

# MySQL 数据库 \*\*\*

---

作者: 王猛

Email : [bietushiwo@gmail.com](mailto:bietushiwo@gmail.com)

微博: : [@王猛](#)

QQ : 672725440

技术博客: [倾奕的官方网站](#)

说明: 本文档用于上课教案和学员复习, 可传播可分享, 如有错误, 请联系老王, 感谢矫正与探讨。

## 第1章 数据概念

---

### 1.1 什么是数据库？

---

留言本将留言存在哪里？ ---> 文本文件

文本文件管理的不够好？ ---> 我们可以放在 **数据库** 里

#### 数据库是什么呢？

它就是一个软件，它能帮我们管理数据的一个软件。

我们将数据给它，比如 一篇文章，一个人的年龄，名字，放进数据库里，它能帮我们稳妥的管理起来，且效率挺高，这种软件就叫**数据库**。

### 1.2 常用数据库概述

---

#### 1 ) Oracle数据库

世界上最好的数据库，没有之一，全平台，闭源收费，甲骨文公司主要产品。

#### 2 ) SQLServer

windows平台最好的数据库，微软研发。

#### 3 ) MySQL

世界上最容易上手的开源数据库，被Sun公司收购，后又被甲骨文公司收购，准备闭源。

#### 4 ) Maria DB

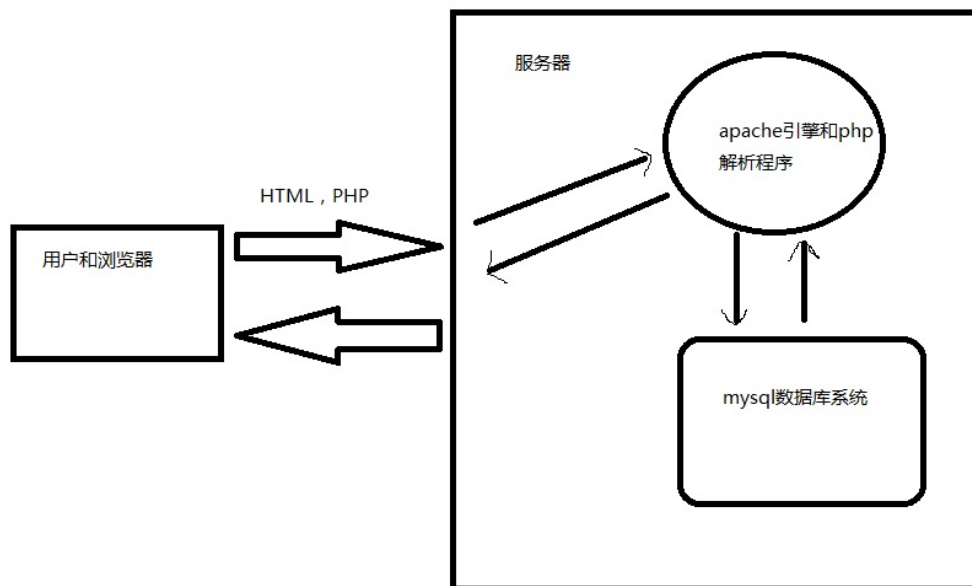
原MySQL团队利用mysql源码重新改名的产品，内容与mysql一模一样，只是换了一个名字。

## 1.3 为什么选择MySQL

- 1) 相对其他数据库，MySQL 免费，成本低。
- 2) 跨平台，Windows/Linux 跨平台，兼容性好。
- 3) 功能强大且方便，软件人性化，一键安装。

## 1.4 Web 开发原理

复习，留言板是如何实现数据的存放的？



## 第2章 SQL 结构化查询语言

### 2.1 SQL 概述

对数据库服务器中数据的管理，必须使用客户机程序成功连接以后，再通过必要的操作指令对其进行操作，这种数据库操作指令被称为 SQL ( Structured Query Language ) 语言，即结构化查询语言。

MySQL 支持 SQL 作为自己的数据库语言，SQL 是一种专门用于查询和修改数据库里的数据，以及对数据库进行管理和维护的标准化语言。

SQL 语言结构简洁，功能强大，简单易学，所以自从 IBM 公司 1981 年退出以来，SQL 语言得到了广泛的应用，逐步成为世界数据库语言标准。

世界主流数据库均支持 SQL 语句，互通性非常强。

## 2.2 SQL 语言四大分类

---

SQL 语言包含四个部分：

- 1) 数据定义语言 ( DDL ) ：用于定义和管理数据对象，包括数据库，数据表等。

例如：CREATE，DROP，ALTER 等。

- 2) 数据操作语言 ( DML ) ：用于操作数据库对象中所包含的数据。

例如：INSERT，UPDATE，DELETE 语句。

- 3) 数据查询语言 ( DQL ) ：用于查询数据库对象中所包含的数据，能够进行单表查询，连接查询，嵌套查询，以及集合查询等各种复杂程度不同的数据库查询，并将数据返回客户机中显示。

例如：SELETE

- 4) 数据控制语言 ( DCL ) ：是用来管理数据库的语言，包括管理权限及数据更改。

例如：GRANT，REVOKE，COMMIT，ROLLBACK 等。

## 第3章 MySQL 服务器登录和退出

---

### 3.1 配置环境变量

---

1. 计算机右键 ---> 属性 ---> 高级系统设置 ---> 环境变量 ---> 系统变量(S) ---> Path ( 变量 ) --->选中 ---> 编辑

2. 找到当时服务器安装目录的路径

x:\wamp64\bin\mysql\mysql5.7.14\bin

3. 复制整个路径

4. 粘贴到刚才1步骤中打开的Path的最后一行，注意，前面的内容要用分号隔开。如下图



5. 确定，保存，退出

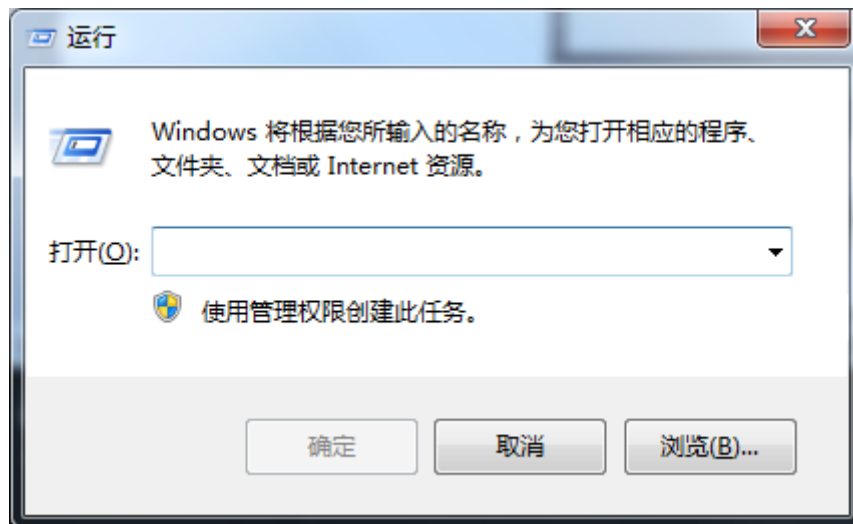
6. 重启服务器

## 3.2 登录服务器

第一种：

1) win 键 + r 呼出运行框

2) 出现如下界面



3) 运行框中输入下面内容

```
1 mysql -h localhost -uroot -p123456 -- (不推荐使用，密码明文)
2
3 mysql -h localhost -u root -p 回车 -- (推荐使用)
4
5 password:*****
6
```

