MySQL

Introduction

These functions allow you to access MySQL database servers. More information about MySQL can be found at » http://www.mysql.com/.

Documentation for MySQL can be found at » http://dev.mysql.com/doc/.

Installing/Configuring

Requirements

In order to have these functions available, you must compile PHP with MySQL support.

Installation

For compiling, simply use the *--with-mysql[=DIR]* configuration option where the optional *[DIR]* points to the MySQL installation directory.

Although this MySQL extension is compatible with MySQL 4.1.0 and greater, it doesn't support the extra functionality that these versions provide. For that, use the MySQLi extension.

If you would like to install the mysql extension along with the mysqli extension you have to use the same client library to avoid any conflicts.

Installation on Linux Systems

PHP 4

The option *--with-mysql* is enabled by default. This default behavior may be disabled with the *--without-mysql* configure option. If MySQL is enabled without specifying the path to the MySQL install DIR, PHP will use the bundled MySQL client libraries.

Users who run other applications that use MySQL (for example, auth-mysql) should not use the bundled library, but rather specify the path to MySQL's install directory, like so: *--with-mysql=/path/to/mysql*. This will force PHP to use the client libraries installed by MySQL, thus avoiding any conflicts.

PHP 5+

MySQL is not enabled by default, nor is the MySQL library bundled with PHP. Read this FAQ for details on why. Use the *--with-mysql[=DIR]* configure option to include MySQL support. You can download *headers and libraries* from » MySQL.

Installation on Windows Systems

PHP 4

The PHP MySQL extension is compiled into PHP.

PHP 5+

MySQL is no longer enabled by default, so the *php_mysql.dll* DLL must be enabled inside of *php.ini*. Also, PHP needs access to the MySQL client library. A file named *libmysql.dll* is included in the Windows PHP distribution and in order for PHP to talk to MySQL this file needs to be available to the Windows systems PATH. See the FAQ titled " How do I add my PHP directory to the PATH on Windows " for information on how to do this. Although copying *libmysql.dll* to the Windows system directory also works (because the system directory is by default in the system's PATH), it's not recommended.

As with enabling any PHP extension (such as *php_mysql.dll*), the PHP directive extension_dir should be set to the directory where the PHP extensions are located. See also the Manual Windows Installation Instructions. An example extension_dir value for PHP 5 is *c:\php\ext*

Note
If when starting the web server an error similar to the following occurs: *"Unable to load dynamic library './php_mysql.dll'"*, this is because *php_mysql.dll* and/or *libmysql.dll* cannot be found by the system.

MySQL Installation Notes

Warning
Crashes and startup problems of PHP may be encountered when loading this extension in conjunction with the recode extension. See the recode extension for more information.

Note
If you need charsets other than *latin* (default), you have to install external (not bundled) libmysql with compiled charset support.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

MySQL Configuration Options

Name	Default	Changeable	Changelog
mysql.allow_persistent	"1"	PHP_INI_SYSTEM	
mysql.max_persistent	"-1"	PHP_INI_SYSTEM	
mysql.max_links	"-1"	PHP_INI_SYSTEM	
mysql.trace_mode	"0"	PHP_INI_ALL	Available since PHP 4.3.0.
mysql.default_port	NULL	PHP_INI_ALL	
mysql.default_socket	NULL	PHP_INI_ALL	Available since PHP 4.0.1.
mysql.default_host	NULL	PHP_INI_ALL	
mysql.default_user	NULL	PHP_INI_ALL	
mysql.default_password	NULL	PHP_INI_ALL	
mysql.connect_timeout	"60"	PHP_INI_ALL	PHP_INI_SYSTEM in PHP <= 4.3.2. Available since PHP 4.3.0.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

mysql.allow_persistent boolean
: Whether to allow persistent connections to MySQL.

mysql.max_persistent integer
: The maximum number of persistent MySQL connections per process.

mysql.max_links integer
: The maximum number of MySQL connections per process, including persistent connections.

mysql.trace_mode boolean
: Trace mode. When *mysql.trace_mode* is enabled, warnings for table/index scans, non free result sets, and SQL-Errors will be displayed. (Introduced in PHP 4.3.0)

mysql.default_port string
: The default TCP port number to use when connecting to the database server if no

other port is specified. If no default is specified, the port will be obtained from the MYSQL_TCP_PORT environment variable, the *mysql-tcp* entry in */etc/services* or the compile-time **MYSQL_PORT** constant, in that order. Win32 will only use the **MYSQL_PORT** constant.

mysql.default_socket string
: The default socket name to use when connecting to a local database server if no other socket name is specified.

mysql.default_host string
: The default server host to use when connecting to the database server if no other host is specified. Doesn't apply in SQL safe mode.

mysql.default_user string
: The default user name to use when connecting to the database server if no other name is specified. Doesn't apply in SQL safe mode.

mysql.default_password string
: The default password to use when connecting to the database server if no other password is specified. Doesn't apply in SQL safe mode.

mysql.connect_timeout integer
: Connect timeout in seconds. On Linux this timeout is also used for waiting for the first answer from the server.

Resource Types

There are two resource types used in the MySQL module. The first one is the link identifier for a database connection, the second a resource which holds the result of a query.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Since PHP 4.3.0 it is possible to specify additional client flags for the mysql_connect() and mysql_pconnect() functions. The following constants are defined:

MySQL client constants

Constant	Description
MYSQL_CLIENT_COMPRESS	Use compression protocol
MYSQL_CLIENT_IGNORE_SPACE	Allow space after function names
MYSQL_CLIENT_INTERACTIVE	Allow interactive_timeout seconds (instead of wait_timeout) of inactivity before closing the connection.
MYSQL_CLIENT_SSL	Use SSL encryption. This flag is only available with version 4.x of the MySQL client library or newer. Version 3.23.x is bundled both with PHP 4 and Windows binaries of PHP 5.

The function mysql_fetch_array() uses a constant for the different types of result arrays. The following constants are defined:

MySQL fetch constants

Constant	Description
MYSQL_ASSOC	Columns are returned into the array having the fieldname as the array index.
MYSQL_BOTH	Columns are returned into the array having both a numerical index and the fieldname as the array index.
MYSQL_NUM	Columns are returned into the array having a numerical index to the fields. This index starts with 0, the first field in the result.

Examples

This simple example shows how to connect, execute a query, print resulting rows and disconnect from a MySQL database.

Example #1 - MySQL extension overview example

```
<?php
// Connecting, selecting database
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
   or die('Could not connect: ' . mysql_error());
echo 'Connected successfully';
mysql_select_db('my_database') or die('Could not select database');

// Performing SQL query
$query = 'SELECT * FROM my_table';
$result = mysql_query($query) or die('Query failed: ' . mysql_error());

// Printing results in HTML
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
   echo "\t<tr>\n";
   foreach ($line as $col_value) {
       echo "\t\t<td>$col_value</td>\n";
   }
   echo "\t</tr>\n";
}
echo "</table>\n";

// Free resultset
mysql_free_result($result);

// Closing connection
mysql_close($link);
?>
```

MySQL Functions

Notes

Note
Most MySQL functions accept *`link_identifier`* as the last optional parameter. If it is not provided, last opened connection is used. If it doesn't exist, connection is tried to establish with default parameters defined in *php.ini*. If it is not successful, functions return **FALSE**.

mysql_affected_rows

mysql_affected_rows -- Get number of affected rows in previous MySQL operation

Description

int **mysql_affected_rows** ([resource `$link_identifier`])

Get the number of affected rows by the last INSERT, UPDATE, REPLACE or DELETE query associated with *`link_identifier`*.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns the number of affected rows on success, and -1 if the last query failed.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero with MySQL versions prior to 4.1.2.

When using UPDATE, MySQL will not update columns where the new value is the same as the old value. This creates the possibility that mysql_affected_rows() may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

The REPLACE statement first deletes the record with the same primary key and then inserts the new record. This function returns the number of deleted records plus the number of inserted records.

Examples

Example #2 - mysql_affected_rows() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
```

```
mysql_select_db('mydb');

/* this should return the correct numbers of deleted records */
mysql_query('DELETE FROM mytable WHERE id < 10');
printf("Records deleted: %d\n", mysql_affected_rows());

/* with a where clause that is never true, it should return 0 */
mysql_query('DELETE FROM mytable WHERE 0');
printf("Records deleted: %d\n", mysql_affected_rows());
?>
```

The above example will output something similar to:

```
Records deleted: 10
Records deleted: 0
```

Example #3 - mysql_affected_rows() example using transactions

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db('mydb');

/* Update records */
mysql_query("UPDATE mytable SET used=1 WHERE id < 10");
printf ("Updated records: %d\n", mysql_affected_rows());
mysql_query("COMMIT");
?>
```

The above example will output something similar to:

```
Updated Records: 10
```

Notes

Note

Transactions

If you are using transactions, you need to call mysql_affected_rows() after your INSERT, UPDATE, or DELETE query, not after the COMMIT.

