CHAPTER 1

Introduction to Performance Forecasting

It's not just about numbers, and it certainly is not just for geeks. While forecasting is attractive to the more technically minded, the consequences of forecasting impact nearly every human being. All of our friends and all of our families step out into the world each day equipped with the results of multiple forecasts. The classic is the weather forecast. Don't we all look outside when we wake up? Don't we take a quick look at the weather page (online or offline)? It's something we do without thinking, and it would seem strange—even unnatural—not to have a clue about the day's weather.

Just like a weather forecast, the most valuable forecasts seem natural, normal, and in a way, hidden or unnoticed. We don't expect to be impressed by some big mathematical marvel. Rather, it's just part of our existence. When forecasting becomes embedded in our lives, you can be sure that forecasting is adding real value. Why? Because the focus is not on the forecast but on the forecast result: the weather, the stock market, interest rates, political elections, service-level management, and so on.

What is so fascinating to me is that every person loves to forecast. Now they won't tell you that, but they do . . . nearly every minute of every day. Before you step outside in the morning, you take a quick look up into the sky and at the trees to make a quick weather forecast. If you are driving, you make millions of predictions! You predict the person next to you is not going to merge into your lane, but just in case they do, you plan how you would react. When you approach an intersection and the light is green, you predict that no one will run the red light and crash into you. But you may also quickly perform some risk mitigating permutations, just in case someone does run the light. I could go on and on, but as you can see, forecasting is deeply entrenched into each one of us. In fact, it's just not us who needs forecasting. It's broader than that. The world needs forecasting, and it needs people who can do forecasting.

Forecasting and risk assessment go hand in hand. If you want to assess risk and all the information is not before you, you must forecast. Anticipating risk through forecasting provides yourself, your family, your company, your country, and our world with information to avoid disasters.

The bigger the impact of something bad happening, the more value forecasting provides. For example, what's the impact of a Windows PC locking up on you? Well, it's inconvenient, but you've learned to live with it. But what's the impact of a Category Five hurricane? People can die. Because people can die, the forecast value dramatically increases, and people suddenly care strongly about forecasting. In fact, they care so much that governments invest heavily in developing and using forecasting to avert massive disasters.

Is there a significant risk impact when implementing a new business application on a 64 CPU HP Superdome using Oracle Database 10*g* Enterprise Edition with a combined cost in the multiple millions? You bet! This is when forecasting's value skyrockets. It would be foolish to implement a system like this without forecasting to ensure (at some level of comfort) service levels will be met. Not forecasting in situations like these is what I like to call a "career decision."

With IT-related forecasting, our focus tends to be related to answering these questions:

- Does significant risk exist?
- What is at risk?
- When will the risk occur?
- What can we do to mitigate the risk?

Short-term risk assessment is relatively straightforward. It's like looking into the sky to forecast the weather for the next few hours. But don't expect a reliable ten-day forecast without some additional complexity! When systems are dynamic and we forecast far out into the future, forecast complexity will undoubtedly increase. To keep risk to an acceptable minimum, various forecast methods and models will be needed to suit the wide variety of situations in which we find ourselves.

Hurricanes, tornadoes, earthquakes, and Oracle-based systems are full of risk. And because people's lives are intertwined with these potential disasters, forecasting absolutely must occur. With the exception of Oracle-based systems, I think most people would say forecasting has come a long way over the past couple of decades. My hope is that the application of what is contained in this book will help bring a little more respect to forecasting Oracle performance. But that will not happen until forecasting becomes commonplace and something people don't consciously think about . . . like looking up into the sky.

Risk: A Four-Letter Word

Question: What is equally unsettling to a manager who hears someone yell, "Fire!" It's someone who says, "We've got some risk here, and it's significant." A manager doesn't need an exclamation point to start feeling like it's going to be a long day. Managers are trained to identify (hopefully early) and mitigate risk. A good manager knows what risk looks like and knows how to deal with it. A bad manager thinks risk is a board game by Parker Brothers.

I was involved in a capacity-planning project where usage of the word *risk* was closely guarded. We found out early on in the project that the word *risk* carried so much baggage and fear that we should only use terms like *significant risk*, *no significant risk*, or my favorite, *poses no significant risk to the project's success*. Notice the word *risk* was always qualified so managers could better frame the risk situation. Equally entertaining to me was that the word *failure* was never, under any circumstances, to be used. It was acceptable to say, "There exists significant risk to the project's success" but never "There exists significant risk the project may fail." As you can see, management lives and dies by appropriately identifying and artfully dealing with risk.

