

Windows Platform: CREATE TABLESPACE WORK
datafile 'C:\Oracle11g\oradata\wrk01.dbf' size 500M online;

Unix/ Linux Platform: CREATE TABLESPACE WORK
datafile '/u01/oradata/wrk01.dbf' size 500M online;

Creating Tablespaces

How a user uses Tablespace?

Create user **ejaz** identified by **pwd** default tablespace **WORK** temporary tablespace **TEMP**;

--Assigning default roles to user(owner)

Grant connect, resource to ejaz;

Grant create table to ejaz; Grant create view to ejaz;

Creating Tablespaces

Space Management in Tablespace

Tablespaces allocate space in extents. Tablespaces can be created to use one of the following two different methods of keeping track of free and used space:

Locally managed tablespaces: The extents are managed within the tablespace via bitmaps. Each bit in the bitmap corresponds to a block or a group of blocks.

Creating Tablespaces

Space Management in Tablespace

When an extent is allocated or freed for reuse, the Oracle server changes the bitmap values to show the new status of the blocks. Locally managed is the default beginning Oracle.

Dictionary-managed tablespaces: The extents are managed by the data dictionary. The Oracle server updates the appropriate tables in the data dictionary whenever an extent is allocated or deallocated.

Creating Tablespaces

Locally-Managed Tablespaces

Reduced contention on data dictionary tables

No undo generated when space allocation or deallocation occurs

No coalescing required

```
CREATE TABLESPACE userdata datafile 'C:\Ora11g\oradata\usr01.dbf' size 500M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

Creating Tablespaces

Dictionary-Managed Tablespaces

Extents are managed in the data dictionary
Each segment stored in the tablespace can have a different storage clause.
No coalescing required

```
CREATE TABLESPACE userdata datafile 'C:\Ora11g\oradata\usr01.dbf' size 500M EXTENT
MANAGEMENT DICTIONARY DEFAULT STORAGE (Initial 1M NEXT 1M
PCTINCREASE 0);
```

Creating Tablespaces

Create UNDO Tablespace

Used to store undo segments
Cannot contain any other objects
Extents are locally managed
Can only use the DATAFILE and EXTENT
MANAGEMENT Clause

```
CREATE UNDO TABLESPACE undo01
datafile 'C:\Oracle11g\oradata\undo01.dbf' size 40M;
```

Creating Tablespaces

Create Temporary Tablespace

Used for sort operations
Can be shared by multiple users
Cannot contain any permanent objects
Locally managed extents recommended

```
CREATE Temporary TABLESPACE temp1 datafile 'C:\Oracle11g\oradata\temp101.dbf' size
20M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4M;
```

Creating a default Temporary Tablespace

After database creation: ALTER DATABASE
DEFAULT TEMPORARY TABLESPACE

default_temp1;

To see default temporary tablespace, use
Select property_name from database_properties;
Creating Tablespaces

Restrictions

Default temporary tablespace cannot be:

Dropped until after a new default is
made available
Taken offline
Alter to a permanent tablespace

End
Introduction to Database Administration

TOPIC 74

Specifying Non Standard Block Sizes for Tablespaces

Specifying Non Standard Block Sizes for Tablespaces

Block Sizes for Tablespaces

Bigfile Tablespaces
Creating Bigfile Tablespaces
VIEWS for Bigfile Tablespaces
Non Standard Block Sizes for Tablespaces
Creating Non Standard Tablespaces
Specifying Non Standard Block Sizes for Tablespaces

Bigfile Tablespaces

A bigfile tablespace is a tablespace with a single, but potentially very large (up to 4G blocks) data file. Traditional small file tablespaces, in contrast, can contain multiple data files, but the files cannot be as large.

Specifying Non Standard Block Sizes for Tablespaces

Bigfile Tablespaces

The benefits of bigfile tablespaces are as follows:

A bigfile tablespace with 8K blocks can contain a 32 terabyte data file. A bigfile tablespace with 32K blocks can contain 128 terabyte data file. The maximum number of data files in an Oracle Database is limited (usually to 64K files). Therefore, bigfile tablespaces can significantly enhance the storage capacity of an Oracle DB

Specifying Non Standard Block Sizes for Tablespaces

Bigfile Tablespaces

Bigfile tablespaces can reduce the number of data files needed for a database. An additional benefit is that the DB_FILES initialization parameter and MAXDATAFILES parameter of the CREATE DATABASE and CREATE CONTROLFILE statements can be adjusted to reduce the amount of SGA space required for data file information and the size of the control file.

Specifying Non Standard Block Sizes for Tablespaces

Bigfile Tablespaces

Bigfile tablespaces simplify database management by providing data file transparency. SQL syntax for the ALTER TABLESPACE statement lets you perform operations on tablespaces, rather than the underlying individual data files.

Specifying Non Standard Block Sizes for Tablespaces

Creating Bigfile Tablespaces

```
CREATE BIGFILE TABLESPACE bigtbs  
DATAFILE '/u02/oracle/data/bigtbs01.dbf'  
SIZE 50G
```

...

You can specify SIZE in kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T).

If the default tablespace type was set to BIGFILE at database creation then bigfile tablespace will be created by default.

Specifying Non Standard Block Sizes for Tablespaces

VIEWS for Bigfile Tablespaces

```
DBA_TABLESPACES  
USER_TABLESPACES
```

V\$TABLESPACE

Specifying Non Standard Block Sizes for Tablespaces

Non Standard Block Sizes for Tablespaces

You can create tablespaces with block sizes different from the standard database block size, which is specified by the DB_BLOCK_SIZE initialization parameter. This feature lets you transport tablespaces with unlike block sizes between databases.

Specifying Non Standard Block Sizes for Tablespaces

Non Standard Block Sizes for Tablespaces

Use the BLOCKSIZE clause of the CREATE TABLESPACE statement to create a tablespace with a block size different from the database standard block size. In order for the BLOCKSIZE clause to succeed, you must have already set the DB_CACHE_SIZE and at least one DB_nK_CACHE_SIZE initialization parameter.

Specifying Non Standard Block Sizes for Tablespaces

Non Standard Block Sizes for Tablespaces

Further, and the integer you specify in the BLOCKSIZE clause must correspond with the setting of one DB_nK_CACHE_SIZE parameter setting. Although redundant, specifying a BLOCK SIZE equal to the standard block size, as specified by the DB_BLOCK_SIZE initialization parameter, is allowed.

Specifying Non Standard Block Sizes for Tablespaces

Creating Non Standard Tablespaces

```
CREATE TABLESPACE lmtbsb DATAFILE  
'u02/oracle/data/lmtbsb01.dbf' SIZE 50M  
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K BLOCKSIZE 8K;
```

End

Introduction to Database Administration

TOPIC 75

Altering Tablespaces

Altering Tablespaces

Altering Tablespaces

Altering Tablespaces Availability

Taking Tablespaces Offline

Tablespaces cannot Offline

Altering Tablespace SQL

Owner of Tablespace

Making Tablespace Readonly

Dictionary View for use of Altering Tablespace

Wait for Altering Tablespace to Readonly

Altering Tablespaces

Altering Tablespaces Availability

Before you can make a tablespace read- only, the following conditions must be met

You can take an online tablespace offline so that it is temporarily unavailable for general use. The rest of the database remains open and available for users to access data.

Altering Tablespaces

Altering Tablespaces Availability

Conversely, you can bring an offline tablespace online to make the schema objects within the tablespace available to database users.

The database must be open to alter the availability of a tablespace.

Altering Tablespaces

Taking Tablespaces Offline

To make a portion of the database unavailable while allowing normal access to the remainder of the database

To perform an offline tablespace backup (even though a tablespace can be backed up while online and in use)

Altering Tablespaces

Taking Tablespaces Offline

To make an application and its group of tables temporarily unavailable while updating or maintaining the application

To rename or relocate tablespace data files

Altering Tablespaces

Tablespaces cannot Offline

SYSTEM

The undo tablespace

Temporary tablespaces

Altering Tablespaces

Altering Tablespace SQL

Here are sample examples ALTER TABLESPACE users OFFLINE; ALTER TABLESPACE users ONLINE;

ALTER TABLESPACE work

add datafile 'C:\oracle11g\oradata\wrk02.dbf' size 500M;

Altering Tablespaces

Owner of Tablespace

Following SQL statement assigns tablespace to a particular user:

CREATE USER meduser Identified by pwd

default tablespace WORK temporary tablespace TEMP;

Altering Tablespaces

Making Tablespace Read only

The tablespace must be online. This is necessary to ensure that there is no undo information that must be applied to the tablespace.

The tablespace cannot be the active undo tablespace or SYSTEM tablespace.

Altering Tablespaces

Making Tablespace Read only

The tablespace must not currently be involved in an online backup, because the end of a backup updates the header file of all data files in the tablespace.

The tablespace cannot be a temporary tablespace.

Altering Tablespaces

Dictionary View for use of Altering Tablespace

```

SELECT SQL_TEXT, SADDR
FROM V$SQLAREA,V$SESSION
WHERE V$SQLAREA.ADDRESS = V$SESSION.SQL_ADDRESS
AND SQL_TEXT LIKE 'alter tablespace%';

```

SQL_TEXT	SADDR
alter tablespace tbs1 read only	80034AF0

Altering Tablespaces

Dictionary View for use of Altering Tablespace

```

SELECT SES_ADDR, START_SCNB
FROM V$TRANSACTION
ORDER BY START_SCNB;

```

SES_ADDR	START_SCNB	
800352A0	3621	--> waiting on this txn
80035A50	3623	--> waiting on this txn
80034AF0	3628	--> this is the ALTER TABLESPACE statement
80037910	3629	--> don't care about this txn

End
Introduction to Database Administration

TOPIC 76

Readonly & Online Status

of Tablespaces

Readonly & Online Status of Tablespaces

Readonly & Online Status of Tablespaces

Steps to Create Readonly Tablespaces

SQL Alter Tablespace Command

Delaying the opening of Data Files

Side Effect of Delaying the opening of Data Files

Important Issues of Delaying the opening of Data Files

Readonly & Online Status of Tablespaces

Steps to Create Readonly Tablespaces

Create a writable tablespace on another device. Create the objects that belong in the tablespace and insert your data.

Alter the tablespace to make it read- only.

Copy the data files of the tablespace onto the WORM device. Use operating system commands to copy the files.

Take the tablespace offline.

Readonly & Online Status of Tablespaces

Steps to Create Readonly Tablespaces

Rename the data files to coincide with the names of the data files you copied onto your WORM device. Use ALTER TABLESPACE with the RENAME DATAFILE clause.

Renaming the data files changes their names in the control file.

Bring the tablespace back online.

Readonly & Online Status of Tablespaces

SQL Alter Tablespace Command

ALTER TABLESPACE userdata READ ONLY;

Causes a checkpoint

Data available only for read operations

Objects can be dropped from tablespace

Readonly & Online Status of Tablespaces

SQL Alter Tablespace Command

RMAN>

SQL 'ALTER TABLESPACE hr READ ONLY'; CONVERT TABLESPACE hr
TO PLATFORM 'Solaris[tm] OE (64-bit)' FORMAT
'/tmp/transport_to_solaris/%U'; We will discuss RMAN later.
Readonly & Online Status of Tablespaces

Delaying the opening of Data Files

When substantial portions of a very large database are stored in read- only tablespaces that are located on slow-access devices or hierarchical storage, you should consider setting the READ_ONLY_OPEN_DELAYED initialization parameter to TRUE.
Readonly & Online Status of Tablespaces

Delaying the opening of Data Files

This speeds certain operations, primarily opening the database, by causing data files in read- only tablespaces to be accessed for the first time only when an attempt is made to read data stored within them.
Readonly & Online Status of Tablespaces

Side Effect of Delaying the opening of Data Files

Setting READ_ONLY_OPEN_DELAYED=TRUE has the following side-effects
A missing or bad read-only file is not detected at open time. It is only discovered when there is an attempt to access it.
Readonly & Online Status of Tablespaces

Side Effect of Delaying the opening of Data Files

ALTER SYSTEM CHECK DATAFILES does not check read- only files.
ALTER TABLESPACE...ONLINE and ALTER DATABASE DATAFILE...ONLINE do not check read-only files. They are checked only upon the first access.
Readonly & Online Status of Tablespaces

Side Effect of Delaying the opening of Data Files

V\$RECOVER_FILE, V\$BACKUP, and V\$DATAFILE_HEADER do not access read-only files.
Read-only files are indicated in the results list with the error "DELAYED OPEN", with zeroes for the values of other columns.
Readonly & Online Status of Tablespaces

Side Effect of Delaying the opening of Data Files

V\$DATAFILE does not access read-only files. Read-only files have a size of "0" listed.
V\$RECOVERY_LOG does not access read-only files. Logs they could need for recovery are not added to the list.

Readonly & Online Status of Tablespaces

Side Effect of Delaying the opening of Data Files

ALTER DATABASE NOARCHIVELOG does not access read-only files. It proceeds even if there is a read-only file that requires recovery.

Readonly & Online Status of Tablespaces

Important Issues of Delaying the opening of Data Files

RECOVER DATABASE and ALTER DATABASE OPEN RESETLOGS continue to access all read-only data files regardless of the parameter value. To avoid accessing read-only files for these operations, take those files offline.

Readonly & Online Status of Tablespaces

Important Issues of Delaying the opening of Data Files

If a backup control file is used, the read-only status of some files may be inaccurate. This can cause some of these operations to return unexpected results. Care should be taken in this situation.

End

Introduction to Database Administration

TOPIC 77

Altering & Maintaining Tablespaces

Altering & Maintaining Tablespaces

Maintaining Tablespaces

Altering a Locally Managed Tablespace

Add a bigfile Tablespace

Altering a Locally Managed Temporary Tablespace

Shrinking a Locally Managed Temporary Tablespace Altering & Maintaining Tablespaces

Altering a Locally Managed Tablespace

You cannot alter a locally managed tablespace to a locally managed temporary tablespace, nor can you change its method of segment space management. Coalescing free extents is unnecessary for locally managed tablespaces. However, you can use the ALTER TABLESPACE statement on locally managed tablespaces for some operations, including the following:

Altering & Maintaining Tablespaces

Altering a Locally Managed Tablespace

Adding a data file. For example: ALTER TABLESPACE lmtbsb
ADD DATAFILE

'u02/oracle/data/lmtbsb02.dbf' SIZE 1M;

Altering & Maintaining Tablespaces

Altering a Locally Managed Tablespace

Altering tablespace availability (ONLINE/OFFLINE).

Making a tablespace read-only or read/write

Renaming a data file, or enabling or disabling the autoextension of the size of a data file in the tablespace.

Altering & Maintaining Tablespaces

Add a bigfile Tablespace

Two clauses of the ALTER TABLESPACE statement support data file transparency when you are using bigfile tablespaces:

RESIZE: The RESIZE clause lets you resize the single data file in a bigfile tablespace to an absolute size, without referring to the data file. For example: ALTER TABLESPACE bigtbs
RESIZE 80G;

AUTOEXTEND(used outside of the ADD DATAFILE clause):

Altering & Maintaining Tablespaces

Add a bigfile Tablespace

With a bigfile tablespace, you can use the AUTOEXTEND clause outside of the ADD DATAFILE clause.

For example:

ALTER TABLESPACE bigtbs AUTOEXTEND ON NEXT 20G;

An error is raised if you specify an ADD DATAFILE clause for a bigfile tablespace.
Altering & Maintaining Tablespaces

Altering a Locally Managed Temporary Tablespace

You can use ALTER TABLESPACE to add a temp file, take a temp file offline, or bring temp file online, as illustrated in the following examples:

```
ALTER TABLESPACE lmtmp ADD TEMPFILE  
'u02/oracle/data/lmtmp02.dbf' SIZE 18M REUSE;
```

Altering & Maintaining Tablespaces

Altering a Locally Managed Temporary Tablespace

```
ALTER TABLESPACE lmtmp TEMPFILE OFFLINE;
```

```
ALTER TABLESPACE lmtmp TEMPFILE ONLINE;
```

Altering & Maintaining Tablespaces

Altering a Locally Managed Temporary Tablespace

The ALTER DATABASE statement can be used to alter temp files.
The following statements take offline and bring online temp files. They behave identically to the last two ALTER TABLESPACE statements in the previous example.
Altering & Maintaining Tablespaces

Altering a Locally Managed Temporary Tablespace

```
ALTER DATABASE TEMPFILE  
'u02/oracle/data/lmtmp02.dbf' OFFLINE;  
ALTER DATABASE TEMPFILE  
'u02/oracle/data/lmtmp02.dbf' ONLINE;
```

Altering & Maintaining Tablespaces

Altering a Locally Managed Temporary Tablespace

The following statement resizes a temp file:
ALTER DATABASE TEMPFILE
'u02/oracle/data/lmtmp02.dbf' RESIZE 18M;

Altering & Maintaining Tablespaces

Altering a Locally Managed Temporary Tablespace

The following statement drops a temp file and deletes its operating system file:

```
ALTER DATABASE TEMPFILE
```

```
'u02/oracle/data/lmtemp02.dbf' DROP INCLUDING DATAFILES;
```

The tablespace to which this temp file belonged remains. A message is written to the alert log for the temp file that was deleted.

Altering & Maintaining Tablespaces

Shrinking a Locally Managed Temporary Tablespace

Large sort operations performed by the database may result in a temporary tablespace growing and occupying a considerable amount of disk space. After the sort operation completes, the extra space is not released; it is just marked as free and available for reuse.

Altering & Maintaining Tablespaces

Shrinking a Locally Managed Temporary Tablespace

Therefore, a single large sort operation might result in a large amount of allocated temporary space that remains unused after the sort operation is complete.

For this reason, the database enables you to shrink locally managed temporary tablespaces and release unused space.

Altering & Maintaining Tablespaces

Shrinking a Locally Managed Temporary Tablespace

```
ALTER TABLESPACE lmtemp1 SHRINK SPACE KEEP 20M;
```

The following example shrinks the temp file lmtemp02.dbf of the locally managed temporary tablespace lmtemp2.

