

UNIT 3

Tablespace and Storage Management

(4 Hours) — Database Administration (CACS405)

*Note: These notes are prepared for study purposes and **may contain errors or omissions**. Syntax, file paths and screen names differ between Oracle versions and operating systems (examples here follow Oracle 19c on Linux). Please verify important points with your textbook, class notes and the official Oracle documentation, and test commands on a practice database, never on a production system.*

3.1 Overview of Oracle Storage Structures

Oracle separates **logical** storage (how data is organised for users and the DBA) from **physical** storage (files on the operating system). This lets a DBA move or add files without changing applications.

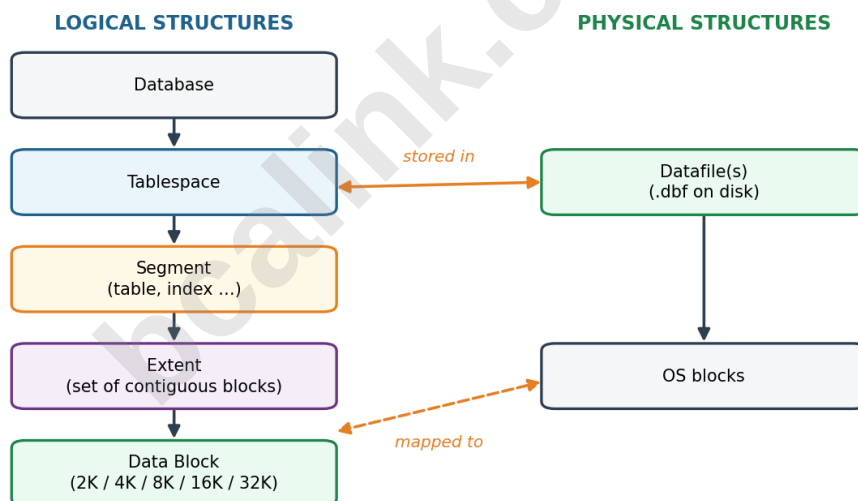


Figure 3.1: Logical and physical storage structures

Logical unit	Description
Data block	Smallest unit of I/O (default 8 KB). Set by DB_BLOCK_SIZE at database creation.
Extent	A set of contiguous data blocks allocated together to one segment.
Segment	All the extents of one object such as a table, index, undo or temporary data.
Tablespace	Logical container for segments; made of one or more datafiles.

3.1.1 Types of Tablespaces

Tablespace	Purpose
SYSTEM	Data dictionary and system-managed objects. Always online. Do not store user data here.
SYSAUX	Auxiliary tablespace for components such as AWR and Enterprise Manager repository.
UNDO	Stores undo data for rollback, read consistency and flashback. One active undo tablespace per instance (UNDO_TABLESPACE).
TEMP (temporary)	Used for sorting, hash joins and temporary tables. Uses <i>tempfiles</i> .
USERS / application	Permanent tablespaces that hold user tables and indexes.

Smallfile tablespaces (default) can have many datafiles; a **bigfile** tablespace has exactly one very large datafile, which simplifies management for large databases. Tablespaces are normally **locally managed** (extent information stored in bitmaps in the datafile headers) with **Automatic Segment Space Management (ASSM)**. The older dictionary-managed method is obsolete.

3.2 Working with Tablespaces and Data Files

3.2.1 Creating Tablespaces

```
-- permanent tablespace with auto-extending datafile
CREATE TABLESPACE app_data
  DATAFILE '/u01/oradata/ORCL/app_data01.dbf' SIZE 100M
  AUTOEXTEND ON NEXT 10M MAXSIZE 1G
  EXTENT MANAGEMENT LOCAL
  SEGMENT SPACE MANAGEMENT AUTO;

-- temporary tablespace
CREATE TEMPORARY TABLESPACE temp2
  TEMPFILE '/u01/oradata/ORCL/temp02.dbf' SIZE 50M;

-- undo tablespace
CREATE UNDO TABLESPACE undotbs2
  DATAFILE '/u01/oradata/ORCL/undotbs02.dbf' SIZE 200M;

-- bigfile tablespace
CREATE BIGFILE TABLESPACE big_ts
  DATAFILE '/u01/oradata/ORCL/big_ts01.dbf' SIZE 5G;
```

Listing 3.1: Creating different kinds of tablespaces

Note: If Oracle Managed Files (OMF) is configured with DB_CREATE_FILE_DEST, you can simply write `CREATE TABLESPACE app_data;` and Oracle chooses the file name and location.

