Administration d'ORACLE

Mohamed EL ADNANI

Plan

Structure logique et physique d'une base de données

- La structure interne d'une BD oracle est divisée en des unités **physiques** (ont une existence physique tel un datafile) et d'autres dites **logiques** dans le sens ou leur existence est essentiellement liée à des conventions.

Structure logique

- Il existe plusieurs niveaux de structures logiques allant du **schema object** (la structure la plus importante) au **datablock** (la plus petite structure, indépendamment des données, sur laquelle on puisse apposer un contrôle).
- **schema object:** notion vague comportant tous les moyens d'accès à la BD. On y trouve notamment les **tables**, mais aussi les **vues**, les **index**, les **clusters**, les **liens**, les **synonymes**, les **procédures PL/SQL** et les **packages PL/SQL**

Structure logique: schema object

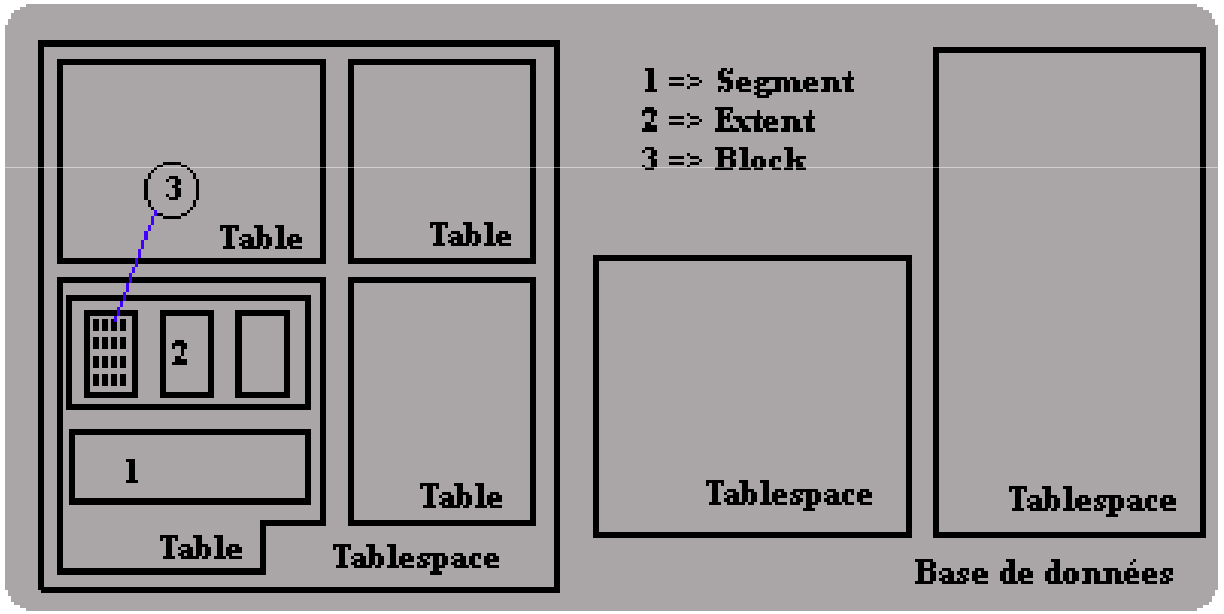
- **Les tables** : sont des Schema Objects qui permettent directement d'accéder aux données
- **Les vues** : ces éléments permettent de donner accès à un sous-ensemble d'une table ou de plusieurs tables (jointes) elles peuvent aussi être utilisées pour cloisonner le champ d'action d'un utilisateur
- **Les index** : ces éléments permettent à une instance du serveur d'accéder plus rapidement à des éléments.
- **Les clusters** : ces schema objects permettent aussi un accès plus rapide aux données. L'astuce consiste à supprimer des données doubles et donc à avoir moins de données à charger à partir des disques.

Structure logique: schema object

- **Les liens** : ces schema objects permettent d'accéder à des données sur une DB distante.
- **Les synonymes** : ils consistent en un nom de remplacement sur un autre schema object.
- **Les procédures et les packages** : une procédure est un ensemble d'ordres PL/SQL permettant de réaliser une action sur des données. Un package est un ensemble de procédures. Pour qu'elles puissent être utilisées, ces unités de stockage ont besoin d'être stockées sur la BD et comme elles permettent la manipulation des données, ce sont des schema objects.
- **L'ensemble de tous les schema objects pour un utilisateur est appelé user's schema.**

Positionnement des éléments d'une BD Oracle

Le schéma suivant replace, les unes en rapport aux autres, les diverses unités logiques existantes



Structure logique: Les tablespaces

- Une BD est composée d'unités logiques appelées **tablespaces**. Un tablespace est utilisé pour regrouper un ensemble d'objets logiques. Une base est dénie avec au moins un tablespace d'origine de nom SYSTEM qui contient le dictionnaire de données.
- La composante principale d’un tablespace est la table qui elle-même composée de **segments**

Les segments

- Un segment est constitué d'**extents** et rentre dans la constitution de la table.
- Une table est constituée d'au moins deux segments : les **data** segments et le **rollback** segment.
- Deux autres segments peuvent apparaître dans une table : **l'index** segment et le **temporary** segment.

Les segments

- **Le data segment** : sert à stoker toutes les données (les valeurs des différentes lignes) que contient la table
- **Le rollback segement** : ce segment stocke des données relatives aux transactions pour les ré-utiliser en cas d'annullation de la transaction par la commande ROLLBACK et permettre ainsi de restituer la base dans l'état initiale ou elle était au départ de la transaction en cours.

Les segments

- **L'index segment** : ce segment optionnel sert à stocker les informations relatives aux index crées sur la table.
- **Le temporary segment** : cet autre segment est utilisé pour stocker les résultats temporaires d'une requête PL/SQL ne pouvant directement s'exécuter en mémoire. Pour ce faire un segment est alloué pour les traitements intermédiaires puis désalloué directement à la fin de la transaction (d'ou son nom).

