ORACLE TABLESPACES

A Resource Guide

JUNE 15, 2016

TECHGOEASY.COM

WHAT IS ORACLE TABLESPACE?

It is logical storage unit in Oracle Database. Tablespace consists of one or more datafiles.

Tablespace is further divided in to logical units Segments, Segment is divided in to Extent and Extent in to Block

Various type of Tablespace are BIGFILE, SYSTEM, SYSAUX, and UNDO

BASICS ABOUT SEGMENT/EXTENT/DATA BLOCK

Segment	**It** is a space allocated for a specific logical structure (table/index, partition etc.), cannot span tablespaces but it can span datafiles belonging to tablespace. It is made up of one or more extents
Extent	**It is** a set of contiguous data blocks, cannot span a datafile (must exist in only one). When the segment grows more extents are allocated.
Data Block	The smallest unit of data in Oracle server, one or more blocks corresponds to one or more operating system blocks (should be a multiple of operating system block size), initial size determined by DB_BLOCK_SIZE parameter in the parameter file

WHAT IS DATAFILES?

-It is physical structure to store oracle data
-One or more physical datafile are logically grouped together to make a tablespace
-A Datafile can be associated with only one tablespace

CREATE TABLESPACE STATEMENT

The tablespace can be created by the user having sysdba privilege to hold various tables and index objects.

Complete Syntax for Create table statement

```
CREATE [TEMPORARY / UNDO] TABLESPACE <tablespace_name>
 DATAFILE / TEMPFILE      '<datafile and Path where file to create>' SIZE
<integer M>
 BLOCKSIZE  <DB_BLOCK_SIZE parameter /2k/4k/8k/16k/32k >
 AUTOEXTEND { [OFF/ON (NEXT <integer K/M >  MAXSIZE<integer K/M >)
/ UNLIMITED] }
 LOGGING/NOLOGGING (LOGGING default)
FORCE LOGGING {ON/OFF(default)}
 ONLINE/OFFLINE (Online default)
 EXTENT MANAGEMENT { [DICTIONARY] /
                    [LOCAL Default (AUTOALLOCATE / UNIFORM SIZE <integer
K/M >)] }
 AUTO SEGMENT MANAGEMENT { AUTO/MANUAL}
 PERMANENT  / TEMPORARY (Permanent default)
 MINIMUM EXTENT
 DEFAULT STORAGE  {    [INITIAL <integer K/M >]
                      [NEXT <integer K/M >]
                      [PCTINCREASE <integer K/M >]
                      [MINEXTENTS <integer>]
                      [MAXEXTENTS <integer> / UNLIMITED]
                      [FREELISTS <integer>]
                      [FREELIST GROUPS <integer>]
                      [OPTIMAL <integer>/NULL]
                      [BUFFER_POOL < DEFAULT/KEEP/RECYCLE >] }
;
```

VARIOUS OPTION OF CREATE TABLESPACE STATEMENT ARE EXPLAINED BELOW	
UNDO/TEMPORARY	Temp or temporary tablespaces are used to store data with short lifespan (transient data) Undo tablespaces are used to store "before image" data that can be used to undo transactions If nothing is given then it is a default tablespace which can store normal table/index segments
DATAFILE	Specifies the location and name of datafile
SIZE	specifies the size of datafile in Kb/Mb
BLOCKSIZE	specified the block size for the tablespace
LOGGING/NOLOGGING–	specifies if writing to the redo log is done
FORCE LOGGING	If it is On, even no logging operation in any segment in the tablespace are written to redo
ONLINE/OFFLINE	It specifies if tablespace will be placed online after creation
AUTOEXTEND	ON means datafile can be auto extended. Off means no auto extension
TEMPORARY/PERMANENT	It specifies if tablespace holds permanent or temporary objects
DEFAULT STORAGE	specifies storage parameters for all segments created in the dictionary managed tablespace
MINIMUM EXTENT	specifies the minimum size for any extent in the tablespace
EXTENT MANAGEMENT	There are 2 types of space management LOCALLY MANGED Locally managed tablespaces manage all extent allocations in the datafile header using a bitmap. The advantages of using locally managed tablespaces are less space in system tablespace (extent management not saved in data dictionary), reduce contention on data dictionary tables,