第二种：

服务器右下角选择你的服务器，左键找到 mysql---> 选择mysql控制台

出现 Enter password : \*\*\*\*\* 回车

第三种（基本等同于第一种）：

Windows 开始菜单 ---> 所有程序 ---> 附件 ---> 命令提示符

出现小黑框，按照第一种，步骤输入命令即可

当出现 MySQL 的一系列提示信息的时候，说明 MySQL 服务器登录成功。

## 3.3 退出服务器

---

1. `exit` 回车退出
2. `\q` 回车退出

注意:

- 1.每个 SQL 命令都需要使用分号来结尾。
- 2.可以将一行命令拆分成多行命令。
- 3.可以通过 `\c` 来取消本行命令。
- 4.可以通过 `exit` 或者 `\q` 退出。

我们可以通过一些快捷键进行使用 help

- a. 将查询结果立起来：`\G`
- b. 取消当前为完成的操作：`\c`
- c. 退出客户端 `\q`
- d. 显示当前服务器状态：`\s`
- e. 显示帮助信息：`\h`

## 第4章 数据库操作

---

### 4.1 查看数据库

---

```
1 //（推荐使用大写）
2 SHOW DATABASES;
3 //show databases;
```

### 4.2 创建数据库

---

```
1 //如果创建的数据库不存在 我们则创建
2 //中括号代表可写可不写，不是必须有的内容
3 CREATE DATABASE [IF NOT EXISTS] 数据库名（小写）；
```

## 4.3 选择数据库

---

```
1 //数据库必须使用USE选中后才能进行下一步操作，否则报错
2 USE 数据库名；
3 //ERROR 1046 (3D000): No database selected
```

## 4.4 查看表

---

```
1 //注意，查看库和查看表都需要最后加s复数
2 SHOW TABLES；
```

## 4.5 删除数据库

---

```
1 //如果存在我们就删除
2 DROP DATABASE [IF EXISTS] 数据库名；
```

注意:

1. MySQL 数据库中命令不区分大小写。
2. 每创建一个数据库就会在 data 目录下创建一个对应的名字的文件夹。
3. 在 Windows 下数据库名称也是不区分大小写的 但是 Linux 下数据库名称严格区分大小写。

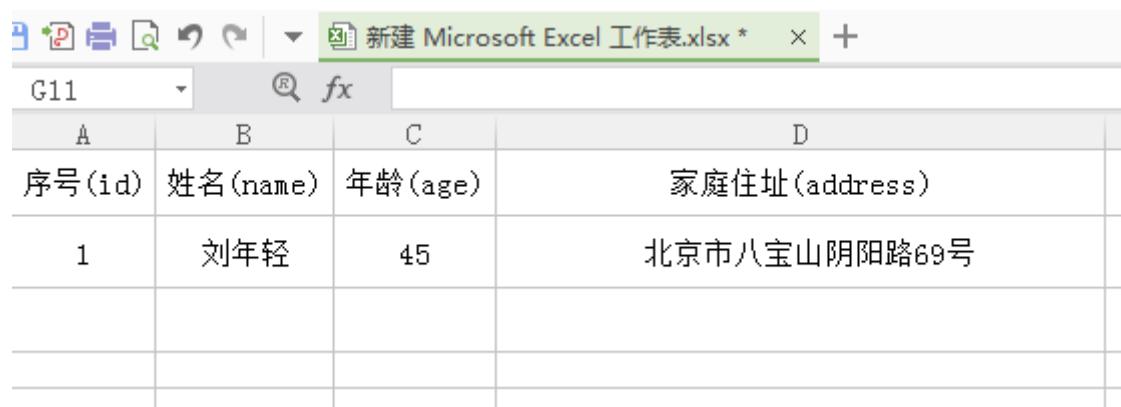
# 第5章 表操作

---

## 5.0 表概述

---

所有的数据，都以表的形式存放在数据库，就像一张表格一样，比如我们入学的时候会填写各种表，那些表就是我们数据库转换成数字以后的表，就连格式都差不多。



A	B	C	D
序号(id)	姓名(name)	年龄(age)	家庭住址(address)
1	刘年轻	45	北京市八宝山阴阳路69号

## 5.1 创建第一张表

创建一个简单的数据表（学员跟着创建，里面的细节后面讲）：

```
1 CREATE TABLE IF NOT EXISTS user1(  
2   id int UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
3   name varchar(255) NOT NULL,  
4   age tinyint NOT NULL DEFAULT 0  
5 )ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

注意：

- 1.需要先选中库，才能创建表。
- 2.字段最后一行没有逗号。

## 5.2 表内容的增删改查

### 5.2.1 insert 增加数据

备注：利用我们之前创建的简单表，做练习。

语法：

```
1 -- 添加到哪个表，字段叫啥名字，值是多少？  
2 INSERT INTO 表名('字段', '字段', '字段') VALUES('值','值','值');
```

id name age 三列

1.添加所有列

```
1 INSERT INTO user (id,name,age) VALUES (1,'lisi',23);
```

2.一行中有多个列，我们可以插入全部列，也可以插入部分列

```
1 | INSERT INTO user (uid,name) VALUES (1,'lucy');
```

3.插入所有列的简写

```
1 | -- insert into user values ('kimi',25); //报错:列计数不匹配值计数
2 | insert into user values (3,'kimi',25);
```

注意：

数字可以加单引号，它也会转成int来理解。

但是字符串必须加单引号，不加会理解为一个列名或者变量，会报错。

```
1 | INSERT INTO user VALUES ('4','zhangsan','25');
2 | INSERT INTO user VALUES (5,zhangsan,25);
```

4. 一次添加多行数据

```
1 | INSERT INTO user VALUES (5,'test1',44),(6,'test2',23),(7,'test3',18);
```

注意:

列与值，严格对应 (id 自增列也必须对应)。

数字和字符串的注意点。

数字不必加单引号，字符串必须加单引号。

1 次添加多行数据，用逗号隔开。

测试查询暂时使用如下语句：

```
1 | SELECT * FROM user;
```

## 5.2.2 update 修改操作

语法：

```
1 | -- 改哪个表？改几列的值？分别改为什么值？在哪些行生效？
2 | UPDATE 表名 SET 列名=新值,列名=新值 WHERE 条件;
```



举例：

