

1

The Challenge of Performance Tuning

Database Administrators, or DBAs, perform a wide variety of tasks – some exciting, and others more routine. At most companies, the DBA doesn't have the option to pick and choose which tasks to perform. Typically, all the DBAs have to "pitch-in" and work on a wide variety of tasks, some pleasant, and others dull.

On the down side, not too many DBAs get really excited about setting up backups, monitoring disk space, or checking to see if the exports ran OK last night. For most DBAs, these tasks are not especially challenging, nor do they typically call for much creativity. Nevertheless, they are all critical to ensuring the integrity of important systems on which a firm depends.

There is another facet of DBA work, however, which many have found to be a definite plus. On the up side, performance tuning offers DBAs an entirely new arena in which to flex their muscles. Far from being routine or dull, performance tuning allows the analyst to exercise creativity and intelligence, and to solve baffling problems that are threatening a firm's success.

In this chapter, we take a step back and look at the big picture of performance tuning. We explore the proposition that performance tuning is not all science but, rather, a mixture of art and science. This means that the DBA must be *more* than a superb mathematician who cranks out answers to complex problems. Instead, the performance expert must have a flexible process that allows him to change tack as needed.

We will present an adaptable approach to performance tuning that I call **Physician-to-Magician**; with this method, the DBA "changes hats" in the various stages of the performance tuning process. Each stage has its own unique tasks, suited to what needs to be accomplished at that point.

Real-life scenarios will illustrate the challenge and excitement of Oracle performance tuning. In this chapter (and throughout the book), we will be using actual **case studies** whenever possible. This is useful for several reasons:

- ❑ Real-life examples are often more obscure than we could ever imagine.
- ❑ Historical examples can be used to illustrate a sound approach (or a blunder to avoid).
- ❑ It is easier to remember colorful cases compared to merely theoretical axioms; they are often much more interesting.
- ❑ Solutions to real cases may provide the reader with an answer to similar problems.

In addition to the various case studies in this chapter, we introduce our first performance tuning **hint**. These hints are observations and suggestions that I believe are critical points in understanding the performance tuning process. Some of the hints might seem obvious at first, but many of these common sense tips are not so commonly practiced.

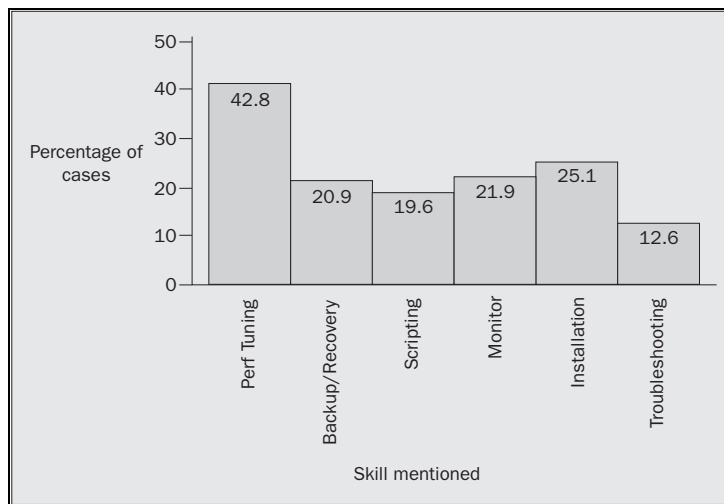
Along with the Physician-to-Magician approach, we will also discuss some obstacles to performance tuning that the specialist may encounter. These include the "silver bullet" tendency, as well as the "guessing method" approach. These troublesome obstacles are covered in more depth in Chapter 2.

Finally, we conclude with a discussion of "preventative medicine." Ideally, DBAs, designers, and developers will strive to prevent many of the performance problems that are seen so frequently today. Let's begin...

Why Performance Tuning?

In contrast to more routine DBA tasks, performance tuning can be a fascinating part of database work. Most people do not realize, however, how rewarding it can be. This is unfortunate, because performance tuning is one of the DBA's greatest challenges, with both tangible and less obvious rewards. How many jobs provide the opportunity to be an instant "hero" who saves the day by fixing critical systems for a large corporation? Yet this is exactly what a skilled performance expert can accomplish.

Performance tuning skills are also in high demand. The following figure illustrates some skills that employers desire in newly hired Oracle DBAs. Notice that performance tuning is mentioned far more frequently than even basic skills, such as backup and recovery. Of all employers looking for Oracle DBAs, 42% mentioned performance tuning as a desired skill:



Source: DICE web site (<http://www.dice.com/>) on 3/7/2002. Sample size: 593 employers. DICE (Data Processing Independent Contractor Exchange) is a popular web site for software professionals.

