<http://www.sdau.edu.cn/support/mysq_doc/manual_Clients.html>

MYSQL

这个结构表示对一个数据库连接的句柄，它被用于几乎所有的**MySQL** 函数。

MYSQL\_RES

这个结构代表返回行的一个查询的(SELECT , SHOW , DESCRIBE , EXPLAIN )的结果。从查询返回的信息在本章下文称为*结果集合* 。

MYSQL\_ROW

这是一个行数据的类型安全(type-safe)的表示。当前它实现为一个计数字节的字符串数组。（如果字段值可能包含二进制数据，你不能将这些视为空终止串，因为这样的值可以在内部包含空字节) 行通过调用mysql\_fetch\_row() 获得。

MYSQL\_FIELD

这个结构包含字段信息，例如字段名、类型和大小。其成员在下面更详细地描述。你可以通过重复调用mysql\_fetch\_field() 对每一列获得MYSQL\_FIELD 结构。字段值不是这个结构的部分；他们被包含在一个MYSQL\_ROW 结构中。

MYSQL\_FIELD\_OFFSET

这是一个相对一个**MySQL** 字段表的偏移量的类型安全的表示。（由mysql\_field\_seek() 使用。) 偏移量是在一行以内的字段编号，从0开始。

my\_ulonglong

该类型用于行编号和mysql\_affected\_rows() 、mysql\_num\_rows() 和mysql\_insert\_id() 。这种类型提供0到1.84e19 的一个范围。在一些系统上，试图打印类型my\_ulonglong 的值将不工作。为了打印出这样的值，将它变换到unsigned long 并且使用一个%lu 打印格式。例如：

printf (Number of rows: %lu/n", (unsigned long) mysql\_num\_rows(result));

MYSQL\_FIELD 结构包含列在下面的成员：

char \* name

字段名，是一个空结尾的字符串。

char \* table

包含该字段的表的名字，如果它不是可计算的字段。对可计算的字段，table 值是一个空字符串。

char \* def

这字段的缺省值，是一个空结尾的字符串。只要你使用，只有你使用mysql\_list\_fields() 才可设置它。

enum enum\_field\_types type

字段类型。type 值可以是下列之一：

|  |  |
| --- | --- |
| **类型值** | **类型含义** |
| FIELD\_TYPE\_TINY | TINYINT 字段 |
| FIELD\_TYPE\_SHORT | SMALLINT 字段 |
| FIELD\_TYPE\_LONG | INTEGER 字段 |
| FIELD\_TYPE\_INT24 | MEDIUMINT 字段 |
| FIELD\_TYPE\_LONGLONG | BIGINT 字段 |
| FIELD\_TYPE\_DECIMAL | DECIMAL 或NUMERIC 字段 |
| FIELD\_TYPE\_FLOAT | FLOAT 字段 |
| FIELD\_TYPE\_DOUBLE | DOUBLE 或REAL 字段 |
| FIELD\_TYPE\_TIMESTAMP | TIMESTAMP 字段 |
| FIELD\_TYPE\_DATE | DATE 字段 |
| FIELD\_TYPE\_TIME | TIME 字段 |
| FIELD\_TYPE\_DATETIME | DATETIME 字段 |
| FIELD\_TYPE\_YEAR | YEAR 字段 |
| FIELD\_TYPE\_STRING | 字符串(CHAR 或VARCHAR )字段 |
| FIELD\_TYPE\_BLOB | BLOB 或TEXT 字段(使用max\_length 决定最大长度） |
| FIELD\_TYPE\_SET | SET 字段 |
| FIELD\_TYPE\_ENUM | ENUM 字段 |
| FIELD\_TYPE\_NULL | NULL - 类型字段 |
| FIELD\_TYPE\_CHAR | 不推荐；使用FIELD\_TYPE\_TINY 代替 |

你可以使用IS\_NUM() 宏来测试字段是否有一种数字类型。将type 值传给IS\_NUM() 并且如果字段是数字的，它将计算为TRUE：

if (IS\_NUM(field->type))

printf("Field is numeric/n");

unsigned int length

字段宽度，在表定义中指定。

unsigned int max\_length

对结果集合的字段的最大宽度(对实际在结果集合中的行的最长字段值的长度)。如果你使用mysql\_store\_result() 或mysql\_list\_fields() ，这包含字段最大长度。如果你使用mysql\_use\_result()，这个变量的值是零。

