Database Administration Maturity Model

Version 1.1

Table of Contents

Levels of the DBA MM (Database Administration Maturity Model)

This document discusses database administration, and outlines the tasks required to administer an Oracle database in both development and production environments.

There are five levels to the DBA CMM:

Level	Focus	Key Process Areas
1. Initial	None Key tool: Heroism	• successes are sporadic and not repeatable • work is charactarized by fire-fighting, emergencies, and heroism
2. Repeatable	focus on a repeatable DBA process Key tool: DBA Daily Checklist	• project planning (dba projects; dba support of development) • project tracking and oversight (dba projects) • subcontract management (dba outsourcing or subcontracting) • configuration management (database software, dba tools, change control)
3. Defined	focus on defining and documenting the DBA process Key tool: DBA training	• requirements management (service level) • quality assurance (dba tasks) • best practices are identified and propagated (training) • tasks are defined, and measurements of tasks become meaningful
4. Managed	process control: the DBA process is both stable and measured; variation is noted and cause identified Key tool: SLA	• changes to defined tasks are made in a controlled fashion; before-and-after measures show if the change helped
5. Optimizing	Continuous process improvement:	• improvements in management practices are sought out and refined • causes of variation are systematically controlled; "lessons learned" lead to better processes

You cannot *improve* what you cannot first *repeat*.

You cannot *measure* what you have not first *defined*.

You cannot *manage* what you do not first *measure*.

Differences Between Data Administrators, DBAs, and Data Architects

There are three distinct sets of tasks that a database needs to have performed upon it. These can be grouped into (1) Data Administration tasks, (2) Data Architect tasks, and (3) Database Administration (DBA) tasks. It's quite common for a single person to perform tasks in two or all three areas -- a "DBA" may take on architecture tasks during development -- but the task areas are distinct.

One crucial distinction involves the qualitative differences between the task areas. Another distinction is that these different tasks take place in different parts of the database's life cycle.

Qualitative Differences Between Task Areas

There are qualitative differences between the task areas.

Data Architecture

The first area, Data Architecture, is concerned with ***creating logical and physical models*** of data structures that may later be stored as database objects. These models exist in order to solve specific *business problems* that are within the *scope* of the overall application being created. The dependency order is this:

Application scope => business needs => logical models for data => physical models for data

The Data Architect has responsibility for such things as:

- creating models that adequately serve the needs of the business
- ensuring that the models stay within the scope of the application. (It is common to model a bit past the edges of the application scope, in order to understand how the application will interface with other systems. The data architect is responsible for keeping this under control and not modelling things that are too far out of scope.)
- creating a physical model that can provide acceptable performance to users. (The vast majority of performance improvements for an application come from changes to application code. Since the architect can make or break performance, it is up to the architect to be responsible for making it, not breaking it.)

Data Administration

The Data Administrator needs to deal with ***the political reality that information is power***, and the data in the database needs to belong to somebody. She gets to find out (or proclaim, if she's senior enough and has been given that power) *who owns which data elements*, and who gets to create, change, and delete what. If the database keeps a copy of the corporate master price list, then who is responsible for keeping that copy current -- the price list owner, who must then push changes in, or the application owner, who would have to poll for changes and pull them in? Do appropriate people have the right permissions to read the data they need? (I can't pull in changes from the master price list if I can't read that list.)

The Data Administrator needs to track *Data Ownership* for data elements, which is a task independent of the table design (Architect) and also independent of the details of placing data on particular disks (DBA).

The Data Administrator has responsibility for tracking, for each data element:

- who owns the data -- who may insert, update, or delete it
- who owns the data definition -- who is allowed to change that definition (i.e. a Part Number is an unpunctuated alphanumeric string with a length of exactly 9)
- for data that changes ownership during its lifetime, when and how ownership changes (i.e. the sales person owns an order until it is submitted; then shipping owns it until shipment is confirmed; then billing owns it until payment is received)

Database Administration

The Database Administrator deals with the ***physical details of keeping an actual database running*** (tasks such as placing data on particular disks, etc.).

