

XTIVIA

**Oracle DBA:
Best Practices**

*by Ron Kranz - Sr. Oracle Consultant
(OCP)*

July 2008

Introduction

CONTENTS

2	INTRODUCTION
3	ORACLE ENVIRONMENT
5	ORACLE BACKOVER FAILOVER & DISASTER RECOVERY
7	ORACLE SYSTEM SECURITY
9	ORACLE DESIGN
10	ORACLE CODING
13	ORACLE PERFORMANCE MANAGEMENT & TUNING
15	REFERENCES
15	FOR MORE INFORMATION

Defining Oracle best practices is a difficult process. The problem is that there are always exceptions to each best practice. Alas, each best practice has to include its own **"IF"** and **"WHEN"** clauses to be valid. Further, the versions of both the operating system and the Oracle software are vitally important to whether a best practice will work.

Under all considerations, this paper presents a list of the Oracle best practices applicable in most situations. For ease of implementation, these best practices are divided into several areas from your Oracle environment through the design and coding of your Oracle databases to performance management and tuning.

Oracle Environment

Optimal Flexible Architecture

With respect to the environmental aspects of Oracle, the first best practice is to use the **Optimal Flexible Architecture** (OFA) standards to layout the database. This provides for a logical as well as physical setup and should even be used under Stripe and Mirror Everything (SAME) setups.

- For SAME configurations, use the proper stripe width and depth or performance will suffer.
 - It must be a factor of **db_block_size*db_file_multiblock_read_count**.

The OFA principles breakdown simply into a series of recommendations for naming files and folders when installing and implementing an Oracle database in order to isolate REDO, rollback, temp, data and indexes files as much as possible. The other benefit from using OFA is that when you have to apply patches or upgrades, Oracle installer and other tools will be able to find the files where expected (inventory, etc.).

Set Points

A critical best practice for the Oracle environment is to verify proper operating system set points.

- Set **shared memory and semaphores** properly in UNIX and Linux environments:
 - Shared memory segments should be set to at least the desired **SGA size + 10%**, but not more than ½ total memory.
 - Set semaphores to **(n1 + n2 + n3...)+(2* nx)** where **n1..n3** are the semaphores for each instance on the box and **nx** is the largest value from any instance on the box.
 - For Unix and Linux, **set swap area size** to at least the size of the largest expected SGA or to a multiple of the size of physical memory.
- Set proper **background/foreground** settings and **virtual memory** in Windows and set Windows for **application serving** and not file serving.

File System Tunings

Another frequently overlooked best practice is to utilize File System Tunings. Some examples of this are:

- Use **Async IO**, where appropriate (but first, verify OS and file system supports this) and **turn off write ahead caching**.
- Tied in with this is the best practice of reducing buffering at the OS level. Generally speaking, you will achieve better performance by placing the buffer (for read operations) closer to the disk.

- With more and more shops looking at Linux on commodity-priced, Intel-based boxes, you also need to consider tuning the IDE interface; of course, this is only applicable if you use IDE drives for data or temporary areas.
 - On Linux, the **hdparm** command is used to both view and set the IDE parameters.
 - Default installations will have this parameter set inefficiently.

Multiple Channels

Given that disk controllers are also part of the environment, disk controllers should utilize multiple channels and use channel load balancing.

For example, use *Veritas Fastpath*.

Proper Stripe Width and Depth

The DBA should always review all pre-installed disk array configurations for proper **stripe width** and **depth**. For example, EMC has provided preconfigured arrays in the past that were set at an 8K stripe width; this size is not acceptable for some Oracle products, like applications.

- Set stripe width (amount of each stripe on each disk) to at least **db_block_size*db_file_multiblock_read_count**.
- Set stripe depth (number of disks in the stripe set) to at least **(expected maximum IO/sec)/90 for RAID1+0** or **(expected maximum IO/sec)/50 for RAID5**.

“Your DBA should always review all preconfigured systems.”

