11

Managing Space

Objectives

After completing this lesson, you should be able to:

- Describe how the Oracle Database server automatically manages space
- Save space by using compression
- Proactively monitor and manage tablespace space usage
- Describe segment creation in the Oracle database
- Control deferred segment creation
- Use the Segment Advisor
- Reclaim wasted space from tables and indexes by using the segment shrink functionality
- Manage resumable space allocation

Space Management: Overview

Space is automatically managed by the Oracle Database server. It generates alerts about potential problems and recommends possible solutions. Features include:

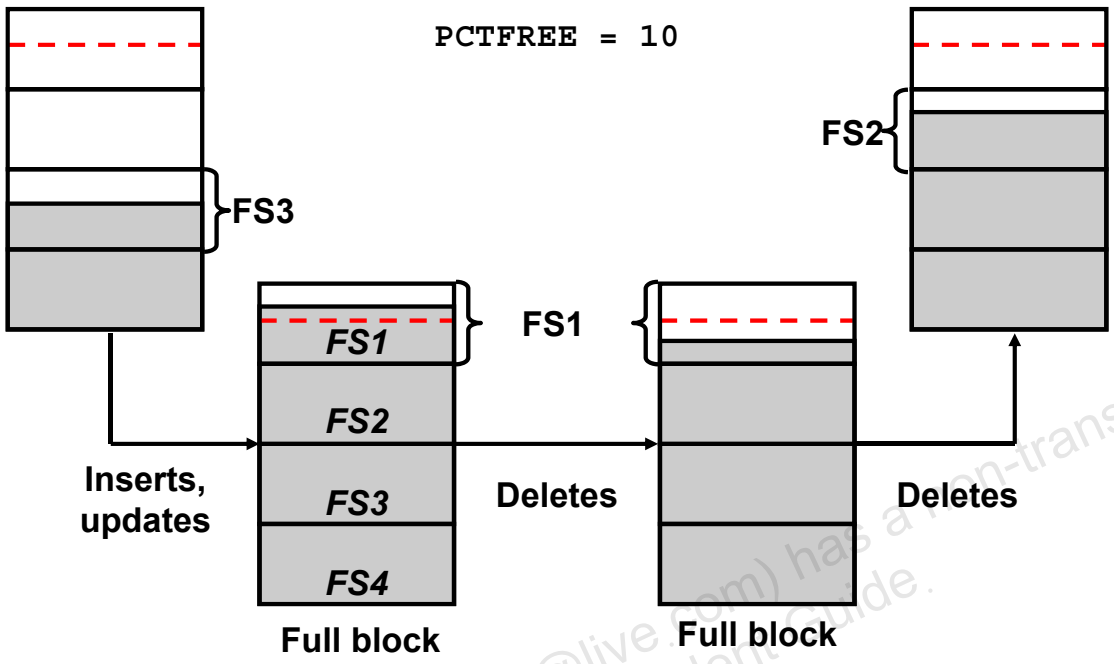
- Oracle Managed Files (OMF)
- Free-space management with bitmaps (“locally managed”) and automatic data file extension
- Proactive space management (default thresholds and server-generated alerts)
- Space reclamation (shrinking segments, online table redefinition)
- Capacity planning (growth reports)

With Oracle Managed Files (OMF), you can specify operations in terms of database objects rather than file names. The Oracle Database server can manage free space within a tablespace with bitmaps. This is known as a “locally managed” tablespace. In addition, free space within segments located in locally managed tablespaces can be managed using bitmaps. This is known as Automatic Segment Space Management. The bitmapped implementation eliminates much space-related tuning of tables, while providing improved performance during peak loads. Additionally, the Oracle Database server provides automatic extension of data files, so the files can grow automatically based on the amount of data in the files.

When you create a database, proactive space monitoring is enabled by default. (This causes no performance impact.) The Oracle Database server monitors space utilization during normal space allocation and deallocation operations and alerts you if the free space availability falls below the predefined thresholds (which you can override). Advisors and wizards assist you with space reclamation.

For capacity planning, the Oracle Database server provides space estimates based on table structure and number of rows and a growth trend report based on historical space utilization stored in the Automatic Workload Repository (AWR).

Block Space Management



Space management involves the management of free space at the block level. With Automatic Segment Space Management, each block is divided into four sections, named FS1 (between 0 and 25% of free space), FS2 (25% to 50% free), FS3 (50% to 75% free), and FS4 (75% to 100% free).

Depending on the level of free space in the block, its status is automatically updated. That way, depending on the length of an inserted row, you can tell whether a particular block can be used to satisfy an insert operation. Note that a “full” status means that a block is no longer available for inserts.

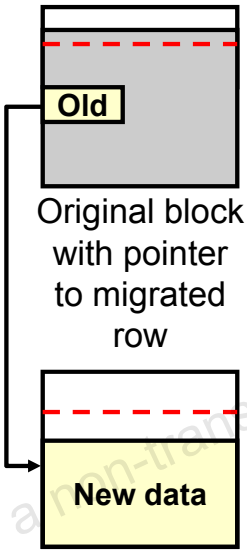
In the slide example, the block on the left is an FS3 block because it has between 50% and 75% free space. After some insert and update statements, `PCTFREE` is reached (the dashed line) and it is no longer possible to insert new rows in that block. The block is now considered as a “full” or FS1 block. The block is considered for insertion again, as soon as its free space level drops below the next section. In the preceding case, it gets status FS2 as soon as the free space is more than 25%.

Note: Large object (LOB) data types (`BLOB`, `CLOB`, `NCLOB`, and `BFILE`) do not use the `PCTFREE` storage parameter. Uncompressed and OLTP-compressed blocks have a default `PCTFREE` value of `10`; basic compressed blocks have a default `PCTFREE` value of `0`.

Row Chaining and Migration

Example:

- On update: Row length increases, exceeding the available free space in the block.
- Data needs to be stored in a new block.
- Original physical identifier of row (ROWID) is preserved.
- The Oracle Database server needs to read two blocks to retrieve data.
- The Segment Advisor finds segments containing the migrated rows.
- There is automatic coalescing of fragmented free space inside the block.



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, the Oracle Database server 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 data type 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, the Oracle Database server migrates the data for the entire row to a new data block, assuming that the entire row can fit in a new block. The 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, input/output (I/O) performance associated with this row decreases because the Oracle Database server must scan more than one data block to retrieve the information for the row.

The Segment Advisor finds the segments containing migrated rows that result from an UPDATE.

The Oracle Database server automatically and transparently coalesces the free space of a data block when:

- An `INSERT` or `UPDATE` statement attempts to use a block with sufficient free space for a new row piece
- The free space is fragmented, so that the row piece cannot be inserted in a contiguous section of the block

After coalescing, the amount of free space is identical to the amount before the operation, but the space is now contiguous.

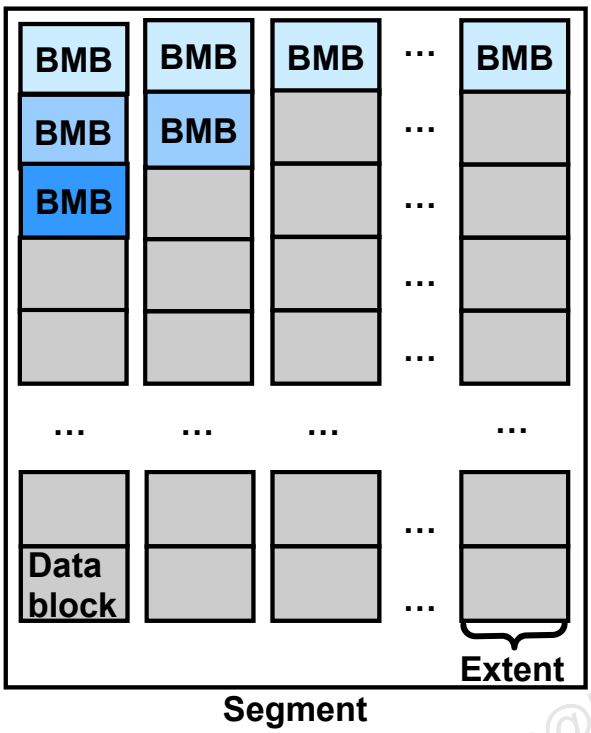
Quiz

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

a. True
b. False

Answer: a

Free Space Management Within Segments



- Tracked by bitmaps in segments

Benefits:

- More flexible space utilization
- Runtime adjustment
- Multiple process search of BMBs

Free space can be managed automatically inside database segments. The in-segment free or used space is tracked with bitmaps. To take advantage of this feature, specify Automatic Segment Space Management when you create a locally managed tablespace. Your specification then applies to all segments subsequently created in this tablespace.