Risk management implies identification and mitigation. If you cannot identify risk, then you cannot mitigate risk. Initially, forecasting is focused on exposing and identifying where risk resides, how significant the risk is, and when it will likely occur. Not addressing the identified risk is like telling someone there is a small fire in their house, but not telling them where it is—criminal. Mitigating something you have not identified is futile. Once the risk is understood, your effort shifts to focusing on developing strategies to mitigate the identified risk. Identification and mitigation are two separate terms and operations, yet must harmoniously exist for forecasting to bring value to your company.

Much of your forecasting time will be focused on gaining a clear understanding of risk and what to do about it. This may sound simple, but it's not. It can take quite a while to develop strategies to mitigate risk that are politically, financially, and responsibly sound. Risk management is vital; it is an integral part of and a central theme in service-level management.

Service-Level Management

The worldwide IT infrastructure market is in the billions of US dollars.[1] It is that important. A significant part of managing an IT infrastructure is managing the level of service it provides.

Service-level management (SLM) is a very broad and powerful term. It encompasses just about everything we do in IT. SLM is the primary management of IT services, ensuring the agreed-upon services are delivered when and where they are supposed to be delivered. If the services are not delivered as agreed, it can be like a hurricane—devastating.

More than ever, companies live and die by how well IT delivers. It is not uncommon to meet people working for a company whose entire existence is based on IT. If their systems are down, so is the company. So crucial is SLM that companies will spend billions of dollars ensuring IT delivers on its promises.

If you are a developer or a database administrator, SLM is probably not something you think about often. Sure, you are thinking about performance, writing optimal SQL, and tuning Oracle, but the concept of managing "services" seems somewhat abstract and not quite precise enough. It is important for your company and for your career to widen your thinking and the way you speak about performance. Start thinking about how what you are doing affects the service your users will experience. And start thinking about how this service can be measured, monitored, and improved.

SLM focuses on value. *Value* is one of the words that seem somewhat intangible and not precise enough. But if you focus on the return on your investment (ROI), or what your business gains from the service you are providing, you will be thinking in value terms. Here are some examples to contrast the differences between thinking technically versus thinking in value terms:

1. According to the IDC study, "Worldwide Distributed System Management Software 2005–2009 Forecast and 2004 Vendor Shares," the worldwide distributed system management software market achieved revenue of $5.54 billion in 2004 and is forecast to grow to $8.99 billion in 2009, an increase of 62.3% for the period. IDC defines the worldwide system management software market as "including those software tools that are routinely used in IT operations or by end users to manage system and application resources." IDC defines the distributed system management software as the portion of system management software used to manage systems and applications for distributed operating environments, principally Unix, Windows, and Linux.

Technical	Value
This query completes in less then two seconds! I'm a flipping genius!	We can now process 500 more customer calls each hour. As a result, we can reduce our call center head count by 50%, saving us $500,000 each year.
Our upgraded order-entry system is incredible!	We can enter orders twice as fast, saving our company 200 hours each day in order-entry-related costs.
We can now do hot backups!	Now our business can operate 24 hours each day. This allows us to bring on an additional shift, which will increase our manufacturing capacity by 15%, resulting in a 10% revenue increase.

SLM focuses on processes such as reviewing existing services, negotiating with customers, producing and monitoring the service-level agreements (SLAs), implementing service improvement policies and processes, establishing priorities, planning for growth, and the involvement in the accounting process. This stuff bores a technician to death. But to those responsible for IT operations and its future existence, it is life everlasting.

The processes of SLM can be grouped into four management areas:

Capacity management: Ensures the infrastructure is adequate to provide services at the right time, at the right volume, and at the right price. There is a definite focus on efficiency; that is, minimizing cost while maximizing service. As you might imagine, capacity management relies on performance and workload monitoring, application sizing, forecasting, and performance modeling. Forecasting Oracle performance is most closely aligned with capacity management.

Continuity management: Focuses on ensuring IT can recover and continue operations should a fault occur. This is not simply about backup and recovery. The focus is the process, the plans, and their implementation. For example, continuity management involves prioritizing the processes to be recovered and performing a risk assessment for each IT service to identify the risks, threats, vulnerabilities, and countermeasures. Also included are evaluating recovery options, producing a contingency plan, and ensuring the plan actually works and is kept up-to-date.

