CHAPTER 4

The DBA as Designer

by Melanie Caffrey

Consider the following scenario. It is Friday, five o’clock p.m. You have plans for the evening. You are signing out and logging off, when your phone rings. That new application that was just moved into production appears to be having problems. The question posed to you is “Can you log on and see if there is something wrong with the database?” Something wrong with the database. How many times have you heard a question posed like that? And because production code is deemed the responsibility of the DBAs at your company, this problem now falls to you and your team.

So you log in to the new application and notice several problems right away:

- Clicking a link (presumably parameterized) presents you with the familiar hourglass icon for what seems like an eternity.
- The application allows the user to dynamically create Data Manipulation Language (DML) statements, and when you do so, with very few filters, you again find yourself staring at the hourglass icon for an interminable amount of time.
- You decide to have a look at exactly how many users are accessing this application right now and discover that it is fewer than a hundred. Some of them have active sessions that have been working for nearly 20 minutes.

In the middle of this research, you are approached by a rather panicked project manager who implores, “Can you please do something with the database to fix this?” Soon the panicked project manager is joined by several lead developers and perhaps even one rather annoyed chief information officer. All of them want to know what is wrong with production. Because this code was tested and it was working! So what is wrong with the database?

If you are like many DBAs at many companies, you have no idea what this application is or what it’s supposed to do. Earlier that afternoon, one of the DBAs on your team was given a set of scripts to run to create this application. The DBA ran the set of scripts that created a nice resultant log file ensuring him, as well as the developers who gave him the scripts, that all objects and code were created without error. Like many DBAs, he has been told that the developers are in charge of designing and writing applications.

And if you were to ask him to look into it, he might immediately begin to increase temporary space, or create some extra indexes or redo logs, or implement that cool new initialization parameter setting he just read about on insert_your_favorite_listserve_here. Because, if he gets it right tonight—even if it’s just for tonight—in alleviating some of the pain of the running of this application, he will be the company hero. And what could be better? It may not even occur to him to start pulling aside one or more of the lead developers leaning on the back of his chair to start asking questions about the
design of this application. But it should. It could save him hours of troubleshooting time and frustration (not to mention the money it could potentially save the company).

Therefore, you decide to look into it yourself. And if you’re smart, the first question you ask lead developer 1 and lead developer 2 is “Why weren’t the DBAs involved in the design of this application?” After you ask this question, you’ll still need to help solve the problem. However, your approach will now be (hopefully) different from that taken by your cowboy colleague illustrated earlier. You now need to try to figure out a way to introduce design improvements that are scalable and maintainable while causing minimal interruption to what is now in place in production. Good luck!

When to Get Involved in Application Design

Ideally, a DBA gets involved in application design right from the beginning. If you are not invited to an initial planning meeting for a brand new application (or even an existing one that will be overhauled), invite yourself. Trust me—it’s worth the investment of your time. For as any experienced DBA knows, though the poor design in production may not be your fault (if you were never involved in its inception or implementation), it will certainly become your problem. Be proactive, not reactive. Burnout takes place quickly when you have little say in a problem you are asked (and expected) to fix. Though poor communication and lack of proper planning is a reason why many DBAs refer to developers as duh-velopers, there is little good that can be said about a DBA who does not make herself available to the development team for design consultation and collaboration.

In other words, if you are not helping the design effort as soon as possible, especially if you are aware that it is taking place, you are part of the problem. Given that you have enough to do in your day—what with new systems, data warehousing efforts, reorganizations, and what-have-you—spending a little time up front to alleviate the need for spending lots of time later on, sometimes late at night, under duress, seems like a small sacrifice in the long run. Putting yourself on the radar of the development team and those in charge of gathering requirements and planning project schedules, though initially may seem tedious, will save you, personally, many hours, many headaches, and hopefully a few friendships.

So how do you do that? There are several ways in which very skilled DBAs accomplish this task designed (no pun intended) to make everyone’s lives easier. Personally, I have found that adhering to the following modus operandi works well for me.

DBAS AND DEVELOPERS

This chapter focuses on the relationship between DBAs and developers (and others involved in application design sessions involving the database, solely and explicitly). To keep the chapter at a manageable length while covering all I wanted to write about the importance of the DBA/developer relationship, this chapter does not discuss the (by no means less important) relationships a DBA has with other parties such as system administrators, storage administrators, network administrators, and so forth.

Be Approachable

This cannot be overstated. Developers, like DBAs, are often under huge amounts of stress and time constraints. Many developers are skilled, professional, hardworking, and capable of accomplishing great things. When they approach you for help, take them seriously, be respectful, and more