UF3: Tècniques d'accés a dades

Desenvolupament web en entorn servidor

3.3

API

Extensió MySQL millorada

JESUÏTES El Clot
Escola del Clot
Sergi Grau

Introducció

Sergi Grau
sergi.grau@fje.edu

L'EXTENSIO MYSQLI, O COM DE VEGADES SE LI CONEIX, L'EXTENSIO DE MYSQL MILLORADA, ES VA DESENVOLUPAR PER APROFITAR LES NOVES FUNCIONALITATS TROBADES EN ELS SISTEMES MYSQL AMB VERSIO 4.1.3 O POSTERIOR. L'EXTENSIO MYSQLI VE INCLOSA EN LES VERSIONS PHP 5 I POSTERIORS.

http://www.mysql.com/

Característiques

Sergi Grau
sergi.grau@fje.edu

L'extensió mysqli conté nombrosos beneficis, sent aquestes les millores principals respecte a l'extensió mysql:

Interfície orientada a objectes

Suport per Declaracions Preparades

Suport per Múltiples Declaracions

Suport per Transaccions

Millorades les opcions de depuració

Suport per servidor encastat

Característiques

Sergi Grau
sergi.grau@fje.edu

A més de la interfície orientada a objectes, aquesta extensió també proporciona una interfície procedural.

L'extensió mysqli està desenvolupada mitjançant el framework d'extensions de PHP. El seu codi font s'ubica al directori ext / mysqli.

Dos paradigmes

Sergi Grau
sergi.grau@fje.edu

L'extensió mysqli ofereix una interfície dual. Suporta el paradigme de programació procedimental i l'orientat a objectes.

Els usuaris que migren des de l'extensió mysql antiga poden preferir la interfície procedimental. Aquesta interfície és similar a la de l'extensió antiga de mysql. En la majoria dels casos, els noms de funcions difereixen únicament pel prefix. Algunes funcions de mysqli prenen com a primer argument un gestor de connexió, mentre que les funcions similars de l'antiga interfície de mysql el prenen com l'últim argument opcional.

A més de la clàssica interfície procedimental, els usuaris poden optar per usar la interfície orientada a objectes. La documentació està organitzada segons la interfície orientada a objectes. Aquesta interfície mostra les funcions agrupades pel seu propòsit, fent més fàcil els començaments. La secció de referència proveeix exemples de les dues variants de sintaxi.

No hi ha diferències significatives de rendiment entre les dues interfícies. Els usuaris pot basar la seva elecció en les seves preferències personals.

Exemple

Sergi Grau
sergi.grau@fje.edu

```php
<?php
$link = mysqli_connect ( "host" , "usuari" , "contrasenya" , "basedades" );
if ( mysqli_connect_errno ( $link )) {
echo "problema al connectar MySQL: " . mysqli_connect_error ();
}

$resultat = mysqli_query ( $link , "SELECT 'prova' AS _msg FROM DUAL" );
$fila = mysqli_fetch_assoc ( $resultat );
echo $fila [ '_msg' ];

$mysqli = new mysqli ( "host" , "usuari" , "contrasenya" , "basedades" );

if ( $mysqli -> connect_errno ) {
echo "problema al connectar MySQL: " . $mysqli -> connect_error ;

}

$resultat = $mysqli -> query ( "SELECT 'prova OO.' AS _msg FROM DUAL" );
$fila = $resultat -> fetch_assoc ();
echo $fila [ '_msg' ];
?>
```

Connexions

Sergi Grau
sergi.grau@fje.edu

El servidor MySQL suporta l'ús de diferents capes de transport per a connexions. Les connexions usen TCP / IP, sòcols de domini Unix o canonades amb nom de Windows.

