Cours DBA Oracle 1

Une instance oracle (un lancement de la base) :

- Permet d'accéder à une base de données Oracle
- N'ouvre qu'une seule base de données
- Est constituée de structure de processus d'arrière plan et de structure mémoire (SGA).

Instance		
Mémoire SGA (structures mémoire)		
Zone de mémoire partagée Cache « librairy » Cache du dictionnaire de données	Gestion de tampons de la base de données Zone de mémoire Java	Tampons de journalisation Zone de mémoire LARGE SPOOL
Structure de processus d'arrière-plan		
Pmon | Smon | Dbwr | Lgwr | Ckpt | Autres		

Mémoire SGA (system or shared global area) : alloué au démarrage de l'instance, composante fondamentale d'une instance Oracle.
On peut la voir par **show SGA :**

Mémoire PGA (Program global area) : alloué au démarrage du processus serveur.

3 types de fichiers :

1) fichiers de paramètres : a) fichier PFILE init.ora, b) fichier de mots de passe (pwd …).
2) Fichiers de la base : a) fichiers de données (datafile), b) fichiers de contrôle, c) fichiers de journalisation ON LINE (redo log),
3) Fichiers de journalisation archivés OFF LINE (arch log),.

En V9, La mémoire SGA :

- Est dynamique,
- Sa taille est définie par le paramètre SGA_MAX_SIZE
- Son allocation et son suivi sont effectués sous la forme de granules, par les composant de la SGA :
- => allocation e mémoire virtuelle contiguë,
- => taille des granules définie en fonction de la valeur totale estimée de SGA_MAX_SIZE

La Zone de mémoire partagée :

- permet de stocker :
- => les dernières instructions SQL exécutées,
- => les dernières définitions de données utilisées
- est constituée de 2 structures mémoire clés liées aux performances :
- => cache « librairy »,
- => du dictionnaire de données

- est définie par le paramètre SHARED_POOL_SIZE (alter system set SHARED_POOL_SIZE = 64M;).

Le Cache “librairy” :

- conserve des informations sur les dernières instructions SQL et PL/SQL utilisées,
- permet le partage des instructions communes,
- est gérée par l’algorithme LRU (list recent use),
- est composée de 2 structures :
- => Zone SQL partagée
- => Zone PL/SQL partagée
- sa taille dépend du dimensionnement de la zone de mémoire partagée.

Le cache du dictionnaire de données :

- contient les dernières définitions utilisées dans la base,
- contient des informations sur les fichiers, les tables, les index, les colonnes, les utilisateurs, les privilèges et d’autres objets de la base de données.
- Au cours de l’analyse, le processus serveur recherche des informations dans le cache du dictionnaire, pour résoudre les noms d’objet et valider l’accès.
- La mise en mémoire cache des informations du dictionnaire de données réduit le temps de réponse aux interrogations et aux instructions LMD,
- Sa taille dépend du dimensionnement de la zone mémoire partagée.

Le cache de tampons de la base :

- Conserve des copies des blocs de données extraits des fichiers de données,
- Permet des gains de performances considérables, lors de l’obtention et de la mise à jour de données,
- Est géré par un algorithme LRU,
- Le paramètre DB_BLOCK_SIZE détermine la taille du bloc principal.
- Est composé de sous-caches indépendants :
- => DB_CACHE_SIZE
- => DB_KEEP_CACHE_SIZE
- => DB_RECYCLE_CACHE_SIZE
- peut être redimensionné dynamiquement :
- alter system set DB_CACHE_SIZE = 96M;
- le paramètre DB_CACHE_ADVICE peut être défini pour collecter des statistiques permettant de prévoir le comportement du serveur en fonction de différentes tailles du cache.
- La vue V$DB_CACHE_ADVICE affiche les statistiques collectées.

Le tampon de journalisation :

- Enregistre toutes les modifications apportées aux blocs de données de la base.
- Sa principale fonction est la récupération des données.
- Les modifications enregistrées constituent des entrées de journalisation.
- Les entrées de journalisation contiennent des informations permettant de reconstruire des modifications,
- La taille du tampon est définie par le paramètre LOG_BUFFER.

La zone de mémoire LARGE POOL :

- Est une zone facultative de la mémoire SGA,
- Réduit la charge de la zone de mémoire partagée,
- Est utilisée pour :
- => La mémoire allouée par session (UGA) au serveur partagé,

- => les processus serveurs d'E/S
- => les opérations de sauvegarde ou de restauration ou RMAN (recovery manager).
- => les mémoire tampon des messages d'exécution en parallèle : PARALLEL_AUTOMATIC_TUNING = TRUE
- n'utilise pas de liste LRU,
- sa taille est définie par le paramètre : LARGE_POOL_SIZE,
- peut-être redimensionné dynamiquement.

La zone de mémoire JAVA :
- répond aux besoins d'analyse des commandes Java.
- Est nécessaire si Java est installé et utilisé.
- Sa taille est définie par le paramètre JAVA_POOL_SIZE.

La mémoire PGA :
- Est réservée à chaque processus utilisateur qui se connecte à une base de données Oracle.
- Est alloué lors qu'un processus Oracle est crée,
- N'est utilisé que par un seul processus.

Différence d'allocation mémoire PGA, entre serveur dédié et le serveur partagé :

1 Zone de mémoire	**Serveur dédié**	**Serveur partagé**
Type de mémoire allouée par session	Privée	Partagée
Emplacement de la zone persistante	Mémoire PGA	Mémoire SGA
Emplacement de la zone d'exécution (SELECT)	Mémoire PGA	Mémoire SGA
Emplacement de la zone d'exécution (LMD/LDD)	Mémoire PGA	Mémoire SGA

Structure de processus :

Oracle utilise différents types de processus :

- **Processus utilisateur** : est démarré au moment où un utilisateur de la base de données effectue une demande de connexion au serveur Oracle
- **Processus serveur** : se connecte à l'instance Oracle et démarre lorsqu'un utilisateur établit une session.
- **Processus d'arrière-plan** : sont lancé au démarrage d'une instance Oracle.

Le processus utilisateur :

- Est un programme qui demande une interaction avec le serveur Oracle,
- Doit d'abord établir une connexion,
- N'entre pas directement en interaction avec le serveur Oracle.

Le processus serveur :
- Est un programme qui entre directement en interaction avec le serveur Oracle.
- Répond aux appels générés et renvoie les résultats.
- Peut être un serveur dédié ou un serveur partagé.

Les processus d'arrière-plan :

- Gèrent et appliquent les relations entre les structures physiques et les structures mémoire :
- => processus d'arrière-plan obligatoires : DBWn, pmon, smon, lgwr, ckpt
- => processus d'arrière-plan facultatifs : ARCn (archivage), CJQ0 (coordinateur job queue), Dnnn (répartiteur), LCKn (verrous d'instance), LMDn (verrous à distance), lmon (verrous globaux), LMS (RAC Global cache service), Pnnn (Processus esclave « parallel Query »), QMNn (Adevenced Queuing), reco (récupérateur), Snnn (serveur partagé).

Le processus database writer (DBWn) écrit dans les cas suivants :

- Point de reprise (chekpoint),
- Seuil des tampons « dirty » atteint,
- Aucune mémoire tampon disponible,
- Temps imparti dépassé,
- Demande de ping RAC,
- Tablespace hors ligne
- Tablespace en lecture seule,
- DROP ou TRUNCATE sur une table
- BEGIN BACKUP sur une tablespace.

Le processus log writer (LGWR) effectue une opération d'écriture dans les cas suivants :

- Validation,
- Un tier du cache est occupé,
- La journalisation atteint 1 Mo,
- Toutes les 3 secondes,
- Avant que le processus DBWn ne procède à une opération d'écriture.

Le processus system monitor (SMON) a :

- Comme responsabilité :
- => 1) la récupération d'instance :
- => => réimplantation des modifications dans les fichiers de journalisation en ligne.
- => => ouverture de la base de données pour permettre l'accès aux utilisateurs,
- => => annulation des transactions non validées,
- => 2) la fusion de l'espace libre,
- => 3) la libération des segments temporaires.

Le processus « processus monitor » (PMON) :

- Suite à l'échec du processus, PMON exécute des opérations de nettoyage :
- => annulation de la transaction en cours,
- => libération des verrous,
- => libération d'autres ressources,
- => redémarrage des répartiteurs interrompus.

Le processus Checkpoint (CKPT) à pour responsabilité :

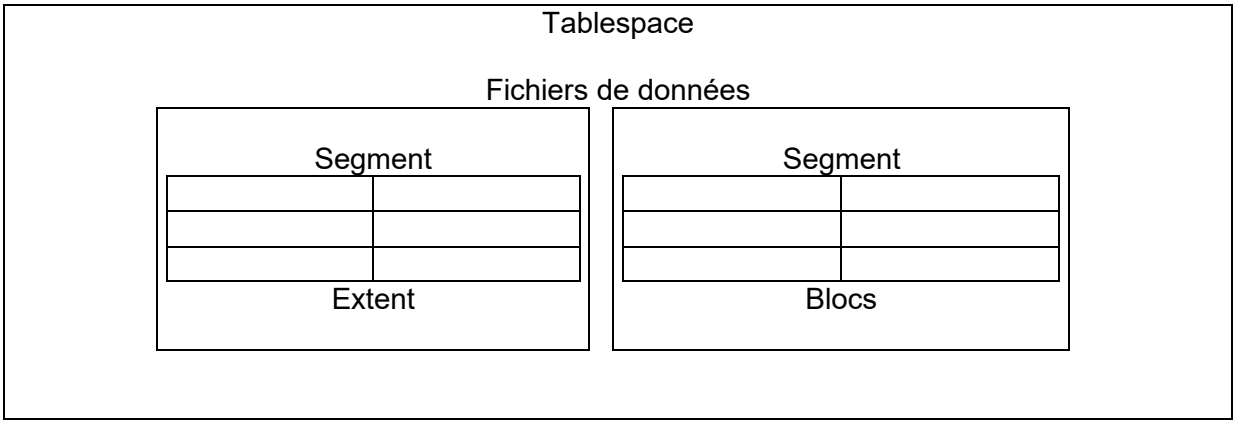
- => du signalement de DBWn aux points de reprise,
- => de la mise à jour des en-têtes de fichiers de données avec les informations sur le point de reprise
- => de la mise à jour des fichiers de contrôle avec les informations sur le point de reprise.

Le processus d'archivage (ARCn) :

- est un processus d'arrière-plan facultatif,

- en mode ARCHIVELOG, archive automatiquement les fichiers de journalisation en ligne,
- enregistre toutes les modifications apportées à la base de données.

La structure logique :
- définit le mode d'utilisation de l'espace physique d'une base de données.
- Possède une hiérarchie composée de tablespaces, de segments, d'extents et de blocs.



BD ⊂ (1 ou plusieurs Tablespace(s)),
1 Tablespace ⊂ (1 ou plusieurs datafiles _ fichiers de données),
1 Tablespace ⊂ (1 ou plusieurs segments),
1 segment est composé d'extents.
1 extent est composé de blocs logiques
(1 bloc = plus petite unité de lecture et d'écriture).

A l'exception du tablespace SYSTEM et des tablespaces contenant un segment d'annulation actif, les tablespaces peuvent être mis hors ligne, sans nécessité de l'arrêt de la base.

Traitement d'une instruction SQL :

a) Parse : analyse (vérif syntaxe, verrouillage objets, création plan exécution),
b) Bind : affectation de valeurs (getb valeurs variables),
c) Execute,
d) Fetch (extraction).

Traitement d'une instruction DML :

e) Parse : analyse (vérif syntaxe, verrouillage objets, création plan exécution),
f) Bind : affectation de valeurs (getb valeurs variables),
g) Execute,

2 Outils d'administration

Outils	**Description**
Oracle Universal Installer (OUI)	Permet d'installe, de mettre à niveau ou de supprimer des composants logiciels : ./runInstaller (mode intéractif) ./runInstaller –responsefile myResponsfile -silent (mode non intéractif)

Outils	Description
Oracle Database Configuration Assistant	Outil graphique, en interaction avec OUI, pour créer, supprimer, modifier une base
SQL*plus	Utilitaire permettrant d'accéder à des données d'une base de données Oracle
Oracle Enterprise Manager	Interface utilisateur permettant d'administrer, de surveiller et de régler une ou plusieurs bases de données

3 Administrateurs de la base de données

SYS : propriétaire du dictionnaire de données (rôle administrateur, MdP : change_on_install),
SYSTEM : propriétaire des tables et vues internes utilisées par les outils Oracle (rôle administrateur, MdP : manager).

SQPLUS (en V9) :
Rôle : Démarrer, arrêter, manipuler, d'interroger la base de données.
Sqlplus /nolog
Connect / as sysdba

3.1 Oracle Enterprise Manager

Usages : tâches administration (ajout fichiers, modifications, augmentation de tailles, de paramètres, réglages, diagnostics) sur plusieurs bases.

3.2 Fichier de paramétrage d'initialisation PFILE « init<sid>.ora »

Contient 2 types de paramètres :

- Explicite : le fichier contient une entrée.
- Implicite : le fichier ne contient pas d'entrée (l'instant prend les valeurs oracle par défaut).

Une instance peut présenter plusieurs fichiers de paramètres d'initialisation.

- Pfile : fichier de paramètres, statique
- Spfile : fichier de paramètres serveur, persistant.

Ces fichiers se trouvent :

1) Sous unix :
$ORACLE_HOME/dbs/initSID.ora : fichier PFILE.
$ORACLE_HOME/dbs/spfileSID.ora : fichier SPFILE.

2) Sous Windows :
%ORACLE_HOME%\database\initSID.ora : fichier PFILE.
%ORACLE_HOME%\database\spfileSID.ora : fichier SPFILE.

On ne peut pas éditer le fichier SPFILE, car contenant du binaire. Mais on peut le modifier par :

ALTER SYSTEM … (permet de modifier le SPFILE).

Création d'un fichier SPFILE :

CREATE SPFILE='$ORACLE_HOME/dbs/spfileDBA01.ora' FROM PFILE=
'$ORACLE_HOME/dbs/initDBA01.ora';
(on peut faire l'inverse : CREATE PFILE FROM SPFILE ;).
La vue V$PARAMETER permet de visualiser les paramètres du SPFILE.