```
ALTER TABLESPACE lmtemp2 SHRINK TEMPFILE
```

```
'u02/oracle/data/lmtemp02.dbf'
```

Introduction to Database Administration

TOPIC 78

Renaming & Dropping

Tablespaces

Renaming & Dropping Tablespaces

Renaming & Dropping Tablespaces

Renaming Tablespaces

Affects of Renaming Tablespaces

Dropping Tablespaces

Dropping Tablespaces - Example

Segments in a Tablespace

Renaming & Dropping Tablespaces

Renaming Tablespaces

Using the RENAME TO clause of the ALTER TABLESPACE, you can rename a permanent or temporary tablespace. For example, the following statement renames the users tablespace:

```
ALTER TABLESPACE users RENAME TO  
usersts;
```

Renaming & Dropping Tablespaces

Renaming Tablespaces

When you rename a tablespace the database updates all references to the tablespace name in the data dictionary, control file, and (online) data file headers.

To confirm use DBA_USERS.

Renaming & Dropping Tablespaces

Affects of Renaming Tablespaces

The COMPATIBLE parameter must be set to 10.0.0 or higher.

If the tablespace being renamed is the SYSTEM tablespace or the SYSAUX tablespace, then it will not be renamed.

If any data file or the tablespace is offline then the tablespace is not renamed.

Renaming & Dropping Tablespaces

Affects of Renaming Tablespaces

If the tablespace is read only, then data file headers are not updated. This should not be regarded as corruption; instead, it causes a message to be written to the alert log indicating that data file headers have not been renamed. The data dictionary and control file are updated.

Renaming & Dropping Tablespaces

Affects of Renaming Tablespaces

If the tablespace is the default temporary tablespace, then the corresponding entry in the database properties table is updated and the DATABASE_PROPERTIES view shows the new name.

Renaming & Dropping Tablespaces

Affects of Renaming Tablespaces

If the tablespace is an undo tablespace and if the following conditions are met, then the tablespace name is changed to the new tablespace name in the server parameter file (SPFILE). The server parameter file was used to start up the database.

The tablespace name is specified as the UNDO_TABLESPACE for any instance.

Renaming & Dropping Tablespaces

Dropping Tablespaces

You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. You must have the DROP TABLESPACE system privilege to drop a tablespace.

Renaming & Dropping Tablespaces

Dropping Tablespaces

When you drop a tablespace, the file pointers in the control file of the associated database are removed. You can optionally direct Oracle Database to delete the operating system files (data files) that constituted the dropped tablespace.

Off line tablespace before dropping it.

Renaming & Dropping Tablespaces

Dropping Tablespaces

You cannot drop a tablespace that contains any active segments.

For example, if a table in the tablespace is currently being used or the tablespace contains undo data needed to roll back uncommitted transactions, you cannot drop the tablespace.

Renaming & Dropping Tablespaces

Dropping Tablespaces

To drop a tablespace, use the DROP TABLESPACE statement. The following statements drops the users tablespace, including the segments in the tablespace:

Alter tablespace users offline; DROP TABLESPACE users INCLUDING CONTENTS;
Renaming & Dropping Tablespaces

Dropping Tablespaces

To delete the data files associated with a tablespace at the same time that the tablespace is dropped, use the INCLUDING CONTENTS AND DATAFILES clause. The following statement drops the users tablespace and its associated data files:

DROP TABLESPACE users INCLUDING CONTENTS AND DATAFILES;
Renaming & Dropping Tablespaces

Dropping Tablespaces - Example

```
create tablespace test
datafile 'E:\app\Haider\oradata\tst01.dbf'
size 100M; Tablespace created.
create user usr1 identified by xyz default tablespace test temporary tablespace temp;
```

```
grant connect, resource to usr1; grant create table to usr1;
```

Renaming & Dropping Tablespaces

Dropping Tablespaces - Example

```
alter tablespace test offline;
```

```
shutdown immediate startup open
```

```
DROP TABLESPACE test including contents;
```

Renaming & Dropping Tablespaces

Dropping Tablespaces - Example

```
conn usr1/xyz
SQL> create table AA
2 (id number); create table
*
```

```
ERROR at line 1:
ORA-00959: tablespace 'TEST' does not exist
```

Renaming & Dropping Tablespaces

Segments in a Tablespace

```
SQL> set linesize 200;
SQL> select segment_name, segment_type
       2  from dba_segments
       3  where tablespace_name='WRK_NOVUS';
```

SEGMENT_NAME	SEGMENT_TYPE
PATIENT	TABLE
PATIENT_VISIT	TABLE
PK_EMP_ID	INDEX
PK_PATVISIT_PAT_VSN0	INDEX
IX_PATVISIT_PID_VDATE	INDEX
IX_PATVISIT_PID_DPNSNO	INDEX
IX_PATVISIT_EMPID_VDATE	INDEX
IX_PATIENT_VISIT_CREATE_DATE	INDEX

```
8 rows selected.
```

End

Introduction to Database Administration

TOPIC 79

Managing SYSAUX Tablespaces

Managing SYSAUX Tablespaces

SYSAUX Tablespaces

- Introduction
- Monitoring Occupants
- Benefits of using SYSAUX Tablespace
- Controlling the Size of the SYSAUX Tablespace - Guidelines

Managing SYSAUX Tablespaces

Introduction

- The SYSAUX tablespace was installed as an auxiliary tablespace to the SYSTEM tablespace when you created your database.
 - Some database components that formerly created and used separate tablespaces now occupy the SYSAUX tablespace.
 - SYSAUX tablespace is used to reduce the load managed by SYSTEM tablespace.
- Managing SYSAUX Tablespaces

Introduction

- If the SYSAUX tablespace becomes unavailable, core database functionality will remain operational.
- The database features that use the SYSAUX tablespace could fail, or function with limited capability.
- SYSTEM tablespace will continue running for the users to work on if some problem occurs in SYSAUX tablespace.

Managing SYSAUX Tablespaces

Introduction

- SYSAUX tablespace mainly used for performance and comparison purposes

Managing SYSAUX Tablespaces

Monitoring Occupants

You can monitor the occupants of the SYSAUX tablespace using the V\$SYSAUX_OCCUPANTS view. This view lists the following information about the occupants of the SYSAUX tablespace:

- Name of the occupant
- Occupant description
- Schema name
- Move procedure
- Current space usage

Managing SYSAUX Tablespaces

Monitoring Occupants

Another major occupant of the SYSAUX tablespace is the embedded Oracle Enterprise Manager repository. This repository is used by Oracle Enterprise Manager Database Control to store its metadata. The size of this repository depends on database activity and on configuration-related information stored in the repository.

Managing SYSAUX Tablespaces

Benefits of using SYSAUX Tablespace

The SYSAUX tablespace is always created at database creation.

The SYSAUX tablespace serves as an auxiliary tablespace to the SYSTEM tablespace. Because it is the default tablespace for many Oracle Database features and products that previously required their own tablespaces, it reduces the number of tablespaces required by the database. It also reduces the load on the SYSTEM tablespace.

Managing SYSAUX Tablespaces

Controlling the Size of the SYSAUX Tablespace

The SYSAUX tablespace is occupied by several database components, and its total size is governed by the space consumed by those components.

The space consumed by the components, in turn, depends on which features or functionality are being used and on the nature of the database workload.

Managing SYSAUX Tablespaces

Controlling the Size of the SYSAUX Tablespace - Guidelines

Parameter/Recommendation	Small	Medium	Large
Number of CPUs	2	8	32
Number of concurrently active sessions	10	20	100
Number of user objects: tables and indexes	500	5,000	50,000
Estimated SYSAUX size at steady state with default configuration	500 MB	2 GB	5 GB

End

Introduction to Database Administration

TOPIC 80

Repairing Tablespaces

Repairing Tablespaces

Repairing Tablespaces

Repairing Locally Managed Tablespace

Diagnosing using DBMS_SPACE_ADMIN package

Scenario-1

Scenario-2



Scenario-3

Scenario-4

Scenario-5

Repairing Tablespaces

Repairing Locally Managed Tablespace

DROP_EMPTY_SEGMENTS	Drops segments from empty tables or table partitions and dependent objects
MATERIALIZER_DEFERRED_SEGMENTS	Materializes segments for tables and table partitions with deferred segment creation and their dependent objects.
SEGMENT_CORRUPT	Marks the segment corrupt or valid so that appropriate error recovery can be done
SEGMENT_DROP_CORRUPT	Drops a segment currently marked corrupt (without reclaiming space)

Oracle Database includes the DBMS_SPACE_ADMIN package, which is a collection of aids for diagnosing and repairing problems in locally managed tablespaces.

Following are the some of DBMS_SPACE_ADMIN Package Procedures

Repairing Tablespaces

Diagnosing using DBMS_SPACE_ADMIN package

The following scenarios describe typical situations in which you can use the DBMS_SPACE_ADMIN package to diagnose and resolve problems.

Note: Some of these procedures can result in lost and unrecoverable data if not used properly. You should work with Oracle Support Services if you have doubts about these procedures.

Repairing Tablespaces

Scenario-1

Fixing Bitmap When Allocated Blocks are Marked Free out these procedures.

The TABLESPACE_VERIFY procedure discovers that a segment has allocated blocks that are marked free in the bitmap, but no overlap between segments is reported.

In this scenario, perform the following tasks:

Call the SEGMENT_DUMP procedure to dump the ranges that the administrator allocated to the segment.

For each range, call the TABLESPACE_FIX_BITMAPS procedure with the TABLESPACE_EXTENT_MAKE_USED option to mark the space as used.

Call TABLESPACE_REBUILD_QUOTAS to rebuild quotas.

Repairing Tablespaces

Scenario-2

Dropping a corrupted segment:

In this scenario, perform the following tasks:

Call the SEGMENT_VERIFY procedure with the SEGMENT_VERIFY_EXTENTS_GLOBAL option. If no overlaps are reported, then proceed with steps 2 through 5.

Call the SEGMENT_DUMP procedure to dump the DBA ranges allocated to the segment.

For each range, call TABLESPACE_FIX_BITMAPS with the TABLESPACE_EXTENT_MAKE_FREE option to mark the space as free.

Call SEGMENT_DROP_CORRUPT to drop the SEG\$ entry.

Call TABLESPACE_REBUILD_QUOTAS to rebuild quotas

Repairing Tablespaces

Scenario-3

Fixing Bitmap Where Overlap is Reported:

The TABLESPACE_VERIFY procedure reports some overlapping. Some of the real data must be sacrificed based on previous internal errors. After choosing the object to be sacrificed, in this case say, table t1, perform the following tasks:

Make a list of all objects that t1 overlaps.

Drop table t1. If necessary, follow up by calling the SEGMENT_DROP_CORRUPT procedure.

Call the SEGMENT_VERIFY procedure on all objects that t1 overlapped. If necessary, call the TABLESPACE_FIX_BITMAPS procedure to mark appropriate bitmap blocks as used.

Rerun the TABLESPACE_VERIFY procedure to verify that the problem is resolved

Repairing Tablespaces

Scenario-4

Correcting Media Corruption of Bitmap Blocks A set of bitmap blocks has media corruption.

In this scenario, perform the following tasks:

Call the TABLESPACE_REBUILD_BITMAPS procedure, either on all bitmap blocks, or on a single block if only one is corrupt.

Call the TABLESPACE_REBUILD_QUOTAS procedure to rebuild quotas.

Call the TABLESPACE_VERIFY procedure to verify that the bitmaps are consistent.

Repairing Tablespaces

Scenario-5

Assume that the database block size is 2K and the existing extent sizes in tablespace tbs_1 are 10, 50, and 10,000 blocks (used, used, and free). The MINIMUM EXTENT value is 20K (10 blocks). Allow the system to choose the bitmap allocation unit. The value of 10 blocks is chosen, because it is the highest common denominator and does not exceed MINIMUM EXTENT.

The statement to convert tbs_1 to a locally managed tablespace is as follows: EXEC DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL ('tbs_1');

If you choose to specify an allocation unit size, it must be a factor of the unit size calculated by the system.

Introduction to Database Administration

TOPIC 81

Migrating the System Tablespace

Migrating the System Tablespaces

Migrating the System Tablespace

Introduction

Other tablespaces

Pre-requisites of migration

Command for migration

Importing and exporting data of tablespaces

Migrating the System Tablespaces

Introduction

What is SYSTEM tablespace

It is required when database is created and when database is getting start using.

Pre bundled code and data dictionary-managed tablespace.
It contains System Objects, your objects and Users/ Privileges.
It contains AWR: Automatic Workload Repository
Migrating the System Tablespaces

Other Tablespaces

Following tablespaces are required with the SYSTEM tablespace
UNDO TS

User ability to rollback
old values of DML changes
Read Consistency of Data
Temporary TS
contents are temp stored in TS
order by, distinct, Index --> PGA is active
Kind rough work area
Migrating the System Tablespaces

Other Tablespaces

USERS TS
Create users, which users will use which TS information
Migrating the System Tablespaces

Pre-requisites of Migration

Before performing the migration the following conditions must be met:
The database has a default temporary tablespace that is not SYSTEM.
There are no rollback segments in the dictionary-managed tablespace.
There is at least one online rollback segment in a locally managed tablespace, or if using automatic undo management, an undo tablespace is online.
Migrating the System Tablespaces

Pre-requisites of Migration

All tablespaces other than the tablespace containing the undo space (that is, the tablespace containing the rollback segment or the undo tablespace) are in read- only mode.
The SYSAUX tablespace is offline.
The system is in restricted mode.
There is a cold backup of the database.
Migrating the System Tablespaces

Command for Migration

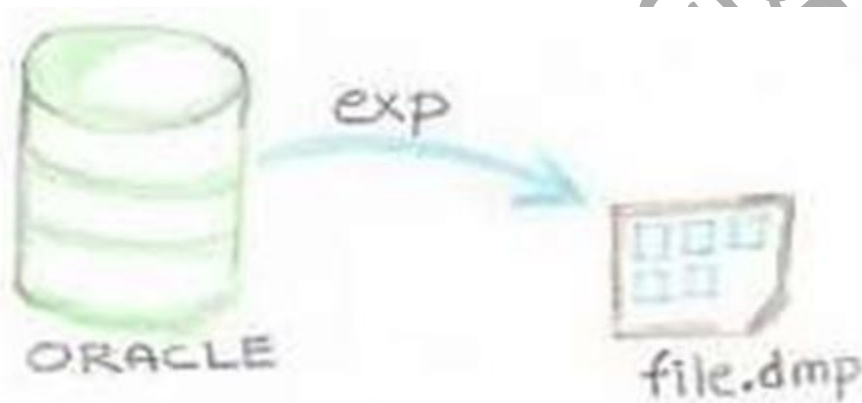
The following statement performs the migration:

```
SQL> EXECUTE  
DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE  
_TO_LOCAL('SYSTEM');  
Migrating the System Tablespaces
```

Importing & Exporting Data of Tablespaces

Use following command of export on source database

```
C:\>exp usr/pwd file=c:\test\fl.dmp
```



Migrating the System Tablespaces

Importing & Exporting Data of Tablespaces

Following are two tablespaces of source database. We will create sure same tablespaces in the target tablespace.

```
Create tablespace MedCare Datafile  
'D:\oracle9i\ora92\oradata\MCare01.dbf' Size 150M online  
MINIMUM EXTENT 64K  
DEFAULT STORAGE (INITIAL 64K NEXT 64K);
```

```
CREATE TABLESPACE MCare_indx  
DATAFILE 'D:\oracle9i\ora92\oradata\MCare_indx01.dbf' SIZE 100M  
MINIMUM EXTENT 64K  
DEFAULT STORAGE (INITIAL 64K NEXT 64K);
```

Use following command of import on target database

C:\>exp usr/pwd file=c:\test\f1.dmp

Migrating the System Tablespaces

Importing & Exporting Data of Tablespaces

```
E:\app\Haider\product\11.2.0\dbhome_1\BIN>expdp

Export: Release 11.2.0.1.0 - Production on Wed Aug 12 11:02:54 2020

Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.

