

# MySQL Functions

## Introduzione

Queste funzioni consentono l'accesso ai server di database MySQL. Maggiori informazioni riguardo MySQL possono essere trovate su <http://www.mysql.com/>.

La documentazione su MySQL può essere trovata su <http://dev.mysql.com/doc/>.

---

## Requisiti

Al fine di rendere queste funzioni disponibili, si deve compilare PHP con il supporto MySQL.

---

## Installazione

Usando l'opzione di configurazione `--with-mysql[=DIR]` si abilita *PHP* l'accesso ai database MySQL.

In PHP 4, il parametro `--with-mysql` è abilitato per default. Per disabilitarlo si può utilizzare `--without-mysql`. Inoltre in PHP 4, se si abilita MySQL senza indicare il percorso directory di installazione di MySQL, il *PHP* utilizzerà le librerie incluse nella distribuzione. In Windows non vi sono DLL, è semplicemente compilato nel PHP. Gli utenti che eseguano altre applicazioni basate su MySQL (come, ad esempio, auth-mysql), non dovrebbero utilizzare le librerie allegate al PHP, ma, piuttosto, indicare il percorso alla directory di installazione di MySQL, come, ad esempio; `--with-mysql=/percorso/a/mysql`. Questo forzerà *PHP* ad usare le librerie client installate da MySQL evitando ogni conflitto.

In PHP 5, MySQL non sarà più abilitato per default, e nè la libreria MySQL sarà allegata al PHP. Per maggiori dettagli sul perchè leggere: [FAQ](#).

Questo modulo di interfaccia a MySQL non supporta completamente le tutte le funzioni presenti nelle versioni di MySQL successive alla 4.1.0. Per avere il pieno supporto a queste vedere [MySQLi](#).

Se si desidera installare sia il modulo `mysql` sia il modulo `mysqli` utilizzare la stessa libreria client per evitare conflitti.

Avvertimento
Problemi di blocco e di avvio di <i>PHP</i> possono essere riscontrati quando si carica questa estensione insieme ad estensioni <code>recode</code> . Vedere anche l'estensione <a href="#">recode</a> per maggiori informazioni.

**Nota:** Se occorre utilizzare set di caratteri differenti rispetto al *latin* (il default), si deve installare la libreria esterna (non allegata al PHP) `libmysql` compilata con il supporto dei set di caratteri.

---

## Configurazione di Runtime

Il comportamento di queste funzioni è influenzato dalle impostazioni di `php.ini`.

**Tabella 1. Opzioni di configurazione di MySQL**

Nome	Predefinito	Modificabile in
<code>mysql.allow_persistent</code>	"On"	PHP_INI_SYSTEM
<code>mysql.max_persistent</code>	"-1"	PHP_INI_SYSTEM
<code>mysql.max_links</code>	"-1"	PHP_INI_SYSTEM
<code>mysql.default_port</code>	NULL	PHP_INI_ALL
<code>mysql.default_socket</code>	NULL	PHP_INI_ALL
<code>mysql.default_host</code>	NULL	PHP_INI_ALL
<code>mysql.default_user</code>	NULL	PHP_INI_ALL
<code>mysql.default_password</code>	NULL	PHP_INI_ALL
<code>mysql.connect_timeout</code>	"0"	PHP_INI_SYSTEM

Per ulteriori dettagli e definizione delle costanti `PHP_INI_*` vedere [ini\\_set\(\)](#).

Breve descrizione dei parametri di configurazione.

`mysql.allow_persistent` [boolean](#)

Determina se consentire le [connessioni persistenti](#) a MySQL.

`mysql.max_persistent` [integer](#)

Il numero massimo di connessioni persistenti MySQL per processo.

`mysql.max_links` [integer](#)

Il numero massimo di connessioni MySQL per processo, incluse le connessioni persistenti.

`mysql.default_port` [string](#)

Il numero di porta TCP predefinito da usare per connettersi ad un server di database se nessuna altra porta viene specificata. Se nessun valore predefinito è specificato, la porta sarà ottenuta dalla variabile d'ambiente `MYSQL_TCP_PORT`, dalla voce `mysql-tcp` in `/etc/services` o dalla costante `MYSQL_PORT` in fase di compilazione, in questo ordine. Win32 userà solo la costante `MYSQL_PORT`.

*mysql.default\_socket* [string](#)

Il nome del socket predefinito da usare per connettersi ad un server di database locale se nessun altro nome di socket viene specificato.

*mysql.default\_host* [string](#)

L'host di default del server da usare per connettersi al server di database se nessun altro host viene specificato. Non si applica in [safe mode](#).

*mysql.default\_user* [string](#)

Il nome utente predefinito da usare per connettersi al server di database se nessun altro nome viene specificato. Non si applica in [safe mode](#).

*mysql.default\_password* [string](#)

La password predefinita da usare per connettersi al server di database se nessuna altra password viene specificata. Non si applica in [safe mode](#).

*mysql.connect\_timeout* [integer](#)

Timeout di connessione in secondi. Per Linux questo timeout è usato anche per attendere la prima risposta dal server.

---

## Tipi di risorse

Ci sono due tipi di risorse usati nel modulo MySQL. Il primo è l'identificativo di connessione per una connessione ad un database, del secondo tipo sono le risorse che contengono i risultati di una query.

---

## Costanti predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata compilata nel PHP o se è stata caricata dinamicamente a runtime.

Fin dal PHP 4.3.0 è possibile specificare flag aggiuntivi per il client per le funzioni [mysql\\_connect\(\)](#) e [mysql\\_pconnect\(\)](#). Sono definite le seguenti costanti:

**Tabella 2. Costanti client MySQL**

Costante	Descrizione
MYSQL_CLIENT_COMPRESS	Usa la compressione del protocollo

Costante	Descrizione
ESS	
MYSQL_CLIENT_IGNORE_SPACE	Consente lo spazio dopo i nomi delle funzioni
MYSQL_CLIENT_INTERACTIVE	Lascia trascorrere interactive_timeout secondi (anziché wait_timeout) di inattività prima di chiudere la connessione

La funzione [mysql\\_fetch\\_array\(\)](#) usa una costante per i diversi tipi di array risultato. Sono definite le seguenti costanti:

**Tabella 3. Costanti caricamento MySQL**

Costante	Descrizione
MYSQL_ASSOC	Le colonne sono restituite in un array avente il nome del campo come indice dell'array
MYSQL_BOTH	Le colonne sono restituite in un array avente sia un indice numerico sia un indice costituito dal nome del campo
MYSQL_NUM	Le colonne sono restituite in un array avente un indice numerico per i campi. Questo indice inizia da 0, il primo campo nel risultato

## Esempi

Questo esempio mostra come connettersi, eseguire una query, stampare le righe risultanti e disconnettersi dal database MySQL.

### Esempio 1. Esempio dell'estensione MySQL

```
<?php
/* Connessione e selezione del database */
$connessione = mysql_connect("host_mysql", "utente_mysql", "password_mysql")
    or die("Connessione non riuscita: " . mysql_error());
print "Connesso con successo";
mysql_select_db("mio_database") or die("Selezione del database non riuscita");

/* Esecuzione di una query SQL */
$query = "SELECT * FROM mia_tabella";
$resultato = mysql_query($query) or die("Query fallita: " . mysql_error() );

/* Stampa dei risultati in HTML */
echo "<table>\n";
while ($linea = mysql_fetch_array($resultato, MYSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($linea as $valore_colonna) {
        echo "\t\t<td>$valore_colonna</td>\n";
    }
    echo "\t</tr>\n";
}
print "</table>\n";

/* Liberazione delle risorse del risultato */
mysql_free_result($resultato);
```

```
/* Chiusura della connessione */  
mysql_close($connessione);  
?>
```

## Sommario

[mysql\\_affected\\_rows](#) -- Ottiene il numero di righe coinvolte nelle precedenti operazioni MySQL

[mysql\\_change\\_user](#) -- Cambia l'utente della connessione attiva

[mysql\\_client\\_encoding](#) -- Restituisce il nome del set di caratteri

[mysql\\_close](#) -- Chiude una connessione MySQL

[mysql\\_connect](#) -- Apre una connessione ad un server MySQL

[mysql\\_create\\_db](#) -- Crea un database MySQL

[mysql\\_data\\_seek](#) -- Muove il puntatore interno del risultato

[mysql\\_db\\_name](#) -- Ottiene dei dati dal risultato

[mysql\\_db\\_query](#) -- Invia una query MySQL

[mysql\\_drop\\_db](#) -- Elimina (cancella) un database MySQL

[mysql\\_errno](#) -- Restituisce il valore numerico del messaggio di errore della precedente operazione MySQL

[mysql\\_error](#) -- Restituisce il testo del messaggio di errore della precedente operazione MySQL

[mysql\\_escape\\_string](#) -- Aggiunge le sequenze di escape in una stringa per l'uso in `mysql_query`.

[mysql\\_fetch\\_array](#) -- Carica una riga del risultato come un array associativo, un array numerico o entrambi.