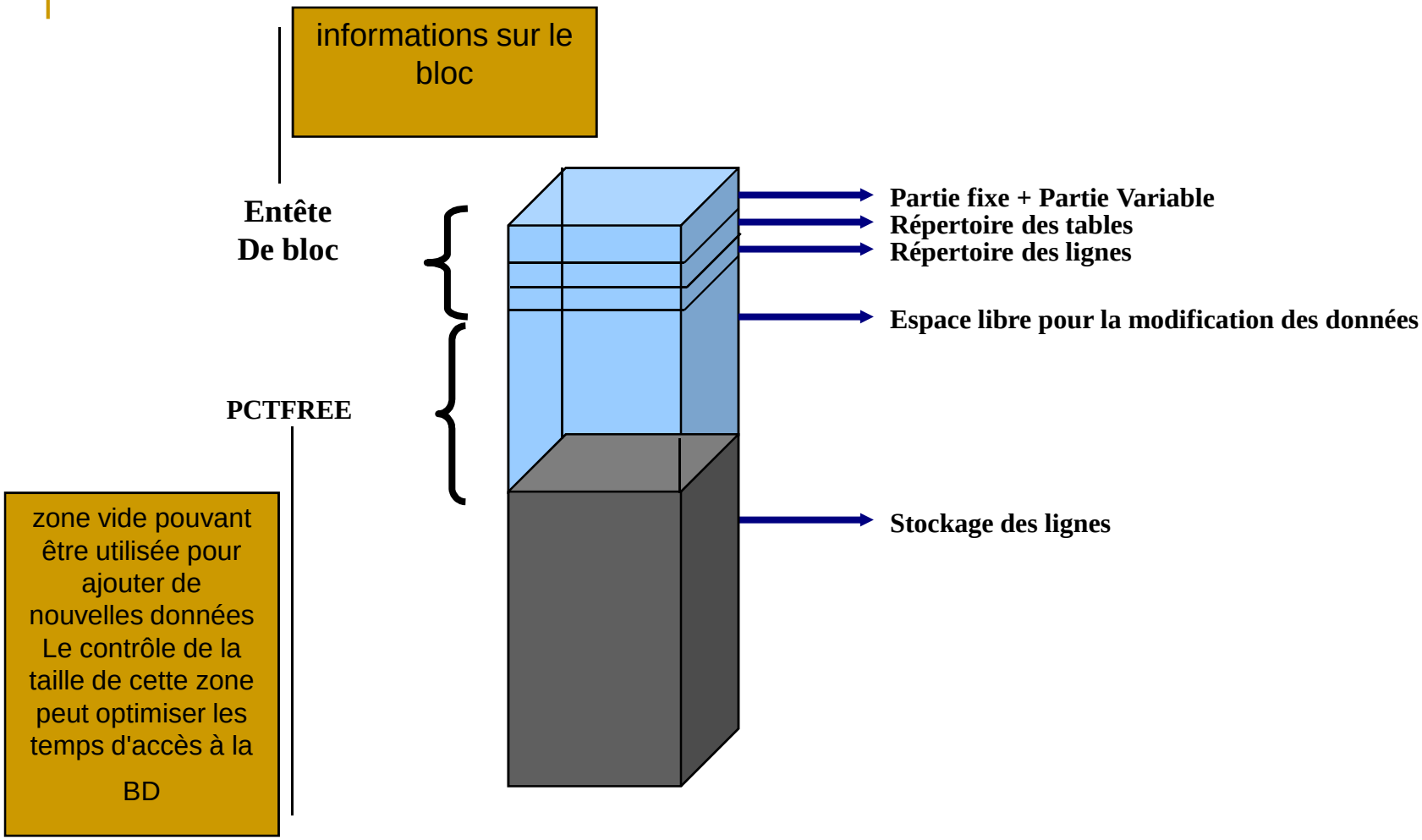
L'extent

- un segment est constitué de plusieurs extents
- L'extent est un ensemble de blocks consécutifs
- Tout comme il existe différents types de segments, il existe différents types d'extents. L'allocation des extents, constituant le segment, est dynamique a condition d'avoir suffisamment de place sur le tablespace courant

Les blocks

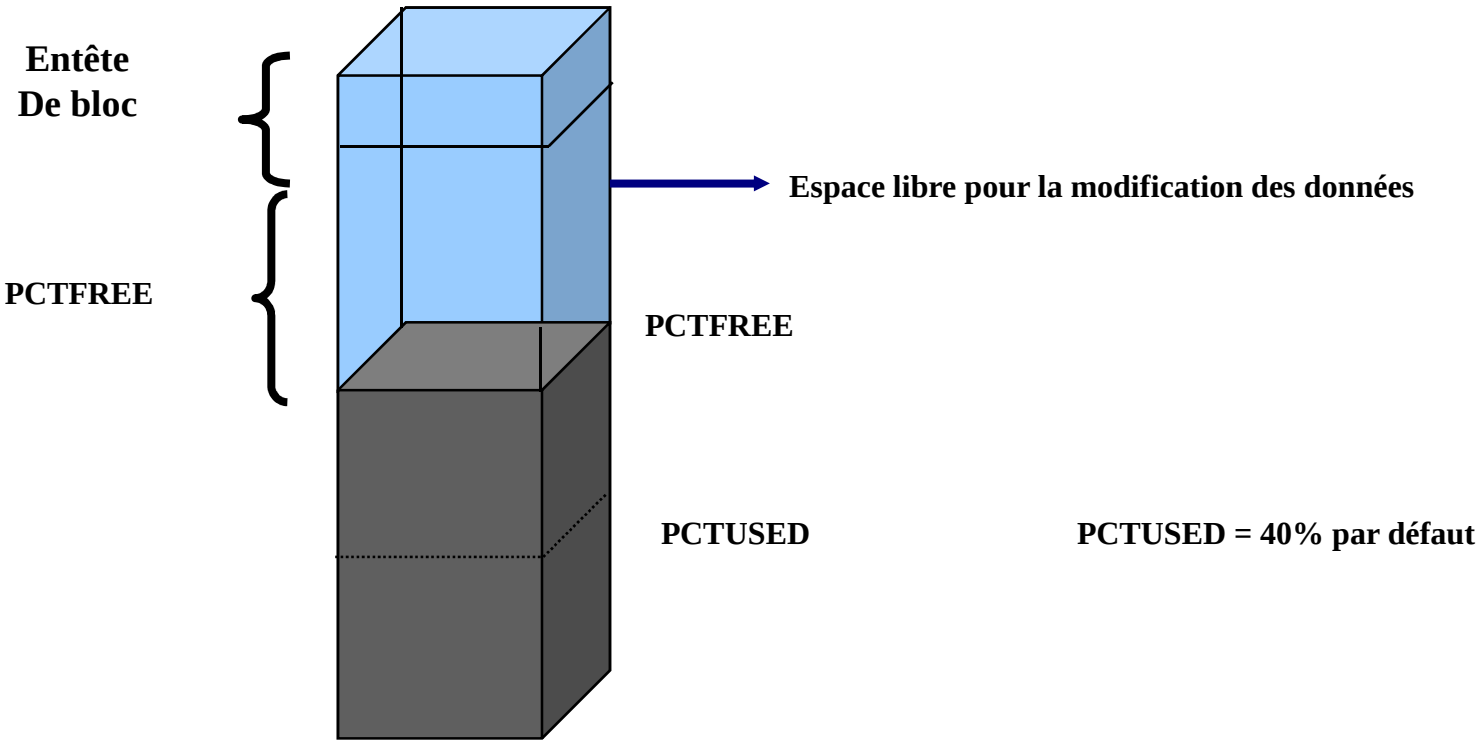
- Le **block** est la plus petite unité logique de stockage que peut manipuler le système
- Tout les blocs constituant la BD ont tous la **même** taille.
- Cette taille peut soit être celle par défaut (fixée par le SGBD), soit être fixée par l'administrateur de la base. Dans ce dernier cas, le paramètre **DB_BLOCK_SIZE** du fichier d'initialisation doit contenir cette valeur

LE BLOC ORACLE POUR UNE TABLE



PCTFREE = 10% par défaut, augmenter cette valeur pour des tables qui ont Un fort taux de modification

