Oracle Database 10g Release 2
Automatic Storage Management
Overview and Technical Best Practices

Table of Contents

Introduction

In Oracle Database 10g Release 2, storage management and provisioning for the database has been greatly simplified using *Automatic Storage Management* (*ASM*). ASM provides filesystem and volume manager capabilities built into the Oracle database kernel. With this capability, ASM simplifies storage management tasks, such as creating/laying out databases and diskspace management. Since ASM allows disk management to be done using familiar create/alter/drop SQL statements, DBAs do not need to learn a new skill set or make crucial decisions on provisioning. An Enterprise Manager interface as well as a new command line utility (new in Oracle Database 10g Release 2) is also available for those ASM administrators who are not familiar with SQL.

ASM is a management tool specifically built to simplify the job of the DBA. It provides a simple storage management interface across all server and storage platforms. ASM provides the DBA flexibility to manage a dynamic database environment with increased efficiency. This feature is a key component of Grid Computing and Database Storage Consolidation.

The following are some key benefits of ASM:

- I/O is spread evenly across all available disk drives to prevent hot spots and maximize performance.
- ASM eliminates the need for over provisioning and maximizes storage resource utilization facilitating database consolidation.
- Inherent large file support.
- Performs automatic online redistribution after the incremental addition or removal of storage capacity.
- Maintains redundant copies of data to provide high availability, or leverage 3rd party RAID functionality.
- Supports Oracle Database 10g as well as Oracle Real Application Clusters (RAC).
- Capable of leveraging 3rd party multipathing technologies.
- For simplicity and easier migration to ASM, an Oracle Database 10g Release 2 database can contain ASM and non-ASM files. Any new files can be created as ASM files whilst existing files can also be migrated to ASM.
- RMAN commands enable non-ASM managed files to be relocated to an ASM disk group.
- Oracle Database 10g Enterprise Manager can be used to manage ASM disk and file management activities.
- ASM reduces Oracle Database 10g cost and complexity without compromising performance or availability.

This document will discuss the essentials of ASM, and cover the steps to discover disks, create a diskgroup, and eventually create a database within an ASM diskgroup, discussing some best practices along the way.

The following components of ASM will be discussed:

- Instances – ASM instance and database instance
- ASM Disks
- Diskgroups
- Failure Groups
- Rebalancing Capabilities
- ASM Files, Templates and Aliases
- ASM and Cluster Synchronization Services (CSS)

Overview of Oracle Database 10g Release 2 ASM New Features

Automatic Storage Management in Oracle Database 10g Release 2 expands the capabilities of ASM and provides features and enhancements in the following categories:

- Database consolidation and clustering
  - Database storage consolidation with Single Instance and RAC
  - Multiple database version support
  - Independent ASM Oracle home
- Easy database migration to ASM
  - Enterprise Manger (EM) ASM migration utility
- Management flexibility and simplicity
  - Command Line Interface (CLI)
  - ASM file system view through XMLDB virtual folders
  - Enhanced ASM manageability and alerts
  - Enhancements in DBMS_FILE_TRANSFER utility

This paper will introduce the new features of Automatic Storage Management (ASM) in Oracle Database 10g Release 2, and describe the benefits and improvements in simplification and automation for managing Oracle database files.

See the following document for a high level overview of all the new Oracle Database 10g Release 2 ASM Features:
ASM 10gR2 New Features – http://asm.us.oracle.com/pdf/ASM R2 New Features.pdf

ASM Instances

In Oracle Database 10g there are two types of instances: database and ASM instances. The ASM instance, which is generally named +ASM[1], is started with the INSTANCE_TYPE=ASM init.ora parameter. This parameter, when set, signals the Oracle initialization routine to start an ASM instance and not a standard database instance. Unlike the standard database instance, the ASM instance contains no physical files; such as logfiles, controlfiles or datafiles, and only requires a few init.ora parameters for startup.

Upon startup, an ASM instance will spawn all the basic background processes, plus some new ones that are specific to the operation of ASM. The STARTUP clauses for ASM instances are similar to those for database instances. For example, RESTRICT prevents database instances from connecting to this ASM instance. NOMOUNT starts up an ASM instance without mounting any disk group. MOUNT option simply mounts all defined diskgroups[2].

The illustration in Figure 1 shows the basic init.ora parameters required to start ASM. Observe that all ASM processes begin with *asm,* as opposed to the database instance, whose processes begin with *ora*.

ASM is the volume manager for all databases that employ ASM on a given node. Therefore, only one ASM instance is required per node regardless of the number of database instances on the node. Additionally, ASM seamlessly works with the RAC architecture to support clustered storage environments. In RAC environments, there will be one ASM instance per clustered node, and the ASM instances communicate with each other on a peer-to-peer basis using the interconnect.

Figure 1.

```
Instance type
SQL> select instance_name from v$instance

INSTANCE_NAME
----------------
+ASM

ASM init.ora parameters
*.background_dump_dest='/opt/app/admin/+ASM/bdump'
*.core_dump_dest='/opt/app/admin/+ASM/cdump'
*.instance_type=asm
*.asm_diskgroups=+DATA
*.large_pool_size=12M
*.asm_diskstring='/dev/rdsk/c3t19d*s4'
*.remote_login_passwordfile='SHARED'
*.user_dump_dest='/opt/app/admin/+ASM/udump'
```

Note that most database parameters are not allowed (or applicable) to be set in ASM instance. If you set an invalid parameter in the ASM instance, you will get an ORA-15021 error. For example, if you try to set the control_files parameter for the ASM instance, you'll receive the following error message:
ORA-15021: parameter "control_files" is not valid in asm instance

[1] For RAC configurations, the ASM SID is +ASMx instance, where x represents the instance number.
[2] OPEN startup option performs the same function as MOUNT option; i.e., mount all asm_diskgroups.

ASM background processes

```
oracle    2423     1  0 Apr30 ?        00:00:00 asm_pmon_+ASM1
oracle    2425     1  0 Apr30 ?        00:00:00 asm_diag_+ASM1
oracle    2434     1  0 Apr30 ?        00:00:00 asm_mman_+ASM1
oracle    2436     1  0 Apr30 ?        00:00:00 asm_dbw0_+ASM1
oracle    2438     1  0 Apr30 ?        00:00:00 asm_lgwr_+ASM1
oracle    2440     1  0 Apr30 ?        00:00:00 asm_ckpt_+ASM1
oracle    2442     1  0 Apr30 ?        00:00:00 asm_smon_+ASM1
oracle    2444     1  0 Apr30 ?        00:16:03 asm_rbal_+ASM1
```

ASM installation

In cases where a single ASM instance is managing only one database instance, it may be sufficient to maintain a single ORACLE_HOME for ASM and the database.
However, for systems that have ASM instance managing the storage for several database instances and require higher availability, it is recommended that the ASM instance be installed in a separate ORACLE_HOME (ASM_HOME) than the database ORACLE_HOME[3]. If installing ASM in a separate ORACLE_HOME, then the listener should be run from that ASM_HOME.

In Oracle Database 10g Release 2, Oracle Universal Installer (OUI) and Database Configuration Assistant (DBCA) have been enhanced to allow the user to seamlessly create and install an ASM instance in a separate ORACLE_HOME. OUI now has options to:

- Install and configure a database that uses ASM for storage management
- Install and configure an ASM instance, without creating a database.
- Install and configure ASM on a system that already has a running database, where subsequently, the DBA can use the EM Migration Utility to migrate the database to ASM.

Additionally, in Oracle Database 10g Release 2, ASM transparently supports both older and newer software versions of the database. See *ASM – Database Multi-Version Support* section for more details. This new feature provides higher availability as well as the foundation for the Database Storage Consolidation Feature. For more information on Database Storage Clustering see the following section: *Database consolidation and clustering*

Upgrading the ASM instance
There are no special procedures for upgrading ASM instance from 10.1 to 10.2. From an ASM point of view, upgrading is just a matter of upgrading the software; i.e.; there are no persistent changes to disk groups as part of upgrade. Simply, install 10.2 as a separate ORACLE_HOME, then all that DBUA should need to do is move the configuration files from the old ORACLE_HOME to the new ORACLE_HOME and make any updates to those files that are ORACLE_HOME specific.

The upgrade of the ASM instance from 10.1 to 10.2 can be accomplished through DBUA. First all the databases that use the ASM instance being upgraded must be shutdown.
1) Start DBUA (Database Upgrade Assistant) from 10.2 ORACLE_HOME/bin
2) DBUA has two options
   a) upgrade ASM instance
   b) upgrade database
3) Select "upgrade ASM" to upgrade the ASM instance.

[3] If a separate ASM_HOME is created with a different user id , then the oracle user for each of the DB instances needs to be a member of the dba group for the ASM instance; the ASM oracle user does not need to be in the dba group for the DB instances.

4) Oracle Cluster Synchronization Services(CSS) needs to be upgraded before upgrading ASM instance or Database instance. When prompted by DBUA please run the following script :
`/ORACLE_HOME/bin/localconfig reset` to upgrade CSS.

5). Once step 4 is completed DBUA will a display summary screen of operations to be performed

There is no support for "rolling upgrade" in 10gR2 ASM, thus if installing a patchset or "non-rolling upgrade safe" patches, then all the ASM instances in RAC cluster must be shutdown. Currently in 10.2, rolling upgrades is only supported w/ CRS installs and between Dataguard sites.

Additionally, if using a separate ASM_HOME then ASM should be patched accordingly, just like the RDBMS home. This includes applying Critical Patch Updates (CPU) as well.

ASM SGA and parameter sizing

Enabling the ASM instance requires configuring only a handful of init.ora parameters. The parameter file for ASM can be a Pfile or Spfile. DBCA will create an ASM Pfile by default during database/ASM creation. The Spfile can be manually configured later if desired. However, if an Spfile is used in clustered ASM environments, then it must on a shared raw device. The init.ora parameters specified in Figure 1 are the essential parameters required to start up ASM[4].
The following describes the usage of SGA components.

