Managing an Oracle Instance

Date: 07.10.2009

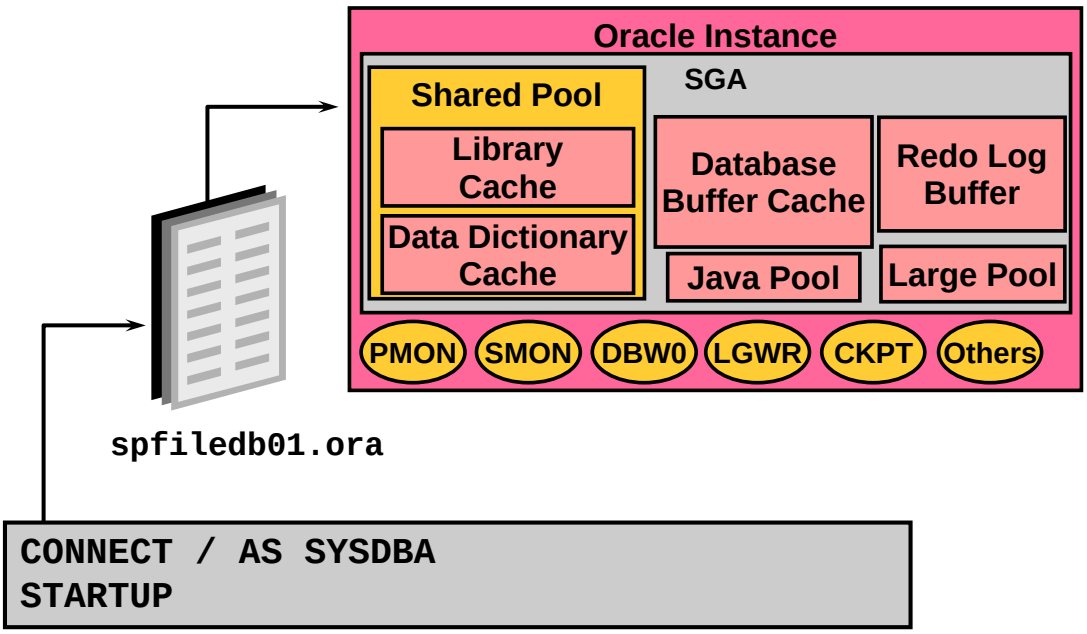
Instructor: SL. Dr. Ing. Ciprian Dobre

Objectives

After completing this lesson, you should be able to do the following:

- **Create and manage initialization parameter files**
- **Start up and shut down an instance**
- **Monitor and use diagnostic files**

Initialization Parameter Files



Initialization Parameter Files

In order to start an instance and open the database, you must connect as SYSDBA and enter the STARTUP command. The Oracle server will then read the initialization parameter file and prepare the instance according to the initialization parameters contained within.

Note: You must have SYSDBA privilege. Authentication and the SYSDBA privilege will be discussed in later lessons.

Initialization Parameter Files

- **Entries are specific to the instance being started**
- **Two types of parameters:**
  - **Explicit: Having an entry in the file**
  - **Implicit: No entry within the file, but assuming the Oracle default values**
- **Multiple initialization parameter files can exist**
- **Changes to entries in the file take effect based on the type of initialization parameter file used:**
  - **Static parameter file, PFILE**
  - **Persistent server parameter file, SPFILE**

Initialization Parameter Files

To start an instance, the Oracle server reads the initialization parameter file. Two types of initialization parameter files exist:

Static parameter file, PFILE, commonly referred to as `initSID.ora`.

Persistent server parameter file, SPFILE, commonly referred to as `spfileSID.ora`.

Initialization Parameter File Contents

A list of instance parameters

The name of the database the instance is associated with

Allocations for memory structures of the System Global Area (SGA)

What to do with filled online redo log files

The names and locations of control files

Information about undo segments

Multiple initialization parameter files can exist for an instance in order to optimize performance in different situations.

Initialization Parameter Files (continued)

Using Oracle Enterprise Manager to View Initialization Parameters

From the OEM Console:

Navigate to Instance > Configuration.

Highlight Configuration.

Select All Initialization Parameters from the General tabbed page.

PFILE
initSID.ora

- Text file
- Modified with an operating system editor
- Modifications made manually
- Changes take effect on the next start up
- Only opened during instance start up
- Default location is $ORACLE_HOME/dbs

PFILE

The PFILE is a text file that can be maintained using a standard operating system editor. The PFILE is read only during instance startup. If the file is modified, the instance must be shut down and restarted in order to make the new parameter values effective.

By default, the PFILE is located in the `$ORACLE_HOME/dbs` directory and named `initSID.ora`.

Creating a PFILE

- **Created from a sample `init.ora` file**
  - **Sample installed by the Oracle Universal Installer**
  - **Copy sample using operating system copy command**
  - **Uniquely identified by database SID**

```
cp init.ora $ORACLE_HOME/dbs/initdba01.ora
```

- **Modify the `initSID.ora`**
  - **Edit the parameters**
  - **Specific to database needs**

Creating a PFILE

A sample `init.ora` file is created by the Universal Installer during installation. This sample `init.ora` file can be used to create an instance-specific `initSID.ora`. A text editor can be used to modify the parameters within the `initSID.ora` file.

PFILE Example

```
# Initialization Parameter File: initdba01.ora
db_name              = dba01
instance_name        = dba01
control_files        = (
    /home/dba01/ORADATA/u01/control01dba01.ctl,
    /home/dba01/ORADATA/u02/control01dba02.ctl)
db_block_size        = 4096
db_cache_size        = 4M
shared_pool_size     = 50000000
java_pool_size       = 50000000
max_dump_file_size   = 10240
background_dump_dest = /home/dba01/ADMIN/BDUMP
user_dump_dest       = /home/dba01/ADMIN/UDUMP
core_dump_dest       = /home/dba01/ADMIN/CDUMP
undo_management      = AUTO
undo_tablespace      = UNDOTBS
. . .
```

PFILE Example

- Specify the values in the following format: `keyword=value`.
- The server has a default value for each parameter. This value may be operating system dependent, depending on the parameter.
- Parameters can be specified in any order, although there are some exceptions.
- Comment lines begin with the # symbol.
- Enclose parameters in double quotation marks to include character literals.
- Additional files can be included with the keyword `IFILE`.
- If case is significant for the operating system, then it is also significant in filenames.
- Multiple values are enclosed in parentheses and separated by commas.

Note: Develop a standard for listing parameters; either list them alphabetically or group them by functionality. The PFILE varies from instance to instance and does not necessarily look like the preceding example.

