

# ENABLING THE MAXIMUM AVAILABILITY ARCHITECTURE ON LINUX

*Edward Whalen, Performance Tuning Corporation*

## **INTRODUCTION**

High availability architecture and disaster recovery are an important part of any critical database and application system. In the event of downtime, many applications could lose thousands or millions of dollars in revenue. The Sarbanes-Oxley Act of 2002 (SOX) regulates internal controls used for corporate information of publicly traded companies. This includes the need for system integrity and disaster recovery planning and implementation. Because of the fact that a fully functional Disaster Recovery site involves a heavy investment in equipment, Linux is an ideal platform. Linux provides both stability and value.

The Oracle Maximum Availability Architecture (MAA) is a blueprint developed by Oracle in order to provide both High Availability (HA) and Disaster Recovery (DR) capabilities for Oracle environments. The HA components provide protection from the loss of a single component, whereas the DR components provide protection from a loss of a complete system or the loss of an entire data center. The Maximum Availability Architecture is made up of redundant components at all levels of the application stack; web servers, application servers and database servers. This paper explores how the MAA can be used to provide protection in a number of disaster scenarios.

## **HIGH AVAILABILITY AND DISASTER RECOVERY**

High Availability (HA) and Disaster Recovery (DR) are the key goals of the Oracle Maximum Availability Architecture. HA and DR are similar in that they both involve the goal of maximum uptime, however, HA usually revolves around surviving the loss of a single point of failure, while DR involves picking up the pieces after the loss of an entire system or data center.

### **HIGH AVAILABILITY**

High Availability or HA is a design implementation with the purpose of creating a system that is operational to a particular standard (under normal circumstances). These standards are typically referred to in terms of 9's. A system with an uptime requirement of three 9's is said to need to be up 99.9% of the time. Four nines will be 99.99% of the time, etc. The standard for HA is typically the ability to withstand any single point of failure. For example, the system would be designed with a RAID disk subsystem that can handle the loss of any single disk drive, or any single component. The single-point-of-failure criteria doesn't usually allow for multiple points of failure. For example, the system can handle the loss of a single disk drive because drive mirroring has been employed; however, it cannot withstand the loss of multiple disk drives at the same time.

HA systems employ redundant components - such as redundant memory, RAID disks, network cards - in order to provide the ability to survive a single point of failure. At a higher level, redundant application servers, network components, and database servers are used in order to create a system made up of many redundant components. A system can also be designed to have multiple redundant components; however, this does not make it capable of surviving a disaster, such as the loss of the entire data center.

### **DISASTER RECOVERY**

Disaster Recovery or DR systems are designed to resume normal business functions in case of a disaster. There are many types of disasters, but this paper will focus on disasters that affect the entire data center, or the entire database system. In the event of one of these types of disasters, having redundant components such as RAID disks doesn't solve the problem. It is important that the entire database system and application servers are redundant. In the event of a disaster, the user community will be redirected to the standby data center in order to resume operations.

Often the switch over to the standby data center is not instantaneous and some data might even be temporarily lost, depending on how things are configured. The main goal of the disaster recovery solution is for your company or organization to stay in business. No business can afford to be down for long periods of time. For some companies each hour that the system is out of service might result in thousands or millions of dollars in lost revenue. Thus it is important to have some sort of DR system in place, even if it runs at a degraded performance; at least it keeps the company in business.

## **TYPES OF DISASTERS**

There are a number of different types of disasters that can affect database systems. They can be widespread or isolated and include some of the following:

- System Disasters such as code bugs, application errors, human errors or sabotage. A loss of a system due to a hardware failure could also be considered a system disaster.
- Natural Disasters such as hurricanes, tornadoes, floods, earthquakes could disable an entire area for an extended period of time
- Isolated Disasters such as fire could disable an entire building for a significant amount of time

Each type of disaster affects single or multiple components of the system and must be protected from slightly differently. However, some of the concepts are the same.

### *SYSTEM DISASTERS*

System disasters can be the easiest to deal with in some respects and the most difficult to deal with in other respects. They can be easy to deal with if they simply involve the loss of a component or a system for a short period of time. These types of disasters can be resolved by simply fixing the problem and moving on with business as usual. Other types of system disasters can be much more difficult to solve, since sometimes you might not even realize that the problem has occurred. For example, a user might input the wrong data and either try to cover it up, or not realize it for days or weeks. It can be very difficult to fix a problem that happened weeks ago, since other activity has been going on in the system in the meantime.

