MySQL 是最流行的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所使用的SQL语言是用于访问数据库的最常用标准化语言。

MySQL由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择MySQL作为网站数据库。

**MySQL 安装**

本教程的系统平台：CentOS release 6.6 (Final) 64位。

**一、安装编译工具及库文件**

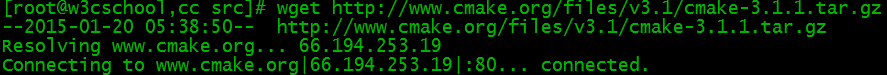
yum -y install gcc gcc-c++ make autoconf libtool-ltdl-devel gd-devel freetype-devel libxml2-devel libjpeg-devel libpng-devel openssl-devel curl-devel bison patch unzip libmcrypt-devel libmhash-devel ncurses-devel sudo bzip2 flex libaio-devel

**二、 安装cmake 编译器**

cmake 版本：cmake-3.1.1。

1、下载地址：<http://www.cmake.org/files/v3.1/cmake-3.1.1.tar.gz>

[root@w3cschool,cc src]# wget http://www.cmake.org/files/v3.1/cmake-3.1.1.tar.gz



2、解压安装包

[root@w3cschool,cc src]# tar zxvf cmake-3.1.1.tar.gz

3、进入安装包目录

[root@w3cschool,cc src]# cd cmake-3.1.1

4、编译安装

[root@w3cschool,cc cmake-3.1.1]# ./bootstrap

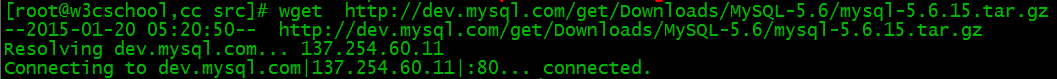
[root@w3cschool,cc cmake-3.1.1]# make && make install

**三、安装 MySQL**

MySQL版本：mysql-5.6.15。

1、下载地址： <http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.15.tar.gz>

[root@w3cschool,cc src]# wget http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.15.tar.gz



2、解压安装包

[root@w3cschool,cc src]# tar zxvf mysql-5.6.15.tar.gz

3、进入安装包目录

[root@w3cschool,cc src]# cd mysql-5.6.15

4、编译安装

[root@w3cschool,cc mysql-5.6.15]# cmake -DCMAKE\_INSTALL\_PREFIX=/usr/local/webserver/mysql/ -DMYSQL\_UNIX\_ADDR=/tmp/mysql.sock -DDEFAULT\_CHARSET=utf8 -DDEFAULT\_COLLATION=utf8\_general\_ci -DWITH\_EXTRA\_CHARSETS=all -DWITH\_MYISAM\_STORAGE\_ENGINE=1 -DWITH\_INNOBASE\_STORAGE\_ENGINE=1 -DWITH\_MEMORY\_STORAGE\_ENGINE=1 -DWITH\_READLINE=1 -DWITH\_INNODB\_MEMCACHED=1 -DWITH\_DEBUG=OFF -DWITH\_ZLIB=bundled -DENABLED\_LOCAL\_INFILE=1 -DENABLED\_PROFILING=ON -DMYSQL\_MAINTAINER\_MODE=OFF -DMYSQL\_DATADIR=/usr/local/webserver/mysql/data -DMYSQL\_TCP\_PORT=3306

[root@w3cschool,cc mysql-5.6.15]# make && make install

5、查看mysql版本:

[root@w3cschool,cc mysql-5.6.15]# /usr/local/webserver/mysql/bin/mysql --version

mysql3

到此，mysql安装完成。

**MySQL 配置**

1、创建mysql运行使用的用户mysql：

[root@bogon conf]# /usr/sbin/groupadd mysql

[root@bogon conf]# /usr/sbin/useradd -g mysql mysql

2、创建binlog和库的存储路径并赋予mysql用户权限

[root@w3cschool,cc mysql]# mkdir -p /usr/local/webserver/mysql/binlog /www/data\_mysql

[root@w3cschool,cc mysql]# chown mysql.mysql /usr/local/webserver/mysql/binlog/ /www/data\_mysql/

2、创建my.cnf配置文件

将/etc/my.cnf替换为下面内容

[root@w3cschool,cc mysql]# cat /etc/my.cnf

[client]

port = 3306

socket = /tmp/mysql.sock

[mysqld]

