Oracle 10g DBA Fundamentals

Example and Code Reference

v. 1.0   Feb, 2007

Edited by: Ahmed Baraka

Contents

Usage Terms

- Anyone is authorized to copy this document to any means of storage and present it in any format to any individual or organization for *non-commercial* purpose free.

- No individual or organization may use this document for *commercial* purpose without a written permission from the editor.

- There is no warranty of any type for the code or information presented in this document. The editor is not responsible for any loses or damage resulted from using the information or executing the code in this document.

- If any one wishes to correct a statement or a typing error or add a new piece of information, please send the request to **info@ahmedbaraka.com** . If the modification is acceptable, it will be added to the document, the version of the document will be incremented and the modifier name will be listed in the version history list.

Version History

Version	Date	Updates
1.0	Feb, 2007	Initial document.

Document Purpose

This document is a quick how-to reference of performing fundamental DBA tasks. It just shows the task, the steps and code to perform it and any warnings or limitations. It is quit useful for new Oracle DBAs.

Oracle 10g DBA Fundamentals Example and Code Reference

Administrative Privileges and Authentication

Administrative Privileges

When you connect with SYSDBA or SYSOPER privileges, you connect with a default schema, not with the schema that is generally associated with your username. For SYSDBA this schema is SYS; for SYSOPER the schema is PUBLIC.

Authentication Methods

Using Operating System Authentication

Two special operating system groups control database administrator connections when using operating system authentication:

UNIX User Group: dba and oper

Windows User Group: ORA_DBA and ORA_OPER

```
CONNECT / AS SYSDBA
CONNECT / AS SYSOPER
```

For a remote database connection over a secure connection, the user must also specify the net service name of the remote database:

```
CONNECT /@net_service_name AS SYSDBA
CONNECT /@net_service_name AS SYSOPER
```

Using Password File Authentication

Preparing to Use Password File Authentication

1. Log in to the database by using OS authentication.

2. Set the REMOTE_LOGIN_PASSWORDFILE parameter to NONE and restart the database.

3. If not already created, create the password file using the ORAPWD utility:

```
ORAPWD FILE=filename PASSWORD=password ENTRIES=max_users
```

4. Set the REMOTE_LOGIN_PASSWORDFILE initialization parameter to EXCLUSIVE.

5. Connect to the database as user SYS (or as another user with the administrative privileges).

6. Grant the SYSDBA or SYSOPER system privilege to the user.

7. Restart the instance.

Caution: It is critically important to the security of your system that you protect your password file.

Note: If the server does not have an EXCLUSIVE password file, Oracle Database issues an error if you attempt to grant SYSDBA or SYSOPER privileges.

Viewing Password File Members

Use the V$PWFILE_USERS view to see the users who have been granted SYSDBA or SYSOPER system privileges for a database.

Creating an Oracle Database

Planning for Database Creation

1. Structure and size of tables and indexes.

2. Plan the layout of the underlying DB files.

3. Select the global database name by setting both the DB_NAME and DB_DOMAIN initialization parameters.

4. Decide values of important parameters.

5. Select the database character set.

Caution: AL32UTF8 is the Oracle Database character set that is appropriate for XMLType data.

6. Consider what time zones your database must support (The default time zone file is timezonelrg.dat).

Oracle 10g DBA Fundamentals Example and Code Reference

7. Select the standard database block size (`DB_BLOCK_SIZE`).

8. Plan to use an undo tablespace to manage your undo data.

9. Develop a backup and recovery strategy to protect the database from failure. It is important to protect the control file by multiplexing, to choose the appropriate backup mode, and to manage the online and archived redo logs.

Creating the Database Manually

Tip: You can use the DBCA to create the databse creation script. Then use the generated script to manually create the database.

1: Decide on Your Instance Identifier (SID)

```
setenv ORACLE_SID mynewdb
```

2: Establish the Database Administrator Authentication Method

Decide about the password file and OS authentication methods.

3: Create the Initialization Parameter File

You can use the sample file `$ORACLE_HOME/dbs/init.ora`

See section "Basic Initialization Parameter Settings" for list of commonly parameter settings.

See section "Using SPFILE" for using SPFILe.

4: Connect to the Instance

```
$ SQLPLUS /nolog
CONNECT SYS/password AS SYSDBA
```

5: Create a Server Parameter File

```
-- create the server parameter file
CREATE SPFILE='/u01/oracle/dbs/spfilemynewdb.ora' FROM
        PFILE= '/u01/oracle/admin/initmynewdb/scripts/init.ora';
SHUTDOWN IMMEDIATE
-- the next startup will use the server parameter file
EXIT
```

6: Start the Instance

```
STARTUP NOMOUNT
```

7: Issue the CREATE DATABASE Statement

You can use the following options with the `CREATE DATABASE` command:

- `FORCE LOGGING` (important for Data Guard)

- `SET TIME ZONE`
  `SELECT * FROM V$TIMEZONE_NAMES`

```
CREATE DATABASE mynewdb
    USER SYS IDENTIFIED BY pz6r58
    USER SYSTEM IDENTIFIED BY y1tz5p
    LOGFILE GROUP 1 ('/u01/oracle/oradata/mynewdb/redo01.log') SIZE 100M,
            GROUP 2 ('/u01/oracle/oradata/mynewdb/redo02.log') SIZE 100M,
            GROUP 3 ('/u01/oracle/oradata/mynewdb/redo03.log') SIZE 100M
    MAXLOGFILES 5
    MAXLOGMEMBERS 5
    MAXLOGHISTORY 1
    MAXDATAFILES 100
    MAXINSTANCES 1
    CHARACTER SET US7ASCII
    NATIONAL CHARACTER SET AL16UTF16
    DATAFILE '/u01/oracle/oradata/mynewdb/system01.dbf' SIZE 325M
    EXTENT MANAGEMENT LOCAL
    SYSAUX DATAFILE '/u01/oracle/oradata/mynewdb/sysaux01.dbf' SIZE 325M
    DEFAULT TABLESPACE tbs_1
    DEFAULT TEMPORARY TABLESPACE tempts1
    TEMPFILE '/u01/oracle/oradata/mynewdb/temp01.dbf'
    SIZE 20M REUSE
    UNDO TABLESPACE undotbs
    DATAFILE '/u01/oracle/oradata/mynewdb/undotbs01.dbf'
    SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
```

Oracle 10g DBA Fundamentals Example and Code Reference

8: Create Additional Tablespaces

```
CONNECT SYS/password AS SYSDBA
-- create a user tablespace to be assigned as the default tablespace for users
CREATE TABLESPACE users LOGGING
 DATAFILE '/u01/oracle/oradata/mynewdb/users01.dbf'
 SIZE 25M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
 EXTENT MANAGEMENT LOCAL;
-- create a tablespace for indexes, separate from user tablespace
CREATE TABLESPACE indx LOGGING
 DATAFILE '/u01/oracle/oradata/mynewdb/indx01.dbf'
 SIZE 25M REUSE AUTOEXTEND ON NEXT  1280K MAXSIZE UNLIMITED
 EXTENT MANAGEMENT LOCAL;
```

9: Run Scripts to Build Data Dictionary Views

```
CONNECT SYS/password AS SYSDBA
@/u01/oracle/rdbms/admin/catalog.sql
@/u01/oracle/rdbms/admin/catproc.sql
EXIT
```

10: Run Scripts to Install Additional Options (Optional)

If you plan to install other Oracle products to work with this database, see the installation instructions for those products (Oracle Database Reference).

11: Back Up the Database.

Take a full backup of the database.

Basic Initialization Parameter Settings

Determining the Global Database Name

```
DB_NAME = test (max of 8 characters)
DB_DOMAIN = us.acme.com
```

You can rename the GLOBAL_NAME of your database using the ALTER DATABASE RENAME GLOBAL_NAME

Specifying a Flash Recovery Area

```
DB_RECOVERY_FILE_DEST_SIZE (in bytes)
DB_RECOVERY_FILE_DEST
```

Note: You cannot enable these parameters if you have set values for the LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST parameters.

Specifying Control Files

```
CONTROL_FILES=C:\...\CONTROL01.CTL, C:\...\CONTROL02.CTL
DB_BLOCK_SIZE Initialization Parameter
```

A larger data block size provides greater efficiency in disk and memory I/O

```
DB_BLOCK_SIZE=4096 DB_BLOCK_SIZE=8192
```

Limiting the Size of the SGA

```
SGA_MAX_SIZE=293601280 (in bytes)
```

Using Automatic Shared Memory Management

```
SGA_TARGET=293601280 (in bytes)
```

Using Manual Shared Memory Management

```
DB_CACHE_SIZE
SHARED_POOL_SIZE
LARGE_POOL_SIZE
JAVA_POOL_SIZE
STREAMS_POOL_SIZE
```

To obtain information about SGA use:

```
V$SGA, V$SGAINFO, V$SGASTAT, V$SGA_DYNAMIC_COMPONENTS, V$SGA_DYNAMIC_FREE_MEMORY
```

Specifying the Maximum Number of Processes

```
PROCESSES
```

The COMPATIBLE Parameter

- If you create an Oracle Database 10g database, but specify COMPATIBLE = 9.2.0.2 in the initialization parameter file, then features that requires 10.0 compatibility will generate an error if you try to use them.

- If you do advance the compatibility of your database with the `COMPATIBLE` initialization parameter, there is no way to start the database using a lower compatibility level setting, except by doing a point-in-time recovery to a time before the compatibility was advanced.
- This parameter enables or disables the use of features in the database that affect file format on disk.

Troubleshooting Database Creation

- If database creation fails, you can look at the alert log to determine the reason for the failure and to determine corrective action.
- You should shut down the instance and delete any files created by the `CREATE DATABASE` statement before you attempt to create it again.

Dropping a Database

The `DROP DATABASE` statement deletes all control files and all other database files listed in the control file. To use the `DROP DATABASE` statement successfully, all of the following conditions must apply:

- The database must be mounted and closed.
- The database must be mounted exclusively--not in shared mode.
- The database must be mounted as `RESTRICTED`

Using SPFILE

Migrating to a Server Parameter File

```
CREATE SPFILE='/u01/oracle/dbs/test_spfile.ora'
FROM PFILE='/u01/oracle/dbs/test_init.ora'
```

The SPFILE initialization parameter contains the name of the current server parameter file.

To change initialization parameter values:

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES=50
COMMENT='temporary change on Nov 29'
SCOPE=MEMORY|SPFILE|BOTH
```

Viewing Parameter Settings

```
V$PARAMETER
```
  initialization parameters that are currently in effect for the session.

```
V$PARAMETER2
```
  initialization parameters that are currently in effect for the session. It enables you to view values of parameters that may take two values.