unsigned int flags

字段的不同位标志。flags 值可以是零个或多个下列位设置：

|  |  |
| --- | --- |
| **标志值** | **标志含义** |
| NOT\_NULL\_FLAG | 字段不能是NULL |
| PRI\_KEY\_FLAG | 字段是一个主键的一部分 |
| UNIQUE\_KEY\_FLAG | 字段是一个唯一键的一部分 |
| MULTIPLE\_KEY\_FLAG | 字段是一个非唯一键的一部分。 |
| UNSIGNED\_FLAG | 字段有UNSIGNED 属性 |
| ZEROFILL\_FLAG | 字段有ZEROFILL 属性 |
| BINARY\_FLAG | 字段有BINARY 属性 |
| AUTO\_INCREMENT\_FLAG | 字段有AUTO\_INCREMENT 属性 |
| ENUM\_FLAG | 字段是一个ENUM （不推荐） |
| BLOB\_FLAG | 字段是一个BLOB 或TEXT （不推荐） |
| TIMESTAMP\_FLAG | 字段是一个TIMESTAMP （不推荐） |

BLOB\_FLAG 、ENUM\_FLAG 和TIMESTAMP\_FLAG 标志的使用是不推荐的，因为他们指出字段的类型而非它的类型属性。对FIELD\_TYPE\_BLOB 、FIELD\_TYPE\_ENUM 或FIELD\_TYPE\_TIMESTAMP ，最好是测试field->type 。下面例子演示了一个典型的flags 值用法：

if (field->flags & NOT\_NULL\_FLAG)

printf("Field can't be null/n");

你可以使用下列方便的宏决来确定flags 值的布尔状态：

|  |  |
| --- | --- |
| IS\_NOT\_NULL(flags) | 真，如果该字段被定义为NOT NULL |
| IS\_PRI\_KEY(flags) | 真，如果该字段是一个主键 |
| IS\_BLOB(flags) | 真，如果该字段是一个BLOB 或TEXT （不推荐；相反测试field->type ） |

unsigned int decimals

对数字字段的小数位数。

**[20.4.14 mysql\_fetch\_field ()](http://blog.csdn.net/xiangyu5945/article/details/manual_toc.html" \l "mysql_fetch_field)**

MYSQL\_FIELD \*mysql\_fetch\_field (MYSQL\_RES \*result)

**20.4.14.1 说明**

返回作为一个MYSQL\_FIELD 结构的一个结果集合的一个列的定义。重复调用这个函数在结果集合中检索所有关于列的信息。当没有剩下更多的字段时，mysql\_fetch\_field () 返回NULL 。

在每次你执行一个新的SELECT 查询，mysql\_fetch\_field () 被重置（reset)以返回有关第一列的信息。由mysql\_fetch\_field () 返回的字段也受调用mysql\_field\_seek() 的影响。

如果你调用mysql\_query() 在一张表上执行一个SELECT ，但是没调用mysql\_store\_result() ，如果你调用mysql\_fetch\_field () 询问一个BLOB 字段的长度，[**MySQL**](http://lib.csdn.net/base/14) 返回缺省BLOB长度(8K字节)。（选择8K的长度是因为**MySQL** 不知道BLOB 的最大长度。这应该在某个时候是它可配置) 一旦你已经检索了结果集合，field->max\_length 包含了在特定查询中对于该列最大值的长度。

**20.4.14.2 返回值**

当前列的MYSQL\_FIELD 结构。如果没有列剩下，NULL 。

**20.4.14.3 错误**

无。

**20.4.14.4 范例**

MYSQL\_FIELD \*field;

while((field = mysql\_fetch\_field

(result)))

{

printf("field name %s/n", field->name);

}

**[20.4.15 mysql\_fetch\_fields()](http://blog.csdn.net/xiangyu5945/article/details/manual_toc.html" \l "mysql_fetch_fields)**

MYSQL\_FIELD \*mysql\_fetch\_fields(MYSQL\_RES \*result)

**20.4.15.1 说明**

返回一个结果集合的所有MYSQL\_FIELD 结构的数组。每个结构提供结果结合中一列的字段定义。

**20.4.15.2 返回值**

一个结果集合的所有MYSQL\_FIELD 结构的一个数组。

**20.4.15.3 错误**

无。

**20.4.15.4 范例**

unsigned int num\_fields;

unsigned int i;

MYSQL\_FIELD \*fields;

num\_fields = mysql\_num\_fields(result);

fields = mysql\_fetch\_fields(result);

for(i = 0; i < num\_fields; i++)

{

printf("Field %u is %s/n", i, fields[i].name);

}