

ORACLE DATABASE ADMINISTRATION

ABBREVIATED

By Mark E. Donaldson

ORACLE ARCHITECTURE

An Oracle Server consists of:

- **Oracle Instance**
- **Oracle Database**

An **Oracle Instance** consists of Processes and Memory Structures. **Processes** can be subdivided as follows:

- Background Processes
- Server Processes
- User Processes

Memory Structures can be subdivided as follows:

- System Global Area (SGA)
- Program Global Area (PGA)
- Sort Area

An Oracle Instance is identified by its **ORACLE_SID** can open and use only one database at any point in time. The ORACLE_SID is set in the operating system with the following command:

SET ORACLE_SID=ORCL

THE ORACLE DATABASE

An **Oracle Database** represents the physical structures and is composed of operating system files. And is identified by the database name **DB_NAME**. The files in the database contain user data and the additional information that is needed to ensure proper database operation.

An Oracle Database consists of the following types of files:

- **Data Files** – store the actual information in the database to include the data dictionary, user objects, and before images of data that are modified by current transactions. A database must have at least one Data File.
- **Redo Log Files** – contain a record of changes made to the database to ensure reconstruction of the data in case of failures. A database must have at least two Redo Log Files.
- **Control Files** – contain the information needed to maintain and verify database integrity, and include the names of all the Data Files and the online and archived log files. A database must have at least one Control File.

In addition to the database files, the Oracle Server also needs the following files:

- **Parameter File** – used to define the characteristics and configuration parameters of an Oracle Instance (**INIT<SID>.ORA**). The **INIT<SID>.ORA** file will typically point to the actual parameter file named **INIT.ORA** using the keyword **IFILE** as follows:
IFILE='D:\Oracle\admin\DB1\pfile\init.ora'

Revised February 1, 2009

The **INIT.ORA** file points to Control Files.

- **Password File** – used to authenticate privileged database users.
- **Archived Redo Log Files** – Offline copies of the Redo Log Files that may be necessary to recover from media failures.
- **Alert File** – records alert messages from the Oracle Server.
- **Configuration File** – referred to as the **CONFIG.ORA** file, it is an optional file that also contains parameters like the **INIT.ORA** file. It can only be used if the **INIT.ORA** file contains an Include line specifying the name of the **CONFIG.ORA**. This file is actually merged into the **INIT.ORA** file upon startup of the Instance. Used primarily to segregate a particular set of standard initialization parameters.

When you START the Instance, the Parameter Files is read. When you MOUNT the database, the Control Files are read. When you OPEN the database, the Data Files are referenced.

In addition to the physical files used by the database, Oracle maintains the following logical structures:

- **Tablespaces** – The basic storage allocation in an Oracle database. Each Tablespace is composed of one or more physical (operating system) files.
- **Schemas** – Essentially the same as an account or username. Every initial Oracle database is created with the initial **SYS** and **SYSTEM** Schemas. There is no necessary relationship between a Schema and a Tablespace.
- **Segments** – Each object that takes up space is created as one or more segments. Each Segment can be in one and only one Tablespace.
- **Extents** – Each Segment is composed of one or more extents. An Extent is a contiguous allocation of space within a Data File of a Tablespace. At the time a Segment is created, you can specify the size of the initial and next Extents, as well as the minimum and maximum number of Extents.
- **Rollback Segments** – The space Oracle writes the old value when a Table is updated, allowing users to maintain a consistent read on the Table. It also allows Oracle to restore the contents of the Table if the change is not committed.
- **Temporary Segments** – Used by Oracle during Table and Index creation and for sorting.
- **Tables** – All data in a database is stored in a table to include user data and the Data Dictionary.
- **Indexes** – Used to facilitate quick retrieval of data from the Table. Indexes are stored in separate Segments from the Table data.

THE ORACLE INSTANCE

System Global Area (SGA)

The SGA contains the memory structures of the Oracle Instance which contains data and control information for the Oracle Server. The SGA is allocated in virtual memory. The SGA comprises the following memory structures:

- **Shared Pool** – used to store information such as the most recently executed SQL and the most recently used data from the data dictionary.
- **Database Buffer Cache** – used to store the most recently used data.
- **Redo Log Buffer** – used to register changes made to the database using the instance.
- **Large Pool** – an optional memory area used for IO by RMAN (Recovery Manager) backup and restore to allocate sequential IO buffers from shared memory.
- **Locks**

Shared Pool

The Shared Pool is a part of the SGA that is used during the parse phase of a query. The size of the shared pool is specified by the initialization parameter **SHARED_POOL_SIZE** in the parameter files. The Shared Pool consists of the following areas:

- **Library Cache** – stores the text, parse tree, and execution plan of an SQL statement.
- **Data Dictionary Cache** – part of the shared pool that stores the most recently used data dictionary information. During the parse phase, the server process looks for information in the dictionary cache to resolve the object names specified in the SQL statement and to validate the access privileges. If necessary, the server process initiates the loading of this information from the data files.
- **Control Structures** – contains other data required for the operation and control of the Oracle Instance, including information about the state of the database, the Instance, locks, and individual processes. The information is shared by all processes for the Instance.

Database Buffer Cache

The Database Buffer Cache is an area in the SGA that is used to store the most recently used data blocks. The size of each buffer in the buffer cache is equal to the size of the data block and is specified by the **DB_BLOCK_SIZE** parameter. The number of buffers is equal to the value of the **DB_BLOCK_SIZE** parameter. The Oracle Server uses a least recently used (LRU) algorithm to age out buffers that have not been accessed recently to make way for new blocks in the buffer cache.

Redo Log Buffer

The Server Process records changes made by an Instance in the Redo Log Buffer. Its size in bytes is defined by the **LOG_BUFFER** parameter. The Redo Log Buffer is used sequentially. It is also a circular buffer that is reused after it is filled up, but only after the old redo entries are recorded in the Redo Log Files.

Program Global Area (PGA)

The PGA is a memory region that contains data and controls information for a single server process or a single background process. In contrast to the SGA which is shared and written to by several processes, the

PGA is used by only one process. The PGA contains the following:

- **Sort Area** – used for any sort that may be necessary before rows are processed or returned to the user.
- **Session Information** – user privileges for the user.
- **Cursor State** – indicates the stage in the processing of various cursors that are currently used by the session.
- **Stack Space** – contains the session variables.

Sort Area

Sort Area are memory structures used by Oracle to sort data and it is created in memory space of the user process that requests the sort.

Background Processes

The Background Processes perform common functions that are needed to service the requests from several concurrent users. Each Oracle Instance may use several background processes depending on the configuration.

Default Processes:

- **Database Writer (DBWR)** – responsible for writing changed data to the database.
- **Log Writer (LGWR)** – records changes registered in the Redo Log Buffer to the database.
- **System Monitor (SMON)** – primary function is to check for consistency and initiate recovery of the database when the database is opened.
- **Process Monitor (PMON)** – cleans up the resources if one of the processes fails.
- **Checkpoint Process (CKPT)** – responsible for updating the database status information whenever changes in the Database Buffer Cache are permanently recorded in the database.

