

Administration Base de Données

ch3: Understanding Storage and Space Management

Working with Oracle Managed File Tablespaces

Oracle Managed Files (OMF)

Purpose:

Simplifies file management by letting Oracle **automatically create and manage data/temp files**.

Enabling OMF:

Set the following initialization parameter:

```
ALTER SYSTEM SET db_create_file_dest = '/u02/oradata/' SCOPE=BOTH;
```

Creating a Tablespace Using OMF:

```
CREATE TABLESPACE hr_data;
```

→ File: `o1_mf_hr_data_<unique_id>.dbf`

→ Location: `<DBNAME>/datafile/` inside `db_create_file_dest`.

→ Defaults: 100MB file with **autoextend** enabled, **SMALLFILE** by default.

Extent Management

What Is "Extent Management"?

In Oracle, when you create a table or index, it needs **space** to store data.

- Oracle allocates that space in **chunks** called **extents**.
- Oracle must **track** which extents are free or used.
- That tracking is called **extent management**.

Types of Extent Management

1. Dictionary-Managed Tablespaces

- **Extent info stored in the data dictionary** (`FET$`, `UET$`).
- Generates **undo/rollback** info during extent changes.
- **Less efficient** than locally managed.
- Example:

```
CREATE TABLESPACE appl_data  
DATAFILE '/disk3/oradata/DB01/appl_data01.dbf' SIZE 100M  
EXTENT MANAGEMENT DICTIONARY;
```

2. Locally Managed Tablespaces (LMT).

- **Default in Oracle 12c.**
- Uses **bitmaps** in data files to track extents.
- **More efficient**, no recursive SQL or undo generation.

► Extent Allocation Options:

a) **UNIFORM**

- Fixed extent size (default 1MB or specified).
- Ideal for temp tablespaces.
- Example:

```
CREATE TABLESPACE hr_index
DATAFILE '/u02/oradata/12CR11/hr_index01.dbf' SIZE 2G
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 10M;
```

b) **AUTOALLOCATE**

- Oracle automatically adjusts extent sizes as the segment grows.
- Start with 64KB → 1MB → 8MB → 64MB, etc.
- Best for **mixed workloads** (small + large tables).
- Example:

```
CREATE TABLESPACE hr_index
DATAFILE '/u02/oradata/12CR11/hr_index01.dbf' SIZE 2G
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

Choosing Segment Space Management

What Is Segment Space Management?

Segment space management is **how Oracle tracks free space inside extents**—basically, inside the blocks that store rows in a table or index.

It determines **which blocks** are available for new rows (inserts), based on how full they are.

Two Types of Segment Space Management

1. Manual Segment Space Management

► How it works:

- Uses **free block lists**.
- Relies on two key parameters:
 - **PCTFREE**: Minimum % of block to **keep free** (for updates).
 - **PCTUSED**: If space used in a block drops **below this %**, it goes back on the **insert list**.

```
CREATE TABLESPACE hr_index
```

```
DATAFILE '/u02/oradata/hr_index01.dbf' SIZE 2G
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
SEGMENT SPACE MANAGEMENT MANUAL;
```

► Disadvantages:

- More overhead
- You must tune `PCTFREE` and `PCTUSED`
- Not recommended for modern use

Automatic Segment Space Management (AUTO)

► How it works:

- Uses **bitmaps** instead of free lists.
- Oracle internally tracks which blocks have enough space—you **don't have to manage anything**.
- Ignores `PCTFREE`, `PCTUSED`, `FREELISTS`, etc.

► Best for:

- Most modern use cases
- Large or mixed workloads
- Easier maintenance and better performance

```
CREATE TABLESPACE hr_index
DATAFILE '/u02/oradata/hr_index01.dbf' SIZE 2G
EXTENT MANAGEMENT LOCAL AUTOALLOCATE
SEGMENT SPACE MANAGEMENT AUTO;
```

Choosing Other Tablespace Options

Table 4

DEFAULT STORAGE	Default extent settings	Dictionary-managed only
<code>BLOCKSIZE</code>	Sets tablespace block size	Both types
<code>MINIMUM EXTENT</code>	Prevents fragmentation	Dictionary-managed only
<code>LOGGING</code> / <code>NOLOGGING</code>	Controls redo logging	Both types
<code>FORCE LOGGING</code>	Forces logging for all objects	Both types
<code>ONLINE</code> / <code>OFFLINE</code>	Sets availability after creation	Both types
<code>FLASHBACK ON/OFF</code>	Enables/disables flashback	Both types