- db_cache_size - This value determines the size of the cache. This buffer cache area is used to cache metadata blocks.

- shared_pool - Used for standard memory usage (control structures, etc.) to manage the instance. Also used to store extent maps.

- large_pool - Used for large allocations.

The recommended SGA parameter sizes for ASM are as follows:

- shared_pool_size = 128M
- large_pool = 12M
- db_cache_size = 64M

The "processes" init.ora parameter for ASM may need to be modified. The following recommendation pertains to versions 10.1.0.3 and later of Oracle, and will work for RAC and non-RAC systems. The formula below can used to determine an optimal value for this parameter:

```
Processes = 25 + (10 + [max number of concurrent database file
creations, and file extend operations possible])*n
```

Where n is the number of databases connecting to ASM (ASM clients).

The source of concurrent file creations can be any of the following:

- Several concurrent create tablespace commands
- Creation of a Partitioned table with several tablespaces creations
- RMAN backup channels
- Concurrent archive logfile creations

[4] ASM currently does not employ the Oracle Database 10g Automatic Memory Management feature. However, since ASM instances have static memory needs, this is not an issue.

ASM and privileges

Access to the ASM instance is comparable to a standard instance; i.e., SYSDBA and SYSOPER. Note however, since there is no data dictionary, authentication is done from an Operating System level and/or an Oracle password file. Typically, the SYSDBA privilege is granted through the use of an operating system group. On Unix, this is typically the dba group. By default, members of the dba group have SYSDBA privileges on all instances on the node, including the ASM instance[5]. Users who connect to the ASM instance with the SYSDBA privilege have complete administrative access to all disk groups in the system. The SYSOPER privilege is supported in ASM instances and limits the set of allowable SQL commands to the minimum required for basic operation of an already-configured system. The following commands are available to SYSOPER users:
•STARTUP/SHUTDOWN
•ALTER DISKGROUP MOUNT/DISMOUNT
•ALTER DISKGROUP ONLINE/OFFLINE DISK
•ALTER DISKGROUP REBALANCE
•ALTER DISKGROUP CHECK
•Access to all V$ASM_* views

All other commands, such as CREATE DISKGROUP, ADD/DROP/RESIZE DISK, and so on, require the SYSDBA privilege and are not allowed with the SYSOPER privilege.

ASMLIB

Oracle has developed a storage management interface called the ASMLIB API. ASMLIB is not required to run ASM; it is simply an add-on module that simplifies the management and discovery of ASM disks. The ASMLIB provides an alternative, to the standard operating system interface, for ASM to identify and access block devices. The ASMLIB API provides storage and operating system vendors the opportunity to supply extended storage-related features and exploit the capabilities and strengths of vendors' storage array.

The ASMLIB API provides two major feature enhancements over standard interfaces:

- Disk discovery – Providing more information about the storage attributes to the Database and the DBA
- I/O processing – To enable more efficient I/O

Oracle is providing an ASMLIB, called the Oracle ASMLIB, for the Linux platform (current version as of this writing is 2.0). This implementation of ASMLIB makes it easier to provision storage for a clustered pool of storage. Providing a kernel level driver access to disk means that permissions and device mapping are more stable upon the reboot of a server.
ASMLIB also makes the discovery process simpler to manage and removes the challenge of having disks added to one node and not be known to other nodes in the cluster.

The Oracle ASMLIB for Linux is available and can be down loaded from OTN. See the following URL for more information on ASMLib configuration and support matrix.
http://www.oracle.com/technology/tech/linux/asmlib/index.html

[5] On Windows systems, the sqlnet.ora must specify NTS.

Disks

The first task in building the ASM infrastructure is to discover and associate (adding) disks under ASM management. This step is best done with the coordination of Storage and Systems administrators. The Storage administrator will identify a set of disks and create the appropriate LUNs from the storage array and finally present them to the host. The term *disk* may be used in loose terms. A disk can be partition of a physical spindle, the entire spindle or a LUN in a RAID group set (defined in the storage array), this depends on how the storage array presents the logical unit number (LUN) to the Operating System (OS). In this document we will refer generically to LUNs or disks presented to the OS as simply, *disks.*

There are some restrictions on disk sizes that can be added to an ASM diskgroup. The maximum disk size that can be included as an ASM disk is 2^32 MB and the minimum disk size is 4Mb

On Solaris systems, disks will generally have the following SCSI name format: *CwTxDySz*. Where the C is controller number, T is the target, D is the LUN/disk number, and S is the partition. Note that each OS will have its unique representation of SCSI disk naming.

For example, it is a best practice on a Solaris system, to create a partition on the disk; such as slice 4 or 6, which skips the first 1Mb into the disk. Creating partition serves several purposes. A partition creates a placeholder to identify that the disk is being used. An un-partitioned disk could be accidentally misused or overwritten. Skipping 1Mb into the disk is done to skip the OS label/VTOC, as well as to preserve alignment between ASM stripping and storage array internal stripping.

Keep in mind that you'll never get perfect alignment between ASM and the underlying storage LUN layout. Anything other than perfect 1 MB alignment will not achieve that goal. However this is not an issue if you are not using a storage array that stripes a LUN across multiple disks with stripes that are a power of 2. In any case, if your fdisk tool does not allow you to do this easily, then don't worry.

Different Operating Systems will have varying requirements for the OS label; i.e., some may require an OS label before it is used while others will not. The same principles apply for different Operating systems although the procedures may be different.

In SAN environments, it assumed that the disks are identified and configured correctly; i.e., they are appropriately zoned or LUN masked within the SAN fabric and can be seen by the OS. Once the disks are identified, they will need to be discovered by ASM. This requires that the disk devices (Unix filenames) have their ownership changed from root to oracle[6].

Throughout this document a running example consisting of diskgroup DATA will be used to procedurally outline the steps for creating a diskgroup and other ASM objects.

In our example, disks c3t19d5s4, c3t19d16s4, c3t19d17s4, c3t19d18s4 are identified, and their ownership set to the correct "*oracle:dba*" ownership.

Now, these disks can be defined and discovered via the init.ora parameter, asm_diskstring. In our example, we will use the following wildcard setting:

```
*.asm_diskstring='/dev/rdsk/c3t19d*s4'.
```

ASM will only scan for disks that match that ASM search string. There are two forms of ASM disk discovery; shallow and deep. Shallow discovery is simply ASM scanning (essentially an 'ls-l' listing) of all the devices that are eligible to opened; i.e; these disk devices have the appropriate permissions. Deep discovery is physical open of those eligible disk devices.

[6] Although logical volumes can be presented to ASM as disks, it is generally not a good practice, since an extra layer is added to IO code path, and thus some level of overhead.

A key point to mention with regards to ASM in a RAC environment, is that ASM does not require that all disks have consistent names across servers. For example, an ASM disk can be /dev/rdsk/c3t19d1s4 on server1 and the same disk could be seen as /dev/rdsk/c7t22d1s4 on server2. ASM will discover the disks as long as the discovery string is set appropriately (on each server). This is all possible because ASM does not persistently store OS path names in "on-disk" headers, it simply stores ASM disk names.

In Figure 2 below, notice that the NAME column is empty and the group_number is set to 0. Disks that are discovered, but not yet associated with a diskgroup have a null name, and a group number of 0.

Figure 2.

```
SQL> select name, path, group_number from v$asm_disk

NAME                           PATH                GROUP_NUMBER
------------------------------ ----------------    ------------
                               /dev/rdsk/c3t19d5s4                0
                               /dev/rdsk/c3t19d16s4               0
                               /dev/rdsk/c3t19d17s4               0
                               /dev/rdsk/c3t19d18s4               0
```

Disks have various header statuses that reflect its membership state with a diskgroup. Disks can have the following header statuses:

- Former - This state declares that the disk was formerly part of a diskgroup
- Candidate - When a disk is in this state, it indicates that it is available to be added to a diskgroup.
- Member - This state indicates that a disk is already part of a diskgroup.
- Provisioned - This state is similar to candidate, in that its available to diskgroups. However, the provisioned state indicates that this disk has been configured or made available using ASMLIB.

ASM and Multipathing

An I/O path generally consists of an initiator port, fabric port, target port, and LUN. Each permutation of this I/O path is considered an independent path. Dynamic Multi-pathing/failover tools aggregate these independent paths into a single logical path. This path abstraction provides I/O load balancing across the host bus adapters (HBA), as well as non-disruptive failovers on I/O path failures. Multi-pathing (MP) software requires all the required disks to be visible on each available and eligible HBA. A MP driver will detect multi-paths by performing a SCSI inquiry command[7].

Multi-pathing software also provides multi-path software drivers. Most multi-pathing drivers support multipath services for fibre channel attached SCSI-3 devices. These drivers receive naming and transport services from one or more physical HBA devices. To support multi-pathing, a physical HBA driver must comply with the multi-pathing services provided by this driver. Multipathing tools provides the following benefits:

- Provide a single block device interface for a multi-pathed LUN
- Detect any component failures in the I/O path; e.g., fabric port, channel adapter, or HBA.
- When a loss of path occurs, ensure that I/Os are re-routed to the available paths, with no process disruption.
- Reconfigure the multipaths automatically when events occur.
- Ensure that failed paths get revalidated as soon as possible and provide auto-failback capabilities.
- Configure the multi-paths to maximize performance using various load balancing methods; e.g., round robin, least I/Os queued, or least service time.

[7] Probes for the World Wide Device Identifiers (WWD ID).

When a given disk has several paths defined, each one will be presented as a unique path name at the OS level; e.g.; /dev/rdsk/c3t19d1s4 and /dev/rdsk/c7t22d1s4 could be pointing to same disk device. ASM, however, can only tolerate the discovery of one unique device path per disk. For example, if the asm_diskstring is '/dev/rdsk/*', then several paths to the same device will be discovered, and ASM will produce an error message stating this. When using a multipath driver, which sits above this SCSI-block layer, the driver will generally produce a pseudo device that virtualizes the sub-paths. For example, in the case of EMC's PowerPath, you can use the following asm_diskstring setting of '/dev/rdsk/emcpower*'. When I/O is issued to this disk device, the multipath driver will intercept it and provide the necessary load balancing to the underlying subpaths.

