字段类型

1、int[(M)]
正常大小类型
2、double[(m,d)][zerofill]
正常大小（双精度）浮点数字类型
3、date 类型
日期类型。MySql 是以 YYY-MM-DD 格式来显示 date 值
4、char(m)
定义字符串类型，当存储时，总是用空格填满右边指定的长度
5、blob text
blob 或者 Text 类型，最大长度 65535 个字符
6、varchar

一些常用的基本指令

1、显示数据库
show databases;
mysql> SHOW DATABASES;
+--------------------+
| Database |
+--------------------+
| information_schema |
| mysql |
| students |
| users |
| webshop |
+--------------------+
5 rows in set (0.00 sec)
2、选择数据库和显示当前选择的数据库

选择数据库
mysql> USE USERS
Database changed

当前选择的数据库
select database();
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| users |
+------------+

1 row in set (0.00 sec)

3、当前数据库包含的表信息

SHOW TABLES;

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

4、获取表结构

```
mysql> desc userinfo;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(45) | NO   |     | NULL    |                |
| password | varchar(45) | NO   |     | NULL    |                |
| remark   | varchar(100)| NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.03 sec)
```

5、导入数据库表

source 命令

```
mysql> SELECT * FROM userinfo;
Empty set (0.00 sec)

mysql> SOURCE InsertDatasToUserInfo.sql
Query OK, 1 row affected (0.03 sec)
Query OK, 1 row affected (0.02 sec)
Query OK, 1 row affected (0.03 sec)
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM userinfo;
+----+---------+----------+--------+
| id | name    | password | remark |
+----+---------+----------+--------+
|  5 | Tom     | 12345    | good   |
|  6 | Johnson | 123456   | good   |
|  7 | Lucy    | 12345    | good   |
|  8 | MySql   | 12345    | good   |
+----+---------+----------+--------+
4 rows in set (0.00 sec)
```

其中 InsertDatasToUserInfo.sql 放在 C:\Program Files\MySQL\MySQL Server 5.0\bin 目录下，如果不在这个目录下则要指定路径

查看版本号

```
mysql> SELECT VERSION();
+---------------------+
| VERSION()           |
+---------------------+
| 5.0.22-community-nt |
+---------------------+
1 row in set (0.01 sec)
```

查看 mysql 服务器版本号和当前日期

```
mysql> SELECT VERSION(),CURRENT_DATE();
+---------------------+----------------+
| VERSION()           | CURRENT_DATE() |
+---------------------+----------------+
| 5.0.22-community-nt | 2012-05-14     |
+---------------------+----------------+
1 row in set (0.00 sec)
```

修改数据库

在表中增加字段

alter table dbname add column gender varchar(2) not null

这样就再 dbname 表中添加了一个字段 gender，类型为 varchar(2)

```
mysql> ALTER TABLE userinfo ADD COLUMN gender VARCHAR(2) NOT NULL;
Query OK, 4 rows affected (0.25 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

新建数据库

新建一个名为 example 的数据库

```
mysql> CREATE DATABASE example;
Query OK, 1 row affected (0.00 sec)
mysql> USE example;
Database changed
```

新建一个表

在名为 example 的数据库中建立一个 myTable 的表

```
mysql> CREATE TABLE myTable(
    -> name CHAR(30),
    -> phone CHAR(11)
    -> );
```

```
Query OK, 0 rows affected (0.11 sec)
```

因为现在的电话大部分都是 12 位的，例如“010- 82618800”

```
mysql> ALTER TABLE myTable MODIFY COLUMN phone CHAR(12);
Query OK, 0 rows affected (0.17 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

添加数据

```
mysql> INSERT INTO myTable VALUES("Homer Simpson","010-82618800");
Query OK, 1 row affected (0.03 sec)
mysql> INSERT INTO myTable VALUES("Bart Simpson","029-82312054");
Query OK, 1 row affected (0.05 sec)
mysql> INSERT INTO myTable VALUES("Lisa Simpson","028-82582134");
Query OK, 1 row affected (0.03 sec)
mysql> INSERT INTO myTable VALUES("Marge Simpson","022-85241234");
Query OK, 1 row affected (0.03 sec)
mysql> INSERT INTO myTable VALUES("Maggie Simpson","0316-7922655");
Query OK, 1 row affected (0.03 sec)
```

