```

```
DESCRIBE dept;
```

> 在MySQL 8.x版本中，不再推荐为INT类型指定显示长度，并在未来的版本中可能去掉这样的语法。

3.2 创建方式2（基于现有表）

- 使用 AS subquery 选项，**将创建表和插入数据结合起来**

创建表，基于现有的表导入数据

```
CREATE TABLE table
        [(column, column...)]
AS subquery;
```

- 指定的列和子查询中的列要一一对应
- 通过列名和默认值定义列

例子一：

```
-- 基于现有的表 建表的同时插入数据
CREATE TABLE emp1
AS
SELECT * FROM employees;

Query OK, 107 rows affected, 2 warnings (0.02 sec)
Records: 107  Duplicates: 0  Warnings: 2


-- 查看新建的表的数据
```

```
SELECT last_name,email,salary FROM emp1;

+-------------+----------+----------+
| last_name   | email    | salary   |
+-------------+----------+----------+
| King        | SKING    | 24000.00 |
| Mavris      | SMAVRIS  |  6500.00 |
| Baer        | HBAER    | 10000.00 |
| Higgins     | SHIGGINS | 12000.00 |
| Gietz       | WGIETZ   |  8300.00 |
+-------------+----------+----------+
107 rows in set (0.00 sec)


-- 查看表结构
DESC emp1;

+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| employee_id    | int         | NO   |     | 0       |       |
| first_name     | varchar(20) | YES  |     | NULL    |       |
| last_name      | varchar(25) | NO   |     | NULL    |       |
| email          | varchar(25) | NO   |     | NULL    |       |
| phone_number   | varchar(20) | YES  |     | NULL    |       |
| hire_date      | date        | NO   |     | NULL    |       |
| job_id         | varchar(10) | NO   |     | NULL    |       |
| salary         | double(8,2) | YES  |     | NULL    |       |
| commission_pct | double(2,2) | YES  |     | NULL    |       |
| manager_id     | int         | YES  |     | NULL    |       |
| department_id  | int         | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
11 rows in set (0.01 sec)
```

例子二：

```
-- 基于现有表，多表连接，建表并导入数据
CREATE TABLE emp3
AS
SELECT e.last_name name,e.employee_id id,d.department_name department
FROM employees e INNER JOIN departments d
ON e.department_id=d.department_id;

Query OK, 106 rows affected (0.01 sec)
Records: 106  Duplicates: 0  Warnings: 0



-- 查看新建的表
SELECT * FROM emp3;

+-------------+-----+------------------+
```

```
| name        | id   | department       |
+-------------+------+------------------+
| Whalen      | 200  | Administration   |
| Hartstein   | 201  | Marketing        |
| McCain      | 194  | Shipping         |
| Greenberg   | 108  | Finance          |
| Higgins     | 205  | Accounting       |
| Gietz       | 206  | Accounting       |
+-------------+------+------------------+
106 rows in set (0.00 sec)



-- 查看表结构
DESC emp3;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| name       | varchar(25) | NO   |     | NULL    |       |
| id         | int         | NO   |     | 0       |       |
| department | varchar(30) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

说明：

1、查询语句中字段的别名,可以作为新创建的表的字段的名称

2、此时的查询语句可以为结构比较丰富,使用前面章节讲过的各种SELECT

仅要表结构

```
-- 创建的emp2是空表
-- 仅要表结构
CREATE TABLE emp2
AS
SELECT * FROM employees
WHERE 1=2;



-- 查看表结构
DESC emp2;

+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| employee_id    | int         | NO   |     | 0       |       |
| first_name     | varchar(20) | YES  |     | NULL    |       |
| last_name      | varchar(25) | NO   |     | NULL    |       |
```

```
| email          | varchar(25) | NO   |     | NULL    |       |
| phone_number   | varchar(20) | YES  |     | NULL    |       |
| hire_date      | date        | NO   |     | NULL    |       |
| job_id         | varchar(10) | NO   |     | NULL    |       |
| salary         | double(8,2) | YES  |     | NULL    |       |
| commission_pct | double(2,2) | YES  |     | NULL    |       |
| manager_id     | int         | YES  |     | NULL    |       |
| department_id  | int         | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
11 rows in set (0.01 sec)
```

