

# **Introduction au réglage des performances**

# Questions relatives au réglage

- **Pourquoi faire le réglage des performances?**
- **Sur quoi porte le réglage ?**
- **Qui effectue le réglage ?**
- **Comment effectuer le réglage ?**

# Questions relatives au réglage des performances...

## Pourquoi effectuer un réglage ?

### ✓ Réduire les temps de réponse:

- Améliorer le temps de traitement des requêtes
- Réduire ou éliminer les attentes
- Réduire les nombre de lectures / écritures disque.
  - Utiliser le plus petit nombre de blocs possible
  - Garder les blocs de données à disposition en mémoire cache

### ✓ Réduire le temps de récupération:

En cas d'incidents (coupure électricité; crache mémoire disque, ...), Il faut retourner à la normale dans les plus brefs délais en récupérant toutes les données et en s'assurant que les données sont cohérentes

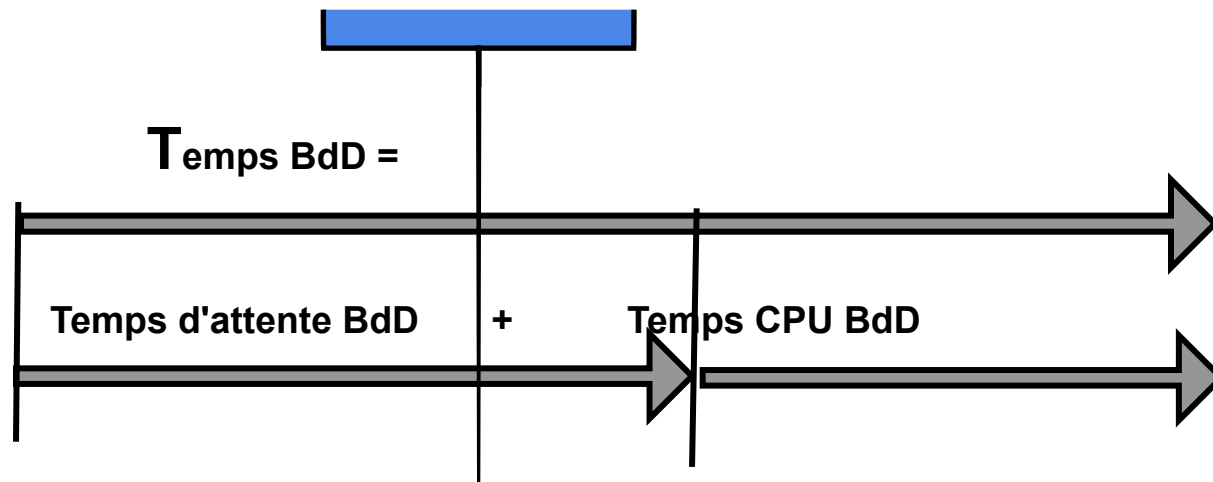
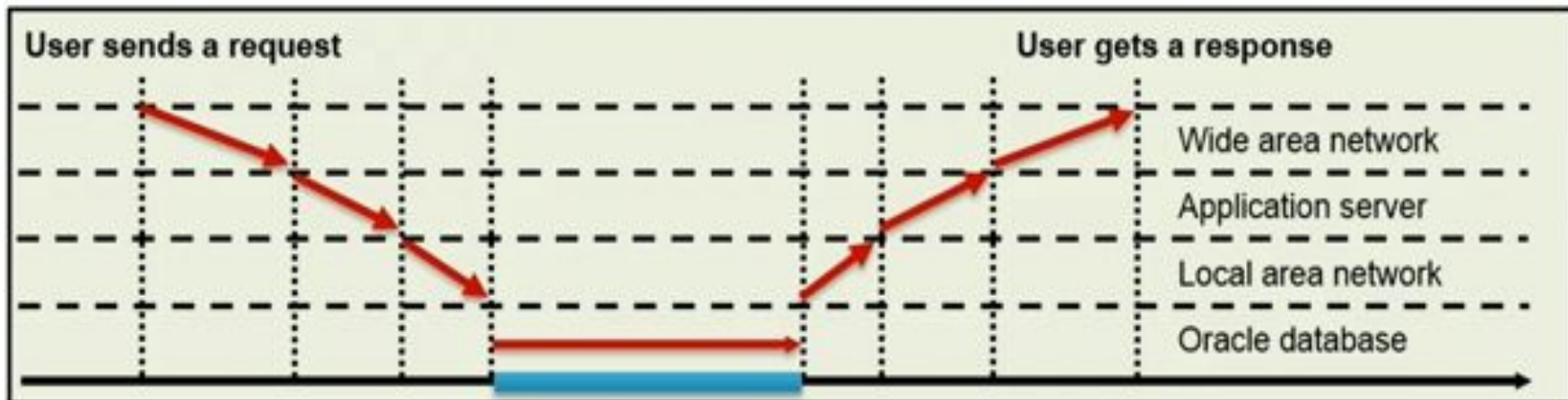
### ✓ Répondre aux SLAs (Engagements envers les clients), Ex :

- Finir quelques tâches avant une certaine heure de la journée
- Extraire des Dashboard / reporting chaque heures,... etc

### ✓ Prévenir les défaillances et les régler en proactive:

- Surveiller les dysfonctionnements avant qu'elles ne se produisent
- Retour à la normale dans les meilleurs délais en cas de défaillance

# Temps de réponse et temps base de données



Pour un réglage efficace, il faut Réduire les temps d'attentes et régler le temps CPU des instructions SQL

# Questions relatives au réglage des performances...

## Sur quoi porte le réglage ?

### Domaines de réglage des performances :

- **Applications :**
  - Code SQL de mauvaise qualité
  - Ressources sérialisées
  - Mauvaise gestion des sessions
- **Instances :**
  - Utilisation de la Mémoire
  - Structure de la base de données
  - Configuration d'instance
- **Système d'exploitation :**
  - E/S
  - Swap
  - Paramètres



# Questions relatives au réglage des performances...

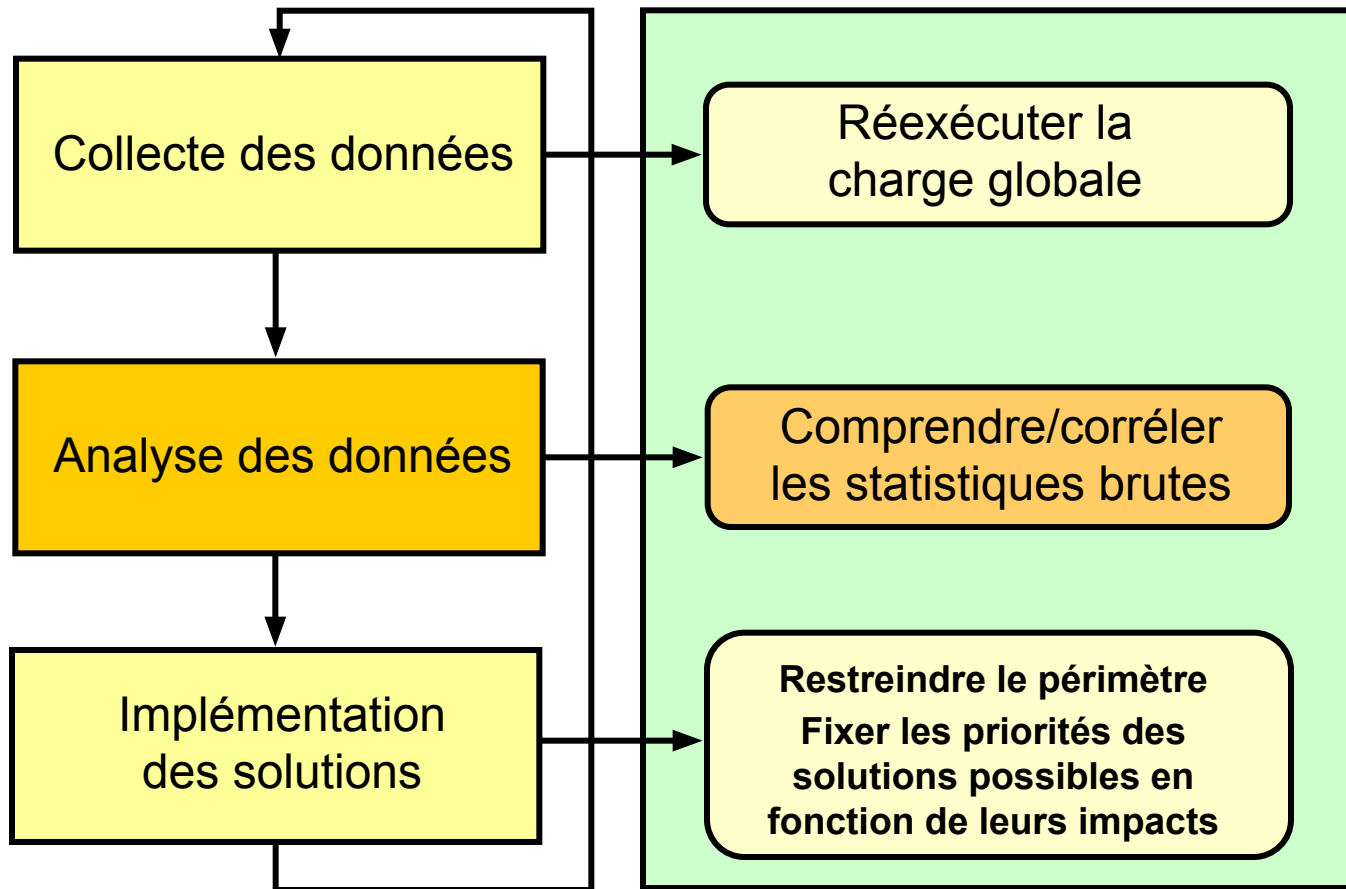
## Qui effectue le réglage ?

De manière général, les personnes qui effectuent le réglage sont:

- Concepteurs d'applications
- Développeurs d'applications
- Administrateurs de base de données
- Administrateurs système

# Questions relatives au réglage

## Comment effectuer le réglage ?



# Exemples de problèmes de réglages courants

- **Une mauvaise gestion des sessions :**
  - Trop de sessions ouvertes rend le système jusqu'à deux fois plus lent que la normale
- **Une mauvaise gestion des curseurs :**
  - Instructions SQL ne sont pas partagées
  - Dégrade les performances de la shared pool
  - Augmente le temps de réponse
- **Une mauvaise conception de système relationnel**
  - entraîne l'exécution de jointures de tables inutiles



# Concilier performances et sécurité

- **Souvent les DBA appliquent les recommandations de sécurité de manière excessive.**
- **Ces facteurs de sécurité peuvent affecter les performances.**
- **Il faudra trouver un juste milieu pour assurer un maximum de sécurité sans impacter les performance**
  - Fichiers de contrôle multiples
  - Nombreux fichiers journaux membres dans un groupe
  - Points de reprise fréquents
  - Sauvegarde des fichiers de données
  - Chemins d'Archivage
  - Nombre d'utilisateurs et de transactions simultanés

# **Chapitre 01**

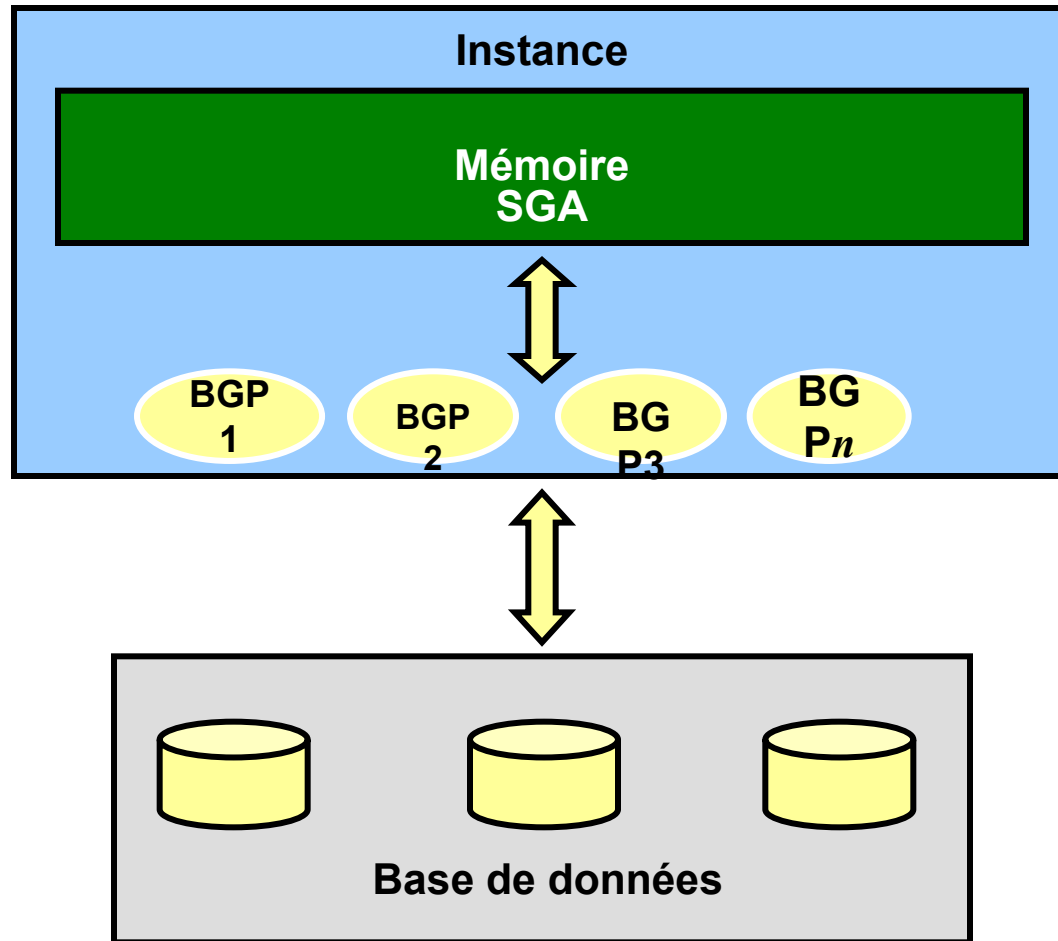
## **Rappel de l'architecture d'un serveur de base de données Oracle**

# Objectifs

**A la fin de ce chapitre, vous pourrez :**

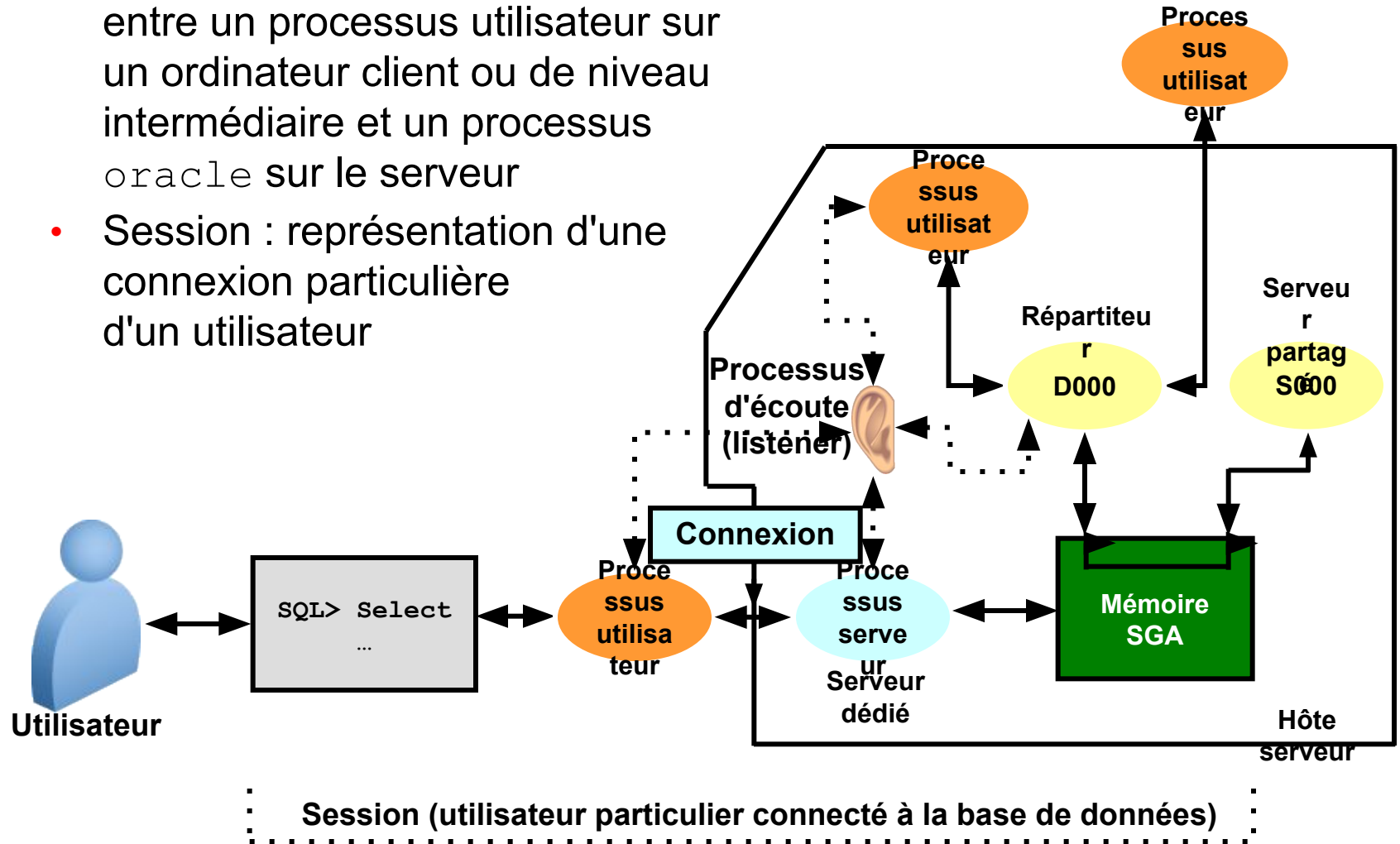
- **énumérer les principaux composants de l'architecture d'un serveur de base de données Oracle**
- **décrire les structures mémoire**
- **décrire les processus en arrière-plan**
- **corrélérer les structures de stockage logique et physique**

