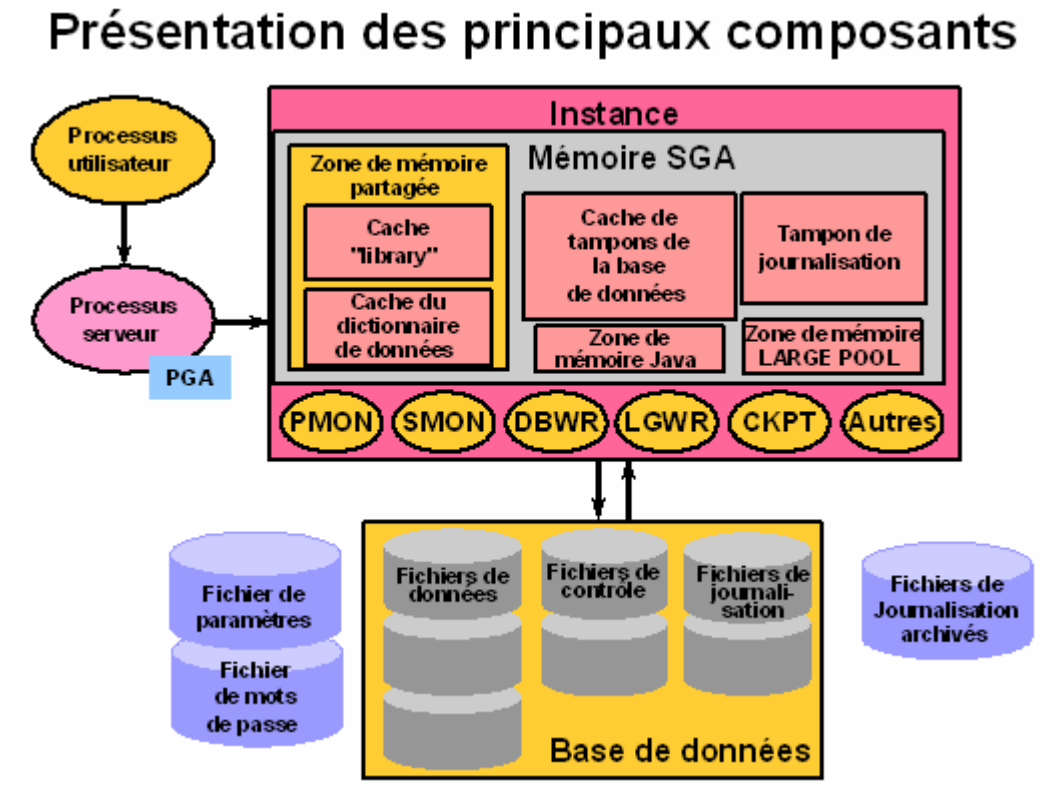
COURS D'ADMINISTRATION DE LA BASE DE DONNEES ORACLE

Niveau I

1. ARCHITECTURE DU SERVEUR ORACLE

Le serveur de bases de données est primordial pour la gestion des informations. En général, il doit gérer de façon fiable, dans un environnement multi utilisateur, une quantité importante de données pour que de nombreux utilisateurs puissent y accéder simultanément, et ce sans affecter les performances.
L'architecture Oracle comporte plusieurs composants principaux, présentés dans le schéma suivant :



➔Définitions & généralitées:

- **Le serveur de bases de données est constitué d'une instance et d'une base de données Oracle**
- **Une instance Oracle est constituée de la mémoire SGA et des processus d'arrière-plan utilisés pour gérer une base de données. Elle ne peut ouvrir et utiliser qu'une seule base à la fois.**
- **Une base de données Oracle possède une structure logique et une structure physique.**
  La structure physique correspond à l'ensemble de fichiers du système d'exploitation constituant la base de données.
  Une base Oracle est composée de trois types de fichier :
  - les fichiers de données, qui contiennent les données de la base,
  - les fichiers de journalisation (fichiers redo log), qui contiennent un enregistrement des modifications apportées à la base afin de permettre la récupération des données en cas de panne,
  - les fichiers de contrôle, qui contiennent les informations nécessaires au maintien et à la vérification de l'intégrité de la base de données.

- **Autres fichiers importants : Les fichiers qui ne sont pas des fichiers de base de données permettent de configurer l'instance, d'authentifier les utilisateurs ayant des privilèges et de récupérer la base en cas de défaillance du disque.**
- **Processus utilisateur et serveur : Les processus utilisateur et serveur sont les principaux processus intervenant dans l'exécution d'une instruction SQL. Néanmoins, d'autres processus peuvent être utilisés par le serveur pour le traitement des instructions SQL.**

2. Instance Oracle :

Une instance Oracle est une combinaison des processus d'arrière-plan et des structures mémoire. Pour accéder aux données de la base, il est nécessaire de démarrer l'instance. A chaque démarrage d'instance, une mémoire SGA (System Global Area) est allouée et des processus d'arrière-plan Oracle sont lancés. Ces processus exécutent des fonctions pour le compte du processus appelant. Ils regroupent des fonctions qui, sinon, seraient gérées par plusieurs programmes Oracle exécutés par chaque utilisateur. Les processus d'arrière-plan effectuent des opérations d'entrée/sortie et surveillent d'autres processus Oracle afin de permettre un plus grand parallélisme et d'améliorer les performances et la fiabilité.

2.1. Les composants de la SGA

La mémoire SGA est également appelée Shared Global Area. Elle stocke les informations de la base qui sont partagées par les processus de base de données.
La mémoire SGA est constituée de plusieurs structures mémoire :
– la zone de mémoire partagée,
– le cache de tampons de la base de données,
– le tampon de journalisation,
– d'autres structures (gestion des verrous externes (lock) et des verrous internes (latch), données statistiques, par exemple).

Deux structures mémoire supplémentaires peuvent également être configurées dans la mémoire SGA :
– la zone de mémoire LARGE POOL,
– la zone de mémoire Java.

2.1.1. Zone de mémoire partagée (Shared pool)

Elle est constituée de deux structures mémoire clés liées **aux performances** :
- **Cache "library"** : permet de stocker et de partager les dernières instructions SQL et leurs plans d'exécution ainsi que les dernières instructions PL/SQL exécutées
- **Cache du dictionnaire de données** : permet de stocker les dernières définitions de données utilisées, telles que les informations sur les fichiers, les tables, les index, les colonnes, les utilisateurs, les privilèges et d'autres objets de la base de données

La mise en mémoire cache des informations des codes SQL/PLSQL et du dictionnaire de données réduit le temps de réponse aux interrogations et aux instructions LMD.

Sa taille est définie par le paramètre SHARED_POOL_SIZE.

2.1.2. Cache de tampons de la base de données (D.B Buffer Cache)

Définitions

- Un bloc de données : est la plus petite unité d'entrée/sortie
- Un bloc OS = 512 Octés en général
- Un bloc Oracle vaut un ou plusieurs blocs OS, la taille du bloc Oracle est définie par le paramètre DB_BLOC_SIZE=2 puissance n
- Une table est constitué d'un ou plusieurs blocs Oracle
- Chaque bloc est affecté à une seule table (Contient un ou plusieurs ligne de la même table)

Lors du traitement d'une interrogation, le processus serveur Oracle recherche les blocs dont il a besoin dans le cache de tampons de la base. Si un bloc nécessaire ne s'y trouve pas, **le processus serveur** lit le bloc dans le fichier de données, puis place une copie de ce bloc dans le cache de tampons. Etant donné que ce bloc pourra par la suite être trouvé dans la mémoire, les demandes suivantes ne nécessitent pas de lecture physique (ce qui permet des gains de performances considérables lors de l'obtention et de la mise à jour de données). Le serveur Oracle utilise un algorithme LRU (Least Recently Used) pour retirer de la mémoire les tampons (buffers) qui n'ont pas été utilisés récemment, afin de libérer de l'espace pour de nouveaux blocs dans le cache de tampons.

2.1.3. Tampon de journalisation

Le tampon de journalisation est un tampon circulaire qui contient les modifications apportées aux blocs de fichiers de données. Ces informations sont stockées dans des entrées de journalisation. Ces entrées contiennent les informations **nécessaires à la recréation (donc, récupération) des données** avant toute modification via les opérations INSERT, UPDATE, DELETE, CREATE, ALTER ou DROP.
La taille du tampon de journalisation est définie par le paramètre d'initialisation LOG_BUFFER.

2.1.4.Zone de mémoire LARGE POOL

C'est une zone facultative de la mémoire SGA, elle permet de réduire la charge de la zone de mémoire partagée.
Il est très souhaitable de la configurée dans les cas suivant :
- la mémoire allouée par session (UGA) au serveur partagé
- les processus serveur d'E/S (configuration des processus esclave du processus DBWn)
- les opérations de sauvegarde et de restauration ou RMAN
- les mémoires tampon des messages d'exécution en parallèle

Elle n'utilise pas de liste LRU.
Sa taille est définie par le paramètre LARGE_POOL_SIZE.

2.1.5.Zone de mémoire Java

La configuration de la zone de mémoire Java est facultative, mais nécessaire si vous installez et utilisez l'option Java dans oracle. La taille de la zone est définie en octets par le paramètre JAVA_POOL_SIZE.

2.2. Processus d'arrière-plan

L'architecture Oracle possède cinq processus d'arrière-plan obligatoires qui seront présentés dans ce chapitre. Oracle possède, en outre, un certain nombre de processus d'arrière-plan facultatifs qui sont démarrés via l'exécution de l'option correspondante. Ces processus ne sont pas décrits dans ce chapitre, à l'exception du processus d'arrière-plan ARCn.
Processus d'arrière-plan obligatoires sont :
- DBWn
- PMON
- CKPT
- LGWR
- SMON

2.2.1.Processus database writer (DBWn)

Le processus serveur enregistre les modifications des blocs d'annulation et des blocs de données dans le cache de tampons de la base de données. Le processus DBWn écrit les tampons "dirty" du cache de tampons de la base de données dans les fichiers de données. Il garantit qu'un nombre suffisant de mémoires tampon libres (tampons qui peuvent être écrasés lorsque les processus serveur doivent lire des blocs dans les fichiers de données) est disponible dans le cache de tampons de la base de données. Les performances de la base sont améliorées puisque les processus serveur n'effectuent les modifications que dans le cache de tampons de la base de données.
Le processus DBWn diffère l'écriture dans les fichiers de données jusqu'à ce que l'un des événements suivants se produise :
- Point de reprise (checkpoint) normal ou incrémentiel.
- Le nombre de tampons "dirty" a atteint une valeur seuil.
- Un processus balaie un certain nombre de blocs à la recherche de mémoires tampon libres et n'en trouve pas.
- Le temps imparti est dépassé.
- Une demande de ping est émise dans l'environnement Real Application Clusters (RAC).
- Un tablespace normal ou temporaire est mis hors ligne.
- Un tablespace est mis en lecture seule.
- Une table est supprimée ou vidée.
- ALTER TABLESPACE nom du tablespace BEGIN BACKUP

2.2.2.Processus LGWR (Log Writer)

Le processus LGWR effectue des opérations d'écriture séquentielles à partir du tampon de journalisation vers le fichier de journalisation dans les cas suivants :
- Lorsqu'une transaction est validée (commit).
- Lorsqu'un tiers du tampon de journalisation est occupé.
- Lorsque le tampon de journalisation contient plus d'un mégaoctet de modifications enregistrées.
- Avant que le processus DBWn n'écrive les blocs modifiés du cache de tampons de la base de données vers les fichiers de données.
- Toutes les trois secondes.

2.2.3.Processus SMON (System Monitor)

Le processus SMON a les responsabilitées suivantes :

✓ **La récuperation d'instance :**

En cas d'échec de l'instance Oracle (ex : copure de courrant ou bloquage d'un composant de l'instance, …), les informations présentes dans la mémoire SGA qui n'ont pas été écrites sur disque sont perdues.