Modification des paramètres d'un fichier SPFILE :

ALTER SYSTEM SET undo_tablespace = UNDO2 ;
ALTER SYSTEM SET undo_tablespace = UNDO2 SCOPE BOTH ;
ALTER SYSTEM RESET undo_suppress_errors SCOPE BOTH SID='*' ;

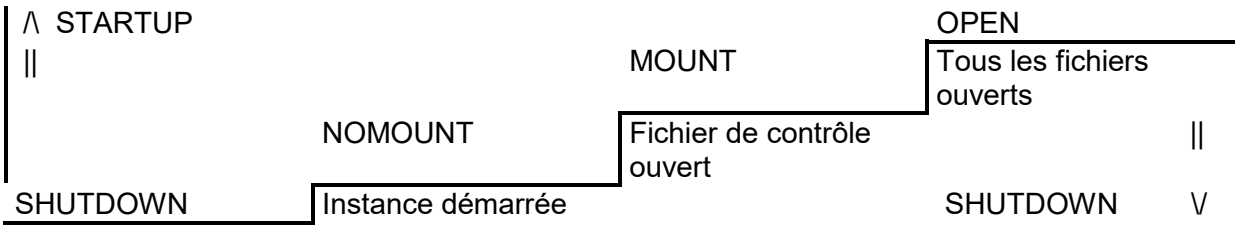
D'une manière générale :
ALTER SYSTEM SET parameter_name = parameter_value
[COMMENT 'text'] [SCOPE = MEMORY | SPFILE | BOTH] [SID = 'sid' | '*'] ;

4 Fonctionnement de la commande STARTUP :

1) Ordre des priorité :
spfileSID.ora > SPFILE par défaut > initSID.ora > pfile par défaut.

2) STARTUP PFILE = $ORADCLE_HOME/dbs/initDBA1.ora

3) un fichier PFILE doit indiquer qu'un fichier SPFILE doit être utilisé :
SPFILE = /database/startup/spfileDBA1.ora



NOMOUNT :
a) pour création base de données,
b) recréation de fichiers de contrôle.

MOUNT :
Usage :
a) renommer des fichiers de données,
b) activer ou désactiver des options d'archivage de journalisation en ligne.
c) Effectuer une récupération complète de la base de données.

OPEN :
Accès à la base de données.

Alter database :

ALTER DATABASE sid MOUNT ;
ALTER DATABASE sid OPEN ;
ALTER DATABASE sif OPEN READ ONLY ; : pas de journalisation, lecture seule, tris, récupérer des fichiers de données hors ligne;

STARTUP RESTRICT ;

ALTER SYSTEM ENABLE RESTRICTED SESSION ;

4.1 Ouvrir une base de données en mode d'accès restreint

STARTUP RESTRICT ;
ALTER SYSTEM ENABLE RESTRICTED SESSION ;

4.1.1 Arrêter la base de données

Mode d'arrêt	**A**	**I**	**T**	**N**
Permet de nouvelles connexions	Non	Non	Non	Non
Attend la fin des cessions en cours	Non	Non	Non	Oui
Attend à la fin de la transaction en cours	Non	Non	Oui	Oui
Applique un point de reprise et ferme les fichiers	Non	Oui	Oui	Oui

Mode d'arrêt :
A = ABORT
I = Immédiate
T = Transactional
N = Normal

4.1.2 SHUTDOWN NORMAL

Conditions :
Aucune connexion ne peut être établie.
Attend la déconnexion préalable de tous les utilisateurs.
Les tampons de journalisation et de la base données sont écris 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.

Ce que cela provoque :
Le cache de tampons de la base est écrit dans les fichiers de données.
Les modifications non validées sont annulées.
Les ressources sont libérées.

⇨ à la fin, base de données cohérente (« clean »).

4.1.3 SHUTDOWN TRANSACTIONAL

Ce type d'arrêt évite aux clients de perdre leurs travaux en cours.
Aucun client ne peut lancer une transaction, pour l'instance en cours.
Le client est déconnecté lorsqu'il termine une 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 démarrage.

4.1.4 SHUTDOWN IMMEDIATE

Les instructions SQL en cours de traitement ne sont pas terminées.
Le serveur Oracle n'attend pas la déconnexion des utilisateurs de la base de données.
Oracle annuel 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 démarrage.

4.1.5 SHUTDOWN ABORT

Les instructions SQL en cours de traitement sont immédiatement interrompues.
Le serveur 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 de fichiers (!).
La base de données n'est ni fermée, ni démontée (!).
Une récupération est nécessaire au démarrage ; elle s'effectue automatiquement.
Remarque : il déconseillé de sauvegarder une base de données incohérente.
⇨ à la fin, base de données incohérente (« dirty »).

Ce que cela provoque :
Les mémoire tampon modifiées ne ont pas écrites dans les fichiers de données.
Les modifications non validées ne sont pas annulées.

4.1.6 STARTUP FORCE

Les fichiers de journalisation en ligne permettent de réappliquer les modifications.
Des segments d'annulation sont utilisés pour annuler les modifications non validées.
Les ressources sont libérées.

4.1.6.1 Fichiers de diagnostic permettant de surveiller une instance

3 types :
- alert<SID>.log
- trace de processus d'arrière-plan : <sid>_<processname>_<PID>.trc
- trace utilisateur : ora_<pid>_<sid>.trc

Ils contiennent info sur événements …

exemples :
snp1_<pid>_<sid>.trc
snp2_<pid>_<sid>.trc
snp3_<pid>_<sid>.trc
snp4_<pid>_<sid>.trc
ckpt_<pid>_<sid>.trc
ckpt_<pid>_<sid>.trc
dbw0_<pid>_<sid>.trc
smon_<pid>_<sid>.trc

Emplacement des fichiers traces des processus en arrière-plan défini par le paramètre « BACKGROUND_DUMP_DEST ».
Visualisables par la commande SHOW PARAMETER BACKGROUND_DUMP_DEST ;

Emplacement des fichiers traces des utilisateurs défini par le paramètre « USER_DUMP_DEST ».
Visualisables par la commande SHOW PARAMETER USER_DUMP_DEST ;

Leur taille est déterminée par le paramètre : MAX_DUMP_FILE_SIZE

Activer ou désactiver la fonction de trace utilisateur :

a) au niveau session, exécuter commande :
ALTER SESSION SET SQL_TRACE = TRUE

b) au niveau session, executer procedure DBMS :

dbms_system.SET_SQL_TRACE_IN_SESSION

c) au niveau instance, definer parameter initialisation : SQL_TRACE=TRUE

Créer une base de données

Planification de la création d'un base :

1. Fonction rôle de la base,
2. Type de la base,
3. Architecture de la base,
4. Choix nom de la base.
5. Créer la base.

« Oracle data migration assistant » permet de migrer à partir d'une version antérieure de la base (dangereux).

Commande de création de base de données : CREATE DATABASE …

Préconisations de l'architecture OFA (Optmal Flexible Architecture)

Préconisation par Oracle Corp : 3 règles :

- structure de répertoire, facilitant le stockage de n'importe quel fichier de la base sur n'importe quel disque,
- mettre objets ayant des comportements différents, dans des tablespaces différents.
- Pour optimiser la fiabilité et les performances, mettre les composants d'un base sur des disques différents.

Authentification par fichier mot de passe

orapwd file=$ORACLE_HOME/dbs/orapw<sid> password=xxxxx entries=nb

Et rajouter la ligne, dans le fichier "init<SID>.ora' :
REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE

Variables de l'environnement Oracle

ORACLE_HOME :
ORACLE_SID :
PATH :
LD_LIBRAIRY-PATH :
NLS_LANG (ou ORA_NLS33) :

Syntaxe de la commande « create database »

```
create database database
user sys identified by MdP (en general "change_on_install").
user system identified by MdP2 (en general "manager").
controlfile reuse
logfile group integer filespec
maxlogfiles integer
maxlogmembers integer
maxloghistory integer
maxdatafiles integer
maxinstances integer
archivelog | noarchivelog
character set charset
```

```
national character set charset
datafile filespec |autoextentd_clause]
filespec:= 'filename' [size integer][K|M] [reuse]
autoextend_clause :=
     [autoextend {OFF|ON [NEXT integer [K|M]] [MAXSIZE {unlimited|integer [K|M]}}]
default temporary tablespace tablespace filespec
   temp_tablespace_extent_clause
   temp_tablespace_extent_clause :=
                  extent management local uniform [size integer [K|M] ]
undo tablespace tablespace datafile filespec |autoextentd_clause]
set time_zone [ time_zone_region ]
```

Voici un exemple d'un tel script :

```
-------------------------------------------------------------------------------
-- BACUS : Script de création de la database
-------------------------------------------------------------------------------
-- NOTES
--     appelé par buildxxx et recre_db
-------------------------------------------------------------------------------
-- spool information to log file --
set echo on
spool /util/admindb/bac/oper/crdb.log
-- démarrage instance database monted (non ouverte)
--
connect internal
startup nomount
--
--  Création db BACxxx ---------------------------------------------------

create database BACDBO
    controlfile reuse
    logfile '/ora_redo_log/bacdbo/log1bac.ora' size 5000K reuse,
            '/ora_redo_log/bacdbo/log2bac.ora' size 5000K reuse
    maxlogfiles 5
    datafile '/ora_data_tbl/bacdbo/bactssy1.ora' size 70M reuse
    maxinstances 1
    noarchivelog
    character set WE8ISO8859P1;

-- Création 2ème RB segment dans TS system et activation -----------------
create public rollback segment rb_temp
       tablespace system ;
alter rollback segment rb_temp online;

--
alter tablespace system
    default storage (initial 4K  next 4k
                     pctincrease 10 minextents 1 maxextents 2048);
--
-- création des tablespaces  -----------------
@/util/admindb/bac/oper/tsdbo.sql
--
-- script de création du catalogue/ PL/SQL CATBLOCK TOOLS LOADER
@?/rdbms/admin/catalog
@?/rdbms/admin/catproc
connect system/manager
@?/sqlplus/admin/pupbld.sql
--
-- arrêt database
connect internal
shutdown
disconnect
```

Après la création de la base, elle contient 2 utilisateurs « sys » et « system » et des tables internes (mais pas les vues du dictionnaire de données) et les vues de performances v$logfile, v$controlfile, v$datafile.

Création des autres objets intégrés de la base :

- Dictionnaire de données,
- Tables des performances
- Packages PL/SQL (créées par « catproc.sql »).
- Déclencheurs d'évènements de la base de données.

Erreurs au moment de la création de la base

Cas :
- Erreurs de syntaxe
- Fichiers existants déjà,
- Ressources trop petites (partition trop petite), droits d'accès restreints.

Dictionnaire de données

- Tables (créées par CREATE DATABASE) et vues (créées par catalog.sql) en lecture seule.
- Stockés dans le tablespace « SYSTEM ».
- Propriétés de l'utilisateur « SYS ».
- (accessible avec le privilège SELECT).

Il fournit des informations sur :
- la structure logique et la structure physique de la base de données,
- les définitions d'objets et l'espace alloué aux objets (toutes infos sur les objets du schéma),
- les contraintes d'intégrité
- les utilisateurs
- les rôles
- les privilèges
- la fonction d'audit.

3 types de vues :
- DBA_xxx : contenu de tous les schémas
- ALL_xxx : objets, éléments auxquelles les utilisateurs en cours ont accès
- USER_xxxx : : contenu du schéma de l'utilisateur (objets appartenant à l'utilisateur en cours).

Quelques exemples :

Type	**Nom**	**Rôle / affiche …**
Présentation générale	Dictionnary	
	Dict_columns	
Objets de schéma	Dba_tables	
	Dba_indexes	
	Dba_tab_columns	
	Dba_constraints	
	Dba_views	
	Dba_objects	

Type	Nom	Rôle / affiche …
Affectation de l'espace	Dba_segments	
	Dba_extents	
	Dba_free_space	Volume d'espace disponible de la bd
	Dba_segments	
Structure de la base de données	Dba_tablespaces	
	Dba_data_files	
utilisateurs	Dba_users	

Tables dynamiques de performance

- Tables virtuelles
- Enregistre l'activité en cours de la base de données
- Sont constamment mises à jour, lorsque la base de données est active
- Les informations sont lues à partir de la mémoire et du fichier de contrôle,
- Permettent de surveiller et de régler la base de données,
- Propriétés de l'utilisateur « SYS ».
- Les synonymes commencent pa le préfixe « V$ »
- Sont répertoriées dans la vue « V$FIXED_TABLE ».

Exemples de ces vues dynamiques

Nom	Rôle (affiche …)
V$controlfile	Noms des fichiers de contrôle
V$database	Infos sur la base, provenant du fichier de contrôle
V$datafile	Infos sur fichiers de données, venant du fichier de contrôle
V$instance	Etat de l'instance en cours
V$parameter	Paramètres et valeurs utilisées par la session
V$cession	Infos sur la session en cours
V$sga	Infos de synthèse sur la SGA (System Global Area)
V$spparameter	Contenu du SPFILE
V$tablespace	Infos sur le tablespace, provenant du fichier de contrôle
V$thread	Infos sur le thread, provenant du fichier de contrôle
V$version	N° de version des composants de la bibliothèque principale (core librairy) du serveur Oracle
V$backup	
V$tempfile	
V$archive	
V$log	
V$logfile	
V$loghist	
V$archived_log	

On peut avoir des informations par :
Select * from v$parameter ;
Ou par
Show parameter
Select * from v$sga ;
Ou par
Show sga ;
select HOST_NAME from v$INSTANCE;

Convention d'appellation des scripts d'administration

Convention	Description	Exemple
cat*.sql	Information du catalogue et du dictionnaire de données	catalog.sql catproc.sql catadt.sql catnoadt.sql
dbms*.sql	Spécifications de package de la base de données	dbmspool.sql
prvt*.plb	Code de package de la base de données crypté	
utl*.sql	Vues et tables des utilitaires de la base de données	

Fichiers de contrôle

- Fichier binaire (en général de petite taille)
- Définit l'état actuel de la base
- Assure l'intégrité de la base de données
- Utilisé et indispensable à l'étape MOUNT du démarrage de la base et pour le fonctionnement de la base,
- Lié à une seule base,
- Sa perte nécessite la récupération de la base,
- Sa taille initiale est définie au moment du « create database … »'.

Il contient les entrées suivantes :

- Nom et identificateur de la base de données,
- Horodatage de création de la base de données,
- Nom des tablespaces
- Nom et emplacement des fichiers de données et fichiers de journalisation en ligne (on line redo log)
- N° de séquence du fichier de journalisation en ligne en cours.
- Informations sur les points de reprise (checkpoints)
- Début et fin des segments d'annulation,
- Informatiions sur l'archivage des fichiers de journalisation
- Informatiions sur les sauvegardes.

