



The background image is a collage of PHP code snippets. A magnifying glass is positioned over a line of code that reads: `$sql.=" OR ";` followed by `$sql.="contenu LIKE '%$mots%'` and `$i++;`. This represents a SQL injection attack. Other visible code includes `$q=$_GET['q'];`, `$s=explode(" ", $q);`, `$sql="SELECT * FROM news";`, `$i=0;`, `foreach($s as $mot) {`, `if(strlen($mot)>3) {`, `if(isset($_POST['pnp']))`, `require_once('db.php');`, `$q=$_GET['q'];`, and `$s=explode(" ", $q);`.

Administration d'Oracle 10g



ORACLE®



Introduction

❑ Oracle: Vue d'ensemble

- Oracle est le leader du marché des SGBDR, avec une part de marché allant jusqu'à 48.6% en 2007 (Gartner Group).
- Oracle commercialise différents produits; Oracle Database, Oracle Developer Suite, Oracle Application Server, Oracle Applications, Oracle Collaboration Suite, Oracle Services...



Introduction


❑ Oracle Database

- Le produit principal d'Oracle est passé depuis 1977 de la version 1 à la version 11g en 30 ans. Chaque version est commercialisée sous différentes éditions:
 - Enterprise Edition: inclut toutes les fonctionnalités d'Oracle.
 - Standard Edition: basique, destinée aux serveurs à 4 processeurs.
 - Standard Edition One: basique, destinée aux serveurs biprocesseurs.
 - Personal Edition: uniquement sur Windows, destinée aux développeurs.
 - Express Edition: édition gratuite, fonctionne sur des machines à 1 processeur.
 - Lite Edition: destinée aux machines mobiles.

Introduction

❑ Interaction avec Oracle Database





Architecture d'Oracle Database

❑ Composantes principales d'un serveur Oracle

1. Base de données

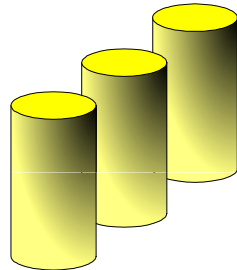
Ensemble de fichiers (sur disque) contenant les données, les informations sur les données, le journal de modifications

2. Instance

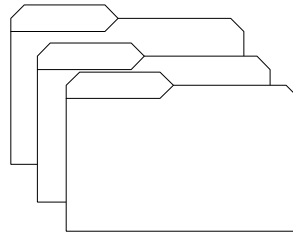
Ensemble de processus et de zones mémoires (mémoire centrale) qui permettent de gérer la base de données.

Base de données

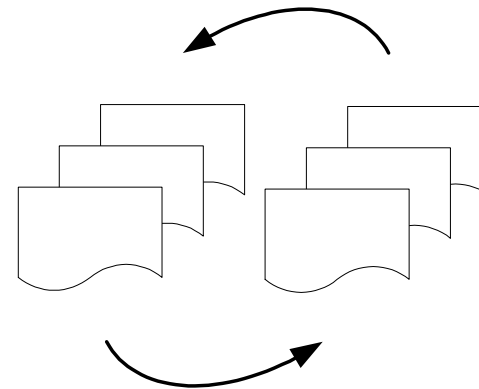
Base de Données



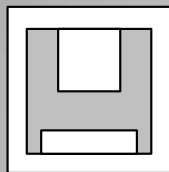
Fichiers de données



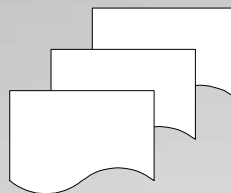
Fichiers de contrôle



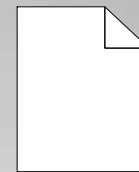
Groupes des fichiers de
journalisation



Fichier de paramètres



Archives des fichiers de
journalisation



Fichier de mot de
passe



Fichier de contrôle

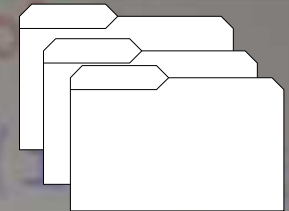
❑ Fichier de contrôle (Control File)

- Contient des informations de contrôle sur la base:

Nom de la base, noms et chemins des fichiers de données et de journalisation, informations de restaurations etc...

- Premier fichier lu par l'instance lors du démarrage.

- La vue **V\$CONTROLFILE** nous renseigne sur le contenu du fichier de contrôle.



Fichier de contrôle



Fichier de données

❑ Fichier de données (Data Files)

- Stockent les données sous un format spécial à Oracle.
- Physiquement, un fichier de données est un ensemble de blocs SE. Un bloc SE constitue l'unité d'E/S (écriture/lecture) des fichiers.
- Les fichiers de données sont logiquement regroupés sous forme de **tablespaces**.
- Une BD Oracle 10g inclut au moins deux tablespaces, **SYSTEM** et **SYSAUX**.

Fichiers de données



Les tablespaces

❑ Les tablespaces

- Un *tablespace* est une unité logique qui correspond physiquement à un ou plusieurs fichiers de données.
- L'administrateur (DBA) agit sur les tablespaces et non sur les fichiers de données.
- Une BD est organisée sous forme de plusieurs tablespaces, chacun correspondant à un contexte (thème).
- EX: On peut créer plusieurs tablespaces dans une BD d'une E^{se} commerciale qui gère la FACTURATION, la GRH, et le PARC INFO.



Les tablespaces

- Le tablespace FACTURATION → fact01.dbf et fact02.dbf
- Les tables FACTURE, LIGNE_FACTURE, PRODUIT, CLIENT, REGLEMENT.
- **Le tablespace GRH → grh01.dbf**
- **Les tables PERSONNEL, PAIE, SANCTION etc.**
- Le tablespace PARC → parc01.dbf
- Les tables EQUIPEMENT, CATEGORIE, REPARATION, MAINTENANCE etc.
- **Avantage:** On peut administrer par tablespace, et donc par partie (par rapport à la BD). Par exemple, pour maintenir certaines tables relatives à la facturation, on peut mettre uniquement le tablespace FACTURATION en offline, au lieu de rendre toute la base indisponible ce qui touchera des centaines d'utilisateurs qui ont besoin de manipuler les données de la GRH, PARC INFO etc.



Les tablespaces

❑ Les tablespaces

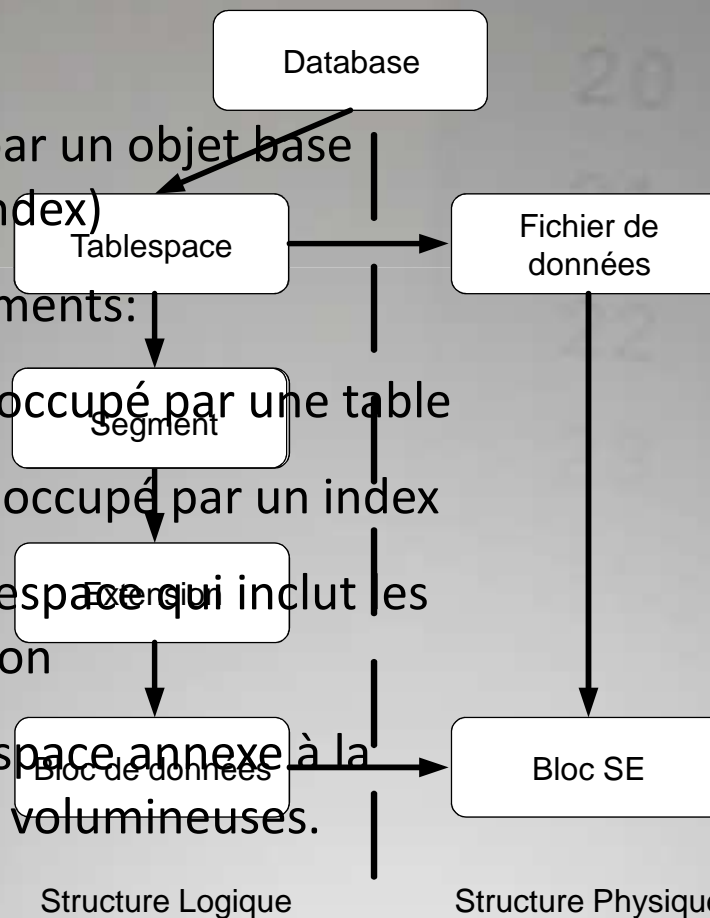
- Les vues `DBA_TABLESPACES` et `DBA_DATA_FILES` incluent toutes les informations relatives aux tablespaces et aux fichiers de données de la base.