[mysql\\_fetch\\_assoc](#) -- Carica una riga del risultato come array associativo

[mysql\\_fetch\\_field](#) -- Ottiene informazioni sulla colonna da un risultato e le restituisce come oggetto

[mysql\\_fetch\\_lengths](#) -- Ottiene la lunghezza di ogni output nel risultato

[mysql\\_fetch\\_object](#) -- Carica una riga del risultato come un oggetto

[mysql\\_fetch\\_row](#) -- Ottiene una riga del risultato come un array enumerato

[mysql\\_field\\_flags](#) -- Ottiene i flag associati al campo specificato di un risultato

[mysql\\_field\\_len](#) -- Restituisce la lunghezza del campo specificato

[mysql\\_field\\_name](#) -- Ottiene il nome del campo specificato in un risultato

[mysql\\_field\\_seek](#) -- Imposta il puntatore al risultato ad un determinato indice di campo

[mysql\\_field\\_table](#) -- Ottiene il nome della tabella in cui si trova il campo specificato

[mysql\\_field\\_type](#) -- Ottiene il tipo del campo specificato in un risultato

[mysql\\_free\\_result](#) -- Libera la memoria occupata dal risultato

[mysql\\_get\\_client\\_info](#) -- Ottiene informazioni sul client MySQL

[mysql\\_get\\_host\\_info](#) -- Ottiene le informazioni sull'host MySQL

[mysql\\_get\\_proto\\_info](#) -- Ottiene le informazioni sul protocollo MySQL

[mysql\\_get\\_server\\_info](#) -- Ottiene le informazioni sul server MySQL

[mysql\\_info](#) -- Ottiene le informazioni relative alla query più recente.

[mysql\\_insert\\_id](#) -- Ottiene l'identificativo generato dalla precedente operazione INSERT

[mysql\\_list\\_dbs](#) -- Elenca i database disponibili in un server MySQL

[mysql\\_list\\_fields](#) -- Elenca i campi di un risultato MySQL

[mysql\\_list\\_processes](#) -- Elenca i processi MySQL

[mysql\\_list\\_tables](#) -- Elenca le tabelle in un database MySQL

[mysql\\_num\\_fields](#) -- Ottiene il numero di campi nel risultato

[mysql\\_num\\_rows](#) -- Ottiene il numero di righe in un risultato

[mysql\\_pconnect](#) -- Apre una connessione persistente ad un server MySQL

[mysql\\_ping](#) -- Esegue un ping su una connessione al server o riconnette se non esiste la connessione

[mysql\\_query](#) -- Invia una query MySQL

[mysql\\_real\\_escape\\_string](#) -- Aggiunge le sequenze di escape ai caratteri speciali in una stringa per

l'uso in una istruzione SQL, tenendo conto dell'attuale set di caratteri della connessione.

[mysql\\_result](#) -- Ottiene i dati dal risultato

[mysql\\_select\\_db](#) -- Seleziona un database MySQL

[mysql\\_stat](#) -- Ottiene l'attuale stato del sistema

[mysql\\_tablename](#) -- Ottiene il nome della tabella

[mysql\\_thread\\_id](#) -- Restituisce l'attuale thread ID

[mysql\\_unbuffered\\_query](#) -- Invia una query SQL a MySQL senza caricare e bufferare le righe risultanti

## mysql\_affected\_rows

(PHP 3, PHP 4, PHP 5)

`mysql_affected_rows` -- Ottiene il numero di righe coinvolte nelle precedenti operazioni MySQL

### Descrizione

`int mysql_affected_rows ( [resource identificativo_connesione] )`

`mysql_affected_rows()` restituisce il numero di righe coinvolte nell'ultima query INSERT, UPDATE o DELETE associata a *identificativo\_connesione*. Se l'identificativo di connessione non è specificato, viene considerata l'ultima connessione aperta con [mysql\\_connect\(\)](#).

**Nota:** Se sono usate le transazioni, è necessario richiamare `mysql_affected_rows()` dopo le query INSERT, UPDATE, o DELETE e non dopo il commit.

Se l'ultima query era una query DELETE senza clausola WHERE, tutti i record saranno cancellati dalla tabella ma questa funzione restituirà zero.

**Nota:** Usando UPDATE, MySQL non aggiornerà le colonne nelle quali il nuovo valore è uguale al vecchio valore. Questo crea la possibilità che `mysql_affected_rows()` può non uguagliare realmente il numero di righe corrispondenti ma solo il numero di righe effettivamente coinvolte dalla query.

`mysql_affected_rows()` non funziona con l'istruzione SELECT ma solo con le istruzioni che modificano i record. Per ricavare il numero di righe restituite da SELECT, usare [mysql\\_num\\_rows\(\)](#).

Se l'ultima query fallisce, questa funzione restituisce -1.

#### Esempio 1. Query di eliminazione

```
<?php
/* connessione al database */
mysql_pconnect("localhost", "utente_mysql", "password_mysql") or
die("Connessione non riuscita: " . mysql_error());

/* questo dovrebbe restituire il numero corretto di record eliminati */
```

```
mysql_query("DELETE FROM mia_tabella WHERE id < 10");
printf ("Records eliminati: %d\n", mysql_affected_rows());

/* senza la clausola WHERE nell'istruzione DELETE, dovrebbe restituire 0 */
mysql_query("DELETE FROM mia_tabella");
printf ("Record eliminati: %d\n", mysql_affected_rows());
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
Records eliminati: 10
Records eliminati: 0
```

## Esempio 2. Query di aggiornamento

```
<?php
/* connessione al to database */
mysql_pconnect("localhost", "utente_mysql", "password_mysql") or
die("Connessione non riuscita: " . mysql_error());

/* aggiornamento dei record */
mysql_query("UPDATE mia_tabella SET used=1 WHERE id < 10");
printf ("Record aggiornati: %d\n", mysql_affected_rows());
mysql_query("COMMIT");
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
Record aggiornati: 10
```

Vedere anche: [mysql\\_num\\_rows\(\)](#), [mysql\\_info\(\)](#).

# mysql\_change\_user

(PHP 3 >= 3.0.13)

mysql\_change\_user -- Cambia l'utente della connessione attiva

## Descrizione

int **mysql\_change\_user** ( string nome\_utente, string password [, string database [, resource identificativo\_connessione]] )

**mysql\_change\_user()** cambia l'utente dell'attuale connessione attiva o della connessione specificata dal parametro opzionale *identificativo\_connessione*. Se un database is specificato, questo sarà il database corrente dopo che l'utente è stato cambiato. Se l'autorizzazione del nuovo utente e password fallisce, l'attuale utente connesso rimane attivo. Restituisce **TRUE** in caso di successo, **FALSE** in caso di fallimento.

**Nota:** Questa funzione è stata introdotta nel PHP 3.0.13 e richiede MySQL 3.23.3 o successivo. Non è disponibile nel PHP 4.

# mysql\_client\_encoding

(PHP 4 >= 4.3.0, PHP 5)

mysql\_client\_encoding -- Restituisce il nome del set di caratteri

## Descrizione

int **mysql\_client\_encoding** ( [resource identificativo\_connessione] )

**mysql\_client\_encoding()** restituisce il nome del set di caratteri predefinito per l'attuale connessione.

### Esempio 1. Esempio di mysql\_client\_encoding()

```
<?php
$connessione = mysql_connect('localhost', 'utente_mysql', 'password_mysql');
$charset = mysql_client_encoding($connessione);
printf ("il set di carattei attuale è %s\n", $charset);
?>
```

L'esempio riportato sopra produce il seguente output:

```
il set di caratteri attuale è latin1
```

Vedere anche: [mysql\\_real\\_escape\\_string\(\)](#)

# mysql\_close

(PHP 3, PHP 4, PHP 5)

mysql\_close -- Chiude una connessione MySQL

## Descrizione

bool **mysql\_close** ( [resource identificativo\_connessione] )

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

**mysql\_close()** chiude la connessione al server MySQL associata all'identificativo di connessione specificato. Se *identificativo\_connessione* non è specificato, viene usata l'ultima connessione aperta.

L'uso di **mysql\_close()** non è normalmente necessario, dal momento che le connessioni non persistenti sono chiuse automaticamente alla fine dell'esecuzione dello script. Vedere anche [freeing resources](#).

**Nota:** `mysql_close()` non chiude le connessioni persistenti create da [mysql\\_pconnect\(\)](#).

### Esempio 1. Esempio di chiura connessione MySQL

```
<?php
    $connessione = mysql_connect("localhost", "utente_mysql", "password_mysql")
        or die("Connessione non riuscita: " . mysql_error());
    print ("Connesso con successo");
    mysql_close($connessione);
?>
```

Vedere anche: [mysql\\_connect\(\)](#), e [mysql\\_pconnect\(\)](#).

## mysql\_connect

(PHP 3, PHP 4, PHP 5)

`mysql_connect` -- Apre una connessione ad un server MySQL

### Descrizione

resource **mysql\_connect** ( [string server [, string nome\_utente [, string password [, bool nuova\_connessione [, int client\_flags]]]] )

Restituisce un identificativo di connessione MySQL in caso di successo oppure **FALSE** in caso di fallimento.

**mysql\_connect()** stabilisce una connessione ad un server MySQL. I seguenti valore predefiniti sono assunti per i parametri opzionali mancanti: *server* = 'localhost:3306', *nome\_utente* = nome dell'utente proprietario del processo server e *password* = password vuota.

Il parametro *server* può anche includere un numero di porta. Es. "hostname:porta" o un percorso ad un socket es. ":/percorso/al/socket" per il localhost.

**Nota:** il supporto per " :porta" è stato aggiunto nel PHP 3.0B4.

Il supporto per " :/percorso/al/socket" è stato aggiunto nel PHP 3.0.10.

Si può eliminare il messaggio di errore nel caso di fallimento aggiungendo il prefisso [@](#) al un nome della funzione.

Se si fa una seconda chiamata a **mysql\_connect()** con gli stessi argomenti, nessuna nuova connessione sarà stabilita, ma sarà restituito l'identificativo della connessione già aperta. Il parametro *nuova\_connessione* modifica questo comportamento e fa sì che **mysql\_connect()** apra sempre una nuova connessione, anche se **mysql\_connect()** era stata chiamata prima con gli stessi parametri. il parametro *client\_flags* può essere combinato con le costanti `MYSQL_CLIENT_COMPRESS`, `MYSQL_CLIENT_IGNORE_SPACE` o

MYSQL\_CLIENT\_INTERACTIVE.