These figures can be interpreted in several ways. Possibly, these skill sets are needed to address newly discovered performance problems that haven't yet been dealt with by a tuning specialist. It is also possible that they have been addressed – but not successfully! Whatever the explanation, it is clear that there is a big demand for Oracle DBAs who are also competent tuning professionals.

Note that the slightly lower demand for skill sets such as backup and recovery does NOT mean that these skills are less important to the company. Rather, it is a reflection of the *scarcity* of competent tuning professionals.

Why This Book?

Although many DBAs and designers are confronted with a challenge in Oracle performance tuning, many are not really sure where to start. They know that *something* must be done, but what? "Should I start changing the `init.ora` parameters? Or perhaps I should add more data files?"

Without a sound approach to performance tuning, one guess is as good as another, and will rarely lead anywhere. This guessing approach may be fun at first, but our bosses will seldom find the results satisfactory. We all want to employ a competent, logical approach to Oracle performance tuning – and get the corresponding results. This is a reasonable goal, whether the performance analyst is a permanent employee or a consultant.

This book is designed for those readers who want to learn and apply a systematic, step-by-step approach to resolving tough performance issues. The message of this book is a positive one: **There is no need to employ a guesswork, seat-of-the-pants approach to Oracle performance tuning.**

Of course, when we talk about a sound approach, that does not mean that technical content and expertise is unimportant. On the contrary, the best approach in the world is useless without a sound grounding in Oracle database operation. In the earlier portions of this book we emphasize a sound process; in later chapters, however, we introduce analytical methods. These methods include how to use the `V$WAIT` interface, how to solve table joins, how to use Consumer Resource Groups, and so on.

A Quick Note on the Term DBA

Throughout this book, we will use the term, **DBA**, or **Database Administrator**, when referring to a specialist who is tackling Oracle performance problems. In actuality, this person may not be a DBA at all – the performance analyst could be a lead developer, a senior designer, or perhaps a programmer. Depending on the individual organization and project, the company's "Oracle Magician" could be any of these individuals.

What's important is not whether a person is called a DBA, but how well they understand database fundamentals, and how well they apply a consistent, logical process to solving performance problems. After all, results are what the performance tuning game is all about. When a tough performance problem is resolved, management will be very appreciative – regardless of the person's title.

Who Should Read This Book?

This book is aimed at two groups of readers. The first group consists of those DBAs who have limited or moderate experience of Oracle performance tuning. Perhaps you have been a DBA for only a short time, or you have struggled with even how to *begin* solving performance issues. If this describes your situation, this book is for you. We will show you how to get started with Oracle performance tuning, and how to maximize your odds of success.

Besides new DBAs, this book will also help those readers who have already had some exposure to performance troubleshooting. Some of you have already analyzed a few simple performance issues. You might have successfully solved some problems by adding an index or two, or perhaps by rewriting some poorly defined SQL. Nevertheless, although you have achieved some success, you sense that you need to beef up your skills in Oracle tuning. You would like to handle much tougher performance issues that really challenge the analyst. If this describes your background and desires, this book is also for you.

If you are an experienced DBA, you will benefit from the discussion of some recent innovations in Oracle performance tuning. For instance, we will explore sophisticated techniques, such as using the **wait events** facility to isolate bottlenecks, and how to use the **star transformation** join to optimize data warehouse queries. Additionally, you will probably find the discussion of the Consumer Resource Manager and the section on a graphical method of analyzing table joins to be very helpful.

What You Need to Know

To gain the greatest benefit from reading this book, some basic knowledge of database operation is assumed. For instance, we assume you know what `init.ora` parameters are, or what we mean by the Listener. You should already have experience in starting and stopping an Oracle database.

You should also understand some of the very basic concepts in relational databases, such as the various types of SQL statements. Although you need not be a SQL guru, you should know what a join is.

Beyond these basics, you do not need to have any special background in performance tuning.

Good News for the Reader

Since you are reading this book, you are most likely interested in becoming an accomplished performance tuning analyst. You might already be employed in this capacity as a DBA tasked with improving performance, or you might be curious as to exactly how performance experts operate.

As with all professions, there will be new things to learn and obstacles to overcome. This book is written as a guide, or mentor, to help you get around those obstacles, by answering questions such as these:

- ❑ How do I get started?
- ❑ How can I master such a complex subject?
- ❑ How can I achieve remarkable performance tuning results?