replicate-ignore-db = mysql

replicate-ignore-db = test

replicate-ignore-db = information\_schema

user = mysql

port = 3306

socket = /tmp/mysql.sock

basedir = /usr/local/webserver/mysql

datadir = /www/data\_mysql

log-error = /usr/local/webserver/mysql/mysql\_error.log

pid-file = /usr/local/webserver/mysql/mysql.pid

open\_files\_limit = 65535

back\_log = 600

max\_connections = 5000

max\_connect\_errors = 1000

table\_open\_cache = 1024

external-locking = FALSE

max\_allowed\_packet = 32M

sort\_buffer\_size = 1M

join\_buffer\_size = 1M

thread\_cache\_size = 600

#thread\_concurrency = 8

query\_cache\_size = 128M

query\_cache\_limit = 2M

query\_cache\_min\_res\_unit = 2k

default-storage-engine = MyISAM

default-tmp-storage-engine=MYISAM

thread\_stack = 192K

transaction\_isolation = READ-COMMITTED

tmp\_table\_size = 128M

max\_heap\_table\_size = 128M

log-slave-updates

log-bin = /usr/local/webserver/mysql/binlog/binlog

binlog-do-db=oa\_fb

binlog-ignore-db=mysql

binlog\_cache\_size = 4M

binlog\_format = MIXED

max\_binlog\_cache\_size = 8M

max\_binlog\_size = 1G

relay-log-index = /usr/local/webserver/mysql/relaylog/relaylog

relay-log-info-file = /usr/local/webserver/mysql/relaylog/relaylog

relay-log = /usr/local/webserver/mysql/relaylog/relaylog

expire\_logs\_days = 10

key\_buffer\_size = 256M

read\_buffer\_size = 1M

read\_rnd\_buffer\_size = 16M

bulk\_insert\_buffer\_size = 64M

myisam\_sort\_buffer\_size = 128M

myisam\_max\_sort\_file\_size = 10G

myisam\_repair\_threads = 1

myisam\_recover

interactive\_timeout = 120

wait\_timeout = 120

skip-name-resolve

#master-connect-retry = 10

slave-skip-errors = 1032,1062,126,1114,1146,1048,1396

#master-host = 192.168.1.2

#master-user = username

#master-password = password

#master-port = 3306

server-id = 1

loose-innodb-trx=0

loose-innodb-locks=0

loose-innodb-lock-waits=0

loose-innodb-cmp=0

loose-innodb-cmp-per-index=0

loose-innodb-cmp-per-index-reset=0

loose-innodb-cmp-reset=0

loose-innodb-cmpmem=0

loose-innodb-cmpmem-reset=0

loose-innodb-buffer-page=0

loose-innodb-buffer-page-lru=0

loose-innodb-buffer-pool-stats=0

loose-innodb-metrics=0

loose-innodb-ft-default-stopword=0

loose-innodb-ft-inserted=0

loose-innodb-ft-deleted=0

loose-innodb-ft-being-deleted=0

loose-innodb-ft-config=0

loose-innodb-ft-index-cache=0

loose-innodb-ft-index-table=0

loose-innodb-sys-tables=0

loose-innodb-sys-tablestats=0

loose-innodb-sys-indexes=0

loose-innodb-sys-columns=0

loose-innodb-sys-fields=0

loose-innodb-sys-foreign=0

loose-innodb-sys-foreign-cols=0

slow\_query\_log\_file=/usr/local/webserver/mysql/mysql\_slow.log

long\_query\_time = 1

[mysqldump]

quick

max\_allowed\_packet = 32M

3、初始化数据库

[root@w3cschool,cc mysql]#/usr/local/webserver/mysql/scripts/mysql\_install\_db --defaults-file=/etc/my.cnf  --user=mysql

显示如下信息：

Installing MySQL system tables...2015-01-26 20:18:51 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit\_defaults\_for\_timestamp server option (see documentation for more details).

OK

Filling help tables...2015-01-26 20:18:57 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit\_defaults\_for\_timestamp server option (see documentation for more details).

OK

...

4、创建开机启动脚本

[root@w3cschool,cc mysql]# cd /usr/local/webserver/mysql/

[root@w3cschool,cc mysql]# cp support-files/mysql.server /etc/rc.d/init.d/mysqld

[root@w3cschool,cc mysql]# chkconfig --add mysqld

[root@w3cschool,cc mysql]# chkconfig --level 35 mysqld on

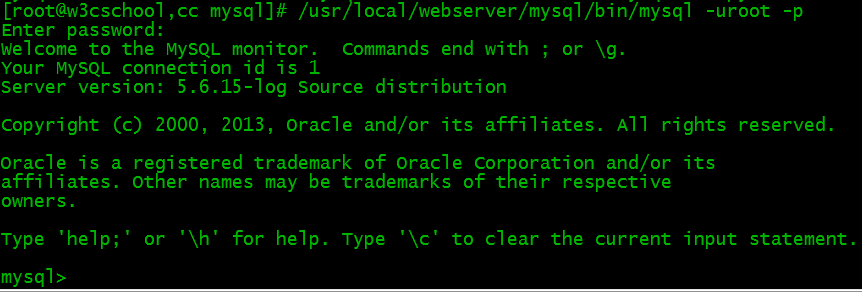
4、启动mysql服务器

[root@w3cschool,cc mysql]# service mysqld start

mysql4

5、连接 MySQL

[root@w3cschool,cc mysql]# /usr/local/webserver/mysql/bin/mysql -u root -p



**修改MySQL用户密码**

mysqladmin -u用户名 -p旧密码 password 新密码

或进入mysql命令行

SET PASSWORD FOR '用户名'@'主机' = PASSWORD(‘密码');

创建新用户并授权:

grant all privileges on \*.\* to 用户名@'%' identified by '密码' with grant option;

**其他命令**

* 启动：service mysqld start
* 停止：service mysqld stop
* 重启：service mysqld restart
* 重载配置：service mysqld reload

# 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，冗余可以使系统速度更快。
* **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，所以你不需要支付额外的费用。
* 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可以允许于多个系统上，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](http://www.w3cschool.cc/php/php-tutorial.html)来了解该语言。

# MySQL 安装

所有平台的Mysql下载地址为： [MySQL 下载](http://www.mysql.com/downloads). 挑选你需要的 *MySQL Community Server* 版本及对应的平台。

## 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和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。

以下安装Mysql RMP的实例是在SuSE Linux系统上进行，当然该安装步骤也适合应用于其他支持RPM的Linux系统，如:Centos。

安装步骤如下：

使用root用户登陆你的Linux系统。

下载Mysql RPM包，下载地址为：[MySQL 下载](http://www.mysql.com/downloads)。

通过以下命令执行Mysql安装，rpm包为你下载的rpm包：

[root@host]# rpm -i MySQL-5.0.9-0.i386.rpm

以上安装mysql服务器的过程会创建mysql用户，并创建一个mysql配置文件my.cnf。

你可以在/usr/bin和/usr/sbin中找到所有与MySQL相关的二进制文件。所有数据表和数据库将在/var/lib/mysql目录中创建。

以下是一些mysql可选包的安装过程，你可以根据自己的需要来安装：

[root@host]# rpm -i MySQL-client-5.0.9-0.i386.rpm

[root@host]# rpm -i MySQL-devel-5.0.9-0.i386.rpm

[root@host]# rpm -i MySQL-shared-5.0.9-0.i386.rpm

[root@host]# rpm -i MySQL-bench-5.0.9-0.i386.rpm

## Window上安装Mysql

Window上安装Mysql相对来说会较为简单，你只需要载 [MySQL 下载](http://www.mysql.com/downloads)中下载window版本的mysql安装包，并解压安装包。