**Nota:** Il parametro *nuova\_connessione* è stato introdotto dal PHP 4.2.0

Il parametro *client\_flags* è stato introdotto dal PHP 4.3.0

La connessione al server sarà chiusa prima della fine dell'esecuzione dello script, a meno che questa non sia precedentemente chiusa esplicitamente richiamando [mysql\\_close\(\)](#).

### Esempio 1. Esempio di connessione MySQL

```
<?php
    $connessione = mysql_connect("localhost", "utente_mysql", "password_mysql")
        or die("Connessione non riuscita: " . mysql_error());
    print ("Connesso con successo");
    mysql_close($connessione);
?>
```

Vedere anche [mysql\\_pconnect\(\)](#) e [mysql\\_close\(\)](#).

## mysql\_create\_db

(PHP 3, PHP 4, PHP 5)

mysql\_create\_db -- Crea un database MySQL

### Descrizione

bool **mysql\_create\_db** ( string nome\_database [, resource identificativo\_connessione] )

**mysql\_create\_db()** tenta di creare un nuovo database nel server associato all'identificativo di connessione specificato.

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

### Esempio 1. Esempio creazione database MySQL

```
<?php
    $connessione = mysql_pconnect("localhost", "utente_mysql", "password_mysql")
        or die("Connessione non riuscita: " . mysql_error());

    if (mysql_create_db("mio_db")) {
        print ("Database creato con successo\n");
    } else {
        printf ("Errore nella creazione del database: %s\n", mysql_error());
    }
?>
```

Per motivi di compatibilità con il passato, anche **mysql\_createdb()** può essere usata. Questo è

comunque sconsigliato.

**Nota:** La funzione `mysql_create_db()` è sconsigliata. Al suo posto è preferibile usare [mysql\\_query\(\)](#) per inviare un'istruzione SQL CREATE DATABASE.

Vedere anche: [mysql\\_drop\\_db\(\)](#), [mysql\\_query\(\)](#).

## mysql\_data\_seek

(PHP 3, PHP 4, PHP 5)

`mysql_data_seek` -- Muove il puntatore interno del risultato

### Descrizione

bool **mysql\_data\_seek** ( resource identificativo\_risultato, int numero\_riga )

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

**mysql\_data\_seek()** muove il puntatore di riga interno del risultato MySQL associato all'identificativo specificato per puntare ad un determinato numero di riga. La successiva chiamata a [mysql\\_fetch\\_row\(\)](#) dovrebbe restituire questa riga.

*numero\_riga* inizia da 0. *numero\_riga* dovrebbe essere un valore nell'intervallo che va da 0 a `mysql_num_rows - 1`.

**Nota:** La funzione **mysql\_data\_seek()** può essere usata solo insieme a [mysql\\_query\(\)](#), non con [mysql\\_unbuffered\\_query\(\)](#).

### Esempio 1. Esempio seek dati MySQL

```
<?php
    $connessione = mysql_pconnect("localhost", "utente_mysql", "password_mysql")
        or die("Connessione non riuscita: " . mysql_error());

    mysql_select_db("samp_db")
        or die("Selezione del database non riuscita: " . mysql_error());

    $query = "SELECT cognome, nome FROM amici";
    $risultato = mysql_query($query)
        or die("Query fallita: " . mysql_error());

    /* caricamento righe in ordine inverso */
    for ($i = mysql_num_rows($risultato) - 1; $i >= 0; $i--) {
        if (!mysql_data_seek($risultato, $i)) {
            echo "Non si può eseguire il seek alla riga $i: " . mysql_error() . "\n";
            continue;
        }

        if(!($riga = mysql_fetch_object($risultato)))
```

```
        continue;

        echo "$riga->cognome $riga->nome<br />\n";
    }

    mysql_free_result($risultato);
?>
```

Vedere anche: [mysql\\_query\(\)](#), [mysql\\_num\\_rows\(\)](#).

## mysql\_db\_name

(PHP 3 >= 3.0.6, PHP 4, PHP 5)

mysql\_db\_name -- Ottiene dei dati dal risultato

### Descrizione

string **mysql\_db\_name** ( resource risultato, int riga [, mixed campo] )

**mysql\_db\_name()** accetta come primo parametro il puntatore al risultato dalla chiamata a [mysql\\_list\\_dbs\(\)](#). Il parametro *riga* è un indice compreso nell'intervallo del risultato.

Se intercorre un errore, viene restituito **FALSE**. Usare [mysql\\_errno\(\)](#) e [mysql\\_error\(\)](#) per determinare la natura dell'errore.

#### Esempio 1. Esempio di mysql\_db\_name()

```
<?php
    error_reporting(E_ALL);

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

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

Per motivi di compatibilità con il passato, anche **mysql\_dbname()** è accettata. Questo comunque è sconsigliato.

## mysql\_db\_query

(PHP 3, PHP 4, PHP 5)

mysql\_db\_query -- Invia una query MySQL

## Descrizione

resource **mysql\_db\_query** ( string database, string query [, resource identificativo\_connessione] )

Restituisce una risorsa risultato della query se è stato possibile eseguire quest'ultima, oppure **FALSE** in caso d'errore. La funzione restituisce **TRUE/FALSE** anche per le query *INSERT/UPDATE/DELETE* per indicarne il successo/fallimento.

**mysql\_db\_query()** seleziona un database ed esegue una query su di esso. Se l'identificativo di connessione opzionale non è specificato, la funzione proverà a cercare una connessione aperta al server MySQL e se tale connessione non viene trovata cercherà di crearne una come se [mysql\\_connect\(\)](#) fosse chiamata senza argomenti.

Si informa che questa funzione **NON** commuta al database al quale si era connessi prima. In altre parole, non si può usare questa funzione per eseguire *temporaneamente* una query sql su un altro database, si deve commutare manualmente. Gli utenti sono fortemente incoraggiati ad usare la sintassi *database.tabella* nelle loro query sql in questa funzione.

Vedere anche [mysql\\_connect\(\)](#) e [mysql\\_query\(\)](#).

**Nota:** Questa funzione è stata sconsigliata a partire dal PHP 4.0.6. Non usare questa funzione. Usare invece [mysql\\_select\\_db\(\)](#) e [mysql\\_query\(\)](#).

## mysql\_drop\_db

(PHP 3, PHP 4, PHP 5)

mysql\_drop\_db -- Elimina (cancella) un database MySQL

## Descrizione

bool **mysql\_drop\_db** ( string nome\_database [, resource identificativo\_connessione] )

Restituisce **TRUE** in caso di successo, **FALSE** in caso di fallimento.

**mysql\_drop\_db()** tenta di eliminare (cancellare) un intero database dal server associato all'identificativo di connessione specificato.

Per motivi di compatibilità con il passato, anche **mysql\_dropdb()** può essere usato. Questa è comunque sconsigliato.

**Nota:** La funzione **mysql\_drop\_db()** è sconsigliata. Al suo posto è preferibile usare

[mysql\\_query\(\)](#) per inviare una istruzione SQL DROP DATABASE.

Vedere anche: [mysql\\_create\\_db\(\)](#), [mysql\\_query\(\)](#).

## mysql\_errno

(PHP 3, PHP 4, PHP 5)

mysql\_errno -- Restituisce il valore numerico del messaggio di errore della precedente operazione MySQL

### Descrizione

int **mysql\_errno** ( [resource identificativo\_connesione] )

Restituisce il numero di errore dall'ultima funzione MySQL, oppure 0 (zero) se nessun errore è intercorso.

Gli errori ritornano dal database MySQL senza visualizzare i messaggi di avvertimento. Usando invece **mysql\_errno()** si recupera il codice di errore. Notare che questa funzione restituisce solo il codice errore della più recente funzione MySQL eseguita (ad esclusione di [mysql\\_error\(\)](#) e **mysql\_errno()**), quindi se la si vuole usare, assicurarsi di controllare il valore prima di richiamare un'altra funzione MySQL.

#### Esempio 1. Esempio di mysql\_errno

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql");

mysql_select_db("db_inesistente");
echo mysql_errno() . ": " . mysql_error() . "\n";

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

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
1049: Unknown database 'db_inesistente'
1146: Table 'kossu.tabella_inesistente' doesn't exist
```

**Nota:** Se l'argomento opzionale è specificato la connessione indicata viene usata per recuperare il codice d'errore. Altrimenti viene usata l'ultima connessione aperta.

Vedere anche: [mysql\\_error\(\)](#)

# mysql\_error

(PHP 3, PHP 4, PHP 5)

mysql\_error -- Restituisce il testo del messaggio di errore della precedente operazione MySQL

## Descrizione

string **mysql\_error** ( [resource identificativo\_connessione] )

Restituisce il testo dell'errore dall'ultima funzione MySQL, oppure "" (la stringa vuota) se nessun errore intercorre.

Gli errori ritornano dal database MySQL senza visualizzare i messaggi di avvertimento. Si usa invece **mysql\_error()** per recuperare il testo dell'errore. Notare che questa funzione restituisce solo il testo dell'errore della più recente funzione MySQL eseguita (ad esclusione di **mysql\_error()** e [mysql\\_errno\(\)](#)), quindi se la si vuole usare, assicurarsi di controllare il valore prima di richiamare un'altra funzione MySQL.