Lorsqu'une instance a échoué, le processus d'arrière-plan SMON la récupère automatiquement lors de la réouverture de la base de données.
La récupération d'une instance implique les 3 étapes suivantes :

1. Réimplémenter les modifications pour récupérer les données non enregistrées dans les fichiers de données, mais enregistrées dans le fichier de journalisation en ligne (online). Ces données n'ont pas été enregistrées sur disque du fait de l'effacement de la mémoire SGA suite à l'échec de l'instance. Lors de cette opération de réimplémentation, le processus SMON lit les fichiers de journalisation et applique aux blocs de données les modifications qui y sont enregistrées. Etant donné que toutes les transactions validées ont été enregistrées dans les fichiers de journalisation, ce processus permet de récupérer ces transactions dans leur intégralité.
2. Ouvrir la base de données pour permettre aux utilisateurs de s'y connecter. Les données qui ne sont pas Verrouillées par des transactions non récupérées sont immédiatement disponibles.
3. Annuler les transactions non validées. Ces transactions sont annulées (rollback) par le processus SMON ou par les processus serveur lorsqu'ils accèdent aux données verrouillées.

- ✓ **Il combine ou fusionne les espaces libres adjacents dans les fichiers de données**.
- ✓ **Il libère les segments temporaires et augmente ainsi l'espace disponible dans les fichiers de données**.

2.2.4.Processus PMON (Process Monitor)

Suite à l'échec de processus, le processus d'arrière-plan PMON exécute des opérations de nettoyage :

- ✓ Il annule la transaction en cours de l'utilisateur
- ✓ Il libère tous les verrous posés sur des tables ou des lignes
- ✓ Il libère d'autres ressources réservées par l'utilisateur
- ✓ Il redémarre les répartiteurs interrompus (En cas du mode Shared Server)

2.2.5.Processus CKPT (Checkpoint)

Ce processus exécute les opérations du point de reprise qui consistent à :
1. Faire appel au processus DBWn pour ecrire le contenu, modifié, du tampon de la base de données sur disque afin d'éviter une perte des données en cas de panne du système ou de la base de données
2. Mettre à jour les en-têtes des fichiers de données du fait qu'un point de reprise a été effectué.
3. Mettre à jour les fichiers CONTROL FILE afin de spécifier que l'action de CHECKPOINT s'est bien déroulée

Le **CHECKPOINT** est **un évènement** qui se déclenche **lors** :

- ✓ D'un changement de groupe de REDO LOG FILE (switch).
- ✓ D'un arrêt normal de la base de données (c'est à dire sans l'option ABORT)
- ✓ D'une demande explicite de l'administrateur par la commande : ALTER SYSTEM CHECKPOINT.
- ✓ D'une limite définie par les paramètres d'initialisation LOG_CHECKPOINT_INTERVAL, LOG_CHECKPOINT_TIMEOUT, et FAST_START_IO_TARGET

Le **CHECKPOINT** est **un évènement** qui se déclenche :

- ✓ Pour garantir que les blocs de données modifiés qui se trouvent en mémoire sont régulièrement écrits sur disque afin d'éviter une perte des données en cas de panne du système ou de la base de données.
- ✓ Pour réduire le temps de récupération d'une instance. Seules les entrées de journalisation postérieures au dernier point de reprise doivent être traitées pour que la récupération ait lieu.
- ✓ Pour garantir que toutes les données validées (et reportées dans les fichiers de journalisation) ont été écrites dans les fichiers de données lors de l'arrêt.

2.2.6.Processus d'archivage ARCn

- ✓ Processus d'arrière-plan facultatif
- ✓ En mode ARCHIVELOG, il archive, sauvegarde, automatiquement les fichiers de journalisation en ligne

2.3. Démarrage d'une instance

Pour lancer le démarrage d'une base de donnée on exécute la commande suivante :

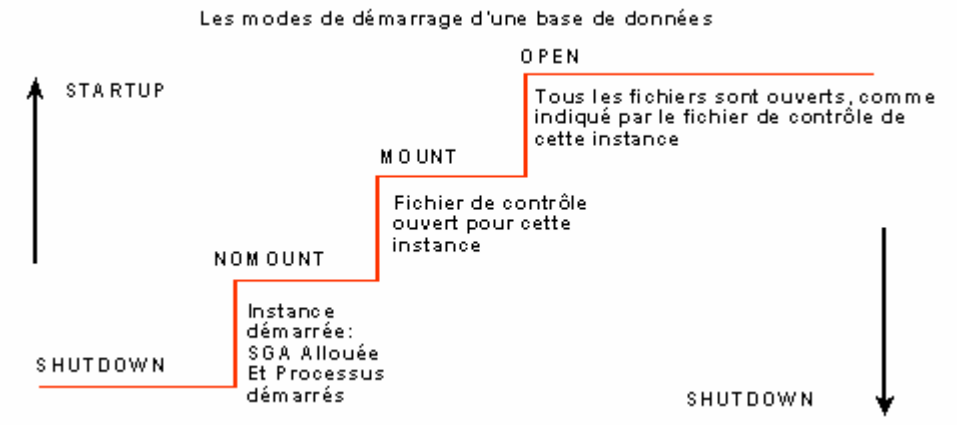
```
Connect sys/password as sysdba   ou   Connect / as sysdba
ou
```

```
Connect sys/password as sysoper  ou   Connect / as sysoper

Startup [NoMount | Mount | open ] [pfile='c:\pfile\init<SID>.ora']
```

Lorsque vous lancez le démarrage de la base de données, il faut sélectionner son statut ou mode de démarrage. Les scénarios suivants décrivent les différentes étapes de démarrage d'une instance.

Les scénarios suivants décrivent les différentes étapes de démarrage d'une instance.



A-
Instance démarrée (NOMOUNT)
Le démarrage d'une instance en mode NOMOUNT ne s'effectue que pour la création d'une base de données ou à la recréation de fichiers de contrôle.
Le démarrage d'une instance comprend les tâches suivantes :
• La lecture du fichier de paramètres d'initialisation dans le répertoire %ORACLE_HOME%\database pour Windows ou dans $ORACLE_HOME/dbs en cas d'unix
• L'affectation de la mémoire SGA.
• Le démarrage des processus d'arrière-plan.
• L'ouverture du fichier alert<SID>.log et des fichiers trace. Où SID=le nom de l'instance.

B- Base de données montée (MOUNT) :
Pour effectuer des opérations de maintenance, vous démarrez une instance et montez une base de données sans l'ouvrir pour :
• renommer des fichiers de données,
• activer ou désactiver des options d'archivage de fichiers de journalisation,
• effectuer une récupération complète de la base de données.

Le montage d'une base de données comprend les tâches suivantes :
• Association d'une base de données à une instance démarrée
• Localisation et ouverture des fichiers de contrôle indiqués dans le fichier de paramètres
• Lecture des fichiers de contrôle pour extraire le nom et le statut des fichiers de données et des fichiers de journalisation. Toutefois, l'existence des fichiers de données et des fichiers de journalisation en ligne n'est pas vérifiée à ce stade.

C- Ouverture de la base de données (OPEN) :
En mode de fonctionnement normal, vous démarrez une instance avant de monter et d'ouvrir la base de données. Ainsi, les utilisateurs autorisés peuvent se connecter à la base et effectuer des opérations standard sur les données.
L'ouverture de la base de données comprend les tâches suivantes :

•Ouverture des fichiers de données en ligne
•Ouverture des fichiers de journalisation en ligne

Si un fichier de données ou de journalisation en ligne est absent lorsque vous tentez d'ouvrir la base de données, le serveur Oracle renvoie une erreur.
Au cours de cette dernière étape, le serveur Oracle vérifie que tous les fichiers de données et de journalisation en ligne peuvent être ouverts et contrôle la cohérence de la base de données. Si nécessaire, le processus d'arrière-plan SMON (System Monitor) déclenche la récupération de l'instance.

Changement de statut/mode de démarrage :
Si la base est déjà démarré dans les modes Nomount ou Mount on peut la passer dans le mode suivant par la commande :
ALTER DATABASE MOUNT ;
ou
ALTER DATABASE OPEN;

Obtenir des information sur le statut actuel de la BD.
Sql> SELECT status FROM V$instance

2.4. Le fichier de paramètres:

Le fichier de paramètre est un fichier qui contient tous les paramètres explicites (dont la valeurs est définie par l'administrateur et non pas affectée par défaut).
A partir de la version Oracle 9i , il existe deux types de fichiers de paramètres :

- ✓ **Le fichier PFILE :**
  Est un fichier texte que vous pouvez mettre à jour à l'aide d'un éditeur standard du système d'exploitation. Ce fichier est en lecture seule pendant le démarrage de l'instance. S'il est modifié, l'instance doit être interrompue et redémarrée pour que les nouvelles valeurs des paramètres soient effectives. Par défaut, le fichier PFILE se trouve dans le répertoire :
  $ORACLE_HOME/dbs sous unix ou **%ORACLE_HOME%\database**
  son nom par defaut est **init<SID>.ora où <SID> est le nom de l'instance**

- ✓

- ✓ **Le fichier SPFILE :**
  Oracle9i propose un nouveau **fichier binaire** nommé SPFILE. Ce fichier ne doit pas être modifié manuellement et doit toujours résider côté serveur. Une fois créé, le fichier est mis à jour par le serveur Oracle. S'il est modifié manuellement, il devient inutile. Il permet d'apporter à la base de données des modifications qui seront conservées après l'arrêt et le redémarrage.
  Par défaut, le fichier SPFILE se trouve dans le répertoire :
  $ORACLE_HOME/dbs sous unix ou **%ORACLE_HOME%\database**
  son nom par defaut est spfile**<SID>.ora où <SID> est le nom de l'instance**

  Pour créer le fichier spfile on execute la commande suivante :
  Sql> Create spfile from pfile ;

2.5. Gestion des paramètres

Exemple de paramètres:

```
db_name              = dba01
instance_name        = dba01
control_files        = (home/dba01/ORADATA/u01/control01dba01.ctl,
       home/dba01/ORADATA/u02/control01dba02.ctl)
db_block_size        = 4096
db_cache_size        = 4M
shared_pool_size     = 50000000
java_pool_size       = 50000000
max_dump_file_size   = 10240
background_dump_dest = /home/dba01/ADMIN/BDUMP
undo_management      = AUTO
```

```
undo_tablespace      = UNDOTBS
. . .
```

2.5.1.Types de paramètres :

Il existe Deux types de paramètres :

- ✓ **Paramètre dynamiques** : On peut les modifier sans redémarrer la B.D avec l'instruction suivante :
  ALTER SYSTEM SET nom_parmetre= nouvel_valeur ;

- ✓ **Paramètre statiques :** Ce type de paramètre ne peut être modifier effectivement que si on redémarre la B.D. La manière pour changer la valeur d'un paramètre statique dépend selon est ce qu'on utilise le fichier PFILE ou SPFILE :
  - Cas PFILE :
    1- Arrêter la BD (shutdown immediate)
    2- Editer le fichier init<SID> .ora et changer le paramètre souhaité
    3- Redémarrer la BD (startup)
  - Cas SPFILE
    1- Alter system set nom_param=valeur scope=SPFILE ;
    2- Shutdown immediate
    3- startup

Consulter la valeur d'un paramètre :

```
Sql> Select name,value from v$parameter where name='nom_parametre' ;
```

Ou

```
Sql> show parameter nom_parametre
```

2.6. Arrêter une instance

Pour arrêter une instance, connectez-vous en tant que SYSOPER ou SYSDBA et utilisez la commande suivante :

```
SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT ]
```

Options d'arrêt :

Arrêt en mode Normal :

Le mode Normal est le mode d'arrêt par défaut. Il s'effectue dans les conditions suivantes :

- Aucune nouvelle connexion ne peut être établie.
- Le serveur Oracle attend la déconnexion préalable de tous les utilisateurs.
- Les tampons de journalisation et de la base de données sont écrits sur disque.
- Les processus d'arrière-plan prennent fin et la zone SGA est supprimée de la mémoire.
- Oracle ferme et démonte la base de données avant d'arrêter l'instance.
- La récupération de l'instance n'est pas nécessaire lors du redémarrage.