Note

SELECT Statements

To retrieve the number of rows returned by a SELECT, it is possible to use mysql_num_rows().

See Also

- mysql_num_rows()
- mysql_info()

mysql_change_user

mysql_change_user -- Change logged in user of the active connection

Description

int **mysql_change_user** (string `$user`, string `$password` [, string `$database` [, resource `$link_identifier`]])

mysql_change_user() changes the logged in user of the current active connection, or the connection given by the optional *`link_identifier`* parameter. If a database is specified, this will be the current database after the user has been changed. If the new user and password authorization fails, the current connected user stays active.

This function is deprecated and no longer exists in PHP.

Parameters

`user`
: The new MySQL username.

`password`
: The new MySQL password.

`database`
: The MySQL database. If not specified, the current selected database is used.

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ChangeLog

Version	Description
3.0.14	This function was removed from PHP.

Notes

Note
Requirements This function requires MySQL 3.23.3 or higher.

See Also

- mysql_connect()
- mysql_select_db()
- mysql_query()

mysql_client_encoding

mysql_client_encoding -- Returns the name of the character set

Description

string **mysql_client_encoding** ([resource `$link_identifier`])

Retrieves the *character_set* variable from MySQL.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns the default character set name for the current connection.

Examples

Example #4 - mysql_client_encoding() example

```
<?php
$link    = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$charset = mysql_client_encoding($link);

echo "The current character set is: $charset\n";
?>
```

The above example will output something similar to:

```
The current character set is: latin1
```

See Also

- mysql_real_escape_string()

mysql_close

mysql_close -- Close MySQL connection

Description

bool **mysql_close** ([resource `$link_identifier`])

mysql_close() closes the non-persistent connection to the MySQL server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is used.

Using mysql_close() isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution. See also freeing resources.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #5 - mysql_close() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
?>
```

The above example will output:

```
Connected successfully
```

Notes

Note
mysql_close() will not close persistent links created by mysql_pconnect().

See Also

- mysql_connect()
- mysql_free_result()

mysql_connect

mysql_connect -- Open a connection to a MySQL Server

Description

resource **mysql_connect** ([string `$server` [, string `$username` [, string `$password` [, bool `$new_link` [, int `$client_flags`]]]]])

Opens or reuses a connection to a MySQL server.

Parameters

`server`
: The MySQL server. It can also include a port number. e.g. "hostname:port" or a path to a local socket e.g. ":/path/to/socket" for the localhost. If the PHP directive mysql.default_host is undefined (default), then the default value is 'localhost:3306'. In SQL safe mode, this parameter is ignored and value 'localhost:3306' is always used.

`username`
: The username. Default value is defined by mysql.default_user. In SQL safe mode, this parameter is ignored and the name of the user that owns the server process is used.

`password`
: The password. Default value is defined by mysql.default_password. In SQL safe mode, this parameter is ignored and empty password is used.

`new_link`
: If a second call is made to mysql_connect() with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned. The *`new_link`* parameter modifies this behavior and makes mysql_connect() always open a new link, even if mysql_connect() was called before with the same parameters. In SQL safe mode, this parameter is ignored.

`client_flags`
: The *`client_flags`* parameter can be a combination of the following constants: 128 (enable *LOAD DATA LOCAL* handling), **MYSQL_CLIENT_SSL**, **MYSQL_CLIENT_COMPRESS**, **MYSQL_CLIENT_IGNORE_SPACE** or **MYSQL_CLIENT_INTERACTIVE**. Read the section about MySQL client constants for further information. In SQL safe mode, this parameter is ignored.

Return Values

Returns a MySQL link identifier on success, or **FALSE** on failure.

ChangeLog

Version	Description
4.3.0	Added the *client_flags* parameter.
4.2.0	Added the *new_link* parameter.
3.0.10	Added support for ":/path/to/socket" with *server*.
3.0.0	Added support for ":port" with *server*.

Examples

Example #6 - mysql_connect() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
?>
```

Example #7 - mysql_connect() example using *hostname:port* syntax

```
<?php
// we connect to example.com and port 3307
$link = mysql_connect('example.com:3307', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);

// we connect to localhost at port 3307
$link = mysql_connect('127.0.0.1:3307', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
?>
```

Example #8 - mysql_connect() example using ":/path/to/socket" syntax

```
<?php
// we connect to localhost and socket e.g. /tmp/mysql.sock

//variant 1: ommit localhost
$link = mysql_connect(':/tmp/mysql', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);


// variant 2: with localhost
$link = mysql_connect('localhost:/tmp/mysql.sock', 'mysql_user',
'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
?>
```

Notes

Note

Whenever you specify "localhost" or "localhost:port" as server, the MySQL client library will override this and try to connect to a local socket (named pipe on Windows). If you want to use TCP/IP, use "127.0.0.1" instead of "localhost". If the MySQL client library tries to connect to the wrong local socket, you should set the correct path as in your PHP configuration and leave the server field blank.

Note

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling mysql_close().

Note

You can suppress the error message on failure by prepending a @ to the function name.

Note
Error "Can't create TCP/IP socket (10106)" usually means that the variables_order configure directive doesn't contain character *E*. On Windows, if the environment is not copied the *SYSTEMROOT* environment variable won't be available and PHP will have problems loading Winsock.

See Also

- mysql_pconnect()
- mysql_close()

mysql_create_db

mysql_create_db -- Create a MySQL database

Description

bool **mysql_create_db** (string `$database_name` [, resource `$link_identifier`])

mysql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Parameters

`database_name`
The name of the database being created.

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #9 - mysql_create_db() alternative example

The function mysql_create_db() is deprecated. It is preferable to use mysql_query() to issue a sql *CREATE DATABASE* statement instead.

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}

$sql = 'CREATE DATABASE my_db';
if (mysql_query($sql, $link)) {
   echo "Database my_db created successfully\n";
} else {
   echo 'Error creating database: ' . mysql_error() . "\n";
}
?>
```

The above example will output something similar to:

```
Database my_db created successfully
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_createdb()**

Note
This function will not be available if the MySQL extension was built against a MySQL 4.x client library.

See Also

- mysql_query()
- mysql_select_db()

mysql_data_seek

mysql_data_seek -- Move internal result pointer

Description

bool **mysql_data_seek** (resource `$result`, int `$row_number`)

mysql_data_seek() moves the internal row pointer of the MySQL result associated with the specified result identifier to point to the specified row number. The next call to a MySQL fetch function, such as mysql_fetch_assoc(), would return that row.

`row_number` starts at 0. The *`row_number`* should be a value in the range from 0 to mysql_num_rows() - 1. However if the result set is empty (mysql_num_rows() == 0), a seek to 0 will fail with a E_WARNING and mysql_data_seek() will return **FALSE**.

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

`row_number`
The desired row number of the new result pointer.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #10 - mysql_data_seek() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
$db_selected = mysql_select_db('sample_db');
if (!$db_selected) {
   die('Could not select database: ' . mysql_error());
}
$query = 'SELECT last_name, first_name FROM friends';
$result = mysql_query($query);
if (!$result) {
   die('Query failed: ' . mysql_error());
```

```
}
/* fetch rows in reverse order */
for ($i = mysql_num_rows($result) - 1; $i >= 0; $i--) {
   if (!mysql_data_seek($result, $i)) {
       echo "Cannot seek to row $i: " . mysql_error() . "\n";
       continue;
   }

   if (!($row = mysql_fetch_assoc($result))) {
       continue;
   }

   echo $row['last_name'] . ' ' . $row['first_name'] . "<br />\n";
}

mysql_free_result($result);
?>
```

Notes

Note
The function mysql_data_seek() can be used in conjunction only with mysql_query(), not with mysql_unbuffered_query().

See Also

- mysql_query()
- mysql_num_rows()
- mysql_fetch_row()
- mysql_fetch_assoc()
- mysql_fetch_array()
- mysql_fetch_object()

mysql_db_name

mysql_db_name -- Get result data

Description

string **mysql_db_name** (resource `$result`, int `$row` [, mixed `$field`])

Retrieve the database name from a call to mysql_list_dbs().

Parameters

result
The result pointer from a call to mysql_list_dbs().

row
The index into the result set.

field
The field name.

Return Values

Returns the database name on success, and **FALSE** on failure. If **FALSE** is returned, use mysql_error() to determine the nature of the error.

Examples

Example #11 - mysql_db_name() example

```
<?php
error_reporting(E_ALL);

$link = mysql_connect('dbhost', 'username', 'password');
$db_list = mysql_list_dbs($link);

$i = 0;
$cnt = mysql_num_rows($db_list);
while ($i < $cnt) {
   echo mysql_db_name($db_list, $i) . "\n";
   $i++;
}
?>
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_dbname()**

See Also

- mysql_list_dbs()
- mysql_tablename()

mysql_db_query

mysql_db_query -- Send a MySQL query

Description

resource **mysql_db_query** (string `$database`, string `$query` [, resource `$link_identifier`])

mysql_db_query() selects a database, and executes a query on it.