Examples of multi-pathing software include EMC PowerPath, Veritas DMP, Sun Traffic Manager, Hitachi HDLM, and IBM SDDPCM. Linux 2.6 has a kernel based multipathing driver called Device Mapper. Additionally, some HBA vendors also provide multipathing solutions.
Oracle Corporation does not certify or qualify these multipathing tools. Although ASM does not provide multi-pathing capabilities, ASM does leverage multi-pathing tools, as long the path or device produced by the Multi-pathing tool returns a successful return code from an fstat system call. Metalink Note 294869.1 provides more details on ASM and Multipathing.

DiskGroups

Once the disks are discovered, a diskgroup can be created that will encapsulate one or more of these disks. A diskgroup, which is the highest-level data structure in ASM, is comparable to a LVM's volume group. However, there are several differentiators between typical LVM volume groups and ASM diskgroups:

- An ASM filesystem layer is implicitly created within a diskgroup. This filesystem is transparent to users and only accessible through ASM, interfacing databases, and 10.2 ASM command line tool.

- There are inherent automatic file-level striping and mirroring capabilities. A database file created within an ASM diskgroup will have its file extents (not to be confused with its database extents) distributed equally across all online disks in the diskgroup, which provides an even IO load.

The creation of a diskgroup involves the validation of the disks to be added. These disks must have the following attributes:

- Cannot already be in use by another diskgroup
- Must not have a pre-existing ASM header
- Cannot have an Oracle file header (from a Oracle raw device datafile).

This check and validation prevents ASM from destroying any in use data device. Disks with a valid header status, which include candidate, former, or provisioned, are the only ones allowed to be diskgroup members[8].

The disk group is created using SQL commands. See Figures 4a, 4b and 5 below.
Figure 4a.

```
SQL> create diskgroup DATA external redundancy disk
'/dev/rdsk/c3t19d5s4',
'/dev/rdsk/c3t19d16s4',
'/dev/rdsk/c3t19d17s4',
'/dev/rdsk/c3t19d18s4';
```

[8] Disks with a status of *member,* require the use force option when adding to a diskgroup. Use the force option with extreme caution.

The output below, from V$ASM_DISGROUP, shows the newly created diskgroup.
Figure 4b.

```
SQL> select name, state, type, total_mb, free_mb from v$asm_diskgroup;

NAME                           STATE       TYPE     TOTAL_MB    FREE_MB
------------------------------ ----------- ------ ---------- ----------
DATA                           MOUNTED     EXTERN      34512      34101
```

In the example above, a DATA diskgroup is created using four disks, which reside in a storage array, with the redundancy being handled externally via the storage array.
Figure5 shows how the V$ASM_DISK view reflects the disk state change after being incorporated into the diskgroup.
Figure 5.

```
SQL> select name, path, mode_status, state, disk_number from v$asm_disk

NAME         PATH                    MODE_ST STATE    DISK_NUMBER
------------ ----------------------- ------- -------- -----------
DATA_0000    /dev/rdsk/c3t19d5s4     ONLINE  NORMAL             0
DATA_0001    /dev/rdsk/c3t19d16s4    ONLINE  NORMAL             1
DATA_0002    /dev/rdsk/c3t19d17s4    ONLINE  NORMAL             2
DATA_0003    /dev/rdsk/c3t19d18s4    ONLINE  NORMAL             3
```

ASM will generate and assign a disk name with a sequence number to each disk added to the diskgroup[9]. The Disk Name is used when performing any disk management activities. The ASM disk name is different from the SCSI address. This allows for consistent naming across RAC nodes as well as protects from SCSI address name changes due to array re-configurations. In 10.2, disk names are now unique only within diskgroups, whereas, in 10.1 disk names were unique to an ASM instance (or clustered ASM instances in RAC).

After the diskgroup is successfully created, metadata information, which includes creation date, diskgroup name, and redundancy type, is stored in the SGA and on each disk (in disk header) within the DATA diskgroup. The V$ASM_DISK view will now reflect this disk header information. Once these disks are under ASM management, all subsequent mounts of the diskgroup will cause ASM to re-read and validate the ASM disk headers.

When mounting diskgroups, either at ASM startup or for subsequent mounts, it is advisable to mount all required diskgroups at once. This is done to minimize the overhead of multiple ASM disk discovery scans.

When a diskgroup becomes mounted, ASM registers the diskgroup name, the instance name, and the corresponding Oracle Home path name with Cluster Synchronization Services (CSS). This registered data is then used by the database instance to build the TNS connect string. This string is subsequently used by the database instance to connect into the ASM instance for volume management activities. This is discussed in the Database Instance Section.

The ASM instance also houses I/O statistics in the V$ASM_DISK and V$ASM_DISK_STAT views. This view includes reads/writes processed, read/write blocks handled, and read/write errors incurred. An ASM based *iostat* like utility, called asmiostat, can be used to show I/O statistics for ASM based disks. Appendix F contains the code for this utility. This script can be copied into a file and executed against any 10.2 ASM instance.

In 10.1, querying V$ASM_DISK and V$ASM_DISKGROUP was an expensive operation because each execution involved a disk discovery. To minimize overhead, and allow lightweight access to this dataset,

[9] A generic disk name is generated only if the disk name is not provided in the 'create diskgroup' statement.

10.2 introduces two new views: V$ASM_DISK_STAT and V$ASM_DISKGROUP_STAT. These two views are identical to V$ASM_DISK and V$ASM_DISKGROUP, however, V$ASM_DISK_STAT and V$ASM_DISKGROUP_STAT views are polled from memory and therefore do not require disk discovery. Since these new views provide efficient lightweight access, EM can periodically query performance statistics at disk level and aggregate space usage statistics at diskgroup level, without incurring significant overhead. Note, to get accurate real-time statistics, it maybe be prudent to query the V$ASM_DISK and V$ASM_DISKGROUP views, but, caution should be exercised when running queries against these views during peak workloads.

Diskgroups and databases

A diskgroup can contain files from many different 10g Oracle databases, and these databases can be either from the same server or can reside on different servers. However, a disk and a database file can only be part of one diskgroup. Alternatively, one Oracle database may also store its files in multiple disk groups managed by the same ASM instance. Allowing multiple databases to share a diskgroup provides greater potential for improved disk utilization and greater overall throughput.

To reduce the complexity of managing ASM and its diskgroups, Oracle recommends that generally no more than two diskgroups be maintained and managed per RAC cluster or single ASM instance[10].

- Database area: This is where active database files such as datafiles, control files, online redo logs, and change tracking files used in incremental backups are stored
- Flash recovery area: Where recovery-related files are created, such as multiplexed copies of the current control file and online redo logs, archived redo logs, backup sets, and flashback log files

To provide higher availability for the database, when a Flash Recovery Area is chosen at database creation time, an active copy of the controlfile and one member set of the redo log group is stored in the Flash Recovery Area. Note, additional copies of the controlfile or extra logfiles can be created and placed in either diskgroup as desired. Please refer to the following documentation for a detailed discussion on the usage and best practices of the Flash Recovery Area:

http://otn.oracle.com/deploy/availability/pdf/40104_Bednar_doc.pdf

[10] Additional diskgroups may be added to support tiered storage classes in Information Lifecycle Management (ILM) or Hierarchical Storage Management (HSM) deployments

ASM redundancy and Failure Groups

For systems that do not use external redundancy, ASM provides its own internal redundancy mechanism and additional high availability by way of *failure groups*. A failure group, which is a subset of a diskgroup, by definition is a collection of disks that can become unavailable due to a failure of one of its associated components; e.g., controllers or entire arrays. Thus, disks in two separate failure groups (for a given diskgroup) must not share a common failure component. If you define failure groups for your disk group, ASM can tolerate the simultaneous failure of multiple disks in a single failure group

ASM uses a unique mirroring algorithm. ASM does not mirror disks; rather, it mirrors extents. As a result, to provide continued protection in event of failure, only spare capacity in your disk group is needed rather than having to provision a hot spare disk. It is not advisable to create failure groups of different sizes, as this will create problems when allocating secondary extents. Primary extents on a given disk will have their respective mirror extents on one of several partner disks in the other fail group. ASM ensures that a primary extent and its mirror copy never reside in the same failure group.

Redundancy for disk groups can be either *Normal redundancy* (the default); where files are two-way mirrored (requiring at least two failure groups), or *High redundancy*, which provides a higher degree of protection using three-way mirroring (requiring at least three failure groups). Once a disk group has been created its redundancy level cannot be changed. To change the redundancy of a diskgroup, another diskgroup must be created with the appropriate redundancy desired, then move the datafiles (using RMAN restore or DBMS_FILE_TRANSFER) to this newly created diskgroup.

Figure 6 shows an example of creating a disk group using failure groups. In this example, ASM normal redundancy is being deployed over a storage array that has four internal trays, with each tray having four disks. The failing component to isolate is the storage tray, thus the failure group boundary is the storage tray; i.e., each storage tray will be associated with a failure group. Figure 6.

```
SQL> create diskgroup DATA_NRML normal redundancy

Failure group flgrp1 disk
‘/dev/rdsk/c3t19d3s4’,‘/dev/rdsk/c3t19d4s4’,‘/dev/rdsk/c3t19d5s4’,
‘/dev/rdsk/c3t19d6s4’

Failure group flgrp2 disk
‘/dev/rdsk/c4t20d3s4’,‘/dev/rdsk/c4t20d4s4’,‘/dev/rdsk/c4t20d5s4’,
‘/dev/rdsk/c4t19ds4’

Failure group flgrp3 disk
/dev/rdsk/c5t21d3s4’,‘/dev/rdsk/c5t21d4s4’,‘/dev/rdsk/c5t21d5s4’,
‘/dev/rdsk/c5t21ds4’

Failure group flgrp4 disk
/dev/rdsk/c6t22d3s4’,‘/dev/rdsk/c6t22d4s4’,‘/dev/rdsk/c6t22d5s4’,
‘/dev/rdsk/c6t22ds4’;
```

Diskgroups created with normal or high redundancy contain files that are double or triple mirrored; respectively[11]. The first file extent allocated is chosen as primary extent, and the mirrored extent is called the secondary extent. In the case of high redundancy there will two secondary extents. This logical grouping of primary and secondary extents is called an extent set. An extent set always contains the exact same data, since they are mirrored versions of each other. Thus when a block is written to a file, each extent in the extent set is written in parallel. However, when a block is read from disk, it is always read from the primary extent, unless the primary extent cannot be read. Keep in mind that each disk in a

[11] Diskgroup metadata is always triple mirrored with normal; or high redundancy.

diskgroup (and failure groups) contains nearly the same number of primary and secondary extents. This provides an even distribution of read IO activity across all the disks. This is different than most logical volume managers, which have a primary and mirrored disk set.

