16

Oracle Database Instance

This chapter explains the nature of an Oracle database instance, the parameter and diagnostic files associated with an instance, and what occurs during instance creation and the opening and closing of a database.

> **Note:**
>
> For the purposes of this chapter, the term "database" refers to a multitenant container database (CDB) unless otherwise noted.

- Introduction to the Oracle Database Instance
  A **database instance** is a set of memory structures that manage database files.
- Overview of Database Instance Startup and Shutdown
  A **database instance** provides user access to a database. The instance and the database can be in various states.
- Overview of Checkpoints
  A **checkpoint** is a crucial mechanism in consistent database shutdowns, instance recovery, and Oracle Database operation generally.
- Overview of Instance Recovery
  Instance recovery is the process of applying records in the online redo log to data files to reconstruct changes made after the most recent checkpoint.
- Overview of Parameter Files
  To start a database instance, Oracle Database must read either a **server parameter file**, which is recommended, or a **text initialization parameter file**, which is a legacy implementation. These files contain a list of configuration parameters.
- Overview of Diagnostic Files
  Oracle Database includes a **fault diagnosability infrastructure** for preventing, detecting, diagnosing, and resolving database problems. Problems include critical errors such as code bugs, metadata corruption, and customer data corruption.

Introduction to the Oracle Database Instance

A **database instance** is a set of memory structures that manage database files.

At the physical level, a CDB is a set of files on disk created by the `CREATE DATABASE` statement. A CDB contains one or more user-created PDBs. A PDB contains its own set of data files within the overall set of data files that belongs to the CDB. The database instance manages the data associated with the CDB and its PDBs and serves their users.

Every running CDB is associated with at least one Oracle database instance. Because an instance exists in memory and a database (in the narrowest sense of term) is a set of files on disk, an instance can exist without a database and a database can exist without an instance.

Database Instance Structure

When an instance is started, Oracle Database allocates a memory area called the **system global area (SGA)** and starts one or more **background processes**.

The SGA serves various purposes, including the following:

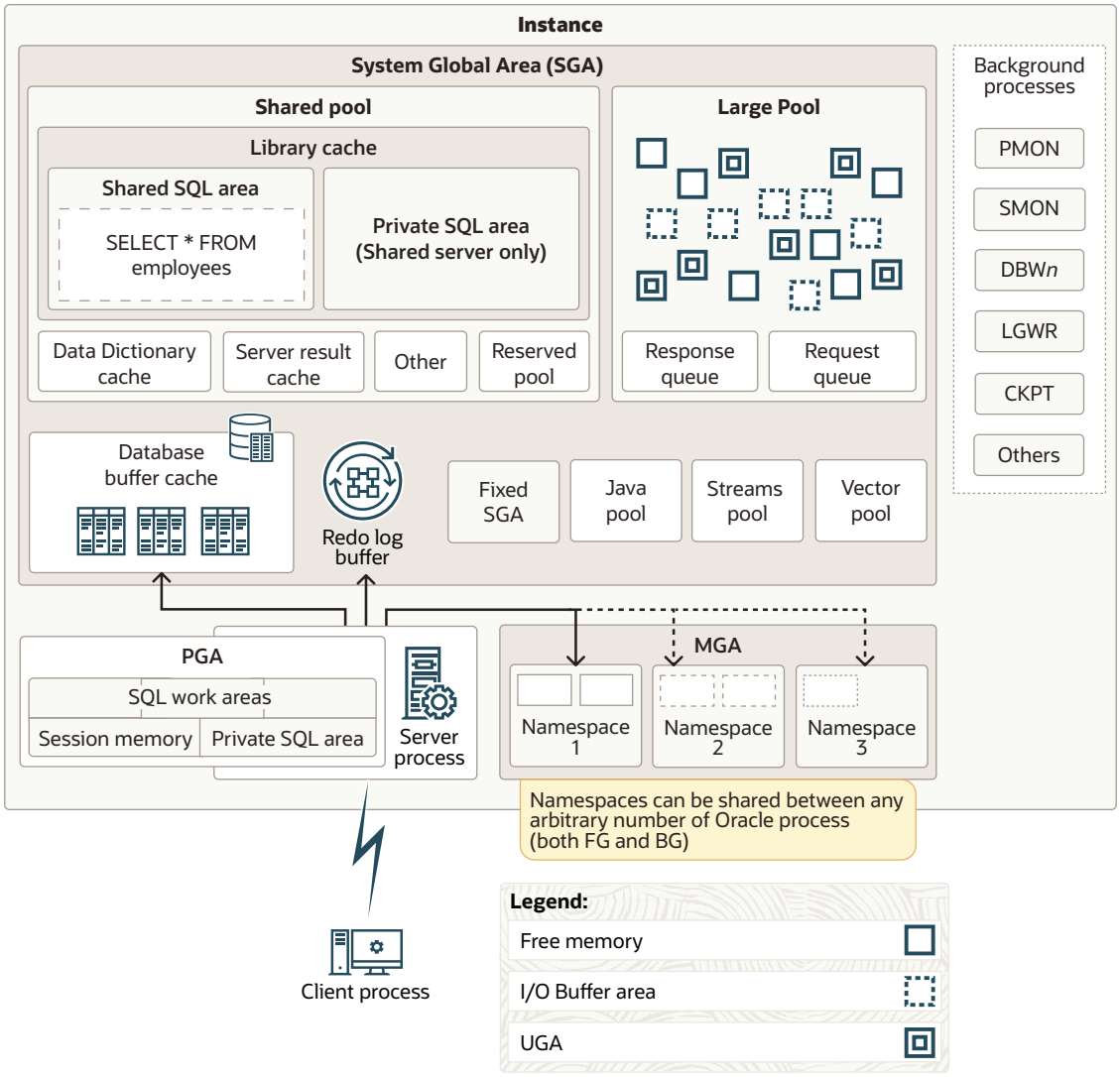
- Maintaining internal data structures that many processes and threads access concurrently
- Caching data blocks read from disk
- Buffering redo data before writing it to the online redo log files
- Storing SQL execution plans

Oracle processes running on a single computer share the SGA. The way in which Oracle processes associate with the SGA varies according to operating system.

A database instance also includes background processes and server processes. Each process has its own dedicated memory considered as part of the instance. The instance continues to function when server processes terminate.

The following graphic shows the main components of an Oracle database instance.

Figure 16-1 Database Instance



See Also:

- "Overview of the System Global Area (SGA)"
- "Overview of Background Processes"

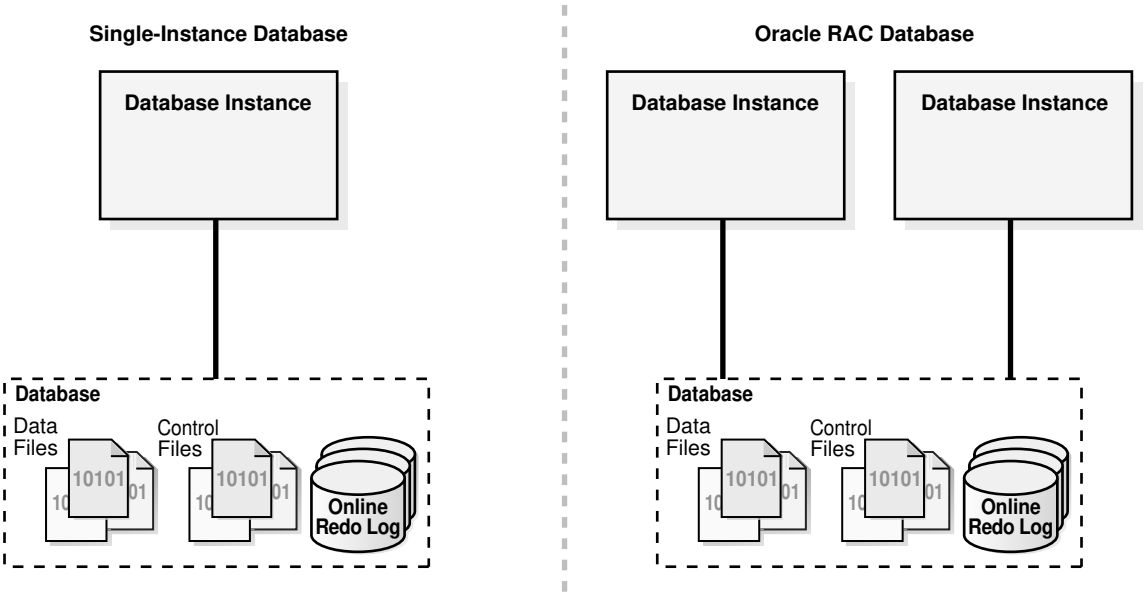
Database Instance Configurations

Oracle Database runs in either a single-instance configuration or an Oracle Real Application Clusters (Oracle RAC) configuration. These configurations are mutually exclusive.

