附录 H　PHP API 参考

本附录描述了 MySQL 的 PHP 应用程　序编程接口（API）。API 由与 MySQL 服务器通信和访问数据库的一组函数组成。

由于这个附录是参考性的，因此它只包含说明 PHP API 应用的简短代码段。要想获得它们的全部客户机脚本和编写注意事项，请参阅第 8 章。这里介绍的函数是完全适合 MySQL 的函数。PHP手册（现在大约有800页）包括了 600 多页参考材料，因此很明显，这个附录说明的内容还不到 PHP 性能概要的一小部分。要获得完整的 PHP 手册，请访问下面的 PHP Web 网站：

```
http://www.php.net/
```

H.1　编写 PHP 脚本

PHP 脚本是可以包括 HTML 和 PHP 代码混合的无格式文本文件。对脚本的解释生成一个 Web 页面，作为发送到客户机的输出。HTML 不用解释就可拷贝到输出中。PHP 代码需要进行解释，并用该代码所生成的输出（可能没有输出）来替换。

PHP 开始在 HTML 模式下解释文件。可以使用表明 PHP 代码开始和结束的特殊标记进行 PHP 代码模式的转换。在一个文件中可以在此两种模式之间进行多次切换。PHP 支持四种类型的标记，虽然其中一些标记在使用它们时需要明确启用。一种方法就是在 PHP 初始化文件 php3.ini 中将它们打开。这个文件的位置由系统决定，通常位于 /usr/local/lib 中。

PHP 支持下面的标记风格：

■ 缺省的标记风格，用‘ <?php ’和‘ ?> ’标记：

```
<?php echo ("Some PHP code here"); ?>
```

■ 简短打开的标记风格，用‘ <? ’和‘ ?> ’标记：

```
<? echo ("Some PHP code here"); ?>
```

这种标记风格由 PHP 初始文件的一个命令来启用：

```
short_open_tag = On;
```

■ 与活动服务器页面兼容的风格，用‘ <% ’和‘ %> ’标记：

```
<% echo ("Some PHP code here"); %>
```

ASP风格的标记由 PHP 初始文件的一个指令来启用：

```
asp_tags = On;
```

对 ASP 标记的支持是在 PHP 3.0.4 中引入的。

■ 如果使用的是不能支持其他标记的 HTML 编辑器，可以使用 <SCRIPT> 和 </SCRIPT> 标记：

```
<SCRIPT LANGUAGE="php"> echo ("Some PHP code here"); </SCRIPT>
```

H.2　函数

下面的描述讨论了与 MySQL 相关的 PHP 的每个函数。在方括号（[]）中指出可选参数。

许多函数用一个可选的 link_id 参数（连接标识符）表示与 MySQL 服务器的连接。如果函数调用中没有连接标识符，则使用最近打开的连接。如果没有指定连接，并且也没有打开的连接，有的函数会设法建立连接。

参数 result_id 指出了一个结果集标识符，一般是由 mysql_db_query( ) 或者 mysql_query( ) 返回的。

除了返回状态值之外，如果出现错误，一些函数还会产生错误消息。在 Web 环境下，这个消息出现在发送到客户机浏览器的输出中。一般来说，在函数名称的前面加一个字符‘ @’取消了（可能是隐含的）函数产生的错误消息。例如，要想从 mysql_pconnect() 调用中取消错误消息，以便可以以更合适的方式报告失败，则可以如下进行：

```
<?php
    $link = @mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
?>
```

另一种方法是利用函数 error_reporting() 来取消错误消息：

```
<?php
    error_reporting (0); # suppress all error messages
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
?>
```

本附录中的许多实例脚本在建立了与 MySQL 服务器的连接之后，都将打印“ Connected successfully ”。它们这样做的原因是要确认脚本打印了一些输出，以免要自己去试这个脚本。没有产生输出的 PHP 脚本在一些浏览器中引发了“ 页面不包括数据 ”的警告。