Network Interface Cards

The environment extends beyond disks and memory. The connectivity to the outside world comes from the network. You must utilize proper network configurations; this means, use the fastest Network Interface Cards (NICs) possible. In order to gain from fast NICs, you must also tune **TCP buffers** to attain max throughput.

- On Unix and Linux, this is accomplished through setting the proper buffers.
 - On Linux, setting the values for **/proc/sys/net/ipv4/tcp_wmem** and **/proc/sys/net/ipv4/tcp_rmem** can make a significant difference in performance.
 - A setting as high as 4 meg is possible.

The other side of enhancing connectivity is **tuning the Oracle packet sizes** through the use of **TDU** and **SDU** settings.

- These can be set as high as 32K and can provide instant performance increases for data-intensive operations across the network.

For Oracle RAC (which uses the **Unsecure Data Packet** protocol), tune the UDP buffers.

- These are set in a similar manner to the TCP buffers.

- You should also utilize the **CLUSTER_INTERCONNET** setting for multiple NIC card systems and RAC to ensure that the proper NIC is being utilized for the cluster interconnect.

Summary

Some *general best practices* for the Oracle environment are:

- ➔ Do not place *non-Oracle* applications on Oracle servers.
- ➔ Minimize placing multiple databases on a single server (unless the server has been properly sized and configured for multiple instances).
- ➔ Do not place applications with different maintenance windows and backup recovery requirements in the same database.

Oracle Backup, Failover, & Disaster Recovery

With the latest round of disasters, blackouts, and other site-wide failure occurrences, the concepts of backup, recovery, failover, and disaster recovery are not only a DBA's best practice, but also a business best practice.

- ➔ *Is your site protected?*
- ➔ *Could your customers get to your database or website in case of a wide-spread outage (multiple states)?*

“Many companies in New York utilize disaster sites here in Colorado to insure data availability and survivability.”

When we think of a disaster recovery site, most of us think in terms of a site that is a couple of miles away. It is a best practice to use an Oracle hot site (standby database, Data guard, etc.) to provide a Geo-mirrored site that is far away from your primary site. The hot site should be far enough away so that it does not get caught in site disaster (power, phone, network or physical access limitations) and it should fully mirror the regular site in terms of backup capability and bandwidth for all connections. Remember you may need to operate there for a while.

On this subject, we had a “shop” discussion about what would be the safest place for a global or near-global disaster. The general consensus is to have a data center suspended **100 feet below the ocean's surface** (one in the Atlantic and one in the Pacific). These sites could survive undamaged from just about anything that could happen to the earth's atmosphere or crust. They would have to be on a separate power supply and a dedicated communication link, possibly satellite. As far as we know, no one has established such a data center under the ocean's surface. Look for one in the near future.

RMAN

There are known exceptions (for example, data warehousing), but generally it is a best practice to use RMAN (Recovery Manager). RMAN encapsulates backup commands into RMAN syntax so that a script created on one site will most likely run on most others using RMAN.

“It is a best practice to use Recovery Manager.”

- Use RMAN for OLTP and small to medium sized databases, while other solutions may apply for large DBs or DWH.
- RMAN is an important component in HA systems using Data Guard, Fast Recovery Area, and RAC.

Mirroring

On sites where mirroring is used, use more than 2 mirrors. Multiple mirrors allow tools to break off the mirror for making backups, thus eliminating downtime. If possible, utilize 3 way mirrors so redundancy is not lost during the backup time frame.

Encrypt Backups

Another best practice is to encrypt backups of databases containing sensitive data. There are numerous tales of companies losing control of sensitive data due to improper disposal of backup media containing un-encrypted backups. If someone obtains an old backup, a simple restore operation can put your entire corporate data set at their fingertips.

RAC

You should consider it a best practice to use Oracle Real Application Clusters (RAC). RAC provides for instance redundancy preventing a downed box from bringing your database down. Combined with standby databases, it makes a truly high availability setup. RAC allows for use of low cost hardware in a redundant configuration.