查看所有信息

```
mysql> SELECT * FROM myTable;
+----------------+--------------+
| name           | phone        |
+----------------+--------------+
| Homer Simpson  | 010-82618800 |
| Bart Simpson   | 029-82312054 |
| Lisa Simpson   | 028-82582134 |
| Marge Simpson  | 022-85241234 |
| Maggie Simpson | 0316-7922655 |
+----------------+--------------+
5 rows in set (0.00 sec)
```

数据库种常见的问题

DATE 值的格式是’YYYY-MM-DD’。按照标准的 SQL，不允许其他格式。在 UPDATE 表达式以及 SELECT 语句的 WHERE 子句中应使用该格式，例如：

```
MySQL> SELECT * FROM tb_name WHERE date >= ’2003-05-05’;
```

为了方便，如果日期是在数值环境下使用的，MySQL 会自动将日期转换为数值（反之亦然）。它还具有相当的智能，在更新时或在与 TIMESTAMP、DATE 或 DATETIME 列比较日期的 WHERE 子句中，允许“宽松的”字符串形式（“宽松形式”表示，任何标点字符均能用作各部 分之间的分隔符。例如，

'2004-08-15'和'2004#08#15'是等同的)。MySQL 还能转换不含任何分隔符的字符串(如 '20040815'),前体是它必须是有意义的日期。

使用<、<=、=、>=、>、或 BETWEEN 操作符将 DATE、TIME、DATETIME 或 TIMESTAMP 与常量 字符串进行比较时,MySQL 通常会将字符串转换为内部长整数,以便进行快速比较(以及略为“宽松”的字符串检查)。但是,该转换具有下述例外:

比较两列时将 DATE、TIME、DATETIME 或 TIMESTAMP 列与表达式进行比较时使用其他比较方法时,如 IN 或 STRCMP()。对于这些例外情形,会将对象转换为字符串并执行字符串比较,采用该方式进行比较。为了保持安全,假定按字符串比较字符串,如果你打算比较临时值和字符串,将使用恰当的字符串函数。对于特殊日期'0000-00-00',能够以'0000-00-00'形式保存和检索。在 MyODBC 中使用'0000-00-00'日期时,对于 MyODBC 2.50.12 或更高版本,该日期将被自动转换为 NULL,这是因为 ODBC 不能处理这类日期。

由于 MySQL 能够执行前面所介绍的转换,下述语句均能正常工作:

```
mysql> INSERT INTO tb_name (idate) VALUES (19970505);
mysql> INSERT INTO tb_name (idate) VALUES ('19970505');
mysql> INSERT INTO tb_name (idate) VALUES ('97-05-05');
mysql> INSERT INTO tb_name (idate) VALUES ('1997.05.05');
mysql> INSERT INTO tb_name (idate) VALUES ('1997 05 05');
mysql> INSERT INTO tb_name (idate) VALUES ('0000-00-00');
mysql> SELECT idate FROM tb_name WHERE idate >= '1997-05-05';
mysql> SELECT idate FROM tb_name WHERE idate >= 19970505;
mysql> SELECT MOD(idate,100) FROM tb_name WHERE idate >= 19970505;
mysql> SELECT idate FROM tb_name WHERE idate >= '19970505';
```

但是,下述语句不能正常工作:

```
mysql> SELECT idate FROM tbl_name WHERE STRCMP(idate,'20030505')=0;
```

STRCMP()是一种字符串函数,它能将 idate 转换为'YYYY-MM-DD'格式的字符串,并执行字符串比较。它不能将'20030505'转换为日期'2003-05-05'并进行日期比较。

如果你正在使用 ALLOW_INVALID_DATES SQL 模式,MySQL 允许以仅执行给定的有限检查方式保存日期:MySQL 仅保证天位于 1～31 的范围内,月位于 1～12 的范围内。