SPFILE
`spfileSID.ora`

- **Binary file**
- **Maintained by the Oracle server**
- **Always resides on the server side**
- **Ability to make changes persistent across shut down and start up**
- **Can self-tune parameter values**
- **Can have Recovery Manager support backing up to the initialization parameter file**

SPFILE

An SPFILE, new to Oracle9*i*, is a binary file. The file is not meant to be modified manually and must always reside on the server side. After the file is created it is maintained by the Oracle server. If modified manually, the SPFILE is rendered useless. The SPFILE provides the ability to make changes to the database persistent across shutdown and startup. It also provides the ability to self-tune parameter values, which are recorded in the file. RMAN support for backing up the initialization parameter file is possible because the SPFILE resides on the server side. By default, the file is located in `$ORACLE_HOME/dbs` and has a default name in the format of `spfileSID.ora`.

Creating an SPFILE

- **Created from a PFILE file**

```
CREATE SPFILE = '$ORACLE_HOME/dbs/spfileDBA01.ora'
FROM PFILE = '$ORACLE_HOME/dbs/initDBA01.ora';
```

  where
  - **SPFILE-NAME: SPFILE to be created**
  - **PFILE-NAME: PFILE creating the SPFILE**
- **Can be executed before or after instance start up**

Creating an SPFILE

An SPFILE is created from a PFILE file using the CREATE SPFILE command. This command requires the SYSDBA privilege to execute. This command can be executed before or after instance startup.

```
CREATE SPFILE [='SPFILE-NAME']
FROM PFILE[='PFILE-NAME'];
```

where:

SPFILE-NAME: Name of the SPFILE to be created

PFILE-NAME: Name of the PFILE being used to create the SPFILE. The PFILE must be available on the server side

If the SPFILE-NAME and PFILE-NAME are not included in the syntax, Oracle will use the default PFILE to generate an SPFILE with a system-generated name.

```
SQL> CREATE SPFILE FROM PFILE;
```

Creating an SPFILE (continued)

Exporting an SPFILE

The contents of an SPFILE can be exported into a PFILE.

```
SQL> CREATE PFILE FROM SPFILE;
```

The PFILE is created as a text file on the server side. This command can be executed either before or after instance startup. This provides an easy way to view the SPFILE and make modifications by:

- Exporting the SPFILE to a PFILE
- Editing the PFILE
- Re-creating the SPFILE from the edited PFILE

Exporting an SPFILE to a PFILE can also serve as another alternative to creating a backup of the server parameter file.

Note: With Oracle9*i*, `RMAN` can also back up server parameter files.

`V$SPPARAMETER`

As shown above, there are several options for viewing the parameter settings within the SPFILE. `V$SPPARAMETER` is another source for presenting and viewing contents of the SPFILE.

Creating an SPFILE

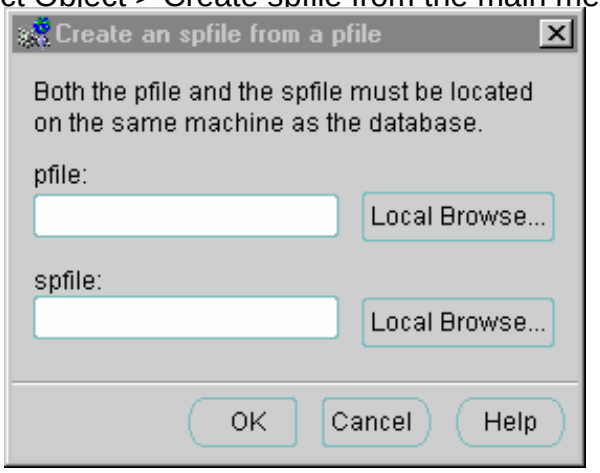
Using Oracle Enterprise Manager to Create an SPFILE

From the OEM Console:

Navigate to Instance > Configuration

Highlight Configuration.

Select Object > Create spfile from the main menu.



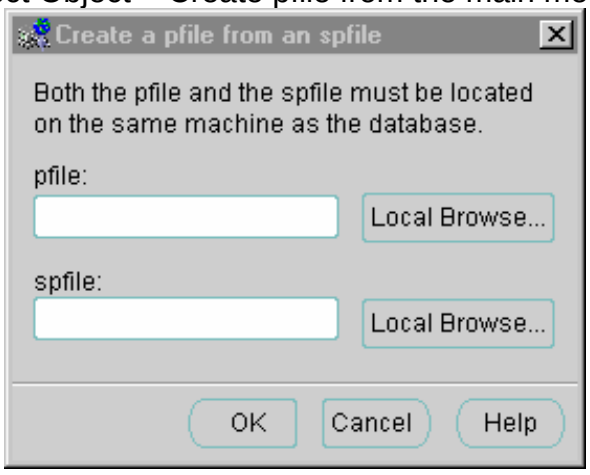
Using Oracle Enterprise Manager to Export an SPFILE

From the OEM Console:

Navigate to Instance > Configuration

Highlight Configuration.

Select Object > Create pfile from the main menu.



SPFILE Example

```
*.background_dump_dest= '/home/dba01/ADMIN/BDUMP'
*.compatible='9.2.0'
*.control_files='/home/dba01/ORADATA/u01/ctrl01.ctl'
*.core_dump_dest= '/home/dba01/ADMIN/CDUMP'
*.db_block_size=4096
*.db_name='dba01'
*.db_domain= 'world'
*.global_names=TRUE
*.instance_name='dba01'
*.remote_login_passwordfile='exclusive'
*.java_pool_size=50000000
*.shared_pool_size=50000000
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS'
. . .
```

SPFILE Example

The comments specified on the same lines as a parameter setting in the PFILE are maintained in the SPFILE. All other comments are ignored.

Although the text of an SPFILE is easily viewed in UNIX, the SPFILE is binary, and manual modification of the SPFILE will render it unusable. If you need to view the specific contents of an SPFILE or make some modification, export the SPFILE to a PFILE.

Modifying Parameters in SPFILE

- **Changing parameter values**

```
ALTER SYSTEM SET undo_tablespace = UNDO2;
```

- **Specifying temporary or persistent changes**

```
ALTER SYSTEM SET undo_tablespace = UNDO2
SCOPE=BOTH;
```

- **Deleting or resetting values**

```
ALTER SYSTEM RESET undo_suppress_errors
SCOPE=BOTH SID='*';
```

Modifying Parameters in SPFILE

The `ALTER SYSTEM SET` command is used to change the value of instance parameters.

```
ALTER SYSTEM SET parameter_name = parameter_value
  [COMMENT 'text'] [SCOPE = MEMORY|SPFILE|BOTH]
  [SID= 'sid'|'*']
```

where

`parameter_name`: Name of the parameter to be changed

`parameter_value`: Value the parameter is being changed to

`COMMENT`: A comment to be added into the SPFILE next to the parameter being altered

`SCOPE`: Determines if change should be made in memory, SPFILE, or in both areas

`MEMORY`: Changes the parameter value only in the currently running instance

`SPFILE`: Changes the parameter value in the SPFILE only

`BOTH`: Changes the parameter value in the currently running instance and the SPFILE