Username: scott/tiger

Connected to: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0
With the Partitioning, OLAP, Data Mining and Real Application Testing
Starting "SCOTT"."SYS_EXPORT_SCHEMA_01": scott/*****
Estimate in progress using BLOCKS method...
. . exported "SCOTT"."STAGE_MIGRLOG"                                0 KB
Master table "SCOTT"."SYS_EXPORT_SCHEMA_01" successfully loaded/unloaded
*****
Dump file set for SCOTT.SYS_EXPORT_SCHEMA_01 is:
  E:\APP\HAIDER\ADMIN\ORCL\DPDUMP\EXPDAT.DMP
Job "SCOTT"."SYS_EXPORT_SCHEMA_01" successfully completed at 11:03:47
```

Alternatively, use export/ import data pump command. For example,

End

Introduction to Database Administration

TOPIC 82

Creating Alternate TEMP
Tablespace

Creating Alternate TEMP Tablespace

Creating Alternate TEMP Tablespace

What is Temporary Tablespace?

Creating TEMP Tablespace

Verifying TEMP Tablespace

Create Temporary Tablespace Group

Temporary Tablespace Files

Creating Alternate TEMP Tablespace

What is TEMP Tablespace?

What is SYSTEM tablespace

It is required when database is created and when database is getting start using.

Creating Alternate TEMP Tablespace

Creating TEMP Tablespace

Create a new tablespace and then drop an existing temp tablespace

```
CREATE  
TEMPORARY TABLESPACE "TEMP_DEFAULT2" TEMPFILE  
'D:\ORACLE9I\ORA92\ORADATA\MEDCARE\TEMP_DEFAULT2.DBF' SIZE  
1000M EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;
```

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE "TEMP_DEFAULT2";
```

Creating Alternate TEMP Tablespace

Creating TEMP Tablespace

shutdown and startup database,

run complicated query to see whether new tablespace getting filled or not

see file alert.log for more details

To check default tablespace is created or DONE or NOT:

```
select * from database_properties;
```

Creating Alternate TEMP Tablespace

Creating TEMP Tablespace

Tablespace must be empty before dropping off

Drop tablespace temp
including contents and datafiles;

Shutdown and startup database again

Drop related files from folder like TEMP01.DBF

Creating Alternate TEMP Tablespace

Verifying TEMP Tablespace

```
SQL> select property_name from database_properties;

PROPERTY_NAME
-----
DICT.BASE
DEFAULT_TEMP_TABLESPACE
DEFAULT_PERMANENT_TABLESPACE
DEFAULT_EDITION
Flashback Timestamp TimeZone
TDE_MASTER_KEY_ID
DST_UPGRADE_STATE
DST_PRIMARY_TT_VERSION
DST_SECONDARY_TT_VERSION
DEFAULT_TBS_TYPE
NLS_LANGUAGE
```

Creating Alternate TEMP Tablespace

Create Temporary Tablespace Group

Temporary Tablespace Groups allow an Oracle database to write to multiple temp tablespaces simultaneously.

A user is assigned to a group of temp tablespaces instead of a single temporary tablespace

It allows a single SQL operation to use multiple Temporary Tablespaces for sorting

Creating Alternate TEMP Tablespace

Create Temporary Tablespace Group

Create Temporary Tablespace temp1 tempfile 'C:\Oracle\Oradata\temp01.dbf' size 50M

```
SQL> select *
  2  from database_properties
  3  where property_name like '%TABLESPACE%';

PROPERTY_NAME
-----
PROPERTY_VALUE
-----
DESCRIPTION
-----
DEFAULT_TEMP_TABLESPACE
TEMP
Name of default temporary tablespace
DEFAULT_PERMANENT_TABLESPACE
USERS
Name of default permanent tablespace
```

tablespace group tempgroup1;

Creating Alternate TEMP Tablespace

Temporary Tablespace Files

Using data dictionary view;

```
SQL> select tablespace_name
  2  from dba_temp_files;

TABLESPACE_NAME
-----
TEMP
```

End

Introduction to Database Administration

TOPIC 83

Viewing Tablespace using Data Dictionary Views

Viewing Tablespaces using Data Dictionary Views

Data Dictionary Views of Tablespace

Dictionary Views of Tablespaces

V\$Tablespaces

DBA_DATA_FILES

DBA_TEMP_FILES

Finding Used Size of Tablespaces

Finding Free Space of Tablespaces

Viewing Tablespaces using Data Dictionary Views

Dictionary Views of Tablespaces

Use Dictionary to display views:

```

SQL> select table_name
      2  from dict
      3  where table_name like '%TABLESPACE%';

TABLE_NAME
-----
DBA_FILE_GROUP_TABLESPACES
DBA_HIST_TABLESPACE_STAT
DBA_TABLESPACES
DBA_TABLESPACE_GROUPS
DBA_TABLESPACE_THRESHOLDS
DBA_TABLESPACE_USAGE_METRICS
USER_FILE_GROUP_TABLESPACES
USER_TABLESPACES
ALL_FILE_GROUP_TABLESPACES
V$TABLESPACE
GV$ENCRYPTED_TABLESPACES

TABLE_NAME
-----
V$ENCRYPTED_TABLESPACES
GV$TABLESPACE

13 rows selected.

```

Viewing Tablespaces using Data Dictionary Views

V\$Tablespaces

To display columns of V\$Tablespaces:

```

SQL> set linesize60
SQL> desc v$tablespace

Name                                         Null?     Type
-----
TS#                                           NUMBER
NAME                                         VARCHAR2(30)
INCLUDED_IN_DATABASE_BACKUP                 VARCHAR2(3)
BIGFILE                                     VARCHAR2(3)
FLASHBACK_ON                               VARCHAR2(3)
ENCRYPT_IN_BACKUP                           VARCHAR2(3)

```

Viewing Tablespaces using Data Dictionary Views

V\$Tablespaces

Select column name V\$Tablespaces:

```
SQL> select name
      2   from v$tablespace;

NAME
-----
SYSTEM
SYSAUX
UNDOTBS1
USERS
TEMP
IND_TSMYDEV
TSNDEV
IND_TSNDEV
PTSDEV
PIND_TSDEV
PTSIMG

NAME
-----
PTS_STUCRSREG
PTS_EMARKS
TSIMG
WRK_NOVUS

15 rows selected.
```

Viewing Tablespaces using Data Dictionary Views

DBA_DATA_FILES

Display name and data file size in MB:

```

SQL> select tablespace_name, file_name,
2 bytes/1024/1024 size_MB
3 from dba_data_files
4 where tablespace_name='&tablespace_name';
Enter value for tablespace_name: WRK_NOVUS
old 4: where tablespace_name='&tablespace_name'
new 4: where tablespace_name='WRK_NOVUS'

TABLESPACE_NAME
-----
FILE_NAME
-----
SIZE_MB
-----
WRK_NOVUS
E:\APP\HAIDER\PRODUCT\11.2.0\DBHOME_1\ORADATA\NOVUS01.DBF
800

```

Viewing Tablespaces using Data Dictionary Views

DBA_TEMP_FILES

Display name and data file size in MB:

```

SQL> select file_name,
2 bytes/1024/1024 size_MB
3 from dba_temp_files;

FILE_NAME
-----
SIZE_MB
-----
E:\APP\HAIDER\ORADATA\ORCL\TEMP01.DBF
114

```

TEMP01.DBF	13-Aug-20...	DBF File	116,744 KB
------------	--------------	----------	------------

Viewing Tablespaces using Data Dictionary Views

Finding Used Data Size of Tablespaces

```

SQL> SELECT TABLESPACE_NAME,
2      ROUND(SUM(BYTES)/(1024*1024), 2)||' MB' AS USED_SIZE
3      FROM DBA_SEGMENTS
4      --WHERE TABLESPACE_NAME='EDRDBA'
5      GROUP BY TABLESPACE_NAME;

```

TABLESPACE_NAME	USED_SIZE
PIND_TSDEV	197.31 MB
SYSAUX	648.5 MB
TSIMG	1120.5 MB
UNDOTBS1	21.69 MB
TSNDEV	643.56 MB
PTS_EMARKS	48 MB
WRK_NOVUS	703.75 MB
USERS	4.19 MB
SYSTEM	707.63 MB
PTSDEV	364.63 MB
PTSIMG	62 MB
PTS_STUCRSREG	480 MB
IND_TSNDEV	.06 MB

13 rows selected.

Using data dictionary view DBA_SEGMENTS :
Viewing Tablespaces using Data Dictionary Views

Finding Free Space of Tablespaces

```

SQL> SELECT TABLESPACE_NAME,
2      ROUND(SUM(BYTES)/(1024*1024), 2)||' MB' AS FREE_SIZE
3      FROM DBA_FREE_SPACE
4      GROUP BY TABLESPACE_NAME;

```

TABLESPACE_NAME	FREE_SIZE
IND_TSMYDEV	999 MB
PIND_TSDEV	301.75 MB
SYSAUX	50.5 MB
UNDOTBS1	282.31 MB
TSIMG	925.5 MB
TSNDEV	531.44 MB
PTS_EMARKS	1950 MB
WRK_NOVUS	367.25 MB
USERS	3.56 MB
SYSTEM	1.38 MB
IND_TSNDEV	498.94 MB
PTSDEV	2704.5 MB
PTSIMG	4030 MB
PTS_STUCRSREG	1518 MB

14 rows selected.

Using data dictionary view DBA_FREE_SPACE

End

Introduction to Database Administration

TOPIC 84

Logical Storage Structure & Hierarchy

Logical Storage Structure & Hierarchy

Logical Storage Structure & Hierarchy

Physical & Logical Database

Tablespace

Database Block

Database Block, Extents and Segments

Database Block, Extents and Segments within Database

Logical Storage Structure & Hierarchy

Physical & Logical Database

Physical Database

Control Files

Redo Log Files

Data Files

Logical Database

Tablespace

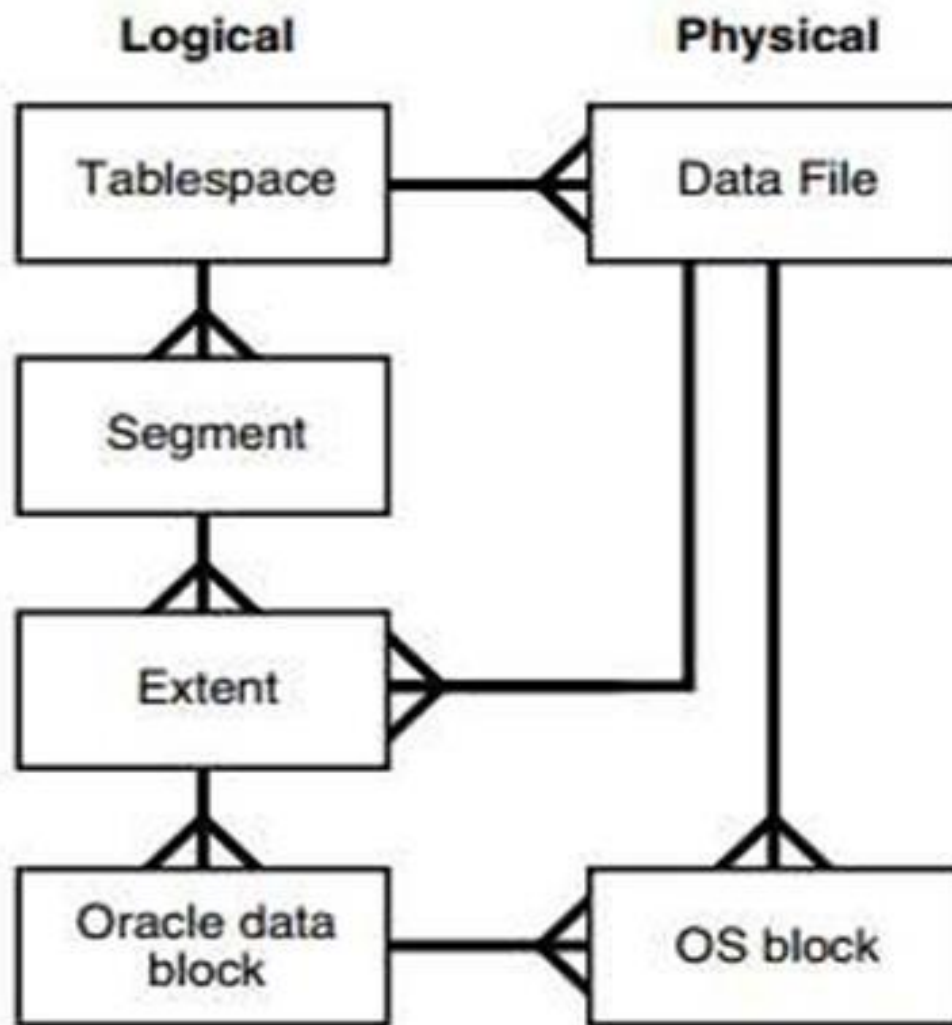
Segments

Extents

DB Blocks

Logical Storage Structure & Hierarchy

Physical & Logical Database



Logical Storage Structure & Hierarchy

Tablespace

Tablespace stores space logically to create logical objects such as Tables, Indexes and other objects.

Tablespace's physically space stored in associated files.

Logical Storage Structure & Hierarchy

Database Block

Logical minimum storage unit is called database block.

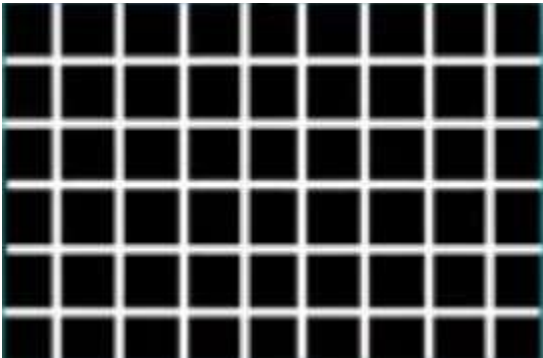
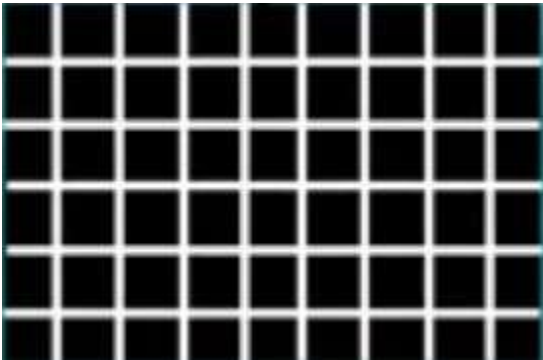
Its size is decided during database creation.