3.3 查看数据表结构

在MySQL中创建好数据表之后，可以查看数据表的结构

1、MySQL支持使用`DESCRIBE/DESC`语句查看数据表结构

2、支持使用`SHOW CREATE TABLE`语句查看数据表结构

语法格式如下：

```
SHOW CREATE TABLE 表名\G   -- 大写
```

注意：

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

4. 修改表

修改表指的是修改数据库中已经存在的数据表的结构。

使用 ALTER TABLE 语句可以实现：

- 向已有的表中添加列
- 修改现有表中的列
- 删除现有表中的列
- 重命名现有表中的列

4.1 追加一个列

语法格式如下：

> ALTER TABLE 表名 ADD 【COLUMN】 字段名 字段类型 【FIRST|AFTER 字段名】;

```
-- 查看表结构
```

```
DESC emp;

+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| emp_id   | int         | YES  |     | NULL    |       |
| emp_name | varchar(20) | YES  |     | NULL    |       |
| salary   | double      | YES  |     | NULL    |       |
| birthday | date        | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

-- 追加一个列
ALTER TABLE emp
ADD email VARCHAR(12);  -- 默认添加到表中最后一个字段的位置
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0


-- 查看表结构
DESC emp;

+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| emp_id   | int         | YES  |     | NULL    |       |
| emp_name | varchar(20) | YES  |     | NULL    |       |
| salary   | double      | YES  |     | NULL    |       |
| birthday | date        | YES  |     | NULL    |       |
| email    | varchar(12) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)


-- 在第一个字段的位置中添加一个字段
ALTER TABLE emp
ADD phone VARCHAR(11) FIRST;

Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0


-- 在某个字段的后面添加一个字段
ALTER TABLE emp
ADD hire_date DATE AFTER emp_name;

Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0


-- 查看表结构
DESC emp;

+-----------+-------------+------+-----+---------+-------+
```

```
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| phone     | varchar(11) | YES  |     | NULL    |       |
| emp_id    | int         | YES  |     | NULL    |       |
| emp_name  | varchar(20) | YES  |     | NULL    |       |
| hire_date | date        | YES  |     | NULL    |       |
| salary    | double      | YES  |     | NULL    |       |
| birthday  | date        | YES  |     | NULL    |       |
| email     | varchar(12) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

4.2 修改一个列

- 可以修改列的数据类型，长度、默认值和位置
- 修改字段数据类型、长度、默认值、位置的语法格式如下：

> ALTER TABLE 表名 MODIFY 【COLUMN】 字段名1 字段类型 【DEFAULT 默认值】【FIRST|AFTER 字段名2】;

- 举例：

```
-- 查看表结构
DESC emp;

+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| phone     | varchar(11) | YES  |     | NULL    |       |
| emp_id    | int         | YES  |     | NULL    |       |
| emp_name  | varchar(20) | YES  |     | NULL    |       |
| hire_date | date        | YES  |     | NULL    |       |
| salary    | double      | YES  |     | NULL    |       |
| birthday  | date        | YES  |     | NULL    |       |
| email     | varchar(12) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)


-- 修改表的某个字段
ALTER TABLE emp
MODIFY emp_name VARCHAR(30);

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

-- 查看表结构
DESC emp;