# Architecture d'un serveur de base de données Oracle : Présentation



# Connexion à une instance de base de données

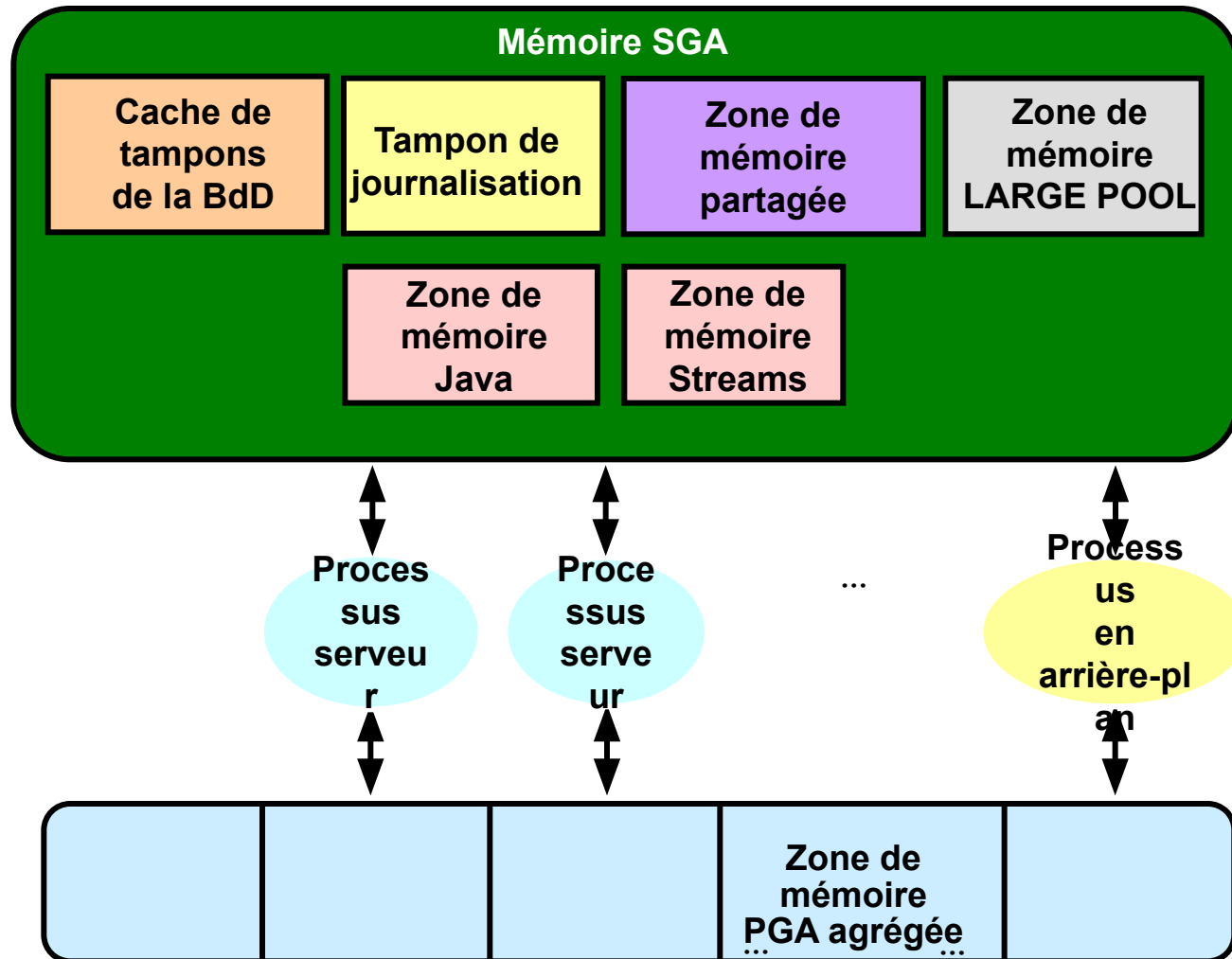
- Connexion : chemin réseau bidirectionnel entre un processus utilisateur sur un ordinateur client ou de niveau intermédiaire et un processus oracle sur le serveur
- Session : représentation d'une connexion particulière d'un utilisateur





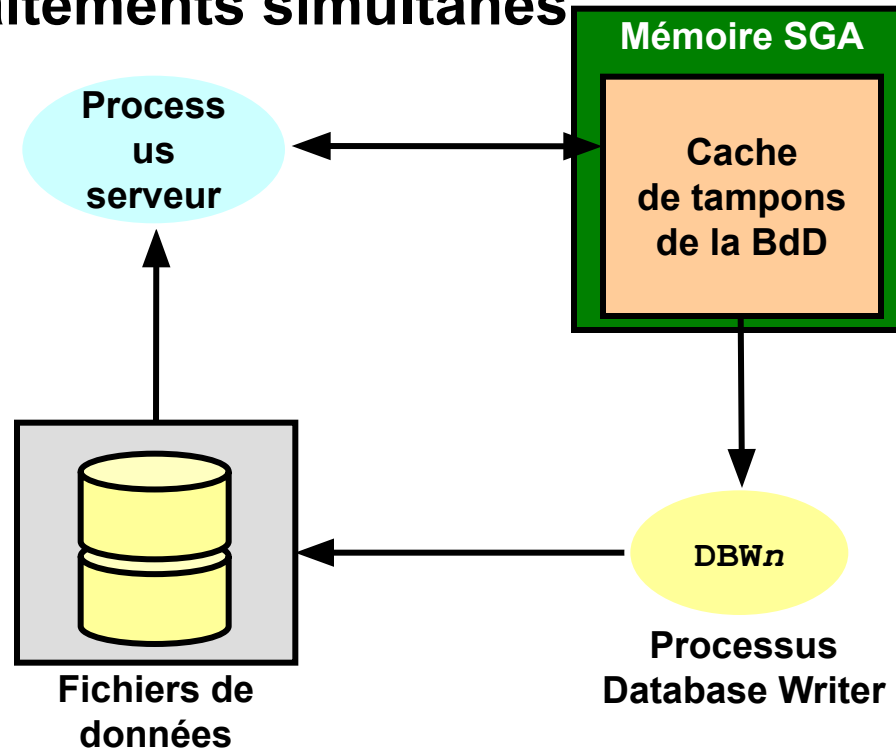
# Structures mémoire d'une base de données

## Oracle : Présentation



# Cache de tampons de la base de données

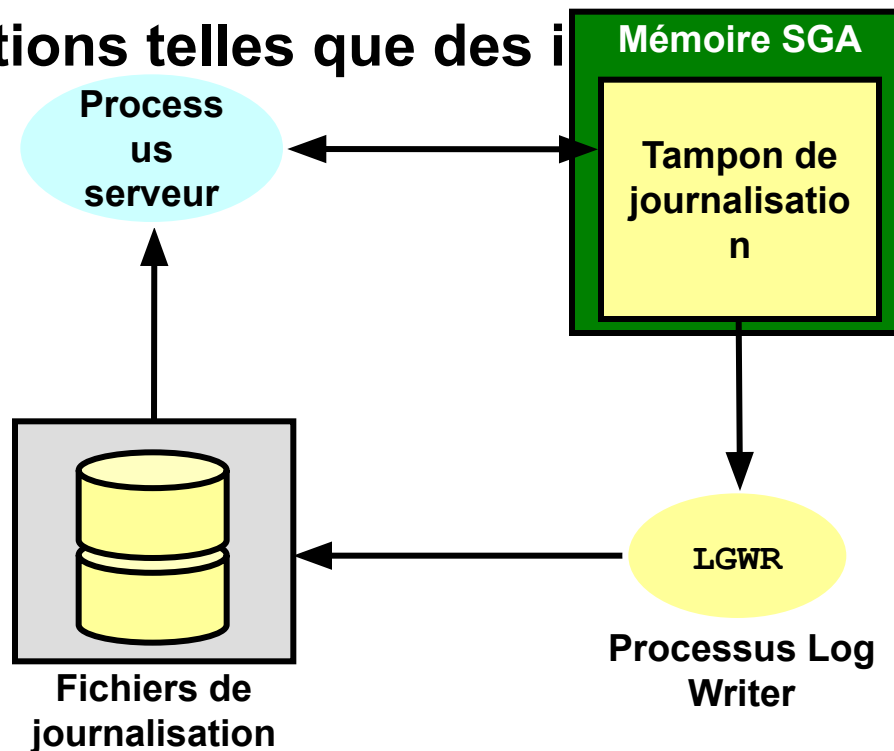
- Il fait partie de la mémoire SGA.
- Il contient des copies des blocs de données qui sont lus depuis les fichiers de données.
- Il est partagé par tous les traitements simultanés





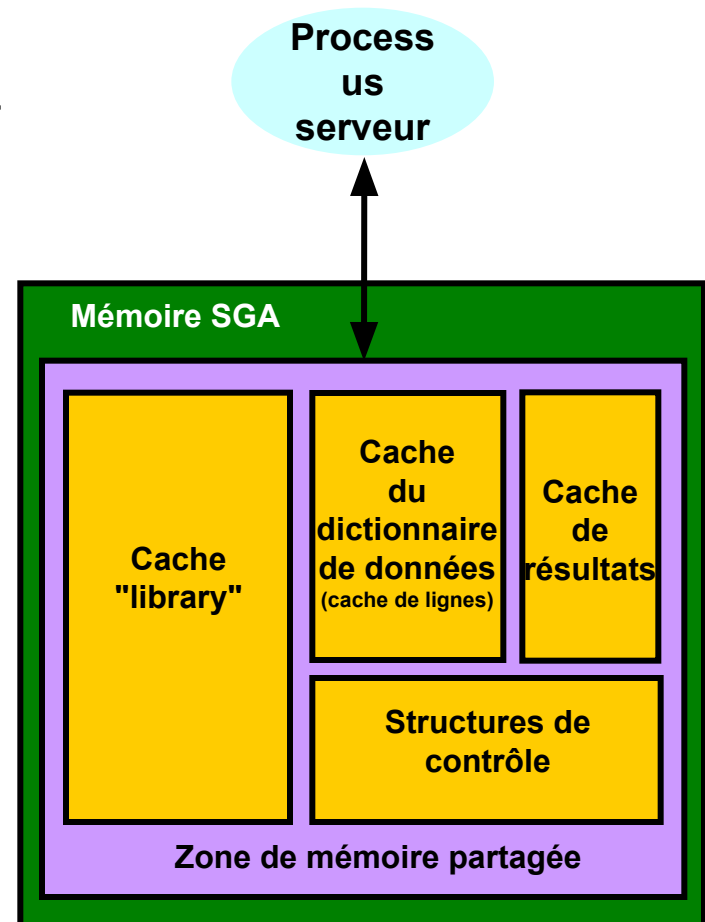
# Tampon de journalisation

- Il correspond à une mémoire tampon réutilisable dans la mémoire SGA (basée sur le nombre de CPU).
- Il contient les entrées de journalisation permettant de reproduire les modifications effectuées par des opérations telles que des insertions, mises à jour ou suppressions de données (LMD ou LDD).

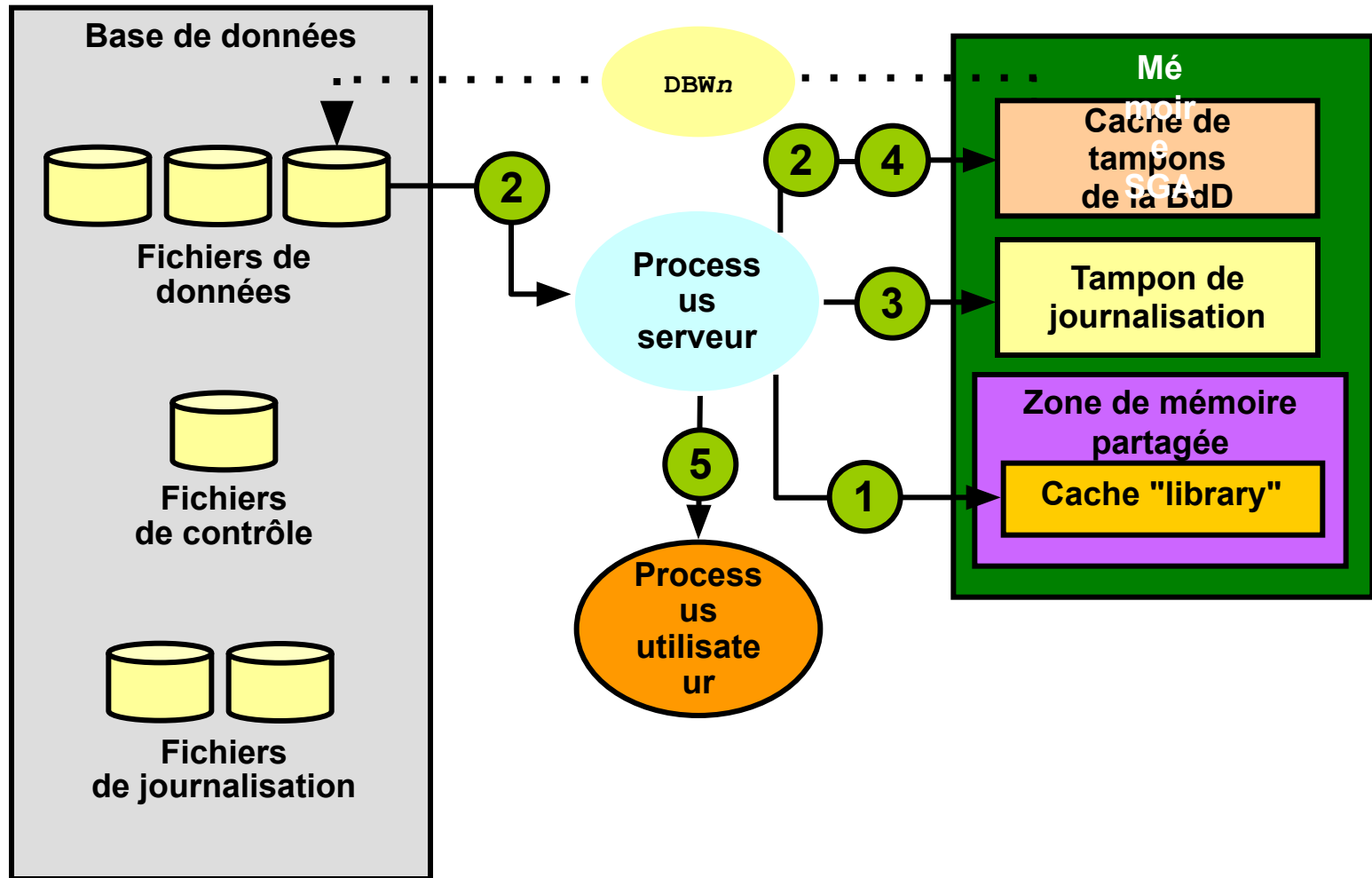


# Zone de mémoire partagée

- Elle fait partie de la mémoire SGA.
- Elle contient les éléments suivants :
  - Cache "library"  
Parties des instructions SQL et PL/SQL qui peuvent être partagées
  - Cache du dictionnaire de données
  - Cache de résultats  
Interrogations SQL  
Fonctions PL/SQL
  - Structures de contrôle  
**Verrous externes (locks)**



# Traitement d'une instruction LMD : Exemple

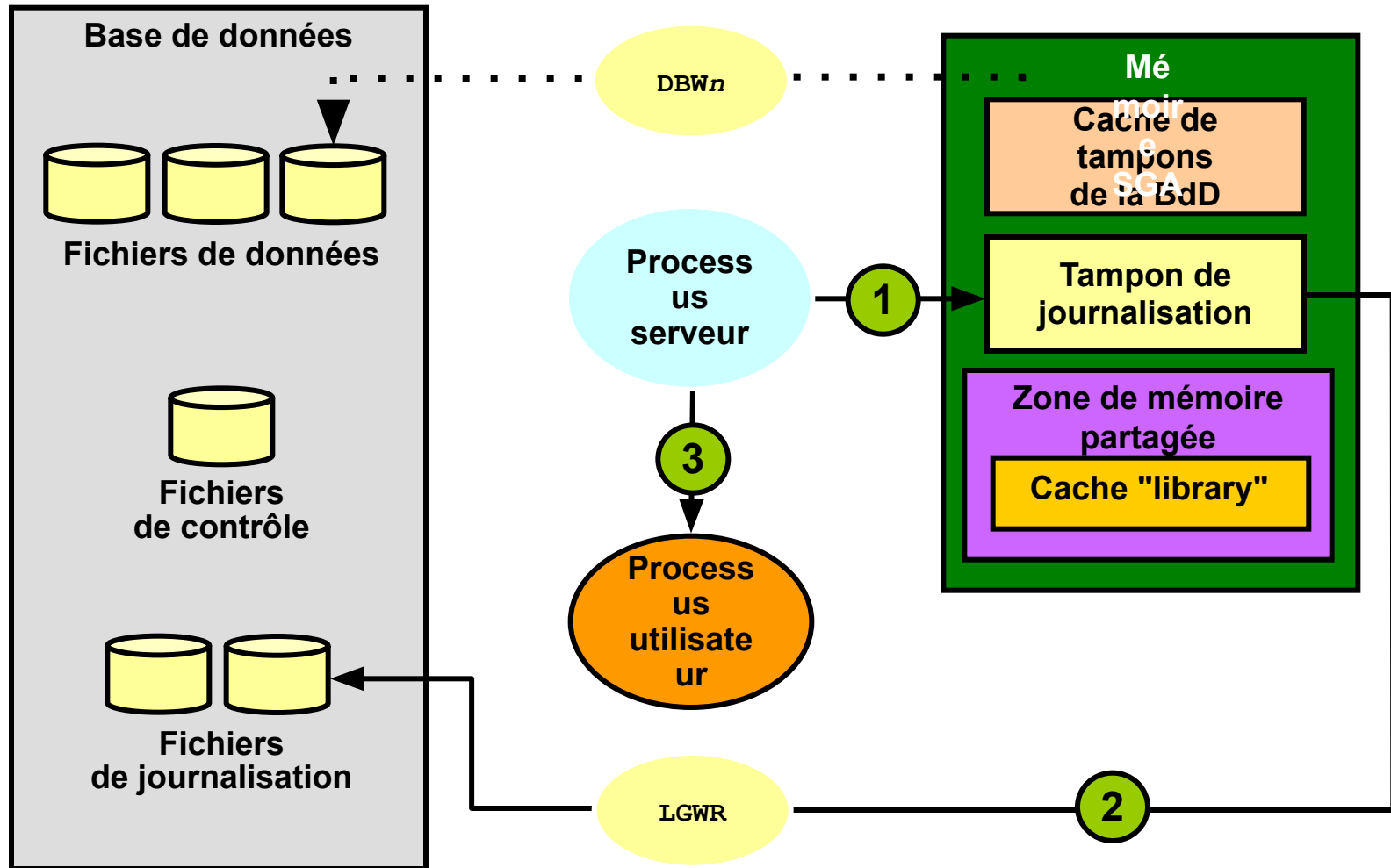


**Traitement d'une instruction LMD** : Les étapes intervenant dans l'exécution d'une instruction LMD (Langage de manipulation de données) sont les suivantes :

1. Le processus serveur reçoit l'instruction et recherche dans le cache "library" une zone SQL partagée contenant une instruction SQL similaire. S'il trouve une telle zone, il vérifie les privilèges d'accès de l'utilisateur aux données demandées et utilise cette zone pour traiter l'instruction. Dans le cas contraire, une nouvelle zone SQL partagée est allouée pour l'instruction pour qu'elle puisse être analysée (parsed) et traitée.
2. Si les blocs de données et les blocs de segments d'annulation ne figurent pas encore dans le cache de tampons (buffer cache), le processus serveur les lit à partir des fichiers de données et les place dans le cache de tampons. Le processus serveur verrouille les lignes à modifier.
3. Le processus serveur enregistre les modifications à apporter aux tampons de données, ainsi que les modifications d'annulation (undo). Ces modifications sont écrites dans le tampon de journalisation (redo log buffer) avant la modification des tampons de données et de segments d'annulation en mémoire. Cette opération est appelée journalisation à écriture anticipée.
4. Les tampons de segments d'annulation contiennent les valeurs des données avant leur modification. Ils sont utilisés pour stocker l'image "avant" des données afin que les instructions LMD puissent être annulées si nécessaire. Les tampons de données enregistrent les nouvelles valeurs des données.
5. L'utilisateur obtient les informations relatives au traitement de l'opération LMD (par exemple, le nombre de lignes modifiées par l'opération).

