Mysql C API提供函数主要分为如下五种：

1. 控制类函数

2. 信息获取类函数

3. 行列类操作函数

4. 线程类操作函数

5. 出错处理类函数

本文归纳了C API可使用的函数，具体用法，参考文末地址中对应条目，均有详细说明。

函数

描述

mysql\_affected\_rows()

返回上次UPDATE、DELETE或INSERT查询更改／删除／插入的行数。

mysql\_autocommit()

切换 autocommit模式，ON/OFF

mysql\_change\_user()

更改打开连接上的用户和数据库。

mysql\_charset\_name()

返回用于连接的默认字符集的名称。

mysql\_close()

关闭服务器连接。

mysql\_commit()

提交事务。

mysql\_connect()

连接到MySQL服务器。该函数已不再被重视，使用mysql\_real\_connect()取代。

mysql\_create\_db()

创建数据库。该函数已不再被重视，使用SQL语句CREATE DATABASE取而代之。

mysql\_data\_seek()

在查询结果集中查找属性行编号。

mysql\_debug()

用给定的字符串执行DBUG\_PUSH。

mysql\_drop\_db()

撤销数据库。该函数已不再被重视，使用SQL语句DROP DATABASE取而代之。

mysql\_dump\_debug\_info()

让服务器将调试信息写入日志。

mysql\_eof()

确定是否读取了结果集的最后一行。该函数已不再被重视，可以使用mysql\_errno()或mysql\_error() 取而代之。

mysql\_errno()

返回上次调用的MySQL函数的错误编号。

mysql\_error()

返回上次调用的MySQL函数的错误消息。

mysql\_escape\_string()

为了用在SQL语句中，对特殊字符进行转义处理。

mysql\_fetch\_field()

返回下一个表字段的类型。

mysql\_fetch\_field\_direct()

给定字段编号，返回表字段的类型。

mysql\_fetch\_fields()

返回所有字段结构的数组。

mysql\_fetch\_lengths()

返回当前行中所有列的长度。

mysql\_fetch\_row()

从结果集中获取下一行

mysql\_field\_seek()

将列光标置于指定的列。

mysql\_field\_count()

返回上次执行语句的结果列的数目。

mysql\_field\_tell()

返回上次mysql\_fetch\_field()所使用字段光标的位置。

mysql\_free\_result()

释放结果集使用的内存。

mysql\_get\_client\_info()

以字符串形式返回客户端版本信息。

mysql\_get\_client\_version()

以整数形式返回客户端版本信息。

mysql\_get\_host\_info()

返回描述连接的字符串。

mysql\_get\_server\_version()

以整数形式返回服务器的版本号。

mysql\_get\_proto\_info()

返回连接所使用的协议版本。

mysql\_get\_server\_info()

返回服务器的版本号。

mysql\_info()

返回关于最近所执行查询的信息。

mysql\_init()

获取或初始化MYSQL结构。

mysql\_insert\_id()

返回上一个查询为AUTO\_INCREMENT列生成的ID。

mysql\_kill()

杀死给定的线程。

mysql\_library\_end()

最终确定MySQL C API库。

mysql\_library\_init ()

初始化MySQL C API库。

mysql\_list\_dbs()

返回与简单正则表达式匹配的数据库名称。

mysql\_list\_fields()

返回与简单正则表达式匹配的字段名称。

mysql\_list\_processes()

返回当前服务器线程的列表。

mysql\_list\_tables()

返回与简单正则表达式匹配的表名。

mysql\_more\_results ()

检查是否还存在其他结果。

mysql\_next\_result()

在多语句执行过程中返回/初始化下一个结果。

mysql\_num\_fields()

返回结果集中的列数。

mysql\_num\_rows()

返回结果集中的行数。

mysql\_options()

为mysql\_connect()设置连接选项。

mysql\_ping()

检查与服务器的连接是否工作，如有必要重新连接。

mysql\_query()

执行指定为“以Null终结的字符串”的SQL查询。

mysql\_real\_connect ()