- Pour afficher les noms des fichiers de données ainsi que les tablespaces auxquelles ils correspondent:

```
SELECT tablespace_name, file_name  
FROM DBA_DATA_FILES  
ORDER BY tablespace_name;
```


Structure de stockage

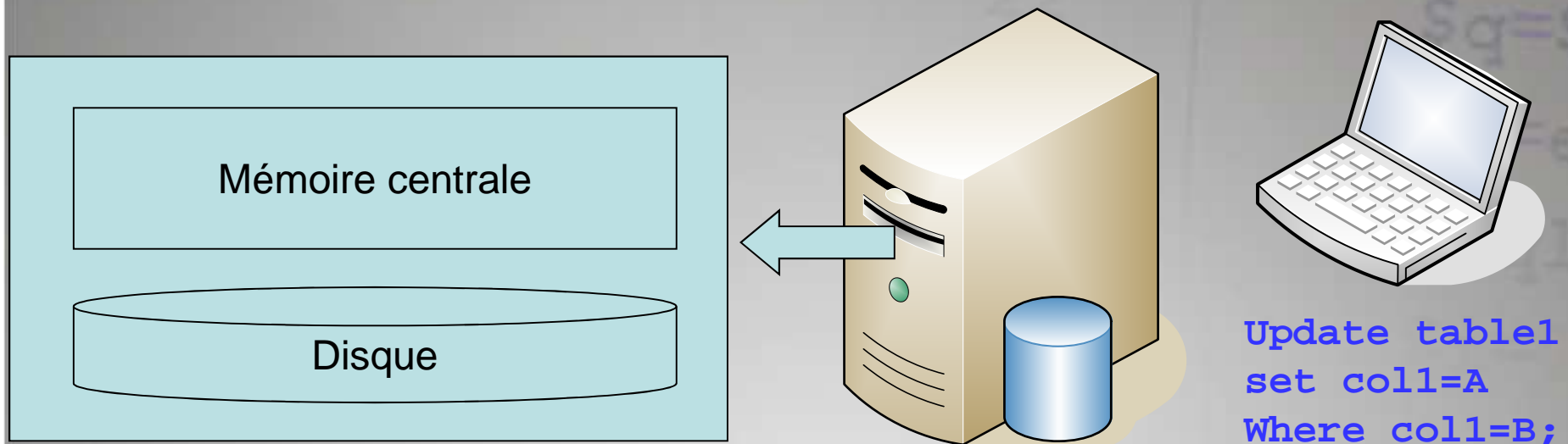
- C'est l'espace occupé par un objet base de données (Table ou Index)
- Il existe 4 types de segments:
 1. Segment table: espace occupé par une table
 2. Segment index: espace occupé par un index
 3. Segment d'annulation: espace qui inclut les informations d'annulation
 4. Segment temporaire: espace annexe à la MC pour les opérations volumineuses.



Le segment d'annulation

❑ **Le segment d'annulation** sert à stocker les données nécessaires:

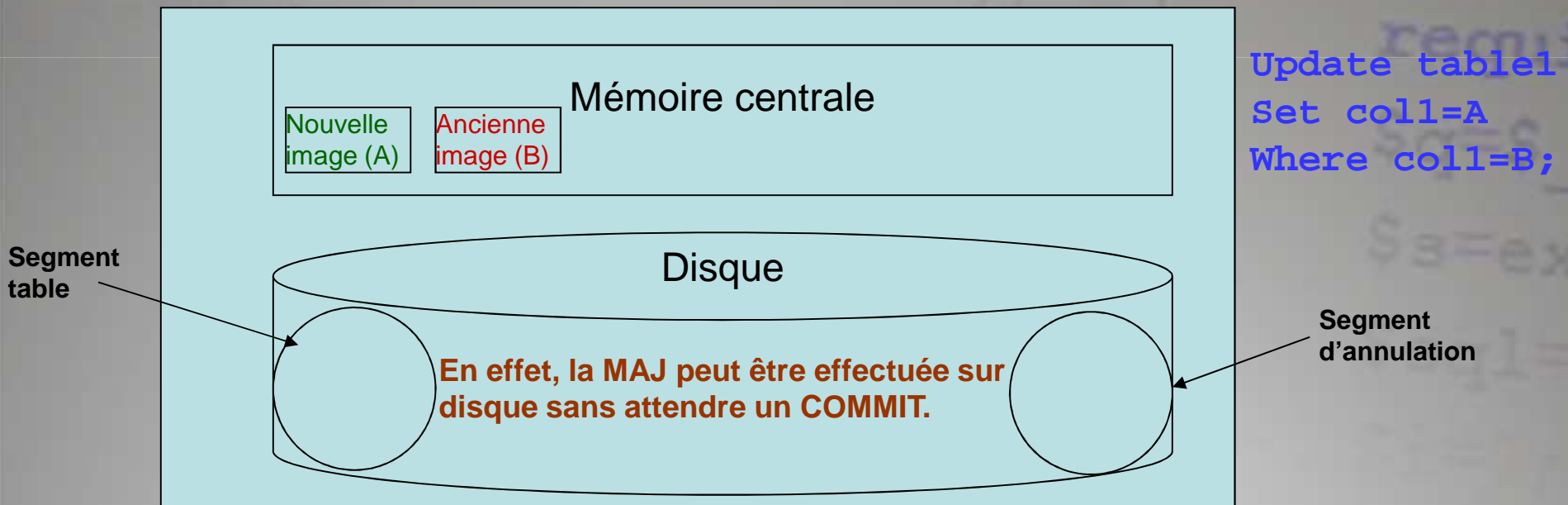
1. A l'annulation




Le segment d'annulation

❑ **Le segment d'annulation** sert à stocker les données nécessaires:

1. A l'annulation



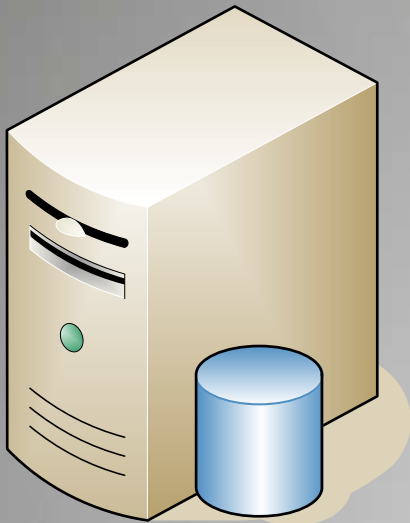
La nouvelle image est enregistrée dans un segment table, l'ancienne dans un segment d'annulation. Au cas où un ROLLBACK est effectué, c'est cette image qu'on utilisera pour rétablir les données.



Le segment d'annulation

❑ **Le segment d'annulation** sert à stocker les données nécessaires:

1. A l'annulation
2. A la lecture cohérente (ou dite aussi consistante).




And the value
ou ancienne?

12:01
Select col1 From table1
Where col2=C;



11:58
Update table1
set col1=A
Where col2=C;



Le segment d'annulation

❑ **Le segment d'annulation** sert à stocker les données nécessaires:

1. A l'annulation

2. A la lecture cohérente (ou dite aussi consistante).

- a. Si **pas d'anticipation d'enregistrement des MAJ**, Alors **la lecture se fait directement à partir du segment table**.
- b. Si **anticipation d'enregistrement des MAJ**, Alors **la lecture se fait à partir du segment d'annulation**.



Le segment temporaire

❑ **Le segment temporaire** sert à stocker les données relatives à des opérations volumineuses si la mémoire centrale ne suffit pas à les exécuter.

Exemple d'opérations volumineuses:

- Certains tris.
- Certaines jointures.
- Création d'index.
- Etc.

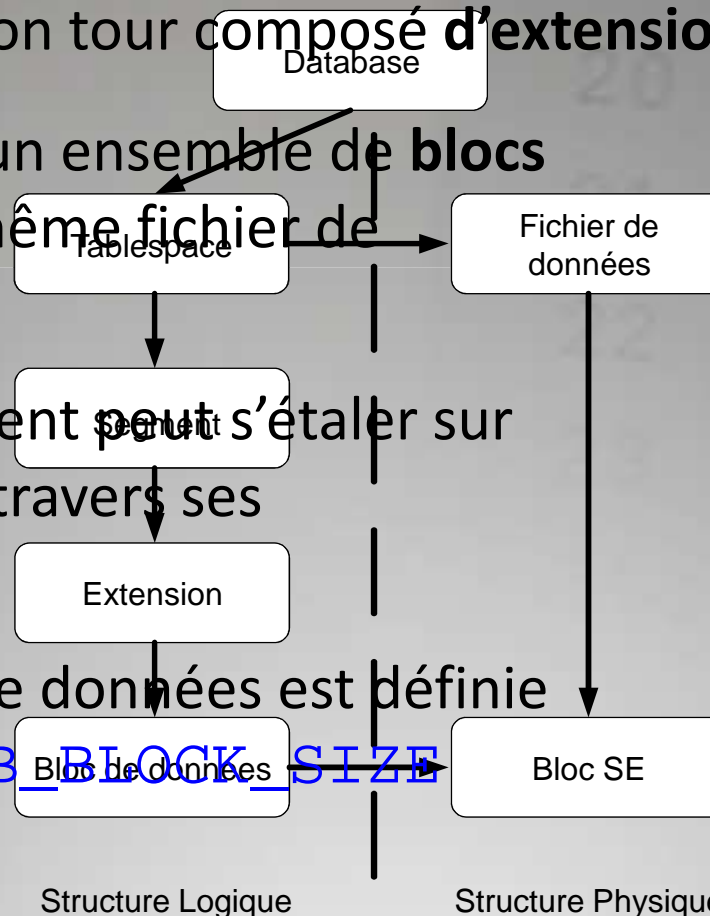
Extension

❑ Un **segment** est à son tour composé d'**extensions**

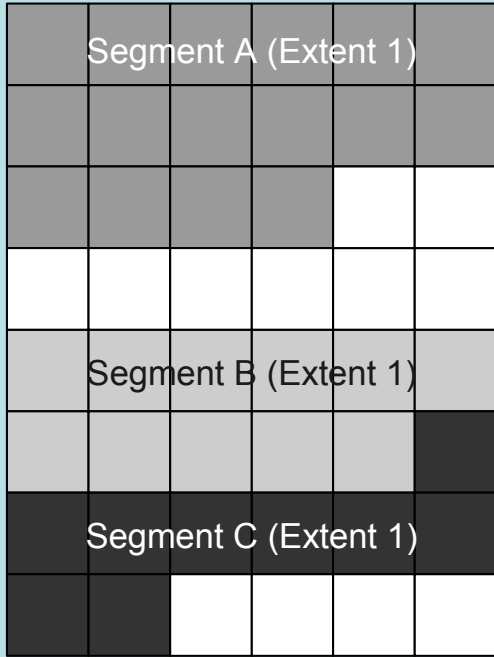
❑ Une **extension** est un ensemble de **blocs** appartenant à un même **fichier de données**

❑ Par contre un segment peut s'étaler sur plusieurs fichiers à travers ses extensions.