As stated above, failure groups are used to isolate component failures, and thus failing components needs to be understood. If this failing component cannot be determined or the failing component is the disk itself (as opposed to the array controller or storage tray), then it may be advisable to not specify any failure groups when defining normal or high redundancy diskgroups. This results in every ASM disk being in its own failure group, with ASM internally determining the disk mirroring partnership.
If the storage array has fully redundant components with dual paths to the disk, it is best to not to specify failure groups and let ASM manage the disk mirroring partnerships.

Using the example in Figure 6, in the event of a disk failure in failure group flgrp1, which will induce a rebalance, the contents (data extents) of the failed disk are reconstructed using the redundant copies of the extents from the partnered disk. This partnered disk can be either from failure group flgrp2 or flgrp3. Let's say that the partnered disk is c5t21d3s4 from failure group 3. During the rebalance, if the database instance needs to access an extent whose primary extent was on the failed disk, then the database will read the mirror copy from the appropriate disk in failure group flgrp3. Once the rebalance is complete, and the disk contents fully reconstructed, the database instance will return to reading primary copies only.

Since disk drives are mechanical devices, they have a tendency to fail. As drives begin to fail or have sporadic I/O errors, the probability for database corruptions becomes more likely. ASM takes proactive measures with regards to I/O errors. This is done irrespective of using Failure Groups. A permanent I/O error is only signaled to the caller (Oracle IO process) after several retries in the device driver. If a permanent disk IO error is incurred during an Oracle write operation, then the affected disk is removed from the diskgroup by ASM, thus preventing more application failures[12]. If the loss of a disk results in data loss, ASM will automatically dismount the diskgroup to protect the integrity of the diskgroup data.

New Space related columns in Oracle Database 10g Release 2

There are two new columns introduced in Oracle Database 10g Release 2 that provide more accurate information on free space usage.

- USABLE_FREE_SPACE - In 10.1, the FREE_MB value that is reported in V$ASM_DISKGROUP does not take into account mirroring[13]. 10.2, introduces a new column in V$ASM_DISKGROUP called USABLE_FREE_SPACE to indicate the amount of free space that can be "safely" utilized taking mirroring into account. The column provides a more accurate view of usable space in the diskgroup.

- REQUIRED_MB_FREE - Along with *usable_free_space*, a new column has been added to V$ASM_DISKGROUP to more accurately indicate the amount of space that is required to be available in a given diskgroup in order to restore redundancy after one or more disk failures. The amount of space displayed in this column takes mirroring into account.

[12] Disks are removed from the diskgroup only on write operation I/O error and not for read operations. For read errors, the block is read from the secondary extents (only for normal or high redundancy) or from external redundancy mirror pair (this is performed by the storage array).
[13] Note for external redundancy, column FREE_MB is equal to USABLE_FREE_SPACE.

Cluster Synchronization Services – CSS

ASM was designed to work with single instance as well as with RAC 10g clusters. ASM, even in single-instance, requires that Cluster Synchronization Services (CSS) is installed and started before ASM becomes available. In a single instance, CSS maintains synchronization between the ASM and database instances. CSS, which is a component of Oracle's Cluster Ready Services (CRS), is automatically installed on every node that runs Oracle Database 10g ASM and starts up automatically on server boot-up. In RAC 10g environments, the full Oracle Cluster-ware (CRS) is installed on every RAC node.

Since CSS provides cluster management and node monitor management, it inherently monitors ASM and its shared storage components (disks and diskgroups). Upon startup, ASM will register itself and all diskgroups it has mounted, with CSS. This allows CSS across all RAC nodes to keep diskgroup metadata in-sync. Any new diskgroups that are created are also dynamically registered and broadcasted to other nodes in the cluster.

As with the database, internode communication is used to synchronize activities in ASM instances. CSS is used to heartbeat the health of the ASM instances. ASM internode messages are initiated by structural changes that require synchronization; e.g. adding a disk. Thus, ASM uses the same integrated lock management infrastructure that is used by the database for efficient synchronization.

If CSS does not start automatically on reboot or if there is a hostname configuration change. The following command can be used to reconfigure CSS on that host. Note, this command needs to run from the ORACLE_HOME where CSS is running from.

```
1. $ORACLE_HOME/bin/localconfig delete
2. $ORACLE_HOME/bin/localconfig add
```

Database Instances

A Database instance is the standard Oracle instance, and is the client of the ASM instance. The database to ASM communication is always intra-node; i.e., the database will not contact the remote (in case of RAC) ASM instance for servicing database requests.

After the ASM disk group is created, DBCA can now be used to create the database[14]. DBCA has three options for database file structures: filesystem, raw, or ASM. See the ASM Installation section for an overview of the 10.2 new features of DBCA.

If ASM is selected, then all available diskgroups, if already created for that ASM instance, will be listed. Selecting ASM and a diskgroup will invoke DBCA to create a database within ASM. If no diskgroups exist or a new diskgroup is desired, then DBCA offers the opportunity to create a new diskgroup. Note, an ASM diskgroup can house all the Oracle physical files; ranging from controlfiles, datafiles, spfiles and RMAN backup files. However, Oracle executables, CRS files (OCR and voting disks) and non-database files cannot be housed in ASM diskgroups. Throughout this document all Oracle physical files will be generically referred to as *database files*.

An active database instance which uses ASM storage, can then operate just like a typical database instance; i.e., all file access is directly performed without extensive ASM intervention. Database instances interact with the ASM instance when files are created, deleted or opened. At this time the file layout is read from the ASM instance and all subsequent I/O's are done using the extent map stored in the database instance. ASM and DB instances also interact if the storage configuration changes; e.g., when disks are added, dropped, or fail.

Although the ASM layer is transparent to the database clients and users on the server, all datafile access can only be done via the database instance and its utilities. For example, database backups of ASM based files can only be performed with RMAN. Note, utilities like the Unix *dd* command are not recommended for backing up or restoring ASM diskgroups.

The database file level access (read/write) of ASM files is similar to pre-10g, except that any database file name that begins with a *"+"*, will automatically be handled and managed using the ASM code path. However, with ASM files, the database file access inherently has the characteristics of raw devices; i.e., un-buffered (direct IO) with kernelized asynchronous IO (KAIO).

The database structure is generally transparent to ASM; thus concepts like database extent allocation, automatic segment space management (ASSM), or usage of database such as objects CLOBS or BLOBS is inherently supported. The only exception to this rule is external tables. External tables are ASCII files and currently not supported for storage in ASM.

Database consolidation and clustering

In Oracle Database 10g Release 1, RAC and single instance databases could not be managed by the same ASM instance. This created challenges in implementing storage Grid architectures and consolidated database solutions. Oracle Database 10g Release 2 enhances the ASM functionality in a clustered environment allowing one ASM instance per node to manage all database instances in the cluster. Therefore, an ASM instance on a given node can now manage storage for single instance or RAC database instances. This feature relieves the customer from maintaining more than one ASM instance needed to serve the different database types that might exist in the cluster, thus obviating the need for DBAs to manage separate storage pools.

[14] DBCA can also be used to create the ASM instance and disk groups

This new feature leverages Oracle Clusterware to economically consolidate multiple islands of databases into a single clustered pool of storage managed by ASM. This essentially allows customers to optimize their storage utilization by eliminating wasted over-provisioned storage and save money by reducing their overall footprint of database storage.

Once the database is created and the instance is active, the database instance will become a client of ASM. This is reflected in the `V$ASM_CLIENT` view[15]. `V$ASM_CLIENT` contains one row for every ASM diskgroup that is opened by a database instance [16]. In Figure 7 below, V$ASM_CLIENT displays two databases connected to ASM, with each instance using two diskgroups. Note, instance *cubs* is a 10.2 RAC enabled database and brew is a 10.1.0.4 single instance.

Figure 7.

```
SQL> select INSTANCE,NAME,STATUS,SOFTWARE_VERSION,COMPATIBLE_VERSION from
v$asm_client;

INSTANCE STATUS       SOFTWARE_VRSN COMPATIBLE_VRSN

------- ------------ ------------ ------------

cubs1   CONNECTED    10.2.0.1.0   10.2.0.1.0

cubs1   CONNECTED    10.2.0.1.0   10.2.0.1.0

brew    CONNECTED    10.1.0.3.0   10.1.0.2.0

brew    CONNECTED    10.1.0.3.0   10.1.0.2.0
```

ASM – Database Multi-Version Support

In Oracle Database 10g Release 2, ASM transparently supports both older and newer software versions of the database. Both forward and backward compatibility is also maintained between all Oracle Database 10g Release 1 and Release 2. Therefore, any combination of 10.1.u.v and 10.2.x.y for either ASM or database instances interoperate seamlessly without administrative intervention. The only requirement is that a database connecting to a 10.2 ASM instance must be at version 10.1.0.3 or later. Additionally, for RAC environments, CRS must be at the highest level.

Allowing mixed versions of the ASM and database instances, allows for relaxed migration strategies and higher availability. For example, a database upgrade will not force an upgrade of the ASM instance.