Availability management: Focuses on identifying availability requirements and mapping those to service levels. The objective is to leave no service level unmeasured and undefined to enable SLA measurement and compliance. Areas to focus on are reliability, recoverability, maintainability, resilience, and security. For example, the production database availability SLA may require the database to be available 24 hours a day, Monday through Friday. The availability manager says we must devise a method to measure the database availability to ensure it operates within the defined service levels.

IT financial management: Focuses on ensuring the IT infrastructure is obtained at the most cost-effective price while meeting service levels. Those involved or contributing to this management area will be very familiar with cost benefit analysis (CBA) and ROI analysis. All of IT is included: equipment, software (operating system, database, tools, business applications, and infrastructure software), organization (staff labor), facilities, and outsourced operations (such as human resources activities). Capacity, continuity, and availability management all feed into IT financial management.

Personally, growing up in IT focusing on Oracle database management, the whole concept, process, and implementation of SLM has been overwhelming. I simply could not get my hands around it. I was not trained to be a manager of people and organizations. I was trained to be a manager of data and to effectively communicate. Over the years, and in particular as the result of bringing an SLM product to the market, I have learned (more like had forced down my throat) to view value in terms of service to a company—not to myself or even to my users, but to the company. You and I might not like what all this may imply, but the reality is that companies employ us because they must provide their customers with certain specific values and their hope is we can help in this process. In exchange for our services, they compensate us in a variety of ways that we must deem acceptable.

Our forecasting work must feed directly into the management of service levels. In fact, SLM should drive and provide focus for our forecasting-related projects. If you are a hard-core technology person, this can be difficult. I suspect that most of you reading this book have this kind of background, so throughout this book, I have taken extra care in helping you demonstrate the value of what you are doing and how that can support SLM-related initiatives. If our forecasting work does not support the management of service levels, then eventually its shrouded value will be unveiled to show a pure technical exercise, and our efforts will be put to an uncomfortable halt. Performance forecasting provides an opportunity for technologists to provide a significant value to their company—much more value than just simply performing a technical exercise and forwarding the answer to someone.

Modeling: Making the Complex Simple

The world is much too complex for us to understand. To compensate, we create models to simplify the world around us, turning it into something we can comprehend. Models help us understand our world. When focused on a specific area in which the scope is very limited, a model can teach, expose, describe, reveal, and draw out the understanding we may desperately need. Modeling is all about making the complex simple.

Models are designed to be very focused and limited. This allows them to reveal to us what we need to know, while at the same time not address the other areas of complexity. For example, consider a paper airplane. Although it's extremely simple to construct and examine, it can help teach us about basic aerodynamics. But like all models, it has a very focused and limited use. That same paper airplane will not teach us about landing gear or traveling at supersonic speeds. To learn about these things, a different model is required. In our work, we may choose a ratio model to forecast batch processing load, while choosing queuing theory to forecast online transaction processing (OLTP) load. The trick is to match the requirements with the model.

SLM requires models in many different areas. IT financial management will have, at a minimum, cost benefit and ROI models. Capacity management will have resource forecasting models (like the ones I will present in this book). Any time a forecast (financial, CPU utilization, workload activity, or stock price) is made, it is based on a model. Modeling is central for all forms of forecasting.

I like to say that models are an abstraction of reality. From paper airplanes to supermodels, they are not real but an abstraction. One of the dangers to those of us deeply involved with modeling is believing that our models are real. This may seem strange, but after spending weeks developing a forecasting system, you can start to get a little strange. People who suggest or hint that your model is not perfect will rub you the wrong way. So please, remember that models are not reality. They are not perfect, cannot be expected to be perfect, and are not designed to be perfect. It would be foolish to think what a model says will absolutely 100% occur. Even proprietary forecast models are not perfect—close, but not perfect.

Have you ever seen hurricane path forecasts? They usually contain many different forecast models, all created for the sole purpose of forecasting where a hurricane will travel. But notice that no two models forecast the exact same path! That's because each path is based on a forecasting model. Even the best models are not perfect, because nature, life, and computer systems are so very complex.

Don't be impressed by seemingly complex models. Just because a model is complex does not mean it produces more value. For example, adding independent values to a regression formula typically reduces the forecast precision. On the flip side, a simple model may not have the wherewithal to provide the insight you need and to grapple with real system complexities. A model car is not going to show you how fast your real car can take a turn without sliding off the road. The perfect model is one that allows you to understand something you need to understand with minimal cost and effort.

Models are central to managing service levels and to forecasting Oracle performance. Contained in this book are some very simple to some fairly complex models. As you'll come to understand, picking the right model for the right job is crucial.