	eliminates the need to coalesce free extents, do not generate redo DICTIONARY-MANAGED Dictionary managed tablespace records all extent allocations in the data dictionary tables. One can specify the EXTENT MANAGEMENT clause in the CREATE TABLESPACE command and specify the management type (DICTIONARY/LOCAL). If LOCAL option is selected, one can specify the extent size using UNIFORM SIZE n Kb/Mb clause. The extent size and allocation table are kept in the datafile header. Altering storage parameters for locally managed tablespace is not allowed. In order to change storage parameters, one needs to create a new tablespace with new storage parameters and move all segments to newly created tablespace. Dictionary managed tablespaces record extent allocation in the data dictionary, each segment created in the tablespace can have its own storage parameters, needs to be coalesced
Segment Space management	There are two options **Manual** Manual you want to use free lists for managing free space within segments. Free lists are lists of data blocks that have space available for inserting rows. This form of managing space within segments is called manual segment-space management because of the need to specify and tune the PCTUSED, FREELISTS, and FREELISTS GROUPS storage parameters for schema objects created in the tablespace. MANUAL is the default. **Auto**

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

VARIOUS EXAMPLE OF CREATE TABLESPACE STATEMENTS

PERMANENT TABLESPACES

Permanent tablespaces are used to store user data and user created objects like tables, indexes and materialized views.

Single datafile

```
CREATE TEMPORARY TABLESPACE EXAMPLE  DATAFILE
'/u01/oracle/TEST/oradata/example_1.dbf' SIZE 1000M;
```

Multiple tempfile

```
CREATE TABLESPACE EXAMPLE DATAFILE
 '/u01/oracle/TEST/oradata/example_1.dbf' SIZE 1000M
'/u01/oracle/TEST/oradata/example_2.dbf' SIZE 1000M
;
```

TEMP TABLESPACES

Temp or temporary tablespaces are used to store data with short lifespan (transient data), for example: global temporarily tables or sort results.

Single tempfile

```
CREATE TEMPORARY TABLESPACE TEMP TEMPFILE
```

```
'/u01/oracle/TEST/oradata/temp_1.dbf' SIZE 1000M;
```

Multiple tempfile

```
CREATE TABLESPACE TEMP TEMPFILE
 '/u01/oracle/TEST/oradata/temp_1.dbf' SIZE 1000M
'/u01/oracle/TEST/oradata/temp_2.dbf' SIZE 1000M
;
```

UNDO TABLESPACES

Undo tablespaces are used to store "before image" data that can be used to undo transactions.

Single datafile

```
CREATE UNDO TABLESPACE UNDO_TBS1  DATAFILE
'/u01/oracle/TEST/oradata/undo_1.dbf' SIZE 1000M;
```

Multiple datafile

```
CREATE UNDO TABLESPACE UNDO_TBS1  DATAFILE
 '/u01/oracle/TEST/oradata/undo_1.dbf' SIZE 1000M
'/u01/oracle/TEST/oradata/undo_2.dbf' SIZE 1000M
;
```

OTHER EXAMPLES

Tablespace created with extent management local of uniform size 1M and segment space management auto

```
CREATE TABLESPACE TEST  DATAFILE '/u01/oracle/TEST/oradata/test_1.dbf'
SIZE 1000M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M
SEGMENT SPACE MANAGEMENT AUTO;
```

Tablespace created with extent management local of uniform size 1M and no automatic segment space management

```
CREATE TABLESPACE TEST DATAFILE '/u01/oracle/TEST/oradata/test_1.dbf'
SIZE 1000MEXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;
```

Tablespace created with extent management local of Auto allocate size and no automatic segment space management

```
CREATE TABLESPACE TEST DATAFILE '/u01/oracle/TEST/oradata/test_1.dbf'
SIZE 1000M

EXTENT MANAGEMENT LOCAL Autoallocate;
```

Tablespace created with dictionary extent management

```
CREATE TABLESPACE TEST DATAFILE '/u01/oracle/TEST/oradata/test_1.dbf'
SIZE 1000M

EXTENT MANAGEMENT dictionary;
```

Tablespace created with local extent management and OMF with ASM storage

```
CREATE TABLESPACE TEST DATAFILE '+DATA'    SIZE 1000M

EXTENT MANAGEMENT local;
```

File name is automatically managed by Oracle

VARIOUS TABLESPACE ALTERATION OPTIONS:

HOW TO ADD DATAFILE IN A TABLESPACE

we can use ALTER TABLESPACE to add datafile in tablespace like

Default Tablespace

Syntax

```
ALTER TABLESPACE <tablespace_name> ADD DATAFILE
<location_of_datafile>;
```