Additional Processes:

- **Archiver (ARCH)** – copies the contents of an online Redo Log File to another location (disk file) when that log becomes full.
- **Shared Server Process (S)** – created when using the Multi-threaded Server (MTS), these processes allow many user processes to create connections to an Oracle Instance.
- **Dispatcher (D)** – responsible for receiving connection requests from the listener and directing each request to the least busy Shared Server process when the MTS is being used.
- **Recovery (RECO)** – used by the Oracle distributed transaction facility to recover from failures involving distributed transactions.
- **Lock (LCK)** – used by Oracle running with the Parallel Server Option to provide inter-instance locking.
- **Parallel Query (P)** - used by Oracle running with the Parallel Server Option and are responsible for executing SQL statements in parallel. The number is determined by the **PARALLEL_MIN_SERVERS** parameter in the INIT.ORA file.

- **Queue Monitor (QMN)** – used by the Oracle Advanced Queuing Option to monitor message queues.
- **Job Queue (JNP)** – used when snapshots are implemented in an Oracle database running the Distributed Option.

Database Writer (DBWR)

The server process records changes to rollback and data blocks in the buffer cache. The Database Writer (DBWR) writes the dirty buffers from the Database Buffer Cache to the data files. The DBWR defers writing to the Data Files until one of the following events occurs:

- The number of dirty buffers reaches a threshold value.
- A process scans a specified number of blocks when scanning for free buffers and cannot find any.
- A timeout occurs (Every three seconds).
- Triggered by an event such as closing the database.

Log Writer (LGWR)

The Log Writer (LGWR) is a background process that writes entries from the Redo Log Buffer into the Redo Log Files. The LGWR performs sequential writes to the Redo Log File under the following situations:

- When the Redo Log Buffer is 1/3 full.
- When a timeout occurs (every three seconds) **LOG_CHECKPOINT_TIMEOUT**.
- Before DBWR writes modified blocks in the database buffer cache to the data files.
- When a transaction commits. When a transaction commits, the Oracle server places a commit record along with the System Change Number (SCN) in the Redo Log Buffer.

Server Processes

Server Processes are created by Oracle to handle requests made by connected user processes. A Server Process handles communication with the User Process and interacts with Oracle to carry out specific requests.

User Processes

User Processes are created to execute the code of an application program and to handle communications between the application program and the Server Process.

QUICK REFERENCE	
Parameters	SHARED_POOL_SIZE DB_BLOCK_SIZE DB_BLOCK_BUFFERS LOG_BUFFER DB_BLOCK_MAX_DIRTY_TARGET DB_WRITER_PROCESSES CHECKPOINT_PROCESS ARCHIVELOG SORT_AREA_SIZE LOG_CHECKPOINT_TIMEOUT
Dynamic Performance Views	V\$RESOURCE_LIMIT V\$SGASTAT

Data Dictionary Views	None
Commands	SHOW SGA
Packaged Procedures & Functions	None

MANAGING AN ORACLE INSTANCE

ORACLE CONNECTION & AUTHENTICATION

AUTHENTICATION METHODS

Authentication is required when connecting to the database with administrative privileges for STARTUP and SHUTDOWN of the database or other activities. There are two types of authentication:

- Operating System Authentication
- Oracle Password File Authentication

Basic **CONNECT** syntax is:

CONNECT [LOGON[@SERVICE]] AS ROLE

OS Logon = CONNECT / AS ROLE

Oracle Password File =

CONNECT USERNAME/pw AS ROLE

It is also possible to connect to SQLPLUS or SERVER MANAGER from the command line without logging on or authentication, prior to further actions:

C:\> SQLPLUS /NOLOG

C:\> SVRMGR1 /NOLOG

Operating System Authentication

UNIX:

1. Create UNIX group **dba**. The user (root, oracle, etc.) must be a member of the **dba** group. Confirm in /etc/passwd and /etc/group files.
2. Set REMOTE_LOGIN_PASSWORDFILE parameter to NONE (default).
3. Connect to the database with special administrator privileges **SYSDBA** or **SYSOPER** (connects in SYS schema):

CONNECT / AS (SYSDBA | SYSOPER) or

\$ SQLPLUS "/ AS SYSDBA"

NT:

1. Create user groups **ORA_DBA** and **ORA_OPER**.
2. Add NT operating system user to the new groups. Once the domain is accessed by the user, the user is automatically authorized as DBA.
3. Set REMOTE_LOGIN_PASSWORDFILE parameter to NONE (default).
4. Connect to the database with special administrator privileges **SYSDBA** or **SYSOPER** (connects in SYS schema):

CONNECT / AS (SYSDBA | SYSOPER) or

CONNECT INTERNAL/pw AS SYSDBA or

CONNECT INTERNAL AS SYSDBA or

CONNECT INTERNAL@SERVICE AS SYSDBA or

C:\> SQLPLUSW "/ AS SYSDBA"

Oracle Password File Authentication

The password utility and file allows connection to the Oracle Server using a standard username and password, but connects the user to the **SYS** schema instead of as the username provided. Access to the database using the password file is provided by special GRANT commands issued by privileged users.

1. Create the password file using the password utility ORAPWD on UNIX or ORAPWD on NT:

```
orapwd file=<fname> password=<password>
entries=<entries>
```

where:

fname is the name of the password file.

password is the password for SYS and INTERNAL.

entries is the maximum number of distinct database administrators.

The following command creates a password file with the password "admin" for the user SYS and INTERNAL and accepts up to five users with different passwords:

```
orapwd file=$ORACLE_HOME/dbs/orapwU15
password=admin entries=5
```

2. Set REMOTE_LOGIN_PASSWORDFILE parameter to **EXCLUSIVE** or **SHARED**

where:

EXCLUSIVE indicates that only one instance can use the password file and that the password file contains other names than SYS and SYSTEM.

SHARED indicates that more than one instance can use the password file and the only users recognized by the password file are SYS and INTERNAL.

3. Connect to the database as follows:

```
SQLPLUS> CONNECT internal/admin
```

On UNIX the password files are usually in the \$ORACLE_HOME/dbs directory. On NT, the password file is a hidden file and is usually located in the %ORACLE_HOME%\DATABASE directory. The default INTERNAL password is ORACLE.

Changing the INTERNAL Password

On UNIX and NT delete the existing password file and create a new password file using the **ORAPWD** utility. On NT you can also use the **ORADMIN** utility to recreate the password file:

```
C:\> ORADMIN -DELETE -SID sid
```

```
C:\>ORADMIN -NEW -SID [-INTPWD internal_pwd]
[SRVC svrcname] [MAXUSERS n] [STARTMODE
auto | manual] [-PFILE filename]
```

ORACLE STARTUP/SHUTDOWN

Starting Up In Stages

When starting the database, you choose the state in which it starts. The following are the different stages of starting up a database:

1. STARTING THE INSTANCE

Usually the Instance is only started without mounting the database during database creation or recreation of the control files. **Starting the Instance includes the following tasks:**

- Reading the parameter file INIT<sid>.ORA.
- Allocating the SGA.
- Starting the background processes.
- Opening the Trace and ALERT files.

The database must be named either with the DB_NAME parameter in the INIT<sid>.ORA file or in the STARTUP command.

2. MOUNTING THE DATABASE

To perform specific maintenance operations, you start an instance and MOUNT a database, but do not OPEN the database. The database must be mounted but not open to perform the following tasks:

- Renaming data files.
- Enabling and disabling redo log archiving options.
- Performing full database recovery.

Mounting a database includes the following tasks:

- Associating a database with a previously started instance
- Locating and opening the control files specified in the parameter file.
- Reading the control files to obtain the names and status of the data files and redo log files (however no checks are performed to verify the existence of the data files and online log files at this time).