When mixing software versions, ASM functionality reverts to the functionality of the earliest version in use. For example, a 10.1.0.3 database instance working with a 10.2.0.0 ASM instance does not exploit new features in ASM 10.2.0.0. Conversely, a 10.2 database instance working with a 10.1.0.3 ASM instance, will not exploit any of the new 10.2 features of ASM; e.g., XDB Folders. V$ASM_CLIENT has two new columns that gives you information about the software version number and corresponding database compatibility level. Note V$ASM_CLIENT exists on both ASM and database instance, therefore the content of the view will be in respect to the instance from where its queried.

[15] This view on database instance contains rows if the database has open files in ASM (CONNECTED state) or has accessibility to any diskgroup (MOUNTED state).

The SOFTWARE_VERSION column gives the version number of the database or ASM instance for the selected diskgroup connection. The COMPATIBLE_VERSION column gives the compatible setting of the database or ASM instance for the selected diskgroups connection. Figure 7 above shows a 10.1.0.4 database connected to 10.2 ASM instance.

Database processes to support ASM

In a database instance there are three sets of processes added to support the ASM diskgroups and infrastructure.

- RBAL – This process performs global opens of all the disks in the disk groups
- ASMB – This process contacts CSS using the diskgroup name, and acquires the associated ASM connect string. This connect string is then used to connect into ASM instance[17]. Using this persistent connection, periodic messages are exchanged to update statistics and provide a heartbeat mechanism. During operations that require ASM intervention, such as a file creation by a database foreground, the database foreground connects directly to the ASM instance to perform the operation. Upon successful completion of file creation, database file extent maps are sent by ASM to ASMB. Additionally, ASMB also sends database IO statistics to ASM instance.
- O00x – A group of slave processes establish connections to the ASM instance, where x is a number from 1 to 10. Through this connection pool, database processes can send messages to the ASM instance. For example opening a file sends the open request to the ASM instance via a slave. However slaves are not used for long running operations such as creating a file. The slave (pool) connections eliminate the overhead of logging into the ASM instance for short requests. These slaves are shutdown when not in use.

[17] This connection is always a bequeath connection into ASM instance.

Database Init.ora parameters to support ASM

The SGA parameters for database instance needs slight modification to support ASM extent maps and other ASM information. Note if the 10g Automatic Memory Management feature is being used on the database instance, then the following sizing data can be treated as informational only or as supplemental data in gauging appropriate values for the SGA. Oracle highly recommends using the Automatic Memory Management feature. The following are guidelines for SGA sizing on the database instance:
Processes = Add 16

Large_pool = Add additional 600k

Shared_pool – Additional memory is required to store extent maps. Aggregate the values from the following queries to obtain current database storage size that is either already on ASM or will be stored in ASM. Then determine the redundancy type that is used (or will be used), and calculate the shared_pool, using the aggregated value as input.

```
select sum(bytes)/(1024*1024*1024) from v$datafile;

select sum(bytes)/(1024*1024*1024) from v$logfile a, v$log b
 where a.group#=b.group#;

select sum(bytes)/(1024*1024*1024) from v$tempfile where
status='ONLINE';

For diskgroups using external redundancy = (Every 100Gb of space needs
1Mb of extra shared pool) + 2M

For diskgroups using Normal redundancy: (Every 50Gb of space needs 1Mb of
extra shared pool) + 4M.

For diskgroups using High redundancy:  (Every 33Gb of space needs 1Mb of
extra shared pool) + 6M.
```

ASM and database shutdown dependencies

Since ASM manages diskgroups and hold the database files and its metadata, a shutdown of the ASM instance will cause all client-database instances to shutdown as well[18]. In normal shutdowns, the ASM instance will begin shutdown and wait for all sessions to disconnect, just as typical database instances. Note, however that ASM has a persistent database instance connection, thus the database instances must be shutdown first, in order for ASM to complete shutdown.

In case of ASM `SHUTDOWN IMMEDIATE or ABORT`, ASM will immediately terminate any open connections (including the database instance connections), and as a result, all dependent databases will immediately abort; i.e., databases will ungracefully shutdown.

In a single ASM instance configuration, if the ASM instance fails while diskgroups are open for update, ASM instance recovery will be performed upon the restart of ASM by reading the disk group logs. In RAC environments, with multiple ASM instances sharing disk groups, if one ASM instance should fail, another node's ASM instance automatically recovers transient ASM metadata changes caused by the failed instance; i.e. performs instance recovery.

[18] If an ASM instance fails, all Oracle database instances dependent on that ASM instance also fail. This is similar to a failure of a volume manager in the OS.

ASM and database deployment Best Practices

ASM provides out-of-the-box enablement of redundancy and optimal performance. However, the following items should be considered to increase performance and/or availability:

1. Implement multiple access paths to the storage array using two or more HBAs or initiators.

2. Deploy multi-pathing software over these multiple HBAs to provide IO load-balancing and failover capabilities.

3. Use diskgroups with similarly sized and performing disks. A diskgroup containing large number of disks provides a wide distribution of data extents, thus allowing greater concurrency for I/O and reduces the occurrences of hotspots. Since a large diskgroup can easily sustain various I/O characteristics and workloads, a single (database area) diskgroup can be used to house database files, logfiles, and controlfiles.

4. Use diskgroups with four or more disks, and making sure these disks span several backend disk adapters.

5. As stated earlier, Oracle generally recommends no more than two diskgroups. For example, a common deployment can be four or more disks in a database diskgroup (DATA diskgroup for example) spanning all back-end disk adapters/directors, and 8-10 disks for the Flash Recovery Area Diskgroup. The size of the Flash area will depend on what is stored and how much; i.e., full database backups, incremental backups, flashback database logs and archive logs. Note, an active copy of the controlfile and one member of each of the redo log group are stored in the Flash Recovery Area.

See the *Oracle High Availability Architecture and Best Practices Manual* for more details on these topics.

`Storage Management and Allocation`

A database created under the constructs of ASM will be striped by default and mirrored; as specified in the SAME methodology; i.e., the I/O load is evenly distributed and balanced across all disks within the diskgroup. In our example, the database will be striped across four disks, and redundancy will be handled by the storage array. The striping is done on a file-by-file basis, using a 1MB stripe size, as opposed to other LVMs that do striping and mirroring at a disk-volume level. ASM 1MB stripe depth has proved to be the best stripe depth for Oracle databases. This optimal stripe depth coupled with even distribution of extents in the diskgroup, prevents the occurrence of hot spots.

ASM allocates space in units called allocation units or AU. ASM always creates one-allocation-unit (AU) extents across all of the disks in a disk group[19]. For a diskgroup with similarly sized disks, there should be an equal number of AU extents on every disk. A database file is broken up into file extents. There are two types of file extent distributions: coarse and fine. For coarse distribution, each coarse grain extent file extent is mapped to a single allocation unit. With fine grain distribution, each grain is interleaved 128K across groups of 8 AUs. Fine distribution breaks up large sized I/O operations, into multiple 128K I/O operations that can execute in parallel, which benefits sequential I/Os. Coarse and fine grain attributes are pre-defined, as part of system templates, for all system related files; e.g., redo and archive log files are defined as fine grain, whereas, datafiles are coarse.

[19] The AU size is currently not user configurable.

Rebalance and Redistribution

With traditional volume managers, expansion/growth or shrinkage of striped filesystems has typically been difficult. With ASM, these disk/volume changes are now seamless redistribution (rebalancing) of the striped data and can be performed online.

Any change in the storage configuration will trigger a rebalance. The main objective of the rebalance operation is to always provide an even distribution of file extents and space usage across all disks in the diskgroup. Rebalancing is performed on all database files on a per file basis[20]; however, some files may not require a rebalance. The Oracle background process, RBAL, from the ASM instance manages this rebalance.

The Rebalance process examines each file extent map, and the new extents are re-plotted on to the new storage configuration. For example, consider an 8-disk diskgroup, with a datafile with 40 extents (each disk will house 5 extents). When 2 new drives of same size are added, that datafile is rebalanced and spread across 10 drives, with each drive containing 4 extents. Only 8 extents need to move to complete the rebalance; i.e., a complete redistribution of extents is not necessary, only the minimum number of extents are moved to reach equal distribution.

The following is a typical process flow for ASM rebalancing:

1. On the ASM instance, a DBA adds (or drops) a disk to (from) a diskgroup.
2. This invokes the RBAL process to create the rebalance plan and then begin coordination of the redistribution
3. RBAL will calculate estimation time and work required to perform the task and then message the ARB*x* processes to actually handle the request. The number of ARB*x* processes invoked is directly determined by the asm_power_limit.
4. The Continuing Operations Directory (metadata) will be updated to reflect a rebalance activity.
5. Each extent to be relocated is assigned to an ARBx process.
6. ARBx performs rebalance on these extents. Each extent is locked, relocated, and unlocked. This is shown as Operation *REBAL* in V$ASM_OPERATION.

Rebalancing involves physical movement of file extents. This impact is generally low because the rebalance is done one AU at a time; therefore, there will only be one outstanding I/O at any given time, per ARBx processes. This should not adversely affect online database activity. However, it is generally advisable to schedule the rebalance operation during off-peak hours.

The init.ora parameter *asm_power_limit* is used to influence the throughput and speed of the rebalance operation[21]. The range of values for asm_power_limit are 0 to 11; where a value of 11 is full throttle and a value of 1 is low speed. A value of 0, which turns off automatic rebalance, should be used with caution.

The power value can also be set for a specific rebalance activity using the *alter diskgroup* command. This value is only effective for the specific rebalance task. See Figure 8. A *power value of 0* indicates that no rebalance should occur for this rebalance. This setting is particularly important when adding or removing storage (that has external redundancy), and then deferring the rebalance to a later scheduled time.