### Esempio 1. mysql\_error Example

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql");

mysql_select_db("db_inesistente");
echo mysql_errno() . ": " . mysql_error() . "\n";

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

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
1049: Unknown database 'db_inesistente'
1146: Table 'kossu.tabella_inesistente' doesn't exist
```

**Nota:** Se l'argomento opzionale è specificato la connessione indicata viene usata per recuperare il codice d'errore. Altrimenti viene usata l'ultima connessione aperta.

Vedere anche: [mysql\\_errno\(\)](#)

## mysql\_escape\_string

(PHP 4 >= 4.0.3, PHP 5)

mysql\_escape\_string -- Aggiunge le sequenze di escape in una stringa per l'uso in mysql\_query.

## Descrizione

string **mysql\_escape\_string** ( string stringa\_senza\_escape )

Questa funzione aggiunge le sequenze di escape a *stringa\_senza\_escape*, in modo che sia sicuro usarla in [mysql\\_query\(\)](#).

**Nota:** `mysql_escape_string()` non aggiunge le sequenze di escape a `%` ed a `_`.

Questa funzione è identica a [mysql\\_real\\_escape\\_string\(\)](#) eccetto che [mysql\\_real\\_escape\\_string\(\)](#) accetta un identificativo di connessione ed aggiunge le sequenze di escape alla stringa in base al set di caratteri corrente.

`mysql_escape_string()` non accetta come argomento un identificativo di connessione e non rispetta le impostazioni del corrente set di caratteri.

### Esempio 1. Esempio di `mysql_escape_string()`

```
<?php
    $voce = "Zak's Laptop";
    $voce_con_escape = mysql_escape_string($voce);
    printf ("La stringa con le sequenze di escape: %s\n", $voce_con_escape);
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
La stringa con le sequenze di escape: Zak\'s Laptop
```

Vedere anche: [mysql\\_real\\_escape\\_string\(\)](#), [addslashes\(\)](#), e la direttiva [magic\\_quotes\\_gpc](#) .

## mysql\_fetch\_array

(PHP 3, PHP 4, PHP 5)

`mysql_fetch_array` -- Carica una riga del risultato come un array associativo, un array numerico o entrambi.

## Descrizione

array **mysql\_fetch\_array** ( resource risultato [, int tipo\_risultato] )

Restituisce un array che corrisponde alla riga caricata o **FALSE** se non ci sono più righe.

`mysql_fetch_array()` è una versione estesa di [mysql\\_fetch\\_row\(\)](#). Oltre a memorizzare i dati del risultato in array con indice numerico, questa li memorizza anche con indici associativi usando i nomi dei campi come chiavi.

Se due o più colonne del risultato hanno gli stessi nomi di campo, l'ultima colonna avrà la

precedenza. Per accedere alle altre colonne con lo stesso nome, si deve usare l'indice numerico della colonna o farne un alias. Per le colonne-alias, non si può accedere al contenuto con il nome della colonna originale (in questo esempio si usa *'campo'*).

### Esempio 1. Query con nomi di campo duplicati

```
SELECT tabella1.campo as foo tabella2.campo as bar from tabella1, tabella2
```

Una cosa importante da notare è che l'uso di `mysql_fetch_array()` non è significativamente più lento dell'uso di `mysql_fetch_row()`; questo fornisce un significativo valore aggiunto.

Il secondo argomento opzionale *tipo\_risultato* in `mysql_fetch_array()` è una costante e può assumere i seguenti valori: `MYSQL_ASSOC`, `MYSQL_NUM` e `MYSQL_BOTH`. Questa caratteristica è stata aggiunta nel PHP 3.0.7. `MYSQL_BOTH` è il valore predefinito per questo argomento.

Usando `MYSQL_BOTH`, si ottiene un array con entrambe gli indici (associativo e numerico). Usando `MYSQL_ASSOC`, si ottengono solo gli indici associativi (stesso funzionamento di `mysql_fetch_assoc()`), usando `MYSQL_NUM`, si ottengono solo gli indici numerici (stesso funzionamento di `mysql_fetch_row()`).

**Nota:** I nomi dei campi restituiti da questa funzione sono *case-sensitive*.

### Esempio 2. mysql\_fetch\_array() con MYSQL\_NUM

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql") or
    die("Connessione non riuscita: " . mysql_error());
mysql_select_db("mio_db");

$resultato = mysql_query("SELECT id, nome FROM mia_tabella");

while ($riga = mysql_fetch_array($resultato, MYSQL_NUM)) {
    printf ("ID: %s Nome: %s", $riga[0], $riga[1]);
}

mysql_free_result($resultato);
?>
```

### Esempio 3. mysql\_fetch\_array() con MYSQL\_ASSOC

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql") or
    die("Connessione non riuscita: " . mysql_error());
mysql_select_db("mio_db");

$resultato = mysql_query("SELECT id, nome FROM mia_tabella");

while ($riga = mysql_fetch_array($resultato, MYSQL_ASSOC)) {
    printf ("ID: %s Nome: %s", $riga["id"], $riga["name"]);
}

mysql_free_result($resultato);
?>
```

#### Esempio 4. `mysql_fetch_array()` con `MYSQL_BOTH`

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql") or
    die("Connessione non riuscita: " . mysql_error());
mysql_select_db("mio_db");

$risultato = mysql_query("SELECT id, nome FROM mia_tabella");

while ($riga = mysql_fetch_array($risultato, MYSQL_BOTH)) {
    printf ("ID: %s Nome: %s", $riga[0], $riga["nome"]);
}

mysql_free_result($risultato);
?>
```

Per maggiori dettagli, vedere anche [mysql\\_fetch\\_row\(\)](#) e [mysql\\_fetch\\_assoc\(\)](#).

## mysql\_fetch\_assoc

(PHP 4 >= 4.0.3, PHP 5)

`mysql_fetch_assoc` -- Carica una riga del risultato come array associativo

### Descrizione

array `mysql_fetch_assoc` ( resource risultato )

Restituisce un array associativo che corrisponde alla riga caricata o `FALSE` se non ci sono più righe.

`mysql_fetch_assoc()` è equivalente alla chiamata di [mysql\\_fetch\\_array\(\)](#) con `MYSQL_ASSOC` come secondo parametro opzionale. Questa restituisce solo un array associativo. Questo è il modo in cui [mysql\\_fetch\\_array\(\)](#) funzionava originalmente. Se è necessario l'indice numerico al posto di quello associativo, usare [mysql\\_fetch\\_array\(\)](#).

Se due o più colonne del risultato hanno gli stessi nomi di campo, l'ultima colonna avrà la precedenza. Per accedere alle altre colonne con lo stesso nome, si deve accedere al risultato con l'indice numerico usando [mysql\\_fetch\\_row\(\)](#) oppure aggiungere degli alias. Vedere l'esempio nella descrizione di [mysql\\_fetch\\_array\(\)](#) per quanto concerne gli alias.

Una cosa importante da notare è che l'uso di `mysql_fetch_assoc()` *non è significativamente* più lento dell'uso di [mysql\\_fetch\\_row\(\)](#); questo fornisce un significativo valore aggiunto.

#### Esempio 1. Un esteso esempio di `mysql_fetch_assoc()`

```
<?php

$conn = mysql_connect("localhost", "utente_mysql", "password_mysql");
```

```

if (!$conn) {
    echo "Impossibile connettersi al DB: " . mysql_error();
    exit;
}

if (!mysql_select_db("mio_nome_db")) {
    echo "Impossibile selezionare mio_nome_db: " . mysql_error();
    exit;
}

$sql = "SELECT id as id_utente, nome_intero, stato_utente
FROM    una_tabella
WHERE   stato_utente = 1";

$resultato = mysql_query($sql);

if (!$resultato) {
    echo "Fallimento nell'esecuzione della query ($sql) dal DB: " . mysql_error();
    exit;
}

if (mysql_num_rows($resultato) == 0) {
    echo "Nessuna riga trovata, niente da stampare quindi si esce";
    exit;
}

// Finché esiste una riga di dati, si pone questa riga in $riga come un array associativo
// Nota: Se ci si aspetta solo una riga, non è necessario usare un ciclo
// Nota: Se si mette extract($riga); all'interno del seguente ciclo,
//       si creeranno $id_utente, $nome_intero, and $stato_utente
while ($riga = mysql_fetch_assoc($resultato)) {
    echo $riga["id_utente"];
    echo $riga["nome_intero"];
    echo $riga["stato_utente"];
}

mysql_free_result($resultato);

?>

```

Vedere anche [mysql\\_fetch\\_row\(\)](#), [mysql\\_fetch\\_array\(\)](#), [mysql\\_query\(\)](#) e [mysql\\_error\(\)](#).

## mysql\_fetch\_field

(PHP 3, PHP 4, PHP 5)

`mysql_fetch_field` -- Ottiene informazioni sulla colonna da un risultato e le restituisce come oggetto

### Descrizione

object `mysql_fetch_field` ( resource risultato [, int indice\_campo] )

Restituisce un oggetto contenente le informazioni di un campo.

`mysql_fetch_field()` può essere usata al fine di ottenere informazioni circa i campi di un

determinato risultato di una query. Se l'indice del campo non è specificato, il successivo campo non ancora recuperato da **mysql\_fetch\_field()** viene considerato.

Le proprietà dell'oggetto sono:

- name - nome della colonna
- table - nome della tabella a cui appartiene la colonna
- max\_length - massima lunghezza della colonna
- not\_null - 1 se la colonna non può essere NULL
- primary\_key - 1 se la colonna è una chiave primaria
- unique\_key - 1 se la colonna è una chiave unica
- multiple\_key - 1 se la colonna è una chiave non-unica
- numeric - 1 se la colonna è numerica
- blob - 1 se la colonna è un BLOB
- type - il tipo della colonna
- unsigned - 1 se la colonna è senza segno
- zerofill - 1 se la colonna è completata da zeri

### Esempio 1. mysql\_fetch\_field()

```
<?php
mysql_connect('localhost:3306', $utente, $password)
    or die("Connessione non riuscita: " . mysql_error());
mysql_select_db("database");
$resultato = mysql_query("select * from tabella")
    or die("Query fallita: " . mysql_error());
/* ottiene i metadata della colonna */
$i = 0;
while ($i < mysql_num_fields($resultato)) {
    echo "Informazioni della colonna $i:<br />\n";
    $meta = mysql_fetch_field($resultato);
    if (!$meta) {
        echo "Nessuna informazione disponibile<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
```

```
unique_key:    $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
</pre>";
    $i++;
}
mysql_free_result($risultato);
?>
```

Vedere anche [mysql\\_field\\_seek\(\)](#).

