# 14 Physical Storage Structures

The physical database structures of an Oracle database are viewable at the operating system level.

- Introduction to Physical Storage Structures
  One characteristic of an RDBMS is the independence of logical data structures such as tables, views, and indexes from physical storage structures such as files.
- Overview of Data Files
  At the operating system level, Oracle Database stores data in structures called **data files**. Every Oracle database, whether a CDB or PDB, must have at least one data file.
- Overview of Control Files
  The database **control file** is a small binary file associated with only one CDB.
- Overview of the Online Redo Log
  The most crucial structure for recovery is the **online redo log**, which consists of two or more preallocated files that store changes to the database as they occur. The online redo log records changes to the data files.

## Introduction to Physical Storage Structures

One characteristic of an RDBMS is the independence of logical data structures such as tables, views, and indexes from physical storage structures such as files.

Because physical and logical structures are separate, you can manage physical storage of data without affecting access to logical structures. For example, renaming a database file does not rename the tables stored in it.

An Oracle database is a set of files that store Oracle data in persistent storage. Starting in Oracle Database 21c, a multitenant container database (CDB) is the only supported architecture. A CDB contains one or more PDBs.

This section discusses the database files generated when you issue a `CREATE DATABASE` statement to create a CDB:

- Data files and temp files

  A data file is a physical file in persistent storage that was created by Oracle Database and contains data structures such as tables and indexes. A temp file is a data file that belongs to a temporary tablespace. The database writes data to these files in an Oracle proprietary format that cannot be read by other programs.

  When you create a PDB within a PDB, the PDB has its own set of data files within the overall set of data files that makes up the CDB. For example, you could create a CDB named `mycdb`, and then create two PDBs within it: `hrpdb` and `salespdb`. In this case, `mycdb` would have its own set of data files and temp files, as would `hrpdb` and `salespdb`.

- Control files

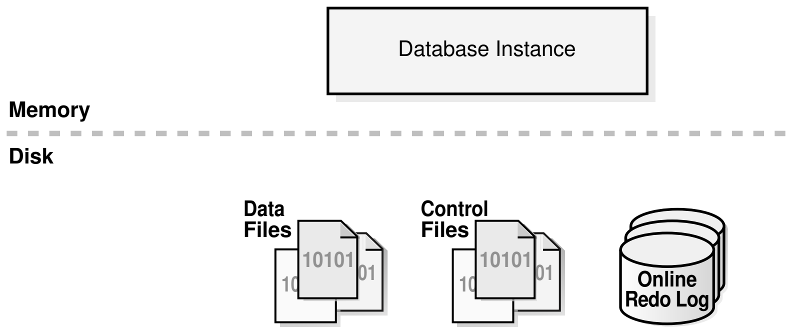
  A control file is a root file that tracks the physical components of the CDB. PDBs do not have their own separate control files.

- Online redo log files

The online redo log is a set of files containing records of changes made to data within the CDB. PDBs do not have their own separate online redo log files.

A database instance is a set of memory structures that manage database files in a CDB. The following graphic shows the relationship between the instance and the files that it manages.

**Figure 14-1 Database Instance and Database Files**



- Mechanisms for Storing Database Files
  Several mechanisms are available for allocating and managing the storage of these files.
- Oracle Managed Files and User-Managed Files
  **Oracle Managed Files** is a file naming strategy that enables you to specify operations in terms of database objects rather than file names. For example, you can create a tablespace without specifying the names of its data files.
- Oracle Automatic Storage Management (Oracle ASM)
  **Oracle ASM** is a high-performance, ease-of-management storage solution for Oracle Database files. Oracle ASM is a volume manager and provides a file system designed exclusively for use by the database.
- Oracle Persistent Memory Filestore (PMEM Filestore)
  PMEM is byte-addressable persistent memory. Oracle Persistent Memory Filestore (PMEM Filestore) can store data files, online redo log files, and control files.

**See Also:**

- *Oracle Database Administrator's Guide* to learn how to create a database
- *Oracle Database SQL Language Reference* for `CREATE DATABASE` semantics and syntax

## Mechanisms for Storing Database Files

Several mechanisms are available for allocating and managing the storage of these files.

Mechanisms include:

- Operating system file system

  Most Oracle databases store files in a file system, which is a structure for storing and retrieving data.

A file system enables storage space to be used for application data in a standard and consistent way. Each file has a name and appears as a contiguous address space to applications such as Oracle Database. The database can create, read, write, resize, and delete files.

A file system is commonly built on top of a logical volume constructed by a software package called a logical volume manager (LVM). The LVM enables pieces of multiple physical disks to combine into what appears as a single contiguous address space.

- Cluster file system

  A cluster file system is a distributed file system that is a cluster of servers that collaborate to provide consistency and high performance to their clients. In an Oracle RAC environment, a cluster file system makes shared storage appear as a file system shared by many computers in a clustered environment. With a cluster file system, the failure of a computer in the cluster does not make the file system unavailable. In an operating system file system, however, if a computer sharing files through NFS or other means fails, then the file system is unavailable.

- Oracle Persistent Memory Filestore (PMEM Filestore)

  PMEM Filestore is designed to store database files in Persistent Memory (PMEM), which is byte-addressable persistent memory. You create a filestore using the `CREATE PMEM FILESTORE` command. At the Linux operating system level, the filestore stores data in a large ext4 file system mounted in `fsdax` mode. PMEM Filestore appears to Oracle Database as a native file system on Linux.

  > **Note:**
  >
  > Unlike a POSIX-compliant file system, PMEM Filestore cannot be shared by multiple Oracle databases.

- Oracle Automatic Storage Management (Oracle ASM)

  Oracle ASM includes a file system designed exclusively for use by Oracle Database.