Automatic space management segments have a set of bitmap blocks (BMBs) describing the space utilization of the data blocks in that segment. BMBs are organized in a tree hierarchy. The root level of the hierarchy, which contains references to all intermediate BMBs, is stored in the segment header. The leaves of this hierarchy represent the space information for a set of contiguous data blocks that belong to the segment. The maximum number of levels inside this hierarchy is three.

Benefits of using automatic space management include:

- Better space utilization, especially for the objects with highly varying row sizes
- Better runtime adjustment to variations in concurrent access
- Better multi-instance behavior in terms of performance or space utilization

Types of Segments

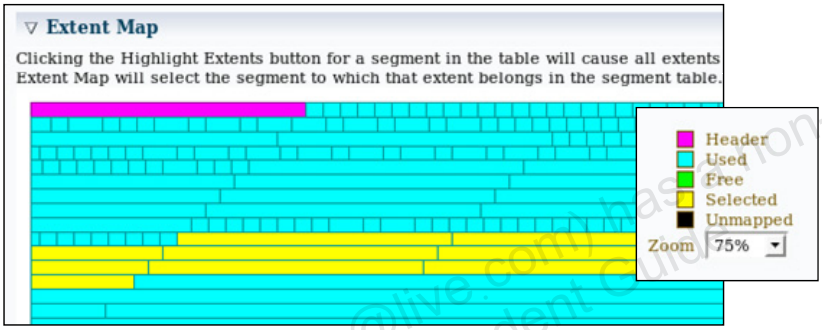
- A segment is a set of extents allocated for a certain logical structure. The different types of segments include:
  - Table and cluster segments
  - Index segment
  - Undo segment
  - Temporary segment
- Segments are dynamically allocated by the Oracle Database server.

- **Table and cluster segments:** Each nonclustered table has a data segment. All table data is stored in the extents of the table segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
- **Index segment:** Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
- **Undo segment:** 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.
- **Temporary segment:** A temporary segment is created by the Oracle Database server when a SQL statement needs a temporary database area to complete execution. When the statement finishes execution, the extents in the temporary segment are returned to the system for future use.

The Oracle Database server dynamically allocates space when the existing extents of a segment become full. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

Allocating Extents

- Searching the data file's bitmap for the required number of adjacent free blocks
- Sizing extents with storage clauses:
  - `UNIFORM`
  - `AUTOALLOCATE`
- Viewing extent map
- Obtaining deallocation advice



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

Two clauses affect the sizing of extents:

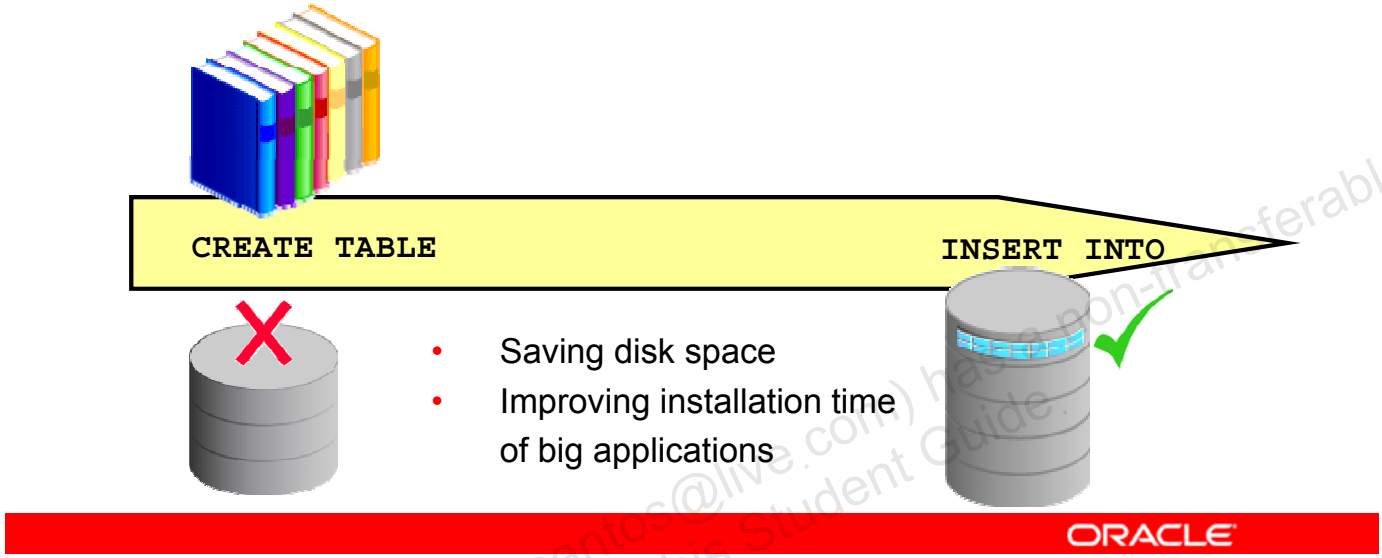
- With the `UNIFORM` clause, the database creates all extents of a uniform size that you specified (or a default size) for any objects created in the tablespace.
- With the `AUTOALLOCATE` clause, the database determines the extent-sizing policy for the tablespace.

To view the extent map in Enterprise Manager Cloud Control, choose Administration > Storage > Tablespaces. Select the tablespace and click View. Select Show Tablespace Contents in the Action menu and click Go. Expand Extent Map.

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

Understanding Deferred Segment Creation

- `DEFERRED_SEGMENT_CREATION = TRUE` is the default.
- Segment creation takes place as follows:
  1. Table creation > Data dictionary operation
  2. DML > Segment creation



When you create a nonpartitioned heap table, table segment creation is deferred to the first row insert. This functionality is enabled by default with the `DEFERRED_SEGMENT_CREATION` initialization parameter set to `TRUE`.

Advantages of this space allocation method:

- A significant amount of disk space can be saved for applications that create hundreds or thousands of tables upon installation, many of which might never be populated.
- The application installation time is reduced.

When you insert the first row into the table, the segments are created for the base table, its LOB columns, and its indexes. During segment creation, cursors on the table are invalidated. These operations have a small additional impact on performance.

Note: With this allocation method, it is essential that you do proper capacity planning so that the database has enough disk space to handle segment creation when tables are populated. For more details, see the *Oracle Database Administrator's Guide*.

Viewing Deferred Segment Information

```
SQL> SHOW PARAMETERS deferred_segment_creation
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------
deferred_segment_creation            boolean     TRUE

SQL> CREATE TABLE seg_test(c number, d varchar2(500));
Table created.
SQL> SELECT segment_name FROM user_segments;
no rows selected
```

Inserting rows and creating segments:

```
SQL> INSERT INTO seg_test VALUES(1, 'aaaaaaa');
1 row created.

SQL> SELECT segment_name FROM user_segments;
SEGMENT_NAME
--------------------------------------------------------------------------------
SEG_TEST
```


This slide shows you how to check the DEFERRED_SEGMENT_CREATION parameter. Then a table is created without segments, which you can verify by querying the USER_SEGMENTS data dictionary view. After the insert of a row, you query this view again to see that the segment now exists.

You can also query the SEGMENT_CREATED column of the USER_TABLES, USER_INDEXES, or USER_LOBS views. For nonpartitioned tables, indexes, and LOBs, this column shows YES if the segment is created.

The SYS.SEG$ data dictionary table stores the storage parameters that you specified during table or index creation.

Controlling Deferred Segment Creation

With the DEFERRED_SEGMENT_CREATION parameter in the:

- Initialization file
- ALTER SESSION command
- ALTER SYSTEM command

With the SEGMENT CREATION clause:

- IMMEDIATE
- DEFERRED (default)

```
CREATE TABLE SEG_TAB3(C1 number, C2 number)
  SEGMENT CREATION IMMEDIATE TABLESPACE SEG_TBS;
CREATE TABLE SEG_TAB4(C1 number, C2 number)
  SEGMENT CREATION DEFERRED;
```

Note: Indexes inherit table characteristics.

Segment creation can be controlled in two ways:

- With the DEFERRED_SEGMENT_CREATION initialization parameter set to TRUE or FALSE. This parameter can be set in the initialization file. You can also control it via the ALTER SESSION or ALTER SYSTEM commands.