## mysql\_fetch\_lengths

(PHP 3, PHP 4, PHP 5)

mysql\_fetch\_lengths -- Ottiene la lunghezza di ogni output nel risultato

### Descrizione

array **mysql\_fetch\_lengths** ( resource risultato )

Restituisce un array che corrisponde alle lunghezze di ogni campo nell'ultima riga caricata da [mysql\\_fetch\\_row\(\)](#) oppure FALSE in caso di errore.

**mysql\_fetch\_lengths()** memorizza le lunghezze di ogni colonna dell'ultima riga restituita da [mysql\\_fetch\\_row\(\)](#), [mysql\\_fetch\\_array\(\)](#) e [mysql\\_fetch\\_object\(\)](#) in un array, iniziando con un indice pari a 0.

Vedere anche: [mysql\\_fetch\\_row\(\)](#).

## mysql\_fetch\_object

(PHP 3, PHP 4, PHP 5)

mysql\_fetch\_object -- Carica una riga del risultato come un oggetto

### Descrizione

object **mysql\_fetch\_object** ( resource risultato )

Restituisce un oggetto con proprietà che corrispondono alla riga caricata oppure FALSE se non ci sono più righe.

**mysql\_fetch\_object()** è simile a [mysql\\_fetch\\_array\(\)](#), con una differenza - viene restituito un

oggetto invece che un array. Indirettamente, questo significa che si ha solo accesso ai dati attraverso i nomi dei campi e non attraverso il loro indice (i numeri sono nomi di proprietà illeciti).

```
<?php
/* questo è valido */
echo $riga->campo;
/* questo non è valido */
echo $riga->0;

?>
```

In termini di velocità, la funzione è identica a [mysql\\_fetch\\_array\(\)](#) e quasi veloce come [mysql\\_fetch\\_row\(\)](#) (la differenza è insignificante).

### Esempio 1. Esempio di [mysql\\_fetch\\_object\(\)](#)

```
<?php
mysql_connect("hostname", "utente", "password");
mysql_select_db($db);
$resultato = mysql_query("select * from tabella");
while ($riga = mysql_fetch_object($resultato)) {
    echo $riga->id_utente;
    echo $riga->nome_intero;
}
mysql_free_result($resultato);
?>
```

Vedere anche: [mysql\\_fetch\\_array\(\)](#) e [mysql\\_fetch\\_row\(\)](#).

## mysql\_fetch\_row

(PHP 3, PHP 4, PHP 5)

`mysql_fetch_row` -- Ottiene una riga del risultato come un array enumerato

### Descrizione

array `mysql_fetch_row` ( resource risultato )

Restituisce un array che corrisponde ad una riga caricata oppure `FALSE` se non ci sono più righe.

`mysql_fetch_row()` carica una riga di dati dal risultato associato all'identificativo specificato. La riga è restituita con un array. Ogni colonna del risultato è memorizzata in un indice dell'array, partendo dall'indice 0.

La susseguente chiamata a `mysql_fetch_row()` restituisce la successiva riga nell'intervallo del risultato oppure `FALSE` se non ci sono più righe.

Vedere anche: [mysql\\_fetch\\_array\(\)](#), [mysql\\_fetch\\_object\(\)](#), [mysql\\_data\\_seek\(\)](#),

[mysql\\_fetch\\_lengths\(\)](#) e [mysql\\_result\(\)](#).

## mysql\_field\_flags

(PHP 3, PHP 4, PHP 5)

mysql\_field\_flags -- Ottiene i flag associati al campo specificato di un risultato

### Descrizione

string **mysql\_field\_flags** ( resource risultato, int indice\_campo )

**mysql\_field\_flags()** restituisce i flag del campo specificato. I flag sono restituiti come singole parole per flag separate da un singolo spazio, in modo che sia possibile suddividere il valore restituito usando [explode\(\)](#).

I seguenti flag sono restituiti, se la versione di MySQL è abbastanza aggiornata da supportarli: "not\_null", "primary\_key", "unique\_key", "multiple\_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto\_increment", "timestamp".

Per motivi di compatibilità con il passato, anche **mysql\_fieldflags()** può essere usata. Questo comunque è sconsigliato.

## mysql\_field\_len

(PHP 3, PHP 4, PHP 5)

mysql\_field\_len -- Restituisce la lunghezza del campo specificato

### Descrizione

int **mysql\_field\_len** ( resource risultato, int indice\_campo )

**mysql\_field\_len()** restituisce la lunghezza del campo specificato.

Per motivi di compatibilità con il passato, anche **mysql\_fieldlen()** può essere usata. Questo comunque è sconsigliato.

## mysql\_field\_name

(PHP 3, PHP 4, PHP 5)

`mysql_field_name` -- Ottiene il nome del campo specificato in un risultato

## Descrizione

string **mysql\_field\_name** ( resource risultato, int indice\_campo )

**mysql\_field\_name()** restituisce il nome del campo specificato dall'indice. *risultato* deve essere un identificativo di risultato valido e *indice\_campo* è lo spiazzamento numerico del campo.

**Nota:** *indice\_campo* inizia da 0.

Es. L'indice del terzo campo è in realtà 2, l'indice del quarto campo è 3 e così via.

### Esempio 1. Esempio di `mysql_field_name()`

```
/* La tabella utenti è costituita da tre campi:
 *   id_utente
 *   nome_utente
 *   password
 */
$connessione = mysql_connect('localhost', "utente_mysql", "password_mysql");
mysql_select_db($nome_db, $connessione)
    or die("Errore nella selezione di $dbname: " . mysql_error());
$risultato = mysql_query("select * from utenti", $connessione);

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

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
id_utente
password
```

Per motivi di compatibilità con il passato, anche **mysql\_fieldname()** può essere usata. Questo comunque è sconsigliato.

## mysql\_field\_seek

(PHP 3, PHP 4, PHP 5)

`mysql_field_seek` -- Imposta il puntatore al risultato ad un determinato indice di campo

## Descrizione

int **mysql\_field\_seek** ( resource risultato, int indice\_campo )

Esegue il seek ad uno specifico indice di campo. Se la successiva chiamata a [mysql\\_fetch\\_field\(\)](#) non include un indice di campo, quello specificato in **mysql\_field\_seek()** viene restituito.

Vedere anche: [mysql\\_fetch\\_field\(\)](#).

## mysql\_field\_table

(PHP 3, PHP 4, PHP 5)

mysql\_field\_table -- Ottiene il nome della tabella in cui si trova il campo specificato

### Descrizione

string **mysql\_field\_table** ( resource risultato, int indice\_campo )

Ottiene il nome della tabella in cui si trova il campo specificato.

Per motivi di compatibilità con il passato, anche **mysql\_fieldtable()** può essere usata. Questo comunque è sconsigliato.

## mysql\_field\_type

(PHP 3, PHP 4, PHP 5)

mysql\_field\_type -- Ottiene il tipo del campo specificato in un risultato

### Descrizione

string **mysql\_field\_type** ( resource risultato, int indice\_campo )

**mysql\_field\_type()** è simile alla funzione [mysql\\_field\\_name\(\)](#). Gli argomenti sono identici, ma viene restituito il tipo del campo. Il tipo del campo sarà uno dei seguenti: "int", "real", "string", "blob" ed altri come dettagliati nella [Documentazione di MySQL](#).

#### Esempio 1. Tipi di campo MySQL

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql");
mysql_select_db("mysql");
$resultato = mysql_query("SELECT * FROM func");
$campi = mysql_num_fields($resultato);
$righe = mysql_num_rows($resultato);
$tabella = mysql_field_table($resultato, 0);
echo "La tabella ".$table."' ha ".$fields." campi e ".$righe." record\n";
echo "La tabella ha i seguenti campi:\n";
for ($i=0; $i < $campi; $i++) {
    $tipo = mysql_field_type($resultato, $i);
    $nome = mysql_field_name($resultato, $i);
    $lung = mysql_field_len($resultato, $i);
    $flag = mysql_field_flags($resultato, $i);
```

```
        echo $tipo." ".$nome." ".$lung." ".$flag."\n";
    }
    mysql_free_result($risultato);
    mysql_close();
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
La tabella 'func' ha 4 campi e 1 record
La tabella ha i seguenti campi:
string name 64 not_null primary_key binary
int ret 1 not_null
string dl 128 not_null
string type 9 not_null enum
```

Per motivi di compatibilità con il passato, anche **mysql\_fieldtype()** può essere usata. Questo comunque è sconsigliato.

## mysql\_free\_result

(PHP 3, PHP 4, PHP 5)

mysql\_free\_result -- Libera la memoria occupata dal risultato

### Descrizione

bool **mysql\_free\_result** ( resource risultato )

**mysql\_free\_result()** libera tutta la memoria associata all'identificativo del risultato *risultato*.

**mysql\_free\_result()** deve essere richiamata solo se si è preoccupati sulla quantità di memoria usata dalle query che restituiscono dei grandi risultati. Tutta la memoria associata al risultato è automaticamente liberata al termine dell'esecuzione dello script.

Restituisce **TRUE** in caso di successo, **FALSE** in caso di fallimento.

Per motivi di compatibilità con il passato, anche **mysql\_freeresult()** può essere usata. Questo comunque è sconsigliato.

## mysql\_get\_client\_info

(PHP 4 >= 4.0.5, PHP 5)

mysql\_get\_client\_info -- Ottiene informazioni sul client MySQL

## Descrizione

string **mysql\_get\_client\_info** ( void )

**mysql\_get\_client\_info()** restituisce una stringa che rappresenta la versione della libreria client.

### Esempio 1. Esempio di **mysql\_get\_client\_info**

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

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
Informazioni sul client MySQL: 3.23.39
```

Vedere anche: [mysql\\_get\\_host\\_info\(\)](#), [mysql\\_get\\_proto\\_info\(\)](#) e [mysql\\_get\\_server\\_info\(\)](#).