连接到MySQL服务器。

mysql\_real\_escape\_string()

考虑到连接的当前字符集，为了在SQL语句中使用，对字符串中的特殊字符进行转义处理。

mysql\_real\_query()

执行指定为计数字符串的SQL查询。

mysql\_refresh()

刷新或复位表和高速缓冲。

mysql\_reload()

通知服务器再次加载授权表。

mysql\_rollback()

回滚事务。

mysql\_row\_seek()

使用从mysql\_row\_tell()返回的值，查找结果集中的行偏移。

mysql\_row\_tell()

返回行光标位置。

mysql\_select\_db()

选择数据库。

mysql\_server\_end()

最终确定嵌入式服务器库。

mysql\_server\_init()

初始化嵌入式服务器库。

mysql\_set\_server\_option()

为连接设置选项（如多语句）。

mysql\_sqlstate()

返回关于上一个错误的SQLSTATE错误代码。

mysql\_shutdown()

关闭数据库服务器。

mysql\_stat()

以字符串形式返回服务器状态。

mysql\_store\_result ()

检索完整的结果集至客户端。

mysql\_thread\_id()

返回当前线程ID。

mysql\_thread\_safe()

如果客户端已编译为线程安全的，返回1。

mysql\_use\_result()

初始化逐行的结果集检索。

mysql\_warning\_count()

返回上一个SQL语句的告警数。

与MySQL交互时，应用程序应使用该一般性原则：

1.    通过调用mysql\_library\_init()，初始化MySQL库。库可以是mysqlclient C客户端库，或mysqld嵌入式服务器库，具体情况取决于应用程序是否与“-libmysqlclient”或“-libmysqld”标志链接。

2.    通过调用mysql\_init()初始化连接处理程序，并通过调用mysql\_real\_connect()连接到服务器。

3.    发出SQL语句并处理其结果。（在下面的讨论中，详细介绍了使用它的方法）。

4.    通过调用mysql\_close()，关闭与MySQL服务器的连接。

5.    通过调用mysql\_library\_end()，结束MySQL库的使用。

调用mysql\_library\_init()和mysql\_library\_end()的目的在于，为MySQL库提供恰当的初始化和结束处理。对于与客户端库链接的应用程序，它们提供了改进的内存管理功能。如果不调用mysql\_library\_end()，内存块仍将保持分配状态（这不会增加应用程序使用的内存量，但某些内存泄漏检测器将抗议它）。对于与嵌入式服务器链接的应用程序，这些调用会启动并停止服务器。

mysql\_library\_init()和mysql\_library\_end()实际上是#define符号，这类符号使得它们等效于mysql\_server\_init()和mysql\_server\_end()，但其名称更清楚地指明，无论应用程序使用的是mysqlclient或mysqld库，启动或结束MySQL库时，应调用它们。对于早期的MySQL版本，可调用mysql\_server\_init()和mysql\_server\_end()取而代之。

如果愿意，可省略对mysql\_library\_init()的调用，这是因为，必要时，mysql\_init()会自动调用它。

要想连接到服务器，可调用mysql\_init()来初始化连接处理程序，然后用该处理程序（以及其他信息，如主机名、用户名和密码）调用mysql\_real\_connect()。建立连接后，在低于5.0.3版的API中，mysql\_real\_connect()会将再连接标志（MYSQL结构的一部分）设置为1，或在较新的版本中，将其设置为0。对于该标志，值“1”指明，如果因连接丢失而无法执行语句，放弃之前，会尝试再次连接到服务器。从MySQL 5.0.13开始，可以在mysql\_options()上使用MYSQL\_OPT\_RECONNECT选项，以控制再连接行为。完成连接后，调用mysql\_close()中止它。

当连接处于活动状态时，客户端或许会使用mysql\_query()或mysql\_real\_query()向服务器发出SQL查询。两者的差别在于，mysql\_query()预期的查询为指定的、由Null终结的字符串，而mysql\_real\_query()预期的是计数字符串。如果字符串包含二进制数据（其中可能包含Null字节），就必须使用mysql\_real\_query()。