The good news is that:

- ❑ Yes, you too can do these things!
- ❑ I'll show you a good way to start.
- ❑ It isn't really that hard!

The Physician-to-Magician Approach

The true way to become a performance tuning expert is not through technical sophistication, but rather by a good understanding of the overall approach. Many extremely bright DBAs do not do well at performance tuning because of a poor approach. Despite their intelligence and drive, they unwittingly hinder their own progress.

Performance problems take place in a certain *context*, typically a corporation with many other software professionals involved. This means that there are many aspects of human interaction that need to be considered. As we will see, this context means that DBAs sometimes have to play detective, separating the real information given to them by their colleagues from the illusory. At other times, they must try to figure out which problem should be solved.

Even with the root cause identified, the DBA must somehow fashion a solution that is acceptable to the other software engineers. The Oracle performance expert must deal with a variety of complications, many of which are not purely technical in nature. This wide scope of activity means that they must be **adaptable**.

The Need for Adaptability

To be effective, the performance expert must have a flexible approach that can address the different requirements within each performance assignment. In other words, the DBA must wear different "hats" as the performance tuning process unfolds. The effective performance analyst realizes that the methods used must *adapt* to the particular problem and to the people involved.

This multi-dimensional approach recognizes that much fact-gathering, analysis, and deduction must be applied – each according to its order in the performance assignment:

The performance expert must use a process that adapts to the various stages of the tuning assignment.

Let's take a look at the different stages that we can identify in our successful performance tuning process:

The Steps From Physician to Magician

We begin with a summary:

- 1. Define problem:** What is the chief complaint?
- 2. Investigate:** Confirm/quantify the problem.
- 3. Isolate root cause:** Home in on the essence of the problem.
- 4. Devise a solution:** Fashion a solution to the environment.
- 5. Implement a solution:** Make sure the solution fixes the problem.

We can liken each of these steps to the jobs carried out by other professionals. Let's take a look at each of these analogies in turn:

Step 1 – Physician

The performance analyst starts off as the physician, listening to his patient. When any of us visit a doctor, the doctor will soon ask, "What seems to be the trouble?" In medical clinics, this is called finding the **chief complaint**. The DBA must do the same thing; for instance, he may ask the questions, "How long is the query delay?" or "When does the problem occur?"

Step 2 – Detective

After defining the problem, the next step is investigation. Here, the DBA changes hats from physician to detective. The main objective in this step is to **recreate and quantify** the problem. For instance, the DBA-detective will strive to find important facts, such as the actual elapsed time of the query, or the number of disk and logical reads that are performed. In many cases, he will discover that the problem has nothing to do with the database at all. Sometimes, he or she will conclude that there isn't even a problem!

Step 3 – Pathologist

After the detective has gathered clues (and confirmed that a problem really exists), the Oracle Pathologist tries to find the **root cause** of the problem and isolate what it is that is causing our database so much distress. At this point, we have a great deal of good information, so there will be no need to make wild guesses. The Oracle Physician and Oracle Detective have given us their files, which contain a great amount of useful information. This information will help guide us to the source of the problem.

Step 4 – Artist

With the root cause of the problem identified, the task of the Oracle Artist is to **create an acceptable solution**. Whereas previous stages have required *analysis*, this stage requires *synthesis*. The Oracle Artist will have to consider the particular environment in fashioning a solution that is not just technically accurate, but also feasible. Both technical and organizational issues will need to be considered, so that the final solution is both technically accurate and acceptable to the customer.

Step 5 – Magician

Finally, the Oracle Magician will **implement and confirm the solution** devised by the previous steps. This is the most exciting part of the whole process; we now activate a solution that has been carefully prepared to clear the performance bottleneck. As a final step, we will also quantify the performance improvement for a subsequent report to management.

With the completion of Step 5, our database patient, formerly so sick and sluggish, now skips out of the infirmary with a clean bill of health. His "chart" has been updated to show his new vital signs, with a copy to grateful hospital management.

Underlying this approach, there are some fundamental principles that we will examine now.

Understanding, Not Trickery

Ironic as it sounds, the successful performance expert will strive to *avoid* tricks, such as special setups or unnecessarily complicated code.

In fact, using exotic solutions, understandable only by the select few in the "DBA-guru club," is a cruel disservice to your employer. The reason for this is simple. In future years, the firm will probably not understand what you did (or *why*). Documentation that you may have left will probably be lost. Other DBAs who were familiar with your work will be gone. The result of the tricky solution by the tricky DBA, is that the application is almost impossible to maintain.