```
V$SPPARAMETER
```
  contents of the server parameter file

```
V$SYSTEM_PARAMETER, V$SYSTEM_PARAMETER2
```
  initialization parameters that are currently in effect for the instance

```
V$PARAMETER_VALID_VALUES
```
  list of valid values for list parameters

Considerations After Creating a Database

- Read and Configure Oracle Security Checklist document
- Enable Transparent Data Encryption
- Consider using Secure External Password Store
- If required, install sample schemas. Refer to the documentation "Oracle® Database Sample Schemas 10g"

Starting Up and Shutting Down the Database

Manual Startup Steps

- Follow the following steps:
```
set ORACLE_SID=mydb
SQLPLUS /NOLOG
```

```
CONNECT username/password AS SYSDBA
STARTUP NOMOUNT | MOUNT | OPEN
```

- You can use the `RESTRICT` clause in combination with the `MOUNT`, `NOMOUNT`, and `OPEN` clauses.

- To disable the `RESTRICTED SESSION` feature:

```
ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

- To place an instance back into the restricted mode, where only users with administrative privileges can access it, use the SQL statement:

```
ALTER SYSTEM ENABLE RESTRICTED SESSION;
```

- If you cannot cleanly shut down the instance or you experience problems when starting an instance, you can force starting up the database:

```
STARTUP FORCE
```

In this case, if an instance is running, STARTUP FORCE shuts it down with mode ABORT before restarting it.

- For information about automatic database startup, see your operating system specific Oracle documentation.

Altering Database Availability

```
ALTER DATABASE MOUNT
ALTER DATABASE OPEN READ ONLY
ALTER DATABASE OPEN READ WRITE
```

Shutting Down the Database

```
set ORACLE_SID=mydb
Connect / as sysdba
SHUTDOWN NORMAL
SHUTDOWN IMMEDIATE
SHUTDOWN TRANSACTIONAL
SHUTDOWN ABORT
```

Quiescing a Database

- In this state, the database allows only DBA transactions (SYS and SYSTEM)

```
ALTER SYSTEM QUIESCE RESTRICTED
```

  o The statement will take effect when Non-DBA active sessions become inactive.

  o To display sessions that block quiesce operations:

```
select bl.sid, user, osuser, type, program
from v$blocking_quiesce bl, v$session se
where bl.sid = se.sid;
```

  o Until the database is later unquiesced, any session trying to be active will hang.

  o You cannot perform a cold backup when the database is in the quiesced state.

  o You can still take online backups while the database is in a quiesced state.

- The following statement restores the database to normal operation:

```
ALTER SYSTEM UNQUIESCE
```

- To view the quiesce state of an instance, query the ACTIVE_STATE column of the V$INSTANCE . Its possible values are: NORMAL, QUIESCING, QUIESCED

```
Select active_state from v$instance
```

Configuring Shared Server Database

Shared Server Configuration Steps

- Shared server is enabled by setting the `SHARED_SERVERS` initialization parameter to a value greater than 0.

- The SHARED_SERVERS initialization parameter specifies the minimum number of shared servers that you want created when the instance is started.

```
ALTER SYSTEM SET SHARED_SERVERS = 5;
```

- Other Related Parameters
  - o DISPATCHERS
  - o MAX_SHARED_SERVERS
  - o PROCESSES
  - o SHARED_SERVER_SESSIONS
  - o CIRCUITS sets a maximum limit on the number of virtual circuits that can be created in shared memory.
- Determining the Number of Dispatchers

You should know the number of possible connections for each process for the operating system. Then use the formula:

Number of dispatchers = CEIL (max. concurrent sessions / connections for each dispatcher)

- Set Dispatchers:

```
DISPATCHERS='(PROT=tcp)(DISP=5)', '(PROT-tcps)(DISP=3)'
```

Shutting Down Specific Dispatcher Processes

- When you change the DESCRIPTION, ADDRESS, PROTOCOL, CONNECTIONS, TICKS, MULTIPLEX, and POOL attributes of a dispatcher configuration, in order for the change to be effective for all dispatchers associated with a configuration, you must forcibly kill existing dispatchers after altering the DISPATCHERS parameter.

```
SELECT NAME, NETWORK FROM V$DISPATCHER;
ALTER SYSTEM SHUTDOWN IMMEDIATE 'D002'
```

Disabling Shared Servers

- You disable shared server by setting SHARED_SERVERS to 0.
- To terminate dispatchers once all shared server clients disconnect, enter this statement:

```
ALTER SYSTEM SET DISPATCHERS = '';
```

Monitoring Shared Server

- V$DISPATCHER
- V$DISPATCHER_CONFIG
- V$DISPATCHER_RATE Provides rate statistics for the dispatcher processes.
- V$QUEUE
- V$SHARED_SERVER
- V$CIRCUIT Contains information about virtual circuits, which are user connections to the database through dispatchers and servers.
- V$SHARED_SERVER_MONITOR Contains information for tuning shared server.
- V$SGA Contains size information about various system global area (SGA) groups. May be useful when tuning shared server.
- V$SGASTAT Contains detailed statistical information about the SGA, useful for tuning.
- V$SHARED_POOL_RESERVED

Managing Background Processes

Terminating Sessions

- When an inactive session has been terminated, the STATUS of the session in the V$SESSION view is KILLED. The row for the terminated session is removed from V$SESSION after the user attempts to use the session again and receives the ORA-00028 message.

```
ALTER SYSTEM KILL SESSION 'SID,Serial#5';
```

Managing Server-Generated Alerts

- To set an threshold
```
-- warning and critical threshold for tablespace usage
begin
DBMS_SERVER_ALERT.SET_THRESHOLD(
METRICS_ID=>dbms_server_alert.tablespace_pct_full,
WARNING_OPERATOR=>dbms_server_alert.operator_ge,
WARNING_VALUE=>80,
CRITICAL_OPERATOR=>dbms_server_alert.operator_ge,
CRITICAL_VALUE=>95,
OBSERVATION_PERIOD=>1,
CONSECUTIVE_OCCURRENCES=>1,
INSTANCE_NAME=>NULL,
OBJECT_TYPE=>dbms_server_alert.object_type_tablespace,
OBJECT_NAME=>NULL);
end;
```

- To view threshold
  - o Use GET_THRESHOLD procedure
  - o You can also check specific threshold settings with the DBA_THRESHOLDS view.

- Viewing Alert Data
  - o DBA_THRESHOLDS   lists the threshold settings defined for the instance.
  - o DBA_OUTSTANDING_ALERTS   describes the outstanding alerts in the database.
  - o DBA_ALERT_HISTORY   lists a history of alerts that have been cleared.
  - o V$ALERT_TYPES   provides information such as group and type for each alert.

Monitoring the Database Using Trace Files and the Alert Log

- Search for all internal errors (ORA-600), block corruption errors (ORA-1578), and deadlock errors (ORA-60) that occur.

- Check the alert log and other trace files of an instance periodically to learn whether the background processes have encountered errors.

- To obtain the Location of Trace Files
```
select value
from v$parameter
where name in ('background_dump_dest','user_dump_dest')
```

- You can control the maximum size of all trace files (excluding the alert log) using the initialization parameter MAX_DUMP_FILE_SIZE, which limits the file to the specified number of operating system blocks.

Monitoring Locks

- Run catblock.sql then utllockt.sql to view the sessions in the system that are waiting for locks and the locks that they are waiting for.

- Query: V$LOCK, DBA_BLOCKERS, DBA_WAITERS, DBA_DDL_LOCKS, DBA_DML_LOCKS, DBA_LOCK
```
select *
from v$session
where username is not null
and sid  in ( select l.sid from v$lock l where l.block=1)
```

Managing Control Files

Managing Control File

Viewing Control Files Locations
```
SELECT VALUE FROM V$PARAMETER WHERE NAME = 'control_files'
SELECT NAME FROM V$CONTROLFILE;
```

Creating Initial Control Files
```
CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
                 /u02/oracle/prod/control02.ctl,
                 /u03/oracle/prod/control03.ctl)
```

Creating Additional Copies, Renaming, and Relocating Control Files
1. Shut down the database.
2. Copy an existing control file to a new location, using operating system commands.
3. Edit the CONTROL_FILES parameter
4. Restart the database.

Note: if you are using SPFILE, STARTUP NOMOUNT then use ALTER SYSTEM SET .. SCOPE=SPFILE command.

Creating New Control Files

1. Make a list of all datafiles and redo log files of the database.

2. Shut down the database.

3. Back up all datafiles and redo log files of the database.

4. STARTUP NOMOUNT

5. Use the CREATE CONTROLFILE statement:
```
CREATE CONTROLFILE
   SET DATABASE prod
   LOGFILE GROUP 1 ('/u01/oracle/prod/redo01_01.log',
                    '/u01/oracle/prod/redo01_02.log'),
           GROUP 2 ('/u01/oracle/prod/redo02_01.log',
                    '/u01/oracle/prod/redo02_02.log'),
           GROUP 3 ('/u01/oracle/prod/redo03_01.log',
                    '/u01/oracle/prod/redo03_02.log')
   RESETLOGS | NORESETLOGS
   DATAFILE '/u01/oracle/prod/system01.dbf' SIZE 3M,
            '/u01/oracle/prod/rbs01.dbs' SIZE 5M,
            '/u01/oracle/prod/users01.dbs' SIZE 5M,
            '/u01/oracle/prod/temp01.dbs' SIZE 5M
   MAXLOGFILES 50
   MAXLOGMEMBERS 3
   MAXLOGHISTORY 400
   MAXDATAFILES 200
   MAXINSTANCES 6
   ARCHIVELOG;
```

Specify the RESETLOGS clause if you have lost any redo log groups in addition to control files. In this case, you will need to recover from the loss of the redo logs (step 8). You must specify the RESETLOGS clause if you have renamed the database. Otherwise, select the NORESETLOGS clause.

Caution: The CREATE CONTROLFILE statement can potentially damage specified datafiles and redo log files. Omitting a filename can cause loss of the data in that file, or loss of access to the entire database.

6. Store a backup of the new control file on an offline storage device.

7. Edit the CONTROL_FILES initialization parameter

8. If you are renaming the database, edit the DB_NAME parameter in your instance parameter file.

9. Recover the database if necessary.
   - If the new control file was created using the NORESETLOGS clause, you can recover the database with complete, closed database recovery.

Oracle 10g DBA Fundamentals Example and Code Reference

- If the new control file was created using the RESETLOGS clause, you must specify USING BACKUP CONTROL FILE in your RECOVER command.

10. If you did not perform recovery, open the database normally.

    ALTER DATABASE OPEN;

    If you specified RESETLOGS when creating the control file:

    ALTER DATABASE OPEN RESETLOGS;

Handling Errors During CREATE CONTROLFILE

If Oracle Database sends you an error (usually error ORA-01173, ORA-01176, ORA-01177, ORA-01215, or ORA-01216) when you attempt to mount and open the database after creating a new control file, the most likely cause is that you omitted a file from the CREATE CONTROLFILE statement or included one that should not have been listed.

Checking for Missing Files after Creating Control Files

Check the alert log to see if the database has detected inconsistencies between the data dictionary and the control file.

- If a datafile exists in the data dictionary but not in the new control file, the database creates a placeholder entry in the control file under the name MISSINGnnnn, where nnnn is the file number in decimal. MISSINGnnnn is flagged in the control file as being offline and requiring media recovery.
    - o If the actual datafile corresponding to MISSINGnnnn is read-only or offline normal, then you can make the datafile accessible by renaming MISSINGnnnn to the name of the actual datafile.
    - o If MISSINGnnnn corresponds to a datafile that was not read-only or offline normal, you must drop the tablespace containing the datafile.