El nom del host localhost té un significat especial. Està vinculat a l'ús de sockets de domini Unix. No és possible obrir una connexió TCP / IP usant com a nom de host localhost, s'ha d'usar 127.0.0.1 al seu lloc.

```
$mysqli = new mysqli ( "localhost","usuari","contrasenya" , "basedades" );
if ( $mysqli -> connect_errno ) {
echo "problema a MySQL: (" . $mysqli -> connect_errno . ") " . $mysqli ->
connect_error ;
}
echo $mysqli -> host_info . "\n" ;

$mysqli = new mysqli ( "127.0.0.1" , "usuari" , "contrasenya" ,
"basedatos" , 3306 );
if ( $mysqli -> connect_errno ) {
echo "problema a MySQL: (" . $mysqli -> connect_errno . ") " . $mysqli ->
connect_error ;
}

echo $mysqli -> host_info . "\n" ;
```

Opcions de Connexió

Sergi Grau
sergi.grau@fje.edu

Depenent de la funció de connexió utilitzada es poden ometre diversos paràmetres. Si no es proporciona un paràmetre, l'extensió intentarà utilitzar els valors predeterminats que estan establerts en el fitxer de configuració de PHP.

Si el valor del host no està establert o és buit, la biblioteca client farà servir una connexió de sòcol Unix sobre localhost. Si el sòcol no està establert o és buit, i és sol · licitada una connexió de sòcol Unix, s'intentarà una connexió al sòcol predeterminat de / tmp / mysql.sock.

Les opcions de connexió estan disponibles per, per exemple, establir ordre inicials que són executats sobre la connexió, o per sol · licitar l'ús de certs conjunts de caràcters. Les opcions de connexió han de ser establertes abans que es establezcla una connexió de xarxa.

Per configurar una opció de connexió, l'operació de connexió ha de ser realitzada en tres passos: crear un gestor de connexió amb mysqli_init () , establir les opcions sol · licitades usant mysqli_options () , i establir la connexió de xarxa amb mysqli_real_connect () .

Memòria cau de les connexions

Sergi Grau
sergi.grau@fje.edu

L'extensió mysqli suporta connexions persistents a bases de dades, les quals són un tipus especial de connexions en memòria cau. Per defecte, cada connexió a una base de dades oberta per un script és tancada explícitament per l'usuari durant el temps d'execució o alliberada automàticament en finalitzar l'script. Una connexió persistent no. En el seu lloc, es col · loca en una memòria cau per a la seva reutilització posterior, si una connexió és oberta al mateix servidor usant el mateix nom d'usuari, contrasenya, socket, port i base de dades per defecte. La reutilització estalvia despeses de connexió.

Cada processos de PHP utilitza la seva pròpia memòria cau de connexions mysqli. Depenent de model de distribució del servidor web, un procés PHP pot servir una o múltiples peticions. Per tant, una connexió en memòria cau pot ser utilitzada posteriorment per un o més scripts.

Connexions persistents

Sergi Grau
sergi.grau@fje.edu

Si una connexió persistent no usada amb una combiació donada d'amfitrió, nom d'usuari, contrasenya, sòcol, port i base de dades per defecte no es pot trobar en la memòria cau de connexions, mysqli obrirà una nova connexió. L'ús de connexions persistents es pot habilitar i deshabilitar usant la directiva de PHP mysqli.allow_persistent . El nombre total de connexions obertes per un script pot ser limitat amb mysqli.max_links . El nombre màxim de connexions persistents per procés de PHP pot restringir amb mysqli.max_persistent .

Observa que el servidor web pot engendrar molts processos de PHP.

Una queixa comuna sobre les connexions persistents és que el seu estat no és reiniciat abans del seu ús. Per exemple, les transaccions obertes i no finalitzades no són automàticament represes. També, els canvis d'autorització que ocorrin durant la col · locació de la connexió a la memòria cau i la seva reutilització no estan reflectits. Això es pot veure com un efecte secundari no desitjat.

Connexions persistents

Sergi Grau
sergi.grau@fje.edu

L'extensió mysqli suporta dues interpretacions d'una connexió persistent: l'estat persisteix, i l'estat es reinicia abans de la reutilització. El predeterminat és la reiniciació. Abans que una connexió sigui reutilitzada, l'extensió crida implícitament a mysqli_change_user () per reiniciar l'estat. La connexió persistent apareix a l'usuari com si estigués acabada d'obrir.

La funció mysqli_change_user () és una operació costosa. Per a un millor rendiment, els usuaris poden recompilar l'extensió amb el flag de compilació MYSQLI_NO_CHANGE_USER_ON_PCONNECT establerta.

Correspon a l'usuari triar entre comportament segur o millor rendiment. Ambdues són metes d'optimització vàlides. Per facilitar l'ús, el comportament segur és el predeterminat a costa d'un rendiment màxim.