Parameters

`database`
The name of the database that will be selected.

`query`
The MySQL query.

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns a positive MySQL result resource to the query result, or **FALSE** on error. The function also returns **TRUE** / **FALSE** for *INSERT* / *UPDATE* / *DELETE* queries to indicate success/failure.

ChangeLog

Version	Description
4.0.6	This function is deprecated, do not use this function. Use mysql_select_db() and mysql_query() instead.

Examples

Example #12 - mysql_db_query() alternative example

```
<?php

if (!$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')) {
   echo 'Could not connect to mysql';
   exit;
}

if (!mysql_select_db('mysql_dbname', $link)) {
   echo 'Could not select database';
   exit;
}

$sql    = 'SELECT foo FROM bar WHERE id = 42';
$result = mysql_query($sql, $link);

if (!$result) {
   echo "DB Error, could not query the database\n";
   echo 'MySQL Error: ' . mysql_error();
   exit;
}

while ($row = mysql_fetch_assoc($result)) {
   echo $row['foo'];
}

mysql_free_result($result);

?>
```

Notes

Note

Be aware that this function does *NOT* switch back to the database you were connected before. In other words, you can't use this function to *temporarily* run a sql query on another database, you would have to manually switch back. Users are strongly encouraged to use the *database.table* syntax in their sql queries or mysql_select_db() instead of this function.

See Also

- mysql_query()
- mysql_select_db()

mysql_drop_db

mysql_drop_db -- Drop (delete) a MySQL database

Description

bool **mysql_drop_db** (string `$database_name` [, resource `$link_identifier`])

mysql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier. This function is deprecated, it is preferable to use mysql_query() to issue a sql *DROP DATABASE* statement instead.

Parameters

`database_name`
: The name of the database that will be deleted.

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #13 - mysql_drop_db() alternative example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}

$sql = 'DROP DATABASE my_db';
if (mysql_query($sql, $link)) {
   echo "Database my_db was successfully dropped\n";
} else {
   echo 'Error dropping database: ' . mysql_error() . "\n";
}
?>
```

Notes

Warning
This function will not be available if the MySQL extension was built against a MySQL 4.x client library.

Note
For backward compatibility, the following deprecated alias may be used: **mysql_dropdb()**

See Also

- mysql_query()

mysql_errno

mysql_errno -- Returns the numerical value of the error message from previous MySQL operation

Description

int **mysql_errno** ([resource `$link_identifier`])

Returns the error number from the last MySQL function.

Errors coming back from the MySQL database backend no longer issue warnings. Instead, use mysql_errno() to retrieve the error code. Note that this function only returns the error code from the most recently executed MySQL function (not including mysql_error() and mysql_errno()), so if you want to use it, make sure you check the value before calling another MySQL function.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns the error number from the last MySQL function, or *0* (zero) if no error occurred.

Examples

Example #14 - mysql_errno() example

```
<?php
$link = mysql_connect("localhost", "mysql_user", "mysql_password");

if (!mysql_select_db("nonexistentdb", $link)) {
   echo mysql_errno($link) . ": " . mysql_error($link). "\n";
}

mysql_select_db("kossu", $link);
if (!mysql_query("SELECT * FROM nonexistenttable", $link)) {
   echo mysql_errno($link) . ": " . mysql_error($link) . "\n";
}
?>
```

The above example will output something similar to:

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

See Also

- mysql_error()
- » MySQL error codes

mysql_error

mysql_error -- Returns the text of the error message from previous MySQL operation

Description

string **mysql_error** ([resource `$link_identifier`])

Returns the error text from the last MySQL function. Errors coming back from the MySQL database backend no longer issue warnings. Instead, use mysql_error() to retrieve the error text. Note that this function only returns the error text from the most recently executed MySQL function (not including mysql_error() and mysql_errno()), so if you want to use it, make sure you check the value before calling another MySQL function.

Parameters

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns the error text from the last MySQL function, or '' (empty string) if no error occurred.

Examples

Example #15 - mysql_error() example

```
<?php
$link = mysql_connect("localhost", "mysql_user", "mysql_password");

mysql_select_db("nonexistentdb", $link);
echo mysql_errno($link) . ": " . mysql_error($link). "\n";

mysql_select_db("kossu", $link);
mysql_query("SELECT * FROM nonexistenttable", $link);
echo mysql_errno($link) . ": " . mysql_error($link) . "\n";
?>
```

The above example will output something similar to:

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

See Also

- mysql_errno()
- » MySQL error codes

mysql_escape_string

mysql_escape_string -- Escapes a string for use in a mysql_query

Description

string **mysql_escape_string** (string `$unescaped_string`)

This function will escape the *`unescaped_string`*, so that it is safe to place it in a mysql_query(). This function is deprecated.

This function is identical to mysql_real_escape_string() except that mysql_real_escape_string() takes a connection handler and escapes the string according to the current character set. mysql_escape_string() does not take a connection argument and does not respect the current charset setting.

Parameters

`unescaped_string`
The string that is to be escaped.

Return Values

Returns the escaped string.

ChangeLog

Version	Description
4.3.0	This function became deprecated, do not use this function. Instead, use mysql_real_escape_string().

Examples

Example #16 - mysql_escape_string() example

```
<?php
$item = "Zak's Laptop";
$escaped_item = mysql_escape_string($item);
printf("Escaped string: %s\n", $escaped_item);
```

```
?>
```

The above example will output:

```
Escaped string: Zak\'s Laptop
```

Notes

Note
mysql_escape_string() does not escape % and _.

See Also

- mysql_real_escape_string()
- addslashes()
- The magic_quotes_gpc directive.

mysql_fetch_array

mysql_fetch_array -- Fetch a result row as an associative array, a numeric array, or both

Description

array **mysql_fetch_array** (resource `$result` [, int `$result_type`])

Returns an array that corresponds to the fetched row and moves the internal data pointer ahead.

Parameters

`result`
: The result resource that is being evaluated. This result comes from a call to mysql_query().

`result_type`
: The type of array that is to be fetched. It's a constant and can take the following values: **MYSQL_ASSOC**, **MYSQL_NUM**, and the default value of **MYSQL_BOTH**.

Return Values

Returns an array of strings that corresponds to the fetched row, or **FALSE** if there are no more rows. The type of returned array depends on how *`result_type`* is defined. By using **MYSQL_BOTH** (default), you'll get an array with both associative and number indices. Using **MYSQL_ASSOC**, you only get associative indices (as mysql_fetch_assoc() works), using **MYSQL_NUM**, you only get number indices (as mysql_fetch_row() works).

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column. For aliased columns, you cannot access the contents with the original column name.

Examples

Example #17 - Query with aliased duplicate field names

```
SELECT table1.field AS foo, table2.field AS bar FROM table1, table2
```

Example #18 - mysql_fetch_array() with MYSQL_NUM

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
   die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_NUM)) {
   printf("ID: %s  Name: %s", $row[0], $row[1]);
}

mysql_free_result($result);
?>
```

Example #19 - mysql_fetch_array() with MYSQL_ASSOC

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
   die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
   printf("ID: %s  Name: %s", $row["id"], $row["name"]);
}

mysql_free_result($result);
?>
```

Example #20 - mysql_fetch_array() with MYSQL_BOTH

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password") or
   die("Could not connect: " . mysql_error());
mysql_select_db("mydb");

$result = mysql_query("SELECT id, name FROM mytable");

while ($row = mysql_fetch_array($result, MYSQL_BOTH)) {
   printf ("ID: %s  Name: %s", $row[0], $row["name"]);
}

mysql_free_result($result);
?>
```

Notes

Note
Performance An important thing to note is that using mysql_fetch_array() is *not significantly* slower than using mysql_fetch_row(), while it provides a significant added value.