3.2.2 Adding and Managing Datafiles

When a tablespace becomes full you can (1) add a new datafile, (2) resize an existing datafile, or (3) enable auto-extension.

```
-- 1. add a datafile
ALTER TABLESPACE app_data
  ADD DATAFILE '/u01/oradata/ORCL/app_data02.dbf' SIZE 200M;

-- 2. resize a datafile
ALTER DATABASE DATAFILE '/u01/oradata/ORCL/app_data01.dbf' RESIZE 500M;

-- 3. enable autoextend
ALTER DATABASE DATAFILE '/u01/oradata/ORCL/app_data01.dbf'
  AUTOEXTEND ON NEXT 50M MAXSIZE 2G;

-- add a tempfile
ALTER TABLESPACE temp2 ADD TEMPFILE '/u01/oradata/ORCL/temp03.dbf' SIZE 50M;

-- rename a tablespace, change its state
ALTER TABLESPACE app_data RENAME TO app_data_new;
ALTER TABLESPACE app_data_new OFFLINE NORMAL;
ALTER TABLESPACE app_data_new ONLINE;

ALTER TABLESPACE app_data_new READ ONLY;
ALTER TABLESPACE app_data_new READ WRITE;

-- drop a tablespace and its files
DROP TABLESPACE app_data_new INCLUDING CONTENTS AND DATAFILES;
```

Listing 3.2: Managing tablespaces and datafiles

To **move or rename** a datafile of a permanent tablespace: take the tablespace offline (or mount the database for SYSTEM/UNDO files), copy the file at OS level, run `ALTER DATABASE RENAME FILE 'old' TO 'new';` (or `ALTER TABLESPACE ... RENAME DATAFILE`), and bring the tablespace online. (From 12c, `ALTER DATABASE MOVE DATAFILE` can move a file online.)

3.2.3 Monitoring Tablespaces

```

SELECT tablespace_name, status, contents, extent_management FROM dba_tablespaces;
SELECT file_name, tablespace_name, bytes/1024/1024 mb, autoextensible
FROM dba_data_files;
SELECT file_name, tablespace_name, bytes/1024/1024 mb FROM dba_temp_files;

-- used and free space per tablespace (MB)
SELECT d.tablespace_name,
       ROUND(d.total_mb) total_mb,
       ROUND(d.total_mb - NVL(f.free_mb,0)) used_mb,
       ROUND(NVL(f.free_mb,0)) free_mb
FROM (SELECT tablespace_name, SUM(bytes)/1048576 total_mb
      FROM dba_data_files GROUP BY tablespace_name) d
LEFT JOIN
      (SELECT tablespace_name, SUM(bytes)/1048576 free_mb
      FROM dba_free_space GROUP BY tablespace_name) f
ON d.tablespace_name = f.tablespace_name;

```

Listing 3.3: Queries to monitor space

Example 3.1

The USERS tablespace is 95% full. List two ways to solve the problem.

Solution

- (a) ALTER TABLESPACE users ADD DATAFILE '/u01/oradata/ORCL/users02.dbf' SIZE 500M;
- (b) Enable autoextend or resize the existing datafile: ALTER DATABASE DATAFILE '...users01.dbf' RESIZE 1G;
- Also consider purging unneeded data or moving large objects.

3.3 Managing Control Files

The **control file** is a small binary file that records the physical structure of the database. Without it, the database cannot be mounted. It is read at the MOUNT stage and updated continuously by Oracle.

Contents of a control file:

- Database name and unique DBID, and creation timestamp.
- Names and locations of all datafiles and online redo log files.
- Current log sequence number and checkpoint information.
- Tablespace information, archived log history and RMAN backup metadata.