+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
```

```
+----------+-------------+------+-----+---------+-------+
| phone    | varchar(11) | YES  |     | NULL    |       |
| emp_id   | int         | YES  |     | NULL    |       |
| emp_name | varchar(30) | YES  |     | NULL    |       |
| hire_date| date        | YES  |     | NULL    |       |
| salary   | double      | YES  |     | NULL    |       |
| birthday | date        | YES  |     | NULL    |       |
| email    | varchar(12) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

```
-- 修改某个字段的默认值
ALTER TABLE emp
MODIFY salary double default 1000; #注意要写上类型，即使不需要修改

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0


-- 查看表结构
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| phone    | varchar(11) | YES  |     | NULL    |       |
| emp_id   | int         | YES  |     | NULL    |       |
| emp_name | varchar(30) | YES  |     | NULL    |       |
| hire_date| date        | YES  |     | NULL    |       |
| salary   | double      | YES  |     | 1000    |       |
| birthday | date        | YES  |     | NULL    |       |
| email    | varchar(12) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

- 对默认值的修改只影响今后对表的修改
- 此外，还可以通**过此种方式修改列的约束**。这里暂先不讲

4.3 重命名一个列

使用 CHANGE old_column new_column dataType子句重命名列。语法格式如下：

> **ALTER TABLE 表名 CHANGE 【column】 列名 新列名 新数据类型;**

举例：

```
-- 查看表结构
DESC emp;

+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
```

```
+----------+-------------+------+-----+---------+-------+
| phone    | varchar(11) | YES  |     | NULL    |       |
| emp_id   | int         | YES  |     | NULL    |       |
| emp_name | varchar(30) | YES  |     | NULL    |       |
| hire_date| date        | YES  |     | NULL    |       |
| salary   | double      | YES  |     | 1000    |       |
| birthday | date        | YES  |     | NULL    |       |
| email    | varchar(12) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)



-- 重命名字段名
ALTER TABLE  emp
CHANGE emp_name last_name varchar(15); -- 重命名并修改字段类型
# 所以说CHANGE 里面是有MODIFY的行为在里面的

Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0



-- 查看表结构
DESC emp;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| phone    | varchar(11) | YES  |     | NULL    |       |
| emp_id   | int         | YES  |     | NULL    |       |
| last_name| varchar(15) | YES  |     | NULL    |       |
| hire_date| date        | YES  |     | NULL    |       |
| salary   | double      | YES  |     | 1000    |       |
| birthday | date        | YES  |     | NULL    |       |
| email    | varchar(12) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

4.4 删除一个列

删除表中某个字段的语法格式如下：

> ALTER TABLE 表名 DROP 【COLUMN】字段名

举例：

```
-- 删除某个字段
ALTER TABLE  emp
DROP COLUMN  email;
```

```
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0



-- 查看表结构
DESC emp;

+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| phone     | varchar(11) | YES  |     | NULL    |       |
| emp_id    | int         | YES  |     | NULL    |       |
| last_name | varchar(15) | YES  |     | NULL    |       |
| hire_date | date        | YES  |     | NULL    |       |
| salary    | double      | YES  |     | 1000    |       |
| birthday  | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

5. 重命名表

- 方式一：使用RENAME

```
RENAME TABLE emp
TO myemp;

Query OK, 0 rows affected (0.01 sec)



-- 查看数据库中的表
SHOW TABLES;
+---------------------+
| Tables_in_atguigudb |
+---------------------+
| countries           |
| departments         |
| emp1                |
| emp2                |
| emp3                |
| emp_details_view    |
| employees           |
| job_grades          |
| job_history         |
| jobs                |
| locations           |
```

```
| myemp                | --
| order                |
| regions              |
+----------------------+
14 rows in set (0.00 sec)
```

- 方式二（推荐）：

```
ALTER table myemp
RENAME  emp;  -- [TO]可以省略

+----------------------+
| Tables_in_atguigudb |
+----------------------+
| countries            |
| departments          |
| emp                  |
| emp1                 |
| emp2                 |
| emp3                 |
| emp_details_view     |
| employees            |
| job_grades           |
| job_history          |
| jobs                 |
| locations            |
| order                |
| regions              |
+----------------------+
14 rows in set (0.00 sec)
```

- 必须是对象的拥有者

6. 删除表

- 在MySQL中，当一张数据表`没有与其他任何数据表形成关联关系`时，可以将当前数据表直接删除。
- **数据和结构都被删除**
- 所有正在运行的相关事务被提交
- **所有相关索引被删除**

- 语法格式：

> DROP TABLE [IF EXISTS] 数据表1 [, 数据表2, ..., 数据表n];

`IF EXISTS`的含义为：如果当前数据库中存在相应的数据表，则删除数据表；

如果当前数据库中不存在相应的数据表，则忽略删除语句，不再执行删除数据表的操作

- 举例：