```
CREATE TABLESPACE APPL_DATA
DATAFILE '/disk3/oradata/DB01/appl_data01.dbf' SIZE 100M
DEFAULT STORAGE (
    INITIAL 256K
    NEXT 256K
```

```
MINEXTENTS 2
PCTINCREASE 0
MAXEXTENTS 4096)
BLOCKSIZE 16K
MINIMUM EXTENT 256K
LOGGING
ONLINE
FORCE LOGGING
FLASHBACK ON
EXTENT MANAGEMENT DICTIONARY
SEGMENT SPACE MANAGEMENT MANUAL;
```

Temporary Tablespaces

Purpose

- Used **only** for **temporary segments** during:
 - `ORDER BY`, `GROUP BY`, `CREATE INDEX`
 - Hash joins**, temporary table operations, etc.
- Frees permanent tablespaces from temporary segment overhead.

Key Characteristics

- Created with `CREATE TEMPORARY TABLESPACE`
- Use `TEMPFILE` instead of `DATAFILE`
- No redo logging (changes aren't written to redo logs)
- Not backed up
- Not fully allocated at creation (especially on UNIX)

```
CREATE TEMPORARY TABLESPACE temp
TEMPFILE '/u01/oradata/12CR11/temp01.dbf' SIZE 1G;
```

Undo Tablespaces

✓ Purpose

Used to store **undo segments**, which support:

- Explicit/implicit **ROLLBACK**
- Read consistency**
- Flashback queries**
- Recovery from logical corruption**

```
CREATE UNDO TABLESPACE undo
DATAFILE '/ORADATA/PROD/UND001.DBF' SIZE 2G;
```



Seul le tablespace Undo est concerné par le paramètre `UNDO_RETENTION` : Il détermine combien de temps Oracle tente de conserver les données d'annulation (undo) dans le tablespace Undo, même après la fin de la transaction.

Dropping Tablespaces (DROP TABLESPACE)

Basic command:

```
DROP TABLESPACE USER_DATA;
```

To drop the tablespace contents (tables, indexes, etc.):

| If the tablespace is not empty:

```
DROP TABLESPACE dba_sandbox INCLUDING CONTENTS;
```

To also drop foreign key constraints related:

```
DROP TABLESPACE USER_DATA INCLUDING CONTENTS CASCADE CONSTRAINTS;
```

To also drop the associated datafiles:

```
sql> DROP TABLESPACE hr_data INCLUDING CONTENTS AND DATAFILES;
```



Le tablespace **SYSTEM** ne peut pas être supprimé. SYSAUX peut être supprimé en mode **UPGRADE** et avec **SYSDBA**.

Modification d'un Tablespace (ALTER TABLESPACE)

Possibles Modifications :

- Change default storage clauses.
- Add datafiles.
- Change to read-only / read-write.
- Make offline or online.
- Enable/disable flashback or guaranteed retention.
- Resize or rename files.

→ Adding a Datafile :

```
ALTER TABLESPACE receivables ADD DATAFILE  
'/u02/oradata/ORAI10/receivables01.dbf' SIZE 2G;
```

→ Putting a tablespace online / offline:

► Online :

```
ALTER TABLESPACE USER_DATA ONLINE;
```

II Offline modes :

- **NORMAL:** no recovery needed.
- **TEMPORARY:** checkpoint, recovery possible.
- **IMMEDIATE:** no checking, recovery mandatory.
- **FOR RECOVER:** used for point-in-time recovery.

Exemple :

```
ALTER TABLESPACE USER_DATA OFFLINE IMMEDIATE;
```



! SYSTEM ne peut jamais être mis offline.

Read Only :

- Marquer comme lecture seule :

```
ALTER TABLESPACE sales2007 READ ONLY;
```

- Revenir en lecture/écriture :

```
ALTER TABLESPACE sales2007 READ WRITE;
```

Mode Backup manuel (non-RMAN) :

Before manual backup :

```
ALTER TABLESPACE system BEGIN BACKUP;
```

After :

```
ALTER TABLESPACE system END BACKUP;
```



▲ Si tu oublies `END BACKUP`, un redémarrage déclenchera une demande de **récupération média**.

Getting Tablespace Information:

Vues utiles :

- DBA_TABLESPACES
- DBA_DATA_FILES
- V\$TABLESPACE

Example :