Classes de l'API

Sergi Grau
sergi.grau@fje.edu

mysqli : Aquesta classe representa una connexió entre PHP i una base de dades MySQL.

mysqli_stmt : Aquesta classe representa una consulta preparada

mysql_result: Representa el conjunt de resultats obtinguts a partir d'una consulta a la base de dades.

mysqli_driver: representa un driver per a la connexió.

mysqli_warning : Representa un advertiment de MySQL.

mysqli_sql_exception : La classe de maneig d'excepcions de mysqli.

Atributs de la classe mysqli

Sergi Grau
sergi.grau@fje.edu

- int $ affected_rows ;
- string $ client_info ;
- int $ client_version ;
- string $ connect_errno ;
- string $ connect_error ;
- int $ errno ;
- array $ error_list ;
- string $ error ;
- int $ field_count ;
- int $ client_version ;
- string $ host_info ;
- string $ protocol_version ;
- string $ server_info ;
- int $ server_version ;
- string $ info ;
- mixed $ insert_id ;
- string $ SQLSTATE ;
- int $ thread_id ;
- int $ warning_count

Mètodes de la classe mysqli

Sergi Grau
sergi.grau@fje.edu

- __construct ([string $host = ini_get ("mysqli.default_host") [, string $username = ini_get ("mysqli.default_user") [, string $passwd = ini_get ("mysqli.default_pw") [, string $dbname = " "[, int $port = ini_get ("mysqli.default_port") [, string $socket = ini_get ("mysqli.default_socket")]]]]]])
- bool autocommit (bool $mode)
- bool change_user (string $user , string $password , string $database)
- string character_set_name (void)
- bool close (void)
- bool commit (void)
- bool debug (string $message)
- bool dump_debug_info (void)
- object get_charset (void)
- string get_client_info (void)
- bool get_connection_stats (void)
- mysqli_warning get_warnings (void)
- mysqli init (void)
- bool kill (int $processid)
- bool more_results (void)
- bool multi_query (string $query)

Mètodes de la classe mysqli

Sergi Grau
sergi.grau@fje.edu

- bool next_result (void)
- bool options (int $option , mixed $value)
- bool ping (void)
- public int poll (array &$read , array &$error , array &$reject , int $sec [, int $usec])
- mysqli_stmt prepari (string $query)
- mixed query (string $query [, int $resultmode = MYSQLI_STORE_RESULT])
-)
- string escape_string (string $escapestr)
- bool real_query (string $query)
- public mysqli_result reap_async_query (void)
- public bool refresh (int $options)
- bool rollback (void)
- int rpl_query_type (string $query)
- bool select_db (string $dbname)
- bool send_query (string $query)
- bool set_charset (string $charset)
- bool set_local_infile_handler (mysqli $link , callable $read_func)
- bool ssl_set (string $key , string $cert , string $ca , string $capath , string $cipher)
- string stat (void)
- mysqli_stmt stmt_init (void)
- mysqli_result store_result (void)
- mysqli_result use_result (void)

La classe mysqli

Sergi Grau
sergi.grau@fje.edu

- mysqli :: $ affected_rows - Obté el nombre de files afectades en l'última operació MySQL
- mysqli :: autocommit - Activa o desactiva les modificacions de la base de dades autoconsignadas
- mysqli :: change_user - Canvia l'usuari de la connexió de bases de dades especificada
- mysqli :: character_set_name - Retorna el joc de caràcters per defecte per a la connexió a la base de dades
- mysqli :: $ client_info - Obté informació de la versió client de MySQL
- mysqli :: $ client_version - Retorna la versió client de MySQL com una cadena
- mysqli :: close - Tanca una connexió a base de dades prèviament oberta
- mysqli :: commit - Realitza un "commit" de la transacció actual
- mysqli :: $ connect_errno - Retorna el codi d'error de l'última trucada
- mysqli :: $ connect_error - Retorna una cadena amb la descripció de l'últim error de connexió
- mysqli :: __construct - Obre una nova connexió al servidor MySQL
- mysqli :: debug - Realitza operacions de depuració
- mysqli :: dump_debug_info - Bolcat d'informació de depuració al registre
- mysqli :: $ errno - Retorna el codi de l'error de l'última funció anomenada
- mysqli :: $ error_list - Retorna una llista d'errors des de l'últim ordre executada
- mysqli :: $ error - Retorna una cadena que descriu l'últim error