Note
Field names returned by this function are *case-sensitive*.

Note
This function sets NULL fields to the PHP **NULL** value.

See Also

- mysql_fetch_row()
- mysql_fetch_assoc()
- mysql_data_seek()
- mysql_query()

mysql_fetch_assoc

mysql_fetch_assoc -- Fetch a result row as an associative array

Description

array **mysql_fetch_assoc** (resource `$result`)

Returns an associative array that corresponds to the fetched row and moves the internal data pointer ahead. mysql_fetch_assoc() is equivalent to calling mysql_fetch_array() with MYSQL_ASSOC for the optional second parameter. It only returns an associative array.

Parameters

result
The result resource that is being evaluated. This result comes from a call to mysql_query().

Return Values

Returns an associative array of strings that corresponds to the fetched row, or **FALSE** if there are no more rows.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you either need to access the result with numeric indices by using mysql_fetch_row() or add alias names. See the example at the mysql_fetch_array() description about aliases.

Examples

Example #21 - An expanded mysql_fetch_assoc() example

```
<?php

$conn = mysql_connect("localhost", "mysql_user", "mysql_password");

if (!$conn) {
   echo "Unable to connect to DB: " . mysql_error();
   exit;
}

if (!mysql_select_db("mydbname")) {
   echo "Unable to select mydbname: " . mysql_error();
   exit;
}

$sql = "SELECT id as userid, fullname, userstatus
```

```
       FROM   sometable
       WHERE  userstatus = 1";

$result = mysql_query($sql);

if (!$result) {
   echo "Could not successfully run query ($sql) from DB: " . mysql_error();
   exit;
}

if (mysql_num_rows($result) == 0) {
   echo "No rows found, nothing to print so am exiting";
   exit;
}

// While a row of data exists, put that row in $row as an associative array
// Note: If you're expecting just one row, no need to use a loop
// Note: If you put extract($row); inside the following loop, you'll
//       then create $userid, $fullname, and $userstatus
while ($row = mysql_fetch_assoc($result)) {
   echo $row["userid"];
   echo $row["fullname"];
   echo $row["userstatus"];
}

mysql_free_result($result);

?>
```

Notes

Note

Performance

An important thing to note is that using mysql_fetch_assoc() is *not significantly* slower than using mysql_fetch_row(), while it provides a significant added value.

Note

Field names returned by this function are *case-sensitive*.

Note

This function sets NULL fields to the PHP **NULL** value.

See Also

- mysql_fetch_row()
- mysql_fetch_array()
- mysql_data_seek()
- mysql_query()
- mysql_error()

mysql_fetch_field

mysql_fetch_field -- Get column information from a result and return as an object

Description

object **mysql_fetch_field** (resource `$result` [, int `$field_offset`])

Returns an object containing field information. This function can be used to obtain information about fields in the provided query result.

Parameters

`result`
: The result resource that is being evaluated. This result comes from a call to mysql_query().

`field_offset`
: The numerical field offset. If the field offset is not specified, the next field that was not yet retrieved by this function is retrieved. The *`field_offset`* starts at *0*.

Return Values

Returns an object containing field information. The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- def - default value of the column
- max_length - maximum length of the column
- not_null - 1 if the column cannot be **NULL**
- primary_key - 1 if the column is a primary key
- unique_key - 1 if the column is a unique key
- multiple_key - 1 if the column is a non-unique key
- numeric - 1 if the column is numeric
- blob - 1 if the column is a BLOB
- type - the type of the column
- unsigned - 1 if the column is unsigned
- zerofill - 1 if the column is zero-filled

Examples

Example #22 - mysql_fetch_field() example

```
<?php
$conn = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$conn) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db('database');
$result = mysql_query('select * from table');
if (!$result) {
   die('Query failed: ' . mysql_error());
}
/* get column metadata */
$i = 0;
while ($i < mysql_num_fields($result)) {
   echo "Information for column $i:<br />\n";
   $meta = mysql_fetch_field($result, $i);
   if (!$meta) {
       echo "No information available<br />\n";
   }
   echo "<pre>
blob:         $meta->blob
max_length:   $meta->max_length
multiple_key: $meta->multiple_key
name:         $meta->name
not_null:     $meta->not_null
numeric:      $meta->numeric
primary_key:  $meta->primary_key
table:        $meta->table
type:         $meta->type
default:      $meta->def
unique_key:   $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
</pre>";
   $i++;
}
mysql_free_result($result);
?>
```

Notes

Note

Field names returned by this function are *case-sensitive*.

See Also

- mysql_field_seek()

mysql_fetch_lengths

mysql_fetch_lengths -- Get the length of each output in a result

Description

array **mysql_fetch_lengths** (resource `$result`)

Returns an array that corresponds to the lengths of each field in the last row fetched by MySQL.

mysql_fetch_lengths() stores the lengths of each result column in the last row returned by mysql_fetch_row(), mysql_fetch_assoc(), mysql_fetch_array(), and mysql_fetch_object() in an array, starting at offset 0.

Parameters

result
The result resource that is being evaluated. This result comes from a call to mysql_query().

Return Values

An array of lengths on success, or **FALSE** on failure.

Examples

Example #23 - A mysql_fetch_lengths() example

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}
$row     = mysql_fetch_assoc($result);
$lengths = mysql_fetch_lengths($result);

print_r($row);
print_r($lengths);
?>
```

The above example will output something similar to:

```
Array
(
   [id] => 42
```

```
    [email] => user@example.com
)
Array
(
    [0] => 2
    [1] => 16
)
```

See Also

- mysql_field_len()
- mysql_fetch_row()
- strlen()

mysql_fetch_object

mysql_fetch_object -- Fetch a result row as an object

Description

object **mysql_fetch_object** (resource `$result` [, string `$class_name` [, array `$params`]])

Returns an object with properties that correspond to the fetched row and moves the internal data pointer ahead.

Parameters

`result`
: The result resource that is being evaluated. This result comes from a call to mysql_query().

`class_name`
: The name of the class to instantiate, set the properties of and return. If not specified, a stdClass object is returned.

`params`
: An optional array of parameters to pass to the constructor for *`class_name`* objects.

Return Values

Returns an object with string properties that correspond to the fetched row, or **FALSE** if there are no more rows.

mysql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

ChangeLog

Version	Description
5.0.0	Added the ability to return as a different object.

Examples

Example #24 - mysql_fetch_object() example

```
<?php
mysql_connect("hostname", "user", "password");
mysql_select_db("mydb");
$result = mysql_query("select * from mytable");
while ($row = mysql_fetch_object($result)) {
   echo $row->user_id;
   echo $row->fullname;
}
mysql_free_result($result);
?>
```

Example #25 - mysql_fetch_object() example

```
<?php
class foo {
   public $name;
}

mysql_connect("hostname", "user", "password");
mysql_select_db("mydb");

$result = mysql_query("select name from mytable limit 1");
$obj = mysql_fetch_object($result, 'foo');
var_dump($obj);
?>
```

Notes

Note

Performance

Speed-wise, the function is identical to mysql_fetch_array(), and almost as quick as mysql_fetch_row() (the difference is insignificant).

Note

mysql_fetch_object() is similar to mysql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Note
Field names returned by this function are *case-sensitive*.

Note
This function sets NULL fields to the PHP **NULL** value.

See Also

- mysql_fetch_array()
- mysql_fetch_assoc()
- mysql_fetch_row()
- mysql_data_seek()
- mysql_query()

mysql_fetch_row

mysql_fetch_row -- Get a result row as an enumerated array

Description

array **mysql_fetch_row** (resource `$result`)

Returns a numerical array that corresponds to the fetched row and moves the internal data pointer ahead.

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

Return Values

Returns an numerical array of strings that corresponds to the fetched row, or **FALSE** if there are no more rows.

mysql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Examples

Example #26 - Fetching one row with mysql_fetch_row()

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}
$row = mysql_fetch_row($result);

echo $row[0]; // 42
echo $row[1]; // the email value
?>
```

Notes

Note
This function sets NULL fields to the PHP **NULL** value.

See Also

- mysql_fetch_array()
- mysql_fetch_assoc()
- mysql_fetch_object()
- mysql_data_seek()
- mysql_fetch_lengths()
- mysql_result()

mysql_field_flags

mysql_field_flags -- Get the flags associated with the specified field in a result

Description

string **mysql_field_flags** (resource `$result`, int `$field_offset`)

mysql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using explode().

