

# MySQL 教程

MySQL 是最流行的关系型数据库管理系统，在WEB应用方面 MySQL 是最好的RDBMS(Relational Database Management System: 关系数据库管理系统)应用软件之一。

在本教程中，会让大家快速掌握 MySQL 的基本知识，并轻松使用 MySQL 数据库。

## 什么是数据库？

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库，

每个数据库都有一个或多个不同的API用于创建，访问，管理，搜索和复制所保存的数据。

我们也可以将数据存储在文件中，但是在文件中读写数据速度相对较慢。

所以，现在我们使用关系型数据库管理系统（RDBMS）来存储和管理的大数据量。所谓的关系型数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。

RDBMS即关系数据库管理系统(Relational Database Management System)的特点：

- 1.数据以表格的形式出现
- 2.每行为各种记录名称
- 3.每列为记录名称所对应的数据域
- 4.许多的行和列组成一张表单
- 5.若干的表单组成database

## RDBMS 术语

在我们开始学习MySQL 数据库前，让我们先了解下RDBMS的一些术语：

**数据库：**数据库是一些关联表的集合。

**数据表：**表是数据的矩阵。在一个数据库中的表看起来像一个简单的电子表格。

**列：**一列(数据元素) 包含了相同的数据，例如邮政编码的数据。

**行：**一行（=元组，或记录）是一组相关的数据，例如一条用户订阅的数据。

**冗余：**存储两倍数据，冗余降低了性能，但提高了数据的安全性。

**主键：**主键是唯一的。一个数据表中只能包含一个主键。你可以使用主键来查询数据。

**外键：**外键用于关联两个表。

**复合键：**复合键（组合键）将多个列作为一个索引键，一般用于复合索引。

**索引：**使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。类似于书籍的目录。

**参照完整性：**参照的完整性要求关系中不允许引用不存在的实体。与实体完整性是关系模型必须满足的完整性约束条件，目的是保证数据的一致性。

MySQL 为关系型数据库(Relational Database Management System), 这种所谓的"关系型"可以理解为"表格"的概念，一个关系型数据库由一个或数个表格组成，如图所示的一个表格：

| runoob_id | runoob_title | runoob_author | submission_date | 表头 |
|-----------|--------------|---------------|-----------------|----|
| 1         | 学习 PHP       | 菜鸟教程          | 2017-05-28      |    |
| 2         | 学习 Python    | 菜鸟教程          | 2018-09-21      |    |
| 3         | 学习 HTML      | 菜鸟教程          | 2018-08-23      |    |
| 4         | 学习 CSS       | 菜鸟教程          | 2018-09-18      | 行  |
| 5         | 学习 SQL       | 菜鸟教程          | 2018-09-10      |    |
|           |              |               |                 | 值  |
|           |              |               |                 |    |
|           |              |               |                 |    |
|           |              |               |                 |    |
|           |              |               |                 |    |

表头(header): 每一列的名称;

列(row): 具有相同数据类型的数据的集合;

行(col): 每一行用来描述某条记录的具体信息;

值(value): 行的具体信息, 每个值必须与该列的数据类型相同;

键(key): 键的值在当前列中具有唯一性。

## MySQL数据库

MySQL 是一个关系型数据库管理系统, 由瑞典 MySQL AB 公司开发, 目前属于 Oracle 公司。MySQL 是一种关联数据库管理系统, 关联数据库将数  
据保存在不同的表中, 而不是将所有数据放在一个大仓库内, 这样就增加了速度并提高了灵活性。

MySQL 是开源的, 所以你不需支付额外的费用。

MySQL 支持大型的数据库。可以处理拥有上千万条记录的大型数据库。

MySQL 使用标准的SQL数据语言形式。

MySQL 可以运行于多个系统上, 并且支持多种语言。这些编程语言包括C、C++、Python、Java、Perl、PHP、Eiffel、Ruby和Tcl等。

MySQL 对PHP有很好的支持, PHP是目前最流行的Web开发语言。

MySQL 支持大型数据库, 支持5000万条记录的数据仓库, 32位系统表文件最大可支持4GB, 64位系统支持最大的表文件为8TB。

MySQL 是可以定制的, 采用了GPL协议, 你可以修改源码来开发自己的 MySQL 系统。

## 在开始学习本教程前你应该了解?

在开始学习本教程前你应该了解PHP和HTML的基础知识, 并能简单的应用。

本教程的很多例子都跟PHP语言有关, 我们的实例基本上是采用PHP语言来演示。

如果你还不了解PHP, 你可以通过本站的[PHP教程](#)来了解该语言。

MySQL 安装 ☐

☐ 点我分享笔记

反馈/建议

# MySQL 安装

所有平台的 MySQL 下载地址为：[MySQL 下载](#)。挑选你需要的 *MySQL Community Server* 版本及对应的平台。

**注意：**安装过程我们需要通过开启管理员权限来安装，否则会由于权限不足导致无法安装。

## Linux/UNIX 上安装 MySQL

Linux 平台上推荐使用RPM包来安装Mysql,MySQL AB提供了以下RPM包的下载地址：

- MySQL** - MySQL服务器。你需要该选项，除非你只想连接运行在另一台机器上的MySQL服务器。
- MySQL-client** - MySQL 客户端程序，用于连接并操作Mysql服务器。
- MySQL-devel** - 库和包含文件，如果你想要编译其它MySQL客户端，例如Perl模块，则需要安装该RPM包。
- MySQL-shared** - 该软件包包含某些语言 and 应用程序需要动态装载的共享库(libmysqlclient.so\*)，使用MySQL。
- MySQL-bench** - MySQL数据库服务器的基准和性能测试工具。

安装前，我们可以检测系统是否自带安装 MySQL:

```
rpm -qa | grep mysql
```

如果你系统有安装，那可以选择进行卸载:

```
rpm -e mysql      // 普通删除模式

rpm -e --nodeps mysql  // 强力删除模式，如果使用上面命令删除时，提示有依赖的其它文件，则用该命令可以对其进行强力删除
```

安装 MySQL:

接下来我们在 **Centos7** 系统下使用 **yum** 命令安装 **MySQL**，需要注意的是 **CentOS 7** 版本中 **MySQL**数据库已从默认的程序列表中移除，所以在安装前我们需要先去官网下载 **Yum** 资源包，下载地址为：<https://dev.mysql.com/downloads/repo/yum/>  
Please report any bugs or inconsistencies you observe to our [Bug Database](#).

Thank you for your support!

|   |      |          |
|---|------|----------|
| Red Hat Enterprise Linux 7 / Oracle<br>Linux 7 (Architecture Independent), RPM<br>Package<br>(mysql57-community-release-el7-9.noarch.rpm) | 9.0K | Download |
| Red Hat Enterprise Linux 6 / Oracle   | 9.0K | Download |

```
wget https://dev.mysql.com/get/mysql57-community-release-el7-9.noarch.rpm

rpm -ivh mysql57-community-release-el7-9.noarch.rpm

yum install mysql-server
```

初始化 MySQL:

```
mysqld --initialize
```

启动 MySQL:

```
systemctl start mysqld
```

查看 MySQL 运行状态:

```
systemctl status mysqld
```

注意: 如果我们是第一次启动 **mysql** 服务, **mysql** 服务器首先会进行初始化的配置。

此外,你也可以使用 **MariaDB** 代替, **MariaDB** 数据库管理系统是 **MySQL** 的一个分支, 主要由开源社区在维护, 采用 **GPL** 授权许可。开发这个分支的原因之一是: 甲骨文公司收购了 **MySQL** 后, 有将 **MySQL** 闭源的潜在风险, 因此社区采用分支的方式来避开这个风险。

**MariaDB**的目的是完全兼容**MySQL**, 包括**API**和命令行, 使之能轻松成为**MySQL**的代替品。

```
yum install mariadb-server mariadb
```

**mariadb**数据库的相关命令是:

```
systemctl start mariadb  #启动MariaDB

systemctl stop mariadb  #停止MariaDB

systemctl restart mariadb  #重启MariaDB

systemctl enable mariadb  #设置开机启动
```

## 验证 MySQL 安装

在成功安装 **MySQL** 后, 一些基础表会表初始化, 在服务器启动后, 你可以通过简单的测试来验证 **MySQL** 是否工作正常。

使用 **mysqladmin** 工具来获取服务器状态:

使用 **mysqladmin** 命令俩检查服务器的版本, 在 **linux** 上该二进制文件位于 **/usr/bin** 目录, 在 **Windows** 上该二进制文件位于**C:\mysql\bin**。

```
[root@host]# mysqladmin --version
```

**linux**上该命令将输出以下结果, 该结果基于你的系统信息:

```
mysqladmin  Ver 8.23 Distrib 5.0.9-0, for redhat-linux-gnu on i386
```

如果以上命令执行后未输入任何信息, 说明你的**Mysql**未安装成功。

## 使用 **MySQL Client(MySQL客户端)** 执行简单的**SQL**命令

你可以在 **MySQL Client(MySQL客户端)** 使用 **mysql** 命令连接到**MySQL**服务器上，默认情况下**MySQL**服务器的密码为空，所以本实例不需要输入密码。  
命令如下：

```
[root@host]# mysql
```

以上命令执行后会输出 **mysql>**提示符，这说明你已经成功连接到**MySQL**服务器上，你可以在 **mysql>** 提示符执行**SQL**命令：

```
mysql> SHOW DATABASES;

+-----+
| Database |
+-----+
| mysql    |
| test     |
+-----+

2 rows in set (0.13 sec)
```

## **MySQL**安装后需要做的

**MySQL**安装成功后，默认的**root**用户密码为空，你可以使用以下命令来创建**root**用户的密码：

```
[root@host]# mysqladmin -u root password "new_password";
```

现在你可以通过以下命令来连接到**MySQL**服务器：

```
[root@host]# mysql -u root -p

Enter password:*****
```

**注意：**在输入密码时，密码是不会显示了，你正确输入即可。

## **Windows** 上安装 **MySQL**

**Windows** 上安装 **MySQL** 相对来说会较为简单，地那就链接 <https://cdn.mysql.com//Downloads/MySQL-8.0/mysql-8.0.11-winx64.zip> 下载 **zip** 包。  
最新版本可以在 [MySQL 下载](#) 中下载中查看。

## MySQL Community Server 5.7.13

Select Platform:

Microsoft Windows  
Select Platform...  
Microsoft Windows  
Ubuntu Linux  
Debian Linux  
SUSE Linux Enterprise Server  
Red Hat Enterprise Linux / Oracle Linux  
Fedora  
Linux - Generic  
Sun Solaris  
Mac OS X  
FreeBSD  
Source Code

Starting with MySQL 5.6 the MySQL Installer package replaces the server-only MSI packages.

### Other Downloads:

Windows (x86, 64-bit), ZIP Archive

8.0.11

183.3M

Download

(mysql-8.0.11-winx64.zip)

MD5: 0b4efe256a28cd391bf057d4c61ade09 | Signature

Windows (x86, 64-bit), ZIP Archive

8.0.11

230.5M

Download

Debug Binaries & Test Suite

(mysql-8.0.11-winx64-debug-test.zip)

MD5: 51bb19a79a9956fb325ec198bcca64fd | Signature

点击 **Download** 按钮进入下载页面，点击下图中的 **No thanks, just start my download**，就可立即下载：

Login »

using my Oracle Web account

Sign Up »

for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account following the instructions.

No thanks, just start my download.

下载完后，我们将 zip 包解压到相应的目录，这里我将解压后的文件夹放在 **C:\web\mysql-8.0.11** 下。

接下来我们需要配置下 **MySQL** 的配置文件

打开刚刚解压的文件夹 **C:\web\mysql-8.0.11**，在该文件夹下创建 **my.ini** 配置文件，编辑 **my.ini** 配置以下基本信息：

```
[mysql]

# 设置mysql客户端默认字符集

default-character-set=utf8


[mysqld]

# 设置3306端口

port = 3306

# 设置mysql的安装目录
```

```
basedir=C:\web\mysql-8.0.11

# 设置mysql数据库的数据的存放目录

datadir=C:\web\sqldata

# 允许最大连接数

max_connections=20

# 服务端使用的字符集默认为8比特编码的latin1字符集

character-set-server=utf8

# 创建新表时将使用的默认存储引擎

default-storage-engine=INNODB
```

接下来我们来启动下 **MySQL** 数据库：

以管理员身份打开 **cmd** 命令行工具，切换目录：

```
cd C:\web\mysql-8.0.11\bin
```

初始化数据库：

```
mysqld --initialize --console
```

执行完成后，会输出 **root** 用户的初始默认密码，如：

```
...

2018-04-20T02:35:05.464644Z 5 [Note] [MY-010454] [Server] A temporary password is generated for root@localhost: APWCY
5ws&hjQ

...
```

**APWCY5ws&hjQ** 就是初始密码，后续登录需要用到，你也可以在登陆后修改密码。

输入以下安装命令：

```
mysqld install
```

启动输入以下命令即可：

```
net start mysql
```

*注意: 在 5.7 需要初始化 **data** 目录:*

```
cd C:\web\mysql-8.0.11\bin

mysqld --initialize-insecure
```

*初始化后再运行 **net start mysql** 即可启动 **mysql**。*

## 登录 **MySQL**

当 **MySQL** 服务已经运行时, 我们可以通过 **MySQL** 自带的客户端工具登录到 **MySQL** 数据库中, 首先打开命令提示符, 输入以下格式的命名:

```
mysql -h 主机名 -u 用户名 -p
```

参数说明:

**-h**: 指定客户端所要登录的 **MySQL** 主机名, 登录本机(**localhost** 或 **127.0.0.1**)该参数可以省略;

**-u**: 登录的用户名;

**-p**: 告诉服务器将会使用一个密码来登录, 如果所要登录的用户名密码为空, 可以忽略此选项。

如果我们要登录本机的 **MySQL** 数据库, 只需要输入以下命令即可:

```
mysql -u root -p
```

按回车确认, 如果安装正确且 **MySQL** 正在运行, 会得到以下响应:

```
Enter password:
```

若密码存在, 输入密码登录, 不存在则直接按回车登录。登录成功后你将会看到 **Welecome to the MySQL monitor...** 的提示语。

然后命令提示符会一直以 **mysql>** 加一个闪烁的光标等待命令的输入, 输入 **exit** 或 **quit** 退出登录。

[MySQL 教程](#)

[MySQL 管理](#)



4 篇笔记

[写笔记](#)

[反馈/建议](#)



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[MySQL 安装](#)

[MySQL PHP 语法](#)

## MySQL 管理

### 启动及关闭 MySQL 服务器

#### Windows 系统下

在 **Windows** 系统下, 打开命令窗口(cmd), 进入 **MySQL** 安装目录的 **bin** 目录。

启动:

```
cd c:/mysql/bin
```

```
mysqld --console
```

关闭:



```
cd c:/mysql/bin

mysqladmin -uroot shutdown
```

## Linux 系统下

首先，我们需要通过以下命令来检查MySQL服务器是否启动：

```
ps -ef | grep mysqld
```

如果MySQL已经启动，以上命令将输出mysql进程列表， 如果mysql未启动，你可以使用以下命令来启动mysql服务器：

```
root@host# cd /usr/bin

./mysqld_safe &
```

如果你想关闭目前运行的 MySQL 服务器, 你可以执行以下命令：

```
root@host# cd /usr/bin

./mysqladmin -u root -p shutdown

Enter password: *****
```

## MySQL 用户设置

如果你需要添加 MySQL 用户，你只需要在 mysql 数据库中的 user 表添加新用户即可。

以下为添加用户的实例，用户名为guest，密码为guest123，并授权用户可进行 SELECT, INSERT 和 UPDATE操作权限：

```
root@host# mysql -u root -p

Enter password:*****

mysql> use mysql;

Database changed

mysql> INSERT INTO user

      (host, user, password,

      select_priv, insert_priv, update_priv)

      VALUES ('localhost', 'guest',

      PASSWORD('guest123'), 'Y', 'Y', 'Y');

Query OK, 1 row affected (0.20 sec)

mysql> FLUSH PRIVILEGES;
```

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

```
mysql> SELECT host, user, password FROM user WHERE user = 'guest';
```

```
+-----+-----+-----+
| host      | user   | password          |
+-----+-----+-----+
| localhost | guest  | 6f8c114b58f2ce9e |
+-----+-----+-----+

1 row in set (0.00 sec)
```

在添加用户时，请注意使用MySQL提供的 `PASSWORD()` 函数来对密码进行加密。你可以在以上实例看到用户密码加密后为：6f8c114b58f2ce9e。

注意：在 MySQL5.7 中 user 表的 password 已换成了 **authentication\_string**。

注意：password() 加密函数已经在 8.0.11 中移除了，可以使用 MD5() 函数代替。

注意：在需要执行 **FLUSH PRIVILEGES** 语句。这个命令执行后会重新载入授权表。

如果你不使用该命令，你就无法使用新创建的用户来连接mysql服务器，除非你重启mysql服务器。

你可以在创建用户时，为用户指定权限，在对应的权限列中，在插入语句中设置为 'Y' 即可，用户权限列表如下：

```
Select_priv
Insert_priv
Update_priv
Delete_priv
Create_priv
Drop_priv
Reload_priv
Shutdown_priv
Process_priv
File_priv
Grant_priv
References_priv
Index_priv
Alter_priv
```

另外一种添加用户的方法为通过SQL的 **GRANT** 命令，以下命令会给指定数据库TUTORIALS添加用户 zara，密码为 zara123。

```
root@host# mysql -u root -p
```

```
Enter password:*****
```

```
mysql> use mysql;
```

```
Database changed
```

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP
```

```
-> ON TUTORIALS.*

-> TO 'zara'@'localhost'

-> IDENTIFIED BY 'zara123';
```

以上命令会在mysql数据库中的user表创建一条用户信息记录。

注意: MySQL 的SQL语句以分号 (;) 作为结束标识。

## /etc/my.cnf 文件配置

一般情况下, 你不需要修改该配置文件, 该文件默认配置如下:

```
[mysqld]

datadir=/var/lib/mysql

socket=/var/lib/mysql/mysql.sock


[mysql.server]

user=mysql

basedir=/var/lib


[safe_mysqld]

err-log=/var/log/mysqld.log

pid-file=/var/run/mysqld/mysqld.pid
```

在配置文件中, 你可以指定不同的错误日志文件存放的目录, 一般你不需要改动这些配置。

## 管理MySQL的命令

以下列出了使用Mysql数据库过程中常用的命令:

### USE 数据库名 :

选择要操作的Mysql数据库, 使用该命令后所有Mysql命令都只针对该数据库。

```
mysql> use RUNOOB;

Database changed
```

### SHOW DATABASES:

列出 MySQL 数据库管理系统的数据库列表。

```
mysql> SHOW DATABASES;

+-----+
```

```

| Database          |
+-----+
| information_schema |
| RUNOOB            |
| cdcol             |
| mysql             |
| onethink           |
| performance_schema |
| phpmyadmin         |
| test              |
| wecenter           |
| wordpress          |
+-----+

10 rows in set (0.02 sec)

```

### SHOW TABLES:

显示指定数据库的所有表，使用该命令前需要使用 **use** 命令来选择要操作的数据库。

```

mysql> use RUNOOB;

Database changed

mysql> SHOW TABLES;

+-----+
| Tables_in_runoob |
+-----+
| employee_tbl     |
| runoob_tbl       |
| tcount_tbl       |
+-----+

3 rows in set (0.00 sec)

```

### SHOW COLUMNS FROM 数据表:

显示数据表的属性，属性类型，主键信息，是否为 **NULL**，默认值等其他信息。

```

mysql> SHOW COLUMNS FROM runoob_tbl;

+-----+-----+-----+-----+-----+-----+

```

| Field           | Type         | Null | Key | Default | Extra |
|-----------------|--------------|------|-----|---------|-------|
| runoob_id       | int(11)      | NO   | PRI | NULL    |       |
| runoob_title    | varchar(255) | YES  |     | NULL    |       |
| runoob_author   | varchar(255) | YES  |     | NULL    |       |
| submission_date | date         | YES  |     | NULL    |       |

4 rows in set (0.01 sec)

### SHOW INDEX FROM 数据表:

显示数据表的详细索引信息，包括PRIMARY KEY（主键）。

```
mysql> SHOW INDEX FROM runoob_tbl;
```

| Table      | Non_unique | Key_name | Seq_in_index  | Column_name | Collation | Cardinality | Sub_part | Packed |
|------------|------------|----------|---------------|-------------|-----------|-------------|----------|--------|
| Null       | Index_type | Comment  | Index_comment |             |           |             |          |        |
| runoob_tbl | 0          | PRIMARY  | 1             | runoob_id   | A         | 2           | NULL     | NULL   |
|            | BTREE      |          |               |             |           |             |          |        |

1 row in set (0.00 sec)

### SHOW TABLE STATUS LIKE [FROM db\_name] [LIKE 'pattern'] \G:

该命令将输出Mysql数据库管理系统的性能及统计信息。

```
mysql> SHOW TABLE STATUS FROM RUNOOB;    # 显示数据库 RUNOOB 中所有表的信息

mysql> SHOW TABLE STATUS from RUNOOB LIKE 'runoob%';    # 表名以runoob开头的表的信息

mysql> SHOW TABLE STATUS from RUNOOB LIKE 'runoob%'\G;    # 加上 \G，查询结果按列打印
```

Gif 图演示:



MySQL 安装

MySQL PHP 语法

7 篇笔记

写笔记

反馈/建议

Copyright © 2013-2018 菜鸟教程 runoob.com All Rights Reserved. 备案号: 闽ICP备15012807号-1

首页 HTML CSS JS 本地书签

MySQL 管理

MySQL 连接

## MySQL PHP 语法

MySQL 可应用于多种语言, 包括 PERL, C, C++, JAVA 和 PHP。在这些语言中, Mysql在PHP的web开发中是应用最广泛。

在本教程中我们大部分实例都采用了 PHP 语言。如果你想了解 Mysql 在 PHP 中的应用, 可以访问我们的 [PHP 中使用 Mysql 介绍](#)。

PHP提供了多种方式来访问和操作Mysql数据库记录。PHP `Mysql11` 函数格式如下:

```
mysqli_function(value,value,...);
```

以上格式中 `function`部分描述了mysql函数的功能, 如

```
mysqli_connect($connect);
```

```
mysqli_query($connect,"SQL 语句");

mysqli_fetch_array()

mysqli_close()
```

以下实例展示了PHP调用mysql函数的语法：

## 实例 (MySQLi)

```
<?php
$retval = mysqli_function(value, [value,...]);
if( !$retval )
{
    die ( "相关错误信息" );
}
// 其他 MySQL 或 PHP 语句
?>
```

从下一章开始，我们将学习到更多的MySQL功能函数。

[MySQL 管理](#)

[MySQL 连接](#)

[点我分享笔记](#)

[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[MySQL PHP 语法](#)

[MySQL 创建数据库](#)

# MySQL 连接

## 使用mysql二进制方式连接

您可以使用MySQL二进制方式进入到mysql命令提示符下来连接MySQL数据库。

### 实例

以下是从命令行中连接mysql服务器的简单实例：

```
[root@host]# mysql -u root -p

Enter password:*****
```

在登录成功后会出现 `mysql>` 命令提示窗口，你可以在上面执行任何 SQL 语句。

以上命令执行后，登录成功输出结果如下：

```
Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 2854760 to server version: 5.0.9


Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

在以上实例中，我们使用了root用户登录到mysql服务器，当然你也可以使用其他mysql用户登录。  
如果用户权限足够，任何用户都可以在mysql的命令提示窗口中进行SQL操作。  
退出 mysql> 命令提示窗口可以使用 exit 命令，如下所示：

```
mysql> exit

Bye
```

## 使用 PHP 脚本连接 MySQL

PHP 提供了 mysqli\_connect() 函数来连接数据库。  
该函数有 6 个参数，在成功链接到 MySQL 后返回连接标识，失败返回 FALSE 。

### 语法

```
mysqli_connect(host,username,password,dbname,port,socket);
```

参数说明：

| 参数       | 描述                          |
|----------|-----------------------------|
| host     | 可选。规定主机名或 IP 地址。            |
| username | 可选。规定 MySQL 用户名。            |
| password | 可选。规定 MySQL 密码。             |
| dbname   | 可选。规定默认使用的数据库。              |
| port     | 可选。规定尝试连接到 MySQL 服务器的端口号。   |
| socket   | 可选。规定 socket 或要使用的已命名 pipe。 |

你可以使用PHP的 mysqli\_close() 函数来断开与MySQL数据库的连接。  
该函数只有一个参数为 mysqli\_connect() 函数创建连接成功后返回的 MySQL 连接标识符。