Model Types

Models come in all shapes and sizes. When forecasting Oracle performance, you need to be comfortable using a wide variety of mathematical models. Each model has a natural affinity to certain types of forecasting. For example, regression analysis may work wonderfully when you want to determine the maximum number of orders your system can handle. But add to that question a hardware configuration change, and you'll need a different model.

Most of us tend to gravitate toward certain models. Some people really like to use ratio modeling; some are dangerously fanatical about queuing theory. Don't be like the new Oracle DBA who goes to a conference, listens to a presentation on latch contention, and then upon return, proclaims that Oracle is suffering from significant latch contention. Take the time to understand, appreciate, and use a wide variety of models. Knowing which model to use in a variety of circumstances greatly increases your forecasting prowess.

There are three fundamental model types; mathematical, benchmark, and simulation. For sure, there are other forecasting models available. But these broad model categories will cover our work nicely, not leave gaping holes, and prepare us to assimilate other models when necessary.

Mathematical Models

Mathematical models are fast, flexible, portable, and therefore extremely useful. They are also the focus of this book! While the underlying math can be complicated (but as you'll see in upcoming chapters, it does not have to be), once developed, the models typically run in under a second. Flexibility is important when forecasting Oracle performance. It seems like no two projects are the same, and you'll need the flexibility and portability to modify an existing model. Most mathematical models can be implemented in a spreadsheet or in a third-generation programming language.

In this book, I'll present many mathematical forecasting models. We'll cover basic forecasting math, how to improve on the basics, ratio modeling, queuing theory, and regression analysis. This will equip you to forecast in a wide variety of situations.

Someone once asked me to forecast the performance improvement if he upgraded his Oracle database. Another person asked me which Oracle latch would raise its ugly head first. I told them both the same thing: "The closer you get to Oracle's internal and proprietary algorithms, the less useful a mathematical model becomes." While a mathematical model can forecast a variety of things, implementing algorithmic details is usually beyond the scope of our knowledge, time, money, and sanity. In fact, many algorithmic details are proprietary and simply not meant for us to know. When you need to see what specifically will break, you need to benchmark.

Benchmark Models

Benchmarking gets a lot of good press because that's what MBAs and project managers are taught. At the same time, these folks tend to loathe mathematical models, even though they use them heavily in financial analyses . . . go figure. Many implementation processes require some type of performance assurance test to be performed. When project schedules and budgets get tight, the project manager will retreat to what is the most comfortable and seemingly the lowest risk.

A benchmark is a model because it simplifies something complex, highlighting areas we want to study. As explained in the "Differences Between Benchmarks and Simulations" section, a benchmark is not a true simulation because it runs the system it is studying.

The beauty of a benchmark is it is a copy of the real system and therefore will behave exactly like the real system. This is true until the benchmark deviates from the real system, which is nearly always the case. When this occurs, all sorts of problems can occur. If a benchmark fails (and many do), it is probably due to the workload. You must spend a great deal of time, energy, and money to get the workload correct for a benchmark to be meaningful. The scary thing about a bogus benchmark is that most people don't know it's bogus until the real system is placed into production. By then, it's too late.

The other problem is benchmarks contain physical computing components. Depending on the benchmark scope and architectural complexity, it can take a prohibitive amount of time to physically construct the system, load the initial data, warm up the benchmark, finally run the benchmark, and then repeat this for each scenario. Doing this many times can take what seems like forever.

While there are many negatives surrounding benchmarks, a benchmark has the potential to provide the most precise forecast of any model.

Simulation Models

I like to say a simulation is a *living* forecast model. Simulations are very different from mathematical equation-based models like regression analysis or queuing theory. With simulations, the solution is not solved or evaluated. With simulation, we create a controlled environment where artificial transactions are placed into the system. The transactions work their way through the system just like real transactions in a real system. The beauty of simulation is that the model is instrumented to closely observe and record transaction activity. To find out how the real system would react to certain changes, we can introduce these changes into our model and run the simulation. When the simulation is complete, the data is analyzed, and we can learn a lot about the real or proposed system.

Nearly every scientific discipline embraces simulation: manufacturing floor layout, managerial staffing decisions, chemical engineering, civil engineering, nuclear physics, mathematical research, computer games (think SimCity)—the list goes on and on. Simulation is fascinating. It can include eye-popping graphics, produce a plethora of statistical data, and be incredibly precise. However, it can also take a long time to complete a single forecast and, in many cases, be practically impossible to construct.