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

`field_offset`
The numerical field offset. The *`field_offset`* starts at *0*. If *`field_offset`* does not exist, an error of level **E_WARNING** is also issued.

Return Values

Returns a string of flags associated with the result, or **FALSE** on failure.

The following flags are reported, if your version of MySQL is current enough to support them: *"not_null"*, *"primary_key"*, *"unique_key"*, *"multiple_key"*, *"blob"*, *"unsigned"*, *"zerofill"*, *"binary"*, *"enum"*, *"auto_increment"* and *"timestamp"*.

Examples

Example #27 - A mysql_field_flags() example

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}
$flags = mysql_field_flags($result, 0);

echo $flags;
print_r(explode(' ', $flags));
?>
```

The above example will output something similar to:

```
not_null primary_key auto_increment
Array
(
   [0] => not_null
   [1] => primary_key
   [2] => auto_increment
)
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_fieldflags()**

See Also

- mysql_field_type()
- mysql_field_len()

mysql_field_len

mysql_field_len -- Returns the length of the specified field

Description

int **mysql_field_len** (resource `$result`, int `$field_offset`)

mysql_field_len() returns the length of the specified field.

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

`field_offset`
The numerical field offset. The *`field_offset`* starts at *0*. If *`field_offset`* does not exist, an error of level **E_WARNING** is also issued.

Return Values

The length of the specified field index on success, or **FALSE** on failure.

Examples

Example #28 - mysql_field_len() example

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}

// Will get the length of the id field as specified in the database
// schema.
$length = mysql_field_len($result, 0);
echo $length;
?>
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_fieldlen()**

See Also

- mysql_fetch_lengths()
- strlen()

mysql_field_name

mysql_field_name -- Get the name of the specified field in a result

Description

string **mysql_field_name** (resource `$result`, int `$field_offset`)

mysql_field_name() returns the name of the specified field index.

Parameters

result
: The result resource that is being evaluated. This result comes from a call to mysql_query().

field_offset
: The numerical field offset. The *field_offset* starts at *0*. If *field_offset* does not exist, an error of level **E_WARNING** is also issued.

Return Values

The name of the specified field index on success, or **FALSE** on failure.

Examples

Example #29 - mysql_field_name() example

```
<?php
/* The users table consists of three fields:
*   user_id
*   username
*   password.
*/
$link = @mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect to MySQL server: ' . mysql_error());
}
$dbname = 'mydb';
$db_selected = mysql_select_db($dbname, $link);
if (!$db_selected) {
   die("Could not set $dbname: " . mysql_error());
}
$res = mysql_query('select * from users', $link);

echo mysql_field_name($res, 0) . "\n";
echo mysql_field_name($res, 2);
```

```
?>
```

The above example will output:

```
user_id
password
```

Notes

Note
Field names returned by this function are *case-sensitive*.

Note
For backward compatibility, the following deprecated alias may be used: **mysql_fieldname()**

See Also

- mysql_field_type()
- mysql_field_len()

mysql_field_seek

mysql_field_seek -- Set result pointer to a specified field offset

Description

bool **mysql_field_seek** (resource `$result`, int `$field_offset`)

Seeks to the specified field offset. If the next call to mysql_fetch_field() doesn't include a field offset, the field offset specified in mysql_field_seek() will be returned.

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

`field_offset`
The numerical field offset. The *`field_offset`* starts at *0*. If *`field_offset`* does not exist, an error of level **E_WARNING** is also issued.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- mysql_fetch_field()

mysql_field_table

mysql_field_table -- Get name of the table the specified field is in

Description

string **mysql_field_table** (resource `$result`, int `$field_offset`)

Returns the name of the table that the specified field is in.

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

`field_offset`
The numerical field offset. The *`field_offset`* starts at *0*. If *`field_offset`* does not exist, an error of level **E_WARNING** is also issued.

Return Values

The name of the table on success.

Examples

Example #30 - A mysql_field_table() example

```
<?php

$query = "SELECT account.*, country.* FROM account, country WHERE
country.name = 'Portugal' AND account.country_id = country.id";

// get the result from the DB
$result = mysql_query($query);

// Lists the table name and then the field name
for ($i = 0; $i < mysql_num_fields($result); ++$i) {
   $table = mysql_field_table($result, $i);
   $field = mysql_field_name($result, $i);

   echo  "$table: $field\n";
}

?>
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_fieldtable()**

See Also

- mysql_list_tables()

mysql_field_type

mysql_field_type -- Get the type of the specified field in a result

Description

string **mysql_field_type** (resource `$result`, int `$field_offset`)

mysql_field_type() is similar to the mysql_field_name() function. The arguments are identical, but the field type is returned instead.

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

`field_offset`
The numerical field offset. The *`field_offset`* starts at *0*. If *`field_offset`* does not exist, an error of level **E_WARNING** is also issued.

Return Values

The returned field type will be one of *"int"*, *"real"*, *"string"*, *"blob"*, and others as detailed in the » MySQL documentation.

Examples

Example #31 - mysql_field_type() example

```
<?php
mysql_connect("localhost", "mysql_username", "mysql_password");
mysql_select_db("mysql");
$result = mysql_query("SELECT * FROM func");
$fields = mysql_num_fields($result);
$rows   = mysql_num_rows($result);
$table  = mysql_field_table($result, 0);
echo "Your '" . $table . "' table has " . $fields . " fields and " . $rows .
" record(s)\n";
echo "The table has the following fields:\n";
for ($i=0; $i < $fields; $i++) {
   $type  = mysql_field_type($result, $i);
   $name  = mysql_field_name($result, $i);
   $len   = mysql_field_len($result, $i);
   $flags = mysql_field_flags($result, $i);
   echo $type . " " . $name . " " . $len . " " . $flags . "\n";
}
```

```
mysql_free_result($result);
mysql_close();
?>
```

The above example will output something similar to:

```
Your 'func' table has 4 fields and 1 record(s)
The table has the following fields:
string name 64 not_null primary_key binary
int ret 1 not_null
string dl 128 not_null
string type 9 not_null enum
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_fieldtype()**

See Also

- mysql_field_name()
- mysql_field_len()

mysql_free_result

mysql_free_result -- Free result memory

Description

bool **mysql_free_result** (resource `$result`)

mysql_free_result() will free all memory associated with the result identifier *`result`*.

mysql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

If a non-resource is used for the *`result`*, an error of level E_WARNING will be emitted. It's worth noting that mysql_query() only returns a resource for SELECT, SHOW, EXPLAIN, and DESCRIBE queries.

Examples

Example #32 - A mysql_free_result() example

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}
/* Use the result, assuming we're done with it afterwards */
$row = mysql_fetch_assoc($result);

/* Now we free up the result and continue on with our script */
mysql_free_result($result);

echo $row['id'];
echo $row['email'];
?>
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_freeresult()**

See Also

- mysql_query()
- is_resource()

mysql_get_client_info

mysql_get_client_info -- Get MySQL client info

Description

string **mysql_get_client_info** (void)

mysql_get_client_info() returns a string that represents the client library version.

Return Values

The MySQL client version.

Examples

Example #33 - mysql_get_client_info() example

```
<?php
printf("MySQL client info: %s\n", mysql_get_client_info());
?>
```

The above example will output something similar to:

```
MySQL client info: 3.23.39
```

See Also

- mysql_get_host_info()
- mysql_get_proto_info()
- mysql_get_server_info()

mysql_get_host_info

mysql_get_host_info -- Get MySQL host info

Description

string **mysql_get_host_info** ([resource `$link_identifier`])

Describes the type of connection in use for the connection, including the server host name.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns a string describing the type of MySQL connection in use for the connection or **FALSE** on failure.

Examples

Example #34 - mysql_get_host_info() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
printf("MySQL host info: %s\n", mysql_get_host_info());
?>
```

The above example will output something similar to:

```
MySQL host info: Localhost via UNIX socket
```

See Also

- mysql_get_client_info()

- mysql_get_proto_info()
- mysql_get_server_info()

mysql_get_proto_info

mysql_get_proto_info -- Get MySQL protocol info

Description

int **mysql_get_proto_info** ([resource `$link_identifier`])

Retrieves the MySQL protocol.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns the MySQL protocol on success, or **FALSE** on failure.

Examples

Example #35 - mysql_get_proto_info() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
printf("MySQL protocol version: %s\n", mysql_get_proto_info());
?>
```

The above example will output something similar to:

```
MySQL protocol version: 10
```

See Also

- mysql_get_client_info()
- mysql_get_host_info()

- mysql_get_server_info()

mysql_get_server_info

mysql_get_server_info -- Get MySQL server info

Description

string **mysql_get_server_info** ([resource `$link_identifier`])

Retrieves the MySQL server version.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns the MySQL server version on success, or **FALSE** on failure.