In a single-instance configuration, a one-to-one relationship exists between the database and a database instance. In Oracle RAC, a one-to-many relationship exists between the database and database instances.

The following figure shows possible database instance configurations.

Figure 16-2 Database Instance Configurations



Whether in a single-instance or Oracle RAC configuration, a database instance is associated with only one database at a time. You can start a database instance and **mount** (associate the instance with) one database, but not mount two databases simultaneously with the same instance.

> **Note:**
>
> This chapter discusses a single-instance database configuration unless otherwise noted.

Multiple instances can run concurrently on the same computer, each accessing its own database. For example, a computer can host two distinct databases: `prod1` and `prod2`. One database instance manages `prod1`, while a separate instance manages `prod2`.

> **See Also:**
>
> *Oracle Real Application Clusters Administration and Deployment Guide* for information specific to Oracle RAC

Read/Write and Read-Only Instances

Every database instance is either read/write or read-only.

A read/write database instance, which is the default, can process DML and supports direct connections from client applications. In contrast, a read-only database instance can process

queries, but does not support modification DML (`UPDATE`, `DELETE`, `INSERT`, and `MERGE`) or direct client connections.

> **Note:**
>
> Unless stated otherwise in this manual, all references to database instances are to read/write instances.

In previous releases, all database instances—unless they accessed a standby database—were read/write. Starting in Oracle Database 12c Release 2 (12.2), read-only and read/write instances can coexist within a single database. This configuration is useful for parallel SQL statements that both query and modify data, because both read/write and read-only instances can query, while the read/write instances modify.

Unlike read/write instances, read-only instances have the following characteristics:

- Can only open a database that has already been opened by a read/write instance
- Disable many background processes, including the checkpoint and archiver processes, which are not necessary
- Can mount a disabled redo thread or a thread without any online redo log

To designate an instance as read-only, set the `INSTANCE_MODE` initialization parameter to `READ_ONLY`. The default value of the parameter is `READ_WRITE`.

> **See Also:**
>
> - "Overview of Background Processes" to learn more about the checkpoint and archiver background processes
> - "Overview of the Online Redo Log"
> - *Oracle Database Reference* to learn more about the `INSTANCE_MODE` initialization parameter

Duration of a Database Instance

A database instance begins when it is created with the `STARTUP` command and ends when it is terminated.

During this period, a database instance can associate itself with one and only one database. Furthermore, the instance can mount a database only once, close it only once, and open it only once. After a database has been closed or shut down, you must start a *different* instance to mount and open this database.

The following table illustrates a database instance attempting to reopen a database that it previously closed.

Table 16-1 Duration of an Instance

Statement	Explanation

`SQL> STARTUP`
`ORACLE instance started.`
 	
`Total System Global Area 468729856 bytes`
`Fixed Size 1333556 bytes`
`Variable Size 440403660 bytes`
`Database Buffers 16777216 bytes`
`Redo Buffers 10215424 bytes`
`Database mounted.`
`Database opened.`
	The `STARTUP` command creates an instance, which mounts and opens the database.

`SQL> SELECT`
`TO_CHAR(STARTUP_TIME,'MON-DD-RR HH24:MI:SS')`
`AS "Inst Start Time" FROM V$INSTANCE;`
 	
`Inst Start Time`
`------------------`
`JUN-18-14` **`13:14:48`**
	This query shows the time that the current instance was started.

`SQL> SHUTDOWN IMMEDIATE`	The instance closes the database and shuts down, ending the life of this instance.

`SQL> STARTUP`
`Oracle instance started.`
`. . .`
	The `STARTUP` command creates a new instance and mounts and open the database.

`SQL> SELECT`
`TO_CHAR(STARTUP_TIME, 'MON-DD-RR HH24:MI:SS')`
`AS "Inst Start Time" FROM V$INSTANCE;`
 	
`Inst Start Time`
`------------------`
`JUN-18-14` **`13:16:40`**
	This query shows the time that the current instance was started. The different start time shows that this instance is different from the one that shut down the database.

Identification of a Database Instance

Multiple database instances can reside on a single host. Therefore, you must have a means of specifying which instance you want to access.

Oracle Optimal Flexible Architecture (OFA) rules are a set of configuration guidelines created to ensure well-organized Oracle installations. The examples in this section assume an OFA architecture.

This section contains the following topics:

- Oracle Base Directory
  The Oracle base directory stores the binaries for Oracle products.
- Oracle Home Directory
  The **Oracle home** is the software location for an Oracle database.
- Oracle System Identifier (SID)
  The **system identifier (SID)** is a unique name for an Oracle database instance on a specific host.

> **See Also:**
>
> *Oracle Database Installation Guide for Linux* for an overview of the OFA

Oracle Base Directory

The Oracle base directory stores the binaries for Oracle products.

The Oracle Base directory is the database home directory for Oracle Database installation owners. There can be many Oracle Database installations on a host, and many Oracle Database software installation owners.

The following example shows the Oracle base directory of the operating system user account `oracle`:

```
/u01/app/oracle
```

In the preceding path, `/u01/` is the mount point, and `/u01/app/` is the subtree for application software.

> **See Also:**
>
> *Oracle Database Installation Guide for Linux* for an overview of the Oracle base

Oracle Home Directory

The **Oracle home** is the software location for an Oracle database.

You must specify a new Oracle home directory for each new installation of Oracle Database software. Starting in Oracle Database 21c, the Oracle home is read-only. The Oracle home stores static files such as binaries.

By default, the Oracle home directory is a descendent within the Oracle base (`ORACLE_BASE`) directory tree. You can install this release, or earlier releases of the database software, more

than once on the same host, in different Oracle home directories within a single Oracle base. Multiple databases, of different versions and owned by different user accounts, can coexist concurrently.

The following example shows the full path names of three different Oracle homes, all within the same Oracle base directory of `/u01/app/oracle/`:

```
/u01/app/oracle/product/19.3.0/dbhome_1
/u01/app/oracle/product/19.3.0/dbhome_2
/u01/app/oracle/product/20.1.0/dbhome_1
```

The part of the path name after the Oracle base (`/u01/app/oracle/`) includes the product release number (for example, `19.3.0`) and Oracle home relative directory (for example, `dbhome_1`). The `/u01/app/oracle/product/19.3.0/` directory contains two separate Oracle homes: `dbhome_1` and `dbhome_2`.

The Oracle base home (`ORACLE_BASE_HOME`) directory, which is located in `ORACLE_BASE/homes/home_name`, stores dynamic files specific to an Oracle home. The Oracle base configuration directory (`ORACLE_BASE_CONFIG`), which is shared by all Oracle homes in an Oracle base, stores instance-specific dynamic files.

In the following example, the first path is an Oracle home, and the second path is the Oracle base home for this Oracle home:

```
/u01/app/oracle/product/20.1.0/dbhome_1
/u01/app/oracle/homes/dbhome_1
```

> **See Also:**
>
> *Oracle Database Installation Guide for Linux* for an overview of the Oracle home

Oracle System Identifier (SID)

The **system identifier (SID)** is a unique name for an Oracle database instance on a specific host.

On UNIX and Linux, Oracle Database uses the SID and Oracle home values to create a key to shared memory. Also, Oracle Database uses the SID by default to locate the initialization parameter file, which locates relevant files such as the database control files.

On most platforms, the `ORACLE_SID` environment variable sets the SID, and the `ORACLE_HOME` variable sets the Oracle home. When connecting to a database instance, clients can specify the SID in an Oracle Net connection or use a net service name. Oracle Database converts a net service name to an `ORACLE_HOME` and `ORACLE_SID`.

Instance-specific files are stored separately in the Oracle base. Files whose names include the SID reside in the `dbs` subdirectory of the Oracle base configuration directory (`ORACLE_BASE_CONFIG`). Because of this separation, you can create a database using software in an existing Oracle home, and then start an instance for this database using software that resides in a new Oracle home.