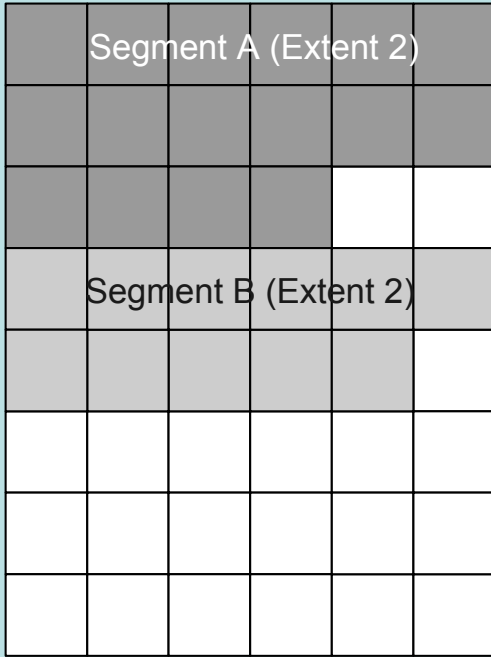
❑ La taille d'un bloc de données est définie par le paramètre **DB_BLOCK_SIZE**



Example

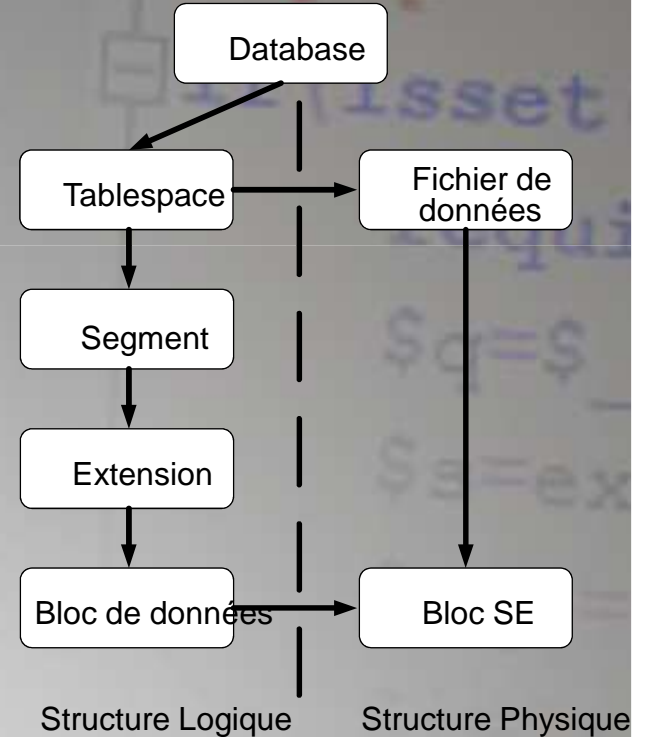



Fact01.dbf



Fact02.dbf

Structure du tablespace FACTURATION





Fichiers de journalisation

- ☐ Les **fichiers de journalisation** contiennent toutes les modifications effectuées sur les données depuis une certaine durée.
- ☐ En cas de crash du système, ou d'altération des fichiers de données, on peut reconstituer les données à partir des fichiers journaux.
- ☐ L'écriture sur les fichiers journaux est **multiplexée** et **cyclique**.
- ☐ L'ensemble des fichiers multiplexés (qui contiennent donc les mêmes informations) sont appelés **membres** et forment un **groupe**.
- ☐ L'écriture est donc **multiplexée** à l'intérieur d'un groupe, et **cyclique** entre les groupes.
- ☐ La vue **V\$LOGFILE** contient les informations des fichiers journaux.

Fichiers de journalisation

Groupe 1

DELETE
UPDATE
INSERT INTO
UPDATE
UPDATE...
INSERT INTO

Membre 1

DELETE
UPDATE
INSERT INTO
UPDATE
UPDATE...
INSERT INTO

Membre 2

Groupe 2

INSERT INTO
UPDATE...
DELETE...

Membre 1

INSERT INTO
UPDATE...
DELETE...

Membre 2



Archives des fichiers journaux

- ☐ L'écriture des fichiers journaux est cyclique, ce qui fait qu'à un certain moment, certaines transactions seront écrasées.
- ☐ La solution, c'est d'archiver les fichiers journaux avant de les écraser.
- ☐ Les fichiers journaux archivés peuvent être stockés sur des disques (serveurs) distants, ce qui optimisera la sécurité de la BD.
- ☐ Une base de données n'est pas forcément en mode ARCHIVELOG, si elle ne l'est pas, les fichiers journaux ne sont pas archivés.



Le fichier de paramètres

- Consultable via la vue **V\$PARAMETER**

SELECT name, value FROM V\$PARAMETER

PFILE (parameter file)	SPFILE (server parameter file)
Fichier texte	Fichier binaire
Modifiable via un éditeur texte	Modifiable via SQL
Disponible sur la machine de démarrage	Centralisé (sur le serveur uniquement)
Nommé <code>init%.ora</code>	Nommé <code>spfile%.ora</code>
Modification à froid	Modification à chaud



Le fichier mot de passe

- ❑ Oracle doit authentifier le DBA avant que la base ne soit ouverte, et donc le mot de passe DBA est enregistré hors de la BD → dans un fichier de mot de passe
- ❑ L'authentification du DBA se fait ou bien via le **système d'exploitation** (`REMOTE_LOGIN_PASSWORDFILE=none`), ou bien via le **fichier de mot de passe** (`REMOTE_LOGIN_PASSWORDFILE=exclusive` ou `shared`).
 - `none` : Authentification via le SE uniquement
 - `exclusive` : utilisation d'un Fichier MDP dédié à une seule BD. Possibilité de donner le role SYSDBA ou SYSOPER à d'autres utilisateurs
 - `shared` : utilisation d'un FMP partagé par plusieurs BDs. Seul SYS peut utiliser les privilèges SYSDBA ou SYSOPER

Diapositive 24

MAB1

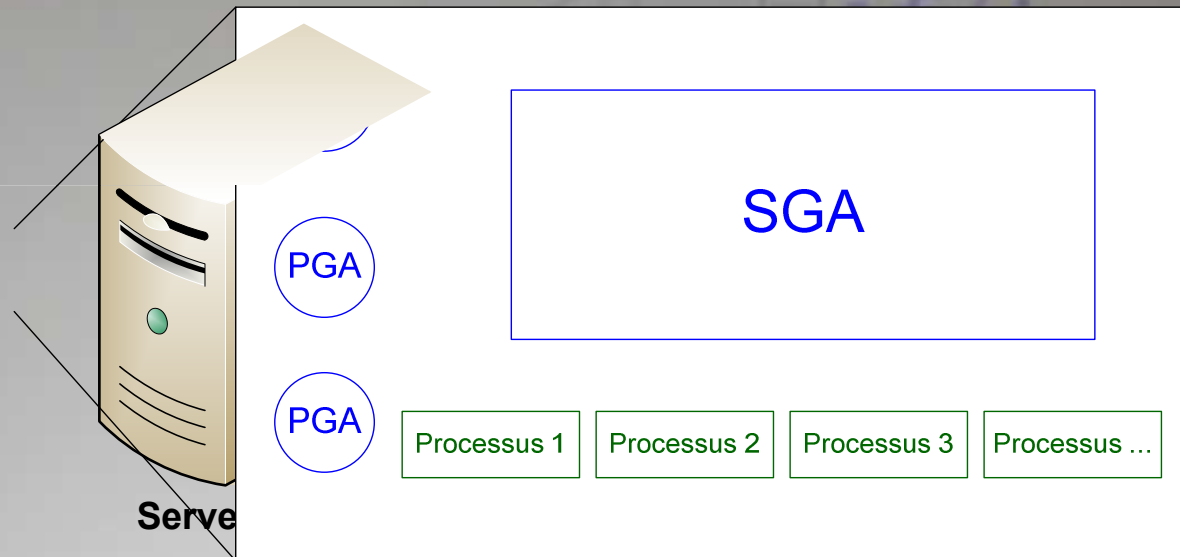
À ajouter dans le cours détaillé
Mohamed Anis BACH TOBJI; 21/10/2011



L'instance

- ☐ C'est l'ensemble de **structures mémoire** et de **processus** qui assurent la gestion de la base de données.
- ☐ Le fichier de paramètres est utilisé pour configurer l'instance lors de son démarrage.
- ☐ Une instance ne peut ouvrir qu'une seule base de données.
- ☐ Une instance emploie deux zones mémoire principales; la **SGA** (System Global Area) et la **PGA** (Program Global Area).

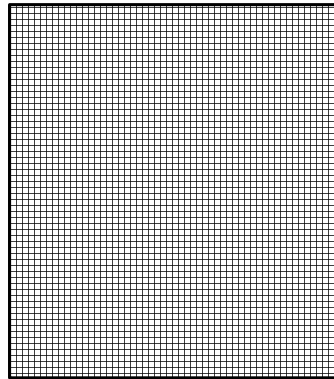
L'instance



Instance: la SGA

- ☐ C'est un **SGA** mémoire partagé par tous les processus de l'instance. Elle est arrêtée lors de son arrêt. Elle
- ☐ Plus la SGA est grande, plus la performance est maximale.
- ☐ Le paramètre `SGA_MAX_SIZE` définit la taille maximale de la SGA physique.