5 default sizes: 2K, 4K, 8K, 16K, 32K (K: Kilobytes)

Logical Storage Structure & Hierarchy

Database Block, Extents and Segments

Data Files



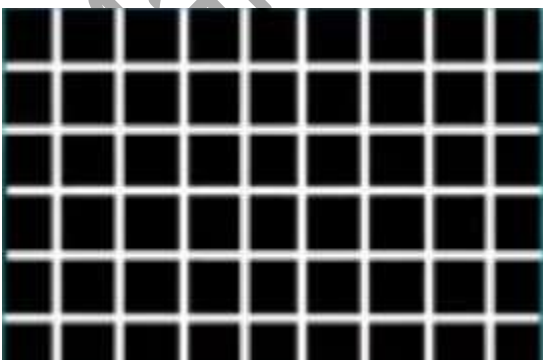
Database blocks

Database Blocks

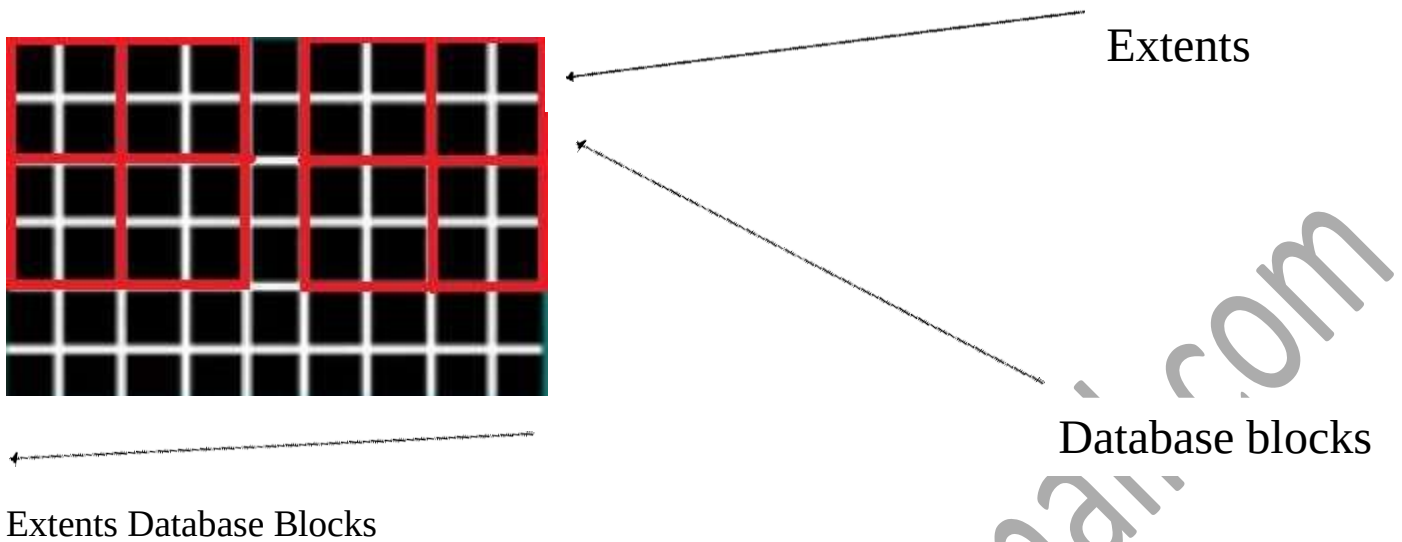
OS Blocks

Logical Storage Structure & Hierarchy

Database Block, Extents and Segments

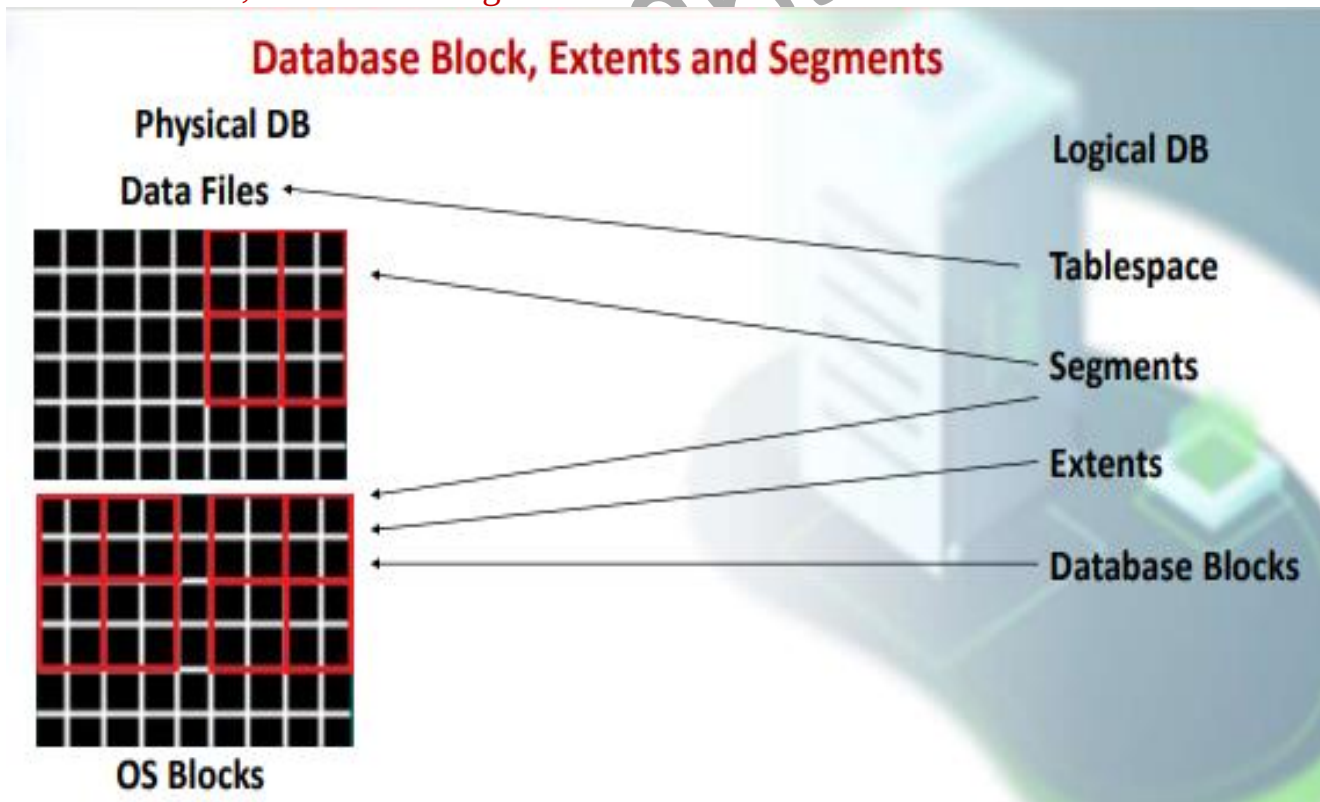


Data Files



OS Blocks
Logical Storage Structure & Hierarchy

Database Block, Extents and Segments



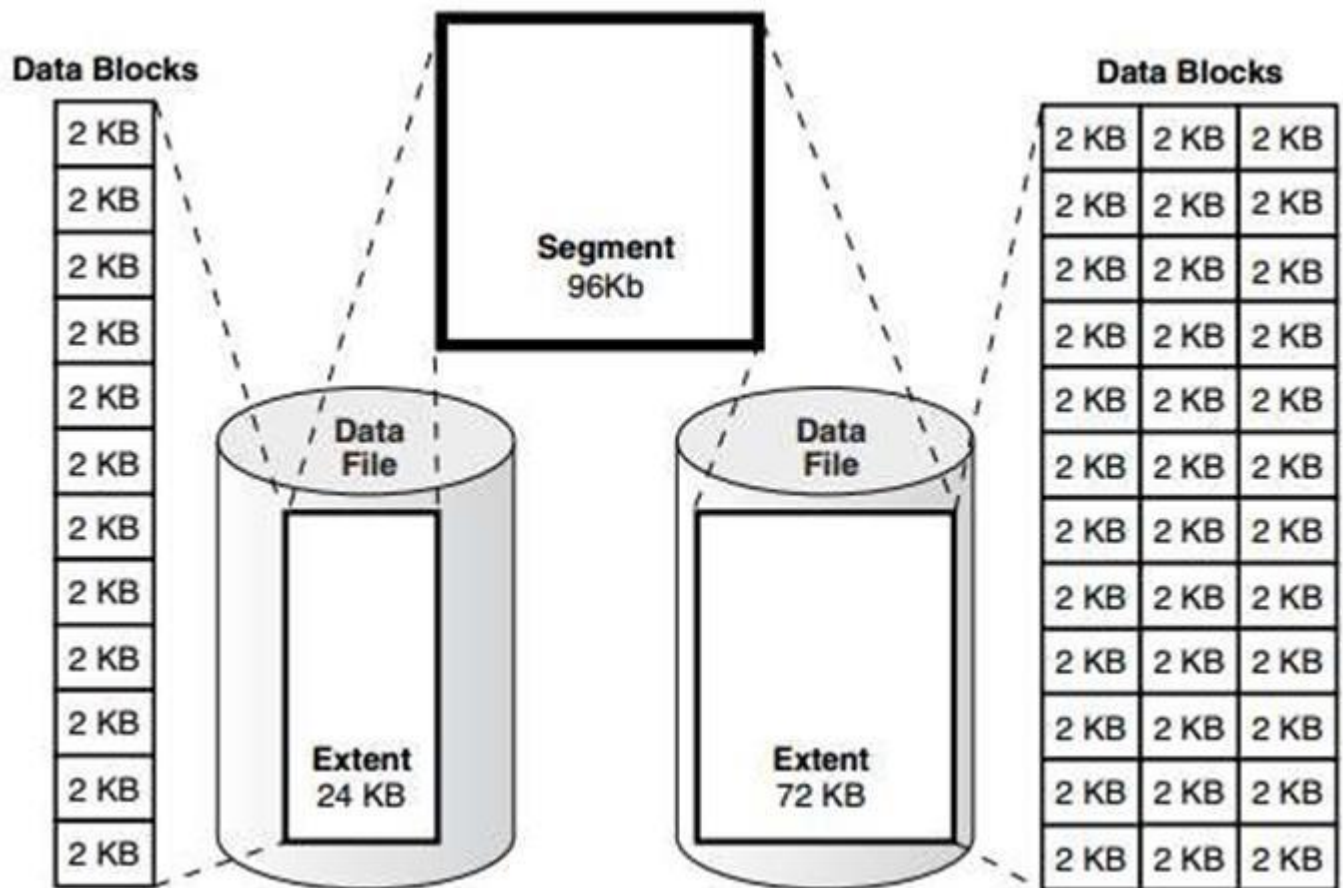
Database Block, Extents and Segments within Database

At the finest level of granularity, Oracle Database stores data in data blocks. One logical data block corresponds to a specific number of bytes of physical disk space. Block sizes can be 2 KB, 4KB or more.

Data block is the smallest unit of storage that Oracle Database can use or allocate.

Logical Storage Structure & Hierarchy

Database Block, Extents and Segments within Database



Introduction to Database Administration

TOPIC 85

Overview of Data Blocks

Overview of Data Blocks

Overview of Database Blocks

What is Data Block?

Data Blocks & OS Blocks

Interaction of Data Blocks & OS Blocks

Database Block Size

Data Block Format

Row Format

RowID Pseudocolumn

Overview of Data Blocks

What is Data Block?

Oracle Database manages the logical storage space in the data files of a database in units called data blocks, also called Oracle blocks or pages. A data block is the minimum unit of database I/O.

Overview of Data Blocks

Data Blocks & OS Blocks

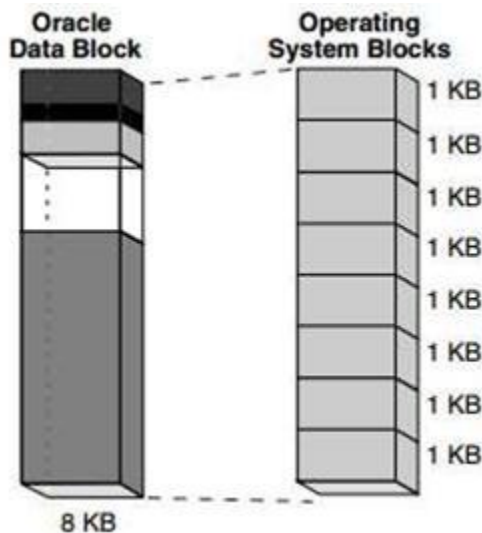
At the physical level, database data is stored in disk files made up of operating system blocks. An operating system block is the minimum unit of data that the operating system can read or write.

In contrast, an Oracle block is a logical storage structure whose size and structure are not known to the operating system.

Overview of Data Blocks

Data Blocks & OS Blocks

Figure shows that operating system blocks may differ in size from data blocks. The database requests data in multiples of data blocks, not operating system blocks.



Overview of Data Blocks

Interaction of Data Blocks & OS Blocks

When the database requests a data block, the operating system translates this operation into a requests for data in permanent storage. The logical separation of data blocks from operating system blocks has the following implications:

Applications do not need to determine the physical addresses of data on disk.

Database data can be striped or mirrored on multiple physical disks.

Overview of Data Blocks

Database Block Size

Every database has a database block size. The `DB_BLOCK_SIZE` initialization parameter sets the data block size for a database when it is created.

The size is set for the `SYSTEM` and `SYSAUX` tablespaces and is the default for all other tablespaces.

The database block size cannot be changed except by re-creating the database.

Overview of Data Blocks

Database Block Size

If `DB_BLOCK_SIZE` is not set, then the default data block size is operating system-specific.

The standard data block size for a database is 4 KB or 8 KB.

If the size differs for data blocks and operating system blocks, then the data block size must be a multiple of the operating system block size

Overview of Data Blocks

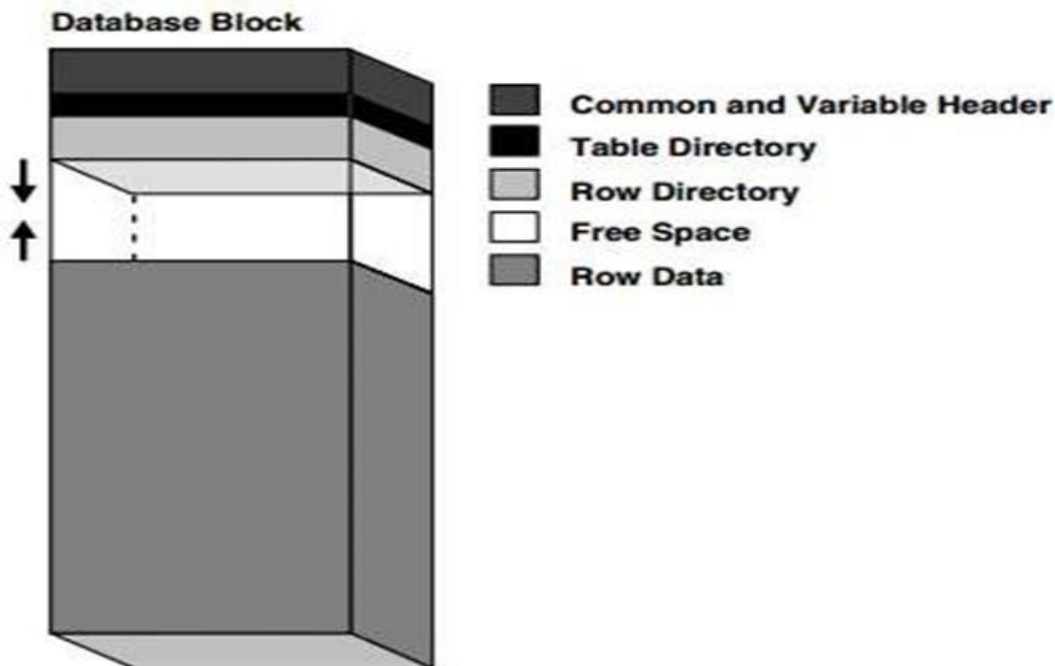
Database Block Size

You can create individual tablespaces whose block size differs from the DB_BLOCK_SIZE setting.

A nonstandard block size can be useful when moving a transportable tablespace to a different platform.

Overview of Data Blocks

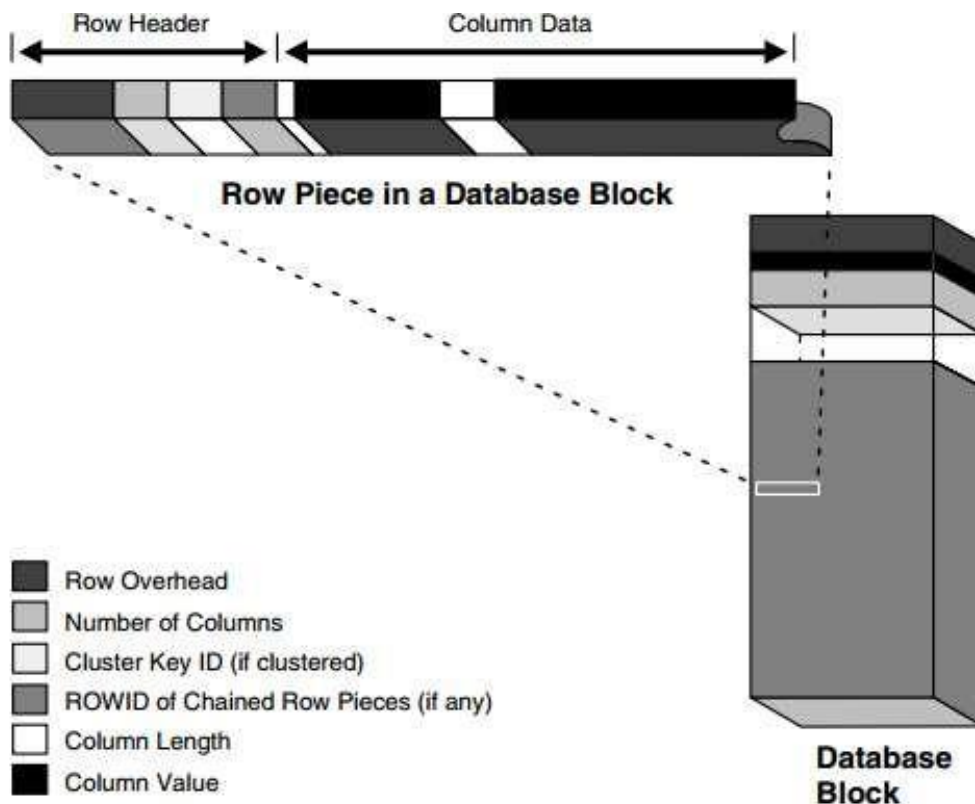
Data Block Format



Every data block has a format or internal structure that enables the database to track the data and free space in the block. This format is similar whether the data block contains table, index, or table cluster data.