3.3.1 Multiplexing Control Files

Because the control file is so important, Oracle recommends keeping **at least two copies** (three is common) on **different disks or controllers**. Oracle writes to all copies at the same time and reads from the first one listed in the `CONTROL_FILES` parameter.

```
-- 1. see the current control files
SELECT name FROM v$controlfile;
SHOW PARAMETER control_files

-- 2. record the new list in the SPFILE
ALTER SYSTEM SET control_files =
  '/u01/oradata/ORCL/control01.ctl',
  '/u02/oradata/ORCL/control02.ctl'
  SCOPE = SPFILE;

-- 3. shut down, copy the existing file at OS level, and restart
SHUTDOWN IMMEDIATE
-- $ cp /u01/oradata/ORCL/control01.ctl /u02/oradata/ORCL/control02.ctl
STARTUP
```

Listing 3.4: Steps to multiplex a control file

3.3.2 Backing Up Control Files

```
ALTER DATABASE BACKUP CONTROLFILE TO '/backup/control.bkp';
ALTER DATABASE BACKUP CONTROLFILE TO TRACE; -- writes a re-create script
```

RMAN can also back up the control file automatically (`CONFIGURE CONTROLFILE AUTOBACKUP ON`; see Unit 7). If all copies are lost, the control file can be restored from a backup or re-created with `CREATE CONTROLFILE`.

3.4 Online Redo Logs

The **online redo log** records every change made to the database so that committed work can be recovered after a failure. It consists of two or more **redo log groups**, each with one or more identical **members** (files). LGWR writes to one group at a time (the **CURRENT** group); when it is full, a **log switch** moves LGWR to the next group. After the last group, LGWR goes back to the first, reusing the log circularly.

Group status	Meaning
CURRENT	LGWR is writing to this group now.
ACTIVE	Not current, but still needed for instance recovery (checkpoint not complete). If ARCHIVELOG, it may also be waiting to be archived.

Group status	Meaning
INACTIVE	No longer needed for instance recovery; can be reused (after archiving in ARCHIVELOG mode).
UNUSED	Newly added group not yet used.

```
SELECT group#, sequence#, bytes/1024/1024 mb, members, status, archived FROM v$log;
SELECT group#, member, status FROM v$logfile;
```

```
ALTER DATABASE ADD LOGFILE GROUP 4
  ('/u01/oradata/ORCL/redo04a.log', '/u02/oradata/ORCL/redo04b.log') SIZE 200M;
ALTER DATABASE ADD LOGFILE MEMBER '/u02/oradata/ORCL/redo01b.log' TO GROUP 1;
ALTER DATABASE DROP LOGFILE MEMBER '/u02/oradata/ORCL/redo01b.log';
ALTER DATABASE DROP LOGFILE GROUP 4;
```

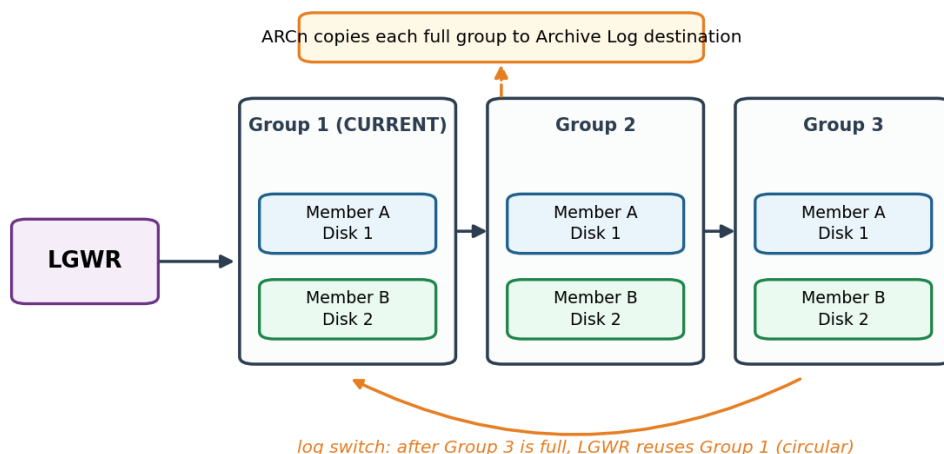
```
ALTER SYSTEM SWITCH LOGFILE;    -- force a log switch
ALTER SYSTEM CHECKPOINT;        -- force a checkpoint
```