SGA

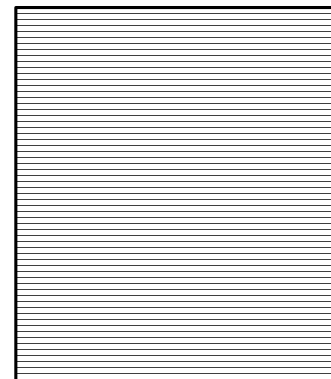


DB Buffer
Cache

Dictionary
Cache

Library
Cache

Shared
Pool



Redo Log
Buffer

Java Pool

Large Pool

Streams
Pool



Shared Pool

- ❑ Elle est composée de deux structures; le **library cache** et le **dictionary cache**. Sa taille est définie via le paramètre **SHARED_POOL_SIZE**.
- ❑ Le **LC** contient pour chaque requête récemment exécutée trois informations:
 - Son texte
 - Sa compilation
 - Son plan d'exécution
- ❑ Lorsqu'une requête existe encore dans le LC, Oracle ne perd pas son temps à la ré-exécuter.
- ❑ Le **DC** stocke toutes les informations nécessaires à l'analyse sémantique de la requête (table? Colonnes? Droits d'accès?...)



DB Buffer Cache

- ☐ Il stocke les blocs de données les plus récemment utilisés.
- ☐ Lorsqu'une requête est exécutée, Oracle vérifie d'abord si les blocs de données à ramener ne soient pas déjà chargés dans le DB buffer.
- ☐ Si les blocs à renvoyer n'existent pas dans le DB buffer cache, Oracle les y stocke selon l'algorithme LRU (et libère d'autres blocs s'il le faut).
- ☐ La taille du DB Buffer Cache est définie via le paramètre `DB_CACHE_SIZE`



Redo Log Buffer

- ❑ Contient les informations sur les transactions exécutées.
- ❑ Le Redo Log Buffer est écrit de manière séquentielle (les transactions y sont mélangées) et cyclique (taille limitée de la zone mémoire oblige).
- ❑ Chaque modification correspond à une entrée Redo. Cette dernière est constituée de plusieurs vecteurs, chacun correspond à un bloc de données modifié (ancienne + nouvelle valeur).
- ❑ Le contenu du Redo Log Buffer est écrit dans les fichiers journaux.
- ❑ La taille du Redo Log Buffer est définie via le paramètre `LOG_BUFFER`



Autres zones mémoire

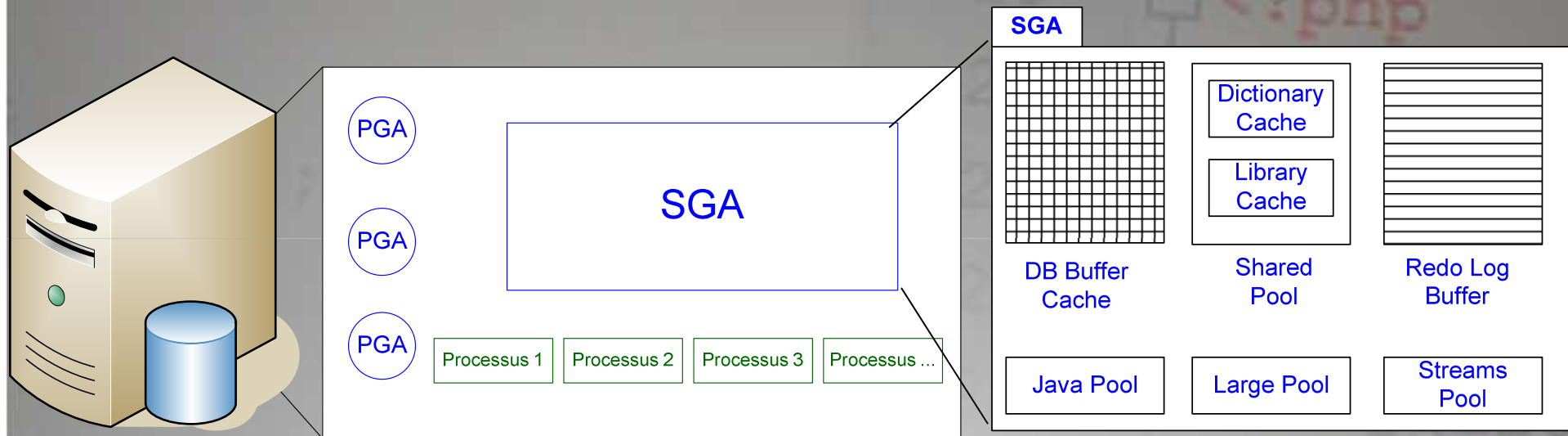
- ❑ Le **JAVA Pool** : Stocke les objets et applications Java les plus récemment utilisés lorsque la machine virtuelle Java JVM optionnelle est utilisée. Paramètre `JAVA_POOL_SIZE`.
- ❑ Le **Large Pool** : Stocke des données lors de l'exécution d'opérations volumineuses. Il est utilisé dans le cas d'une configuration serveur partagé. Paramètre `LARGE_POOL_SIZE`.
- ❑ Le **Streams Pool** : C'est le cache des données relatives à la queue de messages utilisées par Oracle. Paramètre `STREAMS_POOL_SIZE`.
- ❑ A noter que dans la SGA, il y a une petite zone mémoire appelée SGA fixe. Elle inclut des infos sur l'état de la base, l'instance, les verrous...
- ❑ Les vues `V$SGA` et `V$SGA_DYNAMIC_COMPONENTS`.



La gestion automatique de la SGA

- ☐ Les tailles des zones mémoire peuvent être définies manuellement ou automatiquement.
- ☐ Gestion manuelle : donner une valeur spécifique à chacun des paramètres des tailles des zones mémoire. La somme ne doit pas dépasser `SGA_MAX_SIZE`.
- ☐ Gestion automatique : activé si `SGA_TARGET` est différente de zéro.
- ☐ Dans ce cas, le DB Buffer Cache, le Shared Pool, le Large Pool et le Java Pool sont dimensionnés automatiquement. La taille du Log Buffer n'est pas prise en charge par la gestion automatique, sa taille est déduite du `SGA_TARGET`.

Récapitulatif SGA



SERVEUR BD =
Instance
+
BD (fichiers)

INSTANCE =
Zones mémoires
+
Processus

ZONES MÉMOIRES
=
3 + 3



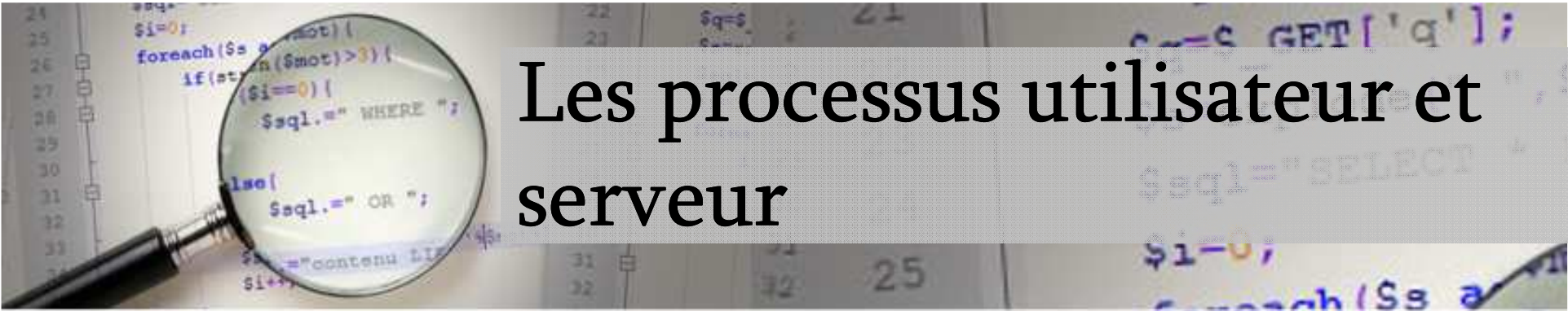
La PGA

- ☐ C'est une zone mémoire privée dédiée aux utilisateurs. Elle est créée pour chaque processus serveur.
- ☐ Elle stocke des informations spécifiques aux utilisateurs, tel que les variables hôtes, les variables de session, l'état des curseurs utilisés, des informations de tri etc.
- ☐ La PGA totale allouée à tous les processus serveur est appelée PGA agrégée.
- ☐ Le paramètre `PGA_AGGREGATE_TARGET` définit la taille de la PGA agrégée, c'est Oracle qui se charge de la répartir entre les différentes processus serveur.




Les processus

- ❑ Ils permettent une interaction entre les différentes composantes du serveur ainsi qu'avec les utilisateurs.
- ❑ Il existe trois types de processus
 1. Les processus utilisateur
 2. Les processus serveur
 3. Les processus d'arrière plan



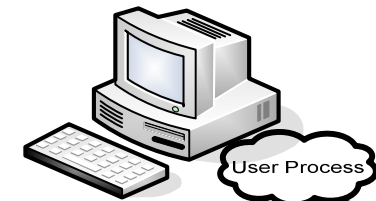
Les processus utilisateur et serveur

- ☐ Le **processus utilisateur** s'exécute au niveau client.
- ☐ Le **processus serveur** s'exécute au niveau serveur BD.
- L'interaction entre le serveur et les clients se fait en réalité grâce à ces deux processus, chacun de son côté.
- ☐ Lorsque le client est lié au serveur, on parle d'une **connexion**.
- ☐ Lorsque l'utilisateur s'identifie, il ouvre une **session**.
- ☐ Plusieurs sessions peuvent être ouvertes en même temps. Elles ne doivent pas dépasser la valeur du paramètre **SESSIONS**.