Multiplexer le fichier de contrôle

Pour se préserver d'une panne en un point unique affectant le fichier de contrôle, il est conseillé de :

- Multiplexer (« dupliquer ») le fichier de contrôle (jusqu'à 8 fois)
- Mettre chaque copie de ce fichier, sur un disque physique distinct.
- Sauvegarder ce fichier à chaque modification de la structure physique de la base.

Mutiplexage du fichier de contrôle avec le SPFILE

1. Modifier le SPFILE :

```
alter system set control_files =
'path1/ctrl01.ctl',
'path2/ctrl02.ctl' SCOPE=SPFILE;
```

2. Arrêter la base : shutdown immediate

3. Créer des fichiers de contrôle supplémentaires :

cp path1/ctrl01.ctl path2/ctrl02.ctl

4. Démarrer la base : startup

Mutiplexage du fichier de contrôle avec le fichier PFILE

1. Modifier le fichier PFILE :

CONTROL_FILES = path1/ctrl01.ctl, path2/ctrl02.ctl

2. Arrêter la base : shutdown immediate

3. Créer des fichiers de contrôle supplémentaires :

cp path1/ctrl01.ctl path2/ctrl02.ctl

4. Démarrer la base : startup

Informations sur les fichiers de contrôle

Source de l'information	**Ce qu'elle donne**
V$controlfile	Nom et statut de tous les control files associés à l'instance
V$parameter	Statut et emplacement de tous les paramètres
V$controlfile_record_section	Infos sur enregistrements des différentes sections des control files
Show parameters control_files	Nom statut et l'emplacement de tous les control files

Exemple :

Select name from V$controlfile;
Select name, value from V$parameter where name = 'control_files';
Select type, record_size, record_total, records_used, from V$controlfile_record_section where type = 'DATAFILE';

Vues dynamiques dont les infos sont extraites du fichier de contrôle :

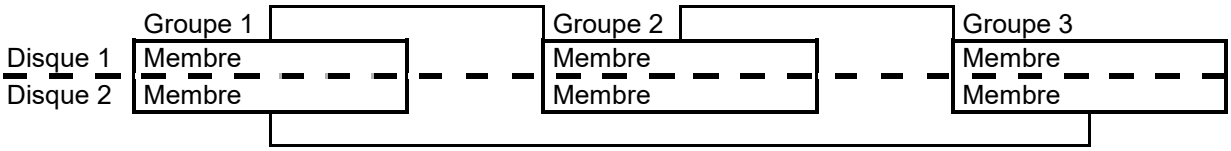
V$backup
V$datafile
V$tempfile
V$tablespace
V$archive
V$log
V$logfile
V$loghist
V$archived_log
V$database

On peut aussi obtenir des infos sur le du fichier de contrôle , par « O.E.M. ».

Utiliser les fichiers de journalisation en ligne

Ils enregistrent toutes les modifications apportées aux données
Ils offrent un mécanisme de récupération,
Ils peuvent être organisés en groupes (2 groupes minimum sont requis).

Structure des fichiers de journalisation en ligne



Groupe de fichiers de journalisation en ligne

- Un groupe est un ensemble de copie identique de journalisation en ligne.
- Le processus PGWR écrit simultanéement les mêmes informations dans tous les fichiers de journalisation en ligne d'un groupe.
- Le serveur oracle a besoin de 2 groupes de fichiers de journalisation en ligne pour garantir un fonctionnement correct des bases de données.

Membre de fichiers de journalisation en ligne

- Chaque fichier de journalisation en ligne d'un groupe est nommé membre (du groupe).
- Les membres d'un groupe portent tous le même n° de séquence de journal et ont tous la même taille. Ce type de n°, permettant d'identifier de manière unique chaque fichier de journalisation en ligne, est attribué lorsque le serveur Oracle écrit dans un groupe de fichiers de journalisation. Le n° en cours est stocké dans le fichier de contrôle et dans l'entête de tous les fichiers de données.

Créer des fichiers de journalisation en ligne initiaux

Les groupes de fichiers de journalisation en ligne (on line) et membres initiaux sont créés en même temps que la base de données.

Les paramètres suivants limitent le nombre de fichiers de journalisation en ligne :

- Le paramètre MAXLOGFILES de la commande CREATE DATABASE définit le nombre max absolu de groupes de fichiers de journalisation en ligne.
- La valeur max et la valeur par défaut du paramètre MAXLOGFILES dépendent du système d'exploitation.
- Le paramètre MAXLOGMEMBERS de la commande CREATE DATABASE définit le nombre max de membres par groupes. La valeur max et la valeur par défaut de ce paramètre dépendent du système d'exploitation.

Mode de fonctionnement des fichiers de journalisation en ligne

Ils sont utilisés de façon cyclique.
Lorsqu'un fichier de ce type est rempli, le processus LGWR passe au groupe de fichiers de journalisation suivant.
On parle alors de changement de fichier de journalisation.
Une opération de point de reprise (checkpoint) se produit également.
Les informations sont écrites dans le fichier de contrôle.

LGWR écrit dans ce groupe :

- Lorsqu'une transaction est validée,
- Lorsqu'un tiers du tampon de journalisation est occupé.
- Lorsque le tampon de journalisation contient plus d' 1 Mo d'enregistrements modifiés

- Avant que le processsus DBWn n'écrive les blocs modifiés du cache de tampon,s de la base de données dans les fichiers modifiés.

Changement de fichier de journalisation

LGWR écrit, de façon séquentielle, dans les fichiers de journalisation en ligne. Lorsqu'un groupe est complet, il écrit alors dans le groupe suivant. Ainsi de suite, jusqu'au retour au 1er groupe.

Points de reprise :

Au cours d'un point de reprise :

Plusieurs tampons « dirty », générés par les les fichiers de journalisation, faisant l'objet d'un point de reprise, sont écrits par DBWn dans les fichiers de données.
Le processus des points de reprise CKPT met à jour le fichier de contrôle pour indiqué qu'il s'est exécuté correctement.

Ce point de reprise se produit dans les cas suivants :
A chaque changement de fichier de journalisation,
Lors de l'arrêt d'une instance, avec l'option NORMAL, TRANSACTIONAL, IMMEDIATE.
Lorsque son exécution est forcée par le paramètre d'initialisation FAST_START_MTTP_TARGET
Lorsque le DBA l'exécute manuellement.
Lorsque la commande ALTER TABLESPACE [OFFLINE NORMAL|READ ONLY| BEGIN BACKUP] entrave l'exécution de points de reprise sur certains fichiers de données.

Des informations sur chaque point de reprise sont enregistrées dans le fichier alert_<sid>.log Si le paramètre d'initialisation LOG_CHECKPOINTS_ALERT = TRUE

Forcer les changements de fichiers de journalisation :

- ALTER SYSTEM SWITCH LOGFILE ;
- ALTER SYSTEM CHECKPOINT ;
- Forcer application points de reprise : FAST_START_MTTR_TARGET = 600 (celui-ci remplace ces 2 paramètres de la V8 : FAST_START_IO_TARGET et LOG_CHECKPOINT_TIMEOUT). Note : 600 = 600 secondes.

Ajouter des groupes de fichiers de journalisation

ALTER DATABASE [database] ADD LOGFILE GROUP integer (filespec, filespec …) [, GROUP integer (filespec, filespec …)] ;

Exemple :
ALTER DATABASE ADD LOGFILE
GROUP 3 ('$HOME/oradata/u01/log3a.rdo', '$HOME/oradata/u01/log3b.rdo') SIZE 1 M ;

Ajouter des membres à des fichiers de journalisation en ligne

```
ALTER DATABASE [database]
ADD LOGFILE MEMBER
    [ 'filename' [REUSE] ,      [ 'filename' [REUSE] ] …
TO { GOUP integer | ('filename' [,'filename'] … )
    }
] …
```

Exemple :
ALTER DATABASE ADD LOGFILE MEMBER
'$HOME/oradata/u01/log1c.rdo' TO GROUP 1,
'$HOME/oradata/u02/log2c.rdo' TO GROUP 2,
'$HOME/oradata/u04/log3c.rdo' TO GROUP 3 ;

Supprimer des groupes de fichiers de journalisation

ALTER DATABASE [database] DROP LOGFILE { GROUP integer | ('filename' [, 'filename' [, 'filename'] …) }
[, GROUP integer ('filename' [, 'filename'] …) }] ….

Exemple: ALTER DATABASE DROP LOGFILE GROUP 3 ;

Supprimer des membres de fichiers de journalisation en ligne

ALTER DATABASE [database]
DROP LOGFILE MEMBER 'filename' , ['filename'] …

Exemple: ALTER DATABASE DROP LOGFILE MEMBER '$HOME/oradata/u04/log3c.rdo' ;

Transférer ou renommer des fichiers de journalisation en ligne

1. ALTER DATABASE CLEAR LOGFILE ;
2. Arrêter la base de données : SHUTDOWN
3. copier les fichiers de journalisation en ligne dans le nouvel emplacement : mv fic1 fic2
4. CONNECT / AS SYSDBA
5. STARTUP MOUNT
6. ALTER DATABASE RENAME FILE 'fic1.rdo' TO 'fic2.rdo' ;
7. ALTER DATABASE OPEN ;
8. [ajouter ou supprimer des members].

Annuler des fichiers de journalisation en ligne

Réinitialiser un fichier de journalisation en ligne :
ALTER DATABSE CLEAR LOGFILE GROUP integer ;
Pour ne pas archiver le fichier de journalisation en ligne endommagé :
ALTER DATABSE CLEAR UNARCHIVED LOGILE GROUP integer ;

Infos sur groupes et membres : vlog, vlogfile

Fichier de journalisation archivés

Archivage en ligne et complet.
L'exécution de la base de données en mode ARCHIVELOG et l'archivage des fichiers de journalisation en ligne présente deux avantages :

1. récupération : la sauvegarde de la BD ainsi que les fichiers de journalisation en ligne et archivés peuvent garantir la récupération de toutes les transactions validées.
2. sauvegarde : celle-ci peut s'effectuer lorsque la base de données est ouverte.

Par défaut une base est ouverte en mode NOARCHIVELOG.

Le paramètre LOG_ARCHIVE_START indique si l'archivage doit automatique (TRUE) ou manuel (FALSE) lors du démarrage de l'instance.
Archivage fait par le processus ARCn.
Archivage réalisé automatiquement par :

- le processus ARCn
- à l'aide d'instructions SQL.

Lorsque les fichiers sont correctement archivés :

- une entrée est générée dans le fichier de contrôle.
- Le nom des fichiers archivés, le n° de séquence du journal, le n° SCN le élévé et le plus faible sont enregistrés.

Les fichiers de journalisation en ligne complets ne peuvent pas être réutilisés :

- Tant qu'un point de reprise (checkpoint) n'a pas eu lieu.
- Tant qu'ils n'ont pas été archivés par ARCn.

Les fichiers archivés ne peuvent être multiplexés.
Ils sont mis à jour par le DBA.

Infos : SELECT archiver FROM v$instance ; => STARTED ou STOPPED

Gestion des tablespaces et fichiers de données (datafiles)

Type de tablespace

Le Tablespace SYSTEM :

- Créé en même temps que la base de données
- Contient le dictionnaire de données
- Contient le segment d'annulation SYSTEM (UNDO …).

Les Tablespaces non SYSTEM :

- Sépare les segments
- Facilite l'administration de l'espace
- Gère la quantité d'espace allouée aux utilisateurs.

CREATE TABLESPACE *tablespace*
[DATAFILE clause]
[MINIMUM EXTENT integer [K | M]]
[BLOCKSIZE integer [K]]
[LOGGING | NOLOGING]
[DEFAULT storage_clause]
[ONLINE | OFFILE]
[PERMANENT | TEMPORARY]
[extent_management_clause]
[segment_management_clause]

avec :
extent_management_clause :
[EXTENT MANAGEMENT [DICTIONNARY | LOCAL [AUTOALLOCATE | UNIFORM [SIZE integer [K | M]]]]]

Exemple de la commande CREATE TABLESPACE :
CREATE TABLESPACE user_data

DATAFILE ‘/u01/oradata/userdata01.dbf’ SIZE 100M ;

Tablespace géré localement :

- Extents libres gérés dans le tablespace
- Un bitmap est utilisé pour enregistrer les extents libres,
- Chaque bit correspond à un bloc ou à un groupe de blocs
- Le valeurs des bits inidique si ceux-ci sont utilisés ou disponibles (libres).
- La contention au niveau de tables du dictionnaire de données est réduite.
- Aucune annulation n’est générée lors de l’allocation ou de la libération d’espace.
- Aucune fusion n’est requise.

Exemple :
CREATE TABLESPACE userdata
DATAFILE ‘/u01/oradata/userdata01.dbf’ SIZE 500M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K ;

Tablespace géré au moyen du dictionnaire de données :

A ne pas utiliser …
Les extents libres sont gérés par le dictionnaire de données.
Les tables appropriées sont mises à jour lorsque les extents sont allouées ou libérées.

Exemple :
CREATE TABLESPACE userdata
DATAFILE ‘/u01/oradata/userdata01.dbf’ SIZE 500M
EXTENT MANAGEMENT DICTIONNARY
DEFAULT STORAGE (initial 1 M NEXT 1M PCTINCREASE 0);

Migration d’un tablespace SYSTEM géré au moyen du dictionnaire

DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL(‘SYSTEM’);

Tablespace d’annulation (undo) :

- Permet de stocker des segments d’annulation.
- Ne peut contenir aucun autre objet
- Les extents sont gérés localement.
- Ne peut être utilisés qu’avec les clauses DATAFILE et EXTENT.

CREATE UNDO TABLESPACE *tablespace* [DATAFILE clause] ;

Exemple :
CREATE UNDO TABLESPACE undo1
DATAFILE ‘/u01/oradata/undo01.dbf’ SIZE 40M ;

Tablespaces TEMPORARY :

- Sont utilisées pour les opérations de tri.
- Peuvent être partagés par plusieurs utilisateurs.
- Ne peuvent pas contenir d’objets permanents.
- La gestion locale des extents est recommandée.

CREATE TEMPORARY TABLESPACE temp

DATAFILE ‘/u01/oradata/temp01.dbf’ SIZE 20M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4M ;

Tablespaces TEMPORARY par défaut :

Les tablespaces TEMPORARY par défaut ne peuvent pas être :

- Supprimé tant qu’une nouveau tablespace par défaut n’est pas disponible.
- Mis hors ligne
- Transformés en tablespace permanents.