在下面的描述中，术语“ SELECT 查询 ”用来指 SELECT 查询或者其他返回行的查询，如 DESCRIBE、EXPLAIN 或者 SHOW。

H.2.1 连接管理例程

本节的例程允许打开和关闭对 MySQL 服务器的连接。

■ int

```
mysql_close ([int link_id]);
```

关闭由 link_id 标识的与 MySQL 服务器的连接。如果没有指定连接，则 mysql_close() 关闭最近打开的连接。如果成功，则 mysql_close() 返回真，失败则返回假。对由 mysql_pconnect() 打开的永久连接，mysql_close() 忽略相应的关闭请求，只是返回真。如果要关闭一个连接，就应该用 mysql_connect() 而不是 mysql_pconnect() 来打开它。

```
<?php
    $link = mysql_connect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    mysql_close ($link);
?>
```

■int

```
mysql_connect ([string host_name] [, string user_name [, string
password]]);
```

在主机 host_name 上对给定口令的用户 user_name打开与 MySQL 服务器的连接。连接成功，则返回与新连接相关的连接标识符（正数），如果出现错误，则返回假。

自 PHP 3.0B4 起，主机名称参数就由可选的端口号以“ *host_name: port_num* ”形式指定。自 PHP 3.0.10 起，如果主机名称是 localhost，则主机名称参数就可以由可选路径以“ localhost:*socket_name* ”形式指定 UNIX 领域的套接字路径。应该将套接字指定为完整的路径名称。

如果没有主机名称参数，则缺省值为 localhost。如果没有用户名称，或者用户名称为空，则缺省值为 PHP 运行的用户名称（这是 Web 正在处理的用户名，好像 PHP 按 Apache 模块运行一样，或者是 PHP 正在处理的用户名，好像 PHP 作为一个独立程序运行一样）。如果没有口令参数或者口令参数为空，就发送空口令。

在打开连接时，如果用同样参数（主机名称、用户名称和口令）调用 mysql_connect()，不会产生新的连接，mysql_connect() 将返回已有的连接标识符。

这个连接可以通过调用 mysql_close() 来关闭。在脚本终止时，如果连接是打开的，则自动关闭该连接。

```
<?php
    $link = mysql_connect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    mysql_close ($link);
?>
```

■ int

```
mysql_pconnect ([string host_name] [, string user_name [, string
password]]);
```

mysql_pconnect() 除了打开一个永久的连接之外，与 mysql_connect() 类似，即，在脚本终止时，这个连接仍然是打开的。当连接打开时，如果用同样的连接参数（主机名称、用户名称和口令）调用 mysql_pconnect(),将再次使用连接。这就避免了删除和再次打开连接的开销，并且比非永久的连接效率更高。

永久连接只在 PHP 作为在脚本终止之后继续运行的 Web 服务器内部的模块执行时有意义。在独立的 PHP 版本执行的脚本中，当脚本终止时连接关闭，因为 PHP 处理也终止了。

在永久的连接上调用 mysql_close()是无意义的，在此情形下，mysql_close() 返回真,但是连接仍然是打开的。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
?>
```

H.2.2 状态报告和错误报告例程

函数 mysql_errno() 和 mysql_error() 返回与 MySQL 有关的 PHP 函数的错误号或错误信息。然而，不会从无有效连接标识符的任何函数得到错误信息。这表示它们对报告失败的 mysql_errno() 和 mysql_error() 调用的结果没有用处，因为在成功建立连接之前，不能得到连

接标识符。如果想获得连接失败的 MySQL 错误消息，可用 PHP 初始化文件的一条命令来启用变量 track_errors：

```
track_errors = On;
```

然后，如果 PHP 按 Apache 模块运行，则重新启动 Web 服务器。完成之后，可通过访问变量 $php_errormsg 获得连接失败的错误字符串：

```
<?php
    $link = @mysql_connect("badhost","baduser", "badpass")
        or die ("Could not connect: " . $php_errormsg);
    print ("Connected successfully");
?>
```

■ int

```
mysql_errno ([int link_id]);
```

对于给定的连接，返回最近返回状态的与 MySQL 相关的函数错误号。零值表示未出现错误。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("query failed, error code = " . mysql_errno ());
?>
```