Usually when forecasting computer system activity, mathematical equations and benchmarks work just fine. But sometimes a mathematical solution is impossible given the current state of mathematics and the complexities of what we are trying to model. In these situations, simulation may offer a reasonable alternative. If we insist that all forecasts must be solved mathematically, we may find ourselves oversimplifying or overabstracting the system so that it can be mathematically solved in order to come up with a mathematical solution. This can sacrifice precision and result in bogus forecasts. Simulation enables us to forecast what may be mathematically impossible or mathematically inappropriate.

Not too long ago, simulation had to be performed by hand or using handheld calculators. This made running even a single simulation very time-consuming and expensive. This, combined with the model callibration process, prohibited the use of many simulation models. However, the world is not as it was. The introduction of grid computing (by BigBlueRiver, `www.bigblueriver.com`) provides massive computing power at a relatively low cost. As a result, I suspect we will see a rise in simulation modeling in the coming years.

What prohibits Oracle forecasting using simulation is model construction time and the time required to run a single simulation. Callibration of a simulation model requires thousands of iterations, and when identifying risk, we typically run hundreds of forecasts. Even a single simulation can take a few minutes to complete on a standard office PC. And, of course, there is the cost of the simulation software. Unless you are creating a professional forecasting product or have access to a lot of computing power, using simulation for forecasting Oracle performance just doesn't make good sense.

Differences Between Benchmarks and Simulations

There are significant differences between a simulation and a benchmark. And calling a benchmark a simulation will cause communication confusion with both IT and just about every scientific community.

A simulation program does not run on what is being studied. For example, during a nuclear physics simulation, actual atoms are not split. Traffic simulations are not run on the highway, but rather inside a computer. This is in stark contrast to a benchmark, where the benchmark is actually running Oracle, the software application, and the operating system on particular hardware.

When I started creating Oracle load generators, I used to say I was simulating an Oracle load. I had a rude awakening when I begin looking for *simulation* books. I quickly discovered there is a massive industry and field of study whose focus is squarely on true simulation. And it had absolutely nothing to do with my *Oracle simulations*. In fact, I felt a little embarrassed at being so presumptions by calling my load generator a simulation. As I begin to learn, study, and actually write simulation programs,[2] it became clear that Oracle benchmarks and simulations

2. OraPub's SLM product, HoriZone (`www.horizone.orapub.com`), sometimes uses simulation models to forecast Oracle performance. Creating the simulation program to specifically forecast Oracle performance brought about a whole new level of respect for the simulation community. Simulation programs can be incredibly complex. Calling an Oracle benchmark a simulation is like calling a bicycle an automobile.

where two distinct industries and topics. While they have a common goal, how they are constructed, run, and analyzed are very different.

To avoid confusion and not offend those in the simulation industry, I started using the words *synthetically loaded system* to describe a benchmark. While I admit it sounds strange, especially when talking to a group of Oracle-centric folks, it certainly differentiates the two models and eliminates any confusion.

NEURAL NETWORK MODELS

To me, neural networks are the most fascinating models of all. However, I have yet to find an appropriate Oracle-centric application of neural networks.

Just what is a neural network? A *neuron* is a relatively simple biological cell, and we have a lot of them in our brains. An artificial neuron is the simple mathematical equivalent. Each neuron can be connected to many other neurons. When a neuron receives energy, it holds the energy until it reaches a threshold, and then releases the energy to all the neurons to which it's connected. The same thing happens to all the neurons in the network, resulting in a type of cascade reaction. By specifically manipulating and controlling a neuron's threshold activity in conjunction with the network architecture, a neural network can hold an amazing amount of information.

Neural networks are terrific at recognizing patterns. They will beat any other model in recognizing a known pattern. However, if the neural network has never seen the pattern, then it essentially guesses. And guessing is not good when forecasting Oracle performance. I have spent untold hours working with neural networks, heard academics speak on the subject, and conversed with an author on the subject. I have yet to hear of a neural network that can forecast Oracle performance. However, proprietary forecasting models do sometimes incorporate aspects of neural network technology.

Challenges in Forecasting Oracle Performance

No one knows better how difficult it is to do a good job forecasting Oracle performance than traditional mainframe capacity planners. A few years ago, I attended a major capacity planning conference. During one of the presentations, the speaker was describing an architecture. When he mentioned the word "database," there was a slight grumbling in the room, and you could feel the angst. Being an Oracle person, I was starting to feel a little uncomfortable. The speaker continued and all was well. But later, he mentioned the word "Oracle." Even the speaker grumbled, and the entire room joined in. At this point, I quickly located the nearest exit. I am not exaggerating, and I was perplexed.