> **See Also:**
>
> - "Service Names"
> - *Oracle Database Administrator's Guide* to learn how to specify an Oracle SID

Overview of Database Instance Startup and Shutdown

A **database instance** provides user access to a database. The instance and the database can be in various states.

- Overview of Instance and Database Startup
  Typically, you manually start an instance, and then mount and open the database, making it available for users. You can use the SQL*Plus `STARTUP` command, Oracle Enterprise Manager (Enterprise Manager), or the SRVCTL utility to perform these steps.
- Overview of Database and Instance Shutdown
  In a typical use case, you manually shut down the database, making it unavailable for users while you perform maintenance or other administrative tasks. You can use the SQL*Plus `SHUTDOWN` command or Enterprise Manager to perform these steps.

Overview of Instance and Database Startup

Typically, you manually start an instance, and then mount and open the database, making it available for users. You can use the SQL*Plus `STARTUP` command, Oracle Enterprise Manager (Enterprise Manager), or the SRVCTL utility to perform these steps.

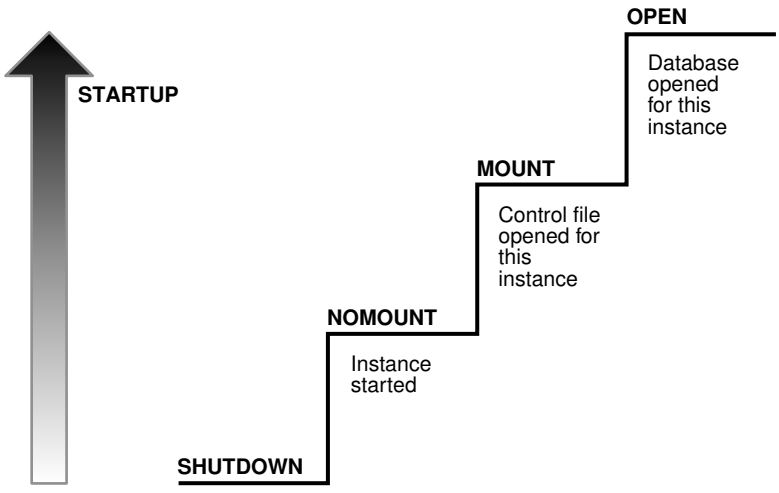
To start a database instance using Oracle Net, the following must be true:

- The database is statically registered with an Oracle Net listener.
- Your client is connected to the database with the `SYSDBA` privilege.

The listener creates a dedicated server, which can start the database instance.

The following graphic shows the database progressing from a shutdown state to an open state.

Figure 16-3 Instance and Database Startup Sequence



A database goes through the following phases when it proceeds from a shutdown state to an open database state.

Table 16-2 Steps in Instance Startup

Phase	Mount State	Description	To Learn More
1	Instance started without mounting database	The instance is started, but is not yet associated with a database.	"How an Instance Is Started"
2	Database mounted	The instance is started and is associated with a database by reading its control file. The database is closed to users.	"How a Database Is Mounted"
3	Database open	The instance is started and is associated with an open database. The data contained in the data files is accessible to authorized users.	"How a Database Is Opened"

- Connection with Administrator Privileges
  Database startup and shutdown are powerful administrative options that are restricted to users who connect to Oracle Database with administrator privileges.
- How an Instance Is Started
  When Oracle Database starts an instance, it proceeds through stages.
- How a Database Is Mounted
  The instance **mounts** a database to associate the database with this instance.
- How a Database Is Opened
  Opening a mounted database makes it available for normal database operation.

See Also:

- "The Oracle Net Listener"
- "Overview of Control Files"
- *Oracle Database Administrator's Guide* to learn how to start an instance
- *Oracle Database Administrator's Guide* to learn how to use SRVCTL

Connection with Administrator Privileges

Database startup and shutdown are powerful administrative options that are restricted to users who connect to Oracle Database with administrator privileges.

Normal users do not have control over the current status of an Oracle database. Depending on the operating system, one of the following conditions establishes administrator privileges for a user:

- The operating system privileges of the user enable the user to connect using administrator privileges.
- The user is granted special system privileges, and the database uses password files to authenticate database administrators over the network.

The following special system privileges enable access to a database instance even when the database is not open:

- SYSDBA
- SYSOPER
- SYSBACKUP
- SYSDG
- SYSKM

Control of the preceding privileges is outside of the database itself. When you connect to a database with the SYSDBA system privilege, you are in the schema owned by SYS. When you connect as SYSOPER, you are in the public schema. SYSOPER privileges are a subset of SYSDBA privileges.

> **See Also:**
>
> - "SYS and SYSTEM Accounts"
> - *Oracle Database Security Guide* to learn about predefined administrative accounts
> - *Oracle Database Administrator's Guide* to learn about system privileges
> - *Oracle Database Installation Guide* to learn more about operating system privilege groups

How an Instance Is Started

When Oracle Database starts an instance, it proceeds through stages.

The stages are as follows:

1. Searches for a server parameter file in a platform-specific default location and, if not found, for a text initialization parameter file (specifying STARTUP with the SPFILE or PFILE parameters overrides the default behavior)
2. Reads the parameter file to determine the values of initialization parameters
3. Allocates the SGA based on the initialization parameter settings
4. Starts the Oracle background processes
5. Opens the alert log and trace files and writes all explicit parameter settings to the alert log in valid parameter syntax

At this stage, no database is associated with the instance. Scenarios that require a NOMOUNT state include database creation and certain backup and recovery operations.

> **See Also:**
>
> *Oracle Database Administrator's Guide* to learn how to manage initialization parameters using a server parameter file

How a Database Is Mounted

The instance **mounts** a database to associate the database with this instance.

To mount the database, the instance obtains the names of the database control files specified in the `CONTROL_FILES` initialization parameter and opens the files. Oracle Database reads the control files to find the names of the data files and the online redo log files that it will attempt to access when opening the database.

In a mounted database, the database is closed and accessible only to database administrators. Administrators can keep the database closed while completing specific maintenance operations. However, the database is not available for normal operations.

If Oracle Database allows multiple instances to mount the same database concurrently, then the `CLUSTER_DATABASE` initialization parameter setting can make the database available to multiple instances. Database behavior depends on the setting:

- If `CLUSTER_DATABASE` is `false` (default) for the first instance that mounts a database, then only this instance can mount the database.
- If `CLUSTER_DATABASE` is `true` for the first instance, then other instances can mount the database if their `CLUSTER_DATABASE` parameter settings are set to `true`. The number of instances that can mount the database is subject to a predetermined maximum specified when creating the database.

> **See Also:**
>
> - *Oracle Database Administrator's Guide* to learn how to mount a database
> - *Oracle Real Application Clusters Administration and Deployment Guide* for more information about the use of multiple instances with a single database

How a Database Is Opened

Opening a mounted database makes it available for normal database operation.

Any valid user can connect to an open database and access its information. Usually, a database administrator opens the database to make it available for general use.

When you open the database, Oracle Database performs the following actions:

- Opens the online data files in tablespaces other than undo tablespaces

  If a tablespace was offline when the database was previously shut down, then the tablespace and its corresponding data files will be offline when the database reopens.
- Acquires an undo tablespace

  If multiple undo tablespaces exists, then the `UNDO_TABLESPACE` initialization parameter designates the undo tablespace to use. If this parameter is not set, then the first available undo tablespace is chosen.
- Opens the online redo log files

- Read-Only Mode
  By default, the database opens in **read/write mode**. In this mode, users can make changes to the data, generating redo in the online redo log. Alternatively, you can open in **read-only mode** to prevent data modification by user transactions.
- Database File Checks
  If any of the data files or redo log files are not present when the instance attempts to open the database, or if the files are present but fail consistency tests, then the database returns an error. Media recovery may be required.

> **See Also:**
>
> - "Online and Offline Tablespaces"
> - *Oracle Database Backup and Recovery User's Guide* to learn more about data repair

Read-Only Mode

By default, the database opens in **read/write mode**. In this mode, users can make changes to the data, generating redo in the online redo log. Alternatively, you can open in **read-only mode** to prevent data modification by user transactions.

> **Note:**
>
> By default, a physical standby database opens in read-only mode.