```
-- 删除表
DROP TABLE IF EXISTS emp1;

Query OK, 0 rows affected (0.01 sec)



-- 查看表
SHOW TABLES;

+---------------------+
| Tables_in_atguigudb |
+---------------------+
| countries           |
| departments         |
| emp                 |
| emp2                |
| emp3                |
| emp_details_view    |
| employees           |
| job_grades          |
| job_history         |
| jobs                |
| locations           |
| order               |
| regions             |
+---------------------+
13 rows in set (0.00 sec)
```

- DROP TABLE 语句不能回滚

7. 清空表

- TRUNCATE TABLE语句：
  - 删除表中所有的数据
  - 释放表的存储空间

- 举例：

```
-- 查看表中的数据据
SELECT * FROM emp3;
```

```
+-------------+-----+------------------+
| name        | id  | department       |
+-------------+-----+------------------+
| Whalen      | 200 | Administration   |
| Hartstein   | 201 | Marketing        |
| Higgins     | 205 | Accounting       |
| Gietz       | 206 | Accounting       |
+-------------+-----+------------------+
106 rows in set (0.00 sec)


# 清空表数据
TRUNCATE TABLE emp3;

Query OK, 0 rows affected (0.02 sec)



-- 查看表中的数据据
SELECT * FROM emp3;

Empty set (0.01 sec)
```

- TRUNCATE语句**不能回滚**，而使用 DELETE 语句删除数据，可以回滚

- 对比：

```
SET autocommit = FALSE;

DELETE FROM emp2;
#TRUNCATE TABLE emp2;

SELECT * FROM emp2;

ROLLBACK;

SELECT * FROM emp2;
```

DCL中COMMIT 和 ROLLBACK

COMMIT:提交数据；一旦执行了COMMIT，则数据就会被永久地保存在数据库中；意味着数据不能被回滚

ROLLBACK：一旦指向ROLLBACK，则可以实现数据的回滚；回滚到最近的一次COMMIT之后

这两个也可以称为TCL，事务控制语言

对比TRUNCATE TABLE 、 DELETE FROM

TRUNCATE TABLE:一旦执行此操作，表数据全部清除；同时，数据是不可以回滚；

DELETE FROM ： 一旦执行此操作，表数据全部可以清除（没有过滤条件）；同时，数据是可以实现回滚的；

DDL与DCL的说明

1、DDL的操作一旦执行，就不可回滚；指令 SET autocommit=FALSE ; 对DDL的操作失效;因为在执行完DDL操作后一定会进行一次提交操作；

2、DML的操作默认情况下是一旦执行是不可回滚的； 但是如果在执行DML之前执行了SET autocommit=FALSE ; 则执行的DML操作既可以实现回滚；

示例1：

```
-- 基于现有表建表并导入数据
CREATE TABLE emp
AS
SELECT employee_id id,last_name,salary
FROM employees
LIMIT 5;

Query OK, 5 rows affected, 1 warning (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 1



-- 先直接提交一次
COMMIT;

Query OK, 0 rows affected (0.00 sec)




-- 查询表数据
SELECT * FROM emp;

+-----+-----------+----------+
| id  | last_name | salary   |
+-----+-----------+----------+
| 100 | King      | 24000.00 |
| 101 | Kochhar   | 17000.00 |
| 102 | De Haan   | 17000.00 |
| 103 | Hunold    |  9000.00 |
| 104 | Ernst     |  6000.00 |
+-----+-----------+----------+
```

```
5 rows in set (0.00 sec)

-- 设置自动提交为假；则需要手动提交
SET autocommit = FALSE;
Query OK, 0 rows affected (0.00 sec)



-- 删除表中的数据
DELETE FROM emp;
Query OK, 5 rows affected (0.00 sec)