In the very first case study of the book, we see how exciting performance tuning can be. We also get an illustration of the importance of a thorough understanding of basic principles of database operation:

Case Study: Police Department Arrests Otrace

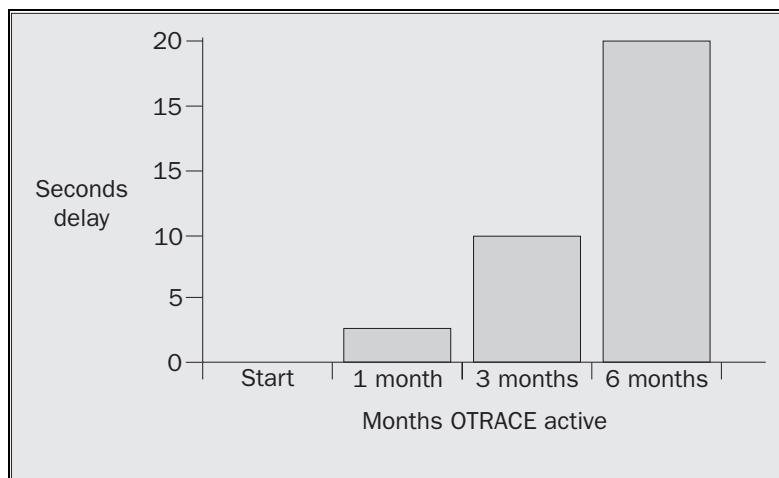
Years ago, Rob went to an interview at a firm that creates public safety programs for police departments. The firm's IT manager wanted to see how the candidates approached their work; thus, the interview was actually an all-day mini-contract. In order to see how the candidate would tackle some difficult database issues, the IT manager demonstrated some performance problems the firm was experiencing with its Oracle 7.3 databases. He did not expect the DBA to solve the performance bottlenecks, but just to discuss what approach he would take.

After some small talk, the manager connected into an important production database and illustrated a particularly vexing problem. He demonstrated how the time to establish a connection in SQL*Plus seemed to be inordinately long (perhaps 30 seconds).

As Rob listened to the manager and observed the problem, he recalled that the symptoms being described perfectly matched a well-documented problem with an Oracle utility called **Otrace**.

The idea behind the Otrace utility was to record various types of database and SQL*Net activity; this information could then be used for later debugging.

With Otrace active, connections to SQL*Plus will take longer and longer over a period of months. This happens because Otrace builds and scans several special data files, which grow larger and larger with each new connection. As the file reaches several megabytes, delays become very noticeable. The following figure illustrates a typical degradation in time to establish a SQL*Plus connection:



After Rob double-checked the documentation, he asked the manager to check the directory where the Otrace files were kept. Sure enough, there were several files that had grown large enough to be a problem. Rob suggested the manager remove the files and follow a simple procedure to reset Otrace. The manager did so, thereby removing the delay due to Otrace.

A quick test confirmed that SQL*Plus connections could now be established instantaneously. One of the firm's biggest performance problems was solved during the interview!

Needless to say, Rob felt pretty good about his showing in the interview. Unfortunately for him, it now appeared that one of the firm's biggest performance problems (and reasons for hiring a DBA) was now solved!

The Otrace performance problem was quickly solved, not because the DBA was incredibly smart, or knew some clever tricks, but because he was aware of some fundamental problems with the database, and recognized symptoms that matched a particular problem. In fact, that particular problem was well documented with an **Oracle Alert**.

An Oracle Alert is a document created by Oracle Corporation that warns the user community of some serious, or even catastrophic problem.

Anyone who had read the alert and had a good understanding of database operations would also have found the root cause rapidly.

This case demonstrates the dynamics of performance tuning and shows how a good grasp of database fundamentals can lead to remarkable results. On the other hand, without a sound grasp of the fundamentals, the DBA must resort to guessing. Of course, Oracle has long since solved the Otrace problem, but similar odd problems will undoubtedly pop-up in future database releases:

Nothing can substitute for a solid understanding of database principles.

This hint might appear to be ridiculously obvious, but experience indicates that this maxim is not often followed in practice. I have witnessed many attempts, by both management and software engineers, to avoid the labor necessary to gain actual knowledge of Oracle fundamentals.

Avoiding Guesswork

We have stressed the importance of the tuning analyst following a sound *process*. The opposite of a sound approach is *guessing*. We will discuss the guessing approach in detail in Chapter 2.