Example

```
ALTER TABLESPACE TEST ADD DATAFILE
'/u01/oracle/TEST/oradata/test_4.dbf' SIZE 1000M
```

TEMP table space

Syntax

```
ALTER TABLESPACE <tablespace_name> ADD TEMPFILE
‘<location_of_tempfile>’ SIZE <size>;
```

Example

```
ALTER TABLESPACE TEMP ADD TEMPFILE
'/u01/oracle/TEST/oradata/temp_4.dbf' SIZE 1000M
```

HOW TO MODIFY THE EXISTING DATAFILE AUTO EXTEND CHARACTERISTICS IN A TABLESPACE

you can modify datafile auto extend   using alter database datafile option

Syntax

```
ALTER DATABASE DATAFILE <location_of_datafile> AUTOEXTEND ON|OFF
NEXT <size> MAXSIZE <size>;
```

Example

```
ALTER DATABASE DATAFILE ‘/u01/oracle/TEST/oradata/test_4.dbf’
AUTOEXTEND ON NEXT 50M MAXSIZE 2400M;
```

HOW TO ALTER THE TABLESPACE TO OFFLINE

-Taking a tablespace offline/online is done using the ALTER TABLESPACE 'name' OFFLINE/ONLINE clause.

There are several options for taking tablespace offline:

Normal	It flushes all data blocks that belong to the tablespace from the SGA
Temporary	It performs checkpoint for the selected tablespace only
Immediate	It does not perform checkpoint or flush data blocks – requires media recovery
For recover	It is used for tablespace point in time recovery (TSPITR)

HOW TO ALTER THE TABLESPACE TO READ-ONLY

It allows read only operations on tablespace (no DML), objects can be dropped from the tablespace

Syntax

```
ALTER TABELSPACE 'name' READ ONLY;
```

Example

```
ALTER TABELSPACE TEST READ ONLY;
```

HOW TO DROP THE TABLESPACE

Dropping the tablespace means tablespace is removed from data dictionary, contents are removed from data dictionary (optional), and datafiles are deleted (optional),

When the tablespace does not have any contents

Syntax

DROP TABELSPACE 'name';

Example

DROP TABELSPACE 'TEST';

When the tablespace has any contents

Syntax

DROP TABELSPACE 'name' INCLUDING CONTENTS;

Example

DROP TABELSPACE 'TEST' INCLUDING CONTENTS;

When the tablespace has any contents and datafiles from OS also need to deleted

Syntax

DROP TABELSPACE 'name' INCLUDING CONTENTS and DATAFILES;

Example

DROP TABELSPACE 'TEST' INCLUDING CONTENTS and DATAFILES;

HOW TO INCREASE THE SIZE OF THE TABLESPACE

The increase can be done automatically or manually.

Automatically – using the AUTOEXTEND ON MAXSIZE n Kb/Mb clause.

```
Syntax
ALTER DATABASE DATAFILE <location_of_datafile> AUTOEXTEND ON|OFF
NEXT <size> MAXSIZE <size>;
Example
ALTER DATABASE DATAFILE '/u01/oracle/TEST/oradata/test_4.dbf'
AUTOEXTEND ON NEXT 50M MAXSIZE 2400M;
```

Manually – ALTER TABLESPACE 'name' ADD DATAFILE 'name' clause or

ALTER DATABASE DATAFILE ‘DATAFILE NAME’ resize <bigger size>;

```
Syntax
ALTER TABLESPACE <tablespace_name> ADD DATAFILE
<location_of_datafile>;
Example
ALTER TABLESPACE TEST ADD DATAFILE
'/u01/oracle/TEST/oradata/test_4.dbf' SIZE 1000M
ALTER DATABASE DATAFILE '/u01/oracle/TEST/oradata/test_4.dbf' RESIZE
2000M;
```

HOW TO MOVE TABLESPACE/DATAFILE TO ANOTHER LOCATION

In case of single or multiple datafile move in a tablespace, steps are

1) Bring the tablespace offline
2) Move all the datafiles using OS utility
3) Rename the datafile using alter tablespace command