-- 查询表数据
SELECT * FROM emp;
Empty set (0.00 sec)


-- 回滚
ROLLBACK;
Query OK, 0 rows affected (0.00 sec)


-- 查询表数据
SELECT * FROM emp;

+-----+-----------+----------+
| id  | last_name | salary   |
+-----+-----------+----------+
| 100 | King      | 24000.00 |
| 101 | Kochhar   | 17000.00 |
| 102 | De Haan   | 17000.00 |
| 103 | Hunold    |  9000.00 |
| 104 | Ernst     |  6000.00 |
+-----+-----------+----------+
5 rows in set (0.00 sec)
```

示例2：

```
-- 先直接提交一次
COMMIT;
Query OK, 0 rows affected (0.00 sec)


-- 查询表数据
SELECT * FROM emp;

+-----+-----------+----------+
| id  | last_name | salary   |
+-----+-----------+----------+
| 100 | King      | 24000.00 |
| 101 | Kochhar   | 17000.00 |
```

```
| 102 | De Haan   | 17000.00 |
| 103 | Hunold    |  9000.00 |
| 104 | Ernst     |  6000.00 |
+-----+-----------+----------+
5 rows in set (0.00 sec)



-- 设置自动提交为假；则需要手动提交
SET autocommit = FALSE;
Query OK, 0 rows affected (0.00 sec)



-- 删除表中的数据
TRUNCATE TABLE emp;
Query OK, 0 rows affected (0.02 sec)


-- 查询表数据
SELECT * FROM emp;
Empty set (0.00 sec)


-- 回滚
ROLLBACK;
Query OK, 0 rows affected (0.00 sec)


-- 查询表数据
SELECT * FROM emp;
Empty set (0.00 sec)
```

阿里开发规范：

> 【参考】TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少，但 TRUNCATE 无事务且不触发 TRIGGER，有可能造成事故，故**不建议**在开发代码中使用此语句。
>
> 说明：TRUNCATE TABLE 在功能上与不带 WHERE 子句的 DELETE 语句相同

在安全面前，占用部分资源不值一提

8. 内容拓展

拓展1：阿里巴巴《Java开发手册》之MySQL字段命名

- 【强制】表名、字段名必须使用小写字母或数字，禁止出现数字开头，禁止两个下划线中间只出现数字。数据库字段名的修改代价很大，因为无法进行预发布，所以字段名称需要慎重考虑。
  - 正例：aliyun_admin，rdc_config，level3_name

- 反例：AliyunAdmin，rdcConfig，level_3_name

- 【强制】禁用保留字，如 desc、range、match、delayed 等，请参考 MySQL 官方保留字

- 【强制】表必备三字段：id, gmt_create, gmt_modified。
  - 说明：其中 id 必为主键，**类型为BIGINT UNSIGNED、单表时自增、步长为 1**
  - gmt_create, gmt_modified 的类型均为 **DATETIME** 类型，**前者现在时表示主动式创建，后者过去分词表示被动式更新**

- 【推荐】表的命名最好是遵循 “业务名称_表的作用”。
  - 正例：alipay_task 、 force_project、 trade_config

- 【推荐】库名与应用名称尽量一致。

- 【参考】合适的字符存储长度，不但节约数据库表空间、节约索引存储，更重要的是提升检索速度
- 正例：无符号值可以避免误存负数，且扩大了表示范围。

对象	年龄区间	类型	字节	表示范围
人	150 岁之内	tinyint unsigned	1	无符号值：0 到 255
龟	数百岁	smallint unsigned	2	无符号值：0 到 65535
恐龙化石	数千万年	int unsigned	4	无符号值：0 到约 43 亿
太阳	约 50 亿年	bigint unsigned	8	无符号值：0 到约 10 的 19 次方

拓展2：如何理解清空表、删除表等操作需谨慎？！

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

同样的，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

拓展3：MySQL8新特性—DDL的原子化

在MySQL 8.0版本中，InnoDB表的DDL支持事务完整性，即`DDL操作要么成功要么回滚`。

DDL操作回滚日志写入到data dictionary数据字典表mysql.innodb_ddl_log（该表是隐藏的表，通过show tables无法看到）中，用于回滚操作

通过设置参数，可将DDL操作日志打印输出到MySQL错误日志中

分别在MySQL 5.7版本和MySQL 8.0版本中创建数据库和数据表，结果如下：