```
1 | UPDATE user SET age=99,name='liu' WHERE id=5;
```

注意：

修改操作必须加 where 条件，不加会修改所有数据。

```
1 | UPDATE user SET age=69;
```

## 5.2.3 delete 删除操作

语法：

```
1 | -- 从哪个表删除？条件是什么？  
2 | DELETE FROM 表名 WHERE 条件;
```

举例：

```
1 | DELETE FROM user WHERE id=3;
```

注意：

删除操作必须加 where 条件，不加会删除所有数据。

## 5.2.4 select 查询入门操作

语法：

```
1 | -- 查询哪些列？从哪张表？条件是什么？  
2 | SELECT 列名 FROM 表名 WHERE 条件;
```

案例：

```

1  -- *代表所有列
2  SELECT * FROM user;
3
4  -- 查询一行
5  SELECT * FROM user WHERE id=2;
6
7  -- 查询多行
8  SELECT * FROM user WHERE id>=6;
9
10 -- 不使用*, 准确到具体的列
11 SELECT id,name FROM user WHERE id=2;
12 SELECT name FROM user WHERE id>5;

```

在语言的四大类中，增删改查占了 80% 以上的操作，而查询又在增删改查中占了 80% 以上的操作，所以，查询是一个重点需要练习，我们以上所学的是最基础的查询语句；

想写出高难度复杂的查询，我们还需要专门讨论有关查询的问题，后续我们会逐步加强查询练习。

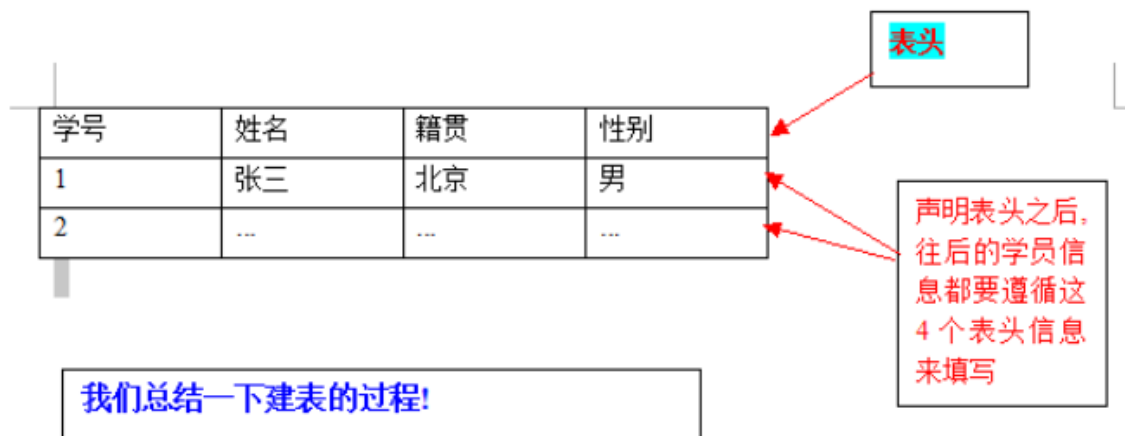
## 第6章 表类型

### 6.1 创建 table 表

别人给我们建好了表，我们可以增删改查数据，特别是查询,变化非常大。

如果别人没有把表给我们,我们连上数据库，能不能自己建一张表，更具项目的要求，做一个博客，一个商城，具体分析这个商城项目应该如何建表。

如何建一张表？



我们把表头声明完之后,表就已经声明好了；

所谓建表的过程 ---> 就是声明表头的过程；

表头的第一行 ---> 我们叫做列，而里面起的名字，我们叫 列名；

拿到数据中来说：我们建表的过程 => 就是 声明列 的过程；

建立一个简单的表：

```
1 CREATE TABLE t1(  
2   sn INT,  
3   name VARCHAR(10)  
4 );
```

我们用 word 建表,每个列是都不是等宽分布的，同样在数据库中,表中的列并不是等宽的。

像: 学号 姓名 家庭地址 性别

我们要根据这个字段有可能要存储的最大的长度，来给列分布一个最合理的宽度。

如果分配的太宽了，A4 纸就浪费了，如果分配的太窄了，有可能内容就放不下了。

数据库将我们的内容放在磁盘上，我们的内存也是有限的，所以我们在建表的时候需要考虑列的属性。

我们要分析，就是这个列，最大能存多少，然后我们选取一个合理的列类型，合理的列宽度；

使之既能放下内容，又不浪费磁盘空间，还要查询速度快。

列的属性声明规则：

```
1 CREATE TABLE 表名 (  
2   列1 列类型 [列属性 默认值],  
3   列2 列类型 [列属性 默认值],  
4   .....  
5   列n 列类型 [ 列属性 默认值]  
6 );
```

最后还需要声明引擎和字符集

```
1 CREATE TABLE 表名 (  
2   列1 列类型 [列属性 默认值],  
3   列2 列类型 [列属性 默认值],  
4   .....  
5   列n 列类型 [ 列属性 默认值]  
6 )ENGINE=MyISAM DEFAULT CAHRSET=utf8;
```

HTML 之前学的设置为 UTF-8

MySQL 设置为 utf8，没有杠。

## 6.2 列的三大分类

列类型大致分为 3 类:

## 1、数值型

整型,浮点型,定点型

## 2、字符串

char、varchar、text

## 3、日期时间类型 2012-12-13 14:26:23

# 6.3 整型

类型	字节	最小值 (带符号的/无符号的)	最大值 (带符号的/无符号的)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

MySQL 利用的是字节来表示范围和宽度的。

我们之前创建过一个表，是 int 类型的，占 4 个字节。

1 个字节有 8 个位，每一个位上都有 0、1 两种可能。

0000 0000

那 4 个字节 int 类型在磁盘中比如说存放 1 这个数字，存法如下：

00000000 00000000 00000000 00000001

int 类型存储的范围是多少能不能算出来？

11111111 11111111 11111111 11111111

2 的 32 次方 -1 大概是 42 亿

说明:

一个列，占的字节越多，存储的范围越大。

没必要死记，记住大致范围即可。

## 什么时候用无符号的？

比如：人的年龄，tinyint 类型无符号

### 6.3.1 整型列的可选参数

理解并能应用 UNSIGNED、ZEROFILL 及 M 属性

以 tinyint 为例

有符号：-128 ~ 127

无符号：0 ~ 255

#### 1) UNSIGNED 表示无符号，列的值从0开始，不能为负数

新建一个表t2