Examples

Example #36 - mysql_get_server_info() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
printf("MySQL server version: %s\n", mysql_get_server_info());
?>
```

The above example will output something similar to:

```
MySQL server version: 4.0.1-alpha
```

See Also

- mysql_get_client_info()
- mysql_get_host_info()

- mysql_get_proto_info()
- phpversion()

mysql_info

mysql_info -- Get information about the most recent query

Description

string **mysql_info** ([resource `$link_identifier`])

Returns detailed information about the last query.

Parameters

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns information about the statement on success, or **FALSE** on failure. See the example below for which statements provide information, and what the returned value may look like. Statements that are not listed will return **FALSE**.

Examples

Example #37 - Relevant MySQL Statements

Statements that return string values. The numbers are only for illustrating purpose; their values will correspond to the query.

```
INSERT INTO ... SELECT ...
String format: Records: 23 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
String format: Records: 37 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
String format: Records: 42 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
String format: Records: 60 Duplicates: 0 Warnings: 0
UPDATE
String format: Rows matched: 65 Changed: 65 Warnings: 0
```

Notes

Note
mysql_info() returns a non- **FALSE** value for the INSERT ... VALUES statement only if multiple value lists are specified in the statement.

See Also

- mysql_affected_rows()
- mysql_insert_id()
- mysql_stat()

mysql_insert_id

mysql_insert_id -- Get the ID generated from the previous INSERT operation

Description

int **mysql_insert_id** ([resource `$link_identifier`])

Retrieves the ID generated for an AUTO_INCREMENT column by the previous INSERT query.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

The ID generated for an AUTO_INCREMENT column by the previous INSERT query on success, *0* if the previous query does not generate an AUTO_INCREMENT value, or **FALSE** if no MySQL connection was established.

Examples

Example #38 - mysql_insert_id() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
mysql_select_db('mydb');

mysql_query("INSERT INTO mytable (product) values ('kossu')");
printf("Last inserted record has id %d\n", mysql_insert_id());
?>
```

Notes

Caution

mysql_insert_id() converts the return type of the native MySQL C API function *mysql_insert_id()* to a type of *long* (named int in PHP). If your AUTO_INCREMENT column has a column type of BIGINT, the value returned by mysql_insert_id() will be incorrect. Instead, use the internal MySQL SQL function *LAST_INSERT_ID()* in an SQL query.

Note

Because mysql_insert_id() acts on the last performed query, be sure to call mysql_insert_id() immediately after the query that generates the value.

Note

The value of the MySQL SQL function *LAST_INSERT_ID()* always contains the most recently generated AUTO_INCREMENT value, and is not reset between queries.

See Also

- mysql_query()
- mysql_info()

mysql_list_dbs

mysql_list_dbs -- List databases available on a MySQL server

Description

resource **mysql_list_dbs** ([resource `$link_identifier`])

Returns a result pointer containing the databases available from the current mysql daemon.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns a result pointer resource on success, or **FALSE** on failure. Use the mysql_tablename() function to traverse this result pointer, or any function for result tables, such as mysql_fetch_array().

Examples

Example #39 - mysql_list_dbs() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
?>
```

The above example will output something similar to:

```
database1
database2
database3
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_listdbs()**

See Also

- mysql_db_name()
- mysql_select_db()

mysql_list_fields

mysql_list_fields -- List MySQL table fields

Description

resource **mysql_list_fields** (string `$database_name`, string `$table_name` [, resource `$link_identifier`])

Retrieves information about the given table name.

This function is deprecated. It is preferable to use mysql_query() to issue a SQL *SHOW COLUMNS FROM table [LIKE 'name']* statement instead.

Parameters

`database_name`
: The name of the database that's being queried.

`table_name`
: The name of the table that's being queried.

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

A result pointer resource on success, or **FALSE** on failure.

The returned result can be used with mysql_field_flags(), mysql_field_len(), mysql_field_name() and mysql_field_type().

Examples

Example #40 - Alternate to deprecated mysql_list_fields()

```
<?php
$result = mysql_query("SHOW COLUMNS FROM sometable");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}
if (mysql_num_rows($result) > 0) {
```

```
    while ($row = mysql_fetch_assoc($result)) {
        print_r($row);
    }
}
?>
```

The above example will output something similar to:

```
Array
(
    [Field] => id
    [Type] => int(7)
    [Null] =>
    [Key] => PRI
    [Default] =>
    [Extra] => auto_increment
)
Array
(
    [Field] => email
    [Type] => varchar(100)
    [Null] =>
    [Key] =>
    [Default] =>
    [Extra] =>
)
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_listfields()**

See Also

- mysql_field_flags()
- mysql_info()

mysql_list_processes

mysql_list_processes -- List MySQL processes

Description

resource **mysql_list_processes** ([resource `$link_identifier`])

Retrieves the current MySQL server threads.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

A result pointer resource on success, or **FALSE** on failure.

Examples

Example #41 - mysql_list_processes() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$result = mysql_list_processes($link);
while ($row = mysql_fetch_assoc($result)){
   printf("%s %s %s %s %s\n", $row["Id"], $row["Host"], $row["db"],
       $row["Command"], $row["Time"]);
}
mysql_free_result($result);
?>
```

The above example will output something similar to:

```
1 localhost test Processlist 0
4 localhost mysql sleep 5
```

See Also

- mysql_thread_id()
- mysql_stat()

mysql_list_tables

mysql_list_tables -- List tables in a MySQL database

Description

resource **mysql_list_tables** (string `$database` [, resource `$link_identifier`])

Retrieves a list of table names from a MySQL database.

This function is deprecated. It is preferable to use mysql_query() to issue a SQL *SHOW TABLES [FROM db_name] [LIKE 'pattern']* statement instead.

Parameters

`database`
The name of the database

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

A result pointer resource on success, or **FALSE** on failure.

Use the mysql_tablename() function to traverse this result pointer, or any function for result tables, such as mysql_fetch_array().

ChangeLog

Version	Description
4.3.7	This function became deprecated.

Examples

Example #42 - mysql_list_tables() alternative example

```
<?php
$dbname = 'mysql_dbname';

if (!mysql_connect('mysql_host', 'mysql_user', 'mysql_password')) {
   echo 'Could not connect to mysql';
   exit;
}

$sql = "SHOW TABLES FROM $dbname";
$result = mysql_query($sql);

if (!$result) {
   echo "DB Error, could not list tables\n";
   echo 'MySQL Error: ' . mysql_error();
   exit;
}

while ($row = mysql_fetch_row($result)) {
   echo "Table: {$row[0]}\n";
}

mysql_free_result($result);
?>
```

Notes

Note

For backward compatibility, the following deprecated alias may be used:
mysql_listtables()

See Also

- mysql_list_dbs()
- mysql_tablename()

mysql_num_fields

mysql_num_fields -- Get number of fields in result

Description

int **mysql_num_fields** (resource `$result`)

Retrieves the number of fields from a query.

Parameters

result
The result resource that is being evaluated. This result comes from a call to mysql_query().

Return Values

Returns the number of fields in the result set resource on success, or **FALSE** on failure.

Examples

Example #43 - A mysql_num_fields() example

```
<?php
$result = mysql_query("SELECT id,email FROM people WHERE id = '42'");
if (!$result) {
   echo 'Could not run query: ' . mysql_error();
   exit;
}

/* returns 2 because id,email === two fields */
echo mysql_num_fields($result);
?>
```

Notes

Note

For backward compatibility, the following deprecated alias may be used: **mysql_numfields()**

See Also

- mysql_select_db()
- mysql_query()
- mysql_fetch_field()
- mysql_num_rows()

mysql_num_rows

mysql_num_rows -- Get number of rows in result

Description

int **mysql_num_rows** (resource `$result`)

Retrieves the number of rows from a result set. This command is only valid for statements like SELECT or SHOW that return an actual result set. To retrieve the number of rows affected by a INSERT, UPDATE, REPLACE or DELETE query, use mysql_affected_rows() .

Parameters

`result`
The result resource that is being evaluated. This result comes from a call to mysql_query().

Return Values

The number of rows in a result set on success, or **FALSE** on failure.