```
-- 建库、建表
CREATE DATABASE mytest;

USE mytest;

CREATE TABLE book1(
book_id INT ,
book_name VARCHAR(255)
);
Query OK, 0 rows affected (0.02 sec)



-- 查看当前库中所有的表
SHOW TABLES;

+------------------+
| Tables_in_mytest |
+------------------+
| book1            |
+------------------+
1 row in set (0.00 sec)
```

（1）在MySQL 8.0 版本中，测试步骤如下：

删除数据表book1和数据表book2，结果如下：

```
DROP TABLE book1,book2;
ERROR 1051 (42S02): Unknown table 'mytest.book2'

-- 因为没有book2 所以报错
```

再次查询数据库中的数据表名称，结果如下：

```
SHOW TABLES;

+------------------+
| Tables_in_mytest |
+------------------+
| book1            |
+------------------+
1 row in set (0.00 sec)
```

从结果可以看出，数据表book1并没有被删除

（2）在MySQL 5.7版本中，测试步骤如下：
删除数据表book1和数据表book2，结果如下：

```
DROP TABLE book1,book2;
ERROR 1051 (42S02): Unknown table 'mytest.book2'

-- 因为没有book2 所以报错
```

再次查询数据库中的数据表名称，结果如下：

```
SHOW TABLES;
Empty set (0.00 sec)
```

课后练习题

1. 创建数据库test01_office,指明字符集为utf8。并在此数据库下执行下述操作

```
-- 指定字符集 建表 character set='utf8'
CREATE DATABASE IF NOT EXISTS test01_office CHARACTER SET='utf8';

Query OK, 1 row affected, 1 warning (0.00 sec)


-- 查看表结构
 SHOW CREATE DATABASE test01_office;
+---------------+-----------------------------------------------------------------
-----------------------------------------------+
| Database      | Create Database
                                                |
```

```
+--------------+-----------------------------------------------------------------
-----------------------------------------------+
| test01_office | CREATE DATABASE `test01_office` /*!40100 DEFAULT CHARACTER
SET utf8 */ /*!80016 DEFAULT ENCRYPTION='N' */ |
+--------------+-----------------------------------------------------------------
-----------------------------------------------+
1 row in set (0.00 sec)
```

2. 创建表dept01

```
字段       类型
id       INT(7)
NAME     VARCHAR(25)
```

```
USE test01_office;
Database changed


CREATE TABLE IF NOT EXISTS dept01(
id INT(7),
`NAME`  VARCHAR(25)
);

Query OK, 0 rows affected, 1 warning (0.02 sec)


DESC dept01;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int         | YES  |     | NULL    |       |
| NAME  | varchar(25) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

3. 将表departments中的数据插入新表dept02中

```
-- 创表
CREATE TABLE IF NOT EXISTS dept02
AS
SELECT *
FROM atguigudb.departments;   # 要有访问atguidudb的权限

Query OK, 27 rows affected (0.02 sec)
Records: 27  Duplicates: 0  Warnings: 0
```

```
-- 查看表结构
DESC dept02;

+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| department_id   | int         | NO   |     | 0       |       |
| department_name | varchar(30) | NO   |     | NULL    |       |
| manager_id      | int         | YES  |     | NULL    |       |
| location_id     | int         | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

4. 创建表emp01

```
id INT(7)
first_name VARCHAR (25)
last_name VARCHAR(25)
dept_id INT(7)
```

```
CREATE TABLE IF NOT EXISTS emp01(
            id INT(7),
            first_name VARCHAR (25),
            last_name VARCHAR(25),
            dept_id INT(7)
);

Query OK, 0 rows affected, 2 warnings (0.02 sec)