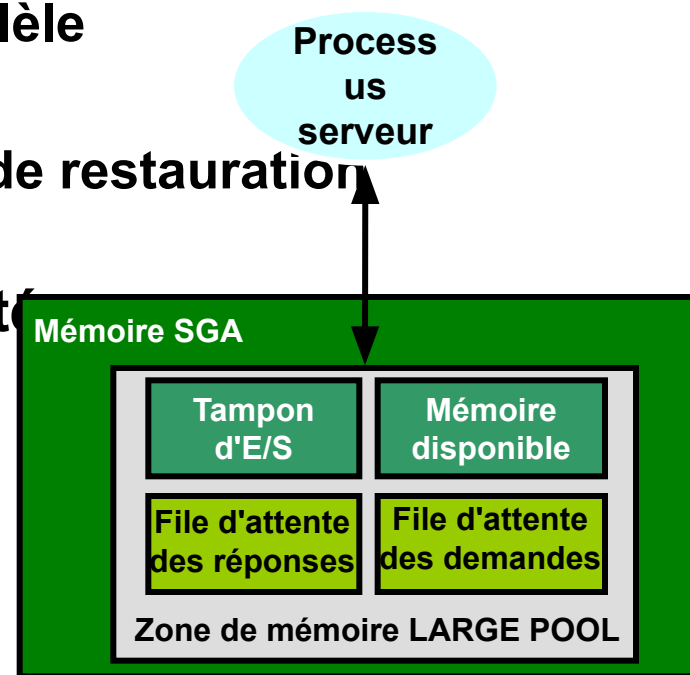
**Remarque** : Les blocs modifiés figurant dans le cache de tampons (buffer cache) sont marqués en tant que tampons "dirty" ; cela signifie qu'ils ne sont pas identiques aux blocs correspondants sur le disque. Ces tampons ne sont pas écrits immédiatement sur le disque par le processus `DBWn`.

# Traitement des opérations COMMIT : Exemple



# Zone de mémoire LARGE POOL

- Elle permet d'allouer des espaces mémoire volumineux pour les éléments suivants :
  - Mémoire allouée par session pour le serveur partagé et l'interface Oracle XA
  - Tampons d'exécution en parallèle
  - Processus serveur d'E/S
  - Opérations de sauvegarde et de restauration de la base de données Oracle
- Zone facultative la mieux adaptée dans les cas suivants :
  - Exécution en parallèle
  - Recovery Manager
  - Serveur partagé

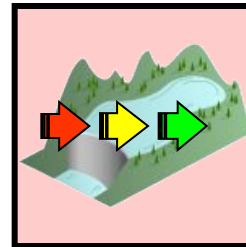


# Zones de mémoire Java et Streams

- La zone de mémoire Java est utilisée dans la mémoire du serveur pour tout le code Java propre à la session et pour toutes les données au sein de la Java Virtual Machine (JVM).
- La zone de mémoire Streams est utilisée exclusivement par Oracle Streams pour :
  - stocker les messages dans une file d'attente tampon
  - fournir de la mémoire aux processus Oracle Streams



Zone de mémoire  
Java



Zone de mémoire  
Streams

# Mémoire PGA (Program Global Area)

- La mémoire PGA est une zone de mémoire qui contient les éléments suivants :

- Informations relatives à la session
- Informations relatives aux curseurs
- Zones de travail d'exécution du

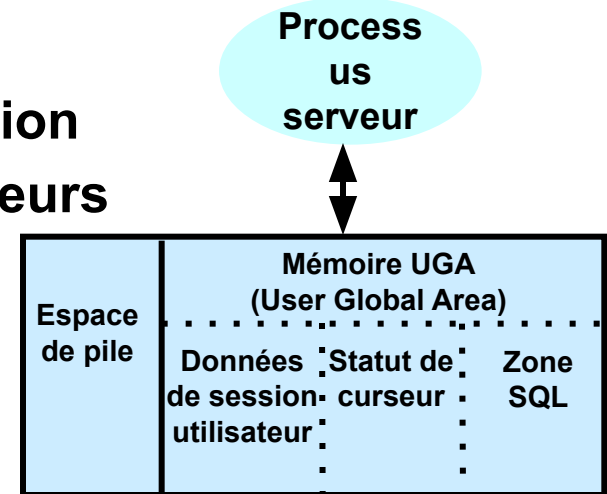
Zone de tri

Zone de jointure de hachage  
(hash join)

Zone de fusion d'index bitmap

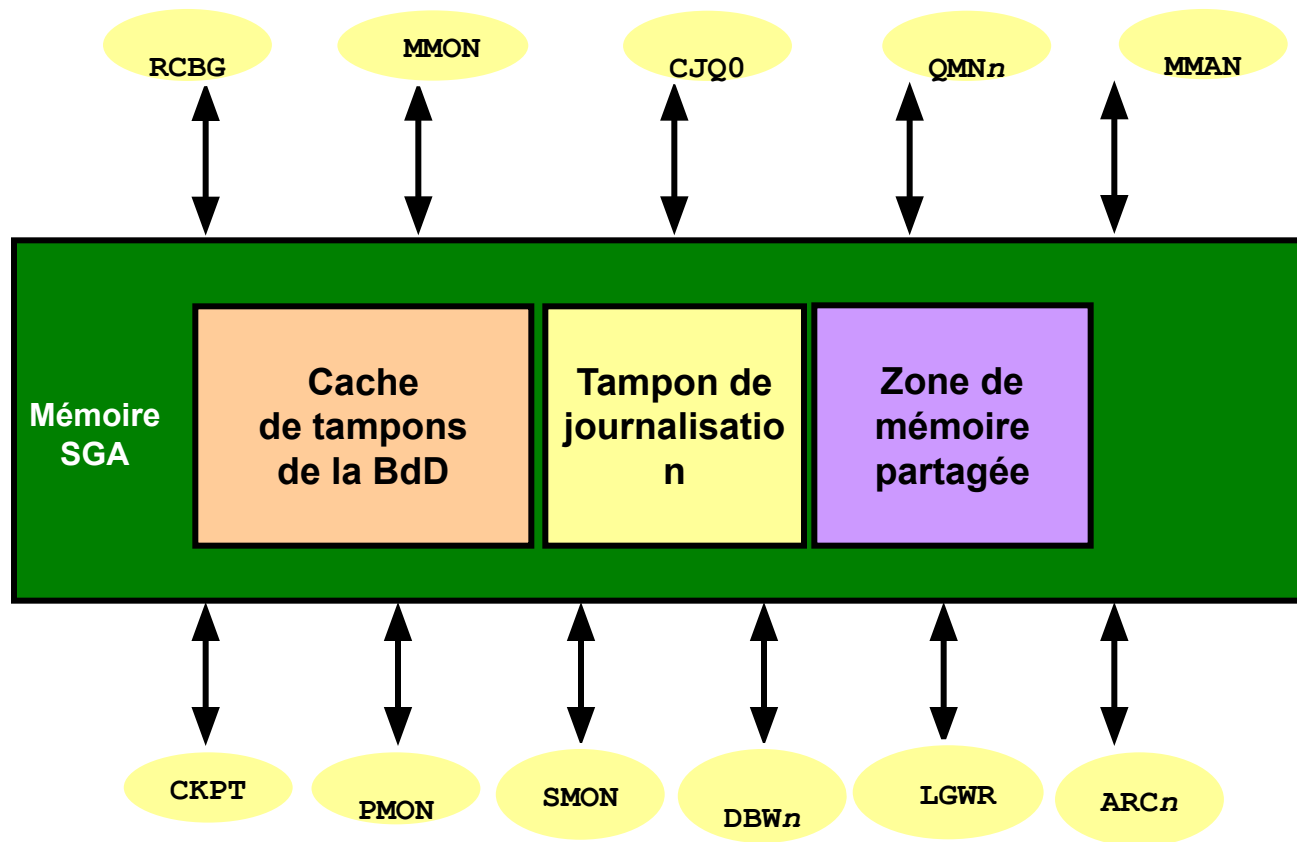
Zone de création d'index bitmap

- La taille de la zone de travail influe sur les performances des instructions SQL.
- Les zones de travail peuvent être gérées automatiquement ou manuellement.





# Processus en arrière-plan

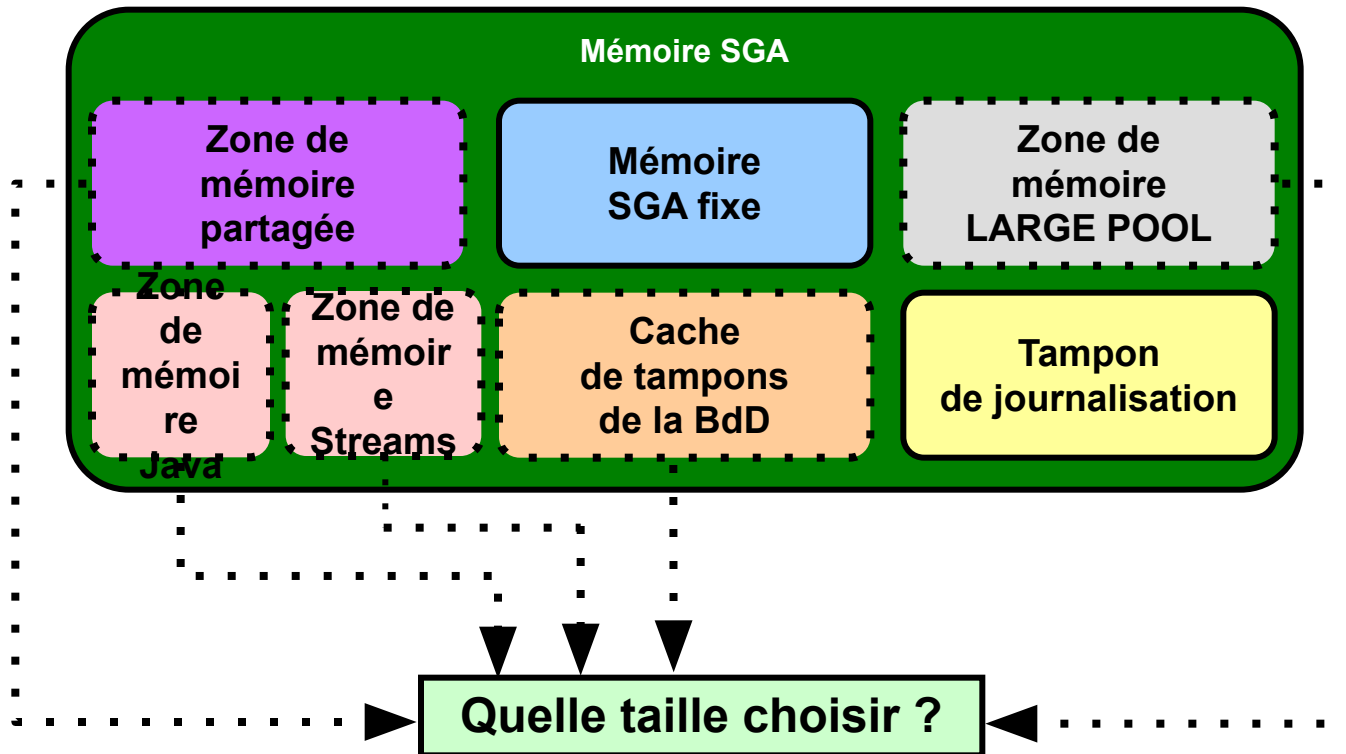


Processus en arrière-plan : Les environnements autres que RAC ou ASM comprennent généralement les processus en arrière-plan suivants :

- **Le processus Database Writer (DBWR<sub>n</sub>)** écrit de manière asynchrone sur le disque les tampons modifiés ("dirty") figurant dans le cache de tampons (buffer cache) de la base de données.
- **Le processus Log Writer (LGWR)** écrit dans un fichier de journalisation (fichier redo log) situé sur le disque les informations de récupération (ou informations de journalisation) figurant dans le tampon de journalisation.
- **Le processus de point de reprise (CKPT)** enregistre les informations relatives au point de reprise dans les fichiers de contrôle et dans chaque en-tête de fichier de données.
- **Le processus System Monitor (SMON)** effectue la récupération au démarrage de l'instance et nettoie les segments temporaires non utilisés.
- **Le processus Process Monitor (PMON)** effectue une récupération en cas d'échec d'un processus utilisateur.
- **Le processus en arrière-plan du cache de résultats (RCBG)** sert à gérer le cache de résultats dans la zone de mémoire partagée.
- **Le processus de gestion de la file d'attente des travaux (CJQO)** exécute les travaux utilisateur par lots via le planificateur.
- **Les processus d'archivage (ARC<sub>n</sub>)** copient les fichiers de journalisation (fichiers redo log) sur un périphérique de stockage spécifique après un changement de fichier de journalisation.
- **Les processus de surveillance de file d'attente (QMN<sub>n</sub>)** contrôlent les files d'attente de messages d'Oracle Streams.
- **Les processus Manageability Monitor (MMON)** effectuent les tâches de gestion en arrière-plan.
- **Le processus en arrière-plan de gestion de la mémoire (MMAN)** est utilisé pour gérer automatiquement les composants de mémoire SGA et PGA.

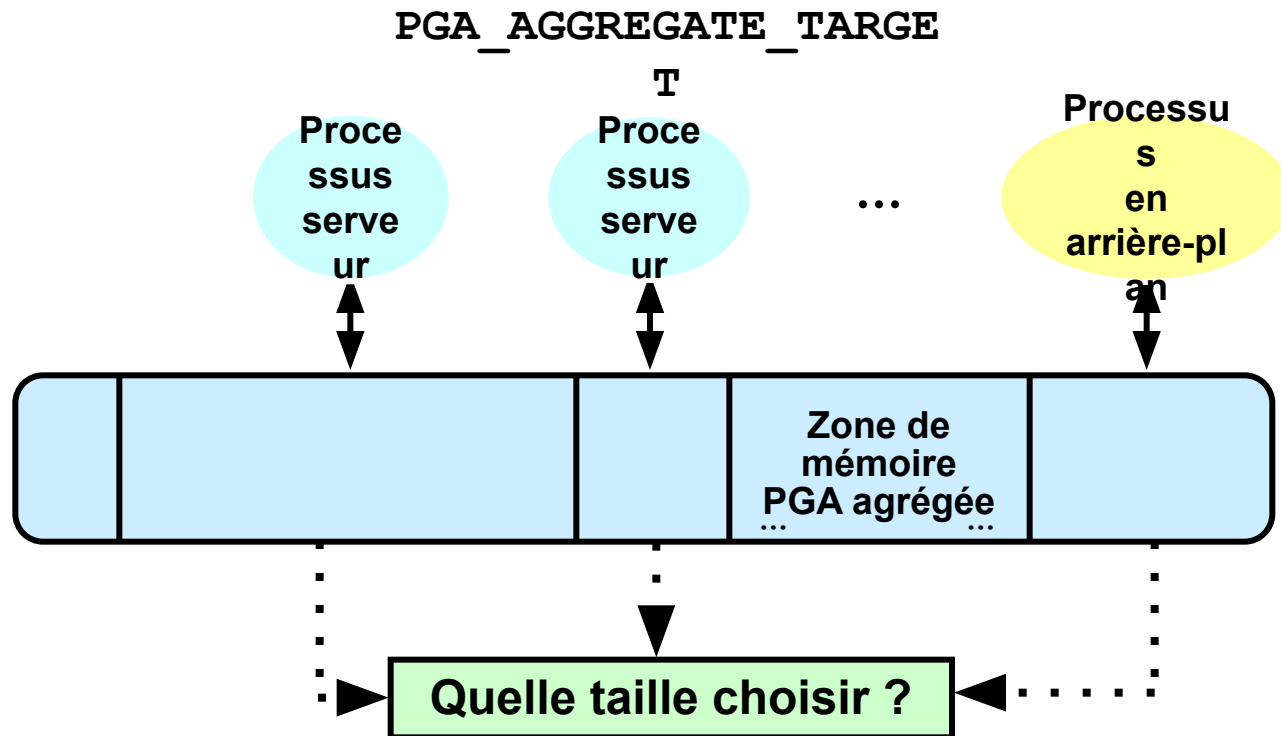
# Gestion automatique de la mémoire partagée