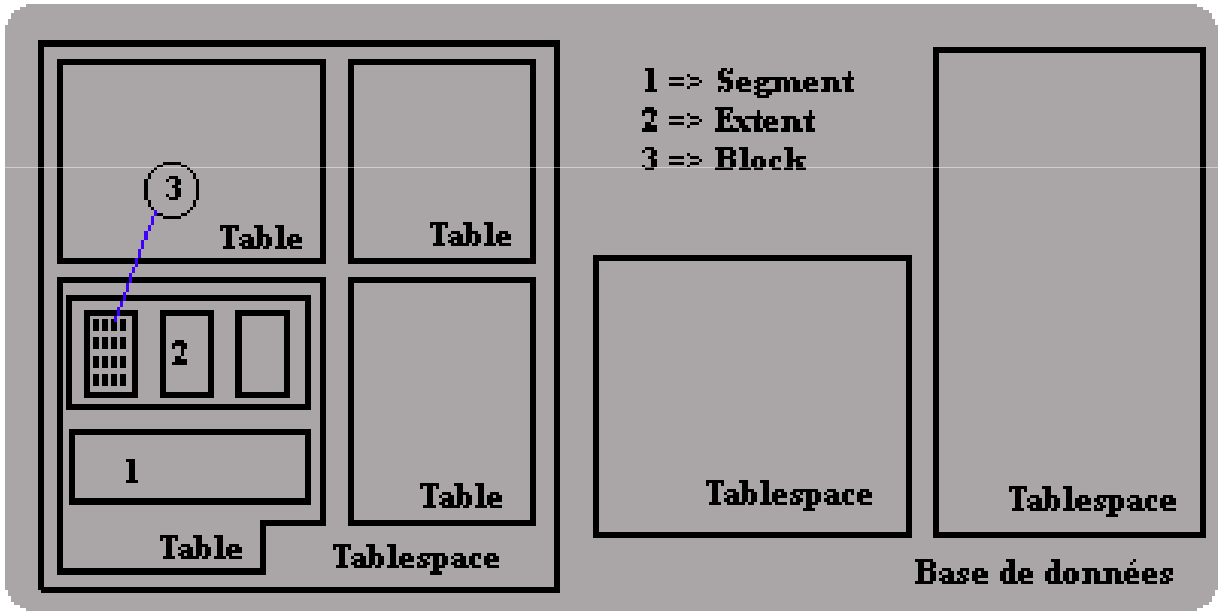
LE BLOC ORACLE POUR UNE TABLE



ESPACE POUR LES DONNES
Si le taux de remplissage est inférieur à PCTUSED ➔ le bloc est inséré en tête
De la FREE LIST et devient disponible pour les insertions de lignes

Récapitulation

Le schéma suivant replace, les unes en rapport aux autres, les diverses unités logiques existantes



Le data dictionary

- Ce dictionnaire est un ensemble de tables et de vues contenant des informations sur la BD
- La BD stocke toutes données sous forme de tables. Ce dictionnaire est généré lors de la création de la BD.
- Ce dictionnaire contient par exemple, des informations sur les utilisateurs, sur les privilèges qu'ont ces utilisateurs, sur les schema objects définis sur la BD, ...
- Exemple de requetes pour ce dictionnaire de données

SELECT * FROM all_db_links;	Demande l'ensemble des liens utilisables sur la BD.
SELECT * FROM V$DBFILE	Demande tous les datafiles de la BD.
SELECT * FROM V$DISPATCHER	Demande l'ensemble des processus dispatchers.
V$LOGFILE V$CONTROLFILE V$QUEUE ...	Encore quelques vues.

Structure physique

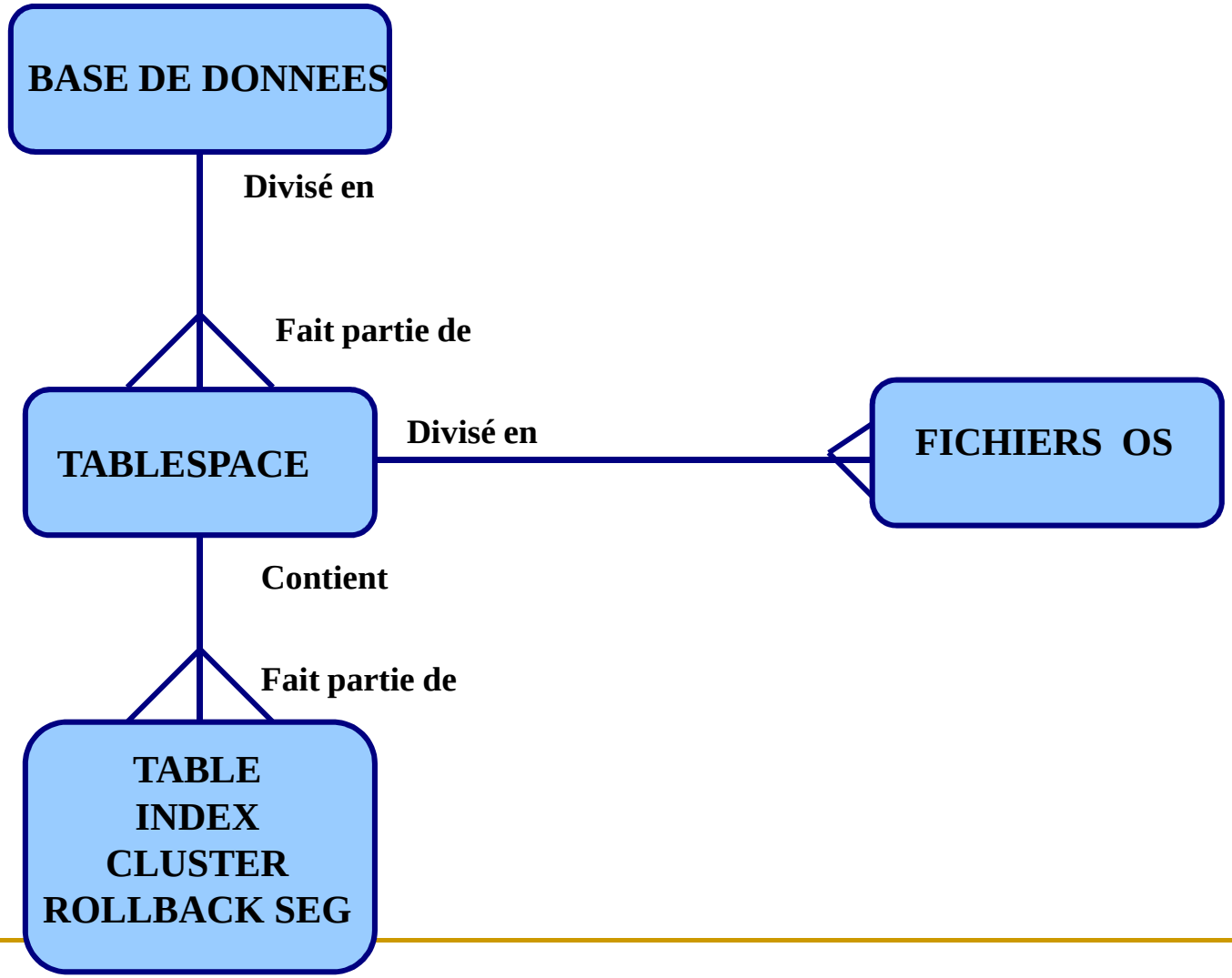
- Les structures physiques sont toutes les structures qui ont une réelle existence : par exemple, un **datafile** (un fichier de données)
- **Les fichiers de données (datafile):** Ces fichiers contiennent l'ensemble des données constituant la BD. Toutes les données accessibles par les **schema objects**.
- Un tablespace doit mettre l'ensemble de ses informations stockées dans un ou plusieurs datafiles. Plus précisément, un schema object peut être stocké dans un ou plusieurs datafiles