### 语法

```
bool mysqli_close ( mysqli $link )
```

本函数关闭指定的连接标识所关联的到 MySQL 服务器的非持久连接。如果没有指定 link\_identifier，则关闭上一个打开的连接。  
**提示：**通常不需要使用 mysqli\_close()，因为已打开的非持久连接会在脚本执行完毕后自动关闭。

### 实例

你可以尝试以下实例来连接到你的 MySQL 服务器：



## 连接 MySQL

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysqli_error());
}
echo '数据库连接成功! ';
mysqli_close($conn);
?>
```

[MySQL PHP 语法](#)

[MySQL 创建数据库](#)

[点我分享笔记](#)

[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号: 闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[MySQL 连接](#)

[MySQL 删除数据库](#)

## MySQL 创建数据库

我们可以在登陆 MySQL 服务后, 使用 `create` 命令创建数据库, 语法如下:

```
CREATE DATABASE 数据库名;
```

以下命令简单的演示了创建数据库的过程, 数据名为 RUNOOB:

```
[root@host]# mysql -u root -p

Enter password:***** # 登录后进入终端

mysql> create DATABASE RUNOOB;
```

### 使用 mysqladmin 创建数据库

使用普通用户, 你可能需要特定的权限来创建或者删除 MySQL 数据库。

所以我们这边使用root用户登录, root用户拥有最高权限, 可以使用 `mysql mysqladmin` 命令来创建数据库。

以下命令简单的演示了创建数据库的过程, 数据名为 RUNOOB:

```
[root@host]# mysqladmin -u root -p create RUNOOB
```

```
Enter password:*****
```

以上命令执行成功后会创建 MySQL 数据库 RUNOOB。

### 使用 PHP脚本 创建数据库

PHP 使用 `mysqli_query` 函数来创建或者删除 MySQL 数据库。

该函数有两个参数，在执行成功时返回 `TRUE`，否则返回 `FALSE`。

#### 语法

```
mysqli_query(connection,query,resultmode);
```

| 参数                | 描述  |
|-------------------|---|
| <i>connection</i> | 必需。规定要使用的 MySQL 连接。   |
| <i>query</i>      | 必需，规定查询字符串。   |
| <i>resultmode</i> | 可选。一个常量。可以是下列值中的任意一个：<br><br>MYSQLI_USE_RESULT（如果需要检索大量数据，请使用这个）<br><br>MYSQLI_STORE_RESULT（默认） |

#### 实例

以下实例演示了使用PHP来创建一个数据库：

#### 创建数据库

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接错误: ' . mysqli_error($conn));
}
echo '连接成功<br />';
$sql = 'CREATE DATABASE RUNOOB';
$retval = mysqli_query($conn,$sql );
if(! $retval )
{
    die('创建数据库失败: ' . mysqli_error($conn));
}
echo "数据库 RUNOOB 创建成功\n";
mysqli_close($conn);
?>
```

执行成功后，返回如下结果：

如果数据库已存在，执行后，返回如下结果：

MySQL 连接

MySQL 删除数据库



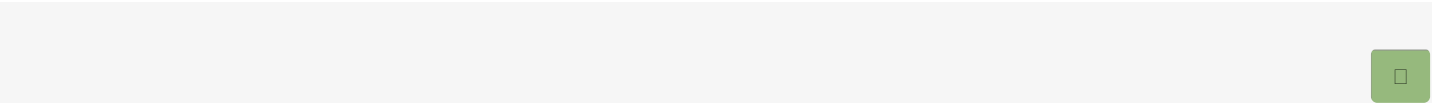
使用root登录后，可以使用

```
CREATE DATABASE IF NOT EXISTS RUNOOB DEFAULT CHARSET utf8 COLLATE utf8_general_ci;
```

创建数据库，该命令的作用：  
1. 如果数据库不存在则创建，存在则不创建。  
2. 创建RUNOOB数据库，并设定编码集为utf8



反馈/建议



# MySQL 删除数据库

使用普通用户登陆 MySQL 服务器，你可能需要特定的权限来创建或者删除 MySQL 数据库，所以我们这边使用 root 用户登录，root 用户拥有最高权限。

在删除数据库过程中，务必要十分谨慎，因为在执行删除命令后，所有数据将会消失。

## drop 命令删除数据库

drop 命令格式：

```
drop database <数据库名>;
```

例如删除名为 RUNOOB 的数据库：

```
mysql> drop database RUNOOB;
```

## 使用 mysqladmin 删除数据库

你也可以使用 mysql **mysqladmin** 命令在终端来执行删除命令。  
以下实例删除数据库 RUNOOB(该数据库在前一章节已创建)：

```
[root@host]# mysqladmin -u root -p drop RUNOOB

Enter password:*****
```

执行以上删除数据库命令后，会出现一个提示框，来确认是否真的删除数据库：

```
Dropping the database is potentially a very bad thing to do.
```

Any data stored in the database will be destroyed.

Do you really want to drop the 'RUNOOB' database [y/N] y

Database "RUNOOB" dropped

## 使用PHP脚本删除数据库

PHP使用 `mysqli_query` 函数来创建或者删除 MySQL 数据库。

该函数有两个参数，在执行成功时返回 `TRUE`，否则返回 `FALSE`。

### 语法

```
mysqli_query(connection,query,resultmode);
```

| 参数                | 描述  |
|-------------------|---|
| <i>connection</i> | 必需。规定要使用的 MySQL 连接。   |
| <i>query</i>      | 必需，规定查询字符串。   |
| <i>resultmode</i> | 可选。一个常量。可以是下列值中的任意一个：<br><br>MYSQLI_USE_RESULT（如果需要检索大量数据，请使用这个）<br><br>MYSQLI_STORE_RESULT（默认） |

### 实例

以下实例演示了使用PHP `mysqli_query`函数来删除数据库：

#### 删除数据库

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
echo '连接成功<br />';
$sql = 'DROP DATABASE RUNOOB';
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('删除数据库失败: ' . mysqli_error($conn));
}
echo "数据库 RUNOOB 删除成功\n";
mysqli_close($conn);
?>
```

执行成功后，数结果为：

□

**注意：** 在使用PHP脚本删除数据库时，不会出现确认是否删除信息，会直接删除指定数据库，所以你在删除数据库时要特别小心。

## MySQL 选择数据库

在你连接到 MySQL 数据库后, 可能有多个可以操作的数据库, 所以你需要选择你要操作的数据库。

### 从命令提示窗口中选择MySQL数据库

在 `mysql>` 提示窗口中可以很简单的选择特定的数据库。你可以使用SQL命令来选择指定的数据库。

#### 实例

以下实例选取了数据库 RUNOOB:

```
[root@host]# mysql -u root -p

Enter password:*****

mysql> use RUNOOB;

Database changed

mysql>
```

执行以上命令后, 你就已经成功选择了 RUNOOB 数据库, 在后续的操作中都会在 RUNOOB 数据库中执行。

**注意:**所有的数据库名, 表名, 表字段都是区分大小写的。所以你在使用SQL命令时需要输入正确的名称。

### 使用PHP脚本选择MySQL数据库

PHP 提供了函数 `mysqli_select_db` 来选取一个数据库。函数在执行成功后返回 `TRUE`, 否则返回 `FALSE`。

#### 语法

```
mysqli_select_db(connection,dbname);
```

| 参数                | 描述                  |
|-------------------|---------------------|
| <i>connection</i> | 必需。规定要使用的 MySQL 连接。 |
| <i>dbname</i>     | 必需, 规定要使用的默认数据库。    |

#### 实例

以下实例展示了如何使用 `mysqli_select_db` 函数来选取一个数据库:

# 选择数据库

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
echo '连接成功';
mysqli_select_db($conn, 'RUNOOB' );
mysqli_close($conn);
?>
```

点我分享笔记

反馈/建议



## MySQL 数据类型

MySQL中定义数据字段的类型对你数据库的优化是非常重要的。  
MySQL支持多种类型，大致可以分为三类：数值、日期/时间和字符串(字符)类型。

### 数值类型

MySQL支持所有标准SQL数值数据类型。  
这些类型包括严格数值数据类型(INTEGER、SMALLINT、DECIMAL和NUMERIC)，以及近似数值数据类型(FLOAT、REAL和DOUBLE PRECISION)。  
关键字INT是INTEGER的同义词，关键字DEC是DECIMAL的同义词。  
BIT数据类型保存位字段值，并且支持MyISAM、MEMORY、InnoDB和BDB表。  
作为SQL标准的扩展，MySQL也支持整数类型TINYINT、MEDIUMINT和BIGINT。下面的表显示了需要的每个整数类型的存储和范围。

| 类型              | 大小   | 范围（有符号）                         | 范围（无符号）            | 用途   |
|-----------------|------|---------------------------------|--------------------|------|
| TINYINT         | 1 字节 | (-128, 127)                     | (0, 255)           | 小整数值 |
| SMALLINT        | 2 字节 | (-32 768, 32 767)               | (0, 65 535)        | 大整数值 |
| MEDIUMINT       | 3 字节 | (-8 388 608, 8 388 607)         | (0, 16 777 215)    | 大整数值 |
| INT或<br>INTEGER | 4 字节 | (-2 147 483 648, 2 147 483 647) | (0, 4 294 967 295) | 大整数值 |

|         |   |   |   |             |
|---------|---|---|---|-------------|
| BIGINT  | 8 字节                                    | (-9 233 372 036 854 775 808, 9 223 372 036 854 775 807)   | (0, 18 446 744 073 709 551 615)                                   | 极大整数值       |
| FLOAT   | 4 字节                                    | (-3.402 823 466 E+38, -1.175 494 351 E-38), 0, (1.175 494 351 E-38, 3.402 823 466 351 E+38)   | 0, (1.175 494 351 E-38, 3.402 823 466 E+38)                       | 单精度<br>浮点数值 |
| DOUBLE  | 8 字节                                    | (-1.797 693 134 862 315 7 E+308, -2.225 073 858 507 201 4 E-308), 0, (2.225 073 858 507 201 4 E-308, 1.797 693 134 862 315 7 E+308) | 0, (2.225 073 858 507 201 4 E-308, 1.797 693 134 862 315 7 E+308) | 双精度<br>浮点数值 |
| DECIMAL | 对DECIMAL(M,D)<br>，如果M>D, 为<br>M+2否则为D+2 | 依赖于M和D的值  | 依赖于M和D的值  | 小数值         |

## 日期和时间类型

表示时间值的日期和时间类型为DATETIME、DATE、TIMESTAMP、TIME和YEAR。

每个时间类型有一个有效值范围和一个"零"值，当指定不合法的MySQL不能表示的值时使用"零"值。

TIMESTAMP类型有专有的自动更新特性，将在后面描述。

| 类型        | 大小<br>(字节) | 范围   | 格式                  | 用途           |
|-----------|------------|--|---------------------|--------------|
| DATE      | 3          | 1000-01-01/9999-12-31  | YYYY-MM-DD          | 日期值          |
| TIME      | 3          | '-838:59:59'/838:59:59'  | HH:MM:SS            | 时间值或持续时间     |
| YEAR      | 1          | 1901/2155  | YYYY                | 年份值          |
| DATETIME  | 8          | 1000-01-01 00:00:00/9999-12-31 23:59:59  | YYYY-MM-DD HH:MM:SS | 混合日期和时间值     |
| TIMESTAMP | 4          | 1970-01-01 00:00:00/2038<br>结束时间是第 <b>2147483647</b> 秒，北京时间 <b>2038-1-19 11:14:07</b> ，格林尼治时间 2038年1月19日 凌晨 03:14:07 | YYMMDD HHMMSS       | 混合日期和时间值，时间戳 |

## 字符串类型

字符串类型指CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT、ENUM和SET。该节描述了这些类型如何工作以及如何在查询中使用这些类型。

| 类型         | 大小                | 用途                 |
|------------|-------------------|--------------------|
| CHAR       | 0-255字节           | 定长字符串              |
| VARCHAR    | 0-65535 字节        | 变长字符串              |
| TINYBLOB   | 0-255字节           | 不超过 255 个字符的二进制字符串 |
| TINYTEXT   | 0-255字节           | 短文本字符串             |
| BLOB       | 0-65 535字节        | 二进制形式的长文本数据        |
| TEXT       | 0-65 535字节        | 长文本数据              |
| MEDIUMBLOB | 0-16 777 215字节    | 二进制形式的中等长度文本数据     |
| MEDIUMTEXT | 0-16 777 215字节    | 中等长度文本数据           |
| LONGBLOB   | 0-4 294 967 295字节 | 二进制形式的极大文本数据       |

|          |                   |        |
|----------|-------------------|--------|
| LONGTEXT | 0-4 294 967 295字节 | 极大文本数据 |
|----------|-------------------|--------|

CHAR 和 VARCHAR 类型类似，但它们保存和检索的方式不同。它们的最大长度和是否尾部空格被保留等方面也不同。在存储或检索过程中不进行大小写转换。

BINARY 和 VARBINARY 类似于 CHAR 和 VARCHAR，不同的是它们包含二进制字符串而不要非二进制字符串。也就是说，它们包含字节字符串而不是字符串。这说明它们没有字符集，并且排序和比较基于列值字节的数值值。

BLOB 是一个二进制大对象，可以容纳可变数量的数据。有 4 种 BLOB 类型：TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB。它们区别在于可容纳存储范围不同。

有 4 种 TEXT 类型：TINYTEXT、TEXT、MEDIUMTEXT 和 LONGTEXT。对应的这 4 种 BLOB 类型，可存储的最大长度不同，可根据实际情况选择。

2 篇笔记  
#2

写笔记

MySQL 5.0 以上的版本：

1、一个汉字占多少长度与编码有关：

UTF—8: 一个汉字=3个字节

GBK: 一个汉字=2个字节

2、varchar(n) 表示 n 个字符，无论汉字和英文，Mysql 都能存入 n 个字符，仅是实际字节长度有所区别

3、MySQL 检查长度，可用 SQL 语言来查看：

```
select LENGTH(fieldname) from tablename
```

1个月前 (08-24)

#1

1、整型

| MySQL数据类型    | 含义（有符号）                         |
|--------------|---------------------------------|
| tinyint(m)   | 1个字节 范围(-128~127)               |
| smallint(m)  | 2个字节 范围(-32768~32767)           |
| mediumint(m) | 3个字节 范围(-8388608~8388607)       |
| int(m)       | 4个字节 范围(-2147483648~2147483647) |
| bigint(m)    | 8个字节 范围(+/-9.22*10的18次方)        |

取值范围如果加了 unsigned，则最大值翻倍，如 tinyint unsigned 的取值范围为(0~256)。

int(m) 里的 m 是表示 SELECT 查询结果集中的显示宽度，并不影响实际的取值范围，没有影响到显示的宽度，不知道这个 m 有什么用。

2、浮点型(float 和 double)

| MySQL数据类型   | 含义                          |
|-------------|-----------------------------|
| float(m,d)  | 单精度浮点型 8位精度(4字节) m总个数，d小数位  |
| double(m,d) | 双精度浮点型 16位精度(8字节) m总个数，d小数位 |

设一个字段定义为 float(5,3)，如果插入一个数 123.45678,实际数据库里存的是 123.457，但总个数还以实际为准，即 6 位。

3、定点数

浮点型在数据库中存放的是近似值，而定点类型在数据库中存放的是精确值。

decimal(m,d) 参数 m<65 是总个数，d<30 且 d<m 是小数位。

4、字符串(char,varchar,text)

| MySQL数据类型 | 含义            |
|-----------|---------------|
| char(n)   | 固定长度，最多255个字符 |



|            |                    |
|------------|--------------------|
| varchar(n) | 固定长度，最多65535个字符    |
| tinytext   | 可变长度，最多255个字符      |
| text       | 可变长度，最多65535个字符    |
| mediumtext | 可变长度，最多2的24次方-1个字符 |
| longtext   | 可变长度，最多2的32次方-1个字符 |

char 和 varchar:

- 1.char(n) 若存入字符数小于n，则以空格补于其后，查询之时再将空格去掉。所以 char 类型存储的字符串末尾不能有空格，varchar 不限于此。
- 2.char(n) 固定长度，char(4) 不管是存入几个字符，都将占用 4 个字节，varchar 是存入的实际字符数 +1 个字节（n<=255）或2个字节(n>255)，所以 varchar(4),存入 3 个字符将占用 4 个字节。
- 3.char 类型的字符串检索速度要比 varchar 类型的快。

varchar 和 text:

- 1.varchar 可指定 n，text 不能指定，内部存储 varchar 是存入的实际字符数 +1 个字节（n<=255）或 2 个字节(n>255)，text 是实际字符数 +2 个字节。
- 2.text 类型不能有默认值。
- 3.varchar 可直接创建索引，text 创建索引要指定前多少个字符。varchar 查询速度快于 text, 在都创建索引的情况下，text 的索引似乎不起作用。

5.二进制数据 (\_Blob)

- 1.\_BLOB和\_text存储方式不同，\_TEXT以文本方式存储，英文存储区分大小写，而\_Blob是以二进制方式存储，不分大小写。
- 2.\_BLOB存储的数据只能整体读出。
- 3.\_TEXT可以指定字符集，\_BLO不用指定字符集。

6.日期时间类型

| MySQL数据类型 | 含义                        |
|-----------|---------------------------|
| date      | 日期 '2008-12-2'            |
| time      | 时间 '12:25:36'             |
| datetime  | 日期时间 '2008-12-2 22:06:44' |
| timestamp | 自动存储记录修改时间                |

若定义一个字段为timestamp，这个字段里的时间数据会随其他字段修改的时候自动刷新，所以这个数据类型的字段可以存放这条记录最后被修改的时间。

数据类型的属性

| MySQL关键字           | 含义            |
|--------------------|---------------|
| NULL               | 数据列可包含NULL值   |
| NOT NULL           | 数据列不允许包含NULL值 |
| DEFAULT            | 默认值           |
| PRIMARY KEY        | 主键            |
| AUTO_INCREMENT     | 自动递增，适用于整数类型  |
| UNSIGNED           | 无符号           |
| CHARACTER SET name | 指定一个字符集       |

34x1周前 (09-21)

反馈/建议



## MySQL 创建数据表

创建MySQL数据表需要以下信息：

表名

表字段名

定义每个表字段

### 语法

以下为创建MySQL数据表的SQL通用语法：

```
CREATE TABLE table_name (column_name column_type);
```

以下例子中我们将在 RUNOOB 数据库中创建数据表runoob\_tbl:

```
CREATE TABLE IF NOT EXISTS `runoob_tbl` (

  `runoob_id` INT UNSIGNED AUTO_INCREMENT,

  `runoob_title` VARCHAR(100) NOT NULL,

  `runoob_author` VARCHAR(40) NOT NULL,

  `submission_date` DATE,

  PRIMARY KEY ( `runoob_id` )

)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

实例解析：

如果你不想字段为 **NULL** 可以设置字段的属性为 **NOT NULL**， 在操作数据库时如果输入该字段的数据为**NULL**， 就会报错。

**AUTO\_INCREMENT**定义列为自增的属性，一般用于主键，数值会自动加1。

**PRIMARY KEY**关键字用于定义列为主键。 您可以使用多列来定义主键，列间以逗号分隔。

**ENGINE** 设置存储引擎，**CHARSET** 设置编码。

### 通过命令提示符创建表

通过 **mysql>** 命令窗口可以很简单的创建MySQL数据表。你可以使用 **SQL** 语句 **CREATE TABLE** 来创建数据表。

### 实例

以下为创建数据表 runoob\_tbl 实例：

```
root@host# mysql -u root -p

Enter password:*****

mysql> use RUNOOB;
```

Database changed

```
mysql> CREATE TABLE runoob_tbl(

-> runoob_id INT NOT NULL AUTO_INCREMENT,

-> runoob_title VARCHAR(100) NOT NULL,

-> runoob_author VARCHAR(40) NOT NULL,

-> submission_date DATE,

-> PRIMARY KEY ( runoob_id )

-> )ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Query OK, 0 rows affected (0.16 sec)

mysql>

注意：MySQL命令终止符为分号 (;)。

## 使用PHP脚本创建数据表

你可以使用 PHP 的 `mysqli_query()` 函数来创建已存在数据库的数据表。

该函数有两个参数，在执行成功时返回 `TRUE`，否则返回 `FALSE`。

### 语法

```
mysqli_query(connection,query,resultmode);
```

| 参数                | 描述  |
|-------------------|---|
| <i>connection</i> | 必需。规定要使用的 MySQL 连接。   |
| <i>query</i>      | 必需，规定查询字符串。   |
| <i>resultmode</i> | 可选。一个常量。可以是下列值中的任意一个：<br><br>MYSQLI_USE_RESULT（如果需要检索大量数据，请使用这个）<br><br>MYSQLI_STORE_RESULT（默认） |

### 实例

以下实例使用了PHP脚本来创建数据表：

#### 创建数据表

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
echo '连接成功<br />';
$sql = "CREATE TABLE runoob_tbl( ".
"runoob_id INT NOT NULL AUTO_INCREMENT, ".
"runoob_title VARCHAR(100) NOT NULL, ".
"runoob_author VARCHAR(40) NOT NULL, ".
"submission_date DATE, ";
```

```
"PRIMARY KEY ( runoob_id ))ENGINE=InnoDB DEFAULT CHARSET=utf8; ";
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if( ! $retval )
{
die('数据表创建失败: ' . mysqli_error($conn));
}
echo "数据表创建成功\n";
mysqli_close($conn);
?>
```

执行成功后，就可以通过命令行查看表结构：

□

□ MySQL 数据类型



MySQL 删除数据表 □



1 篇笔记  
#1

□ 写笔记



创建 **MySQL** 的表时，表名和字段名外面的符号  不是单引号，而是英文输入法状态下的反单引号，也就是键盘左上角 **esc** 按键下面的那一个  按键，坑惨了。反引号是为了区分 **MySQL** 关键字与普通字符而引入的符号，一般的，表名与字段名都使用反引号。

江湖人称二舅1年前  
(2017-07-31)

反馈/建议

Copyright © 2013-2018 菜鸟教程 **runoob.com** All Rights Reserved. 备案号：闽ICP备15012807号-1



首页 HTML CSS JS 本地书签

□ MySQL 创建数据表

MySQL 插入数据 □

## MySQL 删除数据表

MySQL中删除数据表是非常容易操作的， 但是你再进行删除表操作时要非常小心，因为执行删除命令后所有数据都会消失。

### 语法

以下为删除**MySQL**数据表的通用语法：

```
DROP TABLE table_name ;
```

### 在命令提示窗口中删除数据表

在mysql>命令提示窗口中删除数据表SQL语句为 **DROP TABLE** ：

### 实例

以下实例删除了数据表runoob\_tbl:

```
root@host# mysql -u root -p
```

```
Enter password:*****
```

```
mysql> use RUNOOB;
```

```
Database changed
```

```
mysql> DROP TABLE runoob_tbl
```

```
Query OK, 0 rows affected (0.8 sec)
```

```
mysql>
```

## 使用PHP脚本删除数据表

PHP使用 `mysqli_query` 函数来删除 MySQL 数据表。

该函数有两个参数，在执行成功时返回 `TRUE`，否则返回 `FALSE`。

### 语法

```
mysqli_query(connection,query,resultmode);
```

| 参数                | 描述  |
|-------------------|---|
| <i>connection</i> | 必需。规定要使用的 MySQL 连接。   |
| <i>query</i>      | 必需，规定查询字符串。   |
| <i>resultmode</i> | 可选。一个常量。可以是下列值中的任意一个：<br><br>MYSQLI_USE_RESULT（如果需要检索大量数据，请使用这个）<br><br>MYSQLI_STORE_RESULT（默认） |

### 实例

以下实例使用了PHP脚本删除数据表 `runoob_tbl`：

#### 删除数据库

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
echo '连接成功<br />';
$sql = "DROP TABLE runoob_tbl";
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('数据表删除失败: ' . mysqli_error($conn));
}
echo "数据表删除成功\n";
mysqli_close($conn);
?>
```

执行成功后，我们使用以下命令，就看不到 `runoob_tbl` 表了：

```
mysql> show tables;
```

Empty set (0.01 sec)

MySQL 创建数据表

MySQL 插入数据



2 篇笔记  
#2

写笔记



删除表内数据，用 **delete**。格式为：

```
delete from 表名 where 删除条件;
```

实例：删除学生表内姓名为张三的记录。

```
delete from student where T_name = "张三";
```

清除表内数据，保存表结构，用 **truncate**。格式为：

```
truncate table 表名;
```

实例：清除学生表内的所有数据。

```
truncate table student;
```

删除表用 **drop**。就是啥都没了。格式为：

```
drop table 表名;
```

实例：删除学生表。

```
drop table student;
```

- 1、当你不再需要该表时，用 **drop**；
- 2、当你仍要保留该表，但要删除所有记录时，用 **truncate**；
- 3、当你要删除部分记录时，用 **delete**。