La classe mysqli

Sergi Grau
sergi.grau@fje.edu

mysqli :: $ field_count - Retorna el nombre de columnes per a la consulta més recent

- mysqli :: get_charset - Retorna un objecte que conté el conjunt de caràcters
- mysqli :: get_client_info - Obté informació de la biblioteca client de MySQL
- mysqli_get_client_stats - Retorna estadístiques de clients per procés
- mysqli_get_client_version - Retorna la versió clients de MySQL com una cadena
- mysqli :: get_connection_stats - Retorna estadístiques sobre la connexió del client
- mysqli :: $ host_info - Retorna una cadena que representa el tipus de connexió utilitzada
- mysqli :: $ protocol_version - Retorna la versió del protocol MySQL utilitzada
- mysqli :: $ server_info - Retorna la versió del servidor MySQL
- mysqli :: $ server_version - Retorna la versió del servidor MySQL com un valor sencer
- mysqli :: get_warnings - Obté el resultat de SHOW warnings
- mysqli :: $ info - Obté la informació de la darrera consulta executada
- mysqli :: init - Inicialitza i retorna un recurs per utilitzar amb mysqli_real_connect ()
- mysqli :: $ insert_id - Retorna el id autogenerat que es va utilitzar en l'última consulta
- mysqli :: kill - Demana al servidor matar un bri de MySQL
- mysqli :: more_results - Comprova si hi ha més resultats d'una multi consulta
- mysqli :: multi_query - Realitza una consulta a la base de dades
- mysqli :: next_result - Prepara el següent resultat de multi_query
- mysqli :: options - Establir opcions
- mysqli :: ping - Comprova la connexió al servidor, o tracta de reconnectar si es va perdre la connexió
- mysqli :: poll - Emmagatzema en memòria cau connexions

La classe mysqli

Sergi Grau
sergi.grau@fje.edu

- mysqli :: prepari - Prepara una sentència SQL per executar
- mysqli :: query - Realitza una consulta a la base de dades
- mysqli :: real_connect - Obre una connexió a un servidor mysql
- mysqli :: real_escape_string - Escapa els caràcters especials d'una cadena per usar-la en una sentència SQL, tenint en compte el conjunt de caràcters actual de la connexió
- mysqli :: real_query - Executa una consulta SQL
- mysqli :: reap_async_query - Obté el resultat d'una consulta asincrònica
- mysqli :: refresh - Refresca
- mysqli :: rollback - Reverteix la transacció actual
- mysqli :: rpl_query_type - Retorna un tipus de consulta RPL
- mysqli :: select_db - Selecciona la base de dades per defecte per a realitzar les consultes
- mysqli :: send_query - Enviar una consulta i tornar
- mysqli :: set_charset - Estableix el conjunt de caràcters per defecte del client
- mysqli :: set_local_infile_default - Desestablece el gestor definit per l'usuari per una ordre load local infile
- mysqli :: set_local_infile_handler - Establir la crida de retorn per la comanda LOAD DATA LOCAL INFILE
- mysqli :: $ SQLSTATE - Retorna l'error SQLSTATE de l'operació de MySQL prèvia
- mysqli :: ssl_set - Utilitzada per estableix connexions segures utilitzant SSL
- mysqli :: stat - Obté l'estat actual del sistema

La classe mysqli

Sergi Grau
sergi.grau@fje.edu

- mysqli :: stmt_init - Inicialitza una sentència i retorna un objecte per usar amb mysqli_stmt_prepare
- mysqli :: store_result - Transfereix un conjunt de resulados de l'última consulta
- mysqli :: $ thread_id - Deveulve l'ID del fil de la connexió actual
- mysqli :: thread_safe - Retorna si la seguretat a nivell de fils està donada o no
- mysqli :: use_result - Inicia la resuperación d'un conjunt de resultats
- mysqli :: $ warning_count - Retorna el nombre de missatges d'advertència de l'última consulta per a un enllaç donat

Executar sentències

Sergi Grau
sergi.grau@fje.edu

Les sentències es poden executar amb les funcions mysqli_query () , mysqli_real_query () i mysqli_multi_query () .