Arrêt en mode Transactional

L'arrêt en mode Transactional évite aux clients de perdre leurs travaux en cours. Il s'effectue dans les conditions suivantes :

- Aucun client ne peut lancer de nouvelle transaction pour l'instance indiquée.
- Le client est déconnecté lorsqu'il termine la transaction en cours.
- La fin de toutes les transactions entraîne l'arrêt immédiat de la base de données.
- La récupération de l'instance n'est pas nécessaire lors du redémarrage.

Arrêt en mode Immediate

L'arrêt en mode Immediate s'effectue dans les conditions suivantes :

- Les instructions SQL en cours de traitement par Oracle ne sont pas terminées.
- Le serveur Oracle n'attend pas la déconnexion des utilisateurs de la base de données.
- Oracle annule les transactions actives et déconnecte tous les utilisateurs.
- Oracle ferme et démonte la base de données avant d'arrêter l'instance.

- La récupération de l'instance n'est pas nécessaire lors du redémarrage.

Arrêt en mode Abort

Si les arrêts en modes Normal et Immediate échouent, vous pouvez abandonner l'instance de base de données en cours. Cette opération s'effectue dans les conditions suivantes :

- Les instructions SQL en cours de traitement par le serveur Oracle sont immédiatement interrompues.
- Oracle n'attend pas la déconnexion des utilisateurs de la base de données.
- Les tampons de journalisation et de la base de données ne sont pas écrits sur disque.
- Les transactions non validées ne sont pas annulées.
- L'instance est interrompue sans fermeture des fichiers.
- La base de données n'est pas fermée, ni démontée.
- Une récupération est nécessaire au redémarrage ; elle s'effectue automatiquement par SMON.

3. La structure memoire Oracle

La structure mémoire d'Oracle est constituée des deux zones de mémoire suivantes :

1. **la mémoire SGA (System Global Area)**, qui est allouée au démarrage de l'instance et qui est une composante fondamentale d'une instance Oracle
2. **La mémoire PGA (Program Global Area),** qui est allouée au démarrage de chaque processus serveur ou d'arrière plan :

La PGA contient les information sur la session utilisateur, la zone de trie, état des curseurs et l'espace de pile :

- ✓ Le curseur contient des valeurs de mémoire d'exécution pour l'instruction SQL, par exemple les lignes extraites.
- ✓ Les données des sessions utilisateur comprennent des informations sur la sécurité et l'utilisation des ressources.
- ✓ L'espace de pile contient des variables locales pour le processus.

4. La base de données

La base de donnée oracle est constituée de l'ensemble de fichiers oracle, il existe trois types de fichiers

- Fichiers de contrôle
- Fichiers de journalisation en ligne (ou fichiers Redo log)
- Fichiers de données

Nous allons décrire la fonctionalité de chaque type de fichier dans les chapitres à venir……

4.1. Création d'une BD :

Etapes pour créer Une Base de données manuellement

- ✓ Préparer les disques et les répertoires pour les fichiers de la BD
- ✓ Préparer votre fichier de paramètre « init<SID>.ora »
  où <SID>=nom d'instance
- ✓ Positionner les variables d'environnement suivantes:
  `ORACLE_HOME=<répertoire où oracle est installé>`
  `ORACLE_SID=<nom de l'instance à créer>`
- ✓ Dans le cas de Windows (uniquement):
  Il faut créer le service Windows : « OracleService<ORACLE_SID> »
  Pour cela utiliser l'utilitaire « oradim » fourni par oracle :
  Ex: `c:\>oradim -NEW -SID TEST`
  (où TEST est le nom de l'instance à créer)
- ✓ Ouvrez un terminal OS (DOS ou UNIX), puis connectez vous et démarrez l'instance en mode « NOMOUNT » comme suit:

```
sqlplus /nolog
sql>connect / as sysdba
sql>startup nomount
```

- ✓ Executer la commande de creation de la BD :

```
sql> create database  ... . . .
```

- ✓ Ouvrez la BD : `sql>Alter database open;`
- ✓ Créer les vues du dictionnaire de données en exécutant les scripts :

```
sql>@?\rdbms\admin\catalog.sql
sql>@?\rdbms\admin\catproc.sql
```

5. Le Fichier de contrôle

C'est un petit fichier binaire qui
définit l'état actuel de la base de données physique.
Il permet d'assure l'intégrité de la base de données.
Il est requis :

–lors de l'étape `MOUNT` lors du démarrage de BD, s'il est perdu BD ne dépasse pas le mode nomount

–pour le fonctionnement de la base de données.

5.1. Contenu du fichier de contrôle :

Le fichier de contrôle contient les informations suivantes :
•Le nom de la base de données correspond au nom indiqué par le paramètre d'initialisation DB_NAME ou à celui utilisé dans l'instruction CREATE DATABASE.
•L'identificateur de la base de données est enregistré à la création de cette dernière.
•L'horodatage de création de la base de données est également enregistré à la création de cette dernière.
•Le nom et l'emplacement des fichiers de données et des fichiers de journalisation en ligne (online redo logs) associés sont mis à jour lors de l'ajout, du changement de nom ou de la suppression de l'un de ces fichiers dans la base de données.
•Les informations relatives aux tablespaces sont mises à jour lors de l'ajout ou de la suppression de tablespaces.
•L'historique de journalisation est enregistré lors des changements de fichier de journalisation.
•L'emplacement et le statut des journaux archivés sont enregistrés lors de l'archivage.
•L'emplacement et le statut des sauvegardes sont enregistrés par l'utilitaire Recovery Manager.
•Le numéro de séquence du journal en cours est enregistré lors des changements de fichier de journalisation.
•Les informations relatives aux points de reprise (checkpoints) sont enregistrées lorsque les points de reprise ont lieu.

5.2. Obtenir des information sur le fichier de contrôle

Pour obtenir des information sur les fichiers de contrôle de la BD on exécute l'une des commandes suivantes :

Select name from v$controlfile ou Show parameter control_files
ou
Select name,value from v$parameter where name='control_files'

5.3. Multiplexer le fichier de contrôle :

Vu son importance un fichier de control doit être multiplexer (plusieurs copies en miroirs sur des disks différents). Le multiplexage n'est pas qu'une simple copie du fichier mais c'est le fait que cette copie doit être maintenue par le serveur Oracle.
La procédure du multiplexage dépend du fait qu'on a démarré la B.D avec le fichier SPFILE ou PFILE :

5.3.1.Multiplexer le fichier de contrôle lorsqu'un fichier SPFILE est utilisé :

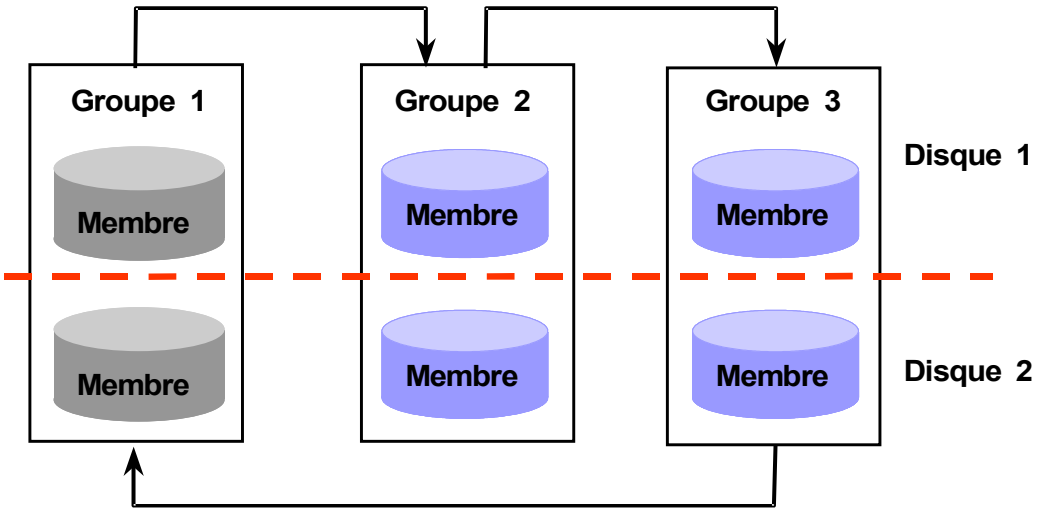
1.Modifiez le `SPFILE` : à l'aide de la commande `ALTER SYSTEM SET`, modifiez le fichier `SPFILE` afin d'y inclure la liste des fichiers de contrôle à utiliser (fichier de contrôle principal et copies multiplexées).
2.Arrêtez la base de données : arrêtez la base afin de créer les fichiers de contrôle supplémentaires sur le système d'exploitation.
3.Créez les fichiers de contrôle supplémentaires : à l'aide de la commande de copie du système d'exploitation, créez autant de fichiers de contrôle supplémentaires que nécessaire dans les répertoires appropriés.
4.Démarrez la base de données : au démarrage de la base, le fichier `SPFILE` est lu et le serveur Oracle met à jour tous les fichiers de contrôle répertoriés dans le paramètre `CONTROL_FILES`.

5.3.2.Multiplexer le fichier de contrôle lorsqu'un fichier `PFILE` est utilisé:

1.Arrêtez la base de données : arrêtez la base afin de créer les fichiers de contrôle supplémentaires sur le système d'exploitation.
2.Créez les fichiers de contrôle supplémentaires : à l'aide de la commande de copie du système d'exploitation, créez autant de fichiers de contrôle supplémentaires que nécessaire dans les répertoires appropriés.
3.Ajoutez les noms des fichiers de contrôle au fichier `PFILE` : modifiez le parametre `CONTROL_FILES` dans le fichier `PFILE` afin d'y inclure la liste des fichiers de contrôle.
4.Démarrez la base de données : au démarrage de la base, le fichier `PFILE` est lu et le serveur Oracle met à jour tous les fichiers de contrôle répertoriés dans le paramètre `CONTROL_FILES`.

6. Les fichiers de journalisation

Le serveur Oracle enregistre de manière séquentielle toutes les modifications apportées à la base de données dans le tampon de journalisation (appelé les entrées de journalisation).