`SID`: Identifies the `ORACLE_SID` for the SPFILE being used

'`sid`': Specific `SID` to be used in altering the SPFILE

Modifying Parameters in SPFILE (continued)

Example

```
SQL>  SHOW PARAMETERS undo_suppress_errors
 NAME                    TYPE        VALUE
 ----------------------- ----------- -------
 undo_suppress_errors    boolean     FALSE

SQL> ALTER SYSTEM SET undo_suppress_errors = TRUE
   2 COMMENT = 'temporary testing' SCOPE=BOTH
   3 SID='DBA01';
SQL> SHOW PARAMETERS undo_suppress_errors
 NAME                    TYPE        VALUE
 ----------------------- ----------- -------
 undo_suppress_errors    boolean     TRUE
```

The `ALTER SYSTEM RESET` command is used to delete or revert to the default value.

```
ALTER SYSTEM RESET parameter_name [SCOPE = MEMORY|SPFILE|BOTH]
[SID= 'sid'|'*']
```

Example

```
SQL> ALTER SYSTEM RESET undo_suppress_errors
   2 SCOPE=BOTH SID='dba01';
```

There are several ways to remove a parameter from the `SPFILE`:

- Set the parameter back to its default value to simulate deleting using `ALTER SYSTEM SET`.
- Re-create the SPFILE using `CREATE SPFILE FROM PFILE`.
- Use `ALTER SYSTEM RESET` to delete the parameter from the SPFILE.

Modifying Parameters in SPFILE (continued)

Using Oracle Enterprise Manager to Modify the SPFILE Configuration

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Click All Initialization Parameters in the General tabbed page.
4. Modify a parameter in the value column.
5. Click Apply.

STARTUP Command Behavior

- **Order of precedence:**
  - **`spfileSID.ora`**
  - **Default SPFILE**
  - **`initSID.ora`**
  - **Default PFILE**
- **Specified PFILE can override precedence.**

```
STARTUP PFILE = $ORACLE_HOME/dbs/initDBA1.ora
```

- **PFILE can indicate to use SPFILE.**

```
SPFILE = /database/startup/spfileDBA1.ora
```


STARTUP Command Behavior

Order of Precedence

When the command `STARTUP` is used, the `spfileSID.ora` on the server side is used to start up the instance.

If the `spfileSID.ora` is not found, the default SPFILE on the server side is used to start the instance.

If the default SPFILE is not found, the `initSID.ora` on the server side will be used to start the instance.

A specified PFILE can override the use of the default SPFILE to start the instance.

A PFILE can optionally contain a definition to indicate use of an SPFILE. This is the only way to start the instance with an SPFILE in a nondefault location.

To start the database with an SPFILE not in the default location:

`SPFILE=<full path and filename>` must be placed in the PFILE.

Example: `SPFILE=$HOME/ADMIN/PFILE/$ORACLE_SID.ora`.

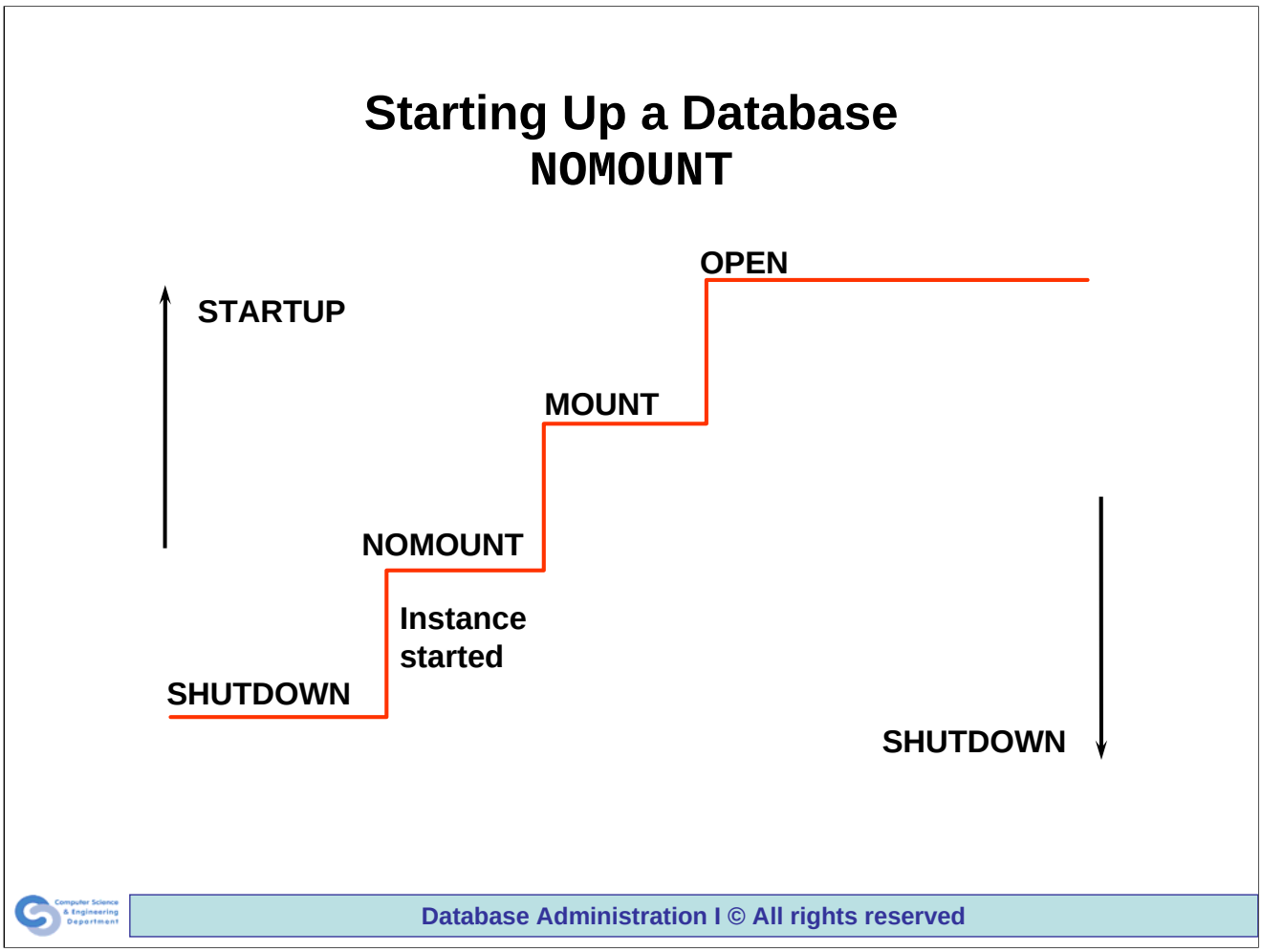
Parameters That Should be Specified in the Initialization Parameter File

Parameter	Description
BACKGROUND_DUMP_DEST	Location where background process trace files are written (LGWR, DBWn, and so on). Also the location for the alert log file
COMPATIBLE	Version of the server with which this instance should be compatible
CONTROL_FILES	Names of the control files
DB_CACHE_SIZE	Specifies the size of the cache for standard block size buffers
DB_NAME	Database identifier of eight characters or fewer. This is the only parameter that is required when a new database is created
SHARED_POOL_SIZE	Size in bytes of the shared pool
USER_DUMP_DEST	Location where user debugging trace files are created on behalf of a user process

Note: The default values depend on the version of the Oracle server.