La funció mysqli_query () és la més comuna, i combina la sentència de execució amb el seu conjunt de resultats obtingut d'un buffer.

Cridar a mysqli_query () és idèntic de cridar a mysqli_real_query () seguit de mysqli_store_result ().

```
$mysqli = new mysqli ( "host" , "usuari" , "contrasenya" , "basedades" );
if ( $mysqli -> connect_errno ) {
echo "problema amb MySQL: (" . $mysqli -> connect_errno . ") " . $mysqli
-> connect_error ;
}

if (! $mysqli -> query ( "DROP TABLE IF EXISTS test" ) ||
! $mysqli -> query ( "CREATE TABLE test(id INT)" ) ||
! $mysqli -> query ( "INSERT INTO test(id) VALUES (1)" )) {
echo "problema: (" . $mysqli -> errno . ") " . $mysqli -> error ;
}
```

Conjunts de resultats emmagatzemats en buffer

Sergi Grau
sergi.grau@fje.edu

Després de l'execució de sentències, els resultats poden recuperar-se d'un sol cop perquè siguin emmagatzemats en buffer pel client o llegint fila a fila. El emmagatzemament en buffer de conjunts de resultats en el costat del client permet al servidor alliberar recursos associats amb els resultats de la sentència tan aviat com sigui possible. Generalment parlant, els clients són lents consumint conjunts de resultats. Per tant, es recomana utilitzar conjunts de resultats emmagatzemats en buffer. mysqli_query () combina l'execució de sentències i l'emmagatzematge en buffer de conjunts de resultats.

Les aplicacions de PHP poden navegar lliurement a través de resultats emmagatzemats en buffer. La navegació és més ràpida perquè els conjunts de resultats es mantenen en la memòria del client. Tingueu en compte que sovint és més fàcil processar dades en el client de fer el servidor.

Si no volem emmagatzemar en buffer:

```
$mysqli -> real_query ( "SELECT id FROM test ORDER BY id ASC" );
$resultado = $mysqli -> use_result ();
```

Conjunts de resultats emmagatzemats en buffer

Sergi Grau
sergi.grau@fje.edu

```
$mysqli = new mysqli ( "host" , "usuari" , "contrasenya" , "basedades" );
if ( $mysqli -> connect_errno ) {
echo "problema a MySQL: (" . $mysqli -> connect_errno . ") " . $mysqli -> connect_error ;
}

if (! $mysqli -> query ( "DROP TABLE IF EXISTS test" ) ||
! $mysqli -> query ( "CREATE TABLE test(id INT)" ) ||
! $mysqli -> query ( "INSERT INTO test(id) VALUES (1), (2), (3)" )) {
echo "Problema en la creació de la taula: (" . $mysqli -> errno . ") " . $mysqli -> error ;
}
$resultat = $mysqli -> query ( "SELECT id FROM test ORDER BY id ASC" );
echo "Ordre invers...\n" ;
for ( $num_fila = $resultat -> num_rows - 1 ; $num_fila >= 0 ; $num_fila --) {

$resultat -> data_seek ( $num_fila );

$fila = $resultat -> fetch_assoc ();

echo " id = " . $fila [ 'id' ] . "\n" ;
}

echo "Ordre normal...\n" ;
$resultat -> data_seek ( 0 );

while ( $fila = $resultat -> fetch_assoc ()) {

echo " id = " . $fila [ 'id' ] . "\n" ;
}
```

Tipus de dades dels valors del conjunt de resultats

Sergi Grau
sergi.grau@fje.edu

Els mètodes mysqli_query () , mysqli_real_query () i mysqli_multi_query () s'usen per executar sentències no preparades. Al nivell del Protocol Client Servidor de MySQL, la comanda COM_QUERY i el protocol de text s'usen per a l'execució de sentències. Amb el protocol text, el servidor MySQL converteix totes les dades dels conjunts de resultats en cadenes abans d'enviar-los. Aquesta conversió es realitza sense considerar el tipus de dades de les columnes del conjunt de resultats SQL. Les biblioteques client de mysql reben tots els valors de les columnes com cadenes. No es realitza cap conversió del costat del client al tornar a convertir les columnes a tipus nadius. En el seu lloc, tots els valors són proporcionats com cadenes de PHP.

```
$resultat = $mysqli -> query ( "SELECT id, etiqueta FROM test
WHERE id = 1" );
$fila = $resultat -> fetch_assoc ();

printf ( "id = %s (%s)\n" , $fila [ 'id' ], gettype ( $fila [ 'id'
]));
printf ( "etiqueta = %s (%s)\n" , $fila [ 'etiqueta' ], gettype
( $fila [ 'etiqueta' ]));
//en tots dos casos obtenim string
```