  Examples:

  ```
  ALTER SESSION SET DEFERRED_SEGMENT_CREATION = TRUE;
  ALTER SYSTEM SET DEFERRED_SEGMENT_CREATION = FALSE;
  ```

- With the SEGMENT CREATION clause of the CREATE TABLE command:
  - SEGMENT CREATION DEFERRED: If specified, segment creation is deferred until the first row is inserted into the table. This is the default behavior for Oracle Database 11*g* Release 2 and later releases.
  - SEGMENT CREATION IMMEDIATE: If specified, segments are materialized during table creation. This is the default behavior in Oracle databases before Oracle Database 11*g* Release 2.

This clause takes precedence over the DEFERRED_SEGMENT_CREATION parameter.

It is possible to force the creation of segments for an already created table with the ALTER TABLE … MOVE command.

However, it is not possible to directly control the deferred segment creation for dependent objects such as indexes. They inherit this characteristic from their parent object—in this case, the table.

Restrictions and Exceptions

- Segment creation on demand is:
  - Only for nonpartitioned tables and indexes
  - Not for IOTs, clustered tables, or other special tables
  - Not for tables in dictionary-managed tablespaces
- If you were to migrate a table without segments from a locally managed to a dictionary-managed tablespace, you must drop and re-create it.

Deferred segment creation is restricted to nonpartitioned tables and nonpartitioned indexes.

Segment creation on demand is not supported for IOTs, clustered tables, global temp tables, session-specific temp tables, internal tables, typed tables, AQ tables, `SYS`-owned tables, external tables, bitmap join indexes, and domain indexes. Tables owned by `SYSTEM`, `PUBLIC`, `OUTLN`, and `XDB` are also excluded.

Segment creation on demand is not supported for tables created in dictionary-managed tablespaces and for clustered tables. An attempt to do so creates segments.

If you create a table with deferred segment creation on a locally managed tablespace, it has no segments. If at a later time, you migrate the tablespace to be dictionary-managed, any attempt to create segments produces errors. In this case, you must drop the table and re-create it.

Additional Automatic Functionality

Without user intervention:

- No segments for unusable indexes and index partitions
- Creating an index without a segment:

```
CREATE INDEX test_i1 ON seg_test(c) UNUSABLE;
```

- Removing any allocated space for an index:

```
ALTER INDEX test_i UNUSABLE;
```

- Creating the segment for an index:

```
ALTER INDEX test_i REBUILD;
```

```
SELECT segment_name, partition_name, segment_type
FROM  user_segments
WHERE segment_name like '%DEMO';
```


Additional features are implemented in Oracle Database to save space. All `UNUSABLE` indexes and index partitions are created without a segment. This functionality is completely transparent for you.

Example: If you have a `DEMO` table with three partitions and a local index, you see three table and three index segments when executing the query, which is shown in the slide.

If you execute the same query after you move one table partition to a new tablespace, you see three table segments and only two index segments because the unusable one is automatically deleted.

Quiz

Which of the following statements are true?

a. Deferred segment creation is always enabled. You cannot control it.
b. You can control the deferred segment creation with the `SEGMENT CREATION` clause of the `CREATE TABLE` command.
c. Segment creation on demand is available for all types of tables, including those owned by the `SYS` user.
d. Segment creation on demand is available for nonpartitioned tables.

Answer: b, d

Table Compression: Overview

Reducing storage costs by compressing all data:

- Basic compression for direct-path insert operations: 10x
- Advanced row compression for all DML operations: 2–4x

Compression Method	Compression Ratio	CPU Overhead	CREATE and ALTER TABLE Syntax	Typical Applications
Basic table compression	High	Minimal	`COMPRESS [BASIC]`	DSS
Advanced row compression	High	Minimal	`ROW STORE COMPRESS ADVANCED`	OLTP, DSS

Oracle Database supports three methods of table compression:

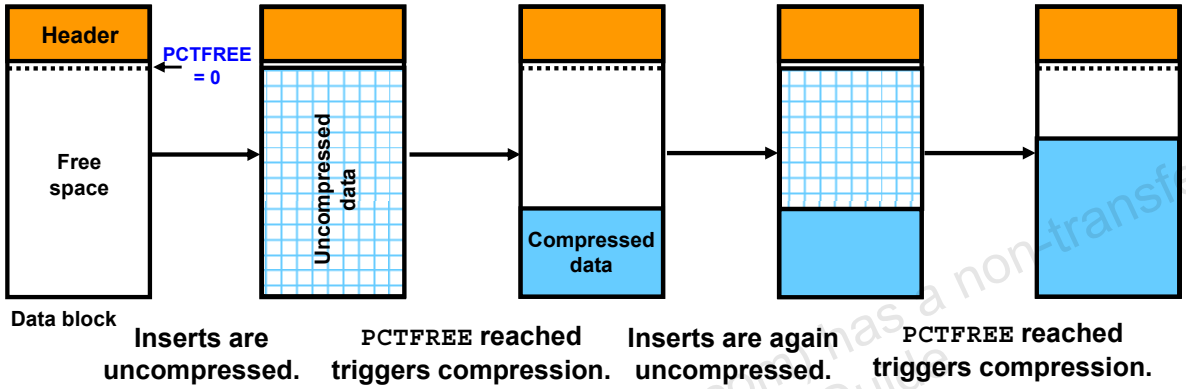
- Basic table compression
- Advanced row compression
- Hybrid columnar compression (with Exadata)

Oracle Corporation recommends compressing all data to reduce storage costs. The Oracle Database server can use table compression to eliminate duplicate values in a data block. For tables with highly redundant data, compression saves disk space and reduces memory use in the database buffer cache. Table compression is transparent to database applications.

- The `table_compression` clause is valid only for heap-organized tables. The `COMPRESS` keyword enables table compression. The `NOCOMPRESS` keyword disables table compression. `NOCOMPRESS` is the default.
- With basic compression, the Oracle Database server compresses data at the time of performing bulk load using operations such as direct loads or `CREATE TABLE AS SELECT`.
- With `ROW STORE COMPRESS ADVANCED`, the Oracle Database server compresses data during all DML operations on the table.

Compression for Direct-Path Insert Operations

- Is enabled with `CREATE TABLE … COMPRESS BASIC …;`
- Is recommended for bulk loading data warehouses
- Maximizes contiguous free space in blocks



With `COMPRESS` or `COMPRESS BASIC`, you enable basic table compression.

- The Oracle Database server attempts to compress data during the following direct-path insert operations when it is productive to do so:
  - Direct-path SQL*Loader
  - `CREATE TABLE AS SELECT` statements
  - Parallel `INSERT` statements
  - `INSERT` statements with an `APPEND` hint
- The original import utility (imp) does not support direct-path `INSERT`, and therefore cannot import data in a compressed format.
- In earlier releases, this type of compression was called DSS table compression and was enabled using `COMPRESS FOR DIRECT_LOAD OPERATIONS`. This syntax has been deprecated.
- Compression eliminates holes created due to deletions and maximizes contiguous free space in blocks.

The slide shows you a data block evolution when that block is part of a compressed table. You should read it from left to right. At the start, the block is empty and available for inserts. When you start inserting into this block, data is stored in an uncompressed format (as for uncompressed tables). However, as soon as the block is filled based on the `PCTFREE` setting of the block, the data is automatically compressed, potentially reducing the space it originally occupied.

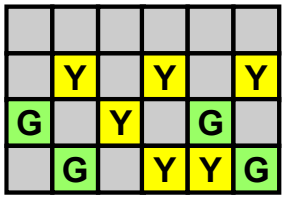
This allows for new uncompressed inserts to take place in the same block, until it is once again filled based on the `PCTFREE` setting. At that point, compression is triggered again to reduce the amount of space used in the block.

Note: Tables with `COMPRESS` or `COMPRESS BASIC` use a `PCTFREE` value of `0` to maximize compression, unless you explicitly set a value for `PCTFREE` clause.

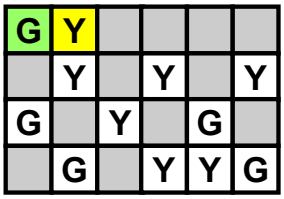
Tables with `ROW STORE COMPRESS ADVANCED` or `NOCOMPRESS` use the `PCTFREE` default value of `10` to maximize compression while still allowing for some future DML changes to the data, unless you override this default explicitly.

