RESOURCES
About Me
discourse.org
stackexchange.com
Learn Markdown
Recommended Reading

Subscribe in a reader
Subscribe via email

17 May 2005

Stored Procedures vs. Ad-Hoc SQL

In a recent article, Doug Reilly makes a fairly well reasoned case for the use of stored procedures in lieu of ad-hoc SQL:

> So, should you use SPs or ad-hoc SQL? The answer is "it depends." I have placed myself firmly on the side of doing all database access through SPs. I do so knowing that I am not getting any unique security benefits using SPs, knowing that the performance benefits are not as clear cut as I once might have thought (but are still real in some cases), knowing how to leverage SPs to minimize the maintenance load, and understanding that I am more tied to SQL Server than I might be if I were to use ad-hoc SQL. What do you think?

There's excellent followup commentary on his blog entry for this article. In the comments, Frans Bouma immediately links to a formal debate at TheServerSide on the same topic, which he also participated in.

I agree with Doug when he says the answer is "it depends." However, as I've said before, I think it's generally better to err on the side of simplicity whenever possible. Writing a bunch of mindless stored procedures to perform every database operation you *think* you may need is definitely not what I'd call simple. Parameterized SQL, on the other hand, really is simple. Safe and fast too. I'm certainly not ruling out the use of stored procedures, but to *start* with procs? That seems like a fairly extreme case of premature optimization to me.

At the risk of repeating myself, I've observed two recurring themes in these discussions that I don't feel are being properly addressed:

1. **If your primary goal is abstraction, stored procedures are a terrible place to do that.**

   The idea that you're abstracting away the database (for reasons of access control, coherency, etcetera) by creating a stored procedure "API" is weak at best. Stored procedures only provide the *illusion of abstraction*. They're incredibly tightly coupled to the database. Make a few changes to the tables and your procs are toast-- just like parameterized SQL. Contrast that with a web service, which provides nearly infinite opportunities for designing an API with access control, abstraction, and decoupling. All accessible from port 80 on any platform, and without the inevitable limitations of your particular vendor's stored procedure implementation and database language.

2. **Embedding domain-specific languages in your code is a *good* thing.**

   Some programmers sneer at the idea of "naked SQL statements clumsily embedded in other languages". This is insane. On the contrary, you should embrace as many domain-specific languages in as much of your code as possible! Use SQL to manipulate set-based data, Regular Expressions to manipulate strings, VB.NET to do COM interop, and C# for bitwise operations. Why in the world would you write a 3-level deep For..Next loop to manipulate a string when you can express that same logic in 12 characters of regex? If anything, we should be railing against the stupidity of being limited to a single, general-purpose language!

Of course, your mileage may vary; every project is different. And always measure actual performance before jumping to any conclusions either way.

NEXT
A Group Is Its Own Worst Enemy

PREVIOUS
The Code-First Dictum

Written by Jeff Atwood

Indoor enthusiast. Co-founder of Stack Overflow and Discourse. Disclaimer: I have no idea