Les processus utilisateur et serveur

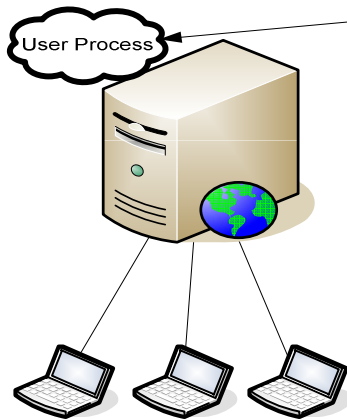
- ☐ Le **processus serveur** (PS) reçoit les requêtes utilisateur, les exécute et renvoie le résultat.
- ☐ Un serveur BD est soit en mode **DÉDIÉ**, soit en mode **PARTAGÉ**.
- ☐ En mode **dédié**, on dédie un PS à chaque utilisateur.
- ☐ En mode **partagé**, on partage un PS pour un ensemble d'utilisateurs.
- ☐ Le mode par défaut est le mode dédié. Pour activer le mode partagé, il faut modifier la valeur de **SHARED_SERVERS** (par défaut =0).
- ☐ La valeur de **SHARED_SERVERS** indique le nombre de processus serveur partagés.
- ☐ Rappelons qu'une zone de mémoire **PGA** est allouée à chaque PS.



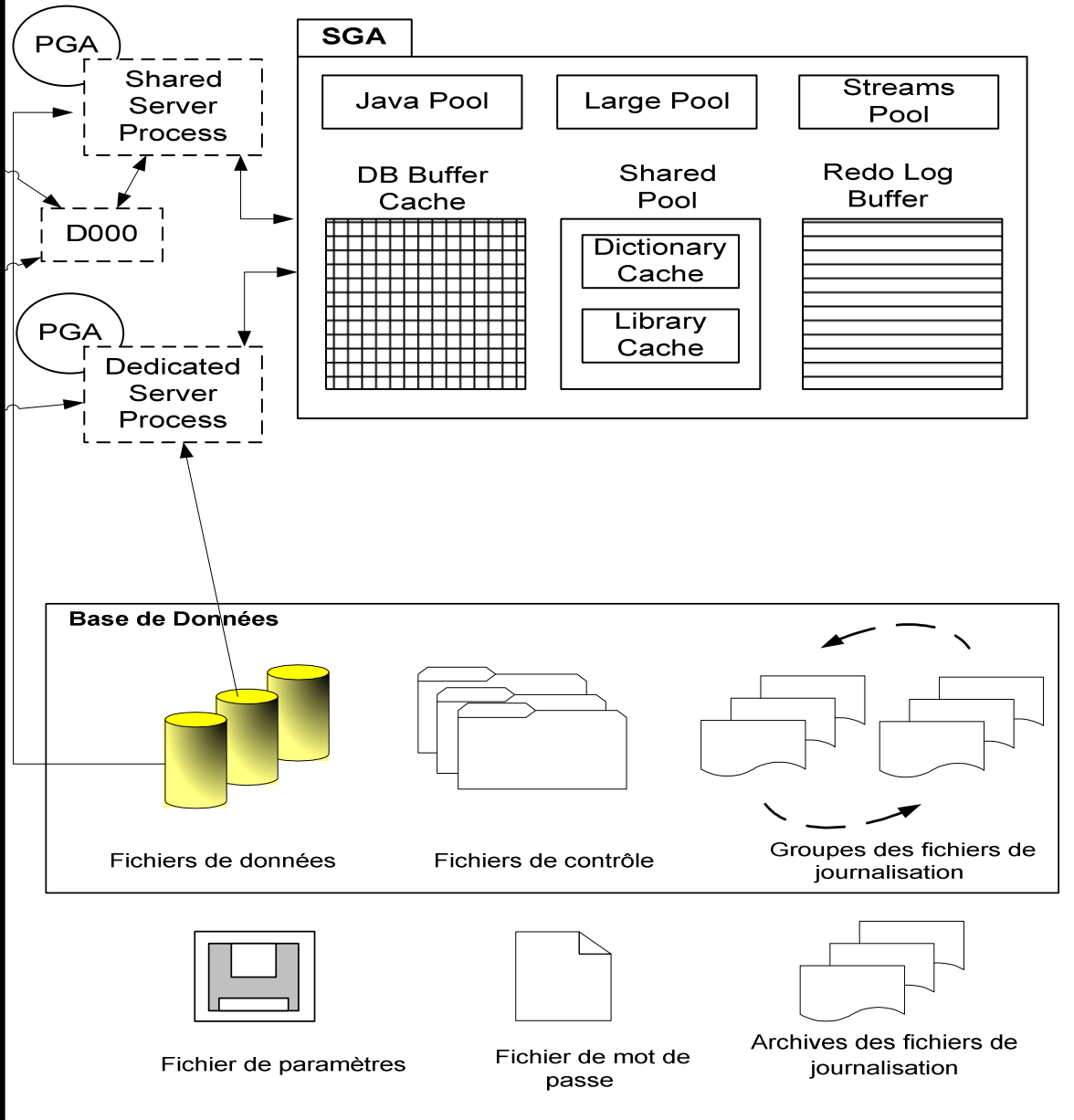
User Process



User Process



User Process



Serveur de Base de Données Oracle



Les processus d'arrière plan

- ☐ Assurent le bon fonctionnement du serveur.
- ☐ Maximisent la performance du serveur.
- ☐ Démarrent avec ou après (sur demande) le démarrage de l'instance.
- ☐ Certains peuvent être exécutés en n exemplaires.



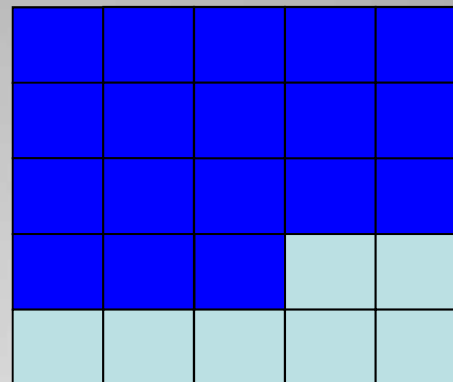
Le DB Writer (DBW_n)

- ❑ « **Nettoie** » le DB Buffer Cache des blocs « **dirty** » les moins récemment utilisés en les écrivant sur les fichiers de données.
- Ainsi, les PS trouvent de la place dans le cache pour y charger les blocs de données.
- ❑ Peuvent fonctionner en 20 exemplaires en cas de forte charge transactionnelle (inutile en cas de serveurs monoprocesseurs).
- ❑ Le paramètre qui détermine ce nombre `DB_WRITER_PROCESSES`.
- ❑ L'écriture des blocs modifiés est différée; le COMMIT par exemple ne déclenche pas une écriture par le DBW_n.

Le DB Writer (DBW_n)

- ❑ Le **DBW_n** se déclenche (en écrivant les blocs dirty les moins récemment utilisées sur le disque) par ces deux événements:
 1. Le PS ne trouve pas de tampons « clean » après avoir scanné un nombre seuil de tampons.

Seuil = 13



DB Buffer Cache

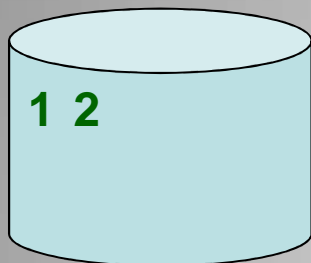
Le DB Writer (DBW_n)

❑ Le DBW_n se déclenche (en écrivant les blocs dirty les moins récemment utilisés sur le disque) par ces deux événements:

2. Après une certaine période pour faire avancer le point de reprise.

×	×	3	4
5			

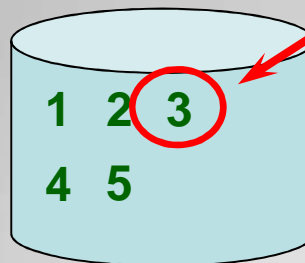
DB Buffer Cache



Data Files

1
2
3
4
5

LOG Buffer



LOG Files

CRASH!!!

1. Insert...;

2. Update...;

Le DBW est déclenché !

3. Update...;

commit;

4. Insert...;

5. Update...;

commit;

Le point de reprise est la position dans les fichiers journaux à partir de laquelle les données sont récupérables = le plus ancien tampon dirty du DB cache.



Le DB Writer (DBW_n)

- ❑ Le **DBW_n** ne se déclenche pas à la suite d'un COMMIT. Il faut donc savoir qu'on peut se trouver dans l'une de ces deux situations:
 1. Le **DBW_n** écrit des modifications non-confirmés (non commités) dans les fichiers de données. On dit qu'il anticipe.
 2. Des modifications confirmées ne sont pas écrites sur les fichiers de données, et restent en mémoire centrale (dans le DB Buffer Cache).



Le LOG Writer (LGWR)

- ☐ Le **LGWR** écrit le contenu du Redo Log Buffer sur les fichiers journaux pour libérer de l'espace au **PS**.
- ☐ Le **LGWR** est déclenché par :
 1. COMMIT → écriture des entrées redo relatives à la transaction confirmée.
 2. Après un délai (3 secondes)
 3. Quand le Redo Log Buffer est plein au tiers
 4. Avant que le DBWn n'écrive sur les fichiers de données des blocs dirty non confirmés. Ainsi, en cas de crash, l'annulation se fait lors du démarrage du serveur.



Le fast COMMIT & le SCN

- ❑ La notion de **fast COMMIT** est liée à l'écriture **différée** des blocs dirty confirmés sur les fichiers de données.
- ❑ Dans le cas où après un COMMIT, seules les entrées Redo de la transaction confirmée sont écrites sur les fichiers journaux (les blocs dirty ne sont pas écrits sur les fichiers de données par le DBWn), cela s'appelle **Fast COMMIT**.
- ❑ Le **SCN** (**System Change Number**) est un code affecté à toute transaction confirmée par un COMMIT.
- ❑ Le **SCN** est inscrit sur les entrées redo de la transaction confirmée sur les fichiers journaux.
- ❑ Ce système de codage est fondamental pour la restauration des données en cas de crash du système.