Solution(s): Ways to prepare for and to recover from a system disaster include good backups that are kept for a significant amount of time, the use of the UNDO tablespace, good QA and testing, and good security.

### *ISOLATED DISASTERS*

Isolated disasters involve the loss of a computer system, a storage system or even the entire data center. This is not as bad as the natural disaster because you still have access to personnel and resources that are local to your data center. In the event of a loss of a single system, it might be possible to either continue service or to use a spare system in order to get back into service fairly quickly. In the event of the loss of the entire data center, it will be necessary to switch over to the standby data center.

Solution(s): Ways to prepare for and to recover from an isolated disaster include good backups that are kept for a significant amount of time offsite and the use of an offsite data center which could be a local site.

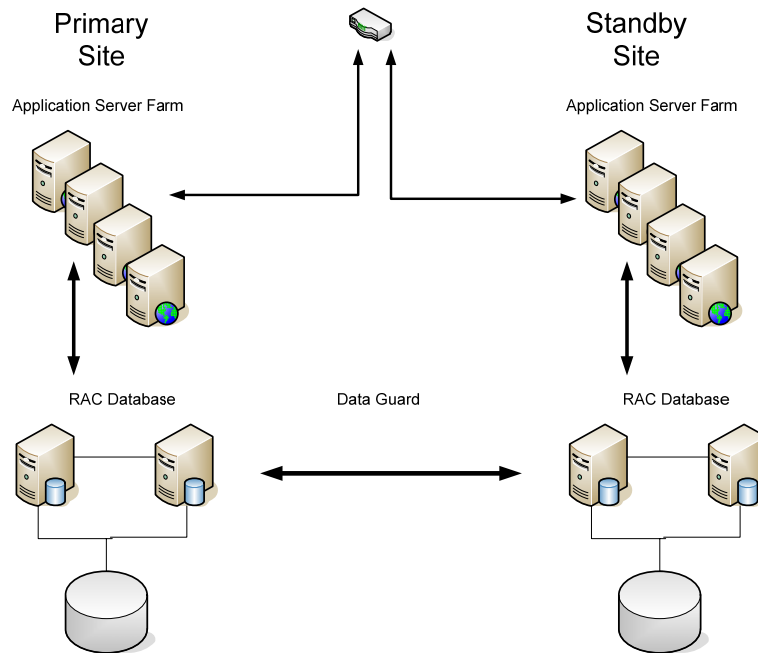
### *NATURAL DISASTERS*

Natural disasters create the most destruction, since this type of disaster is widespread. The entire data center may be out of commission for days, weeks or even months. In addition, all aspects of the data center, including personnel are affected. To survive this type of disaster, the entire data center (or a subset) must be reproduced at a different location. In addition, it cannot be local, since natural disasters, such as floods, could affect the entire region.

Solution(s): In order to protect against a natural disaster, the only solution is to design and implement a full standby data center. This standby data center must be located sufficiently far enough from the primary data center that a widespread outage caused by a flood, earthquake or hurricane will not affect both the primary and standby data centers.

## **THE ORACLE MAXIMUM AVAILABILITY ARCHITECTURE**

The Oracle Maximum Availability Architecture (MAA) is a framework designed to provide both high availability and disaster recovery components for the capability of surviving multiple disaster scenarios. The Oracle MAA involves the creation of fully redundant components including the web server tier, the application tier and the database tier. It employs many technologies including Oracle RAC and Data Guard in order to provide both HA and DR attributes. The entire MAA is illustrated in the following figure:



*Oracle Maximum Availability Architecture Overview.*

The MAA is made up of a number of components and layers. By creating full redundancy and high availability, both performance and protection can be maintained. Locating the standby data center further from the primary site provides more protection from natural disasters, however, the further the standby data center is from the end user, the slower it will be. Thus there are trade-offs to be made between protection and performance. This is why the MAA is designed to provide HA components as well as DR components. HA will take over when it can, and the DR standby site is to be used only when it is absolutely necessary.

Whether a full standby data center that is equivalent to the primary data center is created or a smaller scale or subset is created depends on business needs. If the service level agreement allows for degraded performance for a period of time, then it might be acceptable for the standby data center to have fewer application servers, or a smaller RAC cluster, or maybe even no RAC cluster at all. This all depends on the needs of the business and the Service Level Agreement.