A database can use a combination of the preceding storage mechanisms. For example, a database could store the server parameter file in a traditional file system, store database files in PMEM Filestore, and then archive the redo log files to a cluster file system.

> **See Also:**
>
> - "Oracle Automatic Storage Management (Oracle ASM)"
> - *Oracle Database Administrator’s Guide* to view database storage structures by querying database views

## Oracle Managed Files and User-Managed Files

**Oracle Managed Files** is a file naming strategy that enables you to specify operations in terms of database objects rather than file names. For example, you can create a tablespace without specifying the names of its data files.

Oracle Managed Files eliminates the need for administrators to directly manage the operating system files in a database. Oracle ASM requires Oracle Managed Files.

**Note:**

This feature does not affect the creation or naming of administrative files such as trace files, audit files, and alert logs.

With user-managed files, you directly manage the operating system files in the database. You make the decisions regarding file structure and naming. For example, when you create a tablespace you set the name and path of the tablespace data files.

Through initialization parameters, you specify the file system directory for a specific type of file. The Oracle Managed Files feature ensures that the database creates a unique file and deletes it when no longer needed. The database internally uses standard file system interfaces to create and delete files for data files and temp files, control files, and recovery-related files stored in the fast recovery area.

Oracle Managed Files does not eliminate existing functionality. You can create new files while manually administering old files. Thus, a database can have a mixture of Oracle Managed Files and user-managed files.

**See Also:**

- "Overview of Diagnostic Files"
- *Oracle Database Administrator’s Guide* to learn how to use Oracle Managed Files

## Oracle Automatic Storage Management (Oracle ASM)

**Oracle ASM** is a high-performance, ease-of-management storage solution for Oracle Database files. Oracle ASM is a volume manager and provides a file system designed exclusively for use by the database.

Oracle ASM provides several advantages over conventional file systems and storage managers, including the following:

- Simplifies storage-related tasks such as creating and laying out databases and managing disk space
- Distributes data across physical disks to eliminate hot spots and to provide uniform performance across the disks
- Rebalances data automatically after storage configuration changes

To use Oracle ASM, you allocate partitioned disks for Oracle Database with preferences for striping and mirroring. Oracle ASM manages the disk space, distributing the I/O load across all available resources to optimize performance while removing the need for manual I/O tuning. For example, you can increase the size of the disk for the database or move parts of the database to new devices without having to shut down the database.

- Oracle ASM Storage Components
  Oracle Database can store a data file as an Oracle ASM file in an Oracle ASM disk group. Within a disk group, Oracle ASM exposes a file system interface for database files.
- Oracle ASM Instances
  An **Oracle ASM instance** is a special Oracle instance that manages Oracle ASM disks.

## Oracle ASM Storage Components

Oracle Database can store a data file as an Oracle ASM file in an Oracle ASM disk group. Within a disk group, Oracle ASM exposes a file system interface for database files.

The following figure shows the relationships between storage components in a database that uses Oracle ASM. The diagram depicts the relationship between an Oracle ASM file and a data file, although Oracle ASM can store other types of files. The crow's foot notation represents a one-to-many relationship.

**Figure 14-2 Oracle ASM Components**

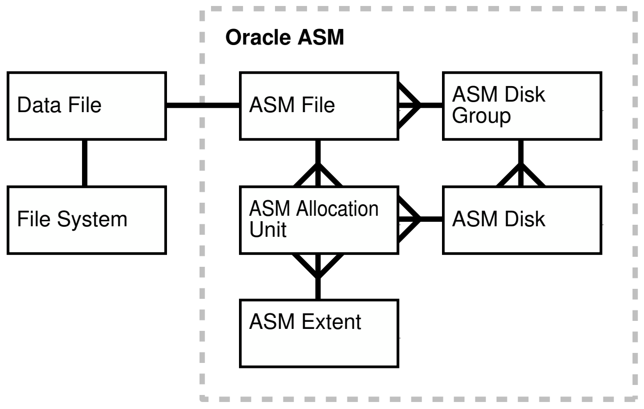


Figure 14-2 illustrates the following Oracle ASM concepts:

- Oracle ASM Disks

  An Oracle ASM disk is a storage device that is provisioned to an Oracle ASM disk group. An Oracle ASM disk can be a physical disk or partition, a Logical Unit Number (LUN) from a storage array, a logical volume, or a network-attached file.

  Oracle ASM disks can be added or dropped from a disk group while the database is running. When you add a disk to a disk group, you either assign a disk name or the disk is given an Oracle ASM disk name automatically.
- Oracle ASM Disk Groups

  An Oracle ASM disk group is a collection of Oracle ASM disks managed as a logical unit. The data structures in a disk group are self-contained and consume some disk space in a disk group.

  Within a disk group, Oracle ASM exposes a file system interface for Oracle database files. The content of files that are stored in a disk group are evenly distributed, or striped, to eliminate hot spots and to provide uniform performance across the disks.
- Oracle ASM Files

  An Oracle ASM file is a file stored in an Oracle ASM disk group. Oracle Database communicates with Oracle ASM in terms of files. The database can store data files, control files, online redo log files, and other types of files as Oracle ASM files. When requested by the database, Oracle ASM creates an Oracle ASM file and assigns it a name beginning with a plus sign (+) followed by a disk group name, as in `+DISK1`.

> **Note:**
>
> Oracle ASM files can coexist with other storage management options, such as third-party file systems. This capability simplifies the integration of Oracle ASM into pre-existing environments.

- Oracle ASM Extents

  An Oracle ASM extent is a section of an Oracle ASM file. An Oracle ASM file consists of one or more file extents. Each Oracle ASM extent consists of one or more allocation units on a specific disk.