The Database Administrator has responsibility for:

- maintaining database performance at or near the level demonstrated by the development team
- tuning, load balancing, patching, upgrading, backing up, recovering, etc.
- all aspects of the database other than the logical (application level) access to data

It is particularly worth noting that DBA tasks don't include writing application SQL. The task of writing application SQL statements belongs to developers. If a DBA takes on this task (which is not uncommon), he is performing a development task, not a DBA task.

(If this distinction seems odd, consider a sales manager who also has his own sales territory – a common arrangement in smaller sales organizations. He may have tasks involving supervising other sales people [performance reviews], and he may also have tasks involving selling products [making sales calls]. People may refer to him as a "manager", but when he performs a sales task, it is still a sales task, not a management task.)a

Database Life Cycle

A new database is born usually in support of a new application (or one sufficiently changed that the old database isn't being used). To create it appropriately, someone must analyze the application and must design appropriate tables (either with an entity-relationship model first, and mapping these to tables, or designing the tables directly). So this task, *Designing Tables* that support the needs of the application, is the first one performed, and happens prior to the birth of the database.

The task of Designing Tables falls under the umbrella of the Data Architect. Other Data Architect tasks include *Impact Analysis*: when the application requirements change, figuring out what table changes are needed.

Database Life Cycle Stage:	Architect	Data Administrator	DBA
1. Strategy, Analysis	Table Design, Security Design	Data Ownership, Security Policy	
2. Design, Build	Impact Analysis	Change Management	Installation, Datafile Creation, Table-to-Tablespace Mapping, Table Creation, Backup/Recovery Planning, Security Procedures
3. Youth			Physical Re-orgs, Performance Tuning
4. Maturity			Exception Monitoring, Performance Monitoring, Performance Tuning, Backup/Recovery Execution, Datafile Management
A. Minor Change	Impact Analysis	Change Management	Table Modification
B. Major Change	Impact Analysis, Table Re-design	Change Management	Table Creation

Whenever the application requirements change, the Data Administrator needs to perform *Change Management* to make sure that all the data is still owned by the right people, and that they know it.

Later in the project, after the tables have been designed, it's time to get the DBA going on *Table Creation*. That includes space planning, writing the storage parameters (based on volumetrics from the application analysts, which they should have provided to the Architect), and a first cut at distributing the data across disks for load balancing. The DBA should determine how many tablespaces to make, where and how many datafiles to make, and take responsibility for *Backup and Recovery Planning*. If the site is following the Optimal Flexible Architecture, then the DBA must implement an OFA-compliant layout of the datafiles.

Qualitative Differences

Some writers differentiate between Architects and DBAs solely on this criterion: Architects deal exclusively with logical database constructs (entities, relationships, table designs) while DBAs deal exclusively with physical issues (datafiles, disk load balancing, backup and recovery).

There is some truth to this distinction, but some danger as well. It's a very good idea for the person performing Architect tasks to have some knowledge of how databases are physically implemented, simply because it tends to lead to designs that are easier to build and maintain. In the automobile industry that's called "designing for manufacturability". Imagine that a designer is choosing between two options that function about equally well, but one of which happens to be much cheaper to make -- say it requires less re-tooling of the production line. If the designer doesn't know about the manufacturing issues, such as re-tooling and other costs, then she won't have the information to choose the option that's cheaper and easier to make.

I've been lucky enough to follow several of my own designs from initial idea all the way to implementation and full production, and it's taught me a lot about design trade-offs. There is no substitute for that experience.

The flip side to this issue is another danger, the danger of putting a DBA into a design role. I've seen several people try this, and they typically try to design things solely so they'll be easy to maintain, with insufficient emphasis on whether the design supports the functional needs of the application. These designs usually fail.