As we have seen, though, it is very common for analysts, administrators, and managers to try to solve a thorny performance problem without attempting any real understanding of database principles. This scenario often manifests itself via some fantastic "tool" that will supposedly analyze the database and churn out the solution. This is sometimes called a **silver bullet**.

The Silver Bullet

Recall that our previous hint suggests that this approach is usually a waste of time. Trying to use some sort of tool without a good understanding of database operation will probably be fruitless.

The point is *not* that all these tools are designed badly. Some *are* poor, but others have been well designed by very competent teams or individuals. The point, rather, is that no tool can compensate for a lack of understanding on the part of the user. In the hands of a competent analyst, a tool can be very helpful.

In other words, *the wrong problem is being addressed*. Instead of focusing on achieving true understanding that will lead to the best solution, the focus is subtly shifted to a vain search for ways to avoid the labor that is required to achieve true understanding. The longing for these types of tricks is so great, that they will be discussed at length in Chapter 2.

In our Otrace case, the firm with the performance problem had experienced the degradation for *months*. The users suffered poor performance while nobody successfully resolved the problem. Were the firm's analysts simply very dull individuals? Of course not. The firm employed very intelligent people; but they simply did not have anyone who truly understood how an Oracle database worked.

Imagine if the firm with the Otrace problem had tried to avoid hiring a competent performance analyst, and persisted in trying to solve the performance problems via a silver bullet like some performance tool. Obviously, that course would have only led to frustration and embarrassment.

Solving the Wrong Problem

In solving the Otrace problem, note also that Rob did not waste time fooling around with the `init.ora` parameters – such as doubling the database buffers, increasing shared pool size, or perhaps some more sophisticated features, such as parallel processing. The novice DBA often applies solutions such as these incorrectly, and therefore fails to solve the real problem.

In the Otrace case study, the symptoms of the problem indicated that steps such as these would have been *completely irrelevant* to the problem in hand. Instead, the solution followed directly from careful listening to the exact problem – just like a doctor listening to the patient describe the symptoms.

Admittedly, it is a bit unusual to solve performance problems during the initial interview, but this scenario emphasizes the theme of this book: the importance of the *approach* that the DBA uses to solve performance problems. This leads us to another hint:

Successful performance tuning requires understanding of database principles, with a focus on uncovering the root cause of the bottleneck.

In other words, the successful performance analyst is no smarter than other DBAs; he has simply learned an overall process that has proved successful. Trickery? No way.

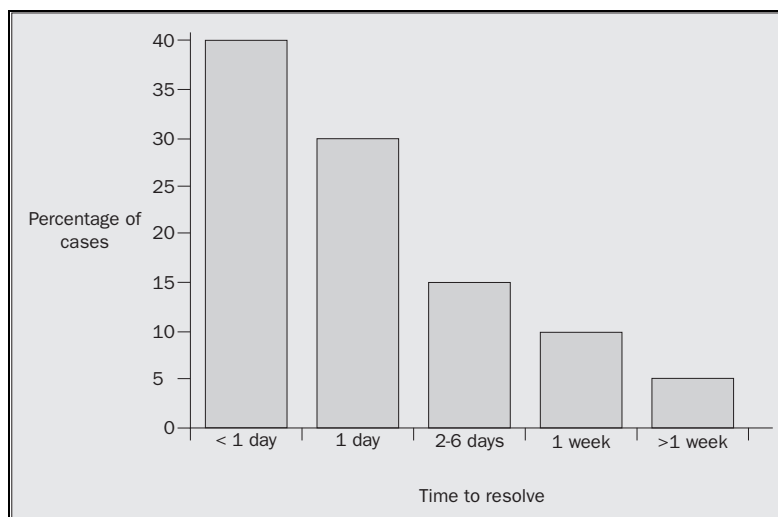
The Big Secret Revealed

After years spent tuning Oracle systems, I have learned a surprising lesson. It is the subject of our next hint:

With a sound approach to performance tuning, many performance bottlenecks are actually very simple to correct.

Considering the big demand for DBAs with performance tuning skills (see the earlier graph), this hint seems counter intuitive. Does this mean that performance tuning is trivial? Not quite. As we have seen with the Otrace situation, many performance tuning problems are incredibly easy to solve. With the wrong approach, these same problems are agonizingly difficult. The high demand for these skills indicates that many analysts do not have good understanding of Oracle principles, or do not bring a sound approach to performance analysis. Perhaps *both* are lacking in some cases.