  > **Note:**
  >
  > An Oracle ASM extent is different from the extent used to store data in a segment.

- Oracle ASM Allocation Units

  An Oracle ASM allocation unit is the fundamental unit of allocation within a disk group. An allocation unit is the smallest contiguous disk space that Oracle ASM allocates. One or more allocation units form an Oracle ASM extent.

> **See Also:**
>
> *Oracle Automatic Storage Management Administrator's Guide* to learn more about Oracle ASM

## Oracle ASM Instances

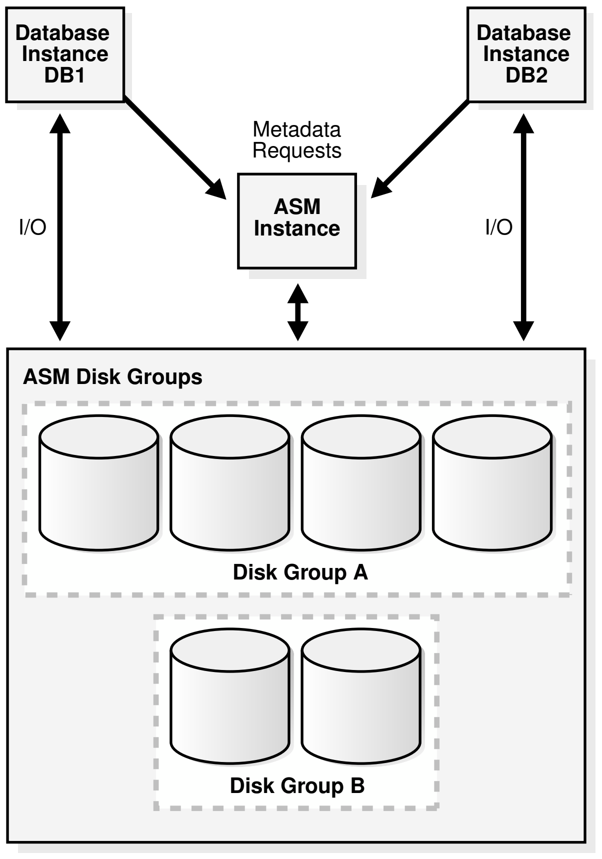
An **Oracle ASM instance** is a special Oracle instance that manages Oracle ASM disks.

Both the Oracle ASM and the database instances require shared access to the disks in an Oracle ASM disk group. Oracle ASM instances manage the metadata of the disk group and provide file layout information to the database instances. Database instances direct I/O to Oracle ASM disks without going through an Oracle ASM instance.

An Oracle ASM instance is built on the same technology as a database instance. For example, an Oracle ASM instance has a system global area (SGA) and background processes that are similar to those of a database instance. However, an Oracle ASM instance cannot mount a database and performs fewer tasks than a database instance.

Figure 14-3 shows a single-node configuration with one Oracle ASM instance and two database instances, each associated with a different single-instance database. The Oracle ASM instance manages the metadata and provides space allocation for the Oracle ASM files storing the data for the two databases. One Oracle ASM disk group has four Oracle ASM disks and the other has two disks. Both database instances can access the disk groups.

**Figure 14-3 Oracle ASM Instance and Database Instances**



**See Also:**

*Oracle Automatic Storage Management Administrator's Guide* to learn more about Oracle ASM

## Oracle Persistent Memory Filestore (PMEM Filestore)

PMEM is byte-addressable persistent memory. Oracle Persistent Memory Filestore (PMEM Filestore) can store data files, online redo log files, and control files.

Queries can read bytes *directly* from PMEM without first migrating blocks into the database buffer cache. This high-performance mechanism avoids data redundancy and unnecessary I/O.

**Note:**

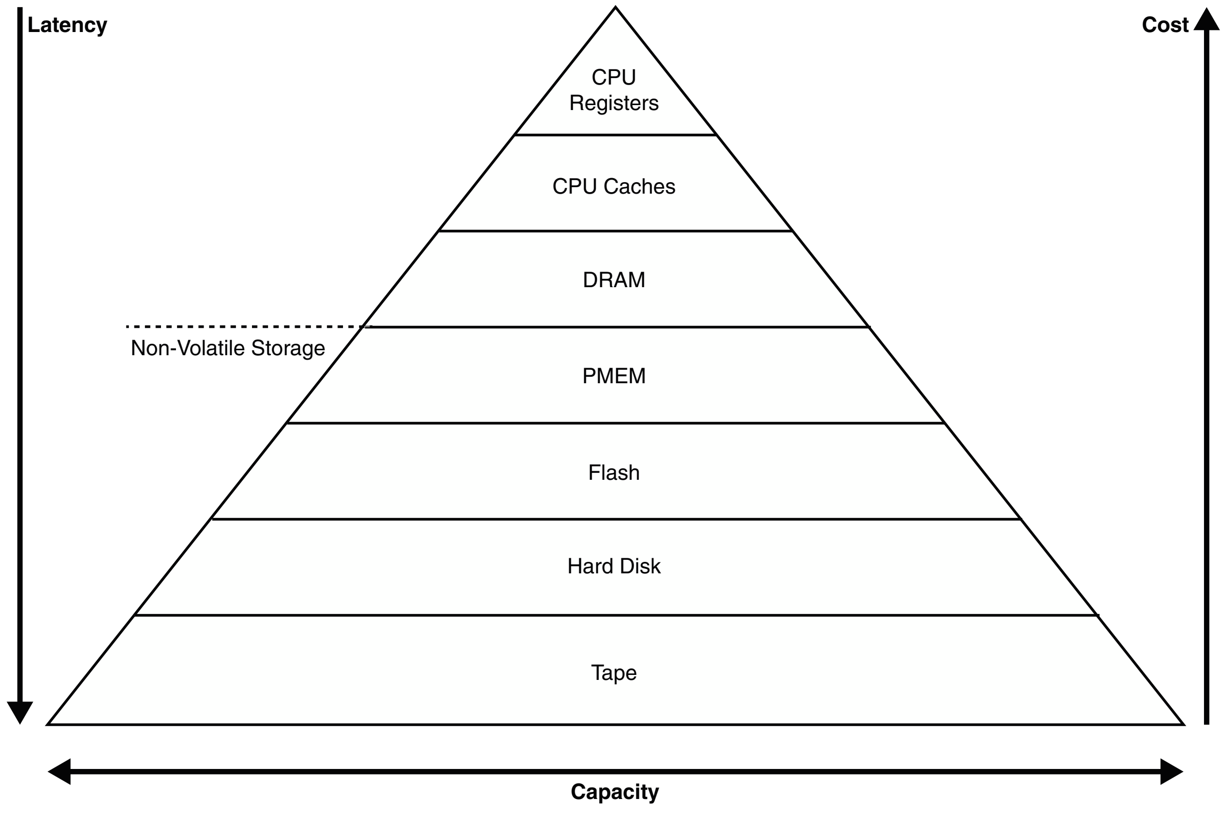
A PMEM filestore cannot store trace files, audit files, or server parameter files. The supported file types for PMEM are similar to the types for Oracle ASM.