Here are some database-life-cycle tasks, by task area:

Area	Task	Frequency	Stages	Description
Data Architecture	Table Design	once	1, B	Design tables that meet the needs of the application. When these needs change, make appropriate design changes.
Data Architecture	Security Design	once	1	Make sure the database design supports anything (special columns, etc.) needed by the application's security needs.
Data Architecture	Impact Analysis	as needed	2, A, B	Whenever an application change is proposed, determine what aspects of the database design must change to support it. (These changes in turn may require changes to some existing code, if any has been written; that impact must also be analyzed by appropriate staff.)
Data Admin	Data Ownership	once	1	Determine who "owns" data and is responsible for its maintenance and accuracy.
Data Admin	Security Policy	once	1	Determine who is authorized to see data, and who may reveal what data to others and when.
Data Admin	Change Management	as needed	2, A, B	Whenever the application or its data change, make sure all data still has clear ownership, and make sure people whose responsibilities have changed are notified.
DBA	Installation	once	2	Install the RDBMS of choice.
DBA	Datafile Creation	once	2	Create appropriate datafiles for the desired tablespaces, using the available disks in such a way as to minimize disk contention. Use the OFA if desired. Requires definition of appropriate Tablespaces.
DBA	Backup/ Recovery Plan	once	2	Create a backup and recovery plan for the database. Document the plan. Test the plan. If resources are not available, escalate to management.
DBA	Security Procedures	once	2	Create roles, grant data permissions and system privileges, create users, issue passwords, etc. Initiate auditing if desired.
DBA	Table-to-Tablespace Mapping	once	2	Based on the OFA or other guidelines, determine which tables will be stored in which tablespaces. Document these decisions.
DBA	Table Creation	once	2, B	Create tables in appropriate tablespaces.

Area	Task	Frequency	Stages	Description
DBA	Physical Re-orgs	as needed	3	Re-arrange tables within tablespaces, create and change indexes, create and change clusters, define horizontal partitioning transparent to the application, etc.
DBA	Performance Tuning	as needed	3, 4	Initially, look for serious problems with the physical layout, missing indexes, etc. Once the database is mature, use the Performance Monitoring task to drive any needed performance tuning.
DBA	Exception Monitoring	daily	2, 3,4	Examine log files and trace files for evidence of problems; monitor e-mail and voice mail for user reports of problems
DBA	Performance Monitoring	weekly*	4	Check the performance of the database against defined metrics and against the database Service Level Agreement.
DBA	Datafile Management	as needed	4	Arrange datafiles (using OFA or a similar discipline) on disks as needed. Create new datafiles to expand tablespaces.
DBA	Backup/ Recovery Execution	daily*	4	Perform backup activities (online backups, offline backups, archive logs) regularly. When recover is needed, execute the appropriate recovery plan.
DBA	Table Modification	as needed	A	When minor changes in the application require that tables be changed, the DBA must make those changes while preserving the data already in that table.

*may vary significantly

Once we start to understand the qualitative differences in these tasks, we can start doing a better job of matching people with appropriate skills to the right job and the right set of tasks, and managing people to those tasks. That will lead to better software, better databases, and more successful projects.

Database Life Cycle Tasks

a reference list

This part lists each task, and describes each in detail. It may be used as a reference for understanding each task.

Installation

Task Name: installation

Frequency: once

Install the RDBMS of choice. Verify that installation succeeded.

Datafile Creation

Task Name: datafile creation

Frequency: once

Create appropriate datafiles for the desired tablespaces, using the available disks in such a way as to minimize disk contention. Use the Optimal Flexible Architecture (OFA) if desired. Requires definition of appropriate Tablespaces.

Backup/Recovery Planning

Task Name: backup/recovery planning

Frequency: once

Backup and Recovery Planning is vital to keeping the database up and available for users, for keeping the data in the database safe, and for making sure that any unavoidable interruptions in service are as brief and harmless as possible.

Steps in task:

- Create a backup and recovery plan for the database.
- Document the backup and recovery plan.
- Test the backup and recovery plan.
- Have the backup and recovery plan reviewed by an outside expert.
- Match the backup and recovery plan against the Service Level Agreement (SLA).

- If resources are not available to execute the plan in a fashion that matches the SLA, escalate to management.

The backup and recovery plan is included in this document as an appendix.