Backing Up Control Files

```
ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/control.bkp';
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

Manage the Size of Control Files

It is affected by MAXDATAFILES, MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, and MAXINSTANCES parameters in the CREATE DATABASE statement. Also it is affected by CONTROL_FILE_RECORD_KEEP_TIME

Displaying Control File Information

The following views display information about control files:

V$DATABASE	Displays database information from the control file
V$CONTROLFILE	Lists the names of control files
V$CONTROLFILE_RECORD_SECTION	Displays information about control file record sections

```
select TYPE, RECORD_SIZE
from V$CONTROLFILE_RECORD_SECTION
union
select 'total' as stype, sum(RECORD_SIZE) RECORD_SIZE
from V$CONTROLFILE_RECORD_SECTION
order by RECORD_SIZE
```

Managing the Redo Log Files

Controlling Archive Lag

- You can force all enabled redo log threads to switch their current logs at regular time intervals by setting the parameter ARCHIVE_LAG_TARGET
- The ARCHIVE_LAG_TARGET initialization parameter specifies the target of how many seconds of redo the standby could lose in the event of a primary shutdown or failure if the Oracle Data Guard environment is not configured in a no-data-loss mode.
- The ARCHIVE_LAG_TARGET parameter must be set to the same value in all instances of an Oracle Real Application Clusters environment. Failing to do so results in unpredictable behavior.

```
ALTER SYSTEM SET ARCHIVE_LAG_TARGET = 1800; — 30 minutes
```

Responding to Redo Log Failure

- Whenever LGWR cannot write to a member of a group, the database marks that member as INVALID and writes an error message to the LGWR trace file and to the database alert log to indicate the problem with the inaccessible files.

Managing Redo Log Groups and Members

Creating Redo Log Groups
```
ALTER DATABASE ADD LOGFILE [GROUP 10] ('/dbs/log1c.rdo', '/dbs/log2c.rdo') SIZE 500K
```

Creating Redo Log Members
```
ALTER DATABASE ADD LOGFILE MEMBER '/dbs/log2b.rdo' TO GROUP 2;
```

Relocating and Renaming Redo Log Members
```
SHUTDOWN IMMEDIATE
```
Copy the redo log files to the new location
```
CONNECT / as SYSDBA
STARTUP MOUNT
ALTER DATABASE
  RENAME FILE '/diska/logs/log1a.rdo', '/diska/logs/log2a.rdo'
          TO '/diskc/logs/log1c.rdo', '/diskc/logs/log2c.rdo';
ALTER DATABASE OPEN;
```

Dropping Log Groups
```
SELECT GROUP#, ARCHIVED, STATUS FROM V$LOG;
ALTER DATABASE DROP LOGFILE GROUP 3;
```

Dropping Redo Log Members
```
ALTER DATABASE DROP LOGFILE MEMBER '/oracle/dbs/log3c.rdo';
```

Forcing Log Switches
```
ALTER SYSTEM SWITCH LOGFILE
```

Verifying Blocks in Redo Log Files
- If you set the initialization parameter DB_BLOCK_CHECKSUM to TRUE, the database computes a checksum for each database block when it is written to disk, including each redo log block as it is being written to the current log.
- If corruption is detected in a redo log block while trying to archive it, the system attempts to read the block from another member in the group. If the block is corrupted in all members of the redo log group, then archiving cannot proceed.

Clearing a Redo Log File
```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```
- If the corrupt redo log file has not been archived, use the UNARCHIVED keyword in the statement.
```
ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 3;
```
- If you clear an unarchived redo log file, you should make another backup of the database.

Viewing Redo Log Information

V$LOG	Displays the redo log file information from the control file
V$LOGFILE	Identifies redo log groups and members and member status
V$LOG_HISTORY	Contains log history information

Managing Archived Redo Logs

General Tasks

Setting the Initial Database Archiving Mode
You set the initial archiving mode as part of database creation in the CREATE DATABASE statement.

Changing the Database Archiving Mode

1. SHUTDOWN IMMEDIATE

2. Back up the database

3. Optionally: set the archive log files destinations (next section)

4. STARTUP MOUNT

5. ALTER DATABASE ARCHIVELOG;

6. ALTER DATABASE OPEN;

7. SHUTDOWN IMMEDIATE

8. Back up the database

Specifying the Archive Destination

1. SHUTDOWN IMMEDIATE

2. STARTUP MOUNT

3. LOG_ARCHIVE_DEST_1 = 'LOCATION = /disk1/archive'

   LOG_ARCHIVE_DEST_2 = 'LOCATION = /disk2/archive'

   LOG_ARCHIVE_DEST_3 = 'SERVICE = standby1'

4. LOG_ARCHIVE_FORMAT=arc_%t_%s_%r.dbf

Understanding Archive Destination Status

```
column DEST_NAME format a40
select dest_name , status from  V$ARCHIVE_DEST
```

VALID	The user has properly initialized the destination, which is available for archiving.
INACTIVE	The user has not provided or has deleted the destination information.
ERROR	An error occurred creating or writing to the destination file; refer to error data.
FULL	Destination is full (no disk space).
DEFERRED	The user manually and temporarily disabled the destination.
DISABLED	The user manually and temporarily disabled the destination following an error; refer to error data.
BAD	A parameter error occurred; refer to error data.

Specifying the Availability State of a Destination

```
LOG_ARCHIVE_DEST_STATE_n = { alternate | reset | defer | enable }
```

enabled	the archive destination is valid and can be used (default)
defer	the destination is excluded from archiving operations until re-enabled.
alternate	the destination is not enabled but will become enabled if communications to another destination fails.

Optional and Mandatory Destinations

• Omitting the MANDATORY attribute for a destination is the same as specifying OPTIONAL

LOG_ARCHIVE_DEST_1 = 'LOCATION = /disk1/archive MANDATORY'

LOG_ARCHIVE_DEST_2 = 'LOCATION = /disk2/archive OPTIONAL'

```
Select dest_name, BINDING from V$ARCHIVE_DEST
```


Specifying the Minimum Number of Successful Destinations

The optional initialization parameter LOG_ARCHIVE_MIN_SUCCEED_DEST=n determines the minimum number of destinations to which the database must successfully archive a redo log group before it can reuse online log files. The default value is 1.

Select value from v$parameter where upper(name)='LOG_ARCHIVE_MIN_SUCCEED_DEST'

Rearchiving to a Failed Destination

• Use the REOPEN attribute of the LOG_ARCHIVE_DEST_n parameter to specify whether and when ARCn should attempt to rearchive to a failed destination following an error.

- If you do not specify the REOPEN keyword, ARCn will never reopen a destination following an error.

- REOPEN=n sets the minimum number of seconds before ARCn should try to reopen a failed destination.

```
LOG_ARCHIVE_DEST_2 = 'LOCATION = /disk2/archive OPTIONAL REOPEN=300'
```

Archive Maintenance

- To stop archive processes:

```
ALTER SYSTEM ARCHIVE LOG STOP;
```

Note: Stopping ARC*n* processes does not set the database in NOARCHIVELOG mode. When all groups of redo logs are used and not archived, the database will stall if it is in ARCHIVELOG mode.

- To manually archive Redo Log Files

```
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM ARCHIVE LOG START;
ALTER SYSTEM ARCHIVE LOG STOP;
ALTER SYSTEM ARCHIVE LOG ALL;
ALTER SYSTEM ARCHIVE LOG THREAD 1 ALL;
ALTER SYSTEM ARCHIVE LOG ALL TO 'C:\oracle\allarch';
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Controlling Trace Output Generated by the Archivelog Process

- You do this by setting the LOG_ARCHIVE_TRACE initialization parameter.

- You can combine tracing levels by specifying a value equal to the sum of the individual levels that you would like to trace.

Trace Level	Meaning
0	Disable archivelog tracing. (only in case or errors) This is the default.
1	Track archival of redo log file.
2	Track archival status for each archivelog destination.
4	Track archival operational phase.
8	Track archivelog destination activity.
16	Track detailed archivelog destination activity.
32	Track archivelog destination parameter modifications.
64	Track ARC*n* process state activity.
128	Track FAL (fetch archived log) server related activities.
256	Supported in a future release.
512	Tracks asynchronous LGWR activity.
1024	RFS physical client tracking.
2048	ARC*n*/RFS heartbeat tracking.
4096	Track real-time apply

```
ALTER SYSTEM SET LOG_ARCHIVE_TRACE=12;
```

Viewing Information About the Archived Redo Log

V$ARCHIVED_LOG	Displays historical archived log information from the control file. If you use a recovery catalog, the RC_ARCHIVED_LOG view contains similar information.
V$ARCHIVE_DEST	Describes the current instance, all archive destinations, and the current value, mode, and status of these destinations.
V$ARCHIVE_PROCESSES	Displays information about the state of the various archive processes for an instance.
V$BACKUP_REDOLOG	Contains information about any backups of archived logs. If you use a recovery catalog, the RC_BACKUP_REDOLOG contains similar information.
V$LOG	Displays all redo log groups for the database and indicates which need to be archived.
V$LOG_HISTORY	Contains log history information such as which logs have been archived and the SCN range for each archived log.

```
-- redolog files
SELECT GROUP#          , SEQUENCE#, MEMBERS,ARCHIVED        ,STATUS
FROM V$LOG


-- archivelog mode
select LOG_MODE from V$DATABASE ;
```

```
-- archived log files
select name , round(BLOCKS*BLOCK_SIZE/1024/1024,2) MB, STATUS, BACKUP_COUNT
from  V$ARCHIVED_LOG ;

-- archive log destination
select DEST_NAME, STATUS , BINDING         ,DESTINATION    , REOPEN_SECS, ERROR
from V$ARCHIVE_DEST
where status <> 'INACTIVE'

-- archiver processes
select PROCESS, STATUS , LOG_SEQUENCE, STATE
from V$ARCHIVE_PROCESSES
where status <> 'STOPPED'
```

Managing Tablespaces

Creating Tablespaces

Creating a Locally Managed Tablespace

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
 EXTENT MANAGEMENT LOCAL AUTOALLOCATE;

CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
 EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K; -- default 1M
```

Specifying Segment Space Management in Locally Managed Tablespaces

```
CREATE TABLESPACE lmtbsb DATAFILE '/u02/oracle/data/lmtbsb01.dbf' SIZE 50M
 EXTENT MANAGEMENT LOCAL
 SEGMENT SPACE MANAGEMENT AUTO;
```

Altering a Locally Managed Tablespace

Adding a datafile

```
ALTER TABLESPACE lmtbsb
 ADD DATAFILE '/u02/oracle/data/lmtbsb02.dbf' SIZE 1M;
```

Altering tablespace availability

Specify TEMPORARY only when you cannot take the tablespace offline normally. In this case, only the files taken offline because of errors need to be recovered before the tablespace can be brought online. Specify IMMEDIATE only after trying both the normal and temporary settings.

```
ALTER TABLESPACE users OFFLINE NORMAL| TEMPORARY| IMMEDIATE
```

Making a Tablespace Read-Only

```
ALTER TABLESPACE flights READ ONLY;
```

This command waits for all transactions started before you issued the ALTER TABLESPACE statement to either commit or rollback. To identify the blocking transactions, check the SQL Address from the following query:

```
SELECT SQL_TEXT, SADDR
 FROM V$SQLAREA,V$SESSION
 WHERE V$SQLAREA.ADDRESS = V$SESSION.SQL_ADDRESS
   AND SQL_TEXT LIKE 'alter tablespace%';
```

Check the SCN number corresponding to the SADDR obtained from the previous statement. All transactions displayed in the following query and of SCN numbers greater than the obtained SCN number are waited for by the statement:

```
SELECT SES_ADDR, START_SCNB
 FROM V$TRANSACTION
 ORDER BY START_SCNB;
```

Making a Read-Only Tablespace Writable

```
ALTER TABLESPACE flights READ WRITE;
```

Delaying the Opening of Datafiles in Read-Only Tablespaces

When substantial portions of a very large database are stored in read-only tablespaces that are located on slow-access devices or hierarchical storage, you should consider setting the **READ_ONLY_OPEN_DELAYED** initialization parameter

Oracle 10g DBA Fundamentals Example and Code Reference

to TRUE. This speeds certain operations, primarily opening the database, by causing datafiles in read-only tablespaces to be accessed for the first time only when an attempt is made to read data stored within them.