SGA\_TARGET +  
STATISTICS\_LEVEL



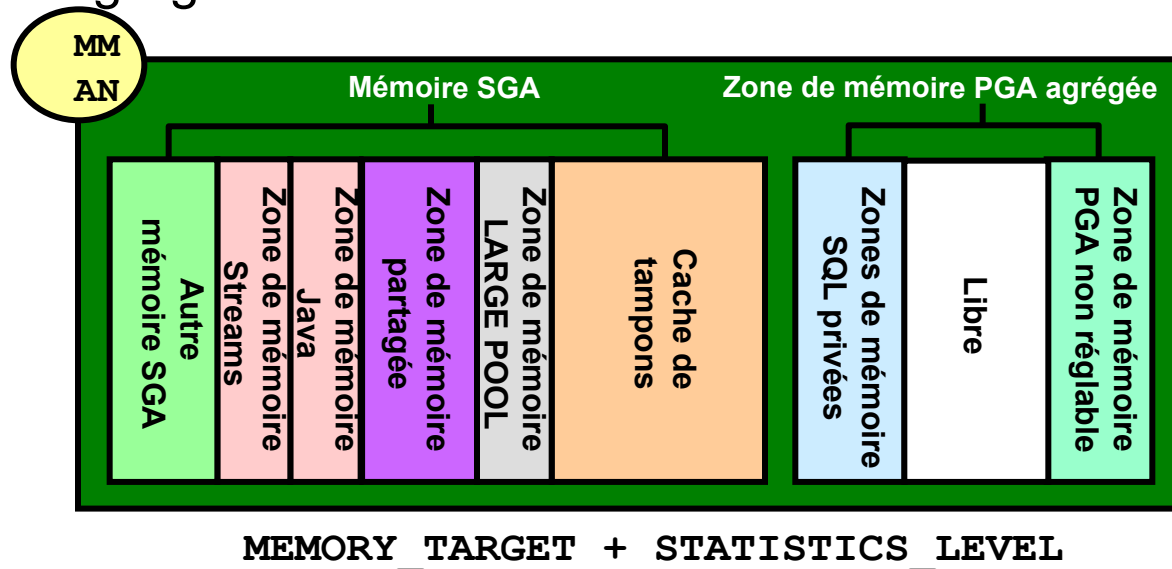
Composants de la mémoire SGA réglés automatiquement

# Gestion automatique de la mémoire d'exécution du code SQL

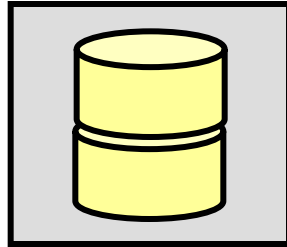


# Gestion automatique de la mémoire

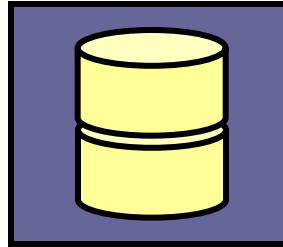
- Le dimensionnement des composants de mémoire est essentiel pour les performances de l'exécution d'instructions SQL.
- Il est difficile de dimensionner manuellement chaque composant.
- La gestion automatique de la mémoire automatise l'allocation de chaque composant de la mémoire SGA et de la mémoire PGA agrégée.



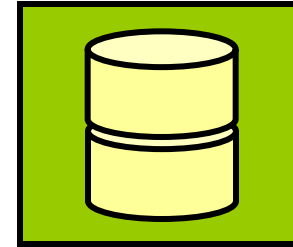
# Architecture de stockage dans la base de données



**Fichier de contrôle**



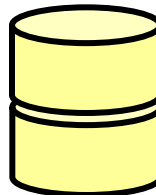
**Fichiers de données**



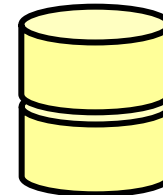
**Fichiers de journalisation en ligne**



**Fichier de paramètres**



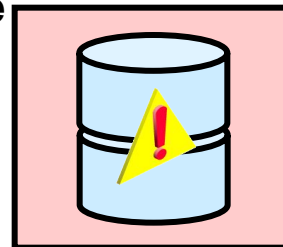
**Fichiers de sauvegarde**



**Fichiers de journalisation archivés**



**Fichier de mots de passe**



**Fichiers d'alertes et fichiers trace**

# Architecture de stockage dans la base de données:

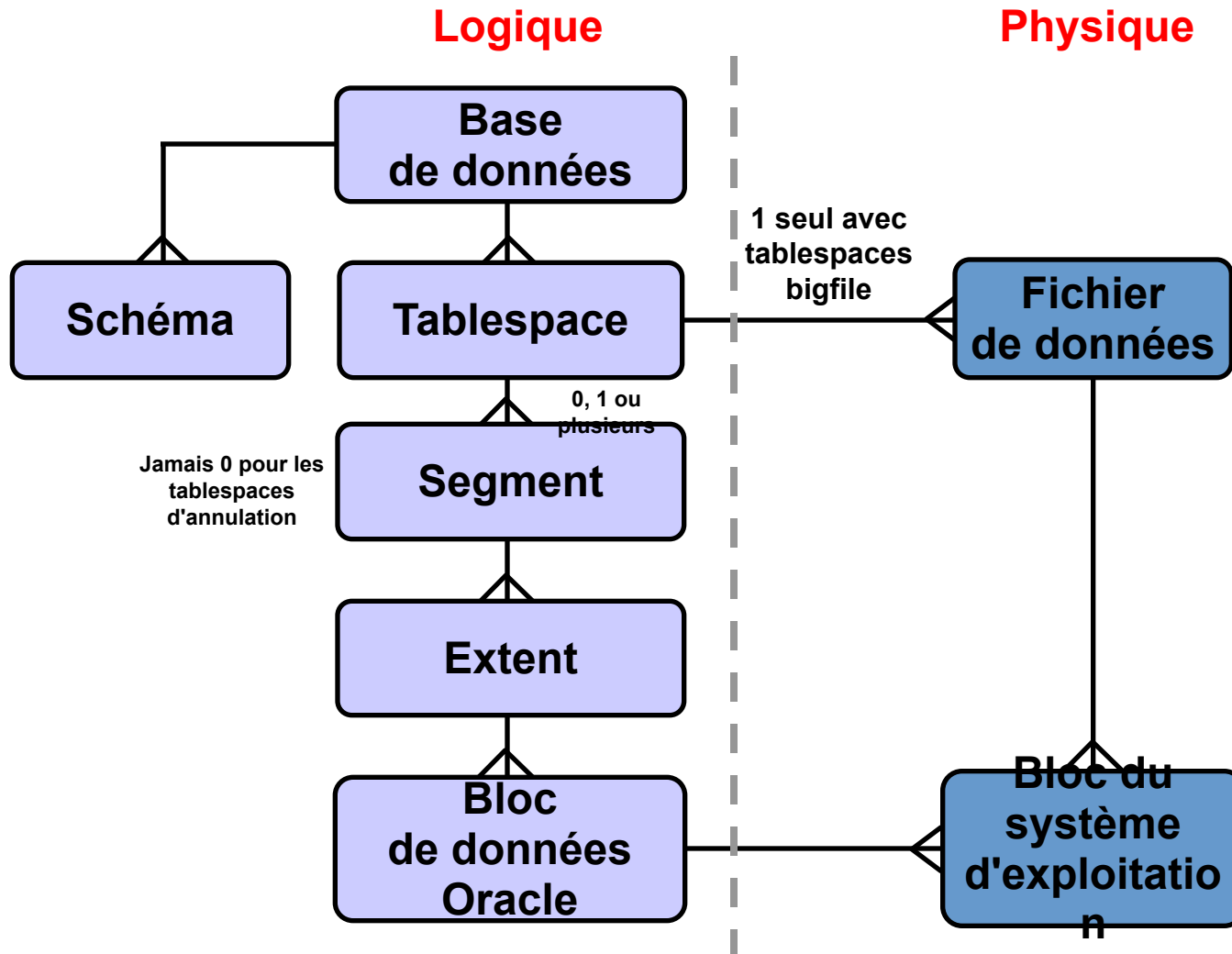
**Les fichiers constituant une base de données Oracle sont organisés de la manière suivante :**

- **Fichiers de contrôle :** Ils contiennent des informations sur la base de données (c'est-à-dire les informations sur sa structure physique). Ces fichiers sont essentiels à la base. Sans eux, vous ne pouvez pas ouvrir de fichiers de données pour accéder aux données contenues dans la base.
- **Fichiers de données :** Ils contiennent les données d'utilisateur ou d'application de la base de données, ainsi que les métadonnées et le dictionnaire de données.
- **Fichiers de journalisation en ligne :** Ils permettent de récupérer la base de données en cas d'échec de l'instance. Si le serveur de base de données tombe en panne sans perdre de fichiers de données, l'instance peut restaurer la base grâce aux informations de ces fichiers.

**Les fichiers supplémentaires suivants sont importants pour la réussite de l'exécution de la base de données :**

- **Fichier de paramètres :** Il est utilisé pour définir la configuration de l'instance au démarrage.
- **Fichier de mots de passe :** Il permet aux utilisateurs `sysdba`, `sysoper` et `sysasm` de se connecter à distance à la base de données et d'effectuer des tâches d'administration.
- **Fichiers de sauvegarde :** Ils sont utilisés pour la récupération de la base de données. Un fichier de sauvegarde est généralement restauré en cas de panne ou lorsqu'une erreur d'un utilisateur a endommagé ou supprimé un fichier d'origine.
- **Fichiers de journalisation archivés :** Ils contiennent un historique mis à jour en permanence des modifications de données (informations de journalisation) qui sont générées par l'instance. A partir de ces fichiers et d'une sauvegarde de la base, vous pouvez récupérer un fichier de données perdu. En d'autres termes, les fichiers de journalisation archivés permettent la récupération de fichiers de données restaurés.

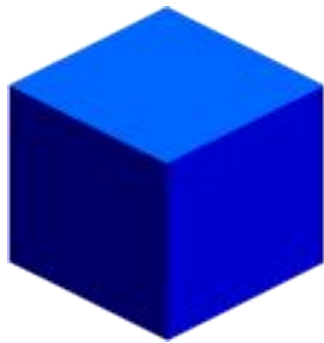
# Structures logiques et physiques de la base de données



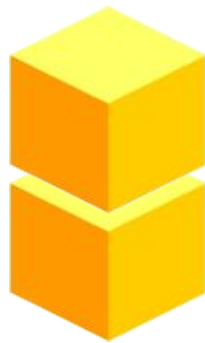


# Segments, extents et blocs

- Les segments sont présents dans un tablespace.
- Les segments sont des ensembles d'extents.
- Les extents sont des ensembles de blocs de données.
- Les blocs de données sont mis en correspondance avec les blocs du disque.



**Segment**



**Extents**



**Blocs de  
données**



**Blocs du  
disque**

# Tablespaces SYSTEM et SYSAUX

- Les tablespaces `SYSTEM` et `SYSAUX` sont obligatoires. Ils sont créés en même temps que la base de données. Ils doivent être en ligne.
- Le tablespace `SYSTEM` est utilisé pour les fonctionnalités principales (les tables du dictionnaire de données, par exemple).
- Le tablespace auxiliaire `SYSAUX` est utilisé pour les composants de base de données supplémentaires (tels que le référentiel Enterprise Manager).

# **Chapitre 02**

## **Outils de réglage et de diagnostic**

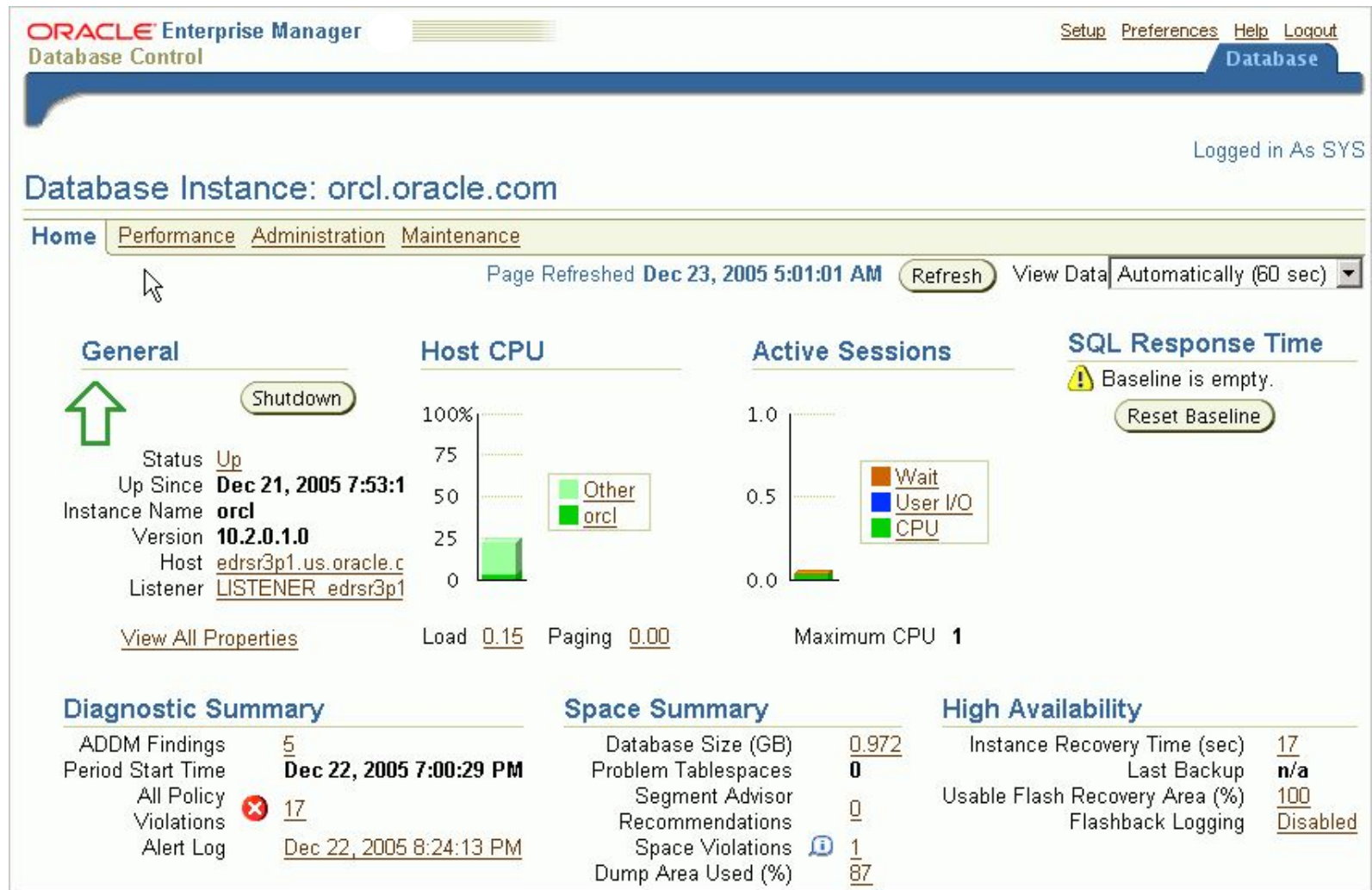
# Outils de réglage des performances

## Outils disponibles :

- **Outils de base :**
  - Pages Oracle Enterprise Manager
  - Fichier d'alertes
  - Fichiers trace
  - Vues et tables dynamiques des performances
- **Module d'extension :**
  - Statspack
  - AWR
  - ADDM
- **Options :**
  - Oracle Diagnostics Pack
  - Oracle Tuning Pack



# Pages Oracle Enterprise Manager sur les performances



# Diagnostic à l'aide du fichier `alert.log`

Les informations contenues dans le fichier `alert.log` peuvent être utilisées pour le réglage de la base de données :

- Contient un journal chronologique de messages et d'erreurs.
- Les messages: « points de reprise incomplets » qui indiquent une *ATTENTE* au niveau du LGWR
- Heures de début et de fin des points de reprise ( que si le paramètre `LOG_CHECKPOINTS_TO_ALERT` a la valeur `TRUE`)
- temps nécessaire à l'archivage
- heures de début et de fin de la récupération d'instance
- erreurs de "verrou mortel" et de dépassement du temps imparti
- les erreurs internes (ORA-600) et autres erreurs . . .

# **Diagnostic à l'aide des fichiers trace de processus en arrière-plan**

- **Le serveur Oracle effectue un dump d'informations dans des fichiers trace, ces données portant sur les erreurs détectées par tout processus en arrière-plan.**
- **En général, ces fichiers ne sont pas exploités par les DBA mais envoyés au support technique Oracle**
- **Le Support technique Oracle utilise ces fichiers trace pour réaliser des diagnostics et résoudre les problèmes.**
- **Ces fichiers sont générés dans le même répertoire que le fichier Alert.log indiqué par  
« background\_dump\_dest »**

# Diagnostic à l'aide des Fichiers trace utilisateur

- Les fichiers trace utilisateur sont propre à chaque session.
- Ils contiennent des statistiques d'exécution des instructions SQL.  
(Telles que: temps d'execution, temps CPU, Nbr de Blocs lu, ...)
- La fonction de trace d'un processus serveur est désactivée par défaut.
- On peut l'activer au niveau de la session ou de l'instance par l'instruction:

- - Dans la session courrante

```
ALTER SESSION SET sql_trace=TRUE;
```

ou

- - Dans la session définie par le couple (sid, serial#)

```
EXEC dbms_system.set_sql_trace_in_session(sid,serila#,TRUE)
```



# Diagnostic et réglage à l'aide des Vues dynamiques v\$

On peut utiliser les vues dynamiques pour:

- Consulter les statistiques concernant l'utilisation mémoire
- Calculer les ratios qui indiquent le niveau de performance de l'instance
- Connaître les attentes au niveau du system
- Consulter les blocages au niveau des sessions
- Connaître les nombres d'E/S au niveau des disques
- ... ect

# Vues dynamiques des performances :

## Exemples d'utilisation

a

```
-- Instructions SQL qui ont consommé un temps CPU > 200 000 ms
SQL> SELECT sql_text, executions
2   FROM v$sqlstats
3   WHERE cpu_time > 200000;
```

b

```
-- Sessions ouvertes depuis l'ordinateur EDRSR9P1 au cours du J-1
SQL> SELECT * FROM v$session
2   WHERE machine = 'EDRSR9P1' and
3   logon_time > SYSDATE - 1;
```

c

```
-- Les sessions détenant actuellement un verrou qui bloque un autre utilisateur et ---
-- depuis combien de temps ce verrou est-il actif ? (block = 1 ou 0, 1 si session bloquante)

SQL> SELECT sid, ctime
2   FROM v$lock WHERE block > 0;
```

# Vues dynamiques des performances :

## Eléments à prendre en compte

- Le propriétaire de ces vues est SYS.
- Différentes vues sont disponibles selon que :
  - l'instance a démarré
  - la base de données est montée
  - la base de données est ouverte
- Vous pouvez interroger `V$FIXED_TABLE` pour afficher les noms de toutes les vues.
- Ces vues sont souvent appelées "vues V\$".
- La cohérence en lecture n'est pas garantie pour ces vues car les données sont dynamiques.