Security Procedures

Task Name: security procedures

Frequency: once

There are at least three levels of security that are relevant to The database: the Unix username and password, the Oracle username and password, and Lawson's application-specific security features. (The network username and password are not directly relevant to the database, but they do form a fourth level of security.)

The DBA must create any Oracle users that need to exist, and assign initial passwords. The DBA must also manage any 'roles' that the application needs.

Steps in task:

- determine what levels of security need to be implemented
- implement those levels of security
- maintain users and roles as needed
- verify security measures

Table-to-Tablespace Mapping

Task Name: table-to-tablespace mapping

Frequency: once

As part of managing the physical storage of data, the DBA needs to specify the tablespace in which each table (and each index) will exist. In some installations, differences in tables (large vs. small, active vs. static, accounting vs. distribution, etc.) dictate which tablespace each will go in. The DBA then manages the physical storage of the datafiles that make up each tablespace, knowing the behavior of the tables inside that tablespace.

Table Creation

Task Name: table creation

Frequency: once

Once the developers and data architects have decided what tables need to exist, it is a DBA task to actually create the table. This involves determining the initial and projected size of the table, whether it needs to be clustered, how it should be indexed, and assumes that table-to-tablespace mapping has already been done.

Physical Re-orgs

Task Name: physical re-orgs

Frequency: as needed

A general rule of thumb for database-related tasks is that anything involving the physical storage of the database is a task for the DBA. There is no task more physical than the ‘physical re-org’. This includes a number of particulars:

- moving a table from one tablespace to another
- creating needed indexes
- creating clusters
- defining horizontal partitioning (if transparent to the application)

⇒ compressing extents of a table with backup/export/drop/import

⇒ compressing extents of an index with drop/re-create

The following is taken from the Oracle7 Server Concepts Manual:

Data Blocks

At the finest level of granularity, an ORACLE database's data is stored in data blocks (also called logical blocks, ORACLE blocks, or pages). One data block corresponds to a specific number of bytes of physical database space on disk. A data block size is specifically set for every ORACLE database when the database is created. This data block size is a multiple of the operating system's block size within the maximum limit. What is important to remember is that a database, at its finest level of granularity, uses and allocates free database space in ORACLE data blocks.

(In contrast, all data at the physical, operating system level is stored in bytes. Each operating system has what is called a block size. An operating system block size is a specific number of bytes on disk.)

Extents

The next level of logical database space is called an extent. An extent is a specific number of contiguous data blocks that is allocated for storing a specific type of information.

Segments

The level of logical database storage above an extent is called a segment. A segment is a set of extents that have been allocated for a specific type of data structure, and that all are stored in the same tablespace. For example, each table's data is stored in its own data segment, while each index's data is stored in its own index segment.

Performance Tuning

Task Name: performance tuning

Frequency: as needed

Database performance tuning is a potentially unending task. Proper tuning can take a great deal of time and involves an enormous number of variables. Improper tuning can make performance worse. An undisciplined approach to tuning can have unpredictable results.

Properly, however, database tuning should only be attempted in support of a pre-existing agreement for the database to perform at a certain level. This in turn assumes that performance can be measured, reliably and accurately.

Only attempt to tune a database when these conditions are true: (1) you have an SLA in place, (2) you have reliable performance metrics in place, and (3) the database is not performing to the level specified in the SLA.

There are two acceptable outcomes to a tuning effort. First, the database may be brought up to the level required by the SLA. Second, tuning may not be possible or feasible, and the SLA may then be re-negotiated to allow the current level of performance.

This document will not cover the details of how to execute performance tuning.

Exception Monitoring

Task Name: exception monitoring

Frequency: daily

Exception monitoring is one of the most important ongoing tasks for a DBA. This task gives the DBA his or her earliest warning of actual problems.

The DBA needs to monitor, at a minimum, these things:

- user reports of unusual database behavior, error messages, etc. (the smart DBA will log these)
- unusual entries in the alert logs and trace files

The DBA should modify this manual if she finds more items that should be monitored.