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	Enables or disables the SQL trace facility for every user session
TIMED_STATISTICS	Enables or disables timing in trace files and in



Starting Up a Database

When starting the database, you select the state in which it starts. The following scenarios describe different stages of starting up an instance.

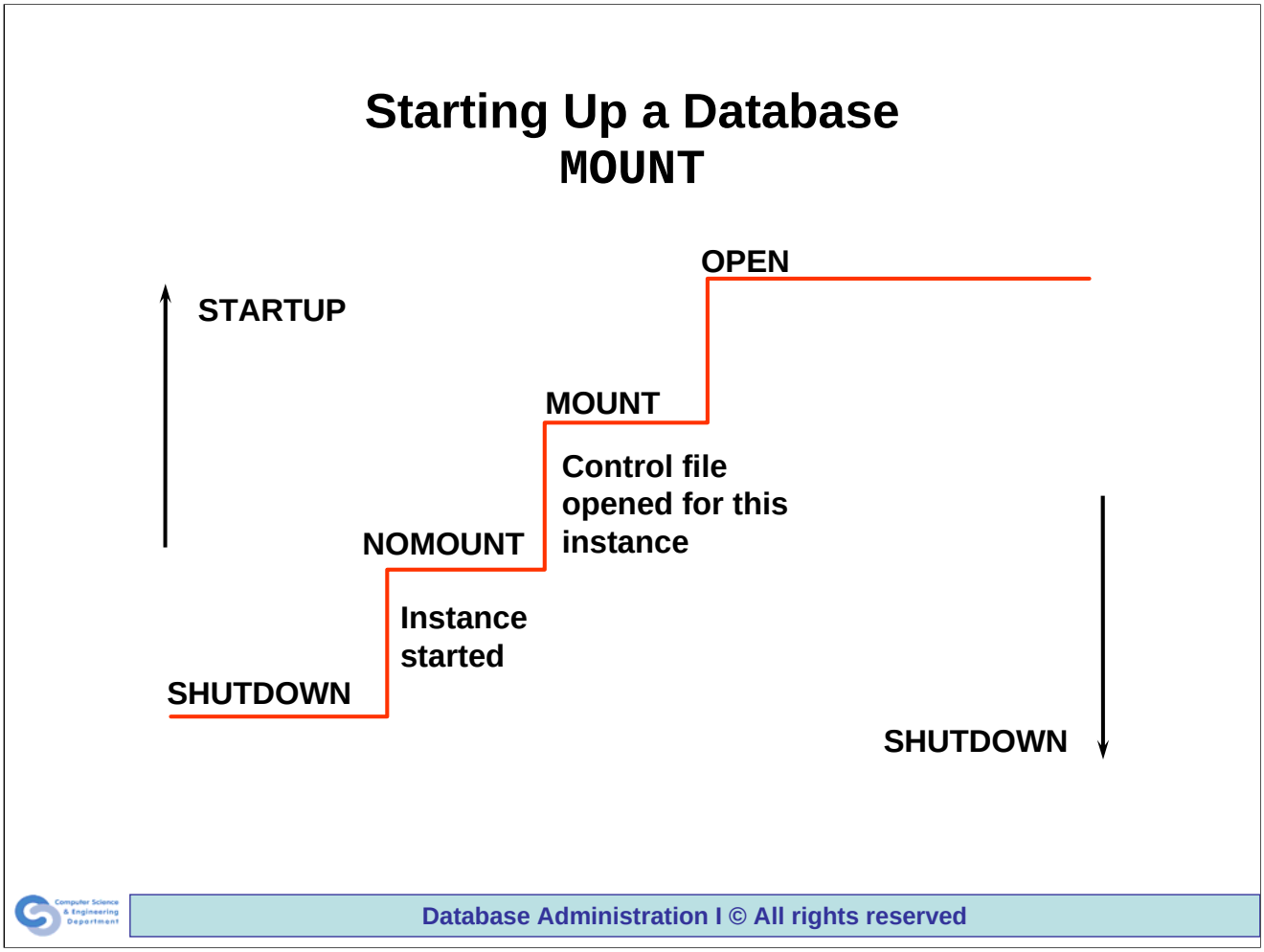
Starting the Instance (`NOMOUNT`)

An instance would be started in the `NOMOUNT` stage only during database creation or the re-creation of control files.

Starting an instance includes the following tasks:

- Reading the initialization file from `$ORACLE_HOME/dbs` in the following order:
  - First `spfileSID.ora`
  - If not found then, `spfile.ora`
  - If not found then, `initSID.ora`
  - Specifying the `PFILE` parameter with `STARTUP` overrides the default behavior.
- Allocating the SGA
- Starting the background processes
- Opening the `alertSID.log` file and the trace files

The database must be named with the `DB_NAME` parameter either in the initialization parameter file or in the `STARTUP` command.



Starting Up a Database (continued)

Mounting the Database (`MOUNT`)