-- 查看表结构
Fdesc emp01;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int         | YES  |     | NULL    |       |
| first_name | varchar(25) | YES  |     | NULL    |       |
| last_name  | varchar(25) | YES  |     | NULL    |       |
| dept_id    | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

5. 将列last_name的长度增加到50

```
ALTER TABLE emp01
MODIFY last_name VARCHAR(50);

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0


-- 查看表结构
DESC emp01;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| id         | int         | YES  |     | NULL    |       |
| first_name | varchar(25) | YES  |     | NULL    |       |
| last_name  | varchar(50) | YES  |     | NULL    |       |
| dept_id    | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

6. 根据表employees创建emp02

```
CREATE TABLE emp02
AS
SELECT *
FROM atguigudb.employees;

Query OK, 107 rows affected, 2 warnings (0.02 sec)
Records: 107  Duplicates: 0  Warnings: 2



SHOW TABLES FROM test01_office; -- 查看某个库中的所有表
+------------------------+
| Tables_in_test01_office |
+------------------------+
| dept01                 |
| dept02                 |
| emp01                  |
| emp02                  |
+------------------------+
4 rows in set (0.00 sec)
```

7. 删除表emp01

```
DROP TABLE emp01;
Query OK, 0 rows affected (0.01 sec)

-- 注意：DDL语句没有回滚操作
```

8. 将表emp02重命名为emp01

```
RENAME TABLE emp02 TO emp01;
Query OK, 0 rows affected (0.01 sec)


SHOW TABLES FROM test01_office;
+-------------------------+
| Tables_in_test01_office |
+-------------------------+
| dept01                  |
| dept02                  |
| emp01                   |
+-------------------------+
3 rows in set (0.00 sec)
```

9.在表dept02和emp01中添加新列test_column，并检查所作的操作

```
ALTER TABLE emp01
ADD test_column VARCHAR(10);

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0



-- 查看表结构
DESC emp01;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| employee_id    | int         | NO   |     | 0       |       |
| first_name     | varchar(20) | YES  |     | NULL    |       |
| last_name      | varchar(25) | NO   |     | NULL    |       |
| email          | varchar(25) | NO   |     | NULL    |       |
| phone_number   | varchar(20) | YES  |     | NULL    |       |
| hire_date      | date        | NO   |     | NULL    |       |
| job_id         | varchar(10) | NO   |     | NULL    |       |
```

```
| salary         | double(8,2) | YES  |     | NULL    |       |
| commission_pct | double(2,2) | YES  |     | NULL    |       |
| manager_id     | int         | YES  |     | NULL    |       |
| department_id  | int         | YES  |     | NULL    |       |
| test_column    | varchar(10) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
12 rows in set (0.00 sec)
```

```
ALTER TABLE dept02
ADD test_column VARCHAR(10);

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0



-- 查看表结构
DESC dept02;

+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| department_id   | int         | NO   |     | 0       |       |
| department_name | varchar(30) | NO   |     | NULL    |       |
| manager_id      | int         | YES  |     | NULL    |       |
| location_id     | int         | YES  |     | NULL    |       |
| test_column     | varchar(10) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

10.直接删除表emp01中的列 department_id

```
ALTER TABLE emp01
DROP COLUMN department_id;

Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0



-- 查看表结构
DESC emp01;

+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| employee_id    | int         | NO   |     | 0       |       |
```

```
| first_name     | varchar(20) | YES  |     | NULL    |       |
| last_name      | varchar(25) | NO   |     | NULL    |       |
| email          | varchar(25) | NO   |     | NULL    |       |
| phone_number   | varchar(20) | YES  |     | NULL    |       |
| hire_date      | date        | NO   |     | NULL    |       |
| job_id         | varchar(10) | NO   |     | NULL    |       |
| salary         | double(8,2) | YES  |     | NULL    |       |
| commission_pct | double(2,2) | YES  |     | NULL    |       |
| manager_id     | int         | YES  |     | NULL    |       |
| test_column    | varchar(10) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
11 rows in set (0.00 sec)
```

