附录G Perl DBI API 参考

本附录介绍了 Perl DBI 应用程序编程接口。API 由与数据库服务器通信及从 Perl 脚本访问数据库的一组方法和属性组成。本附录还描述了由 DBD::mysql 提供的针对 MySQL 进行的 DBI 扩充，即 MySQL 数据库驱动程序。

有一些DBI的方法和属性不在这里介绍，因为它们不用于 MySQL，或者因为它们还是新的，实验性的方法，而随着开发的进行，这些方法可能有变动，甚至被删除。一些 MySQL 特定的 DBD方法由于已经过时，所以也不在这里讨论。如果想了解有关新的或过时的方法的详细情况，请参阅 DBI 文档或 MySQL DBD 文档，运行下面的命令，我们可以获得这些文档：

```
% perldoc DBI
% perldoc DBI::FAQ
% perldoc DBD::mysql
```

本附录仅供参考，所以只含有说明 Perl DBI API 用法的简要代码段。完整的客户机脚本和编写它们时的注意事项，请参阅第 7章。

G.1 编写脚本

使用 DBI 模块的每个 Perl 脚本都必须包括下面的行：

```
use DBI;
```

特定的 DBD 级的模块不需要包括 use 行，因为在与服务器进行连接时， DBI 负责激活相应的模块。

一般来说， DBI 脚本使用 connect( ) 方法打开连接，发布查询，然后用 disconnect( ) 关闭连接。这些方法通常发布各种查询。具有代表性的非 SELECT 查询由 do( ) 方法实现。具有代表性的 SELECT 查询则将这个查询传递给 prepare( )，然后调用 execute( )，最后在重复调用提取行的方法的循环中，如在 fetchrow_array( ) 或 fetchrow_hashref( ) 中，检索查询结果，每次检查一行。

G.2 DBI 方法

这里描述的方法的编写格式与用于附录 F 中的C函数和附录 H 中的PHP函数的编写格式略有不同。那些附录中的函数以原型方式编写，明确地列出了返回值的类型和参数类型。一般来说， Perl 以同样的方式使用原型，所以这里的说明通过变量指定了返回值的类型和参数类型，变量的开始字符表明值的类型：‘ $ ’ 为标量，‘ @ ’ 为数组，‘ % ’ 为散列（关联的数组）。除此之外，列出的以 ‘ \ ’ 开始的任何参数，应该作为对变量的引用来传递，而不是传递变量本身。以 ‘ _ref ’ 结束的变量名称表示变量值为一个引用。

在这个附录中，有几个用于特殊含意的变量，如表 G-1 所示。

许多方法都接受散列参数 %attr，该参数包括影响相应方法的工作方式的属性。这个散列应该通过引用来传递。可以以两种方式来做这件事。一种方式是在调用此方法之前，设置散

列值 %attr 的内容，然后再将它传递给此方法。

```
my (%attr) = (AttrName1 => value1, AttrName2 => value2);
$ret_val = $h->method (..., \%attr);
```

另一种方式为在此方法调用中直接提供匿名的散列：

```
$ret_val = $h->method (..., {AttrName1 => value1, AttrName2 => value2});
```

表G-1 常用的 Perl DBI 变量名称

名 称	含 意
$drh	驱动程序对象的句柄
$dbh	数据库对象的句柄
$sth	语句（查询）对象的句柄
$fh	打开文件的句柄
$h	“通用”句柄；其含意有赖于上下文
$rc	从返回真或假的操作中返回的代码
$rv	从返回整数的操作中返回的值
$rows	从返回行计数的操作中返回的值
@ary	表示查询返回的一行值的数组（列表）

方括号（[]）表示可选的信息。其中使用方法或函数的方式由调用序列来表示。‘DBI->’表示 DBI 类方法，‘DBI::’表示 DBI 函数，‘$DBI::’表示 DBI 变量。对于使用句柄来调用的方法，相应的句柄名称表示方法的范围。‘$dbh->’表示数据库名称方法，‘$sth->’表示语句句柄方法，‘$h->’表示可以用各种不同类型的句柄调用的方法。下面是一个调用序列的实例。

```
@row_ary = $dbh->selectrow_array ($statement, [\%attr [, @bind_values]]);
```