Setting READ_ONLY_OPEN_DELAYED=TRUE has the following side-effects:

- Read-Only tablespaces are not recognized by the following satements: ALTER SYSTEM CHECK DATAFILES, ALTER TABLESPACE…ONLINE, ALTER DATABASE DATAFILE…ONLINE

- Read-Only tablespaces are not recognized by the following views: V$RECOVER_FILE, V$BACKUP, V$DATAFILE_HEADER, V$DATAFILE and V$RECOVER_LOG.

- ALTER DATABASE NOARCHIVELOG does not access read-only files. It proceeds even if there is a read-only file that requires recovery.

Renaming Tablespaces

```
ALTER TABLESPACE users RENAME TO usersts;
```

When you rename a tablespace the database updates all references to the tablespace name in the data dictionary, control file, and (online) datafile headers.

Dropping Tablespaces

```
DROP TABLESPACE users
DROP TABLESPACE users INCLUDING CONTENTS;
DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES;
```

If an operating system error prevents the deletion of a file, the DROP TABLESPACE statement still succeeds, but a message describing the error is written to the alert log.

Managing the SYSAUX Tablespace

Monitoring Occupants of the SYSAUX Tablespace

```
set pagesize 30
column OCCUPANT_NAME format a35
column SCHEMA_NAME format a25
Select OCCUPANT_NAME  , SCHEMA_NAME, SPACE_USAGE_KBYTES/1024 MB
from V$SYSAUX_OCCUPANTS
order by 3
```

Moving Occupants Out Of or Into the SYSAUX Tablespace

You can use the move procedure for that component, as specified in the V$SYSAUX_OCCUPANTS view, to perform the move:

```
column MOVE_PROCEDURE format a40
select OCCUPANT_NAME  , MOVE_PROCEDURE
from V$SYSAUX_OCCUPANTS
order by 1
```

Diagnosing and Repairing Locally Managed Tablespace Problems

Viewing BMB Information

block 1 -> File header
block 2 ->file space bitmap header
block 3 -> the first bitmap block

```
SELECT FILE#, NAME FROM V$DATAFILE;

ALTER SYSTEM DUMP DATAFILE 5 BLOCK 3;

SELECT VALUE
FROM V$PARAMETER
WHERE UPPER(NAME)='USER_DUMP_DEST'
```

When you refer to the generated trace file, convert the hexadecimal value into binary. Then swap every byte. The result is the file bitmap of allocated extents. For testing purpose, you can allocated more extents using the command:

```
alter table demotab allocate extent;
```

Verify the BMB

To verify the integrity of segments created in tablespaces that have automatic segment space management `enabled`:
```
begin
 DBMS_SPACE_ADMIN.ASSM_SEGMENT_VERIFY (
  SEGMENT_OWNER =>'HR',
  SEGMENT_NAME  =>'NAMES',
  SEGMENT_TYPE  =>'TABLE',
  PARTITION_NAME=>NULL );
end;
```

To verify the integrity of the file bitmaps as well perform checks on all the segments
```
begin
DBMS_SPACE_ADMIN.ASSM_TABLESPACE_VERIFY (
   tablespace_name =>'USERS',
   ts_option       =>DBMS_SPACE_ADMIN.TS_VERIFY_DEEP,
   segment_option  =>DBMS_SPACE_ADMIN.SEGMENT_VERIFY_DEEP);
end;
```

Refer to the generated file `sid_ora_process_id.trc` in the location of `USER_DUMP_DEST` :
```
SELECT VALUE
FROM V$PARAMETER
WHERE UPPER(NAME)='USER_DUMP_DEST'
```

Fixing Bitmap When Allocated Blocks are Marked Free (No Overlap)

The TABLESPACE_VERIFY procedure discovers that a segment has allocated blocks that are marked free in the bitmap, but no overlap between segments is reported.

In this scenario, perform the following tasks:

1. Call the SEGMENT_DUMP procedure to dumps the segment header and bitmap blocks of a specific segment to the location of USER_DUMP_DEST
```
DECLARE
 v_fn NUMBER;
 v_bheader NUMBER;
BEGIN
 -- retreive corresponding file #
 SELECT HEADER_FILE , HEADER_BLOCK
 INTO  v_fn, v_bheader
 FROM   DBA_SEGMENTS
 WHERE  OWNER='HR' AND SEGMENT_NAME='NAMES';
DBMS_SPACE_ADMIN.SEGMENT_DUMP (
   TABLESPACE_NAME => 'USERS',
   HEADER_RELATIVE_FILE =>v_fn,
   HEADER_BLOCK =>v_bheader,
   DUMP_OPTION =>DBMS_SPACE_ADMIN.SEGMENT_DUMP_EXTENT_MAP -- DEFAULT
   );
END;
```

2. For each range, call the TABLESPACE_FIX_BITMAPS procedure with the TABLESPACE_EXTENT_MAKE_USED option to mark the space as used.
```
DECLARE
 v_fn NUMBER;
 v_bheader NUMBER;
BEGIN
 -- retreive corresponding file #
 SELECT HEADER_FILE , HEADER_BLOCK
 INTO  v_fn, v_bheader
 FROM   DBA_SEGMENTS
 WHERE  OWNER='HR' AND SEGMENT_NAME='NAMES';

DBMS_SPACE_ADMIN.TABLESPACE_FIX_BITMAPS (
   tablespace_name =>'HR',
   dbarange_relative_file => v_fn,
   dbarange_begin_block=>,
   dbarange_end_block=>,
   fix_option =>DBMS_SPACE_ADMIN.TABLESPACE_EXTENT_MAKE_USED);
end;
```

3. Call TABLESPACE_REBUILD_QUOTAS to rebuild quotas.
```
BEGIN
 DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_QUOTAS (tablespace_name =>'USERS');
END;
```

Dropping a Corrupted Segment

You cannot drop a segment because the bitmap has segment blocks marked "free". The system has automatically marked the segment corrupted. In this scenario, perform the following tasks:

1. Call the SEGMENT_VERIFY procedure with the SEGMENT_VERIFY_EXTENTS_GLOBAL option. If no overlaps are reported, then proceed with steps 2 through 5.

```
DECLARE
 v_fn NUMBER;
 v_bheader NUMBER;
BEGIN
 -- retreive corresponding file #
 SELECT HEADER_FILE , HEADER_BLOCK
 INTO   v_fn, v_bheader
 FROM   DBA_SEGMENTS
 WHERE  OWNER='HR' AND SEGMENT_NAME='NAMES';

-- check the consistency of the segment extent map with the tablespace file bitmaps
DBMS_SPACE_ADMIN.SEGMENT_VERIFY (
   tablespace_name =>'HR',
   header_relative_file => v_fn,
   header_block =>v_bheader,
   verify_option =>DBMS_SPACE_ADMIN.SEGMENT_VERIFY_EXTENTS_GLOBAL );
end;
```

2. Call the SEGMENT_DUMP procedure to dump the DBA ranges allocated to the segment.
3. For each range, call TABLESPACE_FIX_BITMAPS with the TABLESPACE_EXTENT_MAKE_FREE option to mark the space as free.
4. Call SEGMENT_DROP_CORRUPT to drop the SEG$ entry.
5. Call TABLESPACE_REBUILD_QUOTAS to rebuild quotas.

Fixing Bitmap Where Overlap is Reported

The TABLESPACE_VERIFY procedure reports some overlapping. Some of the real data must be sacrificed based on previous internal errors. After choosing the object to be sacrificed, in this case say, table t1, perform the following tasks:

1. Make a list of all objects that t1 overlaps.
2. Drop table t1. If necessary, follow up by calling the SEGMENT_DROP_CORRUPT procedure.
3. Call the SEGMENT_VERIFY procedure on all objects that t1 overlapped. If necessary, call the TABLESPACE_FIX_BITMAPS procedure to mark appropriate bitmap blocks as used.
4. Rerun the TABLESPACE_VERIFY procedure to verify that the problem is resolved.

Correcting Media Corruption of Bitmap Blocks

A set of bitmap blocks has media corruption.

1. Call the TABLESPACE_REBUILD_BITMAPS procedure, either on all bitmap blocks, or on a single block if only one is corrupt.
2. Call the TABLESPACE_REBUILD_QUOTAS procedure to rebuild quotas.
3. Call the TABLESPACE_VERIFY procedure to verify that the bitmaps are consistent.

Migrating from a Dictionary-Managed to a Locally Managed Tablespace

```
EXEC DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL ('TBS1');
```

Transporting Tablespaces Between Databases

Limitations on Transportable Tablespace Use

- The source and target database must use the same character set and national character set.
- Objects with underlying objects (such as materialized views) or contained objects (such as partitioned tables) are not transportable unless all of the underlying or contained objects are in the tablespace set.
- You cannot transport the SYSTEM tablespace or objects owned by the user SYS. This means that you cannot use TTS for PL/SQL, triggers, or views. These would have to be moved with export.
- You cannot transport a table with a materialized view unless the mview is in the transport set you create.
- You cannot transport a partition of a table without transporting the entire table.

Transporting Tablespaces Between Databases

1. Check *endian* format of both platforms.
    For cross-platform transport, check the endian format of both platforms by querying the V$TRANSPORTABLE_PLATFORM view.

    You can find out your own platform name:
    ```
    select platform_name from v$database
    ```

2. Pick a self-contained set of tablespaces.

    The following statement can be used to determine whether tablespaces sales_1 and sales_2 are self-contained, with referential integrity constraints taken into consideration:

    ```
    DBMS_TTS.TRANSPORT_SET_CHECK( TS_LIST =>'sales_1,sales_2', INCL_CONSTRAINTS =>TRUE, FULL_CHECK
    =>TRUE )
    ```

    Note: You must have been granted the EXECUTE_CATALOG_ROLE role (initially signed to SYS) to execute this procedure.

    You can see all violations by selecting from the TRANSPORT_SET_VIOLATIONS view. If the set of tablespaces is self-contained, this view is empty.

3. Generate a transportable tablespace set.

    3.1. Make all tablespaces in the set you are copying read-only.

    3.2. Export the metadata describing the objects in the tablespace(s)
    ```
    EXPDP system/password DUMPFILE=expdat.dmp DIRECTORY=dpump_dir TRANSPORT_TABLESPACES =
    sales_1,sales_2
    TRANSPORT_FULL_CHECK=Y
    ```

    3.3. If you want to convert the tablespaces in the source database, use the RMAN
    ```
    RMAN TARGET /
    CONVERT TABLESPACE sales_1,sales_2
    TO PLATFORM 'Microsoft Windows NT'
    FORMAT '/temp/%U'
    ```

4. Transport the tablespace set.

    Transport both the datafiles and the export file of the tablespaces to a place accessible to the target database.