Les fichiers de journalisation ont les caractéristiques suivantes :

- Organisés en groupe, On doit avoir **DEUX groupes au minimum dans une BD**
- Un group est constitué d'un ou plusieurs copies identiques de fichiers de journalisation en ligne appelés les membres du groupe.
- Le processus LGWR ecrit dans un seul groupe à la fois, quand le groupe est saturé LGWR passe à un autre groupe
- Les fichiers de journalisations sont utilisés de façon cyclique : (On revient toujours vers un groupe pour l'écraser et le réutiliser)
- Chaque groupe est identifié par un numéro de séquence remplacé à chaque nouvelle utilisation du journal
- Un groupe RedoLog doit un status à instant donné, les status des groupes sont :
  → **UNUSED** : les fichiers Redolog n'ont jamais été utilisé (ie : On vient de les créer)
  → **CURRENT** : Le LGWR ecrit actuellement dans ce groupe. Un seul groupe est CURRENT à la fois
  → **ACTIF** : Le group est encore necessaire à la récupération. (ie : on ne peut ni l'ecraser ni le supprimer)
  → **INACTIF** : n'est pas actif

6.1. Changements de fichier de journalisation

Le processus LGWR écrit de façon séquentielle dans les fichiers de journalisation en ligne. Lorsque le groupe de fichiers de journalisation en ligne en cours est complet, le processus LGWR passe au groupe suivant. On parle alors de changement de fichier de journalisation ou le « SWITCH »

Le « SWITCH » invoque automatiquement deux événements :
« SWITCH »
➔ **(1) Checkpoint et (2) L'archivage du groupe si la BD est en mode archivelog**

On peut imposer le « switch » , ou le changement du fichier de journalisation par la commande :

```
ALTER SYSTEM SWITCH LOGFILE;
```

6.2. Obtenir des information sur les fichiers de journalisation

- Répertoriez l'emplacement des fichiers journaux existants par group :
  SELECT group#,member group by group#,member from v$logfile

- affichez les groupes de journalisation , le nombre de membres qu'il contienent et leurs status :
  SELECT group#, members, status FROM v$log

6.3. Ajouter des groupes et des membre de fichiers de journalisation :

Vous pouvez ajouter des groupes pour résoudre les problèmes de disponibilité. Utilisez la commande SQL suivante pour créer un groupe de fichiers de journalisation en ligne :

ALTER DATABASE ADD LOGFILE
GROUP numero ('nom_fichier_member_1', …, ' nom_fichier_member_n') size n[K|M]

Exemple :

En respectant les règles de nommage ajouter le group 3 et 4 à votre Base de données :

```
ALTER DATABASE ADD LOGFILE
GROUP 3 ('c:\oracle\ORADATA\MASTER\log03a.rdo',
         'c:\oracle\ORADATA\MASTER\log03b.rdo') SIZE 1024K,
GROUP 4 ('c:\oracle\ORADATA\MASTER\log04a.rdo',
         'c:\oracle\ORADATA\MASTER\log04b.rdo') SIZE 1024K;
```

→ Pour rajouter des member aux groupes déjà existant on utilisera la commande suivante :

```
ALTER DATABASE ADD LOGFILE MEMBER
  'c:\oracle\ORADATA\MASTER\log01b.rdo' to Group 1,
  'c:\oracle\ORADATA\MASTER\log02b.rdo' to Group 2;
```

6.4. Supprimer des groupes et des membres de fichiers de journalisation

La commande suivante permet de supprimer le groupe « n » redo log

```
ALTER DATABASE DROP LOGFILE GROUP n;
```

La commande suivante permet de supprimer un membre du groupe « n » redo log

```
ALTER DATABASE DROP LOGFILE MEMBER 'c:\…\log3c.rdo';
```

7. Tablespaces et fichiers de données

7.1. Définitions

Les bases de données, les tablespaces et les fichiers de données sont très proches, mais ils présentent d'importantes différences.

- Une base de données Oracle est composée d'une ou de plusieurs unités de stockage logiques appelées tablespaces, qui de manière collective stockent toutes les données de la base de données.
- Chaque tablespace d'une base de données Oracle contient un ou plusieurs fichiers appelés fichiers de données. Ces fichiers sont des structures physiques conformes au système d'exploitation sur lequel s'exécute le serveur Oracle.
- Un utilisateur oracle crée des tables dans un tablespace et le serveur Oracle attribue à cette table un espace dans un ou plusieurs fichiers associé à ce tablespace.

7.2. Types de tablespace :

L'administrateur de base de données crée des tablespaces pour améliorer le contrôle et faciliter la gestion de la base. Le serveur Oracle accepte deux types de tablespace : le tablespace SYSTEM et tous les autres tablespaces.

7.2.1. Les types de tablespace :

Il existe TROIS catégories de tablepsace qui dépendent du type du contenu des ces tablepsace :

- **PERMANENT :** Les tablespaces ***permanent*** permettent de stocker des données à caractère permanent, c.à.d : une fois créé ils ne sont supprimés que par une commande SQL exécuté par l'utilisateur même après le redémarrage de la BD.
- **TEMPORARY :** Les fichiers des tablespaces ***temporaire*** sont utilisés uniquement pour les opérations de tries. Si un utilisateur exécute une instruction SQL qui nécessite un trie, ce dernier s'effectue d'abord dans la mémoire PGA du processus serveur associé à l'utilisateur, et si l'espace mémoire réservé aux tries n'est pas suffisent le processus serveur ALLOUE un ESPACE TEMPORAIRE dans les fichiers du tablespace temporaire. Dès que 'lopération de trie est terminé l'espace physique alloué dans les fichiers est libéré. Une BD oracle peut avoir plusieurs tablespace temporaire
- **UNDO :** Le tablespace ***undo*** appelé encore tablespace d'annulation ou de rollback c'est le tablespace ou oracle stocke les valeurs des modifications non encore validées (commit) pourvu de les restituées si jamais l'utilisateur fait une annulation (rollback) ou bien quitte sa session d'une manière anormal (ex : coupure du courant). Une BD oracle utilise toujours un seul tablespace ***undo*** à la fois.

7.3. Créer des tablespaces

La commande CREATE TABLESPACE permet de créer un tablespace :

CREATE [PERMANENT |TEMPORARY|UNDO] TABLESPACE nom_*tablespace*
DATAFILE 'chemin+nom fichier' SIZE integer[K|M] Autoextend [ON|OFF] MAXSIZE integer[K|M]
[EXTENT MANAGEMENT [DICTIONARY|LOLCAL]]
[SEGMENT MANAGEMENT AUTO]
[MINIMUM EXTENT integer[K|M]]
[LOGGING|NOLOGGING]
[DEFAULT storage_clause]
[ONLINE|OFFLINE]

Où :
DATAFILE définit les fichiers de données constituant le tablespace.
MINIMUM EXTENT garantit que la taille de chaque extent (ensemble de blocs contigus) du tablespace est un multiple de integer. Utilisez K ou M pour définir la taille en kilo-octets ou en mégaoctets.
LOGGING indique que toutes les modifications effectuées sur les tables, index et partitions du tablespace seront écrites par défaut dans le fichier de journalisation (LOGGING est la valeur par défaut).
NOLOGGING indique que toutes les modifications effectuées dans les tables, index et partitions du tablespace ne seront pas écrites par défaut dans le fichier de journalisation (NOLOGGING n'affecte que certaines commandes LMD et LDD, telles que les chargements des données par chemin direct).
DEFAULT définit les paramètres de stockage par défaut de tous les objets créés dans le tablespace.
OFFLINE rend le tablespace indisponible dès sa création.
PERMANENT indique que le tablespace peut être utilisé pour stocker des objets permanents. C'est la valeur par défaut
TEMPORARY indique que le tablespace ne contiendra que des objets temporaires, tels que les segments de tri implicite créés par la clause ORDER BY. La définition des clauses EXTENT MANAGEMENT LOCAL est impossible.
extent_management_clause indique la manière dont les extents du tablespace sont gérés (cette clause est décrite dans l'une des sections suivantes du présent chapitre).
segment_management_clause ne s'applique qu'aux tablespaces permanents, gérés localement. Elle vous permet d'indiquer si Oracle doit procéder au suivi de l'espace libre et utilisé dans les segments du tablespace à l'aide des bitmaps ou des listes de blocs libres (free list).
datafile_clause:== filename [SIZE integer[K|M] [REUSE]
[autoextend_clause]
filename correspond au nom d'un fichier de données du tablespace.
SIZE définit la taille du fichier (utilisez K ou M pour définir la taille en kilo-octets ou en mégaoctets).
REUSE permet au serveur Oracle de réutiliser un fichier existant.
autoextend_clause permet d'activer ou de désactiver l'extension automatique du fichier de données.
NEXT indique la taille en octets de la prochaine incrémentation de l'espace disque devant être allouée automatiquement lorsque plusieurs extents sont requis.
MAXSIZE indique l'espace disque maximum autorisé pour l'extension automatique d'un fichier de données.
UNLIMITED indique l'espace disque pouvant être affecté au fichier de données ou indique que le fichier temporaire n'est pas limité.

Exemples :

- Créer un tablespace DATA de type PERMANENT :

```
CREATE TABLESPACE data
 DATAFILE 'c:\…\data01.dbf' size 10M AUTOEXTEND ON NEXT 1M MAXSIZE 100M;
```

- Créer un tablespace TEMPFIN de type TEMPORAIRE:

```
CREATE TEMPORARY TABLESPACE TEMPFIN
 TEMPFILE 'C:\ORACLE\ORADATA\DBA\FINANCE.ora' SIZE 5M
 EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M;
```

- Créer un tablespace UNDO2 de type UNDO:

```
CREATE UNDO TABLESPACE UNDO2
 DATAFILE 'C:\ORACLE\ORADATA\DBA\FINANCE.ora' SIZE 5M UNIFORM SIZE 100M;
```

7.4. Gestion de l'espace dans les tablespaces

Les tablespaces affectent de l'espace dans les extents. Lors de leur création, vous pouvez choisir l'une ou l'autre des méthodes de gestion de l'espace libre et utilisé suivantes :

7.4.1.Tablespaces gérés localement :

Les extents sont gérés dans le tablespace via des bitmaps. Chaque bit du bitmap correspond un bloc ou à un groupe de blocs. Lorsqu'un extent est alloué ou libéré pour être réutilisé, le serveur Oracle modifie les valeurs bitmap pour indiquer le nouveau statut des blocs. Cette méthode est utilisée par défaut dans Oracle*9i*.

7.4.2.Tablespaces gérés au moyen du dictionnaire :

Les extents sont gérés à l'aide du dictionnaire de données. Le serveur Oracle met à jour les tables appropriées dans le dictionnaire de données chaque fois qu'un extent est alloué ou libéré.

7.5. Tablespaces TEMPORARY

Vous pouvez gérer plus efficacement l'espace pour les opérations de tri en définissant des tablespaces TEMPORARY réservés exclusivement aux segments de tri. Aucun objet de schéma permanent ne peut résider dans un tablespace TEMPORARY.
Un segment de tri ou un segment temporaire est utilisé lorsqu'un segment est partagé par plusieurs opérations de tri. Les tablespaces TEMPORARY améliorent les performances lorsque plusieurs tris ne peuvent tenir dans la mémoire. Le segment de tri d'un tablespace TEMPORARY donné est créé lors de la première opération de tri dans l'instance. La taille du segment de tri augmente par allocation d'extents jusqu'à ce qu'elle soit égale ou supérieure au nombre total des demandes de stockage de tous les tris en cours exécutés dans l'instance.

7.6. Tablespaces accessibles en lecture seule

La commande *ALTER TABLESPACE nom_tablespace READ ONLY* place le tablespace en mode lecture seule de transition. Ce mode n'autorise pas les opérations d'écriture dans le tablespace, excepté pour l'annulation de transactions existantes qui ont préalablement modifié des blocs du tablespace. Une fois toutes les transactions existantes validées ou annulées, la commande en lecture seule s'exécute et place le tablespace en lecture seule. Vous pouvez supprimer des éléments, tels que des tables et des index, d'un tablespace en lecture seule, car ces commandes n'affectent que le dictionnaire de données. Cette opération est possible car la commande DROP met à jour le dictionnaire de données et non les fichiers physiques qui constituent le tablespace. Pour les tablespaces gérés localement, le segment supprimé est converti en segment temporaire pour éviter la mise à jour du bitmap. Pour rendre accessible en écriture un tablespace qui est en lecture seule, tous ses fichiers de données doivent être en ligne (online). La mise en lecture seule d'un tablespace crée un point de reprise dans les fichiers de données du tablespace.