Créé à l’aide de :
CREATE DATABASE
ALTER DATABASE

CREATE DATABASE
….
DEFAULT TEMPORARY TABLESPACE temp
TEMPFILE ‘$HOME/oradata/u03/temp01.dbf’ SIZE 4M ;

ALTER DATABASE
DEFAULT TEMPORARY TABLESPACE default_temp2 ;

La vue DATABASE_PROPERTIES permet de retrouver le tablespace TEMPORARY par défaut :

Select * from DATABASE_PROPERTIES ;

Tablespaces en lecture seule :

La mise en lecture seule permet d'empêcher les opérations d'écriture dans ces fichiers de données. Ces derniers peuvent alors résider sur des supports de données en lecture seule, tels que CD-ROM, unité non réinscriptibles … Cela éviter d'avoir à sauver de grandes sections statiques d'une base de données.

ALTER TABLESPACE userdata READ ONLY ;

Mettre une tablespace hors ligne :

Dès qu'elle est hors ligne, les données ne sont plus disponibles.
Tablespaces ne pouvant pas être mis hors ligne :

- Tablespace SYSTEM
- Tablespaces contenant des segments d'annulation actifs.
- Tablepace TEMPORARY par defaut.

ALTER TABLESPACE userdata OFFLINE :
ALTER TABLESPACE userdata ONLINE :

Modifier les parameters de stockage d'un tablespace :

Valable uniquement si le tablespace est géré dans le dictionnaire. Inutile en gestion locale. (les paramètres de stockage des tablespaces gérés localement ne peuvent être modifiés).

ALTER TABLESPACE tablespace
[MINIMUM EXTENT integer [K | M]]
[DEFAULT storage_clause] ;

Exemples :
ALTER TABLESPACE userdata MINIMUM EXTENT 2 M;
ALTER TABLESPACE userdata DEFAULT STORAGE (INITIAL 2 M NEXT 2M MAXEXTENTS 999);

Redimensionner un tablespace

En modifiant la taille d'un fichier de données :
Automatiquement à l'aide d'AUTOEXTEND
Manuellement à l'aide d'ALTER TABLESPACE
En ajoutant un fichier de données, à l'aide d'ALTER TABLESPACE.

Exemples :
CREATE TABLESPACE userdata02
DATAFILE '/u01/oradata/userdata02.dbf' SIZE 5M
AUTOEXTEND ON NEXT 2M MAXSIZE 200M;
ALTER DATABASE DATAFILE '/u01/oradata/userdata02.dbf' AUTOEXTEND ON NEXT 2M;
ALTER DATABASE DATAFILE '/u01/oradata/userdata02.dbf' RESIZE 5M;

Activer l'extension automatique des fichiers de données :

Par :
CREATE DATABASE …
CREATE TABLESPACE …
ALTER TABLESPACE … ADD DATAFILE ..

ALTER DATABASE DATAFILE filespec [autoextend_clause]
Autoextend_clause :== [AUTOEXTEND [OFF|ON[NEXT integer [K|M]]
[MAXSIZE UNLIMITED | integer [K|M]] } } ;

Exemples :
CREATE TABLESPACE userdata02
DATAFILE '/u01/oradata/userdata02.dbf' SIZE 5M
AUTOEXTEND ON NEXT 2M MAXSIZE 200M;

L'interrogation de la vue DBA_DATAF8LES permet de déterminer si l'AUTOEXTEND est activé.

Méthode de déplacement des fichiers de données :

ALTER TABLESPACE :
- Le tablespace doit être hors ligne,
- Les fichiers de données cible doivent exister.
- Mettre le tablespace OFFLINE, déplacer le fichier, puis faire l'ALTER :

ALTER TABLESPACE userdata RENAME
DATAFILE 'filename' ['filename' …
TO 'filename' ['filename' … ;

Exemples :
ALTER TABLESPACE userdata RENAME
DATAFILE '/u01/oradata/userdata01.dbf'
TO '/u02/oradata/userdata01.dbf' ;

Déplacement des fichiers de données du tablespace SYSTEM :

ALTER DATABASE RENAME FILE '/u01/oradata/system01.dbf'
TO '/u03/oradata/system01.dbf' ;

1. Arrêter la base de données.
2. utiliser les commandes du S.O. pour déplacer le fichier.
3. Monter la base de données (STARTUP MOUNT)
4. Exécuter la commande ALTER DATABASE RENAME FILE ;
5. Ouvrer la base de données (STARTUP OPEN) ;

Supprimer des tablespaces :

Un tablespace ne peut être supprimé s'il :
- S'agit du tablespace SYSTEM
- S'il possède des segments actifs.

INCLUDING CONTENTS supprime les segments,
INCLUDING CONTENTS AND DATAFILES supprime les fichiers de données.
CASCADE CONSTRAINTS supprime les contraintes d'intégrité référentielle.

DROP TABLESPACE userdata
INCLUDING CONTENTS AND DATAFILE ;

Informations sur les tablespaces :

- Informations sur les tablespaces :
- DBA_TABLESPACES
- V$TABLESPACES
- Informations sur les fichiers de données :
- DBA_DATA_FILES
- V$DATAFILES
- Informations sur les fichiers temporaires :
- DBA_TEMP_FILES
- V$TEMPFILE

==========

==========

Gérer les données d'annulation (en V9)

Deux méthode de gestion des données d'annulation sont disponibles :

- Gestion automatique,
- Gestion manuelle.

En V9, le terme « undo » remplace le terme « rollback ».
On le traduit par « annulation », en français.

Un segment d'annulation permet d'enregistrer l'ancienne valeur (données d'annulation) lorsqu'un processus modifie les données d'une base.
Il enregistre l'emplacement des données et leur valeur avant modification.

Les données d'annulation d'une transaction séquentielle sont stockées dans un seul segment d'annulation.
Plusieurs transactions simultanées peuvent écrire des données dans un même segment d'annulation.

Fonction des segments d'annulation :

1. Annulation d'une transaction :
2. Récupération d'une transaction (avec la fonction journal rdo log).
3. Cohérence en lecture :

Cohérence en lecture :

Le serveur Oracle permet à une instruction de voir les données de manière cohérente à un instant donné, même si celles-ci sont modifiées par d'autres transactions.
Lorsque le serveur Oracle lance l'exécution d'une instruction SELECT, il détermine le n° SCN (System Change Number) en cours et s'assure que l'instruction ne traite pas les modifications non validées, avant ce n°.
Une instruction SQL est toujours cohérente en lecture.

On peut demander la cohérence en lecture d'une transaction en lecture seule, par la commande en début de celle-ci :
SET TRANSACTION READ ONLY ;

Demande de la cohérence en lecture d'une transaction LMD :
SET TRANSACTION ISOLATION LEVEL SERILIZABLE ;

Types de segments d'annulation :

- Le segment SYSTEM est utilisé pour les objets du tablespace SYSTEM.
- Le segment non SYSTEM est utilisé pour les objets d'autres tablespaces.
- ➢ Le mode automatique nécessite un tablespace UNDO.
- ➢ Mode manuel :
  - Privé : acquis par une instance spécifique,
  - Public : : acquis par une instance quelconque.
- Les segments différés sont utilisés lorsque des tablespaces sont mis hors ligne immédiatement, temporairement ou à des fins de récupération.

Les noms de ces segments sont de la forme suivante : _SYSSMUn$ avec n un nombre.

Conseils :

a) configurer les 2 paramètres suivant dans le fichier PFILE (d'initialisation) :

UNDO_MANAGEMENT
UNDO_TABLESPACE

Si la base ne contient qu'un tablespace d'annulation (UNDO) et si la valeur du paramètre UNDo_MANAGEMENT est AUTO, le paramètre UNDO_TABLESPACE est facultatif, car le serveur Oracle choisira automatiquement le tablespace d'annulation.

b) créer au moins un tablespace d'annulation.

c) allouer un tablespace d'annulation (UNDO) en prévoyant suffisamment grand pour la charge globale de l'instance.

```
CREATE DATABASE db01
…
UNDO TABLESPACE undo1
DATAFILE '/u01/oradata/undo1db01.dbf' SIZE 20M
AUTOEXTEND ON ;

CREATE UNDO TABLESPACE undo1
DATAFILE '/u01/oradata/undo1db01.dbf' SIZE 20M ;

ALTER TABLESPACE undo1
ADD DATAFILE '/u01/oradata/undo1db01.dbf' SIZE 20M
AUTOEXTEND ON ;

ALTER TABLESPACE tablespace
[ ADD DATAFILE … |
RENAME … |
DATAFILE [ONLINE | OFFLINE ] |
BEGIN BACKUP |
END BACKUP ] ;
```

Changer de tablespace d'annulation :

- Vous pouvez passer d'un tablespace d'annulation à un autre.
- Vous ne pouvez affecter à une base de données, qu'un seul tablespace d'annulation.
- Une instance peut contenir plusieurs tablespaces d'annulation, mais un seul est actif.
- La commande suivante personne changer dynamiquement de tablespace d'annulation : ALTER SYSTEM SET UNDO_TABLESPACE=UNDOTBS ;

Supprimer un tablespace d'annulation

1) ALTER SYSTEM SET undo_tablespace = UNDOTBS2 ;

2) lancer le SELECT :

```
SELECT a.name, b.status
 FROM    v$rollname a, v$rollstat b
WHERE a.name IN (SELECT segment_name
FROM  dba_segments )
AND a.usn = b.usn ;
  2    3    4    5
NAME                           STATUS
------------------------------ ---------------
RB1                              ONLINE
RB2                              ONLINE
SYSTEM                           ONLINE
```

Si vous avez quelque chose comme :

```
NAME                           STATUS
------------------------------ ---------------
_SYSSMU4$                      PENDING OFFLINE
```

Un segment d'annulatiin, possédant le satut « PENDING OFFLINE » a encore des transactions en cours.

3) Si le SELECT précédent ne ramène aucune ligne, vous pouvez lancer :
DROP TABLESPACE UNDOTBS2 ;

Le serveur peut référencer UNDOTBS après être passé à un autre tablespace d'annulation pour garantir la cohérence en lecture des interrogations. Les interrogations qui s'adressent au tablespace UNDOTBS lors que celui-ci n'est plus disponible pour assurer une lecture cohérente aboutissent à l'erreur :
ORA-01555 : snapshot too old : rollback segment number *nnn* with name *XXXX* too small

Autres paramètres de gestion automatique des annulations :
UNDO_SUPPRESS_ERRORS :
Lorsqu'il possède la valeur TRUE, ce paramètre supprime les erreurs qui surviennent lors de tentatives d'exécution d'opération manuelles en mode AUTO.
UNDO_RETENTION :
Ce paramètre contrôle le volume de données d'annulation à conserver, pour une lecture cohérente.

Script (en V9) pour la calcul du dimensionnement d'un tablespace d'annulation :

```
SELECT (UR * (UPS * DBS)) + (DBS * 24) Bytes
FROM (SELECT value AS UR
FROM v$parameter
WHERE name = 'undo_retention'),
```

```
(SELECT SUM(undoblks) / SUM ((end_time - begin_time) * 86400 ) AS UPS
FROM v$undostat ),
(SELECT value AS DBS
FROM v$parameter
WHERE name = 'db_block_size');
```

Affichage de toutes les informations sur les rollback segments en V8

Toutes infos sur les rollback segments en V8

```
SELECT segment_name, tablespace_name, status
FROM   sys.dba_rollback_segs;

SELECT segment_name, tablespace_name, bytes, blocks, extents
FROM     sys.dba_segments
WHERE segment_type = ‘ROLLBACK’ ;
```

Rollback segments OFFLINE en V8

```
SELECT name, xacts "ACTIVES TRANSACTIONS"
FROM    v$rollname, v$rollstat
WHERE status = 'PENDING OFFLINE'
AND       v$rollname.usn = v$rollstat.usn ;
```

Affichage de tous les rollback segments deferrés en v8

```
SELECT segment_name, tablespace_name, owner
FROM   sys.dba_rollback_segs;

SELECT segment_name, segment_type, tablespace_name
FROM   sys.dba_segments
WHERE  segment_type = 'DEFERRED ROLLBACK';
```

Suppression d'un tablespace UNDO décrite dans le cours Oracle V9

Pour supprimer un tablespace UNDO (rollback), il faudrait suivre la procédure suivante :

1) Créer un second tablespace UNDO, par exemple : TBSGENIO2

```
CREATE UNDO TABLESPACE TBSGENIO2
DATAFILE ‘/ora_data_tbl/geniodbo/rbsGENIO2.ora’ SIZE nnnM ;
```
[ou /genio_RBS2/geniodbo/rbsGENIO.ora]

La taille "nnn" de ce table space est obtenue par le script suivant (fonctionnant en V9) :

```
SELECT (UR * (UPS * DBS)) + (DBS * 24) Bytes
FROM (SELECT value AS UR
FROM v$parameter
WHERE name = 'undo_retention'),
(SELECT SUM(undoblks) / SUM ((end_time - begin_time) * 86400 ) AS UPS
FROM v$undostat ),
(SELECT value AS DBS
FROM v$parameter
WHERE name = 'db_block_size');
```

2) il faut ensuite rendre actif le 2° tablespace UNDO (donc basculer dessus) :

```
ALTER SYSTEM SET undo_tablespace = TBSGENIO2 ;
```

3) lancer :
```
SELECT a.name, b.status
  FROM   v$rollname a, v$rollstat b
 WHERE a.name IN (SELECT segment_name
                    FROM  dba_segments )
  AND a.usn = b.usn ;
```

Si on a quelque chose comme :

```
NAME                         STATUS
---------------------------- ---------------
_SYSSMU4$                    PENDING OFFLINE
```

Un segment d'annulation, possédant le satut « PENDING OFFLINE » a encore des transactions en cours.
Donc, on ne peut pas faire cette opération.
Sinon, on peut continuer à faire cette suppression.

4) Si le SELECT précédent ne ramène aucune ligne, on peut alors lancer (1) la suppression du 1er tablespace :

```
DROP TABLESPACE UNDOTBS1 ;
```

Note : Dans certains cas, il faudrait ensuite faire l'opération inverse, pour avoir [de nouveau] le tablespace de départ (ici RBSGENIO), mais avec un datafile dans /rbs_GENIO/geniodbo ... d'une taille plus petite.

