Parameterized SQL Columns?

Asked 16 years, 3 months ago Modified 14 years, 9 months ago

Viewed 3k times

9

I have some code which utilizes parameterized queries to prevent against injection, but I also need to be able to dynamically construct the query regardless of the structure of the table. What is the proper way to do this?

Here's an example, say I have a table with columns Name, Address, Telephone. I have a web page where I run **Show Columns** and populate a select drop-down with them as options.

Next, I have a textbox called **Search**. This textbox is used as the parameter.

Currently my code looks something like this:

```
result = pquery('SELECT * FROM contacts WHERE `' +
escape(column) + '`=?', search);
```

I get an icky feeling from it though. The reason I'm using parameterized queries is to avoid using **escape**. Also, **escape** is likely not designed for escaping column names.

How can I make sure this works the way I intend?

Edit: The reason I require dynamic queries is that the schema is user-configurable, and I will not be around to fix anything hard-coded.

sql-injection parameterized

Share

Improve this question

Follow

edited Mar 4, 2010 at 18:57

Jon Seigel

12.4k ● 8 ● 60 ● 93

asked Sep 19, 2008 at 21:59

Martin

6,015 ● 7 ● 51 ● 79

6 Answers

Sorted by: Highest score (default)

7

Instead of passing the column names, just pass an identifier that you code will translate to a column name using a hardcoded table. This means you don't need to worry about malicious data being passed, since all the data is either translated legally, or is known to be invalid. Psudoish code:

```
@columns = qw/Name Address Telephone/;
if ($columns[$param]) {
  $query = "select * from contacts where $columns[$par
} else {
  die "Invalid column!";
}
```

```
run_sql($query, $search);
```

Share Improve this answer

Follow

answered Sep 19, 2008 at 22:03

zigdon

15k ● 6 ● 37 ● 54

1 Hard coding is not an option for me, thank you for the suggestion! – Martin Sep 19, 2008 at 22:07

My point is not to hardcode the column names, but not to pass them from the interface. You can get the list of columns dynamically, and still use the same solution above. Just don't make the column names part of the data passed -- that will protect you from any SQL injection. – zigdon Sep 20, 2008 at 1:38

1 +1 Yes, I like to say: let users enter data, but do not let users supply code. – Bill Karwin Jun 25, 2010 at 18:23

0

The trick is to be confident in your escaping and validating routines. I use my own SQL escape function that is overloaded for literals of different types. Nowhere do I insert expressions (as opposed to quoted literal values) directly from user input.

Still, it can be done, I recommend a separate — and strict — function for validating the column name. Allow it to accept only a single identifier, something like

```
/^\w[\w\d_]*$/
```

You'll have to rely on assumptions you can make about your own column names.

Share Improve this answer

Follow

answered Sep 19, 2008 at 22:10

harpo

43.1k ● 14 ● 100 ● 133

0

I use ADO.NET and the use of SQL Commands and SQLParameters to those commands which take care of the Escape problem. So if you are in a Microsoft-tool environment as well, I can say that I use this very sucesfully to build dynamic SQL and yet protect my parameters

best of luck

Share Improve this answer

Follow

answered Sep 19, 2008 at 22:12

Brad Osterloo

841 ● 5 ● 9

0

Make the column based on the results of another query to a table that enumerates the possible schema values. In that second query you can hardcode the select to the column name that is used to define the schema. if no rows are returned then the entered column is invalid.

Share Improve this answer

Follow

answered Sep 19, 2008 at 22:15

Turnkey

9,406 ● 3 ● 29 ● 36

0

In standard SQL, you enclose delimited identifiers in double quotes. This means that:

```
SELECT * FROM "SomeTable" WHERE "SomeColumn" = ?
```

will select from a table called SomeTable with the shown capitalization (not a case-converted version of the name), and will apply a condition to a column called SomeColumn with the shown capitalization.

Of itself, that's not very helpful, but...if you can apply the escape() technique with double quotes to the names entered via your web form, then you can build up your query reasonably confidently.

Of course, you said you wanted to avoid using escape - and indeed you don't have to use it on the parameters where you provide the ? place-holders. But where you are putting user-provided data into the query, you need to protect yourself from malicious people.

Different DBMS have different ways of providing delimited identifiers. MS SQL Server, for instance, seems to use square brackets [SomeTable] instead of double quotes.

Share Improve this answer

Follow

answered Sep 19, 2008 at 22:16

Jonathan Leffler

752k ● 145 ● 946 ● 1.3k

0

Column names in some databases can contain spaces, which mean you'd have to quote the column name, but if your database contains no such columns, just run the column name through a regular expression or some sort of check before splicing into the SQL:

```
if ( $column !~ /^\w+$/ ) {
  die "Bad column name [$column]";
}
```

Share Improve this answer Follow

edited Oct 17, 2008 at 3:13

answered Sep 19, 2008 at 22:16

runrig

6,524 ● 2 ● 30 ● 46