5. Convert tablespace set, if required, in the destination database.

    ```
    Use RMAN as follows:
    RMAN> CONVERT DATAFILE
    '/hq/finance/work/tru/tbs_31.f',
    '/hq/finance/work/tru/tbs_32.f',
    '/hq/finance/work/tru/tbs_41.f'
    TO PLATFORM="Solaris[tm] OE (32-bit)"
    FROM PLATFORM="HP TRu64 UNIX"
    DBFILE_NAME_CONVERT=
    "/hq/finance/work/tru/", "/hq/finance/dbs/tru"
    PARALLELISM=5
    ```

    Note: The source and destination platforms are optional.

    Note: By default, Oracle places the converted files in the Flash Recovery Area, without changing the datafile names.

    Note: If you have CLOB data on a small-endian system in an Oracle database version before 10g and with a varying-width character set and you are transporting to a database in a big-endian system, the CLOB data must be converted in the destination database. RMAN does not handle the conversion during the CONVERT phase. However, Oracle database automatically handles the conversion while accessing the CLOB data.

    If you want to eliminate this run-time conversion cost from this automatic conversion, you can issue the CREATE TABLE AS SELECT command before accessing the data.

6. Plug in the tablespace.
    ```
    IMPDP system/password DUMPFILE=expdat.dmp DIRECTORY=dpump_dir
    TRANSPORT_DATAFILES=
    /salesdb/sales_101.dbf,
    /salesdb/sales_201.dbf
    REMAP_SCHEMA=(dcranney:smith) REMAP_SCHEMA=(jfee:williams)
    ```

    If required, put the tablespace into READ WRITE mode.

Using Transportable Tablespaces: Scenarios

Transporting and Attaching Partitions for Data Warehousing

1. In a staging database, you create a new tablespace and make it contain the table you want to transport. It should have the same columns as the destination partitioned table.

2. Create an index on the same columns as the local index in the partitioned table.

3. Transport the tablespace to the data warehouse.

4. In the data warehouse, add a partition to the table.
    ```
    ALTER TABLE sales ADD PARTITION jul98 VALUES LESS THAN (1998, 8, 1)
    ```

5. Attach the transported table to the partitioned table by exchanging it with the new partition:
    ```
    ALTER TABLE sales EXCHANGE PARTITION jul98 WITH TABLE jul_sales
    INCLUDING INDEXES WITHOUT VALIDATION
    ```

Publishing Structured Data on CDs

A data provider can load a tablespace with data to be published, generate the transportable set, and copy the transportable set to a CD. When customers receive this CD, they can plug it into an existing database without having to copy the datafiles from the CD to disk storage.

Note: In this case, it is highly recommended to set the `READ_ONLY_OPEN_DELAYED` initialization parameter to `TRUE`.

Mounting the Same Tablespace Read-Only on Multiple Databases
You can use transportable tablespaces to mount a tablespace read-only on multiple databases.

Archiving Historical Data Using Transportable Tablespaces

Using Transportable Tablespaces to Perform TSPITR

Viewing Tablespace Information

V$TABLESPACE	Name and number of all tablespaces from the control file.
DBA_TABLESPACES	Descriptions of all tablespaces.
DBA_TABLESPACE_GROUPS	Displays the tablespace groups and the tablespaces that belong to them.
DBA_SEGMENTS	Information about segments within all (or user accessible) tablespaces.
DBA_EXTENTS	Information about data extents within all (or user accessible) tablespaces.
DBA_FREE_SPACE	Information about free extents within all (or user accessible) tablespaces.
V$DATAFILE	Information about all datafiles, including tablespace number of owning tablespace.
V$TEMPFILE	Information about all tempfiles, including tablespace number of owning tablespace.
DBA_DATA_FILES	Shows files (datafiles) belonging to tablespaces.
DBA_TEMP_FILES	Shows files (tempfiles) belonging to temporary tablespaces.
V$TEMP_EXTENT_MAP	Information for all extents in all locally managed temporary tablespaces.
V$TEMP_EXTENT_POOL	For locally managed temporary tablespaces: the state of temporary space cached and used for by each instance.
V$TEMP_SPACE_HEADER	Shows space used/free for each tempfile.
DBA_USERS	Default and temporary tablespaces for all users.
DBA_TS_QUOTAS	Lists tablespace quotas for all users.
V$SORT_SEGMENT	Information about every sort segment in a given instance. The view is only updated when the tablespace is of the TEMPORARY type.
V$TEMPSEG_USAGE	Describes temporary (sort) segment usage by user for temporary or permanent tablespaces.

The following are just a few examples of using some of these views:

```
-- list the names, sizes, and associated tablespaces of a database
-- group by datafiles
Set linesize 100
column file_name format a40
SELECT  FILE_NAME, bytes/1024/1024 MB, TABLESPACE_NAME
FROM DBA_DATA_FILES
ORDER BY TABLESPACE_NAME;
-- group by tablespaces
SELECT  TABLESPACE_NAME , sum(bytes)/1024/1024 TOTAL_MB
FROM DBA_DATA_FILES
GROUP BY TABLESPACE_NAME

-- space used and free in tablespaces
SELECT S.TABLESPACE_NAME, ROUND(SUM(S.BYTES)/1024/1024) USED_MB, T.TOTAL_MB, ROUND((1-
SUM(S.BYTES)/1024/1024/T.TOTAL_MB)*100,2) FREE_PCT
FROM dba_segments S, (SELECT  TABLESPACE_NAME , sum(bytes)/1024/1024 TOTAL_MB
FROM DBA_DATA_FILES
                   GROUP BY TABLESPACE_NAME ) T
WHERE S.TABLESPACE_NAME = T.TABLESPACE_NAME
GROUP BY S.TABLESPACE_NAME, T.TOTAL_MB
ORDER BY FREE_PCT;
```

Oracle 10g DBA Fundamentals Example and Code Reference

```
-- statistics about free extents and coalescing activity for each tablespace in the database
set linesize 100
column TABLESPACE format a15
SELECT TABLESPACE_NAME "TABLESPACE", FILE_ID,
    COUNT(*)    "FREE_EXT",
    MAX(blocks) "MAXIMUM",
    MIN(blocks) "MINIMUM",
    AVG(blocks) "AVERAGE",
    SUM(blocks) "TOTAL"
FROM DBA_FREE_SPACE
GROUP BY TABLESPACE_NAME, FILE_ID;
```

Managing Datafiles and Tempfiles

The DB_FILES parameter

The static DB_FILES initialization parameter indicates the amount of SGA space to reserve for datafile information and thus, the maximum number of datafiles that can be created for the instance.

Set its value to amount appropriate to your database.

Creating and Modifying Datafiles

Adding Datafiles

```
ALTER TABLESPACE users
 ADD DATAFILE '/u02/oracle/rbdb1/users03.dbf' SIZE 10M
 AUTOEXTEND ON NEXT 512K MAXSIZE 250M;
```

To disable the automatic extension for the datafile:

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/users03.dbf'
    AUTOEXTEND OFF;
```

Manually Resizing a Datafile

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/stuff01.dbf'
   RESIZE 100M;
```

Bringing Datafiles Online or Taking Offline in ARCHIVELOG Mode

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/stuff01.dbf' ONLINE | OFFLINE;
```

Taking Datafiles Offline in NOARCHIVELOG Mode

```
ALTER DATABASE DATAFILE '/u02/oracle/rbdb1/users03.dbf' OFFLINE FOR DROP;
```

Rename and Re-locate the datafiles

1. `ALTER TABLESPACE users OFFLINE NORMAL;`
2. Rename the datafiles using the operating system
3. `ALTER TABLESPACE users`
    ```
     RENAME DATAFILE '/u02/oracle/rbdb1/user1.dbf',
                     '/u02/oracle/rbdb1/user2.dbf'
                 TO '/u02/oracle/rbdb1/users01.dbf',
                     '/u02/oracle/rbdb1/users02.dbf';
    ```
4. `Back` up the database. After making any structural changes to a database, always perform an immediate and complete backup.

Or

1. Ensure that the database is mounted but closed.
2. Copy the datafiles to be renamed to their new locations and new names, using the operating system.   You can copy the files using the DBMS_FILE_TRANSFER package Use ALTER DATABASE to rename the file pointers in the database control file.
    ```
         ALTER DATABASE
            RENAME FILE '/u02/oracle/rbdb1/sort01.dbf',
                         '/u02/oracle/rbdb1/user3.dbf'
                     TO '/u02/oracle/rbdb1/temp01.dbf',
                         '/u02/oracle/rbdb1/users03.dbf;
    ```
3. Back up the database.

Oracle 10g DBA Fundamentals Example and Code Reference

Dropping Datafiles

Restrictions:

- If a datafile is not empty, it cannot be dropped.
- You cannot drop the first or only datafile in a tablespace. This means that DROP DATAFILE cannot be used with a bigfile tablespace.
- You cannot drop datafiles in a read-only tablespace.
- You cannot drop datafiles in the SYSTEM tablespace.
- If a datafile in a locally managed tablespace is offline, it cannot be dropped.

```
ALTER TABLESPACE example DROP DATAFILE '...';
ALTER TABLESPACE lmtemp DROP TEMPFILE '...';
ALTER DATABASE TEMPFILE '...' DROP INCLUDING DATAFILES;
```

Verifying Data Blocks in Datafiles

Make sure that DB_BLOCK_CHECKSUM is set to TRUE.

Mapping Files to Physical Devices

This requires library provided by third-party vendor. Refer to documentation for further details " Oracle® Database Administrator's Guide".

Managing Undo Tablespace

Sizing the Undo Tablespace

You can size the undo tablespace appropriately either by using automatic extension of the undo tablespace or by using the Undo Advisor for a fixed sized tablespace.

Setting the Undo Retention Period

This parameter is taken into consideration by Oracle when:

- The undo tablespace has the AUTOEXTEND option enabled
- You want to set undo retention for LOBs
- You want retention guarantee

In all other cases, this parameter is ignored, and the database automatically tunes for maximum undo retention.

```
ALTER SYSTEM SET UNDO_RETENTION = 2400; -- in seconds
```

Retention Guarantee

```
ALTER TABLESPACE <> RETENTION GUARANTEE|NOGUARANTEE
```

Managing Undo Tablespace

```
CREATE UNDO TABLESPACE undotbs_02
    DATAFILE '/u01/oracle/rbdb1/undo0201.dbf' SIZE 2M REUSE AUTOEXTEND ON;

ALTER TABLESPACE undotbs_01
    ADD DATAFILE '/u01/oracle/rbdb1/undo0102.dbf' AUTOEXTEND ON NEXT 1M
        MAXSIZE UNLIMITED;
```

You can use the ALTER DATABASE...DATAFILE statement to resize or extend a datafile.

```
DROP TABLESPACE undotbs_01;

ALTER SYSTEM SET UNDO_TABLESPACE = undotbs_02;
```

Viewing Information about Undo

UNDOSTAT	Contains statistics for monitoring and tuning undo space. Use this view to help estimate the amount of undo space required for the current workload.
V$ROLLSTAT	For automatic undo management mode, information reflects behavior of the undo segments in the undo tablespace
V$TRANSACTION	Contains undo segment information
DBA_UNDO_EXTENTS	Shows the status and size of each extent in the undo tablespace.
DBA_HIST_UNDOSTAT	Contains statistical snapshots of V$UNDOSTAT information.