Les Redo log file

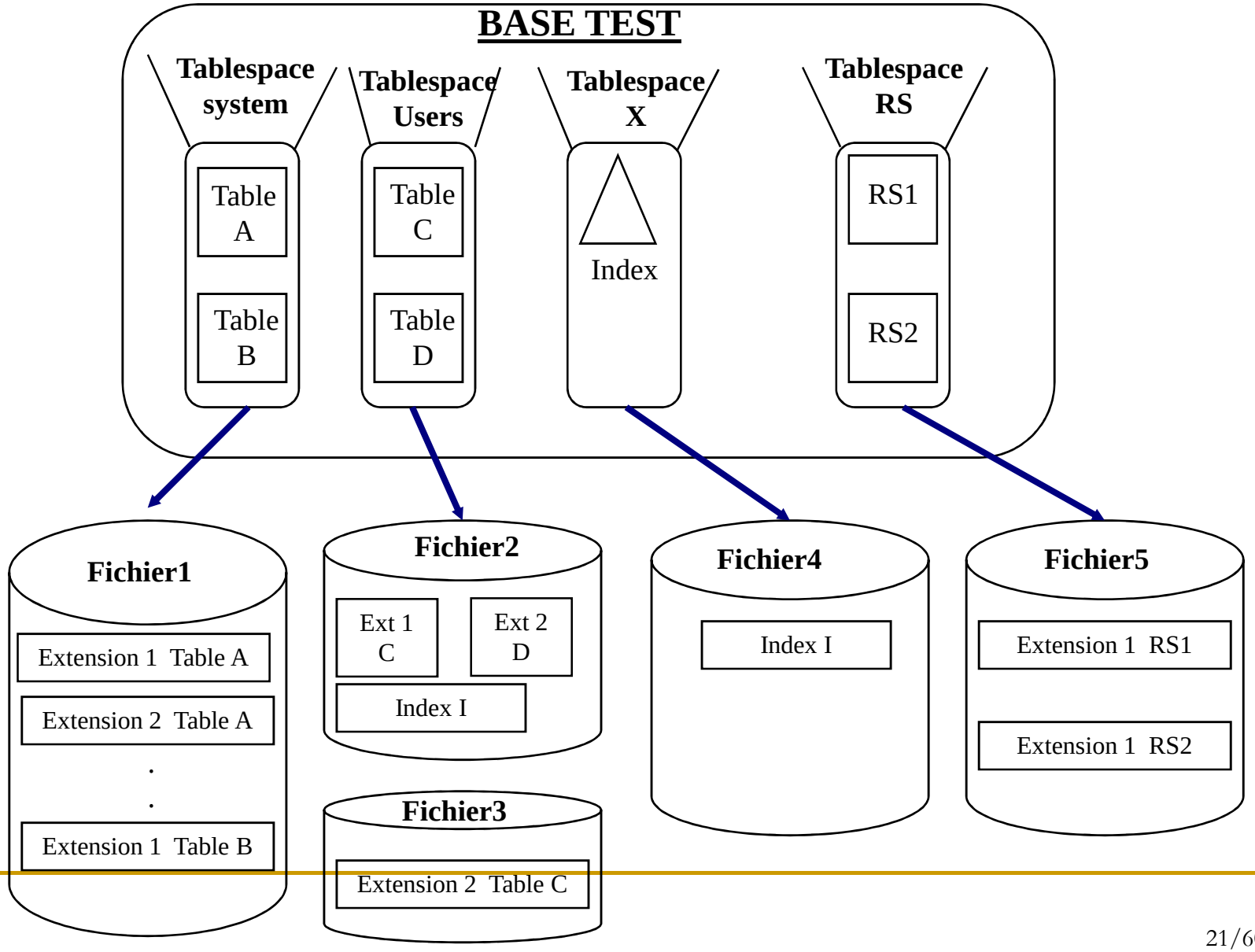
- Ce type de fichiers contient l'ensemble des changements effectués sur la DB (l'ensembles des transactions sur la BD).
- Grâce à un mécanisme de récupération, ces fichiers permettraient alors de reconstituer (de façon consistante) au mieux la BD après une panne brutale
- **Les fichiers de contrôles (Control files):**

  Ces fichiers permettent à une instance de serveur d'utiliser les fichiers précédemment cités : en effet, ils stockent les emplacements physiques des fichiers, leur nature, ...

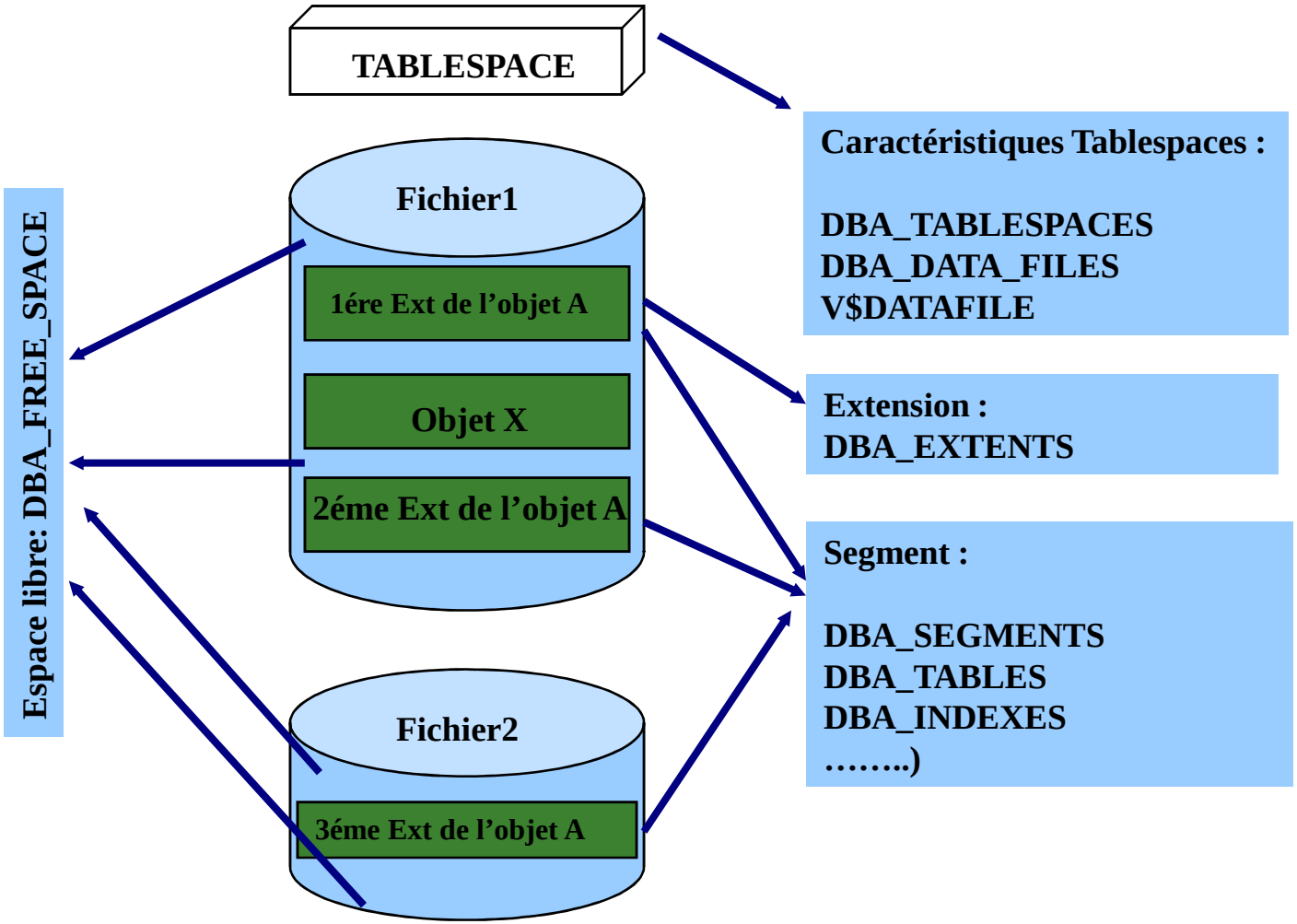
STRUCTURE LOGIQUE D'UNE BASE DE DONNEES ORACLE