## mysql\_get\_host\_info

(PHP 4 >= 4.0.5, PHP 5)

**mysql\_get\_host\_info** -- Ottiene le informazioni sull'host MySQL

## Descrizione

string **mysql\_get\_host\_info** ( [resource identificativo\_connessione] )

**mysql\_get\_host\_info()** restituisce una stringa che descrive il tipo di connessione in uso per *identificativo\_connessione*, includendo il nome dell'host del server. Se *identificativo\_connessione* è omissso, sarà usata l'ultima connessione aperta.

### Esempio 1. Esempio di **mysql\_get\_host\_info**

```
<?php
    mysql_connect("localhost", "utente_mysql", "password_mysql") or
        die("Connessione non riuscita: " . mysql_error());
    printf ("Informazioni sull'host MySQL: %s\n", mysql_get_host_info());
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
Informazioni sull'host MySQL: Localhost via UNIX socket
```

Vedere anche: [mysql\\_get\\_client\\_info\(\)](#), [mysql\\_get\\_proto\\_info\(\)](#) e [mysql\\_get\\_server\\_info\(\)](#).

# mysql\_get\_proto\_info

(PHP 4 >= 4.0.5, PHP 5)

mysql\_get\_proto\_info -- Ottiene le informazioni sul protocollo MySQL

## Descrizione

int **mysql\_get\_proto\_info** ( [resource identificativo\_connessione] )

**mysql\_get\_proto\_info()** restituisce la versione del protocollo usata dalla connessione *identificativo\_connessione*. Se *identificativo\_connessione* è omissa, sarà usata l'ultima connessione aperta.

### Esempio 1. Esempio di mysql\_get\_proto\_info

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql") or
    die("Connessione non riuscita: " . mysql_error());
printf ("Versione del protocollo MySQL: %s\n", mysql_get_proto_info());
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
Versione del protocollo MySQL: 10
```

Vedere anche: [mysql\\_get\\_client\\_info\(\)](#), [mysql\\_get\\_host\\_info\(\)](#) e [mysql\\_get\\_server\\_info\(\)](#).

# mysql\_get\_server\_info

(PHP 4 >= 4.0.5, PHP 5)

mysql\_get\_server\_info -- Ottiene le informazioni sul server MySQL

## Descrizione

string **mysql\_get\_server\_info** ( [resource identificativo\_connessione] )

**mysql\_get\_server\_info()** restituisce la versione del server usato dalla connessione *identificativo\_connessione*. Se *identificativo\_connessione* è omissa, sarà usata l'ultima connessione aperta.

### Esempio 1. Esempio di mysql\_get\_server\_info

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql") or
die("Connessione non riuscita: " . mysql_error());
printf ("Versione server MySQL: %s\n", mysql_get_server_info());
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
Versione server MySQL: 4.0.1-alpha
```

Vedere anche: [mysql\\_get\\_client\\_info\(\)](#), [mysql\\_get\\_host\\_info\(\)](#) e [mysql\\_get\\_proto\\_info\(\)](#).

## mysql\_info

(PHP 4 >= 4.3.0, PHP 5)

mysql\_info -- Ottiene le informazioni relative alla query più recente.

### Descrizione

string **mysql\_info** ( [resource identificativo\_connessione] )

**mysql\_info()** restituisce informazioni dettagliate relative all'ultima query usando lo specifico *identificativo\_connessione*. Se *identificativo\_connessione* non è specificato, viene considerata l'ultima connessione aperta.

**mysql\_info()** restituisce una stringa per tutte le istruzioni elencate di seguito. Per tutte le altre restituisce FALSE. Il formato della stringa dipende dall'istruzione data.

#### Esempio 1. Istruzioni MySQL significative

```
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
```

I numeri sono indicati solo a titolo esemplificativo; i loro valori corrispondono alla query.

**Nota:** **mysql\_info()** restituisce un valore non FALSE per le istruzioni INSERT ... VALUES solo se nell'istruzione sono specificate liste di valori multipli.

Vedere anche: [mysql\\_affected\\_rows\(\)](#)

# mysql\_insert\_id

(PHP 3, PHP 4, PHP 5)

mysql\_insert\_id -- Ottiene l'identificativo generato dalla precedente operazione INSERT

## Descrizione

int **mysql\_insert\_id** ( [resource identificativo\_connessione] )

**mysql\_insert\_id()** restituisce l'identificativo generato per una colonna AUTO\_INCREMENT dal precedente query INSERT usando lo specifico *identificativo\_connessione*. Se *identificativo\_connessione* non è specificato, viene considerata l'ultima connessione aperta.

**mysql\_insert\_id()** restituisce 0 se la precedente query non ha generato un valore AUTO\_INCREMENT. Se è necessario salvare il valore per usarlo in seguito, assicurarsi di richiamare **mysql\_insert\_id()** immediatamente dopo la query che ha generato il valore.

**Nota:** Il valore della funzione SQL *LAST\_INSERT\_ID()* di MySQL contiene sempre il più recente valore AUTO\_INCREMENT generato e non è azzerato dalle query.

### Avvertimento

**mysql\_insert\_id()** converte il tipo restituito dalla funzione nativa dell'API C di MySQL *mysql\_insert\_id()* al tipo *long* (chiamata [int](#) nel PHP). Se la colonna AUTO\_INCREMENT è del tipo BIGINT, il valore restituito da **mysql\_insert\_id()** sarà inesatto. In questo caso si usi la funzione SQL di MySQL *LAST\_INSERT\_ID()* in una query SQL.

### Esempio 1. Esempio di mysql\_insert\_id

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql") or
    die("Connessione non riuscita: " . mysql_error());
mysql_select_db("mio_db");

mysql_query("INSERT INTO mia_tabella (prodotto) VALUES ('kossu')");
printf ("L'ultimo record inserito ha l'identificativo %d\n", mysql_insert_id());
?>
```

Vedere anche: [mysql\\_query\(\)](#).

# mysql\_list\_dbs

(PHP 3, PHP 4, PHP 5)

mysql\_list\_dbs -- Elenca i database disponibili in un server MySQL

## Descrizione

resource **mysql\_list\_dbs** ( [resource identificativo\_connessione] )

**mysql\_list\_dbs()** restituirà un risultato puntatore contenete i database resi disponibili dal demone mysql. Usare la funzione [mysql\\_tablename\(\)](#) per esplorare questo risultato puntatore o qualsiasi funzione per i risultati delle tabelle, come [mysql\\_fetch\\_array\(\)](#).

### Esempio 1. Esempio di mysql\_list\_dbs()

```
<?php
$connessione = mysql_connect('localhost', 'utente_mysql', 'password_mysql');
$lista_db = mysql_list_dbs($connessione);

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

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
database1
database2
database3
...
```

**Nota:** Il codice riportato sopra dovrebbe funzionare facilmente con [mysql\\_fetch\\_row\(\)](#) o altre funzioni simili.

Per motivi di compatibilità con il passato, anche **mysql\_listdbs()** può essere usata. Questo comunque è sconsigliato.

Vedere anche [mysql\\_db\\_name\(\)](#).

## mysql\_list\_fields

(PHP 3, PHP 4, PHP 5)

mysql\_list\_fields -- Elenca i campi di un risultato MySQL

## Descrizione

resource **mysql\_list\_fields** ( string nome\_database, string nome\_tabella [, resource identificativo\_connessione] )

**mysql\_list\_fields()** recupera le informazioni relative ad una data tabella. Gli argomenti sono il nome del database ed il nome della tabella. Viene restituito un risultato puntatore che può essere usato con [mysql\\_field\\_flags\(\)](#), [mysql\\_field\\_len\(\)](#), [mysql\\_field\\_name\(\)](#) e [mysql\\_field\\_type\(\)](#).

## Esempio 1. Esempio di `mysql_list_fields()`

```
<?php
$connessione = mysql_connect('localhost', 'utente_mysql', 'password_mysql');

$campi = mysql_list_fields("database1", "tabella1", $connessione);
$colonne = mysql_num_fields($campi);

for ($i = 0; $i < $colonne; $i++) {
    echo mysql_field_name($campi, $i) . "\n";
}
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
campo1
campo2
campo3
...
```

Per motivi di compatibilit  con il passato, anche `mysql_listfields()` pu  essere usata. Questo comunque   sconsigliato.

# mysql\_list\_processes

(PHP 4 >= 4.3.0, PHP 5)

`mysql_list_processes` -- Elenca i processi MySQL

## Descrizione

resource `mysql_list_processes` ( [resource identificativo\_connessione] )

`mysql_list_processes()` restituisce un risultato puntatore che descrive i thread attuali del server.

## Esempio 1. Esempio di `mysql_list_processes()`

```
<?php
$connessione = mysql_connect('localhost', 'utente_mysql', 'password_mysql');

$resultato = mysql_list_processes($connessione);
while ($riga = mysql_fetch_row($resultato)){
    printf("%s %s %s %s %s\n", $riga["Id"], $riga["Host"], $riga["db"],
        $riga["Command"], $riga["Time"]);
}
mysql_free_result ($resultato);
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

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

Vedere anche: [mysql\\_thread\\_id\(\)](#)

# mysql\_list\_tables

(PHP 3, PHP 4, PHP 5)

mysql\_list\_tables -- Elenca le tabelle in un database MySQL

## Descrizione

resource **mysql\_list\_tables** ( string database [, resource identificativo\_connessione] )

**mysql\_list\_tables()** accetta un nome di database e restituisce un risultato puntatore in modo molto simile alla funzione [mysql\\_query\(\)](#). Usare la funzione [mysql\\_tablename\(\)](#) per esplorare questo risultato puntatore o qualsiasi funzione per i risultati delle tabelle, come [mysql\\_fetch\\_array\(\)](#).