萌面超人8个月前 (01-26)

#1



MySQL 删除表的几种情况：

- 1、**drop table table\_name**：删除表全部数据和表结构，立刻释放磁盘空间，不管是 InnoDB 和 MyISAM

实例，删除学生表：

```
drop table student;
```

- 2、**truncate table table\_name**：删除表全部数据，保留表结构，立刻释放磁盘空间，不管是 InnoDB 和 MyISAM

实例，删除学生表：

```
truncate table student;
```

- 3、**delete from table\_name**：删除表全部数据，表结构不变，对于 MyISAM 会立刻释放磁盘空间，InnoDB 不会释放磁盘空间；

实例，删除学生表：

```
delete from student;
```

- 4、**delete from table\_name where xxx**：带条件的删除，表结构不变，不管是 InnoDB 还是 MyISAM 都不会释放磁盘空间；

实例，删除学生表中姓名为 "张三" 的数据：

```
delete from student where T_name = "张三";
```

5、delete 操作以后，使用 `optimize table table_name` 会立刻释放磁盘空间，不管是 innodb 还是 myisam;  
实例，删除学生表中姓名为 "张三" 的数据：

```
delete from student where T_name = "张三";
```

实例，释放学生表的表空间：

```
optimize table student;
```

6、delete from 表以后虽然未释放磁盘空间，但是下次插入数据的时候，仍然可以使用这部分空间。

lcode4个月前 (05-23)

反馈/建议



# MySQL 插入数据

MySQL 表中使用 **INSERT INTO SQL**语句来插入数据。  
你可以通过 `mysql>` 命令提示窗口中向数据表中插入数据，或者通过PHP脚本来插入数据。

## 语法

以下为向MySQL数据表插入数据通用的 **INSERT INTO SQL**语法：

```
INSERT INTO table_name ( field1, field2,...fieldN )

VALUES

( value1, value2,...valueN );
```

如果数据是字符型，必须使用单引号或者双引号，如： "value"。

## 通过命令提示窗口插入数据

以下我们将使用 **SQL INSERT INTO** 语句向 MySQL 数据表 runoob\_tbl 插入数据

## 实例

以下实例中我们将向 runoob\_tbl 表插入三条数据：

```
root@host# mysql -u root -p password;

Enter password:*****
```

```
mysql> use RUNOOB;

Database changed

mysql> INSERT INTO runoob_tbl

    -> (runoob_title, runoob_author, submission_date)

    -> VALUES

    -> ("学习 PHP", "菜鸟教程", NOW());

Query OK, 1 rows affected, 1 warnings (0.01 sec)

mysql> INSERT INTO runoob_tbl

    -> (runoob_title, runoob_author, submission_date)

    -> VALUES

    -> ("学习 MySQL", "菜鸟教程", NOW());

Query OK, 1 rows affected, 1 warnings (0.01 sec)

mysql> INSERT INTO runoob_tbl

    -> (runoob_title, runoob_author, submission_date)

    -> VALUES

    -> ("JAVA 教程", "RUNOOB.COM", '2016-05-06');

Query OK, 1 rows affected (0.00 sec)

mysql>
```

**注意：** 使用箭头标记 `->` 不是 **SQL** 语句的一部分，它仅仅表示一个新行，如果一条**SQL**语句太长，我们可以通过回车键来创建一个新行来编写 **SQL** 语句，**SQL** 语句的命令结束符为分号 `;`。

在以上实例中，我们并没有提供 `runoob_id` 的数据，因为该字段我们在创建表的时候已经设置它为 `AUTO_INCREMENT`(自动增加) 属性。 所以，该字段会自动递增而不需要我们去设置。实例中 `NOW()` 是一个 **MySQL** 函数，该函数返回日期和时间。

接下来我们可以通过以下语句查看数据表数据：

## 读取数据表：

```
select * from runoob_tbl;
```

输出结果：

```
mysql> select * from runoob_tbl;
+-----+-----+-----+-----+
| runoob_id | runoob_title | runoob_author | submission_date |
+-----+-----+-----+-----+
| 1         | 学习 PHP    | 菜鸟教程     | 2017-04-12     |
| 2         | 学习 MySQL  | 菜鸟教程     | 2017-04-12     |
| 3         | JAVA 教程   | RUNOOB.COM   | 2016-05-06     |
+-----+-----+-----+-----+
```

## 使用PHP脚本插入数据

你可以使用PHP 的 `mysqli_query()` 函数来执行 **SQL INSERT INTO**命令来插入数据。

该函数有两个参数，在执行成功时返回 `TRUE`，否则返回 `FALSE`。

## 语法



```
mysqli_query(connection,query,resultmode);
```

| 参数                | 描述  |
|-------------------|---|
| <i>connection</i> | 必需。规定要使用的 MySQL 连接。   |
| <i>query</i>      | 必需，规定查询字符串。   |
| <i>resultmode</i> | 可选。一个常量。可以是下列值中的任意一个：<br><br>MYSQLI_USE_RESULT（如果需要检索大量数据，请使用这个）<br><br>MYSQLI_STORE_RESULT（默认） |

## 实例

以下实例中程序接收用户输入的三个字段数据，并插入数据表中：

### 添加数据

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
echo '连接成功<br />';
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
$runoob_title = '学习 Python';
$runoob_author = 'RUNOOB.COM';
$submission_date = '2016-03-06';
$sql = "INSERT INTO runoob_tbl ".
"(runoob_title,runoob_author, submission_date) ".
"VALUES ".
"('$runoob_title','$runoob_author','$submission_date')";
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法插入数据: ' . mysqli_error($conn));
}
echo "数据插入成功\n";
mysqli_close($conn);
?>
```

对于含有中文的数据插入，需要添加 `mysqli_query($conn , "set names utf8");` 语句。

接下来我们可以通过以下语句查看数据表数据：

### 读取数据表：

```
select * from runoob_tbl;
```

输出结果：

```
mysql> select * from runoob_tbl;
+-----+-----+-----+-----+
| runoob_id | runoob_title | runoob_author | submission_date |
+-----+-----+-----+-----+
| 1         | 学习 PHP    | 菜鸟教程     | 2017-04-12     |
| 2         | 学习 MySQL  | 菜鸟教程     | 2017-04-12     |
| 3         | JAVA 教程   | RUNOOB.COM   | 2016-05-06     |
| 4         | 学习 Python | RUNOOB.COM   | 2016-03-06     |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

MySQL 删除数据表

MySQL 查询数据

1 篇笔记

#1

写笔记

INSERT 插入多条数据

```
INSERT INTO table_name (field1, field2,...fieldN) VALUES (valueA1,valueA2,...valueAN),(valueB1,valueB2,...valueBN),(valueC1,valueC2,...valueCN).....;
```

YANGYANG

12个月

前 (10-16)

反馈/建议

首页

HTML

CSS

JS

本地书签

MySQL 插入数据

MySQL WHERE 子句

MySQL 查询数据

MySQL 数据库使用SQL SELECT语句来查询数据。

你可以通过 `mysql>` 命令提示窗口中在数据库中查询数据，或者通过PHP脚本来查询数据。

### 语法

以下为在MySQL数据库中查询数据通用的 `SELECT` 语法：

```
SELECT column_name,column_name

FROM table_name

[WHERE Clause]

[LIMIT N][ OFFSET M]
```

查询语句中你可以使用一个或者多个表，表之间使用逗号(,)分割，并使用WHERE语句来设定查询条件。

`SELECT` 命令可以读取一条或者多条记录。

你可以使用星号(\*)来代替其他字段，`SELECT`语句会返回表的所有字段数据

你可以使用 `WHERE` 语句来包含任何条件。

你可以使用 `LIMIT` 属性来设定返回的记录数。

你可以通过`OFFSET`指定`SELECT`语句开始查询的数据偏移量。默认情况下偏移量为0。

## 通过命令提示符获取数据

以下实例我们将通过 `SQL SELECT` 命令来获取 MySQL 数据表 `runoob_tbl` 的数据：

## 实例

以下实例将返回数据表 runoob\_tbl 的所有记录:

### 读取数据表:

```
select * from runoob_tbl;
```

输出结果:

```
mysql> select * from runoob_tbl;
+-----+-----+-----+-----+
| runoob_id | runoob_title | runoob_author | submission_date |
+-----+-----+-----+-----+
| 1         | 学习 PHP    | 菜鸟教程     | 2017-04-12      |
| 2         | 学习 MySQL  | 菜鸟教程     | 2017-04-12      |
| 3         | JAVA 教程   | RUNOOB.COM   | 2016-05-06      |
| 4         | 学习 Python | RUNOOB.COM   | 2016-03-06      |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

## 使用PHP脚本来获取数据

使用 PHP 函数的 `mysqli_query()` 及 SQL `SELECT` 命令来获取数据。

该函数用于执行 SQL 命令, 然后通过 PHP 函数 `mysqli_fetch_array()` 来使用或输出所有查询的数据。

`mysqli_fetch_array()` 函数从结果集中取得一行作为关联数组, 或数字数组, 或二者兼有 返回根据从结果集取得的行生成的数组, 如果没有更多行则返回 `false`。

以下实例为从数据表 runoob\_tbl 中读取所有记录。

## 实例

尝试以下实例来显示数据表 runoob\_tbl 的所有记录。

### 使用 `mysqli_fetch_array` `MYSQLI_ASSOC` 参数获取数据:

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码, 防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'SELECT runoob_id, runoob_title,
runoob_author, submission_date
FROM runoob_tbl';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 mysqli_fetch_array 测试</h2>';
echo '<table border="1"><tr><td>教程 ID</td><td>标题</td><td>作者</td><td>提交日期</td></tr>';
while($row = mysqli_fetch_array($retval, MYSQLI_ASSOC))
{
    echo "<tr><td> {$row['runoob_id']}</td> ".
    "<td>{$row['runoob_title']} </td> ".
    "<td>{$row['runoob_author']} </td> ".
    "<td>{$row['submission_date']} </td> ".
    "</tr>";
}
echo '</table>';
mysqli_close($conn);
?>
```

输出结果如下所示:

## 菜鸟教程 mysqli\_fetch\_array 测试

| 教程 ID | 标题        | 作者         | 提交日期       |
|-------|-----------|------------|------------|
| 1     | 学习 PHP    | 菜鸟教程       | 2017-04-12 |
| 2     | 学习 MySQL  | 菜鸟教程       | 2017-04-12 |
| 3     | JAVA 教程   | RUNOOB.COM | 2016-05-06 |
| 4     | 学习 Python | RUNOOB.COM | 2016-03-06 |

以上实例中，读取的每行记录赋值给变量 `$row`，然后再打印出每个值。

**注意：**记住如果你需要在字符串中使用变量，请将变量置于花括号。

在上面的例子中，PHP `mysqli_fetch_array()` 函数第二个参数为 **MYSQLI\_ASSOC**，设置该参数查询结果返回关联数组，你可以使用字段名称来作为数组的索引。

PHP 提供了另外一个函数 `mysqli_fetch_assoc()`，该函数从结果集中取得一行作为关联数组。返回根据从结果集取得的行生成的关联数组，如果没有更多行，则返回 **false**。

### 实例

尝试以下实例，该实例使用了 `mysqli_fetch_assoc()` 函数来输出数据表 `runoob_tbl` 的所有记录：

#### 使用 `mysqli_fetch_assoc` 获取数据：

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'SELECT runoob_id, runoob_title,
runoob_author, submission_date
FROM runoob_tbl';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 mysqli_fetch_assoc 测试<h2>';
echo '<table border="1"><tr><td>教程 ID</td><td>标题</td><td>作者</td><td>提交日期</td></tr>';
while($row = mysqli_fetch_assoc($retval))
{
    echo "<tr><td> {$row['runoob_id']}</td> ".
    "<td>{$row['runoob_title']} </td> ".
    "<td>{$row['runoob_author']} </td> ".
    "<td>{$row['submission_date']} </td> ".
    "</tr>";
}
echo '</table>';
mysqli_close($conn);
?>
```

输出结果如下所示：

## 菜鸟教程 mysqli\_fetch\_assoc 测试

| 教程 ID | 标题        | 作者         | 提交日期       |
|-------|-----------|------------|------------|
| 1     | 学习 PHP    | 菜鸟教程       | 2017-04-12 |
| 2     | 学习 MySQL  | 菜鸟教程       | 2017-04-12 |
| 3     | JAVA 教程   | RUNOOB.COM | 2016-05-06 |
| 4     | 学习 Python | RUNOOB.COM | 2016-03-06 |

你也可以使用常量 MYSQLI\_NUM 作为 PHP mysqli\_fetch\_array() 函数的第二个参数，返回数字数组。

### 实例

以下实例使用 **MYSQLI\_NUM** 参数显示数据表 runoob\_tbl 的所有记录：

### 使用 mysqli\_fetch\_array MYSQLI\_NUM 参数获取数据：

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'SELECT runoob_id, runoob_title,
runoob_author, submission_date
FROM runoob_tbl';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 mysqli_fetch_array 测试<h2>';
echo '<table border="1"><tr><td>教程 ID</td><td>标题</td><td>作者</td><td>提交日期</td></tr>';
while($row = mysqli_fetch_array($retval, MYSQLI_NUM))
{
    echo "<tr><td> {$row[0]}</td> ".
    "<td>{$row[1]} </td> ".
    "<td>{$row[2]} </td> ".
    "<td>{$row[3]} </td> ".
    "</tr>";
}
echo '</table>';
mysqli_close($conn);
?>
```

输出结果如下所示：

## 菜鸟教程 mysqli\_fetch\_array 测试

| 教程 ID | 标题        | 作者         | 提交日期       |
|-------|-----------|------------|------------|
| 1     | 学习 PHP    | 菜鸟教程       | 2017-04-12 |
| 2     | 学习 MySQL  | 菜鸟教程       | 2017-04-12 |
| 3     | JAVA 教程   | RUNOOB.COM | 2016-05-06 |
| 4     | 学习 Python | RUNOOB.COM | 2016-03-06 |

以上三个实例输出结果都一样。

在我们执行完 **SELECT** 语句后，释放游标内存是一个很好的习惯。

可以通过 **PHP** 函数 `mysqli_free_result()` 来实现内存的释放。

以下实例演示了该函数的使用方法。

### 实例

尝试以下实例：

#### 使用 `mysqli_free_result` 释放内存：

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'SELECT runoob_id, runoob_title,
runoob_author, submission_date
FROM runoob_tbl';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 mysqli_fetch_array 测试<h2>';
echo '<table border="1"><tr><td>教程 ID</td><td>标题</td><td>作者</td><td>提交日期</td></tr>';
while($row = mysqli_fetch_array($retval, MYSQLI_NUM))
{
    echo "<tr><td> {$row[0]}</td> ".
    "<td>{$row[1]} </td> ".
    "<td>{$row[2]} </td> ".
    "<td>{$row[3]} </td> ".
    "</tr>";
}
echo '</table>';
// 释放内存
mysqli_free_result($retval);
mysqli_close($conn);
?>
```

输出结果如下所示：

菜鸟教程 `mysqli_fetch_array` 测试

| 教程 ID | 标题        | 作者         | 提交日期       |
|-------|-----------|------------|------------|
| 1     | 学习 PHP    | 菜鸟教程       | 2017-04-12 |
| 2     | 学习 MySQL  | 菜鸟教程       | 2017-04-12 |
| 3     | JAVA 教程   | RUNOOB.COM | 2016-05-06 |
| 4     | 学习 Python | RUNOOB.COM | 2016-03-06 |



# MySQL WHERE 子句

我们知道从 MySQL 表中使用 SQL SELECT 语句来读取数据。  
如需有条件地从表中选取数据，可将 WHERE 子句添加到 SELECT 语句中。

## 语法

以下是 SQL SELECT 语句使用 WHERE 子句从数据表中读取数据的通用语法：

```
SELECT field1, field2,...fieldN FROM table_name1, table_name2...  
  
[WHERE condition1 [AND [OR]] condition2.....
```

查询语句中你可以使用一个或者多个表，表之间使用逗号，分割，并使用WHERE语句来设定查询条件。

你可以在 WHERE 子句中指定任何条件。

你可以使用 AND 或者 OR 指定一个或多个条件。

WHERE 子句也可以运用于 SQL 的 DELETE 或者 UPDATE 命令。

WHERE 子句类似于程序语言中的 if 条件，根据 MySQL 表中的字段值来读取指定的数据。

以下为操作符列表，可用于 WHERE 子句中。

下表中实例假定 A 为 10, B 为 20

| 操作符    | 描述   | 实例                |
|--------|--|-------------------|
| =      | 等号，检测两个值是否相等，如果相等返回true                        | (A = B) 返回false。  |
| <>, != | 不等于，检测两个值是否相等，如果不相等返回true                      | (A != B) 返回 true。 |
| >      | 大于号，检测左边的值是否大于右边的值, 如果左边的值大于右边的值返回true         | (A > B) 返回false。  |
| <      | 小于号，检测左边的值是否小于右边的值, 如果左边的值小于右边的值返回true         | (A < B) 返回 true。  |
| >=     | 大于等于号，检测左边的值是否大于或等于右边的值, 如果左边的值大于或等于右边的值返回true | (A >= B) 返回false。 |
| <=     | 小于等于号，检测左边的值是否小于或等于右边的值, 如果左边的值小于或等于右边的值返回true | (A <= B) 返回 true。 |

如果我们想在 MySQL 数据表中读取指定的数据，WHERE 子句是非常有用的。

使用主键来作为 WHERE 子句的条件查询是非常快速的。

如果给定的条件在表中没有任何匹配的记录，那么查询不会返回任何数据。

## 从命令提示符中读取数据

我们将在SQL SELECT语句使用WHERE子句来读取MySQL数据表 runoob\_tbl 中的数据：



实例

以下实例将读取 runoob\_tbl 表中 runoob\_author 字段值为 Sanjay 的所有记录：

SQL SELECT WHERE 子句

SELECT \* from runoob\_tbl WHERE runoob\_author='菜鸟教程';

输出结果：

```
mysql> SELECT * from runoob_tbl WHERE runoob_author='菜鸟教程';
```

| runoob_id | runoob_title | runoob_author | submission_date |
|-----------|--------------|---------------|-----------------|
| 1         | 学习 PHP       | 菜鸟教程          | 2017-04-12      |
| 2         | 学习 MySQL     | 菜鸟教程          | 2017-04-12      |

```
2 rows in set (0.02 sec)
```

MySQL 的 WHERE 子句的字符串比较是不区分大小写的。 你可以使用 BINARY 关键字来设定 WHERE 子句的字符串比较是区分大小写的。

如下实例：

BINARY 关键字

```
mysql> SELECT * from runoob_tbl WHERE BINARY runoob_author='runoob.com';
Empty set (0.01 sec)
mysql> SELECT * from runoob_tbl WHERE BINARY runoob_author='RUNOOB.COM';
```

| runoob_id | runoob_title | runoob_author | submission_date |
|-----------|--------------|---------------|-----------------|
| 3         | JAVA 教程      | RUNOOB.COM    | 2016-05-06      |
| 4         | 学习 Python    | RUNOOB.COM    | 2016-03-06      |

```
2 rows in set (0.01 sec)
```

实例中使用了 BINARY 关键字，是区分大小写的，所以 runoob\_author='runoob.com' 的查询条件是没有数据的。

## 使用PHP脚本读取数据

你可以使用 PHP 函数的 mysqli\_query() 及相同的 SQL SELECT 带上 WHERE 子句的命令来获取数据。

该函数用于执行 SQL 命令，然后通过 PHP 函数 mysqli\_fetch\_array() 来输出所有查询的数据。

### 实例

以下实例将从 runoob\_tbl 表中返回使用 runoob\_author 字段值为 RUNOOB.COM 的记录：

MySQL WHERE 子句测试：

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
// 读取 runoob_author 为 RUNOOB.COM 的数据
$sql = 'SELECT runoob_id, runoob_title,
runoob_author, submission_date
FROM runoob_tbl
WHERE runoob_author="RUNOOB.COM"';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 MySQL WHERE 子句测试<h2>';
echo '<table border="1"><tr><td>教程 ID</td><td>标题</td><td>作者</td><td>提交日期</td></tr>';
while($row = mysqli_fetch_array($retval, MYSQL_ASSOC))
```



```
{
echo "<tr><td> {$row['runoob_id']}</td> ".
"<td>{$row['runoob_title']} </td> ".
"<td>{$row['runoob_author']} </td> ".
"<td>{$row['submission_date']} </td> ".
"</tr>";
}
echo '</table>';
// 释放内存
mysqli_free_result($retval);
mysqli_close($conn);
?>
```

输出结果如下所示：

## 菜鸟教程 MySQL WHERE 子句测试

| 教程 ID | 标题        | 作者         | 提交日期       |
|-------|-----------|------------|------------|
| 3     | JAVA 教程   | RUNOOB.COM | 2016-05-06 |
| 4     | 学习 Python | RUNOOB.COM | 2016-03-06 |

[MySQL 查询数据](#)

[MySQL UPDATE 查询](#)

[点我分享笔记](#)

[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[MySQL WHERE 子句](#)

[MySQL DELETE 语句](#)

## MySQL UPDATE 查询

如果我们需要修改或更新 MySQL 中的数据，我们可以使用 SQL UPDATE 命令来操作。

### 语法

以下是 UPDATE 命令修改 MySQL 数据表数据的通用 SQL 语法：

```
UPDATE table_name SET field1=new-value1, field2=new-value2

[WHERE Clause]
```

你可以同时更新一个或多个字段。

你可以在 WHERE 子句中指定任何条件。

你可以在一个单独表中同时更新数据。

当你需要更新数据表中指定行的数据时 WHERE 子句是非常有用的。

通过命令提示符更新数据  
以下我们将在 SQL UPDATE 命令使用 WHERE 子句来更新 runoob\_tbl 表中指定的数据：

实例

以下实例将更新数据表中 runoob\_id 为 3 的 runoob\_title 字段值：

SQL UPDATE 语句：

```
mysql> UPDATE runoob_tbl SET runoob_title='学习 C++' WHERE runoob_id=3;
Query OK, 1 rows affected (0.01 sec)
mysql> SELECT * from runoob_tbl WHERE runoob_id=3;
+-----+-----+-----+-----+
| runoob_id | runoob_title | runoob_author | submission_date |
+-----+-----+-----+-----+
| 3 | 学习 C++ | RUNOOB.COM | 2016-05-06 |
+-----+-----+-----+-----+
1 rows in set (0.01 sec)
```

从结果上看，runoob\_id 为 3 的 runoob\_title 已被修改。

使用 PHP脚本更新数据

PHP 中使用函数 mysqli\_query() 来执行 SQL 语句，你可以在 SQL UPDATE 语句中使用或者不使用 WHERE 子句。

注意：不使用 WHERE 子句将数据表的全部数据进行更新，所以要慎重。

该函数与在mysql>命令提示符中执行SQL语句的效果是一样的。

实例

以下实例将更新 runoob\_id 为3的 runoob\_title 字段的数据。

MySQL UPDATE 语句测试：

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'UPDATE runoob_tbl
SET runoob_title="学习 Python"
WHERE runoob_id=3';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法更新数据: ' . mysqli_error($conn));
}
echo '数据更新成功! ';
mysqli_close($conn);
?>
```

2 篇笔记

#2

写笔记

update 语句可用来修改表中的数据,简单来说基本的使用形式为:  
update 表名称 set 列名称=新值 where 更新条件;  
以下是在表 students 中的实例:  
将 id 为 5 的手机号改为默认的 - : update students settel=default where id=5;  
将所有人的年龄增加 1: update students set age=age+1;

将手机号为 13288097888 的姓名改为 "小明", 年龄改为 19: update students setname="小明", age=19 wheretel="13288097888";

General1年前 (2017-05-08)

#1



UPDATE替换某个字段中的某个字符

当我们需要将字段中的特定字符串批量修改为其他字符串时，可已使用以下操作：

```
UPDATE table_name SET field=REPLACE(field, 'old-string', 'new-string')

[WHERE Clause]
```

实例：

以下实例将更新 runoob\_id 为 3 的runoob\_title 字段值的 "C++" 替换为 "Python"：