```
1  -- 如果只声明TINYINT类型而不给任何参数，表示有符号的，可以写负数
2  CREATE TABLE t2(
3    num TINYINT
4  );
5
6  CREATE TABLE t2(
7    num TINYINT UNSIGNED
8  );
9
10 -- 这个没问题
11 INSERT INTO t2 VALUES(240);
12 -- 这个就会报错
13 INSERT INTO t2 VALUES(-120);
```

#### 2) zerofill 适用于 学号，编码等，固定宽度的数字，可以用 0 填充至固定宽度

```
1  -- 改变表t2添加sn列，属性是tinyint(5)位，不足5位用0填充到5位
2  ALTER TABLE t2 ADD sn TINYINT(5) ZEROFILL;
```

**注意：**填写了 ZEROFILL 属性后不使用 UNSIGNED 进行无符号声明，因为碰到需要填充 0 的数字，生活中根本没有负数的。

**扩展：**

查看表结构使用 DESC

```
1 | DESC t2;
```

我们可以看出，插入的 sn 列，已经为 UNSIGNED 类型。

### 3) 关于TINYINT类型的后面括号中的5 ( M )

因为 INTYINT 类型本身就是有范围的，所以这个括号以后的值，如果没有 ZEROFILL 声明填充，这个值根本没有作用，这个值就是为了跟ZEROFILL做配合使用的。

```
1 | -- 规定好所谓的TINYINT(1)位
2 | Alter table t2 ADD m TINYINT(1) UNSIGNED;
3 |
4 | -- 填写一个三位的100，还是可以添加成功
5 | INSERT INTO t2(m) values(100);
```

## 6.4 浮点列与定点列

**FLOAT**：浮点型，单精度浮点型。

**DOUBLE**：范围更大的浮点型，双精度浮点数。

**DECIMAL**：定点型，无精度损失。

范围：

float -3.402823466E+38 到-1.175494351E-38、 0 和 1.175494351E-38 到 3.402823466E+38。这些是理论限制，基于 IEEE 标准。实际的范围根据硬件或操作系统的不同可能稍微小些。double 更大，没必要去记忆他们的范围，除非搞天文。

FLOAT(M,D) DOUBLE(M,D)

整型的 M 遇到 ZEROFILL 才会起作用。

FLOAT 的 M 和 D 只要设置了，立马就起作用。

**M 是精度，总位数。**

**D 标度, 小数点后面的位数。**

测试：FLOAT ( 5 , 2 )

```
1 | create table t4(
2 | money float(5,2)
3 | );
4 |
5 | -- 报错
6 | insert into t4 values (9999);
```

```

7  -- 正常
8  insert into t4 values (999.99);
9  -- 报错
10 insert into t4 values (1000);
11 -- 正常
12 insert into t4 values (-999.99);
13 -- 报错
14 insert into t4 values (-1000);

```

DECIMAL(M,D) 定点型

更加精准，无精度损失

测试：

```

1  CREATE TABLE t5(
2  f FLOAT(9,2),
3  d DECIMAL(9,2)
4  );
5
6  -- 仔细观察插入的同样的数据，查询出来的结果
7  INSERT INTO t5 VALUES (1234567.23,1234567.23);
8  -- 一个小数点是25 另一个是23

```

我们可以清晰的看到，f 列，我们插入的数据跟显示的数据是不同的。

float/double，有精度损失。

**decimal 定点型，更精确。**

**定点型，是将整数部分和小数部分用分别用数字来存储的，所以定点型更精确。**

## 6.5 字符型列

字符型:

CHAR VARCHAR TEXT/BLOB ENUM

CHAR 定长存储内容

VARCHAR 变长存储内容

### 1. CHAR和VARCHAR有什么区别呢？

CHAR(M)，VARCHAR(M) 都设置10个字符的宽度。

1) 定长的 CHAR(10)，是固定的长度。

给定列 10 个字符的宽度,最多能存 10 个字符。

哪怕你就是写一个字符，它也站 10 个字符，就好比你去吃自助餐，给了 100 块钱，吃多吃少都得花100。

## 2) 变长的 VARCHAR(10)，是可变长度。

给定列 10 个字符的宽度,最多也能存 10 个字符，多了也存不了。

比如说给 1 个字符的宽度，用多少占多少，利用率是否是 100%？当然不是，那就没有 CHAR 类型的必要了。

它会在每一个 10 个长度的前面添加 1 - 3 个字符作为提示信息，用于提示后面还有多少个长度，需要占空间的。

## 2. 那到底是选择 CHAR 还是 VARCHAR 呢？

现在磁盘容量都非常的大，如果存储的字符不是很多，比如 20 个字符以内,其实都用 CHAR，速度会更快，像微博那样的，用的就是 VARCHAR，因为它被限定在 140 个字符，而且还是可变的 140。

## 3. char 没有存够指定的长度,也能占据指定的长度，是如何做到的呢？

char 不够指定长度时用 "\0" (空格) 来填充，取出时，会把右侧的空格全部抹掉。

注：这意味着，如果右侧有本身有空格，将会丢失。

### 实验测试：

```
1 CREATE TABLE t6(  
2   n1 CHAR(10),  
3   n2 VARCHAR(10)  
4 );  
5  
6 INSERT INTO t6 VALUES (' hello ', ' hello ');  
7 SELECT * FROM t6;  
8  
9 -- CONCAT() 函数，内容拼接函数，可以自由往字段中添加值  
10 SELECT CONCAT('!', n1, '!'), CONCAT('!', n2, '!') from t6; -- 会发现CHAR类型后面的  
    空格消失了。
```

所有的空格都一样，MySQL 无法区分这是你本身的空格，所以 char 在取出时，将右侧的空格给删除掉了。

而 varchar 型则不会删除后面本身的空格，因为在它的开始处有 1 - 3 个字节已经说明了这个列占几个字符。

**注意:** char(M)，varchar(M)，限制的是字符，不是字节。

即 char(2) charset utf8，能存2个utf8字符，比如'中国'。



#### 4. char/varchar 最大可存多少个字符：

CHAR 列的长度固定为创建表时声明的长度。

长度可以为从 0 到 255 的任何值。

VARCHAR 列中的值为可变长字符串。长度也可以随便写，但不能超过手册规定的字节数。

也就是说，虽然手册中写的是 0 到 65535 个字节，但规定的是范围是字节数，而参数里写的是字符数。

比如：VARCHAR(100) ---> 就表示可以写 100 个汉字，但实际的最大支持肯定到不了 65535，因为汉字 utf8 编码一个汉字占 3 个字节，最大长度也就是 21845，而 GBK 一个汉字占 2 个字节，最大到 32766。

## 6.6 文本类型

### TEXT(M)

可以存比较大的文本段，搜索速度稍慢。

因此，如果不是特别大的内容，建议用 char，varchar 来代替。

最大长度为 65,535( $2^{16}-1$ ) 字符的 TEXT 列。