Overview of Data Blocks

Row Format



The row data part of the block contains the actual data, such as table rows or index key entries. Just as every data block has an internal format, every row has a row format that enables the database to track the data in the row.

Overview of Data Blocks

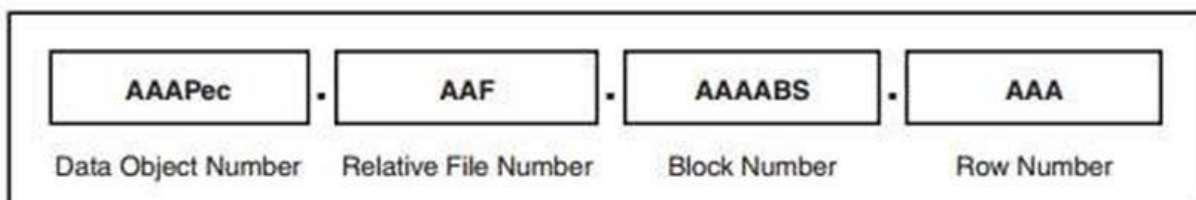
RowID Pseudocolumn

```
SQL> SELECT ROWID FROM employees WHERE employee_id = 100;
```

```
ROWID
-----
AAAPecAAFAAAABSAAA
```

Figure 12–8 illustrates the format of an extended rowid.

Figure 12–8 ROWID Format



Queries the ROWID pseudocolumn to show the extended rowid of the row in the employee stable for employee 100

Introduction to Database Administration

TOPIC 86

Percentage of Free Space in Blocks (PCTFREE)

Percentage of Free Space in Blocks

Percentage of Free Space Blocks

What is Free Space in Data Block?

Data Blocks & OS Blocks

Percentage of Free Space in Data Blocks

PCTFREE Space in Data Block

Optimizing of Free Space

Reuse of Index Space

Row Migration

Percentage of Free Space in Blocks

What is Free Space in Data Block?

As the database fills a data block from the bottom up, the amount of free space between the row data and the block header decreases. This free space can also shrink during updates, as when changing a trailing null to a non-null value.

The database manages free space in the data block to optimize performance and avoid wasted space (**Tuning Feature**).

Percentage of Free Space in Blocks

Data Blocks & OS Blocks

At the physical level, database data is stored in disk files made up of operating system blocks. An operating system block is the minimum unit of data that the operating system can read or write.

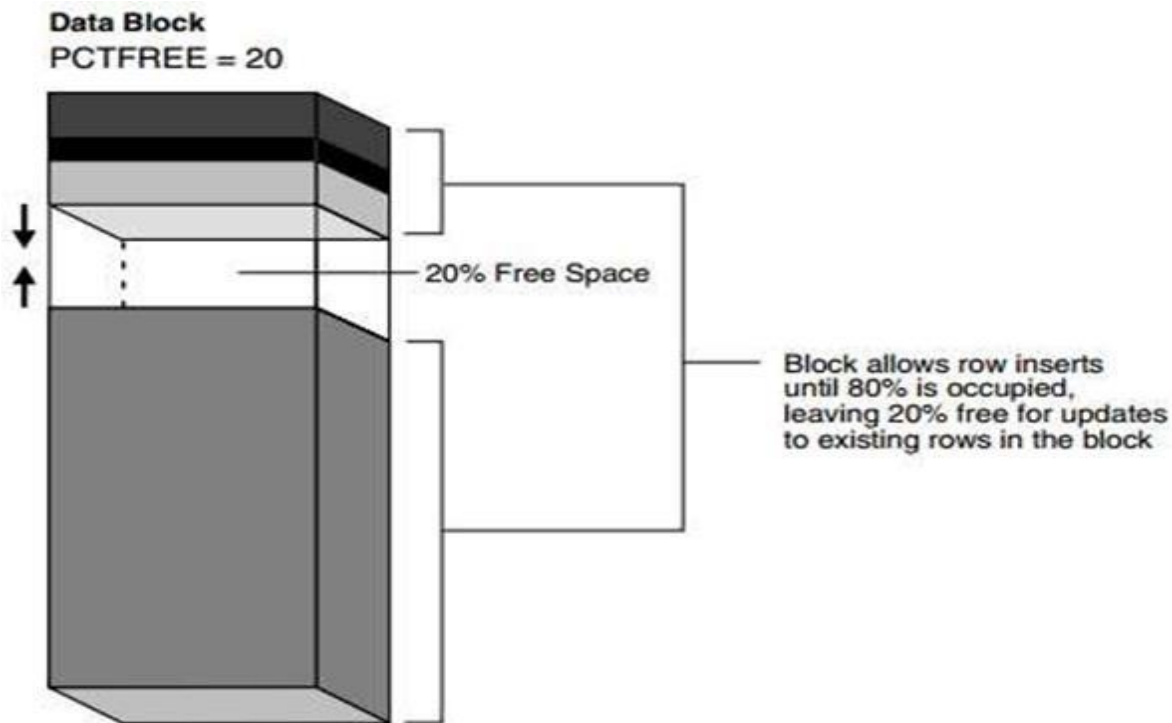
In contrast, an Oracle block is a logical storage structure whose size and structure are not known to the operating system.

Percentage of Free Space in Blocks

Percentage of Free Space in Data Blocks

The **PCTFREE** storage parameter is essential to how the database manages free space. This SQL parameter sets the minimum percentage of a data block reserved as free space for updates to existing rows.

PCTFREE is important for preventing row migration and avoiding wasted space.
Percentage of Free Space in Blocks



PCTFREE Space in Data Block

```
CREATE TABLESPACE userdata datafile 'C:\Ora11g\oradata\usr01.dbf' size 500M  
EXTENT MANAGEMENT DICTIONARY  
DEFAULT STORAGE  
(Initial 1M NEXT 1M PCTINCREASE 0);
```

PCTFREE parameter does not applied on LOB Datatypes
Percentage of Free Space in Blocks

Optimizing of Free Space

While the percentage of free space cannot be less than PCTFREE, the amount of free space can be greater. For example, a PCTFREE setting of 20% prevents the total amount of free space from dropping to 5% of the block, but permits 50% of the block to be free space.

Percentage of Free Space in Blocks

Optimizing of Free Space

The following SQL statements can increase free space:

DELETE statements

UPDATE statements that either update existing values to smaller values or increase existing values and force a row to migrate

INSERT statements on a table that uses OLTP compression. If inserts fill a block with data, then the database invokes block compression, which may result in the block having more free space.

Percentage of Free Space in Blocks

Optimizing of Free Space

The space released is available for INSERT under the following conditions:

If the INSERT statement is in the same transaction and after the statement that frees space, then the statement can use the space.

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

Percentage of Free Space in Blocks

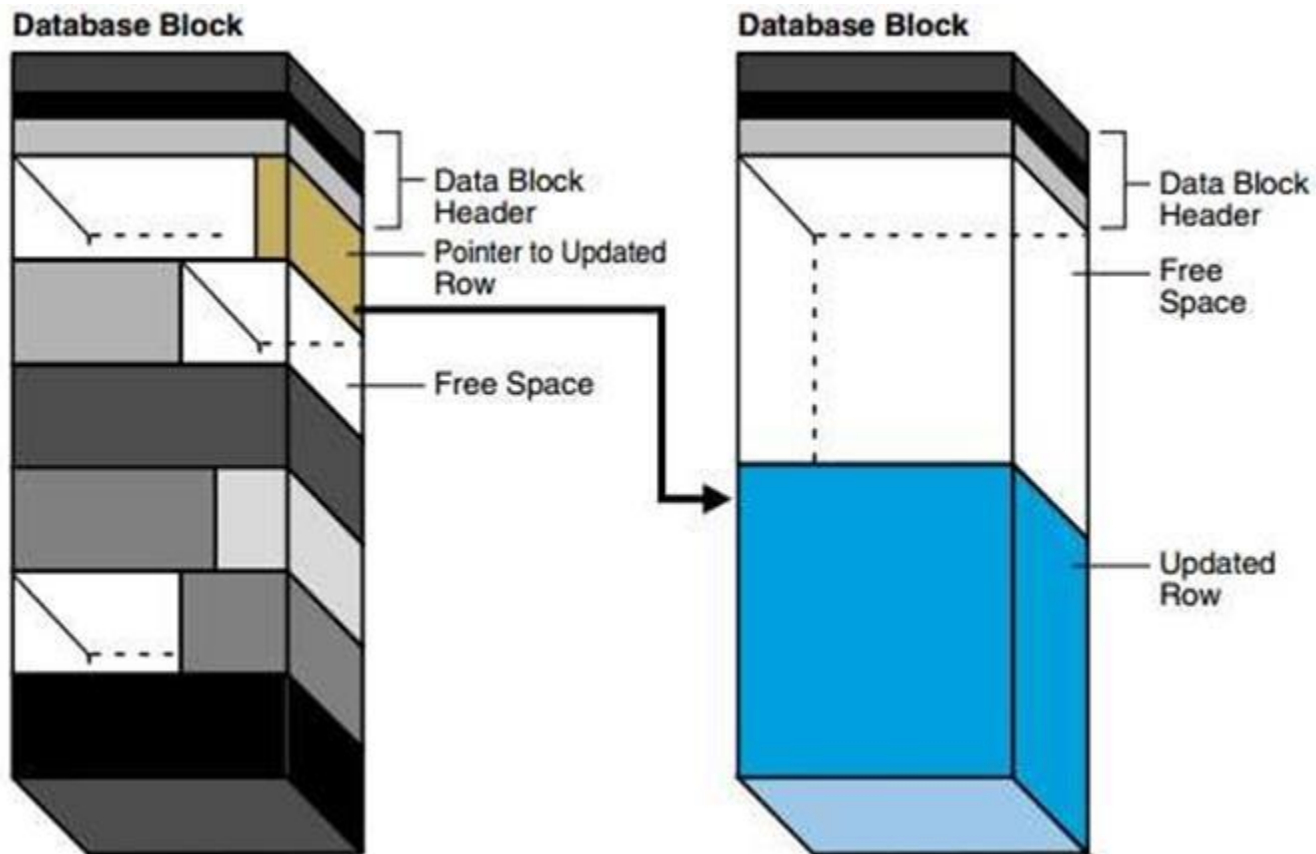
Reuse of Index Space

The database can reuse space within an index block.

For example, if you insert a value into a column and delete it, and if an index exists on this column, then the database can reuse the index slot when a row requires it

Percentage of Free Space in Blocks

Row Migration



The left block contains a row that is updated so that the row is now too large for the block. The database moves the entire row to the right block and leaves a pointer to the migrated row in the left block

Introduction to Database Administration

TOPIC 87

Overview of Extents

Overview of Extents

Overview of Extents

What is Extent?

Incremental Extents of Segment

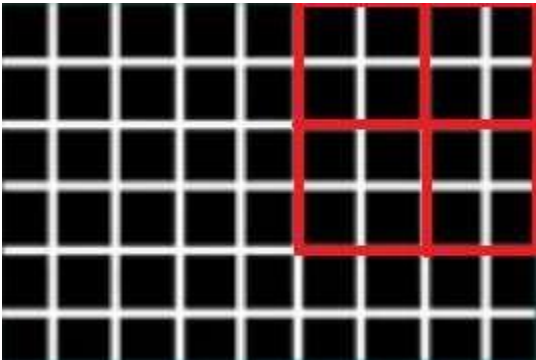
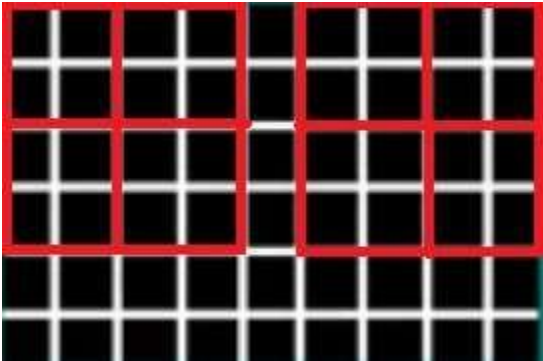
Deallocation of Extents

Manually Deallocate Space

Sample SQL DDL using Extent

Overview of Extents

What is Extent?



An extent is a logical unit of database storage space allocation made up of contiguous data blocks. Data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.

Data Files

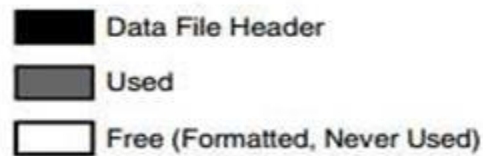
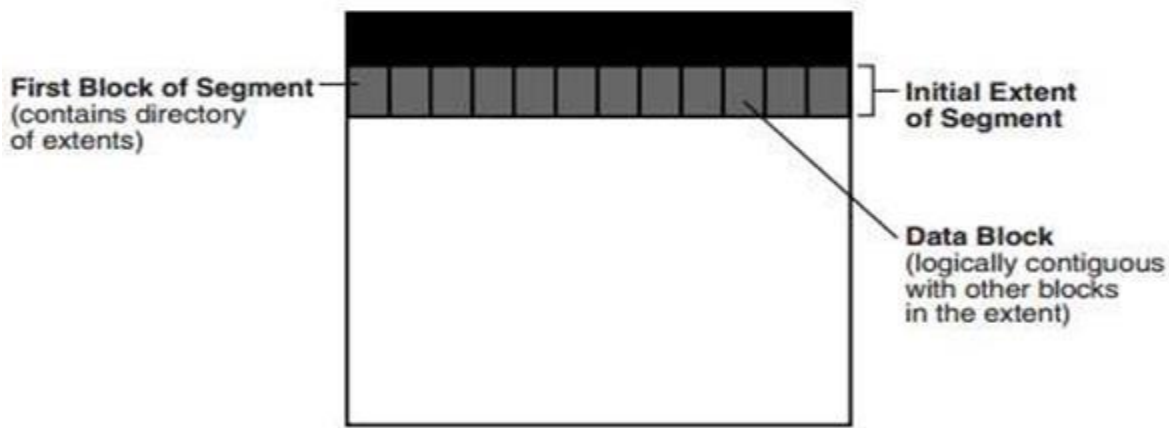
Overview of Extents

What is Extent?

An extent is a logical unit of database storage space allocation made up of contiguous data blocks. Data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.

Overview of Extents

What is Extents?



By default, the database allocates an initial extent for a data segment when the segment is created. An extent is always contained in one data file.

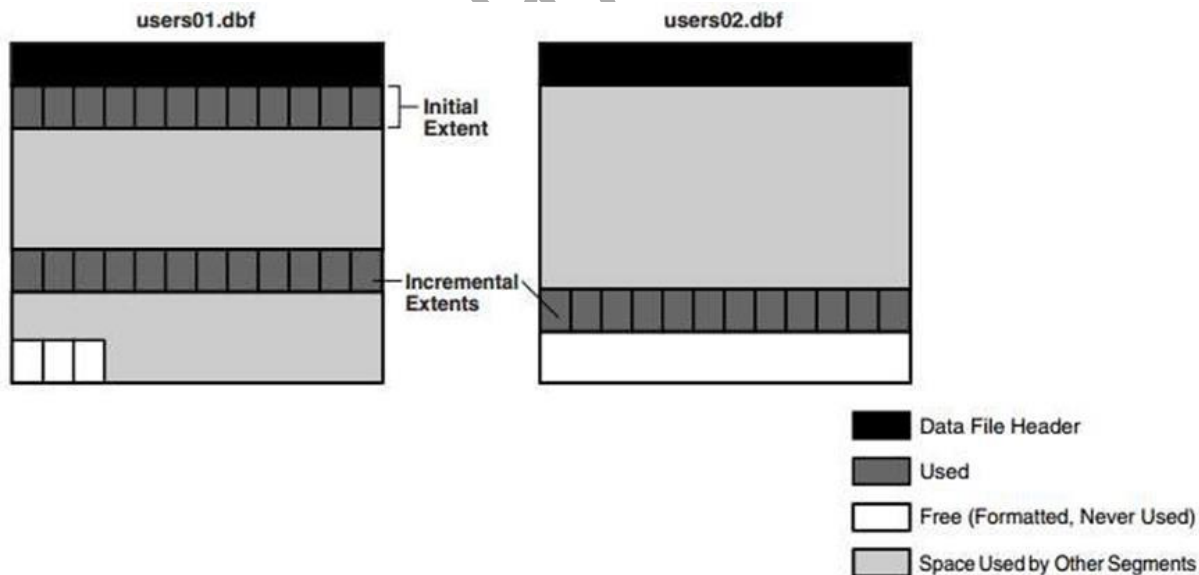
Although no data has been added to the segment, the data blocks in the initial extent are reserved for this segment exclusively.