Listing 3.5: Managing online redo log groups and members

- A database needs **at least two** redo log groups.
- A group that is CURRENT or ACTIVE cannot be dropped; force a switch/checkpoint first.
- In ARCHIVELOG mode a group cannot be reused until it has been archived.
- Size groups so that a log switch occurs every 15–30 minutes under normal load (too small causes frequent checkpoints).

3.4.1 Multiplexing Online Redo Logs

Multiplexing means keeping two or more members in each group, on different disks. LGWR writes the same information to all members of the current group. If one member is damaged, the other keeps the database running and no committed data is lost. All groups should have the same number of members and members of a group should be the same size.



Members of one group hold identical copies (multiplexing) – put them on different disks

Figure 3.2: Redo log groups, multiplexed members and archiving

3.5 Archive Logs and Database Archiving

A database runs in one of two log modes. In **NOARCHIVELOG** mode, a redo log group is overwritten when it is reused, so redo history is lost. In **ARCHIVELOG** mode, each filled group is copied by ARCn into an **archived redo log** file before it can be reused. Archived logs plus a backup allow recovery up to the point of failure and allow hot (online) backups.

Point	NOARCHIVELOG	ARCHIVELOG
Redo history	Overwritten — not kept	Saved as archived logs
Backup type	Cold (consistent) backups only, database must be closed	Hot backups possible while database is open
Recovery	Only to the time of the last full backup	Complete or point-in-time recovery
Use	Test/development databases	Production databases

```

SQL> ARCHIVE LOG LIST; -- show mode and destination
SQL> SELECT log_mode FROM v$database;

-- switch to ARCHIVELOG mode
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP MOUNT
SQL> ALTER DATABASE ARCHIVELOG;
SQL> ALTER DATABASE OPEN;

-- set the archive destination and file format
SQL> ALTER SYSTEM SET log_archive_dest_1 = 'LOCATION=/u03/arch' SCOPE = BOTH;
SQL> ALTER SYSTEM SET log_archive_format = 'arch_%t_%s_%r.arc' SCOPE = SPFILE;

SQL> SELECT name, sequence#, first_time FROM v$archived_log ORDER BY sequence#;
SQL> ALTER SYSTEM ARCHIVE LOG CURRENT; -- switch and archive the current log

```

If no destination is set, archived logs go to the **Fast Recovery Area (FRA)** when `DB_RECOVERY_FILE_DEST` is configured. If the archive destination becomes full, the database can hang until space is freed, so archive space must be monitored.

Example 3.2

Why must the database be in MOUNT state to change the log mode?

Solution

The log mode is stored in the control file. It can be changed only when the database is mounted but not open, so that no user is changing data while the mode is switched. After the change the database is opened normally. Take a full backup afterwards.

Key Takeaways

- ✓ Logical hierarchy: **tablespace** → **segment** → **extent** → **block**; each tablespace is stored in one or more datafiles.
- ✓ Special tablespaces: SYSTEM, SYSAUX, UNDO and TEMP. Use locally managed tablespaces with ASSM.
- ✓ Grow space by adding a datafile, resizing, or enabling AUTOEXTEND.
- ✓ The **control file** holds the physical structure; multiplex it on separate disks and back it up.
- ✓ Online redo logs: at least 2 groups; multiplex members on different disks; LGWR switches groups circularly.
- ✓ **ARCHIVELOG** mode saves redo history and enables hot backups and full recovery; change it in MOUNT state.

Reminder: these notes may contain errors — cross-check with the official syllabus, your textbook and Oracle documentation.

— End of Unit 3 —