2.2. Transactions, sauvegardes et reprises

- Reprise en cas d'incident (*recovery*) : remettre une base en fonctionnement
  - le plus rapidement possible
  - dans un état le plus proche possible de celui dans lequel elle était avant l'incident.
- Incidents : de l'erreur de programmation à la détérioration des unités physiques de stockage.
- Reprendre : reconstruire une base saine à partir de son *histoire* (cf. *log*).

2.2. Transactions et reprises sous Oracle : Plan

2.2.1- Journaux ou *redo logs*

- Minimum deux fichiers *redo logs* par base
- Gestion circulaire
- Possibilité d'archivage automatique (*archivelog*)
- [+ sauvegardes archives et *redo logs actifs*]

1. Archivage automatique
2. Gestion des *redo logs* archivés

2.2.1.a- Archivage automatique des *redo logs*

- Activation du mécanisme : clause *archivelog* de la commande *create/alter database*
- Si *alter database archivelog* : *log_archive_start* = *true* (*init.ora*)
  - Edition fichier initialisation
  - ou *alter system*
- Note : modification d'un fichier d'initialisation d'une base ouverte prise en compte seulement lors du prochain lancement de l'instance Oracle associée.

2.2.1.a- Activation/Désactivation archivage

- *log_archive_start* = *true* dans *init.ora* : mode archivage automatique effectif dès le lancement (*startup*) de l'instance associée.
- Sinon, activation après le startup par *"alter system archive log start"*
- Désactivation :
  1. *log_archive_start* = *false* dans *init.ora*
  2. ou *"alter system archive log stop"*.

2.2.1.a- Principe de l'archivage

- *LGWR* : processus d'écriture dans le journal
- *ARCn* : processus d'archivage
- Fichier indisponible pendant son archivage.
- Si plusieurs processus d'archivage, plusieurs tampons, etc. : voir les paramètres d'initialisation *log_archive_max_processes*, *log_archive_buffers* et *log_archive_buffer_size*.
- Mécanisme :

2.2.1.a- Principe de l'archivage

- Forcer un archivage :
  1. du log courant et ceux non archivés : *alter system archive log current*
  2. d'un groupe : *alter system archive log group* i
  3. des logs non courants : *alter system archive log all*
  - 1 sur base ouverte
  - 2, 3 sur base montée, ouverte ou fermée.

2.2.1.b- Gestion des redo logs archivés

- *Emplacement physique* :
  - Paramètre *log_archive_dest* ou *alter system archive log . . . to*).
- *Nommage* des logs archivés :
  - Paramètre *log_archive_format*
  - Utilisation possible d'un numéro de séquence qu'Oracle associe aux fichiers archives (identique pour tous les membres d'un même groupe)

2.2.2- Gestion des *rollback segments*

- Alternative : *Tablespaces* d'annulation (Version 9)
- Valeurs de données avant leur modification
- Données de transactions non encore validées
- Peuvent servir pour effectuer des lectures "non sales" (*consistent read*), pour :
  - annuler les effets d'une transaction
  - effectuer une reprise sur la base.
- Segment d'annulation = {"entrées d'annulation"} (*rollback entries*)

2.2.2- *Rollback segments* et transactions

- Entrée d'annulation :
  - Identification du fichier de données (*datafile*)
  - Identification du bloc contenant la donnée
  - Valeur de la donnée avant modification
- Chaînage des entrées d'une transaction ;
- En cas d'annulation : Parcours de la chaîne "en arrière"
- Si segment partagé
  - une table T de transactions par segment
  - T[i] = lien vers la liste des entrées de la transaction P_i

Rollback segments et transactions

- *Affectation d'une transactions* à un segment d'annulation :
  - en début de la transaction,
  - lors de la rencontre de la première instruction de mise à jour ou de définition de données,
  - par le système
  - ou par le programmeur (spécification, dans la transaction, de segment à utiliser).
- *Pas d'association* si transaction d'interrogation i.e.
  - Déclarée comme telle (*set transaction read only*)
  - ou ne faisant que de la consultation
- Distribution "équitable" transactions/segments

Types de *Rollback segments*

1. *SYSTEM* :
   - créé lors de la création d'une base dans la *tablespace SYSTEM*
   - avec les paramètres de stockage par défaut de celle-ci
   - Ne peut pas être supprimé ni mis hors ligne.
2. Segments publics/privés : en nombre quelconque (cf. *create rollback segment/undo tablespace*) ;

Types de *Rollback segments*

3. *Segment d'annulation différée* (*defferred rollback segment*) :
   - créé automatiquement dans la *tablespace SYSTEM*
   - contient, pour un espace de données (*tablespace*) mis hors ligne, les entrées d'annulation qui le concernent et qui, du fait de son état hors ligne, n'ont pas pu lui être appliquées lors d'annulations de transactions.
   - Au retour en ligne de la *tablespace* : utilisation du ou des segments d'annulation différée pour éventuellement y annuler les effets de certaines transactions.