The first data block of every segment contains a directory of the extents in the segment.

Figure shows the initial extent in a segment in a data file that previously contained no data.

Overview of Extents

Incremental Extents of Segment



Overview of Extents

Deallocation of Extents

In general, the extents of a user segment do not return to the tablespace unless you drop the object using a DROP command.

In Oracle Database 11g Release 2, you can also drop the segment using the DBMS_SPACE_ADMIN package.

For example, if you delete all rows in a table, then the database does not reclaim the data blocks for use by other objects in the tablespace.

Overview of Extents

Manually Deallocate Space

Following techniques can deallocate extents

You can use an online segment shrink to reclaim fragmented space in a segment. Segment shrink is an online, in-place operation.

You can move the data of a non partitioned table or table partition into a new segment, and optionally into a different tablespace for which you have quota.

Overview of Extents

Manually Deallocate Space

You can rebuild or coalesce the index

You can truncate a table or table cluster, which removes all rows. By default, Oracle Database deallocates all space used by the removed rows except that specified by the MINEXTENTS storage parameter.

You can deallocate unused space, which frees the unused space at the high water mark end of the database segment and makes the space available for other segments in the tablespace

Overview of Extents

Sample SQL DDL using Extent

```
CREATE TABLESPACE MCare_idx  
DATAFILE 'D:\oracle11g\ora11g\oradata\MCare_idx01.dbf'  
SIZE 100M  
MINIMUM EXTENT 64K  
DEFAULT STORAGE (INITIAL 64K NEXT 64K);
```

Introduction to Database Administration

TOPIC 88

Overview of Segments

Overview of Segments

Overview of Segments

What is Segment?

Segment & Data Files

User Segments

User Segments Creation

Creation of Multiple Segments

SQL DDL for Segments under Tablespace

Temporary Segments

UNDO Segments

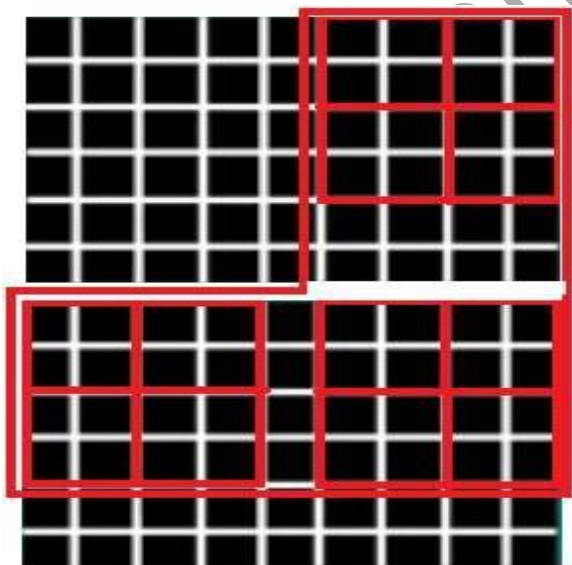
Tablespace with Segments

Overview of Segments

What is Segment?

A segment is a set of extents that contains all the data for a logical storage structure within a tablespace. For example, Oracle Database allocates one or more extents to form the data segment for a table. The database also allocates one or more extents to form the index segment for a table implementations.

Overview of Segments



Segment & Data Files

Data Files

For example, Patient table consists of three segments from two data files

Overview of Segments

User Segments

A single data segment in a database stores the data for one user object. There are different types of segments. Examples of user segments include:

Table, table partition, or table cluster

LOB or LOB partition

Index or index partition

Each non partitioned object and object partition is stored in its own segment. For example, if an index has five partitions, then five segments contain the index data.

Overview of Segments

User Segments Creation

By default, the database uses deferred segment creation to update only database metadata when creating tables and indexes.

When a user inserts the first row into a table or partition, the database creates segments for the table or partition, its LOB columns, and its indexes.

Deferred segment creation enables you to avoid using database resources unnecessarily.

Overview of Segments

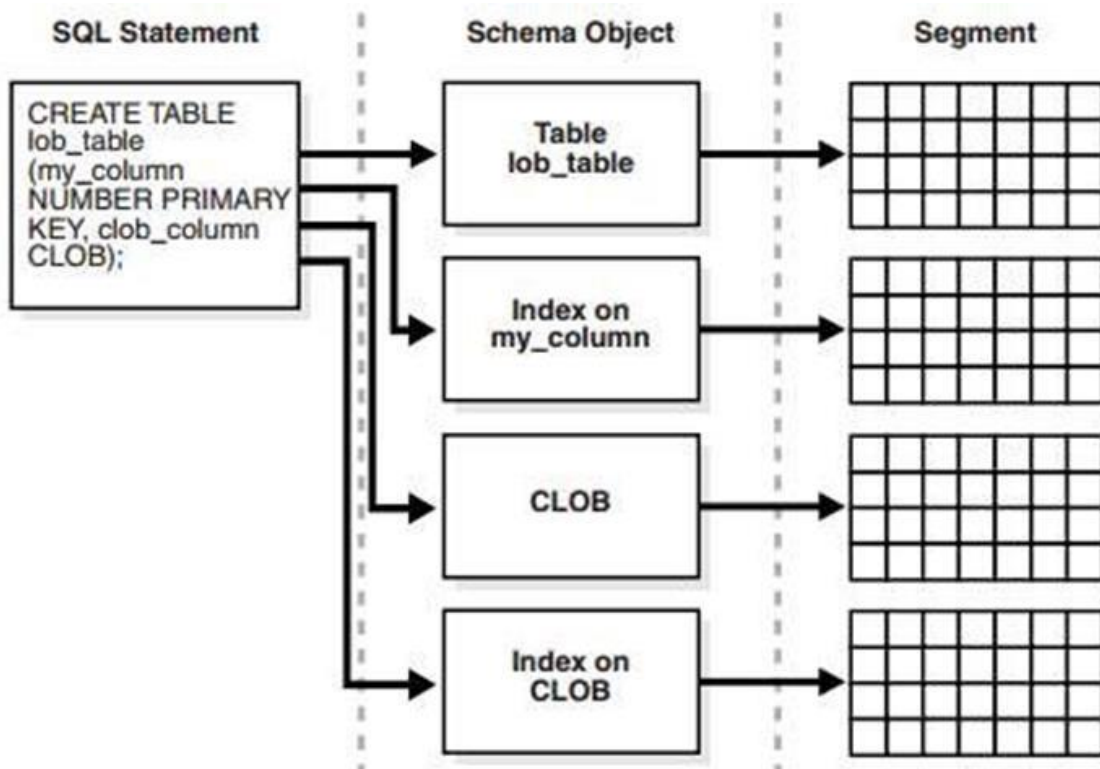
User Segments Creation

For example, installation of an application can create thousands of objects, on consuming significant disk space. Many of these objects may never be used.

You can use the DBMS_SPACE_ADMIN package to manage segments for empty objects.

Overview of Segments

Creation of Multiple Segments



Overview of Segments

SQL DDL for Segments under Tablespace

```
CREATE TABLE PATIENT_HISTORY (PATIENT_ID CHAR (007),
DEPEND_SNO VARCHAR2 (002), ALLERGY_DESC VARCHAR2 (250) NOT NULL,
HISTORY_DESC LONG
) tablespace MCare;
```

```
ALTER TABLE PATIENT_HISTORY
```

```
ADD constraint PK_PATHIST_ID PRIMARY KEY (PATIENT_ID,DEPEND_SNO) USING
INDEX TABLESPACE MCare_idx;
```

Overview of Segments

Temporary Segments

When processing a query, Oracle Database often requires temporary workspace for intermediate stages of SQL statement execution.

Typical operations that may require a temporary segment include sorting, hashing, and merging bitmaps. While creating an index, Oracle Database also places index segments into temporary segments and then converts them into permanent segments when the index is complete

Overview of Segments

UNDO Segments

Oracle Database maintains records of the actions of transactions, collectively known as undo data.

Oracle Database uses undo to do the following:

Roll back an active transaction

Recover a terminated transaction

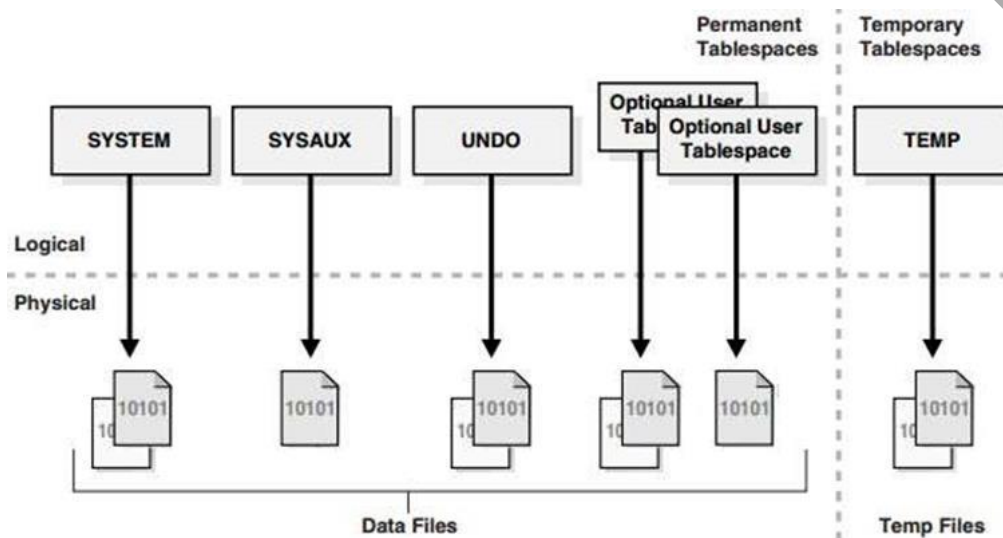
Provide read consistency

Perform some logical flashback operations

Overview of Segments

Tablespace with Segments

A tablespace is a logical storage container for segments. Segments are database objects,



such as tables and indexes, that consume storage space. At the physical level, a tablespace stores data in one Or more data files or temp files.

Introduction to Database Administration

TOPIC 89

Creating Schema Objects Tables, Views

Creating Schema Objects, Tables and Views

Overview of Segments

Introduction to Schema Objects

User & Schema Objects

Schema Objects Types

Overview of Views

Use of Views

SQL to Create View

View of Base Table

Creating Schema Objects, Tables and Views

Introduction to Schema Objects

A database schema is a logical container for data structures, called schema objects.

Examples of schema objects are tables, views, sequences and indexes etc.

Schema objects are created and manipulated with SQL

Creating Schema Objects, Tables and Views

Introduction to Schema Objects

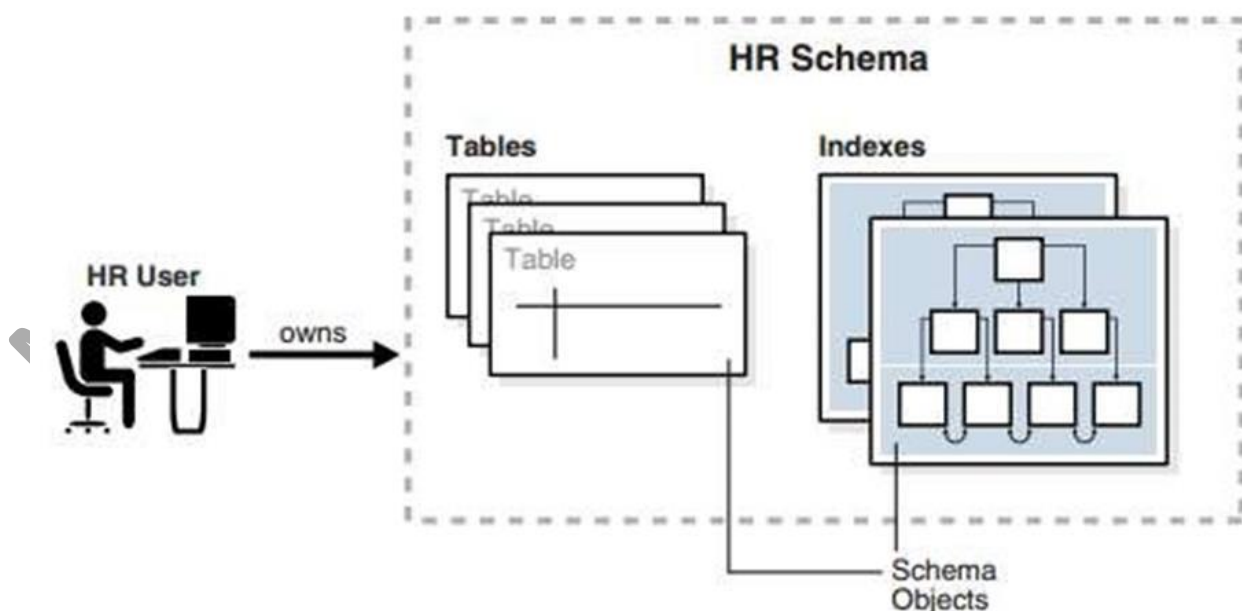
A database user has a password and various database privileges. Each user owns a single schema, which has the same name as the user.

The schema contains the data for the user owning the schema. For example, the hr user owns the hr schema, which contains schema objects such as the employees table.

In a production database, the schema owner usually represents a database application rather than a person.

Creating Schema Objects, Tables and Views

User & Schema Objects



Creating Schema Objects, Tables and Views

Schema Objects Types

Tables

Storing data in rows of tables

Indexes

Indexes are schema objects that contains an entry for each indexed row of the table or table cluster and provide direct, fast access to rows.

Partitions

Partitions are pieces of large tables and indexes.

Creating Schema Objects, Tables and Views

Schema Objects Types

Partitions

Partitions are pieces of large tables and indexes.

Views

Views are customized presentations of data in one or more tables or other views.

Sequences

A sequence is a user-created object that can be shared by multiple users to generate integers.

Creating Schema Objects, Tables and Views

Schema Objects Types

Dimensions

A dimension defines a parent-child relationship between pairs of column sets, where all the columns of a column set must come from the same table.

Synonyms

A synonym is an alias for another schema object.

PL/SQL Programs

Procedural server programming

Creating Schema Objects, Tables and Views

Overview of Views

A view is a logical representation of one or more tables.

In essence, a view is a stored query.

A view derives its data from the tables on which it is based, called base tables.

Base tables can be tables or other

views.

All operations performed on a view actually affect the base tables.

You can use views in most places where tables are used.

Creating Schema Objects, Tables and Views

Use of Views

Provide an additional level of table security by restricting access to a predetermined set of rows or columns of a table

Hide data complexity

Present the data in a different perspective from that of the base table For example, the columns of a view can be renamed without affecting the tables on which the view is based.

Isolate applications from changes in definitions of base tables used.

Creating Schema Objects, Tables and Views

SQL to Create View

```
CREATE VIEW staff AS
```

```
SELECT employee_id, last_name, job_id,  
manager_id, department_id FROM employees;
```

Creating Schema Objects, Tables and Views

View of Base Table

Base Table	employees						
	employee_id	last_name	job_id	manager_id	hire_date	salary	department_id
	203	marvis	hr_rep	101	07-Jun-94	6500	40
	204	baer	pr_rep	101	07-Jun-94	10000	70
	205	higgins	ac_rep	101	07-Jun-94	12000	110
	206	gietz	ac_account	205	07-Jun-94	8300	110
View	staff						
	employee_id	last_name	job_id	manager_id	department_id		
	203	marvis	hr_rep	101	40		
	204	baer	pr_rep	101	70		
	205	higgins	ac_rep	101	110		
	206	gietz	ac_account	205	110		

TOPIC 90

Analyzing Tables, Indexes & Clusters

Analyzing Tables, Indexes & Clusters

Analyzing Tables, Indexes & Clusters

Overview of Table Clusters

Cluster Key

Overview of Indexed Clusters

DDL for Indexed Clusters

Overview of Indexes

Composite Indexes

Analyzing Tables, Indexes & Clusters

Overview of Table Clusters

A table cluster is a group of tables that share common columns and store related data in the same blocks. When tables are clustered, a single data block can contain rows from multiple tables.

For example, a block can store rows from both the employees and Departments tables rather than from only a single table.

Analyzing Tables, Indexes & Clusters

Cluster Key

The cluster key is the column or columns that the clustered tables have in common.

For example, the employees and departments tables share the department_id column.

You specify the cluster key when creating the table cluster and when creating every table added to the table cluster.