Création ou modification d'un rollback segment en V8

1) créer le rollback segment :

```
CREATE PUBLIC ROLLBACK SEGMENT rb1 TABLEPACE tablespace ;
```

2) mettre le rollback segment ONLINE :
mettre le paramètre ROLLBACK_SEGMENTS dans le fichier PFILE, comme ci-après :

```
ROLLBACK_SEGMENTS = (RB1, RB2) ;
```

3) spécifier les paramètres du « rollback segment » avec O.E.M. ou avec l'ordre SQL :

```
CREATE PUBLIC ROLLBACK SEGMENT data1_rs
TABLESPACE tablespace (INITIAL 50K NEXT 50K OPTIMAL 750K MINEXTENTS 15
MAXEXTENTS 100 );
```

4) changer le paramétrage du « rollback segment » avec O.E.M. ou avec l'ordre SQL :

```
ALTER PUBLIC ROLLBACK SEGMENT data1_rs STORAGE (MAXEXTENTS 120)) ;
```

5) Ajuster (décroître) la taille du « rollback segment » :

```
ALTER ROLLBACK SEGMENT rbs1 SHRINK TO 100K ;
```

6) Mettre le « rollback segment » OFFLINE ou ONLINE :

```
ALTER ROLLBACK SEGMENT rbs1 OFFLINE ;
```

7) Assigner une transaction à un « rollback segment » particulier :

```
SET TRANSACTION USE ROLLBACK SEGMENT rbs2 ;
```

Suppression d'un rollback segment en V8

On peut supprimer un rollback segments quand les extents d'un segment deviennent trop fragmentés sur le disque, ou si le segment doit être délocalisés dans un tablespace différent.

Avant de supprimer un ROLLBACK SEGMENT, s'assurer que le status du rollback segment est OFFLINE.
Si le rollback segment, est actuellement ONLINE, PARTLY AVAILABLE, NEEDS RECOVERY ou INVALID, on ne peut le supprimer.
Si le status est INVALID, le segment a déjà été supprimé.
Pour supprimer le rollback segment, on doit avoir le privilège DROP ROLLBACK SEGMENT.

On peut le supprimer par la commande : DROP PUBLIC ROLLBACK SEGMENT RB1 ;
(dans cette commande, il est important de préciser s'il est privé ou public).
Ensuite, il faut mettre à jour la ligne « ROLLBACK_SEGMENTS » dans le fichier PFILE.
Après qu'il a été supprimé, son status change en 'INVALID'.
Il n'apparaîtra plus dans la vue DBA_ROLLBACK_SEGS ;

4.2 Affichage de toutes les informations sur les rollback segments en V8

Toutes infos sur les rollback segments en V8

```
SELECT segment_name, tablespace_name, status
FROM   sys.dba_rollback_segs ;

SELECT segment_name, tablespace_name, bytes, blocks, extents
FROM     sys.dba_segments
WHERE segment_type = 'ROLLBACK' ;
```

Rollback segments OFFLINE en V8

```
SELECT name, xacts "ACTIVES TRANSACTIONS"
FROM    v$rollname, v$rollstat
WHERE status = 'PENDING OFFLINE'
AND       v$rollname.usn = v$rollstat.usn ;
```

Affichage de tous les rollback segments deferrés en v8

```
SELECT segment_name, tablespace_name, owner
FROM   sys.dba_rollback_segs;

SELECT segment_name, segment_type, tablespace_name
FROM   sys.dba_segments
WHERE  segment_type = 'DEFERRED ROLLBACK';
```

Redimensionnement du tablespace utilisés par des rollback segments

Ici on prend un exemple.

0) être connecté « SYSTEM » :

1) création d'un second tablespace pour les rollback segments : RBSGENIO2

```
CREATE TABLESPACE RBSGENIO2
```

```
DATAFILE '/ora_data_tbl/BAANdbo/rbsGENIO2.ora' SIZE 200M REUSE
AUTOEXTEND ON NEXT 50M  MAXSIZE 500M;
```

2) création de 2 nouveaux rollback segments RBS3 et RBS4 :

```
-- Création de segments rollback
CREATE ROLLBACK SEGMENT RBS3
TABLESPACE RBSGENIO2 storage (INITIAL 20M NEXT 20M MINEXTENTS 20 MAXEXTENTS 40 ) ;
CREATE ROLLBACK SEGMENT RBS4
TABLESPACE RBSGENIO2 storage (INITIAL 20M NEXT 20M MINEXTENTS 20 MAXEXTENTS 40 ) ;
```

3) rendre ces 2 ROLLBACK RBS3 et RBS4 "ONLINE" :

```
alter rollback segment RBS3 online;
alter rollback segment RBS4 online;
```

4) vérification qu'il n'y a pas de transaction actives, grâce à cet ordre SQL ci-après :

```
SELECT  name, xacts "ACTIVES TRANSACTIONS"
FROM    v$rollname, v$rollstat
WHERE   status = 'PENDING OFFLINE'
AND     v$rollname.usn = v$rollstat.usn ;
```

On doit avoir : "no rows selected"

5) rendre les 2 rollback segments RBS1 et RBS2 "OFFLINE" :

```
alter rollback segment RB1 offline;
alter rollback segment RB2 offline;
```

6) supprimer ces 2 rollback segments RBS1 et RBS2 :

```
DROP ROLLBACK SEGMENT RB1 ;
DROP ROLLBACK SEGMENT RB2 ;
```

7) mettre OFFLINE et supprime la tablespace RBSGENIO :

```
ALTER TABLESPACE RBSGENIO OFFLINE NORMAL ;
DROP TABLESPACE RBSGENIO INCLUDING CONTENTS;
```

7bis) supprimer le datafile associé :

```
rm /genio_RBS/geniodbo/rbsGENIO.ora
```

8) recréer le 1er tablespace pour les Rollback segments TBSGENIO :

```
CREATE TABLESPACE RBSGENIO
DATAFILE '/genio_RBS/geniodbo/rbsGENIO.ora' SIZE 200M REUSE
AUTOEXTEND ON NEXT 50M  MAXSIZE 500M;
```

9) recréer les 2 rollback segments RB1 et RB1 :

```
CREATE ROLLBACK SEGMENT RB1
TABLESPACE RBSGENIO storage (INITIAL 20M NEXT 20M MINEXTENTS 20 MAXEXTENTS 40 ) ;
CREATE ROLLBACK SEGMENT RB2
TABLESPACE RBSGENIO storage (INITIAL 20M NEXT 20M MINEXTENTS 20 MAXEXTENTS 40 ) ;
```

10) rendre ces 2 ROLLBACK RB1 et RB2 "ONLINE" :

```
alter rollback segment RB1 online;
alter rollback segment RB2 online;
```

11) rendre les 2 rollback segments RBS3 et RBS4 "OFFLINE" :

```
alter rollback segment RBS3 offline;
alter rollback segment RBS4 offline;
```

12) supprimer les 2 rollback segments RBS3 et RBS4 :

```
DROP ROLLBACK SEGMENT RBS3 ;
DROP ROLLBACK SEGMENT RBS4 ;
```

13) mettre OFFLINE et supprimer la tablespace RBSGENIO2 :

```
ALTER TABLESPACE RBSGENIO2 OFFLINE NORMAL ;
DROP TABLESPACE RBSGENIO2 INCLUDING CONTENTS;
```

14bis) puis suppression di datafile associé :

```
rm /ora_data_tbl/BAANdbo/rbsGENIO2.ora
```

Gestion automatique des annulations : quota d'annulation

- Les transactions longues ou incorrectes peuvent utiliser des ressources considérables.
- Les quotas d'annulation permettent de regrouper les utilisateurs et de leur affecter un espace d'annulation maximal défini.
- La directive UNDO_POOL de Ressource Manager définit l'espace alloué à un groupe de ressources.
- Lorsqu'un groupe dépasse la limite définie, il ne peut plus exécuter de nouvelle transaction, jusqu'à ce que les transactions en cours se terminent ou échouent, libérant ainsi de l'espace d'annulation.

Obtenir des informations sur les segments d'annulation

Vue DBA_ROLLBACK_SEGS
Vue dynamique des performances :
V$ROLLNAME
V$ROLLSTAT
V$UNDOSTAT
V$CESSION
V$TRANSACTION

```
SELECT segment_name, owner, tablespace_name, status
FROM   sys.dba_rollback_segs ;
```

Statistiques sur les segments d'annulation en cours d'utilisation :

```
SELECT n.name,  s.extents, s.rssize, s.hwmsize, s.xacts, s.status
FROM    v$rollname n, v$rollstat s
WHERE n.usn = s.usn ;
```

Vérifier que transactions en cours utilisent un segment d'annulation :

SELECT s.username, t.xidusn, t.ubafil, t.ubablk, t.used_ublk
FROM v$session s, v$transaction t
WHERE s.saddr) t.ses_addr ;

Structure de stockage et relation

Base de données PROD																									
TABLES SPACES SYSTEM						USER_DATA										RBS				TEMP					
FICHIERS DE DONNEES DISK1/SYS1.dbf						DISK2/ USER1.dbf				DISK3/ USER2.dbf						DISK1/ UNDO01.dbf				DISK1/ TEMP.dbf					
SEGMENTS D.D. Table Seg données		D.D. index Seg index		Seg RB		S_DEPT Seg Données		S_EMP Seg données		S_dept (suite)		S_emp First_name Index Seg index				Rbs1 Seg RB	Rbs2 Seg RB	Rbs1 suite Seg RB	Rbs2 suite Seg RB	TEMP Seg temp					
EXT 1	EN 2	TS 1	2	1	2	1		1	2	2		1		dispo		1	1	2	2	1					
Blocs	de	don	n	é	e	s	Ora	cle																	

Avec :
D.D. : datafiles
1 segment : 1 objet de la base prenant de la place dans la base de données
(1 vue n'est pas un segment).
1 segment = 1 ensemble d'extends
1 extend = 1 ensemble de blocs contigus
RB : Rollback

Table :

Les données sont en général stockées dans une table.
Un segment de table contient les données d'une table, ni incluses dans un cluster, ni partitionnées.
Toutes les données d'un segment de table doivent être stockées dans un seul tablespace.

Partition de table :

Les données d'une table peuvent être stockées dans plusieurs partitions, chacune se trouvant dans un tablespace différent.
Lorsqu'une table est partionnée, chacune de ses partitions constitue un segment pouvant être contrôlé de manière indépendante à l'aide des paramètres de stockage.
L'utilisation de ce type de segment requière l'option de partionnement d'Oracle 9i Enterprise Edition.

Cluster :
A l'instar d'une table, un cluster est un type de segment de données. Les lignes d'un cluster sont stockées en fonction des valeurs des colonnes de clé. Un cluster peut contenir un ou plusieurs tables

Index :
Toutes les entrées d'un index spécifique sont stockés dans le même segment d'index.

Si une table a n indexes, n segments d'indexes sont utilisés.
Le segment d'index permet de recherche l'emplacement des lignes dans une table, en fonction d'une clé spécifique.

Table organisée en index (I.O.T.) :
Les données d'une table organisée en index sont stockées dans l'index en fonction de la valeur de clé.
Une table organisée en index ne nécessite pas de recherche, car toutes les données peuvent être extraites directement à partir de l'arborescence d'index.

Partition d'index :
Un indexe peut être partitionné et réparti dans plusieurs tablespaces.
Chaque partition correspond à un segment et ne peut pas occuper plusieurs tablespaces.
Un index partitionné est utilisé principalement pour réduire la contention en répartissant les entrées/sorties d'index.
L'utilisation de ce type de segment requiert l'option de partitionnement d'Oracle 9i Enterprise Edition.

Segment d'annulation :
Un segment d'annulation est utilisé par une transaction qui apporte des modifications à une base.
Avant toute modification des blocs de données ou d'index, l'ancienne valeur est stockée dans le segment d'annulation pour permettre à l'utilisateur d'annuler les modifications, s'il le souhaite.

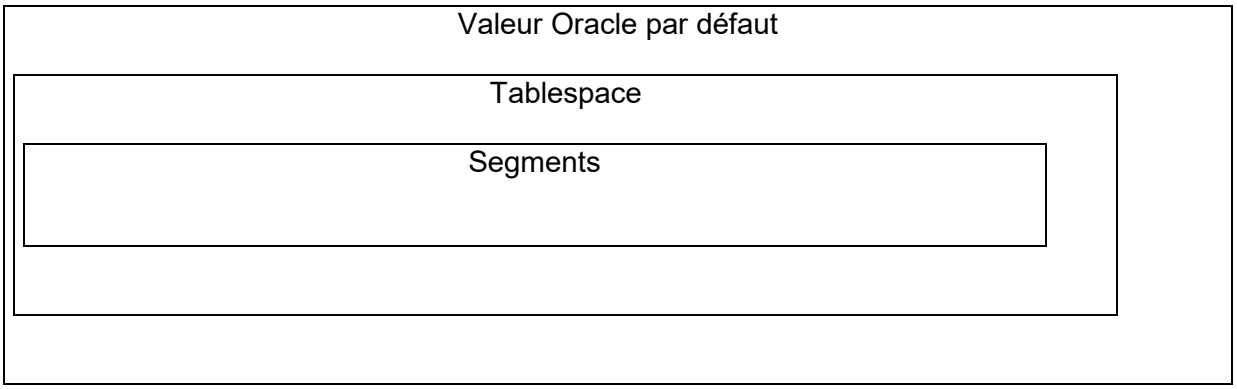
Segments temporaires :
Lorsque l'utilisateur exécute des commandes telles que CREATE INDEX, SELECT DISTINCT et SELECT GROUP BY, le serveur Oracle tente d'effectuer les tris dans la mémoire.
Lorsqu'un tri nécessite d'avantage d'espace disponible que n'en contient la mémoire, des résultats intermédiares sont écrits sur le disque dans des segments temporaires.

Segments LOB (large objects) :
Vous pouvez utiliser une ou plusieurs colonnes d'une table pour stocker des objets LOG (Large Objects), tells que des documents texte, des images ou des données vidéo.
Si les valeurs de ces colonnes sont volumineuses, le serveur oracle les enregistre dans des segments distincts appelés segments LOB. La table contient uniquement un pointeur vers l'emplacement des données LOG correspondantes.

Tables imbriquées (nested table) :
Une colonne de table peut être constituée d'une table définie par l'utilisateur contenant, par exemple, les articles d'une commande. Dans ce cas, la table interne, appelée table imbriquée, est stockée sous la forme de segment distinct.

Segment de bootstrap :
Un segment de bootstrap, appelé également segment de cache, est créé par le script sql.bsq en même temps que la base de données. Ce segment permet d'initialiser le cache du dictionnaire de données lorsqu'une instance ouvre la base.
Il ne peut pas faire l'objet d'une interrogation, ni d'une mise à jour et ne nécessite aucune opération de maintenance de la part de l'administrateur de la base de données.