As mentioned earlier, the MAA is designed around redundancy. This redundancy must exist at all layers of the system so that the entire application stack will function in the event of a failover. Thus the MAA design must include web servers, application servers, database servers, storage and network and storage infrastructure.

### **APPLICATION LAYER REDUNDANCY**

Web and application tiers should be configured using HA techniques by using multiple servers for both web and application services. These servers are typically set up with both load balancing and failover. Thus the loss of a single component might only result in a small performance degradation. Since the web and application servers support load balancing, an additional benefit is the ability to add more servers as you need to. Linux is an ideal platform because of the ease of purchasing additional hardware and the relatively low cost of that hardware.

In order to support Disaster Recovery, a standby data center must be used and must contain web servers, application servers and database servers. In addition, these servers must be maintained at the standby site. The applications themselves can be maintained in several ways including the following:

- Manual synchronization. It is possible to manually maintain the standby site. This is done by updating the applications at both the primary and standby sites whenever changes are made.
- OS Replication. There are replication packages where changes made to OS files are replicated to the standby site. This allows automatic synchronization.

- Storage Replication. Many storage systems support options that allow entire LUNs to be replicated from the primary site to the standby site. This allows automatic and continuous synchronization.

Typically the application tier is not changing dynamically. Applications are being used, but are not continually modified; therefore sometimes simpler methods can be used for synchronization.

There are several ways to use the standby application tier. In the event of the loss of the application tier at the primary site where the database tier is still functional, the standby application tier can be redirected to the primary database tier. However, this is not the norm. Typically the standby data center is an all or nothing design, where the failure of any tier at the primary site brings the entire standby data center into operation. This is more efficient, because the web server tier, application server tier and database tier are all in close proximity with each other. This is another reason why the MAA counts on HA to handle system problems that are localized.

### **DATABASE REDUNDANCY WITH RAC**

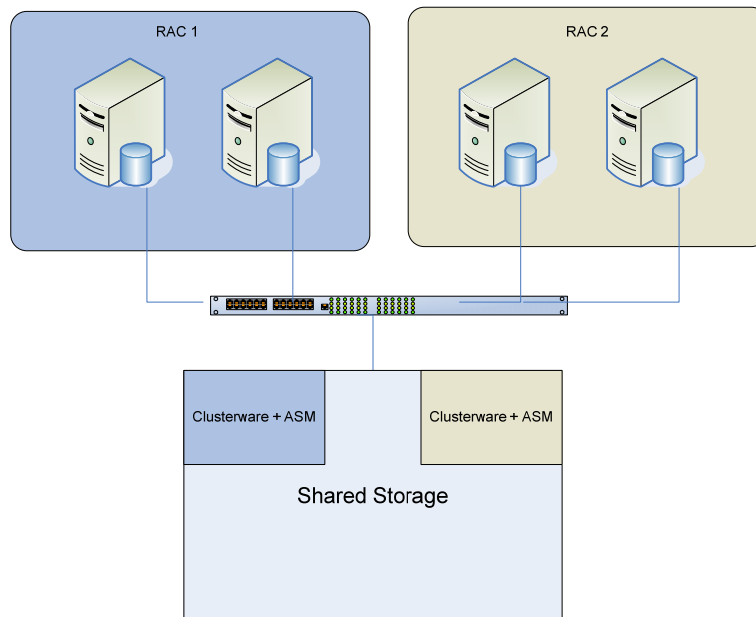
Oracle RAC is used to provide HA at the database tier. Oracle RAC allows multiple instances to access the same database simultaneously, thus providing both high availability and scalability. Oracle RAC has been around for many years and is a stable, high performing product. However, RAC can only be thought of as a HA product. RAC does not provide any DR capabilities.

#### *INDIVIDUAL CLUSTERS VS. SINGLE CLUSTER WITH MULTIPLE RAC DATABASES*

There are several ways that a RAC cluster can be constructed to support multiple databases. One method is to create each RAC database with its own clusterware and shared storage. In this method each cluster would have its own clusterware, Oracle Cluster Registry (OCR) and voting disk. This provides complete separation of resources, but does not easily allow for the Grid concept of RAC in which nodes can easily be added and removed as needed. The second method is to create a single clusterware and ASM that is used by all nodes, and then to individually create RAC databases within that infrastructure. Thus all of the nodes will be treated as a single cluster, but lock and data traffic for each RAC database will be limited to the nodes where that database resides.