双击 setup.exe 文件，接下来你只需要安装默认的配置点击"next"即可，默认情况下安装信息会在C:\mysql目录中。

接下来你可以通过"开始" =》在搜索框中输入 " cmd" 命令 =》 在命令提示符上切换到 C:\mysql\bin 目录，并输入一下命令：

mysqld.exe --console

如果安装成功以上命令将输出一些mysql启动及InnoDB信息。

## 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 on linux ，在window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。

## Linux系统启动时启动 MySQL

如果你需要在Linux系统启动时启动 MySQL 服务器，你需要在 /etc/rc.local 文件中添加以下命令：

/etc/init.d/mysqld start

同样，你需要将 mysqld 二进制文件添加到 /etc/init.d/ 目录中。

# MySQL 管理

## 启动及关闭 MySQL 服务器

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

./safe\_mysqld &

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：**在注意需要执行 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，你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 password;

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。
* **SHOW DATABASES:**列出 MySQL 数据库管理系统的数据库列表。
* **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。
* **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。
* **SHOW INDEX FROM 数据表:** 显示数据表的详细索引信息，包括PRIMARY KEY（主键）。
* **SHOW TABLE STATUS LIKE 数据表\G:** 该命令将输出Mysql数据库管理系统的性能及统计信息。

# 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。你过你想了解Mysql在PHP中的应用，可以访问我们的 [PHP中使用Mysql介绍](http://www.w3cschool.cc/php/php-mysql-intro.html)。

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

mysql\_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 statement");

mysql\_fetch\_array()

mysql\_connect(),mysql\_close()

以下实例展示了PHP调用mysql函数的语法：

<html>

<head>

<title>PHP with MySQL</title>

</head>

<body>

<?php

$retval = mysql\_function(value, [value,...]);

if( !$retval )

{

die ( "Error: a related error message" );

}

// Otherwise MySQL or PHP Statements

?>

</body>

</html>

从下一章开始，我们将学习到更多的MySQL功能函数。

# 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 mysql\_connect() 函数来连接数据库。

该函数有5个参数，在成功链接到MySQL后返回连接标识，失败返回 FALSE 。

### 语法

connection mysql\_connect(server,user,passwd,new\_link,client\_flag);

**参数说明：**

|  |  |
| --- | --- |
| **参数** | **描述** |
| server | 可选。规定要连接的服务器。  可以包括端口号，例如 "hostname:port"，或者到本地套接字的路径，例如对于 localhost 的 ":/path/to/socket"。  如果 PHP 指令 mysql.default\_host 未定义（默认情况），则默认值是 'localhost:3306'。 |
| user | 可选。用户名。默认值是服务器进程所有者的用户名。 |
| passwd | 可选。密码。默认值是空密码。 |
| new\_link | 可选。如果用同样的参数第二次调用 mysql\_connect()，将不会建立新连接，而将返回已经打开的连接标识。参数 new\_link 改变此行为并使 mysql\_connect() 总是打开新的连接，甚至当 mysql\_connect() 曾在前面被用同样的参数调用过。 |
| client\_flag | 可选。client\_flags 参数可以是以下常量的组合：   * MYSQL\_CLIENT\_SSL - 使用 SSL 加密 * MYSQL\_CLIENT\_COMPRESS - 使用压缩协议 * MYSQL\_CLIENT\_IGNORE\_SPACE - 允许函数名后的间隔 * MYSQL\_CLIENT\_INTERACTIVE - 允许关闭连接之前的交互超时非活动时间 |

你可以使用PHP的 mysql\_close() 函数来断开与MySQL数据库的链接。

该函数只有一个参数为mysql\_connect()函数创建连接成功后返回的 MySQL 连接标识符。

### 语法

bool mysql\_close ( resource $link\_identifier );

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 mysql\_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。

**注释：**mysql\_close() 不会关闭由 mysql\_pconnect() 建立的持久连接。

### 实例

你可以尝试以下实例来连接到你的 MySQL 服务器:

# MySQL 创建数据库

## 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。

### 实例

以下命令简单的演示了创建数据库的过程，数据名为 TUTORIALS:

[root@host]# mysqladmin -u root -p create TUTORIALS

Enter password:\*\*\*\*\*\*

以上命令执行成功后会创建 MySQL 数据库 TUTORIALS。

## 使用 PHP脚本 创建数据库

PHP使用 mysql\_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

bool mysql\_query( sql, connection );

|  |  |
| --- | --- |
| **参数** | **描述** |
| sql | 必需。规定要发送的 SQL 查询。注释：查询字符串不应以分号结束。 |
| connection | 可选。规定 SQL 连接标识符。如果未规定，则使用上一个打开的连接。 |

### 实例

以下实例演示了使用PHP来创建一个数据库：

<html>

<head>

<title>Creating MySQL Database</title>

</head>

<body>

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

echo 'Connected successfully<br />';

$sql = 'CREATE DATABASE TUTORIALS';

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not create database: ' . mysql\_error());

}

echo "Database TUTORIALS created successfully\n";

mysql\_close($conn);

?>

</body>

</html>

# MySQL 删除数据库

## 使用 mysqladmin 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，可以使用 mysql mysqladmin 命令来创建数据库。

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。

以下实例删除数据库TUTORIALS(该数据库在前一章节已创建)：

[root@host]# mysqladmin -u root -p drop TUTORIALS

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 'TUTORIALS' database [y/N] y

Database "TUTORIALS" dropped

## 使用PHP脚本删除数据库

PHP使用 mysql\_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

bool mysql\_query( sql, connection );

|  |  |
| --- | --- |
| **参数** | **描述** |
| sql | 必需。规定要发送的 SQL 查询。注释：查询字符串不应以分号结束。 |
| connection | 可选。规定 SQL 连接标识符。如果未规定，则使用上一个打开的连接。 |

### 实例

以下实例演示了使用PHP mysql\_query函数来删除数据库：

<html>

<head>

<title>Deleting MySQL Database</title>

</head>

<body>

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('连接失败: ' . mysql\_error());

}

echo '连接成功<br />';

$sql = 'DROP DATABASE TUTORIALS';

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('删除数据库失败: ' . mysql\_error());

}

echo "数据库 TUTORIALS 删除成功\n";

mysql\_close($conn);