# Dépannage et réglage

## Statistiques relatives à l'ensemble du système

### Instance/Base de données

V\$DATABASE  
V\$INSTANCE  
V\$OPTION  
V\$PARAMETER  
V\$BACKUP  
V\$PX\_PROCESS\_SYSSTAT  
V\$PROCESS  
V\$WAITSTAT  
V\$SYSTEM\_EVENT

### Mémoire

V\$BUFFER\_POOL\_STATISTICS  
T/P  
V\$DB\_OBJECT\_CACHE  
V\$LIBRARYCACHE  
V\$ROWCACHE  
V\$SYSSTAT  
V\$SGASTAT

### Disque

V\$DATAFILE  
V\$FILESTAT  
V\$LOG  
V\$LOG\_HISTORY  
V\$DBFILE  
V\$TEMPFILE  
V\$TEMPSTAT

### Contention

V\$LOCK  
V\$ROLLNAME  
V\$ROLLSTAT  
V\$WAITSTAT  
V\$LATCH

## Statistiques relatives aux sessions

### Utilisateur/Session

V\$LOCK  
V\$OPEN\_CURSOR  
V\$PROCESS  
V\$SORT\_USAGE  
V\$SESSION  
V\$SESSTAT  
V\$TRANSACTION  
V\$SESSION\_EVENT  
V\$SESSION\_WAIT  
V\$PX\_SESSTAT  
V\$PX\_SESSION  
V\$SESSION\_OBJECT\_CACHE

# STATSPACK

## ❑ STATSPACK est un utilitaire Oracle composé de:

- Un schéma utilisateur Oracle « perfstat »
- Des scripts d'administration qui existent dans le répertoire `$ORACLE_HOME/rdbms/admin/sp*.sql`

## ❑ STATSPACK permet de :

- Prendre des clichés « snapshot » des vue V\$ : Sauvegarder les statistiques contenues dans les vues de performances dans les tables du schéma perfstat
- Génère un rapport, en fichier texte, sur les performances de la BD, pour une période donnée, en analysant les « snapshot » pris en début et fin de période

## ❑ Installation de STATSPACK

- Se connecter par SYS et executer le script `spcreate.sql`
- Ce script demandera de saisir le mot de passe , le tablespace par default et le tablespace temporaire par default de l'utilisateur "perfstat"
- Si le script echoue :
  - Consulter le fichier "Alert.log" pour voir et résoudre les erreurs
  - Exécuter le script "spdrop.sql" pour supprimer les objets déjà créés
  - Re-exécuter le script "spcreate.sql"

# Utilisation de STATSPACK

## Pour utiliser STATSPACK :

- On choisit une période de pique durant laquelle la BD est très sollicitée
- On fait une prise de clichées en début et en fin de période

- - Prise de cliché = Collecte des statistiques

```
sqlplus perfstat/perfstat
```

```
sql>exec statspack.snap
```

- On Génère le rapport qui analyse les valeurs des statistiques

- - Génération de rapport

```
sqlplus perfstat/perfstat
```

```
sql>@?\rdbms\admin\spreport.sql
```

- Interpréter le rapport et résoudre les problèmes

# Evénements Wait

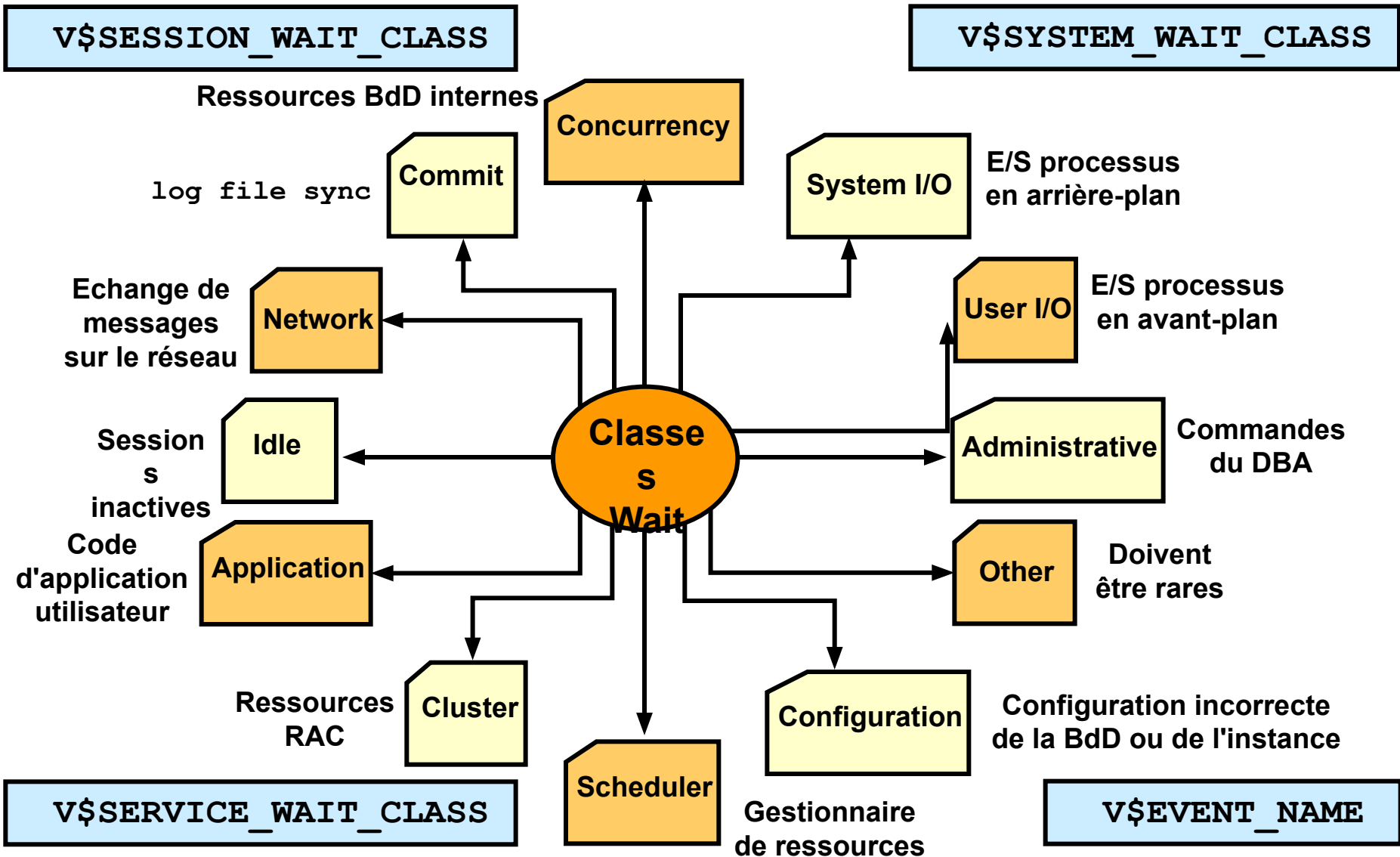
- Une collection d'événements Wait fournit des informations sur les sessions qui ont dû attendre ou doivent attendre pour différentes raisons.
- Ces événements sont répertoriés dans la vue `V$EVENT_NAME`, qui contient les colonnes suivantes :
  - `EVENT#`
  - `NAME`
  - `PARAMETER1`
  - `PARAMETER2`
  - `PARAMETER3`

# Événements Wait courants

Événement Wait	Origine
Buffer busy waits	Cache de tampons, DBWR
Free buffer waits	Cache de tampons, DBWR, E/S
Db file scattered read, Db file sequential read	Réglage des E/S et des instructions SQL
Enqueue waits (enq:)	Verrous externes
Library cache waits	Verrous internes
Log buffer space	E/S tampon de journalisation
Log file sync	Commit trop nombreux, E/S



# Classes Wait



# **Outils de supervision et diagnostic (Suite)**

## **Mesures, Alertes, AWR et ADDM**

# Mesures, alertes

**Une Mesure** : Taux de variation d'une statistique cumulée

**Alerte** : Événement généré quand une mesure surveillée atteint un seuil d'avertissement ou un seuil critique

**Vu que les statistiques ont des valeurs cumulées depuis le démarrage de la BD? Il convient de les comparer entre deux moments distincts.**

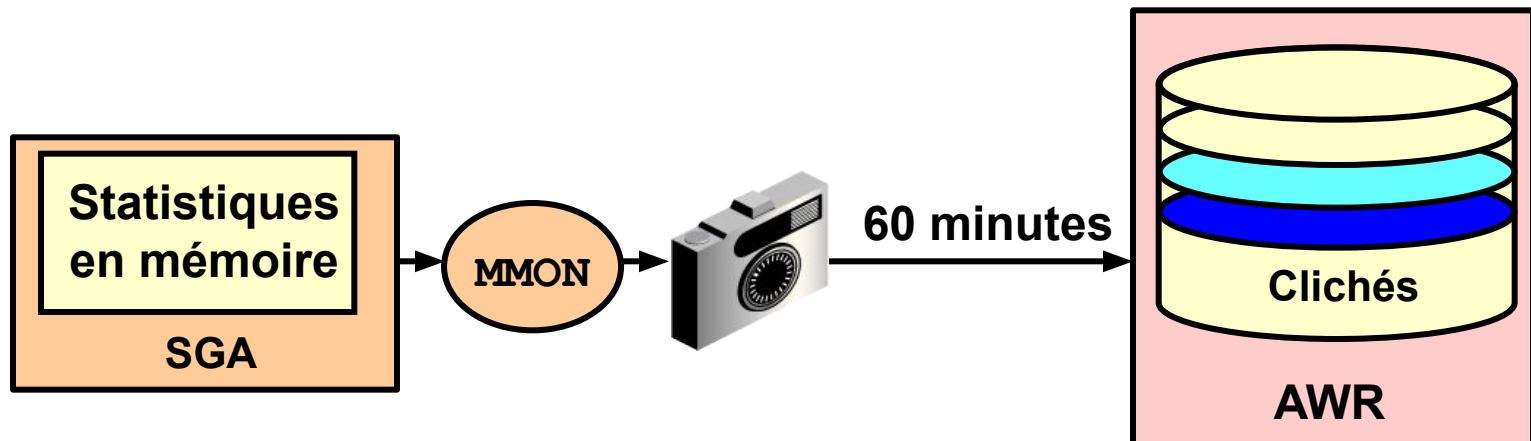
**Les outils suivants produisent des deltas entre 2 périodes :**

- **Etats AWR**
- **Statspack**
- **Scripts personnalisés**

# Référentiel AWR

## (Automatic Workload Repository)

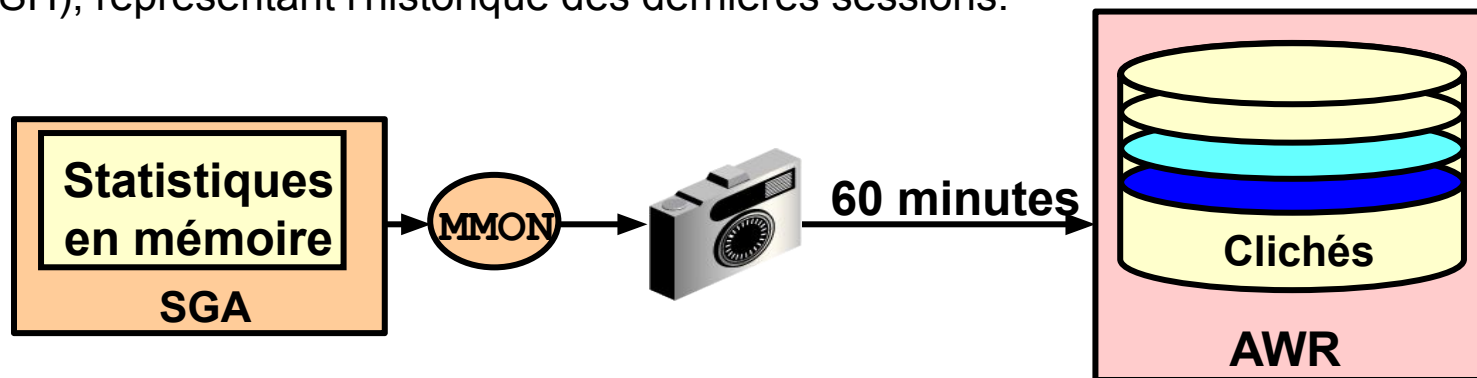
- Le référentiel AWR est un composant intégré à la BD qui permet de **collecter**, de **traiter** et **tenir à jour** des **statistiques de performances** en vue de la détection des problèmes et du réglage automatique. (Self Tuning)
- La collecte se fait automatiquement à partir des vues V\$ par le processus MMON (Manageability Monitor) sur une fréquence spécifique.
- Les statistiques récoltées sont stockées sous forme de clichés / snapshots dans le référentiel en vue de les utiliser



# Les statistiques du référentiel AWR

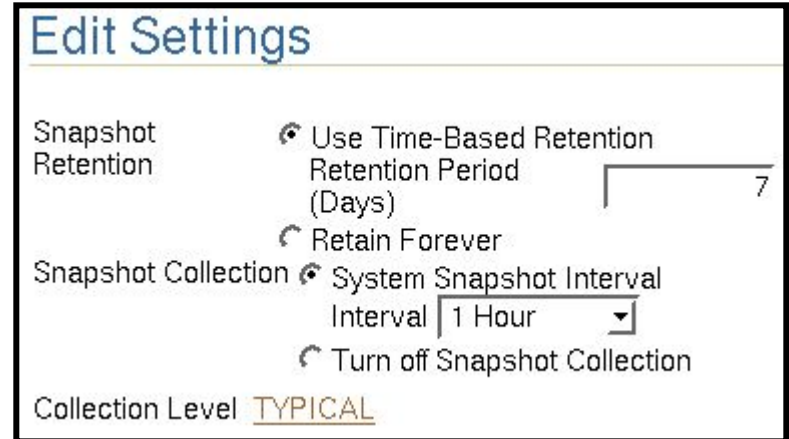
## Les Types de statistiques récoltées:

- **Les statistiques sur les objets**, qui mesurent l'accès aux segments de BD et leur utilisation.
- **Les statistiques temporelles** (Time Model) basées sur l'utilisation du temps pour les activités. Elles s'affichent dans les vues `V$SYS_TIME_MODEL` et `V$SESS_TIME_MODEL`.
- **Quelques-unes des statistiques relatives au système et à la session**, collectées dans les vues `V$SYSSTAT` et `V$SESSTAT`.
- **Les instructions SQL qui génèrent la plus grosse charge sur le système**, en fonction de critères tels que le temps écoulé, le temps CPU, les succès en mémoire tampon (buffer gets), etc.
- **Les statistiques sur l'historique des sessions actives** (Active Session History - ASH), représentant l'historique des dernières sessions.



# Gérer le référentiel AWR

- **Période de conservation:**
  - Par défaut c'est 7 jours
  - Tenez compte des besoins en termes de stockage.
- **Intervalle de collecte**
  - Par défaut, chaque 60 minutes.
  - Tenez compte des besoins en termes de stockage et de l'impact sur les performances.
- **Niveau de collecte des statistiques**
  - **Basic** (désactive la plupart des fonctionnalités ADDM).
  - **Typical** (recommandé pour ne pas impacter les performances).
  - **All** (ajoute aux clichés des informations complémentaires de réglage des instructions SQL).



The screenshot shows the 'Edit Settings' dialog box with the following configuration:

- Snapshot Retention:** ☒ Use Time-Based Retention, Retention Period (Days) set to 7.
- Snapshot Collection:** ☒ System Snapshot Interval, Interval set to 1 Hour.
- Collection Level:** TYPICAL

# Enterprise Manager et référentiel AWR

Paramètres des clichés AWR via l'interface Enterprise Manager

The diagram illustrates the navigation path within the Enterprise Manager interface. It starts with a box labeled "Statistics Management" containing two sub-items: "Automatic Workload Repository" and "Manage Optimizer Statistics". An arrow points from the "Automatic Workload Repository" item to a larger window titled "Automatic Workload Repository".