Le Check Point (CKPT)

- ❑ Le **CKPT** est lancé lorsque le système a besoin de faire un check point (point de vérification).
- ❑ Lorsqu'un point de vérification est déclenché, le **CKPT** envoie un message au **DBWn** pour qu'il écrive tous les blocs dirty sur les fichiers de données.
- Une fois le **DBWn** termine sa tâche, le **CKPT** écrit le **SCN** de la dernière transaction confirmée dans les entêtes des fichiers de données ainsi que dans le fichier de contrôle.
- En cas de crash, les transactions récupérées à partir des fichiers journaux sont celles dont le **SCN** est supérieur à celui écrit dans le fichier de contrôle.



Le Check Point (CKPT)

❑ Les événements qui déclenchent un point de vérification (**CKPT**):

1. Lorsque le **LGWR** bascule d'un groupe à un autre. En fait le LGWR ne prend pas le risque d'écraser des entrées Redo sans écrire les blocs qui leur correspondent dans les fichiers de données.
2. A la fermeture de la base de données.
3. Lors de la mise en offline d'un tablespace.

Exemple

1	2	3	4
5			

DB B

Un message est
envoyé au LGWR

1
2
3
4
5

LOG Buffer

DBWn

CKPT

LGWR

SCN 1

1 2 3 4

Data Files

SCN 1

Control File

1 2 3

4 5

LOG Files

SCN 1

SCN 2

1. Insert...;

2. Update...;

Le DBW est déclenché !

3. Update...;

commit;

4. Insert...;

Check point (fermeture du tblsp) !

5. Update...;

commit;



Le System Monitor (SMON)

- ❑ Récupération des données au démarrage après un crash.
- ❑ Optimisation de l'espace disque (libération des segments temporaires, compactage des extensions libres et contiguës)
- ❑ Le **SMON** effectue ou bien un **ROLLBACK**, ou bien un **ROLLFORWARD**
 - ❖ **ROLLBACK** : annulation des modifications non-confirmées qui étaient enregistrées sur disque par anticipation du **DBWn**.
 - ❖ **ROLLFORWARD** : enregistrement des modifications confirmées sur les fichiers de données à partir des fichiers journaux.



Le Process Monitor (PMON)

- ❑ Détection du plantage des processus utilisateurs.
- ❑ Imaginez ce scénario :
 1. Un **PU** qui a déjà lancé une requête UPDATE
 2. Les données modifiées sont naturellement verrouillées
 3. Le **PU** bloque
 - Les données seront verrouillées indéfiniment !
 - C'est là que le **PMON** entre en action, il détecte le **PU** bloqué, annule la transaction en cours et libère les verrous



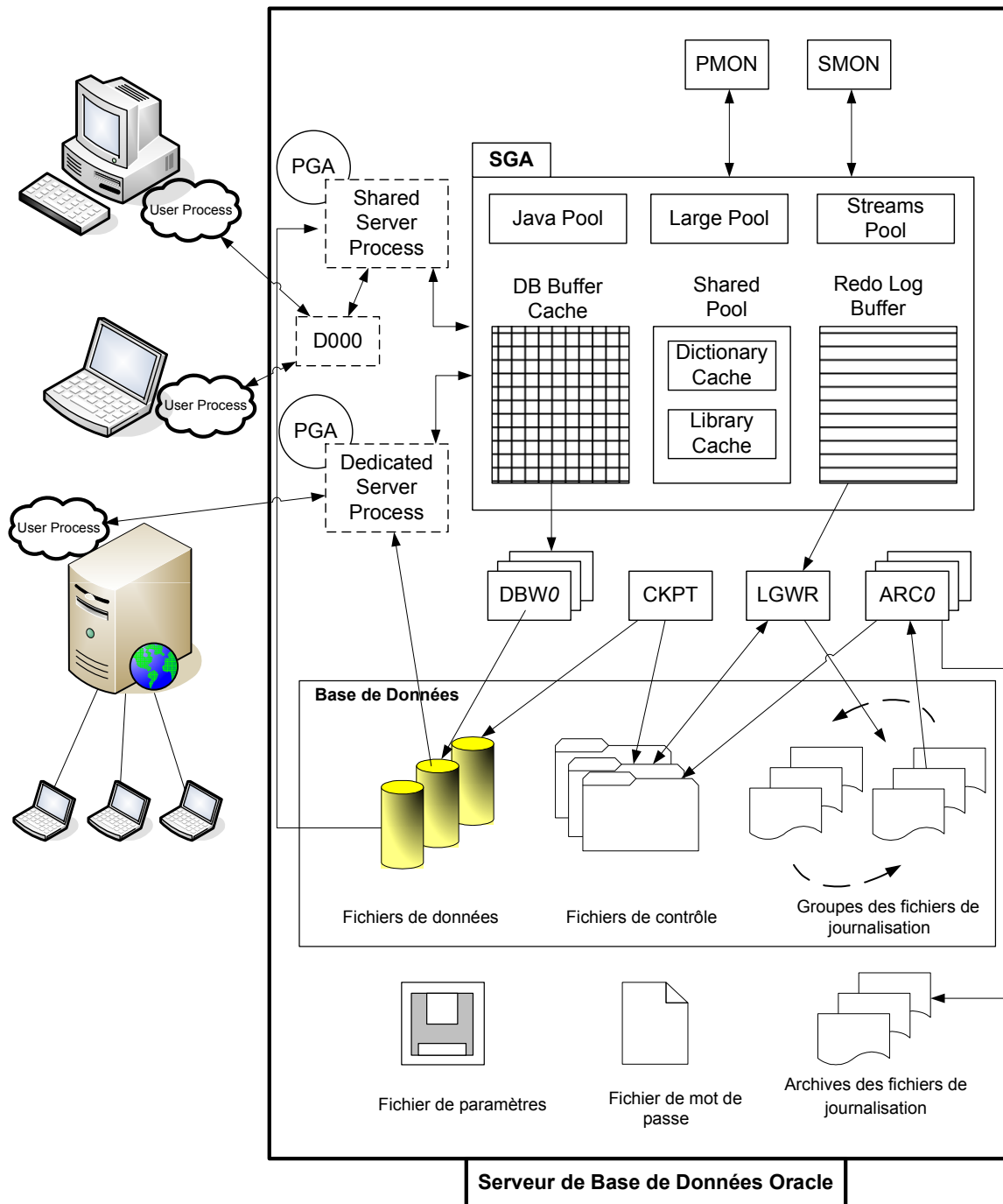
ARCHIVE (ARC n)

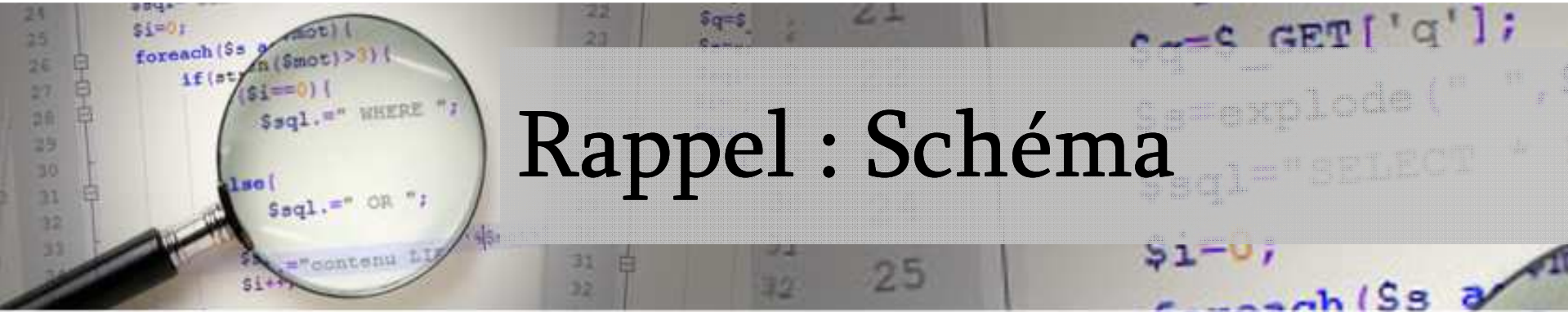
- ☐ Ecriture du contenu des fichiers journaux dans les fichiers journaux archivés.
- ☐ Déclenché par le basculement dans les fichiers journaux d'un groupe à un autre.
- ☐ Activé si le serveur fonctionne en mode ARCHIVELOG.
- ☐ Le paramètre `LOG_ARCHIVE_MAX_PROCESSES` spécifie le nombre d'exemplaires d'ARC n , même si le LGWR peut spécifier ce nombre de manière automatique (si pas d'intervention de l'administrateur).



Les processus d'arrière plan

- ☐ Il existe d'autres processus d'arrière plan, nous nous arrêterons sur l'**ARCn**.
- ☐ Le paramètre d'initialisation **PROCESSES** définit le nombre max de processus connectés en même temps à l'instance.
- ☐ La vue **V\$PROCESS** inclut les informations de tous les processus en cours d'exécution dans l'instance.