```
-- Statistics are available for undo space consumption, transaction concurrency, the tuning of undo
retention, and the length and SQL ID of long-running queries in the instance. Each row in the view
contains statistics collected in the instance for a ten-minute interval.


SELECT TO_CHAR(BEGIN_TIME, 'HH24:MI:SS') BEGIN_TIME,
          TO_CHAR(END_TIME, 'HH24:MI:SS') END_TIME,
          UNDOTSN, UNDOBLKS, TXNCOUNT, MAXCONCURRENCY AS "MAXCON"
FROM v$UNDOSTAT;
```

Managing Schema Objects

Creating Multiple Tables and Views in a Single Operation

- If an individual table, view or grant fails, the entire statement is rolled back
- CREATE SCHEMA statement can include only CREATE TABLE, CREATE VIEW, and GRANT statements
- The CREATE SCHEMA statement does not support Oracle Database extensions to the ANSI CREATE TABLE and CREATE VIEW statements, including the STORAGE clause.

```
CREATE SCHEMA AUTHORIZATION scott
    CREATE TABLE dept (
        deptno NUMBER(3,0) PRIMARY KEY,
        dname VARCHAR2(15),
        loc VARCHAR2(25))
    CREATE TABLE emp (
        empno NUMBER(5,0) PRIMARY KEY,
        ename VARCHAR2(15) NOT NULL,
        job VARCHAR2(10),
        mgr NUMBER(5,0),
        hiredate DATE DEFAULT (sysdate),
        sal NUMBER(7,2),
        comm NUMBER(7,2),
        deptno NUMBER(3,0) NOT NULL
        CONSTRAINT dept_fkey REFERENCES dept)
    CREATE VIEW sales_staff AS
        SELECT empno, ename, sal, comm
        FROM emp
        WHERE deptno = 30
        WITH CHECK OPTION CONSTRAINT sales_staff_cnst
        GRANT SELECT ON sales_staff TO human_resources;
```

Analyzing Tables, Indexes, and Clusters

You must use the ANALYZE statement (rather than DBMS_STATS) for statistics collection not related to the cost-based optimizer, such as:

- To use the VALIDATE or LIST CHAINED ROWS clauses
- To collect information on freelist blocks

Using DBMS_STATS to Collect Table and Index Statistics
The following DBMS_STATS procedures enable the gathering of optimizer statistics:

- o GATHER_INDEX_STATS
- o GATHER_TABLE_STATS
- o GATHER_SCHEMA_STATS
- o GATHER_DATABASE_STATS

Validating Tables, Indexes, Clusters, and Materialized Views
- If a table, index, or cluster is corrupt, you should drop it and re-create it.

```
ANALYZE TABLE emp VALIDATE STRUCTURE;
```

The following statement validates the emp table and all associated indexes:

```
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE;
```

The following statement validates the emp table and all associated indexes online:

```
ANALYZE TABLE emp VALIDATE STRUCTURE CASCADE ONLINE;
```

Listing Chained Rows of Tables and Clusters
- To create the table to accept data returned by an ANALYZE…LIST CHAINED ROWS statement, execute the UTLCHAIN.SQL or UTLCHN1.SQL script.

```
ANALYZE CLUSTER emp_dept LIST CHAINED ROWS INTO CHAINED_ROWS;
```

Eliminating Migrated or Chained Rows in a Table
- You can use the information in the CHAINED_ROWS table to reduce or eliminate migrated and chained rows in an existing table. Use the following procedure.

1. Use the ANALYZE statement to collect information about migrated and chained rows.

See previous section commands

2. Query the output table:

```
SELECT *
FROM CHAINED_ROWS
WHERE TABLE_NAME = 'ORDER_HIST';


OWNER_NAME  TABLE_NAME  CLUST... HEAD_ROWID          TIMESTAMP
----------  ----------  -----... ------------------  ---------
SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAA  04-MAR-96
SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAB  04-MAR-96
SCOTT       ORDER_HIST       ... AAAAluAAHAAAAA1AAC  04-MAR-96
```

The output lists all rows that are either migrated or chained.

If the output table shows that you have many migrated or chained rows, then you can eliminate migrated rows by continuing through the following steps:

3. Create an intermediate table with the same columns as the existing table to hold the migrated and chained rows:

```
CREATE TABLE int_order_hist
   AS SELECT *
      FROM order_hist
      WHERE ROWID IN
         (SELECT HEAD_ROWID
            FROM CHAINED_ROWS
            WHERE TABLE_NAME = 'ORDER_HIST');
```

4. Delete the migrated and chained rows from the existing table:

```
DELETE FROM order_hist
   WHERE ROWID IN
      (SELECT HEAD_ROWID
         FROM CHAINED_ROWS
         WHERE TABLE_NAME = 'ORDER_HIST');
```

5. Insert the rows of the intermediate table into the existing table:

```
INSERT INTO order_hist
   SELECT *
   FROM int_order_hist;
```

6. Drop the intermediate table:

```
DROP TABLE int_order_history;
```

7. Delete the information collected in step 1 from the output table:
```
DELETE FROM CHAINED_ROWS
    WHERE TABLE_NAME = 'ORDER_HIST';
```

8. Use the ANALYZE statement again, and query the output table.

Any rows that appear in the output table are chained. You can eliminate chained rows only by increasing your data block size. It might not be possible to avoid chaining in all situations. Chaining is often unavoidable with tables that have a LONG column or large CHAR or VARCHAR2 columns.

Managing Integrity Constraints

```
CREATE TABLE emp (
    empno NUMBER(5) PRIMARY KEY DISABLE,   . . . ;

ALTER TABLE emp ADD PRIMARY KEY (empno) DISABLE;

ALTER TABLE dept DISABLE CONSTRAINT dname_ukey;

ALTER TABLE dept
    DISABLE PRIMARY KEY KEEP INDEX,
    DISABLE UNIQUE (dname, loc) KEEP INDEX;

ALTER TABLE dept
    ENABLE NOVALIDATE CONSTRAINT dname_ukey;

ALTER TABLE dept
    ENABLE NOVALIDATE PRIMARY KEY,
    ENABLE NOVALIDATE UNIQUE (dname, loc);

ALTER TABLE dept MODIFY CONSTRAINT dname_key VALIDATE;

ALTER TABLE dept MODIFY PRIMARY KEY ENABLE NOVALIDATE;

ALTER TABLE dept ENABLE CONSTRAINT dname_ukey;

ALTER TABLE dept
    ENABLE PRIMARY KEY,
    ENABLE UNIQUE (dname, loc);

ALTER TABLE dept RENAME CONSTRAINT dname_ukey TO dname_unikey;

ALTER TABLE dept DROP UNIQUE (dname, loc);

ALTER TABLE emp
    DROP PRIMARY KEY KEEP INDEX,
    DROP CONSTRAINT dept_fkey;
```

Reporting Constraint Exceptions
You can create an exception table by executing the UTLEXCPT.SQL script or the UTLEXPT1.SQL script.
```
ALTER TABLE dept ENABLE PRIMARY KEY EXCEPTIONS INTO EXCEPTIONS;
SELECT * FROM EXCEPTIONS;
```

Viewing Constraint Information

```
DBA_CONSTRAINTS
DBA_CONS_COLUMNS
```

More Operations

Modifying Table Structure
```
ALTER TABLE PARTS ADD (part_location VARCHAR2(20), part_bin VARCHAR2(30) );
ALTER TABLE parts ADD (photo BLOB)
```

Oracle 10g DBA Fundamentals Example and Code Reference

```
LOB (photo) STORE AS lob_parts_photo
(TABLESPACE parts_lob_tbs);
ALTER TABLE parts MODIFY LOB (photo) (STORAGE(FREELISTS 2));
ALTER TABLE parts MODIFY LOB (photo) (PCTVERSION 50);
ALTER TABLE parts DROP (part_location);
ALTER TABLE parts DROP (part_location, part_bin);
ALTER TABLE parts RENAME COLUMN part_location TO part_loc;
```

Truncating Tables and Clusters

```
TRUNCATE CLUSTER emp_dept REUSE STORAGE;
```

Enabling and Disabling Triggers

```
ALTER TABLE inventory ENABLE ALL TRIGGERS;
ALTER TRIGGER reorder ENABLE;
```

Moving Table

```
ALTER TABLE parts move TABLESPACE parts_new_tbs PCTFREE 10 PCTUSED 60;
```

Renaming Objects

```
RENAME sales_staff TO dept_30;
```

Managing Indexes

create index: Function-Based Index

```
CREATE INDEX fb_upper_last_name_emp ON emp_info (UPPER(last_name) );
```

create index: Global Partitioned Indexes

```
CREATE INDEX ix_part_my_tab_01
 ON store_sales (invoice_number)
 GLOBAL PARTITION BY RANGE (invoice_number)
( PARTITION part_001 VALUES LESS THAN (1000),
  PARTITION part_002 VALUES LESS THAN (10000),
  PARTITION part_003 VALUES LESS THAN (MAXVALUE) );

CREATE INDEX ix_part_my_tab_02
 ON store_sales (store_id, time_id)
 GLOBAL PARTITION BY RANGE (store_id, time_id)
 (PARTITION PART_001 VALUES LESS THAN (1000, TO_DATE('04-01-2003','MM-DD-YYYY') )
   TABLESPACE partition_001 STORAGE (INITIAL 100M NEXT 200M PCTINCREASE 0),
  PARTITION part_002 VALUES LESS THAN (1000, TO_DATE('07-01-2003','MM-DD-YYYY') )
   TABLESPACE partition_002 STORAGE (INITIAL 200M NEXT 400M PCTINCREASE 0),
  PARTITION part_003 VALUES LESS THAN (maxvalue, maxvalue)
   TABLESPACE partition_003 );
```

create index: Local Partitioned Indexes

```
CREATE INDEX ix_part_my_tab_01
 ON my_tab (col_one, col_two, col_three)
 LOCAL (PARTITION tbs_part_01 TABLESPACE part_tbs_01,
        PARTITION tbs_part_02 TABLESPACE part_tbs_02,
        PARTITION tbs_part_03 TABLESPACE part_tbs_03,
        PARTITION tbs_part_04 TABLESPACE part_tbs_04);

CREATE INDEX ix_part_my_tab_01
 ON my_tab (col_one, col_two, col_three)
 LOCAL STORE IN (part_tbs_01, part_tbs_02, part_tbs_03, part_tbs_04);

CREATE INDEX ix_part_my_tab_01
 ON my_tab (col_one, col_two, col_three)
 LOCAL STORE IN
 ( part_tbs_01 STORAGE (INITIAL 10M NEXT 10M MAXEXTENTS 200),
   part_tbs_02,
   part_tbs_03 STORAGE (INITIAL 100M NEXT 100M MAXEXTENTS 200),
   part_tbs_04 STORAGE (INITIAL 1000M NEXT 1000M MAXEXTENTS 200));
```

create index: Local Subpartitioned Indexes