Figure 8.

```
"Session1 SQL"> alter diskgroup DATA add disk '/dev/rdsk/c3t19d39s4' rebalance
power 11
```

[20] A weighting factor, influenced by disk size and file size, affects rebalancing. A larger drive will consume more extents. This is done for even distribution based on overall size.
[21] The asm_power_limit is specific to each ASM instance.

```
From another session

"Session2 SQL"> select * from v$asm_operation

OPERA  STAT      POWER     ACTUAL      SOFAR   EST_WORK   EST_RATE EST_MINUTES

-----  ---- ---------- ---------- ---------- ---------- ---------- -----------

1 REBAL WAIT         11          0          0          0          0          0

1 DSCV  WAIT         11          0          0          0          0          0

(time passes……..)

OPERA STAT      POWER     ACTUAL      SOFAR   EST_WORK   EST_RATE EST_MINUTES

------------ ----- ---- ---------- ---------- ---------- ---------- ----------

1 REBAL REAP         11          2         25        219        485          0
```

If removing or adding several disks, for best practices, it is best to add or remove drives all at once, this will reduce the number rebalance operations that are needed for storage changes.

An ASM diskgroup rebalance is an asynchronous operation, in that the control is returned immediately to DBA after the operation is sent in the background, with the status of the ongoing operation query-able from V$ASM_OPERATION. However, there are situations when the diskgroup operation needs to synchronous; i.e., wait until rebalance is completed. In Oracle Database 10g Release 2, ASM alter diskgroup commands which result in a rebalance now have the option of specifying the option to wait. This allows for accurate [sequential] scripting that may rely on the space change from a rebalance completing before any subsequent action is taken. For instance, if you add 100GB of storage to a completely full disk group, you won't be able to use all 100GB of storage until the rebalance completes.

If a new rebalance command is entered while one is already in progress in *wait* mode, the prior command will not return until the diskgroup is in a balanced state or the rebalance operation encounters an error.

An example SQL script below illustrates how the WAIT option can be used in SQL scripting. The script below adds a new disk, /dev/raw/raw6, and waits until the *add* and *rebalance* operations complete, returning the control back to the script. The subsequent step adds a large tablespace.

```
#An example script to test WAIT option
alter diskgroup data add disk '/dev/raw/raw6' rebalance power 2 wait;

#login into database and create a tablespace for the next month's Order Entry
data
sqlplus  oe_dba/oe1@proddb  << EOF
 create BIGFILE tablespace May_OE  datafile size 800 Gb
<< EOF
```

With external redundancy, the dropping and adding of disks in the diskgroup is very seamless, and becomes more of an exercise of scheduling the rebalancing. However, with failure groups, some planning and forethought maybe required with respect to how disks are removed and added. The best practices for this exercise will be covered in a follow-up article.

Files and Aliases

Files are the objects that database instances access. Files come in the form of datafiles, controlfiles, spfiles, and redo logfiles, and several other file types[22]. The ASM filename syntax is different than the typical naming standards; yet resembles the filesystem approach of the 9*i* OMF feature. ASM file names are derived and generated at the successful creation of a datafile. ASM file names are in the format of

"+diskgroup_name/database_name/database file type/tag_name.file_number.incarnation"

For example, the system tablespace name in the sample case is: *"+DATA/orcl/datafile/system.258.3".*

Note that the tag name in the datafile name corresponds to the tablespace name. For redo logfiles, the tag name is the group number; e.g., *+DATA/orcl/onlinelog/group_3.264.3*
The file number in the ASM instance can be used to correlate and determine filenames in database instance. The illustration below shows the relationship between a database instance's file identity and ASM's. Note, the file number from V$ASM_FILE is embedded in the file name.

Figure 9a.

```
+ASM (ASM instance)
SQL> select file_number , sum(bytes)/(1024*1024) from v$asm_file group by
       file_number

FILE_NUMBER SUM(BYTES)/(1024*1024)
----------- ----------------------
        256             360.007813
        257             35.0078125
        258             450.007813
        261             .002441406
        262             150.007813
        263             23.0078125
        264             10.0004883
        265              5.0078125
        266             10.0004883
        267             10.0004883
        268              2.2109375


ORCL (database instance)
SQL> select name from v$datafile

NAME
----------------------------------------
+DATA/orcl/datafile/sysaux.256.3
+DATA/orcl/datafile/system.258.3
+DATA/orcl/datafile/undotbs1.257.3
+DATA/orcl/datafile/users.265.3
+DATA/orcl/datafile/nitin.263.3
/u01/oradata/orcl/ishan01.dbf

SQL> select member from v$logfile;

MEMBER
--------------------------------------------------------------------------------
-
+DATA/orcl/onlinelog/group_3.264.3
+DATA/orcl/onlinelog/group_2.266.3
```

[22] There are more than 12 file types currently defined .

```
+DATA/orcl/onlinelog/group_1.267.3
```

Note; this database contains ASM files and a non-ASM file named *ISHAN01.dbf*. The ASM list, from the Figure 9a, does not have an entry for the *ISHAN datafile* since it is not ASM managed. It is important to note, that an Oracle Database 10g database can have files that reside on filesystems, raw and ASM, simultaneously.

The filename notation described thus far is called the Fully Qualified FileName (FQFN) notation. FQFN are generally long and awkward, therefore, to make file-naming convention easier to remember the ASM Alias name format was introduced.
ASM Aliases are essentially in hierarchical directory format, similar to the filesystem hierarchy: */u01/oradata/dbname/datafile_name*
Alias names specify a disk group name, but instead of a file and incarnation number, a user-friendly string name is used. Alias ASM filenames are distinguished from fully qualified or numeric names because they do not end in a dotted pair of numbers. The following is an example of an ASM Alias:[23]:

Figure 9b

```
#Create ASM directory

SQL> alter diskgroup DATA add directory '+DATA/oradata/orcl;

#Now create the alias

SQL> alter diskgroup DATA add alias '+DATA/oradata/orcl/nitin01.dbf' for
+DATA/orcl/datafile/nitin.263.3;
```

For best practices, every database should implement the Oracle Managed File (OMF) feature. OMF can be enabled by setting the DB_CREATE_FILE_DEST and DB_RECOVERY_FILE_DEST parameters. OMF will allow simpler file naming and also provides better file handling. For example, if a tablespace is dropped, ASM will delete the file automatically only if it is an OMF datafile. If OMF is not used, then the DBA must drop the file manually by connecting to the ASM instance. OMF files are located in the DB_CREATE_FILE_DEST directory[24]. Please review OMF documentation on usage and implementation. Note, OMF is not used anytime file names are provided in create/alter tablespace add datafile commands.

Aliases are particularly useful when dealing with controlfiles and spfiles; e.g., an Alias ASM filename is normally used in the `CONTROL_FILES` and SPFILE initialization parameter. In Figure 10 below, SPFILE and Control_files parameter are set to the alias, and the db_create_file_dest and db_recovery_file_dest are set to the appropriate OMF destinations.

```
spfile                    = +DATA/orcl/spfileorcl.ora

control_files             = +DATA/orcl/controlfile/control_01.ctl

db_create_file_dest       = +DATA

db_recovery_file_dest     = +FLASH
```

[23] An alias can be assigned during file creation, or can be created for an existing file using the `ALTER DISKGROUP <diskgroup> ADD ALIAS` <alias name> for < file name> command.
[24] There are other *_DEST variables that can be used for other file types.

To show the hierarchical tree of files stored in the diskgroup, use the following connect by clause SQL to generate the full path. Note that you need to have a sufficiently large shared_pool_size in your ASM instance to execute this query. A more efficient way to browse the hierarchy is to use EM.

```
SELECT concat('+'||gname, sys_connect_by_path(aname, '/')) full_alias_path FROM
      (SELECT g.name gname, a.parent_index pindex, a.name aname,
        a.reference_index rindex FROM v$asm_alias a, v$asm_diskgroup g
        WHERE a.group_number = g.group_number)
      START WITH (mod(pindex, power(2, 24))) = 0
    CONNECT BY PRIOR rindex = pindex;
```

Enhanced management for ASM
The following new features provide alternative means to access, manage and manipulate ASM files.

ASM Command Line Interface (ASMCMD)

In Oracle Database Oracle Database 10g Release 1, SQLPlus and EM were the only means to manage and access ASM. In 10.2, a new ASM Command Line Interface (ASMCMD) utility has been introduced, to provide an easy alternative for accessing the files and directories within the ASM diskgroups. The ASMCMD utility provides short commands that have the same functionality as their SQL equivalents; i.e., this obviates the need to build complex SQL statements to query ASM information. Additionally, the ASMCMD syntax is very similar to Unix shell commands, thus simplifying ASM management for those administrators not accustomed to the SQLPlus interface or SQL coding. ASMCMD will work against 10.1 ASM Instances as well.
Note, ASMCMD does not encompass all the functionality of SQLPLus or EM interface, thus Oracle still highly recommends that the Enterprise Manager interface (using either Grid or Database Control) be used as a primary means to manage ASM, and utilize ASMCMD for quick, short commands or for batch command processing.

The ASMCMD interface provides both an interactive and non-interactive modes. The interactive mode provides a shell-like environment where the user is prompted to issue the commands listed below. The non-interactive mode executes a single command and exits the utility. The latter is made available for scripting and batch processing purposes. ASMCMD requires that the ASM instance be started and the diskgroups to be managed are mounted; i.e., ASMCMD cannot mount diskgroups.