Read-only mode restricts database access to read-only transactions, which cannot write to data files or to online redo log files. However, the database can perform recovery or operations that change the database state without generating redo. For example, in read-only mode:

- Data files can be taken offline and online. However, you cannot take permanent tablespaces offline.
- Offline data files and tablespaces can be recovered.
- The control file remains available for updates about the state of the database.
- Temporary tablespaces created with the `CREATE TEMPORARY TABLESPACE` statement are read/write.
- Writes to operating system audit trails, trace files, and alert logs can continue.

> **See Also:**
>
> - *Oracle Database Administrator's Guide* to learn how to open a database in read-only mode
> - *Oracle Data Guard Concepts and Administration*

Database File Checks

If any of the data files or redo log files are not present when the instance attempts to open the database, or if the files are present but fail consistency tests, then the database returns an error. Media recovery may be required.

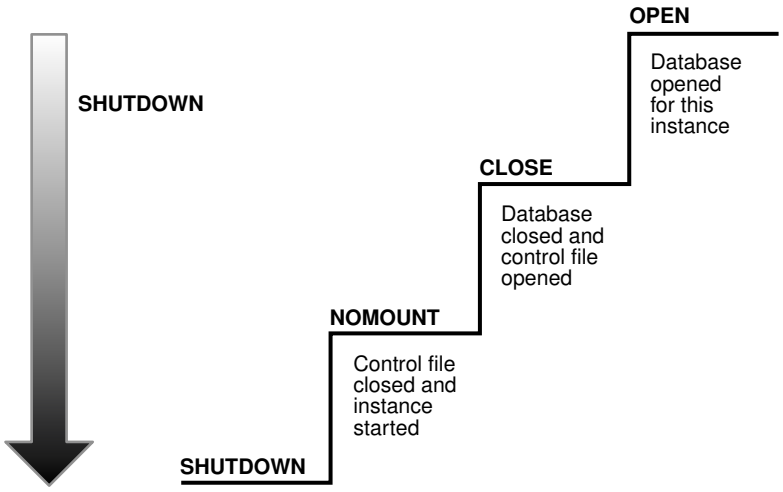
> **See Also:**
>
> *Oracle Database Backup and Recovery User's Guide* for an overview of backup and recovery

Overview of Database and Instance Shutdown

In a typical use case, you manually shut down the database, making it unavailable for users while you perform maintenance or other administrative tasks. You can use the SQL*Plus `SHUTDOWN` command or Enterprise Manager to perform these steps.

The following figure shows the progression from an open state to a consistent shutdown.

Figure 16-4 Instance and Database Shutdown Sequence



Oracle Database automatically performs the following steps whenever an open database is shut down consistently.

Table 16-3 Steps in Consistent Shutdown

Phase	Mount State	Description	To Learn More
1	Database closed	The database is mounted, but online data files and redo log files are closed.	"How a Database Is Closed"

Table 16-3 (Cont.) Steps in Consistent Shutdown

Phase	Mount State	Description	To Learn More
2	Database unmounted	The instance is started, but is no longer associated with the control file of the database.	"How a Database Is Unmounted"
3	Database instance shut down	The database instance is no longer started.	"How an Instance Is Shut Down"

Oracle Database does not go through all of the preceding steps in an instance failure or `SHUTDOWN ABORT`, which immediately terminates the instance.

- Shutdown Modes
  A database administrator with `SYSDBA` or `SYSOPER` privileges can shut down the database using the SQL*Plus `SHUTDOWN` command or Enterprise Manager. The `SHUTDOWN` command has options that determine shutdown behavior.
- How a Database Is Closed
  The database close operation is implicit in a database shutdown. The nature of the operation depends on whether the database shutdown is normal or abnormal.
- How a Database Is Unmounted
  After the database is closed, Oracle Database unmounts the database to disassociate it from the instance.
- How an Instance Is Shut Down
  The final step in database shutdown is shutting down the instance. When the database instance shuts down, the SGA ceases to occupy memory, and the background processes terminate.

> **See Also:**
>
> *Oracle Database Administrator's Guide* to learn how to shut down a database

Shutdown Modes

A database administrator with `SYSDBA` or `SYSOPER` privileges can shut down the database using the SQL*Plus `SHUTDOWN` command or Enterprise Manager. The `SHUTDOWN` command has options that determine shutdown behavior.

The following table summarizes the behavior of the different shutdown modes.

Table 16-4 Database Shutdown Modes

Database Behavior	ABORT	IMMEDIATE	TRANSACTIONAL	NORMAL
Permits new user connections	No	No	No	No
Waits until current sessions end	No	No	No	Yes
Waits until current transactions end	No	No	Yes	Yes

Table 16-4 (Cont.) Database Shutdown Modes

Database Behavior	ABORT	IMMEDIATE	TRANSACTIONAL	NORMAL
Performs a checkpoint and closes open files	No	Yes	Yes	Yes

The possible `SHUTDOWN` statements are:

- `SHUTDOWN ABORT`

  This mode is intended for emergency situations, such as when no other form of shutdown is successful. This mode of shutdown is the fastest. However, a subsequent open of this database may take substantially longer because instance recovery must be performed to make the data files consistent.

  Because `SHUTDOWN ABORT` does not checkpoint the open data files, instance recovery is necessary before the database can reopen. The other shutdown modes do not require instance recovery before the database can reopen.

  > **Note:**
  >
  > In a PDB, issuing `SHUTDOWN ABORT` is equivalent to issuing `SHUTDOWN IMMEDIATE` at the CDB level.

- `SHUTDOWN IMMEDIATE`

  This mode is typically the fastest next to `SHUTDOWN ABORT`. Oracle Database terminates any executing SQL statements and disconnects users. Active transactions are terminated and uncommitted changes are rolled back.

- `SHUTDOWN TRANSACTIONAL`

  This mode prevents users from starting new transactions, but waits for all current transactions to complete before shutting down. This mode can take a significant amount of time depending on the nature of the current transactions.

- `SHUTDOWN NORMAL`

  This is the default mode of shutdown. The database waits for all connected users to disconnect before shutting down.

> **See Also:**
>
> - *Oracle Database Administrator's Guide* to learn about the different shutdown modes
> - *SQL*Plus User's Guide and Reference* to learn about the `SHUTDOWN` command

How a Database Is Closed

The database close operation is implicit in a database shutdown. The nature of the operation depends on whether the database shutdown is normal or abnormal.

How a Database Is Closed During Normal Shutdown

When a database is closed as part of a SHUTDOWN with any option other than ABORT, Oracle Database writes data in the SGA to the data files and online redo log files.

Afterward, the database closes online data files and online redo log files. Any offline data files of offline tablespaces have been closed already. When the database reopens, any tablespace that was offline remains offline.

At this stage, the database is closed and inaccessible for normal operations. The control files remain open after a database is closed.

How a Database Is Closed During Abnormal Shutdown

If a SHUTDOWN ABORT or abnormal termination occurs, then the instance of an open database closes and shuts down the database instantaneously.

In an abnormal shutdown, Oracle Database does not write data in the buffers of the SGA to the data files and redo log files. The subsequent reopening of the database requires instance recovery, which Oracle Database performs automatically.

How a Database Is Unmounted

After the database is closed, Oracle Database unmounts the database to disassociate it from the instance.

After a database is unmounted, Oracle Database closes the control files of the database. At this point, the database instance remains in memory.

How an Instance Is Shut Down

The final step in database shutdown is shutting down the instance. When the database instance shuts down, the SGA ceases to occupy memory, and the background processes terminate.

In unusual circumstances, shutdown of a database instance may not occur cleanly. Memory structures may not be removed from memory or one of the background processes may not be terminated. When remnants of a previous instance exist, a subsequent instance startup may fail. In such situations, you can force the new instance to start by removing the remnants of the previous instance and then starting a new instance, or by issuing a SHUTDOWN ABORT statement.

In some cases, process cleanup itself can encounter errors, which can result in the termination of process monitor (PMON) or the instance. The dynamic initialization parameter INSTANCE_ABORT_DELAY_TIME specifies how many seconds to delay an internally generated instance failure. This delay gives you a chance to respond. The database writes a message to the alert log when the delayed termination is initiated. In some circumstances, by allowing certain database resources to be quarantined, the instance can avoid termination.