#### *INDIVIDUAL CLUSTERS*

With the individual clusters each RAC database has its own clusterware and shared storage for ASM. This provides complete separation from other RAC nodes that might be using the same shared storage. This separation means that maintenance such as patching to the database and clusterware Oracle Homes will not affect any other systems. An illustration of this is shown here:



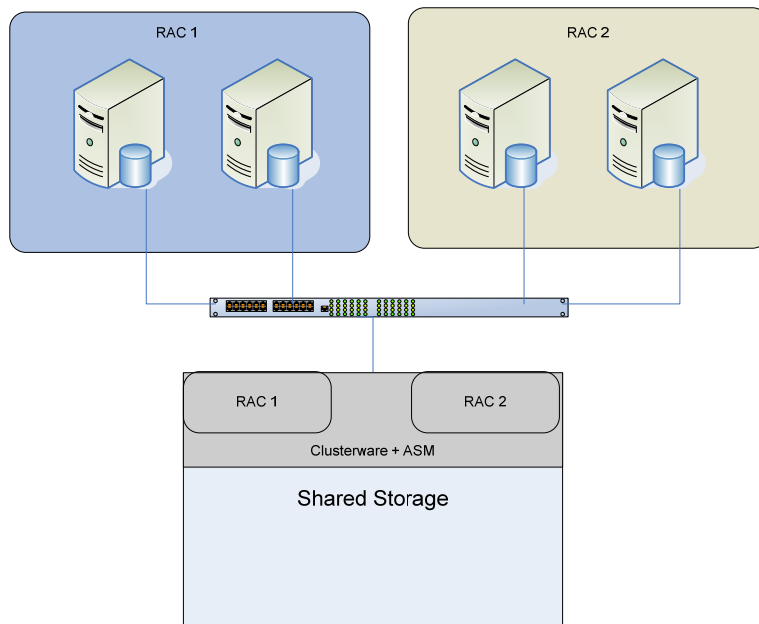
*Separate RAC Clusters.*

In this model, the RAC clusters are completely separated. The same shared storage might be used by both RAC clusters, but it is not required. Individual interconnects need to be configured between each RAC cluster as well.

The advantage of this model is that there is complete separation. The disadvantage is that since each cluster is separate, it is difficult (but not impossible) to interchange systems. It is also impossible to create multiple RAC databases that share the nodes between the two clusters defined here.

### *SINGLE CLUSTER WITH MULTIPLE RAC DATABASES*

The second method is to create one set of clusterware and shared ASM across all nodes in the cluster and then to create separate RAC databases that span separate nodes in the cluster. With this method the nodes can be easily exchanged between RAC databases by using only database commands. No storage or network reconfiguration is required. This method is illustrated in the following figure:



*Separate RAC Databases with Shared Clusterware and ASM.*

Since you can have multiple RAC databases within the same RAC cluster it is even possible to have a third RAC database that spans the entire four nodes of the cluster. With this model, it is the RAC database determines the clusters. This model provides maximum agility for future growth and modification. It is an easy matter to take an existing node out of one RAC database and add it to the other, or simply to extend one database to the other node(s).

As with most things, there are both advantages and disadvantages to this model. The advantage is that nodes can be allocated to each RAC cluster by performing database commands or via DBCA. No system administration intervention is required. It is also possible to have a system be part of two RAC databases by extending RAC 1 to the system that RAC 2 is running on.

The disadvantage is that maintenance and patching must be done across all of the nodes, since the clusterware and ASM instance spans all of the systems shown here.

Tip: Create ASM using its own Oracle Home. This will allow maintenance on the database Oracle Homes for patching without having to bring down ASM.

### *RAC SCALABILITY*

RAC scalability can be severely diminished by improper application design. RAC scales extremely well in a read-only environment, and even when updates, inserts and deletes are being performed on different parts of the database. However, when the same block or row of data is modified by multiple nodes in the cluster, the scalability is diminished. This is due to heavy lock traffic and data passing between nodes in the cluster. If a node in the cluster wants to update a row that has been modified by another node in the cluster, a sophisticated locking mechanism must be used in order to release the lock on the

node that is currently holding it and then pass locking information and data to the new node. If either of these two nodes are not the block owner, a third node is involved in this process as well.

Therefore, a good RAC application design must avoid modifying the same data from multiple nodes. This can be done via the use of Oracle services. With services you can point specific users or batch jobs to specific nodes. In the event that the desired node is not available, the standby node will be used. By segmenting the workload, RAC performance can be optimized.