?>

</body>

</html>

**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 TUTORIALS:

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

Enter password:\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql>

执行以上命令后，你就已经成功选择了 TUTORIALS 数据库，在后续的操作中都会在 TUTORIALS 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 mysql\_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

bool mysql\_select\_db( db\_name, connection );

|  |  |
| --- | --- |
| **参数** | **描述** |
| db\_name | 必需。规定要选择的数据库。 |
| connection | 可选。规定 MySQL 连接。如果未指定，则使用上一个连接。 |

### 实例

以下实例展示了如何使用 mysql\_select\_db 函数来选取一个数据库：

<html>

<head>

<title>Selecting MySQL Database</title>

</head>

<body>

<?php

$dbhost = 'localhost:3036';

$dbuser = 'guest';

$dbpass = 'guest123';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

echo 'Connected successfully';

mysql\_select\_db( 'TUTORIALS' );

mysql\_close($conn);

?>

</body>

</html>

# 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 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 浮点数值 |
| DECIMAL | 对DECIMAL(M,D) ，如果M>D，为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 (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 | 8 | 1970-01-01 00:00:00/2037 年某时 | YYYYMMDD 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 |
| LOGNGBLOB | 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。它们只是可容纳值的最大长度不同。

有4种TEXT类型：TINYTEXT、TEXT、MEDIUMTEXT和LONGTEXT。这些对应4种BLOB类型，有相同的最大长度和存储需求。

# 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 TUTORIALS 数据库中创建数据表tutorials\_tbl：

tutorials\_tbl(

tutorial\_id INT NOT NULL AUTO\_INCREMENT,

tutorial\_title VARCHAR(100) NOT NULL,

tutorial\_author VARCHAR(40) NOT NULL,

submission\_date DATE,

PRIMARY KEY ( tutorial\_id )

);

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。

## 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 tutorials\_tbl 实例:

root@host# mysql -u root -p

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> CREATE TABLE tutorials\_tbl(

-> tutorial\_id INT NOT NULL AUTO\_INCREMENT,

-> tutorial\_title VARCHAR(100) NOT NULL,

-> tutorial\_author VARCHAR(40) NOT NULL,

-> submission\_date DATE,

-> PRIMARY KEY ( tutorial\_id )

-> );

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 mysql\_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

bool mysql\_query( sql, connection );

|  |  |
| --- | --- |
| **参数** | **描述** |
| sql | 必需。规定要发送的 SQL 查询。注释：查询字符串不应以分号结束。 |
| connection | 可选。规定 SQL 连接标识符。如果未规定，则使用上一个打开的连接。 |

### 实例

以下实例使用了PHP脚本来创建数据表：

<html>

<head>

<title>Creating MySQL Tables</title>

</head>

<body>

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

echo 'Connected successfully<br />';

$sql = "CREATE TABLE tutorials\_tbl( ".

"tutorial\_id INT NOT NULL AUTO\_INCREMENT, ".

"tutorial\_title VARCHAR(100) NOT NULL, ".

"tutorial\_author VARCHAR(40) NOT NULL, ".

"submission\_date DATE, ".

"PRIMARY KEY ( tutorial\_id )); ";

mysql\_select\_db( 'TUTORIALS' );

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not create table: ' . mysql\_error());

}

echo "Table created successfully\n";

mysql\_close($conn);

?>

</body>

</html>

# 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tutorials\_tbl:

root@host# mysql -u root -p

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> DROP TABLE tutorials\_tbl

Query OK, 0 rows affected (0.8 sec)

mysql>

## 使用PHP脚本删除数据表

PHP使用 mysql\_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。

h3> 语法

bool mysql\_query( sql, connection );

|  |  |
| --- | --- |
| **参数** | **描述** |
| sql | 必需。规定要发送的 SQL 查询。注释：查询字符串不应以分号结束。 |
| connection | 可选。规定 SQL 连接标识符。如果未规定，则使用上一个打开的连接。 |

### 实例

以下实例使用了PHP脚本删除数据表tutorials\_tbl:

<html>

<head>

<title>Creating MySQL Tables</title>

</head>

<body>

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

echo 'Connected successfully<br />';

$sql = "DROP TABLE tutorials\_tbl";

mysql\_select\_db( 'TUTORIALS' );

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not delete table: ' . mysql\_error());

}

echo "Table deleted successfully\n";

mysql\_close($conn);

?>

</body>

</html>

# 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 tutorials\_tbl 插入数据

### 实例

以下实例中我们将想 tutorials\_tbl 表插入三条数据:

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> INSERT INTO tutorials\_tbl

->(tutorial\_title, tutorial\_author, submission\_date)

->VALUES

->("Learn PHP", "John Poul", NOW());

Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO tutorials\_tbl

->(tutorial\_title, tutorial\_author, submission\_date)

->VALUES

->("Learn MySQL", "Abdul S", NOW());

Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO tutorials\_tbl

->(tutorial\_title, tutorial\_author, submission\_date)

->VALUES

->("JAVA Tutorial", "Sanjay", '2007-05-06');

Query OK, 1 row affected (0.01 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 tutorial\_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。

## 使用PHP脚本插入数据

你可以使用PHP 的 mysql\_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

bool mysql\_query( sql, connection );

|  |  |
| --- | --- |
| **参数** | **描述** |
| sql | 必需。规定要发送的 SQL 查询。注释：查询字符串不应以分号结束。 |
| connection | 可选。规定 SQL 连接标识符。如果未规定，则使用上一个打开的连接。 |

### 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：

<html>

<head>

<title>Add New Record in MySQL Database</title>

</head>

<body>

<?php

if(isset($\_POST['add']))

{

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

if(! get\_magic\_quotes\_gpc() )

{

$tutorial\_title = addslashes ($\_POST['tutorial\_title']);

$tutorial\_author = addslashes ($\_POST['tutorial\_author']);

}

else

{

$tutorial\_title = $\_POST['tutorial\_title'];

$tutorial\_author = $\_POST['tutorial\_author'];

}

$submission\_date = $\_POST['submission\_date'];

$sql = "INSERT INTO tutorials\_tbl ".

"(tutorial\_title,tutorial\_author, submission\_date) ".