3. OPENING THE DATABASE

Normal database operation means that an instance is started and the database is mounted and open.

Opening the database includes the following tasks:

- Opening the online data files.
- Opening the online redo log files.

If any of the s or online redo logs files are not present when you attempt to open the database, Oracle returns an error. During this final stage, Oracle verifies that all the s and online redo log files can be opened, and checks the consistency of the database. If necessary, the background process System Monitor (SMON) initiates instance recovery.

SHUTDOWN OPTIONS

Shutdown Mode	A	I	T	N
Allow New Connections	NO	NO	NO	NO
Wait Until Current Sessions End	NO	NO	NO	YES
Wait Until Current Transactions End	NO	NO	YES	YES
Force Checkpoint & Close Files	NO	YES	YES	YES

Shutdown Mode:

A = Abort
I = Immediate
T = Transactional
N = Normal

Shutting Down In Stages

There are three steps to shutting down an Instance and the database to which it is connected:

1. CLOSING THE DATABASE

The first step in shutting down the database is closing. When the database is closing, Oracle writes the buffer cache changes and redo log buffer cache entries to the s and online redo logs files. After this operation, Oracle closes all online s and online redo log files. The control files remain open while a database is close but still mounted.

2. DISMOUNTING THE DATABASE

The second step is dismounting the database from an instance. After you dismount a database, only an instance remains. When a database dismounts, the control files are closed.

3. SHUTTING DOWN THE INSTANCE

The final step in database shutdown is shutting down the instance. When you shut down an instance, Trace and ALERT files are closed, the SGA is deal located, and the background processes are terminated.

STARTUP COMMAND

Syntax:

```
STARTUP [FORCE] [RESTRICT] [PFILE=filename]
[EXCLUSIVE | PARALLEL | SHARED]
[OPEN [RECOVER] [database]
| MOUNT
| NOMOUNT]
```

where:

OPEN - enables users to access the database.

MOUNT - mounts the database for certain DBA activities but does not allow user access to the database.

NOMOUNT – creates the SGA and starts up the background processes but does not allow access to the database.

EXCLUSIVE – permits only the current instance to access the database.

PARALLEL – allows multiple instances to access the database (used with Oracle Parallel Server).

SHARED – offers an alternative term for parallel.

PFILE= - allows a no default parameter file to be used to configure the Instance.

FORCE – aborts the running Instance before performing a normal startup.

RESTRICT – enables only users with RESTRICTED SESSION privilege to access the database.

RECOVER – begins media recovery when the database starts.

To **OPEN** the database from **STARTUP NOMOUNT** to a **MOUNT** stage, or from **MOUNT** to an **OPEN** stage, use the **ALTER DATABASE** command:

ALTER DATABASE {MOUNT | NOMOUNT}

Examples:

```
# start up the instance and OPEN the database
STARTUP PFILE=/disk1/INIT<sid>.ORA
```

```
# startup default database in various stages
```

```
STARTUP OPEN
STARTUP NOMOUNT
STARTUP MOUNT
```

```
# startup non-default database
```

```
STARTUP NOMOUNT PFILE=/disk1/INIT<sid>.ORA
STARTUP MOUNT PFILE=/disk1/INIT<sid>.ORA
```

Notes:

For Windows NT, to start the Oracle Services at the startup time, use Control Panel in the service dialog box and choose automatic as the startup type. For UNIX, automatic database startup and shutdown can be controlled by entries in a special operating system file such as in the /var/opt/oracle directory.

SHUTDOWN COMMAND

Syntax:

```
SHUTDOWN [NORMAL
| TRANSACTIONAL
| IMMEDIATE]
```

| ABORT]

SHUTDOWN NORMAL

Normal is the default shutdown mode. Normal database shutdown proceeds with the following conditions:

- No new connections are allowed.
- The oracle server waits for all users to disconnect before completing the shutdown.
- Oracle closes and dismounts the database before shutting down the Instance.
- The next startup will not require an Instance recovery.

SHUTDOWN TRANSACTIONAL

A transactional shutdown prevents clients from losing work. A transactional shutdown proceeds with the following conditions:

- No client can start a new transaction on this particular Instance.
- A client is disconnected when the client ends the transaction which is in progress.
- When all transactions have finished, a shutdown immediate occurs.

SHUTDOWN IMMEDIATE

Immediate database shutdown proceeds with the following conditions:

- Current SQL statements being processed by Oracle are not completed.
- Oracle Server does not wait for users currently connected to the database to disconnect.
- Oracle rolls back active transactions and disconnects all connected users.
- Oracle closes and dismounts the database before shutting down the Instance.
- The next startup will not require an instance recovery.

SHUTDOWN ABORT

If the normal and immediate shutdown options do not work, you can abort the current database instance, which proceeds with the following conditions:

- Current SQL statements being processed by the Oracle Server are immediately terminated.
- Oracle does not wait for users currently connected to the database to disconnect.
- Uncommitted transactions are not rolled back.
- The Instance is terminated without closing the files.
- The next startup will require recovery.

PARAMETER FILE

Initialization parameters are specified in the Oracle **INIT.ORA** file. This file is usually named **INIT<SID>.ORA**, where sid is the SID (name) of the Oracle Instance. There are two categories of initialization parameters:

- **Dynamically Modifiable Parameters** – parameters that can be changed dynamically while the Instance is up and the database is OPEN. These parameters

are modified using the **ALTER SESSION**, **ALTER SYSTEM**, and **ALTER SYSTEM DEFERRED** commands while the Instance is running.

ALTER SESSION SET SQL_TRACE=true;
ALTER SYSTEM SET TIMED_STATISTICS=true;
ALTER SYSTEM SET SORT_AREA_SIZE=131072

The **ALTER SESSION** command modifies only the value of the parameter for the session that executes the command. The **ALTER SYSTEM** command changes globally the value of the parameter. The new value is effective until shutdown or until it is changed. The **ALTER SYSTEM DEFERRED** command modifies the value for future sessions that connect to the database.

COMMONLY MODIFIED PARAMETERS	
Parameter	Description
IFILE	Name of another parameter file to be embedded within the current parameter file. Up to three levels of nesting is possible.
LOG_BUFFER	Number of bytes allocated to the Redo Log Buffer in the SGA.
MAX_DUMP_FILE_SIZE	Maximum size of the Trace Files specified as number of operating system blocks.
PROCESSES	Maximum number of operating system processes that can connect simultaneously to this instance.
SQL_TRACE	Enable or disable the SQL Trace Facility for every user session.
TIMED_STATISTICS	Enable or disable timing in Trace Files and in monitor screens

- **Static Parameters** - parameters whose values must be specified in the INIT.ORA file before the Instance is started and cannot be changed during database operation.

MUST PARAMETERS	
Parameter	Description
BACKGROUND_DUMP_DEST	Location where background process Trace Files are written.
COMPATIBLE	Version of the server with which this Instance should be compatible.
CONTROL_FILES	Names of the Control Files
DB_BLOCK_BUFFERS	Number of blocks cached in the SGA. The default and the minimum is 50 buffers.

DB_NAME	Database identifier of eight characters or fewer. This is the only parameter that is required when creating a new database.
SHARED_POOL_SIZE	Size in bytes of the shared pool. Default is 3500000.
USER_DUMP_DEST	Location where user process Trace Files are created.