- RAIS (redundant array of inexpensive servers) instead of RAID.

Offsite Backups

It is a best practice to store a verified copy of backup tapes offsite. A copy of all backups should be sent to an offsite facility for storage. In a disaster, the entire site and the surrounding area could be inaccessible. However, verified backup copies should be available offsite.

“It is a best practice to store a verified copy of backup tapes offsite.”

- This means that you should maintain at least two backup cycles of tapes or other media offsite.

Archive Logging

For databases that aren't easily rebuilt or, where data volumes don't prohibit it, use archive logging. For databases that need up-to-the moment recovery, use archive logging to allow point-in-time recovery. For databases that can be rebuilt (such as a data warehouse), archive logging may not be advisable.

Summary

- ➔ *No matter which option you choose, the best practice is to make the investment to protect your site from internal and external problems so that your customers can find you.*

Oracle System Security

In the current environment where we hear about hackers breaking in to systems, planting worms, virus and other malicious code just for the fun of it, security must take a high priority.

Security Patches

It is a best practice to implement security-related patches as soon as possible. Install all patches, including security-related patches in your test environment first!

Audit Security

You should also frequently audit security implementations. To perform a security audit, you should:

- Audit password, connectivity, and the use of third party-tools for access.
- Automate auditing whenever possible.
- Maintain audit logs that track when auditing was done, findings and corrective actions.
- Review security logs periodically for alerts.

Within Applications

A best practice is to use minimal privilege grants. This means that users should be granted database privileges and object grants that are required to do their job period. It is also considered a best practice to utilize Oracle Roles to group privileges, grants, and other roles.

- **Roles** provide the option of grouping application privileges into job-related groups, then as a person is assigned a specific task or tasks the roles associated with that set of tasks is granted to them, no more and no less.

At the Database Level

In this day and age, some might think of this as a given, but it is a best practice to encrypt important data within the database. Besides external threats, encryption of key data prevents internal snooping.

- For example, Oracle can encrypt single columns within a table and thereby protect SSANs, credit card numbers, patient identification, etc.

“It is a best practice to encrypt important data.”

It is also a best practice to implement password aging, expiry and degree of difficulty checking. Oracle has a **Profile** capability that incorporates into the Oracle database certain requirements on password aging, expiry, and difficulty verification. As a part of this password verification, a table of commonly used and prohibited passwords should be maintained and updated periodically. Also, as part of the best practice of using **Profile** options, Oracle can verify that users are not using the same password over and over again.

- You should implement the Profile options that lock out users that attempt to guess passwords and that generate alerts when this happens.

From the security point of view, it is a best practice to ensure that the latest patch levels are kept up-to-date in Oracle Net. Given that no software is perfect these days, by keeping software up-to-date, the latest security restrictions are put in place.

Even though Oracle provides these, it is considered a best practice to restrict the use of the **DBA**, **RESOURCE** and **CONNECT** roles and develop application specific roles instead. The reasons for not using these Oracle-supplied roles include:

- **DBA** opens the database and gives virtually unrestricted access.
 - If you use the DBA role, grant this only to DBAs.
- **RESOURCE** allows the insert of tables into any tablespace and grants many privileges that are only needed by high-level developers.
- **CONNECT** allows users to create tables and indexes. Most users only need to access objects that are already created.

It is a best practice to audit all third party applications to prevent use of **DBA**, **RESOURCE** and **CONNECT** roles. Many third party packages insist they need **DBA** or **RESOURCE** roles for their "application" users. This usually means these packages had no real DBA support during development and granting the DBA role was the easiest way to get around "all those grants."

- Don't perpetuate this miss guided "worst" practice. Insist on a real solution instead of a work-around. It's your data, your money, and your reputation that is at stake.