Priorité des clauses de storage



Paramètres des clauses de stockage

Une clause de stockage (storage) peut être définie au niveau du segment afin de contrôler le mode d'allocation des extents à un segment.

- Tout paramètre de stockage défini au niveau du segment annule l'option correspondante définie a niveau du tablespace, sauf pour les paramètres de tablespace MINIMUM EXTENT et UNIFORM SIZE.
- Lorsque des paramètres de stockage ne sont pas définis explicitement au niveau du segment, les paramètres défini au niveau du tablespace sont utilisés par défaut.
- Lorsque les paramètres de stockage ne sont pas définis explicitement au niveau du tablespace, les valeurs système par défaut du serveur Oracle sont utilisées.

Autres considérations :

- Lorsque vous modifiez des paramètres de stockage, les nouvelles valeurs s'appliquent uniquement aux extents non encore alloués.
- Certains paramètres ne peuvent pas être définis au niveau du tablespace. Ils doivent être indiqués au niveau du segment uniquement.
- Lorsque vous définissez une taille minimum d'extent pour le tablespace, cette taille s'applique à tous les extents allouées ultérieurement à des segments du tablespace.

Allocation et libération d'extents

Un extent est un ensemble de blocs contigus d'espace utilisé par un segment dans un tablespace.
Il est alloué lorsqu'un segment est crée, étendu ou modifié.
Il est libéré lorsqu'un segment est supprimé, modifié, vidé.

Bloc de base de données

Unité minimale d'E/S.
Comprend un ou plusieurs blocs du système d'exploitation.
Est défini à la création du tablespace.
« DB_BLOCk_SIZE » détermine la taille du bloc par défaut.

Taille de blocs Oracle = 1 bloc du SO (bloc stantard) jusqu'à 4 blocs du SO (bloc non standard).
Egal à une puissance de 2, comprise entre 2 et 32 Ko.

Taille de bloc stantard

Elle est définie à la création de la base, à l'aide du paramètre DB_BLOCK_SIZE.
Pour la modifier, il est nécessaire de recréer la base.
Elle est utilisée par les tablespaces SYSTEM et TEMPORARY.
DB_CACHE_SIZE définit la taille du cache du tampons DEFAULT (sga) pour une taille de bloc standard.

- Taille minimale = un granule (4 Mo ou 16 Mo),
- Valeur par défaut = 48 Mo.

DB_BLOCK_SIZE a généralement une valeur de 4 Ko ou 8 Ko.
Si aucune valeur n'est indiquée, la taille de bloc par défaut est propre au système d'exploitation et généralement parfaitement adaptée.
(Le paramètre DB_CACHE_SIZE remplace le paramètre DB_BLOCk_BUFFER défini à la V8 et les versions précédentes).

DB_nK_CACHE_SIZE : parameters à definer si l'on veut plusieurs tailles du cache des tampons (par défaut, leur valeur = 0).
Avec n : 2, 4, 8, 16, 32, 64 …
Créer des tablespaces de taille de bloc non standard m

CREATE TABLESPACE tablespace DATAFILE 'filename' SIZE nnn[K|M]
BLOCKSIZE n[K|M] … ;

Select tablespace_name, block_size … from dba_tablespaces … ;

Règles relatives à l'utilisation de plusieurs tailles de bloc

- Toutes les partitions d'un objet doivent résider dans des tablespaces ayant la même taille de bloc.
- Tous les tablespaces temporaires, y compris les tablespaces permanents utilisés comme tablespaces temporaires par défaut, doivent utiliser la taille de bloc standard.
- Les segments de débordement d'une table organisée en index et segments LOB en dehors de la ligne peuvent être stockés dans un tablespace ayant une taille de bloc différente de celle de table de base.

Un bloc de données oracle contient : 1) un entête de bloc, 2) un espace de données, 3) un espace libre.

Paramètres de contrôle de la simultanéité d'accès aux données

INITRANS : définit le nombre initial d'espaces de transaction créés dans les blocs d'index ou dans un bloc de données.
Il garantit un niveau minimal de simultanéité d'accès aux données. Sa valeur par défaut est 1 pour les segments de données et 2 pour les segments d'index.
SI vous lui affectez la valeur 3, il permet à 3 transactions au moins de modifier simultanément le bloc.

MAXTRANS : définit le nombre maximal d'espaces de transaction créés dans les blocs d'index ou dans un bloc de données.
Valeur par défaut 255.
Il définit un niveau maximal de simultanéité de transactions pouvant modifier un bloc de données ou un bloc d'index..

PCTFREE : % d'espace réservé à l'augmentation de la taille du bloc, résultant des mise à jour des lignes. La valeur par défaut de ce paramètre est de 10 %.

PCTUSED : % d'espace minimal d'espace utilisé que le serveur Oracle tente de conserver pour chaque bloc de données de la table. Un bloc est remplacé dans la liste de blocs libres (free list) lorsque la quantité d'espace utilisé devient inférieure à la valeur du paramètre PCTUSED.

FREELISTS : voir le cours Oracle 9i, tuning.

Méthode de gestion des blocs de données

- Gestion automatique de l'espace des segments.
- Gestion manuelle.

Gestion automatique de l'espace des segments.

Le suivi de la quantité d'espace libre et utilisé dans les segments s'effectue à l'aide de bitmap et de listes de blocs libres.
Cette méthode facilite la gestion, permet une meilleure utilisation de l'espace, améliore les performances des opérations INSERT simultanées.

Les segments bitmap contiennent un bitmap qui indique le statut de chaque bloc dans le segment par rapport à son espace disponible.

La représentation des blocs est contenue dans un ensemble de blocs particulier, appelés « blocs bitmap » (BMB).

Lorsque vous insérez une ligne, le serveur recherche dans la représentation des blocs, un bloc disposant d'un espace suffisant.
Lorsque la quantité d'espace disponible d'un bloc varie, le nouveau statut de ce dernier est indiqué dans le bitmap.

Configurer la gestion automatique de l'espace de segment

```
CREATE TABLESPACE dataxxx
DATAFILE '/path/datafileN.dbf' SIZE nnnM
EXTENT MANAGEMENT LOCAL UNIFORM SIZE nnnM
SEGMENT SPACE MANAGEMENT AUTO ;
```

Une fois qu'un tablespace est créé, les spécifications s'appliquent à tous les segments créés dans le tablespace.

La clause « SEGMENT SPACE MANAGEMENT AUTO » de la commande CREATE TABLESPACE permet de définir des segments bitmap, qui ne pourront pas être modifiés.
Les paramètres PCTUSED, FREELIST, FREELIST GROUP définis seront ignorés.
Les tables normales, les index, les tables organisées en index (IOT) et les objets LOB (large objects) constituent des segments pouvant être gérés par des bitmap.

Gestion manuelle des blocs de données (la seule possible avec les version < à la V9)

Configuration manuelle des blocs de données à l'aide de PCTFREE, PCTUSED, FREELIST.

Obtenir des informations sur le stockage :

DBA_EXTENTS
DBA_SEGMENTS
DBA_TABLESPACE
DBA_DATA_FILES
DBA_FREE_SPACE

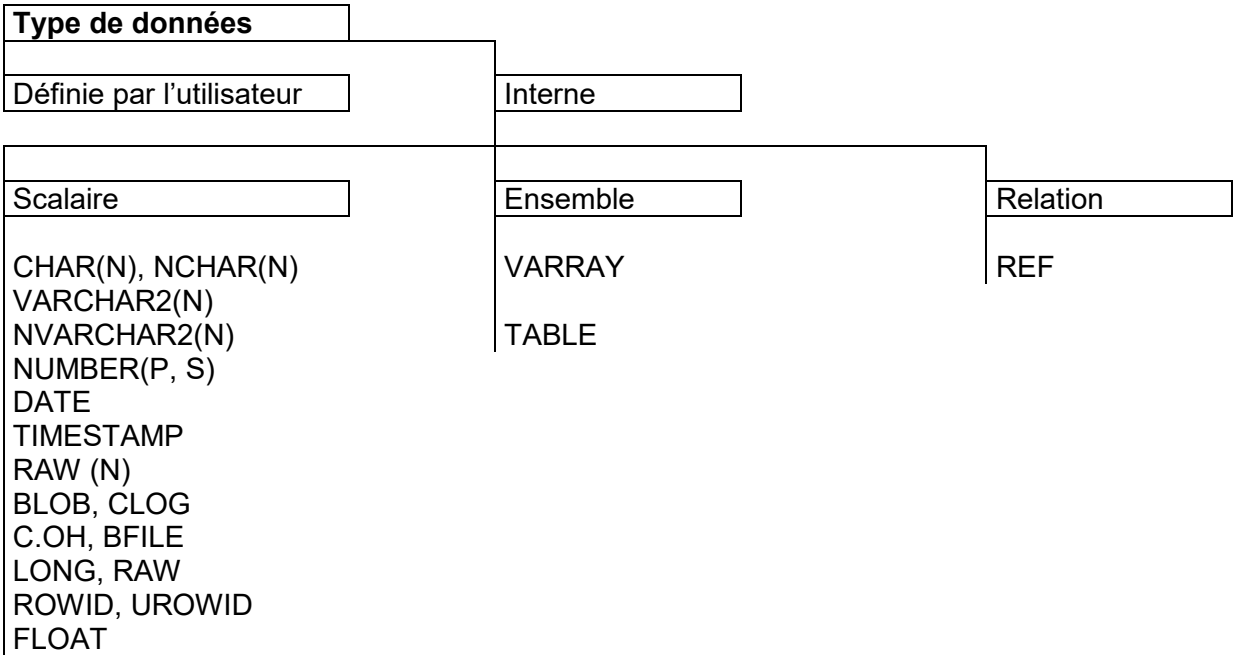
Exemple:

```
SELECT extent_id, file_id, block_id, blocks
FROM    dba_extents
WHERE owner = 'XXXX'
AND segment_name = 'YYYY' ;
```

Types de tables

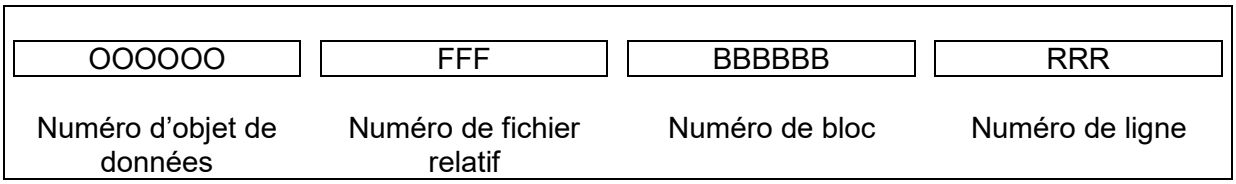
Tables normales,
Tables partitionnées,
Tables organisées en index (TOI).
Tables clustérisées.

Types de données internes Oracle

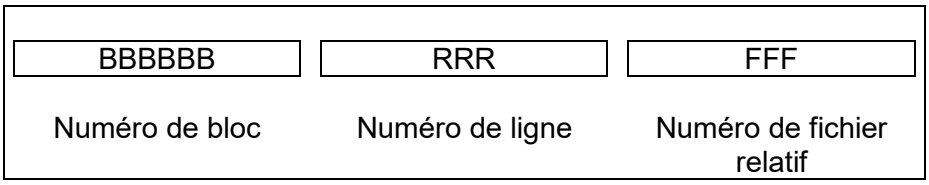


Format de rowid

1. Rowid étendu



2. Rowid restreint



Créer une table

CREATE TABLE owner.table (col1 format1, col2 format2 …) TABLESPACE tbs ;

Pour qu'un utilisateur puisse créer une table, il doit avoir les droits système CREATE TABLE.
Pour pouvoir créer une table dans le schéma d'un autre utilisateur : CREATE ANY TABLE

Règles relatives à la création d'une table

- Placer les tables dans des tablespaces distincts (ne contenant pas de segment d'annulation, de segment temporaire et d'index).
- Utilisez des tables générées localement pour éviter toute fragmentation (et placez les tables dans des tablespaces gérés localement pour éviter la fragmentation).
- Utilisez un nombre limité de tailles d'extent (ensemble de blocs contigus) standatd pour les tables pour éviter la fragmentation des tablespaces.

Créer des tables temporaires

CREATE GLOBAL TEMPORARY TABLE owner2.table2
AS SELECT * FROM owner1.table1 ;

Ces tables conservent les données jusqu'à la fin d'une transactions ou d'une cession.
Aucun verrou LMD n'est placé sur les données de ces tables.
Vous pouvez créer des indexes, des vues, et des déclencheurs dans ces tables/

Définir PCTFREE et PCTUSED

$$\text{PCTFREE} = \frac{(\text{longueur de ligne moyenne} - \text{Longueur de ligne initiale}) * 100}{\text{longueur de ligne moyenne.}}$$

$$\text{PCTUSED} = \frac{(\text{longueur de ligne moyenne}) * 100}{\text{Espace de données disponible}}$$

La commande ANALYSE TABLE permet d'estimer la longueur de la ligne moyenne.

Modifier les paramètres d'utilisation de blocs et de stockage

ALTER TABLE owner.table
PCTFREE nnn
PCTUSED nnn
STORAGE (NEXT nnn [K|M] MINIEXTENTS nn MAXEXTENTS nnn) ;

Allouer manuellement des extents (ensembles de blocs contigus)

ALTER TABLE owner.table
ALLOCATE EXTENT (SIZE nnn [K|M] DATAFILE '/path/dataxxx.DBF');

Réorganisation d'une table non partitionnée

ALTER TABLE owner.table MOVE TABLESPACE tbsXXX ;

La réorganisation d'une table non partitionnée conserve sa structure mais pas son contenu. (Elle permet de déplace une table vers un autre tablespace ou de réorganiser des extents). Après avoir transféré la table, il faut reconstruire l'index pour éviter les erreurs suivantes : ORA-01502 : index 'iiii » or partition of such index is in unusable state.

Vider une table

TRUNCATE TABLE owner.table [{ DROP | REUSE } STORAGE] ;

Lorsque vous videz une table, toutes ses lignes sont supprimées et l'espace utilisé est libéré. Les index correspondants sont également vidés.

Supprimer une table

DROP TABLE owner.table CASCADE CONSTRAINTS ;

Supprimer une colonne (en V9)

ALTER TABLE owner.table DROP COLUMN comments
[CONTINUE | CASCADE CONSTRAINTS [CHECKPOINT nnnn]] ;

Cette opération supprime la longueur et les données d'une colonne de chaque ligne, libérant ainsi de l'espace dans les blocs de données.
La suppression d'une colonne dans une table volumineuse prend énormément de temps.