**Automatic Workload Repository**

The Automatic Workload Repository is used for storing data

**General**

[Edit](#)

Snapshot Retention (days) **7**  
Snapshot Interval (minutes) **60**  
Collection Level **TYPICAL**  
Next Snapshot Capture Time **Jun 3, 2005 6:00:15 PM**

**Manage Snapshots and Preserved Snapshot Sets**

Snapshots 29  
Preserved Snapshot Sets 0  
Latest Snapshot Time **Jun 3, 2005 5:00:15 PM**  
Earliest Snapshot Time **Jun 2, 2005 1:00:43 PM**

# Paramètres des clichés AWR

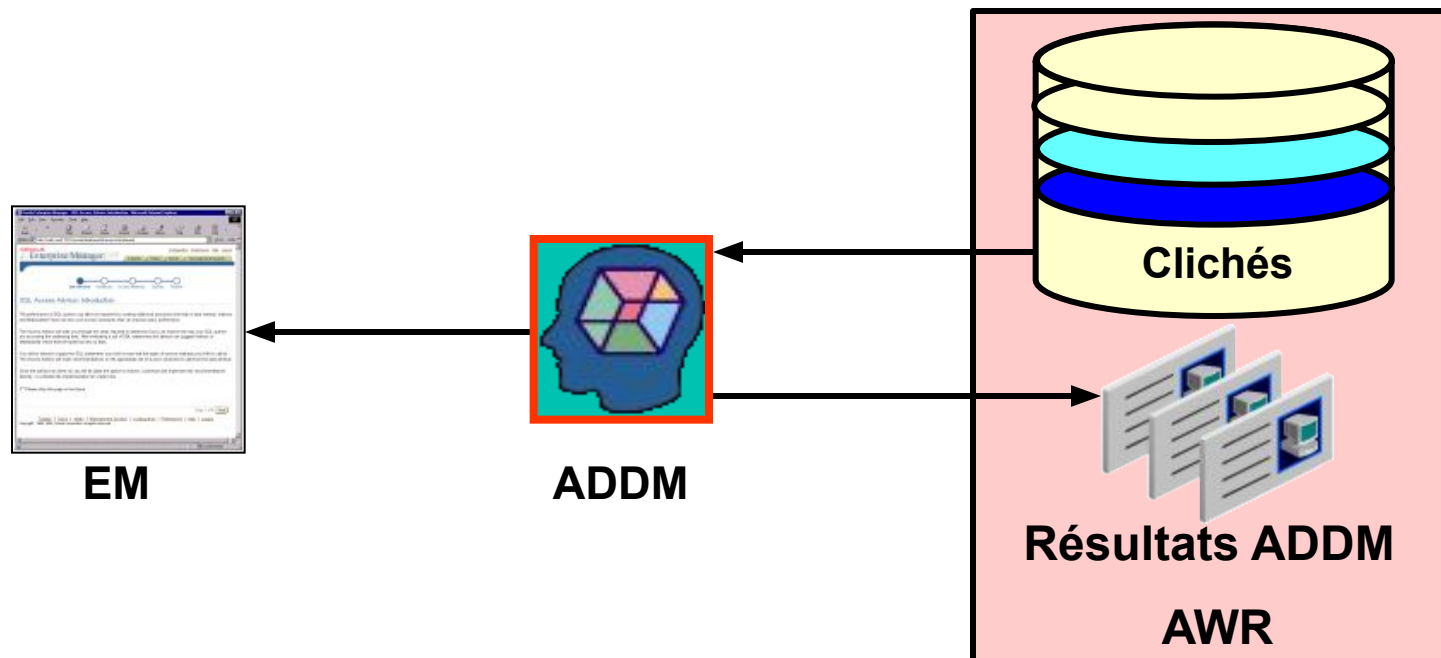
Paramètres des clichés AWR via des commandes SQL

```
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS ( -  
    retention IN NUMBER DEFAULT NULL,  
    interval  IN NUMBER DEFAULT NULL,  
    topnsql   IN NUMBER DEFAULT NULL) ;
```



# Moniteur ADDM (Automatic Database Diagnostic Monitor)

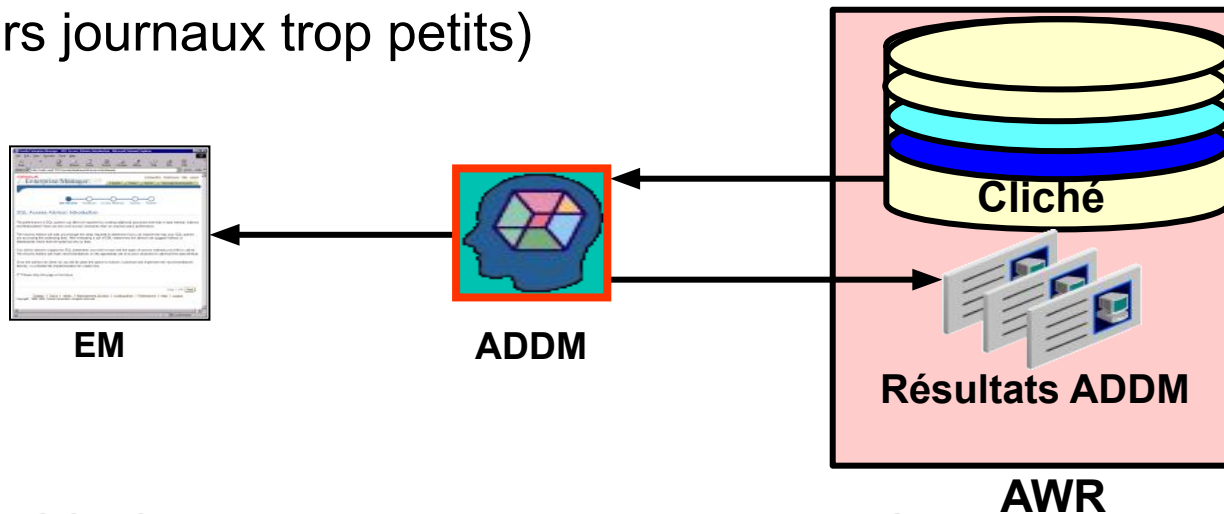
- S'exécute automatiquement après chaque prise de cliché AWR
- Il procède à l'analyse de la période correspondant aux deux derniers clichés.
- Surveillance de l'instance et détection des goulots d'étranglement
- Stockage des résultats dans le référentiel **AWR**



# Moniteur ADDM (Automatic Database Diagnostic Monitor)

Voici quelques-uns des problèmes courants détectés par le moniteur ADDM :

- Goulets d'étranglement CPU
- Gestion inefficace des connexions Oracle Net
- Contention liée à un verrou
- Capacité d'entrée/sortie (E/S)
- Sous-dimensionnement des structures mémoire Oracle
- Instructions SQL à forte consommation de ressources
- Temps PL/SQL et Java élevé
- Nombreux points de reprise (checkpoints) et cause (par exemple, des fichiers journaux trop petits)



# Recommendations ADDM

Database Instance: [EDRSR14P1\\_orcl.oracle.com](#) > [Advisor Central](#) > [Automatic Database Diagnostic Monitor \(ADDM\)](#) >

Performance Finding Details

Logged in As SYS




## Performance Finding Details

Database Time (minutes)	<b>2.6</b>	Period Start Time	<b>Apr 12, 2005 5:10:25 AM PDT</b>	Period Duration (minutes)	<b>3.5</b>
Task Owner	<b>ADDM</b>	Task Name	<b>ADDM:1082506989_1_47</b>	Average Active Sessions	<b>0.7</b>

Finding **Read and write contention on database blocks was consuming significant database time.**  
 Impact (minutes) **0.6**  
 Impact (%)  25.2

## Recommendations

[Show All Details](#) | [Hide All Details](#)

Details	Category	Benefit (%) ▾
<a href="#">Hide</a>	Schema	 25.2
Action	<b>Consider using ORACLE's recommended solution of automatic segment space management in a locally managed tablespace for the tablespace "TBSADDM" containing the TABLE "ADDM.ADDM" with object id 54441. Alternatively, you can move this object to a different tablespace that is locally managed with automatic segment space management.</b> Database Object <a href="#">ADDM.ADDM</a>	
Rationale	<b>There was significant read and write contention on TABLE "ADDM.ADDM" with object id 54441.</b> Database Object <a href="#">ADDM.ADDM</a>	
<a href="#">Show</a>	Schema	 25.2
<a href="#">Show</a>	Schema	 25.2

## Findings Path

[Expand All](#) | [Collapse All](#)

Findings	Impact (%)	Additional Information
<a href="#">Hide</a> Read and write contention on database blocks was consuming significant database time.	 25.2	
Wait class "Concurrency" was consuming significant database time.	 61.1	

# Définir des seuils

Database Instance: orcl > [Manage Metrics](#) > Edit Thresholds

## Edit Thresholds

You can set a warning and critical threshold for each of the metrics below. When a threshold is reached, an alert will be generated and the response action, if specified, executed. The response action can be any command or script, with a fully qualified path, that is accessible to the Management Agent.

Cancel

OK

☒ **TIP** Some metrics do not allow a default set of thresholds for all their monitored objects. Click "Specify Multiple Thresholds" to set thresholds for specific objects.

Related Link [Response to Target Down](#)

Specify Multiple Thresholds

Select Metric	Comparison Operator	Warning Threshold	Critical Threshold	Response Action
<input checked="" type="radio"/> Archive Area Used (%)	>	80		
<input type="radio"/> Archiver Hung Alert Log Error	Contains		ORA-	
<input type="radio"/> Archiver Hung Alert Log Error Status	>	0		
<input type="radio"/> Audited User	=	SYS		
<input type="radio"/> Average File Read Time (centi-seconds)	>			
<input type="radio"/> Average File Write Time (centi-seconds)	>			
<input type="radio"/> Average Users Waiting Count				
<input type="radio"/> Administrative	>	10		
<input type="radio"/> Application	>	10		
<input type="radio"/> Cluster	>	30		
<input type="radio"/> Commit	>	30		

# Créer et tester une alerte

1. **Spécifiez un seuil.**
2. **Créez un scénario de test.**
3. **Vérifiez le fonctionnement de l'alerte.**

Show SQL

2

```
CREATE TABLE "HR"."FILLER" ( "EMPLOYEE_ID" NUMBER(6), "FIRST_NAME"
VARCHAR2(20), "LAST_NAME" VARCHAR2(25), "EMAIL" VARCHAR2(25),
"PHONE_NUMBER" VARCHAR2(20), "HIRE_DATE" DATE, "JOB_ID" VARCHAR2(10),
"SALARY" NUMBER(8, 2), "COMMISSION_PCT" NUMBER(2, 2), "MANAGER_ID"
NUMBER(6), "DEPARTMENT_ID" NUMBER(4)) TABLESPACE "INVENTORY" PCTFREE 10
INITRANS 1 MAXTRANS 255 STORAGE ( INITIAL 64K BUFFER_POOL DEFAULT)
NOLOGGING
```

## Tablespace Full Metric Thresholds

Monitor the fullness of the tablespace using either of

### Space Used (%)

A warning or critical alert will be generated if the percentage of space used exceeds the corresponding threshold.

☐ Use Database Default Thresholds

Modify

Warning (%) 85

Critical (%) 97

☒ Specify Thresholds

Warning (%) 70

Critical (%) 75

☐ Disable Thresholds

1

## Alerts

3

Category All Go Critical ✖ 1 Warning ⚠ 1

Severity	Category	Name	Message	Alert Triggered
<span style="color: red;">✖</span>	Tablespaces Full	Tablespace Space Used (%)	Tablespace INVENTORY is 98 percent full	Jun 3, 2005 10:44:04 AM
<span style="color: yellow;">⚠</span>	User Audit	Audited User	User SYS logged on from EDRSR30P1.	Jun 3, 2005 8:25:04 AM



# Générer des états AWR dans Enterprise Manager (1/2)

**Automatic Workload Repository**

Database Instance: EDRSR14P1\_orcl

The Automatic Workload Repository is

**General**

Snapshot Retention (days) **7**  
Snapshot Interval (minutes) **60**  
Collection Level **TYP**  
Next Snapshot Capture Time **Apr 13, 2005 8:00 AM**

**Manage Snapshots and Preserved Snapshots**

Snapshots **51**  
Preserved Snapshot Sets **0**  
Latest Snapshot Time **Apr 13, 2005 8:00 AM**

**Snapshots**

A snapshot is a collection of database statistics at a single point in time. You can use the information in snapshots to diagnose database problems.

Page Refreshed **Apr 13, 2005 8:00 AM**

**Select Beginning Snapshot**

Go To Time      
(Example: 12/15/03)

Select	ID	Capture Time	Collection Level	Within A Preserved Snapshot Set
<input checked="" type="radio"/>	49	Apr 12, 2005 5:50:48 AM	TYPICAL	
<input type="radio"/>	50	Apr 12, 2005 7:00:16 AM	TYPICAL	
<input type="radio"/>	51	Apr 12, 2005 8:10:55 AM	TYPICAL	

**View Report**

Beginning Snapshot ID **49**  
Beginning Snapshot Capture Time **Apr 12, 2005 5:50:48 AM**

**Select Ending Snapshot**

Go To Time      
(Example: 12/15/03)

Select	ID	Capture Time	Collection Level	Within A Preserved Snapshot Set
<input checked="" type="radio"/>	50	Apr 12, 2005 7:00:16 AM	TYPICAL	

# Générer des états AWR dans Enterprise Manager (2/2)

Le référentiel AWR peut produire un état récapitulatif similaire à l'état Statspack sur les statistiques stockées dans le référentiel de charge globale (Workload Repository).

Cet état contient des informations générales sur le comportement d'ensemble du système pendant une période délimitée par deux clichés (snapshots).

Pour générer un état AWR:

vous pouvez utiliser la page Automatic Workload Repository de Database Control et cliquer sur le lien correspondant au nombre de clichés.

WORKLOAD REPOSITORY report for						
DB Name	DB Id	Instance	Inst num	Release	RAC	Host
ORCL	1090770270	orcl	1	10.2.0.0.0	NO	edrsr14p1
		Snap Id	Snap Time	Sessions	Cursors/Session	
Begin Snap:	140	21-Mar-05 23:00:46	25	10.8		
End Snap:	141	22-Mar-05 00:00:13	27	12.4		
Elapsed:		59.46 (mins)				
DB Time:		0.78 (mins)				

## Report Summary

### Cache Sizes

	Begin	End		
Buffer Cache:	44M	44M	Std Block Size:	8K
Shared Pool Size:	208M	208M	Log Buffer:	256K

### Load Profile

# **Chapitre 03**

## **Régler les instructions SQL**

### **Plan d'exécution et les statistiques de l'optimiseur**



# Qu'est-ce qu'un plan d'exécution ?

- Le plan d'exécution d'une instruction SQL est composé de petits composants de base appelés row sources pour les plans d'exécution en série.
- La combinaison des row sources associés à une instruction est appelée le plan d'exécution.
- En utilisant des relations parent-enfant, le plan d'exécution peut être affiché dans une structure arborescente (texte ou graphique).



# L'optimiseur d'instructions SQL

**L'optimiseur est un composant oracle qui:**

- S'exécute dans la phase d'analyse pour déterminer **le plan** le plus efficace pour l'exécution d'une instruction SQL.
- **Le plan** d'exécution est l'ensemble d'opérations qui sont effectuées pour extraire les données requises de manière efficace.
- Pour déterminer le meilleur chemin/plan possible, l'optimiseur utilise plusieurs types d'information :
  - Conseils (Hints) fournis par le développeur au niveau des instructions SQL
  - Statistiques
  - Informations provenant du dictionnaire
  - Clause WHERE

# Modes de fonctionnement de l'optimiseur

A partir de Oracle10g, oracle propose deux modes de fonctionnement de l'optimiseur :

- L'optimiseur basé sur les règles :
  - utilise un système de classement,  
(privilégie l'utilisation des indexes s'ils existent)
  - orienté syntaxe et dictionnaire de données.
- L'optimiseur basé sur le coût :
  - choisit le chemin dont le coût est le plus faible,
  - orienté statistiques.

# Définir le mode de fonctionnement de l'optimiseur

- **Au niveau de l'instance :**
  - `Alter system set optimizer_mode = {choose|rule|first_rows|first_rows_n|all_rows}`
- **Au niveau de la session :**
  - `alter session set optimizer_mode = {choose|rule|first_rows|first_rows_n|all_rows}`
- **Au niveau des instructions :**
  - Utilisez des conseils  
`Select /*+ all_rows */ id,nom from ...`

# Optimiseur basé sur les règles

Pour établir le plan optimal, l'optimiseur se base uniquement sur des **règles** prédéfinies par Oracle. Ces règles se basent sur:

- La syntaxe : l'ordre des tables dans les jointures

- Le système de classement :

R1: Utiliser toujours, en premier, index unique s'il existe

R2: Utiliser index non unique

.....

R3: Utiliser Accès complet à la table

**Au niveau de la session uniquement:**

- `ALTER SESSION optimizer_mode = RULE;`

**Au niveau de tout le system (toutes les sessions)**

- `ALTER SYSTEM optimizer_mode = RULE;`

**Au niveau d'instruction SQL via les Hints**

- `Select /*+ Rule */ id, nom from tab_1 ...`

# CBO: Cost Based Optimizer

## L'optimiseur basé sur les Coûts

- Pour établir le plan optimal, l'optimiseur se base sur **le coût** de chaque Plan et choisi le plan qui a **le moindre coût**.
- Pour calculer les coûts, oracle se base sur **des statistiques** appelées « **statistiques de l'optimiseurs** » ou « **statistiques Objets** »
- Les statistiques objets ne sont pas calculés **automatiquement**, il faut programmer leur calcul périodiquement.
- Si les statistiques ne sont pas à jours les coûts seront erronés et le plan choisi ne sera pas optimal

### Au niveau de la session uniquement:

- `ALTER SESSION optimizer_mode = {first_rows|first_rows_n|all_rows};`

### Au niveau de tout le system (toutes les sessions)

- `ALTER SYSTEM optimizer_mode = {first_rows|first_rows_n|all_rows};`

### Au niveau d'instruction SQL via les Hints

- `Select /*+ all_rows */ id, nom from tab_1 ...`

# Optimiseur : Option « Choose »

La valeur « choose » de l'optimizer\_mode indique à l'optimiseur de requêtes d'utiliser les règles dans le cas où les statistiques n'existent pas pour certains objets

## **Au niveau de la session uniquement:**

- `ALTER SESSION optimizer_mode = Choose;`

## **Au niveau de tout le system (toutes les sessions)**

- `ALTER SYSTEM optimizer_mode = Choose;`

## **Au niveau d'instruction SQL via les Hints**

- `Select /*+ choose */ id, nom from tab_1 ...`

# Où trouver les plans d'exécution ?

- `PLAN_TABLE` (SQL Developer ou SQL\*Plus)
- `V$SQL_PLAN` (cache "library")
- `V$SQL_PLAN_MONITOR` (Oracle Database 11g)
- `DBA_HIST_SQL_PLAN` (référentiel AWR)
- `STATS$SQL_PLAN` (Statspack)
- Base de gestion SQL (SMB) (SQL Plan Baselines)
- SQL Tuning Set (STS - Ensemble de réglages SQL)
- Fichiers trace générés par `DBMS_MONITOR`
- Fichier trace de l'événement 10053
- Fichier trace d'un dump de l'état des processus, à partir de la version Oracle Database 10g Release 2



# Afficher des plans d'exécution

- **Commande EXPLAIN PLAN suivie par :**
  - `SELECT from PLAN_TABLE`
  - `DBMS_XPLAN.DISPLAY()`
- **Autotrace SQL\*Plus : SET AUTOTRACE ON**
- `DBMS_XPLAN.DISPLAY_CURSOR()`
- `DBMS_XPLAN.DISPLAY_AWR()`
- `DBMS_XPLAN.DISPLAY_SQLSET()`
- `DBMS_XPLAN.DISPLAY_SQL_PLAN_BASELINE()`

# Visualiser le Plan d'exécution par EXPLAIN PLAN

(1°) Créer la table `PLAN_TABLE` à l'aide du script :

```
SQL> @?\rdbms\admin\utlxplan.sql
```