Summary

- ➔ *Implement security-related patches as soon as possible and have a plan to regularly audit security.*
- ➔ *Customize and utilize Oracle Roles based on job-related requirements of the users and implement Profile options to enhance the security of user access.*
- ➔ *Encrypt all important data.*

Oracle Design

If you are in the Oracle environment or any relational database environment, you know that it is a best practice to use *Dr. Codd's* normal forms (usually 3rd for OLTP and at least 2nd normal for data warehouses) for the design and layout of relational databases. Once you've established a "best" logical design, you can denormalize for performance in the physical design. The normal form diagrams, known most commonly as Entity Relationship Diagrams or ERDs, should be active, living documents that are maintained over the life of the database.

"ERDs should be active, living documents that are maintained over the life of the database."

Design Tools

A performance best practice is to pre-aggregate data with materialized views and/or summaries. Oracle automatically maintains properly created materialized views.

- One of the new features of **Oracle Database 11g** is the fully *automated caching* of result sets for frequently executed SQL.

An important design best practice, which is critical to any successful project, is to use **source control** either through a third-party tool, Oracle-provided source control, or manual logging methods. Proper use of a source control package eliminates last minute panic-builds that rob time and efficiency and will minimize reinvention of existing functionality.

Design Conventions

Another best practice for database design is the required use of *naming conventions*. Naming conventions should be implemented for variables, programs, procedures, packages, tables, columns, indexes, views and clusters to start with. Naming conventions prevent confusion and improve readability. Because sensible names are used consistently throughout, naming conventions also enhance the maintainability of code.

Hand-in-glove with naming conventions goes the use of a *standardized coding* convention. Proper coding standards allow multiple developers to maintain the same code sets without inducing complexity because of multiple (or no) standards being used. Coding standards allow for readable, maintainable code that can be understood long after the original programmers have left. An established standard will increase database performance by helping to avoid unnecessary parsing of the same code. Even an extra “white space” can cause the database parser to re-parse code already contained in the **SHARED POOL** buffer.

Another design best practice is to use the proper design technique for the environment (normal forms for OLTP, STAR for DWH and CUBE for DSS). The use of improper development methods will result in a poorly designed and inefficient database.

The final best design practice is probably the most overlooked and important one; each project should have a *carefully defined* scope, project plan, and deliverables. If there isn't a proper project plan, how will you know when you are finished? Without proper milestones, how will you measure project status? Finally, without deliverables, how will you know when a milestone is reached? A proper plan tells you what needs to happen and the order in which things need to be done. Be sure the part *you play* in the project is defined and logical before signing off on it. Most project managers think a buffer will make their shoes shiny and a server brings them lunch.

Summary

- ➔ *Use standardized naming and coding conventions as well as proper design technique to improve the design and efficiency of your database.*

Oracle Coding

It would take a book to completely delineate all SQL and PL/SQL, let alone discuss Oracle-related 3GLs and, indeed, several have been written for that purpose. However, in this section, a list of the “heavy-hitters” in best practices for Oracle coding will be discussed.

Array Processing

It is a best practice in PL/SQL to use array processing (for example: **bulk collect** and **forall**). Bulk processing minimizes **context switching** during PL/SQL processing to improve performance, sometimes by an order of magnitude or more.

- Context switching occurs any time PL/SQL executes a SQL statement, sometimes over and over again in a loop.

Develop Modular Units

It is also considered a best practice to place all code within stored procedures in *pinned* packages. By placing code in procedures, you can develop modular units. By placing procedures into packages, it allows grouping of related procedures and functions. When a single piece of a package is utilized, the entire package is brought into memory.

- By grouping related procedures and functions, you minimize disk access required to read in related procedures and functions as they are needed. This also allows for pinning of an entire application package into memory.
- Pinning of an application package prevents reloading and parsing of code, thus minimizing performance-robbing, code recursion.

Proper Variables and Loop Logic

Another best practice for PL/SQL (and SQL) is to use proper variable types for variable declaration. When improper variables are used, such as using a **VARCHAR2** when a **NUMBER** is needed or visa-versa, it results in index non-use and on-the-fly conversion. One way to ensure proper variable typing is to use **%TYPE** and **%ROWTYPE** to prevent improper variable types from being declared.