这表示 selectrow_array( ) 方法是作为数据库句柄方法调用的，因为利用了‘$dbh->’来调用它。其中，参数为 $statement（标量值）、%attr（应该作为引用来传递的散列，因为以‘\’开头来表示）和 @bind_values（数组）。第二个和第三个参数为可选参数。返回值为数组。

这种方法的说明指出如果出现错误，返回什么样的值，而且仅当禁用 RaiseError 属性时才返回错误。如果启用 RaiseError，则此脚本将自动终止。

在后面的说明中，将用术语“SELECT 查询”来表示 SELECT 查询或返回行的任何其他查询，如DESCRIBE、EXPLAIN或SHOW。

G.2.1 DBI类方法

在本节中，方法的 %attr 参数可能用来指定方法处理的属性。就 MySQL 来说，这些属性中最重要的是 PrintError 和 RaiseError。例如，要想在出现DBI错误时，使自动脚本的作用终止，应该启用 RaiseError：

```
$dbh = DBI->connect ($data_source, $user_name, $password, {RaiseError => 1});
```

PrintError 和 RaiseError 在“通用句柄属性”一节中介绍。

■ @ary = DBI->**available_drivers** ([$quiet]);

返回可得到的 DBI 驱动程序的一个列表。如果发现多个驱动程序具有同样的名称，则发布一条警告。给 $quiet 参数传递值 1 可避免出现这条警告。

■ $dbh = DBI->**connect** ($data_source, $user_name, $password, [, \%attr]);

connect() 创建与数据库服务器的连接，并返回数据库句柄。如果连接失败，则该句柄为undef。要想终止成功创建的连接，用 conncet() 返回的数据库句柄调用 disconnect() 方法即可。

对于 connect()，在 @attr 参数中指定的任何属性都会在全局影响 DBI 的处理。

可以用几种格式给定数据源。但第一部分总是“ DBI:mysql ”，可以用任何大小写字母给出“ DBI ”，而驱动程序名称“ mysql ”必须为小写字母。在数据源的驱动程序名称部分后的所有字符都由该驱动程序解释，所以下面讨论中说明的语法除 DBD::mysql 外，不一定任何驱动程序都要用。也可以在数据源字符串的开头部分、驱动程序名之后，指定数据库名和主机名：

```
$data_source = "DBI:mysql:db_name";
$data_source = "DBI:mysql:db_name:host_name";
```

可以用*db_name*或database=*db_name*指定数据库名。可以用 *host_name* 或host=*host_name*指定主机名。

接着数据源字符串的初始部分，可以用 *attribute=value*的格式指定若干选项。每个选项的前面都必须有一个分号。MySQL驱动程序允许下面的选项：

■ host=*host_name* 要连接的主机。也可以使用host_name:port_num格式来指定端口号。

■ port=*port_num* 要连接的端口号。

■ mysql_compression=1 这个选项在客户机和 MySQL 服务器之间启用压缩通信。

mysql_compression 需要 MySQL 3.22.3 或更新的版本，以及 DBD::mysql 1.19.20 或更新的版本。

■ mysql_read_default_file=*file_name* 缺省设置时，DBI 脚本不读取连接参数的 MySQL 选项文件。此选项允许读取指定的选项文件。文件名称应该为完整的路径名（否则，将会解释为相对于在其中执行脚本的目录，这样会得到不一致的结果）。

为了只要用户运行该脚本就读取主目录中的 .my.cnf 文件，应该指定该文件名为 $ENV{HOME}/.my.cnf。如果期望多个用户使用同一脚本，并想让他们自己连接而不是作为此脚本中固定的一个用户来连接时，这样做很有用。

mysql_read_default_file 需要 MySQL 3.22.10 或更高的版本，以及 DBD::mysql 1.21.06 或更高的版本。

■ mysql_read_default_group=*group_name* 如果读取连接参数的选项文件，则缺省时读取 [client] 组。除了 [client] 组以外，mysql_read_default_group 选项还允许读取一个指定组。例如，mysql_read_default_group=dbi 指定 [dbi] 组应该像 [client] 组一样读取。出现在多个组中的参数的优先权在附录 E 中介绍。

mysql_read_default_group 需要 MySQL 3.22.10 或更高的版本，需要 DBD::mysql 1.21.06 或更高的版本。

■ mysql_socket=*socket_name* 如果缺省套接字路径不合适，则本选项指定了用于连接到 localhost 的 UNIX 域套接字的路径名。

