Don’t Let Your DBA Be a Bottleneck

Picture this scenario: An Oracle employee meets with a client who says their database is slow, and upon investigation the employee learns the client has not taken advantage of Oracle’s optimized features – billions of dollars worth – in its flagship product.

I can say from many instances of personal experience that it happens all the time, even to this day. It continues to happen despite many of these features having been available for over a decade, now.

In addition to maintaining databases, DBAs often get involved in the development process, helping with data modeling, architecture planning, storage planning and optimization, and performance optimization, among other tasks. Many Agile principles get applied fairly seamlessly along most of the development lifecycle, and these DBA functions do not have to be an exception.

This document serves to help you look at the DBA role with some fresh eyes. Let the DBA role meld into the rest of the development process. Let the DBA’s work receive and stand up to the same peer review as the rest of the development process, and make sure your DBAs take advantage of best practices and database features – the same way the rest of your development team is expected to continually search out development improvements and refactor their code.

The wide acceptance and employment of Agile development methods has added many improvements to software delivery. Yet it has been clear through all of my experience at clients who employ Agile techniques that there is still a glaring lack of clarity as to how to integrate the DBA function into that process. Work diligently to change that on your project, and you will reap benefits just the same as the rest of the development process has benefitted from Agile. Do not ship off data modeling tasks to a DBA completely separate from the rest of the team in a de facto outsourcing manner. Have the data modeling occur in line with the rest of the development team and process. Many developers will possess a solid notion of what their persistence model and data model should look like. Let the data modeling process flow naturally, in line with the project, with the DBA in attendance.

In addition to data modeling, make sure the DBA is supporting the development process in supplying database resources as they are needed. Do not allow data model changes to slow down development. Make sure those changes are in the same cycles as the code changes, coordinated the same way the code changes occur. Make sure the data model code changes (DDL) are committed to source control the same as the rest of the code. Make sure the data model changes are included in your project’s continuous integration the same as everything else. Make sure the data model goes through the same refactoring that all other code goes through for enhancement, optimization, and continuous improvement, important tenets of Agile.

Finally on this topic, it is not uncommon these days to have DBAs so busy with production issues that their time to help on development tasks gets squeezed until it is nearly non-existent. If this is the case on your projects, make sure you are not letting this situation itself become a bottleneck. If a DBA is not available to contribute to the data modeling tasks of the project, make sure you have lined up the appropriate data architect resources who will be handling the tasks. If you know you will absolutely need some of the time of an already swamped DBA, then you will have to account for some creative ways to insure the DBA’s availability does not become a bottleneck.

Another significant tenet of Agile is the ability to adapt quickly as things change, as business users point out they want something developed differently, etc. Helping that process is the availability of multiple environments, such as personal sandboxes, development, test, integration, pre-production, and production environments. Migrating across those environments is tedious for all developers. Here, again, is an opportunity to ensure DBAs do not become a bottleneck. DBAs need to make sure they accept the challenges of Agile development the same as other developers. Supporting these multiple environments and migration processes is a requirement for the entire team, and when it works smoothly, including DBA's tasks, the entire team benefits.

I do not plan to go through the rest of the functions DBAs typically serve that I listed earlier, but they apply the same to integrating with the rest of the project. Integrate all the functions into the Agile process on your project and expect the same team interaction on those tasks that you do with every other aspect of the project.

Now that we have gone over integrating DBA functions into your Agile processes, let us discuss the expectations of developers and how they should apply the same to DBAs. Your developers are expected always to stay current on best practices, libraries, design patterns, optimizations, frameworks, you name it. If someone has found a better way to do something, developers are expected to apply those new techniques to the project, application, or infrastructure. Take JQuery as an excellent example. If we call JQuery a feature, developers start using this feature once it proves to be valuable to the project. The same expectations should be held for DBAs. They should stay current and apply improvements the same as other developers. I have experienced many instances of DBAs not applying the current best practices, new features and optimizations, and in general sticking with old, comfortable methods. My experience has been that many of these tendencies have had adverse effects on project productivity, quality, and deadlines.

The last three projects I have worked on have been for companies or divisions of companies whose revenue exceeds $60 billion annually. They all had well-funded IT organizations, all were huge Oracle database users, and, in every case, all had very experienced Oracle DBAs. The first client was using Oracle 10g, but migrating to 11g, and the other two used 11g. Not a single place used Enterprise Manager. None of the DBAs at these clients were using Oracle's Automatic Storage Management. At two of the clients I met significant resistance to using materialized views. At all three clients I met complete resistance to changing and optimizing disk configurations. At all three clients I met complete resistance to using Oracle's compression optimizations. At all three clients, DBAs mandated that we used identical database configurations for data warehouse and data mart databases as were used for OLTP databases, including block sizes. There are many other instances where the DBAs would not allow the features or optimizations that Oracle made available. It was a reluctance to support different paradigms across databases, which again was detrimental to performance. OLTPs and data warehouses are different paradigms – they deserve different support.

When Oracle was faced with stiff performance competition from an upstart data warehousing database called Redbrick, things looked bad for Oracle. How did Oracle respond? It implemented many of the same features that made Redbrick's performance so good – improving database logging, introducing its own bitmap indexing, introducing its own internal compression. In addition to those enhancements, Oracle continued to introduce many other data warehousing performance features, such as materialized views, query rewrite allowing in-database aggregate navigation, analytical functions, enhancements to its partitioning schemes, integrated OLAP services, increased storage block sizes, and the list goes on.

When Oracle was faced with stiff ease-of-use competition from an upstart database called Microsoft

SQL Server, things looked bad for Oracle. Oracle would never be as easy to manage as SQL Server some thought. How did Oracle respond? They automated database management tasks that were often seen as too difficult or made them just as user-friendly as the competition. Those optimizations include easier database management tools, automated processes for management tasks, automated storage utilities, better alerts, greatly enhanced database monitoring tools, and many other user-friendly enhancements. Make sure your DBAs take advantage of these options, both to make their lives easier, and also to improve project productivity.

Oracle had to respond to competition in the above cases, but Oracle also led the way in many other cases. Oracle's XML support has been as good as any other RDBMS's if not the best. Oracle's integration with the web has been among the most innovative. As data warehouses are integrated more and more with applications and web sites, these features become more and more valuable to BI solutions. DBAs must stay on the leading edge of understanding and implementing these features as they prove to be best-in-class solutions.

To wrap this up, I would like to point out that I have emphasized many of the above points specific to Oracle environments, but many of the points apply regardless of the RDBMS installed. Apply these principles on your project regardless of the technology and make sure the DBA is just as valuable, productive, and engaged as every other developer.

Author: Chris Ford, CBIG Consulting

For more information, call CBIG at (800) 334-2078 or visit our Web site at www.cbigconsulting.com.