Displaying Current Parameter Values

Current parameter values can be displayed in two ways:

- With the SQL*PLUS PARAMETER command:
SHOW PARAMETER parameter
- With the **V\$PARAMETER** dynamic performance view.

Example Parameter File

```
# Initialization Parameter File: initDB1.ora
db_name=DB1

db_files = 30                # INITIAL
# db_files = 80              # SMALL
# db_files = 400             # MEDIUM
# db_files = 1000           # LARGE

control_files = (C:\Oracle\DATABASE\ctl01DB1.ora,
                 C:\Oracle\DATABASE\ctl02DB1.ora)

db_file_multiblock_read_count = 8    # INITIAL
# db_file_multiblock_read_count = 8  # SMALL
# db_file_multiblock_read_count = 16 # MEDIUM
# db_file_multiblock_read_count = 32 # LARGE

db_block_buffers = 1000            # INITIAL
# db_block_buffers = 100           # SMALL
# db_block_buffers = 550           # MEDIUM
# db_block_buffers = 3200          # LARGE

shared_pool_size = 11534336         # INITIAL
# shared_pool_size = 3500000        # SMALL
# shared_pool_size = 5000000        # MEDIUM
# shared_pool_size = 9000000        # LARGE

log_checkpoint_interval = 10000

processes = 59                     # INITIAL
# processes = 50                    # SMALL
# processes = 100                   # MEDIUM
# processes = 200                   # LARGE

log_buffer = 8192                   # INITIAL
# log_buffer = 8192                 # SMALL
# log_buffer = 32768                # MEDIUM
# log_buffer = 163840               # LARGE

sequence_cache_entries = 10        # INITIAL
# sequence_cache_entries = 10       # SMALL
# sequence_cache_entries = 30       # MEDIUM
# sequence_cache_entries = 100      # LARGE
```

```
sequence_cache_hash_buckets = 10  # INITIAL
# sequence_cache_hash_buckets = 10 # SMALL
# sequence_cache_hash_buckets = 23 # MEDIUM
# sequence_cache_hash_buckets = 89 # LARGE
```

```
# if you want auditing
audit_trail = true
```

```
# if you want timed statistics
timed_statistics = true
```

```
# limit trace file size to 5 Meg each
max_dump_file_size = 10240
```

```
# automatic archiving if archiving has been enabled
#using ALTER DATABASE ARCHIVELOG.
log_archive_start = true
log_archive_dest =
%ORACLE_HOME%\database\archive
log_archive_format = "%ORACLE_SID%%S.%T"
```

```
# rollback_segments = (r01, r02, r03)
```

```
# Global Naming -- enforce that a dblink has same
#name as the db it connects to
global_names = TRUE
```

```
# global database name is db_name.db_domain
db_domain = us.acme.com
```

```
oracle_trace_enable = TRUE
```

```
# define directories to store trace and alert files
background_dump_dest=%RDBMS80%\trace
user_dump_dest=%RDBMS80%\trace
core_dump_dest=%RDBMS80%\trace
```

```
# should be a multiple of the OS block size
db_block_size = 2048
```

```
remote_login_passwordfile = shared
```

```
text_enable = TRUE
```

MANAGING SESSIONS

Enabling Restricted Sessions

Restricted Session is useful when you perform structure maintenance or a database IMPORT or EXPORT. The database can be started in RESTRICTED mode so that it is available only to users with the **RESTRICTED SESSION** privilege.

The database can also be put in **RESTRICTED** mode by using the **ALTER SYSTEM SQL** command:

ALTER SYSTEM {ENABLE | DISABLE} RESTRICTED SESSION

The ALTER SYSTEM command does not disconnect current sessions, but allows future connections only to

users with the **RESTRICTED SESSION** privilege. The dynamic performance view **V\$INSTANCE** contains information about the restricted mode.

Examples:

```
# use startup command to restrict access to database
STARTUP RESTRICT
```

```
# use alter system to place instance in restricted mode
ALTER SYSTEM ENABLE RESTRICTED SESSION
```

```
# query v$instance for information
SELECT logins FROM v$instance
```

Terminating Sessions

After placing an Instance in restricted mode, you may want to kill all current user sessions before performing administrative tasks with the **ALTER SYSTEM KILL SESSION** command:

```
ALTER SYSTEM KILL SESSION 'integer1, integer2';
```

Where integer1 is the value of the SID column in V\$INSTANCE, and interger2 is the value of the SERIAL# column V\$SESSION:

```
SELECT sid, serial#
FROM v$session
WHERE username='SCOTT';
```

```
ALTER SYSTEM KILL SESSION '7, 15';
```

The session ID and serial number are used to uniquely identify a session. This guarantees that the ALTER SYSTEM command is applied to the correct session even if the user logs off and a new session uses the same session ID.

The **ALTER SYSTEM KILL SESSION** command causes the background process PMON to perform the following steps upon execution:

- Roll back the user's current transactions.
- Release all currently held or row locks.
- Free all resources currently reserved by the user.
- Sends message to user informing of being killed.

TRACE FILES

If an error occurs while the Oracle Instance is running, the messages are written to the **ALERT** file. During startup of the database, if the **ALERT** file does not exist, Oracle creates one. The **ALERT** file of a database is a chronological log of messages and errors. The following information is logged in the **ALERT** file:

- All internal errors (ORA-00600) and block corruption errors (ORA-01578).
- Operations that affect database structures and parameters, and SQL*PLUS statements such as STARTUP, SHUTDOWN, ARCHIVE LOG, and RECOVER.
- The values of all nondefault initialization parameters at the time the instance starts.

Tracing can also be generated by server processes at the users request. Tracing can be enables or disabled by the initialization parameter **SQL_TRACE**. The following statement enables writing to a trace file for a particular session:

```
ALTER SESSION SET sql_trace=TRUE | FALSE
```

The parameters in the table below control the location and size of trace files:

TRACE FILE PARAMETERS	
Parameter	Description
BACKGROUND_DUMP_DEST	Defines the location of the background trace file and ALERT file.
USER_DUMP_DEST	A dynamic parameter, defines where trace files will be created at the request of the user.
MAX_DUMP_FILE_SIZE	A dynamic parameter specified in OS blocks, limits the size of user trace files.

DATA DICTIONARY

The Oracle Data Dictionary is a collection of table and related views that enable you to see the inner workings and structures of the Oracle database. The Data Dictionary consists of:

- **Static Data Dictionary Views**
- **Dynamic Performance Data Dictionary Views**

Static Data Dictionary Views

The Static Data dictionary Views give you the ability to find information about database objects. It consists of a series of views owned by SYS. Views can be divided into four groups:

- **USER_** are views that allow you to see objects owned by the user.
- **ALL_** are views that allow you to see the objects owned by the user and objects granted to the user.
- **DBA_** are view allow the DBA to see all objects in the database.
- General interest views.

Static Data Dictionary Views can only be accessed and read when the database is MOUNTED and OPEN.

Dynamic Performance Data Dictionary Views

Commonly referred to as the **V\$ Views**, Dynamic Performance Views (identified by the prefix **V_** but provided with public synonyms with the prefix **V\$**) provide information about the instance and information that the instance has about the database. These views are based on a set of internal memory structures maintained by Oracle as virtual tables, which all begin with an **X\$** prefix.

