Donuts for DBAs

By Redmond Bim.

The aim of this column is to provide and outlet for Oracle Database Administrators and Developers who are tired of hearing the same marketing blurb and reading the same articles about managing and developing an Oracle database.

The goal is to give a different perspective to the standard topics and offer an insight into the workings of the Oracle database the reader is rarely ever subjected to.

It is hoped that these articles will make the reader question the workings of their environment and hopefully improve their situation.

The material contained in this column and future columns is designed to be provocative and it is important to stress that the views expressed are those of the author only. They do not represent those of the publisher of this magazine, Oracle or any business partners of Oracle.

I am frequently involved in tuning database environments, in fact I have been involved in tuning the Oracle database since Oracle V5 a long time ago. I have seen it evolve, improve and grow stronger, but there has always been a common theme when it comes to tuning the database that has never changed. I will cover this point in a moment.

First though what is the point of tuning? Typically to make the database run faster, get more bang for the buck, make the users happy and ensure the DBA is not working to 3am waiting for indexes to recreate.

The art of tuning is becoming a lost science and it is disappearing because the focus is moving away from the cause and to the cure.

The cure typically involves reading countless books on Oracle Database Management, analysing them and looking for those small gems that will save the day.

It also involves running those wonderful tools supplied by Oracle and other Vendors which offer the panacea for tuning. More on these tools in another column.

The cure can at time involve attending Oracle training courses getting indoctrinated as to when to create an index, when to defragment a table and when to get certified. More on this topic in another column also.

The point is, is that sometimes the above cures work and the performance issue is resolved. Wonderful, everyone is happy. Well not really, because one day later, one week later, one month later the same problem re-occurs and there is another tuning issue.

Database administrators across the world are always fire fighting, and if you haven't heard this term before, it means working very hard to solve last minute but crucial issues. They are overworked, underpaid and never have time to do the important tasks. Sound familiar? Most cry out to their bosses, give me more staff, more training and I will be able to do more much better, be more efficient and get more done.

Well I have some bad news. The world doesn't work that way, gone are the days of unlimited resources on a computer project and be grateful they have gone, because they never worked anyway. More resources thrown at a

problem typically made it worse not better. So next time you feel overworked and stressed because you have an impossible workload take a different tact to resolve it. How? Well I'll cover this later as well. There's nothing like a little bit of suspense to keep you reading.

To resolve the tuning issues focus on the cause of the problem not the cure. Well you might say, what's new here? The Oracle tuning manual pretty much says the same thing. Start at step one, look at the database structure, get the design right, move to step two, the environment, then look at the database structure. Stop there. It might seem like the cause, but from what I see this is still the cure. Your mindset is still set into looking at the database and fixing it.

Enough and lets get to the point. The cause in nearly all sites I have been too, why there is poor tuning is to do with the relationship between the DBA team and the Developers. Now's the time to close the magazine and walk away because the author sounds like a lunatic. If you do, you will never know why and will forever lose out.

For starters, throw out the window the concept that tuning is logical. You follow these magical steps and voila you have a perfectly tuned database. Sometimes this works, but that's more by luck than anything else. To many DBAs and Developers are logical. I guess it can't be helped because technology is logical and by programming you are training the mind to follow a logical path. How much fun is poked at Vulcans because they are logical.

Follow logic and you are doomed to eventual failure. The human brain isn't based on logic, it's based on neural networks that are better suited for pattern matching. The best developers who debug programs resolve them by using techniques found in an Edward De Bono book on lateral thinking. Most don't realise the technique they are using. It's beyond the scope of this article to cover these tuning methods, but suffice it to say they might get included in a later article.

I deliberately digressed to highlight the point. Most DBA's and Developers are caught in the trap about tuning an environment and can't see where the real problem is. Sure the database might be running slow, but why did it get this way in the first place?

That's a very good question and I am glad I asked it, because it raises another point that gets me very frustrated with tuning manuals. They assume you have control over the application. Ten years ago there were hardly many off the shelf products, so most sites had to develop their own. Now it's very different. Thousands of business partners offering the complete range of products designed to meet an organisations needs. In these cases the code is a black box, can't be touched or tuned and this leaves the DBA in a quandary on tuning. How do you tune these products?