The following figure illustrates the time required to solve the various performance tuning problems that I have encountered. In my opinion, this graph should be more or less similar for any competent performance analyst. Perhaps super-gurus could do better. In any case, note the large percentage of problems that were resolved within a few days or so; relatively few require many days of intense analysis:



Solutions to the vast majority of performance problems do not require discussions with Oracle Corporation about tweaks or new patches. Instead, it is not unusual for a good performance tuning expert to make just a few changes that lead to markedly improved performance.

Of course, there will always be the odd problem that tests our intellectual prowess. Contrary to expectations raised by some, there will continue to be bugs in database software – whether Oracle, Sybase, Informix, DB2, or SQL Server. We will continue to see things like internal errors and bizarre database problems that test our skill. Nevertheless, the pattern is clear: many performance bottlenecks are resolved in just a few days.

Bizarre database problems seem to have a certain appeal though; thus, I have reserved for the final chapter of this book some of the craziest "database mysteries" I have seen.

Case Study: A Master Tuner Solves a Table Join

Years ago, I consulted for one of the world's aircraft manufacturers. The DBA group comprised about eight DBAs – some very experienced, but others just beginning their careers. I had the good fortune to work with a much more experienced DBA, who also happened to be an accomplished SQL tuning expert.

One morning, while sitting at my desk, I noticed that he was looking at something in a small package. He appeared to be quietly chuckling over some private joke. When I asked him about the small package, he showed me what it was – a nice pen and pencil set with the logo of the firm. He explained that the gift had just been presented to him, in appreciation for some SQL performance tuning work he had recently performed.

He (Ron), being a SQL performance expert, had been asked by another department to look at some long-running reports, and see if he could improve their runtime. Many of these troublesome reports involved table joins. This was an easy (actually trivial) assignment for Ron, because he fully understood the basics of table join techniques.

Ron explained that he had investigated the problem and isolated the bottleneck to one particularly bad piece of SQL code. Having found the root cause, Ron made a slight change in the table join, and *voila* – the problem was gone! The reports now ran in a very short time, the users were thrilled, and Ron immediately became the "hero" to this other department. A week later, to show their appreciation, they presented a gift to Ron.

We will take a closer look at the actual mechanics of table joins, and how we can code them for best performance, in Chapter 9.

As should be obvious from the scenarios presented, the path to tuning expertise is not based on clever equations, or silver bullets, but on an understanding of what steps must be performed to succeed in the performance tuning process.

Keeping Up With Oracle

Faced with a bewildering assortment of performance problems, many DBAs are understandably unsure of themselves when it comes to performance tuning. Without some time-proven approach, the whole process can be intimidating. After all, how can anyone keep up with all the new features?

The marketers of some SQL performance tuning tools exploit this apparent obstacle posed by the enormous breadth of Oracle options. In their marketing literature, some tool-sellers emphasize the impossibility of the DBA fully understanding how to deal with the many flavors of SQL. Continuing with this line of reasoning, the DBA must surely buy a tool (*their* tool) that can rapidly formulate and test a myriad of different SQL combinations.

The huge flaw in this line of reasoning is the assumption that effective performance tuning requires that the DBA understand all the features available in the most recent database version. In truth, the DBA who has a good grasp of database principles can easily account for many of the new bells and whistles in the latest database release. As the first two hints explained, sound understanding is the best long-term strategy to win in the game of performance tuning.

Consider the game of chess. In a chess match, there might be hundreds of possible moves, leading to thousands, then millions of combinations for all future moves. Does a good chess player actually consider *every* possible move? Of course not! It is only the poorest of players (or the crudest of computer programs) who uses that strategy. In reality, the master chess player knows the *principles* of the game, and only considers the options that show some promise, or appear to be relevant. The master's grasp of the game leads him to reject the vast majority of moves summarily.

Calculate Every Move?

The performance analyst can use the same reasoning. He doesn't need to try every option, because he knows that 99% of the choices are irrelevant. Consider the expert tuning analyst who sees a SQL statement that generates a huge amount of disk I/O. His good understanding leads him to reject 99% of the ideas that might theoretically be considered. The seasoned performance expert knows, for instance, that he must first focus on *why* the disk I/O's are occurring. Therefore, he will narrow down his investigation to answering questions that strike at the core of the problem. These include issues such as:

- Is there a full table scan occurring?
- What types of indexes are being used?
- Are the proper columns indexed?

We cover the detail of how the expert knows that these are the questions he needs answers to later in the book.