Tipus de dades dels valors del conjunt de resultats

Sergi Grau
sergi.grau@fje.edu

És possible convertir valors de columnes de tipus integer i float a números de PHP establint l'opció de connexió MYSQLI_OPT_INT_AND_FLOAT_NATIVE , si fem servir la biblioteca mysqlnd. Si s'estableix, la biblioteca mysqlnd comprovarà els tipus de columna de les metadades del conjunt de resultats i convertirà les columnes nombres de SQL a nombres de PHP, si el rang de valors del tipus de dades de PHP ho permet. D'aquesta manera, per exemple, les columnes INT de SQL són retornades com enters.

```
$mysqli = mysqli_init ();
$mysqli -> options ( MYSQLI_OPT_INT_AND_FLOAT_NATIVE , 1 );
$mysqli -> real_connect ( host" , "user" , "contrasenya" ,
"basedades" );
```

Sentències Preparades

Sergi Grau
sergi.grau@fje.edu

Les bases de dades MySQL suporten sentències preparades. Una sentència preparada o una sentència parametritzada s'usa per executar la mateixa sentència repetidament amb gran eficiència.

L'execució de sentències preparades consisteix en dos etapes: la preparació i l'execució. En l'etapa de preparació s'envia una plantilla de sentència al servidor de base de dades. El servidor realitza una comprovació de sintaxi i inicialitza els recursos interns del servidor per al seu ús posterior.

El servidor de MySQL suporta l'ús de paràmetres de substitució posicionals anònims amb?.

```
if (!( $sentencia = $mysqli -> prepare ( "INSERT INTO test(id) VALUES (?)"
))) {
echo "problema en la preparacio: (" . $mysqli -> errno . ") " .
$mysqli -> error ;
}
```

Sentències Preparades

Sergi Grau
sergi.grau@fje.edu

La preparació és seguida de l'execució. Durant l'execució el client vincula els valors dels paràmetres i els envia al servidor. El servidor crea una sentència des de la plantilla de la sentència i els valors vinculats per executar utilitzant els recursos interns prèviament creats.

```
$id = 1 ;
if (! $sentencia -> bind_param ( "i" , $id )) {
echo "problema en la vinculació de paràmetres: (" . $sentencia ->
errno . ") " . $sentencia -> error ;
}

if (! $sentencia -> execute ()) {
echo "Problema en la execució: (" . $sentencia -> errno . ") " .
$sentencia -> error ;
}
```

Sentències Preparades

Sergi Grau
sergi.grau@fje.edu

Una sentència preparada es pot executar repetidament. A cada execució el valor actual de la variable vinculada s'avalua i s'envia al servidor. La sentència no s'analitza de nou. La plantilla de la sentència no es fa altra vegada al servidor. Cada sentència preparada ocupa recursos del servidor. Les sentències haurien de tancar explícitament immediatament després del seu ús. Si no es realitza explícitament, la sentència serà tancada quan el gestor de la sentència sigui alliberat per PHP. “i” indica integer, s string, d double i b boolean

```
$id = 1 ;
if (! $sentencia -> bind_param ( "i" , $id )) {
 echo "Problema: (" . $sentencia -> errno . ") " . $sentencia -> error ;
}
if (! $sentencia -> execute ()) {
echo "Problema: (" . $sentencia -> errno . ") " . $sentencia -> error ;
}
for ( $id = 2 ; $id < 5 ; $id ++) {
if (! $sentencia -> execute ()) {
 echo "Problema: (" . $sentencia -> errno . ") " . $sentencia -> error ;
}
}
$sentencia -> close ();
$resultat = $mysqli -> query ( "SELECT id FROM test" );
var_dump ( $resultat -> fetch_all ());
```

Sentències Preparades

Sergi Grau
sergi.grau@fje.edu

Utilitzar una sentència preparada no és sempre la manera més eficient d'executar una sentència. Una sentència preparada executada una vegada causa més viatges d'anada i tornada des del client al servidor que una sentència no preparada. A vegades els SELECT és millor que no siguin sentències preparades.