Renomer une colonne (en V9)

ALTER TABLE owner.table RENAME COLUMN col1 TO col2 ;

Utiliser l'option UNUSED

1. Marquer la colonne comme non utilisée :

ALTER TABLE owner.table SET UNUSED COLUMN comments CASCADE CONSTRAINTS;

2. Supprimer les colonnes non utilisées :

ALTER TABLE owner.table DROP UNUSED COLUMN CHECKPOINT nnnn ;

3. Poursuivre la suppression des colonnes :

ALTER TABLE owner.table DROP UNUSED COLUMN CHECKPOINT nnnn ;

Identifier les tables contenant des colonnes non utilisées

SELECT * FROM dba_unused_col_tabs ;

Identifier les tables pour lesquelles une opération DROP COLUMN n'est pas terminée

SELECT * FROM dba_partial_drop_tabs ;

Restriction relative à la suppression d'une colonne

On ne peut pas supprimer :

1. Une colonne d'une table de type objets,
2. des colonnes de tables de type imbriquées,
3. toutes les colonnes d'une table.
4. une colonne d'une clé de partitionnement,
5. une colonne d'une table appartenant à SYS.
6. d'une table organisée en index, une colonne qui constitue une clé primaire.
7. une colonne de type LONG ou LONG RAW marqué comme non utilisée mais non supprimé.

Infos sur les tables : DBA_TABLES, DBA_OBJECTS

Types d'Index

1. Logiques :

    - Index basés sur une colonne ou concaténés
    - Index uniques ou non uniques
    - Index basés sur une fonction
    - Index de domaine

2. Physiques :

    - Index partitionnés ou non partitionnés
    - Index B-tree : index normaux ou à clé inversée
    - Index bitmap

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

- Index basé sur une seule colonne,
- Index concaténé, ou index composé, créé sur plusieurs colonnes de tables. (nombres max d'index pour une colonne : 32).

Note : il y a :

- Des indexes basés sur une fonction,
- Index de domaine.
- Index partitionnés ou index non partitionnés.

Index b-tree

Racine
Branche
Feuille

Index bitmap

Une structure bitmap présente également une structure b-tree, mais le noeud 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.

Comparaison index b-tree et bitmap

Index b-tree	Index bitmap
Adaptés aux colonnes de fortes cardinalités	Adaptés aux colonnes de faibles cardinalités
Les mises à jour des clés consomment peu de ressources	Les mises à jour des colonnes de clé consomment de grandes quantités de ressources
Inefficace pour les interrogations utilisant des prédicats OR	Efficace pour les interrogations utilisant des prédicats OR
Utiles pour OLTP (traitement des transactions en ligne)	Utile pour le data warehouse

Créer un index b-tree

CREATE INDEX owner.idxxxx ON owner.tabxxxx (col1 [, col2] …) PCTFREE nnn
STORAGE (INITIAL nnnK NEXT nnnK PACTINCREASE n MAXEXTENTS nnnn)
TABLESPACE tbsidxxxx ;

Note: on peut créer un index avec un compte different du compte de la table, sur lequel est créé l'index.

UNIQUE : désigne un index unique (la valeur par défaut est non unique).
Schéma : propriétaire de l'index et de la table.
PCTFREE : indique l'espace réservé dans chaque bloc (pourcentage de l'espace total moins l'en-tête de bloc) au moment de la création pour l'insertion de nouvelles entrées d'index.
INITRANS : définit le nombre d'entrées de transactions préallouées dans chaque bloc (la valeur par défaut et minimale est 2).
MAXTRANS : limite le nombre d'entrées de transaction pouvant être allouées à chaque bloc (la valeur par défaut est 255).
STORAGE clause : identifie la clause de stockage qui détermine la mode d'allocation des extents (ensemble de blocs contigus) à l'index.
LOGGING : indique que la création de l'index et les opérations suivante sur l'index sont consignées dans le fichier de journalisation (redo log) en ligne (c'est la valeur par défaut).
NOLOGGING : indique que la création de l'index et certains type de chargement de données ne sont pas consignées dans le fichier de journalisation en ligne.
NOSORT : indique que les lignes sont stockées dans la base de données en ordre croissant. Par conséquent, le serveur Oracle n'a pas besoin de les trier lors de la création de l'index.

Règles relatives à la création d'index

- Equilibrez les besoins des interrogations et des opérations LMD.
- Placer les index dans un tablespace distinct.
- Utiliser des tailles d'extent (ensemble de blocs contigus) uniformes : multiples de cinq blocs ou MINIMUM EXTENT pour le tablespace.
- Utiliser la clause NOLOGGING pour les index volumineux.
- La valeur INITRANS doit généralement être plus élevé pour les index que pour les tables correspondantes.

Créer un index bitmap

CREATE BITMAP INDEX idxxxxx
ON tabxxxx (col1 [ASC | DESC] [, col2] …) PCTFREE nnn
STORAGE (INITIAL nnnK NEXT nnnK PACTINCREASE n MAXEXTENTS nnnn)
TABLESPACE tbsidxxxx ;

Modifier les parameters de stockage des index

ALTER INDEX idxxxxx STORAGE (NEXT mmmK MAXEXTENTS nnnnn) ;

Allouer et libérer de l'espace d'indexation

Allouer de l'espace à un index :
ALTER INDEX idxxxx ALLOCATE EXTENT (SIZE nnnK DATAFILE '/path/file') ;

Libérer de l'espace inutilisé situé au-delà du repère high-water mark d'in index :

ALTER INDEX idxxxx DEALLOCATE UNUSED ;

Note : lorsqu'une table est vidée, son index aussi.

Reconstruire des index

Utiliser la commande ALTER INDEX pour :

- Déplacer un idex vers un autre tablespace
- Optimiser l'utilisation de l'espace en retirant les entrées supprimées :

ALTER INDEX idxxxxx REBUILD [tablespace idxxxxx2] ;

Reconstruire des index en ligne :

- La reconstruction des index peut s'effectuer avec un verrouillage de table minimal : ALTER INDEX idxxxxx REBUILD ON LINE ;
- Certaines restrictions subsistent cependant :
- Vous ne pouvez pas reconstruire un index sur une table temporaire.
- Vous ne pouvez pas reconstruire l'ensemble d'un index partitionné. Vous devez reconstruire chaque partition ou sous-partition.
- Vous ne pouvez pas modifier la valeur du paramètre PCTFREE de l'index dans on intégralité.

Fusionner des index

Lorsqu'un index est fragmenté, vous pouvez le reconstruire ou le fusionner.
La fusion d'un index correspond à la reconstruction de blocs en ligne (Durant la fusion, on remplit un bloc pour en vider un autre). Syntaxe :

ALTER INDEX idxxxxxx COALESCE ;

Vérifier la validité des index

L'analyse d'un index vous permet :

- De contrôler la corruption de bloc. Cette commande, par contre, ne vérifie pas si les entrées d'index correspondent aux données de la table.
- D'insérer dans la vue INDEX_STATS des informations sur l'index.

ANALYSE INDEX owner.idxxxxx VALIDATE STRUCTURE ;

Select blocks, pct_used, distinct_keys, lf_rows, del_lf_rows from index_stats ;

Conseil : il faut réorganiser l'index si la proportion de lignes supprimées est élevée, par exemple, si le rapport entre DEL_LF_ROWS et LF_ROWS est > à 30 %.

Supprimer des index

- Supprimez et recréez un index avant de procéder à des chargements en masse.
- Supprimez les index rarement utilisés et créez les lorsqu'ils sont nécessaires.
- Supprimez et recréez les index non valide (attribut « INVALID »)

DROP INDEX idxxxxxx ;

Identifier les index non utilisés

Pour lancer la surveillance de l'utilisation d'un index :
ALTER INDEX owner.idxxxxxx MONITORING USAGE ;

Pour arrêter la surveillance de l'utilisation d'un index :
ALTER INDEX owner.idxxxxxx NOMONITORING USAGE ;

Colonnes de la vue V$OBJECT_USAGE :

INDEX_NAME : nom de l'index
TABLE_NAME : nom de la table
MONITORING : indique si la surveillance est activée (ON) ou non (OFF).
USED : indique par YES ou NO si l'index est utilisé durant la surveillance.
START_ MONITORING : heure du début de la surveillance de l'index.
END_ MONITORING : heure de fin de la surveillance de l'index.

Obtenir des informations sur les index

DBA_INDEXES : infos sur les index.
DBA_IND_COLUMNS : infos sur les colonnes indexées.
V$OBJECT_USAGE : infos sur l'usage d'un index.

GERER L'INTEGRITE DES DONNEES

L'intégrité des données garantit que les données d'une base respectent certaines règles. Trois méthodes principales pour la garantir :

- Le code d'application
- Les déclencheurs (triggers) de base de données,
- Les contraintes d'intégrité déclaratives.

Le choix de la méthode dépend de la conception de la base de données.

Le code d'application peut être implémenté sous forme de procédure stockées dans la base ou sous forme d'application exécutée sur le client.

Les contraintes d'intégrité améliore les performances.
Elles centralisent les règles. Elles sont documentées dans le dictionnaire de données.
On peut les activer ou les désactiver.

Types de contrainte

Contrainte	Description
NOT NULL	Indique qu'une colonne ne peut pas contenir de valeur NULL
UNIQUE	Désigne une colonne ou une combinaison de colonne comme unique
PRIMARY KEY	Désigne une colonne ou une combinaison de la colonnes comme clé primaire de la table. Elle est forcément unique.
FOREIGN KEY	Désigne une colonne ou une combinaison de la colonnes comme clé étrangère dans une contrainte d'intégrité référentielle
CHECK	Indique une condition à laquelle doit répondre chaque ligne de la table

Etat des contraintes

ENABLE (activée), DISABLE (désactivée).

DISABLE	NOVALIDATE	Contrainte non contrôlée
DISABLE	VALIDATE	Interdit toute modification de la colonne à laquelle s'applique la contrainte. L'index de la contrainte est supprimée.
ENABLE	NOVALIDATE	Interdit l'entrée de nouvelles de données non conformes à la contrainte. La table peut cependant contenir des données non valides (utile pour des données provenant de téléchargement (?) _ config. De data warehouse téléchargeant vers l serveur des données OLTP valides).
ENABLE	VALIDATE	Interdit l'insertion dans la table de lignes violant la contrainte (sauf si l'on désactive la contrainte). Les lignes qui violent la contrainte doivent être mises à jour ou supprimées pour que la contrainte puisse recevoir l'état ENABLE VALIDATE.

Contraintes différées et immédiates
(Initially immediate & Initially deferred).

Impossible de modifier une contrainte non différable, afin de l'appliquer à la fin d'une transaction.

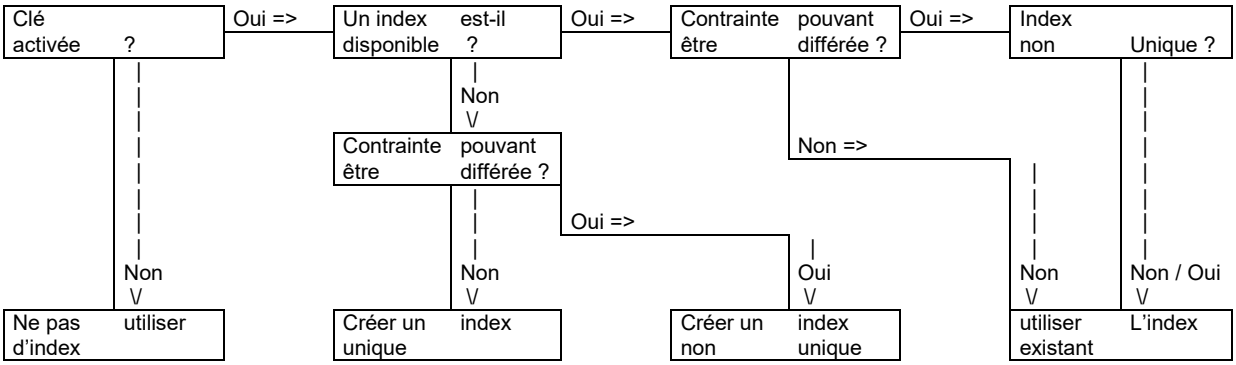
Les contraintes différées ne sont vérifiées que si la transaction est validée.

a) L'instruction "ALTER SESSION" dispose d'une clause permettant d'affecter la valeur IMMEDIATE ou DEFERRED aux contraintes. Cette commande suppose de définir TOUTES les contraintes pouvant être différées (la liste des noms de contrainte ne peut pas être définie). Cette instruction s'applique uniquement à une session en cours.

ALTER SESSION SET CONSTRAINT[S] = { IMMEDIATE | DEFERRED | DEFAULT } ;

b) L'instruction "SET CONSTRAINT" définit des IMMEDIATE ou DEFERRED pour une transaction particulière :

SET CONSTRAINT | CONSTRAINTS { constraint | ALL} { IMMEDIATE | DEFERRED } ;



Informations / conseils / remarques sur les clés étrangères

Action souhaitée	**Solution appropriée**
Supprimer la table parent des contraintes	Mise en cascade
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

LDD impliquant la table parent

- La clé étrangère doit être supprimée avant la table parent. Possible avec :

  DROP TABLE table CASCADE CONSTRAINTS ;

- Vous ne pouvez pas vider la table parent sans supprimer ou désactiver la clé étrangère.
- La clé étrangère doit être supprimée avant le tablespace contenant la table parent. Pour ce faire, utiliser :

  DROP TABLESPACE tablespace INCLUDING CONTENTS CASCADE CONSTRAINTS ;

DELETE … CASCADE ; : permet de supprimer les lignes dans la tables parents et les lignes correspondante dans la table enfant.

Définir les contraintes lors de la création d'une table

Syntaxe :
Col datatype
[CONSTRAINT contrainte]
{ [NOT] NULL
| UNIQUE [USING INDEX index_clause]
| PRIMARY KEY [USING INDEX index_clause]
| REFERENCES [schema.] table [(column)]
| FOREIGN KEY (col1 [, col2] …])
REFERENCES [schema.]table [(col1 [, col2] …)]
[ON DELETE CASCADE]
| CHECK (condition) }
[constraint_state]
avec
constraint_state :==
[NOT DEFERRABLE | DEFERRABLE [INITIALLY { IMMEDIATE | DEFERRED }]]
[DISABLE | ENABLE [VALIDATE | NOVALIDATE]]

où :
CONSTRAINT : identifie la contrainte par le nom "contrainte" stockée 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 contrainte 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 DML (une telle contrainte ne peut être différée par des sessions ou des transactions. « NOT DEFERRABLE » est utilisé par défaut).
INITIALLY IMMEDIATE : indique qu'au début de la transaction, la contrainte doit, par défaut, être vérifiée à la fin de chaque instruction LMD (c'est d'ailleurs la clause par défaut si aucune clause n'est précisée).
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.