Performance Monitoring

Task Name: performance monitoring

Frequency: daily or weekly

Performance monitoring is one of the most important ongoing tasks for a DBA. This task gives the DBA her eyes and ears for anticipating database problems and heading them off before they affect users.

The DBA needs to monitor, at a minimum, these things:

- fragmentation of free space in each tablespace

- total free space available in each tablespace
- the growth rate of the database (i.e. the rate at which free space is being consumed)
- from the above items, calculate how long it will likely be until the database runs out of free space in any tablespace
- the number of extents in each table and index, for any with more than a threshold amount, usually 4 or 5
- response times for specific reports (queries) and transactions (inserts, updates, deletes)
- unusual entries in the alert logs and trace files

The DBA should modify this manual if she finds more items that should be monitored.

Datafile Management

Task Name: datafile management

Frequency: as needed

In response to projected growth, a DBA may need to add free space to a tablespace by adding a new datafile to it. In response to performance tuning needs, a DBA may need to re-arrange the physical locations of datafiles on the available disks.

Backup/Recovery Execution

Task Name: backup/recovery execution

Frequency: daily

The DBA is responsible for the physical well-being of the database's data files. That includes backing them up regularly, and restoring them from backup when needed.

Since The DBA is running Oracle on a set of disks using RAID-5, it is extremely unlikely that the Oracle datafiles will ever be damaged, so the likelihood of needing to restore from backup is quite remote.

Every day, shut down the database and back up its datafiles (and control files) to tape. At that time, purge any archived log files from the archive log destination.

Table Modification

Task Name: table modification

Frequency: as needed

Part II: The Database Life Cycle and DBA Tasks

when to perform which tasks

This part of the manual describes when to perform which tasks. It assumes that the database is in the 'mature' phase of its life cycle.

Performance Monitoring

Performance monitoring must be driven by the Service Level Agreement and by regular feedback from users.

On a regular basis, usually daily when a system is new, weekly once it is stable, monitor these things. When one of them turns up something, here is what to do.

- Fragmentation of free space in each tablespace:
    - drawing on Loney's work (DBA Handbook, pages 243-254) pick a metric for measuring free space fragmentation (Loney uses an arbitrary Free Space Fragmentation Index) and use it;
    - pick a threshold for the FSFI or equivalent, such as .80, and if the FSFI number is below that, then evaluate the free space (pp. 248-250) and continue below;
    - if honeycomb fragmentation is present then run a "glue" utility to re-unite the fragments (Oracle version 7.0 and later is supposed to do this 'coalescing' operation automatically, but you should monitor it to be sure);
    - if "hole" fragmentation is high and free space is low, ignore the holes and add a datafile;
    - if "hole" fragmentation is high and free space is low and disk space is low, perform a backup/export/drop/import to make tables contiguous and free space contiguous.
- Free space in each tablespace:
    - evaluate free space by the growth rate, below.
- Rate of free space usage in each tablespace (compared to the last measure)

  - using a tool like Eventus' Adhawk Spacer, keep careful track of how free space is changing in each tablespace each day. Don't be misled by big changes in TEMP. Make sure rollback segments have an OPTIMAL parameter set in their Storage Clause.

- Projection of how long until all free space is used up

  - a tool like Spacer will automate this, but you can also do it manually: take the amount of free space and divide it by the average amount of free space used up each day, and you should end up with the number of days you have left before you run out of space and the database stops working.

  - calculate for each tablespace separately, and pay special attention to PROD1_DATA and PROD1_IDX. Be cautious of readings on TEMP and RBS since they can fluctuate wildly but tend not to grow steadily over time.

  - decide in advance how many days worth of slack you want to allow yourself. At some point before you run out of space you will want to add a datafile to the tablespace that needs more free space. See 'Datafile Management'.

- Identities of all objects with more than four (4) extents (these become targets for Export/Import to compress the extents -- see 'Physical Re-orgs as needed')

  - set regular times for compressing extents -- Friday afternoons, say -- and compress the extents of all objects found through monitoring.