```
UPDATE runoob_tbl SET runoob_title = REPLACE(runoob_title, 'C++', 'Python') where

runoob_id = 3;
```

CarolLi1年前 (2017-08-21)

反馈/建议



## MySQL DELETE 语句

你可以使用 SQL 的 DELETE FROM 命令来删除 MySQL 数据表中的记录。

你可以在 `mysql>` 命令提示符或 PHP 脚本中执行该命令。

### 语法

以下是 SQL DELETE 语句从 MySQL 数据表中删除数据的通用语法：

```
DELETE FROM table_name [WHERE Clause]
```

如果没有指定 WHERE 子句，MySQL 表中的所有记录将被删除。

你可以在 WHERE 子句中指定任何条件

您可以在单个表中一次性删除记录。

当你想删除数据表中指定的记录时 WHERE 子句是非常有用的。

### 从命令行中删除数据

这里我们将在 SQL DELETE 命令中使用 WHERE 子句来删除 MySQL 数据表 runoob\_tbl 所选的数据。

### 实例

以下实例将删除 runoob\_tbl 表中 runoob\_id 为3 的记录:

DELETE 语句:

```
mysql> use RUNOOB;
Database changed
mysql> DELETE FROM runoob_tbl WHERE runoob_id=3;
Query OK, 1 row affected (0.23 sec)
```

使用 PHP 脚本删除数据

PHP使用 mysqli\_query() 函数来执行SQL语句， 你可以在 SQL DELETE 命令中使用或不使用 WHERE 子句。

该函数与 `mysql>` 命令符执行SQL命令的效果是一样的。

实例

以下PHP实例将删除 runoob\_tbl 表中 runoob\_id 为 3 的记录:

MySQL DELETE 子句测试:

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'DELETE FROM runoob_tbl
WHERE runoob_id=3';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法删除数据: ' . mysqli_error($conn));
}
echo '数据删除成功! ';
mysqli_close($conn);
?>
```

MySQL UPDATE 查询

MySQL LIKE 子句



1 篇笔记  
#1



delete 语句用于删除表中的数据, 基本用法为:  
delete from 表名称 where 删除条件;  
以下是在表 students 中的实例:  
删除 id 为 3 的行: delete from students where id=3;  
删除所有年龄小于 21 岁的数据: delete from students where age<20;  
删除表中的所有数据: delete from students;

General1年前 (2017-05-08)

写笔记

反馈/建议

## MySQL LIKE 子句

我们知道在 MySQL 中使用 SQL SELECT 命令来读取数据，同时我们可以在 SELECT 语句中使用 WHERE 子句来获取指定的记录。

WHERE 子句中可以使用等号 `=` 来设定获取数据的条件，如 `"runoob_author = 'RUNOOB.COM'"`。

但是有时候我们需要获取 runoob\_author 字段含有 `"COM"` 字符的所有记录，这时我们就需要在 WHERE 子句中使用 SQL LIKE 子句。

SQL LIKE 子句中使用百分号 `%` 字符来表示任意字符，类似于UNIX或正则表达式中的星号 `*`。

如果没有使用百分号 `%`，LIKE 子句与等号 `=` 的效果是一样的。

### 语法

以下是 SQL SELECT 语句使用 LIKE 子句从数据表中读取数据的通用语法：

```
SELECT field1, field2,...fieldN

FROM table_name

WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'
```

你可以在 WHERE 子句中指定任何条件。

你可以在 WHERE 子句中使用 LIKE 子句。

你可以使用 LIKE 子句代替等号 `=`。

LIKE 通常与 `%` 一同使用，类似于一个元字符的搜索。

你可以使用 AND 或者 OR 指定一个或多个条件。

你可以在 DELETE 或 UPDATE 命令中使用 WHERE...LIKE 子句来指定条件。

### 在命令提示符中使用 LIKE 子句

以下我们将在 SQL SELECT 命令中使用 WHERE...LIKE 子句来从MySQL数据表 runoob\_tbl 中读取数据。

### 实例

以下是我们将 runoob\_tbl 表中获取 runoob\_author 字段中以 `COM` 为结尾的的所有记录：

#### SQL LIKE 语句：

```
mysql> use RUNOOB;
Database changed
mysql> SELECT * from runoob_tbl WHERE runoob_author LIKE '%COM';
+-----+-----+-----+-----+
| runoob_id | runoob_title | runoob_author | submission_date |
+-----+-----+-----+-----+
| 3 | 学习 Java | RUNOOB.COM | 2015-05-01 |
| 4 | 学习 Python | RUNOOB.COM | 2016-03-06 |
+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

### 在PHP脚本中使用 LIKE 子句

你可以使用PHP函数的 mysqli\_query() 及相同的 SQL SELECT 带上 WHERE...LIKE 子句的命令来获取数据。

该函数用于执行 SQL 命令，然后通过 PHP 函数 mysqli\_fetch\_array() 来输出所有查询的数据。

但是如果是 DELETE 或者 UPDATE 中使用 WHERE...LIKE 子句的SQL语句，则无需使用 mysqli\_fetch\_array() 函数。

实例

以下是我们使用PHP脚本在 runoob\_tbl 表中读取 runoob\_author 字段中以 COM 为结尾的的所有记录：

MySQL LIKE 子句测试：

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'SELECT runoob_id, runoob_title,
runoob_author, submission_date
FROM runoob_tbl
WHERE runoob_author LIKE "%COM"';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 mysqli_fetch_array 测试<h2>';
echo '<table border="1"><tr><td>教程 ID</td><td>标题</td><td>作者</td><td>提交日期</td></tr>';
while($row = mysqli_fetch_array($retval, MYSQLI_ASSOC))
{
    echo "<tr><td> {$row['runoob_id']}</td> ".
    "<td>{$row['runoob_title']} </td> ".
    "<td>{$row['runoob_author']} </td> ".
    "<td>{$row['submission_date']} </td> ".
    "</tr>";
}
echo '</table>';
mysqli_close($conn);
?>
```

输出结果如下图所示：

菜鸟教程 mysqli\_fetch\_array 测试

| 教程 ID | 标题        | 作者         | 提交日期       |
|-------|-----------|------------|------------|
| 3     | 学习 Java   | RUNOOB.COM | 2015-05-01 |
| 4     | 学习 Python | RUNOOB.COM | 2016-03-06 |

MySQL DELETE 语句

MySQL 排序

1 篇笔记

#1

写笔记

like 匹配/模糊匹配，会与 % 和 \_ 结合使用。

- '%a' //以a结尾的数据
- 'a%' //以a开头的数据
- '%a%' //含有a的数据
- '\_a\_' //三位且中间字母是a的

```
'_a'      //两位且结尾字母是a的

'a_'      //两位且开头字母是a的
```

查询以 **java** 字段开头的信息。

```
SELECT * FROM position WHERE name LIKE 'java%';
```

查询包含 **java** 字段的信息。

```
SELECT * FROM position WHERE name LIKE '%java%';
```

查询以 **java** 字段结尾的信息。

```
SELECT * FROM position WHERE name LIKE '%java';
```

**tfbyty**5个月前 (04-24)

反馈/建议



# MySQL UNION 操作符

本教程为大家介绍 MySQL UNION 操作符的语法和实例。

## 描述

MySQL UNION 操作符用于连接两个以上的 SELECT 语句的结果组合到一个结果集合中。多个 SELECT 语句会删除重复的数据。

## 语法

MySQL UNION 操作符语法格式：

```
SELECT expression1, expression2, ... expression_n

FROM tables

[WHERE conditions]

UNION [ALL | DISTINCT]

SELECT expression1, expression2, ... expression_n

FROM tables

[WHERE conditions];
```

## 参数

**expression1, expression2, ... expression\_n:** 要检索的列。

**tables:** 要检索的数据表。

**WHERE conditions:** 可选， 检索条件。

**DISTINCT:** 可选，删除结果集中重复的数据。默认情况下 **UNION** 操作符已经删除了重复数据，所以 **DISTINCT** 修饰符对结果没啥影响。

**ALL:** 可选，返回所有结果集，包含重复数据。

## 演示数据库

在本教程中，我们将使用 RUNOOB 样本数据库。

下面是选自 "Websites" 表的数据：

```
mysql> SELECT * FROM Websites;

+----+-----+-----+-----+-----+
| id | name      | url                | alexa | country |
+----+-----+-----+-----+-----+
| 1  | Google    | https://www.google.cm/ | 1     | USA     |
| 2  | 淘宝      | https://www.taobao.com/ | 13    | CN      |
| 3  | 菜鸟教程  | http://www.runoob.com/  | 4689  | CN      |
| 4  | 微博      | http://weibo.com/      | 20    | CN      |
| 5  | Facebook  | https://www.facebook.com/ | 3     | USA     |
| 7  | stackoverflow | http://stackoverflow.com/ | 0     | IND     |
+----+-----+-----+-----+-----+
```

下面是 "apps" APP 的数据：

```
mysql> SELECT * FROM apps;

+----+-----+-----+-----+
| id | app_name  | url                | country |
+----+-----+-----+-----+
| 1  | QQ APP   | http://im.qq.com/  | CN      |
| 2  | 微博 APP | http://weibo.com/  | CN      |
| 3  | 淘宝 APP | https://www.taobao.com/ | CN      |
+----+-----+-----+-----+

3 rows in set (0.00 sec)
```



## SQL UNION 实例

下面的 SQL 语句从 "Websites" 和 "apps" 表中选取所有不同的 country（只有不同的值）：

### 实例

```
SELECT country FROM Websites
UNION
SELECT country FROM apps
ORDER BY country;
```

执行以上 SQL 输出结果如下：

```
mysql> SELECT country FROM Websites
-> UNION
-> SELECT country FROM apps
:
-> ORDER BY country;

+-----+
| country |
+-----+
| CN      |
| IND     |
| USA     |
+-----+
3 rows in set (0.00 sec)
```

注释：UNION 不能用于列出两个表中所有的 country。如果一些网站和 APP 来自同一个国家，每个国家只会列出一行。UNION 只会选取不同的值。请使用 UNION ALL 来选取重复的值！

## SQL UNION ALL 实例

下面的 SQL 语句使用 UNION ALL 从 "Websites" 和 "apps" 表中选取所有的 country（也有重复的值）：

### 实例

```
SELECT country FROM Websites
UNION ALL
SELECT country FROM apps
ORDER BY country;
```

执行以上 SQL 输出结果如下：

```
mysql> SELECT country FROM Websites
-> UNION ALL
-> SELECT country FROM apps
-> ORDER BY country;

+-----+
| country |
+-----+
| CN      |
| CN      |
| CN      |
| CN      |
| CN      |
| IND     |
| USA     |
| USA     |
| USA     |
+-----+
```

## 带有 WHERE 的 SQL UNION ALL

下面的 SQL 语句使用 UNION ALL 从 "Websites" 和 "apps" 表中选取所有的中国(CN)的数据（也有重复的值）：

### 实例

```
SELECT country, name FROM Websites
```

```
WHERE country='CN'
UNION ALL
SELECT country, app_name FROM apps
WHERE country='CN'
ORDER BY country;
```

执行以上 SQL 输出结果如下：

```
mysql> SELECT country, name FROM Websites
WHERE country='CN'
UNION ALL
SELECT country, app_name FROM apps
WHERE country='CN'
ORDER BY country;
+-----+-----+
| country | name          |
+-----+-----+
| CN      | 淘宝          |
| CN      | QQ APP        |
| CN      | 菜鸟教程      |
| CN      | 微博 APP      |
| CN      | 微博          |
| CN      | 淘宝 APP      |
+-----+-----+
6 rows in set (0.05 sec)
```

[MySQL GROUP BY 语句](#)

[MySQL 函数](#)



1 篇笔记  
#1



[写笔记](#)

**UNION** 语句：用于将不同表中相同列中查询的数据展示出来；（不包括重复数据）

**UNION ALL** 语句：用于将不同表中相同列中查询的数据展示出来；（包括重复数据）

使用形式如下：

```
SELECT 列名称 FROM 表名称 UNION SELECT 列名称 FROM 表名称 ORDER BY 列名称;
```

```
SELECT 列名称 FROM 表名称 UNION ALL SELECT 列名称 FROM 表名称 ORDER BY 列名称;
```

**BAI3**周前 (09-07)

[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[MySQL LIKE 子句](#)

[Mysql 连接的使用](#)

## MySQL 排序

我们知道从 MySQL 表中使用 SQL SELECT 语句来读取数据。

如果我们需要对读取的数据进行排序，我们就可以使用 MySQL 的 **ORDER BY** 子句来设定你想按哪个字段哪种方式来进行排序，再返回搜索结果。

语法

以下是 SQL SELECT 语句使用 ORDER BY 子句将查询数据排序后再返回数据：

```
SELECT field1, field2,...fieldN table_name1, table_name2...

ORDER BY field1, [field2...] [ASC [DESC]]
```

- 你可以使用任何字段来作为排序的条件，从而返回排序后的查询结果。
- 你可以设定多个字段来排序。
- 你可以使用 ASC 或 DESC 关键字来设置查询结果是按升序或降序排列。默认情况下，它是按升序排列。
- 你可以添加 WHERE...LIKE 子句来设置条件。

在命令提示符中使用 ORDER BY 子句

以下将在 SQL SELECT 语句中使用 ORDER BY 子句来读取MySQL 数据表 runoob\_tbl 中的数据：

实例

尝试以下实例，结果将按升序及降序排列。

SQL 排序

```
mysql> use RUNOOB;
Database changed
mysql> SELECT * from runoob_tbl ORDER BY submission_date ASC;
+-----+-----+-----+-----+
| runoob_id | runoob_title | runoob_author | submission_date |
+-----+-----+-----+-----+
| 3 | 学习 Java | RUNOOB.COM | 2015-05-01 |
| 4 | 学习 Python | RUNOOB.COM | 2016-03-06 |
| 1 | 学习 PHP | 菜鸟教程 | 2017-04-12 |
| 2 | 学习 MySQL | 菜鸟教程 | 2017-04-12 |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)
mysql> SELECT * from runoob_tbl ORDER BY submission_date DESC;
+-----+-----+-----+-----+
| runoob_id | runoob_title | runoob_author | submission_date |
+-----+-----+-----+-----+
| 1 | 学习 PHP | 菜鸟教程 | 2017-04-12 |
| 2 | 学习 MySQL | 菜鸟教程 | 2017-04-12 |
| 4 | 学习 Python | RUNOOB.COM | 2016-03-06 |
| 3 | 学习 Java | RUNOOB.COM | 2015-05-01 |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

读取 runoob\_tbl 表中所有数据并按 submission\_date 字段的升序排列。

在 PHP 脚本中使用 ORDER BY 子句

你可以使用PHP函数的 mysqli\_query() 及相同的 SQL SELECT 带上 ORDER BY 子句的命令来获取数据。

该函数用于执行 SQL 命令，然后通过 PHP 函数 mysqli\_fetch\_array() 来输出所有查询的数据。

实例

尝试以下实例，查询后的数据按 submission\_date 字段的降序排列后返回。

MySQL ORDER BY 测试：

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
```

```
die('连接失败: ' . mysqli_error($conn));
}
// 设置编码, 防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'SELECT runoob_id, runoob_title,
runoob_author, submission_date
FROM runoob_tbl
ORDER BY submission_date ASC';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 MySQL ORDER BY 测试</h2>';
echo '<table border="1"><tr><td>教程 ID</td><td>标题</td><td>作者</td><td>提交日期</td></tr>';
while($row = mysqli_fetch_array($retval, MYSQL_ASSOC))
{
echo "<tr><td> {$row['runoob_id']}</td> ".
"<td>{$row['runoob_title']} </td> ".
"<td>{$row['runoob_author']} </td> ".
"<td>{$row['submission_date']} </td> ".
"</tr>";
}
echo '</table>';
mysqli_close($conn);
?>
```

输出结果如下图所示:

## 菜鸟教程 MySQL ORDER BY 测试

| 教程 ID | 标题        | 作者         | 提交日期       |
|-------|-----------|------------|------------|
| 3     | 学习 Java   | RUNOOB.COM | 2015-05-01 |
| 4     | 学习 Python | RUNOOB.COM | 2016-03-06 |
| 1     | 学习 PHP    | 菜鸟教程       | 2017-04-12 |
| 2     | 学习 MySQL  | 菜鸟教程       | 2017-04-12 |

MySQL LIKE 子句

Mysql 连接的使用



1 篇笔记  
#1

写笔记

MySQL 排序我们知道从 MySQL 表中使用 SQL SELECT 语句来读取:

MySQL 拼音排序

如果字符集采用的是 gbk(汉字编码字符集), 直接在查询语句后边添加 ORDER BY:

```
SELECT *

FROM runoob_tbl

ORDER BY runoob_title;
```

如果字符集采用的是 utf8(万国码), 需要先对字段进行转码然后排序:

```
SELECT *

FROM runoob_tbl

ORDER BY CONVERT(runoob_title using gbk);
```

[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)[MySQL 导入数据](#)[MySQL UNION 操作符](#)

## MySQL GROUP BY 语句

GROUP BY 语句根据一个或多个列对结果集进行分组。

在分组的列上我们可以使用 COUNT, SUM, AVG,等函数。

### GROUP BY 语法

```
SELECT column_name, function(column_name)

FROM table_name

WHERE column_name operator value

GROUP BY column_name;
```

### 实例演示

本章节实例使用到了以下表结构及数据，使用前我们可以先将以下数据导入数据库中。

```
SET NAMES utf8;

SET FOREIGN_KEY_CHECKS = 0;

-- -----
-- Table structure for `employee_tbl`
-- -----

DROP TABLE IF EXISTS `employee_tbl`;

CREATE TABLE `employee_tbl` (

  `id` int(11) NOT NULL,

  `name` char(10) NOT NULL DEFAULT '',

  `date` datetime NOT NULL,

  `singin` tinyint(4) NOT NULL DEFAULT '0' COMMENT '登录次数',
```

```

PRIMARY KEY (`id`))

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Records of `employee_tbl`
--

BEGIN;

INSERT INTO `employee_tbl` VALUES ('1', '小明', '2016-04-22 15:25:33', '1'), ('2', '小王', '2016-04-20 15:25:47', '3'),
, ('3', '小丽', '2016-04-19 15:26:02', '2'), ('4', '小王', '2016-04-07 15:26:14', '4'), ('5', '小明', '2016-04-11 15:26:40', '4'), ('6', '小明', '2016-04-04 15:26:54', '2');

COMMIT;

SET FOREIGN_KEY_CHECKS = 1;

```

导入成功后，执行以下 SQL 语句：

```

mysql> set names utf8;

mysql> SELECT * FROM employee_tbl;

+----+-----+-----+-----+
| id | name  | date                | singin |
+----+-----+-----+-----+
|  1 | 小明  | 2016-04-22 15:25:33 |      1 |
|  2 | 小王  | 2016-04-20 15:25:47 |      3 |
|  3 | 小丽  | 2016-04-19 15:26:02 |      2 |
|  4 | 小王  | 2016-04-07 15:26:14 |      4 |
|  5 | 小明  | 2016-04-11 15:26:40 |      4 |
|  6 | 小明  | 2016-04-04 15:26:54 |      2 |
+----+-----+-----+-----+

6 rows in set (0.00 sec)

```

接下来我们使用 **GROUP BY** 语句 将数据表按名字进行分组，并统计每个人有多少条记录：

```

mysql> SELECT name, COUNT(*) FROM employee_tbl GROUP BY name;

+-----+-----+
| name  | COUNT(*) |
+-----+-----+

```

```
+-----+-----+
| 小丽 |          1 |
| 小明 |          3 |
| 小王 |          2 |
+-----+-----+
3 rows in set (0.01 sec)
```

## 使用 WITH ROLLUP

WITH ROLLUP 可以实现在分组统计数据基础上再进行相同的统计（SUM,AVG,COUNT...）。

例如我们将以上的数据表按名字进行分组，再统计每个人登录的次数：

```
mysql> SELECT name, SUM(singin) as singin_count FROM employee_tbl GROUP BY name WITH ROLLUP;

+-----+-----+
| name | singin_count |
+-----+-----+
| 小丽 |          2 |
| 小明 |          7 |
| 小王 |          7 |
| NULL |         16 |
+-----+-----+
4 rows in set (0.00 sec)
```

其中记录 NULL 表示所有人的登录次数。

我们可以使用 coalesce 来设置一个可以取代 NULL 的名称，coalesce 语法：

```
select coalesce(a,b,c);
```

参数说明：如果a==null,则选择b；如果b==null,则选择c；如果a!=null,则选择a；如果a b c 都为null ，则返回为null（没意义）。

以下实例中如果名字为空我们使用总数代替：

```
mysql> SELECT coalesce(name, '总数'), SUM(singin) as singin_count FROM employee_tbl GROUP BY name WITH ROLLUP;

+-----+-----+
| coalesce(name, '总数') | singin_count |
+-----+-----+
| 小丽 |          2 |
```

|                          |    |
|--------------------------|----|
| 小明                       | 7  |
| 小王                       | 7  |
| 总数                       | 16 |
| +-----+-----+            |    |
| 4 rows in set (0.01 sec) |    |

1 篇笔记

#1

写笔记

1、group by 可以实现一个最简单的去重查询，假设想看下有哪些员工，除了用 distinct,还可以用：

```
SELECT name FROM employee_tb1 GROUP BY name;
```

返回的结果集就是所有员工的名字。  
2、group by 语句用法有一个注意点，在 select 语句中，所查询的字段除了聚合函数（SUM,AVG,COUNT...）以外 必须只能是分组的字段，举例：

```
SELECT id,name,count(*) FROM employee_tb1 GROUP BY name;
```

运行该语句程序会报错，因为 id 字段并不包含在 GROUP BY 分组中，改为如下：

```
SELECT id,name ,count(*)FROM employee_tb1 GROUP BY id,name;
```

则程序可以正常运行。  
3、分组后的条件使用 HAVING 来限定，WHERE 是对原始数据进行条件限制。几个关键字的使用顺序为 where 、group by、having、order by，例如：

```
SELECT name ,sum(*) FROM employee_tb1 WHERE id<>1 GROUP BY name HAVING sum(>5 ORDER BY sum(*) DESC;
```

CAI4周前 (09-05)

反馈/建议

# Mysql 连接的使用

在前几章节中，我们已经学会了如何在一张表中读取数据，这是相对简单的，但是在真正的应用中经常需要从多个数据表中读取数据。

本章节我们将向大家介绍如何使用 MySQL 的 JOIN 在两个或多个表中查询数据。

你可以在 SELECT, UPDATE 和 DELETE 语句中使用 Mysql 的 JOIN 来联合多表查询。

JOIN 按照功能大致分为如下三类：



**INNER JOIN**（内连接,或等值连接）：获取两个表中字段匹配关系的记录。

**LEFT JOIN**（左连接）：获取左表所有记录，即使右表没有对应匹配的记录。

**RIGHT JOIN**（右连接）：与 **LEFT JOIN** 相反，用于获取右表所有记录，即使左表没有对应匹配的记录。

本章节使用的数据库结构及数据下载：[runoob-mysql-join-test.sql](#)。

## 在命令提示符中使用 INNER JOIN

我们在RUNOOB数据库中有两张表 `tcount_tbl` 和 `runoob_tbl`。两张数据表数据如下：

### 实例

尝试以下实例：

#### 测试实例数据

```
mysql> use RUNOOB;
Database changed
mysql> SELECT * FROM tcount_tbl;
+-----+-----+
| runoob_author | runoob_count |
+-----+-----+
| 菜鸟教程     | 10          |
| RUNOOB.COM   | 20          |
| Google       | 22          |
+-----+-----+
3 rows in set (0.01 sec)
mysql> SELECT * from runoob_tbl;
+-----+-----+-----+-----+
| runoob_id | runoob_title | runoob_author | submission_date |
+-----+-----+-----+-----+
| 1         | 学习 PHP    | 菜鸟教程     | 2017-04-12     |
| 2         | 学习 MySQL  | 菜鸟教程     | 2017-04-12     |
| 3         | 学习 Java   | RUNOOB.COM   | 2015-05-01     |
| 4         | 学习 Python | RUNOOB.COM   | 2016-03-06     |
| 5         | 学习 C      | FK           | 2017-04-05     |
+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

接下来我们就使用MySQL的**INNER JOIN**(也可以省略 **INNER** 使用 **JOIN**，效果一样)来连接以上两张表来读取`runoob_tbl`表中所有`runoob_author`字段在`tcount_tbl`表对应的`runoob_count`字段值：

#### INNER JOIN

```
mysql> SELECT a.runoob_id, a.runoob_author, b.runoob_count FROM runoob_tbl a INNER JOIN tcount_tbl b ON a.runoob_a
uthor = b.runoob_author;
+-----+-----+-----+
| a.runoob_id | a.runoob_author | b.runoob_count |
+-----+-----+-----+
| 1           | 菜鸟教程       | 10             |
| 2           | 菜鸟教程       | 10             |
| 3           | RUNOOB.COM     | 20             |
| 4           | RUNOOB.COM     | 20             |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