Exemple :
CREATE TABLE owner.tabxxx
(col1 format1 CONSTRAINT nom_contrainte1 PRIMARY KEY DEFERRABLE
USING INDEX STORAGE (INITIAL nnnK NEXT nnnnK) TABLESPACE tbsxxxxx ,
Col2 format2 CONSTRAINT nom_contrainte2 NOT NULL,
Col3 format3) TABLESPACE tbsyyyyyy ;

Remarque : il est recommandé d'adopter une convention d'appellation standard pour les contraintes ; notamment pour les contraintes CHECK, puisque vous pouvez créer plusieurs fois la même contrainte avec des noms différents.
Vous devez utiliser les contraintes de table dans les cas suivants :

- Lorsqu'une contrainte s'applique à plusieurs colonnes,
- Lorsqu'une table est modifiée pour recevoir des contraintes autres que NOT NULL.

La définition d'une contrainte à partir du type NOT NULL une fois la table créée n'est possible que dans le cas suivant :
ALTER TABLE tabxxxx MODIFY colxxx CONSTRAINT constraintexxxxx NOT NULL ;

Exemple de définition de contrainte après la création d'une table

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

Remarque : la clause EXCEPTIONS, décrite dans la section "activer les contraintes" peut être utilisée pour identifier les lignes violant une contrainte ajoutée par la commande ALTER TABLE …

Règles relatives à la définition des contraintes

A) Contraintes UNIQUE et de clé primaire :

- placer les index dans une tablespace distinct.
- Utiliser des index non-uniques si les chargement en masse sont fréquents.

B) Clés étrangères d'autoréférencement :

- Définissez et activez les clés étrangères après le 1er chargement des données.
- Différer la vérification des contraintes.

Activer les contraintes

Type activation	Conditions et syntaxe
ENABLE NOVALIDATE	• Table non verrouillée • Clés primaires & uniques doivent utiliser des index non-uniques ALTER TABLE [schema.]table ENABLE NOVALIDATE { CONSTRAINT contrainte | PRIMARY KEY | UNIQUE (co1 [, col2] …) } [USGIN INDEX index_clause] Ex.: ALTER TABLE owner.tabxxxx ENABLE NOVALIDATE CONSTRAINT constraint_pk ;
ENABLE VALIDATE	• Table verrouillée • Peut utiliser des index uniques ou non-uniques • Requière des données de tables valides ALTER TABLE [schema.]table ENABLE VALIDATE { CONSTRAINT contrainte | PRIMARY KEY | UNIQUE (co1 [, col2] …) } [USGIN INDEX index_clause] [EXCEPTIONS INTO [schema.]table]

Renommer les contraintes

```
ALTER TABLE [schema.]table
RENAME CONSTRAINT old_constraint_name TO new_constraint_name ;
```

Utiliser la table EXCEPTIONS

- Exécuter le script « utilexpt1.sql » pour créer la table EXCEPTIONS .

- Exécuter l'instruction « ALTER TABLE … » avec l'option EXCEPTIONS .
- Lancer une sous-interrogation sur la table EXCEPTIONS pour identifier les lignes contenant des données non valides.
- Corriger ces erreurs.
- Réexécuter l'instruction « ALTER TABLE … » pour activer la contrainte.

Exemple :

```
@ ?/rdbms/admin/utlexpt1

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

ALTER TABLE hr.employee
*
ORA-02298: cannot enable (HR.EMPLOYEE_DEPT_ID_FK) – parent keys not found

Pour identifier les données non valides, dans la table exceptions :

```
SELECT rowid, id, last_name, dept_id FROM hr.employee
WHERE ROWID IN (SELECT row_id FROM exceptions ) FOR UPDATE ;

ROWID              ID   LAST_NAME  DEPT_ID
------------------ ---- ---------- -------
AAAAeyAADAAAAA1AAA 1003 Pirie      50
```

Corriger les erreurs continues dans les données :

```
UPDATE hr.employee
SET         dept_id=10
WHERE  rowed='AAAAeyAADAAAAA1AAA';
COMMIT;
```

Obtenir des informations sur les contraintes

DBA_CONSTRAINTS
DBA_CONS_COLUMNS

Exemple :

```
SELECT constraint_name, constraint_type, deferrable, deferred, validated
FROM   dba_constraints
WHERE  owner='HR'
AND    table_name='EMPLOYEE';
```

GESTION DE LA SECURITE

Profils

Un profil est un ensemble nommé contenant les limites relatives aux mots de passe et aux ressources :

- Durée de vie et expiration des mots de passe,
- Historique des mots de passe,
- Vérification de la complexité des mots de passe,
- Vérouillage de compte,
- Temps CPU ;
- Opérations d'entrée-sorite (E/S),
- Durée d'inactivité,
- Durée de connexion,
- Espace mémoire (zone SQL privée pour serveur partagé uniquement),
- Sessions simultanées.

Les commandes « CREATE USER » et « ALTER USER » permet d'affecter des profils aux utilisateurs.
Les profils peuvent être activés ou désactivées.
Le profil DEFAULT est affecté par défaut.

Pour activer la gestion des mots de passe, exécutez le script « utlpwdmg.sql » avec l'ID utilisateur « SYS ».

Verrouillage d'un compte

Paramètre	Description
FAILED_LOGIN_ATTEMPTS	Nombre d'échecs de connexion avant verrouillage du compte
PASSWORD_LOCK_TIME	Durée, en jours, de verrouillage du compte, après le nombre d'échecs de connexion défini

Durée de vie et expiration des mots de passe

Paramètre	Description
PASSWORD_LIFE_TIME	Durée de vie, en jours, du mot de passe avant expiration
PASSWORD_GRACE_TIME	Période de grâce, en jours, pendant laquelle l'utilisateur peut changer de mot de passe après la 1ière connexion établie, une fois le mot de passe expiré

Historique des mots de passe

La vérification de l'historique des mots de passe permet d'empêcher un utilisateur de réutiliser un mot de passe, pendant une période donnée. On peut la mettre en œuvre avec ces 2 paramètres suivants :

Paramètre	Description
PASSWORD_REUSE_TIME	Période, en jours, pendant laquelle un mot de passe ne peut pas être réutilisé (sinon DEFAULT, UNLIMITED …).
PASSWORD_REUSE_MAX	Nombre maximal de modifications nécessaires avant la réutilisation d'un mot de passe (force l'utilisateur à définir un mot de passe différent des MdP précédents).

Si l'un ce ces 2 paramètres est UNLIMITED, l'autre doit être aussi UNLIMITED.

Vérification des mots de passe

Paramètre	Description
PASSWORD_VERIF_FUNCTION	Fonction PL/SQL vérifiant la complexité d'un mot de passe, avant que celui-ci soit affecté.

Fonction de mot de passe fournie par l'utilisateur

Cette fonction doit être créée dans le schéma « SYS » et respecter la spécification suivante :

```
function_name (
    userid_parameter IN VARCHAR2(30),
    password_parameter IN VARCHAR2(30),
    old_password_parameter IN VARCHAR2(30) )
RETURN BOOLEAN
```

(elle renvoie la valeur TRUE ou FALSE).

Fonction vérification de mot de passe VERIFY_FUNCTION

- Longueur mini d'un mot de passe = 4 caractères.
- Le mot de passe doit être différent du nom utilisateur.
- Il doit comporter au moins une lettre, un caractère numérique et un caractère spécial.
- Il doit comporter au moins trois lettres différentes par rapport à l'ancien mot de passe.

Le script « utlpwdmg.sql » modifie le profil DEFAULT par la commande :

```
ALTER PROFILE DEFAULT LIMIT
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10
PASSWORD_REUSE_TIME 1800
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PASSWORD_VERIF_FUNCTION verify_fonction ;
```

Créer un profil

```
CREATE PROFILE  profile LIMIT
[ FAILED_LOGIN_ATTEMPTS max_value ]
[ PASSWORD_LIFE_TIME max_value ]
[ { PASSWORD_REUSE_TIME |
   PASSWORD_REUSE_MAX  } { max_value | UNLIMITED } ]
[ PASSWORD_LOCK_TIME max_value ]
[ PASSWORD_GRACE_TIME max_value ]
[ PASSWORD_VERIF_FUNCTION { verify_fonction | NULL | DEFAULT } ;
```

Modifier un profil

ALTER PROFILE { profile | DEFAULT } LIMIT
[FAILED_LOGIN_ATTEMPTS max_value]
[PASSWORD_LIFE_TIME max_value]
[{ PASSWORD_REUSE_TIME |
  PASSWORD_REUSE_MAX } { max_value | UNLIMITED }]
[PASSWORD_LOCK_TIME max_value]
[PASSWORD_GRACE_TIME max_value]
[PASSWORD_VERIF_FUNCTION { verify_fonction | NULL | DEFAULT } ;

Exemples de valeurs pour PASSWORD_LOCK_TIME :
1 heure : 1/24
1 mn : 1/1440

Supprimer un profil

DRIOP PROFILE profilexxx [CASCADE] ;

CASCADE retire le profile tous les utilisateurs qui ont ce profile. Il retrouve le profil : DEFAULT.
La suppression ne s'applique qu'aux nouvelles sessions créées et non au sessions en cours.

Gestion des ressources

Les limites relatives à la gestion des ressources peuvent s'appliquer au niveau session, au niveau appel ou aux deux.
Les limites peuvent être définies par des profiles via la commande CREATE PROFILE.

Activer les ressources relatives aux ressources

Affecter la valeur TRUE aux paramètre d'initialisation RESSOURCE_LIMIT .
Activer le paramètre à l'aide de la commande ALTER SYSTEM pour appliquer les limites relatives aux ressources :
ALTER SYSTEM SET RESSOURCE_LIMIT=TRUE ;

Définir des limites relatives aux ressources, au niveau session :

Ressource	Description
CPU_PER_SESSION	Temps CPU total calculé en centième de secondes
SESSIONS_PER_USER	Nombre de sessions simultanées autorisées pour chaque nom utilisateur
CONNECT_TIME	Temps de connexion calculé en minutes
IDLE_TIME	Période d'inactivité calculé en minutes
LOGICAL_READS_PER_SESSION	Nombre de blocs de données (lectures physiques et logiques)
PRIVATE_SGA	Espace privé de la mémoire SGA mesuré en octets (dans le cas d'un serveur partagé uniquement).

Définir des limites relatives aux ressources, au niveau appel

Ressource	Description
CPU_PER_CALL	Temps CPU total calculé en centième de secondes
LOGICAL_READS_PER_CALL	Nombre de blocs de données pouvant être lus par appel

Créer un profil : limites relatives aux ressources

```
CREATE PROFILE profile LIMIT
[ SESSIONS_PER_USER max_value ]
[ CPU_PER_SESSION max_value ]
[ CPU_PER_CALL max_value ]
[ CONNECT_TIME max_value ]
[ IDLE_TIME max_value ]
[ LOGICAL_READS_PER_SESSION max_value ]
[ LOGICAL_READS_PER_CALL max_value ]
[ COMPOSITE_LIMIT  max_value ]
[ PRIVATE_SGA max_bytes ]
 ;
```

Exemple :

```
CREATE PROFILE dev_prof LIMIT
  SESSIONS_PER_USER 2
  CPU_PER_SESSION 10000
  IDLE_TIM 60
  CONNECT_TIME 480 ;
```

Exemple “max_byte” = nnnKB ou nnnnMB ou DEFAULT ou UNLIMITED

Il y a un outil pour changer ces profiles et ressouces « Database Resource Manager ».

Obtenir des informations sur les limites relatives aux mots de passe et aux ressources

DBA_USERS
DBA_PROFILE

```
Select username, password, account_statuts from dba_users ;
```

Schéma de la base de données

- Un schéma est un ensemble nommé d’objets.
- Lorsqu’un utilisateur est créé, un schéma correspondant est également créé.
- Un utilisateur ne peut être associé qu’à un seul schéma.
- Le nom utilisateur et le nom de schéma sont souvent utilisés indifféremment.

Objets de schéma

- Tables
- Déclencheurs (triggers)
- Contraintes
- Index
- Vues
- Clusters
- Sequences
- Programmes stockés (packages, procédures …),
- Synonymes
- Types de données définis par l’utilisateur,
- Liens de bases de données.

Liste de contrôles pour la création d'utilisateurs

- Identifiez les tablespaces dans lesquels l'utilisateur a besoin de stocker des objets,
- Déterminez les quotas applicables pour chaque tablespace.
- Affecter un tablespace par défaut et un tablespace temporaire.
- Créer un utilisateur.
- Accorder des privilèges et des rôles à l'utilisateur.

Créer un utilisateur (authentification par la base de données)

CREATE USER user
IDENTIFIED { BY password | EXTERNALLY | GLOBALLY AS external name }
[DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA { integer [K | M] | UNLIMITED } ON tablespace
[QUOTA { integer [K | M] | UNLIMITED } ON tablespace
] …]
[PASSWORD EXPIRE]
[ACCOUNT { LOCK | UNLOCK }]
[PROFILE { profile | DEFAULT }]

Créer un utilisateur (authentification par le système d'exploitation)

CREATE USER user
IDENTIFIED EXTERNALLY
DEFAULT TABLESPACE tablespace
TEMPORARY TABLESPACE tablespace
[…] ;

la valeur "OPS$" est utilisée par défaut [comme valeur de préfixe aux noms des utilisateurs).

2) dans le fichier PFILE ou SFILE , ajouter une ligne :
OS_AUTHENT_PREFIX = "val"
Si on veut que la valeur du préfixe soit NULL (nulle) on met :
OS_AUTHENT_PREFIX = ""

On se connecte alors à sqlplus très facilement : « sqlplus ./ » (à partir de son compte du système d'exploitation).

Remarques :
On peut mettre :
OS_AUTHENT_PREFIX=OPS$
Ou bien faire :
CREATE USER ops$user IDENTIFIED BY password …. ;

Le paramètre d'initialisation REMOTE_OS_AUTHENT=TRUE indique que l'utilisateur peut être authentifié par un système d'exploitation distant.