mysql_socket 需要 MySQL 3.21.15 或更高的版本。DBI 考虑连接参数的若干环境变量：

■ 如果没有定义数据源或者数据源为空，则使用 DBI_DSN 变量的值。

- 如果该数据源中缺少驱动程序名称，则使用 DBI_DRIVER 变量的值。
- 如果未定义 connect() 调用的 user_name 或 password 参数，则使用 DBI_USER 和 DBI_PASS 变量的值。如果这些参数为空字符串，则不必使用。

就安全而言，使用 DBI_PASS 有些冒险，所以在安全性很重要的情况下，都不应该使用它（依靠像 ps 这样的命令，其他用户可以看到环境变量）。

如果查阅了所有的信息源之后，仍然不知道某些连接参数，则 DBI 使用缺省值。如果未指定主机名称，则其缺省值为 localhost。如果未指定用户名称，则其在 UNIX 下的缺省值为登录名称，Windows 下为 ODBC。如果不给出口令，则无缺省值，不发送口令。

- $drh = DBI->**install_driver** ($driver_name);

激活 DBD 级别的驱动程序，并返回驱动程序句柄。对于 MySQL，这个驱动程序名称为“ mysql ”(必须为小写字母)。一般来说，不一定要使用这种方法，因为在调用 connect() 方法时，DBI 会自动激活恰当的驱动程序。然而，如果使用 func() 方法来实现管理操作，应该可以使用 install_driver()（请参阅“ MySQL的特定管理方法 ”)。

G.2.2 数据库句柄方法

本节中的这些方法可通过数据库句柄进行调用，在通过调用 connect() 方法获得这样一个句柄后，可以使用这些方法。

本节所使用方法的 %attr 参数可用于特定的方法处理属性。对于 MySQL，这些属性中最重要的为 PrintError 和 RaiseError。例如，如果在像这样的特定查询的处理过程中，DBI 出现错误，可如下启用 RaiseError 使脚本自动终止：

```
$rows = $dbh->do ($statement, {RaiseError => 1});
```

在“ 通用句柄属性 ”一节中讨论 PrintError 和 RaiseError。

- $rc = $dbh->**disconnect**( );

终止与此句柄相关的连接。如果该脚本退出时，连接仍然有效，则显示一条警告，并自动终止此连接。

- $rows = $dbh->**do** ($statement [, \%attr [, @bind_values]]);

准备并运行 $statement 表示的查询。返回值为受影响的行数 —— 如果不知道行数，则返回 -1，如果出现错误，则返回 undef。如果受影响的行数为 0，则返回值为字符串“ 0E0 ”，作为数值它与 0 等价，但在判断时它为真。

不检索行的语句，如 DELETE、INSERT、REPLACE 或 UPDATE，主要使用 do()。如果对 SELECT 语句使用它，则不会获得返回的语句句柄，也不能提取任何行。

当语句包括占位符（在此查询字符串内部由‘ ? ’字母表示）时，使用 @bind_values。@bind_values 为给占位符赋值的值的列表。它必须和占位符有一样多的值。如果只指定赋值的值，但没有指定属性，则将 undef 作为 \%attr 参数的值来传递。

- $rc = $dbh->**ping**();

检查与服务器的连接是否仍然有效，并相应返回真或假。

- $sth = $dbh->**prepare** ($statement [, \%attr]);

为以后的执行所准备的由 $statement 表示的查询，并返回一个语句句柄。返回的句柄可用于 execute( )，以便执行该查询。

■ $str = $dbh->**quote** ($value [, $data_type]);

处理字符串以实现 SQL 语句中特定字符的引用和转义，以便在执行这条语句时，该字符串不引起语法错误。例如，“‘ I\ ’ m happy ’”（没有双引号）返回字符串“ I ’ m happy ”。如果 $value 为 undef，则它返回字符串“ NULL ”（没有引号）。

一般来说，$data_type 参数不是必需的，因为 MySQL 将查询中指定为字符串的值自动地转换为其他类型。可以将 $data_type 指定为特殊类型值的提示 —— 例如，DBI::SQL_INTEGER 指出 $value 表示一个整数。