Performance Problems Yield to Sound Analysis

Experience shows that performance problems are good targets for the application of sound principles of database operation. In spite of the bewildering set of new database options that Oracle Corporation releases, there are some things that stay the same, even in a world of rapid database releases:

The principles of sound performance tuning do not change from version to version.

To illustrate this hint, consider how the Oracle optimizer joins two tables. The concept of performing table joins has stayed much the same, even as database bells and whistles have been added. This means that, regardless of database version, the performance specialist must understand the difference between a nested loop and sort-merge join (for more on these, see Chapter 9). Armed with this understanding, a large number of SQL performance problems can be solved, regardless of the exact database version.

Of course, as technology advances there are some changes that must be noted, such as changes in the algorithm that the Oracle optimizer uses, or new efficiencies that are introduced. In the case of table joins, for instance, the DBA should be aware of the hash joins method that is now often used. Nevertheless, this addition is an easy adjustment for the DBA who is already well acquainted with the other two methods that have been around longer. The DBA who already possesses a good understanding of performance tuning principles will have little trouble assimilating additional features such as these.

Art or Science?

Finally, we come to the question of "art or science?" Some specialists have argued that performance tuning is purely a science – simply the application of sound mathematics to a given problem. This notion is very appealing to technical practitioners, because it suggests a nice, clean, deterministic way to solve any problem. There will be no messiness.

Attractive as this idea is, it does not really appear to match reality. In the real world, life is messy and performance tuning is rarely 100% pure science. Rarely are performance problems handed to the DBA on a nice, clean index card, ready for calculations to begin.

In practice, performance problems take place in complex organizations, filled with the potential for misinformation and the effects of human egos. Instead of getting a well-defined problem report, the DBA often gets *half* an index card, with various smudges and contradictory annotations. Problems don't always start off as black and white issues.

Oops! There's Something Wrong with Your Application!

Even when the root cause of a performance problem can be determined through an heuristic, the solution often calls for creative thinking. For instance, creative thinking (and tact) is often needed when the root cause of a performance difficulty is traced to the application. The DBA may be *positive* that the application is to blame, but the developers may not be so anxious to hear that conclusion.

The designers may resist calls for changing their code, arguing that certain changes are too expensive, or simply not feasible. The professional performance expert cannot walk away at this point, but has an obligation to work with the designers to synthesize an acceptable solution. At this moment, the DBA is more of an artist than a scientist.

Thus, solving serious performance problems is not merely a mechanistic process, such as solving an equation in a school textbook. There are frequently other points to consider, such as determining what problem to solve, wading through misinformation, or dealing with the human factor. Of course, good analytical skills are necessary, but so are good *creative* skills. To ignore either aspect is to misunderstand how technical problems are solved *in practice*. Let's take a look at some case studies to illustrate these points:

Case Study: At some Firms, Performance Tuning is *all* Art!

Recently, while discussing the finer points of performance tuning with a friend, I brought up the notion of "art versus science". I should mention that this DBA works for a large financial firm, as a member of the DBA staff. At this company, there are certainly technical challenges, but much of my friend's time is spent using the softer skills, such as coordinating and discussing issues with developers, managers, and other DBAs. Also, since some of the analysts at the firm are just beginning their careers, he must spend much time educating, mentoring, and explaining issues to the more junior staff members.

Over lunch, I made the point that the DBA rarely walks into a situation with the performance problem nicely written out, as an equation to solve. This means, I continued, that the DBA must have a broader perspective than commonly believed. When I suggested that part of a DBA's work is really art, not science, he enthusiastically replied: "It's *all* art!"

For the DBA in this case, the *majority* of his time was spent on solving the "people" problems. Very little of his energies were spent on producing technical solutions.

Now I think, upon reflection, my friend would admit that his job is not *really* 100% art, but his point is instructive. Depending on the environment, simply being an analytical guru is not good enough. Technical problems do not exist in a vacuum.

In our next scenario, we review an interesting tuning problem that illustrates the importance of solving the right problem:

Case Study: Good Performance but Wrong SQL

At a San Francisco training firm, a DBA was tasked with improving the performance of six separate **Actuate** reports.

Actuate is a web-based application that allows reports to be sent over the Internet.

Most of the Actuate reports were completing in a minute or two, but six reports took up to 30 minutes to run. This run-time was considered unacceptable, especially since it represented *worse* performance than the system being replaced. Additionally, the slow reports were for some of the firm's most important customers, so it was important to identify and resolve this problem correctly.