Once the instance is **STARTED** in the **NOMOUNT** stage, the V\$ views that can be read from memory are accessible. Views that read data from the Control File required the database to be **MOUNTED**.

COMMON DYNAMIC PERFORMANCE VIEWS	
View	Description
Accessible In NOMOUNT Stage	
V\$FIXED_TABLE	Displays all Dynamic Performance Views
V\$PARAMETER	Contains information about the initializations parameters
V\$SGA	Contains summary information on the SGA
V\$OPTION	Lists options that are installed with the Oracle server
V\$PROCESS	Information on currently active processes
V\$SESSION	Lists current sessions information
V\$VERSION	List the version number and the components
V\$INSTANCE	Displays the state of the current instance
Accessible In MOUNT Stage	
V\$THREAT	Contains thread information such as redo log groups
V\$CONTROLFILE	Lists names of the Control Files
V\$DATABASE	Contains database information
V\$DATAFILE	Contains data file information from the Control File
V\$DATAFILE_HEADER	Displays data file header information from the Control File
V\$LOGFILE	Contains information about the online redo log files

QUICK REFERENCE	
Parameters	SHARED_POOL_SIZE DB_NAME CONTROL_FILES SHARED_POOL_SIZE DB_BLOCK_BUFFERS LOG_BUFFER IFILE COMPATIBLE PROCESSES SQL_TRACE DB_BLOCK_MAX_DIRTY_TARGET DB_WRITER_PROCESSES CHECKPOINT_PROCESS ARCHIVELOG LOG_CHECKPOINT_TIMEOUT
Dynamic Parameters	USER_DUMP_DEST MAX_DUMP_FILE_SIZE TIMED_STATISTICS SORT_AREA_SIZE
Dynamic Performance Views	V\$FIXED_TABLE V\$CONTROLFILE V\$DATABASE V\$DATAFILE V\$DATAFILE_HEADER V\$LOGFILE

	V\$OPTION V\$PWFILE_USERS V\$SGA V\$VERSION V\$PARAMETER V\$SYSTEM_PARAMETER V\$SESSION V\$INSTANCE
Data Dictionary Views	None
Commands	SHOW SGA SHOW PARAMETER parameter CONNECT / AS SYSDBA CONNECT / AS SYSOPER STARTUP SHUTDOWN ALTER SESSION SET ALTER SYSTEM SET ALTER SYSTEM SET DEFERRED ALTER DATABASE MOUNT ALTER DATABASE OPEN ALTER SYSTEM KILL SESSION ALTER SYSTEM ENABLE RESTRICTED SESSION ALTER SYSTEM DISABLE RESTRICTED SESSION ALTER SYSTEM DISCONNECT SESSION POST_TRANSACTION
Packaged Procedures & Functions	None

OPTIMAL FLEXIBLE ARCHITECTURE (OFA)

The Optimal Flexible Architecture (OFA) is the Oracle standard for organizing the operating system file system for Oracle software and database installation. The key elements of this structure are:

- Each mount point (disk) has the same name and is numbered sequentially (disk00, disk01).
- Below each mount point is a directory called **oracle**. All Oracle files will be created below this point. On the disk that contains Oracle software, the directory to this point (/disk00/oracle) is called **ORACLE_BASE**.
- The next two levels contains the software directory **product** which in turns contains the directory for each **version** of Oracle installed (/disk00/oracle/product/7.3.4). The directory to this level is called **ORACLE_HOME**.
- All Oracle software is organized in predefined directories below ORACLE_HOME that must be defined at the operating system level. Only the account that runs Oracle should have permissions to **access** these directories (root, system, administrator).
- At the same level as the **product** directory, each disk or mount point will have an **oradata** directory. A separate directory for each database (usually

given the **name of the database**) is then created below **oradata**. All Oracle datafiles are created here. Files in these directories should be protected so that only the account that runs Oracle has permission to **alter** them.

- The directory structure should be created prior to installing Oracle software or creating a database.

OPTIMAL FLEXIBLE ARCHITECTURE (OFA)	
Oracle Software Locations	
/u01/app/oracle /product /8.0.3 /bin /dbs /orainst /sqlplus /7.3.3 /admin /DB1/ /DB2/ /local	/u01/app/oem /product /admin /local
Oracle Database Files	
/u02/ /oradata /DB1/ system01.dbf control01.ctl redo0101.rdo /DB2/ system01.dbf control01.ctl redo0101.rdo	/u03/ /oradata /DB1/ tools01.dbf control02.ctl redo0102.rdo /DB2/ users01.dbf control02.ctl redo0102.rdo

CREATING A DATABASE

DATABASE CREATION STEPS (Simple)

1. Prepare operating system file structure (OFA).
2. Decide on unique instance and database name.
3. Decide database character set.
4. Set operating system environmental variables.
5. Prepare the parameter file.
6. Create a password file.
7. Start the Instance.
8. Create the database.
9. Run scripts to generate the data dictionary.
10. Run post creation scripts.

DATABASE CREATION STEPS (Detailed)

1. CHOOSE DATABASE BLOCK SIZE (Multiple of OS Block Size:1024, 2048, 4096, 8192).
2. DETERMINE INSTANCE & DATABASE NAME (Should be Identical).
3. CREATE OS PRIVILEGED ACCOUNTS AND GROUPS AS NECESSARY.

4. CREATE OS FILE AND DIRECTORY SYSTEM (OFA Guidelines).

UNIX:

```
/disk1/oracle = %ORACLE_BASE%
/disk1/oracle/product/8.1.5 = %ORACLE_HOME%
%ORACLE_BASE%/create = loc creation scripts & logs
%ORACLE_HOME%/dbs = PFIL location
%ORACLE_BASE%/admin = administrative scripts
/disk2/oradata/DB1 = Database Files (control, redo, .dbf)
```

NT:

```
D:\oracle = %ORACLE_BASE%
D:\oracle\product\8.1.5 = %ORACLE_HOME%
%ORACLE_HOME%\create = loc of create scripts & logs
%ORACLE_BASE%\database = PFIL location
%ORACLE_BASE%\admin = administrative scripts
D:\oracle\DB1 = Database Files (control, redo, .dbf)
```

5. SET ENVIRONMENTAL VARIABLES

UNIX: ORACLE_HOME
ORACLE_SID
ORACLE_BASE
ORA_NLS33
PATH

NT: CreDB1 batch file
SYSTEM env
Registry HKLM\SOFTWARE\ORACLE
SET ORACLE_SID=DB1
SET ORACLE_HOME=D:\oracle\prod\8.1.5
SET ORACLE_BASE=D:\oracle

6. CREATE PARAMETER FILE

- Copy seed file and edit for particular DB (init%ORACLE_SID%.ora)
- Use to create control files

7. CREATE SID & PASSWORD FILE .

8. BUILD DATABASE CREATION SCRIPT & BATCH FILE
Use to create redo.log and .dbf files

9. CREATE DATABASE (In SYSTEM Tablespace).

10. INSTALL DATA DICTIONARY VIEWS (Normal location %ORACLE_HOME%\rdbms80\admin\catalog.sql).

11. CREATE OBJECT REQUIRED BY PROCEDURAL COMPONENTS
(%ORACLE_HOME%\rdbms\admin\catproc.sql)

12. INSTALL OTHER COMPONENTS
utlsamp1.sql sample tables for SCOTT
dbmsotpt.sql OUTPUT.LINE
dbmspool.sql object sizes
catexp.sql
prvtpool.plb

13. CREATE ROLLBACK SEGMENT IN SYSTEM TABLESPACE.

14. CREATE A ROLLBACK TABLESPACE (Named ROLLBACK).

15. CREATE ROLLBACK SEGMENTS(More than one and add to INIT.ORA file).

16. CREATE A TEMPORARY TABLESPACE (Named TEMP)

17. CREATE A TABLESPACE FOR DATABASE TOOLS (Named TOOLS).
18. CREATE TABLESPACES FOR USER ACTIVITY (Named DATA).
19. CREATE TABLESPACES FOR INDEXES (Named INDEXES).
20. ALTER SYS AND SYSTEM USERS.
21. CREATE DBA VIEWS FOR SYSTEM ACCOUNT (CONNECT SYSTEM/MANAGER catdbsyn.sql).