```
1 CREATE TABLE t8 (  
2   te TEXT  
3 );  
4 -- 添加字符串到t8中，最大长度65535个字符，比VARCHAR大，但速度慢  
5 INSERT INTO t8 values ('asdf asd fsdwerdsf ');  
6 SELECT * FROM t8;
```

### BLOB(M)

#### 二进制类型

其实，如果不是存图像，blob 基本用不上。

blob 是用来存储图像、音频等二进制信息。

但实际开发我们会选择本地存储图像音频等，而不是在数据库存储，数据库只存一个路径而已。

### ENUM('v1','v2',...)

#### 枚举类型

是定义好值，就在某几个枚举范围内，它是个单选值。

比如说定义两个值 ('boy','girl')，INSERT 时，只能选 "boy"，"girl"。

```
1 CREATE TABLE t10 (  
2   gender ENUM ('boy','girl')  
3 );  
4 INSERT INTO t10 VALUES ('girl');  
5 INSERT INTO t10 VALUES ('xxx');  
6 INSERT INTO t10 VALUES ('boy');
```

### SET 集合类型

它是一个设置，字符串对象可以有零个或多个值，每个值必须来自列值'value1', 'value2', ...SET列最多可以有64个成员，它是多选值。

限制太多，一般很少用。

SET 内的数据顺序是由列表顺序决定的。

不可以插入非 SET 中定义好的值。

不可以修改已经使用过的值。

不可以使用重复值。

## 6.7 日期时间列

---

日期时间类型

### Year 年(1字节) 95/1995

范围 [1901-2155]

在 INSERT 时，可以简写面的后 2 位，但是不推荐这样。

如果我们只写两位，计算年份按系统自动算法加载。

[ 00 - 69 ] + 2000

[ 70 - 99 ] + 1900

### Date 日期 1998 - 12 - 31 [ 年 月 日 ]

范围：1000 / 01 / 01 ~ 9999 / 12 / 31

### time 时间 13 : 56 : 23 [ 时 分 秒 ]

范围：-838 : 59 : 59 ~ 838 : 59 : 59

### datetime 日期时间 1998 - 12 - 31 13 : 56 : 23 [ 年 月 日 时 分 秒 ]

范围：1000/01/01 00:00:00 ~ 9999:12:31 23:59:59

**时间戳：**

是 1970 - 01 -01 00:00:00 到当前的秒数

一般存储注册时间，商品发布时间等，并不是用 datetime 存储，而是用时间戳，而且用的还是 PHP 的时间戳，而不是 MySQL 的。

```
1 CREATE TABLE t12 (  
2   ye YEAR,  
3   dt DATE,  
4   tm TIME,  
5   dttm DATETIME  
6 );  
7  
8 -- YEAR  
9 INSERT INTO t12 (ye) VALUES (1901);  
10 INSERT INTO t12 (ye) VALUES (07);  
11 INSERT INTO t12 (ye) VALUES (77);  
12  
13 -- 如果我们只写两位 values (07),(77)  
14 -- 按如下方法计算年份：  
15 -- [00-69]+2000  
16 -- [70-99]+1900  
17 -- 不建议只写两位  
18  
19  
20 -- DATE  
21 INSERT INTO t12 (dt) VALUES ('1990-12-23');  
22  
23 -- TIME  
24 INSERT INTO t12 (tm) VALUES ('12:23:59');  
25  
26 -- DATETIME  
27 INSERT INTO t12 (dttm) VALUES ('1992-12-31 12:23:59');  
28  
29 -- TIMESTAMP时间戳  
30 -- 如果我们插入改行,时间戳这列会自动插入当前时间戳  
31 -- 当我们更改这行数据时,TIMESTAMP列会自动更改时间为更改数据时的时间戳  
32 CREATE TABLE t13 (  
33   id INT,  
34   tt TIMESTAMP  
35 );  
36 INSERT INTO t13 (id) VALUES (1);  
37 SELECT * FROM t13;  
38 UPDATE t13 SET id=2 WHERE id=1;  
39 SELECT * FROM t13;
```

练习一下时间类型即可，无需理会，因为处理时间我们用 PHP 的时间戳和 date 函数，直接往数据库存纯数字，速度更快，更灵活。

## 第7章 列的默认值

### 1. 某些列不插入内容，值是多少？

自动填充 NULL

## 2. NOT NULL 是干嘛的？

不能为 NULL 类型，因为 NULL 类型查询的时候需要加语句 IS NULL 或者 IS NOT NULL

效率低，查询速度慢，开发中我们一般不可以设置为默认的能插入 NULL 类型。

所以会使用 NOT NULL 来限制使用 NULL 类型。

## 3. 既然没有默认值了，我们如何手动设置默认值？

使用 DEFAULT

```
1  -- 设置两个列id和name
2  -- id内容不能为NULL 默认值0
3  -- name内容不能为NULL 默认值空字符串
4  CREATE TABLE t14 (
5  id INT NOT NULL DEFAULT 0,
6  name CHAR(10) NOT NULL DEFAULT ''
7  );
8
9  INSERT INTO t14 VALUES (1,'listi');
10 INSERT INTO t14 (id) VALUES (2);
11
12 SELECT * FROM t14;
13 SELECT * FROM t14 WHERE name='';
```

# 第8章 主键索引与自增

PRIMARY KEY、AUTO\_INCREMENT

## 1. 什么是主键？

PRIMARY KEY，能够区分每一行的列。

以会员为例。

我们为了区分他们，往往给每一个会员加一个独一无二的会员号，这个会员号就是主键，主要在唯一的号码上加。

## 2. 设置主键后一定不能重复

不重复的是不是都可以是主键？比如说手机号，Email，也不重复。

当然不是，它还具备这么几个特征，有顺序，并且递增或者递减的一般才会加主键。

一张表中，不声明主键也可以，取决于你表中数据有没有有顺序递增或者递减的列数据。

## 3. 两种声明主键的方式

```

1  -- 在需要加主键的后面跟着主键语句
2  CREATE TABLE t15 (
3  id INT PRIMARY KEY,
4  name CHAR(5)
5  );
6
7  -- 在语句的最后再声明一行声明哪个列名要加主键
8  CREATE TABLE t16 (
9  id INT,
10 name CHAR(5),
11 PRIMARY KEY(id)
12 );
13
14 -- 不可以添加重复的信息
15 INSERT INTO t15 VALUES (3,'list');
16 INSERT INTO t15 VALUES (3,'list');

```

#### 4.主键往往和 AUTO\_INCREMENT 一起使用

这并不意味着,他们两个必须要绑定在一起使用,我们一般会把自增的列加上主键,提高效率。

但有时候,我们非要用 Email 做主键,让 Email 这一个列提高效率,也是可以的。

#### 5.AUTO\_INCREMENT 不可单独使用

使用自增选项,必须要有索引设置,否则会报错。

索引还有,普通索引 INDEX,唯一 UNIQUE,全文索引 FULLTEXT。

课后作业,把这些索引通过搜索引擎查阅,总结笔记。

## 第9章 列的删除增加与修改

表创建完毕后,能否添加 1 个列?删除 1 个列?修改 1 个列?

新增一个列,或者删除修改一个列,这属于 DDL 操作,数据库定义语言。

区分数据的增删改,插入数据是指表中的数据,不会影响到表的结构。

新建 user\_info 表:

```

1  -- id,名字,性别,体重
2  CREATE TABLE user_info(
3  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
4  username CHAR(20) NOT NULL DEFAULT '',
5  gender TINYINT UNSIGNED NOT NULL DEFAULT 0,
6  weight TINYINT UNSIGNED NOT NULL DEFAULT 0
7  )ENGINE=MyISAM DEFAULT CHARSET=utf8;

```

语法：

ALTER TABLE 表名 ADD 列名 列类型 列属性... (新列 默认在表的最后)

ALTER TABLE 表名 ADD 列名 列类型 列属性... AFTER 列名 (新列出现指定列后)

ALTER TABLE 表名 ADD 列名 列类型 列属性... FIRST (新列为第 1 列)

ALTER TABLE 表名 CHANGE 旧列名 新列名 新类型 列属性....

ALTER TABLE 表名 MODIFY 列名 新属性....

ALTER TABLE 表名 DROP [COLUMN] 列名

```

1  -- 1.添加一个列,默认新增列追加在表的最后
2  ALTER TABLE user_info ADD height TINYINT UNSIGNED NOT NULL DEFAULT 0;
3  -- 查看表结构
4  DESC user_info;
5
6  -- 2.删除身高列
7  ALTER TABLE user_info DROP height;
8  DESC user_info;
9
10 -- 3.再增加身高列,放在username后面
11 ALTER TABLE user_info ADD height TINYINT NOT NULL DEFAULT 0 AFTER username;
12 DESC user_info;
13
14 -- 4.现在人的身高越来越高,255的TINYINT已经不够存了,我们需要改变列类型,改成SMALLINT
15 -- 使用CHANGE可以将列名一起修改了.
16 ALTER TABLE user_info CHANGE height shengao SMALLINT NOT NULL DEFAULT 0;
17 DESC user_info;
18
19 -- 5.MODIFY 也可以修改列,跟CHANGE区别在于,MODIFY不可以修改列名
20 ALTER TABLE user_info MODIFY shengao TINYINT NOT NULL DEFAULT 0;
21 DESC user_info;

```

## 第10章 SELECT 查询加强

在实际工作中，98% 的工作都是查询练习，同学们要重点练习查询。

## 1.先创建一个 info 表

```
1  -- 创建一个表info,分别有名字,年龄,性别,所在城市
2  CREATE TABLE IF NOT EXISTS `info`(
3      `id` INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
4      `name` VARCHAR(255) NOT NULL DEFAULT '',
5      `age` TINYINT UNSIGNED NOT NULL DEFAULT 0,
6      `sex` TINYINT NOT NULL DEFAULT 0,
7      `city` VARCHAR(255) NOT NULL DEFAULT 0
8  )ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

## 2.插入数据

```
1  -- 为表中所有字段插入数据
2  INSERT INTO info VALUES(1,'张根硕',73,0,'上海');
3  INSERT INTO info VALUES(2,'古巨基',84,0,'台湾');
4  INSERT INTO info(name,age,sex,city) VALUES('范涛涛',54,1,'成都');
5  INSERT INTO info(name,age,sex,city) VALUES('刘建国',45,1,'成都');
6  INSERT INTO info(name,age,sex,city) VALUES('李爱党',12,1,'大兴安岭'),('刘华
   强',1,0,'河南'),
7  ('李克强',29,1,'北京'),('习老大',18,1,'北京'),('王傻强',29,1,'北京'),('帅
   猛',18,1,'北京'),('王猛猛',30,1,'山东'),('赵又挺',18,1,'上海');
8
```

# 10.1 WHERE 子句运算符

运算符	说明	运算符	说明
< <= = in	小于 小于或等于 等于 在某集合内	!= 或 <> >= > between	不等于 大于或等于 大于 在某范围内

## 逻辑运算符

运算符	说明
NOT 或 ! OR 或    AND 或 &&	逻辑非 逻辑或 逻辑与

```
1  -- 小于大于等于练习
2  SELECT id,name,sex,age,city FROM info WHERE id=5;
3  SELECT id,name,sex,age,city FROM info WHERE id<5;
4  SELECT id,name,sex,age,city FROM info WHERE id>5;
```

```

5 SELECT id,name,sex,age,city FROM info WHERE id<=5;
6 SELECT id,name,sex,age,city FROM info WHERE id>=5;
7 SELECT id,name,sex,age,city FROM info WHERE id!=5;
8 SELECT id,name,sex,age,city FROM info WHERE id<>5;
9
10 -- 指定范围 BETWEEN AND和NOT BETWEEN AND
11 SELECT id,name,sex,age,city FROM info WHERE id BETWEEN 5 AND 10;
12 SELECT id,name,sex,age,city FROM info WHERE id NOT BETWEEN 5 AND 10;
13 SELECT id,name,sex,age,city FROM info WHERE age BETWEEN 50 AND 100;
14
15 -- 指定集合
16 SELECT id,name,sex,age,city FROM info WHERE id IN(11,15,3,1,10,5,7);
17 SELECT id,name,sex,age,city FROM info WHERE id NOT IN(11,15,3,1,10,5,7);
18
19 -- AND 和 OR
20 -- 查询所有带想字的或者性别为0 的数据全部显示出来
21 SELECT id,name,sex,age,city FROM info WHERE name !='老王' OR sex=1;
22 SELECT id,name,sex,age,city FROM info WHERE name !='老王' AND sex=0;
23
24 -- 去除重复内容得到查询结果 DISTINCT
25 SELECT DISTINCT city FROM info;
26

```

## 10.2 统计函数和 GROUP BY 分组

### 统计函数

COUNT() 计算行数

AVG() 求平均函数

SUM() 求总和

MIN() 求最小

MAX() 求最大

```

1 -- 查询所有人的平均年龄
2 SELECT AVG(age) FROM info;
3
4 -- 查询最大年龄
5 SELECT MAX(age) FROM info;
6
7 -- 查询年龄最小的。
8 SELECT MIN(age) FROM info;
9
10 -- 查询一共多少人。
11 SELECT COUNT(id) FROM info;
12
13 -- 查询所有人的总年龄
14 SELECT SUM(age) FROM info;