També s'ha de considerar l'ús de la sintaxi SQL multi-INSERT de MySQL per sentències INSERT. Per exemple, multi-INSERT requereix menys viatges d'anada i tornada entre el servidor i el client que la sentència preparada pasos.

```
if (! $mysqli -> query ( "INSERT INTO test(id) VALUES (1), (2), (3),
(4)" )) {
echo "Problema amb multi-INSERT: (" . $mysqli -> errno . ") " . $mysqli ->
error ;
```

Sentències Preparades

Sergi Grau
sergi.grau@fje.edu

Les sentències preparades tornen per defecte conjunts de resultats no emmagatzemats en buffer. Els resultats de la sentència no són obtingudes i transferides implícitament des del servidor al client per l'emmagatzematge en memòria intermèdia de costat del client. El conjunt de resultats presa recursos del servidor fins que tots els resultats hagin estat obtinguts pel client. Pel que es recomana consumir resultats quan sigui oportú. Si un client falla en obtenir tots els resultats o el client tanca la consulta abans d'haver obtingut totes les dades, les dades han de ser obtinguts implícitament per mysqli.

També és possible emmagatzemar en buffer els resutats d'una sentència preparada usant mysqli_stmt_store_result () .

la interfície mysqli_result

Sergi Grau
sergi.grau@fje.edu

Utilitzar la interfície mysqli_result ofereix el benefici addicional de la navegació flexible del conjunt de resultats en el costat del client.

```
if (!( $sentencia = $mysqli -> prepare ( "SELECT id, etiqueta FROM
test" ))) {
echo "Problema: (" . $mysqli -> errno . ") " . $mysqli -> error ;
}

if (! $sentencia -> execute ()) {
 echo "Problema: (" . $sentencia -> errno . ") " . $sentencia -> error ;
}

if (!( $resultat = $sentencia -> get_result ())) {
 echo "Problema: (" . $sentencia -> errno . ") " . $sentencia -> error ;
}

for ( $num_fila = ( $resultat -> num_rows - 1 ); $num_fila >= 0 ;
$num_fila --) {
$resultat -> data_seek ( $num_fila );
var_dump ( $resultat -> fetch_assoc ());
}
$resultat -> close ();
```

Tipus de dades dels valors del conjunt de resultats

Sergi Grau
sergi.grau@fje.edu

El Protocol Client Servidor de MySQL defineix un protocol de transport de dades diferent per sentències preparades i no preparades. Les sentències preparades usen l'anomenat protocol binari. El servidor de MySQL envia les dades del conjunt de resultats "tal qual" en format binari. Els resultats no són serialitzats en cadenes abans de l'enviament. Les biblioteques client no reben cadenes només. En el seu lloc, rebran dades binàries i intentaran convertir els valors als tipus de dades de PHP apropiats. Per exemple, els resultats d'una columna de SQL INT seran proporcionats com a variables de tipus integer de PHP.

Aquest comportament difereix de les sentències no preparades. Per defecte, les sentències no preparades tornaran tots els resultats com cadenes. Aquest comportament predeterminat es pot canviar usand una opció de connexió. Si s'utilitza l'opció de connexió no existiran diferències.