DATABASE CREATION SCRIPT (Sample)

```

REM SPOOL to record database creation process
REM Put in same subdirectory as creation script
SPOOL D:\%ORACLE_HOME%\create\CreDB1\%ORACLE_SID%.log
SET ECHO ON

REM CONNECT AS INTERNAL (SYSDBA)
CONNECT INTERNAL/MANAGER

REM START Instance and point to PFILE
STARTUP NOMOUNT
PFILE= D:\%ORACLE_HOME%\database\init\%ORACLE_SID%.ora

REM CREATE the DB1 database
CREATE DATABASE "DB1"
  maxlogfiles 32
  maxlogmembers 5
  maxdatafiles 1000
  maxloghistory 100
  maxinstances 2
LOGFILE
  GROUP 1 ('D:\oradata\log01.log') SIZE 512K,
  GROUP 2 ('D:\oradata\log02.log') SIZE 512K
DATAFILE 'D:\oradata\system01.dbf' SIZE 50M
CHARACTER SET WE8ISO8859P1;

REM Now perform all commands necessary to create
REM the final database after the CREATE DATABASE command has
REM succeeded.

REM install data dictionary:
@/disk00/oracle/software/7.3.4/rdbms/admin/catalog.sql

REM install procedural components:
@/disk00/oracle/software/7.3.4/rdbms/admin/catproc.sql

REM Create additional rollback segment in SYSTEM since
REM at least one non-system rollback segment is required
REM before creating a tablespace.
REM
create rollback segment SYSROLL tablespace system
storage (initial 25K next 25K minextents 2 maxextents 99);

REM Put SYSROLL online without shutting
REM down and restarting the database.
REM
alter rollback segment SYSROLL online;

REM Create a tablespace for rollback segments.
REM
create tablespace ROLLBACK
datafile '/disk01/oracle/oradata/DB1/rbs01.dbf' size 25M
default storage (
  initial 500K
  next 500K

```

```

  pctincrease 0
  minextents 2
);

```

REM Create the "real" rollback segments.

```

REM
create rollback segment RBS01 tablespace ROLLBACK
storage (initial 500K next 500K minextents 2 optimal 1M);
create rollback segment RBS02 tablespace ROLLBACK
storage (initial 500K next 500K minextents 2 optimal 1M);
create rollback segment RBS03 tablespace ROLLBACK
storage (initial 500K next 500K minextents 2 optimal 1M);
create rollback segment RBS04 tablespace ROLLBACK
storage (initial 500K next 500K minextents 2 optimal 1M);

```

REM Use ALTER ROLLBACK SEGMENT ONLINE to put rollback segments online

REM without shutting down and restarting the database.

```

REM
alter rollback segment RBS01 online;
alter rollback segment RBS02 online;
alter rollback segment RBS03 online;
alter rollback segment RBS04 online;

```

REM Since we've created and brought online 4 more rollback segments,

REM we no longer need the rollback segment in the SYSTEM tablespace.

REM We could delete it, but we will leave it here in case we need it REM in the future.

```

alter rollback segment SYSROLL offline;

```

REM Create a tablespace for temporary segments.

```

create tablespace TEMP
datafile '/disk02/oracle/oradata/DB1/temp01.dbf' size 25M
default storage (
  initial 100K
  next 100K
  maxextents UNLIMITED
  pctincrease 0
);

```

REM Create a tablespace for database tools.

```

REM
create tablespace TOOLS
datafile '/disk03/oracle/oradata/DB1/tools01.dbf' size 25M
default storage (
  initial 50K
  next 50K
  maxextents UNLIMITED
  pctincrease 0
);

```

REM Create tablespaces for user activity.

```

REM
Create tablespace DATA
datafile '/disk04/oracle/oradata/DB1/data01.dbf' size 100M
default storage (
  initial 250K
  next 250K
  maxextents UNLIMITED
  pctincrease 0
);

```

REM Create tablespaces for indexes.

```

REM
create tablespace INDEX
datafile '/disk05/oracle/oradata/DB1/index01.dbf' size 100M
default storage (
  initial 250K
  next 250K
  maxextents UNLIMITED
  pctincrease 0
);

```

REM Alter SYS and SYSTEM users, because Oracle will make SYSTEM

```

REM the default and temporary tablespace by default, and we don't
REM want that.
REM
alter user sys temporary tablespace TEMP;
alter user system default tablespace TOOLS temporary tablespace
TEMP;

REM Now run the Oracle-supplied scripts we need for this DB
REM
@/disk00/oracle/software/7.3.4/rdbms/admin/catexp.sql
@/disk00/oracle/software/7.3.4/rdbms/admin/dbmspool.sql
@/disk00/oracle/software/7.3.4/rdbms/admin/prvtpool.plb

REM Now run the Oracle-supplied script to create the DBA views
REM for the SYSTEM account. Change to SYSTEM first.
REM
connect system/manager
@/disk00/oracle/software/7.3.4/rdbms/admin/catdbsyn.sql

REM All done, so close the log file and exit.
REM
SPOOL OFF
EXIT

```

SET OPERATING SYSTEM ENVIRONMENT

UNIX

On UNIX set the following environmental variables:

- ORACLE_HOME
- ORACLE_SID
- ORACLE_BASE
- ORA_NLS33
- PATH

NT

Oracle on NT uses variables in the NT registry the way Oracle on UNIX uses shell environmental variables. Parameters such as ORACLE_HOME, ORACLE_BASE, ORA_NLS33, and ORACLE_SID are stored in HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE

The creation of a new database requires ORACLE_SID to be set with the following command:

```
C:\ SET ORACLE_SID=DB1
```

Create a new Instance, Service and Password File to run the database with the **ORADMIN** utility from command line:

```

ORADMIN -NEW -SID [-INTPWD internal_pwd]
[SRVC svrcname] [MAXUSERS number]
[STARTMODE auto, manual] [-PFILE filename]

```

Example:

```

ORADMIN -NEW -SID DB1
-INTPWD oracle -STARTMODE auto
-PFILE ORACLE_HOME\DATABASE\INIT<sid>.ORA

```

Cleanup:

To cleanup services after a disastrous creation, do the following:

- ORADMIN -DELETE -SID DB1.
- Delete all datafiles, control files, and redo logs.
- Stop related NT services.

- There is no DROP DATABASE command.

If you just want to create a new Password File use the Oracle Password File Utility ORAPWD.

