

Re: Performance concerns in SPs and Tables

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-04/3364.html>

From: Adam Machanic (*amachanic_at_air-worldwide.nospamallowed.com*)

Date: 04/20/04

Date: Tue, 20 Apr 2004 13:04:50 -0400

I'll try to tackle as many of these as I can... comments inline:

"Carlitos" <criveraf@infomedika.com> wrote in message
news:#rP6riuJEHA.2580@TK2MSFTNGP12.phx.gbl...

> 1- *Is there anything wrong to use varchar all the time in stored
procedures?*

> 2- *Same question as #1 for variable declarations inside stored procedures.*

Yes, there are definitely numerous problems with this approach. My first question would be, "why not use strong typing?" Data typing is a feature of any DBMS, because it helps ensure data integrity. Relying on implicit conversions may work most of the time, but at some point it may not give you the expected result. Are you willing to take that risk? You also mentioned performance penalties. Yes, there is a chance you will encounter some. There is no guarantee, for instance, that when you execute a query like the following, that the variable will be implicitly converted to the right datatype first, instead of the column being converted once for each row (causing extreme performance degradation due to lack of index usage):

```
DECLARE @DateString VARCHAR(20)
SET @DateString = '20040201'
```

```
SELECT *
FROM MyTable
WHERE MyRealDateColumn = @DateString
```

There should never be a reason NOT to use strong datatyping. It's there to help you keep your data clean. Remember, all pieces of data in a DBMS require three things: An attribute (a name), a type, and a value. Lose any of these, and you no longer have data.

> 3- *When does the memory get allocated: upon assignment of values or just
right*

> *on declaration? I believe is on assignment. Is it true that if I declare*

> *many varchar(8000) variables or parameters I will mine the server's memory*

> *or cause a memory leak?*

I'm not sure about the first part of this question, although I would assume it would depend on what datatype you're using; for an integer, I would expect memory to be allocated on declaration; for a varchar, I would expect more dynamic behavior. I've never heard of a memory leak problem due to varchar variables.

> 4—Is there a memory leak reported or suspected to exist in the use of
> certain objects or instructions in a stored procedure?

Did you search support.microsoft.com for this?

> 5—I have heard that Microsoft does not recommend nesting calls to stored
> procedures more than 2–3 levels. Is this true? I have search the
internet
> for hours looking for a statement, article, hint, tip (or whatever) that
> would confirm this, but had no success, so I have my doubts. In fact, it
> seems illogical to recommend to restrain the use of nested stored
procedures
> because that is one of the most valuable fundaments of using stored
> procedures: to be able to program in a modular fashion and minimize
network

I don't know what Microsoft recommends, but I don't feel that nesting is fundamental in any way to using stored procedures. Nesting is used for procedural code; database access code should be declarative. If you have a lot of procedural code (loops, nesting, etc), you need to review one or more of the following: A) Your database architecture — is your data denormalized, forcing you to use procedural code? B) Your database developer — does he or she know how to write SQL declaratively? C) Your database usage — are you using the database for something it's not intended to be used for? (search the archives of this group for "reporting" and/or "cross-tab" and you'll find plenty of arguments about that one)

> 6—I know I can declare up to 1,024 parameters in a stored procedure.
> However, is there any performance concern that I need to observe in
regards
> of how many parameters are declared? We have stored procedures of about
100
> parameters, some of them declared as varchar(8000). Is it a good
practice,
> a bad practice or it simply won't matter? What would be the maximum
> recommended?

Declare as many parameters as you need to declare. If you're importing data into a very large table via a stored procedure, you certainly may need quite a few parameters. Nothing wrong with that...

> 8— Which is a better approach: to use temporary tables or variables of
type
> table? I know one of the properties of using temporary tables is that it
> has a scope beyond the stored procedure that creates it. But we are not

> *quite sure about what would be the best time to use a temporary table*

versus

> *a variable of type table, and the performance concerns of using one or the*

> *other (some say temporary tables are better in terms of use of resources,*

> *other say variables of type table are better).*

This is a difficult question to answer. I have heard reports of people getting performance benefits from table variables vs. temporary tables, but have also heard that table variables tend to perform better for 'smaller' datasets, whereas temporary tables should be used for larger temporary storage. Personally, I have never found table variables to outperform temporary tables in my tests. The best option, though, if possible (and it's certainly not always possible), is to get rid of as much temporary storage as you can; it's indicative of procedural code (see above), and a sign of one of the three problems I mentioned.