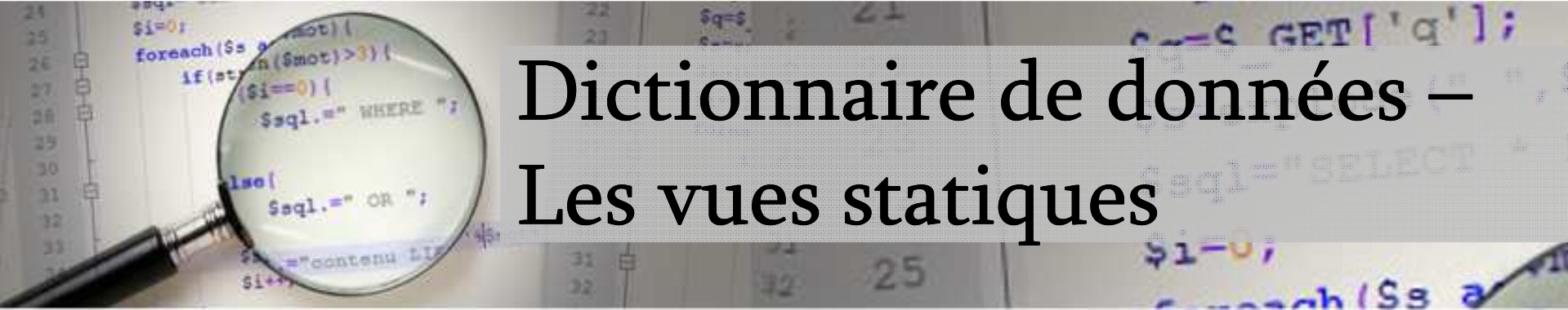
Rappel : Schéma

- ☐ Ensemble d'objets qui ont été créés par un utilisateur.
- ☐ Chaque utilisateur est le propriétaire d'un unique schéma.
- ☐ Les principaux types d'objets d'un schéma sont les **tables**, les **index**, les **vues**, les **synonymes**, les **séquences**, les **déclencheurs**, les **fonctions et procédures stockées** et les **packages PL/SQL**.
- ☐ Seuls les **tables** et les **index** sont stockés sous forme de segments, les autres objets correspondent à des définitions dans le dictionnaire de données.
- ☐ Les objets d'un schéma peuvent être stockés sur différents tablespaces tandis qu'un unique tablespace peut inclure plusieurs schémas.




Rappel : Dictionnaire de données

- ☐ Ensemble de **tables/vues** accédées en lecture seule, créées et maintenues par le système.
- ☐ Elles contiennent toutes les informations de toutes les composantes logiques et physiques de la base de données ainsi que de l'instance.
- ☐ Il est créé dans le tablespace **SYSTEM**, et c'est l'utilisateur **SYS** qui en est le propriétaire.
- ☐ Mise à jour ?
- ☐ Les tables contiennent des données codées. Pour y accéder, on consulte les vues.
- ☐ Il existe deux types de vues; les vues statiques et les vues dynamiques.



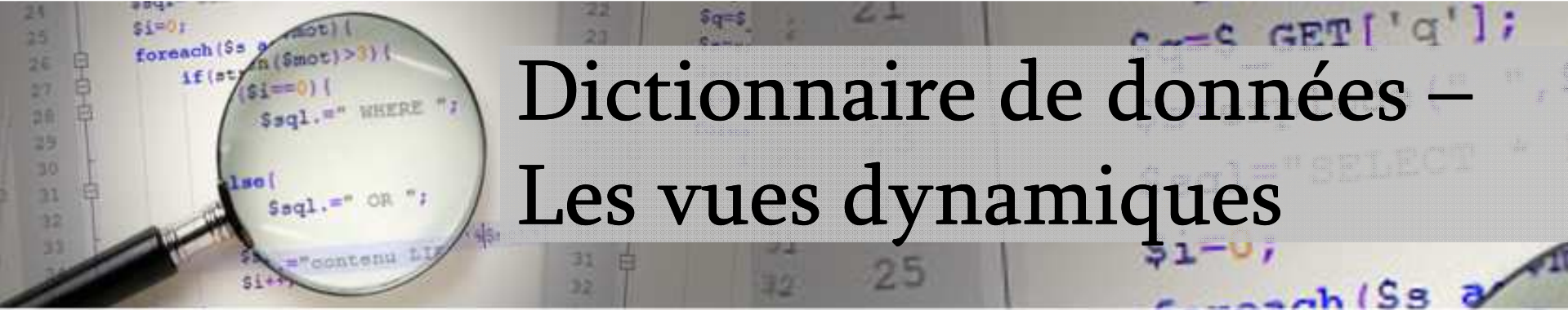
Dictionnaire de données – Les vues statiques

- ❑ Elles sont basées sur des tables créés réellement dans le dictionnaire de données.
- ❑ Accessibles uniquement à l'utilisateur. Commencent par les préfixes:
 - ❑ `USER_` concerne les objets de la base. Accessible par l'utilisateur.
 - ❑ `ALL_` concerne tous les objets de la base. Accessible par l'utilisateur.
 - ❑ `DBA_` concerne tous les objets de la base. Accessible par l'administrateur.
- ❑ La vue `DICTIONARY` inclut des informations sur les vues statiques et dynamiques du dictionnaire de données.



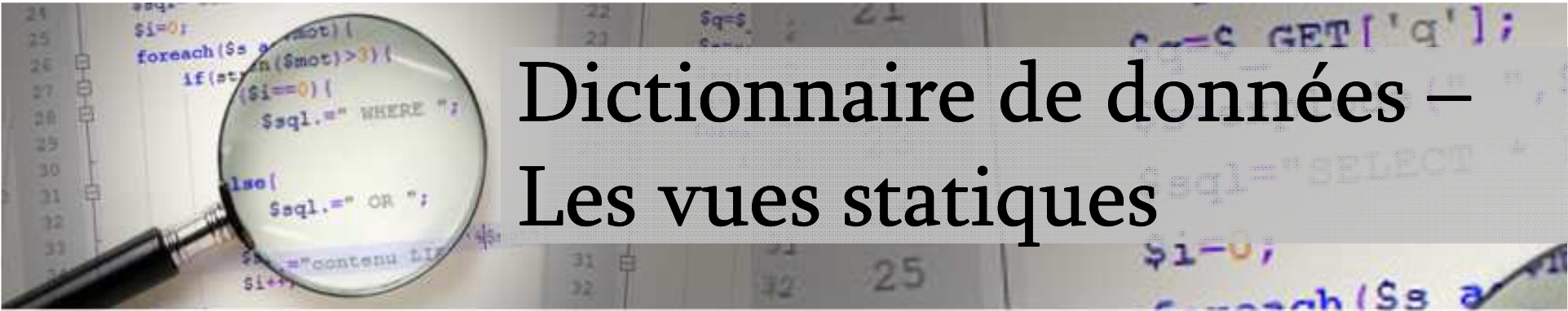
Dictionnaire de données – Les vues statiques

Nom de la vue	Description
<code>%_TABLES</code>	Toutes les informations des tables de la base de données.
<code>%_USERS</code>	Toutes les informations concernant les utilisateurs de la base de données.
<code>%_VIEWS</code>	Toutes les informations des vues de la base de données.
<code>%_SEQUENCES</code>	Toutes les informations concernant les séquences de la base de données.
<code>%_TAB_COLUMNS</code>	Toutes les informations concernant les colonnes des tables de la base de données.
<code>%_INDEXES</code>	Toutes les informations concernant les index de la base de données.
<code>%_OBJECTS</code>	Toutes les informations des objets –tous types confondus- de la base de données.



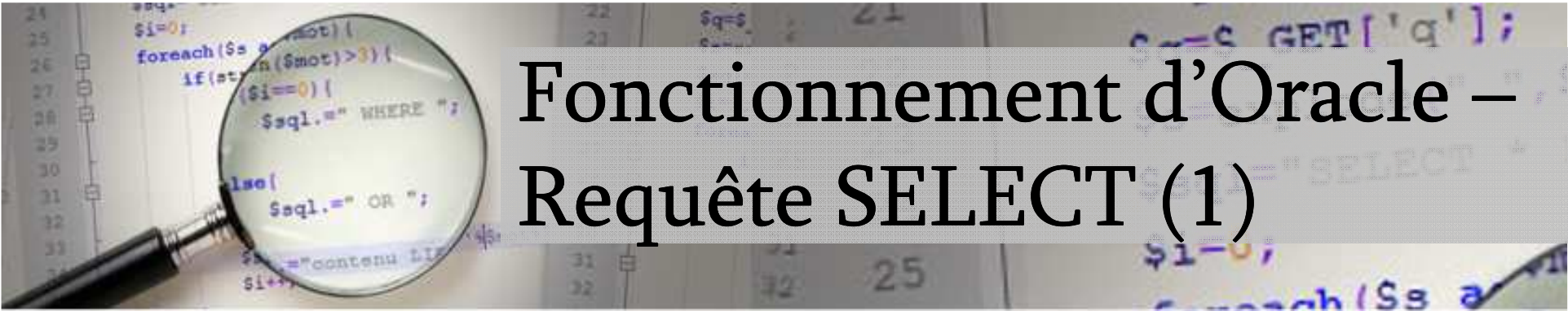
Dictionnaire de données – Les vues dynamiques

- ☐ Elles ne sont pas basées sur des tables du dictionnaire de données. Leurs informations sont extraites de la mémoire et/ou des fichiers de contrôle.
- ☐ Commencent par le préfixe **V\$** et ne sont accessibles que par les administrateurs.
- ☐ Peuvent être consultées même si la base de données n'est pas ouverte



Dictionnaire de données – Les vues statiques

Nom de la vue	Description
V\$DATABASE	Informations de la base de données.
V\$INSTANCE	Informations sur l'instance.
V\$SGA	Informations résumées sur la SGA.
V\$SGA_DYNAMIC_COMPONENTS	Informations détaillées sur les zones mémoire de la SGA.
V\$PARAMETER	Information sur les différents paramètres de l'instance et de la BD.
V\$OPTION	Informations des composantes optionnelles installées sur le serveur BD.
V\$SQL	Informations des requêtes SQL exécutées par tous les utilisateurs de la BD.