The tuning manuals assume you can get in and rewrite and change the underlying code. If you can't do this then the tuning manuals are useless.

So we have two types of situations in today's environment. One where there are Developers building in-house applications and the other where the DBAs having to manage a black box product. In a fair number of environments both situations are true, though each requires a different tactic for resolving it.

I'm going to address both points, starting with the all-familiar relationship between the DBA and the Developers. Lets describe some typical environments.

First, due to some reason found in the organisation's history books, the Developers have determined that the DBAs are only suitable for performing backups and they know what's best for the environment.

Second, because of budget constraints there wasn't time to get the DBAs involved in the original design stage.

Third, the DBAs don't trust the developers because they are all gun-ho and have no concept of change management.

Fourth, due to management's lack of vision, the DBAs are constrained to use an antiquated version of Oracle.

The role of DBAs and Developers in an organisation is rarely understood. The developers feel it is their role to build the application and they see the DBAs as impediments in this goal. Sure they pay them the occasional lip service and acknowledge their presence but most see DBAs as backyard players only to be bought in when things go wrong.

The DBAs see the Developers as small minded, coders who have no vision or global understanding of the organisations environment. They have no concept of the pressures they have to endure dealing with production databases and hate having to come in and clean up the mess left behind by reticent developers. Some long for the good old days when waterfall development was the only respected way of building an application and there was an agreed and formalised arrangement between the two groups.

Given the advent of rapid application development. Actually that's another point. No one really develops using RAD, they think they do, but it's usually just a term coined for not doing development using the Waterfall method.

So with Rapid Application Development, the strict formalised relationship between the DBA and Developer has collapsed. It's withered and died, but it has given rise to numerous opportunities for enterprising consultants to cash in and come into organisations and do specialised tuning. So maybe it's not a bad thing.

At this stage I do feel sorry for the DBAs at really small sites where they are also the Developer, they must be going through an awful identity crisis reading this article. Actually you are the lucky ones, but the biggest problem you face is managing yourself.

So lets stopping being focused on the negative and see what can be done to fix the problem. And that's another issue, it might not be a problem. The relationship between DBAs and Developers is going to be fixed by applying some logic and rules to it. It's a fanciful thought but would never work.

Its not a marriage counselling issue either. Its generally not safe to put a DBA and Developer in the same room together in the best of times. The two don't need to be married to each other to ensure the database environment is perfectly tuned. In fact its best that the two groups don't get on.

By now you are probably very confused as I seem to be contradicting myself. If you see this as happening then you have fallen for the standard logic trap, a contradiction is when two logical events don't add up as the previous scenario seems to point to. As mentioned, we are dealing with the human psyche here and not logic so try and view this from a different point of view to try and understand the point being made.

The goal of the developer is to build a product that satisfies the needs of their customer within budgetary constraints.

The goal of the DBA is to ensure high availability of the database, security of the data, and all applications accessing the database share resource equitably.

Both the Developers and the DBAs have different goals and drives to adhere to. At times the DBA is required to slow down the developer to protect them from themselves. It is also important they do not report to the same management as the Developers. When this happens there is trouble. Because the DBA is locked into the same goals as the developer and then they lose their independence.

At times, the DBA has to say no to a developers request, even if this impacts the developers goals and business objective.

Too often though, the DBA usurps too much power and make the developers goal too difficult to achieve. They boycott developer activities for hitherto, unrealistic goals. It's at this time the clash occurs and the power is eventually rescinded from the control of the DBA.

There could be seen to be a balancing act between the two groups, a constant struggle for control and balance. If either side gains too much power and control the end result will be a poorly performing database.

Lets go through some real life based scenarios. In organisation A, the lead developer is closet managerial material and is focused fully on the goals of building and delivering an application (there is a bonus at stake). The new DBA, fresh out of training arrives on the scene. To better visualise this, try and picture the squeaky voiced kid from the Simpsons who works the till in Krusty Burgers. The two aren't well matched and when it comes to any standoff the developer invariably wins.