- Directly Mapped Buffer Cache
  In the traditional storage pyramid, PMEM sits between DRAM, which is non-persistent, and Flash, which is persistent (non-volatile).
- File System Interface for PMEM Filestore
  PMEM Filestore implements a protocol called Filesystem in Userspace (FUSE).
- User Interface for PMEM Filestore
  To create a PMEM filestore, use the `CREATE PMEM FILESTORE` command.

## Directly Mapped Buffer Cache

In the traditional storage pyramid, PMEM sits between DRAM, which is non-persistent, and Flash, which is persistent (non-volatile).

**Figure 14-4 Storage Pyramid**



The database buffer cache resides in DRAM. Oracle processes have direct byte-addressable access to blocks in a PMEM filestore, which are mapped to the database buffer cache. The mapping mechanism is automatically invoked when you place a data file in a PMEM filestore.

Oracle Database uses an internal algorithm to decide whether to migrate data blocks from the PMEM filestore into DRAM. Typical reasons for migration include DML modifications, read consistency, and faster access for hot data blocks.

**See Also:**

"DRAM and PMEM Buffers"

## File System Interface for PMEM Filestore

PMEM Filestore implements a protocol called Filesystem in Userspace (FUSE).

FUSE enables non-privileged operating system users, such as the Oracle Database software owner, to create and manage file systems and the directories and files within them. For example, you can use the commands `cp`, `rm`, and `ls`.

A typical file system uses raw storage as its backing store, whereas PMEM Filestore gets storage from a native operating system file in a PMEM DAX file system. This file is called the **backing file** and is visible as a file in the operating system. PMEM Filestore subdivides the storage within the backing file and presents it as a local file system.

**See Also:**

- *Oracle Database Administrator's Guide* to learn more about PMEM Filestore

## User Interface for PMEM Filestore

To create a PMEM filestore, use the `CREATE PMEM FILESTORE` command.

The PMEM filestore is automatically mounted after it is created. The PMEM filestore appears under the specified mount point as if it were a native file system.

The `PMEM_FILESTORE` initialization parameter specifies a PMEM filestore that the database instance mounts automatically when it starts. The parameter is set to an ordered pair of strings. The first string in the parameter value list is the directory where PMEM Filestore is mounted. The second string is the backing file.

`V$PMEM_FILESTORE` shows metadata about a PMEM filestore. For example, you can view the directory path of the mount point, file path for the backing file, and the filestore size.

**See Also:**

*Oracle Database Administrator's Guide* to learn how to create and manage a PMEM filestore

## Overview of Data Files

At the operating system level, Oracle Database stores data in structures called **data files**. Every Oracle database, whether a CDB or PDB, must have at least one data file.

- Use of Data Files
  Oracle Database physically stores tablespace data in data files.
- Permanent and Temporary Data Files
  A **permanent tablespace** contains persistent schema objects. Objects in permanent tablespaces are stored in data files.
- Online and Offline Data Files
  Every data file is either online (available) or offline (unavailable).
- Data File Structure
  Oracle Database creates a data file for a tablespace by allocating the specified amount of disk space plus the overhead for the data file header. The operating system under which Oracle Database runs is responsible for clearing old information and authorizations from a file before allocating it to the database.

## Use of Data Files

Oracle Database physically stores tablespace data in data files.