   ALTER TABLESPACE 'name' RENAME 'file_name' TO 'file_name'
4) Bring the tablespace online

```
Alter tablespace TEST offline;

mv /u01/oracle/TEST/oradata/test_4.dbf
```

```
/u02/oracle/TEST/oradata/test_4.dbf

ALTER TABLESPACE 'TEST' RENAME /u01/oracle/TEST/oradata/test_4.dbf  '
TO ‘/u02/oracle/TEST/oradata/test_4.dbf  ‘;

 Alter tablespace TEST online;
```

In case of multiple datafile of different tablespace, if we don’t want to bring all tablespace to offline, we can use the alter datafile command are

1) Bring the datafile offline (This is valid if the database is in archive log mode else we need to start the database in mount state)
2) Move all the datafiles using OS utility
3) Rename the datafile using alter database command
4) Alter datafile RENAME DATAFILE 'file_name' TO 'file_name’;
5) Recover the datafile and bring it online

```
Alter database datafile  ‘/u01/oracle/TEST/oradata/test_4.dbf’   offline;

mv /u01/oracle/TEST/oradata/test_4.dbf
/u02/oracle/TEST/oradata/test_4.dbf

ALTER database datafile RENAME /u01/oracle/TEST/oradata/test_4.dbf  ' TO
‘/u02/oracle/TEST/oradata/test_4.dbf  ‘;

 Recover datafile ‘/u02/oracle/TEST/oradata/test_4.dbf  ‘;

Alter database datafile  ‘/u02/oracle/TEST/oradata/test_4.dbf’   online;
```

HOW TO SHRINK THE DATAFILE

We sometimes have shrink the datafile to reclaim filesystem space.

We can do it like

Alter database datafile <datafile name > resize <smaller size>;

Many times we may get the below error as the extent may spread out on the outer of the file

ORA-03297: FILE CONTAINS USED DATA BEYOND REQUESTED RESIZE VALUE".

To resolve the error, we can find the true resize value for all the datafiles in the tablespace by the below procedure

Script is meant for Oracle version 9 and higher

```
set serveroutput on
exec dbms_output.enable(1000000);
declare
cursor c_dbfile is
select f.tablespace_name,f.file_name,f.file_id,f.blocks,t.block_size
,decode(t.allocation_type,'UNIFORM',t.initial_extent/t.block_size,0) uni_extent
,decode(t.allocation_type,'UNIFORM',(128+(t.initial_extent/t.block_size)),128) file_min_size
from dba_data_files f,
dba_tablespaces t
where f.tablespace_name = t.tablespace_name
and t.tablespace_name='APPS_TS_TX_DATA_10MB'
order by f.tablespace_name,f.file_id;

cursor c_freespace(v_file_id in number) is
select block_id, block_id+blocks max_block
from dba_free_space
where file_id = v_file_id
order by block_id desc;
```

```
dummy number;
checkval varchar2(10);
block_correction number;

file_min_block number;

recycle_bin boolean:=false;
extent_in_recycle_bin boolean;

sqlstr varchar2(100);
table_does_not_exist exception;
pragma exception_init(table_does_not_exist,-942);

space_wastage number;

begin

begin
select value into checkval from v$parameter where name = 'recyclebin';
if checkval = 'on'
then
recycle_bin := true;
end if;
exception
when no_data_found
then
recycle_bin := false;
end;
for c_file in c_dbfile
loop
/* initialization of loop variables */
dummy :=0;
extent_in_recycle_bin := false;
```

```
file_min_block := c_file.blocks;

begin

space_wastage:=0; /* reset for every file check */



for c_free in c_freespace(c_file.file_id)
loop
/* if blocks is an uneven value there is a need to correct
with -1 to compare with end-of-file which is even */
block_correction := (0-mod(c_free.max_block,2));
if file_min_block = c_free.max_block+block_correction
then

/* free extent is at end so file can be resized */
file_min_block := c_free.block_id;

/* Uniform sized tablespace check if space at end of file
is less then uniform extent size */
elsif (c_file.uni_extent !=0) and ((c_file.blocks - c_free.max_block) <
c_file.uni_extent)
then

/* uniform tablespace which has a wastage of space in datafile
due to fact that space at end of file is smaller than uniform extent size */

space_wastage:=c_file.blocks - c_free.max_block;
file_min_block := c_free.block_id;

else
/* no more free extent at end of file, file cannot be further resized */
exit check_free;
```

```
end if;
end loop;
end;

/* check if file can be resized, minimal size of file 128 {+ initial_extent}
blocks */
if (file_min_block = c_file.blocks) or (c_file.blocks <= c_file.file_min_size)
then

dbms_output.put_line('Tablespace: '||c_file.tablespace_name||' Datafile:
'||c_file.file_name);
dbms_output.put_line('cannot be resized no free extents found');
dbms_output.put_line('.');

else

/* file needs minimal no of blocks which does vary over versions,
using safe value of 128 {+ initial_extent} */
if file_min_block < c_file.file_min_size
then
file_min_block := c_file.file_min_size;
end if;


dbms_output.put_line('Tablespace: '||c_file.tablespace_name||' Datafile:
'||c_file.file_name);
dbms_output.put_line('current size:
'||(c_file.blocks*c_file.block_size)/1024||'K'||' can be resized to:
'||round((file_min_block*c_file.block_size)/1024)||'K (reduction of:
'||round(((c_file.blocks-file_min_block)/c_file.blocks)*100,2)||' %)');


/* below is only true if recyclebin is on */
if recycle_bin
```

```
then
begin
sqlstr:='select distinct 1 from recyclebin$ where file#='||c_file.file_id;
execute immediate sqlstr into dummy;

if dummy > 0
then

dbms_output.put_line('Extents found in recyclebin for above
file/tablespace');
dbms_output.put_line('Implying that purge of recyclebin might be needed in
order to resize');
dbms_output.put_line('SQL> purge tablespace
'||c_file.tablespace_name||';');
end if;
exception
when no_data_found
then null;
when table_does_not_exist
then null;
end;
end if;
dbms_output.put_line('SQL> alter database datafile '''||c_file.file_name||'''
resize '||round((file_min_block*c_file.block_size)/1024)||'K;');

if space_wastage!=0
then
dbms_output.put_line('Datafile belongs to uniform sized tablespace and is
not optimally sized.');
dbms_output.put_line('Size of datafile is not a multiple of
NN*uniform_extent_size + overhead');
dbms_output.put_line('Space that cannot be used (space wastage):
'||round((space_wastage*c_file.block_size)/1024)||'K');
dbms_output.put_line('For optimal usage of space in file either resize OR
increase to: '||round(((c_file.blocks+(c_file.uni_extent-
space_wastage))*c_file.block_size)/1024)||'K');
end if;

dbms_output.put_line('.');

end if;

end loop;

end;
/
```