Interactive command line
Invoking ASMCMD with no command arguments starts an interactive shell environment.
Interactive Mode command line examples:

```
% asmcmd
ASMCMD> cd +DATA/RAC/DATAFILE
ASMCMD> ls -l
Type      Redund  Striped  Time           Sys  Name
DATAFILE  MIRROR  COARSE   MAR 28 02:03   Y    EXAMPLE.276.553572261
DATAFILE  MIRROR  COARSE   MAR 28 02:03   Y    SYSAUX.275.553572195
DATAFILE  MIRROR  COARSE   MAR 30 09:03   Y    SYSTEM.256.553536323
DATAFILE  MIRROR  COARSE   MAR 30 09:03   Y    UNDOTBS1.258.553536325
DATAFILE  MIRROR  COARSE   MAR 30 09:03   Y    UNDOTBS2.268.553536665
DATAFILE  MIRROR  COARSE   MAR 30 09:03   Y    USERS.259.553536325
```

```
When the "-p" option is used to launch ASMCMD, the current path is shown in the
prompt. For example:
% asmcmd -p
ASMCMD [+]> cd +DATA/RAC/DATAFILE
ASMCMD [+DATA/RAC/DATAFILE]> ls
EXAMPLE.276.553572261
SYSAUX.275.553572195
SYSTEM.256.553536323
UNDOTBS1.258.553536325
UNDOTBS2.268.553536665
USERS.259.553536325
```

The du command operates similar to the Unix du –s command, it shows used disk space.

```
% asmcmd -p
ASMCMD [+] > cd +DATA/RAC/ARCHIVELOG
ASMCMD [+DATA/RAC/ARCHIVELOG]> du
       Used_Mb      Mirror_used_Mb
       2504         2504
```

List of commands:

```
help [command]                                  Help listing
pwd                                             Show current directory (current
directory)
cd <dir>                                        Change directory
find [-t <type>] <dir> <alias>                  File search
ls [-lsdrtLaH] [alias]                          Directory listing
mkdir <dir1 dir2 . . .>                         Directory creation
rm <file1, file2 . . .>                         File and directory deletion
mkalias <system_alias> <user_alias>             User alias creation
rmalias [-r] <user_alias1 user_alias2 . . .>
                                                User alias deletion
du [-H] [dir]                                   Disk usage
lsdg [-H] [group]                               Diskgroup info
lsct [-H] [group]                               Client info
exit                                            exit
```

Non-interactive Mode command line:

```
% asmcmd ls -L +DATA/RAC/DATAFILE
EXAMPLE.267.553536489
SYSAUX.257.553536323
SYSTEM.256.553536323
UNDOTBS1.258.553536325
UNDOTBS2.268.553536665
USERS.259.553536325

asmcmd find -t DATAFILE +DG1 %
+DG1/ORA102/DATAFILE/TEST.256.560274299

You must use the '%' as the wildcard for non-interactive mode queries.
```

DBMS_FILE_TRANSFER Utility Enhancements

The DBMS_FILE_TRANSFER is a stored procedure that was introduced in 9i. This package provided a means to copy files between two locations (on same host or between database servers). In 10.1, ASM leverages this utility to copy files between ASM diskgroups, and is the primary utility used to instantiate an ASM DataGuard database. In 10.2, DBMS_FILE_TRANSFER has been enhanced to support all combinations of ASM and non-ASM file transfers as follows:

- ASM to ASM
- ASM to OS file
- OS file to ASM
- OS file to OS file

These changes now provide DBAs with another method to migrate database files into and out of ASM storage.

XML ASM Virtual Folder

Oracle Database 10g ASM Release 2 leverages the virtual folder feature in XML DB, providing a mechanism to access and manipulate ASM files and folders via XML DB protocols such as FTP and HTTP/DAV and programmatic APIs[25]. The ASM virtual folder is mounted as /sys/asm within the XML DB hierarchy. The folder is *virtual,* in the sense that the ASM folders and files are not actually stored within XML DB. However, any operation on the ASM virtual folder is transparently handled by the underlying ASM component.
The ASM virtual folder is created by default during the installation of XML DB. If the database is not configured to use ASM, then this folder will be empty and no operations will be permitted on it. If ASM is configured, the ASM virtual folder, /sys/asm, is mounted within the XML DB hierarchy. Any operation on the ASM virtual folder is transparently handled by the underlying ASM component. The ASM folder contains one sub-folder for every mounted disk group. Each diskgroup folder will contain one sub-folder corresponding to every database name. In addition it may contain additional files and folders corresponding to the aliases that have been created by the administrators.

ASM Virtual Folder access via FTP

File Transfer Protocol (FTP) is also available for copying datafiles in and out of the ASM diskgroup offering additional flexibility for managing your files. Determine the appropriate FTP port defined for your environment. The example below describes how to extract, copy, and delete Datapump dumpsets from an ASM directory,+Flash/dumpsets, which equates to the following XML DB folder, /sys/asm/flash/dumpsets.

```
[oracle@edrsr14p1 oracle]$ ftp edrsr14p1 2100
Connected to edrsr14p1 (139.185.35.114).
220- edrsr14p1
220 edrsr14p1 FTP Server (Oracle XML DB/Oracle Database) ready.
Name (edrsr14p1:oracle): system
331 pass required for SYSTEM
Password:
230 SYSTEM logged in
Remote system type is Unix.
ftp> cd /sys/asm/flash/orcl/dumpsets
250 CWD Command successful
ftp> get expdp_5_5.dat
local: expdp_5_5.dat remote: expdp_5_5.dat
227 Entering Passive Mode (139,185,35,114,100,234)
150 ASCII Data Connection
278530 bytes received in 0.153 secs (1.8e+03 Kbytes/sec)
ftp> put expdp_03_26.dat
ftp> del expdp_03_20.dat
ftp> rename expdp_03_26.dat latest_dumpset.dmp
```

Templates

ASM file templates are a named collections of attributes applied to files during file creation. Templates simplify file creation by housing complex file attribute specifications. When an ASM template is applied to a file, that file will have all the attributes of that template. When a disk group is created, ASM establishes a set of initial system default templates associated with that disk group. These templates contain the default attributes for the various Oracle database file types. The administrator may change attributes of the default templates. Additionally, administrators can add their own unique templates, as required. This template specification enables simpler and accurate file creation attributes. System default templates

[25] To determine the XML DB HTTP and FTP ports, go to the EM Administration page, then under the XML Database section go to the XML Configuration link.

cannot be deleted. If you need to change an ASM file attribute after the file has been created, the file must be copied, into a new file with the new attributes. This is the only method of changing a file's attributes.

```
SQL> alter diskgroup DATA add template all_new_files attributes (unprotected
fine)

Once the template is created, it can be applied to a new tablespace.

SQL> create tablespace ISHAN datafile '+DATA/ishan(all_new_files)' size 100M;
```

Simplified ASM migration using EM

The EM Migration utility is a new feature in Oracle Database 10g Release 2 that provides simplified database migration to ASM based storage. This utility migrates 10.2 and later databases residing on file system or raw devices to Automatic Storage Management (ASM) on the same host. The EM Migration utility can be accessed from the Maintenance tab of the Database Instance page of EM.

The purpose of EM Migration utility is to automate and simplify the database migration process to ASM. The EM Migration utility provides the following benefits:

- Supports migrating all database files, recovery related files, and spfile.
- Provides a wizard walk-through to create migration requests, process and schedule migration requests, as well as report statuses of migration requests.
- Migrates database using the same DBNAME, DBID and SID
- Provides a fully recoverable fallback database, if user decides to go back to non-ASM based storage (assuming that ASM storage and existing storage exist at the same time)
- Monitors the migration process
- Supports online and offline database migration

The EM Migration utility essentially uses Recovery Manager (RMAN) commands to migrate control files, datafiles, and recovery related files from filesystem to ASM.[26] The migration task is created and submitted as a job via EM job system. The task specification process includes walking through the wizard, which involves real-time validation of user requests. The submitted job includes the following major sub-steps:

- Construct init parameters
- Migrate database files with RMAN
- Migrate temp files and online logs with PL/SQL procedures
- Cleanup.

The following URL Link also describes how to perform migration to ASM

http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gASMMigration.pdf

[26] For files such as online logs and temp files, the utility will migrate them by dropping and recreating the files, using PL/SQL procedures, and then recreate them in ASM.

ASM Management

ASM is the volume manager for all databases that employ ASM on a given node. Therefore, only one ASM instance is required per node regardless of the number of database instances on the node. Additionally, ASM seamlessly works with the RAC architecture to support clustered storage environments. In RAC environments, there will be one ASM instance per clustered node, and the ASM instances communicate with each other on a peer-to-peer basis using the interconnect.

With the advent of ASM, the lines of responsibility for volume and storage management may seem to become blurred. The roles defined around ASM management can have some variation, depending on the organizational layout. The following are examples of some typical deployments:

- There is an ASM administrator that specifically manages the ASM instance. This ASM administrator can be carved from the Sysadmin department. The Oracle software associated with the ASM instance can be installed by a distinct user from the database user. This can help to enforce the ASM admin role.

- The standard DBA, e.g.; DBAs or owners of the Oracle software, that are responsible for managing the enterprise databases also manage the storage for ASM instance. The Sysadmin still manages the OS LUNs, changes the device file ownership, and defines ASM disk isolation from other host LVM (logical volume manager) disks.

Regardless of the split of roles and responsibilities, the following activities are needed to provision storage to ASM.

- Identifying Storage Requirements for ASM
    - Determine the storage types needed by the application/database; i.e; SATA or FC. This is an important question if deploying a Tiered Storage solution. For example, the DATA diskgroup may require high-end FC disks, while the Flash diskgroup may need only SATA disks.
    - Determine how much space will required to house the database(s).
    - Determine the potential aggregate database IOPs. The combination of this and space requirement will help dictate the number of backend spindles (disk units) necessary to accommodate the database requirements. This step must be done with the assistance of the Storage and System Administrators.
    -

- Prepare Disks from Storage Array
    - Create LUNs from the storage array. These LUNs, which can be created from RAID group sets (RAID10 or RAID5) from the array, are then (in SAN environments) zoned to the appropriate hosts.
    - Leverage the best practices listed in the ASM Deployment Best Practices Section.

- Prepare Disks on host
    - The root user will validate that those LUNs presented to the host are seen by the OS. These new LUNs should have their ownerships changed to match the owner of the Oracle software.
    - Once the ownership/permissions are changed, ASM should discover these disks. These disks can now be added to an existing diskgroup and used in creating a new diskgroup.