After my escape, I spent some time ruminating as to why people whose lives are dedicated to capacity planning did not embrace the elegance of an Oracle database. Then it occurred to me: capacity planners like transactions and their characteristics very well organized and well defined. (That's why I had never got much out of capacity planning books: they didn't address the complexities of an Oracle system.) Oracle transactions and their resulting workloads are very difficult to characterize, enumerate, and understand.

One of the Oracle database's selling points is its speed. This incredible speed is the result of years of research and development focused on *optimization*. Optimization makes Oracle fast, but blurs resource usage responsibility and accountability. For example, Oracle uses caching to reduce physical input/output (IO) for future transactions, utilizes complex transaction concurrency algorithms, and groups multiple data-block changes into a single physical write.

When caching and batch-writing algorithms are used, Oracle is faced with a daunting resource assignment challenge. For example, suppose three sessions change a single cached database block and a fourth session reads the cached block. (Which triggers the cached block to be physically written to disk.) How should the resource usage be assigned? Oracle could divide the IO activity equally between the four sessions, or it could simply assign the IO to the last session. And consider this unfortunate fact: the more perfect resource assignment becomes, the more resources are consumed to determine the resource assignment.

Ultimately, a resource assignment decision must be made. Oracle typically assigns resources to sessions for the code they run and operating system resources they consume, regardless of whether the session is also doing work caused by other sessions. The hope is, with a little abstraction, the error will not be significant. So while resource-sharing strategies can yield improved performance and high concurrency, perfectly assigning responsibility for every activity is impossible.

The job of measuring, understanding, and developing mathematical representations of something that is constantly changing is just plain difficult, if not impossible. For an Oracle system, application activity constantly changes. In fact, computer software application usage changes by the second. Computer hardware components (such as CPU and memory) are frequently upgraded and therefore change. Change also makes a perfect (even a near perfect) workload measurement impossible. To represent these complexities, we must abstract.

Oracle transaction resource requirements fluctuate wildly. You must be very careful in determining, for example, the average CPU time it takes to service a transaction. For example, querying a customer may consume between 500 to 5,000 milliseconds of CPU time. This average is, you guessed it, a method of abstraction.

We talk a lot about the "transaction." This word is probably on just about every page in this book, and nearly every forecast model uses the word *transaction*. But what is a transaction? Unfortunately, there are many accepted definitions for this word. The definition of an Oracle database transaction is different from the definition of a business transaction. An aspect of forecasting is communicating a workload in ways that a person can easily grasp. A person can easily understand a business transaction. For example, the entry of a single order may be considered a business transaction. However, an Oracle transaction is a well-defined technical term.[3] Oracle transactional information, which must be queried from the Oracle database, may have no meaningful relationship to the business. So once again, we must generalize—that is, abstract to better communicate.

Database optimization, transaction resource variance, change of all kinds, and basic word definition challenges all contribute to make typical Oracle workloads very complex and difficult to enumerate. Some people will tell you that the required abstraction is far too great to enable any reasonable forecast. But those of us who are responsible for service levels can't simply say, "It's too hard. Let's buy tons more hardware!"

Reasonable forecasts can be made and are made all the time. In fact, an entire industry is dedicated to forecasting performance. If reasonable forecasts are not possible, then these companies are surely liable and operating with an unethical intent. Every responsible forecast contains steps to validate precision. If the precision is not acceptable, then the model is not used. My personal experiences have shown that forecasting can produce very reasonable forecasts. A reasonable forecast does not imply a high-precision forecast (less then 5% error),

3. An Oracle transaction begins when data changes and ends with either a `commit` or a `rollback`. There is no implied relationship with the business.

although sometimes this is required. The bottom line is this: if validated forecasts are within the acceptable precision tolerance, then forecasting is fulfilling a vital IT service.

These challenges are some of the reasons you don't see many good Oracle forecasting products on the market. What most people typically do is significantly abstract the workload while retaining the relevant aspects of the workload. This process is called *workload characterization*, and I've devoted an entire chapter (Chapter 7) to the subject.

With these challenges, it's no wonder that the conference speaker grumbled when he uttered the word "Oracle." Forecasting can be very frustrating sometimes. But challenges also present opportunity! Sure, you have to be creative and a little gutsy, but it can be done. No matter what people say, it is possible for you to responsibly forecast Oracle performance in a wide variety of real-word situations using a variety of forecasting models. I'll lead you through this process over the remaining chapters of this book. We'll take it a step a time, starting with the essentials and then building some very powerful forecasting techniques.