(2°) Exécuter l'outil Explain Plan pour l'instruction SQL :

```
SQL> Explain plan for  
2 select last_name from hr.employees;
```

(3°) Interrogez la table `plan_table` pour afficher les plans d'exécution de l'une des manières suivantes:

- Interrogez `PLAN_TABLE` directement par select
- Utilisez le script `utlxpls.sql` (informations Parallel Query masquées)
- Utilisez le script `utlxplp.sql` (informations Parallel Query affichées)
- `DBMS_XPLAN.DISPLAY()`

**Remarque:**

- Cet outil peut être utilisé avec ou sans la fonction de trace.
- L'instruction SQL n'est pas réellement exécutée

# Outil Explain Plan dans SQL Developer

The screenshot shows the SQL Developer interface with a query window and an Explain Plan window. The query is:

```
select e.email, d.department_name
FROM EMPLOYEES e, departments d
where email like 'A%';
```

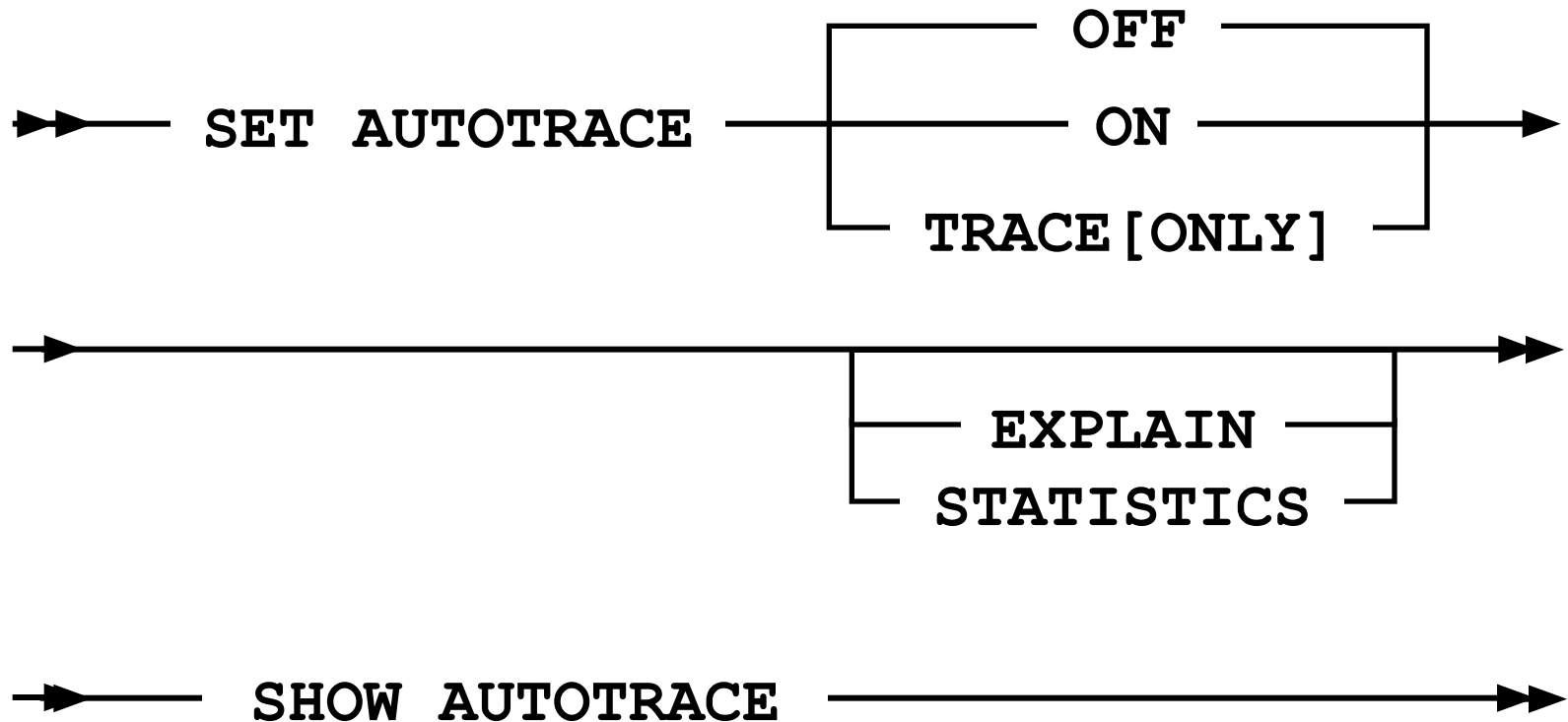
The Explain Plan window shows the execution plan for the query, with a total cost of 0.007 seconds. The plan is as follows:

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			8
MERGE JOIN		CARTESIAN	8
INDEX	EMP_EMAIL_UK	RANGE SCAN	1
Access Predicates		EMAIL LIKE 'A%'	
Filter Predicates		EMAIL LIKE 'A%'	
BUFFER		SORT	7
TABLE ACCESS	DEPARTMENTS	FULL	2

# Fonction AUTOTRACE

- **AUTOTRACE est une fonction de SQL\*Plus et de SQL Developer.**
- **Elle a été proposée à partir de la version Oracle 7.3.**
- **Elle a besoin d'une table `PLAN_TABLE`.**
- **Elle nécessite le rôle `PLUSTRACE` pour extraire des statistiques de certaines vues `V$`.**
- **Par défaut, elle génère le plan d'exécution et des statistiques après l'exécution d'une interrogation.**
- **Elle peut ne pas représenter le plan réel utilisé par l'optimiseur en cas d'examen des variables attachées  
(bind variables) (commande `EXPLAIN PLAN` récursive).**

# Syntaxe AUTOTRACE



# Commande AUTOTRACE : Exemples

- Démarrer la trace des instructions à l'aide d'AUTOTRACE :

```
SQL> set autotrace on
```

- Afficher le plan d'exécution sans exécuter l'instruction :

```
SQL> set autotrace traceonly explain
```

- Afficher les lignes et les statistiques :

```
SQL> set autotrace on statistics
```

- Afficher le plan et les statistiques seulement

```
SQL> set autotrace traceonly
```

# Commande AUTOTRACE : Statistiques

```
SQL> show autotrace
autotrace OFF
SQL> set autotrace traceonly statistics
SQL> SELECT * FROM oe.products;
```

288 rows selected.

## Statistics

```
-----
      1334   recursive calls
           0   db block gets
      686   consistent gets
      394   physical reads
           0   redo size
103919   bytes sent via SQL*Net to client
      629   bytes received via SQL*Net from client
       21   SQL*Net roundtrips to/from client
       22   sorts (memory)
           0   sorts (disk)
      288   rows processed
```

# AUTOTRACE dans SQL Developer

hr x | scott x | sys\_connection x

0.5 seconds

```
select e.email, d.department_name
FROM EMPLOYEES e, departments d
where email like 'A%';
```

Script Output x | Autotrace x

0.5 seconds

OPERATION	OBJECT_NAME	COST	LAST_CR_BUFFER_GETS
SELECT STATEMENT		8	
MERGE JOIN CARTESIAN		8	8
INDEX RANGE SCAN	EMP_EMAIL_UK	1	1
Access Predicates			
EMAIL LIKE 'A%'			
Filter Predicates			

V\$STATNAME Name	V\$MYSTAT Value
recursive calls	1
db block gets	0
consistent gets	8
physical reads	0
redo size	0
bytes sent via SQL*Net to client	1872
bytes received via SQL*Net from client	658
SQL*Net roundtrips to/from client	2
sorts (memory)	2
sorts (disk)	0



# Interprétation de plan d'exécution :

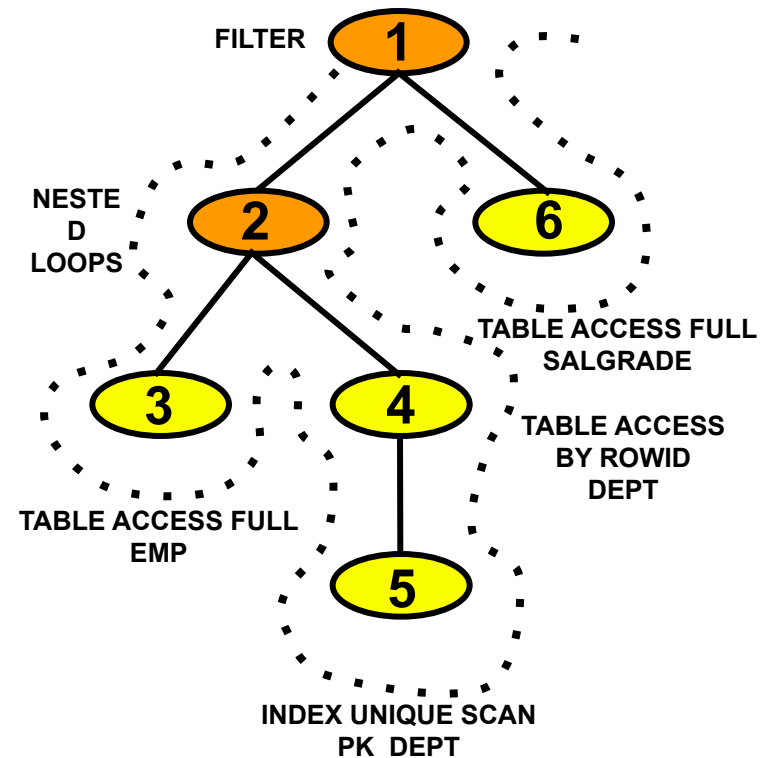
## Exemple 1

```
SELECT /*+ RULE */ ename,job,sal,dname
FROM emp,dept
WHERE dept.deptno=emp.deptno and not exists(SELECT *
                                           FROM salgrade
                                           WHERE emp.sal between losal and hisal);
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	FILTER	
2	NESTED LOOPS	
3	TABLE ACCESS FULL	EMP
4	TABLE ACCESS BY INDEX ROWID	DEPT
* 5	INDEX UNIQUE SCAN	PK_DEPT
* 6	TABLE ACCESS FULL	SALGRADE

Predicate Information (identified by operation id):

- 1 - filter( NOT EXISTS  
(SELECT 0 FROM "SALGRADE" "SALGRADE" WHERE  
"HISAL">=:B1 AND "LOSAL"<=:B2))
- 5 - access("DEPT"."DEPTNO"="EMP"."DEPTNO")
- 6 - filter("HISAL">=:B1 AND "LOSAL"<=:B2)



# Interprétation de plan d'exécution :

```
SQL> alter session set statistics_level=ALL;
```

Session altered.

```
SQL> select /*+ RULE to make sure it reproduces 100% */ ename,job,sal,dname
from emp,dept where dept.deptno = emp.deptno and not exists (select * from salgrade
where emp.sal between losal and hisal);
```

no rows selected

```
SQL> select * from table(dbms_xplan.display_cursor(null,null,'TYPICAL IOSTATS
LAST'));
```

SQL\_ID 274019myw3vuf, child number 0

-----

...

Plan hash value: 1175760222

-----						
Id		Operation	Name	Starts	A-Rows	Buffers
-----						
*	1	FILTER		1	0	61
	2	NESTED LOOPS		1	14	25
	3	TABLE ACCESS FULL	EMP	1	14	7
	4	TABLE ACCESS BY INDEX ROWID	DEPT	14	14	18
*	5	INDEX UNIQUE SCAN	PK_DEPT	14	14	4
*	6	TABLE ACCESS FULL	SALGRADE	12	12	36
-----						

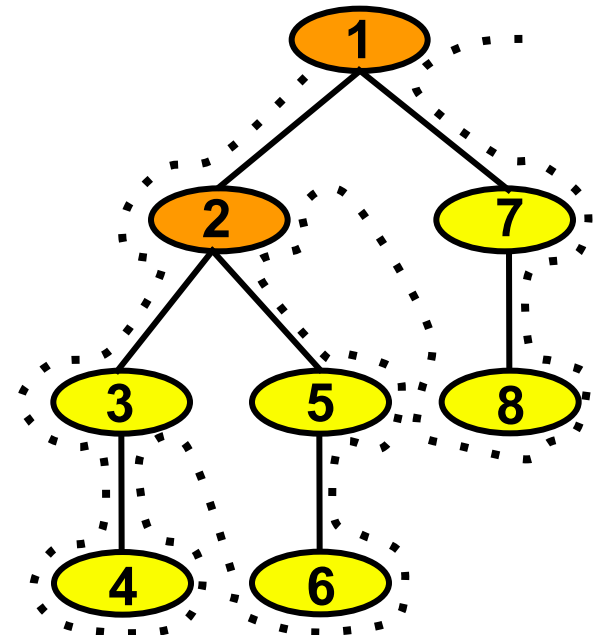
...

# Interprétation de plan d'exécution :

## Exemple 2

```
SQL> select /*+ USE_NL(d) use_nl(m) */ m.last_name as dept_manager
  2   ,      d.department_name
  3   ,      l.street_address
  4 from    hr.employees m   join
  5         hr.departments d on (d.manager_id = m.employee_id)
  6         natural join
  7         hr.locations l
  8 where   l.city = 'Seattle';
```

```
0  SELECT STATEMENT
1 0  NESTED LOOPS
2 1  NESTED LOOPS
3 2  TABLE ACCESS BY INDEX ROWID LOCATIONS
4 3  INDEX RANGE SCAN          LOC_CITY_IX
5 2  TABLE ACCESS BY INDEX ROWID DEPARTMENTS
6 5  INDEX RANGE SCAN          DEPT_LOCATION_IX
7 1  TABLE ACCESS BY INDEX ROWID EMPLOYEES
8 7  INDEX UNIQUE SCAN         EMP_EMP_ID_PK
```



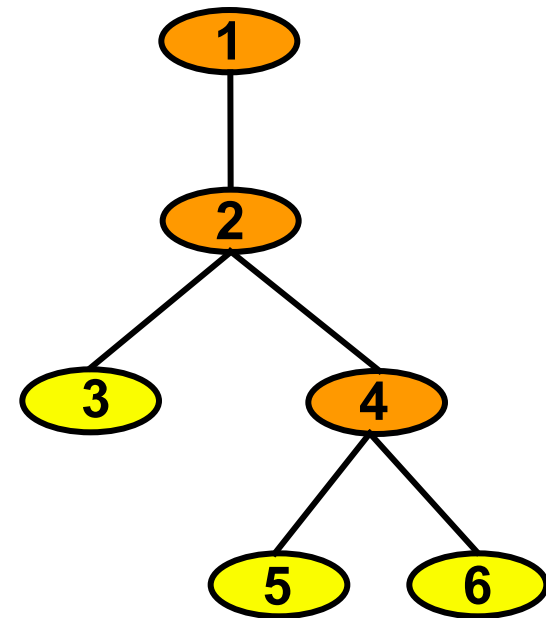
# Interprétation de plan d'exécution :

## Exemple 3

```
select /*+ ORDERED USE_HASH(b) SWAP_JOIN_INPUTS(c) */ max(a.i)
from t1 a, t2 b, t3 c
where a.i = b.i and a.i = c.i;
```

0	SELECT STATEMENT
1	SORT AGGREGATE
2 1	HASH JOIN
3 2	TABLE ACCESS FULL T3
4 2	HASH JOIN
5 4	TABLE ACCESS FULL T1
6 4	TABLE ACCESS FULL T2

<a href="#">Expand All</a>	<a href="#">Collapse All</a>		
Operation	Object	Order	
▼ SELECT STATEMENT		7	
▼ SORT AGGREGATE		6	
▼ HASH JOIN		5	
TABLE ACCESS FULL	<a href="#">T3</a>	1	
▼ HASH JOIN		4	
TABLE ACCESS FULL	<a href="#">T1</a>	2	
TABLE ACCESS FULL	<a href="#">T2</a>	3	



L'ordre de jointure est :  
**T1 - T2 - T3**

# Examiner le plan d'exécution

- **Partez de la table qui présente le filtre le plus sélectif.**
- **Examinez les facteurs suivants :**
  - **La table directrice présente le meilleur filtre.**
  - **Le nombre le plus limité de lignes est renvoyé à l'étape suivante.**
  - **La méthode de jointure convient pour le nombre de lignes renvoyées.**
  - **Les vues sont correctement utilisées.**
  - **Les produits cartésiens sont involontaires.**
  - **L'accès aux tables est efficace.**

# Les statistiques de l'optimiseur

- Ensemble d'informations sur un objet (table ou index) utilisé pour calculer le coût d'un plan d'exécution
- Le CBO : se base sur les statistiques pour calculer les coût des différents plan d'exécution
- Si les statistiques ne sont pas à jour les coût ne seront pas précis, par conséquent le plan choisie ne sera pas forcément le plus optimal
- Doivent être mises à jour assez fréquemment

# Calcul des statistiques de l'optimiseur

- Normalement les statistiques sont générées automatiquement par Oracle, mais on doit aussi les mettre à jours si nécessaire via :
  - Le package DBMS\_STATS :
    - `GATHER_TABLE_STATS`
    - `GATHER_INDEX_STATS`
    - `GATHER_SCHEMA_STATS`
    - `GATHER_DATABASE_STATS`
  - Commande Analyze :
    - `Analyze table nomTab compute statistics;`

# Utilisation des statistiques de l'optimiseur

## Statistiques relatives aux tables

- Nombre de lignes
- Nombre de blocs et de blocs vides
- Quantité moyenne d'espace disponible
- Nombre de lignes chaînées ou migrées
- Longueur moyenne des lignes
- Date de la dernière utilisation de la commande **ANALYZE** et taille de l'échantillon
- Vue du dictionnaire de données : **DBA\_TABLES**



# Utilisation des statistiques de l'optimiseur

## Statistiques relatives aux tables