```



## GROUP BY 分组和 GROUP\_CONCAT() 函数

```
1  -- GROUP BY 分组,名字取得谁的?谁先添加取谁的。
2  -- 你会发现报错,因为mysql无法帮你既要查询id,sex,age,city,还要帮你分组,横行都不对称。
3  -- 就算不报错,你也会发现出来的数据无意义。
4  SELECT id,sex,age,city FROM info GROUP BY city;
5
6  -- GROUP BY 和 GROUP_CONCAT() 归类函数可以实现上面的要求。
7  SELECT city,GROUP_CONCAT(id),GROUP_CONCAT(name) FROM info GROUP BY city;
8
```

## 10.3 HAVING 进一步筛选

HAVING 对结果集进一步筛选,把 having 前面的内容看成一张表,对这个表的结果进一步筛选。

```
1  -- GROUP by 和 having
2  SELECT city, COUNT(id) FROM info GROUP BY city HAVING COUNT(id)>=2;
3  SELECT city,SUM(age) FROM info GROUP BY city HAVING SUM(age)>=50;
4
5  -- 结果集可以起别名形成临时名字用于HAVING
6  SELECT city, COUNT(id) as ha FROM info GROUP BY city HAVING ha>=2;
```

## 10.4 ORDER BY 排序

```
1  -- ORDER BY 排序
2  -- ASC: 升序(默认), DESC: 降序
3  -- ASC 从小到大查询年龄
4  SELECT id,name,sex,age,city FROM info ORDER BY age ASC;
5  -- DESC 从大到小
6  SELECT id,name,sex,age,city FROM info ORDER BY age DESC;
7  -- 如果 age从大到小,我们再使用id从大到小
8  SELECT id,name,sex,age,city FROM info ORDER BY age DESC,id DESC;
9  -- 如果有where条件,先把大范围求出来再求小范围
10 SELECT id,name,sex,age,city FROM info WHERE age > 20 ORDER BY age DESC;
```

## 10.4 LIMIT 限制取出条目

限制取出条目,跳过多少行,取多少行

参数是:LIMIT m,n

```
1  -- LIMIT 不指定初始值
2  SELECT id,name,sex,age,city FROM info LIMIT 5;
3
4  -- LIMIT 指定初始值
5  SELECT id,name,sex,age,city FROM info LIMIT 0,5;
```

## 10.5 CONCAT 拼接函数

```
1  -- CONCAT 拼接函数
2  SELECT id,name,CONCAT(id,name,'hahahh ') FROM info;
```

## 10.6 为字段和表起别名

```
1  -- 为字段起别名
2  SELECT name '姓名',sex '性别',age '年龄',city '城市' FROM info;
3  SELECT name as '姓名',age as '年龄' FROM info;
4
5  -- 为表起别名
6  SELECT id,name,sex,age FROM info as i;
7  -- 起了别名后的表可作为一张新表嵌套使用
8  select name,sex,age,sum(age) from (SELECT id,name,sex,age FROM info) as i;
```

## 10.7 模糊查询

模糊查询多用于搜索.

LIKE和NOT LIKE

```
1  -- 匹配字符 LIKE NOT LIKE
2  -- 字符 _ 代表一个字符 % 代表多个字符
3  -- 模糊查询
4  SELECT id,name,sex,age,city FROM info WHERE name LIKE '张_';
5  SELECT id,name,sex,age,city FROM info WHERE name LIKE '习__';
6  SELECT id,name,sex,age,city FROM info WHERE name LIKE '___强';
7
8  -- 以xx开头,后面无所谓,爱咋滴咋滴
9  SELECT id,name,sex,age,city FROM info WHERE name LIKE '古%';
10
11 -- 以xx结尾,前边无所谓,爱咋滴咋滴
12 SELECT id,name,sex,age,city FROM info WHERE name LIKE '%强';
13
14 -- 所有带xx字符的都出来,只要带 猛 的都出来
15 SELECT id,name,sex,age,city FROM info WHERE name LIKE '%猛%';
16 SELECT id,name,sex,age,city FROM info WHERE name NOT LIKE '%猛%';
17
```

## 10.8 关联查询

新建一个关联表

```
1  -- 新建lian表
```

```

2 CREATE TABLE lian(
3     id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
4     name VARCHAR(255) NOT NULL DEFAULT '',
5     info_id INT UNSIGNED NOT NULL DEFAULT 0
6 )ENGINE=MyISAM DEFAULT CHARSET=utf8;
7
8 -- 为表添加关联内容
9 INSERT INTO lian(name,info_id) VALUES('如花姐',9);
10 INSERT INTO lian(name,info_id) VALUES('石榴姐',9);
11 INSERT INTO lian(name,info_id) VALUES('凤姐',9);
12 INSERT INTO lian(name,info_id) VALUES('芙蓉姐姐',9);
13
14 INSERT INTO lian(name,info_id) VALUES('宝强哥',18);
15 INSERT INTO lian(name,info_id) VALUES('羽凡哥',18);
16 INSERT INTO lian(name,info_id) VALUES('霆锋哥',18);
17 INSERT INTO lian(name,info_id) VALUES('之谦哥',18);
18
19 INSERT INTO lian(name,info_id) VALUES('王思聪',6);
20 INSERT INTO lian(name,info_id) VALUES('韩雪',6);
21 INSERT INTO lian(name,info_id) VALUES('任达华',6);
22 INSERT INTO lian(name,info_id) VALUES('陈楚河',6);
23
24 -- 查询info表中id为6,和lian表中info_id为6的所有值.注意,相同名字需要起别名,否则冲突.
25 -- 多表联合查询首先写select 这里先空着 from info i, lian l where 两个表的关联字段
26 -- select      from info as i,lian as l where i.id=s.info_id;
27 -- 然后再写select和from之间要查询的字段, 注意, 名字重复需要起别名, 否则mysql会懵逼
28 SELECT i.id iid,i.name iname,i.sex,i.age,i.city,s.id sid,s.name
29      sname,s.info_id
30 FROM info i,lian s WHERE i.id = s.info_id;
31
32 -- UNION合并查询,注意,查询的列名数量必须一样,会合并所有相同的.
33 SELECT id,name FROM info
34 UNION
35 SELECT id,name FROM lian;
36
37 -- UNION ALL 合并查询,不合并相同的.
38 (SELECT id,name FROM info)
39 UNION ALL
40 (SELECT id,name FROM lian);

```

## 第11章 奇怪的 NULL

给 user\_info 插入 null

```

1 INSERT INTO user_info VALUES(4,null,20,20,20);
2 -- 报错,因为我们声明了 not null

```

新建一个 tmp 表

```

1 CREATE TABLE tmp(
2   id INT,
3   name CHAR(20)
4 );
5
6 -- 添加
7 INSERT INTO tmp VALUES(1, 'diaobao'), (2, null);
8
9 -- 查询
10 SELECT * FROM tmp WHERE name=null;
11 SELECT * FROM tmp WHERE name!=null;
12 SELECT * FROM tmp WHERE name is null;
13 SELECT * FROM tmp WHERE name is not null;