Advanced Row Compression for DML Operations

- Is enabled with
  `CREATE TABLE … ROW STORE COMPRESS ADVANCED …;`
- Is recommended for active OLTP environments



Uncompressed block



OLTP compression with symbol table at the beginning of the block

With `ROW STORE COMPRESS ADVANCED`, you enable advanced row compression.

- The Oracle database compresses data during all DML operations on the table. This form of compression is recommended for active OLTP environments.
- In earlier releases, OLTP table compression was enabled with `COMPRESS FOR ALL OPERATIONS` and `COMPRESS FOR OLTP`. This syntax has been deprecated.

With advanced row compression, duplicate values in the rows and columns in a data block are stored once at the beginning of the block in a symbol table. Duplicate values are replaced with a short reference to the symbol table (as shown in the slide). Thus, information needed to re-create the uncompressed data is stored in the block.

To illustrate the principle of advanced row compression, the diagram in the slide shows two rectangles. The first gray rectangle contains four small green squares labeled “G” and six yellow ones labeled “Y.” They represent uncompressed blocks. At the beginning of the second gray rectangle, there is only one green square labeled “G” and one yellow “Y” square, representing the symbol table. The second gray diagram shows 10 white squares in the same position as the green and yellow ones. They are white because they are now only a reference, not consuming space for duplicate values.

Specifying Table Compression

You can specify table compression for:

- An entire heap-organized table
- A partitioned table (Each partition can have a different type or level of compression.)
- The storage of a nested table

You cannot :

- Specify basic and advanced row compression on tables with more than 255 columns
- Drop a column if a table is compressed for direct-loads, but you can drop it if the table is advance row compressed

You can specify table compression:

- For an entire heap-organized table (in the *physical_properties* clause of *relational_table* or *object_table*)
- For partitioned tables (Each partition can have a different type or level of compression.)
- For the storage of a nested table (in the *nested_table_col_properties* clause)

Table compression has the following restrictions:

- `ROW STORE COMPRESS ADVANCED` and `COMPRESS BASIC` are not supported for tables with more than 255 columns.
- You cannot drop a column from a table that is compressed for direct-load operations, although you can set such a column as unused. All the operations of the `ALTER TABLE` ... *drop_column_clause* are valid for tables that are compressed for OLTP.

Using the Compression Advisor

The Compression Advisor:

- Analyzes objects to give an estimate of space savings for different compression methods
- Helps in deciding the correct compression level for an application
- Recommends various strategies for compression
  - Picks the right compression algorithm for a particular data set
  - Sorts on a particular column for increasing the compression ratio
  - Presents tradeoffs between different compression algorithms
- Works for OLTP compression (via Enterprise Manager)

The Compression Advisor analyzes database objects and determines the expected compression ratios that can be achieved for each compression level. So it helps you determine the proper compression levels for your application. The advisor recommends various strategies for compression. When you access it from Enterprise Manager, it determines OLTP compression.

Using the DBMS_COMPRESSION Package

To determine optimal compression ratios:

```
BEGIN
DBMS_COMPRESSION.GET_COMPRESSION_RATIO ('USERS','SH','SALES',
 NULL,DBMS_COMPRESSION.COMP_FOR_OLTP, blkcnt_cmp, blkcnt_uncmp,
 rowcnt_cmp, rowcnt_uncmp, comp_ration, comptype);
DBMS_OUTPUT.PUT_LINE('Blk count compressed = ' || blkcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Blk count uncompressed = ' ||
blkcnt_uncmp);
DBMS_OUTPUT.PUT_LINE('Row count per block compressed = ' ||
rowcnt_cmp);
DBMS_OUTPUT.PUT_LINE('Row count per block uncompressed = ' ||
rowcnt_uncmp);
DBMS_OUTPUT.PUT_LINE('Compression type = ' || comptype);
DBMS_OUTPUT.PUT_LINE('Compression ratio = '||comp_ratio||' to
1');
```

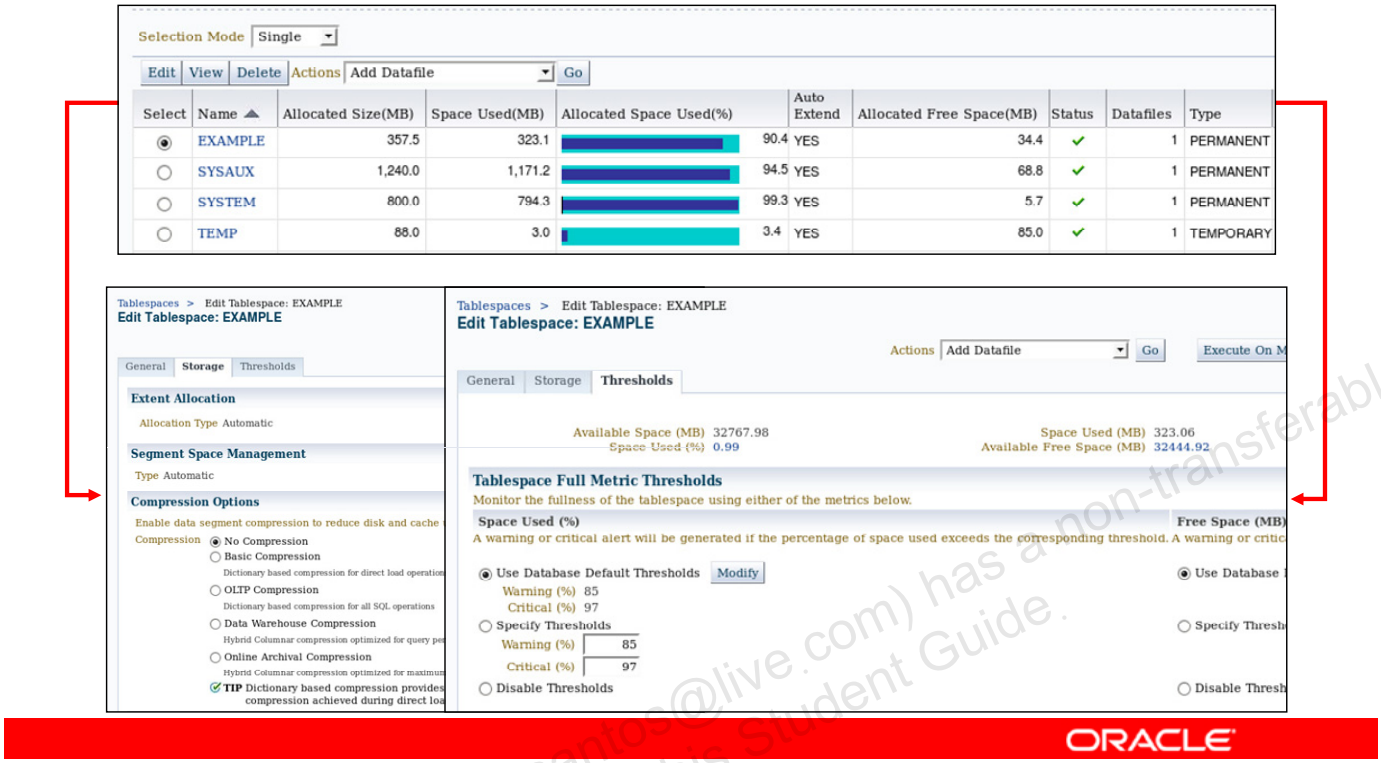

A compression advisor, provided by the DBMS_COMPRESSION package, helps you to determine the compression ratio that can be expected for a specified table. The advisor analyzes the objects in the database, discovers the possible compression ratios that could be achieved, and recommends optimal compression levels. In addition to the DBMS_COMPRESSION package, the compression advisor can also be used within the existing advisor framework (with the DBMS_ADVISOR package).

To determine the compression ratio, the DBMS_COMPRESSION package has the following subprograms:

- The GET_COMPRESSION_RATIO procedure gives you the possible compression ratio for an uncompressed table.
- The GET_COMPRESSION_TYPE function returns the compression type for a given row.

For more details, see the *Oracle Database PL/SQL Packages and Types Reference.*

Proactive Tablespace Monitoring



Tablespace disk space usage is proactively managed by the database in the following ways:

- Through the use of database alerts, you are informed when a tablespace runs low on available disk space as well as when particular segments are running out of space. You can then provide the tablespace with more disk space, thus avoiding out-of-space conditions.
- Information gathered is stored in the Automatic Workload Repository (AWR) and is used to perform growth trend analysis and capacity planning of the database.

To view and modify tablespace information in Enterprise Manager Cloud Control, select Administration > Storage > Tablespaces. Select the tablespace of your choice and click Edit.