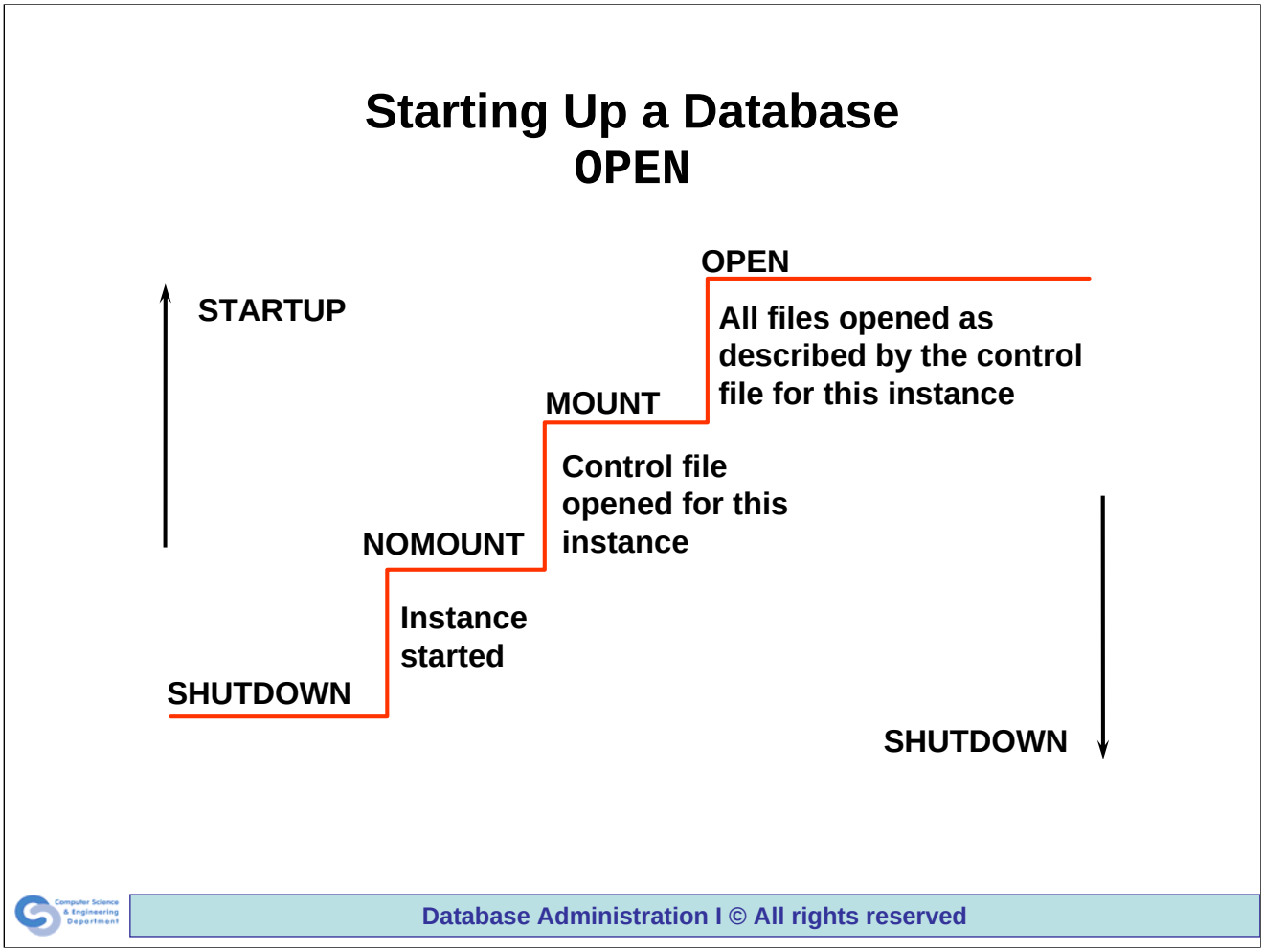
To perform specific maintenance operations, you start an instance and mount a database but do not open the database.

For example, the database must be mounted but not open during the following tasks:

- Renaming data files
- Enabling and disabling online redo log file 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 online redo log files. However, no checks are performed to verify the existence of the data files and online redo log files at this time.



Starting Up a Database (continued)

Opening the Database (OPEN)

Normal database operation means that an instance is started and the database is mounted and open. With normal database operation, any valid user can connect to the database and perform typical data access operations.

Opening the database includes the following tasks:

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

If any of the data files or online redo log files are not present when you attempt to open the database, the Oracle server returns an error.

During this final stage, the Oracle server verifies that all the data files and online redo log files can be opened and checks the consistency of the database. If necessary, the SMON background process initiates instance recovery.

STARTUP Command

Start up the instance and open the database:

```
STARTUP
```

```
STARTUP PFILE=$ORACLE_HOME/dbs/initdb01.ora
```

STARTUP Command

To start up an instance, use the following command:

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

Note: This is not the complete syntax.

where:

`OPEN`: Enables users to access the database

`MOUNT`: Mounts the database for certain DBA activities but does not provide user access to the database

`NOMOUNT`: Creates the SGA and starts up the background processes but does not provide access to the database

`PFILE=`*`parfile`*: Enables a nondefault initialization parameter file to be used to configure the instance

STARTUP Command (continued)

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

Automating Database Startup

On UNIX:

Automating database start up and shut down can be controlled by the entries in a special operating system file; for example, oratab in the /var/opt/oracle directory.

Note: Refer to the installation guide of your operating system for more information.

Troubleshooting

If any errors are encountered while issuing the STARTUP command the SHUTDOWN command must be issued before another STARTUP.

Note: STARTUP and SHUTDOWN commands are SQL*Plus commands, not SQL commands.

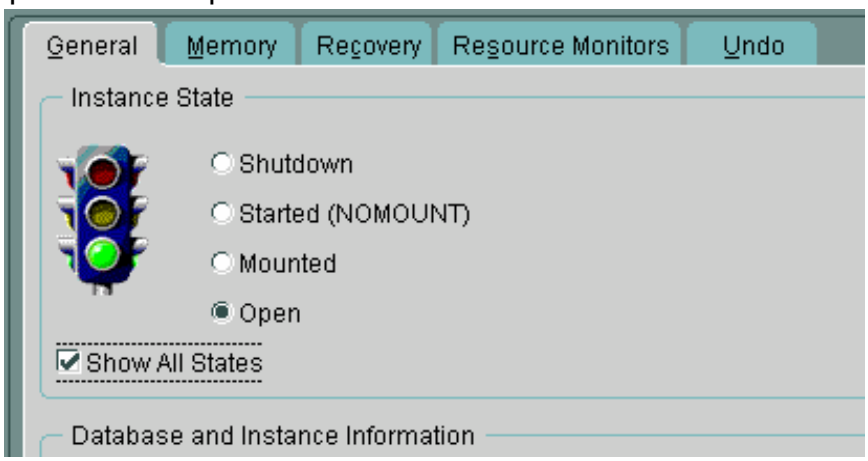
STARTUP Command (continued)

Using Oracle Enterprise Manager to Start Up a Database

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Select the Open option from the General tabbed page.
4. Click Apply.

Note: You must be connected to the database with SYSDBA privileges to perform startup.



ALTER DATABASE Command

- **Change the state of the database from `NOMOUNT` to `MOUNT`:**

```
ALTER DATABASE db01 MOUNT;
```

- **Open the database as a read-only database:**

```
ALTER DATABASE db01 OPEN READ ONLY;
```

ALTER DATABASE Command

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

```
ALTER DATABASE { MOUNT | OPEN }
```

To prevent data from being modified by user transactions, the database can be opened in read-only mode.

To start up an instance, use the following command:

```
ALTER DATABASE OPEN [READ WRITE| READ ONLY]
```

where:

READ WRITE: Opens the database in read/write mode, so that users can generate online redo log files.

READ ONLY: Restricts users to read-only transactions, preventing them from generating online redo log file information.