Script is meant for Oracle version 8 and lower

set serveroutput on

```
exec dbms_output.enable(1000000);

declare

cursor c_dbfile is
select f.tablespace_name,f.file_name,f.file_id,f.blocks
from dba_data_files f,
dba_tablespaces t
where f.tablespace_name = t.tablespace_name
and t.status = 'ONLINE'
order by f.tablespace_name,f.file_id;

cursor c_freespace(v_file_id in number) is
select block_id, block_id+blocks max_block
from dba_free_space
where file_id = v_file_id
```

```
order by block_id desc;

/* variables to check settings/values */
block_correction number;
block_size number;

/* running variable to show (possible) end-of-file */
file_min_block number;

begin

select value into block_size from v$parameter where name='db_block_size';

/* main loop */
for c_file in c_dbfile
loop
/* initialization of loop variables */
file_min_block := c_file.blocks;

begin

<<check_free

for c_free in c_freespace(c_file.file_id)
loop
/* if blocks is an uneven value there is a need to correct with -1 to compare
with end-of-file which is even */
block_correction := (0-mod(c_free.max_block,2));
if file_min_block = c_free.max_block+block_correction
then

/* free extent is at end so file can be resized */
file_min_block := c_free.block_id;
```

```
else
/* no more free extent at end of file, file cannot be further resized */
exit check_free;
end if;
end loop;
end;

/* check if file can be resized, minimal size of file 16 blocks */
if (file_min_block = c_file.blocks) or (c_file.blocks <= 16)
then

dbms_output.put_line('Tablespace: '||c_file.tablespace_name||' Datafile:
'||c_file.file_name);
dbms_output.put_line('cannot be resized no free extents found');
dbms_output.put_line('.');

else

/* file needs minimal no of blocks which does vary over versions */
if file_min_block < 16
then
file_min_block := 16;
end if;

dbms_output.put_line('Tablespace: '||c_file.tablespace_name||' Datafile:
'||c_file.file_name);
dbms_output.put_line('current size: '||(c_file.blocks*block_size)/1024||'K'||'
can be resized to: '||round((file_min_block*block_size)/1024)||'K (reduction
of: '||round(((c_file.blocks-file_min_block)/c_file.blocks)*100,2)||' %)');
dbms_output.put_line('SQL> alter database datafile '''||c_file.file_name||'''
resize '||round((file_min_block*block_size)/1024)||'K;');
dbms_output.put_line('.');
```

```
end if;

end loop;

end;
/
```

HOW TO DROP A DATAFILE FROM TABLESPACE

Sometimes we need to drop the datafile from the tablespace. We have different procedure as per Oracle release

Procedure to drop the datafile till 10gR2

There is no direct SQL statement to drop datafiles from a tablespace. In that case we need to drop the tablespace after all data has been moved to a new tablespace.

