

# MySQLi prepared statements error reporting [duplicate]

Asked 10 years, 5 months ago Active 4 years, 5 months ago Viewed 106k times

This question already has an answer here:

[mysqli\\_fetch\\_assoc\(\) expects parameter / Call to a member function bind\\_param\(\) errors. How to get the actual mysql error and fix it?](#) (1 answer)

Closed 3 years ago.

67

41

I'm trying to get my head around MySQLi and I'm confused by the error reporting. I am using the return value of the MySQLi 'prepare' statement to detect errors when executing SQL, like this:

```
$stmt_test = $mysqliDatabaseConnection->stmt_init();
if($stmt_test->prepare("INSERT INTO testtable VALUES (23,44,56)"))
{
    $stmt_test->execute();
    $stmt_test->close();
}
else echo("Statement failed: ". $stmt_test->error . "<br>");
```

But, is the return value of the prepare statement only detecting if there is an error in the preparation of the SQL statement and not detecting execution errors? If so should I therefore change my execute line to flag errors as well like this:

```
if($stmt_test->execute()) $errorflag=true;
```

And then just to be safe should I also do the following after the statement has executed:

```
if($stmt_test->errno) {$errorflag=true;}
```

...Or was I OK to start with and the return value on the MySQLi 'prepare' statement captures all errors associated with the complete execution of the query it defines?

Thanks C

[php](#) [mysql](#) [mysqli](#) [prepared-statement](#)

edited May 5 '12 at 15:24



[MPelletier](#)

14.6k ● 12 ● 77 ● 124

asked Mar 31 '10 at 11:43



[Columbo](#)

2,575 ● 7 ● 35 ● 51

Why do you use prepare/execute() instead of query() in the first place when there is no variable part in your query string? Or is this just an oversimplified example? – [VolkerK](#) Mar 31 '10 at 14:20

Yes, sorry. It was simplified to show how I was having difficulty understanding where to get definitive error reporting from a prepared statement. – [Columbo](#) Mar 31 '10 at 14:25





I wrote this twice before in the last two days (so for me it's a duplicate even though the questions started a bit different).

Each method of mysqli can fail. You should test each return value. If one fails, think about whether it makes sense to continue with an object that is not in the state you expect it to be. (Potentially not in a "safe" state, but I think that's not an issue here.)

Since only the error message for the last operation is stored per connection/statement you might lose information about *what* caused the error if you continue after something went wrong. You might want to use that information to let the script decide whether to try again (only a temporary issue), change something or to bail out completely (and report a bug). And it makes debugging a lot easier.

```

$stmt = $mysqli->prepare("INSERT INTO testtable VALUES (?, ?, ?)");
// prepare() can fail because of syntax errors, missing privileges, ....
if ( false=== $stmt ) {
    // and since all the following operations need a valid/ready statement object
    // it doesn't make sense to go on
    // you might want to use a more sophisticated mechanism than die()
    // but's it's only an example
    die('prepare() failed: ' . htmlspecialchars($mysqli->error));
}

$stmt->bind_param('iii', $x, $y, $z);
// bind_param() can fail because the number of parameter doesn't match the placeholders in the
// statement
// or there's a type conflict(?), or ....
if ( false=== $rc ) {
    // again execute() is useless if you can't bind the parameters. Bail out somehow.
    die('bind_param() failed: ' . htmlspecialchars($stmt->error));
}

$stmt->execute();
// execute() can fail for various reasons. And may it be as stupid as someone tripping over
// the network cable
// 2006 "server gone away" is always an option
if ( false=== $rc ) {
    die('execute() failed: ' . htmlspecialchars($stmt->error));
}

$stmt->close();

```

edit: just a few notes six years later....

The mysqli extension is perfectly capable of reporting operations that result in an (mysqli) error code other than 0 via exceptions, see [mysqli\\_driver::\\$report\\_mode](#).

[die\(\)](#) is really, really crude and I wouldn't use it even for examples like this one anymore.