> **See Also:**
>
> - *Oracle Database Administrator's Guide* for more detailed information about database shutdown
> - *Oracle Database Reference* to learn more about the `INSTANCE_ABORT_DELAY_TIME` initialization parameter

Overview of Checkpoints

A **checkpoint** is a crucial mechanism in consistent database shutdowns, instance recovery, and Oracle Database operation generally.

The term has the following related meanings:

- A data structure that indicates the **checkpoint position**, which is the SCN in the redo stream where instance recovery must begin

  The checkpoint position is determined by the oldest dirty buffer in the database buffer cache. The checkpoint position acts as a pointer to the redo stream and is stored in the control file and in each data file header.
- The writing of modified database buffers in the database buffer cache to disk

- Purpose of Checkpoints
  Oracle Database uses checkpoints to achieve multiple goals.
- When Oracle Database Initiates Checkpoints
  The **checkpoint process (CKPT)** is responsible for writing checkpoints to the data file headers and control file.

> **See Also:**
>
> "System Change Numbers (SCNs)"

Purpose of Checkpoints

Oracle Database uses checkpoints to achieve multiple goals.

Goals include the following:

- Reduce the time required for recovery in case of an instance or media failure
- Ensure that the database regularly writes dirty buffers in the buffer cache to disk
- Ensure that the database writes all committed data to disk during a consistent shutdown

When Oracle Database Initiates Checkpoints

The **checkpoint process (CKPT)** is responsible for writing checkpoints to the data file headers and control file.

Checkpoints occur in a variety of situations. For example, Oracle Database uses the following types of checkpoints:

- Thread checkpoints

  The database writes to disk all buffers modified by redo in a specific thread before a certain target. The set of thread checkpoints on all instances in a database is a **database checkpoint**. Thread checkpoints occur in the following situations:

  - Consistent database shutdown
  - `ALTER SYSTEM CHECKPOINT` statement
  - Online redo log switch
  - `ALTER DATABASE BEGIN BACKUP` statement

- Tablespace and data file checkpoints

  The database writes to disk all buffers modified by redo before a specific target. A tablespace checkpoint is a set of data file checkpoints, one for each data file in the tablespace. These checkpoints occur in a variety of situations, including making a tablespace read-only or taking it offline normal, shrinking a data file, or executing `ALTER TABLESPACE BEGIN BACKUP`.

- Incremental checkpoints

  An incremental checkpoint is a type of thread checkpoint partly intended to avoid writing large numbers of blocks at online redo log switches. DBW checks at least every three seconds to determine whether it has work to do. When DBW writes dirty buffers, it advances the checkpoint position, causing CKPT to write the checkpoint position to the control file, but not to the data file headers.

Other types of checkpoints include instance and media recovery checkpoints and checkpoints when schema objects are dropped or truncated.

> **See Also:**
>
> - "Checkpoint Process (CKPT)"
> - *Oracle Real Application Clusters Administration and Deployment Guide* for information about global checkpoints in Oracle RAC

Overview of Instance Recovery

Instance recovery is the process of applying records in the online redo log to data files to reconstruct changes made after the most recent checkpoint.

Instance recovery occurs automatically when an administrator attempts to open a database that was previously shut down inconsistently.

- Purpose of Instance Recovery
  Instance recovery ensures that the database is in a consistent state after an instance failure. The files of a database can be left in an inconsistent state because of how Oracle Database manages database changes.
- When Oracle Database Performs Instance Recovery
  Whether instance recovery is required depends on the state of the redo threads.

- Importance of Checkpoints for Instance Recovery
  Instance recovery uses checkpoints to determine which changes must be applied to the data files. The checkpoint position guarantees that every committed change with an SCN *lower than* the checkpoint SCN is saved to the data files.
- Instance Recovery Phases
  The first phase of instance recovery is called **cache recovery** or **rolling forward**, and reapplies all changes recorded in the online redo log to the data files.

Purpose of Instance Recovery

Instance recovery ensures that the database is in a consistent state after an instance failure. The files of a database can be left in an inconsistent state because of how Oracle Database manages database changes.

A redo thread is a record of all of the changes generated by an instance. A single-instance database has one thread of redo, whereas an Oracle RAC database has multiple redo threads, one for each database instance.

When a transaction is committed, log writer process (LGWR) writes both the remaining redo entries in memory and the transaction SCN to the online redo log. However, the database writer (DBW) process writes modified data blocks to the data files whenever it is most efficient. For this reason, uncommitted changes may temporarily exist in the data files while committed changes do not yet exist in the data files.

If an instance of an open database fails, either because of a `SHUTDOWN ABORT` statement or abnormal termination, then the following situations can result:

- Data blocks committed by a transaction are not written to the data files and appear only in the online redo log. These changes must be reapplied to the data files.
- The data files contains changes that had not been committed when the instance failed. These changes must be rolled back to ensure transactional consistency.

Instance recovery uses only online redo log files and current online data files to synchronize the data files and ensure that they are consistent.

> **See Also:**
>
> - "Database Writer Process (DBW)" and "Database Buffer Cache"
> - "Introduction to Data Concurrency and Consistency"

When Oracle Database Performs Instance Recovery

Whether instance recovery is required depends on the state of the redo threads.

A redo thread is marked open in the control file when a database instance opens in read/write mode, and is marked closed when the instance is shut down consistently. If redo threads are marked open in the control file, but no live instances hold the thread enqueues corresponding to these threads, then the database requires instance recovery.

Oracle Database performs instance recovery automatically in the following situations:

- The database opens for the first time after the failure of a single-instance database or all instances of an Oracle RAC database. This form of instance recovery is also called **crash**

recovery. Oracle Database recovers the online redo threads of the terminated instances together.

- Some but not all instances of an Oracle RAC database fail. Instance recovery is performed automatically by a remaining instance in the configuration.

The SMON background process performs instance recovery, applying online redo automatically. No user intervention is required.

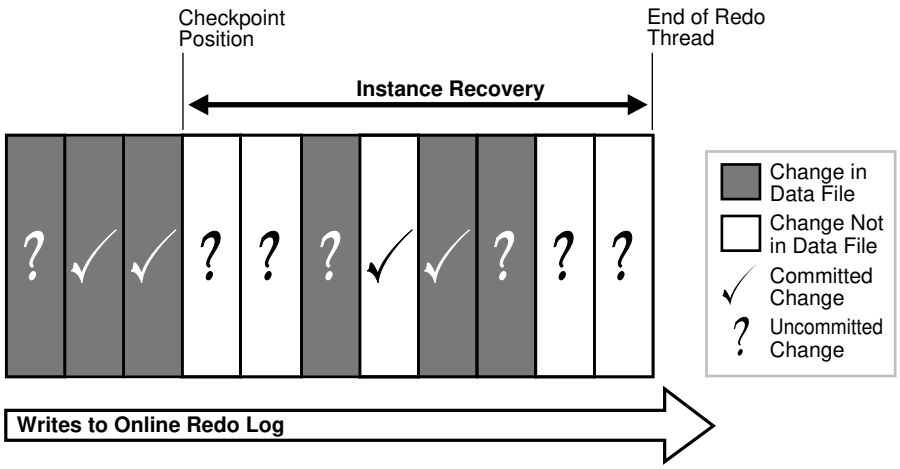
> **See Also:**
>
> - "System Monitor Process (SMON)"
> - *Oracle Real Application Clusters Administration and Deployment Guide* to learn about instance recovery in an Oracle RAC database

Importance of Checkpoints for Instance Recovery

Instance recovery uses checkpoints to determine which changes must be applied to the data files. The checkpoint position guarantees that every committed change with an SCN *lower than* the checkpoint SCN is saved to the data files.

The following figure depicts the redo thread in the online redo log.

Figure 16-5 Checkpoint Position in Online Redo Log



During instance recovery, the database must apply the changes that occur between the checkpoint position and the end of the redo thread. As shown in Figure 16-5, some changes may already have been written to the data files. However, only changes with SCNs lower than the checkpoint position are *guaranteed* to be on disk.