```

null 比较特殊，它需要有自己的谓词来查，不便于优化，所以一半我们要尽量避免用 null。

## 第12章 事务

1.了解事务的概念。

比如：银行转帐

张三 ---> 转账给李四 500 元。

张三的钱 -500；李四的钱 +500。

两个 update 操作 这次事务才算完成，这就叫一次事务。

那：张三的钱刚 -500，打雷闪电机房断电，李四的钱还没加上。

最终这 500 块哪里去了？

日常生活中，汇款二字包含两个小动作，1 扣张三的钱，2 加李四的钱。

汇款成功，那扣钱和增加钱都需完成才算汇款成功，事物就是给你一种保证，什么样的保证呢？

要么让你都完成，要么让你都不完成，从而保证你数据的一个安全性。

否则,张三钱少了,李四没收到钱...俩人决裂了。

如何保证一个或者一次事务的完整性？

事务要选择 innodb 引擎，之前建表都使用 myisam 引擎，但它速度是稍微快一点，但不支持事务。

```

1 CREATE TABLE t29(
2   id INT,
3   name CHAR(10),
4   money int
5 )ENGINE=innodb DEFAULT CHARSET=utf8;

```

插入 zhangsan 和 lisi 的数据

```
1 | INSERT INTO t29 VALUES (1,'zhangsan',5000),(2,'lisi',5000);
```

zhangsan 借给 lisi 500 , 给 lisi 的钱 +500

```
1 | UPDATE t29 SET money=money+500 where id=2;
```

在另外一个 B 窗口查看 lisi 的钱是否多了 500

```
1 | select * from t29;
```

**这个事务还没完成，zhangsan 的钱还没减 500，lisi 不应该看到自己多的 500**

A 窗口启用事务，来试一，看看是什么样的效果。

```
1 | start transaction;
```

先查看连个人各有多少钱，再次给 lisi +500。

```
1 | select * from t29;
2 | update t29 set money=money+500 where id=2;
```

B 窗口查看 lisi 的钱。

```
1 | select * from t29;
```

A 窗口应该给 zhangsan -500。

```
1 | update t29 set money=money-500 where id=1;
```

B 窗口查看 t29。

```
1 | select * from t29;
```

事务完成，A 窗口结束事务。

```
1 | commit;
```

zhangsan 继续借给了 lisi500。

```
1 update t29 set money=money+500 where id=2;
2 update t29 set money=money-500 where id=1;
```

结果在 commit 之前，想撤销借出的钱。

需要用到：回滚 rollback。

```
1 rollback;
2
3 select * from t29;
```

## 第13章 用户操作

我们以上练习使用的全部都是 root 超级管理员权限，如果是大型公司，数据库管理这一块不归我们管，甚至中型公司我们都不可能拿到 root 权限，权限太高了，会对数据库造成很大的不可弥补的损失。

### 13.1 新建普通用户

语法：

**GRANT 权限 ON 库.表 TO '用户名'@'主机名' IDENTIFIED BY '密码';**

权限包括：

SELECT 查询

INSERT 添加

DEELTE 删除

UPDATE 修改

库：

库名.表名

. 表示所有库和所有表

用户名和主机名：

用户名自己起，但一般根据职务不同，分配的名字都具有意义，比如李哥团队，li\_user1..

主机名，本地的是使用 localhost 即可，远程的需要知道远程服务器的 IP 地址。

密码：

一般密码都是有意义并且复杂性奇高的，基本外人感觉就是乱码，常用的是 md5 加密，这里记得就行，复杂读越高，别人相对的越不容易破解。

```
1  -- 给李哥团队分配所有库所有表设置查询权限
2  GRANT SELECT ON *.* TO 'li_user1'@'localhost' IDENTIFIED BY 'lige123';
3
4  -- 给李哥团队的小弟小李子分配test库的查询权限
5  GRANT SELECT ON test.* TO 'li_user2'@'localhost' IDENTIFIED BY 'lige123';
```

## 13.2 查询所有用户

在 MySQL 数据库有，本身就有一个 MySQL 的库，里面有一张表叫 user 表，存放了数据库所有的用户名，它的字段是 user，host，查询这两个字段，即可查询出所有的用户。

```
1  SELECT user,host FROM mysql.user;
```

## 13.3 删除普通用户

当一个员工离职，我们需要删除对应的普通用户，再分配新的用户权限给它。

语法：

```
DROP USER '用户名'@'主机名';
```

```
1  -- 删除李哥团队小李子的用户名
2  DROP USER 'li_user2'@'localhost';
```

## 13.4 修改数据库密码

当我们的密码暴露了或者感觉不安全了，可以进行修改。

### 13.4.1 修改ROOT管理员密码

语法：

```
SET PASSWORD = PASSWORD('新密码');
```

```
1  -- 新密码使用mysql提供的password函数进行更改。
2  SET PASSWORD = PASSWORD('root');
```

## 13.4.2 ROOT修改普通用户密码

语法：

```
SET PASSWORD FOR '用户名'@'主机名'=PASSWORD('新密码');
```

```
1  -- 修改李哥团队的密码为lige456
2  SET PASSWORD FOR 'li_user1'@'localhost'=PASSWORD('lige456');
```

## 13.5 授权

授权的意思是，将自己某一些权限给别人使用。

语法：

```
GRANT 权限 ON 库名.表名 TO '用户名'@'主机名' IDENTIFIED BY '密码';
```

```
1  -- 给大米团队所有权限，密码123456
2  GRANT ALL ON *.* TO 'dami'@'localhost' IDENTIFIED BY '123456';
```

## 13.6 查看和权限收回

发现给大米团队的权限太高了，需要收回权限。

查看权限

语法：

```
SHOW GRANTS FOR '用户名'@'主机名'
```

```
1  -- 查看大米团队的权限,注意：查看权限GRANTS后面有个复数S
2  SHOW GRANTS FOR 'dami'@'localhost';
```

收回权限

语法：

```
REVOKE 要收回的权限 ON 库名.表名 FROM '用户名'@'主机名';
```

```
1  -- 收回大米团队的删除权限
2  REVOKE DELETE ON *.* FROM 'dami'@'localhost';
```

# 第14章 数据的导出和导入



## 14.1 导出数据库

---

当数据需要及时保存的时候，我们需要执行导出命令。

注意：此语句没有分号。

语法：

**mysqldump -u 用户名 -p 库名 表名 > 保存路径**

```
1  -- 导出整个test库
2  mysqldump -uroot -p test>C:\28.sql
3  Enter password:*****
4
5  -- 只导出test库的其中一张表
6  mysqldump -uroot -p test t6>C:\28.sql
```

## 14.2 导入数据库

---

首页必须准备一个空的数据库。

语法：

**mysql -u用户名 -p 库名 < 保存的路径**

```
1  mysql -uroot -p ss28 <c:\ss28.sql
```