

Message from the President's Desk – Michael Olin

Spring 2010

Every so often, I receive some mail addressed to a person I've never heard of as an officer or manager of my company. Since LinkedIn is always asking me if I want to add "colleagues" who work for an Indian outsourcing company with the same name to my network, I occasionally search online to see if my company's name and the addressee produce any hits. Last week, one of these searches found a letter to the editor of the trade magazine *InfoWorld* that I had written in 1998. *InfoWorld's* database columnist had answered a reader's question regarding slow performance of a database system (SQL Server, if I remember correctly) with a myriad of suggestions that started with tuning at the operating system level, or even replacing the hardware. My reply

(<http://books.google.com/books?id=E1EEAAAAMBAJ&lpg=PA83&ots=QA00ZGdiIU&pg=PA83#v=onepage&q=&f=false>) suggested that the better first step was to tune the slow application and mentioned a great new feature in Oracle 7 that would allow database developers to do so - optimizer hints. Coincidentally, the following Saturday I received a call from my client's offshore support team informing me that a scheduled extract which normally ran in 40 minutes had been running for over four-and-a-half hours. I had just tuned the query upon which the extract was based and, in all of our test environments, the query was completing in less than five minutes. I had expected the first production execution to be an order of magnitude faster than the prior run, not slower. I was a little surprised to see that the query plan chosen by the optimizer (taking direction, I expected, from my carefully constructed hints) in the production database was so different from what was generated in an "identical" test environment. A bit of creative query subfactoring convinced the optimizer to see things my way and our production database then displayed the expected order of magnitude performance improvement. It was time for a column about database tuning.

For years, performance tuning has been one of the hottest topics at our NYOUG meetings. We have had presentations about every possible aspect of performance tuning and hosted many performance related Training Days. Presentations at NYOUG meetings have discussed how to determine which aspects of your database's performance really need tuning and the best ways to perform the tuning. We have discussed how to monitor and tune your database by hand, and how to use automated tools. We've talked about the "old" approaches to tuning, and the new. However, one issue that hasn't really gotten much attention is who should really be responsible for database tuning. As a developer and database architect, I have always assumed that it was my responsibility to ensure that the databases I designed and the code that I wrote would perform efficiently. I was quite surprised when, during an "Ask the Experts" session at one of NYOUG's General Meetings, the overwhelming majority of the attendees indicated that in their organizations, tuning was the DBA's responsibility.

I really should have expected this response. When we plan the agenda for NYOUG meetings, the tuning presentations are almost always put in the DBA track. Our most popular Training Days have focused on tuning and almost everyone who attends is a DBA. Why is it that I find the notion of tuning being the DBA's job to be so strange? Perhaps it is because I have been working with Oracle databases for too long. In the "good old days," the only option we had available for "tuning" was to create indexes, craft our WHERE clauses carefully to be sure to use those

indexes and, if you really needed to squeeze some extra performance out of the engine, choose the order in which the tables were listed in the FROM clause to direct the join order of the executing query. There really was not much opportunity for the DBA to get involved in the tuning process. As the Oracle RDBMS became more complex, DBAs started to think about the physical layout of their databases by segregating indexes from data and locating data files on multiple spindles. Subsequent versions of the database saw the configuration options, and opportunities for DBAs to tweak things expand exponentially. Partition the tables, stripe the drives, and distribute the workload with RAC. We've now progressed to the point where the options for configuring an Oracle database are so vast that we are once again being asked to just set a few parameters and let the system take care of itself. Personally, I don't know too many DBAs who are willing to do this just yet. They do, however, use the vastly improved monitoring tools, leverage the knowledge gained by collective decades of experience with the Oracle RDBMS, and examination of database internals (shared via groups like NYOUG) to watch their databases more closely than ever before. Incremental fixes are made constantly. For many DBAs, keeping the database running smoothly is a never ending process of little tweaks. I've recently experienced this myself. An email from one of our DBAs suggested that a query in an overnight job I created be replaced by a slightly different version (conveniently included as an attachment). I tested the revised query, adjusted it to account for some boundary cases that would cause trouble for the DBA's hints, and off it went to production. The DBA's changes were probably good enough and simply applying them as is would have greatly improved the job's performance on almost every execution. I'm glad I was consulted, but in most cases the developers are conspicuously absent.

Why has it come to this? Once upon a time, application developers were able to rely on their intimate knowledge of the system's requirements and the nature of the data to develop efficient data structures and code well-performing SQL. How did things change so much? I think that the primary reason is that code has become a commodity. In many organizations, the developers no longer possess any special knowledge about the business or the data to be stored and manipulated. There are business analysts who take care of that (see my column in the September 2009 NYOUG Technical Journal). The developers are blindly coding from specifications and in many cases, the least expensive or quickest developer is the one who gets the work. Problems with inefficient code arise later, post-deployment. Then it becomes the DBA's job to fix it. I'm not sure that I can come up with an appropriate analogy. Looking at it as stock cars (commodity code) as opposed to hot rods (artfully tuned queries) doesn't quite capture the situation. Another reason is that specialized solutions, such as Oracle's Exadata appliance, analyze the running system and make adjustments automatically. Perhaps we are finally reaching the point where throwing the right hardware at the problem is the most cost-effective solution. Regardless, the one thing I am sure of is that with the way things are progressing, it is a good time to be a DBA. I would also suggest that instead of relying on the plethora of automated tools that claim to do the tuning for you, attending our spring Training Day ("Writing Optimal SQL" with Jonathan Lewis) is not only a fantastic bargain, but also a smart career move.

Upcoming Meeting Dates

IOUG - OAUG - QUEST - Collaborate 2010

DATES: April 18-22, 2010

LOCATION: Mandalay Bay Convention Center Las Vegas, Nevada

REGISTRATION: <http://collaborate10.ioug.org/>

BIWA Training Days at Collaborate 2010

Register using BIWA2010 for member discounts and BI-nefits

<http://collaborate10.ioug.org/Education/BIWATrainingDays/tabid/83/Default.aspx>

ODTUG Kaleidoscope 2010

DATES: June 27-July1, 2010

LOCATION: Marriott Wardman Park Hotel - Washington, DC

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

SAVE THE DATE: NYOUG Summer General Meeting

DATE: June 9, 2010

LOCATION: St. John's University – 101 Murray Street