■ string

```
mysql_error ([int link_id]);
```

对于给定的连接，返回含有最近返回状态的与 MySQL 相关的函数的错误消息字符串。空值意味着未出现错误。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("query failed, error message = " . mysql_error ());
?>
```

H.2.3 查询的构造与执行例程

本节的例程用来发布到 MySQL 服务器的查询。

■ int

```
mysql_db_query (string db_name, string query [, int link_id]);
```

mysql_db_query( ) 除了提取一个额外的数据库名称参数，并在执行查询之前使它成为缺省的数据库之外，与 mysql_query( ) 类似（可将下面的实例与 mysql_query( ) 的实例做一下比较）。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
```

```
    print ("Connected successfully");
    $query = "SELECT * FROM president";
    $result = mysql_db_query ("samp_db", $query)
        or die ("Query failed");
?>
```

■ int

```
mysql_list_dbs ([int link_id]);
```

返回结果标识符，标识给定的连接中这个服务器知道的由数据库名称所组成的结果集，结果集中的每行都是一个数据库名。如果出现错误，则返回假。不必选择缺省的数据库。结果集可由任何通用的提取行的函数或者由 mysql_tablename( ) 来处理。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    $result = mysql_list_dbs ()
        or die ("Query failed");
    print ("Databases (using mysql_fetch_row()):<BR>\n");
    while ($row = mysql_fetch_row ($result))
        printf ("%s<BR>\n", $row[0]);
    $result = mysql_list_dbs ()
        or die ("Query failed");
    print ("Databases (using mysql_tablename()):<BR>\n");
    for ($i = 0; $i < mysql_num_rows ($result); $i++)
        printf ("%s<BR>\n", mysql_tablename ($result, $i));
?>
```

■ int

```
mysql_list_fields (string db_name, string tbl_name [, int link_id]);
```

对于包括表中有关列的信息的结果集，返回一个结果标识符，如果出现错误，则返回假。不必选择缺省的数据库。db_name 和 tbl_name参数指定了感兴趣的数据库和表。结果标识符用于函数 mysql_field_flags( )、mysql_field_len( )、mysql_field_name( ) 和 mysql_field_type()。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    $result = mysql_list_fields ("samp_db", "member")
        or die ("Query failed");
    print ("member table column information:<BR>\n");
    for ($i = 0; $i < mysql_num_fields ($result); $i++)
    {
        printf ("column %d:", $i);
        printf (" name %s,\n", mysql_field_name ($result, $i));
        printf (" len %d,\n", mysql_field_len ($result, $i));
        printf (" type %s,\n", mysql_field_type ($result, $i));
        printf (" flags %s\n", mysql_field_flags ($result, $i));
        print "<BR>\n";
    }
?>
```

■ int

```
mysql_list_tables (string db_name [, int link_id]);
```

在给定的数据库名称中，对于由表的名称组成的结果集，返回一个结果标识符，结果集的每行都是一个表名称。如果出现错误，则返回假。不必选择缺省的数据库。结果集可以由任何通用的提取行的函数或者由 mysql_tablename( ) 来处理。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    $result = mysql_list_tables ("samp_db")
        or die ("Query failed");
    print ("samp_db tables (using mysql_fetch_row()):<BR>\n");
    while ($row = mysql_fetch_row ($result))
        printf ("%s<BR>\n", $row[0]);
    $result = mysql_list_tables ("samp_db")
        or die ("Query failed");
    print ("samp_db tables (using mysql_tablename()):<BR>\n");
    for ($i = 0; $i < mysql_num_rows ($result); $i++)
        printf ("%s<BR>\n", mysql_tablename ($result, $i));
?>
```

■ int

