2

Data Blocks, Extents, and Segments

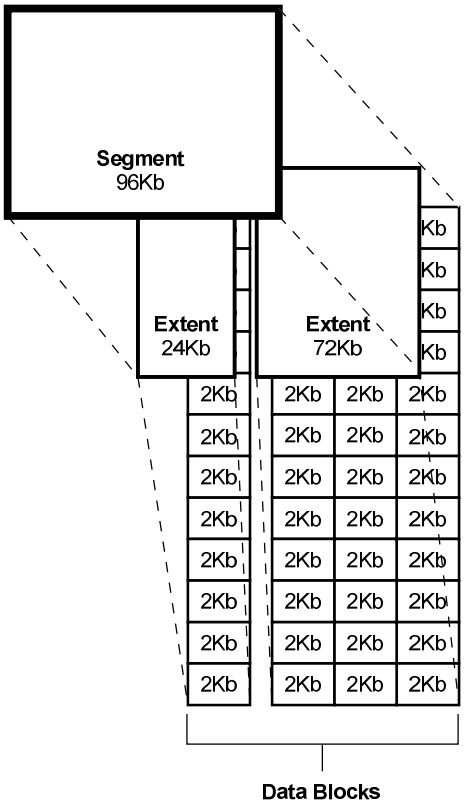
This chapter describes the nature of and relationships among the logical storage structures in the Oracle database server.

This chapter contains the following topics:

Introduction to Data Blocks, Extents, and Segments

Oracle Database allocates logical database space for all data in a database. The units of database space allocation are data blocks, extents, and segments. Figure 2–1 shows the relationships among these data structures.

Figure 2–1 The Relationships Among Segments, Extents, and Data Blocks



At the finest level of granularity, Oracle Database stores data in **data blocks** (also called **logical blocks**, **Oracle blocks**, or **pages**). One data block corresponds to a specific number of bytes of physical database space on disk.

The next level of logical database space is an **extent**. An extent is a specific number of contiguous data blocks allocated for storing a specific type of information.

The level of logical database storage greater than an extent is called a **segment**. A segment is a set of extents, each of which has been allocated for a specific data structure and all of which are stored in the same tablespace. For example, each table's data is stored in its own **data segment**, while each index's data is stored in its own **index segment**. If the table or index is partitioned, each partition is stored in its own segment.

Oracle Database allocates space for segments in units of one extent. When the existing extents of a segment are full, Oracle Database allocates another extent for that segment. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

A segment and all its extents are stored in one tablespace. Within a tablespace, a segment can include extents from more than one file; that is, the segment can span datafiles. However, each extent can contain data from only one datafile.

Although you can allocate additional extents, the blocks themselves are allocated separately. If you allocate an extent to a specific instance, the blocks are immediately allocated to the free list. However, if the extent is not allocated to a specific instance, then the blocks themselves are allocated only when the high water mark moves. The **high water mark** is the boundary between used and unused space in a segment.

Note: Oracle recommends that you manage free space automatically. See "Free Space Management" on page 2-5.

Overview of Data Blocks

Oracle Database manages the storage space in the datafiles of a database in units called **data blocks**. A data block is the smallest unit of data used by a database. In contrast, at the physical, operating system level, all data is stored in bytes. Each operating system has a **block size**. Oracle Database requests data in multiples of Oracle Database data blocks, not operating system blocks.

The standard block size is specified by the `DB_BLOCK_SIZE` initialization parameter. In addition, you can specify of up to five nonstandard block sizes. The data block sizes should be a multiple of the operating system's block size within the maximum limit to avoid unnecessary I/O. Oracle Database data blocks are the smallest units of storage that Oracle Database can use or allocate.

This section includes the following topics:

- Data Block Format
- Free Space Management
- PCTFREE, PCTUSED, and Row Chaining

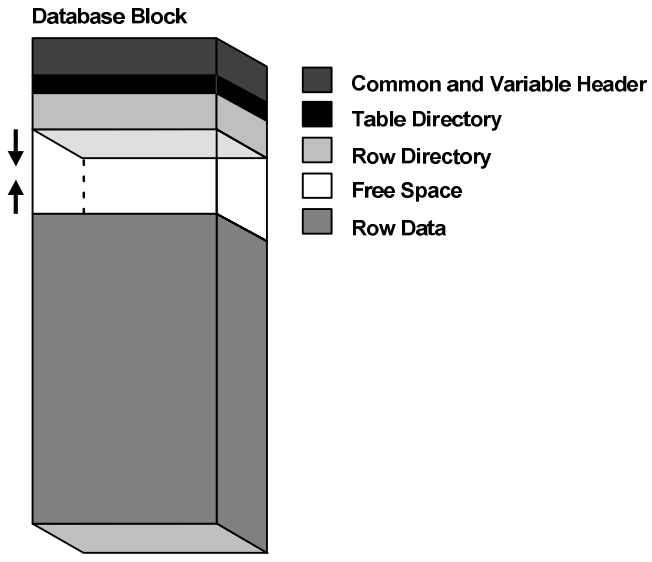
See Also:

- Your Oracle Database operating system-specific documentation for more information about data block sizes
- Multiple Block Sizes on page 3-11

Data Block Format

The Oracle Database data block format is similar regardless of whether the data block contains table, index, or clustered data. Figure 2–2 illustrates the format of a data block.

Figure 2–2 Data Block Format



This section discusses the following components of the data block:

- Header (Common and Variable)
- Table Directory
- Row Directory
- Overhead
- Row Data
- Free Space

Header (Common and Variable)

The header contains general block information, such as the block address and the type of segment (for example, data or index).

Table Directory

This portion of the data block contains information about the table having rows in this block.

Row Directory

This portion of the data block contains information about the actual rows in the block (including addresses for each row piece in the row data area).

After the space has been allocated in the row directory of a data block's overhead, this space is not reclaimed when the row is deleted. Therefore, a block that is currently empty but had up to 50 rows at one time continues to have 100 bytes allocated in the header for the row directory. Oracle Database reuses this space only when new rows are inserted in the block.

Overhead

The data block header, table directory, and row directory are referred to collectively as **overhead**. Some block overhead is fixed in size; the total block overhead size is variable. On average, the fixed and variable portions of data block overhead total 84 to 107 bytes.

Row Data

This portion of the data block contains table or index data. Rows can span blocks.

See Also: "Row Chaining and Migrating" on page 2-5

Free Space

Free space is allocated for insertion of new rows and for updates to rows that require additional space (for example, when a trailing null is updated to a nonnull value).

In data blocks allocated for the data segment of a table or cluster, or for the index segment of an index, free space can also hold transaction entries. A **transaction entry** is required in a block for each `INSERT`, `UPDATE`, `DELETE`, and `SELECT...FOR UPDATE` statement accessing one or more rows in the block. The space required for transaction entries is operating system dependent; however, transaction entries in most operating systems require approximately 23 bytes.

Free Space Management

Free space can be managed automatically or manually.

Free space can be managed automatically inside database segments. The in-segment free/used space is tracked using bitmaps, as opposed to free lists. Automatic segment-space management offers the following benefits:

- Ease of use
- Better space utilization, especially for the objects with highly varying row sizes
- Better run-time adjustment to variations in concurrent access
- Better multi-instance behavior in terms of performance/space utilization

You specify automatic segment-space management when you create a locally managed tablespace. The specification then applies to all segments subsequently created in this tablespace.

See Also: *Oracle Database Administrator's Guide*

This section includes the following topics:

- Availability and Optimization of Free Space in a Data Block
- Row Chaining and Migrating

Availability and Optimization of Free Space in a Data Block

Two types of statements can increase the free space of one or more data blocks: `DELETE` statements, and `UPDATE` statements that update existing values to smaller values. The released space from these types of statements is available for subsequent `INSERT` statements under the following conditions:

- If the `INSERT` statement is in the same transaction and subsequent to the statement that frees space, then the `INSERT` statement can use the space made available.
- If the `INSERT` statement is in a separate transaction from the statement that frees space (perhaps being run by another user), then the `INSERT` statement can use the space made available only after the other transaction commits and only if the space is needed.

Released space may or may not be contiguous with the main area of free space in a data block. Oracle Database coalesces the free space of a data block *only* when (1) an `INSERT` or `UPDATE` statement attempts to use a block that contains enough free space to contain a new row piece, and (2) the free space is fragmented so the row piece cannot be inserted in a contiguous section of the block. Oracle Database does this compression only in such situations, because otherwise the performance of a database system decreases due to the continuous compression of the free space in data blocks.

Row Chaining and Migrating

In two circumstances, the data for a row in a table may be too large to fit into a single data block. In the first case, the row is too large to fit into one data block when it is first inserted. In this case, Oracle Database stores the data for the row in a **chain** of data blocks (one or more) reserved for that segment. Row chaining most often occurs with large rows, such as rows that contain a column of datatype `LONG` or `LONG RAW`. Row chaining in these cases is unavoidable.

However, in the second case, a row that originally fit into one data block is updated so that the overall row length increases, and the block's free space is already completely

filled. In this case, Oracle Database **migrates** the data for the entire row to a new data block, assuming the entire row can fit in a new block. Oracle Database preserves the original row piece of a migrated row to point to the new block containing the migrated row. The rowid of a migrated row does not change.