```
SELECT tablespace_name, status, contents,  
       extent_management AS extents,  
       segment_space_management AS free_space  
FROM dba_tablespaces;
```

Use EM Database Express to manage tablespaces

EM Database Express (EMDE) is a graphical user interface (GUI) used to:

- View tablespaces (menu Storage → Tablespaces)
- Create, modify, drop a tablespace
- Add or resize datafiles
- Change status (online/offline) of a tablespace

→ It's an alternative to SQL*Plus (command line).

Datafile Management

Creation :

- A datafile is created when creating a tablespace.
- It belongs to one tablespace and one database.

→ If Oracle Managed Files (OMF) is not used:

- You must manage the filename, path, and size yourself.

Possible operations on Datafiles:

Resize a file:

```
ALTER DATABASE DATAFILE '...' RESIZE 1500M;
```

- You can **increase** or **decrease** (if nothing specified after new size).

Auto-extend (automatic growth) :

```
CREATE TABLESPACE ...  
DATAFILE '...' SIZE 500M AUTOEXTEND ON NEXT 100M MAXSIZE 2000M;
```

```
-- Pour le cas de modification
```

```
ALTER DATABASE DATAFILE '...' AUTOEXTEND ON NEXT 100M MAXSIZE 2000M;
```

- `NEXT` : the file grows in increments of 100 MB when space runs out. For example, if full at 500 MB, it becomes 600 MB, then 700 MB, etc.
- `MAXSIZE` : maximum size
- `AUTOEXTEND OFF` : to disable

Taking a datafile offline / online:

Offline (often in case of corruption) :

| You tell Oracle that the datafile becomes **temporarily unusable** (usually due to **corruption, disk failure**, etc.).

```
ALTER DATABASE DATAFILE '...' OFFLINE;
```

Online (sometimes requires recovery) :

```
ALTER DATABASE DATAFILE '...' ONLINE;
```

ARCHIVELOG vs NOARCHIVELOG

Mode	Can be recovered?	Required syntax
ARCHIVELOG	✓ Yes	OFFLINE
NOARCHIVELOG	✗ No	OFFLINE FOR DROP

Rename or move a datafile

En mode online :

```
ALTER DATABASE MOVE DATAFILE 'ancien' TO 'nouveau';
```

- Oracle copies, updates metadata, and deletes the old file (unless KEEP is used).
- You can use REUSE if the file already exists at the destination.

mode offline (old methode) :

1. Take the tablespace offline:

```
ALTER TABLESPACE USER_DATA OFFLINE;
```

2. Copy or move the file manually (OS command)
3. Rename inside the database:


```
ALTER DATABASE RENAME FILE 'ancien' TO 'nouveau';
```

OR

```
ALTER TABLESPACE USER_DATA RENAME DATAFILE 'ancien' TO 'nouveau';
```

4. Bring it back online:

```
ALTER TABLESPACE USER_DATA ONLINE;
```



Le **SYSTEM tablespace** ne peut pas être mis offline → on ne peut **pas utiliser RENAME FILE**, mais on peut utiliser **MOVE DATAFILE**.

On Renomer et Resize les datfile de SYSTEM Tablespace en mode MOUNT

Relocating an Entire Tablespace

If multiple files need to be moved:

1. Take the tablespace offline
2. Copy all files
3. Rename all at once:

```
ALTER DATABASE RENAME FILE 'a', 'b', 'c' TO 'x', 'y', 'z';
```

4. Bring it back online

Relocating Files Belonging to Multiple Tablespaces

If you move files of several tablespaces, you must:

- Shut down the database (`SHUTDOWN`)
- Copy the files
- Start in MOUNT mode
- Rename with `ALTER DATABASE RENAME FILE`
- Open the database (`ALTER DATABASE OPEN`)

Parameters:

- `DB_CREATE_FILE_DEST` — For data files
- `DB_CREATE_ONLINE_LOG_DEST_n` — For redo logs and control files

Remarks !!

The SQL command:

```
TRUNCATE TABLE CUST_INFO;
```

means: delete all rows from the table `CUST_INFO` quickly and irreversibly, but do not delete the table itself or its structure. **It is faster than delete.**

Types de Startup Modes ou Mount Options dans Oracle :

Voici les **états ou modes de démarrage ("startup modes")** d'une base de données Oracle :

1. NOMOUNT

- Lance l'instance Oracle sans monter la base de données.
- Utilisé pour la **création** de la base ou la **restauration** (RMAN).
- Charge les fichiers de paramètres (`spfile` ou `pfile`), mais pas le fichier de contrôle.

2. MOUNT

- Monte la base de données, c'est-à-dire lit le **control file**.
- Ne permet pas l'accès aux données (les fichiers de données ne sont pas encore ouverts).
- Utilisé pour certaines opérations d'administration comme la **récupération**.

3. OPEN

- Ouvre la base de données : les utilisateurs peuvent se connecter et utiliser les données.
- Toutes les opérations sont possibles : lecture, écriture, transactions.

Options principales de STARTUP

Option	Description	Use Case
<code>NOMOUNT</code>	Démarre l'instance sans monter la base (le control file n'est pas lu).	Used to create a new database or re-create control files
<code>MOUNT</code>	Monte la base de données (lit le control file), mais n'ouvre pas encore les datafiles.	Used to recover a database or perform maintenance (e.g., renaming datafiles)
<code>OPEN</code> (défaut)	Monte et ouvre la base de données – elle est disponible pour les utilisateurs.	Used in normal operations
<code>FORCE</code>	Force le redémarrage de l'instance. Oracle fait d'abord un <code>SHUTDOWN ABORT</code> , puis un <code>STARTUP</code> .	Forces instance shutdown and restart
<code>RESTRICT</code>	Ouvre la base de données en accès restreint , seuls les utilisateurs avec le privilège <code>RESTRICTED SESSION</code> peuvent s'y connecter.	Allows only DBA access , for maintenance
<code>PFILE=...</code>	Spécifie un fichier <code>PFILE</code> particulier à utiliser pour démarrer la base.	

PFile vs. SPFile

Le **PFILE** (*Parameter File*) est un fichier texte utilisé par Oracle pour **initialiser** la base de données au démarrage. Il contient les **paramètres d'initialisation** nécessaires pour lancer l'instance Oracle.

Le **SPFILE** (Server Parameter File) est un **fichier binaire** utilisé par Oracle pour stocker les **paramètres d'initialisation** de l'instance. Contrairement au **PFILE**, qui est un fichier texte, le SPFILE **ne peut pas être modifié directement avec un éditeur de texte** — il se modifie via des commandes SQL.

✔ **le PFILE (Parameter File) et le SPFILE (Server Parameter File) ont la même utilité principale : ils servent tous les deux à démarrer une instance Oracle en lui fournissant les paramètres d'initialisation (comme la mémoire, les chemins de fichiers, etc.).**

Caractéristique	PFILE	SPFILE
Format	Texte (lisible et modifiable)	Binaire (non modifiable directement)
Modification	Manuelle avec un éditeur (Notepad, vi)	Par commande SQL (<code>ALTER SYSTEM</code>)
Dynamique	❌ Non	✔ Oui (certains paramètres modifiables à chaud)
Utilisé par défaut ?	❌ Non (sauf si spécifié au démarrage)	✔ Oui (s'il existe)
Création	Écrit manuellement	Généré avec <code>CREATE SPFILE FROM PFILE</code>
Emplacement typique	<code>\$ORACLE_HOME/dbs/init<SID>.ora</code>	<code>\$ORACLE_HOME/dbs/spfile<SID>.ora</code>

UNDO_MANAGEMENT

Le paramètre `UNDO_MANAGEMENT` dans Oracle détermine **le mode de gestion des données UNDO**, qui sont utilisées pour :

- annuler une transaction (ROLLBACK),
- garantir la cohérence des lectures (read consistency),
- permettre la récupération après incident (recovery).

Valeurs possibles de `UNDO_MANAGEMENT`

Valeur	Signification
<code>AUTO</code> (recommandé)	Oracle gère automatiquement l'espace UNDO via une <i>undo tablespace</i> .
<code>MANUAL</code>	C'est à l'administrateur de gérer les segments UNDO manuellement.

Mode AUTO (le plus utilisé)

Quand `UNDO_MANAGEMENT=AUTO`, Oracle :

- utilise une **undo tablespace** (définie par le paramètre `UNDO_TABLESPACE`),
- alloue, libère et recycle l'espace automatiquement,
- simplifie la gestion pour les DBA.