```
mysql_query (string query [, int link_id]);
```

在给定的连接上，将查询字符串发送给 MySQL 服务器。对于 DELETE、INSERT、REPLACE 和 UDATE 语句，如果 mysql_query( ) 成功，则返回真；如果出现错误，则返回假。对于成功的查询，可以调用 mysql_affected_rows( ) 找出所修改的行数。

对 SELECT 语句，如果成功，则 mysql_query( ) 返回一个正的结果集标识符，如果出现错误，则返回假。对于成功的查询，结果集标识符可用于提取 result_id 参数的各种结果集处理函数。可以将这个标识符传递给 mysql_free_result( ) 以释放与结果集相关的任何资源。

一个“成功的”查询是无错误执行的查询，但是成功并不意味着查询会返回一些行。下面的查询是非常合法的，只是不返回行：

```
SELECT * FROM president WHERE 1 = 0
```

查询失败可能有许多原因。例如，可能在语句构成上是畸形的、语法是非法的或者不符合规定的，因为您不具有查询中指定表的访问权。

如果没有指定连接标识符，就使用最近打开的连接。如果不存在当前连接，则 mysql_query( ) 试图打开一个连接，就像无参数地调用 mysql_connect( ) 一样。如果连接失败，则 mysql_query( ) 失败。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
?>
```

H.2.4 结果集处理例程

本节的例程用来检索查询的结果。它们也提供了有关结果信息的访问，如涉及多少行的

信息，或者结果集列的元数据信息等。

■ int

```
mysql_affected_rows ([int link_id]);
```

在给定的连接中，返回由最近的 DELETE、INSERT、REPLACE 或者 UPDATE语句所作用的行数。如果没有行被修改，则 mysql_affected_rows() 返回 0，如果出现错误，则返回 -1。

在SELECT查询之后，mysql_affected_rows() 返回所选择的行数。但一般是与 SELECT 语句一道使用 mysql_num_rows()。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "INSERT INTO member (last_name,first_name,expiration)"
        . " VALUES('Brown','Marcia','2002-6-3')";
    $result = mysql_query ($query)
        or die ("Query failed");
    printf ("%d row%s inserted\n",
            mysql_affected_rows (),
            mysql_affected_rows () == 1 ? "" : "s");
?>
```

■ int

```
mysql_data_seek (int result_id, int row_num);.
```

由 SELECT 查询返回的每个结果集都有一个行游标，指示下一个提取行的函数（mysql_fetch_array( )、mysql_fetch_object( ) 或者 mysql_fetch_row( )）调用将返回哪一行。mysql_data_seek( ) 将给定结果集的指针设置到给定的行。行号的范围为0到mysql_num_rows( ) - 1。如果行号合法，则mysql_data_seek( ) 返回真，否则返回假。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT last_name, first_name FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    # fetch rows in reverse order
    for ($i = mysql_num_rows ($result) - 1; $i >=0; $i--)
    {
        if (!mysql_data_seek ($result, $i))
        {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }
        if(!($row = mysql_fetch_object ($result)))
            continue;
        printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
    }
    mysql_free_result ($result);
?>
```

■ array

```
mysql_fetch_array (int result_id [, int result_type]);
```

作为一个数组返回给定结果集的下一行。如果没有更多的行，则返回假。数组含有按数值列索引和按列名称关联的键索引所存储的值。换句话说，每个列值都可以用它的数值列索引或者它的名称来访问。关联索引是区分大小写的，并且给出时必须与查询中使用的列名称的大小写相同。假设发布下面的查询：

```
SELECT last_name, first_name FROM president
```

如果从结果集中获取行，放到一个名为 $row 的数组中，其数组元素可以这样访问：

$row[0]	保存 last_name 值
$row[1]	保存 first_name 值
$row[“last_name”]	保存 last_name 值
$row[“first_name”]	保存 first_name 值

键不受相应列的表名的限制，因此如果选择了不同表中具有相同名称的列，结果会发生名称冲突。在查询选择的列的列表中，最后指定的列优先。要访问隐藏的列，应该使用数值索引，或者编写查询，为该列提供一个别名。

result_type 参数可以为 MYSQL_ASSOC（只返回名称索引值）、MYSQL_NUM（只返回数字索引值）或者MYSQL_BOTH（返回两种类型索引值）。如果不给出 result_type，其缺省值为 MYSQL_BOTH。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT last_name, first_name FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    while ($row = mysql_fetch_array ($result))
    {
        # print each name twice, once using numeric indices,
        # once using associative (name) indices
        printf ("%s %s<BR>\n", $row[0], $row[1]);
        printf ("%s %s<BR>\n", $row["last_name"], $row["first_name"]);
    }
    mysql_free_result ($result);
?>
```

■ object