"VALUES ".

"('$tutorial\_title','$tutorial\_author','$submission\_date')";

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not enter data: ' . mysql\_error());

}

echo "Entered data successfully\n";

mysql\_close($conn);

}

else

{

?>

<form method="post" action="<?php $\_PHP\_SELF ?>">

<table width="600" border="0" cellspacing="1" cellpadding="2">

<tr>

<td width="250">Tutorial Title</td>

<td>

<input name="tutorial\_title" type="text" id="tutorial\_title">

</td>

</tr>

<tr>

<td width="250">Tutorial Author</td>

<td>

<input name="tutorial\_author" type="text" id="tutorial\_author">

</td>

</tr>

<tr>

<td width="250">Submission Date [ yyyy-mm-dd ]</td>

<td>

<input name="submission\_date" type="text" id="submission\_date">

</td>

</tr>

<tr>

<td width="250"> </td>

<td> </td>

</tr>

<tr>

<td width="250"> </td>

<td>

<input name="add" type="submit" id="add" value="Add Tutorial">

</td>

</tr>

</table>

</form>

<?php

}

?>

</body>

</html>

在我们接收用户提交的数据时，为了数据的安全性我们需要使用 get\_magic\_quotes\_gpc() 函数来判断特殊字符的转义是否已经开启。如果这个选项为off（未开启），返回0，那么我们就必须调用addslashes 这个函数来为字符串增加转义。

义。

你也可以添加其他检查数据的方法，比如邮箱格式验证，电话号码验证，是否为整数验证等。

# 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 field1, field2,...fieldN table\_name1, table\_name2...

[WHERE Clause]

[OFFSET M ][LIMIT N]

* 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。
* 你可以使用 LIMIT 属性来设定返回的记录数。

## 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 tutorials\_tbl 的数据：

### 实例

以下实例将返回数据表tutorials\_tbl的所有记录:

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> SELECT \* from tutorials\_tbl

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

| tutorial\_id | tutorial\_title | tutorial\_author | submission\_date |

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

| 1 | Learn PHP | John Poul | 2007-05-21 |

| 2 | Learn MySQL | Abdul S | 2007-05-21 |

| 3 | JAVA Tutorial | Sanjay | 2007-05-21 |

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

3 rows in set (0.01 sec)

mysql>

## 使用PHP脚本来获取数据

使用PHP函数的mysql\_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 mysql\_fetch\_array() 来使用或输出所有查询的数据。

mysql\_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 tutorials\_tbl 中读取所有记录。

### 实例

尝试以下实例来显示数据表 tutorials\_tbl 的所有记录。

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'SELECT tutorial\_id, tutorial\_title,

tutorial\_author, submission\_date

FROM tutorials\_tbl';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_array($retval, MYSQL\_ASSOC))

{

echo "Tutorial ID :{$row['tutorial\_id']} <br> ".

"Title: {$row['tutorial\_title']} <br> ".

"Author: {$row['tutorial\_author']} <br> ".

"Submission Date : {$row['submission\_date']} <br> ".

"--------------------------------<br>";

}

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

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 mysql\_fetch\_array()函数第二个参数为MYSQL\_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mysql\_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mysql\_fetch\_assoc()函数来输出数据表tutorial\_tbl的所有记录：

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'SELECT tutorial\_id, tutorial\_title,

tutorial\_author, submission\_date

FROM tutorials\_tbl';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_assoc($retval))

{

echo "Tutorial ID :{$row['tutorial\_id']} <br> ".

"Title: {$row['tutorial\_title']} <br> ".

"Author: {$row['tutorial\_author']} <br> ".

"Submission Date : {$row['submission\_date']} <br> ".

"--------------------------------<br>";

}

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

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

### 实例

以下实例使用MYSQL\_NUM参数显示数据表tutorials\_tbl的所有记录:

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'SELECT tutorial\_id, tutorial\_title,

tutorial\_author, submission\_date

FROM tutorials\_tbl';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_array($retval, MYSQL\_NUM))

{

echo "Tutorial ID :{$row[0]} <br> ".

"Title: {$row[1]} <br> ".

"Author: {$row[2]} <br> ".

"Submission Date : {$row[3]} <br> ".

"--------------------------------<br>";

}

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

以上三个实例输出结果都一样。

## 内存释放

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mysql\_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:

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'SELECT tutorial\_id, tutorial\_title,

tutorial\_author, submission\_date

FROM tutorials\_tbl';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_array($retval, MYSQL\_NUM))

{

echo "Tutorial ID :{$row[0]} <br> ".

"Title: {$row[1]} <br> ".

"Author: {$row[2]} <br> ".

"Submission Date : {$row[3]} <br> ".

"--------------------------------<br>";

}

mysql\_free\_result($retval);

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

# 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。 |

如果我们想再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 tutorials\_tbl 中的数据：

实例

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

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> SELECT \* from tutorials\_tbl WHERE tutorial\_author='Sanjay';

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

| tutorial\_id | tutorial\_title | tutorial\_author | submission\_date |

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

| 3 | JAVA Tutorial | Sanjay | 2007-05-21 |

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

1 rows in set (0.01 sec)

mysql>

除非你使用 LIKE 来比较字符串，否则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

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> SELECT \* from tutorials\_tbl \

WHERE BINARY tutorial\_author='sanjay';

Empty set (0.02 sec)

mysql>

## 使用PHP脚本读取数据

你可以使用PHP函数的mysql\_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 mysql\_fetch\_array() 来输出所有查询的数据。

### 实例

以下实例将从 tutorials\_tbl 表中返回使用 tutorial\_author 字段值为 Sanjay 的记录：

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'SELECT tutorial\_id, tutorial\_title,

tutorial\_author, submission\_date

FROM tutorials\_tbl

WHERE tutorial\_author="Sanjay"';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_array($retval, MYSQL\_ASSOC))

{

echo "Tutorial ID :{$row['tutorial\_id']} <br> ".

"Title: {$row['tutorial\_title']} <br> ".

"Author: {$row['tutorial\_author']} <br> ".

"Submission Date : {$row['submission\_date']} <br> ".