不要使用具有打算利用占位符插入到查询中的值的 quote()。DBI会自动引用这样的值。

■ $ary_ref = $dbh->**selectall_arrayref** ($statement [, \%attr [,
@bind_values]]);

执行由 $statement 指定的查询，并结合 prepare()、execute() 和 fetchall_arrayref() 返回结果。如果出现错误，则返回 undef。

如果 $statement 参数是以前准备的语句，则省略 prepare() 步骤。

@bind_values 参数和 do() 方法中的该参数具有同样的意义。

■ @ary_ref = $dbh->**selectcol_arrayref** ($statement [, \%attr [,
@bind_values]]);

执行由 $statement 指定的查询，并通过组合 prepare() 和 execute() 返回结果的第一列。返回结果作为对含有每行第一列的数组的引用。如果出现错误，则返回 undef。

如果 $statement 参数是以前准备的语句，则省略 prepare() 步骤。

@bind_values 参数和 do() 方法中的该参数具有同样的意义。

■ @row_ary = $dbh->**selectrow_array** ($statement [, \%attr [,
@bind_values]]);

执行由 $statement 指定的查询，并结合 prepare()、execute() 和 fetchall_arrayref() 返回结果的第一行。

如果参数 $statement 是以前准备的语句，则省略 prepare() 步骤。

如果在列表的上下文中调用时，selectrow_array() 返回代表行值的数组，或者，如果出现错误，则返回空数组。在标量的上下文中，selectrow_array() 返回这个数组的第一个元素的值（行的第一列）。如果出现错误，则返回 undef。

@bind_values 参数和 do() 方法中的相应参数具有同样的意义。

G.2.3 语句句柄方法

本节中的这些方法通过语句句柄来调用，句柄可通过调用 prepare() 获得。

■ $rc = $sth->**bind_col** ($col_num, \$var_to_bind);

将 SELECT 查询的给定列与 Perl 变量相联系，将它作为引用传递。$col_num 的范围为 1 到查询选择的列数。每次提取行时，这个变量用列值自动更新。

bind_col() 应该在 execute() 之前及 prepare() 之后调用。

如果列号范围不在 1 到查询选择的列数之间，则 bind_col() 返回假。

■ $rc = $sth->**bind_columns** (\$var_to_bind1, \$var_to_bind2, ...);

将一系列变量与由准备好的 SELECT 语句返回的列相联系，请参阅 bind_col() 方法的说明。

如果引用的数量与查询选择的列数不匹配，则 bind_columns() 返回假。

■ $rv = $sth->**bind_param** ($n, $value [, \%attr]);

$rv = $sth->**bind_ param** ($n, $value [, $bind_type]);

在一个语句中，将值与占位符 ‘？’ 相联系。应该在 execute() 之前及 prepare() 之后调用它。

$n 指定了占位符的数量，应该限定 $value 值，而且该值范围应该为 1到占位符的数量。为了限定 NULL 值，可传递 undef。

参数 \%attr 或者 $bind_type 可作为要联系的值的类型提示。例如，要指定表示整数的值，可以用下面两种方式调用 bind_param()：

```
$rv = $sth->bind_param ($n, $value , { TYPE => DBI::SQL_INTEGER });
$rv = $sth->bind_param ($n, $value , DBI::SQL_INTEGER);
```

缺省值是将变量作为 VARCHAR 类型。这通常就足够了，因为 MySQL 将查询中字符串的值转换为所需的其他数据类型。

■ $rows = $sth->**dump_results** ([$maxlen [, $line_sep [, $field_sep [, $fh]]]]);

从语句句柄 $sth 中提取所有的行，通过调用实用函数 DBI::neat_list() 将他们格式化，并将他们打印到给定的文件句柄中。返回提取的行数。

$maxlen、$line_sep、$field_sep和$fh的缺省值分别为35、“ \n ”、“ , ”和STDOUT。

■ $rv = $sth->**execute** ([@bind_values]);

执行准备好的语句。如果该语句执行成功，则返回真，如果发生错误，则返回 undef。

参数 @bind_values 与 do() 方法中的有相同的意义。

■ary_ref = $sth->**fetch**();

fetch() 是 fetchrow_arrayref() 的别名。

■$tbl_ary_ref = $sth->**fetchall_arrayref** ([$slice_array_ref]);

