

## Re: Problem with sql statement in VB

**Source:** <http://www.tech-archive.net/Archive/VB/microsoft.public.vb.database.ado/2004-10/0140.html>

---

**From:** Wart (*nospamWart\_at\_epix.net*)

**Date:** 10/07/04

Date: Thu, 7 Oct 2004 08:51:35 -0400

Generally I agree with you, especially where input may be derived from uncertain sources.

However, I think that your admonition to "never use concatenation to build dynamic SQL in this way" is overly didactic. Concatenation is a valid tool to use under certain circumstances, such as using the results of one query to build a second query or as parameters to a second query, for instance. There are circumstances where the number of parameters that might be submitted to a query are not known until a previous result set is analysed.

CF

"Val Mazur" <group51a@hotmail.com> wrote in message news:ujzt2A1qEHA.3744@TK2MSFTNGP10.phx.gbl...

> *Of course you could use, but you never know what are special characters, which should be doubled. It is not just a single quote, but some other characters as well. It also could be different set of these characters and you would need to alter your code to handle it. Using queries with the parameters would solve this issue, because provider will handle all the special characters and all the issues related to the regional settings on a local PC*

>

> --

> *Val Mazur*

> *Microsoft MVP*

>

>

> *"Wart" <nospamWart@epix.net> wrote in message*

> *news:ert2wFmqEHA.1988@TK2MSFTNGP09.phx.gbl...*

>> *I went and looked around at the various postings regarding SQL Injection*

>> *attacks and it looks like there are three main suggestions to avoid this*

>> *type of attack.*

>> *The first is to verify that the parameters returned by the user are*

>> *reasonable. If you are looking for a numeric, check that it is numeric.*

>> *If the expected field is 10 characters long, verify if it is 10*

>> *characters long.*

>> *The second was to pass the parameters through a "Bad character" function*

>> *that looks for semi colons, double dashes, apostrophes, etc. and throws*

>> *an error if one is found.*

>> *The third is to pass the parameters through a function that replaces all*

>> *single apostrophes with a double apostrophe. This will cause most*

>> *malicious insertions to become unintelligible to SQL.*  
>> *There were a bunch of other "good programming" suggestions regarding the*  
>> *building of the queries and aliasing the tables that you can review by*  
>> *searching for SQL Injection Attack. I found it interesting browsing.*  
>> *But the bottom line is that you can use concatenated queries if you*  
>> *verify the parameters before you plug them in to the query. It may not be*  
>> *as secure as the parameterized solution, but it is doable and will turn*  
>> *away the usual insertion attack explorations.*  
>> *CF*  
>> *"Val Mazur" <group51a@hotmail.com> wrote in message*  
>> *news:uudtc1KqEHA.556@TK2MSFTNGP11.phx.gbl...*  
>>> *Hi Corey,*  
>>>  
>>> *There are two issues here. First – never use concatenation to build*  
>>> *dynamic SQL this way, because it could lead to SQL injection attack from*  
>>> *the hackers. Second, to avoid any issues with the dates format or number*  
>>> *formats, use SQL statements with parameters. In this case provider will*  
>>> *handle all regional settings properly. Using parameterized queries*  
>>> *solves both issues. Your code will be longer, but it is worth it. It*  
>>> *would be like (assuming that UD.EmployeeID is integer datatype)*  
>>>  
>>> *Dim Cmd1 As ADODB.Command*  
>>> *Dim Param1 As ADODB.Parameter*  
>>> *Dim Rs1 As ADODB.Recordset*  
>>>  
>>> *SQLText = "Select EmployeeID, [Date], Status From TimeTable Where*  
>>> *EmployeeID = ? And [Date] = ? And Status = Yes"*  
>>>  
>>> *Set Cmd1 = New ADODB.Command*  
>>> *Set Cmd1.ActiveConnection = Conn1*  
>>> *Cmd1.CommandText = SQLText*  
>>>  
>>> *Set Param1 = Cmd1.CreateParameter(, adInteger, adParamInput)*  
>>> *Param1.Value = UD.EmployeeID*  
>>> *Cmd1.Parameters.Append Param1*  
>>>  
>>> *Set Param1 = Cmd1.CreateParameter(, adDate, adParamInput)*  
>>> *Param1.Value = SysDate*  
>>> *Cmd1.Parameters.Append Param1*  
>>>  
>>> *Set Rs1 = Cmd1.Execute()*  
>>>  
>>>  
>>> *--*  
>>> *Val Mazur*  
>>> *Microsoft MVP*  
>>>  
>>>  
>>> *"Corey" <Corey@discussions.microsoft.com> wrote in message*  
>>> *news:A0269693-DCBF-4213-8959-CD624869F50B@microsoft.com...*  
>>>> *I'm trying to pull some records from a table based on two criteria as*

```
>>>> follows.
>>>>
>>>> SQLText = "Select EmployeeID, Date, Status " & _
>>>> "From TimeTable " & _
>>>> "Where EmployeeID = '" & UD.EmployeeID & "' " & _
>>>> "And Date = '" & SysDate & "' " & _
>>>> "And Status = Yes;"
>>>>
>>>> The Date field in the table is a date format (short date) and the
>>>> variable
>>>> (sysdate) local to the vb application has been dimensioned as a date
>>>> variable. When running the statement gives me a data type mismatch
>>>> error in
>>>> criteria statement. I know the error isn't getting caught up on the
>>>> employeeID part of the statement because if I change the Date field
>>>> data type
>>>> in the Access table to a Text format this same SQL statement works just
>>>> fine.
>>>> But I need the Date field in the table to be a date format for later
>>>> SQL
>>>> comparisons and data filtering.
>>>>
>>>>
>>>>
>>>>
>>>>
>>>>
>>>>
>>>>
>>>>
```