ORADIM UTILITY OPTIONS (To Administer The Instance)

Create New Instance:

```

ORADIM -NEW -SID SID [-SRVC SERVICE_NAME [-
INTPWD INTERNAL_PWD] - SHUTTYPE SRVC | INST |
SRVC, INST
[-MAXUSERS NUMBER][[-STARTMODE AUTO | MANUAL][
-PFILE FILENAME]

```

```

C:\> ORADIM -NEW -SID PROD -INTPWD MYPASSWORD1 -
STARTMODE AUTO -PFILE
C:\ORACLE\ADMIN\PROD\PFILE\INIT.ORA

```

Start An Instance:

```

ORADIM -STARTUP -SID SID [-USRPWD USER_PWD] [-
STARTTYPE SRVC | INST | SRVC, INST] [-PFILE
FILENAME]

```

```

C:\> ORADIM -STARTUP -SID PUMA -STARTTYPE SRVC -
PFILE C:\ORACLE\ADMIN\PROD\PFILE\INIT.ORA

```

Stop An Instance:

```

ORADIM -SHUTDOWN -SID SID [-USRPWD USER_PWD] [-
SHUTTYPE SRVC | INST | SRVC, INST] [-SHUTMODE A | I |
N]

```

```

C:\> ORADIM -SHUTDOWN -SID PUMA -SHUTTYPE SRVC
INST

```

Modify An Instance:

```

ORADIM -EDIT -SID SID [-NEWSID NEWSID] [-INTPWD
INTERNAL_PWD]
[-STARTMODE AUTO | MANUAL][[-PFILE FILENAME]

```

```

C:\> ORADIM -EDIT -SID PROD -NEWSID LYNX -INTPWD
MYCAT123 -STARTMODE AUTO -PFILE
C:\ORACLE\ADMIN\LYNX\PFILE\INIT.ORA

```

PREPARING THE PARAMETER FILE

1. When preparing the new database, copy the default INIT.ORA file and rename it with the new name, or place in individual pfile directory.

2. Change parameter setting and values as needed.

3. Specify at least the following parameters.

Parameter	Description
DB_NAME	Database identifier of eight characters or fewer. This is the only parameter that is required when creating a new database. This parameter does not need to match the ORACLE_SID, but must match the name used in the CREATE DATABASE statement.
CONTROL_FILES	Specifies a list of control files (always specify at least two). The control files do not need to exist at this point. The Oracle Server can

create new operating system files when creating the database.

DB_BLOCK_SIZE Determine size. Use Format in NT.

To change the name of a database after it is created, use the **CREATE CONTROLFILE** command to re-create the control file.

STARTING THE INSTANCE

1. Connect as **SYSDBA** using operating system authentication or password file.
2. Start the Instance using the **STARTUP NOMOUNT** command and point to the parameter file:

STARTUP NOMOUNT PFILE=init.ora

CREATE DATABASE COMMAND

Syntax:

```
CREATE DATABASE [database]
  [CONTROLFILE REUSE]
  [LOGFILE [GROUP integer] filespec
  [, [GROUP integer] filespec ]
  [MAXLOGFILES integer]
  [MAXLOGMEMBERS integer]
  [MAXDATAFILES integer]
  [MAXINSTANCES integer]
  [ARCHIVELOG NOARCHIVELOG]
  [CHARACTERSET character set]
  [NATIONAL CHARACTER SET charset]
  [DATAFILE filespec [autoextend_clause]
  [, filespec [autoextend_clause] ] ]
  filespec ::= 'filename' [SIZE integer] [K|M] [REUSE]
  autoextend_clause ::=
  [AUTOEXTEND {OFF | ON [NEXT integer [K|M]]
  MAXSIZE {UNLIMITED | INTEGER [K | M] } } ]
```

Notes:

- Oracle Server allocates as much space in the control files as the values of **MAXLOGMEMBERS**, **MAXLOGFILES**, **MAXDATAFILES**, **MAXLOGHISTORY**, and **MAXINSTANCES** require. To change the value of any of these parameters, use the **CREATE CONTROLFILE** command to re-create the control file.
- There is no **DROP DATABASE** command. To delete a database, delete all datafiles from the operating system.
- To make the new database the default database on NT, change the ORACLE_SID in the registry.
- It is not possible to change character set or the national character set after creating a database.
- On NT you can use the **build_db.sql** script to create a database.

QUICK REFERENCE

Parameters	DB_NAME
------------	---------

	CONTROL_FILES DB_BLOCK_SIZE
Dynamic Performance Views	V\$LOGFILE V\$CONTROLFILE V\$DATAFILE
Data Dictionary Views	None
Commands	CREATE DATABASE
Packaged Procedures & Functions	None

CREATING DATA DICTIONARY VIEWS AND STANDARD PACKAGES

QUICK REFERENCE

Parameters	DB_NAME CONTROL_FILES DB_BLOCK_SIZE
Dynamic Performance Views	V\$LOGFILE V\$CONTROLFILE V\$DATAFILE
Data Dictionary Views	None
Commands	CREATE DATABASE
Packaged Procedures & Functions	None

SEQUENCE OBJECTS

A sequence is a database object created by a user, and can be shared by multiple users to generate unique integers. You can use sequences to automatically generate primary key values.

Sequence NEXTVAL & CURRVAL Pseudocolumns

Once you create a sequence, you can use it to generate sequential numbers for use in tables. The **NEXTVAL pseudocolumn is used to extract successive sequence numbers from a specified sequence. When you reference sequence. NEXTVAL, a new sequence number is generated and the current sequence number is placed in CURRVAL.**

The CURRVAL pseudocolumn is used refer to a sequence number that the current user has just generated. **NEXTVAL must be used to generate a sequence number in the current session before CURRVAL can be referenced.** When sequence.CURRVAL is referenced, the last value returned to the user's process is displayed.

Both NEXTVAL and CURRVAL must be referenced with the sequence name: sequence.NEXTVAL.

Rules For Using NEXTVAL & CURRVAL

You can use NEXTVAL & CURRVAL in the following:

- The SELECT list of a SELECT statement that is not part of a subquery.

- The SELECT list of a subquery in an INSERT statement.
- The VALUES clause of an INSERT statement.
- The SET clause of an UPDATE statement.

You cannot use NEXTVAL & CURRVAL in the following:

- A SELECT list of a view.
- A SELECT statement with the DISTINCT keyword.
- A SELECT statement with the GROUP BY, HAVING, or ORDER BY clauses.
- A subquery in a SELECT, DELETE, or UPDATE statement.
- A DEFAULT expression in a CREATE TABLE or ALTER TABLE statement.

SEQUENCE Syntax

```
CREATE SEQUENCE sequence
  [INCREMENT BY n]
  [START WITH n]
  [ {MAXVALUE n | NOMAXVALUE} ]
  [ {MINVALUE n | NOMINVALUE} ]
  [ {CYCLE | NOCYCLE} ]
  [ {CACHE n | NOCACHE} ];
```

where:

sequence is the name of the sequence generator.

INCREMENT BY n specifies the interval between sequence numbers where n is an integer (If this clause is omitted, the sequence will increment by 1).

START WITH n specifies the first sequence number to be generated (If the clause is omitted, the sequence will start with 1).

MAXVALUE n specifies the maximum value the sequence can generate.

NOMAXVALUE specifies a maximum value of 10²⁷ for an ascending sequence and -1 for a descending sequence (default option).

MINVALUE n specifies the minimum sequence value.

NOMINVALUE specifies a minimum value of 1 for an ascending sequence and -(10²⁶) for a descending sequence (default option).