The developer believes he knows everything about being a DBA, he's off course read the concepts manual and can spout the odd piece of technical jargon. This belief system stems from the marketing propaganda bestowed from high that says the database is now so easy to use any one can set one up. The developer then gives the DBA the token task of setting up backups. A job he believes fits in the same category as muck racking. Of course, his ulterior motive is that if the DBA can't even manage this, he will take the role and move it into his team. If this is sounding all too familiar then it's not too late to panic, but let me continue.

All goes initially well for our all conquering developer. The project is running on schedule and it's now time to go live with the code. It goes live and that's when the problems start to begin. Two months into the project and the whole application is running slow. It's impacting other ones running and no one is happy. Of course the developer has a fall back plan and conveniently blames the DBA, I mean

it's the DBAs job for fixing performance problems isn't it?

I hope by now you are seeing where the real cause of the performance problem is. If you aren't and you believe its now the job of the DBA to fix the developer's problem, I suggest you stop reading this and go back to the comfortable life you are living in. Performance is obviously not an issue for you and blissful ignorance is the contentment you deserve.

Anyhow, back to the real world. The DBA has no idea what to do, and is wondering if they should be attending more training courses. The developer is getting edgy because the users aren't happy. And then they bring in the outside consultant to resolve it. The political quagmire the consultant gets embroiled in, only further justifies their desire to increase their hourly rate. The discussion between the consultant (C) and the developer (D) goes like this. And this conversation is the same wherever you go:

C: "Your design has some fundamental design flaws in it"
D: "It was built using tool (unnamed) and conforms to 3rd Normal Form"
C: "Off course, but being normalised is no guarantee it will perform well."
D: "We had tight budgetary constraints. We didn't have time to seriously performance tune it."
(This is a red flag to the performance consultant. If there were tight budget problems how can they afford to have them in now reviewing performance).
D: "This problem doesn't happen with SQL*Server"
(Gosh, another red flag waved here. Blame the database vendor. The grass is greener on the other side. If they raise this, it means they really have no clue about databases. Time to give them a shovel and walk away – but a good consultant likes a challenge and will stay on, no matter how fast the ship is sinking).
D: "It performed well in acceptance testing"
(Yes and like the point above, so do small database vendor products. It's this issue of scaling that leaves most developers in dry dock)
D: "We added some indexes, and that fixed some of the problems, but other ones then appeared"
(The consultant is biting their tongue at this point. The design is flawed, haphazardly adding indexes is not going to seriously get anywhere).
C: "You need to rebuild these tables, rewrite these and fix here and here"
D: "Off course we will"
C: "And you need to be on a stable version of Oracle."
D: "We didn't have time to upgrade"

The consultant by this stage is wondering where the DBA is. If they had got involved early on in the project these issues would never have materialised.

C: "You should be on the latest release, it fixes a number of problems"
Dba: "I tried but they wouldn't let me, said it wasn't important to the project"
C: "But its your job to stipulate what the platform is"

The conversation could go on and on, but I hope the point has been made. The problem of tuning was due to the lack of a formal relationship between the Developer and the DBA.

Now for the fun bit. Who is to blame or at fault in this situation? It's always enjoyable to point the finger at someone. I mean, that's what this article is all about, trying to ridicule the poor developer or DBA.

No, I'm not going to get caught in that trap, no matter how enjoyable it seems to be.

Strength of character and personality should not be the driving force for who wins arguments between developers and DBAs. Many large organisations can attest to this fact and most have learnt it the hard way. Their solution is to smother the problem in bureaucracy and red tape, which actually makes it worse. How many change control manuals have I seen gathering dust on shelves, written by well meaning individuals with the goal of preventing disasters described above. By writing voluminous amount of documentation just drives each party into a corner and ensures that legal representation is needed by each side.

What is needed, and this is a foreign concept in most organisations, is a good manager who actually understands both sides technically and can balance the needs. Sure some basic procedures are needed, but no more than a page.