$tbl_ary_ref = $sth->**fetchall_arrayref** ([$slice_hash_ref]);

从语句句柄 $sth 中提取所有行，并返回数组的引用，这个数组包含提取的每行的一个引用。数组中每个引用的意义取决于所传递的参数。没有参数或者只有数组部分引用参数，则 $tbl_ary_ref 的每个元素都是包括结果集的一行值的数组引用。对于散列部分的引用参数，$tbl_ary_ref 的每个元素就是对包含结果集的一行值的散列引用。

■ @ary_ref = $sth->**fetchrow_array**()；

当在一个列表的范围中调用时， fetchrow_array() 返回包含结果集下一行列值的数组，如果不再有行或者发生错误，则 fetchrow_array() 返回一个空数组。在标量上下文中，fetchrow_array() 返回数组第一个元素的值（那就是说，行的第一列），如果不再有行或者发生错误，则 fetchrow_array() 返回 undef。

通过检查 $sth->err()，可以将结果集正常结束与出现错误区分开来。零值表明已经无错误地到达了结果集的末尾。

■ @ary_ref = $sth->**fetchrow_arrayref**()；

返回一个包括结果集的下一行列值的数组引用。如果不再有行或者发生错误，则返回 undef。

通过检查 $sth->err()，可以将结果集正常结束与出现错误区分开来。零值表明已经无错误地到达了结果集的末尾。

■$hash_ref = $sth->**fetchrow_hashref** ([$name]);

返回包括结果集的下一行列值的散列引用。如果不再有行或者发生错误，则返回 undef。散列是索引值是列名称，散列的元素是列值。

对于散列的关键值，指定变量 $name 说明使用的语句句柄属性。缺省值为“ NAME ”。这可能导致查询中的列名称不区分大小写的问题，但是散列键是区分大小写的。要强迫散列键为大写字母或者小写字母，可以指定“ NAME_lc ”或“ NAME_uc ”的 $name 值。

通过检查 $sth->err()，可以将结果集正常结束与出现错误区分开来。零值表明已经无错误地到达了结果集的末尾。

■ $rc = $sth->**finish**();

释放有关语句句柄的任何资源。通常不必显式地调用这个方法，但是如果只提取部分结果集，则调用 finish() 使 DBI 了解已经提取了数据。调用 finish() 可能使语句属性无效，最好在调用 execute() 之后立即访问它们。

■ $rv = $sth->**rows**();

返回与 $sth 相关的语句所作用的行数，如果发生错误，则返回 -1。使用这个方法主要用于不返回行的语句。对于SELECT语句，不能依赖 rows( ) 方法在提取行时统计行数。

G.2.4 通用句柄方法

本节中的这些方法不是专用于特定类型的句柄的。可用驱动程序、数据库或语句句柄来调用它们。

■ $h->**err**()

返回最近调用的驱动程序操作的数字错误代码。 0 表示没有错误。

■ $h->**errstr**()

返回最近调用的驱动程序操作的字符串错误消息。空字符串表示没有错误。

■ DBI->**trace** ($trace_level [, $trace_filename]);

$h->**trace** ($trace_level [, $trace_filename]);

设置跟踪级别。跟踪提供有关 DBI 操作的信息。跟踪级别的范围从 0（关闭）到9（最多信息）。通过作为 DBI 类方法或独立的句柄调用跟踪，跟踪可以启用脚本内部的所有 DBI 操作：

```
DBI->trace (2);          打开脚本跟踪
$sth->trace (2);         打开句柄跟踪
```

通过设置 DBI_TRACE 环境变量，也可以对运行的所有 DBI 脚本在全局级别启用跟踪。缺省时，跟踪输出到 STDERR.。提供的 $filename 参数可以直接将结果输出到不同的文件。将输出添加到这个文件的任何已有内容后面。

每个跟踪调用导致来自所有跟踪的句柄中的输出进入相同的文件。如果文件已命名，则所有跟踪就输出到那个文件。如果没有命名的文件，则所有跟踪输出到 STDERR。

■ DBI->**trace_msg** ($str [, $min_level])

$h->**trace_msg** ($str [, $min_level])