Analyzing Tables, Indexes & Clusters

Cluster Key Value

The cluster key value is the value of the cluster key columns for a particular set of rows. All data that contains the same cluster key value, such as department_id=20, is physically stored together. Each cluster key value is stored only once in the cluster and the cluster index, no matter how many rows of different tables contain the value

Analyzing Tables, Indexes & Clusters

Overview of Indexed Clusters

An indexed cluster is a table cluster that uses an index to locate data. The cluster index is a B-tree index on the cluster key. A cluster index must be created before any rows can be inserted into clustered tables

Analyzing Tables, Indexes & Clusters

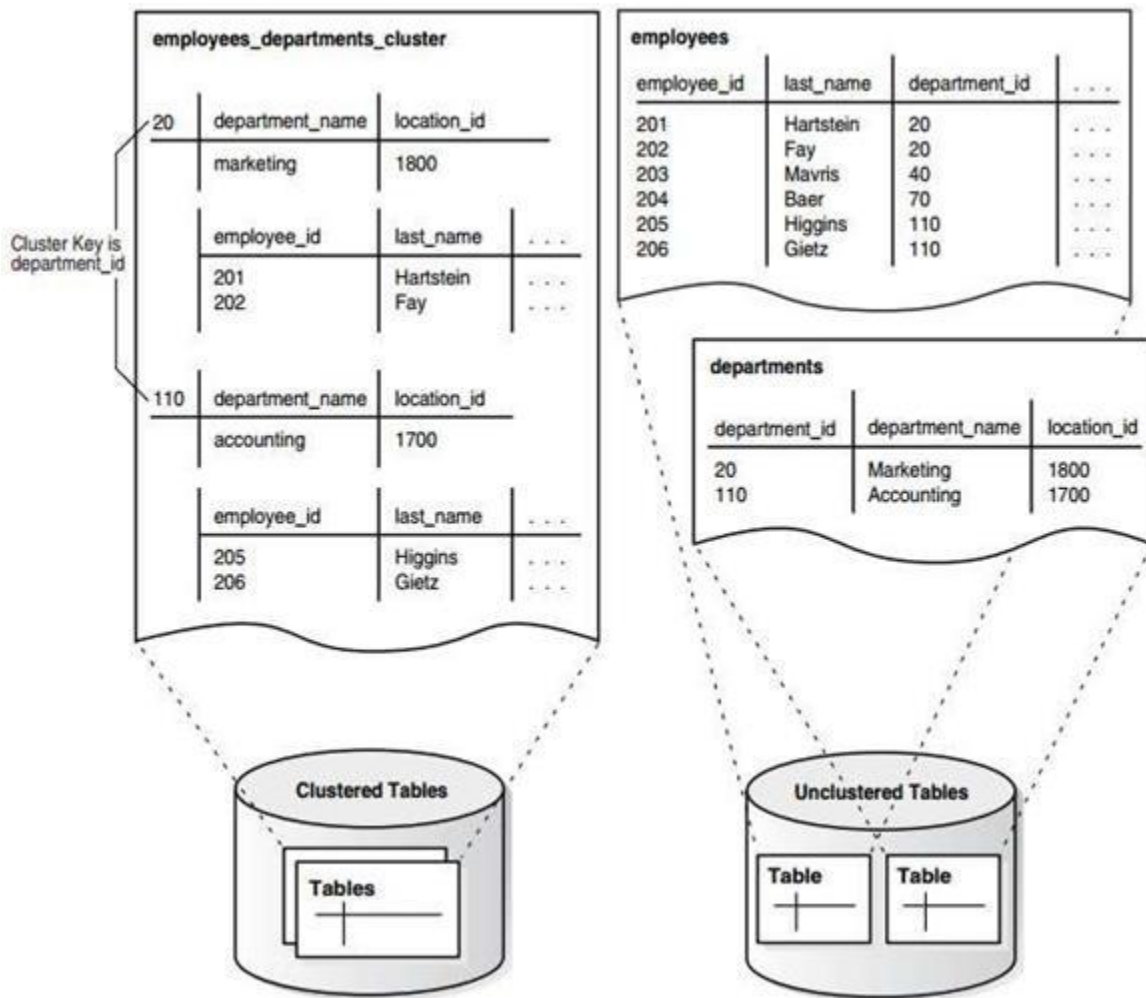
DDL for Indexed Clusters

```
CREATE CLUSTER emp_dept_cluster (department_id NUMBER(4))  
SIZE 512;
```

```
CREATE INDEX idx_emp_dept_cluster ON  
CLUSTER emp_dept_cluster;
```

Analyzing Tables, Indexes & Clusters

DDL for Indexed Clusters



Analyzing Tables, Indexes & Clusters

Overview of Indexes

An index is an optional structure, associated with a table or table cluster, that can sometimes speed data access. By creating an index on one or more columns of a table, you gain the ability in some cases to retrieve a small set of randomly distributed rows from the table.

Indexes are one of many means of reducing disk I/O.

Analyzing Tables, Indexes & Clusters

Composite Indexes

An index is an optional structure, associated with a table or table cluster, that can sometimes speed data access. By creating an index on one or more columns of a table, you gain the ability in some cases to retrieve a small set of randomly distributed rows from the table.

Indexes are one of many means of reducing disk I/O.
Analyzing Tables, Indexes & Clusters

Composite Indexes

A composite index, also called a concatenated index, is an index on multiple columns in a table. Columns in a composite index should appear in the order that makes the most sense for the queries that will retrieve data and need not be adjacent in the table.

```
CREATE INDEX employees_ix  
ON employees (last_name, job_id, salary);
```

Introduction to Database Administration

TOPIC 91

Managing Integrity Constraints-1

Managing Integrity Constraints-1

Integrity Constraints

Data Integrity
Techniques for Data Integrity
Advantages of Integrity Constraints
Types of Integrity Constraints
Managing Integrity Constraints-1

Data Integrity

Business rules specify conditions and relationships that must always be true or must always be false. For example, each company defines its own policies about salaries, employee numbers, inventory tracking, and so on.

It is important that data maintain data integrity, which is adherence to these rules, as determined by the database administrator or application developer.
Managing Integrity Constraints-1

Techniques for Data Integrity

Following are possible options:

Enforcing business rules with triggered stored database procedures

Using stored procedures to completely control access to data

Enforcing business rules in the code of a database application

Using Oracle Database integrity constraints, which are rules defined at the column or object level that restrict values in the database

Managing Integrity Constraints-1

Advantages of Integrity Constraints

An integrity constraint is a schema object that is created and dropped using SQL.

Advantages of integrity constraints over alternatives for enforcing data integrity include:

Declarative ease

Centralized rules

Flexibility when loading data

Managing Integrity Constraints-1

Advantages of Integrity Constraints

Declarative ease

Because you define integrity constraints using SQL DDL statements.

Centralized rules

Integrity constraints are defined for tables and are stored in the data dictionary

Flexibility when loading data

You can disable integrity constraints temporarily to avoid performance overhead when loading large amounts of data.

Managing Integrity Constraints-1

Types of Integrity Constraints

Oracle Database enables you to apply constraints both at the table and column level. A constraint specified as part of the definition of a column or attribute is called an inline specification.

NOT NULL UNIQUE CHECK DEFAULT

PK/ FK: Standard Integrity Constraints

Introduction to Database Administration

TOPIC 92

Managing Integrity Constraints-2

Managing Integrity Constraints-2

Integrity Constraints

NOT NULL Constraint

UNIQUE Constraint

CHECK Constraint

DEFAULT Constraint

FK CASCADE Constraints

FK CASCADE Constraints – Example

Trigger's Constraint

ENABLE/ DISABLE or Drop Constraints

Managing Integrity Constraints-2

NOT NULL Constraint

A NOT NULL constraint requires that a column of a table contain no null values. A null is the absence of a value. By default, all columns in a table allow nulls.

By default, FK is NULL.

```
Create table Student(  
(RollNo CHAR(10),  
FullName VARCHAR2(50) NOT NULL,  
..... );
```

Managing Integrity Constraints-2

UNIQUE Constraint

A unique key (Null able) constraint requires that every value in a column or set of columns be unique. No rows of a table may have duplicate values in a column (the unique key) or set of columns (the composite unique key) with a unique key constraint.

```
Create table Medicine( (ItemNo CHAR(10),  
Name VARCHAR2(50) NOT NULL UNIQUE,  
..... );
```

Managing Integrity Constraints-2

CHECK Constraint

A check constraint on a column or set of columns requires that a specified condition be true or unknown for every row. If DML results in the condition of the constraint evaluating to false, then the SQL statement is rolled back.

Alter table Medicine

Add constraint chk_medType CHECK (MedType IN ('S','P'));

Managing Integrity Constraints-2

DEFAULT Constraint

A default constraint on a column requires that a specified value be true when null value is entered.

Create table Medicine((ItemNo CHAR(10),
Name VARCHAR2(50) NOT NULL,
Status CHAR(1) default 'Y',
.....);

Managing Integrity Constraints-2

FK CASCADE Constraints

CASCADE is applied with INSERT, DELETE or UPDATE.

For example, a foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted. This is called a cascade delete in Oracle.

Managing Integrity Constraints-2

FK CASCADE Constraints - Example

```
CREATE TABLE supplier  
( supplier_id numeric(10) not null, supplier_name varchar2(50) not null, contact_name  
varchar2(50),  
CONSTRAINT supplier_pk PRIMARY KEY (supplier_id));
```

```
CREATE TABLE products  
( product_id numeric(10) not null,  
supplier_id numeric(10) not null,  
CONSTRAINT fk_supplier FOREIGN KEY (supplier_id)  
REFERENCES supplier(supplier_id) ON DELETE CASCADE);
```

Managing Integrity Constraints-2

Trigger's Constraints

```
CREATE TRIGGER cascade_del_stu_record  
BEFORE DELETE ON STUDENT FOR EACH ROW  
BEGIN  
DELETE FROM RECORD  
WHERE RECORD.SID = :OLD.SID;  
END;
```

Managing Integrity Constraints-2

Trigger's Constraints

```
CREATE TRIGGER cascade_del_stu_record BEFORE DELETE ON STUDENT  
FOR EACH ROW BEGIN  
DELETE FROM RECORD  
WHERE RECORD.SID = :OLD.SID;  
END;
```

Managing Integrity Constraints-2

ENABLE/ DISABLE or Drop Constraints

Existing constraints on table can be

Enabled

Disabled

Dropped

For example,

Alter table Invoice

DROP constraint fk_custid;

Introduction to Database Administration

TOPIC 93

Schema Objects, Queries & Views

Schema Objects, Queries & Views

Schema Objects

How a Schema is Created?

How a Schema is Implemented?

Simple & Complex Views

Creating Tables

Creating Sequence or AutoNumber

Data Dictionary for Tables & Views

Schema Objects, Queries & Views

How a Schema is Created

We have studied that DBA has to create default Tablespace and assign temporary Tablespace with storage in terms of data files.

Create first user (owner of schema) with grants such as rights and privileges.

We will cover schema objects Tables and Views.

Ref. DBA-73

Schema Objects, Queries & Views

How a Schema is Implemented?

A view is a logical representation of a table or combination of tables.

In essence, a view is a stored query.

A view derives its data from the tables on which it is based.

These tables are called base tables. Base tables might in turn be actual tables or might be views themselves.

All operations performed on a view actually affect the base table of the view.

Schema Objects, Queries & Views

How a Schema is Implemented?

You can use views in almost the same way as tables. You can query, update, insert into, and delete from views, just as you can standard tables.

Views can provide a different representation (such as subsets or supersets) of the data that resides within other tables and views.

Views can be Simple or Complex

Views can be used for security purposes

View does not contain its own data

Schema Objects, Queries & Views

Simple & Complex Views

```
CREATE VIEW sales_staff AS SELECT empno, ename, deptno FROM emp
WHERE deptno = 10;
```

```
CREATE VIEW sales_staff AS
SELECT job, count(*) FROM emp
WHERE deptno = 10 Group by job;
```

Schema Objects, Queries & Views

Simple & Complex Views

```
CREATE OR REPLACE VIEW division1_staff AS
SELECT ename, empno, job, dname FROM emp, dept
WHERE emp.deptno IN (10, 30) AND emp.deptno = dept.deptno;
```

Insert, delete and update operations for
base tables are applied on simple views.

Schema Objects, Queries & Views

Creating Tables

```
CREATE TABLE dept (
deptno NUMBER(4) PRIMARY KEY,
dname VARCHAR2(14), loc VARCHAR2(13));
```

```
CREATE TABLE emp (
empno NUMBER(4) PRIMARY KEY, ename VARCHAR2(10),
...
deptno NUMBER(2);
...
```

Schema Objects, Queries & Views

Creating Sequence or AutoNumber

```
CREATE SEQUENCE emp_sequence INCREMENT BY 1
START WITH 1 NOMAXVALUE NOCYCLE CACHE 10;
```

Schema Objects, Queries & Views

Data Dictionary for Tables & Views

```
SELECT COLUMN_NAME, UPDATABL
FROM USER_UPDATABL_COLUMNS
WHERE TABLE_NAME = 'EMP_DEPT';
```

COLUMN_NAME	UPD
-----	---
EMPNO	YES
ENAME	YES
DEPTNO	YES
SAL	YES
DNAME	NO
LOC	NO

```
SELECT COLUMN_NAME, UPDATABL FROM USER_UPDATABL_COLUMNS
WHERE TABLE_NAME = 'EMP_DEPT';
```

End

Introduction to Database Administration

TOPIC 94

Creating Tables & Alter Tables

Creating Tables & Alter Tables

Creating & Alter Tables

What is a Table?

Design Table Guidelines

Heap-Organized Table

Clustered Table

Indexed-Organized Table

Partitioned Table

Storage Location of a Table

Alter Table

Tables with Segment Creation

Creating Tables & Alter Tables

What is a Table?

Tables are the basic unit of data storage in an Oracle Database. Data is stored in rows and columns. You define a table with a table name, such as employees, and a set of columns. You give each column a column name, such as emp_id, last_name, and job_id; a data type, such as VARCHAR2, DATE, or NUMBER; and a width. The width can be predetermined by the data type, as in DATE.

Creating Tables & Alter Tables

What is a Table?

If columns are of the NUMBER data type, define precision and scale instead of width. A row is a collection of column information corresponding to a single record.

You can specify rules for each column of a table. These rules are called integrity constraints. One example is a NOT NULL integrity constraint. This constraint forces the column to contain a value in every row.

Some column types, such as LOBs, varrays, and nested tables, are stored in their own segments. LOBs and varrays are stored in LOB segments, while nested tables are stored in storage tables.

Creating Tables & Alter Tables

Design Table Guidelines

Use descriptive names for tables, columns, indexes, and clusters.

Be consistent in abbreviations and in the use of singular and plural forms of table names and columns.

Document the meaning of each table and its columns with the COMMENT command.

Normalize each table.

Creating Tables & Alter Tables

Design Table Guidelines

Select the appropriate datatype for each column. Consider whether your applications would benefit from adding one or more virtual columns to some tables.

Define columns that allow nulls last, to conserve storage space.

Cluster tables whenever appropriate, to conserve storage space and optimize performance of SQL statements.

Creating Tables & Alter Tables

Heap-Organized Table

This is the basic, general purpose type of table which is the primary subject of this chapter. Its data (heap).

Creating Tables & Alter Tables

Clustered Table

A clustered table is a table that is part of a cluster. A cluster is a group of tables that share the same data blocks because they share common columns and are often used together

Creating Tables & Alter Tables

Indexed-Organized Table

Unlike an ordinary (heap-organized) table, data for an index-organized table is stored in a B-tree index structure in a primary key sorted manner.

Besides storing the primary key column values of an index-organized table row, each index entry in the B-tree stores the non key column values as well

Creating Tables & Alter Tables

Partitioned Table

Partitioned tables enable your data to be broken down into smaller, more manageable pieces called partitions, or even subpartitions. Each partition can have separate physical attributes, such as compression enabled or disabled, type of compression, physical storage settings, and tablespace, thus providing a structure that can be better tuned for availability and performance.

Creating Tables & Alter Tables

Partitioned Table

In addition, each partition can be managed individually, which can simplify and reduce the time required for backup and administration.

Creating Tables & Alter Tables

Storage Location of a Table

It is advisable to specify the TABLESPACE clause in a CREATE TABLE statement to identify the tablespace that is to store the new table.

For partitioned tables, you can optionally identify the tablespace that is to store each partition.

Ensure that you have the appropriate privileges and quota on any tablespaces that you use.

If you do not specify a tablespace in a CREATE TABLE statement, the table is created in your default tablespace.

Creating Tables & Alter Tables

Alter Table

Alter table command applies when table is already exist in a schema.

We can add constraints, columns and modifications operations on table using Alter table DDL statements.