Pour remettre le tablespace en mode lecture ecriture :
ALTER TABLESPACE nom_tablespace READ WRITE

7.7. Mettre un tablespace hors ligne

Un tablespace est généralement en ligne, ce qui permet aux utilisateurs de la base d'accéder aux données qu'il contient. Toutefois, l'administrateur de base de données peut le mettre hors ligne pour :

- rendre une partie de la base de données indisponible tout en permettant l'accès normal au reste de la base,
- sauvegarder un tablespace hors ligne (bien qu'il soit possible de sauvegarder un tablespace pendant qu'il est en ligne et en cours d'utilisation),
- restaurer un tablespace ou un fichier de données lorsque la base de données est ouverte,
- déplacer un fichier de données lorsque la base est ouverte.

Statut hors ligne d'un tablespace

Lorsque vous mettez un tablespace hors ligne, le serveur Oracle ne permet pas aux instructions SQL qui suivent de faire référence aux objets contenus dans le tablespace. Dans ce cas, les utilisateurs qui tentent d'accéder à ces objets reçoivent un message d'erreur.
La mise hors ligne ou en ligne des tablespaces est enregistrée dans le dictionnaire de données et le fichier de contrôle. Si un tablespace est hors ligne lorsque vous arrêtez une base de données, il reste hors ligne et n'est pas vérifié au remontage et à la réouverture de la base de données.
ALTER TABLESPACE tablespace{ONLINE |OFFLINE}

7.8. Redimensionner un tablespace

Vous pouvez augmenter la taille d'un tablespace de deux manières :

- en modifiant automatiquement ou manuellement la taille d'un fichier de données,
- en ajoutant un fichier de données au tablespace.

7.8.1. La clause AUTOEXTEND

La clause AUTOEXTEND permet d'activer ou de désactiver l'extension automatique des fichiers de données. La taille des fichiers est incrémentée selon vos indications mais est limitée par une valeur maximale.
Avantages de l'utilisation de la clause AUTOEXTEND :

- Le besoin d'intervention immédiate est réduit lorsque l'espace disponible est insuffisant dans le tablespace.
- Les applications ne s'arrêtent plus à cause des pannes d'allocation des extents.

Lors de la création d'un fichier de données, utilisez les commandes SQL suivantes pour activer l'augmentation automatique de la taille du fichier de données :

- CREATE TABLESPACE nom_ts DATAFILE 'c:\…\ts01.dbf' size 10M autoextend on next 1M maxsize 20M;
- ALTER TABLESPACE nom_ts ADD DATAFILE 'c:\…\ts02.dbf' size 5M autoextend on next 1M maxsize 20M;

Déterminer si AUTOEXTEND est activé ou non
Interrogez la vue DBA_DATA_FILES pour déterminer si AUTOEXTEND est activé et examinez la colonne AUTOEXTENSIBLE.
SQL> select tablespace_name, file_name, autoextensible from dba_data_files;

7.9. Redimensionner manuellement un fichier de données

Pour augmenter l'espace de la base de données, le DBA peut redimensionner un fichier de données au lieu d'ajouter des fichiers de données. La commande ALTER DATABASE permet d'augmenter ou de diminuer manuellement la taille d'un fichier de données.
ALTER DATABASE DATAFILE 'filename' RESIZE integer[K|M]
Où :
Integer correspond à la taille absolue, en octets, du fichier de données final.

7.10. Supprimer un tablespace :

DROP TABLESPACE nom_ts ICLUDING CONTENT AND DATAFILE ;

7.11. Méthodes de déplacement des fichiers de données

7.11.1. Commande ALTER DATABASE

La commande ALTER DATABASE permet de déplacer tous les types de fichier de données.
ALTER DATABASE [database]
RENAME FILE 'filename'[, 'filename']...
TO 'filename'[, 'filename']...
Dans la mesure où le tablespace SYSTEM ne peut pas être mis hors ligne, vous devez utiliser cette méthode pour déplacer ses fichiers de données.
Pour renommer les fichiers des tablespaces qui ne peuvent pas être mis hors ligne, procédez comme suit :
1.Arrêtez la base de données.

2.Utilisez la commande de système d'exploitation appropriée pour déplacer les fichiers.
3.Montez la base de données.
4.Exécutez la commande ALTER DATABASE RENAME FILE.
5.Ouvrez la base de données.

7.11.2. Commande ALTER TABLESPACE

1-Mettez le tablespace hors ligne.
2-Utilisez la commande appropriée du système d'exploitation pour déplacer ou copier les fichiers.
3-Exécutez la commande ALTER TABLESPACE RENAME DATAFILE.
4-Mettez le tablespace en ligne.
5-Au besoin, utilisez la commande appropriée du système d'exploitation pour supprimer le fichier.

7.12. Obtenir des information sur les tablespaces et les fichiers de données

- vue v$datafile : fourni la liste des fichiers de données de la BD

```
SELECT name FROM V$DATAFILE ;
```

- vue DBA_DATA_FILES :

Obtenir la liste des tablespaces de la BD ainsi que leurs fichiers correspondants

```
SELECT  tablespace_name , file_name from dba_data_files
```

Obtenir le chemin et le nom de(s) fichier(s) de données correspondant à un tablespace spécifique :

```
SELECT  file_name from dba_data_files where tablespace_name='nom_TS'
```

Où nom_TS est le nom du tablespace spécifique
Ex : obtenir le nom et le chemin du fichier de donnée du tablespace SYSTEM

```
SELECT  file_name from dba_data_files where tablespace_name='SYSTEM'
```

8. GESTION DES UTILISATEURS et DE PRIVILEGES

8.1. Gestion des utilisateurs

Un utilisateur de base de données va correspondre à un login qui aura reçu certains privilèges (droits)
Un schéma est une collection (ou un ensemble) nommé d'objets tels que des tables, vues, clusters, procédure et packages associés à un utilisateur précis.

→ ***Pour créer un utilisateur en utilise la syntaxe suivante*** :

```
CREATE USER nom_user IDENTIFIED BY mot_pass
DEFAULT TABLESPACE  nom_ts
TEMPORARY TABLESPACE nom_ts_temp ;
```

Où :
nom_user : est le nom de l'utilisateur
nom_ts : Nom du tablespace par défaut qui va contenir les objets (tables, indexes, …) créés par l'utilisateur si la commande de création ne précise pas le tablespace de destination
nom_ts_temp : nom du tablespace temporaire où seront déroulés les tries effectués par l'utilistaeur

→ **Pour *modifier les caracteristiques d'un utilisateur***

```
ALTER USER nom_user [IDENTIFIED BY mot_pass]
[DEFAULT TABLESPACE  nom_ts]
[TEMPORARY TABLESPACE nom_ts_temp] ;
```

Ex : Affecter le tablespace de données USERS à SCOTT

```
ALTER USER scott DEFAULT TABLESPACE users
```

→ ***Pour supprimer un utilisateur en utilise la syntaxe suivante***

```
DROP USER nom_user CASCADE;
```

8.2. Obtenir des information sur les utilisateurs

L'instruction suivantes permet d'afficher le nom des utilisateurs de la BD, leurs tablespace de données et temporaire par défaut :

```
Sql> SELECT USERNMAE,DEFAUT_TABLESAPCE,TEMPORARY_TABLESPACE FROM
DBA_USERS
```

8.3. Privilèges :

Un privilège est un droit d'exécution d'un type donné d'instruction SQL ou un droit d'accès à l'objet d'un autre utilisateur. Il autorise son détenteur à :

- se connecter à une base de données,
- créer une table,
- sélectionner ou mettre à jour des lignes dans la table d'un autre utilisateur,
- exécuter la procédure stockée d'un autre utilisateur.Privilèges système

8.3.1.privilège système

Chaque privilège système permet à un utilisateur d'effectuer une opération spécifique ou une catégorie d'opérations sur la base de données.
Par exemple, le privilège lié à la création de tablespaces est un privilège système.
Ces privilèges peuvent être classés comme suit :

- Privilèges autorisant l'exécution d'opérations sur l'ensemble du système, tels que CREATE SESSION, CREATE TABLESPACE
- Privilèges autorisant la gestion d'objets dans un schéma propre à l'utilisateur, tels que CREATE TABLE Privilèges autorisant la gestion d'objets de n'importe quel schéma, tels que CREATE ANY TABLE

Privilèges système : exemples

Catégorie	Exemples
INDEX	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX
TABLE	CREATE TABLE CREATE ANY TABLE ALTER ANY TABLE DROP ANY TABLE SELECT ANY TABLE UPDATE ANY TABLE DELETE ANY TABLE
SESSION	CREATE SESSION ALTER SESSION RESTRICTED SESSION
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE UNLIMITED TABLESPACE

Catégorie	Exemples
SYSOPER	STARTUP SHUTDOWN ALTER DATABASE OPEN | MOUNT ALTER DATABASE BACKUP CONTROLFILE TO RECOVER DATABASE ALTER DATABASE ARCHIVELOG RESTRICTED SESSION
SYSDBA	SYSOPER PRIVILEGES WITH ADMIN OPTION CREATE DATABASE ALTER TABLESPACE BEGIN/END BACKUP RESTRICTED SESSION RECOVER DATABASE UNTIL

➔**Accorder des privilèges système**
Les privilèges systèmes doivent, en générale, être accordé par un administarteur
Utilisez l'instruction SQL GRANT pour accorder des privilèges système aux utilisateurs
GRANT nom_priv_system to nom [with admin option]
Où nom : est un nom utilisateur ou un nom de role
Ex :
Sql> GRANT create session , create table To scott

➔**Supprimer un privilège système**
Sql> REVOKE create session , create table FROM nom
Où nom : est un nom utilisateur ou un nom de role

Important : Pour accorder ou révoquer un privilège system il faut être Administrateur ou avoir acquis le provilège avec l'option with admin option

8.3.2.Privilèges objet

Chaque privilège objet autorise un utilisateur à exécuter une action (select, update, delete, execute, …) spécifique (nommé) sur un objet tel qu'une table, une vue, une séquence, une procédure, une fonction ou un package. Le tableau ci-dessous décrit les classes des privilèges objet :

Privilèges objet

Priv. objet	Table	Vue	Séquence	Procédure
ALTER	√	√	√	√
DELETE	√	√		
EXECUTE				√
INDEX	√	√		
INSERT	√	√		
REFERENCES	√			
SELECT	√	√	√	
UPDATE	√	√		

Les privilèges objets doivent être accordé par le propriétaire de l'objet

➔ **Accorder des privilèges objet**

GRANT [select | update | delete | execute | all] ON [nom_table | nom_procedure] TO nom [with grant option]

Où nom : est un nom utilisateur ou un nom de role

Exemples:

1- donne le droit de sélectionner la table EMP de l'utilisateur SCOTT à l'utilisateur HR

Sql> Grant select on scott.emp to hr

2- donne le droit d'exécuter la procédure PROC1 de SYSTEM à l'utilistaeur SCOTT

Sql> grant execute SYSTEM.PROC1 to SCOTT

➔ **Révoquer des privilèges objet**

L'instruction REVOKE permet de révoquer des privilèges objet. L'utilisateur qui révoque un privilège objet doit être celui qui l'a accordé

REVOKE [select | update | delete | execute | all] ON [nom_table | nom_procedure] FROM nom

Où nom : est un nom utilisateur ou un nom de role

Exemple:

Empêcher l'utilisateur SCOTT de modifier la table EMPLOYEES de l'utilisteur HR

Sql> REVOKE update on HR.EMPLOYEES FROM SCOTT

Important : Pour accorder ou révoquer un privilège objet il faut être le propriètaire de l'objet ou avoir acquis le provilège avec l'option with grant option

8.4. Rôles

Oracle permet de gérer et de contrôler aisément les privilèges à l'aide de rôles. Les rôles sont des groupes nommés de privilèges associés qui sont accordés à des utilisateurs ou à d'autres rôles. Ils facilitent l'administration des privilèges dans une base de données.