```
CREATE INDEX sales_ix
ON store_sales(time_id, store_id)
STORAGE (INITIAL 1M MAXEXTENTS UNLIMITED) LOCAL
(PARTITION q1_2003,
 PARTITION q2_2003,
 PARTITION q3_2003
 (SUBPARTITION pq3200301, SUBPARTITION pq3200302,
  SUBPARTITION pq3200303, SUBPARTITION pq3200304,
  SUBPARTITION pq3200305),
```

```
PARTITION q4_2003
(SUBPARTITION pq4200301 TABLESPACE tbs_1,
 SUBPARTITION pq4200302 TABLESPACE tbs_1,
 SUBPARTITION pq4200303 TABLESPACE tbs_1,
 SUBPARTITION pq4200304 TABLESPACE tbs_1,
 SUBPARTITION pq4200305 TABLESPACE tbs_1,
 SUBPARTITION pq4200306 TABLESPACE tbs_1,
 SUBPARTITION pq4200307 TABLESPACE tbs_1,
 SUBPARTITION pq4200308 TABLESPACE tbs_1),
PARTITION sales_overflow
(SUBPARTITION pqoflw01 TABLESPACE tbs_2,
 SUBPARTITION pqoflw02 TABLESPACE tbs_2,
 SUBPARTITION pqoflw03 TABLESPACE tbs_2,
 SUBPARTITION pqoflw04 TABLESPACE tbs_2));
```

create index: Nonpartitioned Indexes

```
CREATE INDEX ix_mytab_01 ON mytab(column_1);

CREATE UNIQUE INDEX ix_mytab_01 ON mytab(column_1, column_2, column_3);

CREATE INDEX ix_mytab_01 ON mytab(column_1, column_2, column_3)
TABLESPACE my_indexes COMPRESS
STORAGE (INITIAL 10K NEXT 10K PCTFREE 10) COMPUTE STATISTICS;

CREATE BITMAP INDEX bit_mytab_01 ON my_tab(col_two)
TABLESPACE my_tbs;
```

Allocate and Deallocate Extents

```
ALTER INDEX ix_my_tab ALLOCATE EXTENT;
ALTER INDEX ix_my_tab ALLOCATE EXTENT
DATAFILE '/ora/datafile/newidx.dbf';
ALTER INDEX ix_my_tab DEALLOCATE UNUSED;
ALTER INDEX ix_my_tab DEALLOCATE UNUSED KEEP 100M;
```

Rebuild

```
ALTER INDEX ix_my_tab REBUILD ONLINE;
ALTER INDEX ix_my_tab REBUILD ONLINE
TABLESPACE idx_tbs_new
```

Shrink

```
ALTER INDEX ix_my_tab SHRINK SPACE;
ALTER INDEX ix_my_tab SHRINK SPACE COMPACT CASCADE;
```

Displaying Information About Schema Objects

```
SELECT DBMS_METADATA.GET_DDL('TABLE','NAMES') FROM DUAL;

DBA_OBJECTS
DBA_CATALOG
DBA_DEPENDENCIES
```

Managing Space for Schema Objects

Managing Tablespace Alerts

The following example sets the free-space-remaining thresholds in the USERS tablespace to 10 MB (warning) and 2 MB (critical), and disables the percent-full thresholds.

```
BEGIN
DBMS_SERVER_ALERT.SET_THRESHOLD(
   metrics_id            => DBMS_SERVER_ALERT.TABLESPACE_BYT_FREE,
   warning_operator      => DBMS_SERVER_ALERT.OPERATOR_LE,
   warning_value         => '10240',
   critical_operator     => DBMS_SERVER_ALERT.OPERATOR_LE,
   critical_value        => '2048',
   observation_period    => 1,
   consecutive_occurrences => 1,
   instance_name         => NULL,
   object_type           => DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE,
```

```
        object_name              => 'USERS');

DBMS_SERVER_ALERT.SET_THRESHOLD(
    metrics_id               => DBMS_SERVER_ALERT.TABLESPACE_PCT_FULL,
    warning_operator         => DBMS_SERVER_ALERT.OPERATOR_GT,
    warning_value            => '0',
    critical_operator        => DBMS_SERVER_ALERT.OPERATOR_GT,
    critical_value           => '0',
    observation_period       => 1,
    consecutive_occurrences  => 1,
    instance_name            => NULL,
    object_type              => DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE,
    object_name              => 'USERS');
END;
```

To set your own database wide default threshold values for the Tablespace Space Usage metric:
```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD(
 METRICS_ID=>DBMS_SERVER_ALERT.TABLESPACE_PCT_FULL,
 WARNING_OPERATOR=>DBMS_SERVER_ALERT.OPERATOR_GE,
 WARNING_VALUE=>80,
 CRITICAL_OPERATOR=>DBMS_SERVER_ALERT.OPERATOR_GE,
 CRITICAL_VALUE=>95,
 OBSERVATION_PERIOD=>1,
 CONSECUTIVE_OCCURRENCES=>1,
 INSTANCE_NAME=>NULL,
 OBJECT_TYPE=>DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE,
 OBJECT_NAME=>NULL)
```

To set a warning threshold of 80% and a critical threshold of 95% on the EXAMPLE tablespace, use the same previous example except
OBJECT_NAME parameter should take value of 'EXAMPLE'

To turn off the space-usage tracking mechanism for the EXAMPLE tablespace:
```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD(
 METRICS_ID=>dbms_server_alert.tablespace_pct_full,
 WARNING_OPERATOR=>dbms_server_alert.operator_do_not_check,
 WARNING_VALUE=>'0',
 CRITICAL_OPERATOR=>dbms_server_alert.operator_do_not_check,
 CRITICAL_VALUE=>'0',
 OBSERVATION_PERIOD=>1,
 CONSECUTIVE_OCCURRENCES=>1,
 INSTANCE_NAME=>NULL,
 OBJECT_TYPE=>dbms_server_alert.object_type_tablespace,
 OBJECT_NAME=>'EXAMPLE')
```

Resumable Space Allocation

What Errors are Correctable?
```
ORA-1653 unable to extend table ... in tablespace ...
ORA-1654 unable to extend index ... in tablespace ...

ORA-1631 max # extents ... reached in table ...
ORA-1654 max # extents ... reached in index
ORA-1536 space quote exceeded for tablespace string
```

Enabling Resumable Space Allocation
```
ALTER SYSTEM SET RESUMABLE_TIMEOUT=3600 – one hour
ALTER SESSION ENABLE RESUMABLE;
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600; -- default timeout is 7200
ALTER SESSION DISABLE RESUMABLE;
```
The name of the statement is used to identify the resumable statement in the DBA_RESUMABLE and USER_RESUMABLE
views.
```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600 NAME 'insert into table';
```

Notifying Users: The AFTER SUSPEND System Event and Trigger
Users can use the USER_RESUMABLE or DBA_RESUMABLE views, or the DBMS_RESUMABLE.SPACE_ERROR_INFO
function, within triggers to get information about the resumable statements.
Triggers can also call the DBMS_RESUMABLE package to terminate suspended statements and modify resumable timeout
values. In the following example, the default system timeout is changed by creating a system wide AFTER SUSPEND
trigger that calls DBMS_RESUMABLE to set the timeout to 3 hours:

```
CREATE OR REPLACE TRIGGER resumable_default_timeout
AFTER SUSPEND ON DATABASE
```

```
BEGIN
   DBMS_RESUMABLE.SET_TIMEOUT(10800);
END;
```

Whenever a resumable statement is suspended in any session, this trigger can have either of two effects:
- If an undo segment has reached its space limit, then a message is sent to the DBA and the statement is aborted.
- If any other recoverable error has occurred, the timeout interval is reset to 8 hours.

Here are the statements for this example:

```
CREATE OR REPLACE TRIGGER resumable_default
AFTER SUSPEND
ON DATABASE
DECLARE
   /* declare transaction in this trigger is autonomous */
   /* this is not required because transactions within a trigger
      are always autonomous */
   PRAGMA AUTONOMOUS_TRANSACTION;
   cur_sid          NUMBER;
   cur_inst         NUMBER;
   errno            NUMBER;
   err_type         VARCHAR2;
   object_owner     VARCHAR2;
   object_type      VARCHAR2;
   table_space_name VARCHAR2;
   object_name      VARCHAR2;
   sub_object_name  VARCHAR2;
   error_txt        VARCHAR2;
   msg_body         VARCHAR2;
   ret_value        BOOLEAN;
   mail_conn        UTL_SMTP.CONNECTION;
BEGIN
   -- Get session ID
   SELECT DISTINCT(SID) INTO cur_SID FROM V$MYSTAT;

   -- Get instance number
   cur_inst := userenv('instance');

   -- Get space error information
   ret_value :=
   DBMS_RESUMABLE.SPACE_ERROR_INFO(err_type,object_type,object_owner,
      table_space_name,object_name, sub_object_name);
   /*
   -- If the error is related to undo segments, log error, send email
   -- to DBA, and abort the statement. Otherwise, set timeout to 8 hours.
   --
   -- sys.rbs_error is a table which is to be
   -- created by a DBA manually and defined as
   -- (sql_text VARCHAR2(1000), error_msg VARCHAR2(4000),
   -- suspend_time DATE)
   */

   IF OBJECT_TYPE = 'UNDO SEGMENT' THEN
      /* LOG ERROR */
      INSERT INTO sys.rbs_error (
         SELECT SQL_TEXT, ERROR_MSG, SUSPEND_TIME
         FROM DBMS_RESUMABLE
         WHERE SESSION_ID = cur_sid AND INSTANCE_ID = cur_inst
       );
      SELECT ERROR_MSG INTO error_txt FROM DBMS_RESUMABLE
         WHERE SESSION_ID = cur_sid and INSTANCE_ID = cur_inst;

      -- Send email to receipient via UTL_SMTP package
      msg_body:='Subject: Space Error Occurred

                 Space limit reached for undo segment ' || object_name ||
                 on ' || TO_CHAR(SYSDATE, 'Month dd, YYYY, HH:MIam') ||
                 '. Error message was ' || error_txt;

      mail_conn := UTL_SMTP.OPEN_CONNECTION('localhost', 25);
      UTL_SMTP.HELO(mail_conn, 'localhost');
      UTL_SMTP.MAIL(mail_conn, 'sender@localhost');
      UTL_SMTP.RCPT(mail_conn, 'recipient@localhost');
      UTL_SMTP.DATA(mail_conn, msg_body);
      UTL_SMTP.QUIT(mail_conn);

      -- Abort the statement
      DBMS_RESUMABLE.ABORT(cur_sid);
   ELSE
```

```
        -- Set timeout to 8 hours
        DBMS_RESUMABLE.SET_TIMEOUT(28800);
    END IF;