When a row is chained or migrated, I/O performance associated with this row decreases because Oracle Database must scan more than one data block to retrieve the information for the row.

See Also:

- "Row Format and Size" on page 5-5 for more information on the format of a row and a row piece
- "Rowids of Row Pieces" on page 5-7 for more information on rowids
- "Physical Rowids" on page 26-14 for information about rowids
- *Oracle Database Performance Tuning Guide* for information about reducing chained and migrated rows and improving I/O performance

PCTFREE, PCTUSED, and Row Chaining

For manually managed tablespaces, two space management parameters, `PCTFREE` and `PCTUSED`, enable you to control the use of free space for inserts and updates to the rows in all the data blocks of a particular segment. Specify these parameters when you create or alter a table or cluster (which has its own data segment). You can also specify the storage parameter `PCTFREE` when creating or altering an index (which has its own index segment).

This section includes the following topics:

- The PCTFREE Parameter
- The PCTUSED Parameter
- How PCTFREE and PCTUSED Work Together

Note: This discussion does not apply to LOB datatypes (`BLOB`, `CLOB`, `NCLOB`, and `BFILE`). They do not use the `PCTFREE` storage parameter or free lists.

See "Overview of LOB Datatypes" on page 26-11 for information.

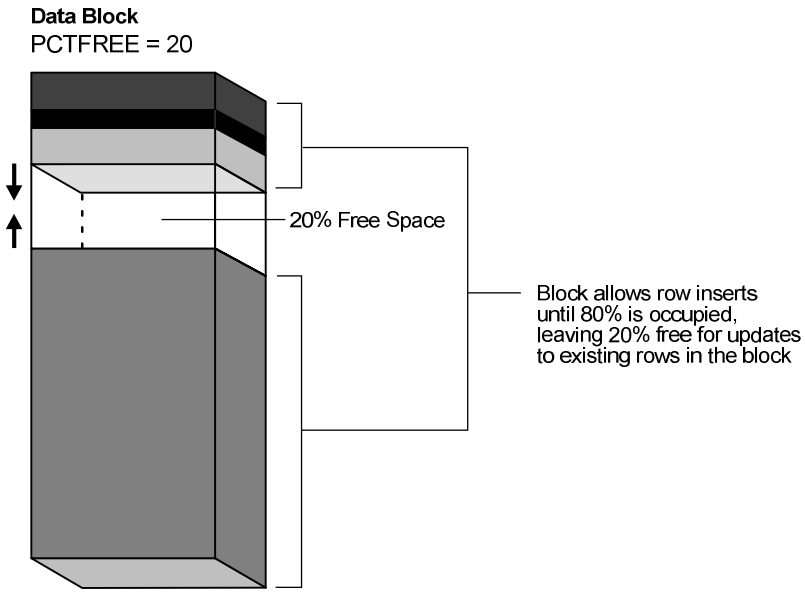
The PCTFREE Parameter

The `PCTFREE` parameter sets the minimum percentage of a data block to be **reserved** as free space for possible updates to rows that already exist in that block. For example, assume that you specify the following parameter within a `CREATE TABLE` statement:

```
PCTFREE 20
```

This states that 20% of each data block in this table's data segment be kept free and available for possible updates to the existing rows already within each block. New rows can be added to the row data area, and corresponding information can be added to the variable portions of the overhead area, until the row data and overhead total 80% of the total block size. Figure 2–3 illustrates `PCTFREE`.

Figure 2–3 PCTFREE



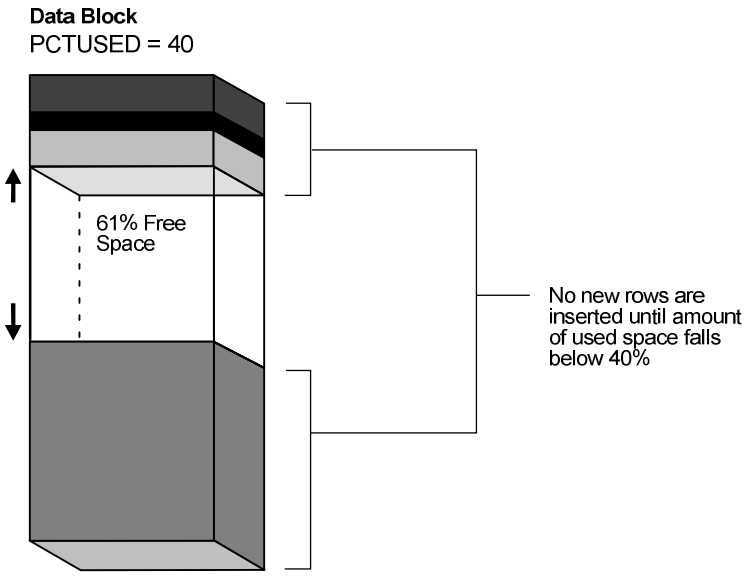
The PCTUSED Parameter

The `PCTUSED` parameter sets the minimum percentage of a block that can be used for row data plus overhead before new rows are added to the block. After a data block is filled to the limit determined by `PCTFREE`, Oracle Database considers the block unavailable for the insertion of new rows until the percentage of that block falls beneath the parameter `PCTUSED`. Until this value is achieved, Oracle Database uses the free space of the data block only for updates to rows already contained in the data block. For example, assume that you specify the following parameter in a `CREATE TABLE` statement:

```
PCTUSED 40
```

In this case, a data block used for this table's data segment is considered unavailable for the insertion of any new rows until the amount of used space in the block falls to 39% or less (assuming that the block's used space has previously reached `PCTFREE`). Figure 2–4 illustrates this.

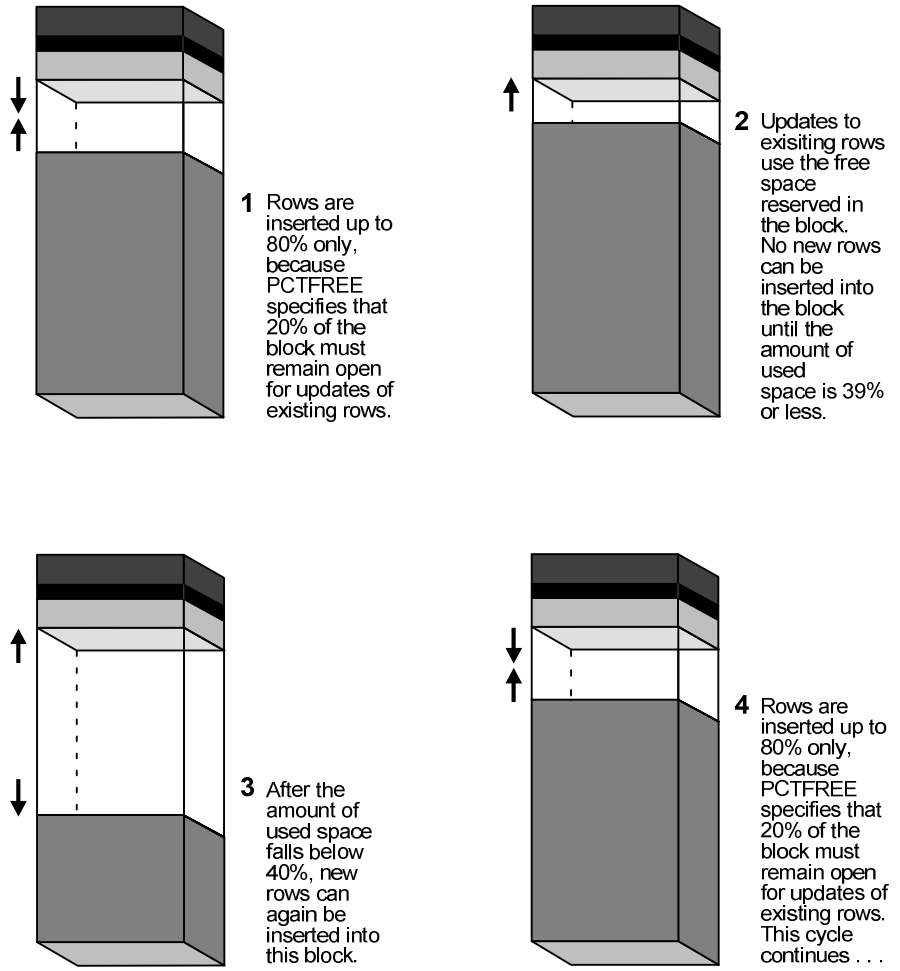
Figure 2–4 PCTUSED



How PCTFREE and PCTUSED Work Together

PCTFREE and PCTUSED work together to optimize the use of space in the data blocks of the extents within a data segment. Figure 2–5 illustrates the interaction of these two parameters.

Figure 2–5 Maintaining the Free Space of Data Blocks with PCTFREE and PCTUSED



In a newly allocated data block, the space available for inserts is the block size minus the sum of the block overhead and free space (`PCTFREE`). Updates to existing data can use any available space in the block. Therefore, updates can reduce the available space of a block to less than `PCTFREE`, the space reserved for updates but not accessible to inserts.

