Confessions of a Quick and Dirty Programmer: YesSQL and the Oracle Database Evangelist Team

By Steven Feuerstein, Oracle Corporation

I often use this column to highlight typical mistakes that I make as I go about implementing the code I need for the PL/SQL Challenge or some other project on which I am working.

In this issue, though, I'd like to take a big step back and talk about a whole 'nother kind of “quick and dirty,” the challenges it presents for end users, for application developers, for everyone who writes applications on Oracle Database, and for Oracle Corporation itself.

The quick and dirty I am referring to can be summed up in one term:

"NoSQL"

I'm sure you've all heard of NoSQL, but if you are, like me, a 100% Oracle Database technology fanatic, you might not have paid a whole lot of attention to it. Until I (re)joined Oracle in March 2014, after a 22-year absence, I just shrugged when I first heard about it and got back to work writing (and writing about) PL/SQL.

There have, after all, always been lots of new technologies, some of them fast-fading fads, some of them reflective of fundamental shifts in society and IT. In the meantime, there has always been more than enough to do in the SQL and PL/SQL world to keep me – and you - busy.

Now, however, that I have gone back to Oracle, I am paying more attention, and have come to realize that it's quite important for all of us *inside* the Oracle Database world (whether or not you work for Oracle Corporation) to understand better what the NoSQL movement means: why it has happened and what impact it could have on all of us.

You can think of NoSQL as a few different things:

- A new kind of database technology, designed to handle a small number of "extreme" use cases that SQL and relational databases were not immediately prepared to handle;
- A reflection of the faddish nature of the software industry;

- A wakeup call to Oracle Corporation and everyone who depends on Oracle Database to meet user requirements.

NoSQL: The Product/Technology

Without a doubt, NoSQL describes a different approach to database that is designed to handle extremely large datasets and large numbers of concurrent users, and support very fast response times for all of that. A nice and very sensible presentation on NoSQL is available in the Oracle Magazine July/August 2014 issue: http://www.oracle.com/technetwork/issue-archive/2014/14-jul/o44interview-2219531.html.

If you read that, by the way, you will then know (if you do not already) that Oracle offers its *own* NoSQL database.

Planet Cassandra (Cassandra being one of the most successful NoSQL databases) offers a definition and set of use cases for "NoSQL (Not Only SQL) databases" here: http://planetcassandra.org/what-is-nosql/

Now, if you think about it, SQL and relational database technology never did handle *all* the requirements of our vast and vastly complex civilization. So from that perspective, it's no big surprise that when massive changes take place in human social experience (social media, mobile apps, turning *everyone* in the world into an end user), existing technology might not be ready to handle every requirement of that experience.

[For my thoughts on what it means when *everyone* becomes an end user, check out http://stevenfeuersteinonplsql.blogspot.com/2014/08/is-it-all-about-end-users-or-end.html]

NoSQL: The Faddish Rejection of a Faddish Industry

Yet that wasn't all that happened when NoSQL hit the scene. There also seemed to be a bit of a gleeful pile-on against SQL, against relational technology, against Oracle.

"Ha, ha, Oracle! Ha, ha, Larry Ellison! You said that Oracle Database and SQL could do anything, handle everything, be faster, and be better than anything else on the planet! You were *so* wrong. Face the facts, Oracle: SQL is *old*. You guys wrote it back in 1979, for Pete's sake. How long do you think we would keep buying your expensive database software, when there's so much free stuff and it's really new, and you can't even handle tweets anyway?"

In other words, lots of people in the industry saw NoSQL not as a specific architecture/product for a specific use case but instead as:

[Confession: This image is taken from my Coding Therapy talk, in which I tell people to "Stop writing so much SQL." But I don't tell you to reject *SQL* in its entirety! For a little comic relief in your day, check out http://www.quest.com/tv/All-Videos/2198245815001/Steven-Feuersteins-Coding-Therapy---Why-Developers-Need-Therapy/Video/]

No to SQL, no to relational technology, no to Oracle Database. It's old, it's tired, it's had its day in the sun, but now it's time to move on, to utilize "modern" database technology that is "web scale" (whatever that means), etc.

Oh, *really*?

Is SQL Old and Tired? Is Database a Commodity?

For many, many years - since 1979, in fact - Oracle Database software and other relational solutions have been at the core of just about every significant human development, whether it be based in private enterprise, government, or the world of NGOs.

SQL, relational technology, Oracle Database: They have been incredibly, outrageously successful.