Comparació entre sentències preparades i no preparades

Sergi Grau
sergi.grau@fje.edu

Comparació entre sentències preparades i no preparades

	Sentència preparada	Sentència no preparada
Viatges d'anada i tornada des del client al servidor, SELECT, execució única	2	1
Cadenes de sentències tranferidas des del client al servidor	1	1
Viatges d'anada i tornada des del client al servidor, SELECT, execució repetida (n)	1 + n	n
Cadenes de sentències tranferidas des del client al servidor	1 plantilla, n vegades parametre vinculat, si existeix	n vegades juntament amb el paràmetre, si existeix
API de vinculació de paràmetres d'entrada	Sí, escapament d'entrades automàtic	No, escapament d'entrades manual
API de vinculació de variables de sortida	Sí	No
Suport per a l'ús de l'API mysqli_result	Sí, utilitzeu mysqli_stmt_get_result ()	Sí
Conjunts de resultats emmagatzemats en buffer	Sí, utilitzeu mysqli_stmt_get_result () o la vinculació amb mysqli_stmt_store_result ()	Sí, ho predeterminat de mysqli_query ()
Conjunts de resultats no emmagatzemats en buffer	Sí, utilitzeu l'API de vinculació de sortida	Sí, utilitzeu mysqli_real_query () amb mysqli_use_result ()
"Sabor" de la transferència de dades del protocol Client Servidor de MySQL	Protocol binari	Protocol de text
Tipus de dades SQL dels valors del conjunt de resultats	Preservats al obtenir	Convertits a cadena o preservats al obetenerlos
Suporta totes les sentència SQL	Versions recent de MySQL suporten moltes però no totes	Sí

Procediments emmagatzemats

Sergi Grau
sergi.grau@fje.edu

Les bases de dades de MySQL suporten procediments emmagatzemats. Un procediment emmagatzemat és una subrutina emmagatzemada en el catàleg de la base de dades. Les aplicacions poden trucar i executar el procediment emmagatzemat. La sentència de SQL CALL s'usa per executar un procediment emmagatzemat.

```
if (! $mysqli -> query ( "DROP PROCEDURE IF EXISTS p" ) ||
! $mysqli -> query ( "CREATE PROCEDURE p(IN id_val INT) BEGIN
INSERT INTO test(id) VALUES(id_val); END;" )) {
echo "problema: (" . $mysqli -> errno . ") " . $mysqli -> error ;
}

if (! $mysqli -> query ( "CALL p(1)" )) {
echo "Problema CALL: (" . $mysqli -> errno . ") " . $mysqli ->
error ;
}
```

Sentències Múltiples

Sergi Grau
sergi.grau@fje.edu

MySQL permet opcionalment tenir múltiples sentències en una cadena de sentències. L'enviament de múltiples sentències de cop redueix els viatges d'anada i tornada des del client al servidor, però requereix un maneig especial.

Les sentències múltiples o multiconsultas han de ser executades amb mysqli_multi_query () . Les sentències individuals de la cadena de sentències estan serparadas per un punt i coma. Llavors, tots els conjunts de resultats retornats per les sentències executades s'han d'obtenir.

El servidor MySQL permet tenir sentències que retornen conjunts de resultats i sentències que no torna conjunts de resultats en una sentència múltiple.

Sentències Múltiples

Sergi Grau
sergi.grau@fje.edu

```
$sql = "SELECT COUNT(*) AS _num FROM test; " ;
$sql .= "INSERT INTO test(id) VALUES (1); " ;
$sql .= "SELECT COUNT(*) AS _num FROM test; " ;

if (! $mysqli -> multi_query ( $sql )) {
echo "problema la multiconsulta: (" . $mysqli -> errno . ") " .
$mysqli -> error ;
}

do {
if ( $resultat = $mysqli -> store_result ()) {
var_dump ( $resultat -> fetch_all ( MYSQLI_ASSOC ));
$resultat -> free ();
}
} while ( $mysqli -> more_results () && $mysqli -> next_result
());
```

Transaccions

Sergi Grau
sergi.grau@fje.edu

El servidor MySQL suporta transaccions segons del motor d'emmagatzematge usat. Des MySQL 5.5, el motor d'emmagatzematge per defecte és InnoDB. InnoDB suporta per transaccions ACID.

Les transaccions es poden controlar utilitzant SQL o crides a l'API. Es recomana utilitzar crides a la API per habilitar i deshabilitar la manera de autoconsignación (auto commit) i per consignar i reiniciar transaccions.

```
$mysqli = new mysqli ( "host" , "usuari" , "contrasenya" ,
"basedatos" );
$mysqli -> autocommit ( false );

$mysqli -> query ( "INSERT INTO test(id) VALUES (1)" );
$mysqli -> rollback ();

$mysqli -> query ( "INSERT INTO test(id) VALUES (2)" );
$mysqli -> commit ();
```