For each data and index segment, Oracle Database maintains one or more **free lists**—lists of data blocks that have been allocated for that segment's extents and have free space greater than `PCTFREE`. These blocks are available for inserts. When you issue an `INSERT` statement, Oracle Database checks a free list of the table for the first available data block and uses it if possible. If the free space in that block is not large enough to accommodate the `INSERT` statement, and the block is at least `PCTUSED`, then Oracle Database takes the block off the free list. Multiple free lists for each segment can reduce contention for free lists when concurrent inserts take place.

After you issue a `DELETE` or `UPDATE` statement, Oracle Database processes the statement and checks to see if the space being used in the block is now less than `PCTUSED`. If it is, then the block goes to the beginning of the transaction free list, and it is the first of the available blocks to be used in that transaction. When the transaction commits, free space in the block becomes available for other transactions.

Overview of Extents

An extent is a logical unit of database storage space allocation made up of a number of contiguous data blocks. One or more extents in turn make up a segment. When the existing space in a segment is completely used, Oracle Database allocates a new extent for the segment.

This section includes the following topics:

- When Extents Are Allocated
- Determine the Number and Size of Extents
- How Extents Are Allocated
- When Extents Are Deallocated

When Extents Are Allocated

When you create a table, Oracle Database allocates to the table's data segment an **initial extent** of a specified number of data blocks. Although no rows have been inserted yet, the Oracle Database data blocks that correspond to the initial extent are reserved for that table's rows.

If the data blocks of a segment's initial extent become full and more space is required to hold new data, Oracle Database automatically allocates an **incremental extent** for that segment. An incremental extent is a subsequent extent of the same or greater size than the previously allocated extent in that segment.

For maintenance purposes, the header block of each segment contains a directory of the extents in that segment.

Note: This chapter applies to serial operations, in which one server process parses and runs a SQL statement. Extents are allocated somewhat differently in parallel SQL statements, which entail multiple server processes.

Determine the Number and Size of Extents

Storage parameters expressed in terms of extents define every segment. Storage parameters apply to all types of segments. They control how Oracle Database allocates free database space for a given segment. For example, you can determine how much space is initially reserved for a table's data segment or you can limit the number of extents the table can allocate by specifying the storage parameters of a table in the `STORAGE` clause of the `CREATE TABLE` statement. If you do not specify a table's storage parameters, then it uses the default storage parameters of the tablespace.

You can have dictionary managed tablespaces, which rely on data dictionary tables to track space utilization, or locally managed tablespaces, which use bitmaps (instead of data dictionary tables) to track used and free space. Because of the better performance and easier manageability of locally managed tablespaces, the default for non-`SYSTEM` permanent tablespaces is locally managed whenever the type of extent management is not explicitly specified.

A tablespace that manages its extents locally can have either uniform extent sizes or variable extent sizes that are determined automatically by the system. When you create the tablespace, the `UNIFORM` or `AUTOALLOCATE` (system-managed) clause specifies the type of allocation.

- For uniform extents, you can specify an extent size or use the default size, which is 1 MB. Ensure that each extent contains at least five database blocks, given the database block size. Temporary tablespaces that manage their extents locally can only use this type of allocation.
- For system-managed extents, Oracle Database determines the optimal size of additional extents, with a minimum extent size of 64 KB. If the tablespaces are created with 'segment space management auto', and if the database block size is 16K or higher, then Oracle Database manages segment size by creating extents with a minimum size of 1M. This is the default for permanent tablespaces.

The storage parameters `INITIAL`, `NEXT`, `PCTINCREASE`, and `MINEXTENTS` cannot be specified at the tablespace level for locally managed tablespaces. They can, however, be specified at the segment level. In this case, `INITIAL`, `NEXT`, `PCTINCREASE`, and `MINEXTENTS` are used together to compute the initial size of the segment. After the segment size is computed, internal algorithms determine the size of each extent.

See Also:

- "Managing Space in Tablespaces" on page 3-9
- "Bigfile Tablespaces" on page 3-5
- *Oracle Database Administrator's Guide*

How Extents Are Allocated

Oracle Database uses different algorithms to allocate extents, depending on whether they are locally managed or dictionary managed.

With locally managed tablespaces, Oracle Database looks for free space to allocate to a new extent by first determining a candidate datafile in the tablespace and then searching the datafile's bitmap for the required number of adjacent free blocks. If that datafile does not have enough adjacent free space, then Oracle Database looks in another datafile.

Note: Oracle strongly recommends that you use locally managed tablespaces.

When Extents Are Deallocated

Oracle Database provides a Segment Advisor that helps you determine whether an object has space available for reclamation based on the level of space fragmentation within the object.

See Also:

- *Oracle Database Administrator's Guide* for guidelines on reclaiming segment space
- *Oracle Database SQL Language Reference* for SQL syntax and semantics

In general, the extents of a segment do not return to the tablespace until you drop the schema object whose data is stored in the segment (using a `DROP TABLE` or `DROP CLUSTER` statement). Exceptions to this include the following:

- The owner of a table or cluster, or a user with the `DELETE ANY` privilege, can truncate the table or cluster with a `TRUNCATE...DROP STORAGE` statement.

- A database administrator (DBA) can deallocate unused extents using the following SQL syntax:

  ```
  ALTER TABLE table_name DEALLOCATE UNUSED;
  ```

- Periodically, Oracle Database deallocates one or more extents of a rollback segment if it has the `OPTIMAL` size specified.

When extents are freed, Oracle Database modifies the bitmap in the datafile (for locally managed tablespaces) or updates the data dictionary (for dictionary managed tablespaces) to reflect the regained extents as available space. Any data in the blocks of freed extents becomes inaccessible.

This section includes the following topics:

- Extents in Nonclustered Tables
- Extents in Clustered Tables
- Extents in Materialized Views and Their Logs
- Extents in Indexes
- Extents in Temporary Segments
- Extents in Rollback Segments

See Also:

- *Oracle Database Administrator's Guide*
- *Oracle Database SQL Language Reference*

Extents in Nonclustered Tables

As long as a nonclustered table exists or until you truncate the table, any data block allocated to its data segment remains allocated for the table. Oracle Database inserts new rows into a block if there is enough room. Even if you delete all rows of a table, Oracle Database does not reclaim the data blocks for use by other objects in the tablespace.

After you drop a nonclustered table, this space can be reclaimed when other extents require free space. Oracle Database reclaims all the extents of the table's data and index segments for the tablespaces that they were in and makes the extents available for other schema objects in the same tablespace.

In dictionary managed tablespaces, when a segment requires an extent larger than the available extents, Oracle Database identifies and combines contiguous reclaimed extents to form a larger one. This is called **coalescing** extents. Coalescing extents is not necessary in locally managed tablespaces, because all contiguous free space is available for allocation to a new extent regardless of whether it was reclaimed from one or more extents.

Extents in Clustered Tables

Clustered tables store information in the data segment created for the cluster. Therefore, if you drop one table in a cluster, the data segment remains for the other tables in the cluster, and no extents are deallocated. You can also truncate clusters (except for hash clusters) to free extents.

Extents in Materialized Views and Their Logs

Oracle Database deallocates the extents of materialized views and materialized view logs in the same manner as for tables and clusters.

> **See Also:** "Overview of Materialized Views" on page 5-18

Extents in Indexes

All extents allocated to an index segment remain allocated as long as the index exists. When you drop the index or associated table or cluster, Oracle Database reclaims the extents for other uses within the tablespace.

Extents in Temporary Segments

When Oracle Database completes the execution of a statement requiring a temporary segment, Oracle Database automatically drops the temporary segment and returns the extents allocated for that segment to the associated tablespace. A single sort allocates its own temporary segment in a temporary tablespace of the user issuing the statement and then returns the extents to the tablespaces.

Multiple sorts, however, can use sort segments in temporary tablespaces designated exclusively for sorts. These sort segments are allocated only once for the instance, and they are not returned after the sort, but remain available for other multiple sorts.

A temporary segment in a temporary table contains data for multiple statements of a single transaction or session. Oracle Database drops the temporary segment at the end of the transaction or session, returning the extents allocated for that segment to the associated tablespace.

> **See Also:**
>
> - "Introduction to Temporary Segments" on page 2-14
> - "Temporary Tables" on page 5-10

Extents in Rollback Segments

Oracle Database periodically checks the rollback segments of the database to see if they have grown larger than their optimal size. If a rollback segment is larger than is optimal (that is, it has too many extents), then Oracle Database automatically deallocates one or more extents from the rollback segment.

Overview of Segments

A segment is a set of extents that contains all the data for a specific logical storage structure within a tablespace. For example, for each table, Oracle Database allocates one or more extents to form that table's data segment, and for each index, Oracle Database allocates one or more extents to form its index segment.

This section contains the following topics:

- Introduction to Data Segments
- Introduction to Index Segments
- Introduction to Temporary Segments
- Introduction to Undo Segments and Automatic Undo Management

Introduction to Data Segments

A single data segment in an Oracle Database database holds all of the data for one of the following:

- A table that is not partitioned or clustered
- A partition of a partitioned table
- A cluster of tables

Oracle Database creates this data segment when you create the table or cluster with the `CREATE` statement.