如果跟踪这个句柄或如果在 DBI 级启用跟踪，则编写这个跟踪输出的消息。如果启用 DBI 级的跟踪，则 trace_msg() 可以作为 DBI->trace_msg() 来调用，编写消息。只有在跟踪级

别至少为这个级别时，才可以提供 $min_level 参数来指定应该编写的消息。

G.2.5 MySQL 的特定管理方法

本节介绍 DBI 作为直接访问驱动程序的手段所供的 func() 函数方法。

```
$rc = $drh->func('createdb',
                 $db_name, $host_name, $user_name, $password, 'admin');
$rc = $drh->func('dropdb',
                 $db_name, $host_name, $user_name, $password, 'admin');
$rc = $drh->func('shutdown',
                 $host_name, $user_name, $password, 'admin');
$rc = $drh->func('reload',
                 $host_name, $user_name, $password, 'admin');

$rc = $dbh->func('createdb', $db_name, 'admin');
$rc = $dbh->func('dropdb', $db_name, 'admin');
$rc = $dbh->func('shutdown', 'admin');
$rc = $dbh->func('reload', 'admin');
```

通过驱动程序句柄或通过数据库句柄访问 func() 方法。驱动程序句柄与打开的连接无关，所以，如果以这种方式访问 func()，则必须提供允许这个方法创建连接的主机名称、用户名称和口令的参数。如果用数据库句柄访问 func()，则不需要那些参数。如果需要，可以像下面这样获得驱动程序句柄：

```
$drh = DBI->install_driver( " mysql " );      # ( " mysql " must be lowercase)
```

- createdb 创建由 $db_name 指定的数据库。要这样做，必须对该数据库拥有 CREAT 权限。
- dropdb 删除由 $db_name 指定的数据库。要这样做，必须对该数据库拥有 DROP 权限。当心，如果删除了一个数据库，则它将会消失，且再也不能恢复。
- shutdown 关闭服务器。必须具有 SHUTDOWN 权限。
- reload 告诉服务器重新加载授权表。如果直接使用 DELETE、INSERT 或 UPDATE 而不是使用 GRANT 或 REVOKE 来修改这个授权表的内容，则这是必需的。要使用 reload，必须具有 RELOAD 权限。

G.3 DBI 实用程序函数

这些函数如像 DBI::func_name() 而不是 DBI->func_name() 那样调用。

- @bool = DBI::**looks_like_numbe**r (@array);

产生一列值，并返回一个数组，列表的每个元素都为数组的一个成员。每个成员表示相应的参数是否为数字：如果是，则返回真，如果不是，则返回假，如果没有定义参数或参数为空，则返回 undef。

- $str = DBI::**neat** ($value [, $maxlen]);

返回含有 $value 参数的格式化表示的字符串。字符串被引用；数字则不被引用（但是，请注意，引用的数字被看作是字符串）。未定义的值作为 undef 报告，不可打印的字符作为 ‘ . ’ 报告。

$maxlen 参数控制结果的最大长度。如果结果比 $maxlen 长，则将 $maxlen 缩短——加

上4个字符和省略号。如果 $maxlen为0、undef、或省略，则缺省值为 $DBI::neat_maxlen (400)。

对查询结构不要使用 neat( )。如果需要实现引用或转义，应该使用 quote( )。

■ $str = DBI::**neat_list** (\@listref [, $maxlen [, $field_sep]]);

对第一个元素指向的列表的每个元素调用 neat( )，将它们与分隔符字符串 $field_sep 连接起来，并作为字符串返回结果。

$maxlen 参数用于单独的参数，而不是用于调用 neat( ) 的结果字符串。

如果省略 $field_sep，则缺省值为“，”。

G.4 DBI属性

DBI 在几个级别上提供了属性信息。大多数属性与数据库句柄相关，或者与语句句柄相关，但不是与两者都相关。一些属性，例如 PrintError 和 RaiseError，可能既与数据库句柄相关，又与语句句柄相关。一般来说，每个句柄都有自己的属性，但是一些保留错误信息的属性，如 err 和 errstr，与最近使用的句柄是动态相关的。

G.4.1 通用句柄属性

这些属性可以用于单独的句柄，或者对于取得影响该方法操作的参数的方法，在 %attr 参数中进行指定。如果用于 connect( ) 方法，则在整个脚本中将全局性地影响 DBI 的处理。