## **DATABASE REDUNDANCY WITH DATA GUARD**

The database DR solution of choice is Oracle Data Guard. Oracle Data Guard maintains a replica of the Oracle database on a standby database system or RAC cluster. This standby system can be used in the event of a failure of the primary system. Data Guard can maintain the standby system in a number of different modes, based on your requirements and configuration.

### *DATA GUARD MODES*

Data Guard can be configured to run in three different modes. Since a primary database can have up to 9 secondary servers, it is possible to mix and match (i.e. use Maximum Protection Mode for local standby databases and Maximum Performance Mode for remote standby databases). The Data Guard modes are:

- **Maximum Protection Mode.** In this mode a transaction does not complete the commit operation until the log record has been written to both the primary redo log and at least one standby redo log file. This guarantees no loss of data in the event of a system failure; however, Maximum Protection Mode can cause performance issues with the primary database if the connection between the primary and standby databases is not fast enough. If the connection is broken between the primary and standby systems, the primary will shut down, thus assuring that there is no data loss possible.
- **Maximum Availability Mode.** As with the Maximum Protection Mode, in this mode a transaction does not complete the commit operation until the log record has been written to both the primary redo log and at least one standby redo log file. This guarantees no loss of data in the event of a system failure; and like Maximum Protection Mode can cause performance issues with the primary database if the connection between the primary and standby databases is not fast enough. The difference with Maximum Availability Mode is that in the event of loss of communication between the primary and standby systems, it will switch to Maximum Performance Mode until the problem is resolved. Once the link has been re-established, Data Guard will switch back into Maximum Availability Mode.
- **Maximum Performance Mode.** This mode provides the highest level of protection without compromising the performance of the primary database. This is done by allowing transactions on the primary to commit when the log record has been written into the primary redo log. The write to the standby redo log is done asynchronously. This provides better performance, but there is the chance of some loss of data if the primary cluster were to fail.

The mode that is chosen depends on your specific needs and configuration. Since both Maximum Protection Mode and Maximum Availability Mode require the standby redo log to be written to before a transaction is considered committed, there is a requirement for a fast network. For every millisecond of delay transmitting data from the primary to secondary sites, the commit operation will wait.

### *DATA GUARD NETWORK CONCERNS*

Since the network between the primary database and standby database(s) is critical for performance, it is important to understand what affects that performance. There are several factors that will degrade the performance between the two sites:

- **Distance.** It is often overlooked that distance actually matters. It is possible to make a telephone call across the country or around the world and it appears to be instantaneous (although Voice Over IP or VOIP can have a noticeable delay). However, since the electronic signal is physically limited to the speed of light (186,282 miles per second), a one way signal from NY City to LA would take approximately 14.5 ms (assuming 2700 miles). A signal from NY City to Denver would take approximately 9.1 ms (assuming 1700 miles) and a signal from NY to Philadelphia would take only .5 ms (assuming 89 miles). In order for the transmission to be completed, it must be transmitted, received, put into the standby redo log file and a return acknowledgement must be sent. Thus distance does really matter, and this isn't even taking into account network components.
- **Network Components.** Every firewall, router and other active network component will add delay to your signal. Thus the more components in between the primary and secondary data centers, the more latency will be introduced. So in

addition to the distance delay, add on significantly more time for network components. Thus it is usually impractical for Maximum Protection Mode to be used over long distances.

These points should all be taken into consideration when deciding how to architect your standby site. Typically a Maximum Protection Data Guard system will be set up locally, with a remote system using Maximum Performance Mode. This is a good compromise for both protection and performance.

Note: There are several ways that the network can be tuned. The network packet size can be tuned through Linux, the IP frame size can be increased with Jumbo Frames and the Oracle SDU size can be increased. Depending on your network, some or all of this might improve network performance. There is nothing that can be done to increase the speed of light.

## **OTHER IMPORTANT CONCERNS**

Other issues that must be considered regard staffing. In the event of a failover to the standby data center, there must be personnel there to manage those systems. In addition, there must be client systems so that DBAs, system administrators and staff can access the standby servers. There must be space to house the staff and workspaces for them to use. This should not be overlooked.

Another issue is the user community. Before designing a complex DR system consider who will be accessing this system. In the case of the internet business where users connect remotely from all parts of the world, the DR site is absolutely critical. If you are down for a long period of time customers will be lost and your business might collapse.