> **See Also:**
>
> *Oracle Database Performance Tuning Guide* to learn how to limit instance recovery time

Instance Recovery Phases

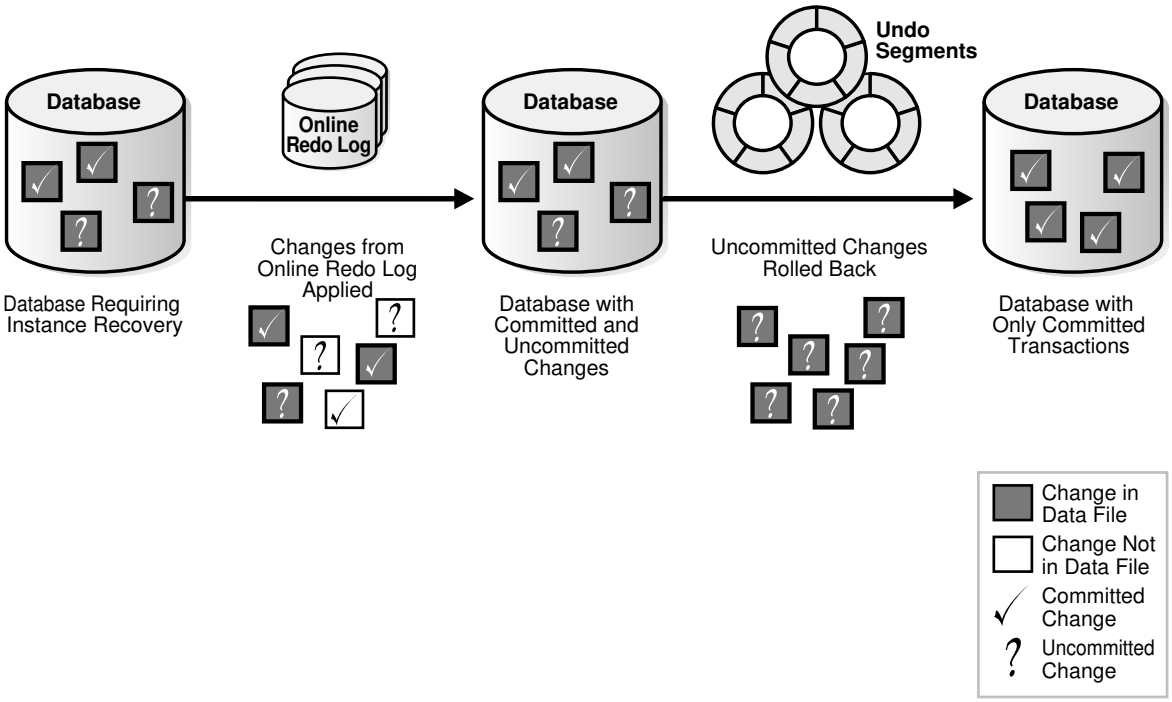
The first phase of instance recovery is called **cache recovery** or **rolling forward**, and reapplies all changes recorded in the online redo log to the data files.

Because the online redo log contains undo data, rolling forward also regenerates the corresponding undo segments. Rolling forward proceeds through as many online redo log files as necessary to bring the database forward in time. After rolling forward, the data blocks contain all committed changes recorded in the online redo log files. These files could also contain uncommitted changes that were either saved to the data files before the failure, or were recorded in the online redo log and introduced during cache recovery.

After the roll forward, any changes that were not committed must be undone. Oracle Database uses the checkpoint position, which guarantees that every committed change with an SCN lower than the checkpoint SCN is saved on disk. Oracle Database applies undo blocks to roll back uncommitted changes in data blocks that were written before the failure or introduced during cache recovery. This phase is called **rolling back** or transaction recovery.

The following figure illustrates rolling forward and rolling back, the two steps necessary to recover from database instance failure.

Figure 16-6 Basic Instance Recovery Steps: Rolling Forward and Rolling Back



Oracle Database can roll back multiple transactions simultaneously as needed. All transactions that were active at the time of failure are marked as terminated. Instead of waiting for the

SMON process to roll back terminated transactions, new transactions can roll back individual blocks themselves to obtain the required data.

> **See Also:**
>
> - "Undo Segments" to learn more about undo data
> - *Oracle Database Performance Tuning Guide* for a discussion of instance recovery mechanics and tuning

Overview of Parameter Files

To start a database instance, Oracle Database must read either a **server parameter file**, which is recommended, or a **text initialization parameter file**, which is a legacy implementation. These files contain a list of configuration parameters.

To create a database manually, you must start an instance with a parameter file and then issue a `CREATE DATABASE` statement. Thus, the instance and parameter file can exist even when the database itself does not exist.

- Initialization Parameters
  Initialization parameters are configuration parameters that affect the basic operation of an instance. The instance reads initialization parameters from a file at startup.
- Server Parameter Files
  A **server parameter file** is a repository for initialization parameters.
- Text Initialization Parameter Files
  A **text initialization parameter file** is a text file that contains a list of initialization parameters.
- Modification of Initialization Parameter Values
  You can adjust initialization parameters to modify the behavior of a database. The classification of parameters as **static** or **dynamic** determines how they can be modified.

Initialization Parameters

Initialization parameters are configuration parameters that affect the basic operation of an instance. The instance reads initialization parameters from a file at startup.

Oracle Database provides many initialization parameters to optimize its operation in diverse environments. Only a few of these parameters must be explicitly set because the default values are usually adequate.

- Functional Groups of Initialization Parameters
  Initialization parameters fall into different functional groups.
- Basic and Advanced Initialization Parameters
  Initialization parameters are divided into two groups: basic and advanced.

Functional Groups of Initialization Parameters

Initialization parameters fall into different functional groups.

Most initialization parameters belong to one of the following groups:

- Parameters that name entities such as files or directories
- Parameters that set limits for a process, database resource, or the database itself
- Parameters that affect capacity, such as the size of the SGA (these parameters are called **variable parameters**)

Variable parameters are of particular interest to database administrators because they can use these parameters to improve database performance.

Basic and Advanced Initialization Parameters

Initialization parameters are divided into two groups: basic and advanced.

Typically, you must set and tune only the approximately 30 basic parameters to obtain reasonable performance. The basic parameters set characteristics such as the database name, locations of the control files, database block size, and undo tablespace.

In rare situations, modification to the advanced parameters may be required for optimal performance. The advanced parameters enable expert DBAs to adapt the behavior of the Oracle Database to meet unique requirements.

Oracle Database provides values in the starter initialization parameter file provided with your database software, or as created for you by the Database Configuration Assistant (DBCA). You can edit these Oracle-supplied initialization parameters and add others, depending on your configuration and how you plan to tune the database. For relevant initialization parameters not included in the parameter file, Oracle Database supplies defaults.

See Also:

- *Oracle Database Administrator's Guide* to learn how to specify initialization parameters
- *Oracle Database Reference* for an explanation of the types of initialization parameters
- *Oracle Database Reference* for a description of `V$PARAMETER` and *SQL*Plus User's Guide and Reference* for `SHOW PARAMETER` syntax

Server Parameter Files

A **server parameter file** is a repository for initialization parameters.

A server parameter file has the following key characteristics:

- Only Oracle Database reads and writes to the server parameter file.
- Only one server parameter file exists for a database. This file must reside on the database host.
- The server parameter file is binary and cannot be modified by a text editor.
- Initialization parameters stored in the server parameter file are persistent. Any changes made to the parameters while a database instance is running can persist across instance shutdown and startup.

A server parameter file eliminates the need to maintain multiple text initialization parameter files for client applications. A server parameter file is initially built from a text initialization

parameter file using the `CREATE SPFILE` statement. It can also be created directly by the Database Configuration Assistant.

> **See Also:**
>
> - *Oracle Database Administrator's Guide* to learn more about server parameter files
> - *Oracle Database SQL Language Reference* to learn about `CREATE SPFILE`

Text Initialization Parameter Files

A **text initialization parameter file** is a text file that contains a list of initialization parameters.

