

Design Tip #70 Architecting Your Data For Microsoft SQL Server 2005

By Joy Mundy

Like many of you, Warren Thornthwaite and I (Joy) have been working with the pre-release versions of Microsoft SQL Server 2005. We're also putting finishing touches on our new book, *The Microsoft Data Warehouse Toolkit* (Wiley, December 2005). One of the issues we've grappled with, for the book and our consulting SQL Server clients, is whether you should implement your Microsoft-based DW/BI system in the relational database, Analysis Services, or both? The answer in most cases is both.

For years people have been building relational data warehouses, so obviously that's possible. Microsoft has added some interesting functionality into the Analysis Service 2005 dimensional database engine that makes it possible—in theory at least—to implement the dimensional database directly from a transactional system, skipping the relational data warehouse. But is that a good idea?

Before I answer that question, I'm going to argue that in any case your Microsoft DW architecture should include Analysis Services. For successful ad hoc access against even a simplified dimensional model, you need a user-oriented layer that streamlines navigation, performs complex calculations and aggregate navigation, and manages data security. For many years people have used relational techniques like views and client-side query tools to deliver this functionality. A dimensional engine like Analysis Services provides an attractive alternative via the following features:

- User-oriented metadata
- Complex analytics stored in the database
- Richness of the analytic language (MDX rather than SQL)
- Query performance
- Aggregate management
- Rich security model
- Server-based—important for scalability, performance, and sharing of metadata across multiple query tools.

Although Analysis Services has become a popular component of SQL Server 2000, there are still several common objections to using Analysis Services:

- Scalability
- Duplication of data
- Changing existing end-user applications.

We find only the third objection to be broadly compelling. Worries about scalability and data duplication have been addressed effectively in SQL Server 2005, and shouldn't prevent the vast majority of implementations from reaping the very real benefits of a data warehouse / business intelligence system built on Analysis Services.

If I've convinced you that Analysis Services is a vital part of your Microsoft data warehouse architecture, your next question may be: why do you even need to store the dimensional data in the RDBMS? You aren't required to do so: Microsoft provides several mechanisms for populating Analysis Services cubes directly from non-dimensional source systems. Why go to the trouble and expense of populating the relational store in addition to Analysis Services? Here's why:

- Conformation of dimensions and facts. In a hypothetical, simple example you could conform data on the way into the Analysis Services database. In the real world, you will have to

update and delete some data in the ETL pipeline, and you really want to do this in a relational database.

- Disaster recovery. The tools and knowledge for managing a relational database for easy recovery are better than those for Analysis Services, though management tools are much improved in the new version.
- Comfort. DBAs and power users are very familiar with SQL and relational databases, and may violently resist the elimination of the relational layer.
- Query flexibility. If you want to modify an Analysis Services database, you usually need to redeploy and reprocess a large chunk of the database. It's much easier to "join and go" in the relational world.
- Future flexibility. The notion of eliminating the relational data warehouse and populating the Analysis Services database directly from transaction systems may sound elegant. But if you choose this architecture, you're sailing on a Microsoft ship with a non-refundable ticket.

There are a few scenarios where the best choice is to skip the relational storage of the dimensional data. But these are edge cases which I'll write about in a future Design Tip. Most of us, most of the time, should plan to store and manage the dimensional data in the relational database, and use that store to feed Analysis Services. Think of the Analysis Services layer primarily as metadata for the dimensional engine, with an ephemeral data cache—similar in spirit to relational indexes—that greatly improves query performance.