    /* commit autonomous transaction */
    COMMIT;
END;
/
```

Using Views to Obtain Information About Suspended Statements

The following views can be queried to obtain information about the status of resumable statements:

`DBA_RESUMABLE`
 	
`USER_RESUMABLE`
	These views contain rows for all currently executing or suspended resumable statements. They can be used by a DBA, AFTER SUSPEND trigger, or another session to monitor the progress of, or obtain specific information about, resumable statements.

`V$SESSION_WAIT`	When a statement is suspended the session invoking the statement is put into a wait state. A row is inserted into this view for the session with the EVENT column containing "statement suspended, wait error to be cleared".

Using the DBMS_RESUMABLE Package

The DBMS_RESUMABLE package helps control resumable space allocation. The following procedures can be invoked:

ABORT(sessionID)	This procedure aborts a suspended resumable statement. The parameter sessionID is the session ID in which the statement is executing. For parallel DML/DDL, sessionID is any session ID which participates in the parallel DML/DDL. The caller of ABORT must be the owner of the session with sessionID, have ALTER SYSTEM privilege, or have DBA privileges.
GET_SESSION_TIMEOUT(sessionID)	This function returns the current timeout value of resumable space allocation for the session with sessionID. This returned timeout is in seconds. If the session does not exist, this function returns -1.
SET_SESSION_TIMEOUT(sessionID, timeout)	This procedure sets the timeout interval of resumable space allocation for the session with sessionID. The parameter timeout is in seconds. The new timeout setting will applies to the session immediately. If the session does not exist, no action is taken.
GET_TIMEOUT()	This function returns the current timeout value of resumable space allocation for the current session. The returned value is in seconds.
SET_TIMEOUT(timeout)	This procedure sets a timeout value for resumable space allocation for the current session. The parameter timeout is in seconds. The new timeout setting applies to the session immediately.

Managing Flash Recovery Area

General Operations

To enable the Flash Recovery Area

You must set the two initialization parameters:

DB_RECOVERY_FILE_DEST_SIZE :

The minimum size of the Flash Recovery Area should be at least large enough to contain archive logs that have not been copied to tape.

DB_RECOVERY_FILE_DEST:

This initialization parameter is a valid destination to create the Flash Recovery Area. The destination can be defined as a directory, file system, or ASM disk group.

Note: DB_RECOVERY_FILE_DEST_SIZE must be set before DB_RECOVERY_FILE_DEST.

Note: LOG_ARCHIVE_DEST_10 is implicitly set to USE_DB_RECOVERY_FILE_DEST if you create a recovery area and do not set any other local archiving destinations.

Note: In a RAC database, all instances must have the same values for these parameters. Even though there are multiple nodes they all share the same controlfiles.

Th

Distributed Databases

Database Links

Required Privileges
```
CREATE DATABASE LINK
CREATE PUBLIC DATABASE LINK
CREATE SESSION
```

Create Statement
```
-- private link using the global database name
CREATE DATABASE LINK supply.us.acme.com;

-- connected user
CREATE [PUBLIC] DATABASE LINK sales.division3.acme.com USING 'sales1';

-- current user (global user on both databases)
CREATE PUBLIC DATABASE LINK hq CONNECT TO CURRENT_USER using 'hq';

-- fixed user
CREATE PUBLIC DATABASE LINK hq CONNECT TO jane IDENTIFIED BY doe USING 'hq';
```

Using Connection Qualifiers
You can create a database link with an optional service name in the database link name. This allows you to use many net service names to connect to the same remote database. Therefore, you can define which node to connect to (in RAC) or which network protocol to use.
```
CREATE PUBLIC DATABASE LINK hq.acme.com@hq_1
  USING 'string_to_hq_1';
CREATE PUBLIC DATABASE LINK hq.acme.com@hq_2
  USING 'string_to_hq_2';
CREATE PUBLIC DATABASE LINK hq.acme.com
  USING 'string_to_hq';

SELECT * FROM scott.emp@hq.acme.com@hq_1
```

Naming of Schema Objects Using Database Links
If you are creating a database link connected to a remote database that enforces global naming, then you must use the remote database global database name as the name of the link.
```
COL NAME FORMAT A12
COL VALUE FORMAT A6
SELECT NAME, VALUE FROM V$PARAMETER
   WHERE NAME = 'global_names'

SELECT * FROM GLOBAL_NAME;
```

If GLOBAL_NAMES is set to TRUE
```
SELECT * FROM scott.emp@sales.division3.acme.com;  # emp table in scott's schema
SELECT loc FROM scott.dept@sales.division3.acme.com;
```

If GLOBAL_NAMES is set to FALSE
```
SELECT name FROM scott.emp@foo;  # link name different from global name
```

Changing the Domain in a Global Database Name
- Use the ALTER DATABASE statement to change the domain in a database global name.
- Changing the initialization parameter DB_DOMAIN has no effect on the global database name or on the resolution of database link names.
```
ALTER DATABASE RENAME GLOBAL_NAME TO sales.us.oracle.com;
```

Authorization for Accessing Remote Schema Objects
To perform any updates, inserts, or deletes on the remote object, you must be granted the SELECT privilege on the object, along with the UPDATE, INSERT, or DELETE privilege.

Using Shared Database Links
Shared database links enable you to limit the number of network connections required between the local server and the remote server. In shared database link configuration, a connection is created for every server process. For the non-

shared database link, a connection is created for every session. Typically, shared links are used for public database links. Also typically it is useful when shared server is configured.

If the remote database is shared server and you want to create a shared database link, the local database must be shared server as well.

To create a shared database link, use the keyword SHARED in the CREATE DATABASE LINK statement but the clause AUTHENTICATED BY is required.

Closing Database Links
To close the link that is active in your current session:

```
ALTER SESSION CLOSE DATABASE LINK linkname;
```

Dropping Database Links
- If the link is private, then it must be in your schema.
- If the link is public, then you must have the DROP PUBLIC DATABASE LINK system privilege.

Limiting the Number of Active Database Link Connections
- The initialization parameter OPEN_LINKS controls the number of remote connections that a single user session can use concurrently in distributed transactions.
- The default value for OPEN_LINKS is 4. If OPEN_LINKS is set to 0, then no distributed transactions are allowed.

Obtaining Information about DB Links
```
COL OWNER FORMAT a10
COL USERNAME FORMAT A8 HEADING "USER"
COL DB_LINK FORMAT A30
COL HOST FORMAT A7 HEADING "SERVICE"
SELECT * FROM DBA_DB_LINKS
```

Determining Which Link Connections Are Open
The following views show the database link connections that are currently open in your **current session**:

```
COL DB_LINK FORMAT A25
COL OWNER_ID FORMAT 99999 HEADING "OWNID"
COL LOGGED_ON FORMAT A5 HEADING "LOGON"
COL HETEROGENEOUS FORMAT A5 HEADING "HETER"
COL PROTOCOL FORMAT A8
COL OPEN_CURSORS FORMAT 999 HEADING "OPN_CUR"
COL IN_TRANSACTION FORMAT A3 HEADING "TXN"
COL UPDATE_SENT FORMAT A6 HEADING "UPDATE"
COL COMMIT_POINT_STRENGTH FORMAT 99999 HEADING "C_P_S"
SELECT * FROM V$DBLINK
```

Tuning Distributed Queries

The most effective way of optimizing distributed queries is to access the remote databases as little as possible and to retrieve only the required data. Use *collocated inline* view for that purpose. For example:

```
CREATE TABLE AS (
            SELECT l.a, l.b, r1.c, r1.d, r1.e, r2.b, r2.c
            FROM local l, remote1 r1, remote2 r2
               WHERE l.c = r.c
               AND r1.c = r2.c
               AND r.e > 300
            );
```

rewrite it as:

```
CREATE TABLE AS (
            SELECT l.a, l.b, v.c, v.d, v.e
            FROM (
                   SELECT r1.c, r1.d, r1.e, r2.b, r2.c
                    FROM remote1 r1, remote2 r2
                    WHERE r1.c = r2.c
                    AND r1.e > 300
                 ) v, local l
            WHERE l.c = r1.c
            );
```

Whenever possible, the optimizer re-writes such select statements into Collocated inline.

Oracle 10g DBA Fundamentals Example and Code Reference

Using Hints

NO_MERGE hint to prevent the optimizer from rewriting your query (because you know it contains an aggregate, subquery, or complex SQL and the optimizer cannot rewrite it)

```
SELECT /*+NO_MERGE(v)*/ t1.x, v.avg_y
   FROM t1, (SELECT x, AVG(y) AS avg_y FROM t2 GROUP BY x) v,
   WHERE t1.x = v.x AND t1.y = 1;

/* in query block */

SELECT t1.x, v.avg_y
   FROM t1, (SELECT /*+NO_MERGE*/ x, AVG(y) AS avg_y FROM t2 GROUP BY x) v,
   WHERE t1.x = v.x AND t1.y = 1;
```

The DRIVING_SITE hint lets you define a remote site to act as the query execution site (helpful when the remote site contains the majority of the data). The optimizer still decides about it, if you do not use the hint.

```
SELECT /*+DRIVING_SITE(dept)*/ * FROM emp, dept@remote.com
   WHERE emp.deptno = dept.deptno;
```

Managing Distributed Transactions

Specifying the Commit Point Strength of a Node
The commit point strength is only used to determine the commit point site in a distributed transaction.

Set the commit point strength for a database relative to the amount of critical shared data in the database. For example, a database on a mainframe computer usually shares more data among users than a database on a PC. Therefore, set the commit point strength of the mainframe to a higher value than the PC.

```
COMMIT_POINT_STRENGTH = 200
```

Naming Transactions
```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
    NAME 'update inventory checkpoint 0';
```
The transaction name is displayed in the NAME column of the V$TRANSACTION view.

Determining the ID Number and Status of Prepared Transactions
```
COL LOCAL_TRAN_ID FORMAT A13
COL GLOBAL_TRAN_ID FORMAT A30
COL STATE FORMAT A8
COL MIXED FORMAT A3
COL HOST FORMAT A10
COL COMMIT# FORMAT A10

SELECT LOCAL_TRAN_ID, GLOBAL_TRAN_ID, STATE, MIXED, HOST, COMMIT#
   FROM DBA_2PC_PENDING

SQL> @pending_txn_script
LOCAL_TRAN_ID GLOBAL_TRAN_ID                 STATE    MIX HOST       COMMIT#
------------- ------------------------------ -------- --- ---------- ----------
1.15.870      HQ.ACME.COM.ef192da4.1.15.870  commit   no  dlsun183   115499
```

This output indicates that local transaction 1.15.870 has been committed on this node, but it may be pending on one or more other nodes. Because LOCAL_TRAN_ID and the local part of GLOBAL_TRAN_ID are the same, the node is the global coordinator of the transaction.

Tracing the Session Tree of In-Doubt Transactions
```
COL LOCAL_TRAN_ID FORMAT A13
COL IN_OUT FORMAT A6
COL DATABASE FORMAT A25
COL DBUSER_OWNER FORMAT A15
COL INTERFACE FORMAT A3
SELECT LOCAL_TRAN_ID, IN_OUT, DATABASE, DBUSER_OWNER, INTERFACE
   FROM DBA_2PC_NEIGHBORS

LOCAL_TRAN_ID IN_OUT DATABASE                 DBUSER_OWNER    INT
------------- ------ ------------------------ --------------- ---
1.15.870      out    SALES.ACME.COM           SYS             C
```

If sales committed the transaction but no other node did, then you know that sales is the commit point site, because the commit point site always commits first.

How to Handle In-Doubt Transactions

One of the following errors will be raised:

```
ORA-02050: transaction ID rolled back, some remote dbs may be in-doubt
```

```
ORA-02053: transaction ID committed, some remote dbs may be in-doubt
```

```
ORA-02054: transaction ID in-doubt
```

A robust application should save information about a transaction if it receives any of the preceding errors. This information can be used later if manual distributed transaction recovery is desired.

Oracle 10g DBA Fundamentals Example and Code Reference