So please, only take away the fact that *each and every* (mysql) operation *can* fail for a number of reasons; even *if* the exact same thing went well a thousand times before....

edited Mar 24 '16 at 20:59

answered Mar 31 '10 at 14:45




VolkerK

90k ● 17 ● 152 ● 219

- 2 Thanks. I've just done some testing and can see what you are saying. If I create a query that inserts a duplicate primary key value into a table then checking the prepare only will not reveal that the insertion failed. On the other hand if I don't check the prepare then the execute will never happen (I'll get some warnings IF warnings are switched on). So I can see that you're right. Thanks. – [Columbo](#) Mar 31 '10 at 14:53



1 if `bind_param()` fails due to Number of elements in type definition string doesn't match number of bind variables. it triggers an `E_WARNING` error. and you will not find that error in `$stmt->error` property – Accountant  Jan 16 '17 at 10:10

## Completeness

17

You need to check both `$mysqli` and `$statement`. If they are false, you need to output `$mysqli->error` or `$statement->error` respectively.



## Efficiency

For simple scripts that may terminate, I use simple one-liners that trigger a PHP error with the message. For a more complex application, an error warning system should be activated instead, for example by throwing an exception.

### Usage example 1: Simple script

```
# This is in a simple command line script
$mysqli = new mysqli('localhost', 'buzUser', 'buzPassword');
$q = "UPDATE foo SET bar=1";
($statement = $mysqli->prepare($q)) or trigger_error($mysqli->error, E_USER_ERROR);
$statement->execute() or trigger_error($statement->error, E_USER_ERROR);
```

### Usage example 2: Application

```
# This is part of an application
class FuzDatabaseException extends Exception {
}

class Foo {
    public $mysqli;
    public function __construct(mysqli $mysqli) {
        $this->mysqli = $mysqli;
    }
    public function updateBar() {
        $q = "UPDATE foo SET bar=1";
        $statement = $this->mysqli->prepare($q);
        if (!$statement) {
            throw new FuzDatabaseException($mysqli->error);
        }

        if (!$statement->execute()) {
            throw new FuzDatabaseException($statement->error);
        }
    }
}

$foo = new Foo(new mysqli('localhost', 'buzUser', 'buzPassword'));
try {
    $foo->updateBar();
} catch (FuzDatabaseException $e) {
    $msg = $e->getMessage();
    // Now send warning emails, write log
}
```


edited Jan 8 '14 at 23:41

answered Dec 13 '13 at 14:05



cmc



1. die() shouldn't be used ever. 2. mysqli is able to throw exceptions by itself, see mysqli\_report() 2. Writing code for "send warning emails, write log" after every database-driven functions is awfully redundant – [Your Common Sense](#) Dec 13 '13 at 14:12 

Anyway you should never use mysqli API as is, but only wrapped in some higher level library. – [Your Common Sense](#) Dec 13 '13 at 14:53

3 Mentioning both \$mysqli->error AND \$statement->error just saved my day, thks^^ – [Fernando Silva](#) Jan 3 '14 at 16:57

2 @FernandoSilva Glad to hear that Fernando, and you just made mine! – [cmc](#) Jan 6 '14 at 21:51

1 @YourCommonSense For the app, yeah it would likely be good to reduce the boilerplate, but how to best do that depends on the app. The example is certainly a decent starting point. – [cmc](#) Jan 9 '14 at 0:07

▲ Not sure if this answers your question or not. Sorry if not

4 ▼ To get the error reported from the mysql database about your query you need to use your connection object as the focus.

↻ so:

```
echo $mysqliDatabaseConnection->error
```

would echo the error being sent from mysql about your query.

Hope that helps

answered Mar 31 '10 at 12:04



[andyface](#)

875 ● 1 ● 9 ● 23

Thanks. So if I request the error for the actual connection object then it will give me the last error for that connection. Since the execution will only succeed if all the previous steps have succeeded then this will tell me if all went well. I suppose the same result could also be generated by just checking the error for the execute command as endorsed below by Col Shrapnel. Am I right in thinking therefore that checking the success/fail flag of the prepare statement serves no real purpose? – [Columbo](#) Mar 31 '10 at 13:27

I think you may already have your answer above, but will still reply out of courtesy. Essentially Prepare can fail, as VolekrK said, but it wont return a mysql error. So you need to find out why prepare failed by getting the mysql connection error which will give you an indication of where the query in your prepare statement failed. I'm not sure about the execute command error. – [andyface](#) Apr 6 '10 at 9:38