这样就使得 MySQL 很适合于 Web 应用程序,其中,你能获得三个不同字段中的年、月、日值,也能准确保存用户插入的值(无日期验证)。

如果未使用 NO_ZERO_IN_DATE SQL 模式,“天”和“月”部分可能为 0。如果你打算将生日保存在 DATE 列而且仅知道部分日期,它十分方便。

如果未使用 NO_ZERO_DATE SQL 模式,MySQL 也允许你将'0000-00-00'保存为“伪日期”。在某些情况下,它比使用 NULL 值更方便。

如果无法将日期转换为任何合理值，“0”将保存在 DATE 列中，并被检索为'0000-00-00'。这是兼顾速度和便利性的事宜。我们认为，数据 库服务器的职责是检索与你保存的日期相同的日期（即使在任何情况下，数据在逻辑上不正确也同样）。我们认为，对日期的检查应由应用程序而不是服务器负责。如果你希望 MySQL 检查所有日期并仅接受合法日期（除非由 IGNORE 覆盖），应将 sql_mode 设置为 "NO_ZERO_IN_DATE,NO_ZERO_DATE"。

如何从一个文本文件运行 SQL 命令

一般地，mysql 客户被交互性地使用，象这样： shell> mysql database

然而，也可以把你的 SQL 命令放在一个文件中并且告诉 mysql 从该文件读取其输入。要想这样做，创造一个文本文件“text_file”，它包含你想要执行的命令。然后如下那样调用 mysql：

shell> mysql database < text_file

你也能启动有一个 USE db_name 语句的文本文件。在这种情况下，在命令行上指定数据库名是不必要的： shell> mysql < text_file

对于 SQL 的新手，NULL 值的概念常常会造成混淆，他们常认为 NULL 是与空字符串''相同的事。情况并非如此。例如，下述语句是完全不同的：

```
mysql> INSERT INTO my_table (phone) VALUES (NULL);
mysql> INSERT INTO my_table (phone) VALUES ('');
```

这两条语句均会将值插入 phone（电话）列，但第 1 条语句插入的是 NULL 值，第 2 条语句插入的是空字符串。第 1 种情况的含义可被解释为“电话号码未知”，而第 2 种情况的含义可被解释为“该人员没有电话，因此没有电话号码”。

为了进行 NULL 处理，可使用 IS NULL 和 IS NOT NULL 操作符以及 IFNULL()函数。

在 SQL 中，NULL 值与任何其它值的比较（即使是 NULL）永远不会为“真”。包含 NULL 的表达式总是会导出 NULL 值，除非在关于操作符的文档中以及表达式的函数中作了其他规定。下述示例中的所有列均返回 NULL：

```
mysql> SELECT NULL, 1+NULL, CONCAT('Invisible',NULL);
```

如果打算搜索列值为 NULL 的列，不能使用 expr = NULL 测试。下述语句不返回任何行，这是因为，对于任何表达式，expr = NULL 永远不为“真”：

```
mysql> SELECT * FROM my_table WHERE phone = NULL;
```

要想查找 NULL 值，必须使用 IS NULL 测试。在下面的语句中，介绍了查找 NULL 电话号码和空电话号码的方式：

```
mysql> SELECT * FROM my_table WHERE phone IS NULL;
mysql> SELECT * FROM my_table WHERE phone = '';
```

更多信息和示例：

如果你正在使用 MyISAM、InnoDB、BDB、或 MEMORY 存储引擎，能够在可能具有 NULL 值的列上增加 1 条索引。如不然，必须声明索引列为 NOT NULL，而且不能将 NULL 插入到列中。

用 LOAD DATA INFILE 读取数据时，对于空的或丢失的列，将用''更新它们。如果希望在列中具有 NULL 值，应在数据文件中使用\N。在某些情况下，也可以使用文字性单词“NULL”。

使用 DISTINCT、GROUP BY 或 ORDER BY 时，所有 NULL 值将被视为等同的。

使用 ORDER BY 时，首先将显示 NULL 值，如果指定了 DESC 按降序排列，NULL 值将最后显示。

对于聚合（累计）函数，如 COUNT()、MIN()和 SUM()，将忽略 NULL 值。对此的例外是 COUNT(*)，它将计数行而不是单独的列值。例如，下述语句产生两个计数。首先计数表中的行数，其次计数 age 列中的非 NULL 值数目：

```
mysql> SELECT COUNT(*), COUNT(age) FROM person;
```

对于某些列类型，MySQL 将对 NULL 值进行特殊处理。如果将 NULL 插入 TIMESTAMP 列，将插入当前日期和时间。如果将 NULL 插入具有 AUTO_INCREMENT 属性的整数列，将插入序列中的下一个编号。