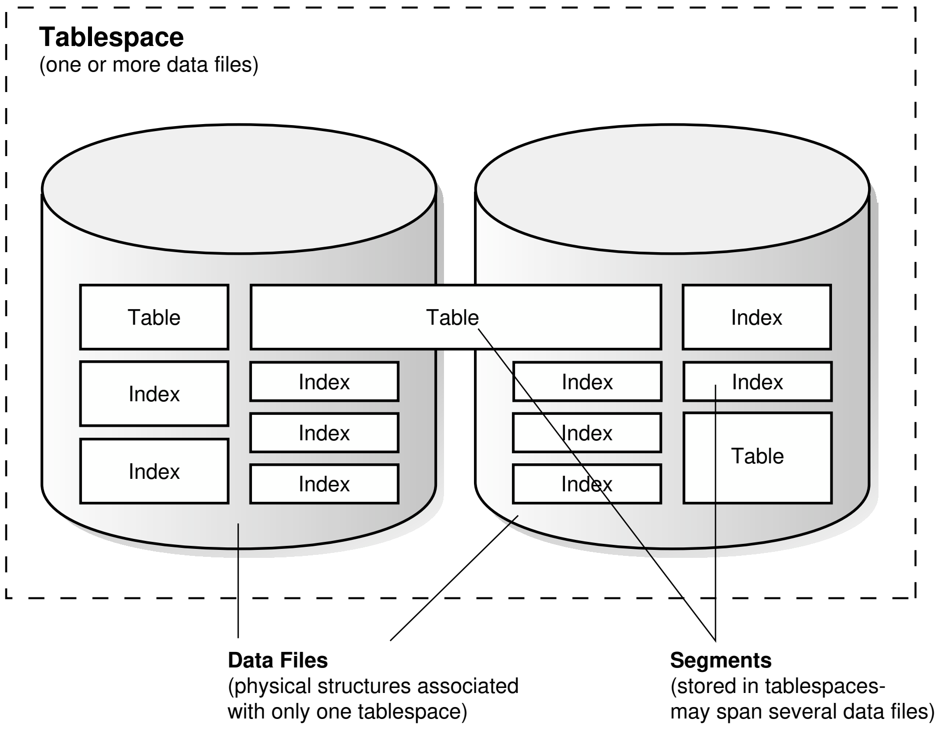
Each nonpartitioned **schema object** and each partition of an object is stored in its own **segment**, which belongs to only one tablespace. For example, the data for a nonpartitioned table is stored in a single segment, which in turn is stored in one tablespace. Tablespaces and data files are closely related, but have important differences:

- Each tablespace consists of one or more data files, which conform to the operating system in which Oracle Database is running.
- The data for a database is collectively stored in the data files located in each tablespace of the database.
- A segment can span one or more data files, but it cannot span multiple tablespaces.
- A CDB must have the `SYSTEM` and `SYSAUX` tablespaces. Each PDB within a CDB has its own `SYSTEM` and `SYSAUX` tablespaces. Oracle Database automatically allocates the first data files of any database for the `SYSTEM` tablespace during database creation.

  The `SYSTEM` tablespace contains the data dictionary, a set of tables that contains database metadata. Typically, a database also has an undo tablespace and a temporary tablespace (usually named `TEMP`).

The following figure shows the relationship between tablespaces, data files, and segments.

**Figure 14-5 Data Files and Tablespaces**



**See Also:**

- "Overview of Tablespaces"
- *Oracle Database Administrator's Guide* to learn how to manage data files

## Permanent and Temporary Data Files

A **permanent tablespace** contains persistent schema objects. Objects in permanent tablespaces are stored in data files.

A temporary tablespace contains schema objects only for the duration of a session. Locally managed temporary tablespaces have temporary files (temp files), which are special files designed to store data in hash, sort, and other operations. Temp files also store result set data when insufficient space exists in memory.

Temp files are similar to permanent data files, with the following exceptions:

- Permanent database objects such as tables are never stored in temp files.
- Temp files are always set to `NOLOGGING` mode, which means that they never have redo generated for them. Media recovery does not recognize temp files.
- You cannot make a temp file read-only.
- You cannot create a temp file with the `ALTER DATABASE` statement.

- When you create or resize temp files, they are not always guaranteed allocation of disk space for the file size specified. On file systems such as Linux and UNIX, temp files are created as *sparse files*. In this case, disk blocks are allocated not at file creation or resizing, but as the blocks are accessed for the first time.

  **Note:**

  Sparse files enable fast temp file creation and resizing; however, the disk could run out of space later when the temp files are accessed.

- Temp file information is shown in the data dictionary view `DBA_TEMP_FILES` and the dynamic performance view `V$TEMPFILE`, but not in `DBA_DATA_FILES` or the `V$DATAFILE` view.

**See Also:**

- "Temporary Tablespaces"
- *Oracle Database Administrator’s Guide* to learn how to manage temp files

## Online and Offline Data Files

Every data file is either online (available) or offline (unavailable).

You can alter the availability of individual data files or temp files by taking them offline or bringing them online. The database cannot access offline data files until they are brought online.

You may take data files offline for many reasons, including performing offline backups or block corruption. The database takes a data file offline automatically if the database cannot write to it.

Like a data file, a tablespace itself is offline or online. When you take a data file offline in an online tablespace, the tablespace itself remains online. You can make all data files of a tablespace temporarily unavailable by taking the tablespace itself offline.

Starting in Oracle Database 12c, you can use the `ALTER DATABASE MOVE DATAFILE` statement to move an online data file from one physical file to another while the database is open and accessing the file. You can use this technique to achieve the following goals:

- Move a tablespace from one kind of storage to another
- Move data files that are accessed infrequently to lower cost storage
- Make a tablespace read-only and move its data files to write-once storage, such as a write once read many (WORM) drive
- Move a database into Oracle ASM

**See Also:**

- "Online and Offline Tablespaces"
- *Oracle Database Administrator's Guide* to learn how to alter data file availability
- *Oracle Database Administrator's Guide* to learn how to move online data files
- *Oracle Database SQL Language Reference* to learn about `ALTER DATABASE . . . MOVE DATAFILE`

## Data File Structure

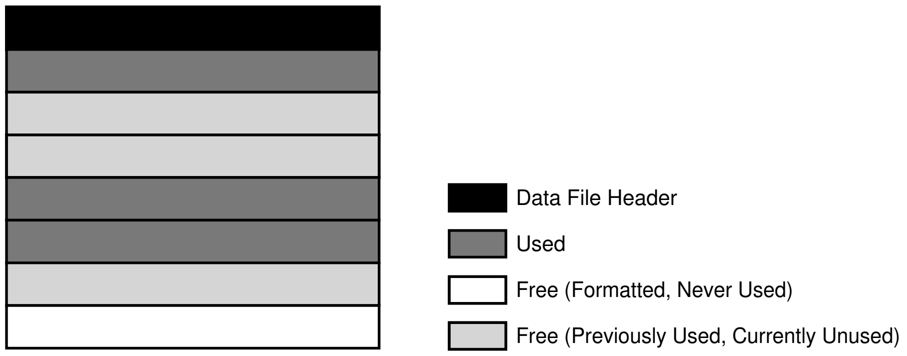
Oracle Database creates a data file for a tablespace by allocating the specified amount of disk space plus the overhead for the data file header. The operating system under which Oracle Database runs is responsible for clearing old information and authorizations from a file before allocating it to the database.