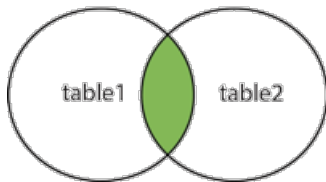
以上 SQL 语句等价于：

#### WHERE 子句

```
mysql> SELECT a.runoob_id, a.runoob_author, b.runoob_count FROM runoob_tbl a, tcount_tbl b WHERE a.runoob_author =
b.runoob_author;
+-----+-----+-----+
| a.runoob_id | a.runoob_author | b.runoob_count |
+-----+-----+-----+
| 1           | 菜鸟教程       | 10             |
| 2           | 菜鸟教程       | 10             |
| 3           | RUNOOB.COM     | 20             |
| 4           | RUNOOB.COM     | 20             |
+-----+-----+-----+
```

4 rows in set (0.01 sec)

### INNER JOIN



## MySQL LEFT JOIN

MySQL left join 与 join 有所不同。MySQL LEFT JOIN 会读取左边数据表的全部数据，即便右边表无对应数据。

### 实例

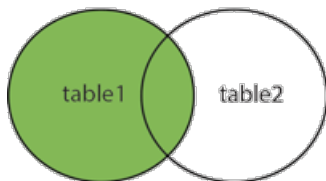
尝试以下实例，以 `runoob_tbl` 为左表，`tcount_tbl` 为右表，理解 MySQL LEFT JOIN 的应用：

### LEFT JOIN

```
mysql> SELECT a.runoob_id, a.runoob_author, b.runoob_count FROM runoob_tbl a LEFT JOIN tcount_tbl b ON a.runoob_au
thor = b.runoob_author;
+-----+-----+-----+
| a.runoob_id | a.runoob_author | b.runoob_count |
+-----+-----+-----+
| 1 | 菜鸟教程 | 10 |
| 2 | 菜鸟教程 | 10 |
| 3 | RUNOOB.COM | 20 |
| 4 | RUNOOB.COM | 20 |
| 5 | FK | NULL |
+-----+-----+-----+
5 rows in set (0.01 sec)
```

以上实例中使用了 LEFT JOIN，该语句会读取左边的数据表 `runoob_tbl` 的所有选取的字段数据，即便在右侧表 `tcount_tbl` 中没有对应的 `runoob_autho` r 字段值。

### LEFT JOIN



## MySQL RIGHT JOIN

MySQL RIGHT JOIN 会读取右边数据表的全部数据，即便左边边表无对应数据。

### 实例

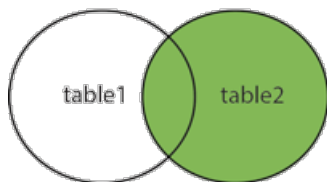
尝试以下实例，以 `runoob_tbl` 为左表，`tcount_tbl` 为右表，理解MySQL RIGHT JOIN的应用：

### RIGHT JOIN

```
mysql> SELECT a.runoob_id, a.runoob_author, b.runoob_count FROM runoob_tbl a RIGHT JOIN tcount_tbl b ON a.runoob_a
uthor = b.runoob_author;
+-----+-----+-----+
| a.runoob_id | a.runoob_author | b.runoob_count |
+-----+-----+-----+
| 1 | 菜鸟教程 | 10 |
| 2 | 菜鸟教程 | 10 |
| 3 | RUNOOB.COM | 20 |
| 4 | RUNOOB.COM | 20 |
| NULL | NULL | 22 |
+-----+-----+-----+
5 rows in set (0.01 sec)
```

以上实例中使用了 RIGHT JOIN，该语句会读取右边的数据表 `tcount_tbl` 的所有选取的字段数据，即便在左侧表 `runoob_tbl` 中没有对应的 `runoob_auth` or 字段值。

## RIGHT JOIN



## 在 PHP 脚本中使用 JOIN

PHP 中使用 `mysqli_query()` 函数来执行 SQL 语句，你可以使用以上的相同的 SQL 语句作为 `mysqli_query()` 函数的参数。

尝试如下实例：

### MySQL ORDER BY 测试：

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
$sql = 'SELECT a.runoob_id, a.runoob_author, b.runoob_count FROM runoob_tbl a INNER JOIN tcount_tbl b ON a.runoob_
author = b.runoob_author';
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 MySQL JOIN 测试<h2>';
echo '<table border="1"><tr><td>教程 ID</td><td>作者</td><td>登陆次数</td></tr>';
while($row = mysqli_fetch_array($retval, MYSQL_ASSOC))
{
    echo "<tr><td> {$row['runoob_id']}</td> ".
    "<td>{$row['runoob_author']} </td> ".
    "<td>{$row['runoob_count']} </td> ".
    "</tr>";
}
echo '</table>';
mysqli_close($conn);
?>
```

输出结果如下图所示：

### 菜鸟教程 MySQL JOIN 测试

| 教程 ID | 作者         | 登陆次数 |
|-------|------------|------|
| 1     | 菜鸟教程       | 10   |
| 2     | 菜鸟教程       | 10   |
| 3     | RUNOOB.COM | 20   |
| 4     | RUNOOB.COM | 20   |

[MySQL 排序](#)

[MySQL NULL 值处理](#)

[点我分享笔记](#)

[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)[MySQL 连接的使用](#)[MySQL 正则表达式](#)

## MySQL NULL 值处理

我们已经知道 MySQL 使用 SQL SELECT 命令及 WHERE 子句来读取数据表中的数据,但是当提供的查询条件字段为 NULL 时,该命令可能就无法正常工作。

为了处理这种情况,MySQL提供了三大运算符:

**IS NULL:** 当列的值是 NULL,此运算符返回 true。

**IS NOT NULL:** 当列的值不为 NULL,运算符返回 true。

**<=>:** 比较操作符(不同于=运算符),当比较的两个值为 NULL 时返回 true。

关于 NULL 的条件比较运算比较特殊。你不能使用 = NULL 或 != NULL 在列中查找 NULL 值。

在 MySQL 中, NULL 值与任何其它值的比较(即使是 NULL)永远返回 false,即 NULL = NULL 返回 false。

MySQL 中处理 NULL 使用 IS NULL 和 IS NOT NULL 运算符。

**注意:**

```
select * , columnName1+ifnull(columnName2,0) from tableName;
```

columnName1, columnName2 为 int 型,当 columnName2 中,有值为 null 时, columnName1+columnName2=null, ifnull(columnName2,0) 把 columnName2 中 null 值转为 0。

## 在命令提示符中使用 NULL 值

以下实例中假设数据库 RUNOOB 中的表 runoob\_test\_tbl 含有两列 runoob\_author 和 runoob\_count, runoob\_count 中设置插入 NULL 值。

### 实例

尝试以下实例:

#### 创建数据表 runoob\_test\_tbl

```
root@host# mysql -u root -p password;
Enter password:*****
mysql> use RUNOOB;
Database changed
mysql> create table runoob_test_tbl
-> (
-> runoob_author varchar(40) NOT NULL,
-> runoob_count INT
-> );
Query OK, 0 rows affected (0.05 sec)
mysql> INSERT INTO runoob_test_tbl (runoob_author, runoob_count) values ('RUNOOB', 20);
mysql> INSERT INTO runoob_test_tbl (runoob_author, runoob_count) values ('菜鸟教程', NULL);
mysql> INSERT INTO runoob_test_tbl (runoob_author, runoob_count) values ('Google', NULL);
mysql> INSERT INTO runoob_test_tbl (runoob_author, runoob_count) values ('FK', 20);
mysql> SELECT * from runoob_test_tbl;
+-----+-----+
| runoob_author | runoob_count |
+-----+-----+
```

```

+-----+
| RUNOOB | 20 |
| 菜鸟教程 | NULL |
| Google | NULL |
| FK | 20 |
+-----+
4 rows in set (0.01 sec)

```

以下实例中你可以看到 = 和 != 运算符是不起作用的：

```

mysql> SELECT * FROM runoob_test_tbl WHERE runoob_count = NULL;
Empty set (0.00 sec)
mysql> SELECT * FROM runoob_test_tbl WHERE runoob_count != NULL;
Empty set (0.01 sec)

```

查找数据表中 runoob\_test\_tbl 列是否为 NULL，必须使用 **IS NULL** 和 **IS NOT NULL**，如下实例：

```

mysql> SELECT * FROM runoob_test_tbl WHERE runoob_count IS NULL;
+-----+
| runoob_author | runoob_count |
+-----+
| 菜鸟教程 | NULL |
| Google | NULL |
+-----+
2 rows in set (0.01 sec)
mysql> SELECT * from runoob_test_tbl WHERE runoob_count IS NOT NULL;
+-----+
| runoob_author | runoob_count |
+-----+
| RUNOOB | 20 |
| FK | 20 |
+-----+
2 rows in set (0.01 sec)

```

## 使用 PHP 脚本处理 NULL 值

PHP 脚本中你可以在 if...else 语句来处理变量是否为空，并生成相应的条件语句。

以下实例中 PHP 设置了 \$runoob\_count 变量，然后使用该变量与数据表中的 runoob\_count 字段进行比较：

### MySQL ORDER BY 测试：

```

<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
mysqli_query($conn , "set names utf8");
if( isset($runoob_count ))
{
    $sql = "SELECT runoob_author, runoob_count
FROM runoob_test_tbl
WHERE runoob_count = $runoob_count";
}
else
{
    $sql = "SELECT runoob_author, runoob_count
FROM runoob_test_tbl
WHERE runoob_count IS NULL";
}
mysqli_select_db( $conn, 'RUNOOB' );
$retval = mysqli_query( $conn, $sql );
if(! $retval )
{
    die('无法读取数据: ' . mysqli_error($conn));
}
echo '<h2>菜鸟教程 IS NULL 测试<h2>';

```

```
echo '<table border="1"><tr><td>作者</td><td>登陆次数</td></tr>';
while($row = mysqli_fetch_array($retval, MYSQL_ASSOC))
{
echo "<tr>".
"<td>{$row['runoob_author']}</td> ".
"<td>{$row['runoob_count']}</td> ".
"</tr>";
}
echo '</table>';
mysqli_close($conn);
?>
```

输出结果如下图所示：

## 菜鸟教程 IS NULL 测试

|        |      |
|--------|------|
| 作者     | 登陆次数 |
| 菜鸟教程   |      |
| Google |      |

[MySQL 连接的使用](#)

[MySQL 正则表达式](#)

[点我分享笔记](#)

[反馈/建议](#)



[MySQL NULL 值处理](#)

[MySQL 事务](#)

## MySQL 正则表达式

在前面的章节我们已经了解到MySQL可以通过 **LIKE ...%** 来进行模糊匹配。

MySQL 同样也支持其他正则表达式的匹配，MySQL中使用 **REGEXP** 操作符来进行正则表达式匹配。

如果您了解PHP或Perl，那么操作起来就非常简单，因为MySQL的正则表达式匹配与这些脚本的类似。

下表中的正则模式可应用于 **REGEXP** 操作符中。

| 模式    | 描述   |
|-------|--|
| ^     | 匹配输入字符串的开始位置。如果设置了 <b>RegExp</b> 对象的 <b>Multiline</b> 属性，^ 也匹配 '\n' 或 '\r' 之后的位置。  |
| \$    | 匹配输入字符串的结束位置。如果设置了 <b>RegExp</b> 对象的 <b>Multiline</b> 属性，\$ 也匹配 '\n' 或 '\r' 之前的位置。 |
| .     | 匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符，请使用象 '[\n]' 的模式。                             |
| [...] | 字符集合。匹配所包含的任意一个字符。例如， '[abc]' 可以匹配 "plain" 中的 'a'。                                 |

|          |  |
|----------|--|
| [^...]   | 负值字符集合。匹配未包含的任意字符。例如， <code>['^abc']</code> 可以匹配 <code>"plain"</code> 中的 <code>'p'</code> 。  |
| p1 p2 p3 | 匹配 <code>p1</code> 或 <code>p2</code> 或 <code>p3</code> 。例如， <code>'z food'</code> 能匹配 <code>"z"</code> 或 <code>"food"</code> 。 <code>'(z f)ood'</code> 则匹配 <code>"zood"</code> 或 <code>"food"</code> 。 |
| *        | 匹配前面的子表达式零次或多次。例如， <code>zo*</code> 能匹配 <code>"z"</code> 以及 <code>"zoo"</code> 。 <code>*</code> 等价于 <code>{0,}</code> 。  |
| +        | 匹配前面的子表达式一次或多次。例如， <code>'zo+'</code> 能匹配 <code>"zo"</code> 以及 <code>"zoo"</code> ，但不能匹配 <code>"z"</code> 。 <code>+</code> 等价于 <code>{1,}</code> 。   |
| {n}      | <code>n</code> 是一个非负整数。匹配确定的 <code>n</code> 次。例如， <code>'o{2}'</code> 不能匹配 <code>"Bob"</code> 中的 <code>'o'</code> ，但是能匹配 <code>"food"</code> 中的两个 <code>o</code> 。                                     |
| {n,m}    | <code>m</code> 和 <code>n</code> 均为非负整数，其中 <code>n &lt;= m</code> 。最少匹配 <code>n</code> 次且最多匹配 <code>m</code> 次。   |

## 实例

了解以上的正则需求后，我们就可以根据自己的需求来编写带有正则表达式的SQL语句。以下我们将列出几个小实例(表名：`person_tbl`)来加深我们的理解：

查找`name`字段中以`'st'`为开头的所有数据：

```
mysql> SELECT name FROM person_tbl WHERE name REGEXP '^st';
```

查找`name`字段中以`'ok'`为结尾的所有数据：

```
mysql> SELECT name FROM person_tbl WHERE name REGEXP 'ok$';
```

查找`name`字段中包含`'mar'`字符串的所有数据：

```
mysql> SELECT name FROM person_tbl WHERE name REGEXP 'mar';
```

查找`name`字段中以元音字符开头或以`'ok'`字符串结尾的所有数据：

```
mysql> SELECT name FROM person_tbl WHERE name REGEXP '^[aeiou]|ok$';
```



## MySQL 事务

MySQL 事务主要用于处理操作量大，复杂度高的数据。比如说，在人员管理系统中，你删除一个人员，你即需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！

在 MySQL 中只有使用了 **InnoDB** 数据库引擎的数据库或表才支持事务。

事务处理可以用来维护数据库的完整性，保证成批的 **SQL** 语句要么全部执行，要么全部不执行。

事务用来管理 **insert,update,delete** 语句

一般来说，事务是必须满足4个条件（**ACID**）：：原子性（**Atomicity**，或称不可分割性）、一致性（**Consistency**）、隔离性（**Isolation**，又称独立性）、持久性（**Durability**）。

**原子性**：一个事务（**transaction**）中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（**Rollback**）到事务开始前的状态，就像这个事务从来没有执行过一样。

**一致性**：在事务开始之前和事务结束以后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作。

**隔离性**：数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交（**Read uncommitted**）、读提交（**read committed**）、可重复读（**repeatable read**）和串行化（**Serializable**）。

**持久性**：事务处理结束后，对数据的修改就是永久的，即便系统故障也不会丢失。

在 **MySQL** 命令行的默认设置下，事务都是自动提交的，即执行 **SQL** 语句后就会马上执行 **COMMIT** 操作。因此要显式地开启一个事务务须使用命令 **BEGIN** 或 **START TRANSACTION**，或者执行命令 **SET AUTOCOMMIT=0**，用来禁止使用当前会话的自动提交。

### 事务控制语句：

**BEGIN**或**START TRANSACTION**：显式地开启一个事务；

**COMMIT**；也可以使用**COMMIT WORK**，不过二者是等价的。**COMMIT**会提交事务，并使已对数据库进行的所有修改成为永久性的；

**ROLLBACK**；有可以使用**ROLLBACK WORK**，不过二者是等价的。回滚会结束用户的事务，并撤销正在进行的所有未提交的修改；

**SAVEPOINT identifier**；**SAVEPOINT**允许在事务中创建一个保存点，一个事务中可以有多多个**SAVEPOINT**；

**RELEASE SAVEPOINT identifier**；删除一个事务的保存点，当没有指定的保存点时，执行该语句会抛出一个异常；

**ROLLBACK TO identifier**；把事务回滚到标记点；

**SET TRANSACTION**；用来设置事务的隔离级别。**InnoDB**存储引擎提供事务的隔离级别有**READ UNCOMMITTED**、**READ COMMITTED**、**REPEATABLE READ**和**SERIALIZABLE**。

### MYSQL 事务处理主要有两种方法：

1、用 **BEGIN**, **ROLLBACK**, **COMMIT**来实现

**BEGIN** 开始一个事务

**ROLLBACK** 事务回滚

**COMMIT** 事务确认

2、直接用 **SET** 来改变 **MySQL** 的自动提交模式：

**SET AUTOCOMMIT=0** 禁止自动提交



## 事务测试

```
mysql> use RUNOOB;
Database changed
mysql> CREATE TABLE runoob_transaction_test( id int(5)) engine=innodb; # 创建数据表
Query OK, 0 rows affected (0.04 sec)
mysql> select * from runoob_transaction_test;
Empty set (0.01 sec)
mysql> begin; # 开始事务
Query OK, 0 rows affected (0.00 sec)
mysql> insert into runoob_transaction_test value(5);
Query OK, 1 rows affected (0.01 sec)
mysql> insert into runoob_transaction_test value(6);
Query OK, 1 rows affected (0.00 sec)
mysql> commit; # 提交事务
Query OK, 0 rows affected (0.01 sec)
mysql> select * from runoob_transaction_test;
+-----+
| id |
+-----+
| 5 |
| 6 |
+-----+
2 rows in set (0.01 sec)
mysql> begin; # 开始事务
Query OK, 0 rows affected (0.00 sec)
mysql> insert into runoob_transaction_test values(7);
Query OK, 1 rows affected (0.00 sec)
mysql> rollback; # 回滚
Query OK, 0 rows affected (0.00 sec)
mysql> select * from runoob_transaction_test; # 因为回滚所以数据没有插入
+-----+
| id |
+-----+
| 5 |
| 6 |
+-----+
2 rows in set (0.01 sec)
mysql>
```

## PHP中使用事务实例

### MySQL ORDER BY 测试:

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('连接失败: ' . mysqli_error($conn));
}
// 设置编码, 防止中文乱码
mysqli_query($conn, "set names utf8");
mysqli_select_db( $conn, 'RUNOOB' );
mysqli_query($conn, "SET AUTOCOMMIT=0"); // 设置为不自动提交, 因为MySQL默认立即执行
mysqli_begin_transaction($conn); // 开始事务定义
if(!mysqli_query($conn, "insert into runoob_transaction_test (id) values(8)"))
{
    mysqli_query($conn, "ROLLBACK"); // 判断当执行失败时回滚
}
if(!mysqli_query($conn, "insert into runoob_transaction_test (id) values(9)"))
{
    mysqli_query($conn, "ROLLBACK"); // 判断执行失败时回滚
}
mysqli_commit($conn); // 执行事务
mysqli_close($conn);
?>
```

点我分享笔记

反馈/建议



# MySQL ALTER命令

当我们需要修改数据表名或者修改数据表字段时，就需要使用到MySQL ALTER命令。

开始本章教程前让我们先创建一张表，表名为：testalter\_tbl。

```
root@host# mysql -u root -p password;

Enter password:*****

mysql> use RUNOOB;

Database changed

mysql> create table testalter_tbl

-> (

-> i INT,

-> c CHAR(1)

-> );

Query OK, 0 rows affected (0.05 sec)

mysql> SHOW COLUMNS FROM testalter_tbl;

+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| i     | int(11) | YES  |     | NULL    |       |
| c     | char(1) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

```
2 rows in set (0.00 sec)
```

## 删除，添加或修改表字段

如下命令使用了 **ALTER** 命令及 **DROP** 子句来删除以上创建表的 **i** 字段：

```
mysql> ALTER TABLE testalter_tbl DROP i;
```

如果数据表中只剩余一个字段则无法使用**DROP**来删除字段。

**MySQL** 中使用 **ADD** 子句来向数据表中添加列，如下实例在表 **testalter\_tbl** 中添加 **i** 字段，并定义数据类型：

```
mysql> ALTER TABLE testalter_tbl ADD i INT;
```

执行以上命令后，**i** 字段会自动添加到数据表字段的末尾。

```
mysql> SHOW COLUMNS FROM testalter_tbl;

+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| c     | char(1) | YES  |     | NULL    |       |
| i     | int(11) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+

2 rows in set (0.00 sec)
```

如果你需要指定新增字段的位置，可以使用**MySQL**提供的关键字 **FIRST** (设定位第一列)，**AFTER** 字段名（设定位于某个字段之后）。

尝试以下 **ALTER TABLE** 语句, 在执行成功后，使用 **SHOW COLUMNS** 查看表结构的变化：

```
ALTER TABLE testalter_tbl DROP i;

ALTER TABLE testalter_tbl ADD i INT FIRST;

ALTER TABLE testalter_tbl DROP i;

ALTER TABLE testalter_tbl ADD i INT AFTER c;
```

**FIRST** 和 **AFTER** 关键字只占用于 **ADD** 子句，所以如果你想重置数据表字段的位置就需要先使用 **DROP** 删除字段然后使用 **ADD** 来添加字段并设置位置。

## 修改字段类型及名称

如果需要修改字段类型及名称，你可以在**ALTER**命令中使用 **MODIFY** 或 **CHANGE** 子句。

例如，把字段 **c** 的类型从 **CHAR(1)** 改为 **CHAR(10)**，可以执行以下命令：

```
mysql> ALTER TABLE testalter_tbl MODIFY c CHAR(10);
```

使用 **CHANGE** 子句, 语法有很大的不同。在 **CHANGE** 关键字之后, 紧跟着的是你要修改的字段名, 然后指定新字段名及类型。尝试如下实例:

```
mysql> ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

```
mysql> ALTER TABLE testalter_tbl CHANGE j j INT;
```

## ALTER TABLE 对 Null 值和默认值的影响

当你修改字段时, 你可以指定是否包含值或者是否设置默认值。

以下实例, 指定字段 **j** 为 **NOT NULL** 且默认值为**100**。

```
mysql> ALTER TABLE testalter_tbl  
-> MODIFY j BIGINT NOT NULL DEFAULT 100;
```

如果你不设置默认值, **MySQL**会自动设置该字段默认为 **NULL**。

## 修改字段默认值

你可以使用 **ALTER** 来修改字段的默认值, 尝试以下实例:

```
mysql> ALTER TABLE testalter_tbl ALTER i SET DEFAULT 1000;  
  
mysql> SHOW COLUMNS FROM testalter_tbl;  
  
+-----+-----+-----+-----+-----+-----+  
| Field | Type   | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| c     | char(1) | YES  |     | NULL    |       |  
| i     | int(11) | YES  |     | 1000    |       |  
+-----+-----+-----+-----+-----+-----+  
  
2 rows in set (0.00 sec)
```

你也可以使用 **ALTER** 命令及 **DROP**子句来删除字段的默认值, 如下实例:

```
mysql> ALTER TABLE testalter_tbl ALTER i DROP DEFAULT;  
  
mysql> SHOW COLUMNS FROM testalter_tbl;  
  
+-----+-----+-----+-----+-----+-----+
```

| Field | Type    | Null | Key | Default | Extra |
|-------|---------|------|-----|---------|-------|
| c     | char(1) | YES  |     | NULL    |       |
| i     | int(11) | YES  |     | NULL    |       |

2 rows in set (0.00 sec)

Changing a Table Type:

修改数据表类型，可以使用 **ALTER** 命令及 **TYPE** 子句来完成。尝试以下实例，我们将表 **testalter\_tbl** 的类型修改为 **MYISAM**：

**注意：**查看数据表类型可以使用 **SHOW TABLE STATUS** 语句。

```
mysql> ALTER TABLE testalter_tbl ENGINE = MYISAM;

mysql> SHOW TABLE STATUS LIKE 'testalter_tbl'\G

***** 1. row *****

      Name: testalter_tbl

      Type: MyISAM

Row_format: Fixed

      Rows: 0

Avg_row_length: 0

      Data_length: 0

Max_data_length: 25769803775

      Index_length: 1024

      Data_free: 0

Auto_increment: NULL

      Create_time: 2007-06-03 08:04:36

      Update_time: 2007-06-03 08:04:36

      Check_time: NULL

Create_options:

      Comment:

1 row in set (0.00 sec)
```