This type of parameter file, which is a legacy implementation of the parameter file, has the following key characteristics:

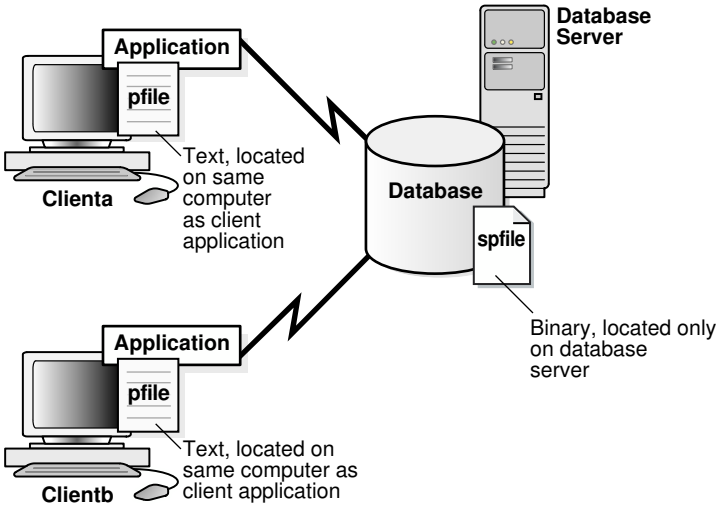
- When starting up or shutting down a database, the text initialization parameter file must reside on the same host as the client application that connects to the database.
- A text initialization parameter file is text-based, not binary.
- Oracle Database can read but not write to the text initialization parameter file. To change the parameter values you must manually alter the file with a text editor.
- Changes to initialization parameter values by `ALTER SYSTEM` are only in effect for the current instance. You must manually update the text initialization parameter file and restart the instance for the changes to be known.

The text initialization parameter file contains a series of `key=value` pairs, one per line. For example, a portion of an initialization parameter file could look as follows:

```
db_name=sample
control_files=/disk1/oradata/sample_cf.dbf
db_block_size=8192
open_cursors=52
undo_management=auto
shared_pool_size=280M
pga_aggregate_target=29M
.
.
.
```

To illustrate the manageability problems that text parameter files can create, assume that you use computers `clienta` and `clientb` and must be able to start the database with SQL*Plus on either computer. In this case, two separate text initialization parameter files must exist, one on each computer, as shown in Figure 16-7. A server parameter file solves the problem of the proliferation of parameter files.

Figure 16-7 Multiple Initialization Parameter Files



> **See Also:**
>
> - *Oracle Database Administrator's Guide* to learn more about text initialization parameter files
> - *Oracle Database SQL Language Reference* to learn about `CREATE PFILE`

Modification of Initialization Parameter Values

You can adjust initialization parameters to modify the behavior of a database. The classification of parameters as **static** or **dynamic** determines how they can be modified.

The following table summarizes the differences.

Table 16-5 Static and Dynamic Initialization Parameters

Characteristic	Static	Dynamic
Requires modification of the parameter file (text or server)	Yes	No
Requires database instance restart before setting takes affect	Yes	No
Described as "Modifiable" in *Oracle Database Reference* initialization parameter entry	No	Yes
Modifiable only for the database or instance	Yes	No

Static parameters include `DB_BLOCK_SIZE`, `DB_NAME`, and `COMPATIBLE`. Dynamic parameters are grouped into **session-level parameters**, which affect only the current user session, and **system-level parameters**, which affect the database and all sessions. For example, `MEMORY_TARGET` is a system-level parameter, while `NLS_DATE_FORMAT` is a session-level parameter.

The **scope** of a parameter change depends on when the change takes effect. When an instance has been started with a server parameter file, you can use the `ALTER SYSTEM SET` statement to change values for system-level parameters as follows:

- `SCOPE=MEMORY`

  Changes apply to the database instance only. The change will not persist if the database is shut down and restarted.

- `SCOPE=SPFILE`

  Changes apply to the server parameter file but do not affect the current instance. Thus, the changes do not take effect until the instance is restarted.

  > **Note:**
  >
  > You must specify `SPFILE` when changing the value of a parameter described as not modifiable in *Oracle Database Reference*.

- `SCOPE=BOTH`

  Oracle Database writes changes both to memory and to the server parameter file. This is the default scope when the database is using a server parameter file.

The database prints the new value and the old value of an initialization parameter to the alert log. As a preventative measure, the database validates changes of basic parameter to prevent invalid values from being written to the server parameter file.

> **See Also:**
>
> - *Oracle Database Globalization Support Guide* to learn how to choose a locale with the `NLS_LANG` environment variable
> - *Oracle Database Administrator's Guide* to learn how to change initialization parameter settings
> - *Oracle Database Reference* for descriptions of all initialization parameters
> - *Oracle Database SQL Language Reference* for `ALTER SYSTEM` syntax and semantics

Overview of Diagnostic Files

Oracle Database includes a **fault diagnosability infrastructure** for preventing, detecting, diagnosing, and resolving database problems. Problems include critical errors such as code bugs, metadata corruption, and customer data corruption.

The goals of the advanced fault diagnosability infrastructure are the following:

- Detecting problems proactively
- Limiting damage and interruptions after a problem is detected
- Reducing problem diagnostic and resolution time

- Improving manageability by enabling trace files to be partitioned, allowing user to define the size per piece and maximum number of pieces to retain, and disabling tracing after a user-specified disk space limit is reached
- Simplifying customer interaction with Oracle Support

- Automatic Diagnostic Repository
  Automatic Diagnostic Repository (ADR) is a file-based repository that stores database diagnostic data such as trace files, the alert log, DDL log, and Health Monitor reports.
- Alert Log
  Every database has an **alert log**, which is a file containing a chronological log of database messages and errors.
- Attention Log
  Every database has an **attention log**, containing critical and highly visible database events.
- DDL Log
  The **DDL log** has the same format and basic behavior as the alert log but contains only DDL statements and details. The database writes DDL information to its own file to reduce the clutter in the alert log.
- Trace Files
  A **trace file** is a file that contains diagnostic data used to investigate problems. Also, trace files can provide guidance for tuning applications or an instance.
- Diagnostic Dumps
  A **diagnostic dump file** is a special type of trace file that contains detailed point-in-time information about a state or structure.

> **See Also:**
>
> *Oracle Database Administrator's Guide* to learn how to manage diagnostic files

Automatic Diagnostic Repository

Automatic Diagnostic Repository (ADR) is a file-based repository that stores database diagnostic data such as trace files, the alert log, DDL log, and Health Monitor reports.

Key characteristics of ADR include:

- Unified directory structure
- Consistent diagnostic data formats
- Unified tool set

The preceding characteristics enable customers and Oracle Support to correlate and analyze diagnostic data across multiple Oracle instances, components, and products.

ADR is located *outside* the database, which enables Oracle Database to access and manage ADR when the physical database is unavailable. A database instance can create ADR before a database has been created.

- Problems and Incidents
  ADR proactively tracks **problems**, which are critical errors in the database.

- ADR Structure
  The **ADR base** is the ADR root directory.

Problems and Incidents

ADR proactively tracks **problems**, which are critical errors in the database.

Critical errors manifest as internal errors, such as `ORA-600`, or other severe errors. Each problem has a **problem key**, which is a text string that describes the problem.

When a problem occurs multiple times, ADR creates a time-stamped **incident** for each occurrence. An incident is uniquely identified by a numeric **incident ID**. When an incident occurs, ADR sends an **incident alert** to Enterprise Manager. Diagnosis and resolution of a critical error usually starts with an incident alert.

Because a problem could generate many incidents in a short time, ADR applies flood control to incident generation after certain thresholds are reached. A **flood-controlled incident** generates an alert log entry, but does not generate incident dumps. In this way, ADR informs you that a critical error is ongoing without overloading the system with diagnostic data.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for detailed information about the fault diagnosability infrastructure

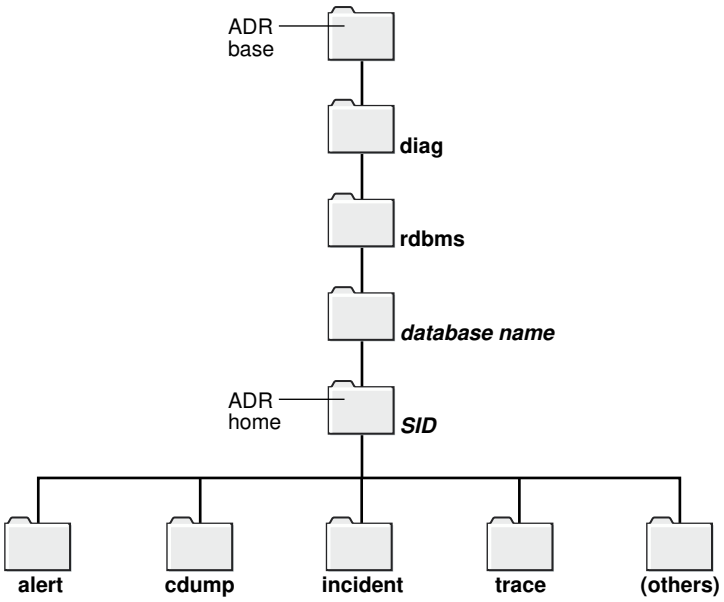
ADR Structure

The **ADR base** is the ADR root directory.