The data file header contains metadata about the data file such as its size and checkpoint SCN. Each header contains an absolute file number, which uniquely identifies the data file within the database, and a relative file number, which uniquely identifies a data file within a tablespace.

When Oracle Database first creates a data file, the allocated disk space is formatted but contains no user data. However, the database reserves the space to hold the data for future segments of the associated tablespace. As the data grows in a tablespace, Oracle Database uses the free space in the data files to allocate extents for the segment.

The following figure illustrates the different types of space in a data file. Extents are either used, which means they contain segment data, or free, which means they are available for reuse. Over time, updates and deletions of objects within a tablespace can create pockets of empty space that individually are not large enough to be reused for new data. This type of empty space is called *fragmented free space*.

**Figure 14-6 Space in a Data File**


Data File Header
Used
Free (Formatted, Never Used)
Free (Previously Used, Currently Unused)

**See Also:**

*Oracle Database Administrator's Guide* to learn how to view data file information

## Overview of Control Files

The database **control file** is a small binary file associated with only one CDB.

Each CDB has one unique control file, although multiple identical copies are permitted. PDBs do not have separate control files.

- Use of Control Files
  Oracle Database uses the control file to locate database files and to manage the state of the database generally.
- Multiple Control Files
  Oracle Database enables multiple, identical control files to be open concurrently and written to the same database. By multiplexing a control file on different disks, the database can achieve redundancy and thereby avoid a single point of failure.
- Control File Structure
  Information about the database is stored in different sections of the control file. Each section is a set of records about an aspect of the database.

### Use of Control Files

Oracle Database uses the control file to locate database files and to manage the state of the database generally.

A control file contains information such as the following:

- The database name and database unique identifier (DBID)
- The time stamp of database creation
- Information about data files, online redo log files, and archived redo log files
- Tablespace information
- RMAN backups

The control file serves the following purposes:

- It contains information about data files, online redo log files, and so on that are required to open the database.

  The control file tracks structural changes to the database. For example, when an administrator adds, renames, or drops a data file or online redo log file, the database updates the control file to reflect this change.
- It contains metadata that must be accessible when the database is not open.

  For example, the control file contains information required to recover the database, including checkpoints. A checkpoint indicates the SCN in the redo stream where instance recovery would be required to begin. Every committed change before a checkpoint SCN is guaranteed to be saved on disk in the data files. At least every three seconds the checkpoint process records information in the control file about the checkpoint position in the online redo log.

Oracle Database reads and writes to the control file continuously during database use and must be available for writing whenever the database is open. For example, recovering a database involves reading from the control file the names of all the data files contained in the database. Other operations, such as adding a data file, update the information stored in the control file.

**See Also:**

- "Overview of Instance Recovery"
- "Checkpoint Process (CKPT)"
- *Oracle Database Administrator's Guide* to learn how to manage the control file

## Multiple Control Files

Oracle Database enables multiple, identical control files to be open concurrently and written to the same database. By multiplexing a control file on different disks, the database can achieve redundancy and thereby avoid a single point of failure.

**Note:**

Oracle recommends that you maintain multiple control file copies, each on a different disk.

If a control file becomes unusable, then the database instance fails when it attempts to access the damaged control file. When other current control file copies exist, then you can remount the database and open it without media recovery. If *all* control files of a database are lost, however, then the database instance fails and media recovery is required. Media recovery is not straightforward if an older backup of a control file must be used because a current copy is not available.

**See Also:**

- *Oracle Database Administrator's Guide* to learn how to maintain multiple control files
- *Oracle Database Backup and Recovery User's Guide* to learn how to back up and restore control files

## Control File Structure

Information about the database is stored in different sections of the control file. Each section is a set of records about an aspect of the database.

For example, one section in the control file tracks data files and contains a set of records, one for each data file. Each section is stored in multiple logical control file blocks. Records can span blocks within a section.

The control file contains the following types of records:

- Circular reuse records

A circular reuse record contains noncritical information that is eligible to be overwritten if needed. When all available record slots are full, the database either expands the control file to make room for a new record or overwrites the oldest record. Examples include records about archived redo log files and RMAN backups.

- Noncircular reuse records

  A noncircular reuse record contains critical information that does not change often and cannot be overwritten. Examples of information include tablespaces, data files, online redo log files, and redo threads. Oracle Database never reuses these records unless the corresponding object is dropped from the tablespace.

You can query the dynamic performance views, also known as `V$` views, to view the information stored in the control file. For example, you can query `V$DATABASE` to obtain the database name and DBID. However, only the database can modify the information in the control file.

Reading and writing the control file blocks is different from reading and writing data blocks. For the control file, Oracle Database reads and writes directly from the disk to the program global area (PGA). Each process allocates a certain amount of its PGA memory for control file blocks.

**See Also:**

- "Overview of the Dynamic Performance Views"
- *Oracle Database Reference* to learn about the `V$CONTROLFILE_RECORD_SECTION` view
- *Oracle Database Reference* to learn about the `CONTROL_FILE_RECORD_KEEP_TIME` initialization parameter

## Overview of the Online Redo Log

The most crucial structure for recovery is the **online redo log**, which consists of two or more preallocated files that store changes to the database as they occur. The online redo log records changes to the data files.