## 修改表名

如果需要修改数据表的名称，可以在 **ALTER TABLE** 语句中使用 **RENAME** 子句来实现。

尝试以下实例将数据表 **testalter\_tbl** 重命名为 **alter\_tbl**：

```
mysql> ALTER TABLE testalter_tbl RENAME TO alter_tbl;
```

ALTER 命令还可以用来创建及删除MySQL数据表的索引，该功能我们会在接下来的章节中介绍。



1 篇笔记  
#1

写笔记



alter其他用途：  
修改存储引擎：修改为myisam

```
alter table tableName engine=myisam;
```

删除外键约束：keyName是外键别名

```
alter table tableName drop foreign key keyName;
```

修改字段的相对位置：这里name1为想要修改的字段，type1为该字段原来类型，first和after二选一，这应该显而易见，first放在第一位，after放在name2字段后面

```
alter table tableName modify name1 type1 first|after name2;
```

Vladimir1年前 (2017-07-17)

反馈/建议



# MySQL 索引

MySQL索引的建立对于MySQL的高效运行是很重要的，索引可以大大提高MySQL的检索速度。

打个比方，如果合理的设计且使用索引的MySQL是一辆兰博基尼的话，那么没有设计和使用索引的MySQL就是一个人力三轮车。

索引分单列索引和组合索引。单列索引，即一个索引只包含单个列，一个表可以有多个单列索引，但这不是组合索引。组合索引，即一个索引包含多个列。

创建索引时，你需要确保该索引是应用在 SQL 查询语句的条件(一般作为 WHERE 子句的条件)。

实际上，索引也是一张表，该表保存了主键与索引字段，并指向实体表的记录。

上面都在说使用索引的好处，但过多的使用索引将会造成滥用。因此索引也会有它的缺点：虽然索引大大提高了查询速度，同时却会降低更新表的速度，如对表进行INSERT、UPDATE和DELETE。因为更新表时，MySQL不仅要保存数据，还要保存一下索引文件。

建立索引会占用磁盘空间的索引文件。

## 普通索引

## 创建索引

这是最基本的索引，它没有任何限制。它有以下几种创建方式：

```
CREATE INDEX indexName ON mytable(username(length));
```

如果是CHAR, VARCHAR类型，length可以小于字段实际长度；如果是BLOB和TEXT类型，必须指定 length。

## 修改表结构(添加索引)

```
ALTER table tableName ADD INDEX indexName(columnName)
```

## 创建表的时候直接指定

```
CREATE TABLE mytable(  
  
  
ID INT NOT NULL,  
  
  
username VARCHAR(16) NOT NULL,  
  
  
INDEX [indexName] (username(length))  
  
);
```

## 删除索引的语法

```
DROP INDEX [indexName] ON mytable;
```

## 唯一索引

它与前面的普通索引类似，不同的就是：索引列的值必须唯一，但允许有空值。如果是组合索引，则列值的组合必须唯一。它有以下几种创建方式：

## 创建索引

```
CREATE UNIQUE INDEX indexName ON mytable(username(length))
```

## 修改表结构

```
ALTER table mytable ADD UNIQUE [indexName] (username(length))
```

## 创建表的时候直接指定

```
CREATE TABLE mytable(  
  
ID INT NOT NULL,  
  
username VARCHAR(16) NOT NULL,  
  
UNIQUE [indexName] (username(length))  
  
);
```

## 使用 **ALTER** 命令添加和删除索引

有四种方式来添加数据表的索引：

**ALTER TABLE tbl\_name ADD PRIMARY KEY (column\_list):** 该语句添加一个主键，这意味着索引值必须是唯一的，且不能为NULL。

**ALTER TABLE tbl\_name ADD UNIQUE index\_name (column\_list):** 这条语句创建索引的值必须是唯一的（除了NULL外，NULL可能会出现多次）。

**ALTER TABLE tbl\_name ADD INDEX index\_name (column\_list):** 添加普通索引，索引值可出现多次。

**ALTER TABLE tbl\_name ADD FULLTEXT index\_name (column\_list):**该语句指定了索引为 FULLTEXT，用于全文索引。

以下实例为在表中添加索引。

```
mysql> ALTER TABLE testalter_tbl ADD INDEX (c);
```

你还可以在 **ALTER** 命令中使用 **DROP** 子句来删除索引。尝试以下实例删除索引：

```
mysql> ALTER TABLE testalter_tbl DROP INDEX c;
```

## 使用 **ALTER** 命令添加和删除主键

主键只能作用于一个列上，添加主键索引时，你需要确保该主键默认不为空（**NOT NULL**）。实例如下：

```
mysql> ALTER TABLE testalter_tbl MODIFY i INT NOT NULL;  
  
mysql> ALTER TABLE testalter_tbl ADD PRIMARY KEY (i);
```

你也可以使用 **ALTER** 命令删除主键：

```
mysql> ALTER TABLE testalter_tbl DROP PRIMARY KEY;
```

删除主键时只需指定**PRIMARY KEY**，但在删除索引时，你必须知道索引名。



## 显示索引信息

你可以使用 **SHOW INDEX** 命令来列出表中的相关的索引信息。可以通过添加 \G 来格式化输出信息。

尝试以下实例：

```
mysql> SHOW INDEX FROM table_name; \G

.....
```

点我分享笔记

反馈/建议



## MySQL 临时表

MySQL 临时表在我们需要保存一些临时数据时是非常有用的。临时表只在当前连接可见，当关闭连接时，Mysql会自动删除表并释放所有空间。

临时表在MySQL 3.23版本中添加，如果你的MySQL版本低于 3.23版本就无法使用MySQL的临时表。不过现在一般很少有再使用这么低版本的MySQL数据库服务了。

MySQL临时表只在当前连接可见，如果你使用PHP脚本来创建MySQL临时表，那每当PHP脚本执行完成后，该临时表也会自动销毁。

如果你使用了其他MySQL客户端程序连接MySQL数据库服务器来创建临时表，那么只有在关闭客户端程序时才会销毁临时表，当然你也可以手动销毁。

### 实例

以下展示了使用MySQL 临时表的简单实例，以下的SQL代码可以适用于PHP脚本的mysql\_query()函数。

```
mysql> CREATE TEMPORARY TABLE SalesSummary (

-> product_name VARCHAR(50) NOT NULL

-> , total_sales DECIMAL(12,2) NOT NULL DEFAULT 0.00

-> , avg_unit_price DECIMAL(7,2) NOT NULL DEFAULT 0.00

-> , total_units_sold INT UNSIGNED NOT NULL DEFAULT 0

);

Query OK, 0 rows affected (0.00 sec)
```

```
mysql> INSERT INTO SalesSummary

-> (product_name, total_sales, avg_unit_price, total_units_sold)

-> VALUES

-> ('cucumber', 100.25, 90, 2);
```

```
mysql> SELECT * FROM SalesSummary;

+-----+-----+-----+-----+
| product_name | total_sales | avg_unit_price | total_units_sold |
+-----+-----+-----+-----+
| cucumber    |      100.25 |          90.00 |                2 |
+-----+-----+-----+-----+

1 row in set (0.00 sec)
```

当你使用 **SHOW TABLES** 命令显示数据表列表时，你将无法看到 **SalesSummary** 表。

如果你退出当前 **MySQL** 会话，再使用 **SELECT** 命令来读取原先创建的临时表数据，那你会发现数据库中没有该表的存在，因为在你退出时该临时表已经被销毁了。

## 删除 **MySQL** 临时表

默认情况下，当你断开与数据库的连接后，临时表就会自动被销毁。当然你也可以在当前 **MySQL** 会话使用 **DROP TABLE** 命令来手动删除临时表。

以下是手动删除临时表的实例：

```
mysql> CREATE TEMPORARY TABLE SalesSummary (

-> product_name VARCHAR(50) NOT NULL

-> , total_sales DECIMAL(12,2) NOT NULL DEFAULT 0.00

-> , avg_unit_price DECIMAL(7,2) NOT NULL DEFAULT 0.00

-> , total_units_sold INT UNSIGNED NOT NULL DEFAULT 0

);

Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO SalesSummary

-> (product_name, total_sales, avg_unit_price, total_units_sold)

-> VALUES

-> ('cucumber', 100.25, 90, 2);
```

```
mysql> SELECT * FROM SalesSummary;
```

```
+-----+-----+-----+-----+
| product_name | total_sales | avg_unit_price | total_units_sold |
+-----+-----+-----+-----+
| cucumber    |      100.25 |          90.00 |                2 |
+-----+-----+-----+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql> DROP TABLE SalesSummary;
```

```
mysql> SELECT * FROM SalesSummary;
```

```
ERROR 1146: Table 'RUN00B.SalesSummary' doesn't exist
```

[MySQL 索引](#)

[MySQL 复制表](#)

[点我分享笔记](#)

反馈/建议

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[MySQL 临时表](#)

[MySQL 元数据](#)

# MySQL 复制表

如果我们需要完全的复制MySQL的数据表，包括表的结构，索引，默认值等。如果仅仅使用**CREATE TABLE ... SELECT** 命令，是无法实现的。本章节将为大家介绍如何完整的复制MySQL数据表，步骤如下：

使用 **SHOW CREATE TABLE** 命令获取创建数据表(**CREATE TABLE**) 语句，该语句包含了原数据表的结构，索引等。

复制以下命令显示的SQL语句，修改数据表名，并执行SQL语句，通过以上命令 将完全的复制数据表结构。

如果你想复制表的内容，你就可以使用 **INSERT INTO ... SELECT** 语句来实现。

## 实例

尝试以下实例来复制表 runoob\_tbl 。

步骤一：

获取数据表的完整结构。

```
mysql> SHOW CREATE TABLE runoob_tbl \G;
```

```
***** 1. row *****
```

```
Table: runoob_tbl
```

```
Create Table: CREATE TABLE `runoob_tbl` (  
  
  `runoob_id` int(11) NOT NULL auto_increment,  
  
  `runoob_title` varchar(100) NOT NULL default '',  
  
  `runoob_author` varchar(40) NOT NULL default '',  
  
  `submission_date` date default NULL,  
  
  PRIMARY KEY (`runoob_id`),  
  
  UNIQUE KEY `AUTHOR_INDEX` (`runoob_author`)  
  
) ENGINE=InnoDB  
  
1 row in set (0.00 sec)
```

```
ERROR:
```

```
No query specified
```

## 步骤二：

修改SQL语句的数据表名，并执行SQL语句。

```
mysql> CREATE TABLE `clone_tbl` (  
  
  -> `runoob_id` int(11) NOT NULL auto_increment,  
  
  -> `runoob_title` varchar(100) NOT NULL default '',  
  
  -> `runoob_author` varchar(40) NOT NULL default '',  
  
  -> `submission_date` date default NULL,  
  
  -> PRIMARY KEY (`runoob_id`),  
  
  -> UNIQUE KEY `AUTHOR_INDEX` (`runoob_author`)  
  
-> ) ENGINE=InnoDB;  
  
Query OK, 0 rows affected (1.80 sec)
```

## 步骤三：

执行完第二步骤后，你将在数据库中创建新的克隆表 `clone_tbl`。如果你想拷贝数据表的数据你可以使用 **INSERT INTO... SELECT** 语句来实现。

```
mysql> INSERT INTO clone_tbl (runoob_id,  
  
  ->                               runoob_title,  
  
  ->                               runoob_author,  
  
  ->                               submission_date)
```

```
-> SELECT runoob_id,runoob_title,

->         runoob_author,submission_date

-> FROM runoob_tbl;
```

Query OK, 3 rows affected (0.07 sec)

Records: 3 Duplicates: 0 Warnings: 0

执行以上步骤后，你将完整的复制表，包括表结构及表数据。

[MySQL 临时表](#)

MySQL 元数据 [MySQL 元数据](#)



2 篇笔记  
#2

[写笔记](#)



另一种完整复制表的方法：

```
CREATE TABLE targetTable LIKE sourceTable;

INSERT INTO targetTable SELECT * FROM sourceTable;
```

其他：  
可以拷贝一个表中其中的一些字段：

```
CREATE TABLE newadmin AS

(

    SELECT username, password FROM admin

)
```

可以将新建的表的字段改名：

```
CREATE TABLE newadmin AS

(

    SELECT id, username AS uname, password AS pass FROM admin

)
```

可以拷贝一部分数据：

```
CREATE TABLE newadmin AS

(

    SELECT * FROM admin WHERE LEFT(username,1) = 's'

)
```

可以在创建表的同时定义表中的字段信息：

```
CREATE TABLE newadmin
```

```
(

    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY

)

AS

(

    SELECT * FROM admin

)
```

Houses Chan12个月前  
(10-06)

#1



来给大家区分下mysql复制表的两种方式。

第一、只复制表结构到新表

create table 新表 select \* from 旧表 where 1=2

或者

create table 新表 like 旧表

第二、复制表结构及数据到新表

create table新表 select \* from 旧表

刘先生12个月前 (10-07)

反馈/建议



## MySQL 元数据

你可能想知道MySQL以下三种信息：

**查询结果信息：** SELECT, UPDATE 或 DELETE语句影响的记录数。

**数据库和数据表的信息：** 包含了数据库及数据表的结构信息。

**MySQL服务器信息：** 包含了数据库服务器的当前状态，版本号等。

在MySQL的命令提示符中，我们可以很容易的获取以上服务器信息。 但如果使用Perl或PHP等脚本语言，你就需要调用特定的接口函数来获取。 接下来我们会详细介绍。

### 获取查询语句影响的记录数

#### PERL 实例

在 DBI 脚本中， 语句影响的记录数通过函数 do( ) 或 execute( )返回：

# 方法 1

```
# 使用do( ) 执行 $query

my $count = $dbh->do ($query);

# 如果发生错误会输出 0

printf "%d 条数据被影响\n", (defined ($count) ? $count : 0);


# 方法 2

# 使用prepare( ) 及 execute( ) 执行 $query

my $sth = $dbh->prepare ($query);

my $count = $sth->execute ( );

printf "%d 条数据被影响\n", (defined ($count) ? $count : 0);
```

## PHP 实例

在PHP中，你可以使用 `mysqli_affected_rows( )` 函数来获取查询语句影响的记录数。

```
$result_id = mysqli_query ($conn_id, $query);

# 如果查询失败返回

$count = ($result_id ? mysqli_affected_rows ($conn_id) : 0);

print (" $count 条数据被影响\n");
```

## 数据库和数据表列表

你可以很容易的在MySQL服务器中获取数据库和数据表列表。 如果你没有足够的权限，结果将返回 `null`。

你也可以使用 `SHOW TABLES` 或 `SHOW DATABASES` 语句来获取数据库和数据表列表。

## PERL 实例

```
# 获取当前数据库中所有可用的表。

my @tables = $dbh->tables ( );

foreach $table (@tables ){

    print "表名 $table\n";

}
```

## PHP 实例

以下实例输出 MySQL 服务器上的所有数据库：

### 查看所有数据库

```
<?php
$dbhost = 'localhost:3306'; // mysql服务器主机地址
$dbuser = 'root'; // mysql用户名
$dbpass = '123456'; // mysql用户名密码
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
```

```
if(! $conn )
{
die('连接失败: ' . mysqli_error($conn));
}
// 设置编码，防止中文乱码
$db_list = mysqli_query($conn, 'SHOW DATABASES');
while ($db = mysqli_fetch_object($db_list))
{
echo $db->Database . "<br />";
}
mysqli_close($conn);
?>
```

## 获取服务器元数据

以下命令语句可以在 MySQL 的命令提示符使用，也可以在脚本中 使用，如PHP脚本。

| 命令                 | 描述             |
|--------------------|----------------|
| SELECT VERSION( )  | 服务器版本信息        |
| SELECT DATABASE( ) | 当前数据库名 (或者返回空) |
| SELECT USER( )     | 当前用户名          |
| SHOW STATUS        | 服务器状态          |
| SHOW VARIABLES     | 服务器配置变量        |

[MySQL 复制表](#)

[MySQL 序列使用](#)

[点我分享笔记](#)

[反馈/建议](#)

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[MySQL 元数据](#)

[MySQL 处理重复数据](#)

## MySQL 序列使用

MySQL 序列是一组整数：1, 2, 3, ..., 由于一张数据表只能有一个字段自增主键， 如果你想实现其他字段也实现自动增加，就可以使用MySQL序列来实现。

本章我们将介绍如何使用MySQL的序列。

### 使用 AUTO\_INCREMENT

MySQL 中最简单使用序列的方法就是使用 MySQL AUTO\_INCREMENT 来定义列。

#### 实例

以下实例中创建了数据表 insect， insect 表中 id 无需指定值可实现自动增长。



```
mysql> CREATE TABLE insect

-> (

-> id INT UNSIGNED NOT NULL AUTO_INCREMENT,

-> PRIMARY KEY (id),

-> name VARCHAR(30) NOT NULL, # type of insect

-> date DATE NOT NULL, # date collected

-> origin VARCHAR(30) NOT NULL # where collected

);

Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO insect (id,name,date,origin) VALUES

-> (NULL,'housefly','2001-09-10','kitchen'),

-> (NULL,'millipede','2001-09-10','driveway'),

-> (NULL,'grasshopper','2001-09-10','front yard');
```

Query OK, 3 rows affected (0.02 sec)

Records: 3 Duplicates: 0 Warnings: 0

```
mysql> SELECT * FROM insect ORDER BY id;
```

```
+----+-----+-----+-----+
| id | name      | date      | origin    |
+----+-----+-----+-----+
|  1 | housefly   | 2001-09-10 | kitchen    |
|  2 | millipede  | 2001-09-10 | driveway   |
|  3 | grasshopper | 2001-09-10 | front yard |
+----+-----+-----+-----+

3 rows in set (0.00 sec)
```

## 获取AUTO\_INCREMENT值

在MySQL的客户端中你可以使用 SQL中的LAST\_INSERT\_ID() 函数来获取最后的插入表中的自增列的值。

在PHP或PERL脚本中也提供了相应的函数来获取最后的插入表中的自增列的值。

## PERL实例

使用 mysql\_insertid 属性来获取 AUTO\_INCREMENT 的值。 实例如下：

```
$dbh->do ("INSERT INTO insect (name,date,origin)

VALUES('moth','2001-09-14','windowsill')");
```

```
my $seq = $dbh->{mysql_insertid};
```

## PHP实例

PHP 通过 `mysql_insert_id()` 函数来获取执行的插入SQL语句中 `AUTO_INCREMENT` 列的值。

```
mysql_query ("INSERT INTO insect (name,date,origin)

VALUES('moth','2001-09-14','windowsill')", $conn_id);

$seq = mysql_insert_id ($conn_id);
```

## 重置序列

如果你删除了数据表中的多条记录，并希望对剩下数据的 `AUTO_INCREMENT` 列进行重新排列，那么你可以通过删除自增的列，然后重新添加来实现。不过该操作要非常小心，如果在删除的同时又有新记录添加，有可能会出现数据混乱。操作如下所示：

```
mysql> ALTER TABLE insect DROP id;

mysql> ALTER TABLE insect

-> ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT FIRST,

-> ADD PRIMARY KEY (id);
```

## 设置序列的开始值

一般情况下序列的开始值为1，但如果你需要指定一个开始值100，那我们可以通过以下语句来实现：

```
mysql> CREATE TABLE insect

-> (

-> id INT UNSIGNED NOT NULL AUTO_INCREMENT,

-> PRIMARY KEY (id),

-> name VARCHAR(30) NOT NULL,

-> date DATE NOT NULL,

-> origin VARCHAR(30) NOT NULL

)engine=innodb auto_increment=100 charset=utf8;
```

或者你也可以在表创建成功后，通过以下语句来实现：

```
mysql> ALTER TABLE t AUTO_INCREMENT = 100;
```



1 篇笔记  
#1

[写笔记](#)



使用函数创建自增序列管理表(批量使用自增表,设置初始值,自增幅度)  
第一步：创建Sequence管理表 `sequence`

```
DROP TABLE IF EXISTS sequence;

CREATE TABLE sequence (

name VARCHAR(50) NOT NULL,

current_value INT NOT NULL,

increment INT NOT NULL DEFAULT 1,

PRIMARY KEY (name)

) ENGINE=InnoDB;
```

第二步：创建取当前值的函数 `currval`

```
DROP FUNCTION IF EXISTS currval;

DELIMITER $

CREATE FUNCTION currval (seq_name VARCHAR(50))

RETURNS INTEGER

LANGUAGE SQL

DETERMINISTIC

CONTAINS SQL

SQL SECURITY DEFINER

COMMENT ''

BEGIN

DECLARE value INTEGER;

SET value = 0;

SELECT current_value INTO value

FROM sequence

WHERE name = seq_name;

RETURN value;

END

$

DELIMITER ;
```

第三步：创建取下一个值的函数 `nextval`

```
DROP FUNCTION IF EXISTS nextval;

DELIMITER $

CREATE FUNCTION nextval (seq_name VARCHAR(50))

RETURNS INTEGER

LANGUAGE SQL

DETERMINISTIC

CONTAINS SQL

SQL SECURITY DEFINER

COMMENT ''

BEGIN

UPDATE sequence

SET current_value = current_value + increment

WHERE name = seq_name;

RETURN currval(seq_name);

END

$

DELIMITER;
```

第四步：创建更新当前值的函数 **setval**

```
DROP FUNCTION IF EXISTS setval;

DELIMITER $

CREATE FUNCTION setval (seq_name VARCHAR(50), value INTEGER)

RETURNS INTEGER

LANGUAGE SQL

DETERMINISTIC

CONTAINS SQL

SQL SECURITY DEFINER

COMMENT ''

BEGIN

UPDATE sequence

SET current_value = value

WHERE name = seq_name;

RETURN currval(seq_name);

END
```

\$

DELIMITER ;

测试函数功能

当上述四步完成后，可以用以下数据设置需要创建的sequence名称以及设置初始值和获取当前值和下一个值。

```
INSERT INTO sequence VALUES ('TestSeq', 0, 1);

----添加一个sequence名称和初始值，以及自增幅度  添加一个名为TestSeq 的自增序列


SELECT SETVAL('TestSeq', 10);

---设置指定sequence的初始值      这里设置TestSeq 的初始值为10


SELECT CURRVAL('TestSeq');

--查询指定sequence的当前值      这里是获取TestSeq当前值


SELECT NEXTVAL('TestSeq');

--查询指定sequence的下一个值      这里是获取TestSeq下一个值
```

Narule3周前 (09-11)

反馈/建议



# MySQL 处理重复数据

有些 MySQL 数据表中可能存在重复的记录，有些情况我们允许重复数据的存在，但有时候我们也需要删除这些重复的数据。本章我们将为大家介绍如何防止数据表出现重复数据及如何删除数据表中的重复数据。

## 防止表中出现重复数据

你可以在MySQL数据表中设置指定的字段为 **PRIMARY KEY**（主键） 或者 **UNIQUE**（唯一） 索引来保证数据的唯一性。让我们尝试一个实例：下表中无索引及主键，所以该表允许出现多条重复记录。

```
CREATE TABLE person_tbl

(
```

```
first_name CHAR(20),

last_name CHAR(20),

sex CHAR(10)

);
```

如果你想设置表中字段`first_name`, `last_name`数据不能重复, 你可以设置双主键模式来设置数据的唯一性, 如果你设置了双主键, 那么那个键的默认值不能为`NULL`, 可设置为`NOT NULL`。如下所示:

```
CREATE TABLE person_tbl

(

first_name CHAR(20) NOT NULL,

last_name CHAR(20) NOT NULL,

sex CHAR(10),

PRIMARY KEY (last_name, first_name)

);
```

如果我们设置了唯一索引, 那么在插入重复数据时, `SQL`语句将无法执行成功,并抛出错误。

`INSERT IGNORE INTO`与`INSERT INTO`的区别就是`INSERT IGNORE`会忽略数据库中已经存在的数据, 如果数据库没有数据, 就插入新的数据, 如果有数据的话就跳过这条数据。这样就可以保留数据库中已经存在数据, 达到在间隙中插入数据的目的。

以下实例使用了`INSERT IGNORE INTO`, 执行后不会出错, 也不会向数据表中插入重复数据:

```
mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)

-> VALUES( 'Jay', 'Thomas');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT IGNORE INTO person_tbl (last_name, first_name)

-> VALUES( 'Jay', 'Thomas');

Query OK, 0 rows affected (0.00 sec)
```

`INSERT IGNORE INTO`当插入数据时, 在设置了记录的唯一性后, 如果插入重复数据, 将不返回错误, 只以警告形式返回。而`REPLACE INTO`如如果存在`primary` 或 `unique`相同的记录, 则先删除掉。再插入新记录。

