

Re: Dynamic SQL

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-05/4857.html>

From: Aaron Bertrand – MVP (aaron_at_TRASHasppfaq.com)

Date: 05/21/04

Date: Fri, 21 May 2004 11:09:43 -0400

Also stored procedures eliminate code redundancy... if you have SQL code that is called from many places, you have to change many places. If the code is in a stored procedure, you only change in one place.

--

Aaron Bertrand
SQL Server MVP

<http://www.asppfaq.com/>

"Louis Davidson" <dr_dontspamme_sql@hotmail.com> wrote in message
news:eechNO0PEHA.1644@TK2MSFTNGP09.phx.gbl...

> Dynamic SQL versus stored procedures is really a minimal discussion.
> Obviously if you CAN use stored procedures, you are going to have a MUCH
> better system in the end. But will it kill you? No. You lose:
>
> * Encapsulation - so you will never be able to be 100% sure of the queries
> that are being sent. If all calls are stored procedures, any changes that
> do
> not affect the public interface are easy. Certainly do not take a
> recompile,
> and testing can be limited to the procedure that has changed. Table
> names,
> structures, etc, can change and you can make a few procedure changes and
> no
> change to the UI unless necessary.
>
> * Security - procs give very nice granular security to activities. This
> is
> not a tremendous problem, as long as your middle tier is built with
> security
> in mind.
>
> * Performance - This is less of a problem in 2000 (maybe 7.0) since they
> cache plans of even dynamic calls.
>
> With only minimal modification you could make your middle tier use stored
> procedures, and greatly reduce the middle tier code. All that would be
> required is to take calls like:
>
> select col1, col2 from table where col3 = 'Fred'
>
> and translate to:
>
> exec table_getCol1Col2 @col3='Fred'
>
> As the complexity of SQL grows, this becomes more and more the easier way.

microsoft.public.sqlserver.programming: Re: Dynamic SQL

```
>
> Bottom line is simple.  You can use dynamic SQL, and you will "probably"
not
> have any problems (you would not be the only person doing it, and most
> package software doesn't use stored procedures for "portability" reasons.)
> It will definitely be easier to code to start with.  But in the future, it
> may come back and bite you hard, especially if you don't write good SQL
> code.  If your SQL is encapsulated into procedures, you just update the
> procedures.  If the load on the system changes appreciably, and database
> performance suffers, then you will not have as easy a time taking care of
> that either.
>
> Good luck!
>
>
> --
> -----
--
> Louis Davidson (drsql@hotmail.com)
> Compass Technology Management
>
> Pro SQL Server 2000 Database Design
> http://www.apress.com/book/bookDisplay.html?bID=266
>
> Note: Please reply to the newsgroups only unless you are
> interested in consulting services.  All other replies will be ignored :)
>
> "Mark" <anonymous@discussions.microsoft.com> wrote in message
> news:102df01c43f38$803e1fd0$a301280a@phx.gbl...
> > I am about to start new project.  This is an HR application.
> > New program should be written in C# which is completely
> > new for me.
> > I have found a few modelers / code generators that I can
> > use to build my
> > business framework and data access components.
> > I particularly like one product.  The only problem (may be
> > it's not a problem)
> > with it creates data components that use dynamic SQL.  So
> > all business rules
> > will be in a middle tier.  No stored procedures.
> > While I am not trying to start another discussion about
> > business rules and where they
> > supposed to live but I need major reasons what's right and
> > what's wrong.
> > I think I am looking for something to convince myself that
> > dynamic SQL is OK.
> >
>
>
```