1. Create a new tablespace to hold moved objects.
2. Move all tables to the new tablespace.
3. Move all indexes to the new tablespace.
4. Move all other segments to the new tablespace.
5. Drop the old tablespace using the INCLUDING CONTENTS and datafiles option.

Procedure to drop the datafile till 10gR2

With 10gR2, we can directly drop the datafile from tablespace with some restriction

1) The database must be open.

2) If a datafile is not empty, it cannot be dropped. If you must remove a datafile that is not empty and that cannot be made empty by dropping schema objects, you must drop the tablespace that contains the datafile.

3)You cannot drop the first or only datafile in a tablespace. This means that DROP DATAFILE cannot be used with a bigfile tablespace.

4) You cannot drop datafiles in a read-only tablespace.

5)You cannot drop datafiles in the SYSTEM tablespace.

6)If a datafile in a locally managed tablespace is offline, it cannot be dropped.

Syntax

ALTER TABLESPACE DROP DATAFILE | TEMPFILE command:

```
ALTER TABLESPACE example DROP DATAFILE '+DATA/example_3.f';

ALTER TABLESPACE TEMO DROP TEMPFILE '/+DATA/temp2.dbf';
```

HOW TO OFFLINE DROP THE DATAFILE

Sometimes a datafile get missing, you get issue opening up the datafile.

Then you can use offline drop to the control file from checking it

```
alter database datafile '/u01/oracle/oradata/ex_01.dbf' offline drop;
```

The file can still be restored and recovered and back in operation.

Or if don't have any backup, then if it is index datafile, then we can recreate all index again. If it is having table segment, we can drop the segment and recreate that table from data from some test database

HOW TO CREATE DATAFILE/TABLESPACE USING OMF

OMF stands for Oracle managed file. It has following features

a) Database files are easily distinguishable from all other files.

b) Files of one database type are easily distinguishable from other database types.

c) Files are clearly associated with important attributes specific to the file type. For example, a datafile name may include the tablespace name to allow for easy association of datafile to tablespace, or an archived log name may include the thread, sequence, and creation date.

We need to set DB_CREATE_FILE_DEST parameter in the database. Once it is set datafiles are created using OMF

Structure of OMF

```
<DB_CREATE_FILE_DEST>/<db_unique_name>/<datafile>/o1_mf_%t_%u
_.dbf
```

Tablespace /datafile creation using OMF

```
CREATE TABLESPACE TEST size 800M;

Alter tablespace add datafile size 800M
```

HOW TO ASSIGN TABLESPACES TO USERS?

Users cannot create objects in a tablespace (even it's their default tablespace) unless they have a quota on it (or UNLIMITED TABLESPACE privilege).

Grant user PER access to use all space in the APPS_TX tablespace:

```
ALTER USER PER QUOTA UNLIMITED ON APPS_TX;
```

DICTIONARY VIEWS FOR VIEWING TABLESPACE INFORMATION

View	Description
V$TABLESPACE	Name and number of all tablespaces from

	the control file.
DBA_TABLESPACES, USER_TABLESPACES	Descriptions of all (or user accessible) tablespaces.
DBA_SEGMENTS, USER_SEGMENTS	Information about segments within all (or user accessible) tablespaces.
DBA_EXTENTS, USER_EXTENTS	Information about data extents within all (or user accessible) tablespaces.
DBA_FREE_SPACE, USER_FREE_SPACE	Information about free extents within all (or user accessible) tablespaces.
V$DATAFILE	Information about all datafiles, including tablespace number of owning tablespace.
V$TEMPFILE	Information about all tempfiles, including tablespace number of owning tablespace.
DBA_DATA_FILES	Shows files (datafiles) belonging to tablespaces.
DBA_TEMP_FILES	Shows files (tempfiles) belonging to temporary tablespaces.
V$TEMP_EXTENT_MAP	Information for all extents in all locally managed temporary tablespaces.
V$TEMP_EXTENT_POOL	For locally managed temporary tablespaces: the state of temporary space cached and used for by each instance.
V$TEMP_SPACE_HEADER	Shows space used/free for each tempfile.
DBA_USERS	Default and temporary tablespaces for all users.
DBA_TS_QUOTAS	Lists tablespace quotas for all users.
V$SORT_SEGMENT	Information about every sort segment in a given instance. The view is only updated when the tablespace is of the TEMPORARY type.
V$SORT_USER	Temporary sort space usage by user and temporary/permanent tablespace.