STRUCTURE LOGIQUE ET STRUCTURE PHYSQIUE



STRUCTURE LOGIQUE ET STRUCTURE PHYSQIUE



utilisation de la mémoire

- La structure interne de toute instance de la BD est constituée de plusieurs parties distinctes:
  - La première est la **SGA** (System Global Area): représente l'ensemble de la mémoire vive utilisée en tant que buffer ou tampon sur l'espace disque de la BD.
  - Les autres correspondent à un ensemble de **processus** qui se partagent les diverses tâches que se doit de réaliser une instance du serveur Oracle

System Global Area (SGA)

- Le SGA est l'une des parties clés que se doit de contrôler tout administrateur
- De cet espace mémoire dépend une utilisation acceptable ou non de la BD en raison de :
  - cet espace sert de mémoire cache pour le disque
  - Une bonne gestion permet de garder en mémoire vive les données les plus utilisées,
  - Ce qui assure alors un facteur d'accélération non négligeable

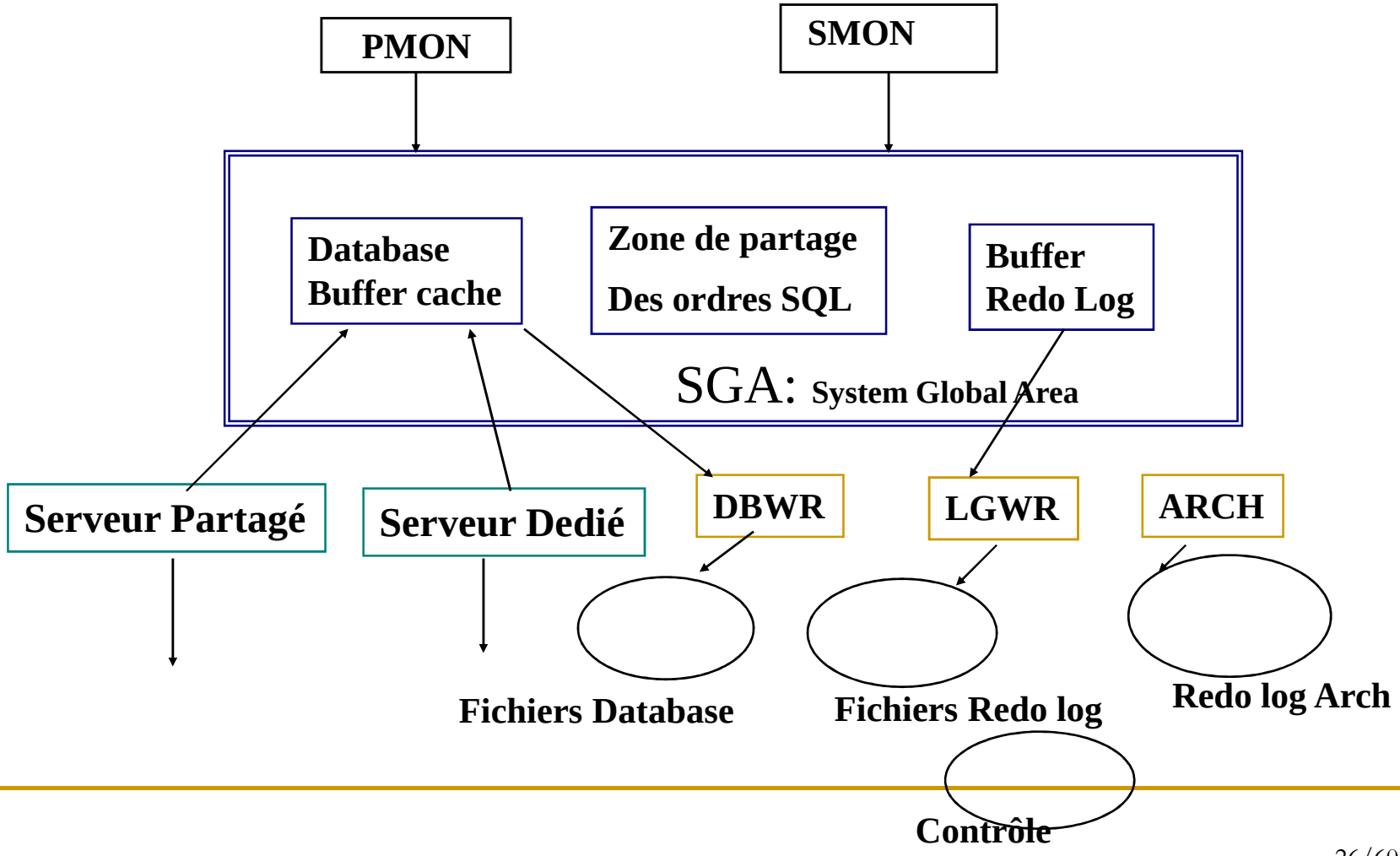
Structure interne du SGA

- Le SGA est constitué de plusieurs tampons (ou buffers) dont les plus important sont:
  - tampon database,
  - le tampon RedoLog
  - les curseurs
  - la librairie
  - le dictionnaire,

Propre à chaque instance du serveur

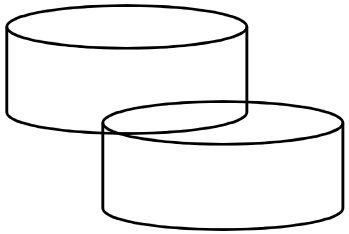
partagée par toutes les instances du serveur.