■ $h->{‘ **ChopBlanks** ’}； 确定提取行的方法是否从 CHAR 列值中剪裁掉后面的空白。对于大多数数据库驱动程序来说，缺省值为禁用 ChopBlanks，但是，对 MySQL 不会产生影响，因为服务器始终去掉 CHAR 值后的空白。

■ $h->{‘ **PrintError** ’}； 如果启用，则出现有关 DBI 的错误时会显示警告消息。缺省为启用 PrintError。

■ $h->{‘ **RaiseError** ’}； 如果启用，则出现与 DBI 有关的错误会使脚本自动终止。缺省为启用 RaiseError。

G.4.2 动态属性

这些属性与最近使用的句柄有关，在下面的描述中用 $h 来表示：

■ **$DBI::err** 与 $h->err( ) 调用相同。

■ **$$DBI::errstr** 与 $h->errstr( ) 调用相同。

■ **$DBI::rows** 与 $h->rows( ) 调用相同。

G.4.3 MySQL特定的数据库句柄属性

这些属性是专门针对 DBI MySQL 驱动程序 DBD::mysql 的。

■ $str = $dbh->{‘ **info** ’}； 此属性含有与 C API函数 mysql_info( ) 返回的信息相同的信息。请参阅附录 F 中该函数的描述。

■ $rv = $dbh->{‘ **mysql_insertid** ’}； AUTO_INCREMENT的值是最近在与 $dbh 有关的连接上生成的。

■ $rv = $dbh->{‘ **thread_id** ’}； 与 $dbh 相关的连接的线程数。

G.4.4 语句句柄属性

通常，这些属性用于 SELECT 查询，而且在将查询传递给 prepare() 获得了语句句柄，而且为这个句柄调用了 execute() 时，这些属性才有效。此外， finish() 可能会使某些语句属性无效。

许多属性具有作为数组值引用的值，查询的每列都有一个值。数组中元素的数量由属性 $sth->{ ‘ NUM_OF_FIELDS ’ } 给出。语句属性 *stmt_attr*是对数组的引用，可以用 @{$sth->{*stmt_attr*}} 访问整个数组，或者由如下数组元素的循环来访问：

```
for (my $i = 0; $i < $sth->{NUM_OF_FIELDS}; $i++)
{
    $value = $sth->{stmt_attr}->[$i];
}
```

- $sth->{ ‘ **NAME** ’ }； 对表示每列的名称的字符串数组的引用。这些名称的大小写字符与 SELECT 语句中给出的相同。
- $sth->{ ‘ **NAME_lc** ’ }； 表示每列名称的字符串数组的引用。返回的名称为小写字符串。
- $sth->{ ‘ **NAME_uc** ’ }； 表示每列名称的字符串数组的引用。返回的名称为大写字符串。
- $sth->{ ‘ **NULLABLE** ’ }； 表示每列是否可以为 NULL 值的数组引用。每个元素的值可为0（非）、1（是）或2（不知道）。至少，DBI 文档是这样说的。如果列不为 NULL，则它实际显示该值为空字符串。
- $sth->{ ‘ **NUM_OF_FIELDS** ’ }； 准备好的语句将返回的列数，或对非 SELECT 语句返回0。
- $sth->{ ‘ **NUM_OF_PARAMS**}； 准备好的语句中的占位符数量。
- $sth->{ ‘ **PRECISION** ’ }； 表示每列精度的值的数组引用。 DBI 采用 ODBC 意义中的“精度”，对MySQL意味着该列的最大宽度。对于数值列，这是显示宽度。对于字符串列，它是该列的最大长度。
- $sth->{ ‘ **SCALE** ’ }； 表示每列的范围的值的数组引用。 DBI 采用 ODBC 意义中的“范围”，对MySQL意味着浮点列的小数位置。对其他列来说，这个范围为 0。
- $sth->{ ‘ **Statement** ’ }； 与 $sth 相关的语句文本。在进行任何占位符替换以前，这个文本是 prepare() 所看到的。
- $sth->{ ‘ **TYPE** ’ }； 表示每列的数值类型的值的数组引用。