"--------------------------------<br>";

}

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

# 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tutorials\_tbl表中指定的数据：

### 实例

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

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> UPDATE tutorials\_tbl

-> SET tutorial\_title='Learning JAVA'

-> WHERE tutorial\_id=3;

Query OK, 1 row affected (0.04 sec)

Rows matched: 1 Changed: 1 Warnings: 0

mysql>

## 使用PHP脚本更新数据

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

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 tutorial\_id 为3的 tutorial\_title 字段的数据。

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'UPDATE tutorials\_tbl

SET tutorial\_title="Learning JAVA"

WHERE tutorial\_id=3';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not update data: ' . mysql\_error());

}

echo "Updated data successfully\n";

mysql\_close($conn);

?>

# 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tutorials\_tbl所选的数据。

### 实例

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

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> DELETE FROM tutorials\_tbl WHERE tutorial\_id=3;

Query OK, 1 row affected (0.23 sec)

mysql>

## 使用 PHP 脚本删除数据

PHP使用 mysql\_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tutorial\_tbl表中tutorial\_id为3的记录:

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'DELETE FROM tutorials\_tbl

WHERE tutorial\_id=3';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not delete data: ' . mysql\_error());

}

echo "Deleted data successfully\n";

mysql\_close($conn);

?>

# 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 "tutorial\_author = 'Sanjay'"。

但是有时候我们需要获取 tutorial\_author 字段含有 "jay" 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 table\_name1, table\_name2...

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 tutorials\_tbl 中读取数据。

### 实例

以下是我们将tutorials\_tbl表中获取tutorial\_author字段中以"jay"为结尾的的所有记录：

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> SELECT \* from tutorials\_tbl

-> WHERE tutorial\_author LIKE '%jay';

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

| tutorial\_id | tutorial\_title | tutorial\_author | submission\_date |

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

| 3 | JAVA Tutorial | Sanjay | 2007-05-21 |

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

1 rows in set (0.01 sec)

mysql>

## 在PHP脚本中使用 LIKE 子句

你可以使用PHP函数的mysql\_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 mysql\_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mysql\_fetch\_array() 函数。

### 实例

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

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'SELECT tutorial\_id, tutorial\_title,

tutorial\_author, submission\_date

FROM tutorials\_tbl

WHERE tutorial\_author LIKE "%jay%"';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_array($retval, MYSQL\_ASSOC))

{

echo "Tutorial ID :{$row['tutorial\_id']} <br> ".

"Title: {$row['tutorial\_title']} <br> ".

"Author: {$row['tutorial\_author']} <br> ".

"Submission Date : {$row['submission\_date']} <br> ".

"--------------------------------<br>";

}

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

# 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中方式来进行排序，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 tutorials\_tbl 中的数据：

### 实例

尝试以下实例，结果将按升序排列

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> SELECT \* from tutorials\_tbl ORDER BY tutorial\_author ASC

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

| tutorial\_id | tutorial\_title | tutorial\_author | submission\_date |

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

| 2 | Learn MySQL | Abdul S | 2007-05-24 |

| 1 | Learn PHP | John Poul | 2007-05-24 |

| 3 | JAVA Tutorial | Sanjay | 2007-05-06 |

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

3 rows in set (0.42 sec)

mysql>

读取 tutorials\_tbl 表中所有数据并按 tutorial\_author 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mysql\_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 mysql\_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 tutorial\_author 字段的降序排列后返回。

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'SELECT tutorial\_id, tutorial\_title,

tutorial\_author, submission\_date

FROM tutorials\_tbl

ORDER BY tutorial\_author DESC';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_array($retval, MYSQL\_ASSOC))

{

echo "Tutorial ID :{$row['tutorial\_id']} <br> ".

"Title: {$row['tutorial\_title']} <br> ".

"Author: {$row['tutorial\_author']} <br> ".

"Submission Date : {$row['submission\_date']} <br> ".

"--------------------------------<br>";

}

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

# Mysql Join的使用

在前几章节中，我们已经学会了如果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。

以下我们将演示MySQL LEFT JOIN 和 JOIN 的使用的不同之处。

## 在命令提示符中使用JOIN

我们在TUTORIALS数据库中有两张表 tcount\_tbl 和 tutorials\_tbl。两张数据表数据如下：

### 实例

尝试以下实例：

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> SELECT \* FROM tcount\_tbl;

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

| tutorial\_author | tutorial\_count |

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

| mahran | 20 |

| mahnaz | NULL |

| Jen | NULL |

| Gill | 20 |

| John Poul | 1 |

| Sanjay | 1 |

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

6 rows in set (0.01 sec)

mysql> SELECT \* from tutorials\_tbl;

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

| tutorial\_id | tutorial\_title | tutorial\_author | submission\_date |

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

| 1 | Learn PHP | John Poul | 2007-05-24 |

| 2 | Learn MySQL | Abdul S | 2007-05-24 |

| 3 | JAVA Tutorial | Sanjay | 2007-05-06 |

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

3 rows in set (0.00 sec)

mysql>

接下来我们就使用MySQL的JOIN来连接以上两张表来读取tutorials\_tbl表中所有tutorial\_author字段在tcount\_tbl表对应的tutorial\_count字段值：

mysql> SELECT a.tutorial\_id, a.tutorial\_author, b.tutorial\_count

-> FROM tutorials\_tbl a, tcount\_tbl b

-> WHERE a.tutorial\_author = b.tutorial\_author;

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

| tutorial\_id | tutorial\_author | tutorial\_count |

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

| 1 | John Poul | 1 |

| 3 | Sanjay | 1 |

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

2 rows in set (0.01 sec)

mysql>

## 在PHP脚本中使用JOIN

PHP 中使用mysql\_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mysql\_query()函数的参数。

尝试如下实例:

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

$sql = 'SELECT a.tutorial\_id, a.tutorial\_author, b.tutorial\_count

FROM tutorials\_tbl a, tcount\_tbl b

WHERE a.tutorial\_author = b.tutorial\_author';

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_array($retval, MYSQL\_ASSOC))

{

echo "Author:{$row['tutorial\_author']} <br> ".