Examples

Example #44 - mysql_num_rows() example

```
<?php

$link = mysql_connect("localhost", "mysql_user", "mysql_password");
mysql_select_db("database", $link);

$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

Notes

Note
If you use mysql_unbuffered_query(), mysql_num_rows() will not return the correct value until all the rows in the result set have been retrieved.

Note
For backward compatibility, the following deprecated alias may be used: **mysql_numrows()**

See Also

- mysql_affected_rows()
- mysql_connect()
- mysql_data_seek()
- mysql_select_db()
- mysql_query()

mysql_pconnect

mysql_pconnect -- Open a persistent connection to a MySQL server

Description

resource **mysql_pconnect** ([string `$server` [, string `$username` [, string `$password` [, int `$client_flags`]]]])

Establishes a persistent connection to a MySQL server.

mysql_pconnect() acts very much like mysql_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (mysql_close() will not close links established by mysql_pconnect()).

This type of link is therefore called 'persistent'.

Parameters

`server`
The MySQL server. It can also include a port number. e.g. "hostname:port" or a path to a local socket e.g. ":/path/to/socket" for the localhost. If the PHP directive mysql.default_host is undefined (default), then the default value is 'localhost:3306'

`username`
The username. Default value is the name of the user that owns the server process.

`password`
The password. Default value is an empty password.

`client_flags`
The *`client_flags`* parameter can be a combination of the following constants: 128 (enable *LOAD DATA LOCAL* handling), **MYSQL_CLIENT_SSL**, **MYSQL_CLIENT_COMPRESS**, **MYSQL_CLIENT_IGNORE_SPACE** or **MYSQL_CLIENT_INTERACTIVE**.

Return Values

Returns a MySQL persistent link identifier on success, or **FALSE** on failure.

ChangeLog

Version	Description
4.3.0	Added the *client_flags* parameter.
3.0.10	Added support for ":/path/to/socket" with *server*.
3.0.0	Added support for ":port" with *server*.

Notes

Note
Note, that these kind of links only work if you are using a module version of PHP. See the Persistent Database Connections section for more information.

Warning
Using persistent connections can require a bit of tuning of your Apache and MySQL configurations to ensure that you do not exceed the number of connections allowed by MySQL.

Note
You can suppress the error message on failure by prepending a @ to the function name.

See Also

- mysql_connect()
- Persistent Database Connections

mysql_ping

mysql_ping -- Ping a server connection or reconnect if there is no connection

Description

bool **mysql_ping** ([resource `$link_identifier`])

Checks whether or not the connection to the server is working. If it has gone down, an automatic reconnection is attempted. This function can be used by scripts that remain idle for a long while, to check whether or not the server has closed the connection and reconnect if necessary.

Note
Since MySQL 5.0.13, automatic reconnection feature is disabled.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns **TRUE** if the connection to the server MySQL server is working, otherwise **FALSE**.

Examples

Example #45 - A mysql_ping() example

```
<?php
set_time_limit(0);

$conn = mysql_connect('localhost', 'mysqluser', 'mypass');
$db   = mysql_select_db('mydb');

/* Assuming this query will take a long time */
$result = mysql_query($sql);
if (!$result) {
   echo 'Query #1 failed, exiting.';
   exit;
```

```
}

/* Make sure the connection is still alive, if not, try to reconnect */
if (!mysql_ping($conn)) {
   echo 'Lost connection, exiting after query #1';
   exit;
}
mysql_free_result($result);

/* So the connection is still alive, let's run another query */
$result2 = mysql_query($sql2);
?>
```

See Also

- mysql_thread_id()
- mysql_list_processes()

mysql_query

mysql_query -- Send a MySQL query

Description

resource **mysql_query** (string `$query` [, resource `$link_identifier`])

mysql_query() sends an unique query (multiple queries are not supported) to the currently active database on the server that's associated with the specified *`link_identifier`*.

Parameters

`query`
A SQL query The query string should not end with a semicolon.

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

For SELECT, SHOW, DESCRIBE, EXPLAIN and other statements returning resultset, mysql_query() returns a resource on success, or **FALSE** on error.

For other type of SQL statements, INSERT, UPDATE, DELETE, DROP, etc, mysql_query() returns **TRUE** on success or **FALSE** on error.

The returned result resource should be passed to mysql_fetch_array(), and other functions for dealing with result tables, to access the returned data.

Use mysql_num_rows() to find out how many rows were returned for a SELECT statement or mysql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

mysql_query() will also fail and return **FALSE** if the user does not have permission to access the table(s) referenced by the query.

Examples

Example #46 - Invalid Query

The following query is syntactically invalid, so mysql_query() fails and returns **FALSE**.

```
<?php
$result = mysql_query('SELECT * WHERE 1=1');
if (!$result) {
   die('Invalid query: ' . mysql_error());
}

?>
```

Example #47 - Valid Query

The following query is valid, so mysql_query() returns a resource.

```
<?php
// This could be supplied by a user, for example
$firstname = 'fred';
$lastname  = 'fox';

// Formulate Query
// This is the best way to perform a SQL query
// For more examples, see mysql_real_escape_string()
$query = sprintf("SELECT firstname, lastname, address, age FROM friends
WHERE firstname='%s' AND lastname='%s'",
   mysql_real_escape_string($firstname),
   mysql_real_escape_string($lastname));

// Perform Query
$result = mysql_query($query);

// Check result
// This shows the actual query sent to MySQL, and the error. Useful for
debugging.
if (!$result) {
   $message  = 'Invalid query: ' . mysql_error() . "\n";
   $message .= 'Whole query: ' . $query;
   die($message);
}

// Use result
// Attempting to print $result won't allow access to information in the
resource
// One of the mysql result functions must be used
// See also mysql_result(), mysql_fetch_array(), mysql_fetch_row(), etc.
while ($row = mysql_fetch_assoc($result)) {
   echo $row['firstname'];
   echo $row['lastname'];
   echo $row['address'];
   echo $row['age'];
}
```

```
// Free the resources associated with the result set
// This is done automatically at the end of the script
mysql_free_result($result);
?>
```

See Also

- mysql_connect()
- mysql_error()
- mysql_real_escape_string()
- mysql_result()
- mysql_fetch_assoc()
- mysql_unbuffered_query()

mysql_real_escape_string

mysql_real_escape_string -- Escapes special characters in a string for use in a SQL statement

Description

string **mysql_real_escape_string** (string `$unescaped_string` [, resource `$link_identifier`])

Escapes special characters in the *`unescaped_string`*, taking into account the current character set of the connection so that it is safe to place it in a mysql_query(). If binary data is to be inserted, this function must be used.

mysql_real_escape_string() calls MySQL's library function mysql_real_escape_string, which prepends backslashes to the following characters: *\x00*, *\n*, *\r*, **, *'*, *"* and *\x1a*.

This function must always (with few exceptions) be used to make data safe before sending a query to MySQL.

Parameters

`unescaped_string`
: The string that is to be escaped.

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns the escaped string, or **FALSE** on error.

Examples

Example #48 - Simple mysql_real_escape_string() example

```
<?php
// Connect
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
   OR die(mysql_error());

// Query
$query = sprintf("SELECT * FROM users WHERE user='%s' AND password='%s'",
```

```
            mysql_real_escape_string($user),
            mysql_real_escape_string($password));
?>
```

Example #49 - An example SQL Injection Attack

```
<?php
// Query database to check if there are any matching users
$query = "SELECT * FROM users WHERE user='{$_POST['username']}' AND
password='{$_POST['password']}'";
mysql_query($query);

// We didn't check $_POST['password'], it could be anything the user wanted!
For example:
$_POST['username'] = 'aidan';
$_POST['password'] = "' OR ''='";

// This means the query sent to MySQL would be:
echo $query;
?>
```

The query sent to MySQL:

```
SELECT * FROM users WHERE user='aidan' AND password='' OR ''=''
```

This would allow anyone to log in without a valid password.

Example #50 - A "Best Practice" query

Using mysql_real_escape_string() around each variable prevents SQL Injection. This example demonstrates the "best practice" method for querying a database, independent of the Magic Quotes setting.