Il parametro *database* è il nome del database da cui recuperare la lista di tabelle. in caso di errore, **mysql\_list\_tables()** restituisce FALSE.

Per motivi di compatibilità con il passato, anche **mysql\_listtables()** può essere usata. Questo comunque è sconsigliato.

### Esempio 1. mysql\_list\_tables() example

```
<?php
    $nome_db = 'nome_db_mysql';

    if (!mysql_connect('host_mysql', 'utente_mysql', 'password_mysql')) {
        print 'Connessione a mysql non riuscita';
        exit;
    }

    $risultato = mysql_list_tables($nome_db);

    if (!$risultato) {
        print "Errore database, Impossibile elencare le tabelle\n";
        print 'Errore MySQL: ' . mysql_error();
        exit;
    }

    while ($riga = mysql_fetch_row($risultato)) {
        print "Tabella: $riga[0]\n";
    }

    mysql_free_result($risultato);
?>
```

Vedere anche: [mysql\\_list\\_dbs\(\)](#) e [mysql\\_tablename\(\)](#).

# mysql\_num\_fields

(PHP 3, PHP 4, PHP 5)

mysql\_num\_fields -- Ottiene il numero di campi nel risultato

## Descrizione

int **mysql\_num\_fields** ( resource risultato )

**mysql\_num\_fields()** restituisce il numero di campi in un risultato.

Vedere anche: [mysql\\_select\\_db\(\)](#), [mysql\\_query\(\)](#), [mysql\\_fetch\\_field\(\)](#) e [mysql\\_num\\_rows\(\)](#).

Per motivi di compatibilità con il passato, anche **mysql\_numfields()** può essere usata. Questo comunque è sconsigliato.

# mysql\_num\_rows

(PHP 3, PHP 4, PHP 5)

mysql\_num\_rows -- Ottiene il numero di righe in un risultato

## Descrizione

int **mysql\_num\_rows** ( resource risultato )

**mysql\_num\_rows()** restituisce il numero di righe in un risultato. Questo comando è valido solo per le istruzioni SELECT. Per recuperare il numero di righe coinvolte dalle query INSERT, UPDATE o DELETE, usare [mysql\\_affected\\_rows\(\)](#).

**Esempio 1. Esempio di mysql\_num\_rows()**

```
<?php
$connessione = mysql_connect("localhost", "utente_mysql", "password_mysql");
mysql_select_db("database", $connessione);

$resultato = mysql_query("SELECT * FROM tabella1", $connessione);
$num_righe = mysql_num_rows($resultato);

echo "$num_righe Righe\n";

?>
```

**Nota:** Se si usa [mysql\\_unbuffered\\_query\(\)](#), **mysql\_num\_rows()** non restituirà il

valore corretto finché non sono recuperate tutte le righe del risultato.

Vedere anche: [mysql\\_affected\\_rows\(\)](#), [mysql\\_connect\(\)](#), [mysql\\_data\\_seek\(\)](#), [mysql\\_select\\_db\(\)](#) e [mysql\\_query\(\)](#).

Per motivi di compatibilità con il passato, anche `mysql_numrows()` può essere usata. Questo comunque è sconsigliato.

## mysql\_pconnect

(PHP 3, PHP 4, PHP 5)

`mysql_pconnect` -- Apre una connessione persistente ad un server MySQL

### Descrizione

resource **mysql\_pconnect** ( [string server [, string nome\_utente [, string password [, int flag\_client]]]] )

Restituisce un identificativo di connessione MySQL valido in caso di successo oppure `FALSE` in caso di errore.

**mysql\_pconnect()** stabilisce una connessione ad un server MySQL. I seguenti valori predefiniti sono usati per i parametri opzionali mancanti: *server* = 'localhost:3306', *nome\_utente* = nome dell'utente proprietario del processo server e *password* = password vuota. Il parametro *flag\_client* può essere una combinazione delle costanti `MYSQL_CLIENT_COMPRESS`, `MYSQL_CLIENT_IGNORE_SPACE` o `MYSQL_CLIENT_INTERACTIVE`.

Il parametro *server* può includere un numero di porta. Es. "hostname:porta" o un percorso ad un socket es. ":/percorso/a/socket" per il localhost.

**Nota:** Il supporto per ":porta" è stato aggiunto nel PHP 3.0B4.

Il supporto per ":/percorso/a/socket" è stato aggiunto nel PHP 3.0.10.

**mysql\_pconnect()** agisce in modo molto simile a [mysql\\_connect\(\)](#) con due differenze principali.

Primo, quando si connette, la funzione tenta innanzitutto di trovare una connessione (persistente) già aperta avente gli stessi host, username e password. Se viene trovata una connessione, viene restituito un identificativo a questa anziché aprirne una nuova.

Secondo, la connessione al server SQL non sarà chiusa quando l'esecuzione dello script termina. La connessione rimane invece aperta per usi futuri ([mysql\\_close\(\)](#) non chiuderà le connessioni stabilite da **mysql\_pconnect()**).

Il parametro opzionale *flag\_client* è diventato disponibile nel PHP 4.3.0.

Questo tipo di link è perciò chiamato 'persistente'.

**Nota:** Notare che questo tipo di connessione funziona solo se si usa PHP come modulo. Vedere la sezione [Persistent Database Connections](#) per maggiori informazioni.

#### Avvertimento

L'uso di connessioni persistenti può richiedere un po' di messa a punto delle configurazioni di Apache e MySQL per assicurarsi di non eccedere il numero di connessioni consentite da MySQL.

## mysql\_ping

(PHP 4 >= 4.3.0, PHP 5)

mysql\_ping -- Esegue un ping su una connessione al server o riconnette se non esiste la connessione

### Descrizione

bool **mysql\_ping** ( [resource identificativo\_connesione] )

**mysql\_ping()** controlla se una connessione al server funziona o meno. Se questa è caduta, viene tentata una riconnessione automatica. Questa funzione può essere usata dagli script che rimangono inattivi per un lungo periodo per controllare se il server ha chiuso la connessione o meno e riconnettersi se necessario. **mysql\_ping()** restituisce TRUE se la connessione al server è funzionante, altrimenti FALSE.

Vedere anche: [mysql\\_thread\\_id\(\)](#) e [mysql\\_list\\_processes\(\)](#).

## mysql\_query

(PHP 3, PHP 4, PHP 5)

mysql\_query -- Invia una query MySQL

### Descrizione

resource **mysql\_query** ( string query [, resource identificativo\_connesione [, int modo\_risultato]] )

**mysql\_query()** invia una query al database attualmente attivo sul server associato all'identificativo di connessione specificato. Se *identificativo\_connesione* non è specificato, viene considerata l'ultima connessione aperta. Se nessuna connessione è aperta, la funzione prova a stabilire una connessione come se [mysql\\_connect\(\)](#) fosse richiamata senza argomenti ed usa questa.

Il parametro opzionale *modo\_risultato* può essere `MYSQL_USE_RESULT` e `MYSQL_STORE_RESULT`. Il valore predefinito `MYSQL_STORE_RESULT`, così il risultato è bufferato. Vedere anche [mysql\\_unbuffered\\_query\(\)](#) per la controparte di questo comportamento.

**Nota:** La stringa della query non dovrebbe terminare con un punto e virgola.

Solo per le istruzioni `SELECT`, `SHOW`, `EXPLAIN` o `DESCRIBE` `mysql_query()` restituisce un identificativo di risorsa o `FALSE` se la query non è stata eseguita correttamente. Per altri tipi di istruzioni SQL, `mysql_query()` restituisce `TRUE` in caso di successo e `FALSE` in caso di errore. Un valore restituito diverso da `FALSE` indica che la query era lecita ed è stata eseguita dal server. Questo non indica niente riguardo il numero di righe coinvolte o restituite. È assolutamente possibile che una query abbia successo ma che non coinvolga o restituisca nessuna riga.

La seguente query non è valida sintatticamente, quindi `mysql_query()` fallisce e restituisce `FALSE`:

**Esempio 1. mysql\_query()**

```
<?php
$resultato = mysql_query("SELECT * WHERE 1=1")
    or die("Query non valida: " . mysql_error());
?>
```

La seguente query non è semanticamente valida se *mia\_colonna* non è una colonna della tabella *mia\_tabella*, quindi `mysql_query()` fallisce e restituisce `FALSE`:

**Esempio 2. mysql\_query()**

```
<?php
$resultato = mysql_query("SELECT mia_colonna FROM mia_tabella")
    or die("Query non valida: " . mysql_error());
?>
```

`mysql_query()` fallisce e restituisce `FALSE` anche se non si hanno i permessi per accedere alle tabelle cui la query fa riferimento.