Thresholds and Resolving Space Problems

Locally managed tablespace

Resolve space problem by:

- Adding or resizing data file
- Setting `AUTOEXTEND ON`
- Shrinking objects
- Reducing `UNDO_RETENTION`
- Checking for long-running queries in temporary tablespaces

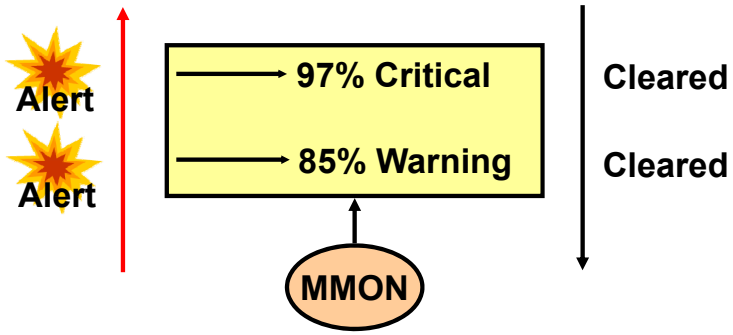
Tablespace thresholds are defined either as full or as available space in the tablespace. Critical and warning thresholds are the two thresholds that apply to a tablespace. The `DBMS_SERVER_ALERT` package contains procedures to set and get the threshold values. When the tablespace limits are reached, an appropriate alert is raised. The threshold is expressed in terms of a percentage of the tablespace size or in remaining bytes free. It is calculated in memory. You can have both a percentage and a byte-based threshold defined for a tablespace. Either or both of them may generate an alert.

The ideal setting for the warning threshold trigger value results in an alert that is early enough to ensure that there is enough time to resolve the problem before it becomes critical, but late enough so that you are not bothered when space is not a problem.

The alert indicates that the problem can be resolved by doing one or more of the following:

- Adding more space to the tablespace by adding a file or resizing existing files, or making an existing file auto-extendable
- Freeing up space on disks that contain any auto-extendable files
- Shrinking sparse objects in the tablespace

Monitoring Tablespace Space Usage



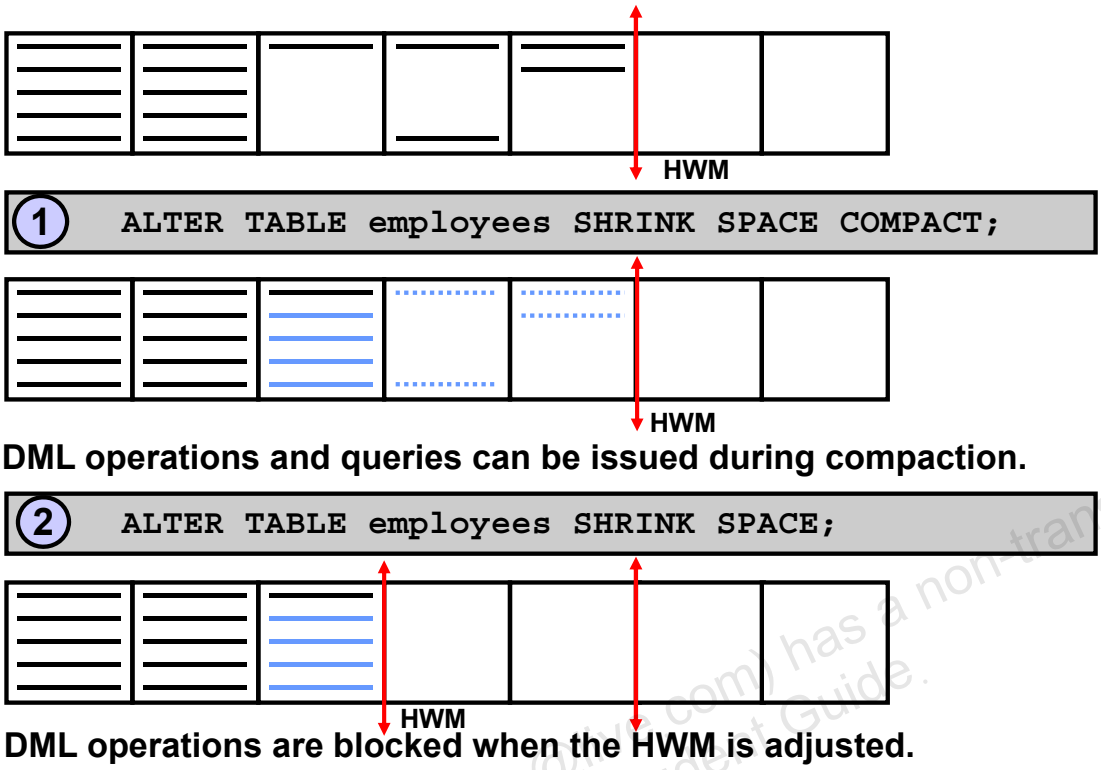
- Read-only and offline tablespaces: Do not set up alerts.
- Temporary tablespace: Threshold corresponds to space currently used by sessions.
- Undo tablespace: Threshold corresponds to space used by active and unexpired extents.
- Auto-extensible files: Threshold is based on the maximum file size.

ORACLE

The database server tracks space utilization while performing regular space management activities. This information is aggregated by the `MMON` process. An alert is triggered when the threshold for a tablespace has been reached or cleared.

- Alerts should not be flagged on tablespaces that are in read-only mode, or tablespaces that were taken offline, because there is not much to do for them.
- In temporary tablespaces, the threshold value has to be defined as a limit on the used space in the tablespace.
- For undo tablespaces, an extent is reusable if it does not contain active or unexpired undo. For the computation of threshold violation, the sum of active and unexpired extents is considered as used space.
- For tablespaces with auto-extensible files, the thresholds are computed according to the maximum file size you specified, or the maximum OS file size.

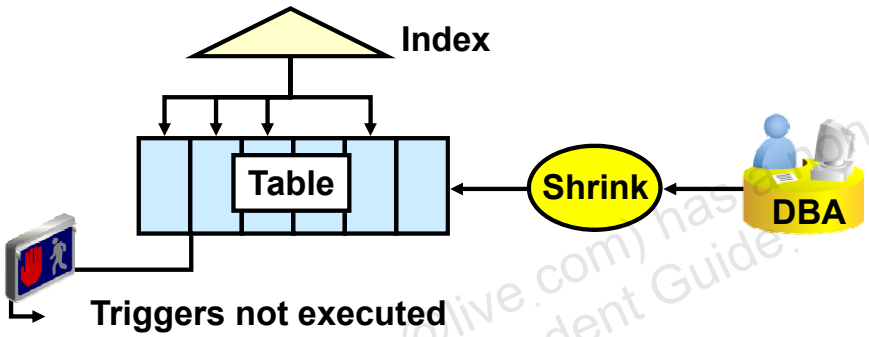
Shrinking Segments



The diagram in the slide describes the two phases of a table shrink operation. Compaction is performed in the first phase. During this phase, rows are moved to the left part of the segment as much as possible. Internally, rows are moved by packets to avoid locking issues. After the rows have been moved, the second phase of the shrink operation is started. During this phase, the high-water mark (HWM) is adjusted and the unused space is released.
The `COMPACT` clause is useful if you have long-running queries that might span the shrink operation and attempt to read from blocks that have been reclaimed. When you specify the `SHRINK SPACE COMPACT` clause, the progress of the shrink operation is saved in the bitmap blocks of the corresponding segment. This means that the next time a shrink operation is executed on the same segment, the Oracle Database server remembers what has been done already. You can then reissue the `SHRINK SPACE` clause without the `COMPACT` clause during off-peak hours to complete the second phase.

Results of Shrink Operation

- Improved performance and space utilization
- Indexes maintained
- Triggers not executed
- Number of migrated rows may be reduced.
- Rebuilding secondary indexes on IOTs recommended



Shrinking a sparsely populated segment improves the performance of scan and DML operations on that segment. This is because there are fewer blocks to look at after the segment has been shrunk. This is especially true for:

- Full table scans (fewer and denser blocks)
- Better index access (fewer I/Os on range `ROWID` scans due to a more compact tree)

Also, by shrinking sparsely populated segments, you enhance the efficiency of space utilization inside your database because more free space is made available for objects in need.

Index dependency is taken care of during the segment shrink operation. The indexes are in a usable state after shrinking the corresponding table. Therefore, no further maintenance is needed.