The storage parameters for a table or cluster determine how its data segment's extents are allocated. You can set these storage parameters directly with the appropriate `CREATE` or `ALTER` statement. These storage parameters affect the efficiency of data retrieval and storage for the data segment associated with the object.

Note: Oracle Database creates segments for materialized views and materialized view logs in the same manner as for tables and clusters.

See Also:

- *Oracle Database Advanced Replication* for information on materialized views and materialized view logs
- *Oracle Database SQL Language Reference* for syntax

Introduction to Index Segments

Every nonpartitioned index in an Oracle database has a single index segment to hold all of its data. For a partitioned index, every partition has a single index segment to hold its data.

Oracle Database creates the index segment for an index or an index partition when you issue the `CREATE INDEX` statement. In this statement, you can specify storage parameters for the extents of the index segment and a tablespace in which to create the index segment. (The segments of a table and an index associated with it do not have to occupy the same tablespace.) Setting the storage parameters directly affects the efficiency of data retrieval and storage.

Introduction to Temporary Segments

When processing queries, Oracle Database often requires temporary workspace for intermediate stages of SQL statement parsing and execution. Oracle Database automatically allocates this disk space called a **temporary segment**. Typically, Oracle Database requires a temporary segment as a database area for sorting. Oracle Database does not create a segment if the sorting operation can be done in memory or if Oracle Database finds some other way to perform the operation using indexes.

This section includes the following topics:

- Operations that Require Temporary Segments
- Segments in Temporary Tables and Their Indexes
- How Temporary Segments Are Allocated

Operations that Require Temporary Segments

The following statements sometimes require the use of a temporary segment:

- CREATE INDEX
- SELECT ... ORDER BY
- SELECT DISTINCT ...
- SELECT ... GROUP BY
- SELECT...UNION
- SELECT ... INTERSECT
- SELECT ... MINUS

Some unindexed joins and correlated subqueries can require use of a temporary segment. For example, if a query contains a DISTINCT clause, a GROUP BY, and an ORDER BY, Oracle Database can require as many as two temporary segments.

Segments in Temporary Tables and Their Indexes

Oracle Database can also allocate temporary segments for temporary tables and indexes created on temporary tables. Temporary tables hold data that exists only for the duration of a transaction or session.

> **See Also:** "Temporary Tables" on page 5-10

How Temporary Segments Are Allocated

Oracle Database allocates temporary segments differently for queries and temporary tables.

This section includes the following topics:

- Allocation of Temporary Segments for Queries
- Allocation of Temporary Segments for Temporary Tables and Indexes

Allocation of Temporary Segments for Queries Oracle Database allocates temporary segments as needed during a user session in one of the temporary tablespaces of the user issuing the statement. Specify these tablespaces with a CREATE USER or an ALTER USER statement using the TEMPORARY TABLESPACE clause.

> **Note:** You cannot assign a permanent tablespace as a user's temporary tablespace.

If no temporary tablespace is defined for the user, then the default temporary tablespace is the SYSTEM tablespace. The default storage characteristics of the containing tablespace determine those of the extents of the temporary segment. Oracle Database drops temporary segments when the statement completes.

Because allocation and deallocation of temporary segments occur frequently, create at least one special tablespace for temporary segments. By doing so, you can distribute I/O across disk devices, and you can avoid fragmentation of the SYSTEM and other tablespaces that otherwise hold temporary segments.

Note: When the SYSTEM tablespace is locally managed, you must define a default temporary tablespace when creating a database. A locally managed SYSTEM tablespace cannot be used for default temporary storage.

Entries for changes to temporary segments used for sort operations are not stored in the redo log, except for space management operations on the temporary segment.

See Also:

- "Bigfile Tablespaces" on page 3-5
- Chapter 20, "Database Security" for more information about assigning a user's temporary segment tablespace

Allocation of Temporary Segments for Temporary Tables and Indexes Oracle Database allocates segments for a temporary table when the first INSERT into that table is issued. (This can be an internal insert operation issued by CREATE TABLE AS SELECT.) The first INSERT into a temporary table allocates the segments for the table and its indexes, creates the root page for the indexes, and allocates any LOB segments.

Segments for a temporary table are allocated in a temporary tablespace of the user who created the temporary table.

Oracle Database drops segments for a transaction-specific temporary table at the end of the transaction and drops segments for a session-specific temporary table at the end of the session. If other transactions or sessions share the use of that temporary table, the segments containing their data remain in the table.

See Also: "Temporary Tables" on page 5-10

Introduction to Undo Segments and Automatic Undo Management

Oracle Database maintains information to reverse changes made to the database. This information consists of records of the actions of transactions, collectively known as **undo**. Undo is stored in undo segments in an *undo tablespace*. Oracle Database uses undo information to do the following:

- Rollback an active transaction
- Recover a terminated transaction
- Provide read consistency
- Recovery from logical corruptions

When a ROLLBACK statement is issued, undo records are used to undo changes that were made to the database by the uncommitted transaction. During database recovery, undo records are used to undo any uncommitted changes applied from the redo log to the datafiles. Undo records provide read consistency by maintaining the before image of the data for users who are accessing the data at the same time that another user is changing it. See "How Oracle Database Manages Data Concurrency and Consistency" on page 13-3 for more information on read consistency.

Oracle Database provides a fully automated mechanism, referred to as **automatic undo management**, for managing undo information and space. In this management mode, for all current sessions, the server automatically manages undo segments and space in the undo tablespace.

Automatic undo management eliminates the complexities of managing rollback segment space. In addition, the system automatically tunes itself to provide the best possible retention of undo information to satisfy long-running queries that may require this undo information. Automatic undo management is the default for new installations of Oracle Database. The installation process automatically creates an undo tablespace.

Oracle Database contains an Undo Advisor that provides advice on and helps automate the establishment of your undo environment.

This section includes the following topics:

- Manual Undo Management
- Undo Quota
- Automatic Undo Retention

> **See Also:** *Oracle Database 2 Day DBA* for information on the Undo Advisor and on how to use advisors and see *Oracle Database Administrator's Guide* for more information on using automatic undo management

Manual Undo Management

A database system can also run in manual undo management mode. In manual undo management mode, undo space is managed through rollback segments, and no undo tablespace is used.

Earlier releases of Oracle Database defaulted to manual undo management mode. To change to automatic undo management, it was necessary to first create an undo tablespace and then change an initialization parameter. If your Oracle Database is release 9*i* or later and you want to change to automatic undo management, see *Oracle Database Upgrade Guide* for instructions.

Note: Space management for rollback segments is complex. Oracle strongly recommends using automatic undo management.

Undo Quota

In automatic undo management mode, the system controls exclusively the assignment of transactions to undo segments, and controls space allocation for undo segments. An ill-behaved transaction can potentially consume much of the undo space, thus paralyzing the entire system. The Resource Manager directive `UNDO_POOL` is a more explicit way to control large transactions. This lets database administrators group users into consumer groups, with each group assigned a maximum undo space limit. When the total undo space consumed by a group exceeds the limit, its users cannot make further updates until undo space is freed up by other member transactions ending.

The default value of `UNDO_POOL` is `UNLIMITED`, where users are allowed to consume as much undo space as the undo tablespace has. Database administrators can limit a particular user by using the `UNDO_POOL` directive.

Automatic Undo Retention

After a transaction is committed, undo data is no longer needed for rollback or transaction recovery purposes. However, for consistent read purposes, long-running queries may require this old undo information for producing older images of data

blocks. Furthermore, the success of several Oracle Flashback features can also depend upon the availability of older undo information. For these reasons, it is desirable to retain the old undo information for as long as possible. If the undo tablespace has space available for new transactions, then old undo information can be retained. When available space in the tablespace becomes short, the database begins to overwrite old undo information for transactions that have been committed.

Oracle Database automatically tunes the system to provide the best possible undo retention for the current undo tablespace. The database collects usage statistics and tunes the undo retention period based on these statistics and the undo tablespace size. If the undo tablespace is configured with the `AUTOEXTEND` option, with maximum size not specified, undo retention tuning is slightly different. In this case, the database tunes the undo retention period to be slightly longer than the longest-running query, if space allows.