The manager needs the power over both sides and needs to have sound business sense as well as the technical knowledge to understand the issues. And that's a challenge. You would think with the huge amounts of money managers get paid they would have both these skill sets. Typically they only have one, and if they only have one of these skill sets and try to bluff their way through the other one, then they are more dangerous than the DBAs and the Developers combined. It's been said many a time that a manager with a little bit of technical knowledge is more dangerous than one without any technical knowledge.

The corollary from this is quite roguish in its implications. Ultimately the manager over the developers and the DBAs is responsible for the performance aspects of the application. Put in a good manager, who can establish good practices for both sides and ensure that they keep each other well balanced, and the result should be an environment conducive to suitable performance for all applications. It won't be a perfect environment, and there will be issues, but at least the fundamental structure will not be flawed.

Don't go away, there is still more. I haven't covered the other scenario where the DBA is the one that has the control. Here the outcome is slightly different.

In this scenario, our DBA through various nefarious activities has put themselves in the central seat of power and control over the database environment. What they say goes and they have the control over anything that goes into and out of the database.

The justification for this control is based on protecting the production environment and ensuring its stability and reliability.

A hefty volume of change management procedures ensures production changes are few and far between. This ensures the production environment remains stable and availability remains high. This position ensures the DBA maintains control. Attempts to usurp this power are always rebuffed by the high availability figures constantly touted by the controlling DBA.

Though the users are happy with the availability, the business is ultimately suffering as the whole environment has become inflexible and cannot adjust quickly to market demands.

New projects take months to initiate and move to production, rather than the weeks needed to remain competitive. The result is that by the time the project is moved to production, so many restrictions are placed on it, to ensure it performs well and does not impact anything else, that the project fails in maintaining its original goal of being of benefit to the users.

When the DBA hears of user complaints they invariably blame the inexperience of the development team and lament for the days when developers knew how to design and build.

Astute developers, unable to break the reigns of power instigated by the DBA, will use a number of ploys to get the project built. These include outsourcing the development, which is in effect hiding the application from the DBA. Only when the project is completed is it given to the DBA for implementation in production. It achieves the short term goal of developing the project, but we don't resolve the performance issue.

Some not so astute developers, fresh out of training college will not see where the problem is, and expound the virtues of "Mickey Mouse" databases as being the solution to quick development. They will label Oracle as a dinosaur and impossible to work with and if they had SQL*Server then development would be so much easier. Taken to the worst extreme, inexperienced managers will actually believe the whimpering cries of these developers and embark on a campaign to replace the Oracle database with the "easier to use one". This actually makes the whole situation worse and is an incredibly dangerous position to be in. The medium to long term costs are actually greater. Rather than fix the core problem, a bandaid cure is given.

Bureaucracies evolve and grow over time. The red tape increases when management determine it's not appropriate to blame the individual for making a mistake but rather a process was not in place to prevent the problem from occurring. In addition management want to protect the organisation from employees who are moving between jobs, on leave or are leaving and taking corporate knowledge with them. Procedures are written down, effectively idiot-proofing the environment. Taken to the extreme and we start to see a situation all to commonly found in the USA where manufacturers have to write obvious "how not to use" instructions on labels. Like "don't drive whilst drinking the hot coffee otherwise it will spill in your lap", "don't wear slippers whilst climbing a ladder", or "don't use the hair dryer whilst having a bath".

There must be a healthy balance between common sense and the need for procedures. Its actually interesting to note that its typically management that want these procedures, but when it comes to management reports, they always want an executive summary to them to save them reading the gory details. It can be seen that life would be much easier for everyone if all reports only contained executive summaries and no details at all. So the same can be said for procedures, keep them simple, concise and brief. Gone are the days of having to justify ones position by the weight of a document written. Rather management should recognise and reward workers who write procedures that are simple, concise and fit on one page of paper.

So getting back to our original issue with the DBA. They have tight control over the machinations of the process of change in an environment and through red tape slow down the process resulting in the business competitiveness being severely hampered.

It gets back to management again and short sightedness of them. The solution is to change the basic belief systems of any organisation. And that is, in today's environment mistakes are made, mistakes aren't necessarily a bad thing, and procedures do not always have to be put in place to prevent mistakes from occurring.