"Count: {$row['tutorial\_count']} <br> ".

"Tutorial ID: {$row['tutorial\_id']} <br> ".

"--------------------------------<br>";

}

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

## 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，理解MySQL LEFT JOIN的应用：

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> SELECT a.tutorial\_id, a.tutorial\_author, b.tutorial\_count

-> FROM tutorials\_tbl a LEFT JOIN tcount\_tbl b

-> ON a.tutorial\_author = b.tutorial\_author;

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

| tutorial\_id | tutorial\_author | tutorial\_count |

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

| 1 | John Poul | 1 |

| 2 | Abdul S | NULL |

| 3 | Sanjay | 1 |

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

3 rows in set (0.02 sec)

以上实例中使用了LEFT JOIN，该语句会读取左边的数据表tutorials\_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tutorial\_author字段值。

# 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），当比较的的两个值为NULL时返回true。

关于 NULL 的条件比较运算是比较特殊的。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。

## 在命令提示符中使用 NULL 值

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

### 实例

尝试以下实例:

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

Enter password:\*\*\*\*\*\*\*

mysql> use TUTORIALS;

Database changed

mysql> create table tcount\_tbl

-> (

-> tutorial\_author varchar(40) NOT NULL,

-> tutorial\_count INT

-> );

Query OK, 0 rows affected (0.05 sec)

mysql> INSERT INTO tcount\_tbl

-> (tutorial\_author, tutorial\_count) values ('mahran', 20);

mysql> INSERT INTO tcount\_tbl

-> (tutorial\_author, tutorial\_count) values ('mahnaz', NULL);

mysql> INSERT INTO tcount\_tbl

-> (tutorial\_author, tutorial\_count) values ('Jen', NULL);

mysql> INSERT INTO tcount\_tbl

-> (tutorial\_author, tutorial\_count) values ('Gill', 20);

mysql> SELECT \* from tcount\_tbl;

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

| tutorial\_author | tutorial\_count |

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

| mahran | 20 |

| mahnaz | NULL |

| Jen | NULL |

| Gill | 20 |

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

4 rows in set (0.00 sec)

mysql>

以下实例中你可以看到 = 和 != 运算符是不起作用的：

mysql> SELECT \* FROM tcount\_tbl WHERE tutorial\_count = NULL;

Empty set (0.00 sec)

mysql> SELECT \* FROM tcount\_tbl WHERE tutorial\_count != NULL;

Empty set (0.01 sec)

查找数据表中 tutorial\_count 列是否为 NULL，必须使用IS NULL和IS NOT NULL，如下实例：

mysql> SELECT \* FROM tcount\_tbl

-> WHERE tutorial\_count IS NULL;

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

| tutorial\_author | tutorial\_count |

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

| mahnaz | NULL |

| Jen | NULL |

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

2 rows in set (0.00 sec)

mysql> SELECT \* from tcount\_tbl

-> WHERE tutorial\_count IS NOT NULL;

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

| tutorial\_author | tutorial\_count |

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

| mahran | 20 |

| Gill | 20 |

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

2 rows in set (0.00 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$tutorial\_count变量，然后使用该变量与数据表中的 tutorial\_count 字段进行比较：

<?php

$dbhost = 'localhost:3036';

$dbuser = 'root';

$dbpass = 'rootpassword';

$conn = mysql\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysql\_error());

}

if( isset($tutorial\_count ))

{

$sql = 'SELECT tutorial\_author, tutorial\_count

FROM tcount\_tbl

WHERE tutorial\_count = $tutorial\_count';

}

else

{

$sql = 'SELECT tutorial\_author, tutorial\_count

FROM tcount\_tbl

WHERE tutorial\_count IS $tutorial\_count';

}

mysql\_select\_db('TUTORIALS');

$retval = mysql\_query( $sql, $conn );

if(! $retval )

{

die('Could not get data: ' . mysql\_error());

}

while($row = mysql\_fetch\_array($retval, MYSQL\_ASSOC))

{

echo "Author:{$row['tutorial\_author']} <br> ".

"Count: {$row['tutorial\_count']} <br> ".

"--------------------------------<br>";

}

echo "Fetched data successfully\n";

mysql\_close($conn);

?>

# 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 RegExp 对象的 Multiline 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RegExp 对象的 Multiline 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， '[^abc]' 可以匹配 "plain" 中的'p'。 |
| p1|p2|p3 | 匹配 p1 或 p2 或 p3。例如，'z|food' 能匹配 "z" 或 "food"。'(z|f)ood' 则匹配 "zood" 或 "food"。 |
| \* | 匹配前面的子表达式零次或多次。例如，zo\* 能匹配 "z" 以及 "zoo"。\* 等价于{0,}。 |
| + | 匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。 |
| {n} | n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。 |
| {n,m} | m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。 |

### 实例

了解以上的正则需求后，我们就可以更加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且以'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）： Atomicity（原子性）、Consistency（稳定性）、Isolation（隔离性）、Durability（可靠性）

* 1、**事务的原子性：**一组事务，要么成功；要么撤回。
* 2、**稳定性** ： 有非法数据（外键约束之类），事务撤回。
* 3、**隔离性：**事务独立运行。一个事务处理后的结果，影响了其他事务，那么其他事务会撤回。事务的100%隔离，需要牺牲速度。
* 4、**可靠性：**软、硬件崩溃后，InnoDB数据表驱动会利用日志文件重构修改。可靠性和高速度不可兼得， innodb\_flush\_log\_at\_trx\_commit选项 决定什么时候吧事务保存到日志里。

### 在Mysql控制台使用事务来操作

1，开始一个事务

start transaction

2, 做保存点

save point 保存点名称

3, 操作

4，可以回滚，可以提交，没有问题，就提交，有问题就回滚。

### PHP中使用事务实例

<?php

$handler=mysql\_connect("localhost","root","password");

mysql\_select\_db("task");

mysql\_query("SET AUTOCOMMIT=0");//设置为不自动提交，因为MYSQL默认立即执行

mysql\_query("BEGIN");//开始事务定义

if(!mysql\_query("insert into trans (id) values('2')"))