Opening a Database in Restricted Mode

- **Use the STARTUP command to restrict access to a database:**

```
STARTUP RESTRICT
```

- **Use the ALTER SYSTEM command to place an instance in restricted mode:**

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


Opening a Database in Restricted Mode

A restricted session is useful, for example, when you perform structure maintenance or a database export and import. 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 ]
```

where:

ENABLE RESTRICTED SESSION: Enables future logins only for users who have the RESTRICTED SESSION privilege

DISABLE RESTRICTED SESSION: Disables RESTRICTED SESSION so that users who do not have the privilege can log on

Terminate Sessions

After placing an instance in restricted mode, you may want to kill all current user sessions before performing administrative tasks. This can be done by the following:

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

where:

integer1: Value of the SID column in the V$SESSION view

integer2: Value of the SERIAL# column in the V$SESSION view

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

Effects of Terminating a Session

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

Roll back the user's current transaction.

Release all currently held table or row locks.

Free all resources currently reserved by the user.

Opening a Database in Restricted Mode (continued)

Using Oracle Enterprise Manager to Open a Database in Restricted Mode

From the OEM Console:

Navigate to Instance > Configuration.

Highlight Configuration.

Select the General tabbed page.

Select the Shutdown option under the Instance State.

Click Apply. The Shutdown Options dialog box will appear.

Select the Immediate option.

Click OK.

Select Close when processing is complete.

Select the Open option under Instance State.

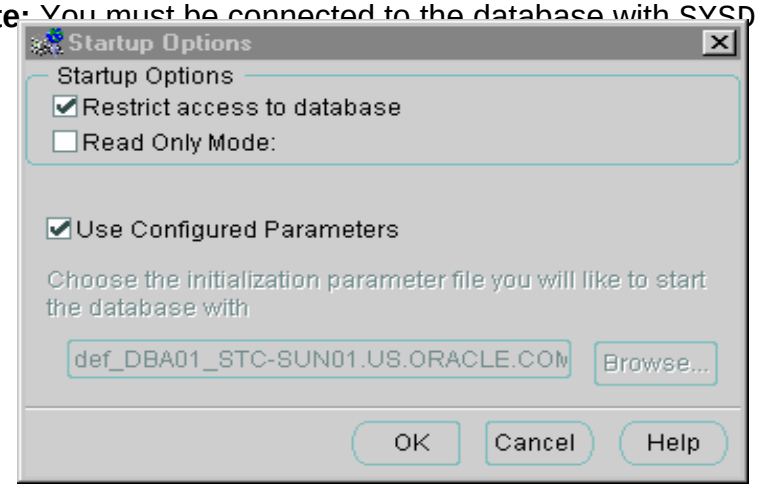
Click OK. The Startup Options dialog box will appear.

Select the “Restrict access to database” option.

Click OK.

Click Close when processing complete.

Note: You must be connected to the database with SYSDBA privileges.



Opening a Database in Read-Only Mode

- **Opening a database in read-only mode:**