Some self-help guru's tout the line that "a mistake is an opportunity for improvement and advancement". In a way this is true. Programs are never written bug free, they contain errors. There are numerous reasons for these errors (to be discussed in another column).

Good managers in an organisation should be like the grease on a wheel axle. They should ensure the organisation runs smoothly. They should control the process and ensure the teams that do the actual real work do it efficiently.

Management need to be constantly reminded that they are at least one degree of separation from the company's customers and are not the actual producers and moneymakers. To say it simply, their role is an optional extra. Putting together fancy project plans or managerial studies only justifies their position; it doesn't make the company more profitable. In fact when it comes to the cause of most of the large companies failing in the marketplace, the common factor stated was poor management.

In the new millennium, the role of managers is going to be analysed closely, especially as more workers due to technological advances become more empowered. The need for managers will change and their importance critiqued. To start with, the whole hierarchical concept instituted and maintained by management (purely because they are at top of the hierarchical positions) needs to be reviewed, rethought and thrown out. And no, this isn't a call to start a revolution.

As an analogy, lets look at this from a technical viewpoint. Unix directory structures look hierarchical but they aren't. They are more like a network. They work well and are very flexible. Microsoft Directory structures are hierarchical and suffer from no end of problems. On the plus side it must be noted that later releases of the operating system are improving on this position.

In addition, the view that all workers can one day rise to the position of management also needs to be quashed. It's a completely inefficient viewpoint that will not survive the rigours of the new competitive decade. For most people moving into a management position this is typically a backward step, and I cite the "Peter Principal" here as proof in point.

When mistakes are made, this doesn't mean an opportunity to write procedures, but rather quickly review and evaluate why. People make mistakes. Leave it at that and don't move into procedure writing mode. On the other hand though, smart managers in an organisation are quick to realise that there are problems in their environment, which are out of their control and which they might have

caused. So they move to a new project to escape them. In some cases, this brilliant manoeuvre gets them promoted, thus further showing how being inefficient moves you up the chain of command.

So once more in attempting to address the core reasons why databases perform poorly in an environment I have successfully turned the discussion around and come back to poor management as one of the core reasons for the badly performing databases.

The irony gets better as most managers aren't even technical and couldn't write a SQL statement let alone spell SQL, yet they are in the enviable position of controlling the IT organisation. In their ivory towers they justify this lack of technical prowess by pointing out its their job to manage people and not resolve the problems. In fact many of the executive retreats these managers attend reinforce this mistaken belief, typically because the people who run these retreats also have no technical knowledge. What they all tend to forget is that when you look at the world's richest and most successful people (case in point Bill Gates), they should realise that they are both technically competent and good managers. For management, they have to be both, not one or the other.

So DBAs and Developers reading this article are now critically looking at their pointy haired manager, wondering which category they fit into and feeling comfortably smug that even though they are given impossible tasks to do and have to fight red tape and bureaucracy, somehow its not their fault, but its due to the incompetence of their management. If at this point you feel this way then this article has been wasted on you, and my apologies for wasting your time. I suggest you go back to reading those Dilbert cartoons that justify your position in the grand cubicle of things.

When it comes to resolving these issues and resolving the spate of problems you find yourself in, don't blame, rather change. Database performance issues have their inherent problems rooted in the structural weaknesses of an organisation. Over the next decade the businesses that grow and prosper are the ones that organise their internal structure to use the human abilities of their employees. If you are an employee change the organisation instead of moping and whining about the plight you are in.

It's time to throw out the old nineties way of doing business and use a different tact. Only then will the performance issues of the database be truly addressed.

If you haven't worked it out, Redmond Bim is a pseudonym for a rather frail, old and haggard DBA languishing in the back waters of a large corporate giant, pining away the hours until retirement. Or maybe not. Redmond could be a young, Volvo driving lunatic, hell bent on destruction and driving to excess. Either way, its not important who Redmond is, but if you want to contact Redmond and either congratulate him on this article or want to know where he lives so you can frail him alive for writing such nonsense, then you are more than welcome to send an email to the publishers of this magazine. Under careful supervision these emails will be forwarded to Redmond's private email address. Suitable versed articles might appear in the next column.