> **See Also:** *Oracle Database Administrator's Guide* for more details on automatic tuning of undo retention

3

Tablespaces, Datafiles, and Control Files

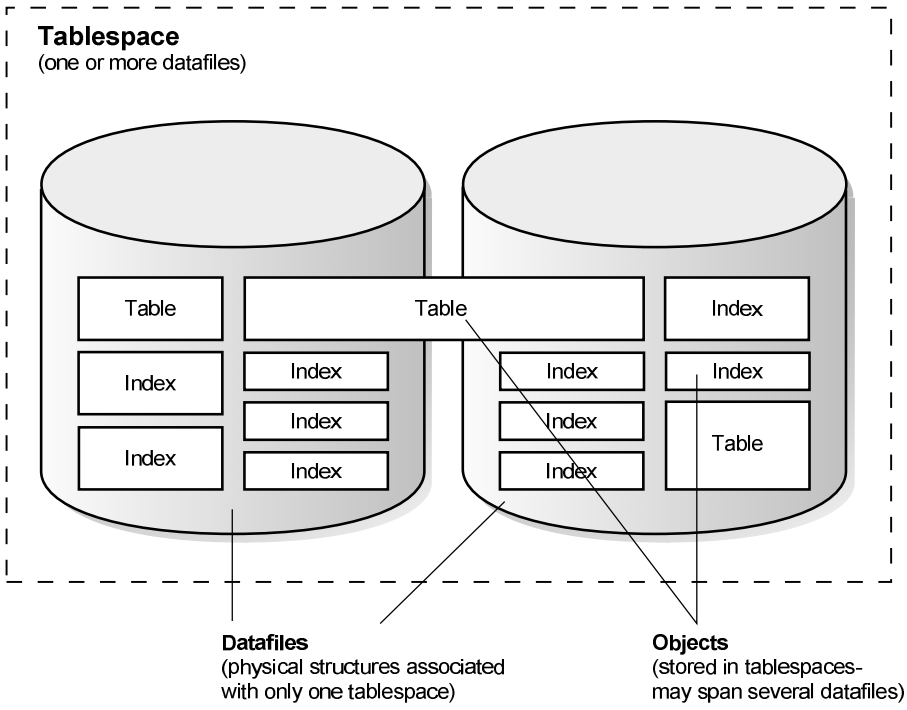
This chapter describes tablespaces, the primary logical database structures of any Oracle database, and the physical datafiles that correspond to each tablespace.

This chapter contains the following topics:

Introduction to Tablespaces, Datafiles, and Control Files

Oracle Database stores data logically in **tablespaces** and physically in **datafiles** associated with the corresponding tablespace. Figure 3–1 illustrates this relationship.

Figure 3–1 Datafiles and Tablespaces



Databases, tablespaces, and datafiles are closely related, but they have important differences:

- An Oracle database consists of at least two logical storage units called tablespaces, which collectively store all of the database's data. You must have the `SYSTEM` and `SYSAUX` tablespaces and a third tablespace, called `TEMP`, is optional.
- Each tablespace in an Oracle database consists of one or more files called datafiles, which are physical structures that conform to the operating system in which Oracle Database is running.
- A database's data is collectively stored in the datafiles that constitute each tablespace of the database. For example, the simplest Oracle database would have one tablespace and one datafile. Another database can have three tablespaces, each consisting of two datafiles (for a total of six datafiles).

This section includes the following topics:

- Oracle-Managed Files
- Allocate More Space for a Database

Oracle-Managed Files

Oracle-managed files eliminate the need for you, the DBA, to directly manage the operating system files comprising an Oracle database. You specify operations in terms of database objects rather than filenames. Oracle Database internally uses standard file system interfaces to create and delete files as needed for the following database structures:

- Tablespaces
- Redo log files
- Control files

Through initialization parameters, you specify the file system directory to be used for a particular type of file. Oracle Database then ensures that a unique file, an Oracle-managed file, is created and deleted when no longer needed.

See Also:

- *Oracle Database Administrator's Guide*
- "Automatic Storage Management" on page 14-14

Allocate More Space for a Database

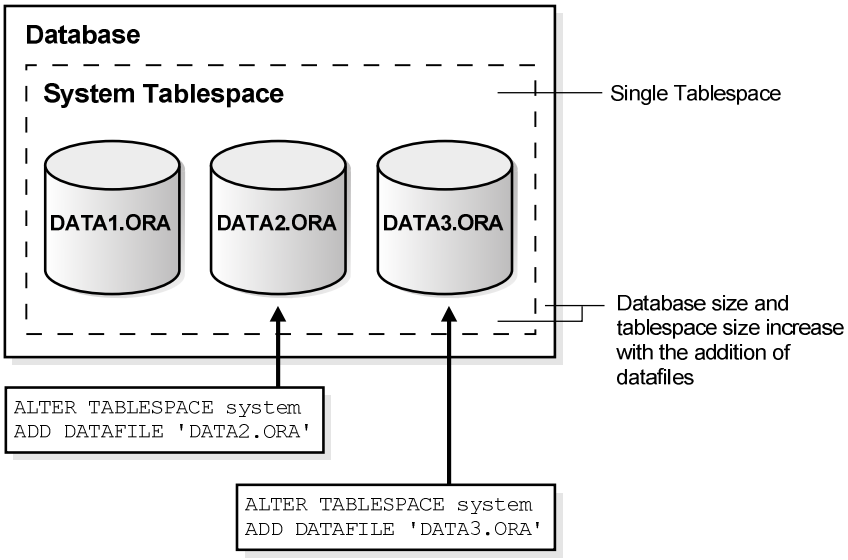
The size of a tablespace is the size of the datafiles that constitute the tablespace. The size of a database is the collective size of the tablespaces that constitute the database.

You can enlarge a database in three ways:

- Add a datafile to a tablespace
- Add a new tablespace
- Increase the size of a datafile

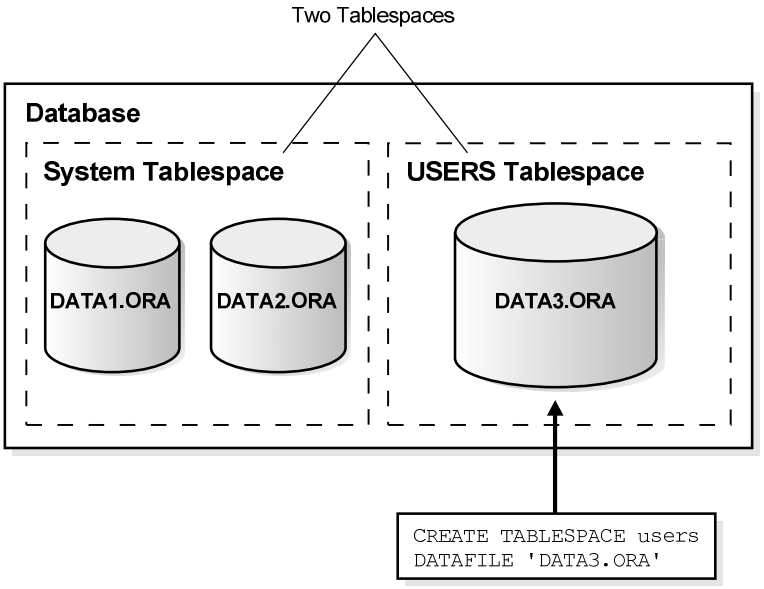
When you add another datafile to an existing tablespace, you increase the amount of disk space allocated for the corresponding tablespace. Figure 3–2 illustrates this kind of space increase.

Figure 3–2 Enlarging a Database by Adding a Datafile to a Tablespace



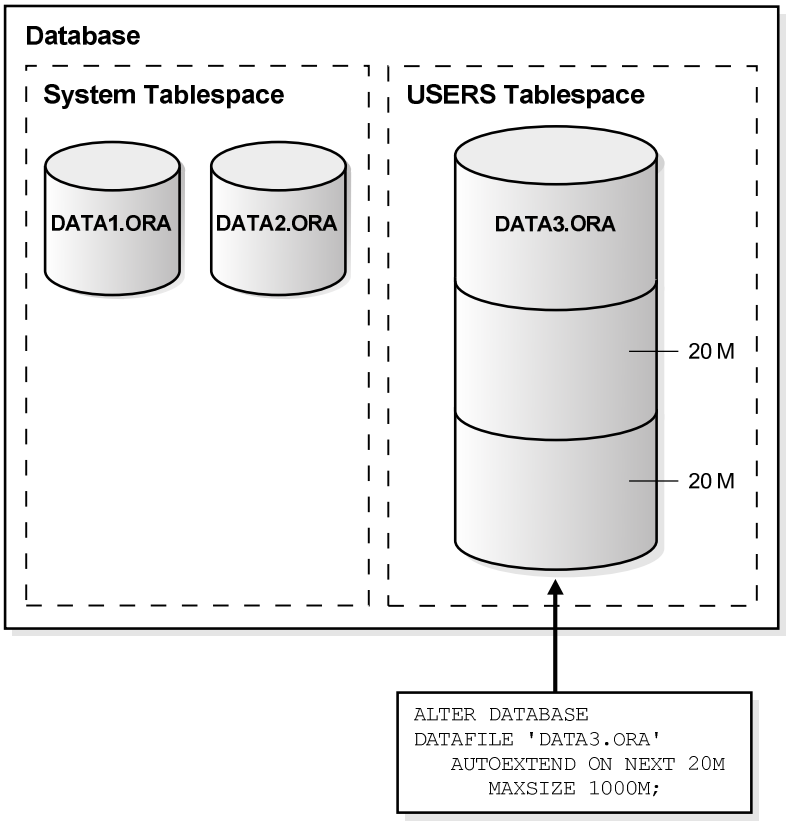
Alternatively, you can create a new tablespace (which contains at least one additional datafile) to increase the size of a database. Figure 3–3 illustrates this.

Figure 3–3 Enlarging a Database by Adding a New Tablespace



The third option for enlarging a database is to change a datafile's size or let datafiles in existing tablespaces grow dynamically as more space is needed. You accomplish this by altering existing files or by adding files with dynamic extension properties. Figure 3–4 illustrates this.