Analysis showed that each of the problem reports had one thing in common: each report requested a huge amount of data, leading to a long delay as it was transferred over the network. In a manner of speaking, then, one could say that the *network* was the bottleneck, but that would be very misleading. Even though the network was in fact restricting the run-time, a more fruitful avenue of investigation would lead to the question: "*Why is the query being executed in the first place?*"

At this firm, the network bandwidth was fine; the delay time was simply due to the *size* of the data transfer. The real problem had nothing to do with how the network, or the database handled a particular SQL query, but why this query was being run.

With these facts in hand, the DBA met with the report designers, and explained why the report was requiring 30 minutes. The designers agreed that the large data transfer was usually unnecessary and they modified the reports to restrict unnecessarily large data transfers.

Consider how a tool would have handled the report problem. The tool could have offered suggestions on increasing the parallelism of the query, or perhaps trying a different join technique. Maybe spreading the disk I/O's across multiple spindles would have been advised. Of course, all "solutions" such as these would have completely missed the point.

The next scenario shows how just getting the right information to start with is often the biggest hurdle:

Case Study: The Cost of Misleading Information

At a division of a publishing and printing house, the IT manager reported a problem with a "slow database." He indicated, however, that he was sure the problem was related to network speed, and advised the DBA to concentrate on network problems. The DBA, being new to the firm, followed the manager's advice, but could not find anything wrong with the network. Suspecting a very serious, but subtle, network problem, he felt unqualified to take any further action.

A few months later, having some spare time, the DBA once again reviewed the slow connection times. This time, the DBA decided not to restrict the analysis to just the network, but to consider *all* possible causes. Investigation soon showed that initial connections to SQL*Plus required about 15 seconds. The same connection *locally* on the server also required 15 seconds. In other words, the network infrastructure was completely irrelevant. Further checking showed that the connections were slow due to some basic mistakes in database setup. The errors could have been easily corrected months earlier, had the DBA not been misled.

The previous case study shows how misleading information can completely sidetrack an investigation. The DBA was too quick to accept the advice, "concentrate on the network." This advice delayed by several months a solution to the performance bottleneck.

Proactive Tuning

Although many of the examples in this book are based on performance problems discovered after the system had already been released to production, it should be noted that it is preferable to correct problems *before* the system is rolled out. The cost to correct problems is orders of magnitude greater once the bug is lodged in a production system. Because of this, many IT shops employ extensive load testing in order to flush-out performance problems well before deployment.

In light of this, many of the case studies here reflect problems that were in fact corrected prior to production. The reality today, however, is that even with capable developers, many performance bottlenecks will be missed in development, and only addressed when the problem becomes so severe that immediate action is warranted.

This state of affairs is sometimes an indication that proper care has not been exercised during the development and testing phase. Ideally, applications should be thoroughly analyzed using load-testing techniques or similar means, in order to confirm the expected performance and scalability of the system.

Much has been written about the need to detect problems as early in the development process as possible. Learning from this, many large projects use strict quality assurance procedures in order to minimize undetected problems. For instance, the Software Engineering Institute (SEI) has developed rigorous procedures that are used by many Department of Defense projects in the United States (<http://www.sei.cmu.edu/>). Prospective contracting companies must be certified at a certain SEI level in order to be considered a qualified vendor. This process is used to minimize the huge costs that would be incurred to correct bugs that are found in production systems.

Database analysts, whether DBAs or designers, will naturally want to encourage thorough testing of applications for performance problems and scalability, and advise management about prospective performance problems *prior* to release into production.

Summary

In this chapter, we have introduced the proposition that Oracle performance tuning is neither fully art nor fully science. Although technical analysis is certainly a major element, many performance problems require the analyst to deal with a bewildering variety of personalities, and to sometimes separate fact from fiction.

Faced with the confusing set of environments in which performance problems occur, the DBA is forced to *adapt* the tuning process to the environment. This means that the DBA must use different tactics – or "change hats" during the process. We call this the Physician-to-Magician approach to performance tuning.

After looking at several real-life scenarios, we also discussed the importance of proactive tuning, in which the DBA or designer strives to *avoid* problems before they are allowed to impact production systems. Most of the principles discussed in this book will apply to both development and production systems.

In future chapters, we will go backstage to explore the steps followed by performance tuning experts. We will see that there *is* a good strategy for performance tuning – a strategy that adapts for each stage as the DBA exchanges one hat for another.

Before we put on our first hat and start the process, we need to pause and clear away some misconceptions about the Oracle tuning process. These pitfalls are the subject of the next chapter.