- Use of the Online Redo Log
  The database maintains online redo log files to protect against data loss. Specifically, after an instance failure, the online redo log files enable Oracle Database to recover committed data that it has not yet written to the data files.
- How Oracle Database Writes to the Online Redo Log
  The online redo log for a database instance is called a **redo thread**.
- Structure of the Online Redo Log
  Online redo log files contain redo records.

### Use of the Online Redo Log

The database maintains online redo log files to protect against data loss. Specifically, after an instance failure, the online redo log files enable Oracle Database to recover committed data that it has not yet written to the data files.

Server processes write every transaction synchronously to the redo log buffer, which the LGWR process then writes to the online redo log. Contents of the online redo log include uncommitted transactions, and schema and object management statements.

As the database makes changes to the undo segments, the database also writes these changes to the online redo logs. Consequently, the online redo log always contains the undo data for permanent objects. You can configure the database to store all undo data for temporary objects in a temporary undo segment, which saves space and improves performance, or allow the database to store both permanent and temporary undo data in the online redo log.

Oracle Database uses the online redo log only for recovery. However, administrators can query online redo log files through a SQL interface in the Oracle LogMiner utility. Redo log files are a useful source of historical information about database activity.

**See Also:**

- "Overview of Instance Recovery"
- "Temporary Undo Segments"
- "Process Architecture" to learn about Oracle processes
- *Oracle Database Administrator's Guide* to learn about temporary undo segments
- *Oracle Database Reference* to learn about the `TEMP_UNDO_ENABLED` initialization parameter

## How Oracle Database Writes to the Online Redo Log

The online redo log for a database instance is called a **redo thread**.

In single-instance configurations, only one instance accesses a database, so only one redo thread is present. In an Oracle Real Application Clusters (Oracle RAC) configuration, however, multiple instances concurrently access a database, with each instance having its own redo thread. A separate redo thread for each instance avoids contention for a single set of online redo log files.

An online redo log consists of two or more online redo log files. Oracle Database requires a minimum of two files to guarantee that one file is always available for writing in case the other file is in the process of being cleared or archived.

- Online Redo Log Switches
  Oracle Database uses only one online redo log file at a time to store records written from the redo log buffer.
- Multiple Copies of Online Redo Log Files
  Oracle Database can automatically maintain two or more identical copies of the online redo log in separate locations.
- Archived Redo Log Files
  An **archived redo log file** is a copy of a filled member of an online redo log group.

**See Also:**

*Oracle Real Application Clusters Administration and Deployment Guide* to learn about online redo log groups in Oracle RAC

## Online Redo Log Switches

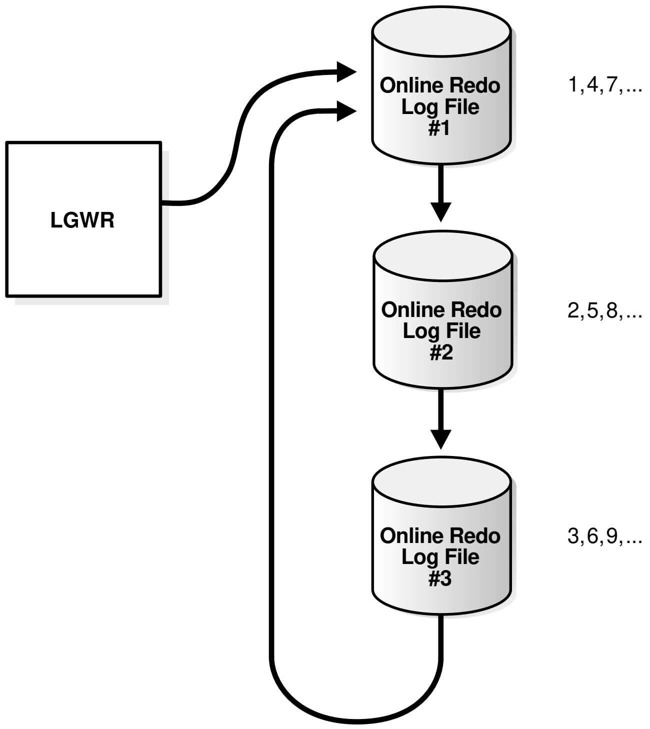
Oracle Database uses only one online redo log file at a time to store records written from the redo log buffer.

The online redo log file to which the log writer process (LGWR) process is actively writing is called the current online redo log file.

A log switch occurs when the database stops writing to one online redo log file and begins writing to another. Normally, a switch occurs when the current online redo log file is full and writing must continue. However, you can configure log switches to occur at regular intervals, regardless of whether the current online redo log file is filled, and force log switches manually.

Log writer writes to online redo log files circularly. When log writer fills the last available online redo log file, the process writes to the first log file, restarting the cycle. Figure 14-7 illustrates the circular writing of the redo log.

**Figure 14-7    Reuse of Online Redo Log Files**



The numbers in Figure 14-7 shows the sequence in which LGWR writes to each online redo log file. The database assigns each file a new log sequence number when a log switches and log writers begins writing to it. When the database reuses an online redo log file, this file receives the next available log sequence number.

Filled online redo log files are available for reuse depending on the archiving mode:

- If archiving is disabled, which means that the database is in `NOARCHIVELOG` mode, then a filled online redo log file is available after the changes recorded in it have been checkpointed (written) to disk by database writer (DBW).
- If archiving is enabled, which means that the database is in ARCHIVELOG mode, then a filled online redo log file is available to log writer after the changes have been written to the data files *and* the file has been archived.