TO LIST TABLESPACES AND ALL IMPORTANT PROPERTIES:

To list the names and various other of all tablespaces in a database, use the following query on the DBA_TABLESPACES view:

```
SELECT TABLESPACE_NAME "TABLESPACE",
   EXTENT_MANAGEMENT, FORCE_LOGGING, BLOCK_SIZE,
SEGMENT_SPACE_MANAGEMNENT
   FROM DBA_TABLESPACES;
```

TO LIST THE DATAFILES AND ASSOCIATED TABLESPACES OF A DATABASE

To list the names, sizes, and associated tablespaces of a database, enter the following query on the DBA_DATA_FILES view

```
SELECT FILE_NAME, BLOCKS, TABLESPACE_NAME
   FROM DBA_DATA_FILES;
```

TO DISPLAY STATISTICS FOR FREE SPACE (EXTENTS) OF EACH TABLESPACE

To produce statistics about free extents and coalescing activity for each tablespace in the database, enter the following query:

```
SELECT TABLESPACE_NAME "TABLESPACE", FILE_ID,
   COUNT(*)    "PIECES",
   MAX(blocks) "MAXIMUM",
   MIN(blocks) "MINIMUM",
   AVG(blocks) "AVERAGE",
   SUM(blocks) "TOTAL"
   FROM DBA_FREE_SPACE
GROUP BY TABLESPACE_NAME, FILE_ID;
```

HOW TO CHECK HIGHEST ALLOCATED EXTENT?

```
column file_name format a50;
column tablespace_name format a15;
column highwater format 9999999999;
```

```
set pagesize 9999

select a.tablespace_name
,a.file_name
,(b.maximum+c.blocks-1)*d.db_block_size highwater
from dba_data_files a
,(select file_id,max(block_id) maximum
from dba_extents
group by file_id) b
,dba_extents c
,(select value db_block_size
from v$parameter
where name='db_block_size') d
where a.file_id = b.file_id
and c.file_id = b.file_id
and c.block_id = b.maximum
order by a.tablespace_name,a.file_name
/
```

To check the free SPACE, largest free chunck and no of free chunck in tablespace.

```
set feedback off
set echo off
set numwidth 15
set linesize 150
set pages 1000

Accept tname Prompt "Enter Tablespace Name : "
Select (Sum(bytes)/1024/1024) Free_space_MB,(max(bytes)/1024/1024)
Largest_Free_chunck_MB,count(*) No_of_free_chunk
```

```
from dba_free_space where tablespace_name=upper('&tname');
```

TO CHECK THE TOTAL SPACE ALLOCATED TO TABLESPACE.

```

Select (sum(bytes)/1024/1024) Space_allocated
from dba_data_files
where tablespace_name=upper('&tname');
```

TO CHECK ALL TABLESPACE INFORMATION IN THE DATABASE

```
set echo off feedback off verify off pages 60

col tablespace_name format a16 head 'Tablespace Name'
col initial_extent format 99,999,999 head 'Initial|Extent(K)'
col next_extent format 99,999,999 head 'Next|Extent(K)'
--col min_extents format 999 head 'Min|Ext'
col max_extents format a4 head 'Max|Ext'
col pct_increase format 999 head 'Pct|Inc'
col extent_management format a10 head 'Extent|Management'
col allocation_type format a10 head 'Allocation|Type'
col status format a7 head 'Status'

select	tbs.tablespace_name
,	tbs.initial_extent
,	tbs.next_extent
--,	tbs.min_extents
,	decode(tbs.max_extents,2147483645,'UL',tbs.max_extents)
max_extents
,	tbs.pct_increase
,	tbs.extent_management
,	tbs.allocation_type
```

```
,       tbs.status
from  dba_tablespaces tbs
order by 1
/
```