Figure 3–4 Enlarging a Database by Dynamically Sizing Datafiles



See Also: *Oracle Database Administrator's Guide* for more information about increasing the amount of space in your database

Overview of Tablespaces

A database is divided into one or more logical storage units called tablespaces. Tablespaces are divided into logical units of storage called **segments**, which are further divided into **extents**. Extents are a collection of contiguous blocks.

This section includes the following topics about tablespaces:

- Bigfile Tablespaces
- The SYSTEM Tablespace
- The SYSAUX Tablespace
- Undo Tablespaces
- Default Temporary Tablespace
- Using Multiple Tablespaces
- Managing Space in Tablespaces
- Multiple Block Sizes
- Online and Offline Tablespaces
- Read-Only Tablespaces
- Temporary Tablespaces

- Transport of Tablespaces Between Databases

See Also:

- Chapter 2, "Data Blocks, Extents, and Segments" for more information about segments and extents
- *Oracle Database Administrator's Guide* for detailed information on creating and configuring tablespaces

Bigfile Tablespaces

Oracle Database lets you create **bigfile tablespaces**. This allows Oracle Database to contain tablespaces made up of single large files rather than numerous smaller ones. This lets Oracle Database utilize the ability of 64-bit systems to create and manage ultralarge files. The consequence of this is that Oracle Database can now scale up to 8 exabytes in size.

With Oracle-managed files, bigfile tablespaces make datafiles completely transparent for users. In other words, you can perform operations on tablespaces, rather than the underlying datafile. Bigfile tablespaces make the tablespace the main unit of the disk space administration, backup and recovery, and so on. Bigfile tablespaces also simplify datafile management with Oracle-managed files and Automatic Storage Management by eliminating the need for adding new datafiles and dealing with multiple files.

The system default is to create a smallfile tablespace, which is the traditional type of Oracle Database tablespace. The `SYSTEM` and `SYSAUX` tablespace types are always created using the system default type.

Bigfile tablespaces are supported only for locally managed tablespaces with automatic segment-space management. There are two exceptions: locally managed undo and temporary tablespaces can be bigfile tablespaces, even though their segments are manually managed.

An Oracle database can contain both bigfile and smallfile tablespaces. Tablespaces of different types are indistinguishable in terms of execution of SQL statements that do not explicitly refer to datafiles.

You can create a group of temporary tablespaces that let a user consume temporary space from multiple tablespaces. A tablespace group can also be specified as the default temporary tablespace for the database. This is useful with bigfile tablespaces, where you could need a lot of temporary tablespace for sorts.

This section includes the following topics:

- Benefits of Bigfile Tablespaces
- Considerations with Bigfile Tablespaces

Benefits of Bigfile Tablespaces

- Bigfile tablespaces can significantly increase the storage capacity of an Oracle database. Smallfile tablespaces can contain up to 1024 files, but bigfile tablespaces contain only one file that can be 1024 times larger than a smallfile tablespace. The total tablespace capacity is the same for smallfile tablespaces and bigfile tablespaces. However, because there is limit of 64K datafiles for each database, a database can contain 1024 times more bigfile tablespaces than smallfile tablespaces, so bigfile tablespaces increase the total database capacity by 3 orders of magnitude. In other words, 8 exabytes is the maximum size of the Oracle database when bigfile tablespaces are used with the maximum block size (32 k).

- Bigfile tablespaces simplify management of datafiles in ultra large databases by reducing the number of datafiles needed. You can also adjust parameters to reduce the SGA space required for datafile information and the size of the control file.
- They simplify database management by providing datafile transparency.

Considerations with Bigfile Tablespaces

- Bigfile tablespaces are intended to be used with Automatic Storage Management or other logical volume managers that support dynamically extensible logical volumes and striping or RAID.
- Avoid creating bigfile tablespaces on a system that does not support striping because of negative implications for parallel execution and RMAN backup parallelization.
- Avoid using bigfile tablespaces if there could possibly be no free space available on a disk group, and the only way to extend a tablespace is to add a new datafile on a different disk group.
- Using bigfile tablespaces on platforms that do not support large file sizes is not recommended and can limit tablespace capacity. Refer to your operating system specific documentation for information about maximum supported file sizes.
- Performance of database opens, checkpoints, and DBWR processes should improve if data is stored in bigfile tablespaces instead of traditional tablespaces. However, increasing the datafile size might increase time to restore a corrupted file or create a new datafile.

> **See Also:** *Oracle Database Administrator's Guide* for details on creating, altering, and administering bigfile tablespaces

The SYSTEM Tablespace

Every Oracle database contains a tablespace named `SYSTEM`, which Oracle Database creates automatically when the database is created. The `SYSTEM` tablespace is always online when the database is open.

To take advantage of the benefits of locally managed tablespaces, you can create a locally managed `SYSTEM` tablespace, or you can migrate an existing dictionary managed `SYSTEM` tablespace to a locally managed format.

In a database with a locally managed `SYSTEM` tablespace, dictionary managed tablespaces cannot be created. It is possible to plug in a dictionary managed tablespace using the transportable feature, but it cannot be made writable.

This section includes the following topics:

The Data Dictionary

The `SYSTEM` tablespace always contains the data dictionary tables for the entire database.

PL/SQL Program Units Description

All data stored on behalf of stored PL/SQL program units (that is, procedures, functions, packages, and triggers) resides in the `SYSTEM` tablespace. If the database

contains many of these program units, then the database administrator must provide the space the units need in the SYSTEM tablespace.

See Also:

- *Oracle Database Administrator's Guide* for information about creating or migrating to a locally managed SYSTEM tablespace
- "Online and Offline Tablespaces" on page 3-11 for information about the permanent online condition of the SYSTEM tablespace
- Chapter 24, "SQL" and Chapter 22, "Triggers" for information about the space requirements of PL/SQL program units

The SYSAUX Tablespace

The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace. Many database components use the SYSAUX tablespace as their default location to store data. Therefore, the SYSAUX tablespace is always created during database creation or database upgrade.

Note: If the SYSAUX tablespace is unavailable, such as due to a media failure, then some database features may fail.

The SYSAUX tablespace provides a centralized location for database metadata that does not reside in the SYSTEM tablespace. It reduces the number of tablespaces created by default, both in the seed database and in user-defined databases.

During normal database operation, Oracle Database does not allow the SYSAUX tablespace to be dropped or renamed. Transportable tablespaces for SYSAUX is not supported.

See Also: *Oracle Database Administrator's Guide* to learn about database components that use the SYSAUX tablespace

Undo Tablespaces

Undo tablespaces are special tablespaces used solely for storing undo information. You cannot create any other segment types (for example, tables or indexes) in undo tablespaces. Undo tablespaces are used only when the database is in automatic undo management mode (the default). A database can contain more than one undo tablespace, but only one can be in use at any time. Undo data is managed within an undo tablespace using undo segments that are automatically created and maintained by the database.

When the first DML operation is run within a transaction, the transaction is bound (assigned) to an undo segment (and therefore to a transaction table) in the current undo tablespace. In rare circumstances, if the instance does not have a designated undo tablespace, the transaction binds to the system undo segment.

Each undo tablespace is composed of a set of datafiles and is locally managed. Like other types of tablespaces, undo blocks are grouped in extents and the status of each extent is represented in the bitmap. At any point in time, an extent is either allocated to (and used by) a transaction table, or it is free.

You can create a bigfile undo tablespace.

See Also:

- *Oracle Database Administrator's Guide* for information on managing the undo tablespace
- "Bigfile Tablespaces" on page 3-5

Creation of Undo Tablespaces

An undo tablespace is automatically created with each new installation of Oracle Database. Earlier versions of Oracle Database may not include an undo tablespace and may instead use rollback segments. This is known as manual undo management mode. When upgrading to Oracle Database 11*g* you can migrate to automatic undo management by creating an undo tablespace and enabling automatic undo management mode. See *Oracle Database Upgrade Guide* for details.

Default Temporary Tablespace

When the `SYSTEM` tablespace is locally managed, you must define at least one default temporary tablespace when creating a database. A locally managed `SYSTEM` tablespace cannot be used for default temporary storage.

If `SYSTEM` is dictionary managed and if you do not define a default temporary tablespace when creating the database, then `SYSTEM` is still used for default temporary storage. However, you will receive a warning in `ALERT.LOG` saying that a default temporary tablespace is recommended and will be necessary in future releases.

How to Specify a Default Temporary Tablespace

Specify default temporary tablespaces when you create a database, using the `DEFAULT TEMPORARY TABLESPACE` extension to the `CREATE DATABASE` statement.

You can create bigfile temporary tablespaces. A bigfile temporary tablespace, like all temporary tablespaces, uses tempfiles instead of datafiles.

Note: You cannot make a default temporary tablespace permanent or take it offline.

See Also:

- *Oracle Database SQL Language Reference* for information about defining and altering default temporary tablespaces
- "Bigfile Tablespaces" on page 3-5

Using Multiple Tablespaces

A very small database may need only the `SYSTEM` tablespace; however, Oracle recommends that you create at least one additional tablespace to store user data separate from data dictionary information. This gives you more flexibility in various database administration operations and reduces contention among dictionary objects and schema objects for the same datafiles.