另一种设置数据的唯一性方法是添加一个`UNIQUE`索引, 如下所示:

```
CREATE TABLE person_tbl

(

first_name CHAR(20) NOT NULL,

last_name CHAR(20) NOT NULL,
```

```
sex CHAR(10),

UNIQUE (last_name, first_name)

);
```

## 统计重复数据

以下我们将统计表中 `first_name` 和 `last_name` 的重复记录数：

```
mysql> SELECT COUNT(*) as repetitions, last_name, first_name

-> FROM person_tbl

-> GROUP BY last_name, first_name

-> HAVING repetitions > 1;
```

以上查询语句将返回 `person_tbl` 表中重复的记录数。 一般情况下，查询重复的值，请执行以下操作：

确定哪一列包含的值可能会重复。

在列选择列表使用 `COUNT(*)` 列出的那些列。

在 `GROUP BY` 子句中列出的列。

`HAVING` 子句设置重复数大于1。

## 过滤重复数据

如果你需要读取不重复的数据可以在 `SELECT` 语句中使用 `DISTINCT` 关键字来过滤重复数据。

```
mysql> SELECT DISTINCT last_name, first_name

-> FROM person_tbl;
```

你也可以使用 `GROUP BY` 来读取数据表中不重复的数据：

```
mysql> SELECT last_name, first_name

-> FROM person_tbl

-> GROUP BY (last_name, first_name);
```

## 删除重复数据

如果你想删除数据表中的重复数据，你可以使用以下的 `SQL` 语句：

```
mysql> CREATE TABLE tmp SELECT last_name, first_name, sex FROM person_tbl GROUP BY (last_name, first_name, sex);

mysql> DROP TABLE person_tbl;

mysql> ALTER TABLE tmp RENAME TO person_tbl;
```

当然你也可以在数据表中添加 INDEX（索引） 和 PRIMAY KEY（主键） 这种简单的方法来删除表中的重复记录。方法如下：

```
mysql> ALTER IGNORE TABLE person_tbl

-> ADD PRIMARY KEY (last_name, first_name);
```

点我分享笔记

反馈/建议



# MySQL 及 SQL 注入

如果您通过网页获取用户输入的数据并将其插入一个MySQL数据库，那么就有可能发生SQL注入安全的问题。

本章节将为大家介绍如何防止SQL注入，并通过脚本来过滤SQL中注入的字符。

所谓SQL注入，就是通过把SQL命令插入到Web表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。

我们永远不要信任用户的输入，我们必须认定用户输入的数据都是不安全的，我们都需要对用户输入的数据进行过滤处理。

以下实例中，输入的用户名必须为字母、数字及下划线的组合，且用户名长度为 8 到 20 个字符之间：

```
if (preg_match("/^\w{8,20}$/", $_GET['username'], $matches))

{

    $result = mysqli_query($conn, "SELECT * FROM users

                                WHERE username=$matches[0]");

}

else

{

    echo "username 输入异常";

}
```

让我们看下在没有过滤特殊字符时，出现的SQL情况：



```
// 设定$name 中插入了我们不需要的SQL语句

$name = "Qadir'; DELETE FROM users;";

mysqli_query($conn, "SELECT * FROM users WHERE name='{$name}'");
```

以上的注入语句中，我们没有对 `$name` 的变量进行过滤，`$name` 中插入了我们不需要的SQL语句，将删除 `users` 表中的所有数据。

在PHP中的 `mysqli_query()` 是不允许执行多个 SQL 语句的，但是在 `SQLite` 和 `PostgreSQL` 是可以同时执行多条SQL语句的，所以我们对这些用户的数据需要进行严格的验证。

防止SQL注入，我们需要注意以下几个要点：

- 1.永远不要信任用户的输入。对用户的输入进行校验，可以通过正则表达式，或限制长度；对单引号和 双 "-" 进行转换等。
- 2.永远不要使用动态拼装sql，可以使用参数化的sql或者直接使用存储过程进行数据查询存取。
- 3.永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。
- 4.不要把机密信息直接存放，加密或者hash掉密码和敏感的信息。
- 5.应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息对原始错误信息进行包装
- 6.sql注入的检测方法一般采取辅助软件或网站平台来检测，软件一般采用sql注入检测工具jsky，网站平台就有亿思网站安全平台检测工具。MDCSOFT SCAN等。采用MDCSOFT-IPS可以有有效的防御SQL注入，XSS攻击等。

## 防止SQL注入

在脚本语言，如Perl和PHP你可以对用户输入的数据进行转义从而防止SQL注入。

PHP的MySQL扩展提供了`mysqli_real_escape_string()`函数来转义特殊的输入字符。

```
if (get_magic_quotes_gpc())

{

    $name = stripslashes($name);

}

$name = mysqli_real_escape_string($conn, $name);

mysqli_query($conn, "SELECT * FROM users WHERE name='{$name}'");
```

## Like语句中的注入

like查询时，如果用户输入的值有 "\_" 和 "%", 则会出现这种情况：用户本来只是想查询"abcd\_"，查询结果中却有"abcd\_"、"abcde"、"abcdf"等等；用户要查询"30%"（注：百分之三十）时也会出现问题。

在PHP脚本中我们可以使用`addslashes()`函数来处理以上情况，如下实例：

```
$sub = addslashes(mysqli_real_escape_string($conn, "%something_"), "%_");

// $sub == \%something\_

mysqli_query($conn, "SELECT * FROM messages WHERE subject LIKE '{$sub}%')");
```

`addslashes()` 函数在指定的字符前添加反斜杠。

语法格式：

addslashes(string, characters)

| 参数         | 描述                              |
|------------|---------------------------------|
| string     | 必需。规定要检查的字符串。                   |
| characters | 可选。规定受 addslashes() 影响的字符或字符范围。 |

具体应用可以查看：[PHP addslashes\(\) 函数](#)

[MySQL 处理重复数据](#)

[MySQL 导出数据](#)

[点我分享笔记](#)

反馈/建议

Copyright © 2013-2018 菜鸟教程 [runoob.com](#) All Rights Reserved. 备案号：闽ICP备15012807号-1



[首页](#) [HTML](#) [CSS](#) [JS](#) [本地书签](#)

[MySQL 及 SQL 注入](#)

[MySQL 导入数据](#)

## MySQL 导出数据

MySQL中你可以使用**SELECT...INTO OUTFILE**语句来简单的导出数据到文本文件上。

### 使用 SELECT ... INTO OUTFILE 语句导出数据

以下实例中我们将数据表 runoob\_tbl 数据导出到 /tmp/runoob.txt 文件中：

```
mysql> SELECT * FROM runoob_tbl
      -> INTO OUTFILE '/tmp/runoob.txt';
```

你可以通过命令选项来设置数据输出的指定格式，以下实例为导出 CSV 格式：

```
mysql> SELECT * FROM passwd INTO OUTFILE '/tmp/runoob.txt'
      -> FIELDS TERMINATED BY ',' ENCLOSED BY '"'
      -> LINES TERMINATED BY '\r\n';
```

在下面的例子中，生成一个文件，各值用逗号隔开。这种格式可以被许多程序使用。

```
SELECT a,b,a+b INTO OUTFILE '/tmp/result.txt'
```

```
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ''

LINES TERMINATED BY '\n'

FROM test_table;
```

## SELECT ... INTO OUTFILE 语句有以下属性：

LOAD DATA INFILE是SELECT ... INTO OUTFILE的逆操作，SELECT句法。为了将一个数据库的数据写入一个文件，使用SELECT ... INTO OUTFILE，为了将文件读回数据库，使用LOAD DATA INFILE。

SELECT...INTO OUTFILE 'file\_name'形式的SELECT可以把被选择的行写入一个文件中。该文件被创建到服务器主机上，因此您必须拥有FILE权限，才能使用此语法。

输出不能是一个已存在的文件。防止文件数据被篡改。

你需要有一个登陆服务器的账号来检索文件。否则 SELECT ... INTO OUTFILE 不会起任何作用。

在UNIX中，该文件被创建后是可读的，权限由MySQL服务器所拥有。这意味着，虽然你就可以读取该文件，但可能无法将其删除。

## 导出表作为原始数据

**mysqldump** 是 mysql 用于转存储数据库的实用程序。它主要产生一个 SQL 脚本，其中包含从头重新创建数据库所必需的命令 CREATE TABLE INSERT 等。

使用 **mysqldump** 导出数据需要使用 **--tab** 选项来指定导出文件指定的目录，该目标必须是可写的。

以下实例将数据表 runoob\_tbl 导出到 /tmp 目录中：

```
$ mysqldump -u root -p --no-create-info \

--tab=/tmp RUNOOB runoob_tbl

password *****
```

## 导出 SQL 格式的数据

导出 SQL 格式的数据到指定文件，如下所示：

```
$ mysqldump -u root -p RUNOOB runoob_tbl > dump.txt

password *****
```

以上命令创建的文件内容如下：

```
-- MySQL dump 8.23

--

-- Host: localhost    Database: RUNOOB

-----

-- Server version      3.23.58

--
```

```
-- Table structure for table `runoob_tbl`

--

CREATE TABLE runoob_tbl (

  runoob_id int(11) NOT NULL auto_increment,

  runoob_title varchar(100) NOT NULL default '',

  runoob_author varchar(40) NOT NULL default '',

  submission_date date default NULL,

  PRIMARY KEY (runoob_id),

  UNIQUE KEY AUTHOR_INDEX (runoob_author)

) TYPE=MyISAM;

--

-- Dumping data for table `runoob_tbl`

--

INSERT INTO runoob_tbl

VALUES (1,'Learn PHP','John Poul','2007-05-24');

INSERT INTO runoob_tbl

VALUES (2,'Learn MySQL','Abdul S','2007-05-24');

INSERT INTO runoob_tbl

VALUES (3,'JAVA Tutorial','Sanjay','2007-05-06');
```

如果你需要导出整个数据库的数据，可以使用以下命令：

```
$ mysqldump -u root -p RUNOOB > database_dump.txt

password *****
```

如果需要备份所有数据库，可以使用以下命令：

```
$ mysqldump -u root -p --all-databases > database_dump.txt

password *****
```

`--all-databases` 选项在 MySQL 3.23.12 及以后版本加入。

该方法可用于实现数据库的备份策略。

## 将数据表及数据库拷贝至其他主机

如果你需要将数据拷贝至其他的 MySQL 服务器上, 你可以在 `mysqldump` 命令中指定数据库名及数据表。

在源主机上执行以下命令, 将数据备份至 `dump.txt` 文件中:

```
$ mysqldump -u root -p database_name table_name > dump.txt

password *****
```

如果完整备份数据库, 则无需使用特定的表名称。

如果你需要将备份的数据库导入到MySQL服务器中, 可以使用以下命令, 使用以下命令你需要确认数据库已经创建:

```
$ mysql -u root -p database_name < dump.txt

password *****
```

你也可以使用以下命令将导出的数据直接导入到远程的服务器上, 但请确保两台服务器是相通的, 是可以相互访问的: </p>

```
$ mysqldump -u root -p database_name \

| mysql -h other-host.com database_name
```

以上命令中使用了管道来将导出的数据导入到指定的远程主机上。

1 篇笔记

#1

写笔记

### 将指定主机的数据库拷贝到本地

如果你需要将远程服务器的数据拷贝到本地, 你也可以在 `mysqldump` 命令中指定远程服务器的IP、端口及数据库名。

在源主机上执行以下命令, 将数据备份到 `dump.txt` 文件中:

请确保两台服务器是相通的:

```
mysqldump -h other-host.com -P port -u root -p database_name > dump.txt

password ****
```

CarolLi1年前 (2017-08-21)

反馈/建议



# MySQL 导入数据

本章节我们为大家介绍几种简单的 MySQL 导出的数据的命令。

## 1、mysql 命令导入

使用 `mysql` 命令导入语法格式为：

```
mysql -u用户名 -p密码 < 要导入的数据库数据(runoob.sql)
```

实例：

```
# mysql -uroot -p123456 < runoob.sql
```

以上命令将将备份的整个数据库 `runoob.sql` 导入。

## 2、source 命令导入

`source` 命令导入数据库需要先登录到数据库终端：

```
mysql> create database abc;      # 创建数据库

mysql> use abc;                  # 使用已创建的数据库

mysql> set names utf8;           # 设置编码

mysql> source /home/abc/abc.sql  # 导入备份数据库
```

## 3、使用 LOAD DATA 导入数据

MySQL 中提供了 `LOAD DATA INFILE` 语句来插入数据。以下实例中将从当前目录中读取文件 `dump.txt`，将该文件中的数据插入到当前数据库的 `mytbl` 表中。

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl;
```

如果指定 `LOCAL` 关键词，则表明从客户主机上按路径读取文件。如果没有指定，则文件在服务器上按路径读取文件。

你能明确地在 `LOAD DATA` 语句中指出列值的分隔符和行尾标记，但是默认标记是定位符和换行符。

两个命令的 `FIELDS` 和 `LINES` 子句的语法是一样的。两个子句都是可选的，但是如果两个同时被指定，`FIELDS` 子句必须出现在 `LINES` 子句之前。

如果用户指定一个 `FIELDS` 子句，它的子句（`TERMINATED BY`、`[OPTIONALLY] ENCLOSED BY` 和 `ESCAPED BY`）也是可选的，不过，用户必须至少指定它们中的一个。

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl

-> FIELDS TERMINATED BY ':'

-> LINES TERMINATED BY '\r\n';
```

LOAD DATA 默认情况下是按照数据文件中列的顺序插入数据的，如果数据文件中的列与插入表中的列不一致，则需要指定列的顺序。如，在数据文件中的列顺序是 **a,b,c**，但在插入表的列顺序为**b,c,a**，则数据导入语法如下：

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt'

-> INTO TABLE mytbl (b, c, a);
```

## 4、使用 **mysqlimport** 导入数据

mysqlimport客户端提供了LOAD DATA INFILEQL语句的一个命令行接口。**mysqlimport**的大多数选项直接对应LOAD DATA INFILE子句。从文件 **dump.txt** 中将数据导入到 **mytbl** 数据表中, 可以使用以下命令：