```
mysql_fetch_field (int result_id [, int col_num]);
```

返回结果集中给定列的相关元数据信息，如果没有这样的列，则返回假。如果省略 col_num，则对 mysql_fetch_field( ) 的后继调用返回结果集后续列的信息。如果不再有剩余的列，则返回值为假。如果指定了 col_num，则其取值范围为 0 到 mysql_num_fields( ) - 1。在此情形下，mysql_num_fields( ) 返回给定列的相关信息，如果 col_num 超出范围，返回假。

信息作为具有表 H-1 中所示属性的对象返回。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
```

```
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    # get column metadata
    for ($i = 0; $i < mysql_num_fields ($result); $i++)
    {
        printf ("Information for column %d:<BR>\n", $i);
        $meta = mysql_fetch_field ($result);
        if (!$meta)
        {
            print ("No information available<BR>\n");
            continue;
        }
        print ("<PRE>\n");
        printf ("blob:         %s\n", $meta->blob);
        printf ("max_length:   %s\n", $meta->max_length);
        printf ("multiple_key: %s\n", $meta->multiple_key);
        printf ("name:         %s\n", $meta->name);
        printf ("not_null:     %s\n", $meta->not_null);
        printf ("numeric:      %s\n", $meta->numeric);
        printf ("primary_key:  %s\n", $meta->primary_key);
        printf ("table:        %s\n", $meta->table);
        printf ("type:         %s\n", $meta->type);
        printf ("unique_key:   %s\n", $meta->unique_key);
        printf ("unsigned:     %s\n", $meta->unsigned);
        printf ("zerofill:     %s\n", $meta->zerofill);
        print ("</PRE>\n");
    }
?>
```

表H-1 mysql_fetch_field( ) 的属性

属 性	说 明
blob	如果列为 BLOB 类型则值为1，否则为0
max_length	结果集中最大列值的长度
multiple_key	如果列为非唯一的索引的成分则值为 1，否则为0
name	列名称
not_null	如果列不能包含 NULL 值则为1，否则为0
numeric	如果列为数值类型则为 1，否则为0
primary_key	如果列为 PRIMARY KEY 的成分则为1，否则为0
table	包含列的表的名称（计算列为空）
type	列的类型的名称
unique_key	如果列为 UNIQUE 索引的成分则为1，否则为0
unsigned	如果列具有 UNSIGNED 属性则为1，否则为0
zerofill	如果列具有 ZEROFILL 属性则值为1，否则为0

■ array

```
mysql_fetch_lengths (int result_id);
```

返回一个数组，此数组含有函数 mysql_fetch_array()、mysql_fetch_object() 或 mysql_fetch_row( ) 最近提取的行中的列值长度。如果没有提取行或者出现错误，则返回假。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
```

```
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    $row_num = 0;
    while (mysql_fetch_row ($result))
    {
        ++$row_num;
        # get lengths of column values
        printf ("Lengths of values in row %d:<BR>\n", $row_num);
        $len = mysql_fetch_lengths ($result);
        if (!$len)
        {
            print ("No information available<BR>\n");
            break;
        }
        print ("<PRE>\n");
       for ($i = 0; $i < mysql_num_fields ($result); $i++)
           printf ("column %d: %s\n", $i, $len[$i]);
       print ("</PRE>\n");
   }
?>
```

■object

mysql_fetch_object (int result_id [, int result_type]);

作为一个对象返回给定结果集的下一行，如果没有更多的行，则返回假。列值可以作为对象的属性访问。这个属性的名称就是在生成结果集的查询中所选择的列的名称。

result_type 参数可能是 MYSQL_ASSOC（只返回名称索引值）、MYSQL_NUM （只返回数字索引值）或者 MYSQL_BOTH（返回两种类型索引值）。如果省略 result_type，则缺省值为 MYSQL_BOTH（假如数值不是合法的属性名，我不清楚指定 MYSQL_NUM 的作用是什么）。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT last_name, first_name FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    while ($row = mysql_fetch_object ($result))
        printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
    mysql_free_result ($result);
?>
```