```
STARTUP MOUNT
ALTER DATABASE OPEN READ ONLY;
```

- **Can be used to:**
  - **Execute queries**
  - **Execute disk sorts using locally managed tablespaces**
  - **Take data files offline and online, but not tablespaces**
  - **Perform recovery of offline data files and tablespaces**

Opening a Database in Read-Only Mode

A database can be opened as read-only, as long as it is not already open in read/write mode. The feature is especially useful for a standby database to offload query processing from the production database.

If a query needs to use a temporary tablespace, for example, to do disk sorts, the current user must have a locally managed tablespace assigned as the default temporary tablespace; otherwise, the query fails.

Note: Locally managed tablespaces are discussed in a later lesson.

Read-only mode does not restrict database recovery or operations that change the database state without generating redo data. For example, in read-only mode:

Data files can be taken offline and online.

Recovery of offline data files and tablespaces can be performed.

Disk writes to other files, such as control files, operating system audit trails, trace files, and alert log files, can continue in read-only mode.

Opening a Database in Read-Only Mode (continued)

Using Oracle Enterprise Manager to Start a Database in Read-Only Mode

From the OEM Console:

Navigate to Instance > Configuration.

Highlight Configuration.

Select the General tabbed page.

Select the Shutdown option under Instance State.

Click Apply. The Shutdown Options dialog box will appear.

Select the Immediate option.

Click OK.

Select Close when processing is complete.

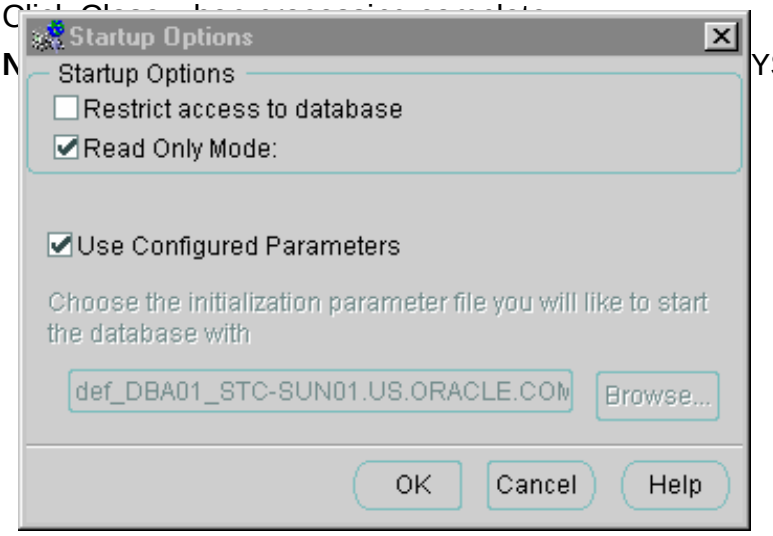
Select the Open option under Instance State.

Click OK. The Startup Options dialog box will appear.

Select the Read Only Mode check box.

Click OK.

Cl



N YSDBA privileges.

Shutting Down the Database

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 a checkpoint and close files	No	Yes	Yes	Yes

Shutdown mode:

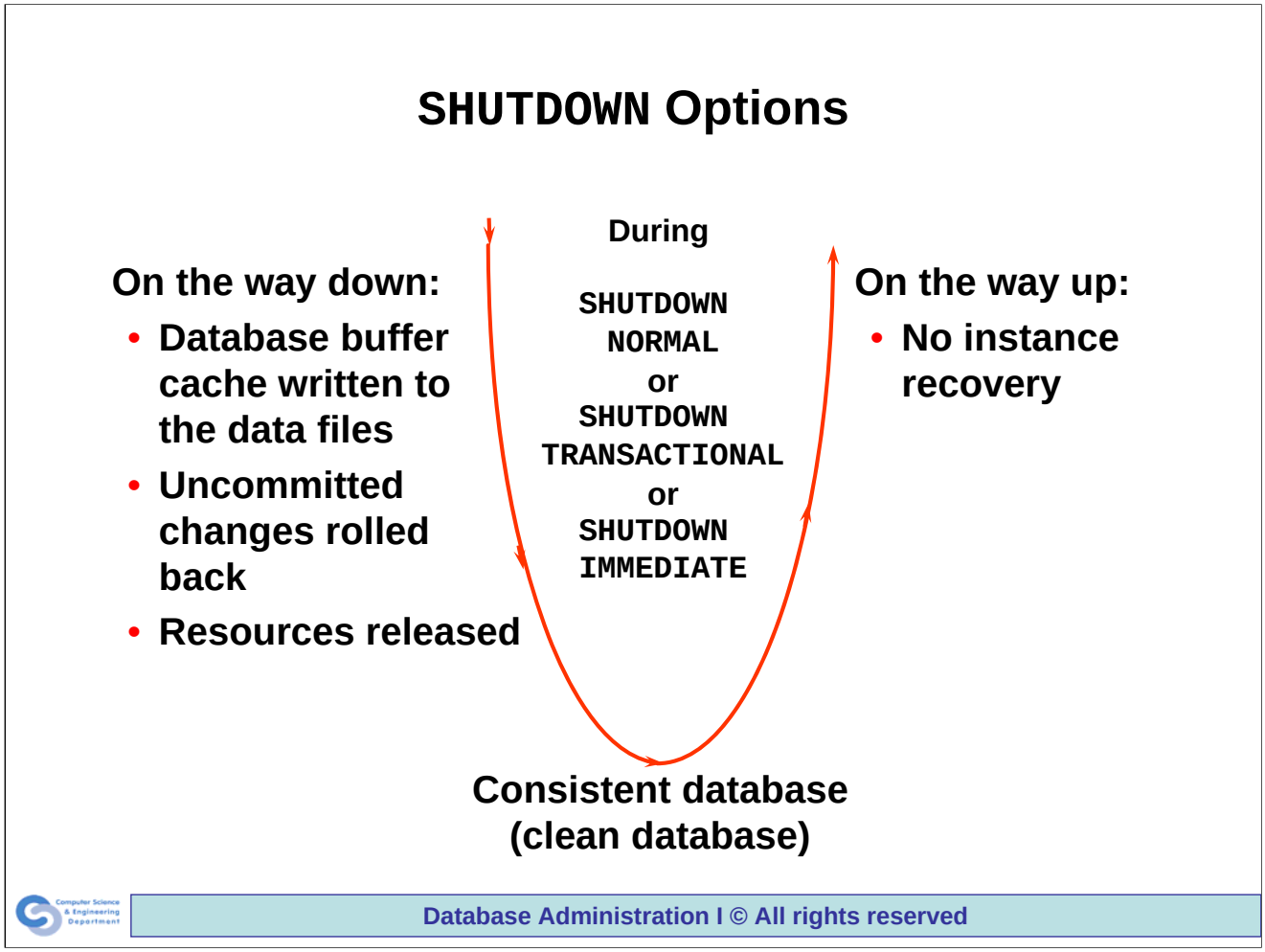
- **A = ABORT**
- **I = IMMEDIATE**
- **T = TRANSACTIONAL**
- **N = NORMAL**

Shutting Down the Database

Shut down the database to make operating system offline backups of all physical structures and to have modified static initialization parameters take effect when restarted.

To shut down an instance you must connect as SYSOPER or SYSDBA and use the following command:

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



SHUTDOWN Options

SHUTDOWN NORMAL

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

No new connections can be made.

The Oracle server waits for all users to disconnect before completing the shutdown.

Database and redo buffers are written to disk.

Background processes are terminated, and the SGA is removed from memory.

Oracle closes and dismounts the database before shutting down the instance.

The next startup does not require an instance recovery.

SHUTDOWN TRANSACTIONAL

A transactional shut down prevents clients from losing work. A transactional database shut down 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 that is in progress.

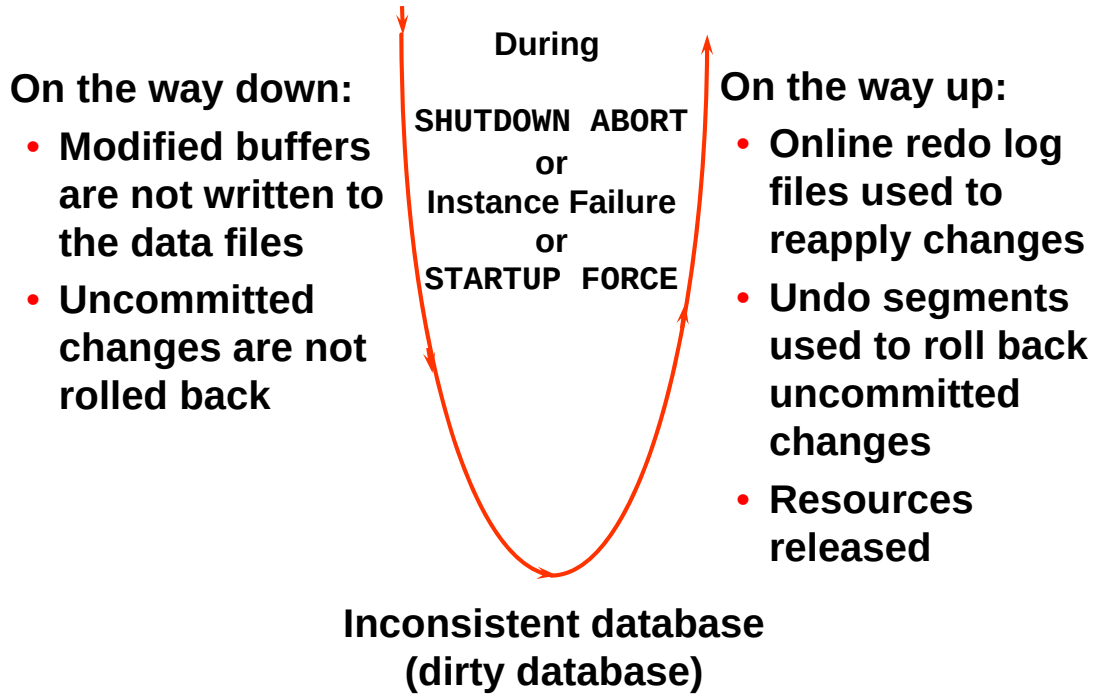
SHUTDOWN Options (continued)

SHUTDOWN IMMEDIATE

Immediate database shut down proceeds with the following conditions:

- Current SQL statements being processed by Oracle are not completed.
- The Oracle server does not wait for the users, who are 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 start up does not require an instance recovery.

SHUTDOWN Options



SHUTDOWN Options (continued)

SHUTDOWN ABORT

If the NORMAL and IMMEDIATE shut down options do not work, you can abort the current database instance. Aborting an instance 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.

Database and redo buffers are not written to disk.

Uncommitted transactions are not rolled back.

The instance is terminated without closing the files.

The database is not closed or dismounted.

The next start up requires instance recovery, which occurs automatically.

Note: It is not advisable to back up a database that is in an inconsistent state.

SHUTDOWN Options (continued)

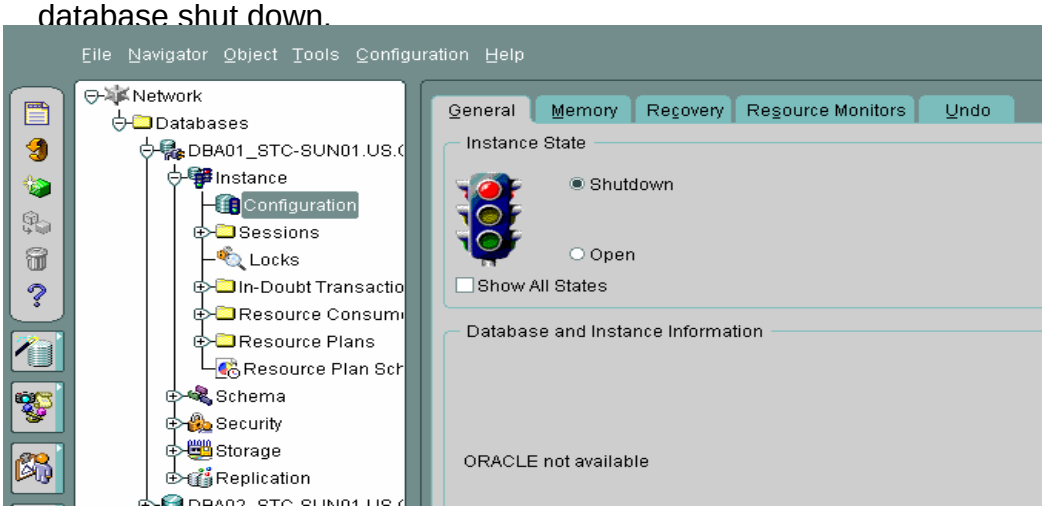
Using Oracle Enterprise Manager to Shut Down a Database

From the OEM Console:

1. Navigate to Instance > Configuration.
2. Highlight Configuration.
3. Select the Shutdown option from the General tabbed page.
4. Click Apply.

Note: You must be connected to the database with SYSDBA privileges to perform a

database shut down.



Monitoring an Instance Using Diagnostic Files

- **Diagnostic files**
  - **Contain information about significant events encountered**
  - **Used to resolve problems**
  - **Used to better manage the database on a day-to-day basis**