8.4.1. Créer des rôles

Sql>CREATE ROLE nom_role

8.4.2. Accorder les roles aux utilisateur ou à un autre rôle

Sql>GRANT nom_role TO nom_user ou nom_rôle2 ;

8.4.3.Accorder des privilèges au Rôles

En utilise la même syntaxe qui permet de donner des privillèges aux utilisateur

```
GRANT  [select | update | delete | execute | all ] ON  [nom_table | nom_procedure ] TO nom_role
 Ou
GRANT nom_priv_system to nom_role
```

8.4.4.Exemples de rôles prédéfinis dans oracle :

Role CONNECT : contient un ensemble de privilèges qui permette d'ouvrir une session et de créer un ensemble d'objet comme des tables , indexes, …

Role RESOURCE : Contient un ensemble de privilèges qui donnent de droit à créer la majorité des type d'objet qui exsiste dans la BD ainsi que le privilège qui défini un quoas illimité sur tous les tablespaces de la BD

Role DBA : Ensemble de privilèges permettant d'administerer le base de données tels que création des utilisateurs, tablespace, ….

8.4.5.Supprimer des privilèges au Rôles

En utilise la même syntaxe qui permet de supprimer des privillèges aux utilisateur

```
REVOKE  [select | update | delete | execute | all ] FROM  [nom_table | nom_procedure ] TO nom_role
 Ou
GRANT nom_priv_system FROM nom_role
```

9. Contraintes d'intégrités

Utilisation :

Les contraintes d'intégrité constituent le mécanisme par excellence pour appliquer des règles pour les raisons suivantes :

- elles améliorent les performances,
- elles sont simples à déclarer et à modifier, dans la mesure où elles nécessitent peu de code,
- elles centralisent les règles,
- elles sont souples (activées ou désactivées),
- elles sont entièrement documentées dans le dictionnaire de données.

Les sections suivantes expliquent le fonctionnement des contraintes d'intégrité et leur mise en oeuvre par le serveur Oracle.

9.1. Types de contrainte

- NOT NULL : Par défaut, toutes les colonnes d'une table acceptent les valeurs NULL (absence de valeur). Une contrainte NOT NULL exige qu'une colonne d'une table contienne des valeurs.

- **UNIQUE** : nécessite que toutes les valeurs d'une colonne ou d'un ensemble de colonnes (clé) soient uniques. Une colonne ou un ensemble de colonnes d'une table ne peut pas contenir deux valeurs de ligne identiques.

- **Clé Primaire :** Chaque table ne peut contenir plus d'une contrainte de clé primaire. Ce type de contrainte garantit :

- que la colonne définie ne peut pas posséder de doublons de valeur de ligne dans une table,
- que les colonnes de clé primaire ne contiennent pas de valeurs NULL.

- **CHECK** : Indique une condition à laquelle doit répondre chaque ligne de la table.
(par ex : saisir une valeur supérieur à zéro).

- **Clé étrangère** : Désigne une colonne ou une combinaison de colonnes comme clé étrangère dans une contrainte d'intégrité référentielle.

9.2. Etats des contraintes

Une contrainte d'intégrité peut être activée (ENABLE) ou désactivée (DISABLE). Si elle est activée, les données sont contrôlées à leur entrée ou leur mise à jour dans la base. Les données qui ne répondent pas à la règle de la contrainte sont refoulées. Si la contrainte est désactivée, il est possible d'entrer des données non conformes dans la base.

Une fois créée une contrainte peut passer aux états :

➔ **ENABLE NOVALIDATE :** Cet état interdit l'entrée de nouvelles données non conformes à la contrainte. La table peut cependant contenir des données non valides (avant l'activation de la contrainte), c'est-à-dire des données qui enfreignent la contrainte. Il s'avère plus utile d'activer des contraintes NOVALIDATE dans des configurations de data warehouse qui téléchargent (vers le serveur) des données OLTP valides.

Syntaxe :

Sql>ALTER TABLE nomtable ENABLE NOVALIDATE CONSTRIANTE nom_contriante
Sql>ALTER TABLE hr.departments ENABLE NOVALIDATE CONSTRAINT dept_pk;

➔**ENABLE VALIDATE :** Cet état interdit l'insertion dans la table de lignes enfreignant la contrainte. Les lignes qui existent, déjà, dans la table (avant activation) doivent absolument vérifiées la contraintes sinon le passage à l'état ANABLE VALIDATE renvoie une erreur.

Syntaxe :

Sql>ALTER TABLE nomtable CONSTRIANTE nom_contriante ENABLE VALIDATE

➔ **DISABLE :** Désactive la vérification de la contrainte

Syntaxe : ***sql>ALTER TABLE nomtable CONSTRIANTE nom_contriante DISABLE***

9.3. Vérification des contraintes

Vous pouvez différer la vérification de la validité des contraintes jusqu'à la fin de la transaction (commit). Il y'a deux type de vérification des contraintes :

➔ **Contraintes non différées ou immédiates**

Les contraintes non différées, également appelées contraintes immédiates, sont appliquées à la fin de chaque instruction LMD.

➔ **Contraintes différées**

Les contraintes différées ne sont vérifiées que lorsqu'une transaction est validée (après le commit). Si une violation de contrainte est détectée lors de la validation, l'ensemble de la transaction est annulé. Ces contraintes s'avèrent très utiles lorsque les lignes parent et les lignes enfant d'une relation de clé étrangère sont entrées simultanément, comme dans le cas d'un système de traitement des commandes où les articles et la commande sont entrés simultanément.

Syntaxe :
SET CONSTRAINT | CONSTRAINTS
{constraint |ALL }
{IMMEDIATE|DEFERRED}
Ex:
sql>set constraint fk_emp deferred;

9.4. Application des contraintes UNIQUE et de CLE PRIMAIRE :

Les contraintes UNIQUE et de CLE PRIMAIRE sont appliquées en utilisant des index.
Un index, s'il n'existe pas déjà, est créé automatiquement une fois qu'on a créé ou activé une contrainte UNIQUE ou de CLE PRIMAIRE
Le serveur Oracle utilise la procédure suivante pour mettre en oeuvre les contraintes UNIQUE et de clé primaire :

- Si la contrainte est désactivée, aucun index n'est nécessaire.
- Si la contrainte est activée et que les colonnes de la contrainte forment la première partie d'un index, cet index, qu'il soit unique ou non-unique, est utilisé pour appliquer la contrainte.
- Si la contrainte est activée et que les colonnes de la contrainte ne correspondent pas au début d'un index, un index portant le nom de la contrainte est créé selon les règles suivantes :
  - S'il s'agit d'une clé pouvant être différée, un index non-unique est créé sur la colonne de clé.
  - S'il s'agit d'une clé ne pouvant pas être différée, un index unique est créé.
- Si un index peut être utilisé et que la contrainte ne peut pas être différée, il convient alors d'utiliser l'index existant. De même, utilisez l'index existant si la contrainte peut être différée et que l'index est non-unique.

9.5. Remarques sur la clé étrangère

Vous devez tenir compte de plusieurs éléments lorsque vous gérez des tables dans une relation de clé étrangère.
- La clé étrangère doit être supprimée avant la table parent. L'instruction suivante permet d'effectuer ces deux opérations :

DROP TABLE table CASCADE CONSTRAINTS
-> supprime d'abord la contrainte puis supprime la table
Ou
***DROP TABLE table* DELETE CASCADE**
->supprime d'abord les lignes de la table fille puis supprime les lignes de la table mère

Le tableau suivant décrit les solutions à faire par rapport aux action effectuées sur la table :

Action souhaitée	Solution appropriée
Supprimer la table parent	Mise en cascade des contraintes
Vider la table parent	Désactiver ou supprimer la clé étrangère
Supprimer le tablespace contenant la table parent	Utiliser la clause CASCADE CONSTRAINTS
Exécuter l'instruction LMD sur la table enfant	S'assurer que le tablespace contenant la clé parent est en ligne

9.6. Création des contraintes

Vous pouvez définir une contrainte lorsque vous créez une table ou lorsque vous la modifiez (après la création)

Syntaxe :

```
column datatype [CONSTRAINT constraint]
                {[NOT] NULL
                |UNIQUE [USING INDEX index_clause]
                |PRIMARY KEY [USING INDEX index_clause]
                |REFERENCES [schema.]table [(column)]
                [ON DELETE CASCADE]
                |CHECK (condition)
                }
                constraint_state :==
                [NOT DEFERRABLE|DEFERRABLE [INITIALLY
        {IMMEDIATE|DEFERRED}]
                ]
                [DISABLE|ENABLE [VALIDATE|NOVALIDATE]]
```

Où :

CONSTRAINT : identifie la contrainte d'intégrité par le nom constraint stocké dans le dictionnaire de données.

USING INDEX : indique que les paramètres définis dans index-clause doivent être utilisés pour l'index auquel fait appel le serveur Oracle afin d'appliquer une contrainte UNIQUE ou de clé primaire (l'index porte le même nom que la contrainte).

DEFERRABLE : indique que la vérification des contraintes peut être différée jusqu'à la fin de la transaction à l'aide de la commande SET CONSTRAINT.

NOT DEFERRABLE : indique que la contrainte est vérifiée à la fin de chaque instruction LMD (une contrainte NOT DEFERRABLE ne peut pas être différée par des sessions ou des transactions. NOT DEFERRABLE est utilisé par défaut.)

INITIALLY IMMEDIATE : indique qu'au début de chaque transaction, la contrainte doit, par défaut, être vérifiée à la fin de chaque instruction LMD (si aucune clause INITIALLY n'est définie, INITIALLY IMMEDIATE est utilisé par défaut).

INITIALLY DEFERRED : indique qu'il s'agit d'une contrainte DEFERRABLE et que, par défaut, elle n'est vérifiée qu'à la fin de chaque transaction.

DISABLE : désactive la contrainte d'intégrité (lorsqu'une contrainte d'intégrité est désactivée, le serveur Oracle ne l'applique pas).

Exemples :

Lorsque vous créez une table

```
CREATE TABLE hr.employee(
id NUMBER(7) CONSTRAINT employee_id_pk PRIMARY KEY DEFERRABLE
   USING INDEX TABLESPACE indx,
last_name VARCHAR2(25) CONSTRAINT empl_last_name_nn NOT NULL,
dept_id NUMBER(7)) TABLESPACE users;
```

Après le création de la table

```
ALTER TABLE hr.employee
 ADD(CONSTRAINT employee_dept_id_fk FOREIGN KEY(dept_id)
 REFERENCES hr.department(id)
 DEFERRABLE INITIALLY DEFERRED);
```

9.7. Recommandations sur la définition des contraintes

Tenez compte des points suivants pour définir des contraintes :

- Placez les index utilisés pour appliquer les contraintes UNIQUE et de clé primaire dans un tablespace différent de celui de la table. Pour ce faire, vous pouvez soit définir la clause USING INDEX, soit créer la table et l'index, puis modifier la table pour ajouter ou activer la contrainte.
- Si les données sont souvent chargées en masse dans une table, il est préférable de désactiver les contraintes, de charger les données, puis de réactiver les contraintes.
- Si une table contient une clé étrangère d'autoréférencement, utilisez l'une des méthodes suivantes pour charger les données :
- Définissez ou activez la clé étrangère après le premier chargement de données.
- Définissez la contrainte comme pouvant être différée.

  La seconde méthode s'avère très utile lorsque des données sont chargées fréquemment

9.8. Identifier une violation de contrainte due aux lignes

Souvent quand on veut activer une contrainte désactivée, en mode ENABLE VALIDATE, on rencontre des erreurs d'activation dues à des lignes qui existent déjà dans la table mais qui ne vérifies pas la contrainte. On doit, alors, identifier ces lignes, les modifier puis réactiver la contrainte.

La clause EXCEPTIONS permet d'identifier les lignes qui violent une contrainte au moment de son activation.