Fonctionnement d'Oracle – Requête SELECT (1)

❑ Le serveur BD reçoit une requête SELECT


I. **Phase de parse** : le processus serveur vérifie si la requête existe dans le Library Cache.

a. **Si elle n'y existe pas (Hard parse)**

- i. Une vérification syntaxique est faite
- ii. Vérification sémantique (exactitude des tables & colonnes utilisées, droits d'accès)
 1. Si le DC contient les informations sémantiques, pas d'accès au disque
 2. Sinon, accès au disque (dictionnaire de données)
- iii. Calcul d'un plan d'exécution


b. **Si elle existe (Soft parse)**

- i. Seule la vérification des privilèges est effectuée



Fonctionnement d'Oracle – Requête SELECT (2)

- ❑ Le serveur BD reçoit une requête SELECT
 - I. **Phase de parse** : le processus serveur vérifie si la requête existe dans le Library Cache.
 - II. **Phase d'exécution** : exécution du plan d'exécution
 - a. Si les blocs de données sont dans le DBC, OK
 - b. Sinon, il les récupère des fichiers de données
 - III. **Phase de fetch** : renvoi du résultat de la requête par le processus serveur, au processus utilisateur.



Fonctionnement d'Oracle – Requête UPDATE (1)

❑ Le serveur BD reçoit une requête UPDATE

I. **Phase de parse** : la même que pour une requête SELECT.

II. **Phase d'exécution** :

- a. Chargement des blocs à modifier dans le DBC, si elles n'y existent pas
- b. Verrouillage des lignes à modifier
- c. Un Redo Entry est généré pour la requête et est écrit dans le RLB
- d. Modification des blocs dans le DBC, et écriture différée (par le DBW) des modifications sur les fichiers de données

III. **Phase de fetch** : pas de résultat renvoyé. Juste un message décrivant le déroulement de l'opération.



Fonctionnement d'Oracle – Commande COMMIT

❑ Le serveur BD reçoit la commande COMMIT

- I. Affectation d'un numéro SCN, par le LGWR, à la transaction validée
- II. Écriture des Redo Entry de la transaction sur les fichiers journaux, par le LGWR
- III. Suppression des informations d'annulation, en relation avec la transaction validée, des segments d'annulation



Fonctionnement d'Oracle – Commande ROLLBACK

❑ Le serveur BD reçoit la commande ROLLBACK

I. Deux scénarii possibles :

- a. Les blocs modifiés (dirty) sont déjà enregistrés dans les fichiers de données.
→ Récupération de l'ancienne image à partir des segments d'annulation
- b. Les blocs modifiés sont encore dans le DBC.
→ L'annulation n'engendre aucune écriture sur le disque.



Démarrage de l'instance (1)

❑ Le démarrage de la base de données passe par trois étapes :

I. Démarrage de l'instance

- Lecture du fichier de paramètres
- Allocation de la SGA et des processus selon les valeurs des paramètres
- État NOMOUNT : seuls les vues dynamiques du DD relatives à l'instance sont consultables (V\$INSTANCE, V\$PARAMETER, V\$SGA etc.)
- Utilisé généralement pour créer une nouvelle BD
- STARTUP NOMOUNT



Démarrage de l'instance (2)

❑ Le démarrage de la base de données passe par trois étapes :

II. Démarrage de la base de données

- L'administrateur précise une BD à monter
 - L'instance ouvre le fichier de contrôle dont le chemin est la valeur du paramètre `CONTROL_FILES`
 - Les chemins des fichiers de données et journaux sont lus, mais pas ouverts
 - État MOUNT : seuls SYSDBA et SYSPOER peuvent interroger la BD
 - La vue `V$DATABASE` est interrogeable, mais pas les vues statiques, puisque la base n'est pas ouverte
 - Exemple de tâches : restauration, activation du mode ARCHIVELOG, déplacement des fichiers de données et journaux
- `STARTUP MOUNT [nom_BD]`



Démarrage de l'instance (3)

- ❑ Le démarrage de la base de données passe par trois étapes :

III. Ouverture de la base de données

- Ouverture des fichiers de données et journaux. État OPEN : tous les utilisateurs peuvent accéder à la BD. Toutes les vues du DD sont accessibles
- Si un tablespace a été mis en OFFLINE lors de l'arrêt, il le sera aussi à l'ouverture de la BD
- Si le dernier arrêt a été anormal, alors une opération de récupération est effectuée par SMON
- `STARTUP OPEN [nom_BD]`
- Il est possible d'ouvrir la BD en mode READ ONLY
- `ALTER DATABASE OPEN READ ONLY`
- Pour remettre la BD en mode READ WRITE
- `ALTER DATABASE OPEN READ WRITE`



Démarrage de l'instance (4)

❑ Démarrage en mode force

- Si le démarrage normal ne se produit pas, on peut forcer le démarrage
- `STARTUP FORCE`

❑ Démarrage en mode restreint

- Donne l'accès uniquement aux utilisateurs ayant le privilège `RESTRICTED SESSION`
- Option utilisée si l'administrateur a des tâches de maintenance exécuter
 - `STARTUP RESTRICT`
- Pour désactiver le mode restreint
 - `ALTER SYSTEM DISABLE RESTRICTED SESSION`




Démarrage de l'instance (5)

❑ Démarrage suivant un PFILE

- Oracle démarre à partir de `spfile<sid>.ora`, si non trouvé, `spfile.ora`, sinon `init<sid>.ora`
- Pour démarrer à partir d'un fichier PFILE spécifié par l'administrateur
- → `STARTUP PFILE=nom_pfile`

❑ Syntaxe complète de la commande STARTUP

→ `STARTUP [NOMOUNT|MOUNT [nom_bd]|OPEN [nom_bd]] [FORCE]
[RESTRICT] [PFILE=nom_pfile]`




Arrêt de la base de données (1)

- ❑ La fermeture de la base passe par les mêmes étapes de son ouverture (mount, nomount et arrêt), mais ne s'arrête pas sur un de ces états
- ❑ Il existe 4 types d'arrêt:

I. Arrêt normal

- Oracle ne permet plus l'ouverture des nouvelles sessions
- Oracle attend que le dernier utilisateur soit déconnecté pour arrêter l'instance
- Il s'agit d'un arrêt propre (checkpoint avant fermeture)

→ **SHUTDOWN NORMAL**



Arrêt de la base de données (2)


❑ Il existe 4 types d'arrêt:

I. Arrêt normal

II. Arrêt Transactionnel

- Oracle ne permet plus le lancement de nouvelles transactions
- Oracle attend que la validation/annulation des transactions courantes
- Il s'agit d'un arrêt propre (checkpoint avant fermeture)

→ **SHUTDOWN TRANSACTIONAL**



Arrêt de la base de données (3)

❑ Il existe 4 types d'arrêt:


I. Arrêt normal

II. Arrêt transactionnel

III. Arrêt immédiat

- Oracle annule toutes les transactions courantes et ferme la BD
- Il s'agit d'un arrêt propre (checkpoint avant fermeture)

→ **SHUTDOWN IMMEDIATE**



Arrêt de la base de données (4)

❑ Il existe 4 types d'arrêt:

- I. Arrêt normal
- II. Arrêt transactionnel
- III. Arrêt immédiat
- IV. Arrêt d'abandon

- Oracle ferme la BD sans faire de checkpoint. C'est l'arrêt le plus brutal.
- Au prochain démarrage, une récupération des données, par le SMON, est effectuée

→ SHUTDOWN ABORT

Gestion de l'instance

Réinitialiser									
Sélectionner	Nom	Aide	Révisions	Valeur	Commentaires	Type	De base	Dynamique	Catégorie
<input checked="" type="radio"/>	db_recovery_file_dest_size	?		2G		Big Integer	✓	✓	Sauvegarde et récupération
<input type="radio"/>	pga_aggregate_target	?		63M		Big Integer	✓	✓	Jointures par hachage, index bitmap
<input type="radio"/>	sga_target	?		190M		Big Integer	✓	✓	Mémoire
<input type="radio"/>	db_cache_size	?	1	4M		Big Integer		✓	Mémoire
<input type="radio"/>	db_keep_cache_size	?				Big Integer		✓	Mémoire
<input type="radio"/>	db_recycle_cache_size	?				Big Integer		✓	Mémoire
<input type="radio"/>	db_16k_cache_size	?				Big Integer		✓	Mémoire
<input type="radio"/>	db_2k_cache_size	?				Big Integer		✓	Mémoire
<input type="radio"/>	db_32k_cache_size	?				Big Integer		✓	Mémoire
<input type="radio"/>	db_4k_cache_size	?				Big Integer		✓	Mémoire
<input type="radio"/>	db_8k_cache_size	?				Big Integer		✓	Mémoire
<input type="radio"/>	java_pool_size	?				Big Integer		✓	Mémoire
<input type="radio"/>	large_pool_size	?				Big Integer		✓	Mémoire

× Rechercher : session ⬇ Suivant ⬆ Précédent ⬆ Tout surligner ☐ Respecter la casse



Gestion de l'instance

- ❑ L'ensemble des paramètres sont consultables sur `V$PARAMETER`
- ❑ L'attribut `ISSYS_MODIFIABLE` peut avoir la valeur :
 1. FALSE : le paramètre est statique, sa modification s'applique après redémarrage
 2. IMMEDIATE : le paramètre est dynamique, et modifié pour la session courante
 3. DEFERRED : le paramètre est dynamique, et modifié pour la session future

- ❑ Syntaxe de l'instanciation des paramètres

```
ALTER SYSTEM SET param=val [COMMENT='txt'] [DEFERRED]  
[SCOPE=MEMORY | SPFILE | BOTH]
```

- ❑ L'attribut `ISSES_MODIFIABLE` peut avoir la valeur :
 1. TRUE : le paramètre est modifiable au niveau de la session
 2. FALSE : le paramètre est non modifiable au niveau de la session

```
ALTER SYSTEM SET param=val
```



Gestion de l'instance

Afficher la valeur actuelle, et celle sur le fichier SPFILE du paramètre **SGA_TARGET** ?

```
SELECT p.name, p.display_value, sp.display_value
FROM v$parameter p, v$spparameter sp
WHERE p.name=sp.name AND p.name='sga_target';
```