```
<?php

if (isset($_POST['product_name']) && isset($_POST['product_description']) &&
isset($_POST['user_id'])) {
   // Connect

   $link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password');

   if(!is_resource($link)) {

       echo "Failed to connect to the server\n";
       // ... log the error properly

   } else {

       // Reverse magic_quotes_gpc/magic_quotes_sybase effects on those vars
```

```
if ON.

       if(get_magic_quotes_gpc()) {
           $product_name        = stripslashes($_POST['product_name']);
           $product_description =
stripslashes($_POST['product_description']);
       } else {
           $product_name        = $_POST['product_name'];
           $product_description = $_POST['product_description'];
       }

       // Make a safe query
       $query = sprintf("INSERT INTO products (`name`, `description`,
`user_id`) VALUES ('%s', '%s', %d)",
                   mysql_real_escape_string($product_name, $link),
                   mysql_real_escape_string($product_description, $link),
                   $_POST['user_id']);

       mysql_query($query, $link);

       if (mysql_affected_rows($link) > 0) {
           echo "Product inserted\n";
       }
   }
} else {
   echo "Fill the form properly\n";
}
?>
```

The query will now execute correctly, and SQL Injection attacks will not work.

Notes

Note

A MySQL connection is required before using mysql_real_escape_string() otherwise an error of level *E_WARNING* is generated, and **FALSE** is returned. If *`link_identifier`* isn't defined, the last MySQL connection is used.

Note

If magic_quotes_gpc is enabled, first apply stripslashes() to the data. Using this function on data which has already been escaped will escape the data twice.

Note

If this function is not used to escape data, the query is vulnerable to SQL Injection Attacks.

Note
mysql_real_escape_string() does not escape % and _. These are wildcards in MySQL if combined with *LIKE*, *GRANT*, or *REVOKE*.

See Also

- mysql_client_encoding()
- addslashes()
- stripslashes()
- The magic_quotes_gpc directive
- The magic_quotes_runtime directive

mysql_result

mysql_result -- Get result data

Description

string **mysql_result** (resource `$result`, int `$row` [, mixed `$field`])

Retrieves the contents of one cell from a MySQL result set.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than mysql_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Parameters

result
: The result resource that is being evaluated. This result comes from a call to mysql_query().

row
: The row number from the result that's being retrieved. Row numbers start at *0*.

field
: The name or offset of the field being retrieved. It can be the field's offset, the field's name, or the field's table dot field name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name. If undefined, the first field is retrieved.

Return Values

The contents of one cell from a MySQL result set on success, or **FALSE** on failure.

Examples

Example #51 - mysql_result() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
$result = mysql_query('SELECT name FROM work.employee');
```

```
if (!$result) {
   die('Could not query:' . mysql_error());
}
echo mysql_result($result, 2); // outputs third employee's name

mysql_close($link);
?>
```

Notes

Note
Calls to mysql_result() should not be mixed with calls to other functions that deal with the result set.

See Also

- mysql_fetch_row()
- mysql_fetch_array()
- mysql_fetch_assoc()
- mysql_fetch_object()

mysql_select_db

mysql_select_db -- Select a MySQL database

Description

bool **mysql_select_db** (string `$database_name` [, resource `$link_identifier`])

Sets the current active database on the server that's associated with the specified link identifier. Every subsequent call to mysql_query() will be made on the active database.

Parameters

`database_name`
The name of the database that is to be selected.

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #52 - mysql_select_db() example

```
<?php

$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
   die('Not connected : ' . mysql_error());
}

// make foo the current db
$db_selected = mysql_select_db('foo', $link);
if (!$db_selected) {
   die ('Can\'t use foo : ' . mysql_error());
}
?>
```

Notes

Note
For backward compatibility, the following deprecated alias may be used: **mysql_selectdb()**

See Also

- mysql_connect()
- mysql_pconnect()
- mysql_query()

mysql_set_charset

mysql_set_charset -- Sets the client character set

Description

bool **mysql_set_charset** (string `$charset` [, resource `$link_identifier`])

Sets the default character set for the current connection.

Parameters

`charset`
A valid character set name.

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function requires MySQL 5.0.7 or later.

See Also

- » List of character sets that MySQL supports

mysql_stat

mysql_stat -- Get current system status

Description

string **mysql_stat** ([resource `$link_identifier`])

mysql_stat() returns the current server status.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

Returns a string with the status for uptime, threads, queries, open tables, flush tables and queries per second. For a complete list of other status variables, you have to use the *SHOW STATUS* SQL command. If *`link_identifier`* is invalid, **NULL** is returned.

Examples

Example #53 - mysql_stat() example

```
<?php
$link   = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$status = explode('  ', mysql_stat($link));
print_r($status);
?>
```

The above example will output something similar to:

```
Array
(
   [0] => Uptime: 5380
   [1] => Threads: 2
   [2] => Questions: 1321299
   [3] => Slow queries: 0
   [4] => Opens: 26
   [5] => Flush tables: 1
   [6] => Open tables: 17
   [7] => Queries per second avg: 245.595
)
```

Example #54 - Alternative mysql_stat() example

```
<?php
$link   = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$result = mysql_query('SHOW VARIABLES', $link);
while ($row = mysql_fetch_assoc($result)) {
   echo $row['Variable_name'] . ' = ' . $row['Value'] . "\n";
}
?>
```

The above example will output something similar to:

```
back_log = 50
basedir = /usr/local/
bdb_cache_size = 8388600
bdb_log_buffer_size = 32768
bdb_home = /var/db/mysql/
bdb_max_lock = 10000
bdb_logdir =
bdb_shared_data = OFF
bdb_tmpdir = /var/tmp/
...
```

See Also

- mysql_get_server_info()
- mysql_list_processes()

mysql_tablename

mysql_tablename -- Get table name of field

Description

string **mysql_tablename** (resource `$result`, int `$i`)

Retrieves the table name from a *`result`*.

This function deprecated. It is preferable to use mysql_query() to issue a SQL *SHOW TABLES [FROM db_name] [LIKE 'pattern']* statement instead.

Parameters

`result`
A result pointer resource that's returned from mysql_list_tables().

`i`
The integer index (row/table number)

Return Values

The name of the table on success, or **FALSE** on failure.

Use the mysql_tablename() function to traverse this result pointer, or any function for result tables, such as mysql_fetch_array().

Examples

Example #55 - mysql_tablename() example

```
<?php
mysql_connect("localhost", "mysql_user", "mysql_password");
$result = mysql_list_tables("mydb");
$num_rows = mysql_num_rows($result);
for ($i = 0; $i < $num_rows; $i++) {
   echo "Table: ", mysql_tablename($result, $i), "\n";
}

mysql_free_result($result);
?>
```

Notes

Note
The mysql_num_rows() function may be used to determine the number of tables in the result pointer.

See Also

- mysql_list_tables()
- mysql_field_table()
- mysql_db_name()

mysql_thread_id

mysql_thread_id -- Return the current thread ID

Description

int **mysql_thread_id** ([resource `$link_identifier`])

Retrieves the current thread ID. If the connection is lost, and a reconnect with mysql_ping() is executed, the thread ID will change. This means only retrieve the thread ID when needed.

Parameters

`link_identifier`
: The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

The thread ID on success, or **FALSE** on failure.

Examples

Example #56 - mysql_thread_id() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$thread_id = mysql_thread_id($link);
if ($thread_id){
   printf("current thread id is %d\n", $thread_id);
}
?>
```

The above example will output something similar to:

```
current thread id is 73
```

See Also

- mysql_ping()
- mysql_list_processes()

mysql_unbuffered_query

mysql_unbuffered_query -- Send an SQL query to MySQL, without fetching and buffering the result rows

Description

resource **mysql_unbuffered_query** (string `$query` [, resource `$link_identifier`])

mysql_unbuffered_query() sends a SQL query *`query`* to MySQL, without fetching and buffering the result rows automatically, as mysql_query() does. On the one hand, this saves a considerable amount of memory with SQL queries that produce large result sets. On the other hand, you can start working on the result set immediately after the first row has been retrieved: you don't have to wait until the complete SQL query has been performed. When using multiple DB-connects, you have to specify the optional parameter *`link_identifier`*.

Parameters

`query`
A SQL query

`link_identifier`
The MySQL connection. If the link identifier is not specified, the last link opened by mysql_connect() is assumed. If no such link is found, it will try to create one as if mysql_connect() was called with no arguments. If by chance no connection is found or established, an **E_WARNING** level error is generated.

Return Values

For SELECT, SHOW, DESCRIBE or EXPLAIN statements, mysql_unbuffered_query() returns a resource on success, or **FALSE** on error.

For other type of SQL statements, UPDATE, DELETE, DROP, etc, mysql_unbuffered_query() returns **TRUE** on success or **FALSE** on error.

Notes

Note
The benefits of mysql_unbuffered_query() come at a cost: You cannot use mysql_num_rows() and mysql_data_seek() on a result set returned from mysql_unbuffered_query(). You also have to fetch all result rows from an unbuffered SQL query, before you can send a new SQL query to MySQL.

See Also

- mysql_query()