Creating Tables & Alter Tables

Tables with Segment Creation

```
CREATE TABLE part_time_employees (  
    empno NUMBER(8),  
    name VARCHAR2(30),  
    hourly_rate NUMBER (7,2)  
)  
SEGMENT CREATION DEFERRED;  
SELECT segment_name, partition_name FROM user_segments;
```

SEGMENT_NAME	PARTITION_NAME
HOURLY_EMPLOYEES	EMPNO_TO_100
HOURLY_EMPLOYEES	EMPNO_TO_200

```
CREATE TABLE hourly_employees (  
    empno NUMBER(8),  
    name VARCHAR2(30),  
    hourly_rate NUMBER (7,2)  
)  
SEGMENT CREATION IMMEDIATE  
PARTITION BY RANGE(empno)  
    (PARTITION empno_to_100 VALUES LESS THAN (100),  
    PARTITION empno_to_200 VALUES LESS THAN (200));
```

Introduction to Database Administration

TOPIC 95

DDL Script Writing-1

DDL Script Writing-1

Script Writing

What is a Script?

A Sample DDL in a Script?

Creating Temporary Tables

Table with TEMP Tablespace

Bulk Data Load in a Table

Alter Table
Index with Tablespace
DDL Partition Table
DDL Script Writing-1

What is a Script?

Normally writing or generating DDL commands to create/ alter tables, views, constraints and storages etc., in an order in plain text using **Notepad or text files**
Mostly scripts contain structure of tables. Sometimes, DML commands are also put in scripts as per management requirement.
Scripts can be with Create, Alter and Drop Tables
DDL Script Writing-1

A Sample DDL in a Script

```
CREATE TABLE hr.admin_emp (  
    empno      NUMBER(5) PRIMARY KEY,  
    ename      VARCHAR2(15) NOT NULL,  
    ssn        NUMBER(9) ENCRYPT USING 'AES256',  
    job        VARCHAR2(10),  
    mgr        NUMBER(5),  
    hiredate   DATE DEFAULT (sysdate),  
    photo      BLOB,  
    sal        NUMBER(7,2),  
    hrly_rate  NUMBER(7,2) GENERATED ALWAYS AS (sal/2080),  
    comm       NUMBER(7,2),  
    deptno     NUMBER(3) NOT NULL  
                CONSTRAINT admin_dept_fkey REFERENCES hr.departments  
                (department_id))  
    TABLESPACE admin_tbs  
    STORAGE ( INITIAL 50K);  
  
COMMENT ON TABLE hr.admin_emp IS 'Enhanced employee table';
```

DDL Script Writing-1

Creating Temporary Tables

Temporary tables are useful in applications where a result set is to be buffered (temporarily persisted), perhaps because it is constructed by running multiple DML operations.
For example, consider the following:

```
CREATE GLOBAL TEMPORARY TABLE admin_work_area
    (startdate DATE,
     enddate DATE,
     class CHAR(20))
ON COMMIT DELETE ROWS;
```

DDL Script Writing-1

Table with TEMP Tablespace

```
CREATE TEMPORARY TABLESPACE tbs_t1
    TEMPFILE 'tbs_t1.f' SIZE 50m REUSE AUTOEXTEND ON
    MAXSIZE UNLIMITED
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;
```

```
CREATE GLOBAL TEMPORARY TABLE admin_work_area
    (startdate DATE,
     enddate DATE,
     class CHAR(20))
ON COMMIT DELETE ROWS
TABLESPACE tbs_t1;
```

DDL Script Writing-1

Bulk Data Load in a Table

```
FORALL i IN 1..numrecords
    INSERT /*+ APPEND_VALUES */ INTO orderdata
    VALUES(ordernum(i), custid(i), orderdate(i), shipmode(i), paymentid(i));
COMMIT;
```

Alter Table

```
TABLE hr.admin_emp MOVE  
STORAGE { INITIAL 20K  
          NEXT 40K  
          MINEXTENTS 2
```

```
TABLESPACE hr_tbs;
```

```
ALTER TABLE hr.admin_emp
```

```
ADD PARTITION
```

```
ADD PARTITION NBFB8R (7,3));
```

DDL Script Writing-1

Index with Tablespace

```
CREATE INDEX int_sales_index ON int_salestable
```

```
(s_saledate, s_productid, s_custid) TABLESPACE tbs_low_freq;
```

DDL Script Writing-1

DDL Partition Table

```
CREATE TABLE salestable  
(s_productid NUMBER,  
s_saledate DATE,  
s_custid NUMBER,  
s_totalprice NUMBER)  
TABLESPACE users  
PARTITION BY RANGE(s_saledate)  
(PARTITION sal03q1 VALUES LESS THAN (TO_DATE('01-APR-2003'  
PARTITION sal03q2 VALUES LESS THAN (TO_DATE('01-JUL-2003'  
PARTITION sal03q3 VALUES LESS THAN (TO_DATE('01-OCT-2003'  
PARTITION sal03q4 VALUES LESS THAN (TO_DATE('01-JAN-2004'
```

Introduction to Database Administration

TOPIC 96

DDL Script Writing-2

DDL Script Writing-2

Script Writing

Writing Script using Notepad

Tables in an Order

Drop Tables in an Order

Other Schema Objects

DDL Script Writing-2

Writing Script using Notepad

Cust-Ord-Prod Schema script.txt - Notepad

```

File Edit Format View Help
create table Customer
(custID CHAR(10),
CustName VARCHAR2(25) NOT NULL,
Address VARCHAR2(50),
City VARCHAR2(50),
TelNo VARCHAR2(15),
Email VARCHAR2(50)
);

Alter table Customer
add constraint PK_Custid primary key (CustID);

Create table Orders
(InvoiceNo CHAR(12),
InvType CHAR(1),
InvDate Date NOT NULL,
TotalPrice Number(6,2),
CustID CHAR(8),
SalesEmpID CHAR(5)
);

Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO);

Alter table Orders
add constraint FK_Cust_CustID foreign key (custID)
references Customer(CustID);

create table Employee
(EmpID CHAR(3))

```

Script is a plain text structure of schema

Tables in an Order

```

create table Customer
(custID          CHAR(10),
CustName        VARCHAR2(25) NOT NULL,
Address         VARCHAR2(50),
City            VARCHAR2(50),
TelNo           VARCHAR2(15),
Email           VARCHAR2(50) UNIQUE)
tablespace TS_Sales;

Alter table Customer
add constraint PK_Custid primary key (CustID);

create table Customer (oustID  CMAR(10).
CustName  VARCHARZ(25) NOT NULL.
Address   VARCHAR2(50)

```

City VARCHAR2(50).
 TelNo VARCHAR2(15).
 Email VARCHAR2(50) UNIQUE)
 tablespace T8_sales:
 Alter table Customer
 add constraint PK_Custid primary key (CustID):

Tables in an Order

```

Create table Orders
(InvoiceNo      CHAR(12),
 InvType        CHAR(1),
 InvDate        Date NOT NULL,
 TotalPrice     Number(6,2),
 SalesEmpID     CHAR(5),
 CustID         CHAR(8))
tablespace TS_Sales;

Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO)
USING INDEX TABLESPACE Sale_indx;

Alter table Orders
add constraint FK_Cust_CustID foreign key (custID)
references Customer(CustID);
  
```

Create table Orders
 (InvoiceNo CHAR(12).
 InvType CHAR(1).
 InvDate Date NOT NULL. alPr Number(6.2).
 SalesEmpID CHAR(5).
 CustID CHAR(8))
 tablespace TS_Bales:
 lter table Orders
 dd constraint PK InvNo primary key (Invoiced0) RING INDEX TABLESPACE Male indx;
 lter table Orders
 adds non I r a i n d F K Y C u s s I C u c I D f o r e i g n k e y (o u s I I D)
 ze Le senses CuzrI omes (CuST I D):

Tables in an Order

```
create table Employee
(EmpID          CHAR(3),
EmpName        VARCHAR2(25) NOT NULL,
Address        VARCHAR2(50),
City           VARCHAR2(50),
TelNo          VARCHAR2(15),
Email          VARCHAR2(50) UNIQUE)
tablespace TS_Sales;

Alter table Employee
add constraint PK_EmpID primary key (EmpID)
USING INDEX TABLESPACE Sale_idx;

Alter table Orders
add constraint FK_Emp_EmpID foreign key (SalesEmpID)
references Employee(EmpID)
USING INDEX TABLESPACE Sale_idx;
```

```
create table Employee (EmpID  CHAR(3).
EmpName  VARCHAR2(25) NOT NULL.
ddr  VARCHAR2(50).
VARCMAR250).
elNo  VARCHAR?(1S).
Email  VARCHAR2(S0) UNIQUE)
tablespace TS_Sales:
lter table Employee
dd oonstraint PK EmpID primary key (Einp I D ) 8 ING INDEX TABLE
8PACE 8ale idx;
```

```
lter table Orders
add constraint FK_Emp_EmpID fore gn key (8a1esEmpID)
re f eres n we  Einp I oyee ( Einp I D )
8 ING INDEX TABLE 8PACE 8a I e idx :
DDL Script Writing-2
```

Tables in an Order

```

Create table Item
(ItemID          CHAR(5),
ItemName        VARCHAR2(50) NOT NULL,
PPrice          NUMBER(10,2),
TotQty          NUMBER(5),
Status          CHAR(1))
tablespace TS_Stock;

Alter table Item
add constraint PK_ItemID primary key (ItemID)
USING INDEX TABLESPACE Sale_idx;

```

Tables in an Order

```

Create table Sold
(
ItemID          CHAR(5),
InvoiceNo       CHAR(12),
SPrice          NUMBER(10,2) Not Null,
SQty            NUMBER(10,2) Not Null)
tablespace TS_ItemSold;

Alter table Sold
add constraint PK_Sold primary key (ItemID, InvoiceNo)
USING INDEX TABLESPACE Sold_idx;

Alter table Sold
add constraint FK_Sold_ItemID foreign key (ItemID)
references Item(ItemID);

Alter table Sold
add constraint FK_Sold_InvoiceNo foreign key (InvoiceNo)
references Invoice(InvoiceNo);

```

```

Create table Sold
ItemID CHAR(5).
InvoiceNo CHAR(12).
SPrice NUMBER(10.2) Not Null.
SQty NUMBER(10*2) Not Null) tablespace TB_ItemBold:

```

```

Alter table Sold
add constraint PK_5old primary key (ItemID. InvoiceNo)
USING INDEX TABLESPACE Sold_idx:

```

Alter table Bold

add constraint FK_Bold_ItemID Foreign key (ItemID) references Item(ItemID);

Alter table Sold

add constraint FK_Sold_InvoiceNo foreign key (InvoiceNo) references Invoice(InvoiceNo);

DDL Script Writing-2

Drop Tables in an Order

```
Drop table Sold;  
Drop table employee;  
Drop table orders;  
Drop table customer;  
Drop table Item;
```

DDL Script Writing-2

Other Schema Objects

Script may contains other objects such as Views

Sequences

Synonym

PL/SQL Triggers & Code

DML for sample data

End

Introduction to Database Administration

TOPIC 97

Drop Tables & Recycle Bin

Drop Tables & Recycle Bin

Overview of Segments

Drop Tables

Drop Tables Precautions

Drop Tables Commands

What is the Recycle Bin?

Object Naming in Recycle Bin

Enable & Disable the Recycle Bin

Purging in the Recycle Bin

Data Dictionary Views for Recycle Bin

Drop Tables & Recycle Bin

Drop Tables

To drop a table that you no longer need, use the DROP TABLE statement. The table must be contained in your schema or you must have the DROP ANY TABLE system privilege.

Drop Tables & Recycle Bin

Drop Tables Precautions

Dropping a table removes the table definition from the data dictionary. All rows of the table are no longer accessible.

All indexes and triggers associated with a table are dropped.

All views and PL/SQL program units dependent on a dropped table remain, yet become invalid (not usable).

All synonyms for a dropped table remain, but return an error when used.

Drop Tables & Recycle Bin

Drop Tables Precautions

All extents allocated for a table that is dropped are returned to the free space of the tablespace and can be used by any other object requiring new extents or new objects. All rows corresponding to a clustered table are deleted from the blocks of the cluster

Drop Tables & Recycle Bin

Drop Tables Commands

```
DROP TABLE hr.int_admin_emp;
```

If the table to be dropped contains any primary or unique keys referenced by foreign keys of other tables and you intend to drop the FOREIGN KEY constraints of the child tables, then include the CASCADE clause in the DROP TABLE statement

```
DROP TABLE hr.admin_emp CASCADE  
CONSTRAINTS;
```

Drop Tables & Recycle Bin

Drop Tables Commands

If you should want to immediately release the space associated with the table at the time you issue the DROP TABLE statement, include the PURGE clause as shown in the following statement:

```
DROP TABLE hr.admin_emp PURGE;
```

Drop Tables & Recycle Bin

What is the Recycle Bin?

The recycle bin is actually a data dictionary table containing information about dropped objects. Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space.

They continue to count against user space quotas, until specifically purged from the recycle bin.

```
SELECT * FROM RECYCLEBIN;
```

Drop Tables & Recycle Bin

Object Naming in Recycle Bin

```
SQL> select * from tab;
```

TNAME	TABTYPE	CLUSTERID
BIN\$h1LPRYaURGeQBt1LS4/DHg==\$0	TABLE	
BIN\$pgI7NTBnRKSh+170M90xHg==\$0	TABLE	
BLOBS	TABLE	
BONUS	TABLE	

The renaming convention is as follows: BIN\$unique_id\$version

unique_id is a 26-character globally unique identifier for this object, which makes the recycle bin name unique across all databases

Version is a version number assigned by the database

Drop Tables & Recycle Bin

Enable & Disable the Recycle Bin

To disable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = OFF;
```

```
ALTER SYSTEM SET recyclebin = OFF SCOPE = SPFILE;
```

2. If you used ALTER SYSTEM, restart the database.

To enable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = ON;
```

```
ALTER SYSTEM SET recyclebin = ON SCOPE = SPFILE;
```

2. If you used ALTER SYSTEM, restart the database.

Drop Tables & Recycle Bin

Purging is the Recycle Bin

If you decide that you are never going to restore an item from the recycle bin, you can use the PURGE statement to remove the items and their associated objects from the recycle bin and release their storage space.

```
PURGE TABLE "BIN$jsleilx392mk2=293$0";
```

```
PURGE TABLESPACE example; PURGE RECYCLEBIN;
```

Drop Tables & Recycle Bin

Data Dictionary Views for Recycle Bin

```
SELECT object_name, original_name FROM dba_recyclebin
WHERE owner = 'HR';
```

OBJECT_NAME	ORIGINAL_NAME
-------------	---------------

BIN\$yrMK1ZaLMhfgNAgAIMenRA==\$0	EMPLOYEES
----------------------------------	-----------

You can also view the contents of the recycle bin using the SQL*Plus command SHOW RECYCLEBIN.

```
SQL> show recyclebin
```

ORIGINAL NAME	RECYCLEBIN NAME	OBJECT TYPE	DROP TIME
EMPLOYEES	BIN\$yrMK1ZaVMhfgNAgAIMenRA==\$0	TABLE	2003-10-27:14:00:19

Introduction to Database Administration

TOPIC 98

Indexed Organized Tables
Indexed Organized Tables

Overview of Segments

Drop Tables
Indexed Organized Tables

Drop Tables

To drop a table that you no longer need, use the DROP TABLE statement. The table must be contained in your schema or you must have the DROP ANY TABLE system privilege.

Introduction to Database Administration

Data Dictionary Views for Tables

Data Dictionary Views for Tables

Integrity Constraints

Three Types of Views

Displaying Views for Tables

Size of a Table

Displaying Table's Constraints

Displaying Table's Columns Details

Data Dictionary Views for Tables

What is an index?

Indexes are optional structures associated with table and clusters that allow SQL queries to execute more quickly against a table

Index helps you locate information faster than if there were no index, an oracle database index provides a faster access path to table data. You can use indexes without rewriting any queries. Your results are the same, but you see them more quickly.

Independency of an index

Indexes are logically and physically independent of the data in the associated table. Being independent structure, they require storage space. You can create and drop an index without affecting the base tables, database application or other indexes.

The database automatically maintains indexes when you insert, update and delete rows of the associated table. If you drop an index all applications

continue to work, access to previously indexed data might be slower.

Data load & index usage

Data is often inserted or loaded into a table using either a SQL * loader or an import utility. It is more efficient to create an index for a table after inserting or loading the data.

If you create one or more indexes before loading data, the database then must update every index as each row is inserted.

Index with segments & Tablespace

- The amount for each user is determined by the initialization parameter SORT_AREA_SIZE.
- The database also swaps sort information to and from temporary segments that are only allocated during the index creation in the user's temporary tablespace.

Suitable Columns for Indexing

- Values are relatively unique in the column.
- There is a wide range of values (good for regular indexes).
- There is a small range of values (good for bitmap indexes).
- The column contains many nulls, but queries often select all rows having a value.

Indexing Limit on Table

- A table can have any number of indexes. However, the more indexes there are, the more overhead is incurred as the table is modified. Specifically, when rows are inserted or deleted, all indexes on the table must be updated as well.
- Also, when a column is updated, all indexes that contain the column must be updated.