对于每个非SELECT查询（例如INSERT、UPDATE、DELETE），通过调用mysql\_affected\_rows()，可发现有多少行已被改变（影响）。

对于SELECT查询，能够检索作为结果集的行。注意，某些语句因其返回行，类似与SELECT。包括SHOW、DESCRIBE和EXPLAIN。应按照对待SELECT语句的方式处理它们。

客户端处理结果集的方式有两种。

一种方式是，通过调用mysql\_store\_result()，一次性地检索整个结果集。该函数能从服务器获得查询返回的所有行，并将它们保存在客户端。

第二种方式是针对客户端的，通过调用mysql\_use\_result()，对“按行”结果集检索进行初始化处理。该函数能初始化检索结果，但不能从服务器获得任何实际行。

在这两种情况下，均能通过调用mysql\_fetch\_row()访问行。通过mysql\_store\_result()，mysql\_fetch\_row()能够访问以前从服务器获得的行。通过mysql\_use\_result()，mysql\_fetch\_row()能够实际地检索来自服务器的行。通过调用mysql\_fetch\_lengths()，能获得关于各行中数据大小的信息。

完成结果集操作后，请调用mysql\_free\_result()释放结果集使用的内存。

这两种检索机制是互补的。客户端程序应选择最能满足其要求的方法。实际上，客户端最常使用的是mysql\_store\_result()。

mysql\_store\_result()的1个优点在于，由于将行全部提取到了客户端上，你不仅能连续访问行，还能使用mysql\_data\_seek()或mysql\_row\_seek()在结果集中向前或向后移动，以更改结果集内当前行的位置。通过调用mysql\_num\_rows()，还能发现有多少行。另一方面，对于大的结果集，mysql\_store\_result()所需的内存可能会很大，你很可能遇到内存溢出状况。

mysql\_use\_result()的1个优点在于，客户端所需的用于结果集的内存较少，原因在于，一次它仅维护一行（由于分配开销较低，mysql\_use\_result()能更快）。它的缺点在于，你必须快速处理每一行以避免妨碍服务器，你不能随机访问结果集中的行（只能连续访问行），你不知道结果集中有多少行，直至全部检索了它们为止。不仅如此，即使在检索过程中你判定已找到所寻找的信息，也必须检索所有的行。

通过API，客户端能够恰当地对查询作出响应（仅在必要时检索行），而无需知道查询是否是SELECT查询。可以在每次mysql\_query()或mysql\_real\_query()后，通过调用mysql\_store\_result()完成该操作。如果结果集调用成功，查询为SELECT，而且能够读取行。如果结果集调用失败，可调用mysql\_field\_count()来判断结果是否的确是所预期的。如果mysql\_field\_count()返回0，查询不返回数据（表明它是INSERT、UPDATE、DELETE等），而且不返回行。如果mysql\_field\_count()是非0值，查询应返回行，但没有返回行。这表明查询是失败了的SELECT。关于如何实现该操作的示例，请参见关于mysql\_field\_count()的介绍。

无论是mysql\_store\_result()还是mysql\_use\_result()，均允许你获取关于构成结果集的字段的信息（字段数目，它们的名称和类型等）。通过重复调用mysql\_fetch\_field()，可以按顺序访问行内的字段信息，或者，通过调用mysql\_fetch\_field\_direct()，能够在行内按字段编号访问字段信息。通过调用mysql\_field\_seek()，可以改变当前字段的光标位置。对字段光标的设置将影响后续的mysql\_fetch\_field()调用。此外，你也能通过调用mysql\_fetch\_fields()，一次性地获得关于字段的所有信息。

为了检测和通报错误，MySQL提供了使用mysql\_errno()和mysql\_error()函数访问错误信息的机制。它们能返回关于最近调用的函数的错误代码或错误消息，最近调用的函数可能成功也可能失败，这样，你就能判断错误是在何时出现的，以及错误是什么。