Another best coding practice is to always verify loop logic using **DBMS_PROFILER** or the *Quest® Code Tester* from *Quest Software*. **DBMS_PROFILER** is an Oracle-provided package that performs generation tracing for the number of times each line is executed and the amount of execution time required.

- Always verify that loops are executed the minimal number of times.

Also verify proper **IF-THEN-ELSE** structures. Place the *most utilized option first* (for example, the exit test). This best practice applies to **CASE** structures as well.

Anonymous PL/SQL, Temporary Tables, and NOCOPY

It is a best practice to utilize anonymous PL/SQL to ensure processing environments are similar when testing PL/SQL-SQL codes. If you replace bind-variables with literals to test in a standard SQL environment, you will not get the same execution plan as with use of anonymous PL/SQL and bind variables. Thus, this type of tuning will probably not produce best results.

- This is where in-situ testing will help to guarantee that needed results are realized.

It is a best programming practice to make proper use of temporary tables and PL/SQL index by tables. Improper use of "normal" tables for temporary storage is a performance robber.

- **Use global temporary tables** instead. Improper use of global temporary tables when the memory space is available to utilize PL/SQL tables can be a big performance hit.
- **Use views to create in-memory tables** for intermediate result sets.

It is a best practice to utilize **NOCOPY** on **IN OUT** and **OUT** variables. If the **NOCOPY** key word is not included in the variable declaration for the header of a PL/SQL procedure that uses **IN OUT** or **OUT variables**, all variables are passed by value using **COPYOUT** routines, instead of by reference. If you utilize **NOCOPY**, the values are passed by reference, which can make a significant difference in performance.

“This best practice is easy to implement.”

Tune All SQL First

One of the more important best practices in PL/SQL (and Java and C, and C++ and all other Oracle-related 3GL's) is to *tune all SQL first*. The most elegant, best thought-out, and constructed programs will perform only as well as the SQL contained within them. The following products from *Quest®* all have features that will assist in creating high performance SQL:

- **SQL Optimizer for Oracle**
- **Performance Analysis for Oracle**
- **Toad® for Data Analysis**
- **SQL Navigator®**

It is considered a best practice to require, at a minimum, that all developers of SQL code generate *explain plans*. All SQL should then be peer-reviewed for performance. In a large project, it can be a daunting task, if not impossible, for the DBA staff to review all code for the best performance. In most cases, DBA staff only stumbles upon troublesome code when it is identified as a bottleneck in production. Therefore, developers should take the lead in SQL code performance tuning. Once the SQL is tuned for index usage, proper logic, and proper technique, the DBAs can then perform more advanced tuning to problematic code.

- In Oracle releases 10g and up, the supplied package **DMBS_SQLTUNE** can be put to effective use in tuning SQL code.

Summary

- ➔ Developers should take the lead in SQL code performance tuning.
- ➔ As with any code, it is a best practice to *pre-tune all SQL statements*.

Oracle Performance Management & Tuning

Again, as with coding, tuning is an area where countless pages of material are written. In the next few paragraphs, I will attempt to distill this mass of material into a few best practices.

Database Monitoring

It is a best practice to perform proactive database monitoring. Proactive database monitoring means that you must, at a minimum:

- Monitor for space usage
- Monitor waits
- Monitor for execution times for standard queries/procedures
- Monitor for statements that use excessive disk IO
- Monitor for statements that use excessive memory IO
- Monitor for excessive recursive SQL
- Monitor latch and lock usage

“It is a best practice to perform proactive database monitoring.”

This type of **monitoring** can be accomplished using a variety of products (Embarcadero, Quest, Symantec, etc.) and services (Xtivia Virtual DBA) on the market or utilities supplied with the Oracle RDBMS (Statspack, ASH, AWR, and OEM).