The actual shrink operation is handled internally as an `INSERT/DELETE` operation. However, DML triggers are not executed because the data itself is not changed.

As a result of a segment shrink operation, it is possible that the number of migrated rows is reduced. However, you should not always depend on reducing the number of migrated rows after a segment has been shrunk. This is because a segment shrink operation may not touch all the blocks in the segment. Therefore, it is not guaranteed that all the migrated rows are handled.

Note: It is recommended to rebuild secondary indexes on an index-organized table (IOT) after a shrink operation.

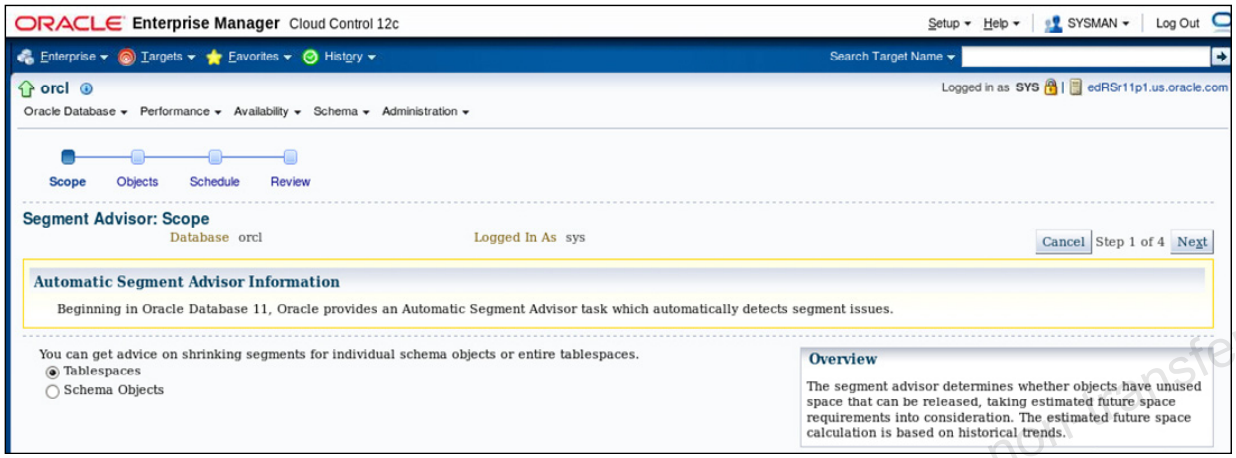
Reclaiming Space Within ASSM Segments

- Online and in-place operation
- Applicable only to segments residing in ASSM tablespaces
- Candidate segment types:
  - Heap-organized tables and index-organized tables
  - Indexes
  - Partitions and subpartitions
  - Materialized views and materialized view logs

A shrink operation is an online and in-place operation because it does not need extra database space to be executed.

- You cannot execute a shrink operation on segments managed by free lists. Segments in automatic segment space–managed tablespaces can be shrunk. However, the following objects stored in ASSM tablespaces cannot be shrunk:
  - Tables in clusters
  - Tables with `LONG` columns
  - Tables with on-commit materialized views
  - Tables with `ROWID`-based materialized views
  - IOT mapping tables
  - Tables with function-based indexes
- `ROW MOVEMENT` must be enabled for heap-organized segments.

Using the Segment Advisor



The Segment Advisor identifies segments that have space available for reclamation. It performs its analysis by examining usage and growth statistics in the Automatic Workload Repository (AWR), and by sampling the data in the segment. It is configured to run automatically at regular intervals, and you can also run it on demand (manually). The regularly scheduled Segment Advisor run is known as the Automatic Segment Advisor.

After the recommendations are made, you can choose to implement the recommendations. The shrink advisor can be invoked at the segment or tablespace level.

Use Enterprise Manager Database Cloud Control to invoke the Segment Advisor. You can access the Segment Advisor from several places within Enterprise Manager Cloud Control:

- Advisor Home page
- Tablespaces page
- Schema object pages

Enterprise Manager Cloud Control provides the option to select various inputs and schedule a job that calls the Segment Advisor to get shrink advice. The Segment Advisor Wizard can be invoked with no context, in the context of a tablespace, or in the context of a schema object.

The Segment Advisor makes recommendation on the basis of sampled analysis, historical information, and future growth trends.

Automatic Segment Advisor

The Automatic Segment Advisor:

- Is started by a Scheduler job set to run during the default maintenance window:
  - Weeknights, Monday–Friday, from 10:00 PM to 2:00 AM
  - Saturday and Sunday, both windows start at 6:00 AM and last for 20 hours
- Examines database statistics, samples segment data, and then selects the following objects to analyze:
  - Tablespaces that have exceeded a critical or warning threshold
  - Segments that have the most activity
  - Segments that have the highest growth rate

The Automatic Segment Advisor is started by a Scheduler job that is configured to run during the default maintenance window. The default maintenance window is specified in the Scheduler, and is initially defined as follows:

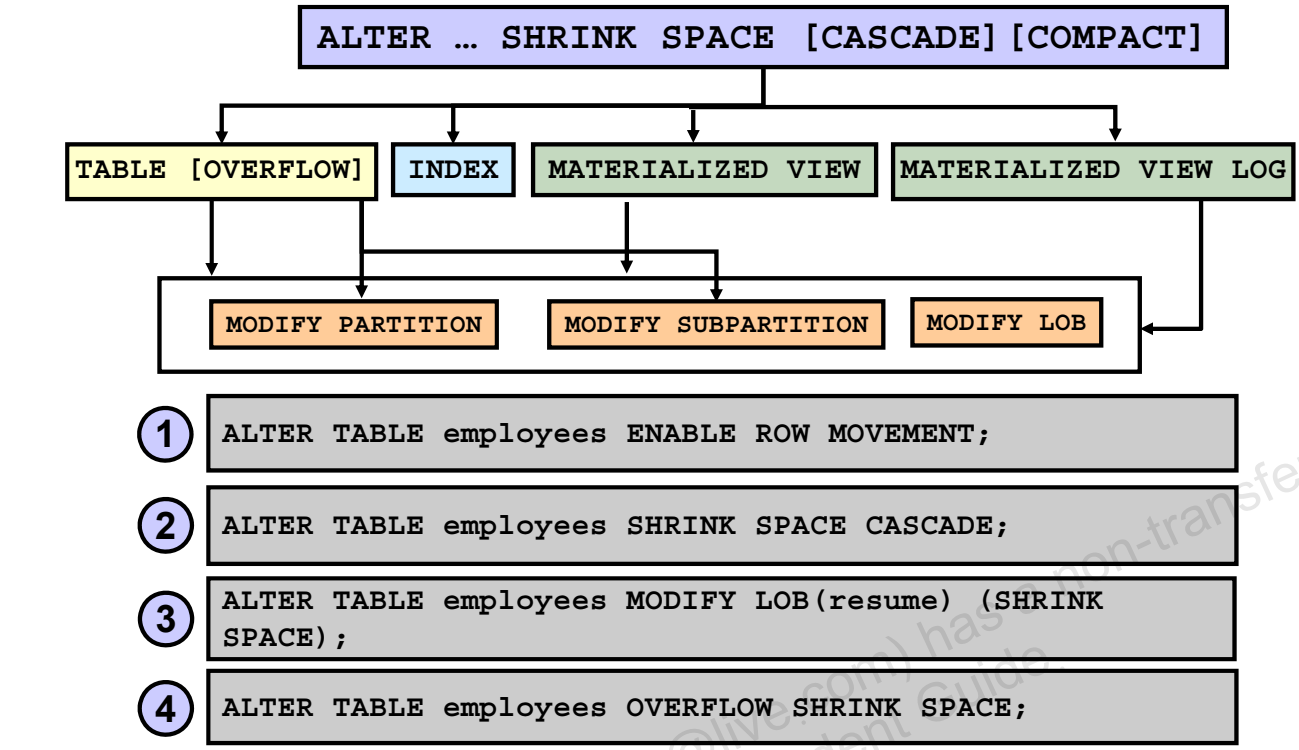
- Weeknights, Monday through Friday, from 10:00 PM to 2:00 AM (4 hours each night)
- Weekends, Saturday and Sunday morning at 6:00 AM and lasting for 20 hours each day.