- **Several types exist:**
  - **`alertSID.log` file**
  - **Background trace files**
  - **User trace files**

Monitoring an Instance Using Diagnostic Files

Diagnostic files are a means to capture information about the database's activities. They are also useful tools for you when you are managing an instance. Several types exist. The type of diagnostic file created depends on the problem that occurred or the information that is needed to be disseminated.

Alert log file (`alertSID.log`): Information for day-to-day operation of the database

Background trace files: Vital information when background processes, such as SMON, PMON, DBWn, and others fail

User trace files: Vital information for fatal user errors or user forced traced files

Alert Log File

- **`alertSID.log` file:**
  - **Records the commands**
  - **Records results of major events**
  - **Used for day-to-day operational information**
  - **Used for diagnosing database errors**
- **Each entry has a time stamp associated with it**
- **Must be managed by DBA**
- **Location defined by `BACKGROUND_DUMP_DEST`**

Alert Log File

Each Oracle instance has an alert log file. If not already created, it is created during instance start up. The alert log file must be managed by the DBA. It continues to grow while the database continues to work. The alert log file should be the first place you look when diagnosing day-to-day operations or errors. The alert log file also contains pointers to trace files for more detailed information.

The alert log file keeps a record of the following information:

When the database was started or shut down

A list of all nondefault initialization parameters

The start up of background processes

The thread being used by the instance

The log sequence number LGWR is writing to

Information regarding a log switch

Creation of tablespaces and undo segments

Alter statements that have been issued

Information regarding error messages such as ORA-600 and extent errors

Alert Log File (continued)

The `alert_SID.log` location is defined by the `BACKGROUND_DUMP_DEST` initialization parameter.

Background Trace Files

- **Background trace files**
  - **Log errors detected by any background process**
  - **Are used to diagnose and troubleshoot errors**
- **Created when a background process encounters an error**
- **Location defined by `BACKGROUND_DUMP_DEST`**

Background Trace Files

Background trace files are used to log errors that have been encountered by a background process, such are SMON, PMON, DBWn, and other background processes. These files exist only when an error requires writing to the trace files.You use these files to diagnose and troubleshoot problems. Initially when a background trace file is created it contains header information indicating the version number of the data server and the operating system.

Naming convention for user trace file: `sid_processname_PID.trc` (`db01_lgwr_23845.trc`).

Its location is defined by the `BACKGROUND_DUMP_DEST` initialization parameter.

User Trace Files

- **User trace files**
  - **Produced by the user process**
  - **Can be generated by a server process**
  - **Contain statistics for traced SQL statements**
  - **Contain user error messages**
- **Created when a user encounters user session errors**
- **Location is defined by `USER_DUMP_DEST`**
- **Size defined by `MAX_DUMP_FILE_SIZE`**

User Trace Files

User trace files contain statistics for traced SQL statements, which are useful for SQL tuning. In addition, user trace files contain user error messages.

Naming convention for user trace file:
`sid_ora_PID.trc(db01_ora_23845.trc)`.

Its location is defined by the `USER_DUMP_DEST` initialization parameter.

Enabling or Disabling User Tracing

- **Session level:**
  - **Using the `ALTER SESSION` command:**
    `ALTER SESSION SET SQL_TRACE = TRUE`
  - **Executing DBMS procedure:**
    `dbms_system.SET_SQL_TRACE_IN_SESSION`
- **Instance level**
  - **Setting the initialization parameter:**
    `SQL_TRACE = TRUE`

Enabling or Disabling User Tracing

Setting `SQL_TRACE=TRUE` at the instance level will produce a significant volume of trace data. This option should be used with care.

User tracing is covered in detail in the *Oracle9i SQL Statement Tuning* course.

Enabling or Disabling User Tracing

Using Oracle Enterprise Manager to Enable or Disable User Tracing

From the OEM Console:

Navigate to Instance > Configuration.

Highlight Configuration.

Select All Initialization Parameters from the General tabbed page.

Set the parameter `SQL_TRACE = TRUE`.

Click Apply.

Summary

In this lesson, you should have learned how to:

- Create and manage initialization parameter files
- Start up and shut down an instance
- Monitor and use diagnostic files