  - if the same object gets compressed all the time, you may want to increase the size of its 'Next' extent in its Storage parameters.

  - if your monitoring shows that an object will reach its maximum number of extents before the time you would normally compress, then act immediately by both increasing the size of 'Next' and by compressing. Once an object reaches the maximum number of extents, it cannot grow, and *that may prevent users from performing their normal work.*

- Response times for certain representative 'test' queries and updates

  - it is hard, but vital, that you pick good test queries and updates to be representatives of overall database performance. This is so important, in fact, that it is included in the SLA. The DBA should not pick these himself, but should work with the users to pick them. This should be part of the SLA negotiation.

  - once these test transactions are chosen, monitor them daily (at first, weekly if all seems stable) to establish a measure of how the database is working.

  - if users complain about database performance, immediately run those test queries or updates that are most similar to the one causing problems, and see if the problem looks real.

  - if the problem proves real but the test queries didn't reveal them, then revise the tests.

- Document response times against the SLA

  - keep a log of every time you measure the response time of the database. This will allow you to track its performance over time.

- Examine appropriate log and trace files daily

  - identify the log files and optional trace files that you want to monitor

  - each day, examine them for unusual activity, error messages, warnings, etc.

- Keep a log of unusual events found in the log and trace files

  - if you find anything unusual in any of the log files, record it in the DBA log of unusual events.

  - use this log to help you decide if you need to take action to head off potential future problems.

Backup/Recovery Planning

- Document a backup plan that allows recovery that meets the Service Level Agreement
- Have an external specialist review the backup plan and confirm that it supports the SLA
- Document a recovery plan, with specific, detailed steps, for each major type of database crash
- Test each recovery plan on a test instance
- If the SLA calls for a service level that cannot be met with the available resources, escalate the issue and either re-negotiate the SLA or obtain the needed resources

Backup/Recovery Execution

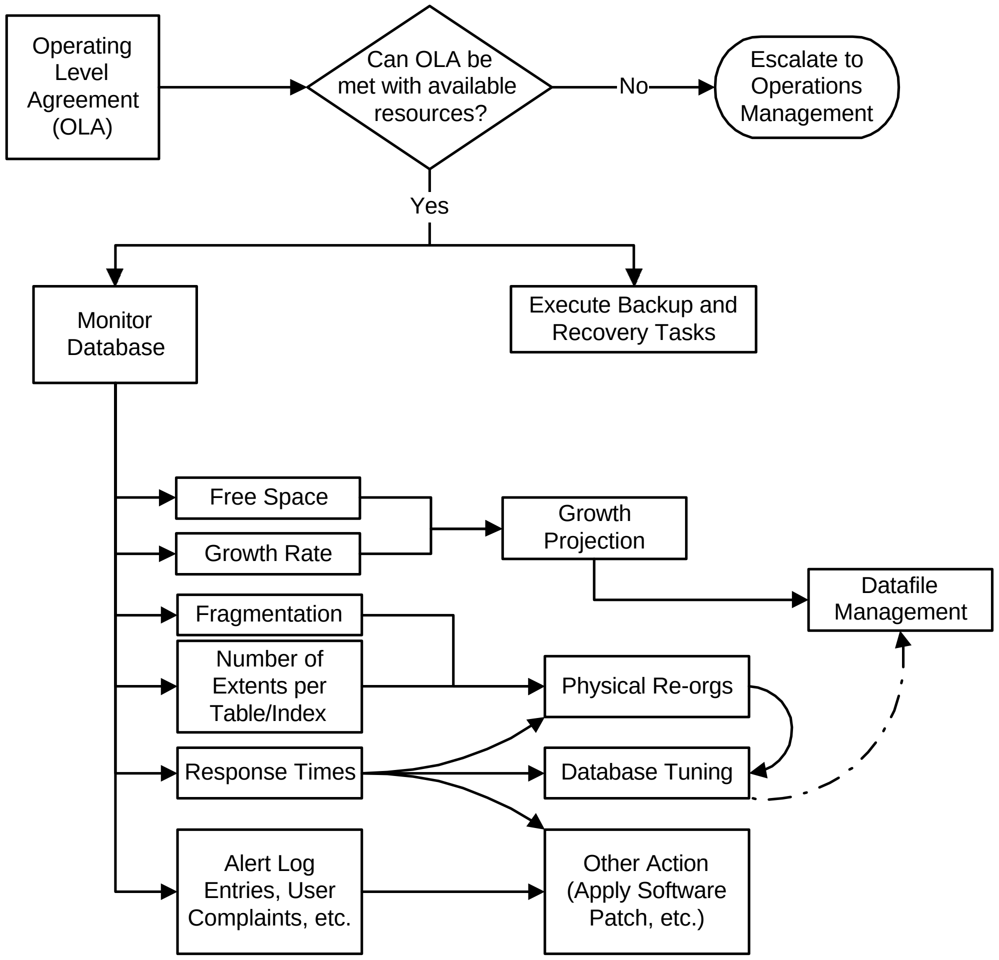
- Execute all steps in the backup plan
- Document each step each time it is done (via a log book, etc.)