■array

mysql_fetch_row (int result_id);

作为一个数组返回给定结果集的下一行，如果没有更多的行，则返回假。

列值可作为数组元素访问，在 0到 mysql_num_fields( ) - 1 范围内使用列索引。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
```

```
        or die ("Could not select database");
    $query = "SELECT last_name, first_name FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    while ($row = mysql_fetch_row ($result))
        printf ("%s %s<BR>\n", $row[0], $row[1]);
    mysql_free_result ($result);
?>
```

■string

mysql_field_name (int result_id, int col_num);.

返回结果集的给定列的名称。

col_num 的范围为0到 mysql_num_fields()—1.

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    # get column names
    for ($i = 0; $i < mysql_num_fields ($result); $i++)
    {
        printf ("Name of column %d: ", $i);
        $name = mysql_field_name ($result, $i);
        if (!$name)
            print ("No name available<BR>\n");
        else
            print ("$name<BR>\n");
    }
?>
```

■ int

mysql_field_seek (int result_id, int col_num);

为随后的 mysql_fetch_field ( ) 调用设置索引。发布没有明确列号的 mysql_fetch_field( ) 的下一次调用，将返回列 col_num 的信息。如果搜索成功，返回真，否则返回假。

col_num 的范围为 0 到 mysql_num_fields( ) - 1。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    # get column metadata
    for ($i = 0; $i < mysql_num_fields ($result); $i++)
    {
    printf ("Information for column %d:<BR>\n", $i);
    if (!mysql_field_seek ($result, $i))
    {
        print ("Cannot seek to column<BR>\n");
```

```
            continue;
        }
        $meta = mysql_fetch_field ($result, $i);
        if (!$meta)
        {
            print ("No information available<BR>\n");
            continue;
        }
        print ("<PRE>\n");
        printf ("blob:         %s\n", $meta->blob);
        printf ("max_length:   %s\n", $meta->max_length);
        printf ("multiple_key: %s\n", $meta->multiple_key);
        printf ("name:         %s\n", $meta->name);
        printf ("not_null:     %s\n", $meta->not_null);
        printf ("numeric:      %s\n", $meta->numeric);
        printf ("primary_key:  %s\n", $meta->primary_key);
        printf ("table:        %s\n", $meta->table);
        printf ("type:         %s\n", $meta->type);
        printf ("unique_key:   %s\n", $meta->unique_key);
        printf ("unsigned:     %s\n", $meta->unsigned);
        printf ("zerofill:     %s\n", $meta->zerofill);
        print ("</PRE>\n");
    }
?>
```

■ string

mysql_field_table (int result_id, int col_num);

返回含有结果集给定列的表名。对于计算列，此名为空。

col_num 的范围为 0 到 mysql_num_fields( ) - 1。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    for ($i = 0; $i < mysql_num_fields ($result); $i++)
    {
        printf ("column %d:", $i);
        printf (" name %s,\n", mysql_field_name ($result, $i));
        printf (" table %s\n", mysql_field_table ($result, $i));
        print "<BR>\n";
    }
?>
```

■ string

mysql_field_type (int result_id, int col_num);

返回结果集给定列的类型名。类型名在附录 B 中列出。

col_num 的范围为 0 到 mysql_num_fields( ) - 1。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db");
    $query = "SELECT * FROM president";
```