➔ **Etapes pour identifier et rectifier les violations de contrainte, et réactiver une contrainte :**

(1) Si la table EXCEPTIONS n'a pas encore été créée, exécutez le script utlexcpt1.sql pour la créer:
SQL> @?/rdbms/admin/utlexcpt1

(2) Vérifier que la table EXCEPTIONS est vide, puis essayer d'activer la contrainte avec la clause EXCEPTIONS :

```
SQL> ALTER TABLE hr.employee  ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
        EXCEPTIONS INTO system.exceptions;
```

Cette commande insère dans la table EXCEPTIONS les lignes qui violent la contraintes

(3) Identifiez les données non valides en lançant une sous-interrogation sur la table EXCEPTIONS :

```
SQL> SELECT rowid, id, last_name, dept_id  FROM hr.employee
      WHERE ROWID in  (SELECT row_id FROM exceptions) ;
```

ROWID	ID	LAST_NAME	DEPT_ID
AAAAeyAADAAAAA1AAA	1003	Pirie	50

(4) Corriger les erreurs contenues dans les données :

```
SQL> UPDATE hr.employee SET dept_id=10  WHERE rowid='AAAAeyAADAAAAA1AAA';
Sql>commit ;
```

(5) Videz la table EXCEPTIONS et réactivez la contrainte :

```
SQL> TRUNCATE TABLE exceptions;
SQL> ALTER TABLE hr.employee ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
EXCEPTIONS INTO system.exceptions;
```

10. Gestion des indexes

Un index est une arborescence qui permet d'accéder directement à une ligne dans une table.

10.1. Classification des index

Les index peuvent être classés en fonction de leur structure logique ou de leur mise en oeuvre physique. La classification logique regroupe les index en fonction de l'application utilisée, alors que la classification physique dépend du mode de stockage des index.

10.1.1. Classification logique des index :

➔ **Index basé sur une colonne :** La clé d'un ***index basé sur une colonne*** ne contient qu'une seule colonne (par exemple : un index sur la colonne des numéros d'employé d'une table d'employés).

➔ **Index concaténé, ou index composé**, est créé sur plusieurs colonnes dans une table. Il n'est pas nécessaire que les colonnes d'un index concaténé respectent l'ordre des colonnes de la table, ni qu'elles soient adjacentes (par exemple : un index sur les colonnes des services et des postes d'une table d'employés).

➔ **Index uniques et index non-uniques**

Les index peuvent être uniques ou non-uniques. Un index unique garantit qu'il n'existe pas deux lignes portant les mêmes valeurs dans une colonne de clé (ou des colonnes) d'une table. Cette restriction ne s'applique pas pour les index non-uniques.

➔ **Index basés sur une fonction**

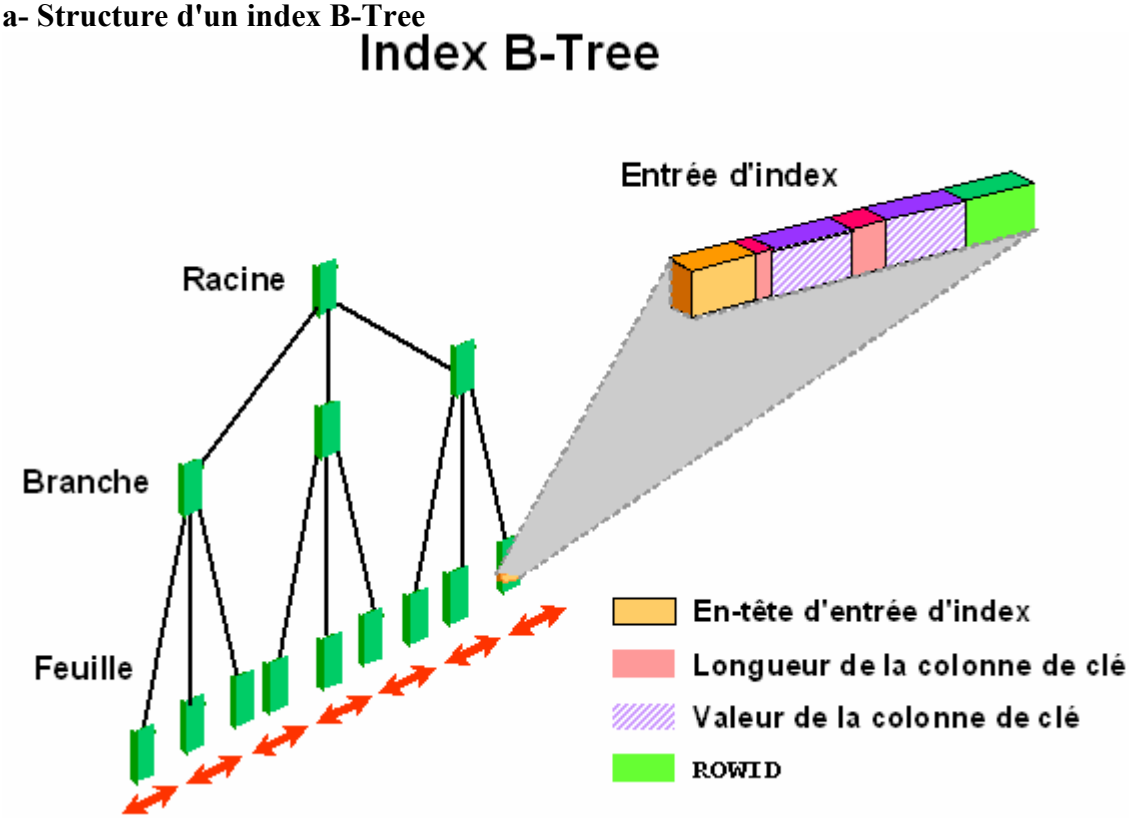
Un index basé sur une fonction est créé lorsque l'utilisateur fait appel à des fonctions ou des expressions impliquant une ou plusieurs colonnes d'une table en cours d'indexation. Ce type d'index précalcule la valeur de la fonction ou de l'expression et la stocke. Ces index peuvent être créés sous forme d'index B-Tree ou bitmap.

10.1.2. Classification physique des index :

➔ Index B-Tree :

Bien que tous les index utilisent une structure B-Tree, l'expression index B-Tree est généralement associée à un index qui stocke une liste de ROWID (adresse physique d'une ligne) pour chaque clé.

a- Structure d'un index B-Tree



Dans la partie supérieure de l'index figure la racine contenant les entrées qui pointent sur le niveau suivant de l'index. Le niveau suivant correspond aux blocs branche qui pointent sur les blocs du niveau suivant de l'index. Au bas de l'arborescence se trouvent les noeuds feuille qui contiennent les entrées d'index pointant sur les lignes de la table. Les blocs feuille comportent deux liens facilitant le balayage de l'index dans l'ordre croissant et décroissant des valeurs de clé.

b- Format des entrées feuille d'un index

Une entrée d'index est constituée :

- d'un en-tête d'entrée contenant le nombre de colonnes et les informations de verrouillage,
- de paires de valeurs correspondant à la longueur de la colonne de clé, qui définissent la taille de la colonne dans la clé et la valeur de la colonne (le nombre de paires correspond au nombre maximal de colonnes dans l'index),
- de l'identificateur ROWID d'une ligne, qui contient les valeurs de clé.

c- Caractéristiques des entrées feuille d'un index :

- Les valeurs de clé se répètent si elles sont communes à plusieurs lignes, sauf si l'index est compressé.
- Il n'existe pas d'entrée d'index pour les lignes dont toutes les colonnes de clé sont NULL. C'est pourquoi une clause WHERE indiquant la valeur NULL aboutira toujours à un balayage complet de table (full table scan).
- L'identificateur ROWID restreint permet de pointer sur les lignes de la table, dans la mesure où toutes les lignes appartiennent au même segment.

d-Créer des index B-Tree normaux :

Vous pouvez créer un index dans le compte de l'utilisateur propriétaire de la table ou dans un compte différent, bien que celui-ci soit généralement créé dans le même compte que celui de la table. L'instruction de la diapositive permet de créer un index sur la table EMPLOYEES à l'aide de la colonne LAST_NAME.

```
CREATE INDEX hr.employees_last_name_idx ON hr.employees(last_name)
PCTFREE 30
STORAGE(INITIAL 200K NEXT 200K  PCTINCREASE 0     MAXEXTENTS 50)
TABLESPACE indx;
```

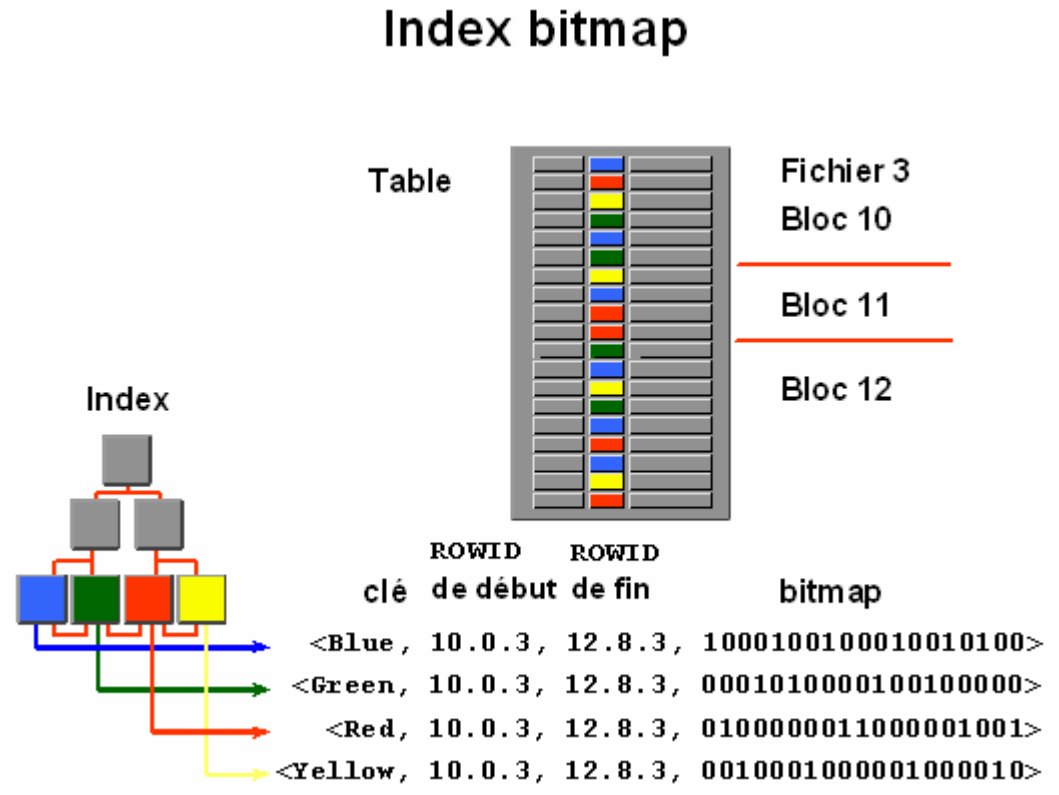
➔ Index bitmap

Dans certains cas, les index bitmap offrent plus d'avantages que les index B-Tree :

- lorsqu'une table contient des millions de lignes et que les colonnes de clé ont une faible cardinalité, c'est-à-dire qu'elles contiennent peu de valeurs distinctes. Par exemple, il est préférable d'utiliser des index bitmap plutôt que des index B-Tree pour les colonnes relatives au sexe et à la situation familiale dans une table contenant des enregistrements de passeport.
- lorsque les interrogations utilisent souvent une combinaison de plusieurs conditions WHERE impliquant l'opérateur OR.
- lorsque les colonnes de clé sont en lecture seule ou qu'elles ne sont pas souvent mises à jour.

a-Structure d'un index bitmap :

Un index bitmap présente également une structure B-Tree, mais le noeud feuille comporte un bitmap pour chaque valeur de clé à la place d'une liste de ROWID. Chaque bit du bitmap correspond à un ROWID possible. Si le bit est défini, la ligne contenant le ROWID correspondant comporte la valeur de clé.