You can use multiple tablespaces to perform the following tasks:

- Control disk space allocation for database data
- Assign specific space quotas for database users
- Control availability of data by taking individual tablespaces online or offline

- Perform partial database backup or recovery operations
- Allocate data storage across devices to improve performance

A database administrator can perform the following actions:

- Create new tablespaces
- Add datafiles to tablespaces
- Set and alter default segment storage settings for segments created in a tablespace
- Make a tablespace read only or read/write
- Make a tablespace temporary or permanent
- Rename tablespaces
- Drop tablespaces
- Transport tablespaces across databases and platforms

Managing Space in Tablespaces

Tablespaces allocate space in extents. Tablespaces can use two different methods to keep track of their free and used space:

- **Locally managed tablespaces**: Extent management by the bitmaps
- **Dictionary managed tablespaces**: Extent management by the data dictionary

When you create a tablespace, you choose one of these methods of space management. Later, you can change the management method with the `DBMS_SPACE_ADMIN` PL/SQL package.

This section includes the following topics:

- Locally Managed Tablespaces
- Segment Space Management in Locally Managed Tablespaces
- Dictionary Managed Tablespaces

See Also: "Overview of Extents" on page 2-10

Locally Managed Tablespaces

A tablespace that manages its own extents maintains a bitmap in each datafile to keep track of the free or used status of blocks in that datafile. Each bit in the bitmap corresponds to a block or a group of blocks. When an extent is allocated or freed for reuse, Oracle Database changes the bitmap values to show the new status of the blocks.

Locally managed tablespaces have the following advantages over dictionary managed tablespaces:

- Local management of extents automatically tracks adjacent free space, eliminating the need to coalesce free extents.
- Local management of extents avoids recursive space management operations. Such recursive operations can occur in dictionary managed tablespaces if consuming or releasing space in an extent results in another operation that consumes or releases space in a data dictionary table or rollback segment.

The sizes of extents that are managed locally are determined automatically by the system. Alternatively, all extents can have the same size in a locally managed tablespace and override object storage options.

The `LOCAL` clause of the `CREATE TABLESPACE` or `CREATE TEMPORARY TABLESPACE` statement is specified to create locally managed permanent or temporary tablespaces, respectively.

Segment Space Management in Locally Managed Tablespaces

When you create a locally managed tablespace using the `CREATE TABLESPACE` statement, the `SEGMENT SPACE MANAGEMENT` clause lets you specify how free and used space within a segment is to be managed. Your choices are:

- `AUTO`

  This keyword tells Oracle Database that you want to use bitmaps to manage the free space within segments. A bitmap, in this case, is a map that describes the status of each data block within a segment with respect to the amount of space in the block available for inserting rows. As more or less space becomes available in a data block, its new state is reflected in the bitmap. Bitmaps enable Oracle Database to manage free space more automatically; thus, this form of space management is called automatic segment-space management.

  Locally managed tablespaces using automatic segment-space management can be created as smallfile (traditional) or bigfile tablespaces. `AUTO` is the default.

- `MANUAL`

  This keyword tells Oracle Database that you want to use free lists for managing free space within segments. Free lists are lists of data blocks that have space available for inserting rows.

  See Also:

  - *Oracle Database SQL Language Reference* for syntax
  - *Oracle Database Administrator's Guide* for more information about automatic segment space management
  - "Determine the Number and Size of Extents" on page 2-10
  - "Temporary Tablespaces" on page 3-12 for more information about temporary tablespaces

Dictionary Managed Tablespaces

If you created your database with Oracle9*i*, you could be using dictionary managed tablespaces. For a tablespace that uses the data dictionary to manage its extents, Oracle Database updates the appropriate tables in the data dictionary whenever an extent is allocated or freed for reuse. Oracle Database also stores rollback information about each update of the dictionary tables. Because dictionary tables and rollback segments are part of the database, the space that they occupy is subject to the same space management operations as all other data.

Note: If you do not specify extent management when you create a tablespace, then the default is locally managed.

Multiple Block Sizes

Oracle Database supports multiple block sizes in a database. The **standard block size** is used for the SYSTEM tablespace. This is set when the database is created and can be any valid size. You specify the standard block size by setting the initialization parameter DB_BLOCK_SIZE. Legitimate values are from 2K to 32K.

In the initialization parameter file or server parameter file, you can configure subcaches within the buffer cache for each of these block sizes. Subcaches can also be configured while an instance is running. You can create tablespaces having any of these block sizes. The standard block size is used for the system tablespace and most other tablespaces.

Note: All partitions of a partitioned object must reside in tablespaces of a single block size.

Multiple block sizes are useful primarily when transporting a tablespace from an OLTP database to an enterprise data warehouse. This facilitates transport between databases of different block sizes.

See Also:

- "Transport of Tablespaces Between Databases" on page 3-13
- *Oracle Database Data Warehousing Guide* for information about transporting tablespaces in data warehousing environments

Online and Offline Tablespaces

A database administrator can bring any tablespace other than the SYSTEM tablespace **online** (accessible) or **offline** (not accessible) whenever the database is open. The SYSTEM tablespace is always online when the database is open because the data dictionary must always be available to Oracle Database.

A tablespace is usually online so that the data contained within it is available to database users. However, the database administrator can take a tablespace offline for maintenance or backup and recovery purposes.

Bringing Tablespaces Offline

When a tablespace goes offline, Oracle Database does not permit any subsequent SQL statements to reference objects contained in that tablespace. Active transactions with completed statements that refer to data in that tablespace are not affected at the transaction level. Oracle Database saves rollback data corresponding to those completed statements in a deferred rollback segment in the SYSTEM tablespace. When the tablespace is brought back online, Oracle Database applies the rollback data to the tablespace, if needed.

When a tablespace goes offline or comes back online, this is recorded in the data dictionary in the SYSTEM tablespace. If a tablespace is offline when you shut down a database, the tablespace remains offline when the database is subsequently mounted and reopened.

You can bring a tablespace online only in the database in which it was created because the necessary data dictionary information is maintained in the SYSTEM tablespace of that database. An offline tablespace cannot be read or edited by any utility other than Oracle Database. Thus, offline tablespaces cannot be transposed to other databases.

Oracle Database automatically switches a tablespace from online to offline when certain errors are encountered. For example, Oracle Database switches a tablespace from online to offline when the database writer process, DBW*n*, fails in several attempts to write to a datafile of the tablespace. Users trying to access tables in the offline tablespace receive an error. If the problem that causes this disk I/O to fail is media failure, you must recover the tablespace after you correct the problem.

See Also:

- "Transport of Tablespaces Between Databases" on page 3-13 for more information about transferring online tablespaces between databases
- *Oracle Database Utilities* for more information about tools for data transfer

Read-Only Tablespaces

The primary purpose of read-only tablespaces is to eliminate the need to perform backup and recovery of large, static portions of a database. Oracle Database never updates the files of a read-only tablespace, and therefore the files can reside on read-only media such as CD-ROMs or WORM drives.

Note: Because you can only bring a tablespace online in the database in which it was created, read-only tablespaces are not meant to satisfy archiving requirements.

Read-only tablespaces cannot be modified. To update a read-only tablespace, first make the tablespace read/write. After updating the tablespace, you can then reset it to be read only.

Because read-only tablespaces cannot be modified, and as long as they have not been made read/write at any point, they do not need repeated backup. Also, if you must recover your database, you do not need to recover any read-only tablespaces, because they could not have been modified.

See Also:

- *Oracle Database Administrator's Guide* for information about changing a tablespace to read only or read/write mode
- *Oracle Database SQL Language Reference* for more information about the `ALTER TABLESPACE` statement
- *Oracle Database Backup and Recovery User's Guide* for more information about recovery

Temporary Tablespaces

You can manage space for sort operations more efficiently by designating one or more temporary tablespaces exclusively for sorts. Doing so effectively eliminates serialization of space management operations involved in the allocation and deallocation of sort space. A single SQL operation can use more than one temporary tablespace for sorting. For example, you can create indexes on very large tables, and the sort operation during index creation can be distributed across multiple tablespaces.

All operations that use sorts, including joins, index builds, ordering, computing aggregates (`GROUP BY`), and collecting optimizer statistics, benefit from temporary tablespaces. The performance gains are significant with Oracle Real Application Clusters.

This section includes the following topics:

- Sort Segments
- Creation of Temporary Tablespaces

Sort Segments

One or more temporary tablespaces can be used only for sort segments. A temporary tablespace is not the same as a tablespace that a user designates for temporary segments, which can be any tablespace available to the user. No permanent schema objects can reside in a temporary tablespace.

Sort segments are used when a segment is shared by multiple sort operations. One sort segment exists for every instance that performs a sort operation in a given tablespace.

Temporary tablespaces provide performance improvements when you have multiple sorts that are too large to fit into memory. The sort segment of a given temporary tablespace is created at the time of the first sort operation. The sort segment expands by allocating extents until the segment size is equal to or greater than the total storage demands of all of the active sorts running on that instance.

> **See Also:** Chapter 2, "Data Blocks, Extents, and Segments" for more information about segments

Creation of Temporary Tablespaces

Create temporary tablespaces by using the `CREATE TABLESPACE` or `CREATE TEMPORARY TABLESPACE` statement.