And SQL in particular is a critical layer within the technology stack that runs the systems that run the world. SQL is a powerful yet relatively accessible interface between algorithmic processing and data.

Rather than write a program to extract, manipulate, and save your data, you describe the set of data that you want (or want to change) and leave it to the underlying database engine to figure out how to get the job done.

It's an incredibly liberating approach, and I have no doubt that SQL and Oracle Database are two of the *defining technologies* that made possible the Information Era and the Internet/Mobile Era. Sure, you could argue that if Oracle hadn't come

along, some other company would have taken its place. But Oracle *did* come along, and from 1979 through 2014 it has continually improved the performance and capabilities of Oracle SQL, providing innovation after innovation.

Let me repeat that, because I think that so many of us have lost perspective on the impact Oracle technology – and we Oracle Database developers* – have had on the world:

SQL and Oracle Database are two of the most important software technologies of the last 40 years. And all of you, all of us, played a key role in *applying* that technology to implement user requirements: literally, to build the applications upon which modern human civilization functions. Us. We did that, and we do it every day.

How cool is that?

OK, great, Steven, thanks for the history lesson. But, you know, the past is the past. Now there's Google and Facebook and Twitter and you name it. Oracle Database and SQL simply don't cut it anymore in the New (Social) World Order.

Oh, *really*?

That's utter nonsense. The features and performance of Oracle Database and SQL are still relied upon by millions of technologists, millions of applications, and billions of end users every day.

And that's not going to change for a very long time.

But stability, consistency, power, the unglamorous under-workings of our systems are by definition unexciting. Un-stimulating. Increasingly lacking in new opportunities for speculative or wild growth. There's just not that much of a story there.

And stories drive how people think about things. If providing an amazingly powerful database platform has been taken for granted, then a new story must be provided. New and better languages! Faster development! Web-scale! NoSQL! No tables! "No, wait a minute. That's too hard." OK, then maybe...NewSQL!

[I kid you not. Look it up. http://en.wikipedia.org/wiki/NewSQL and https://cloud.google.com/products/bigquery/ and https://www.mapr.com/why-hadoop/sql-hadoop and http://hadapt.com/blog/2013/10/02/classifying-the-sql-on-hadoop-solutions/]

I get it. I do.

I get why SQL isn't sexy anymore. I understand why and how a technology such as database would become commoditized in the public eye, become something that is just completely assumed, invisible, taken for granted. I get why so much of the software industry – and our economic system as a whole – is driven by the need to create new things, create new demand, keep consumption levels high and ever higher.

That does not mean, however, that database actually *has* become commoditized. Sure, 2014 is not like 1990, back when I first worked at Oracle. We did battle with Sybase, the first competitor that really gave Oracle a run for its business. And there was Ingres and Informix. Somehow, Oracle muddled through. Well, Oracle did more than muddle.

Now? Our competitors are legion, quite effective, and, many of them, absolutely free. That is quite different from Oracle's model, to say the least.

But for all that you can identify fine alternatives to Oracle, you will have a much more difficult time finding an *equivalent or better* alternative. Oracle Database remains the most powerful and successful database technology of all time.

And back to that idea of database becoming a commodity: Let's take it a step deeper. What's a database for? To hold a company's data. And that's no small thing; data is the lifeblood of a company. Many companies are little more *than* its data, these days.

Which means management must make very careful decisions about what database software to rely on for storage of, protection of, and secure access to corporate data.

And over and over again, however much they may complain about the price, they choose Oracle Database. It almost makes you think that maybe our software is priced right.

[Uh oh, now I may have lost *all* credibility. That is the price one pays, I suppose, for trying to think things through logically. Sometimes you end up in a place that makes other people uncomfortable. But, seriously, if our software was *not* worth it at many levels, things would look very different out at Oracle Headquarters.]

Where does that leave us?

- Database is an invisible layer of technology for most users;
- But database technology is *not* (yet) a commodity.
- You can pick from some very fine, open source databases;
- But Oracle Database remains the worldwide leader in a technology arena that for many companies require the very best solution, regardless of cost.

The New Quick and Dirty

The 21st century macro-version of "quick and dirty" goes right to the heart of the role of database in application development.

If you talk to many "next generation" application developers, the role of database is to "Disappear, Database, Just Go Away." They don't want to deal with designing data structures. They want to build web pages and mobile apps, and they don't want to get slowed down by some backend nonsense.

So they throw all their data into their JSON documents inside their document databases. They change data structures on the fly inside their documents, they snort at the idea of joins, they don't enforce constraints, and they gladly accept – even celebrate - denormalization, all in "service to the application."