- Create the necessary diskgroups

- Once ASM has discovered the disks presented from the storage array, determine how many disks will be placed in a diskgroup.
- Use existing diskgroup or create new diskgroup

Conclusion

ASM provides a point solution focused on unified interface database storage management. This complete solution facilitates Grid Computing, lowers overall manageability costs as well as helping to maintain serviceability levels. The following is a review of the high points of ASM:

Reduce administration complexity:

- Simplifies or eliminates daily database administration tasks
- Automatic I/O tuning for all types of workloads
- Reduces the number of objects to manage, because vertical integration of file system and volume manager into the Oracle kernel reduces complexity, as one disk group replaces multiple file systems. Also, with OMF, you do not have to worry about managing files, just tablespaces.
- Simplifies database storage configuration changes: Automatic data copy on disk, add and drop. Online migration to new storage hardware.
- Reduced downtime: ASM prevents accidental file deletion because there is no file system interface, and ASM is responsible for file management.

Reduces costs of storage management and Increases Utilization

- Reduces the cost of managing storage.
- Provides a mechanism for Database Storage Consolidation
- Provides clustered volume manager and file system functionality integrated with the database.
- Makes block devices as simple to administer as file servers.
- Works with any type of disk from modular storage, NAS devices to SAN disk arrays.

Improves Performance, Scalability, and Reliability

- Raw disk I/O performance for all files
- Stripe files across multiple storage arrays
- Higher storage resource utilization
- ASM-Lib API improves CPU utilization to reap I/O and provide priority and caching hints
- Overcomes file system size limitations
- Implements mirroring to protect against storage array failure.

For more information on ASM, please visit the ASM website on OTN:

http://www.oracle.com/technology/products/database/asm/index.html

Appendix A. Views and Discover Strings

The following table illustrates the various used and their relationship with the instance type.

ASM related V$ Views	ASM Instance	Database Instance
V$ASM_DISKGROUP	row/diskgroup discovered	Row/diskgroup available to the local database instance.
V$ASM_CLIENT	row/client connected	row for local ASM instance if DB contains open ASM files
V$ASM_DISK	Row/disk discovered, across all diskgroups as well as disks that are not in disk groups	Row/disk, across all diskgroups used by the local database instance.
V$ASM_FILE	Row/file allocated; across all client instances-diskgroups	N/A
V$ASM_TEMPLATE	Row/template	Row/template w/ associated-attached diskgroup
V$ASM_ALIAS	Row/file alias	N/A
V$ASM_OPERATION	Row/per active ASM operation (could be more than 1 operation per invocation)	N/A

Appendix B. Migrating individual non-ASM datafiles to ASM

This section will describe how to migrate individual tablespaces to ASM, whilst the database is online. This illustration assumes that there is a tablespace named ISHAN located on a mounted filesystem. Additionally, this same procedure can also be used to move a datafile from one diskgroup to another diskgroup.

1. Connect to RMAN
   RMAN> connect target

2. Make the target tablespace offline or read-only
   RMAN> sql "alter tablespace ISHAN offline";

3. Copy the ISHAN tablespace to the DATA diskgroup
   RMAN> backup as copy tablespace ishan format '+DATA';

4. On successful copy of the tablespace, switch all the datafiles in the ISHAN tablespace to ASM. Determine the file number of the files to be switched (use v$datafile).

   RMAN> switch datafile 6 to copy;

   When the following message is received, it is deemed that the tablespace is migrated successfully.
   datafile 6 switched to datafile copy "+DATA/orcl/datafile/ishan.314.1"

   Note, the original filesystem copy still exists, and can be deleted.

Appendix C. Datapump filesets and ASM

This section describes how to create DataPump export dumpsets within ASM diskgroups.

1. Create a directory from ASM. This step is only necessary, if you plan on storing the dumpset in a specific directory

   alter diskgroup flash add directory '+flash/dumpsets';

2. Create a directory in database

```
create directory dumpsets as '+flash/dumpsets';
```

3. create a logfile directory, since logfiles cant be stored in ASM

```
create directory logfile_dest as '/u01/admin/dumpsets';
```

4. expdp parfile=expdp_parfile
where expdp_parfile contents are:

```
userid=system/manager1 directory='DUMPSETS' dumpfile=expdp_5_5.dat
job_name=full_export logfile=logfile_dest:exp5_5.log
```

5. Validate that the file is created within ASM.

```
SQL> select file_number, bytes, creation_date from v$asm_file
where type ='DUMPSET';

 FILE_NUMBER  BYTES     CREATION_

 -----------  ----------  ---------

 581          2785280     07-MAY-04
```

Appendix D. Creating extra controlfiles in ASM

This following procedure can be used to create additional controlfiles in the ASM diskgroup.

1. Create a new directory in your disk group using commands similar to the following ones (this is optional).

```
alter diskgroup <DG_name> add directory '+<DG_name>/<DB_name>';

alter diskgroup <DG_name> add directory
'+<DG_name>/<DB_name>/controlfile';
```

2. Then, edit the control file entry in your SPFILE/init.ora to point to the new controlfile. Where the new controlfile is +DATA2/ora10g/controlfile/control03.ctl'

```
*.control_files='+DATA/0ra10g/controlfile/current.260.3,'+FLASH/Ora1
0G/controlfile/o1_mf_0fh3t83f_.ctl,'DATA2/ora10g/controlfile/control
03.ctl'
```

3. Shutdown cleanly your database (all instances if using RAC)

```
shutdown immediate;
```

4. Startup the database in nomount mode

```
startup nomount;
```

5. Using RMAN, restore the controlfile from one of the existing locations to the new one.

```
restore controlfile to 'DATA2/Ora10g/controlfile/control03.ctl' from
'+FLASH/Ora10g/controlfile/current.260.3';
```

6. Mount the database and recover the database if necessary.

```
SQL> alter database mount;
```

7. Open the database.

```
SQL> alter database open;
```

Appendix E. ASM SQL scripts

The following scripts are outlined to assist the use in managing ASM files; i.e., listing or deleting. Note, Oracle recommends using 10g EM Grid Control or Database Control for performing these functions.

1. This SQL statement generates a script to list the ASM files for a given database. The list is order by listing first the directory structure and then the files. The script assumes that the files were created using the asm file name conventions, in specific it assumes that the given database name is present in the alias name (full_path column). The query will prompt for the database name.

full_path variable in the query refers to the alias name. The *dir* column indicates if it is a directory or not, and *sys* column indicates if the alias was created by the system or not.

```
COL full_path FORMAT a120
COL dir FORMAT a3
COL sys FORMAT a3
SET ECHO OFF
SET VERIFY OFF
SET FEEDBACK OFF
SET PAGESIZE 39
SET LINESIZE 130

--get database name
ACCEPT database PROMPT 'Enter the database name: '

--spool file
spool lis_&database..lst


--generates  list
SELECT full_path, dir, sys FROM
(SELECT CONCAT('+'||gname, SYS_CONNECT_BY_PATH(aname,'/')) full_path, dir,
sys FROM
(SELECT g.name gname, a.parent_index pindex, a.name aname,
a.reference_index rindex, a.ALIAS_DIRECTORY dir, a.SYSTEM_CREATED sys
FROM v$asm_alias a, v$asm_diskgroup g
WHERE a.group_number = g.group_number)
START WITH (MOD(pindex, POWER(2, 24))) = 0
CONNECT BY PRIOR rindex = pindex
ORDER BY dir desc, full_path asc)
WHERE full_path LIKE UPPER('%/&database%');

--end spool
spool off
-- resetting format
COL full_path CLEAR
COL dir CLEAR
COL sys CLEAR
SET VERIFY ON
SET HEADING ON
SET FEEDBACK ON
```

2. This SQL statement will generate a script that will delete files in ASM for a given database. The script assumes that the files were created using the ASM naming conventions. Specifically, it assumes that the given database name is present in the alias name (full_path column), in the form of "/<db_name>/". The scripts will prompt for the database name.

```
COL gsql FORMAT a300
SET ECHO OFF
SET VERIFY OFF
SET HEADING OFF
SET FEEDBACK OFF
SET PAGESIZE 0
SET LINESIZE 600
SET TRIMSPOOL ON

--get database name
ACCEPT database PROMPT 'Enter the database name: '

--spool file
spool del_&database..sql

--generates file list
SELECT 'ALTER DISKGROUP '||gname||' DROP FILE '''||full_path||''';' gsql
FROM
(SELECT CONCAT('+'||gname, SYS_CONNECT_BY_PATH(aname,'/')) full_path, gname
FROM
(SELECT g.name gname, a.parent_index pindex, a.name aname,
a.reference_index rindex, a.ALIAS_DIRECTORY adir
FROM v$asm_alias a, v$asm_diskgroup g
WHERE a.group_number = g.group_number)
WHERE adir='N'
START WITH (MOD(pindex, POWER(2, 24))) = 0
CONNECT BY PRIOR rindex = pindex)
WHERE full_path LIKE UPPER('%/&database/%');
```

3. This SQL statement will generate a script to drop ASM directories.

```
SELECT 'ALTER DISKGROUP '||gname||' DROP DIRECTORY '''||full_path||''';'
gsql FROM
(SELECT CONCAT('+'||gname, SYS_CONNECT_BY_PATH(aname,'/')) full_path, gname
FROM
(SELECT g.name gname, a.parent_index pindex, a.name aname,
a.reference_index rindex, a.ALIAS_DIRECTORY adir
FROM v$asm_alias a, v$asm_diskgroup g
WHERE a.group_number = g.group_number)
WHERE adir='Y'
START WITH (MOD(pindex, POWER(2, 24))) = 0
CONNECT BY PRIOR rindex = pindex
ORDER BY full_path desc)
WHERE full_path LIKE UPPER('%/&&database%');

--end spool
spool off
-- resetting format
COL gsql CLEAR
SET VERIFY ON
SET HEADING ON
SET FEEDBACK ON
SET PAGESIZE 30
SET LINESIZE 600
SET TRIMSPOOL OFF
```

Oracle Database 10g Release 2 Automatic Storage Management Overview and Technical Best Practices

May 2007
Author: Nitin Vengurlekar
Contributing Authors: Rich Long, Ara Shakian

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com