G.4.5 MySQL 特定的语句句柄属性

大多数语句句柄属性都被认为是只读的，并且应该在 execute() 调用之后访问。但 mysql_store_result 和 mysql_use_result 例外。DBD::mysql 提供控制您的脚本所用的结果集处理风格的功能。这条语句的句柄属性 mysql_store_result 和 mysql_use_result 选择C API 函数 mysql_store_result() 和 mysql_use_result() 的结果集处理的行为。有关这两个函数的讨论及它们的不同，请参阅附录 F。

缺省设置时，DBI 使用 mysql_store_result()，但是可以启用 mysql_use_result 属性，它告

诉 DBI 使用 mysql_use_result( )。在 prepare( ) 之后及 execute( ) 之前做这件事：

```
$sth = $dbh->prepare (...);
$sth->{mysql_use_result} = 1;
$sth->execute();
```

DBD::mysql 的旧版本中可用的若干 MySQL 专用属性现在不赞成使用，已经由更新更好的格式替换，如表G-2 所示。如果 DBD::mysql 是旧版本，不支持这些更新的属性，则试着用这些不赞成的格式（否则应升级为更高的版本）。

请注意，insertid为语句句柄属性，而它的首选形式mysql_insertid是数据库句柄属性。

表G-2 不赞成使用的 MySQL 专用属性

不赞成的属性	首选的属性
insertid	mysql_insertid
is_blob	mysql_is_blob
is_key	mysql_ is_key
is_not_null	mysql_ is_not_null
is_num	mysql_ is_num
is_pri_ key	mysql_ is_pri_ key
length	PRECISION
max_length	mysql_ max_length
table	mysql_ table

- $sth->{‘ **mysql_is_blob** ’}； 表示每列是否为BLOB类型的值的数组引用。
- $sth->{‘ **mysql_is_key** ’}； 表示每列是否为非唯一键的值的数组引用。
- $sth->{‘ **mysql_is_not_null** ’}； 表示每列是否可为 NULL 的值的数组引用。假值表示该列可包括 NULL 值。这个属性的信息为 NULLABLE 属性信息中“反转”的信息。
- $sth->{‘ **mysql_is_ num** ’}； 表示每列是否可为数值类型的值的数组引用。
- $sth->{‘ **mysql_is_pri_key** ’}； 表示每列是否可为 PRIMARY KEY 的组成部分的值的数组引用。
- $sth->{‘ **mysql_ max_length** ’}； 表示结果集中列值实际的最大长度的值的数组引用。
- $sth->{‘ **mysql_store_result** ’}； 如果启动 mysql_store_result （设为1），则利用 mysql_store_result C API 函数，而非 mysql_store_result 从 MySQL 服务器中检索结果集。关于这两个函数及它们之间区别的介绍请参阅附录 F。

如果设置 mysql_store_result 属性，则在 prepare( ) 调用之后及 execute( ) 调用之前再设置。

- $sth->{‘ **mysql_table** ’}； 表示列来自其中的表名的值的数组引用。计算列的表名为空字符串。
- $sth->{‘ **mysql_type** ’}； 表示结果集中每列的 MySQL 类型的数的值的数组引用。
- $sth->{‘ **mysql_type_name** ’}； 表示结果集中每列的 MySQL 类型的名的值的数组引用。
- $sth->{‘ **mysql_use_result** ’}； 如果启动 mysql_use_result（设为1），则使用

mysql_use_result C API 函数，而不是 mysql_use_result 从 MySQL 服务器中检索结果集。请参阅附录F关于这两个函数及它们之间区别的讨论。

请注意，使用这个属性会导致一些属性成为非法的，如 mysql_max_length。虽然在以任何方式提取它们的时候最好对行进行计数，但是，它也会使 rows( ) 方法的使用无效。

如果设置 mysql_use_result 属性，则在 prepare( ) 调用之后及 execute( ) 调用之前再设置。

G.5 DBI 环境变量

DBI 考虑了几个环境变量，如表 G-3 所示。除了 DBI_TRACE 之外，所有变量都由 connect( ) 方法使用。DBI_TRACE 由 trace( ) 方法使用。

表G-3 DBI 环境变量

名 称	含 义
DBI_DRIVER	DBI 级的驱动程序名（MySQL的“ mysql ”）
DBI_DSN	数据源名
DBI_PASS	口令
DBI_TRACE	跟踪级别和/或跟踪输出文件
DBI_USER	用户名称