```
    $result = mysql_query ($query)
        or die ("Query failed");
    for ($i = 0; $i < mysql_num_fields ($result); $i++)
    {
        printf ("column %d:", $i);
        printf (" name %s,\n", mysql_field_name ($result, $i));
        printf (" type %s\n", mysql_field_type ($result, $i));
        print "<BR>\n";
    }
?>
```

■ string

mysql_field_flags (int result_id, int col_num);

作为字符串返回结果集中给定列的相关元数据信息，如果出现错误，则返回假。这个字符串由以空格分开的词组成，说明哪个列的标记值为真。对于假的标记，在字符串中给出相应的词。表 H-2 列出了可能在字符串中给出的词。

col_num 的范围为0到 mysql_num_fields( ) - 1。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db");
    $query = "SELECT * FROM member";
    $result = mysql_query ($query)
        or die ("Query failed");
    for ($i = 0; $i < mysql_num_fields ($result); $i++)
    {
        printf ("column %d:", $i);
        printf (" name %s,\n", mysql_field_name ($result, $i));
        printf (" flags %s\n", mysql_field_flags ($result, $i));
        print "<BR>\n";
    }
?>
```

表H-2 mysql_field_flags( ) 的值

属 性	说 明
auto_increment	列具有AUTO_INCREMENT属性
binary	列具有BINARY属性
blob	列为 BLOB 类型
enum	列为 ENUM 类型
multiple_key	列为非惟一索引的成分
not_null	列不能包含 NULL 值
primary_key	列为 PRIMARY_KEY 的成分
timestamp	列为 TIMESTAMP 类型
unique_key	列 UNIQUE 索引的成分
unsigned	列具有 UNSIGNED 属性
zerofill	列具有 ZEROFILL 属性

■ int

mysql_field_len (int result_id, int col_num);

返回结果集给定列中值可能的最大长度。

col_num 的范围为0到 mysql_num_fields( )-1。

```
col_num 的范围为0到 mysql_num_fields()—1.
   <?php
       $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
           or die ("Could not connect");
       mysql_select_db ("samp_db");
       $query = "SELECT * FROM member";
       $result = mysql_query ($query)
           or die ("Query failed");
       for ($i = 0; $i < mysql_num_fields ($result); $i++)
       {
           printf ("column %d:", $i);
           printf (" name %s,\n", mysql_field_name ($result, $i));
           printf (" len %d\n", mysql_field_len ($result, $i));
           print "<BR>\n";
       }
   ?>
```

■ int

```
mysql_free_result (int result_id);
```

释放与给定结果集相关的任何资源。在脚本终止时，会自动释放结果集，但是你也可以在生成许多结果集的脚本中显式地调用这个函数。例如，下面的脚本将占用相当数量的内存：

```
<?php
    $link = mysql_connect ("localhost", "paul", "secret");
    mysql_select_db ("samp_db");
    for ($i = 0; $i < 10000; $i++)
    {
        $result = mysql_query ("SELECT * from president");
    }
?>
```

在调用 mysql_query( ) 之后加入一个 mysql_free_result( ) 调用，可将结果集内存的使用量减少到几乎没有：

```
<?php
    $link = mysql_connect ("localhost", "paul", "secret");
    mysql_select_db ("samp_db");
    for ($i = 0; $i < 10000; $i++)
    {
        $result = mysql_query ("SELECT * from president");
        mysql_free_result ($result);
    }
?>
```

■ int

```
mysql_insert_id ([int link_id]);
```

在给定的连接中，返回由最近执行的查询生成的 AUTO_INCREMENT 值。在连接期间，如果不生成这样的值，则返回零。一般在期望产生新值的查询之后应该立即调用 mysql_insert_id ()；如果在那个查询和要使用这个值的地方之间有其他的查询干扰，则 mysql_insert_id () 的值将被干扰查询设置为 0。

请注意，mysql_insert_id () 的性能与 SQL 函数中 LAST_INSERT_ID() 的性能不同。mysql_insert_id () 在客户机中维护，并且每次查询都要设置，LAST_INSERT_ID() 在服务器