In some circumstances, log writer may be prevented from reusing an existing online redo log file. An active online redo log file is required for instance recovery, where as an inactive online redo log file is not required for instance recovery. Also, an online redo log file may be in the process of being cleared.

**See Also:**

- "Overview of Background Processes"
- *Oracle Database Administrator's Guide* to learn how to manage the online redo log

## Multiple Copies of Online Redo Log Files

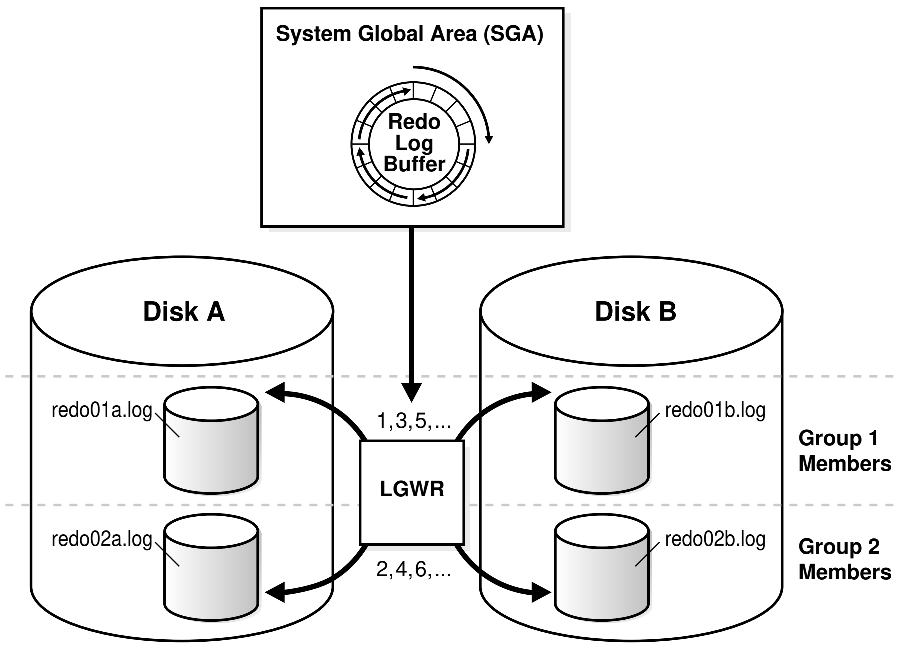
Oracle Database can automatically maintain two or more identical copies of the online redo log in separate locations.

An online redo log group consists of an online redo log file and its redundant copies. Each identical copy is a member of the online redo log group. Each group is defined by a number, such as group 1, group 2, and so on.

Maintaining multiple members of an online redo log group protects against the loss of the redo log. Ideally, the locations of the members should be on separate disks so that the failure of one disk does not cause the loss of the entire online redo log.

In Figure 14-8, `A_LOG1` and `B_LOG1` are identical members of group 1, while `A_LOG2` and `B_LOG2` are identical members of group 2. Each member in a group must be the same size. LGWR writes concurrently to group 1 (members `A_LOG1` and `B_LOG1`), then writes concurrently to group 2 (members `A_LOG2` and `B_LOG2`), then writes to group 1, and so on. LGWR never writes concurrently to members of different groups.

**Figure 14-8 Multiple Copies of Online Redo Log Files**



**Note:**

Oracle recommends that you multiplex the online redo log. The loss of log files can be catastrophic if recovery is required. When you multiplex the online redo log, the database must increase the amount of I/O it performs. Depending on your system, this additional I/O may impact overall database performance.

**See Also:**

- *Oracle Database Administrator's Guide* to learn how to maintain multiple copies of the online redo log files
- *Oracle Data Guard Concepts and Administration* to learn about the automated transfer of redo data between members of a Data Guard configuration

## Archived Redo Log Files

An **archived redo log file** is a copy of a filled member of an online redo log group.

An archived redo log file is not considered part of the database. Rather, it is an offline copy of an online redo log file created by the database and written to a user-specified location.

Archived redo log files are a crucial part of a backup and recovery strategy. You can use archived redo log files to:

- Recover a database backup

- Update a standby database
- Obtain information about the history of a database using the Oracle LogMiner utility

The operation of generating an archived redo log file is known as archiving. This operation is either automatic or manual. It is only possible when the database is in `ARCHIVELOG` mode.

An archived redo log file includes the redo entries and the log sequence number of the identical member of the online redo log group. In "Multiple Copies of Online Redo Log Files", files `A_LOG1` and `B_LOG1` are identical members of Group 1. If the database is in `ARCHIVELOG` mode, and if automatic archiving is enabled, then the archiver process (ARCn) will archive one of these files. If `A_LOG1` is corrupted, then the process can archive `B_LOG1`. The archived redo log contains a copy of every redo log group that existed starting from the time that you enabled archiving.

**See Also:**

- *Oracle Database Utilities* to learn about Oracle LogMiner
- *Oracle Database Administrator's Guide* to learn how to manage the archived redo log
- *Oracle Data Guard Concepts and Administration* to learn how to configure standby redo log archival

## Structure of the Online Redo Log

Online redo log files contain redo records.

A redo record is made up of a group of change vectors, each of which describes a change to a data block. For example, an update to a salary in the `employees` table generates a redo record that describes changes to the data segment block for the table, the undo segment data block, and the transaction table of the undo segments.

The redo records have all relevant metadata for the change, including the following:

- SCN and time stamp of the change
- Transaction ID of the transaction that generated the change
- SCN and time stamp when the transaction committed (if it committed)
- Type of operation that made the change
- Name and type of the modified data segment

**See Also:**

"Overview of Data Blocks"