The Automatic Segment Advisor does not analyze every database object. Instead, it examines database statistics, samples segment data, and then selects the following objects to analyze:

- Tablespaces that have exceeded a critical or warning space threshold
- Segments that have the most activity
- Segments that have the highest growth rate

If an object is selected for analysis but the maintenance window expires before the Segment Advisor can process the object, the object is included in the next Automatic Segment Advisor run. You cannot change the set of tablespaces and segments that the Automatic Segment Advisor selects for analysis. You can, however, enable or disable the Automatic Segment Advisor job, change the times during which the Automatic Segment Advisor is scheduled to run, or adjust Automatic Segment Advisor system resource utilization.

Shrinking Segments by Using SQL



Because a shrink operation may cause `ROWID`s to change in heap-organized segments, you must enable row movement on the corresponding segment before executing a shrink operation on that segment. Row movement by default is disabled at segment level. To enable row movement, the `ENABLE ROW MOVEMENT` clause of the `CREATE TABLE` or `ALTER TABLE` command is used. This is illustrated in the first example in the slide.

Use the `ALTER` command to invoke segment shrink on an object. The object's type can be one of the following: table (heap- or index-organized), partition, subpartition, LOB (data and index segment), index, materialized view, or materialized view log.

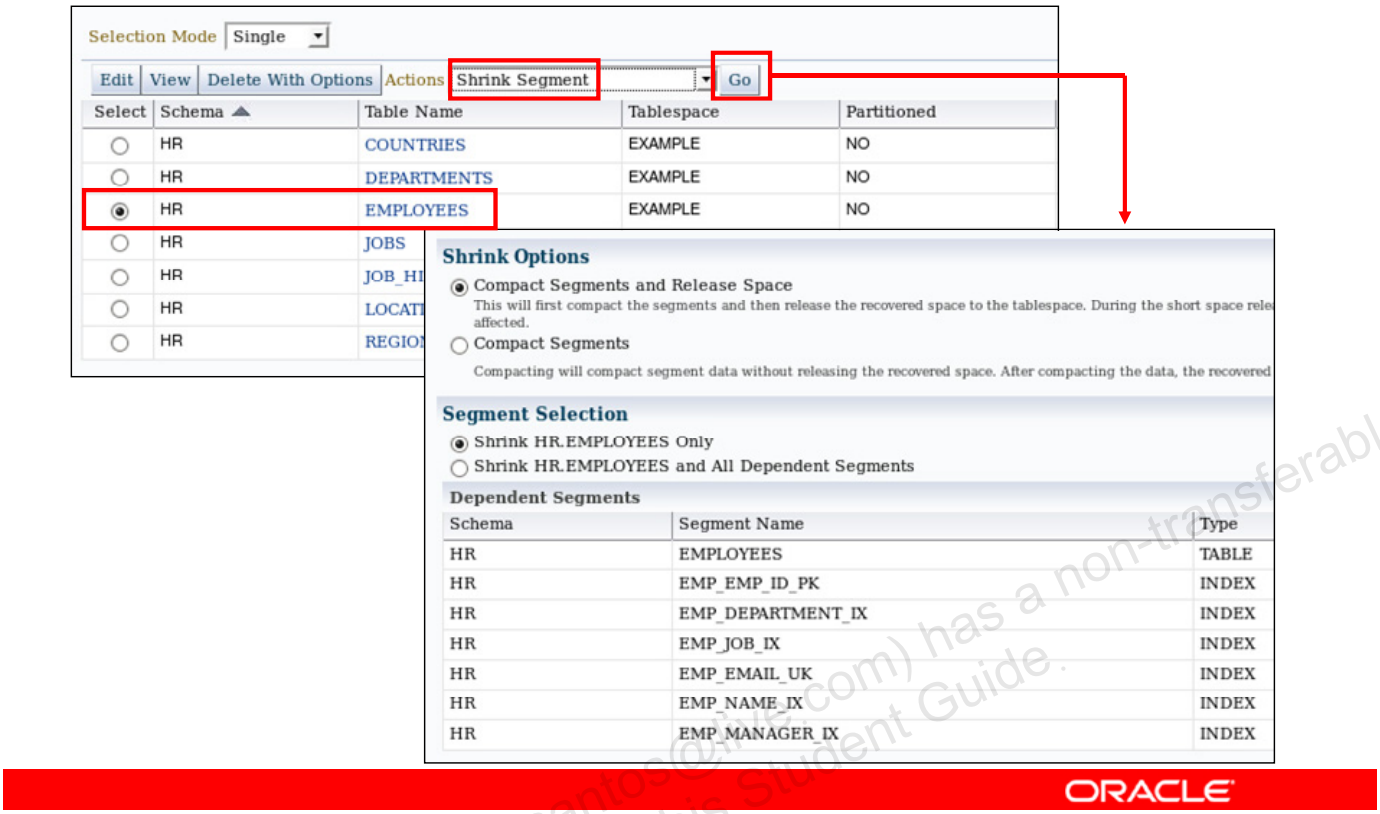
Use the `SHRINK SPACE` clause to shrink space in a segment. If `CASCADE` is specified, the shrink behavior is cascaded to all the dependent segments that support a shrink operation, except materialized views, LOB indexes, and IOT (index-organized tables) mapping tables. The `SHRINK SPACE` clause is illustrated in the second example.

In an index segment, the shrink operation coalesces the index before compacting the data. Example 3 shows a command that shrinks an LOB segment, given that the `RESUME` column is a `CLOB`.

Example 4 shows a command that shrinks an IOT overflow segment belonging to the `EMPLOYEES` table.

Note: For more information, refer to the *Oracle Database SQL Reference* guide.

Shrinking Segments by Using Enterprise Manager



You can manually shrink individual segments associated with specific database objects by using Enterprise Manager Cloud Control. On the Tables page, select your table, and then select Shrink Segment in the Actions drop-down list. Then click the Go button. This brings you to the Shrink Segment page, where you can choose the dependent segments to shrink. You have the opportunity to compact only or to compact and release the space. You can also choose the `CASCADE` option.

When done, click the Continue link. This submits the shrink statements as a scheduled job.

Managing Resumable Space Allocation

A resumable statement:

- Enables you to suspend large operations instead of receiving an error
- Gives you a chance to fix the problem while the operation is suspended, rather than starting over
- Is suspended for the following conditions:
  - Out of space
  - Maximum extents reached
  - Space quota exceeded
- Can be suspended and resumed multiple times

The Oracle Database server provides a means for suspending, and later resuming, the execution of large database operations in the event of space allocation failures. This enables you to take corrective action instead of the Oracle Database server returning an error to the user. After the error condition is corrected, the suspended operation automatically resumes. This feature is called “resumable space allocation.” The statements that are affected are called “resumable statements.” A statement executes in resumable mode only when the resumable statement feature has been enabled for the system or session.

Suspending a statement automatically results in suspending the transaction. Thus, all transactional resources are held through the suspension and resuming of a SQL statement. When the error condition disappears (for example, as a result of user intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution. A resumable statement is suspended when one of the following conditions occur:

- Out of space condition
- Maximum extents reached condition
- Space quota exceeded condition

A suspension timeout interval is associated with resumable statements. A resumable statement that is suspended for the timeout interval (the default is 2 hours) reactivates itself and returns the exception to the user. A resumable statement can be suspended and resumed multiple times.

Note: A maximum extents reached error happens only with dictionary-managed tablespaces.

Using Resumable Space Allocation

- Queries, DML operations, and certain DDL operations can be resumed if they encounter an out-of-space error.
- A resumable statement can be issued through SQL, PL/SQL, SQL*Loader, and Data Pump utilities, or the Oracle Call Interface (OCI).
- A statement executes in resumable mode only if its session has been enabled by one of the following actions:
  - The **RESUMABLE_TIMEOUT** initialization parameter is set to a nonzero value.
  - An **ALTER SESSION ENABLE RESUMABLE** statement is issued:

```
ALTER SESSION ENABLE RESUMABLE;
INSERT INTO sales_new SELECT * FROM sh.sales;
ALTER SESSION DISABLE RESUMABLE;
```

ORACLE

Resumable space allocation is possible only when statements are executed within a session that has resumable mode enabled. There are two means of enabling and disabling resumable space allocation:

- Issue the `ALTER SESSION ENABLE RESUMABLE` command.
- Set the `RESUMABLE_TIMEOUT` initialization parameter to a nonzero value with an `ALTER SESSION` or `ALTER SYSTEM` statement.