La SGA dans une BD Oracle

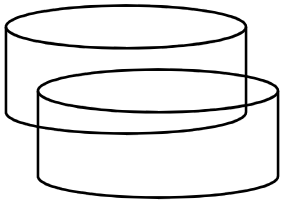


La SGA dans une BD Oracle

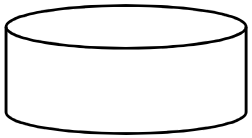
- **Les fichiers**



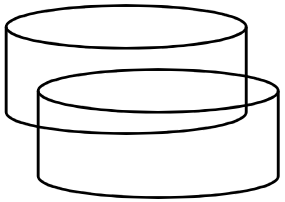
Fichiers de la Base De données



Fichiers Redo Log



Fichiers de contrôle



Fichiers Log Archivés

- **La SGA**

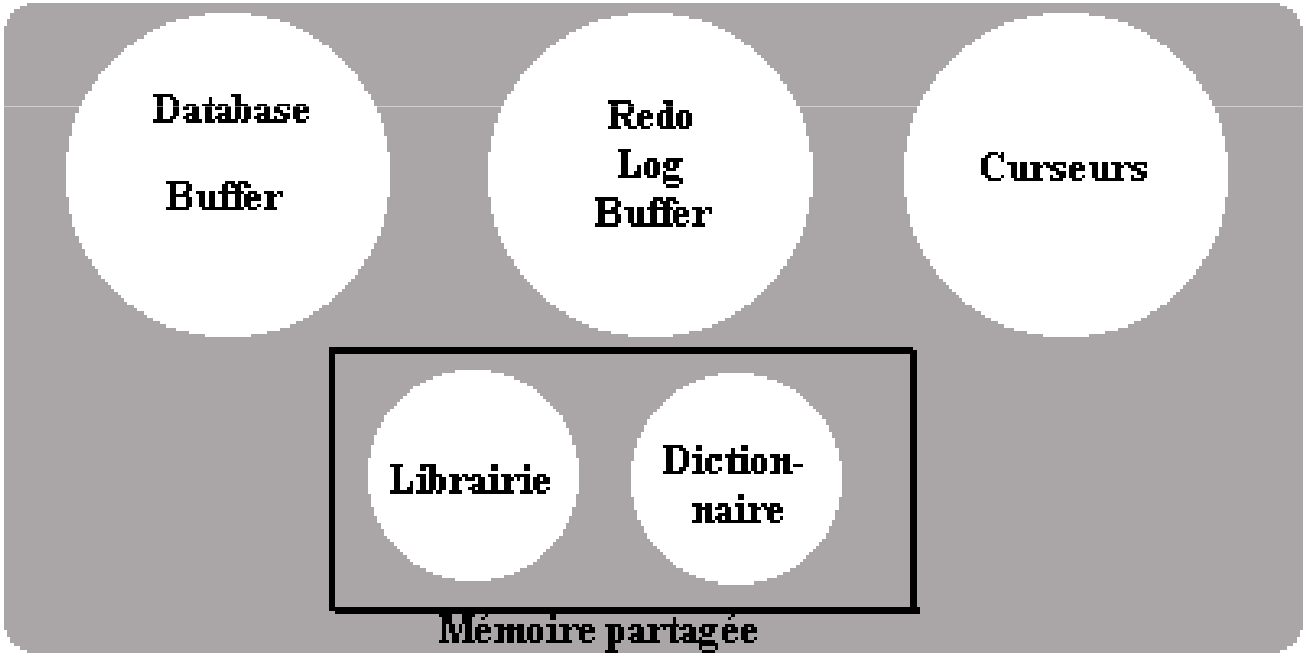
- **Les Process**

DBWR **LGWR** **ARCH** **PMON** **SMON**

USER **S.DEDIE**

Structure interne du SGA

Le plus intéressant des tampons du SGA pour l'administrateur est le **tampon Database**



Le tampon database

- Sert pour la transition de la majorité des informations durant le temps d'exécution d'une instance
- Ce tampon se divise en deux parties : en fait, des listes contenant des éléments chaînés entre eux :
  - La **LRU** (pour Least Recently Used)
  - La liste **dirty**

La LRU

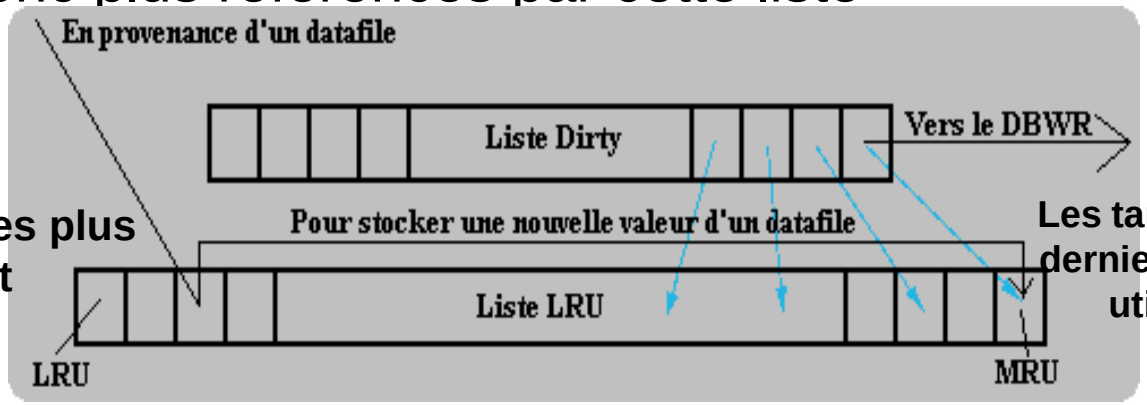
- contient une série d'éléments (des petits tampons) qui sont triés dans un ordre bien précis : des plus anciennement utilisés (Least Recently Used) aux plus récemment utilisés (Most recently Used : MRU) en sélection ou modification
- Les éléments ont trois états possibles :
  - l'état **free:** ce tampon n'est pas référencé par la liste **dirty** et donc les données contenues ne nécessitent pas d'être mises à jour dans un datafile,
  - l'état **pinned** :les données de ce tampon sont en cours d'utilisation pour un traitement quelconque
  - l'état **dirty** : une donnée de ce tampon est plus récente que son homologue sur le datafile requis, en conséquence de quoi il faudra mettre à jour ce tampon sur le datafile.

La dirty et traitement

- référence l'ensemble des tampons notés dirty de la liste LRU
- la liste dirty est stockée sur le disque sur ordre d'un processus gérant cette activité : ce processus se nomme DBWR (DataBase WRiter)
- Si une donnée est requise pour un traitement deux cas:
  - soit le SGA ne contient pas cette donnée,
  - soit il la contient (présente dans la liste LRU).
- Si l'on est dans le premier cas, il faut donc aller la chercher dans un datafile : elle sera stockée dans le tampon le plus anciennement utilisé, noté free, de la liste LRU
- Une fois que la donnée y est stockée (si on était dans le second cas, elle y était déjà), le tampon passe en tête de la même liste
- Ce tampon passe alors de l'état **free** à l'état **pinned**

Traitement

- Si le traitement modifie ce tampon (pined), il est alors référencé par la liste **dirty** et le tampon prend alors l'état du même nom.
- Sinon il repasse dans l'état **free**.
- Quand le DBWR a fini de transférer les données des tampons de la liste **dirty**, ces tampons repassent dans un état **free** et ne sont donc plus référencés par cette liste



Les tampon les plus anciennement utilisés

Les tampon les derniers a être utilisés

Le tampon Redo log

- Ce tampon stocke l'ensemble des dernières modifications effectuées sur la BD, avant qu'elles ne soient définitivement écrites sur les redo log files
- permet d'optimiser les performances du système
- le processus LGWR prend la décision de transférer les données du tampon Redo log vers les redo log files ,

Les processus d'une instance

- les processus (des programmes) se chargent de manipuler toutes les données du SGA (ainsi que celles stockées sur les fichiers) pour faire fonctionner le système
- Il existe plusieurs processus dont le DBWR et le LGWR sont les plus importants
- **Notions de Program Global Area (PGA):**
  - Un PGA est associé à un processus (et inversement).
  - Un PGA sert à temporiser (bufférisér) les données que manipule le processus, toujours dans un souci d'optimisation

Les processus serveurs

- Fonctionne sur la machine serveur
- Prend en charge un processus utilisateur
- Utilise une PGA exclusive
- Inclut L'oracle Program Interface (OPI)
- Traite les appels générés par le client
- Envoie les résultats aux clients

Les processus utilisateurs

- Fonctionne sur la machine du client
- Démarre lors de l'appel d'un outil ou d'une application
- Exécute l'outil ou l'application (SQL Plus,Server Manager,OEM,Dev 2000)
- Inclut l'UPI (User Program Interface)
- Appelle le serveur Oracle
- **Ces deux processus sont créés lors de toutes connexions de la part des utilisateurs.**

Les processus dispatchers

- Ces processus sont optionnels. En effet, ils n'existent que si le serveur Oracle est configuré en mode multi-threads. S'il existe, ils servent d'interprète entre les processus utilisateurs et les processus serveurs.