{

mysql\_query("ROOLBACK");//判断当执行失败时回滚

}

if(!mysql\_query("insert into trans (id) values('4')"))

{

mysql\_query("ROOLBACK");//判断执行失败回滚

}

mysql\_query("COMMIT");//执行事务

mysql\_close($handler);

?>

# 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 TUTORIALS;

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 子句来想数据表中添加列，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，然后指定新字段的类型及名称。尝试如下实例：

mysql> ALTER TABLE testalter\_tbl CHANGE i j BIGINT;

<="" p="" style="color: rgb(0, 0, 0); font-family: 'Microsoft Yahei', 'Helvetica Neue', Helvetica, Arial, sans-serif; font-size: 12px; font-style: normal; font-variant: normal; font-weight: normal; letter-spacing: normal; line-height: normal; orphans: auto; text-align: start; text-indent: 0px; text-transform: none; white-space: normal; widows: 1; word-spacing: 0px; -webkit-text-stroke-width: 0px; background-color: rgb(255, 255, 255);">

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只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 TYPE = 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。

# 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

ALTER mytable ADD INDEX [indexName] ON (username(length))

### 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 mytable ADD UNIQUE [indexName] ON (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;

删除指定时只需指定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

........

# 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，那没当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 'TUTORIALS.SalesSummary' doesn't exist

# 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 tutorials\_tbl 。

**步骤一：**

获取数据表的完整结构。

mysql> SHOW CREATE TABLE tutorials\_tbl \G;

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

Table: tutorials\_tbl

Create Table: CREATE TABLE `tutorials\_tbl` (

`tutorial\_id` int(11) NOT NULL auto\_increment,

`tutorial\_title` varchar(100) NOT NULL default '',

`tutorial\_author` varchar(40) NOT NULL default '',

`submission\_date` date default NULL,

PRIMARY KEY (`tutorial\_id`),

UNIQUE KEY `AUTHOR\_INDEX` (`tutorial\_author`)

) TYPE=MyISAM

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` (

-> `tutorial\_id` int(11) NOT NULL auto\_increment,

-> `tutorial\_title` varchar(100) NOT NULL default '',

-> `tutorial\_author` varchar(40) NOT NULL default '',

-> `submission\_date` date default NULL,

-> PRIMARY KEY (`tutorial\_id`),

-> UNIQUE KEY `AUTHOR\_INDEX` (`tutorial\_author`)

-> ) TYPE=MyISAM;

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 (tutorial\_id,

-> tutorial\_title,

-> tutorial\_author,

-> submission\_date)

-> SELECT tutorial\_id,tutorial\_title,

-> tutorial\_author,submission\_date

-> FROM tutorials\_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 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 rows were affected\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 rows were affected\n", (defined ($count) ? $count : 0);

### PHP 实例

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

$result\_id = mysql\_query ($query, $conn\_id);

# 如果查询失败返回

$count = ($result\_id ? mysql\_affected\_rows ($conn\_id) : 0);

print ("$count rows were affected\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 "Table Name $table\n";

}

### PHP 实例

<?php

$con = mysql\_connect("localhost", "userid", "password");

if (!$con)

{

die('Could not connect: ' . mysql\_error());

}

$db\_list = mysql\_list\_dbs($con);

while ($db = mysql\_fetch\_object($db\_list))

{

echo $db->Database . "<br />";

}

mysql\_close($con);

?>

## 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 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 = 100,

-> PRIMARY KEY (id),

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

-> date DATE NOT NULL, # date collected

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

);

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;

# 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。

本章节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。

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 into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

-> ORDER BY last\_name;

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);

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);

# 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 = mysql\_query("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;";

mysql\_query("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 mysql\_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来防止SQL注入。

PHP的MySQL扩展提供了mysql\_real\_escape\_string()函数来转义特殊的输入字符。

if (get\_magic\_quotes\_gpc())

{

$name = stripslashes($name);

}

$name = mysql\_real\_escape\_string($name);

mysql\_query("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addcslashes()函数来处理以上情况，如下实例：

$sub = addcslashes(mysql\_real\_escape\_string("%something\_"), "%\_");

// $sub == \%something\\_

mysql\_query("SELECT \* FROM messages WHERE subject LIKE '{$sub}%'");

addcslashes() 函数在指定的字符前添加反斜杠。

语法格式:

addcslashes(string,characters)

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

具体应用可以查看：[PHP addcslashes() 函数](http://www.w3cschool.cc/php/func-string-addcslashes.html)

# 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 tutorials\_tbl 数据导出到 /tmp/tutorials.txt 文件中:

mysql> SELECT \* FROM tutorials\_tbl

-> INTO OUTFILE '/tmp/tutorials.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/tutorials.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.text'

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 tutorials\_tbl 导出到 /tmp 目录中：

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

--tab=/tmp TUTORIALS tutorials\_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 TUTORIALS tutorials\_tbl > dump.txt

password \*\*\*\*\*\*

以上命令创建的文件内容如下：

-- MySQL dump 8.23

--

-- Host: localhost Database: TUTORIALS

---------------------------------------------------------

-- Server version 3.23.58

--

-- Table structure for table `tutorials\_tbl`

--

CREATE TABLE tutorials\_tbl (

tutorial\_id int(11) NOT NULL auto\_increment,

tutorial\_title varchar(100) NOT NULL default '',

tutorial\_author varchar(40) NOT NULL default '',

submission\_date date default NULL,

PRIMARY KEY (tutorial\_id),

UNIQUE KEY AUTHOR\_INDEX (tutorial\_author)

) TYPE=MyISAM;

--

-- Dumping data for table `tutorials\_tbl`

--

INSERT INTO tutorials\_tbl

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

INSERT INTO tutorials\_tbl

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

INSERT INTO tutorials\_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 TUTORIALS > 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。

# MySQL 导入数据

MySQL中可以使用两种简单的方式来导入MySQL导出的数据。

## 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);

## 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，mysqlimport将强制继续插入数据 |
| -i or --ignore | mysqlimport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。 |
| --fields-terminated- by=char | 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） |
| --lines-terminated- by=str | 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。 您可以选择用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。