When enabling resumable mode for a session or the database, you can specify a timeout period, after which a suspended statement errors out if no intervention has taken place. The `RESUMABLE_TIMEOUT` initialization parameter indicates the number of seconds before a timeout occurs. You can also specify the timeout period with the following command:

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600;
```

The value of `TIMEOUT` remains in effect until it is changed by another `ALTER SESSION ENABLE RESUMABLE` statement, it is changed by another means, or the session ends. The default timeout interval when using the `ENABLE RESUMABLE TIMEOUT` clause to enable resumable mode is 7,200 seconds, or 2 hours.

You can also give a name to resumable statements. Example:

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600
NAME 'multitab insert';
```

The name of the statement is used to identify the resumable statement in the DBA_RESUMABLE and USER_RESUMABLE views.

Example:

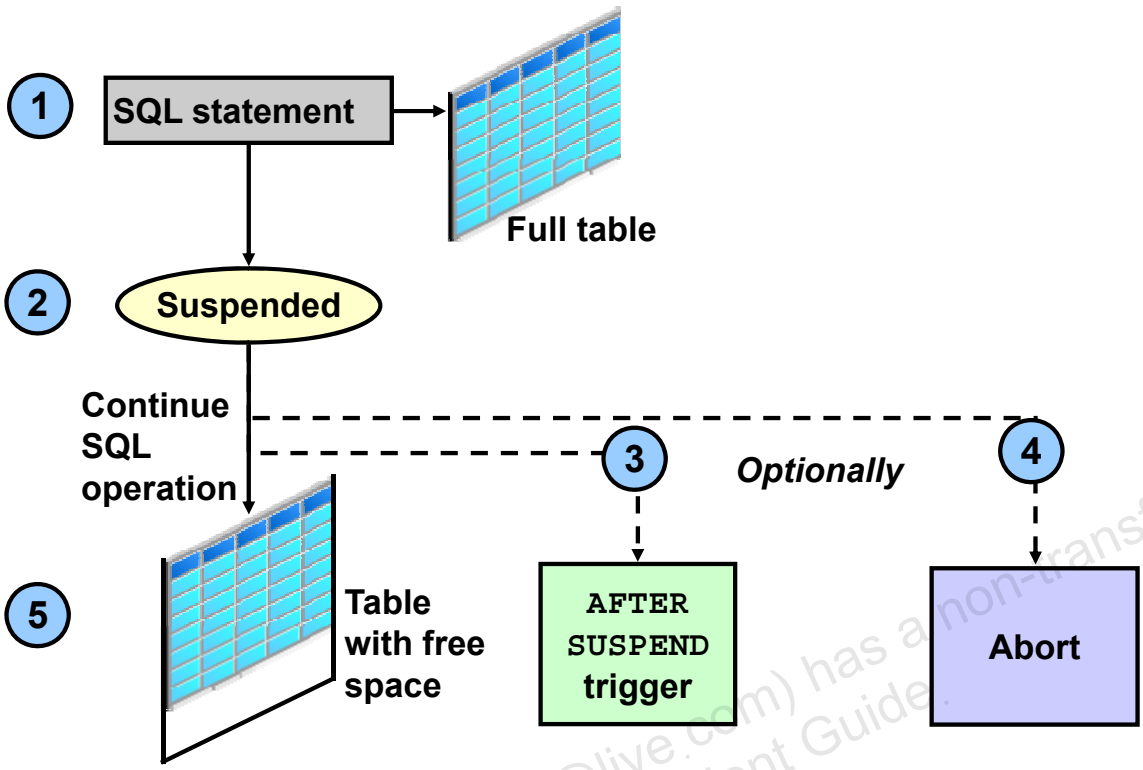
```
SELECT name, sql_text FROM user_resumable;


NAME            SQL_TEXT
--------------- ------------------------------------------------
multitab insert INSERT INTO oldsales SELECT * FROM sh.sales;
```

To automatically configure resumable statement settings for individual sessions, you can create and register a database-level LOGON trigger that alters a user's session. The trigger issues commands to enable resumable statements for the session, specifies a timeout period, and associates a name with the resumable statements issued by the session.

Because suspended statements can hold up some system resources, users must be granted the RESUMABLE system privilege before they are allowed to enable resumable space allocation and execute resumable statements.

Resuming Suspended Statements



Example

1. An `INSERT` statement encounters an error saying the table is full.
2. The `INSERT` statement is suspended, and no error is passed to the client.
3. Optionally, an `AFTER SUSPEND` trigger is executed.
4. Optionally, the `SQLERROR` exception is activated to abort the statement.
5. If the statement is not aborted and free space is successfully added to the table, the `INSERT` statement resumes execution.

Detecting a Suspended Statement

When a resumable statement is suspended, the error is not raised to the client. In order for corrective action to be taken, the Oracle Database server provides alternative methods for notifying users of the error and for providing information about the circumstances.

Possible Actions During Suspension

When a resumable statement encounters a correctable error, the system internally generates the `AFTER SUSPEND` system event. Users can register triggers for this event at both the database and schema level. If a user registers a trigger to handle this system event, the trigger is executed after a SQL statement has been suspended. SQL statements executed within an `AFTER SUSPEND` trigger are always nonresumable and are always autonomous. Transactions started within the trigger use the `SYSTEM` rollback segment. These conditions are imposed to overcome deadlocks and reduce the chance of the trigger experiencing the same error condition as the statement.

Within the trigger code, you can use the `USER_RESUMABLE` or `DBA_RESUMABLE` views, or the `DBMS_RESUMABLE.SPACE_ERROR_INFO` function to get information about the resumable statements.

When a resumable statement is suspended:

- The session invoking the statement is put into a wait state. A row is inserted into `V$SESSION_WAIT` for the session with the `EVENT` column containing "`statement suspended, wait error to be cleared`".
- An operation-suspended alert is issued on the object that needs additional resources for the suspended statement to complete.

Ending a Suspended Statement

When the error condition is resolved (for example, as a result of DBA intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution and the "resumable session suspended" alert is cleared.

A suspended statement can be forced to activate the `SERVERERROR` exception by using the `DBMS_RESUMABLE.ABORT()` procedure. This procedure can be called by a DBA, or by the user who issued the statement. If the suspension timeout interval associated with the resumable statement is reached, the statement aborts automatically and an error is returned to the user.

What Operations Are Resumable?

The following operations are resumable:

- Queries: `SELECT` statements that run out of temporary space (for sort areas)
- DML: `INSERT`, `UPDATE`, and `DELETE` statements
- The following DDL statements:
  - **`CREATE TABLE ... AS SELECT`**
  - **`CREATE INDEX`**
  - **`ALTER INDEX ... REBUILD`**
  - **`ALTER TABLE ... MOVE PARTITION`**
  - **`ALTER TABLE ... SPLIT PARTITION`**
  - **`ALTER INDEX ... REBUILD PARTITION`**
  - **`ALTER INDEX ... SPLIT PARTITION`**
  - **`CREATE MATERIALIZED VIEW`**

ORACLE

The following operations are resumable:

- **Queries:** `SELECT` statements that run out of temporary space (for sort areas) are candidates for resumable execution. When using OCI, the `OCIStmtExecute()` and `OCIStmtFetch()` calls are candidates.
- **DML:** `INSERT`, `UPDATE`, and `DELETE` statements are candidates. The interface used to execute them does not matter; it can be OCI, SQLJ, PL/SQL, or another interface. Also, `INSERT INTO...SELECT` from external tables can be resumable.
- **DDL:** The following statements are candidates for resumable execution:
  - `CREATE TABLE ... AS SELECT`
  - `CREATE INDEX`
  - `ALTER INDEX ... REBUILD`
  - `ALTER TABLE ... MOVE PARTITION`
  - `ALTER TABLE ... SPLIT PARTITION`
  - `ALTER INDEX ... REBUILD PARTITION`
  - `ALTER INDEX ... SPLIT PARTITION`
  - `CREATE MATERIALIZED VIEW`

Quiz

Select the true statements about space management.

a. Segment creation in Oracle Database 12*c* is deferred for all tables. There are no exceptions.
b. All `UNUSABLE` indexes and index partitions are created without a segment.
c. Shrinking segment space is a nonresumable operation.
d. You can set thresholds by tablespace.

Answer: b, d

Summary

In this lesson, you should have learned how to:

- Describe how the Oracle Database server automatically manages space
- Save space by using compression
- Proactively monitor and manage tablespace space usage
- Describe segment creation in the Oracle database
- Control deferred segment creation
- Use the Segment Advisor
- Reclaim wasted space from tables and indexes by using the segment shrink functionality
- Manage resumable space allocation

Practice 11

11-1: Managing Storage