```
SHOW PARAMETER undo_management;
```

Exemple de sortie :

NAME	TYPE	VALUE
undo_management	string	AUTO

Types of SHUTDOWN

1. SHUTDOWN NORMAL

Comportement :

- N'autorise **aucune nouvelle connexion**.
- Attend que **tous les utilisateurs se déconnectent** avant de fermer.
- Base de données proprement arrêtée.
- **C'est le mode par défaut** si aucun paramètre n'est précisé.

Utilisation :

- En production ou en maintenance planifiée.

2. SHUTDOWN IMMEDIATE

Comportement :

- N'autorise **aucune nouvelle connexion**.
- **Déconnecte immédiatement** les utilisateurs actifs.
- **Termine les transactions en cours**.
- Base arrêtée proprement.

Avantages :

- Plus rapide que **NORMAL**.
- Utilisé souvent en administration.

3. SHUTDOWN TRANSACTIONAL

Comportement :

- N'autorise plus de nouvelles connexions.
- Attend que **les transactions en cours se terminent**.
- Puis ferme la base de données proprement.

Cas d'usage :

- Quand on veut éviter d'interrompre les utilisateurs en pleine transaction (comme des achats en ligne).

4. SHUTDOWN ABORT

Comportement :

- **Interrompt immédiatement tout** (sessions, transactions).
- Ne termine **aucune transaction** proprement.
- Aucun nettoyage de mémoire ni des fichiers.
- **Nécessite une récupération** automatique au prochain démarrage (`instance recovery` via SMON).

À utiliser en dernier recours :

- Si `IMMEDIATE` ou `NORMAL` ne fonctionnent pas (ex: base figée).

Comparaison

Commande	Nouvelle connexion ?	Attente de fin de tran	Arrêt propre ?	Vitesse
SHUTDOWN NORMAL	✗ Non	✓ Oui	✓ Oui	Lente
SHUTDOWN IMMEDIATE	✗ Non	✗ Non	✓ Oui	Rapide
SHUTDOWN TRANSACTIONAL	✗ Non	✓ Oui	✓ Oui	Moyen
SHUTDOWN ABORT	✗ Non	✗ Non	✗ Non	Très Rapide

Automatic Space Management Features

1. Locally Managed Tablespaces (LMT)

- Use `EXTENT MANAGEMENT LOCAL` for efficient extent tracking using bitmaps instead of data dictionary tables.

2. Automatic Segment Space Management (ASSM)

- Use `SEGMENT SPACE MANAGEMENT AUTO` to manage free space within segments using bitmaps.
- More efficient than manual management with free lists.

3. Autoextend for Datafiles

- Enables datafiles to grow automatically as space is required using `AUTOEXTEND ON`.
- Helps prevent out-of-space errors.

4. Oracle Managed Files (OMF)

- Automates file creation, naming, and deletion using initialization parameters like `DB_CREATE_FILE_DEST`.
- Simplifies storage administration.

5. Deferred Segment Creation

- Segments for tables/indexes are created **only when the first row is inserted**, not at object creation.
- Useful for ERP systems with thousands of unused tables.

- Saves storage and reduces install time.

6. Automatic Undo Management

- Uses **undo tablespaces** instead of manually managing rollback segments.
- Oracle handles undo allocation, retention, and space reuse.

7. Server-Generated Alerts

- Oracle automatically monitors tablespace usage and other metrics.
- Alerts are triggered when thresholds are crossed (e.g., low free space).
- Viewed in tools like **Enterprise Manager (EM)** or queried via `DBA_OUTSTANDING_ALERTS`.

8. Compression Options

- **Basic compression:** Compresses data during direct-path loads only (`INSERT /*+ APPEND */`).
- **OLTP compression:** Ideal for transactional systems; compresses data at block-level as needed.
- Reduces storage footprint without major performance overhead.

9. Monitoring Space Usage

- Use dictionary views like:
 - `DBA_DATA_FILES`
 - `DBA_FREE_SPACE`
 - `DBA_SEGMENTS`
 - `DBA_TABLESPACES`
- Queries can aggregate allocation, free space, and used space.
- Tools like **EM Database Express** and **SQL Developer** offer GUI-based monitoring.

Default Tablespace Size (If not specified)

Case	Default Size
<code>CREATE TABLESPACE ... DATAFILE 'file.dbf' without SIZE</code>	✅ 100 MB
<code>TEMPORARY TABLESPACE without SIZE</code>	Also defaults to 100 MB , unless specified

CHECKPOINT

Checkpoints occur during:

Event	Does it trigger a checkpoint?
COMMIT	❌ No
Manual checkpoint (<code>ALTER SYSTEM CHECKPOINT</code>)	✅ Yes
Log switch (switch to next redo log)	✅ Yes

Instance shutdown (except ABORT)	✔ Yes
Timeout / periodic interval	✔ Yes
Database in ARCHIVELOG mode during heavy redo activity	✔ Yes

Summary

Action	LGWR writes?	DBWn writes?	Checkpoint triggered?
COMMIT	✔ Yes (redo log)	✗ No	✗ No
CHECKPOINT	✔ Maybe (control files)	✔ Yes (data blocks)	✔ Yes
LOG SWITCH	✔ Yes	✔ Yes	✔ Yes