On the other hand, if your business is the New Orleans traffic division that takes internet payments for traffic violations, in the wake of hurricane Katrina it is unlikely that anybody would notice if you were up and running from a standby location in Nebraska, as paying a traffic violation would probably not be a priority at that time. Thus the need for the DR site depends somewhat on where your users are. In a widespread disaster, the user community might be out of commission for a long period of time as well.

## **MAA KEYS TO SUCCESS**

In order to successfully implement and maintain the Oracle Maximum Availability Architecture there are several key components. These tasks are planning, documentation and execution. If any one of these components are lacking, the MAA is at risk.

### **PLANNING**

It is important to thoroughly plan the MAA. Some of the questions that must be answered include:

- What is the end goal? What are the uptime requirements?
- What facilities are already available for a standby data center? Existing facilities are ideal if you have the right physical facilities and staff.
- Who's going to run the standby site? In the event of a disaster you will need local resources to maintain the standby site once it becomes primary.

The answers to these questions will help you determine how the MAA is to be architected. Once you have planned and architected the solution it should be properly documented.

### **DOCUMENTATION**

Good documentation is one of the keys to success. This documentation should contain information about the configuration of the system, how to access it, when and how to perform the switch to the secondary site, etc. The more information that is documented, the more successful the execution will be. Keep in mind that the people who designed and implemented the standby site might not be available when a switch over is required. Therefore, documentation should include detailed instructions on how to perform the switch over, contact information as to who are the decision makers, and how to make the decision to switch over in case those contacts are not available. In addition, troubleshooting tips should be provided as well. This documentation may be the only guidelines available to the staff at the remote data center. Make it as complete as possible. Also, make sure the documentation clearly labeled and is stored in a location(s) where it can be easily located.

### **EXECUTION**

The success of the execution of a fail over to the standby data center depends on several factors. These include the completeness of the documentation, the skills of the staff, and testing. The documentation should provide all of the

information necessary to decide if you need to activate the standby data center as well as instructions on how to accomplish this.

The standby data center should have available staff that is skilled in Oracle, RAC and Data Guard as well as system administrators and network administrators. This staff might have to run the entire operation for an extended period of time.

Finally, testing should be done on an annual basis where a practice fail over is done. This will not only allow the failover mechanisms to be tested, but will allow the documentation to be QA'd. By testing the failover all aspects can be practiced and documentation can be updated if errors are found. It is only through testing that the validity of the entire system can be assured.

## **SUMMARY**

High availability architecture and disaster recovery are an important part of any critical database and application. Because of the cost of creating a complete standby data center, Linux is an ideal platform. Linux is known for its performance and stability as well as its value. By using a more cost effective platform, it is possible to create a standby data center that more closely replicates the primary data center.

This paper has discussed many aspects of the Oracle Maximum Availability Architecture and different options on how to configure both High Availability and Disaster Recovery systems. As mentioned in this paper, switching over to the DR site is not an easy task, nor is it necessarily instantaneous. Thus HA is needed in order to withstand localized problems and allow for maximum uptime.

In order to survive a major disaster it is necessary to implement a completely redundant data center. This data center could be a complete copy of the primary data center, or it might be a scaled down data center with enough power to run at a degraded performance level until more equipment can be added or until the primary data center comes back on line. In any case, it is important to plan for the worse case and be prepared to maintain your business in the event of a disaster.

## **ABOUT THE AUTHOR**

Edward Whalen is CTO and founder of Performance Tuning Corporation ([www.perftuning.com](http://www.perftuning.com)). Performance Tuning Corporation (PTC) is a technology service company that designs, implements and optimizes database systems and applications for companies worldwide. PTC differentiates itself through its holistic approach to system analysis and performance tuning. Some of PTC services include: performance tuning, application workload testing, data migration, technical training, disaster recovery planning and implementation on Oracle and MS SQL Server. Edward Whalen is considered a leading expert and authority in performance tuning, high availability and disaster recovery.

Edward Whalen has written several books on both Oracle and MS SQL Server including:

- Oracle Performance Tuning and Optimization
- Teach Yourself Oracle8 in 21 Days
- Oracle Performance Tuning
- Oracle Database10g Linux Administration
- SQL Server 7 Administrator's Companion,
- SQL Server 7 Performance Tuning Technical Reference,
- SQL Server 2000 Administrator's Companion
- SQL Server 2000 Performance Tuning Technical Reference
- SQL Server 2005 Administrator's Companion