Rollback segments vs *Undo tablespaces*

- A partir de la version 9i : choix lors de la création de la base
- Utilisation de *rollback segments* : *UNDO_MANAGEMENT = MANUAL* dans init.ora
- Utilisation de *Undo tablespaces* : dans init.ora, paramètres
  1. *UNDO_MANAGEMENT = AUTO*
  2. *UNDO_TABLESPACE =* Nom de la *tablespace*
- Création d'une *undo tablespace* :
  1. lors du *create database* : argument *undo tablespace*
  2. ou commande *create undo tablespace*
- Extension d'une *undo tablespace* : *alter tablespace ... add datafile ...*

Undo tablespaces : Exemple

create database ...
undo tablespace "MonUndoTs"
datafile '...\UndoTs01.dbf' *size* 50M,
'...\UndoTs02.dbf' *size* 50M *reuse*
autoextend on next 5120K *max size* 1000M;

Rollback segments vs *Undo tablespaces*

- Changement de *undo tablespace*
  1. Créer une *undo tablespace*
  2. Arrêter la base
  3. Modifier le fichier d'initialisation de la base
  4. Démarrer la base
  5. Supprimer l'ancienne *undo tablespace* : *drop undo tablespace*.
- *Durée de rétention des images avant* :
  - Pour "interroger le passé"
  - Paramètre *dynamique UNDO_RETENTION*
  - Paquetage PL/SQL *dbms_flashback*

2.2.3- Validation et annulation de transactions

- Début transaction = première instruction SQL exécutable
- Fin si :
  1. *commit* ou *rollback* "total", (i.e. pas *rollback to savepoint*).
  2. Rencontre d'une instruction de définition de données comme (*create, drop alter table*, ...) :
     (a) Validation de la transaction si elle comportait des instructions de modification
     (b) Exécution de l'instruction de définition de données comme une transaction.
  3. Déconnexion par l'utilisateur.
  4. Arrêt en échec du processus utilisateur : transaction avortée.

Validation de transactions : Les effets

1.
2.
3.
4.
5.
6.

2.2.3- Cas Echec des transactions

1. Restaurer les anciennes valeurs des données à partir des segments d'annulation
2. Libérer les ressources verrouillées.

2.2.4.a- Isolation, verrouillage et points de contrôle

- Trois *niveaux d'isolation* :
  1. *read committed*,
  2. *serializable*,
  3. *read only*.
- *read committed* = niveau par défaut
- *read only* ne fait pas partie du standard SQL'92
- Pour une transaction, les trois niveaux d'isolation sont applicables (*set transaction isolation level read committed*, *serializable* ou *read only*)
- Pour une session, seuls les deux premiers (*alter session set isolation_level read committed* ou *serializable*).

2.2.4.b- Isolation, Verrouillage et points de contrôle

- Verrouillage de niveau tuple
- Différents types de verrous (*Share*), *eXclusive*), partagés ou exclusifs de niveau tuple (*Row Share*, *Row EXclusive* respectivement), etc.
- Choix par le système du plus petit niveau de verrouillage lors de la rencontre d'une instruction de modification ou de définition de données.
- Verrous maintenus pendant toute la durée de la transaction
- Demandes explicites de verrous :
  - *set transaction isolation level*,
  - *lock table*,
  - *select for update*
  - *alter session set isolation_level* : pour la durée d'une session.

2.2.4.b- Isolation, Verrouillage et points de contrôle

- *Déverrouillage* :
  - Fin de la transaction (annulation ou validation).
  - Cas d'une annulation partielle (*rollback to savepoint*) : Déverrouillage des ressources acquises entre le point de sauvegarde (*savepoint)* et le point d'annulation.
- *Interblocage* : Détection par Oracle
  - Fine granularité du verrouillage $\longrightarrow$ situations devraient être rares.
  - Cas interblocage :
    - Annulation des seuls effets d'une des instructions impliquées
    - Envoi d'un message à la transaction
    - Utilisateur : annuler toute la transaction ou réexécuter ultérieurement l'instruction annulée.

2.2.4.c- Isolation, Verrouillage et Points de contrôle

- Processus *CHKPT* : met à jour des informations de l'en-tête des fichiers de données (*datafiles*) en y inscrivant des informations relatives à la prise des points de contrôle (*checkpoint*).
- Point de contrôle explicite : *alter system checkpoint*
- Point de contrôle implicite : périodiquement par le système.

2.2.4.c- Isolation, Verrouillage et Points de contrôle

- Paramètres de contrôle de la fréquence :
  1. *log_checkpoint_interval* : Périodicité en termes de nombre de blocs physiques (système hôte et non Oracle) du journal,
  2. *log_checkpoint_timeout* :
     - spécifié en nombre de secondes,
     - limite le temps écoulé entre l'enregistrement du *log* le plus récent et le point de contrôle
     - $\simeq$ temps maximum qu'un tampon modifié du cache peut y séjourner avant d'être inscrit sur disque.