Assumendo che la query abbia successo, si può richiamare [mysql\\_num\\_rows\(\)](#) per scoprire quante righe sono state restituite da un'istruzione `SELECT` o [mysql\\_affected\\_rows\(\)](#) per scoprire quante righe sono state coinvolte da un'istruzione `DELETE`, `INSERT`, `REPLACE` o `UPDATE`.

Solo per le istruzioni `SELECT`, `SHOW`, `DESCRIBE` o `EXPLAIN`, `mysql_query()` restituisce un nuovo identificativo di risultato che si può passare a [mysql\\_fetch\\_array\(\)](#) e ad altre funzioni che si occupano dei risultati delle tabelle. Quando si conclude il trattamento del risultato, si possono liberare le risorse associate ad esso richiamando [mysql\\_free\\_result\(\)](#). Comunque la memoria sarà liberata automaticamente Al termine dell'esecuzione dello script.

Vedere anche: [mysql\\_num\\_rows\(\)](#), [mysql\\_affected\\_rows\(\)](#), [mysql\\_unbuffered\\_query\(\)](#), [mysql\\_free\\_result\(\)](#), [mysql\\_fetch\\_array\(\)](#), [mysql\\_fetch\\_row\(\)](#), [mysql\\_fetch\\_assoc\(\)](#), [mysql\\_result\(\)](#), [mysql\\_select\\_db\(\)](#) e [mysql\\_connect\(\)](#).

# mysql\_real\_escape\_string

(PHP 4 >= 4.3.0, PHP 5)

`mysql_real_escape_string` -- Aggiunge le sequenze di escape ai caratteri speciali in una stringa per l'uso in una istruzione SQL, tenendo conto dell'attuale set di caratteri della connessione.

## Descrizione

string **mysql\_real\_escape\_string** ( string stringa\_senza\_escape [, resource identificativo\_connessione] )

Questa funzione aggiunge le sequenze di escape ai caratteri speciali in *stringa\_senza\_escape*, tenendo conto dell'attuale set di caratteri della connessione in modo che sia sicuro usarla in [mysql\\_query\(\)](#).

**Nota:** `mysql_real_escape_string()` non aggiunge le sequenze di escape a `%` ed a `_`.

### Esempio 1. Esempio di mysql\_real\_escape\_string()

```
<?php
$connessione = mysql_connect('localhost', 'utente_mysql', 'password_mysql');
$voce = "Zak's and Derick's Laptop";
$voce_con_escape = mysql_real_escape_string($voce);
printf ("La stringa con le sequenze di escape: %s\n", $voce_con_escape);
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
La stringa con le sequenze di escape: Zak\'s and Derick\'s Laptop
```

Vedere anche : [mysql\\_escape\\_string\(\)](#) e [mysql\\_character\\_set\\_name\(\)](#).

# mysql\_result

(PHP 3, PHP 4, PHP 5)

`mysql_result` -- Ottiene i dati dal risultato

## Descrizione

mixed **mysql\_result** ( resource risultato, int campo [, mixed campo] )

`mysql_result()` restituisce i contenuti di una cella da un risultato MySQL. L'argomento campo può essere l'indice o il nome del campo oppure il nome della tabella ed il nome del campo separati da un

punto (nome\_tabella.nome\_campo). Se il nome della colonna ha un alias ('select foo as bar from...'), usare l'alias al posto del nome della colonna.

Quando si lavora con un risultato di grandi dimensioni, si dovrebbero considerare l'uso delle funzioni che caricano l'intera riga (specificate di seguito). Poiché queste funzioni restituiscono i contenuti di celle multiple in una chiamata a funzione, sono MOLTO più veloci di **mysql\_result()**. Notare anche che specificare un indice numerico per l'argomento campo è molto più veloce che specificare un argomento del tipo nome\_di\_campo o nome\_tabella.nome\_campo.

Le chiamate a **mysql\_result()** non dovrebbero esserse mescolate con chiamate ad altre funzioni che hanno a che fare con i risultati.

Alternative raccomandate per alte prestazioni: [mysql\\_fetch\\_row\(\)](#), [mysql\\_fetch\\_array\(\)](#) e [mysql\\_fetch\\_object\(\)](#).

## mysql\_select\_db

(PHP 3, PHP 4, PHP 5)

mysql\_select\_db -- Seleziona un database MySQL

### Descrizione

bool **mysql\_select\_db** ( string nome\_database [, resource identificativo\_connessione] )

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

**mysql\_select\_db()** imposta il database attualmente attivo sul server associato all'identificativo di connessione specificato. Se nessun identificativo di connessione è specificato, viene considerata l'ultima connessione aperta. Se nessuna connessione è aperta, la funzione proverà a stabilire una connessione come se [mysql\\_connect\(\)](#) fosse richiamata senza argomenti ed userà questa.

Ogni chiamata successiva a [mysql\\_query\(\)](#) sarà fatta sul database attivo.

Vedere anche: [mysql\\_connect\(\)](#), [mysql\\_pconnect\(\)](#) e [mysql\\_query\(\)](#).

Per motivi di compatibilità con il passato, anche **mysql\_selectdb()** può essere usata. Questo comunque è sconsigliato.

## mysql\_stat

(PHP 4 >= 4.3.0, PHP 5)

mysql\_stat -- Ottiene l'attuale stato del sistema

## Descrizione

string **mysql\_stat** ( [resource identificativo\_connessione] )

**mysql\_stat()** restituisce l'attuale stato del server.

**Nota:** **mysql\_stat()** attualmente restituisce solo le seguenti informazioni: uptime, thread, query, tabelle aperte, tabelle svuotate e query al secondo. Per una lista completa delle altre variabili di stato si usi il comando SQL SHOW STATUS.

### Esempio 1. mysql\_stat() example

```
<?php
$connessione = mysql_connect('localhost', "utente_mysql", "password_mysql");
$stato = explode(' ', mysql_stat($connessione));
print_r($stato);
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
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
)
```

## mysql\_tablename

(PHP 3, PHP 4, PHP 5)

mysql\_tablename -- Ottiene il nome della tabella

## Descrizione

string **mysql\_tablename** ( resource risultato, int i )

**mysql\_tablename()** prende il puntatore risultato dalla funzione [mysql\\_list\\_tables\(\)](#) come un indice intero e restituisce il nome di una tabella. La funzione [mysql\\_num\\_rows\(\)](#) può essere usata per determinare il numero di tabelle nel risultato puntatore. Usare la funzione **mysql\_tablename()** per esplorare questo risultato puntatore o qualsiasi funzione per i risultati delle tabelle, come [mysql\\_fetch\\_array\(\)](#).

**Esempio 1. Esempio di mysql\_tablename()**

```
<?php
mysql_connect("localhost", "utente_mysql", "password_mysql");
$rsultato = mysql_list_tables("mio_db");

for ($i = 0; $i < mysql_num_rows($rsultato); $i++)
    printf ("Tabela: %s\n", mysql_tablename($rsultato, $i));

mysql_free_result($rsultato);
?>
```

Vedere anche: [mysql\\_list\\_tables\(\)](#).

## mysql\_thread\_id

(PHP 4 >= 4.3.0, PHP 5)

mysql\_thread\_id -- Restituisce l'attuale thread ID

### Descrizione

int **mysql\_thread\_id** ( [resource identificativo\_connessione] )

**mysql\_thread\_id()** restituisce l'attuale thread ID. Se la connessione è persa e ci si riconnette con [mysql\\_ping\(\)](#), il thread ID cambia. Questo significa che non si dovrebbe ottenere il thread ID e memorizzarlo per impieghi successivi. Si dovrebbe rilevare il thread ID quando è necessario.

#### Esempio 1. Esempio di mysql\_thread\_id()

```
<?php
$connessione = mysql_connect('localhost', 'utente_mysql', 'password_mysql');
$thread_id = mysql_thread_id($connessione);
if ($thread_id){
    printf ("L'attuale thread è %d\n", $thread_id);
}
?>
```

L'esempio riportato sopra dovrebbe produrre il seguente output:

```
L'attuale thread è 73
```

Vedere anche: [mysql\\_ping\(\)](#) e [mysql\\_list\\_processes\(\)](#).

## mysql\_unbuffered\_query

(PHP 4 >= 4.0.6, PHP 5)

mysql\_unbuffered\_query -- Invia una query SQL a MySQL senza caricare e bufferare le righe

risultanti

## Descrizione

resource **mysql\_unbuffered\_query** ( string *query* [, resource *identificativo\_connessione* [, int *modo\_risultato*]] )

**mysql\_unbuffered\_query()** invia una query SQL *query* a MySQL senza caricare e bufferare le righe risultanti automaticamente come fa [mysql\\_query\(\)](#). Da una parte, questo risparmia un considerevole quantità di memoria con le query SQL che producono risultati di grandi dimensioni. Dall'altra parte, si può iniziare l'elaborazione dei risultati immediatamente dopo che la prima riga è stata recuperata: non si deve attendere finché la query SQL sia completamente eseguita. Quando si usano diverse connessioni a database, si deve specificare il parametro opzionale *identificativo\_connessione*.

Il parametro opzionale *modo\_risultato* può essere MYSQL\_USE\_RESULT e MYSQL\_STORE\_RESULT. Il valore predefinito è MYSQL\_USE\_RESULT, quindi il risultato non è bufferato. Vedere anche [mysql\\_query\(\)](#) per la controparte di questo comportamento.

**Nota:** I benefici di **mysql\_unbuffered\_query()** hanno un costo: non si può usare [mysql\\_num\\_rows\(\)](#) su un risultato restituito da **mysql\_unbuffered\_query()**. Inoltre si debbono caricare tutte le righe risultanti da una query SQL non bufferata prima di poter inviare una nuova query SQL a MySQL.

vedere anche: [mysql\\_query\(\)](#).