Le processus DataBase WRiter (DBWR)

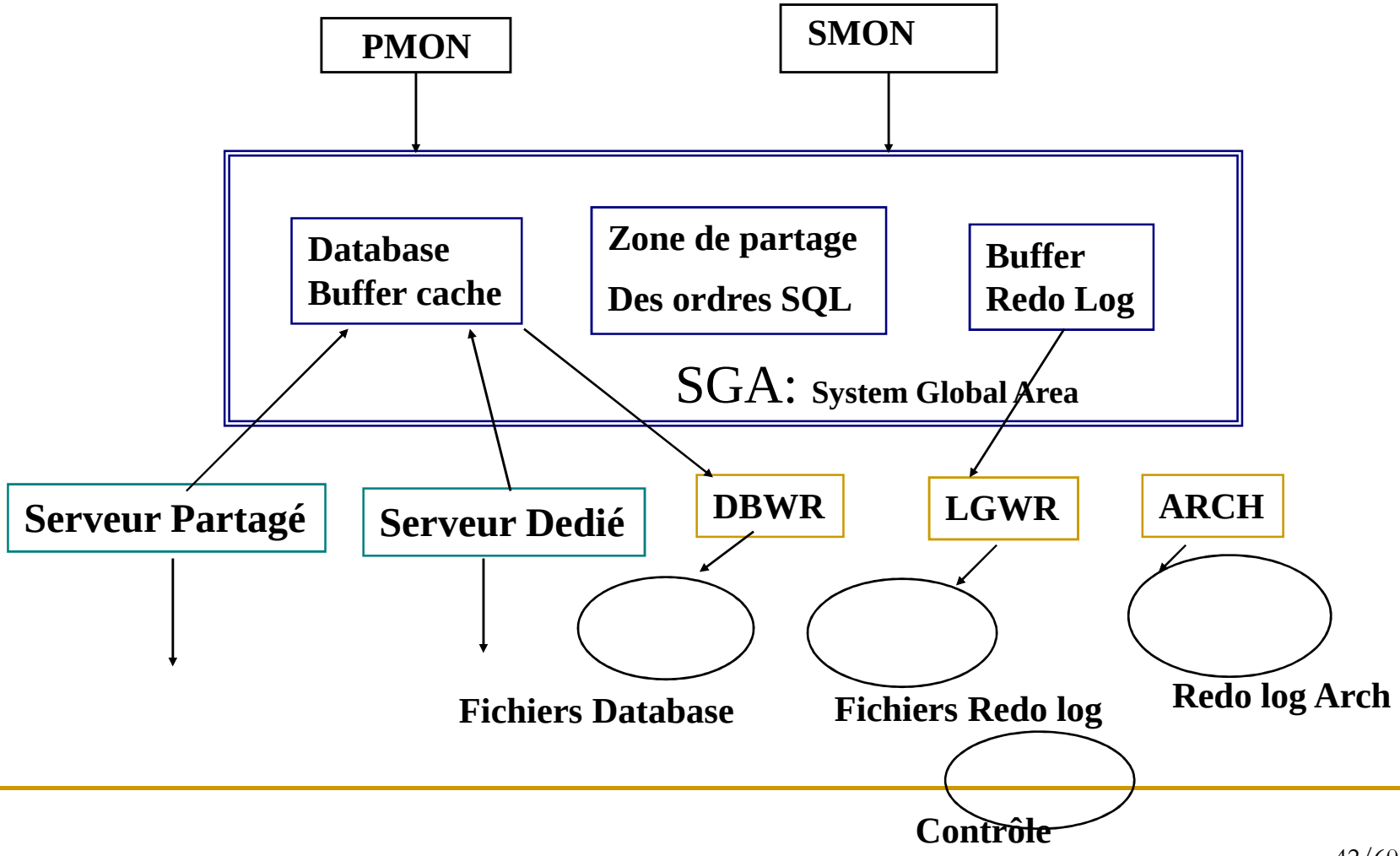
- ce processus se charge de stocker les données du tampon database (du SGA) sur les datafiles nécessaires, et ce à des moments précis : les checkpoints. Ses points de synchronisation sont en fait déterminés par un autre processus : le processus CheckPoinT

- **Le processus LoG WRiter (LGWR)**
  - ce processus contrôle la mise à jour des données des Redo log files à partir du tampon Redo log contenu dans le SGA.
- **Le processus ChecK PoinT (CKPT)**
  - Ce processus est chargé de coordonner la mise à jour des fichiers de données et de contrôle, ce à partir des informations contenues dans le SGA. Cette mise à jour se fait à des moments précis appelés check points. Ce processus est cependant optionnel. Dans le cas où il n'existe pas, c'est le processus LGWR qui se charge de réaliser la sauvegarde

- **Le processus System MONitor (SMON)**
  - Ce processus, père de tous les processus de l'instance, s'occupe de plusieurs tâches. Il se charge notamment d'optimiser l'utilisation de la mémoire dans le système. Il se charge aussi d'assurer la reprise du système lors de tout démarrage d'une instance.
- **Le processus Process MONitor (PMON)**
  - Ce processus se charge notamment de libérer toutes les ressources acquises par un processus client, lorsque celui-ci se termine. Il est aussi chargé de surveiller les processus serveurs et les processus dispatchers : si l'un d'eux s'arrêtait anormalement, le PMON se chargerait de libérer les ressources de ce processus et de le relancer.

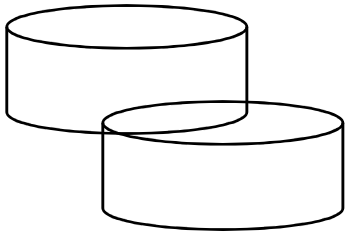
- **Le processus ARCHiver (ARCH)**
  - Ce processus sert à effectuer un archivage des fichiers de Redo log : en effet, si vous n'avez que deux redo log files (leur taille est fixe) et que le tampon Redo log est plein, un des deux fichiers doit être écrasé pour contenir les nouvelles données. Ce processus, s'il est activé (car il est optionnel), permet de palier ce problème. En effet il archivera l'ancien redolog file. Pour que le processus existe, l'instance doit être en ARCHIVELOG mode.
- **Le processus RECOverer (RECO)**
  - Ce processus, très complexe, est utilisé dans le cadre d'une BD distribuée.
- **Les processus LoCK (LCKn)**
  - Ces processus ne sont utilisés que dans un usage du serveur en mode parallèle. Ils servent à gérer des verrous entre les diverses instances de la BD