```
$ mysqlimport -u root -p --local database_name dump.txt

password *****
```

**mysqlimport**命令可以指定选项来设置指定格式,命令语句格式如下：

```
$ mysqlimport -u root -p --local --fields-terminated-by=":" \

--lines-terminated-by="\r\n" database_name dump.txt

password *****
```

**mysqlimport** 语句中使用 **--columns** 选项来设置列的顺序：

```
$ mysqlimport -u root -p --local --columns=b,c,a \

database_name dump.txt

password *****
```

## mysqlimport的常用选项介绍

| 选项                         | 功能  |
|----------------------------|---|
| -d or --delete             | 新数据导入数据表中之前删除数据数据表中的所有信息                                |
| -f or --force              | 不管是否遇到错误， <b>mysqlimport</b> 将强制继续插入数据                  |
| -i or --ignore             | <b>mysqlimport</b> 跳过或者忽略那些有相同唯一 关键字的行， 导入文件中的数据将被忽略。   |
| -l or --lock-tables        | 数据被插入之前锁住表，这样就防止了， 你在更新数据库时，用户的查询和更新受到影响。               |
| -r or --replace            | 这个选项与-i选项的作用相反；此选项将替代 表中有相同唯一关键字的记录。                    |
| --fields-enclosed-by= char | 指定文本文件中数据的记录时以什么括起的， 很多情况下 数据以双引号括起。 默认的情况下数据是没有被字符括起的。 |

|   |   |
|---|---|
| <code>-fields-terminated-by=char</code> | 指定各个数据的值之间的分隔符，在句号分隔的文件中， 分隔符是句号。您可以用此选项指定数据之间的分隔符。 默认的分隔符是跳格符（Tab）                             |
| <code>-lines-terminated-by=str</code>   | 此选项指定文本文件中行与行之间数据的分隔字符串 或者字符。 默认的情况下mysqlimport以newline为行分隔符。 您可以选择一个字符串来替代一个单个的字符： 一个新行或者一个回车。 |

mysqlimport命令常用的选项还有-v 显示版本（version）， -p 提示输入密码（password）等。

[点我分享笔记](#)

反馈/建议



## MySQL 函数

MySQL 有很多内置的函数，以下列出了这些函数的说明。

### MySQL 字符串函数

| 函数             | 描述                       | 实例   |
|----------------|--------------------------|--|
| ASCII(s)       | 返回字符串 s 的第一个字符的 ASCII 码。 | 返回 CustomerName 字段第一个字母的 ASCII 码： <div><pre>SELECT ASCII(CustomerName) AS NumCodeOfFirstChar FROM Customers;</pre></div> |
| CHAR_LENGTH(s) | 返回字符串 s 的字符数             | 返回字符串 RUNOOB 的字符数 <div><pre>SELECT CHAR_LENGTH("RUNOOB") AS LengthOfString;</pre></div>                                  |



|                          |   |   |
|--------------------------|---|---|
| CHARACTER_LENGTH(s)      | 返回字符串 <b>s</b> 的字符数   | 返回字符串 <b>RUNOOB</b> 的字符数 <pre>SELECT CHARACTER_LENGTH("RUNOOB") AS LengthOfString;</pre>                        |
| CONCAT(s1,s2...sn)       | 字符串 <b>s1,s2</b> 等多个字符串合并为一个字符串   | 合并多个字符串 <pre>SELECT CONCAT("SQL ", "Runoob ", "Google ", "Facebook") AS ConcatenatedString;</pre>               |
| CONCAT_WS(x, s1,s2...sn) | 同 <b>CONCAT(s1,s2,...)</b> 函数，但是每个字符串直接要加上 <b>x</b> ， <b>x</b> 可以是分隔符               | 合并多个字符串，并添加分隔符： <pre>SELECT CONCAT_WS("-", "SQL", "Tutorial", "is", "fun!")AS ConcatenatedString;</pre>         |
| FIELD(s,s1,s2...)        | 返回第一个字符串 <b>s</b> 在字符串列表( <b>s1,s2...</b> )中的位置                                     | 返回字符串 <b>c</b> 在列表值中的位置： <pre>SELECT FIELD("c", "a", "b", "c", "d", "e");</pre>                                 |
| FIND_IN_SETT(s1,s2)      | 返回在字符串 <b>s2</b> 中与 <b>s1</b> 匹配的字符串的位置   | 返回字符串 <b>c</b> 在指定字符串中的位置： <pre>SELECT FIND_IN_SET("c", "a,b,c,d,e");</pre>                                     |
| FORMAT(x,n)              | 函数可以将数字 <b>x</b> 进行格式化 " <b>#,###.##</b> ", 将 <b>x</b> 保留到小数点后 <b>n</b> 位，最后一位四舍五入。 | 格式化数字 " <b>#,###.##</b> " 形式： <pre>SELECT FORMAT(250500.5634, 2);      -- 输出 250,500.56</pre>                   |
| INSERT(s1,x,len,s2)      | 字符串 <b>s2</b> 替换 <b>s1</b> 的 <b>x</b> 位置开始长度为 <b>len</b> 的字符串                       | 从字符串第一个位置开始的 6 个字符替换为 <b>runoob</b> : <pre>SELECT INSERT("google.com", 1, 6, "runnob"); -- 输出: runoob.com</pre> |

|                 |   |   |
|-----------------|---|---|
| LOCATE(s1,s)    | 从字符串 <b>s</b> 中获取 <b>s1</b> 的开始位置                       | 获取 <b>b</b> 在字符串 <b>abc</b> 中的位置：<br><pre>SELECT INSTR('abc','b') -- 2</pre>                        |
| LCASE(s)        | 将字符串 <b>s</b> 的所有字母变成小写字母                               | 字符串 <b>RUNOOB</b> 转换为小写：<br><pre>SELECT LOWER('RUNOOB') -- runoob</pre>                             |
| LEFT(s,n)       | 返回字符串 <b>s</b> 的前 <b>n</b> 个字符                          | 返回字符串 <b>runoob</b> 中的前两个字符：<br><pre>SELECT LEFT('runoob',2) -- ru</pre>                            |
| LEFT(s,n)       | 返回字符串 <b>s</b> 的前 <b>n</b> 个字符                          | 返回字符串 <b>abcde</b> 的前两个字符：<br><pre>SELECT LEFT('abcde',2) -- ab</pre>                               |
| LOCATE(s1,s)    | 从字符串 <b>s</b> 中获取 <b>s1</b> 的开始位置                       | 返回字符串 <b>abc</b> 中 <b>b</b> 的位置：<br><pre>SELECT LOCATE('b', 'abc') -- 2</pre>                       |
| LOWER(s)        | 将字符串 <b>s</b> 的所有字母变成小写字母                               | 字符串 <b>RUNOOB</b> 转换为小写：<br><pre>SELECT LOWER('RUNOOB') -- runoob</pre>                             |
| LPAD(s1,len,s2) | 在字符串 <b>s1</b> 的开始处填充字符串 <b>s2</b> ，使字符串长度达到 <b>len</b> | 将字符串 <b>xx</b> 填充到 <b>abc</b> 字符串的开始处：<br><pre>SELECT LPAD('abc',5,'xx') -- xxabc</pre>             |
| LTRIM(s)        | 去掉字符串 <b>s</b> 开始处的空格                                   | 去掉字符串 <b>RUNOOB</b> 开始处的空格：<br><pre>SELECT LTRIM("    RUNOOB") AS LeftTrimmedString;-- RUNOOB</pre> |

|                   |   |   |
|-------------------|---|---|
| MID(s,n,len)      | 从字符串 <b>s</b> 的 <b>start</b> 位置截取长度为 <b>length</b> 的子字符串，同 SUBSTRING(s,n,len) | 从字符串 <b>RUNOOB</b> 中的第 <b>2</b> 个位置截取 <b>3</b> 个字符：<br><br><pre>SELECT MID("RUNOOB", 2, 3) AS ExtractString; -- UNO</pre> |
| POSITION(s1 IN s) | 从字符串 <b>s</b> 中获取 <b>s1</b> 的开始位置   | 返回字符串 <b>abc</b> 中 <b>b</b> 的位置：<br><br><pre>SELECT POSITION('b' in 'abc') -- 2</pre>                                     |
| REPEAT(s,n)       | 将字符串 <b>s</b> 重复 <b>n</b> 次   | 将字符串 <b>runoob</b> 重复三次：<br><br><pre>SELECT REPEAT('runoob',3) -- runoobrunoobrunoob</pre>                                |
| REPLACE(s,s1,s2)  | 将字符串 <b>s2</b> 替代字符串 <b>s</b> 中的字符串 <b>s1</b>                                 | 将字符串 <b>abc</b> 中的字符 <b>a</b> 替换为字符 <b>x</b> ：<br><br><pre>SELECT REPLACE('abc','a','x') --xbc</pre>                      |
| REVERSE(s)        | 将字符串 <b>s</b> 的顺序反过来  | 将字符串 <b>abc</b> 的顺序反过来：<br><br><pre>SELECT REVERSE('abc') -- cba</pre>  |
| RIGHT(s,n)        | 返回字符串 <b>s</b> 的后 <b>n</b> 个字符  | 返回字符串 <b>runoob</b> 的后两个字符：<br><br><pre>SELECT RIGHT('runoob',2) -- ob</pre>  |
| RPAD(s1,len,s2)   | 在字符串 <b>s1</b> 的结尾处添加字符串 <b>s1</b> ，使字符串的长度达到 <b>len</b>                      | 将字符串 <b>xx</b> 填充到 <b>abc</b> 字符串的结尾处：<br><br><pre>SELECT RPAD('abc',5,'xx') -- abcxx</pre>                               |
| RTRIM(s)          | 去掉字符串 <b>s</b> 结尾处的空格   | 去掉字符串 <b>RUNOOB</b> 的末尾空格：<br><br><pre>SELECT RTRIM("RUNOOB   ") AS RightTrimmedString; -- RUNOOB</pre>                   |

|                                       |  |  |
|---------------------------------------|--|--|
| SPACE(n)                              | 返回 n 个空格   | 返回 10 个空格： <pre>SELECT SPACE(10);</pre>  |
| STRCMP(s1,s2)                         | 比较字符串 s1 和 s2，如果 s1 与 s2 相等返回 0，如果 s1>s2 返回 1，如果 s1<s2 返回 -1   | 比较字符串： <pre>SELECT STRCMP("runoob", "runoob"); -- 0</pre>  |
| SUBSTR(s, start, length)              | 从字符串 s 的 start 位置截取长度为 length 的子字符串  | 从字符串 RUNOOB 中的第 2 个位置截取 3 个字符： <pre>SELECT SUBSTR("RUNOOB", 2, 3) AS ExtractString; -- UNO</pre>   |
| SUBSTRING(s, start, length)           | 从字符串 s 的 start 位置截取长度为 length 的子字符串  | 从字符串 RUNOOB 中的第 2 个位置截取 3 个字符： <pre>SELECT SUBSTRING("RUNOOB", 2, 3) AS ExtractString; -- UNO</pre>  |
| SUBSTRING_INDEX(s, delimiter, number) | 返回从字符串 s 的第 number 个出现的分隔符 delimiter 之后的子串。<br>如果 number 是正数，返回第 number 个字符左边的字符串。<br>如果 number 是负数，返回第(number 的绝对值(从右边数))个字符右边的字符串。 | <pre>SELECT SUBSTRING_INDEX('a*b','*',1) -- a  SELECT SUBSTRING_INDEX('a*b','*',-1) -- b  SELECT SUBSTRING_INDEX(SUBSTRING_INDEX('a*b*c*d*e','*',3),'*',-1) -- c</pre> |
| TRIM(s)                               | 去掉字符串 s 开始和结尾处的空格  | 去掉字符串 RUNOOB 的首尾空格： <pre>SELECT TRIM('  RUNOOB  ') AS TrimmedString;</pre>   |
| UCASE(s)                              | 将字符串转换为大写  | 将字符串 runoob 转换为大写： <pre>SELECT UCASE("runoob"); -- RUNOOB</pre>  |

|          |           |  |
|----------|-----------|--|
| UPPER(s) | 将字符串转换为大写 | 将字符串 <code>runoob</code> 转换为大写： <div>SELECT UPPER("runoob"); -- RUNOOB</div> |
|----------|-----------|--|

## MySQL 数字函数

| 函数名             | 描述                                   | 实例   |
|-----------------|--------------------------------------|--|
| ABS(x)          | 返回 <b>x</b> 的绝对值                     | 返回 -1 的绝对值： <div>SELECT ABS(-1) -- 返回1</div>   |
| ACOS(x)         | 求 <b>x</b> 的反余弦值(参数是弧度)              | <div>SELECT ACOS(0.25);</div>  |
| ASIN(x)         | 求反正弦值(参数是弧度)                         | <div>SELECT ASIN(0.25);</div>  |
| ATAN(x)         | 求反正切值(参数是弧度)                         | <div>SELECT ATAN(2.5);</div>   |
| ATAN2(n, m)     | 求反正切值(参数是弧度)                         | <div>SELECT ATAN2(-0.8, 2);</div>  |
| AVG(expression) | 返回一个表达式的平均值， <b>expression</b> 是一个字段 | 返回 <b>Products</b> 表中 <b>Price</b> 字段的平均值： <div>SELECT AVG(Price) AS AveragePrice FROM Products;</div> |
| CEIL(x)         | 返回大于或等于 <b>x</b> 的最小整数               | <div>SELECT CEIL(1.5) -- 返回2</div>   |

|                   |  |  |
|-------------------|--|--|
| CEILING(x)        | 返回大于或等于 <b>x</b> 的最小整数                     | <pre>SELECT CEIL(1.5) -- 返回2</pre>   |
| COS(x)            | 求余弦值(参数是弧度)                                | <pre>SELECT COS(2);</pre>  |
| COT(x)            | 求余切值(参数是弧度)                                | <pre>SELECT COT(6);</pre>  |
| COUNT(expression) | 返回查询的记录总数， <b>expression</b> 参数是一个字段或者 * 号 | 返回 <b>Products</b> 表中 <b>products</b> 字段总共有多少条记录：<br><pre>SELECT COUNT(ProductID) AS NumberOfProducts FROM Products;</pre> |
| DEGREES(x)        | 将弧度转换为角度                                   | <pre>SELECT DEGREES(3.1415926535898) -- 180</pre>  |
| n DIV m           | 整除， <b>n</b> 为被除数， <b>m</b> 为除数            | 计算 10 除以 5：<br><pre>SELECT 10 DIV 5; -- 2</pre>  |
| EXP(x)            | 返回 <b>e</b> 的 <b>x</b> 次方                  | 计算 <b>e</b> 的三次方：<br><pre>SELECT EXP(3) -- 20.085536923188</pre>   |
| FLOOR(x)          | 返回小于或等于 <b>x</b> 的最大整数                     | 小于或等于 1.5 的整数：<br><pre>SELECT FLOOR(1.5) -- 返回1</pre>  |

|   |                          |   |
|---|--------------------------|---|
| <code>GREATEST(expr1, expr2, expr3, ...)</code> | 返回列表中的最大值                | <div>返回以下数字列表中的最大值：</div> <div><pre>SELECT GREATEST(3, 12, 34, 8, 25); -- 34</pre></div> <div>返回以下字符串列表中的最大值：</div> <div><pre>SELECT GREATEST("Google", "Runoob", "Apple"); -- Runoob</pre></div> |
| <code>LEAST(expr1, expr2, expr3, ...)</code>    | 返回列表中的最小值                | <div>返回以下数字列表中的最小值：</div> <div><pre>SELECT LEAST(3, 12, 34, 8, 25); -- 3</pre></div> <div>返回以下字符串列表中的最小值：</div> <div><pre>SELECT LEAST("Google", "Runoob", "Apple"); -- Apple</pre></div>         |
| <code><a href="#">LN</a></code>                 | 返回数字的自然对数                | <div>返回 2 的自然对数：</div> <div><pre>SELECT LN(2); -- 0.6931471805599453</pre></div>  |
| <code>LOG(x)</code>                             | 返回自然对数(以 <b>e</b> 为底的对数) | <div><pre>SELECT LOG(20.085536923188) -- 3</pre></div>  |
| <code>LOG10(x)</code>                           | 返回以 10 为底的对数             | <div><pre>SELECT LOG10(100) -- 2</pre></div>  |
| <code>LOG2(x)</code>                            | 返回以 2 为底的对数              | <div>返回以 2 为底 6 的对数：</div> <div><pre>SELECT LOG2(6); -- 2.584962500721156</pre></div>   |

|                 |                               |  |
|-----------------|-------------------------------|--|
| MAX(expression) | 返回字段 <b>expression</b> 中的最大值  | 返回数据表 <b>Products</b> 中字段 <b>Price</b> 的最大值： <div>SELECT MAX(Price) AS LargestPrice FROM Products;</div> |
| MIN(expression) | 返回字段 <b>expression</b> 中的最小值  | 返回数据表 <b>Products</b> 中字段 <b>Price</b> 的最小值： <div>SELECT MIN(Price) AS LargestPrice FROM Products;</div> |
| MOD(x,y)        | 返回 <b>x</b> 除以 <b>y</b> 以后的余数 | 5 除以 2 的余数： <div>SELECT MOD(5,2) -- 1</div>  |
| PI()            | 返回圆周率(3.141593)               | <div>SELECT PI() --3.141593</div>  |
| POW(x,y)        | 返回 <b>x</b> 的 <b>y</b> 次方     | 2 的 3 次方： <div>SELECT POW(2,3) -- 8</div>  |
| POWER(x,y)      | 返回 <b>x</b> 的 <b>y</b> 次方     | 2 的 3 次方： <div>SELECT POWER(2,3) -- 8</div>  |
| RADIANS(x)      | 将角度转换为弧度                      | 180 度转换为弧度： <div>SELECT RADIANS(180) -- 3.1415926535898</div>  |
| RAND()          | 返回 0 到 1 的随机数                 | <div>SELECT RAND() --0.93099315644334</div>  |



|                 |  |   |
|-----------------|--|---|
| ROUND(x)        | 返回离 x 最近的整数                                  | <pre>SELECT ROUND(1.23456) --1</pre>  |
| SIGN(x)         | 返回 x 的符号，x 是负数、0、正数分别返回 -1、0 和 1             | <pre>SELECT SIGN(-10) -- (-1)</pre>   |
| SIN(x)          | 求正弦值(参数是弧度)                                  | <pre>SELECT SIN(RADIANS(30)) -- 0.5</pre>   |
| SQRT(x)         | 返回x的平方根                                      | 25 的平方根：<br><pre>SELECT SQRT(25) -- 5</pre>   |
| SUM(expression) | 返回指定字段的总和                                    | 计算 OrderDetails 表中字段 Quantity 的总和：<br><pre>SELECT SUM(Quantity) AS TotalItemsOrdered FROM OrderDetails;</pre> |
| TAN(x)          | 求正切值(参数是弧度)                                  | <pre>SELECT TAN(1.75); -- -5.52037992250933</pre>   |
| TRUNCATE(x,y)   | 返回数值 x 保留到小数点后 y 位的值（与 ROUND 最大的区别是不会进行四舍五入） | <pre>SELECT TRUNCATE(1.23456,3) -- 1.234</pre>  |

## MySQL 日期函数

| 函数名          | 描述                 | 实例  |
|--------------|--------------------|---|
| ADDDATE(d,n) | 计算其实日期 d 加上 n 天的日期 | <pre>SELECT ADDDATE("2017-06-15", INTERVAL 10 DAY);</pre><br><pre>-&gt;2017-06-25</pre> |

|                     |                   |   |
|---------------------|-------------------|---|
| ADDTIME(t,n)        | 时间 t 加上 n 秒的时间    | <pre>SELECT ADDTIME('2011-11-11 11:11:11', 5)  -&gt;2011-11-11 11:11:16 (秒)</pre> |
| CURDATE()           | 返回当前日期            | <pre>SELECT CURDATE();  -&gt; 2018-09-19</pre>                                    |
| CURRENT_DATE()      | 返回当前日期            | <pre>SELECT CURRENT_DATE();  -&gt; 2018-09-19</pre>                               |
| CURRENT_TIME        | 返回当前时间            | <pre>SELECT CURRENT_TIME();  -&gt; 19:59:02</pre>                                 |
| CURRENT_TIMESTAMP() | 返回当前日期和时间         | <pre>SELECT CURRENT_TIMESTAMP()  -&gt; 2018-09-19 20:57:43</pre>                  |
| CURTIME()           | 返回当前时间            | <pre>SELECT CURTIME();  -&gt; 19:59:02</pre>                                      |
| DATE()              | 从日期或日期时间表达式中提取日期值 | <pre>SELECT DATE("2017-06-15");  -&gt; 2017-06-15</pre>                           |

|                                   |                               |  |
|-----------------------------------|-------------------------------|--|
| DATEDIFF(d1,d2)                   | 计算日期 d1->d2 之间相隔的天数           | <pre>SELECT DATEDIFF('2001-01-01','2001-02-02')</pre> <p>-&gt; -32</p>   |
| DATE_ADD(d, INTERVAL expr type)   | 计算起始日期 d 加上一个时间段后的日期          | <pre>SELECT ADDDATE('2011-11-11 11:11:11',1)</pre> <p>-&gt; 2011-11-12 11:11:11 (默认是天)</p> <pre>SELECT ADDDATE('2011-11-11 11:11:11', INTERVAL 5 MINUTE)</pre> <p>-&gt; 2011-11-11 11:16:11 (TYPE的取值与上面那个列出来的函数类似)</p> |
| DATE_FORMAT(d,f)                  | 按表达式 f的要求显示日期 d               | <pre>SELECT DATE_FORMAT('2011-11-11 11:11:11','%Y-%m-%d %r')</pre> <p>-&gt; 2011-11-11 11:11:11 AM</p>   |
| DATE_SUB(date,INTERVAL expr type) | 函数从日期减去指定的时间间隔。               | <p>Orders 表中 OrderDate 字段减去 2 天:</p> <pre>SELECT OrderId,DATE_SUB(OrderDate,INTERVAL 2 DAY) AS OrderPayDate</pre> <p>FROM Orders</p>   |
| DAY(d)                            | 返回日期值 d 的日期部分                 | <pre>SELECT DAY("2017-06-15");</pre> <p>-&gt; 15</p>   |
| DAYNAME(d)                        | 返回日期 d 是星期几, 如 Monday,Tuesday | <pre>SELECT DAYNAME('2011-11-11 11:11:11')</pre> <p>-&gt;Friday</p>  |

|                      |   |  |
|----------------------|---|--|
| DAYOFMONTH(d)        | 计算日期 d 是本月的第几天  | <pre>SELECT DAYOFMONTH('2011-11-11 11:11:11')  -&gt;11</pre>           |
| DAYOFWEEK(d)         | 日期 d 今天是星期几，1 星期日，2 星期一，以此类推  | <pre>SELECT DAYOFWEEK('2011-11-11 11:11:11')  -&gt;6</pre>             |
| DAYOFYEAR(d)         | 计算日期 d 是本年的第几天  | <pre>SELECT DAYOFYEAR('2011-11-11 11:11:11')  -&gt;315</pre>           |
| EXTRACT(type FROM d) | <p>从日期 d 中获取指定的值，type 指定返回的值。<br/>type可取值为：</p> <p>MICROSECOND</p> <p>SECOND</p> <p>MINUTE</p> <p>HOUR</p> <p>DAY</p> <p>WEEK</p> <p>MONTH</p> <p>QUARTER</p> <p>YEAR</p> <p>SECOND_MICROSECOND</p> <p>MINUTE_MICROSECOND</p> <p>MINUTE_SECOND</p> <p>HOUR_MICROSECOND</p> <p>HOUR_SECOND</p> <p>HOUR_MINUTE</p> <p>DAY_MICROSECOND</p> <p>DAY_SECOND</p> <p>DAY_MINUTE</p> <p>DAY_HOUR</p> <p>YEAR_MONTH</p> | <pre>SELECT EXTRACT(MINUTE FROM '2011-11-11 11:11:11')  -&gt; 11</pre> |

|                                |   |   |
|--------------------------------|---|---|
| FROM_DAYS(n)                   | 计算从 0000 年 1 月 1 日开始 n 天后的日期                | <pre>SELECT FROM_DAYS(1111)  -&gt; 0003-01-16</pre>                     |
| HOUR(t)                        | 返回 t 中的小时值                                  | <pre>SELECT HOUR('1:2:3')</pre> <pre>-&gt; 1</pre>                      |
| LAST_DAY(d)                    | 返回给定日期的那一月份的最后一天                            | <pre>SELECT LAST_DAY("2017-06-20");</pre> <pre>-&gt; 2017-06-30</pre>   |
| LOCALTIME()                    | 返回当前日期和时间                                   | <pre>SELECT LOCALTIME()</pre> <pre>-&gt; 2018-09-19 20:57:43</pre>      |
| LOCALTIMESTAMP()               | 返回当前日期和时间                                   | <pre>SELECT LOCALTIMESTAMP()</pre> <pre>-&gt; 2018-09-19 20:57:43</pre> |
| MAKEDATE(year, day-of-year)    | 基于给定参数年份 year 和所在年中的天数序号 day-of-year 返回一个日期 | <pre>SELECT MAKEDATE(2017, 3);</pre> <pre>-&gt; 2017-01-03</pre>        |
| MAKETIME(hour, minute, second) | 组合时间，参数分别为小时、分钟、秒                           | <pre>SELECT MAKETIME(11, 35, 4);</pre> <pre>-&gt; 11:35:04</pre>        |

|                               |                       |  |
|-------------------------------|-----------------------|--|
| MICROSECOND(date)             | 返回日期参数所对应的毫秒数         | <pre>SELECT MICROSECOND("2017-06-20 09:34:00.000023");</pre> <p>-&gt; 23</p> |
| MINUTE(t)                     | 返回 t 中的分钟值            | <pre>SELECT MINUTE('1:2:3')</pre> <p>-&gt; 2</p>                             |
| MONTHNAME(d)                  | 返回日期当中的月份名称，如 Janyary | <pre>SELECT MONTHNAME('2011-11-11 11:11:11')</pre> <p>-&gt; November</p>     |
| MONTH(d)                      | 返回日期d中的月份值，1 到 12     | <pre>SELECT MONTH('2011-11-11 11:11:11')</pre> <p>-&gt;11</p>                |
| NOW()                         | 返回当前日期和时间             | <pre>SELECT NOW()</pre> <p>-&gt; 2018-09-19 20:57:43</p>                     |
| PERIOD_ADD(period, number)    | 为 年-月 组合日期添加一个时段      | <pre>SELECT PERIOD_ADD(201703, 5);</pre> <p>-&gt; 201708</p>                 |
| PERIOD_DIFF(period1, period2) | 返回两个时段之间的月份差值         | <pre>SELECT PERIOD_DIFF(201710, 201703);</pre> <p>-&gt; 7</p>                |

|                                  |                       |  |
|----------------------------------|-----------------------|--|
| QUARTER(d)                       | 返回日期d是第几季节，返回 1 到 4   | <pre>SELECT QUARTER('2011-11-11 11:11:11')  -&gt; 4</pre>                            |
| SECOND(t)                        | 返回 t 中的秒钟值            | <pre>SELECT SECOND('1:2:3')  -&gt; 3</pre>   |
| SEC_TO_TIME(s)                   | 将以秒为单位的时间 s 转换为时分秒的格式 | <pre>SELECT SEC_TO_TIME(4320)  -&gt; 01:12:00</pre>                                  |
| STR_TO_DATE(string, format_mask) | 将字符串转变为日期             | <pre>SELECT STR_TO_DATE("August 10 2017", "%M %d %Y");  -&gt; 2017-08-10</pre>       |
| SUBDATE(d,n)                     | 日期 d 减去 n 天后的日期       | <pre>SELECT SUBDATE('2011-11-11 11:11:11', 1)  -&gt;2011-11-10 11:11:11 (默认是天)</pre> |
| SUBTIME(t,n)                     | 时间 t 减去 n 秒的时间        | <pre>SELECT SUBTIME('2011-11-11 11:11:11', 5)  -&gt;2011-11-11 11:11:06 (秒)</pre>    |
| SYSDATE()                        | 返回当前日期和时间             | <pre>SELECT SYSDATE()  -&gt; 2018-09-19 20:57:43</pre>                               |

|                                 |                                     |   |
|---------------------------------|-------------------------------------|---|
| TIME(expression)                | 提取传入表达式的时间部分                        | <pre>SELECT TIME("19:30:10");  -&gt; 19:30:10</pre>                               |
| TIME_FORMAT(t,f)                | 按表达式 f 的要求显示时间 t                    | <pre>SELECT TIME_FORMAT('11:11:11','%r')  11:11:11 AM</pre>                       |
| TIME_TO_SEC(t)                  | 将时间 t 转换为秒                          | <pre>SELECT TIME_TO_SEC('1:12:00')  -&gt; 4320</pre>                              |
| TIMEDIFF(time1, time2)          | 计算时间差值                              | <pre>SELECT TIMEDIFF("13:10:11", "13:10:10");  -&gt; 00:00:01</pre>               |
| TIMESTAMP(expression, interval) | 单个参数时，函数返回日期或日期时间表达式；有 2 个参数时，将参数加和 | <pre>SELECT TIMESTAMP("2017-07-23", "13:10:11");  -&gt; 2017-07-23 13:10:11</pre> |
| TO_DAYS(d)                      | 计算日期 d 距离 0000 年 1 月 1 日的天数         | <pre>SELECT TO_DAYS('0001-01-01 01:01:01')  -&gt; 366</pre>                       |
| WEEK(d)                         | 计算日期 d 是本年的第几个星期，范围是 0 到 53         | <pre>SELECT WEEK('2011-11-11 11:11:11')  -&gt; 45</pre>                           |



|                      |   |   |
|----------------------|---|---|
| WEEKDAY(d)           | 日期 d 是星期几，0 表示星期一，1 表示星期二               | <pre>SELECT WEEKDAY("2017-06-15");  -&gt; 3</pre>             |
| WEEKOFYEAR(d)        | 计算日期 d 是本年的第几个星期，范围是 0 到 53             | <pre>SELECT WEEKOFYEAR('2011-11-11 11:11:11')  -&gt; 45</pre> |
| YEAR(d)              | 返回年份                                    | <pre>SELECT YEAR("2017-06-15");  -&gt; 2017</pre>             |
| YEARWEEK(date, mode) | 返回年份及第几周（0到53），mode 中 0 表示周日，1表示周一，以此类推 | <pre>SELECT YEARWEEK("2017-06-15");  -&gt; 201724</pre>       |

## MySQL 高级函数

| 函数名       | 描述               | 实例   |
|-----------|------------------|--|
| BIN(x)    | 返回 x 的二进制编码      | 15 的 2 进制编码:<br><pre>SELECT BIN(15); -- 1111</pre> |
| BINARY(s) | 将字符串 s 转换为二进制字符串 | <pre>SELECT BINARY "RUNOOB" ;</pre><br>-> RUNOOB   |

|  |   |   |
|--|---|---|
| <pre>CASE expression      WHEN condition1 THE N result1      WHEN condition2 THE N result2      ...      WHEN conditionN THE N resultN      ELSE result  END</pre> | <p>CASE 表示函数开始，END 表示函数结束。如果 <b>condition1</b> 成立，则返回 <b>result1</b>，如果 <b>condition2</b> 成立，则返回 <b>result2</b>，当全部不成立则返回 <b>result</b>，而当有一个成立之后，后面的就不执行了。</p> | <pre>SELECT CASE      WHEN 1 &gt; 0      THEN '1 &gt; 0'      WHEN 2 &gt; 0      THEN '2 &gt; 0'      ELSE '3 &gt; 0'      END  -&gt;1 &gt; 0</pre> |
| <p>CAST(x AS type)</p>   | <p>转换数据类型</p>   | <p>字符串日期转换为日期：</p> <pre>SELECT CAST("2017-08-2 9" AS DATE);  -&gt; 2017-08-29</pre>   |
| <p>COALESCE(expr1, expr2, ...,<br/>expr_n)</p>   | <p>返回参数中的第一个非空表达式（从左向右）</p>   | <pre>SELECT COALESCE(NULL, NULL, NULL, 'runoob.co m', NULL, 'google.com' );  -&gt; runoob.com</pre>   |
| <p>CONNECTION_ID()</p>   | <p>返回服务器的连接数</p>  | <pre>SELECT CONNECTION_ID() ;  -&gt; 4292835</pre>  |
| <p>CONV(x,f1,f2)</p>   | <p>返回 <b>f1</b> 进制数变成 <b>f2</b> 进制数</p>   | <pre>SELECT CONV(15, 10, 2) ;  -&gt; 1111</pre>   |

|                      |  |   |
|----------------------|--|---|
| CONVERT(s USING cs)  | 函数将字符串 <b>s</b> 的字符集变成 <b>cs</b>                               | <pre>SELECT CHARSET('ABC')  -&gt;utf-8  SELECT CHARSET(CONVERT ('ABC' USING gbk))  -&gt;gbk</pre> |
| CURRENT_USER()       | 返回当前用户   | <pre>SELECT CURRENT_USER();  -&gt; guest@%</pre>  |
| DATABASE()           | 返回当前数据库名   | <pre>SELECT DATABASE();  -&gt; runoob</pre>   |
| IF(expr,v1,v2)       | 如果表达式 <b>expr</b> 成立，返回结果 <b>v1</b> ；否则，返回结果 <b>v2</b> 。       | <pre>SELECT IF(1 &gt; 0,'正确', '错误')  -&gt;正确</pre>  |
| <u>IFNULL(v1,v2)</u> | 如果 <b>v1</b> 的值不为 <b>NULL</b> ，则返回 <b>v1</b> ，否则返回 <b>v2</b> 。 | <pre>SELECT IFNULL(null,'He llo Word')  -&gt;Hello Word</pre>                                     |
| ISNULL(expression)   | 判断表达式是否为空  | <pre>SELECT ISNULL(NULL);  -&gt;1</pre>   |

|                      |  |  |
|----------------------|--|--|
| LAST_INSERT_ID()     | 返回最近生成的 AUTO_INCREMENT 值   | <pre>SELECT LAST_INSERT_ID( );  -&gt;6</pre>     |
| NULLIF(expr1, expr2) | 比较两个字符串，如果字符串 <b>expr1</b> 与 <b>expr2</b> 相等 返回 NULL，否则返回 <b>expr1</b> | <pre>SELECT NULLIF(25, 25);  -&gt;</pre>         |
| SESSION_USER()       | 返回当前用户   | <pre>SELECT SESSION_USER();  -&gt; guest@%</pre> |
| SYSTEM_USER()        | 返回当前用户   | <pre>SELECT SYSTEM_USER();  -&gt; guest@%</pre>  |
| USER()               | 返回当前用户   | <pre>SELECT USER();  -&gt; guest@%</pre>         |
| VERSION()            | 返回数据库的版本号  | <pre>SELECT VERSION()  -&gt; 5.6.34</pre>        |

[MySQL UNION 操作符](#)

[MySQL IFNULL\(\) 函数](#)

[点我分享笔记](#)

反馈/建议