Comme le montre le graphique, le noeud feuille d'un index bitmap contient les éléments suivants :

- un en-tête d'entrée, qui contient le nombre de colonnes et les informations de verrouillage,
- des valeurs de clé correspondant à la longueur et aux paires de valeurs de chaque colonne de clé (dans l'exemple, la clé n'est constituée que d'une seule colonne et la valeur de clé de la première entrée est Blue),
- le ROWID de début contenant, dans l'exemple, le numéro de fichier 3, le numéro de bloc 10 et le numéro de ligne 0,
- le ROWID de fin contenant, dans l'exemple, le numéro de bloc 12 et le numéro de ligne 8,

- un segment bitmap constitué d'une chaîne de bits (le bit est défini uniquement lorsque la ligne correspondante contient la valeur de clé. Le serveur Oracle utilise une technique de compression brevetée pour stocker les segments bitmap).

Le ROWID de début est celui de la première ligne sur laquelle pointe le segment bitmap, c'est-à-dire que le premier bit du bitmap correspond au ROWID, le second bit du bitmap correspond à la ligne suivante du bloc, et le ROWID de fin pointe sur la dernière ligne de la table couverte par le segment bitmap. Les index bitmap utilisent des ROWID restreints.

b-Utiliser un index bitmap :

L'index B-Tree permet de localiser les noeuds feuille qui contiennent les segments bitmap d'une valeur donnée de la clé. Le ROWID de début et les segments bitmap permettent de localiser les lignes contenant la valeur de clé.
Lorsque la colonne de clé est modifiée dans la table, les bitmaps doivent être modifiés. Cette opération verrouille les segments bitmap correspondants. Du fait que les verrous sont appliqués sur l'ensemble du segment bitmap, une ligne couverte par le bitmap ne peut pas être mise à jour par d'autres transactions tant que la première transaction n'est pas terminée.

c-Créer des index bitmap :

Syntaxe
Utilisez la commande suivante pour créer un index bitmap :
CREATE BITMAP INDEX [schema.] index
ON [schema.] table
(column [ASC | DESC] [, column [ASC | DESC]] ...)
[TABLESPACE tablespace]
[PCTFREE integer]
[INITRANS integer]
[MAXTRANS integer]
[storage-clause]
[LOGGING| NOLOGGING]
[NOSORT]

Ex :

```
CREATE BITMAP INDEX orders_region_id_idx ON orders(region_id)
PCTFREE 30
STORAGE(INITIAL 200K NEXT 200K  PCTINCREASE 0     MAXEXTENTS 50)
TABLESPACE indx;
```

10.2. Comparer les index B-Tree et les index bitmap

Comparer les index B-Tree et les index bitmap

Index B-Tree	Index bitmap
Adaptés aux colonnes de forte cardinalité	Adaptés aux colonnes de faible cardinalité
Les mises à jour des clés consomment peu de ressources	Les mises à jour des colonnes de clé consomment de grandes quantités de ressources
Inefficaces pour les interrogations utilisant des prédicats OR	Efficaces pour les interrogations utilisant des prédicats OR
Utiles pour le traitement des transactions en ligne	Utiles pour le data warehouse

LES UTILITAIRES ORACLE

10.3. L'EXPORT & L'IMPORT

Présentation

Ces utilitaires permettent d'effectuer les opérations suivantes :
•Sauvegarde logique : Enregistrer des définitions de tables pour les protéger de tout incident utilisateur.
•Déplacer des données entre des ordinateurs et des bases de données, ou entre différentes versions du serveur Oracle

L'utilitaire **Export** fournit un moyen simple de transférer des objets de données entre des bases Oracle même si celles-ci figurent sur des plates-formes dotées de configurations matérielles et logicielles différentes.
Lorsque vous exécutez l'utilitaire Export sur une base de données, les tables sont extraites, suivis par les objets qui leur sont associés (par exemple, les index, les commentaires et les privilèges, les contraintes), le cas échéant. Les données extraites sont écrites dans un fichier d'export, qui est un fichier dump Oracle de format binaire généralement situé sur disque ou sur bande.
L'utilitaire **Import** lit les définitions d'objet et les données de table à partir du fichier dump généré par l'Export, puis il insère les objets de données dans une base Oracle.

10.3.1. Appeler l'utilitaire 'export' : exp

L'export consiste à créer un fichier binaire qui contient les instuctions sql nécéssairent pour reconstruire les tables d'une base de données

Syntaxe :

```
exp keyword = value, value2, … ,valuen
```

On peut appeler l'export en trois modes possible :

- **Mode tables** : Ce mode permet de n'exporter que les tables indiquées
  Exemple :

Créer un fichier d'export appelé `exp1.dmp` qui comprend aux tables `EMPLOYEES` et `DEPARTMENTS` du schéma HR et inclut les lignes :

```
exp hr/hr tables=employees,departments rows=y file=exp1.dmp
```

-

- **Mode utilisateur (Owner) :**

Ce mode permet d'exporter tous les objets du schéma d'un utilisateur. Les utilisateurs ayant les privilèges requis peuvent exporter tous les objets des schémas d'un ensemble d'utilisateurs spécifique.
Exemple :
Créer un fichier d'export appelé `expdat.dmp` qui comprend tous les objets du schéma HR :

```
exp system/manager owner=hr  file=expdat.dmp
```

- **Mode base de données complète (Full) :**

Ce mode permet d'exporter tous les objets de la base de données, hormis ceux du schéma SYS. Seuls les utilisateurs ayant les privilèges requis peuvent exporter des objets dans ce mode.
Exemple :
Créer un fichier d'export appelé `expfull.dmp` qui comprend tous les objets du schéma HR :

```
exp system/manager full=y  file=expfull.dmp
```

10.3.2. Appeler l'utilitaire 'import' : imp

L'import consiste à créer des tables ou/et y inserer les données correspondant à partir d'un fichier déjà généré par l'export

Syntaxe :

```
imp keyword = value, value2, … ,valuen
```

On peut appeler l'import en trois modes possible :

- **Mode tables** : Ce mode permet de n'importer que les tables indiquées à paritir d'un fichier d'export
  Exemple :

`Importer les` tables `EMPLOYEES` et `DEPARTMENTS` du schéma HR et inclut les lignes à partir du fichier d'export appelé `exp1.dmp` **:**

```
imp hr/hr tables=employees,departments rows=y file=exp1.dmp
```

-

- **Mode utilisateur (Owner) :**

Ce mode permet d'importer tous les objets du schéma d'un utilisateur à partir d'un fichier d'export.
Exemple :
`Importé` tous les objets du schéma HR dans le shema de l'utilistaeur scott à partir du fichier d'export appelé `expdat.dmp`:

```
imp system/manager FROMUSER=hr  TOUSER=scott file=expdat.dmp
```

- **Mode complèt (Full) :**

Ce mode permet d'importer tous les objets incluts dans le fichier d'export. **Exemple :**
Importer tous le contenu du fichier d'export appelé `expfull.dmp`

```
imp system/manager full=y  file=expfull.dmp
```

10.4. SQL LOADER

10.4.1. Caractéristiques de SQL*Loader

L'utilitaire SQL*Loader charge les données à partir des fichiers textes externes vers les tables d'une base de données Oracle. Ses caractéristiques sont les suivantes :

•SQL*Loader peut utiliser un ou plusieurs fichiers de donées en entrée.
•Les données en entrée peuvent se présenter sous n'importe quel format : caractère, binaire, décimal ou date
•Les données peuvent être chargées à partir de différents types de support, tels que des disques, des bandes ou des canaux nommés.
•Les données peuvent être chargées simultanément dans plusieurs tables.
•Des options sont disponibles pour ajouter des données aux tables ou remplacer des données existantes.
•Les fonctions SQL peuvent être appliquées aux données en entrée avant le stockage de la ligne dans la base de données.
•Les valeurs de colonne peuvent être générées automatiquement en fonction de règles. Par exemple, une valeur de clé séquentielle peut être générée et stockée dans une colonne.

10.4.2. Fichiers utilisés par SQL*Loader

L'utilitaire SQL*Loader utilise les fichiers suivants :
•**Fichier de contrôle** : (à ne pas confondre avec le fichiers de contrôle de la B.D) C'st un fichier texte qui définit le format des données en entrée, les tables de sortie et les conditions facultatives permettant de ne charger qu'une partie des enregistrements de fichiers de données en entrée.
•**Fichiers de données en entrée** : contiennent les données au format défini dans le fichier de contrôle.
•**Fichier de paramètres :** fichier facultatif permettant de définir les paramètres de ligne de commande pour le chargement.
•**Fichier journal :** fichier créé par SQL*Loader, qui contient l'enregistrement du chargement.
•**Fichier des enregistrements refusés** : utilisé par l'utilitaire pour écrire les enregistrements rejetés lors du chargement des données (cette situation peut se produire lorsque l'utilitaire valide les enregistrements en entrée ou lorsque le serveur Oracle insère les enregistrements).
•**Fichier de rebut :** créé, si nécessaire, pour stocker tous les enregistrements qui ne répondent pas aux critères de sélection.

10.4.3. Utiliser SQL*Loader en

Ligne de commande

Pour appeler sql*loder on utilise la commande suivante :

```
C:\> sqlldr user/password  control=nom_fich_control  log=nom_fichier_log
```

Où
user: l'utlisateur Oracle qui va ouvrir le sssion sql*loader
password : mot de passe

10.4.4. Contenu du fichier de contrôle de SQL*LOADER

L'exemple ci-dessous présente un fichier de contrôle de SQL*Loader standard.

```
1 -- Exemple de fichier control file,
2 LOAD DATA
3 INFILE 'SAMPLE.DAT'
4 BADFILE 'sample.bad'
5 DISCARDFILE 'sample.dsc'
6 APPEND
7 INTO TABLE emp
8 fields terminated by ','
9 TRAILING NULLCOLS
10 (col1 SYSDATE,col2 , …, col 3 )
```

Explication de cet exemple :
1.Les commentaires sont introduits par le signe '- -' dans le fichier de contrôle. Ils peuvent apparaître à n'importe quelle ligne de commande du fichier, mais pas à l'intérieur des données.
2.L'instruction `LOAD DATA` informe SQL*Loader qu'un nouveau chargement de données commence. Si vous poursuivez un chargement interrompu en cours d'exécution, vous devez utiliser l'instruction `CONTINUE LOAD DATA`.
3.Le mot-clé `INFILE` indique le nom d'un fichier de données contenant les données à charger.
4.Le mot-clé `BADFILE` indique le nom d'un fichier dans lequel sont placés les enregistrements rejetés.
5.Le mot-clé `DISCARDFILE` indique le nom d'un fichier dans lequel sont placés les enregistrements de rebut.
6.Le mot-clé `APPEND` est l'une des options que vous pouvez utiliser lors du chargement des données dans une table qui contient déjà des données. Pour charger des données dans une table vide, utilisez le mot-clé `INSERT`.
7.Le mot-clé `INTO TABLE` vous permet d'identifier les tables, les champs et les types de données. Il définit la relation entre les enregistrements du fichier de données et les tables de la base de données.
8.Indique le séparateur des champs dans le fichier de odnnées, ici il sont séparés par ',' .
9.La clause `TRAILING NULLCOLS` demande à SQL*Loader de traiter toutes les colonnes en position relative qui ne figurent pas dans l'enregistrement comme des colonnes NULL.
10.Le reste du fichier de contrôle contient la liste des champs et fournit des informations sur les formats de colonne de la table en cours de chargement.

10.4.5. Exemple de fichier de données

```
101,hamid,8000
103,khalid,9000
. . . .
110 ,hakim,8500
```