CYCLE | NOCYCLE specifies that the sequence continues to generate values after reaching either its maximum or minimum value or does not generate additional values (NOCYCLE is the default option).

CACHE n | NOCACHE specifies how many values the Oracle Server will preallocate and keep in memory (20 values default).

SEQUENCE Examples

Create A Sequence

```
CREATE SEQUENCE dept_deptno
  INCREMENT BY 1
  START WITH 91
  MAXVALUE 100
```

```
NOCACHE
NOCYCLE;
--do not use CYCLE if sequence is to generate primary
-- key values.
```

Confirm A Sequence

```
SELECT sequence_name, min_value, max_value,
       increment_by, last_number
FROM user_sequences;
```

Use A Sequence

```
INSERT INTO dept(deptno, dname, loc)
VALUES (dept_deptno.NEXTVAL, 'MARKETING',
       'SAN DIEGO');
```

```
--return current sequence value
SELECT dept_deptno.CURRVAL
FROM dual;
```

--repeated execution

```
INSERT INTO emp . . .
VALUES (emp_deptno.NEXTVAL, (emp_deptno.NEXTVAL,
dept_deptno.CURRVAL, . . . ));
```

Modify A Sequence

```
ALTER SEQUENCE dept_deptno
  INCREMENT BY 1
  MAXVALUE 999999
  NOCACHE
  NOCYCLE;
```

Remove A Sequence

```
DROP SEQUENCE dept_deptno;
```

INDEX OBJECTS

An Oracle Server index is a schema object that can speed up the retrieval of rows by using a pointer. If you do not have an index, a full table scan will occur. Indexes are logically and physically independent of the table they index so they can be created and dropped without effect on the base tables or other indexes.

Function based indexes are possible where an index is based on expressions. The index expression is built from table columns, constraints, SQL functions, and user-defined functions.

INDEX Syntax

```
CREATE INDEX index
ON table (column[, column . . . ] );
```

where:

index is the name of the index.

table is the name of the table.

column is the name of the column in the table to be indexed.

INDEX Examples

Create An Index

```
CREATE INDEX emp_ename_idx
ON emp(ename);
```

Confirm An Index

```
SELECT ic.index_name, ic.column_name,
       ic.column_position col_pos, ix.uniqueness
FROM user_indexes ix, user_ind_columns ic
WHERE ic.index_name = ix.index_name
AND ic.table_name = 'EMP';
```

Function Based Indexes

```
CREATE INDEX uppercase_idx
ON emp (UPPER (ename) );
```

```
SELECT *
FROM emp
WHERE UPPER (ename) = 'KING';
```

```
SELECT *
FROM emp
WHERE UPPER (ename) IS NOT NULL
ORDER BY UPPER (ename);
```

Remove A Sequence

```
DROP INDEX emp_ename_idx;
```

SYNONYM OBJECTS

A Synonym is another name for an object. It is used to simplify access to objects such as tables owned by another user or to shorten lengthy object name.

SYNONYM Syntax

```
CREATE [PUBLIC] SYNONYM synonym
FOR object;
```

where:

PUBLIC creates a synonym accessible to all users.

synonym is the name of the synonym to be created.

object identifies the object for which the synonym is created.

SYNONYM Examples

Create A Synonym

```
CREATE SYNONYM d_sum
FOR dept_sum_vu;
```

```
CREATE PUBLIC SYNONYM dept
FOR alice.dept;
```

Remove A Sequence

```
DROP SYNONYM d_sum;
DROP SYNONYM dept;
```

CONTROLLING USER ACCESS - SECURITY

Schema

A schema is a collection of objects, such as tables, views, and sequences. **The schema is owned by a database user and has the same name as that user.**

Privileges

Privileges are the right to execute particular SQL statements. Users require **system privileges** to gain access to the database and **object privileges** to manipulate the content of the objects in the database. Users can also be given the privilege to grant additional privileges to other users, or to **roles**, which are named groups of related privileges. Once a user is created, the DBA can grant specific system privileges to the user.

PRIVILEGES	
DBA System Privilege	Operations Authorized
CREATE USER	Allows grantee to create other Oracle users.
DROP USER	Drops another user
DROP ANY TABLE	Drops a table in any schema
BACKUP ANY TABLE	Backs up any table in any schema with the export utility.
User System Privilege	Operations Authorized
CREATE SESSION	Connect to the database
CREATE TABLE	Create tables in the user's schema.
CREATE SEQUENCE	Create a sequence in the user's schema.
CREATE VIEW	Create a view in the user's schema.
CREATE PROCEDURE	Create stored procedure, function, or package in the user's schema.

CREATE USER Syntax & Example

```
CREATE USER user
IDENTIFIED BY password;
```

where:

user is the name of the user to be created.

password specifies that the user must log in with this password.

```
CREATE USER scott
IDENTIFIED BY tiger;
```

```
GRANT create table, create sequence, create view
TO scott;
```

Changing The Password

```
ALTER USER scott
IDENTIFIED BY lion;
```

Roles

A role is a named group of related privileges that can be granted to a user. A user can have access to several roles, and several users can be assigned to the same role.

CREATE ROLE Syntax & Example

```
CREATE ROLE manager;  
  
GRANT create table, create view  
TO manager;  
  
GRANT manager  
TO BLAKE, CLARK;
```

GRANT STATEMENT

Used to grant object privileges to users. SELECT, ALTER, DELETE, EXECUTE, INDEX, INSERT, REFERENCES, And UPDATE privileges can be granted to or revoked from a user.

GRANT Syntax

```
GRANT object_priv [ (columns) ]  
ON object  
TO {user | role | PUBLIC}  
[WITH GRANT OPTION];
```

where:

object_priv is an object privilege to be granted.
ALL specifies all object privileges.

columns specifies the column from a table or view on which privileges are granted.

ON object is object on which the privileges are granted.
TO identifies to whom privilege is granted.
PUBLIC grants object privilege to all users.

WITH GRANT OPTION allows the grantee to grant the object privileges to other users and roles.

GRANT Example

```
GRANT select  
ON emp  
TO sue, rich;  
  
GRANT update (dname, loc)  
ON dept  
TO scott, manager;  
  
GRANT select, insert  
ON dept  
TO scott  
WITH GRANT OPTION;  
  
GRANT select  
ON alice.dept  
TO PUBLIC;
```

	roles.
USER_ROLE_PRIVS	Roles accessible by user.
USER_TAB_PRIVS_MADE	Object privileges granted on the user's objects.
USER_TAB_PRIVS_RECD	Object privileges granted to the user.
USER_COL_PRIVS_MADE	Object privileges granted on the columns of the user's objects.
USER_COL_PRIVS_RECD	Object privileges granted to the user on specific columns.

REVOKE STATEMENT

REVOKE Syntax and Example

```
REVOKE {privilege [, privilege . . . ] | ALL}  
ON object  
FROM {user [, user . . . ] | role | PUBLIC}  
[CASCADE CONSTRAINTS];
```

where:

CASCADE CONSTRAINTS is required to remove any referential constraints made to the object by means of the REFERENCES privilege.

```
REVOKE select, insert  
ON dept  
FROM scott;
```

Confirming Privileges Granted

Data Dictionary Table	Description
ROLE_SYS_PRIVS	System privileges granted to roles.
ROLE_TAB_PRIVS	Table privileges granted to