中维护，而且从一个查询到另一个查询都保持一个值。

由 mysql_insert_id () 返回的值是连接专用的，不受其他连接中 AUTO_INCREMENT 活动的影响。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "INSERT INTO member (last_name,first_name,expiration)"
        . " VALUES('Brown','Marcia','2002-6-3')";
    $result = mysql_query ($query)
        or die ("Query failed");
    printf ("membership number for new member: %d\n", mysql_insert_id());
?>
```

■ int

```
mysql_num_fields (int result_id);
```

返回给定结果集中的列数。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    printf ("Number of columns: %d\n", mysql_num_fields ($result));
?>
```

■ int

```
mysql_num_rows (int result_id);
```

返回给定结果集中的行数。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT * FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    printf ("Number of rows: %d\n", mysql_num_rows ($result));
?>
```

■ int

```
mysql_result (int result_id, int row, mixed field);
```

从给定结果集的行中返回一个值。列由参数 field 标识，可能是数字列索引，或者是在查询中指定的列名称索引。

这个函数速度慢，使用 mysql_fetch_array( )、mysql_fetch_object( ) 或者 mysql_fetch_row( ) 会更合适一些。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
```

```
    mysql_select_db ("samp_db")
        or die ("Could not select database");
    $query = "SELECT last_name, first_name FROM president";
    $result = mysql_query ($query)
        or die ("Query failed");
    for ($i = 0; $i < mysql_num_rows ($result); $i++)
    {
        for ($j = 0; $j < mysql_num_fields ($result); $j++)
        {
            if ($j > 0)
                print (" ");
            print (mysql_result ($result, $i, $j));
        }
        print "<BR>\n";
    }
    mysql_free_result ($result);
?>
```

■ string

mysql_tablename (int result_id, int row_num);

给定一个由 mysql_list_dbs( ) 或者 mysql_list_tables( ) 返回的结果集标识符和一个行索引 row_num，返回在结果集的给定列中存储的名称。

行索引的范围为0到 mysql_num_rows( )-1。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    $result = mysql_list_tables ("samp_db")
        or die ("Query failed");
    print ("samp_db tables:<BR>\n");
    for ($i = 0; $i < mysql_num_rows ($result); $i++)
        printf ("%s<BR>\n", mysql_tablename ($result, $i));
?>
```

H.2.5 数据库例程

本节的例程允许创建和删除数据库。它们也允许为当前的服务器会话选择一个缺省的数据库。

mysql_create_db (string db_name [, int link_id]);

■ int

告诉由 link_id 标识的 MySQL服务器用给定的名称来创建数据库。如果数据库创建成功，则返回真；如果出现错误，则返回假。必须在数据库有创建它的 CREATE 权限。

可能利用 mysql_query( ) 较利用 mysql_create_db( ) 发布 CREATE DATABASE 语句更为适合。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    if (mysql_create_db ("my_db"))
        print ("Database created successfully\n");
    else
        printf ("Error creating database: %s\n", mysql_error ());
?>
```

■int

```
mysql_drop_db (string db_name [, int link_id]);
```

告诉由 link_id 标识的 MySQL 服务器用给定的名称来删除数据库。如果数据库删除成功，则返回真；如果出现错误，则返回假。必须有对数据库进行删除的 DROP 权限。

要小心这个函数；如果删除数据库，它就不存在了，且不能恢复。

使用 mysql_query( ) 较使用 mysql_drop_db( ) 发布 DROP DATABASE 语句更为适合。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    if (mysql_drop_db ("my_db"))
        print ("Database dropped successfully\n");
    else
        printf ("Error dropping database: %s\n", mysql_error ());
?>
```

■ int

```
mysql_select_db (string db_name [, int link_id]);
```

选择给定的数据库使其对给定的连接为缺省数据库。成功时返回真，如果出现错误，则返回假。

如果没有指定连接，则使用当前的连接。如果当前没有连接，则 mysql_select_db( ) 试图打开一个连接，就像无参数地调用 mysql_connect( ) 一样。如果这种连接企图失败，则 mysql_select_db( ) 失败。

```
<?php
    $link = mysql_pconnect ("pit-viper.snake.net", "paul", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    mysql_select_db ("samp_db")
        or die ("Could not select database");
 ?>
```