The ADR base can contain multiple ADR homes, where each ADR home is the root directory for all diagnostic data—traces, dumps, the alert log, and so on—for an instance of an Oracle product or component. For example, in an Oracle RAC environment with shared storage and Oracle ASM, each database instance and each Oracle ASM instance has its own ADR home.

Figure 16-8 illustrates the ADR directory hierarchy for a database instance. Other ADR homes for other Oracle products or components, such as Oracle ASM or Oracle Net Services, can exist within this hierarchy, under the same ADR base.

Figure 16-8 ADR Directory Structure for an Oracle Database Instance



As the following Linux example shows, when you start an instance with a unique SID and database name *before* creating a database, Oracle Database creates ADR by default as a directory structure in the host file system. The SID and database name form part of the path name for files in the ADR Home.

Example 16-1 Creation of ADR

```
% setenv ORACLE_SID osi
% echo "DB_NAME=dbn" > init.ora
% sqlplus / as sysdba
.
.
.
Connected to an idle instance.

SQL> STARTUP NOMOUNT PFILE="./init.ora"
ORACLE instance started.

Total System Global Area  146472960 bytes
Fixed Size                  1317424 bytes
Variable Size              92276176 bytes
Database Buffers           50331648 bytes
Redo Buffers                2547712 bytes

SQL> COL NAME FORMAT a21
SQL> COL VALUE FORMAT a60
SQL> SELECT NAME, VALUE FROM V$DIAG_INFO;

NAME                  VALUE
--------------------- ------------------------------------------------------------
Diag Enabled          TRUE
ADR Base              /d1/3910926111/oracle/log
ADR Home              /d1/3910926111/oracle/log/diag/rdbms/dbn/osi
```

```
Diag Trace              /d1/3910926111/oracle/log/diag/rdbms/dbn/osi/trace
Diag Alert              /d1/3910926111/oracle/log/diag/rdbms/dbn/osi/alert
Diag Incident           /d1/3910926111/oracle/log/diag/rdbms/dbn/osi/incident
Diag Cdump              /d1/3910926111/oracle/log/diag/rdbms/dbn/osi/cdump
Health Monitor          /d1/3910926111/oracle/log/diag/rdbms/dbn/osi/hm
Default Trace File      /d1/3910926111/oracle/log/diag/rdbms/dbn/osi/trace/
osi_ora_29229.trc
Active Problem Count    0
Active Incident Count   0
ORACLE_HOME             /d1/3910926111/oracle
Attention Log           /d1/3910926111/oracle/log/diag/rdbms/dbn/osi/trace/
attention_osi.log
13 rows selected.
```

Alert Log

Every database has an **alert log**, which is a file containing a chronological log of database messages and errors.

The alert log contents include the following:

- All internal errors (`ORA-600`), block corruption errors (`ORA-1578`), and deadlock errors (`ORA-60`)
- Administrative operations such as the SQL*Plus commands `STARTUP`, `SHUTDOWN`, `ARCHIVE LOG`, and `RECOVER`
- Several messages and errors relating to the functions of shared server and dispatcher processes
- Errors during the automatic refresh of a materialized view

If an administrative operation is successful, then Oracle Database writes a message to the alert log as "completed" along with a time stamp.

Oracle Database creates an alert log in the `alert` subdirectory shown in Figure 16-8 when you first start a database instance, even if no database has been created yet. This file is in the XML format. The `trace` subdirectory contains a text-only alert log.

Query `V$DIAG_INFO` to locate the alert log.

Attention Log

Every database has an **attention log**, containing critical and highly visible database events.

The attention log includes only critical and highly visible database events, enabling you to identify an alert and take the appropriate action. Each attention log message has various attributes including target audience, cause, and an action to take. The attention log provides information in the following format:

- Attention message ID
- Attention message type: error, notification, warning
- Attention message urgency: DEFERABLE, IMMEDIATE, INFO, SOON
- Attention message scope: CDB, CDB cluster, CDB instance, PDB, session, PDB instance, process, session
- Attention message text

- Attention message target user: CDB Admin, Clusterware Admin, PDB Admin
- Cause
- Action

Oracle Database creates an attention log named `attention_ORACLE_SID.log` in the `trace` subdirectory shown in Figure 16-8. You can also query `V$DIAG_INFO` to locate the attention log.

DDL Log

The **DDL log** has the same format and basic behavior as the alert log but contains only DDL statements and details. The database writes DDL information to its own file to reduce the clutter in the alert log.

DDL log records are DDL text, optionally augmented with supplemental information. One log record exists for each DDL statement. The DDL log is stored in the `log/ddl` subdirectory of the ADR home.

Trace Files

A **trace file** is a file that contains diagnostic data used to investigate problems. Also, trace files can provide guidance for tuning applications or an instance.

- Types of Trace Files
  Each server and background process can periodically write to an associated trace file. The files contain information on the process environment, status, activities, and errors.
- Locations of Trace Files
  ADR stores trace files in the `trace` subdirectory. Trace file names are platform-dependent and use the extension `.trc`.
- Segmentation of Trace Files
  When the trace file size is limited, the database may automatically split it into a maximum of five segments. Segments are separate files that have the same name as the active trace file, but with a segment number appended, as in `ora_1234_2.trc`.

> **See Also:**
>
> *Oracle Database Get Started with Performance Tuning*

Types of Trace Files

Each server and background process can periodically write to an associated trace file. The files contain information on the process environment, status, activities, and errors.

The SQL trace facility also creates trace files, which provide performance information on individual SQL statements. You can enable tracing for a client identifier, service, module, action, session, instance, or database in various ways. For example, you can execute the appropriate procedures in the `DBMS_MONITOR` package or set events.

> **See Also:**
>
> - "Session Control Statements"
> - *Oracle Database Administrator's Guide* to learn about trace files, dumps, and core files
> - *Oracle Database SQL Tuning Guide* to learn about application tracing

Locations of Trace Files

ADR stores trace files in the `trace` subdirectory. Trace file names are platform-dependent and use the extension `.trc`.

Typically, database background process trace file names contain the Oracle SID, the background process name, and the operating system process number. An example of a trace file for the `RECO` process is `mytest_reco_10355.trc`.

Server process trace file names contain the Oracle SID, the string `ora`, and the operating system process number. An example of a server process trace file name is `mytest_ora_10304.trc`.

Sometimes trace files have corresponding trace metadata files, which end with the extension `.trm`. These files contain structural information called **trace maps** that the database uses for searching and navigation.

> **See Also:**
>
> - "Figure 16-8"
> - *Oracle Database Administrator's Guide* to learn how to find trace files

Segmentation of Trace Files

When the trace file size is limited, the database may automatically split it into a maximum of five segments. Segments are separate files that have the same name as the active trace file, but with a segment number appended, as in `ora_1234_2.trc`.

Each segment is typically 20% of the limit set by `MAX_DUMP_FILE_SIZE`. When the combined size of all segments exceeds the limit, the database deletes the oldest segment (although never the first segment, which may contain relevant information about the initial state of the process), and then creates a new, empty segment.

> **See Also:**
>
> *Oracle Database Administrator's Guide* to learn how to control the size of trace files

Diagnostic Dumps

A **diagnostic dump file** is a special type of trace file that contains detailed point-in-time information about a state or structure.

A trace tends to be continuous output of diagnostic data. In contrast, a dump is typically a one-time output of diagnostic data in response to an event.

- Trace Dumps and Incidents
  Most dumps occur because of incidents.

Trace Dumps and Incidents

Most dumps occur because of incidents.

When an incident occurs, the database writes one or more dumps to the incident directory created for the incident. Incident dumps also contain the incident number in the file name.

During incident creation, an application may take a heap or system state dump as part of an action. In such cases, the database appends the dump name to the incident file name instead of the default trace file name. For example, because of an incident in a process, the database creates file `prod_ora_90348.trc`. A dump in the incident generates the file `prod_ora_90348_`*`incident_id`*`.trc`, where *`incident_id`* is the numeric ID of the incident. A heap dump action created as part of the incident generates the heap dump file `prod_ora_90348_`*`incident_id_dump_id`*`.trc`, where *`dump_id`* is the numeric ID of the trace dump.