Service Level Agreement

- The SLA should describe acceptable levels of database service in these areas:
  - ◊ Scheduled and unscheduled downtime in hours per day, days per week and days per year
  - ◊ Response times for representative queries, reports, and transactions
  - ◊ Maximum recovery time and data loss from a crash
- The details of the SLA will dictate many of the specifics in the daily and weekly DBA tasks. For example:
  - ◊ "Max Unscheduled Down Time Per Year" determines how aggressively the DBA needs to monitor the database and how much time she must spend reviewing bug lists, etc.
  - ◊ "Max data loss" will affect the log file size and the checkpoint interval

The database SLA is included in this document as an appendix.

Select DBA Task Dependencies (mature database)


Operating Level Agreement (OLA)
Can OLA be met with available resources?
No
Escalate to Operations Management
Yes
Monitor Database
Execute Backup and Recovery Tasks
Free Space
Growth Rate
Growth Projection
Datafile Management
Fragmentation
Number of Extents per Table/Index
Physical Re-orgs
Response Times
Database Tuning
Alert Log Entries, User Complaints, etc.
Other Action (Apply Software Patch, etc.)

Appendix A: Draft of Database Service Level Agreement for <Company>

Service	Agreed Level
System up time per day	24 hours
System hours available	24 hours a day, 7 days a week
Frequency of hot backups	daily, plus archiving redo logs
Frequency of cold backups	four times a year
Max data loss per crash	No more than one day's data
Time to come up from a soft crash (power loss, etc.)	Time of power restoration plus 15 minutes
Time to come up from a hard crash (disk failure, etc.) damaging no data	One hour to switch hardware and power up, plus 15 minutes as above: 1 hour 15 minutes total
Time to come up from a hard disk crash that damages a drive containing database data	One hour to switch hardware and power up, plus 15 minutes to start database, plus up to 3 hours to restore hot backup and roll forward: 4 hours 15 minutes total
Down days per year	12 days per year: one day a month, preferably the second Saturday of each month
Response time for bug report	To be negotiated
Response time for enhancement request	To be negotiated
Response time for request for new application	To be negotiated
Query response time	To be negotiated (requires definition of benchmark queries)
Total hours of scheduled down time per year	12 hours
Scheduled down time:	3 hours on each of Labor Day, Thanksgiving Day, Christmas Day, New Year's Day
Projected hours of unscheduled down time per year	4 hours per year

Bibliography

1. Millsap, Cary V. (1994). *The OFA Standard, Oracle7 for Open Systems*.
2. Shallahamer, Craig A. (1994). *Avoiding A Database Reorganization*.
3. Millsap, Cary V. (1995) *Oracle7 Server Space Management* Revision 1.4b (95/10/31) An Oracle Services Advanced Technologies Research Paper
4. Millsap, Cary V. (1996) *Designing your System to Meet Your Requirements*
5. Cox, Thomas B. (1998) *The Low Administration Oracle Specification, Part I* v 1.5