练习2

1、创建数据库 test02_market

```
CREATE DATABASE IF NOT EXISTS test02_market CHARACTER SET 'utf8';

Query OK, 1 row affected, 1 warning (0.00 sec)
```

2、创建数据表 customers

字段名	数据类型
c_num	int
c_name	varchar(50)
c_contact	varchar(50)
c_city	varchar(50)
c_birth	date

```
#指定对哪个数据库进行操作
USE test02_market;

#2、创建数据表 customers
CREATE TABLE customers(
 c_num INT ,
 c_name VARCHAR(50),
 c_contact VARCHAR(50),
 c_city VARCHAR(50),
```

```
 c_birth DATE
);

Query OK, 0 rows affected (0.02 sec)


-- 查看表结构
DESC customers;

+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| c_num     | int         | YES  |     | NULL    |       |
| c_name    | varchar(50) | YES  |     | NULL    |       |
| c_contact | varchar(50) | YES  |     | NULL    |       |
| c_city    | varchar(50) | YES  |     | NULL    |       |
| c_birth   | date        | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

3、将 c_contact 字段移动到 c_birth 字段后面

```
ALTER TABLE customers
MODIFY c_contact VARCHAR(50) AFTER c_birth;

Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0



-- 查看表结构
DESC customers;

+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| c_num     | int         | YES  |     | NULL    |       |
| c_name    | varchar(50) | YES  |     | NULL    |       |
| c_city    | varchar(50) | YES  |     | NULL    |       |
| c_birth   | date        | YES  |     | NULL    |       |
| c_contact | varchar(50) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

4、将 c_name 字段数据类型改为 varchar(70)

```
ALTER TABLE customers
```

```
MODIFY c_name VARCHAR(70);

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0


-- 查看表结构
DESC customers;

+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| c_num     | int         | YES  |     | NULL    |       |
| c_name    | varchar(70) | YES  |     | NULL    |       |
| c_city    | varchar(50) | YES  |     | NULL    |       |
| c_birth   | date        | YES  |     | NULL    |       |
| c_contact | varchar(50) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

5、将c_contact字段改名为c_phone

```
ALTER TABLE customers
CHANGE c_contact c_phone VARCHAR(50);

Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0



-- 查看表结构
DESC customers;

+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| c_num   | int         | YES  |     | NULL    |       |
| c_name  | varchar(70) | YES  |     | NULL    |       |
| c_city  | varchar(50) | YES  |     | NULL    |       |
| c_birth | date        | YES  |     | NULL    |       |
| c_phone | varchar(50) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

6、增加c_gender字段到c_name后面，数据类型为char(1)

```
ALTER TABLE customers
ADD c_gender CHAR(1) AFTER c_name;

Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0


-- 查看表结构
DESC customers;

+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| c_num    | int         | YES  |     | NULL    |       |
| c_name   | varchar(70) | YES  |     | NULL    |       |
| c_gender | char(1)     | YES  |     | NULL    |       |
| c_city   | varchar(50) | YES  |     | NULL    |       |
| c_birth  | date        | YES  |     | NULL    |       |
| c_phone  | varchar(50) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

7、将表名改为customers_info

```
RENAME TABLE customers TO customers_info;

Query OK, 0 rows affected (0.01 sec)



-- 查看数据库中的表
SHOW TABLES FROM test01_office;

+-------------------------+
| Tables_in_test01_office |
+-------------------------+
| dept01                  |
| dept02                  |
| emp01                   |
+-------------------------+
3 rows in set (0.00 sec)
```

8、删除字段c_city

```
ALTER TABLE customers_info
```

```
DROP COLUMN c_city ;

Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0


-- 查看表结构
DESC customers_info;

+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| c_num    | int         | YES  |     | NULL    |       |
| c_name   | varchar(70) | YES  |     | NULL    |       |
| c_gender | char(1)     | YES  |     | NULL    |       |
| c_birth  | date        | YES  |     | NULL    |       |
| c_phone  | varchar(50) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```