> **See Also:**
>
> - "Temporary Datafiles" on page 3-16 for information about `TEMPFILES`
> - "Managing Space in Tablespaces" on page 3-9 for information about locally managed and dictionary managed tablespaces
> - *Oracle Database SQL Language Reference* for syntax
> - *Oracle Database Performance Tuning Guide* for information about setting up temporary tablespaces for sorts and hash joins

Transport of Tablespaces Between Databases

A **transportable tablespace** lets you move a subset of an Oracle database from one Oracle database to another, even across different platforms. You can clone a tablespace and plug it into another database, copying the tablespace between databases, or you can unplug a tablespace from one Oracle database and plug it into another Oracle database, moving the tablespace between databases.

Moving data by transporting tablespaces can be orders of magnitude faster than either export/import or unload/load of the same data, because transporting a tablespace involves only copying datafiles and integrating the tablespace metadata. When you transport tablespaces you can also move index data, so you do not have to rebuild the indexes after importing or loading the table data.

You can transport tablespaces across platforms. (Many, but not all, platforms are supported for cross-platform tablespace transport.) This can be used for the following:

- Provide an easier and more efficient means for content providers to publish structured data and distribute it to customers running Oracle Database on a different platform
- Simplify the distribution of data from a data warehouse environment to data marts which are often running on smaller platforms
- Enable the sharing of read only tablespaces across a heterogeneous cluster
- Allow a database to be migrated from one platform to another

This section includes the following topics:

- Tablespace Repository
- How to Move or Copy a Tablespace to Another Database

Tablespace Repository

A tablespace repository is a collection of tablespace sets. Tablespace repositories are built on file group repositories, but tablespace repositories only contain the files required to move or copy tablespaces between databases. Different tablespace sets may be stored in a tablespace repository, and different versions of a particular tablespace set also may be stored. A version of a tablespace set in a tablespace repository consists of the following files:

- The Data Pump export dump file for the tablespace set
- The Data Pump log file for the export
- The datafiles that comprise the tablespace set

See Also: *Oracle Streams Concepts and Administration*

How to Move or Copy a Tablespace to Another Database

To move or copy a set of tablespaces, you must make the tablespaces read only, copy the datafiles of these tablespaces, and use export/import to move the database information (metadata) stored in the data dictionary. Both the datafiles and the metadata export file must be copied to the target database. The transport of these files can be done using any facility for copying flat files, such as the operating system copying facility, ftp, or publishing on CDs.

After copying the datafiles and importing the metadata, you can optionally put the tablespaces in read/write mode.

The first time a tablespace's datafiles are opened under Oracle Database with the `COMPATIBLE` initialization parameter set to 10 or higher, each file identifies the platform to which it belongs. These files have identical on disk formats for file header blocks, which are used for file identification and verification. Read only and offline files get the compatibility advanced after they are made read/write or are brought online. This implies that tablespaces that are read only before Oracle Database 10*g* must be made read/write at least once before they can use the cross platform transportable feature.

Note: In a database with a locally managed SYSTEM tablespace, dictionary tablespaces cannot be created. It is possible to plug in a dictionary managed tablespace using the transportable feature, but it cannot be made writable.

See Also:

- *Oracle Database Administrator's Guide* for details about how to move or copy tablespaces to another database, including details about transporting tablespaces across platforms
- *Oracle Database Utilities* for import/export information
- *Oracle Database PL/SQL Packages and Types Reference* for information on the DBMS_FILE_TRANSFER package
- *Oracle Streams Concepts and Administration* for more information on ways to copy or transport files

Overview of Datafiles

A tablespace in an Oracle database consists of one or more physical **datafiles**. A datafile can be associated with only one tablespace and only one database.

Oracle Database creates a datafile for a tablespace by allocating the specified amount of disk space plus the overhead required for the file header. When a datafile is created, the operating system under which Oracle Database runs is responsible for clearing old information and authorizations from a file before allocating it to Oracle Database. If the file is large, this process can take a significant amount of time. The first tablespace in any database is always the SYSTEM tablespace, so Oracle Database automatically allocates the first datafiles of any database for the SYSTEM tablespace during database creation.

This section includes the following topics:

- Datafile Contents
- Size of Datafiles
- Offline Datafiles
- Temporary Datafiles

See Also: Your Oracle Database operating system-specific documentation for information about the amount of space required for the file header of datafiles on your operating system

Datafile Contents

When a datafile is first created, the allocated disk space is formatted but does not contain any user data. However, Oracle Database reserves the space to hold the data for future segments of the associated tablespace—it is used exclusively by Oracle Database. As the data grows in a tablespace, Oracle Database uses the free space in the associated datafiles to allocate extents for the segment.

The data associated with schema objects in a tablespace is physically stored in one or more of the datafiles that constitute the tablespace. Note that a schema object does not correspond to a specific datafile; rather, a datafile is a repository for the data of any schema object within a specific tablespace. Oracle Database allocates space for the data

associated with a schema object in one or more datafiles of a tablespace. Therefore, a schema object can span one or more datafiles. Unless table **striping** is used (where data is spread across more than one disk), the database administrator and end users cannot control which datafile stores a schema object.

> **See Also:** Chapter 2, "Data Blocks, Extents, and Segments" for more information about use of space

Size of Datafiles

You can alter the size of a datafile after its creation or you can specify that a datafile should dynamically grow as schema objects in the tablespace grow. This functionality enables you to have fewer datafiles for each tablespace and can simplify administration of datafiles.

> **Note:** You need sufficient space on the operating system for expansion.

> **See Also:** *Oracle Database Administrator's Guide* for more information about resizing datafiles

Offline Datafiles

You can take tablespaces offline or bring them online at any time, except for the `SYSTEM` tablespace. All of the datafiles of a tablespace are taken offline or brought online as a unit when you take the tablespace offline or bring it online, respectively.

You can take individual datafiles offline. However, this is usually done only during some database recovery procedures.

Temporary Datafiles

Locally managed temporary tablespaces have temporary datafiles (**tempfile**s), which are similar to ordinary datafiles, with the following exceptions:

- Tempfiles are always set to `NOLOGGING` mode.
- You cannot make a tempfile read only.
- You cannot create a tempfile with the `ALTER DATABASE` statement.
- Media recovery does not recognize tempfiles:
  - `BACKUP CONTROLFILE` does not generate any information for tempfiles.
  - `CREATE CONTROLFILE` cannot specify any information about tempfiles.
- When you create or resize tempfiles, they are not always guaranteed allocation of disk space for the file size specified. On certain file systems (for example, UNIX) disk blocks are allocated not at file creation or resizing, but before the blocks are accessed.

> **Caution:** This enables fast tempfile creation and resizing; however, the disk could run out of space later when the tempfiles are accessed.

- Tempfile information is shown in the 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: "Managing Space in Tablespaces" on page 3-9 for more information about locally managed tablespaces

Overview of Control Files

The database control file is a small binary file necessary for the database to start and operate successfully. A control file is updated continuously by Oracle Database during database use, so it must be available for writing whenever the database is open. If for some reason the control file is not accessible, then the database cannot function properly.

Each control file is associated with only one Oracle database.

This section includes the following topics:

- Control File Contents
- Multiplexed Control Files

Control File Contents

A control file contains information about the associated database that is required for access by an instance, both at startup and during normal operation. Control file information can be modified only by Oracle Database; no database administrator or user can edit a control file.

Among other things, a control file contains information such as:

- The database name
- The timestamp of database creation
- The names and locations of associated datafiles and redo log files
- Tablespace information
- Datafile offline ranges
- The log history
- Archived log information
- Backup set and backup piece information
- Backup datafile and redo log information
- Datafile copy information
- The current log sequence number
- Checkpoint information

The database name and timestamp originate at database creation. The database name is taken from either the name specified by the DB_NAME initialization parameter or the name used in the CREATE DATABASE statement.

Each time that a datafile or a redo log file is added to, renamed in, or dropped from the database, the control file is updated to reflect this physical structure change. These changes are recorded so that:

- Oracle Database can identify the datafiles and redo log files to open during database startup
- Oracle Database can identify files that are required or available in case database recovery is necessary

Therefore, if you make a change to the physical structure of your database (using `ALTER DATABASE` statements), then you should immediately make a backup of your control file.

Control files also record information about checkpoints. Every three seconds, the checkpoint process (CKPT) records information in the control file about the checkpoint position in the redo log. This information is used during database recovery to tell Oracle Database that all redo entries recorded before this point in the redo log group are not necessary for database recovery; they were already written to the datafiles.

> **See Also:** *Oracle Database Backup and Recovery User's Guide* for information about backing up a database's control file

Multiplexed Control Files

As with redo log files, Oracle Database enables multiple, identical control files to be open concurrently and written for the same database. By storing multiple control files for a single database on different disks, you can safeguard against a single point of failure with respect to control files. If a single disk that contained a control file crashes, then the current instance fails when Oracle Database attempts to access the damaged control file. However, when other copies of the current control file are available on different disks, an instance can be restarted without the need for database recovery.

If *all* control files of a database are permanently lost during operation, then the instance is aborted 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. It is strongly recommended that you adhere to the following:

- Use multiplexed control files with each database
- Store each copy on a different physical disk
- Use operating system mirroring
- Monitor backups