ARCHITECTURE D'UNE BASE DE DONNEES

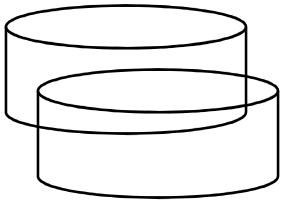


LES COMPOSANTS D'ORACLE

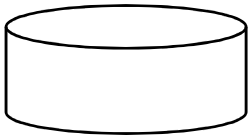
- **Les fichiers**



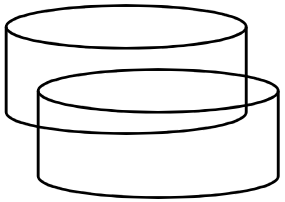
Fichiers de la Base De données



Fichiers Redo Log



Fichiers de contrôle



Fichiers Log Archivés

- **La SGA**

- **Les Process**

DBWR **LGWR** **ARCH** **PMON** **SMON**

USER **S.DEDIE**

Administration de l'espace disque

- C’est une tâche administrative liée à la gestion des supports de stockage
- comment fait-on quand la base de données est pleine ?
- Quelques recommandations importantes:
  - utilisez judicieusement les ressources de la machine
  - Dupliquez l'information sur divers disques différents
  - séparez les données des index afin que les disques puissent travailler simultanément
  - Il est fortement déconseillé de stocker les fichiers de données et les fichiers de Redo log sur le même disque
  - Séparez les données du tablespace SYSTEM. Celui-ci doit se concentrer sur le dictionnaire de données

- **Dupliquer les fichiers de contrôle :**
  - Cela se fait une fois pour toute, dans le fichier d'initialisation.
  - **CONTROL_FILES = '/disk1/base/base_control.ctl', '/disk2/base/base_control.ctl'**
- **Dupliquer les fichiers de Redo log**
  - Cela se fait lors de la création de la base.
  - **CREATE DATABASE base LOGFILE**
    GROUP 1 ('/disk1/base/base_log1.log', '/disk1/base/base_log1.log') SIZE 30K,
    GROUP 2 ('/disk1/base/base_log2.log', '/disk1/base/base_log2.log') SIZE 30K, ...;
  - **Dupliquer les fichiers de données**

Administrer les tablespaces

- **Ajout d'un tablespace**
- Deux manières pour ce faire:
  - soit utiliser le programme **Storage Manager**,
  - soit rentrez la requête correspondante par ligne de commande.
- par **Storage Manager**: il suffit de choisir le champs **create** du menu tablespace et de suivre les instructions.
- Une fois la manipulation terminée, la requête est générée. C'est celle-ci que vous auriez pu rentrer en mode ligne de commande.
- En voici un exemple:
  - **CREATE TABLESPACE essai DATAFILE 'essai1.ora' SIZE 20M, 'essai2.ora' SIZE 30M;**

- **Suppression d'un tablespace** :
  - **DROP TABLESPACE tableSpaceName INCLUDING CONTENTS**;
- **Renommage** :il n'est pas possible de directement renommer un tablespace. Pour ce faire, il faut : exporter tous les objets du tablespace - supprimer le tablespace - créer un nouveau tablespace - y importer les objets.
- **Remarque**: une fois le tablespace supprimé, les fichiers, bien que plus utilisés, ne sont pas supprimés du disque. Il faut alors faire appel aux commandes de suppression de fichiers de l'OS. Méfiez vous de ne pas supprimez un mauvais fichier

Administrer les fichiers de données

- Règle générale: il est plus simple pour le système de gérer un gros fichier plutôt que plusieurs petits (surtout que les OS limitent les nombres de fichiers utilisés par processus).
- Ajout d'un datafile dans un tablespace :
  - **ALTER TABLESPACE tablename ADD DATAFILE 'filename' SIZE 1M;**
- **Remarques**:
  - il vous faut impérativement les privilèges ALTER TABLESPACE
  - si le chemin d'accès du fichier n'est pas complet, c'est le répertoire courant de la BD qui est utilisé.

- **Redimensionnement d'un datafile** :Il est toujours possible d'augmenter la taille d'un datafile (s'il reste de la place sur le disque, bien entendu). Par contre on ne peut pas toujours la diminuer.
  - **ALTER TABLESPACE tablename DATAFILE 'filename' RESIZE 150M**;
- **Extension automatique d'un fichier:** rendre un fichier automatiquement extensible quand celui-ci est plein. Exemple:
  - **ALTER TABLESPACE tablename ADD DATAFILE 'filename' SIZE 1M; AUTOEXTEND ON NEXT 1M MAXSIZE 1024M;**

- Pour figer un fichier auto-extensible utilisez la commande :
  - **ALTER DATABASE DATAFILE 'filename' AUTOEXTEND OFF Fichiers offline et fichiers online**
- Il est possible de rendre un fichier inutilisable puis de le rendre à nouveau utilisable:
  - **ALTER DATABASE DATAFILE 'filename' ONLINE;ALTER DATABASE DATAFILE 'filename' OFFLINE DROP**;

Administrer les datablocks

- **Quelle taille choisir pour les data block?**
  - si vous avez beaucoup de petites transactions, il est préférable de choisir une petite taille.
  - S'il y a peu de transactions mais de taille conséquente, optez pour une taille un peu plus importante.
  - Remarque: Une fois cette taille fixée (dans le fichier d'initialisation : **db_block_size**) cette taille ne pourra plus être changée.

- **Pctused et Pctfree :**
- Ces deux paramètres sont bien utiles pour optimiser les temps d'accès aux fichiers. Nous allons donc voir en quoi ils peuvent jouer dans cette optimisation et comment les utiliser.
- **PCTFREE** (percent free) : ce paramètre fixe le pourcentage de place vide laissée dans un block pour d'éventuels ajouts ou modifications de la table (valeur par défaut 10%).
- **PCTUSED** (percent used) : celui-ci fixe la taille maximum pour que le block appartient à la free list (valeur par défaut 40%).
- **Free list** : liste de blocks susceptible d'être utilisés pour l'ajout de données.

comment tout cela fonctionne?

- Supposons qu'une valeur change dans une ligne (et soit plus grosse en terme de place mémoire). S'il n'y avait pas de place vide dans le bloc (nommé 1), les données de la ligne seraient déplacées dans un nouveau block 2 qui pourrait contenir la ligne nouvellement modifiée (si le 1 contenait d'autres lignes). On appelle cela migration. La migration coûtant du temps, il est préférable de laisser de la place.
- Il faut éviter le chaînage entre blocks pour une même ligne. PCTUSED et la free list sont là dans ce but. Un bloc ne peut rentrer dans la liste seulement s'il est utilisé à **moins** de PCTUSED pourcent.On évite ainsi aussi quelques chaînages coûteux pour les consultations.
- **CREATE TABLE (...)**
  PCTUSED value
  PCTFREE value
  TABLESPACE tablespaceName;

Administrer les extents

- Lorsque de la place manque dans un segment, un nouvel **extent** est recherché (ce qui, d'ailleurs, coûte du temps).
- Vous pouvez fixer certains paramètres (d'un tablespace par exemple) afin d'optimiser au mieux les différentes phases de recherches.
- Pour ce faire, on agit, par exemple, sur les commandes **CREATE TABLESPACES ou ALTER TABLESPACE** :

```
ALTER TABLESPACE "SYSTEM"
 DEFAULT STORAGE (
 INITIAL 100k
 NEXT 100k
 MINEXTENTS 1
MAXEXTENTS 10);
```

Conclusion

- Au terme de ce chapitre, vous voyez donc mieux comment gérer la répartition des données sur le disque. Nous sommes aussi à même de contrôler quelques aspects d'optimisations. Le chapitre suivant pousse plus en avant les choses.