I must admit that I feel a bit jealous of these fast-moving developers using quirkily-named languages. Who among us has not desired greatly the total freedom of being able to add and remove columns, change database design willy-nilly, as we are build our applications? Who among us has not chafed at a DBA's insistence on standards, on controls?

Yet also who among us has also not felt the downstream agony of a poorly designed database? Despaired at the extra nasty code we had to write to compensate for denormalized data, for a lack of anticipation/clarification of user requirements, for a total lack of constraints?

One of my favorite mottos is "Act Now, Perfect Later." You'd think I'd be a perfect candidate to fall in love with JavaScript, NoSQL, and document databases. But I also have a couple of decades of experience that warn me against acting *too* precipitously, *too* carelessly.

We know, don't we, that there are no magic, silvery bullets out there. We know that there are prices to be paid for taking shortcuts up front in the application development process.

And we also know that programmers cannot control themselves (ourselves): When we can take a shortcut, we will (want to) do it, almost every time. And when entire movements in the software industry are cheering us on with, "Social data is unstructured! Modern data is unstructured! The world is just a big bunch of documents! It's OK if an occasional transaction is lost! Go for it!" then "go for it" we will – and with a vengeance.

What the New Quick and Dirty Means to Us

The world has changed a whole lot in the last 35 years. We should not be surprised to find that technologies such as SQL and PL/SQL and relational technology are challenged by some of these changes. We should also be prepared to acknowledge areas that are not well-served by Oracle Database technologies.

But we should *not* do ourselves and our customers the disservice of capitulating to the more faddish elements of our industry, which seem to be more than happy enough to "throw out the baby with the bath water."

It is time for Oracle Corporation – and our tens of thousands of committed, sophisticated expert users – to be proactive and assertive and, above everything else, user-focused. Because we certainly *do* face this reality:

- Oracle Corporation has an enormous installed base of companies that have made sizable investments in Oracle Database and need to *maximize* that investment.
- If our current customers are not wildly successful with Oracle Database, it is hard to imagine being able to convince lots of "next generation" developers to utilize our technology.
- If our current customers *are* wildly successful with Oracle Database, they will lead the way in showing everyone "out there" how and why Oracle Database should be used.
- Application developers will continue to make decisions regarding database.

And that's why we are creating a new Oracle Database Evangelist team with a dual mission:

1. Do everything we can to help our users maximize their investment in Oracle Database technology.

2. Show new generations of developers how leveraging fully SQL, PL/SQL, and relational technology will make *them* more successful.

You'll be hearing lots more from and about and from the Oracle Database Evangelist team in the coming year. You'll be seeing more and more webcasts, tutorials, podcasts, and how-tos. We'll be giving users new ways to contribute their experience and wisdom to the global community, and to help new developers understand Oracle Database.

But, just as important, I hope to be hearing from all of *you*, as you share your expertise and experiences with Oracle technologies, and help Oracle Corporation remind the world of how powerful and important are SQL, PL/SQL, and Oracle Database.

YesSQL! (and YesPLSQL!)

* I used to talk about PL/SQL developers and APEX developers and ADF developers and Javascript developer and so on, but I have recently come to realize that very, very few Oracle technologists can be “pigeon-holed” that way. Sure, I know and use only PL/SQL (and SQL), but just about everyone else on the planet relies on a whole smorgasbord of tools to build applications against Oracle Database. So I’m going to start referring to all of us simply as *Oracle Database Developers*.

About the Author

Steven Feuerstein is considered one of the world's leading experts on the Oracle PL/SQL language, having written ten books on PL/SQL, including Oracle PL/SQL Programming and Oracle PL/SQL Best Practices (all published by O'Reilly Media). Steven has been developing software since 1980, spent five years with Oracle (1987-1992), and has served as PL/SQL Evangelist for Quest Software (acquired by Dell in September 2012) since January 2001.

He is an Oracle ACE Director and writes regularly for Oracle Magazine, which named him the PL/SQL Developer of the Year in both 2002 and 2006. He is also the first recipient of ODTUG's Lifetime Achievement Award (2009). In 2012, Steven won the Best Speaker award at ODTUG's Kscope12.

In 2010, Steven started the PL/SQL Challenge, an online, daily PL/SQL quiz. In 2011, he launched PL/SQL Channel, a library of over 27 hours of detailed training on Oracle PL/SQL.

Find out more about Steven at www.StevenFeuerstein.com.