Monitoring is of no value if no one takes the time to look at it and analyze it. However, watching a screen all the time is probably the most boring thing I can imagine. Therefore, it is a best practice to use an *automated alert* mechanism that uses thresholds to alert you by email or page before an issue becomes a larger problem that is noticed by your valued users. Your ability to create proactive monitoring functionality for all levels of alerts is the Achilles' heel of the database operations under your control. The general “not in the budget” argument against spending corporate dollars on monitoring software or services is nullified by the cost of a database that crashed and burned while giving off warnings and errors possibly hours or days before the final “database is unavailable” response.

- Backups are great to have, but recovery time is always more costly.
- Create a formula that equates downtime to dollars for your company.
- Then discuss the “acceptable” amount of database downtime with your user community. In most cases, that amount will equal zero.

Alert Mechanisms

An unavoidable best practice is to build in *alert mechanisms* for security-related issues, such as multiple unsuccessful login attempts, login attempts to secure accounts, password invalidations, and attempted accesses to restricted tables or restricted types of access to specific tables. It is also best to receive some sort of alert if unauthorized tools are used to access the database.

- Oracle's auditing and login triggers can be utilized for this.

RAID Levels

I don't know how many projects I have been on where the wrong type of disk RAID was in use. It is a best practice to *use the right RAID level* for the right datafile type. Oracle recommends:

- **RAID 1+0** whenever possible for data and indexes
- **RAID 0** for redo logs and archive logs.
- **RAID 5** only if it is not possible to use RAID 1+0.

“It is a best practice to use the right RAID level for the right type of datafile.”

Block Sizes, Buffer Pools, and Index Types

In *Oracle* databases 9i, 10g, and 11g, it is a best practice to use **multiple block sizes**; this allows you to tailor the block size to a specific type of access. Place tables and indexes in tablespaces sized (block size) according to access.

- For single block read type OLTP access, use 8k block sizes.
- For full table scan access, such as with data warehouses, use 16-32K block sizes.
- For index lookups use 8-16K block sizes.
- For indexes that are scanned or bitmap indexes, use 16-32K block sizes.

An important performance best practice is to use **multiple buffer pools** properly.

- Use default, KEEP and RECYCLE pools in Oracle releases 8 and up.
- Use KEEP for lookup tables and indexes (frequently accessed small objects.)
- Use RECYCLE for objects whose data will not be reused multiple times (blob or large full table scan tables.)

More on best performance practice is to use correct *index types*. Only use *reverse-key indexes* in Real Application Cluster environments where index header block contention is an issue (be careful, some third party tools create these by default).

- Do not use *bitmap* indexes for frequent insert, update and delete (IUD) activities on tables in any version.
- Do not use *B*tree* indexes for STAR schema facts in releases 9i, 10g, and 11g.
- Words of caution, Index Organized Tables (IOTs) are difficult to manage and rarely improve performance in any version.

Summary

- ➔ *As with PL/SQL or any code, it is a best practice to pre-tune all SQL statements. No matter how good your table and index layout is, if the SQL is not properly constructed to utilize it, your performance will suffer.*

References

The following resources have more useful information about some of the concepts discussed in this white paper:

- Oracle Database Documentation for 10g Release 2 – Oracle Corporation
- Oracle 10g: The Complete Reference – Kevin Loney – Oracle Press
- Expert Oracle Database 10g Administration – Sam R. Alapati – Apress (2005)
- Advanced Security Administrators Guide – Oracle 10g – Oracle Press (Nov 2005)

For More Information

To learn more about Best Practices and Oracle from Xtivia as well as other Xtivia products and services, please visit www.Xtivia.com or www.Virtual-DBA.com



© Copyright 2008 Xtivia, Inc. All rights reserved.

Xtivia, Inc.
304 South 8th Street
Suite 201
Colorado Springs, CO 80905
(888) 685-3101, Option 2
<http://www.xtivia.com>