```
select num_rows,blocks,empty_blocks  
       avg_space,avg_row_len,last_analyzed  
From dba_tables  
where  table_name = 'ORDERS' and owner = 'SH' ;
```

- **Num\_rows**: nbr de lignes dans la table
- **Last\_analyzed**: dernière date et heure de collecte des statistiques
- **Avg\_row\_len**: taille moyenne d'une ligne dans la table
- **Blocks** : nbr de block alloués pour la table
- **Empty\_blocks**: nbr de block vide alloués pour la table

# Aller au-delà des plans d'exécution

- **Un plan d'exécution seul ne peut vous indiquer s'il est bon ou pas.**
- **Il peut être nécessaire d'effectuer des tests et des réglages supplémentaires :**
  - **Fonction de conseil STA (SQL Tuning Advisor)**
  - **Fonction de conseil SAA (SQL Access Advisor)**
  - **Fonction d'analyse des performances SQL**
  - **Fonction de surveillance SQL**
  - **Fonction de trace**

# **Chapitre 04**

## **Opérateurs de l'optimiseur**

# Objectifs

**A la fin de ce chapitre, vous pourrez :**

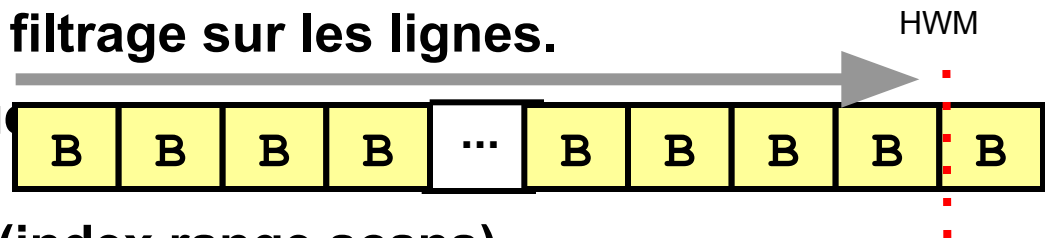
- **décrire les opérateurs SQL pour les tables et les index**
- **répertorier les chemins d'accès possibles**
- **Comprendre l'utilisation des indexes**

# Structures principales et chemins d'accès

Structures	Chemins d'accès
Tables	<ol style="list-style-type: none"><li>1. Balayage complet de table</li><li>2. Balayage par adresse de ligne</li><li>3. Balayage de table par échantillonnage</li></ol>
Index	<ol style="list-style-type: none"><li>4. Balayage unique d'index</li><li>5. Balayage d'intervalle d'index</li><li>6. Balayage complet d'index</li><li>7. Balayage complet et rapide d'index</li><li>8. Balayage par saut d'index</li><li>9. Balayage d'index (jointure d'index)</li><li>10. Utilisation d'index bitmap</li><li>11. Combinaison d'index bitmap</li></ol>

# Balayage complet de table

- Il effectue des lectures multiblocs (ici `DB_FILE_MULTIBLOCK_READ_COUNT = 4`).
- Il lit tous les blocs formatés sous le repère high-water mark.
- Il peut appliquer un filtrage sur les lignes.
- Il est plus rapide que les balayages d'intervalle d'index (index range scans) pour les gros volumes de données.



0.38699999 seconds

scott

`select * from emp where ename='King';`

Explain Plan x

0.387 seconds

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	EMP	FULL
Filter Predicates		
ENAME='King'		

# Balayages complets de table : Cas d'utilisation

- **Absence d'index adapté** (comme l'utilisation de fonction dans le where)
- **Filtres à faible sélectivité (ou absence de filtre)**  
Si l'optimiseur pense que l'interrogation accède à un nombre suffisant de blocs de la table, il peut utiliser un balayage complet de table même si des index sont disponibles.
- **Table de petite taille:** Si le nombre de blocs situés sous le repère high-water mark est inférieur à la valeur de `DB_FILE_MULTIBLOCK_READ_COUNT`
- **Degré élevé de parallélisme:** Lorsqu'une table présente un degré de parallélisme élevé, l'optimiseur privilégie le balayage complet de table plutôt que le balayage par intervalle. Pour connaître le degré de parallélisme d'une table, consultez la colonne `DEGREE` dans `ALL_TABLES`.
- **Imposé par le "Hint" de balayage complet de table dans la requête :** `Select /*+ FULL (Nom-Table) */`

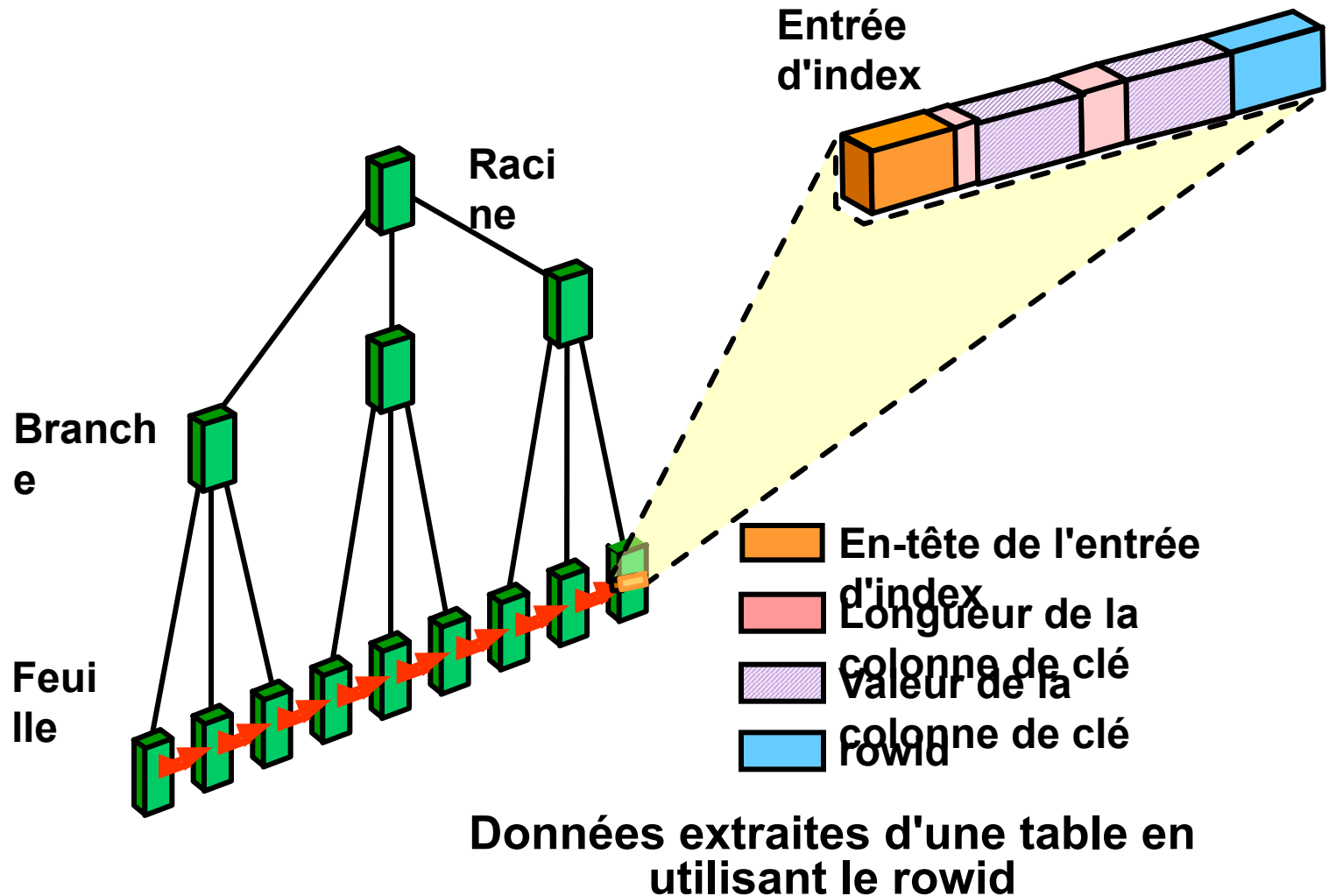
# Index : Présentation

## Techniques de stockage

- **des index :**
  - **Index B\*-tree : technique par défaut, la plus courante**
    - Index normal**
    - Index basé sur une fonction (valeur précalculée d'une fonction ou d'une expression)**
    - Table organisée en index**
  - **Index bitmap**
  - **Index de cluster (défini spécifiquement pour un cluster)**
- **Attributs d'index :**
  - **Clé compressée**
  - **Clé inversée**
  - **Ordre croissant, décroissant**
- **Index de domaine : propre à une application ou une cartouche**



# Index B\*-tree normaux

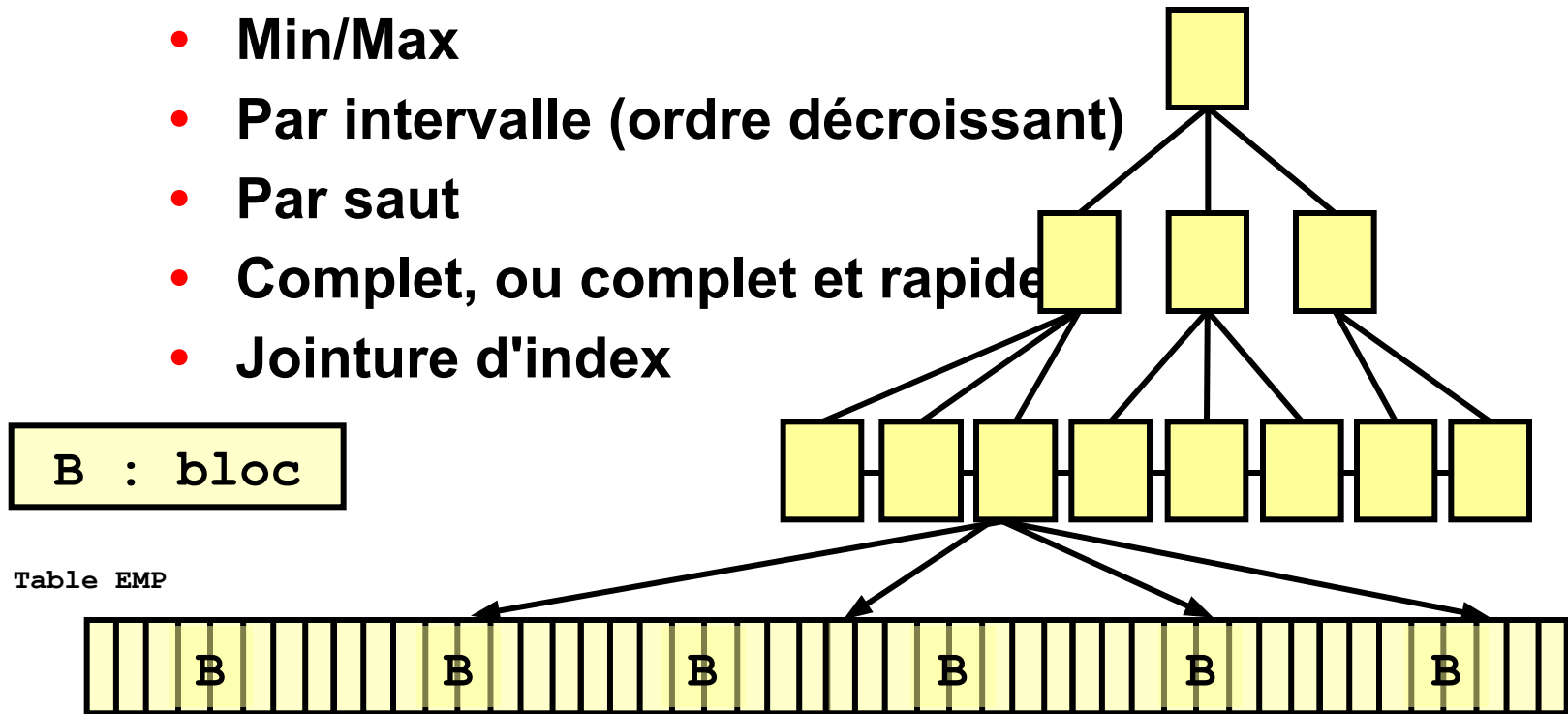


# Balayages d'index

## Types de balayage d'index :

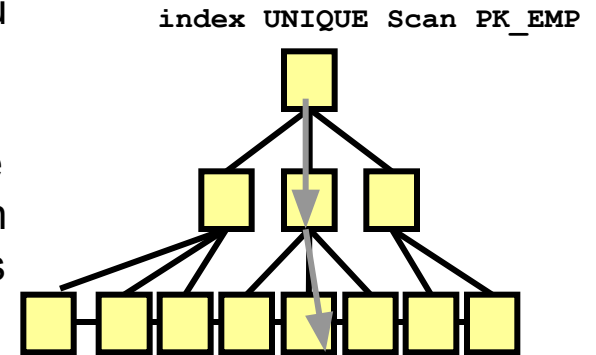
- Unique
- Min/Max
- Par intervalle (ordre décroissant)
- Par saut
- Complet, ou complet et rapide
- Jointure d'index

Index B-tree IX\_EMP



# Balayage unique d'index

Un balayage unique d'index (index unique scan) renvoie au plus un ROWID. Le système effectue un balayage unique lorsqu'une instruction contient une contrainte `UNIQUE` ou `PRIMARY KEY` qui restreint l'accès à une seule ligne. Ce chemin d'accès est utilisé lorsque toutes les colonnes d'un index (B-Tree) unique sont définies avec des conditions d'égalité.



```
create unique index PK_EMP on EMP(empno)
```

```
select * from emp where empno = 9999;
```

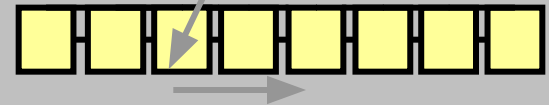
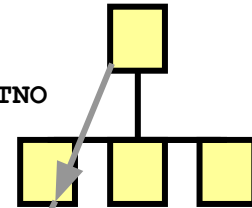
Explain Plan x

0.024 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			1	1
TABLE ACCESS	EMP	BY INDEX ROWID	1	1
INDEX	PK_EMP	UNIQUE SCAN	0	1
Access Predicat		EMPNO=7839		

# Balayage d'intervalle d'index

Index Range SCAN I\_DEPTNO



```
create index I_DEPTNO on EMP(deptno);  
  
select /*+ INDEX(EMP I_DEPTNO) */ *  
from emp where deptno = 10 and sal > 1000;
```

Script Output x Explain Plan x

0.012 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			2	4
TABLE ACCESS	EMP	BY INDEX ROWID	2	4
Filter Predicates SAL>1000				
INDEX	I_DEPTNO	RANGE SCAN	1	5
Access Predicat DEPTNO=10				

## Balayage d'intervalle d'index

- Un balayage d'intervalle d'index (index range scan) est une opération courante permettant d'accéder de manière sélective à des données.
- Il peut être borné (des deux côtés) ou non borné (d'un côté ou des deux). Les données sont renvoyées par ordre croissant en fonction des colonnes d'index. Les lignes contenant des valeurs identiques sont triées par ordre croissant selon ROWID.
- L'optimiseur utilise un balayage par intervalle lorsqu'il trouve une ou plusieurs colonnes de tête d'un index désignées dans les conditions (la clause WHERE), par exemple `col1 = :b1`, `col1 < :b1`, `col1 > :b1`, et toute combinaison des conditions précédentes.
- Les recherches à l'aide d'un caractère générique (`col1 like '%ASD'`) ne doivent pas être en tête, car elles n'entraînent pas un balayage par intervalle.
- Les balayages par intervalle peuvent utiliser des index uniques ou non uniques. Ils peuvent éviter le tri lorsque des colonnes d'index constituent la clause ORDER BY/GROUP BY et que les colonnes indexées sont déclarées NOT NULL, car elles ne sont sinon pas prises en compte.
- Un balayage d'intervalle d'index par ordre décroissant (index range scan descending) est identique à un balayage d'intervalle d'index, à ceci près que les données sont renvoyées dans l'ordre décroissant. L'optimiseur utilise un balayage d'intervalle d'index par ordre décroissant lorsqu'une clause de tri par ordre décroissant peut être satisfaite par un index.
- Dans l'exemple de la diapositive ci-dessus, le système utilise l'index I\_DEPTNO pour accéder aux lignes pour lesquelles EMP.DEPTNO=10. Il récupère le ROWID de ces lignes et extrait les autres colonnes de la table EMP. Pour finir, il applique le filtre EMP.SAL >1000 aux lignes extraites pour renvoyer le résultat final.

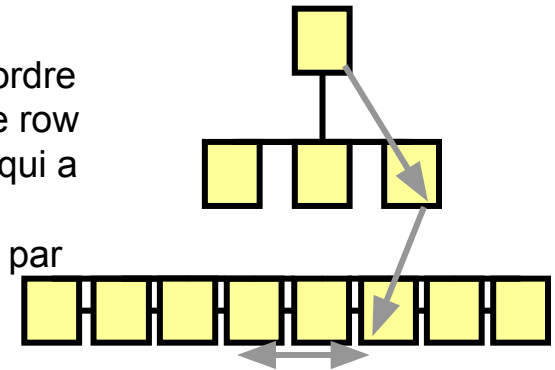
# Balayage d'intervalle d'index par ordre décroissant

En plus des balayages d'intervalle d'index (index range scan) par ordre croissant décrits dans la diapositive de la page précédente, le système peut effectuer des balayages dans l'ordre inverse comme illustré dans la diapositive ci-dessus.

Dans l'exemple considéré, des lignes sont extraites de la table `EMP` par ordre décroissant selon les valeurs de la colonne `DEPTNO`. Vous pouvez voir le row source de l'opération `DESCENDING` pour l'`ID` 2 dans le plan d'exécution qui a donné corps à ce type de balayage d'index.

**Remarque :** Par défaut, les balayages d'intervalle d'index sont exécutés par ordre croissant.

Index Range SCAN IDX

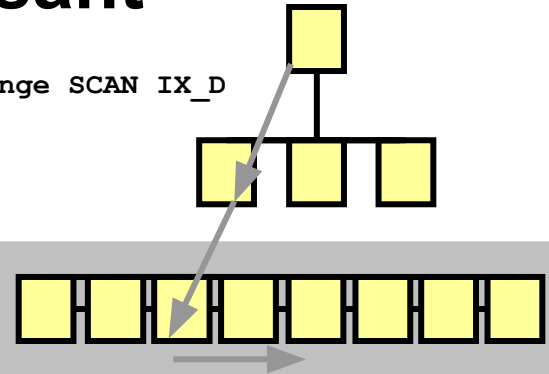


```
select * from emp where deptno>20 order by deptno desc;
```

Script Output x Explain Plan x				
0.426 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			2	7
TABLE ACCESS	EMP	BY INDEX ROWID	2	7
INDEX	I_DEPTNO	RANGE SCAN DESCENDING	1	7
Access Predicat				
DEPTNO>20				

# Balayage d'intervalle d'index par ordre décroissant

Index Range SCAN IX\_D

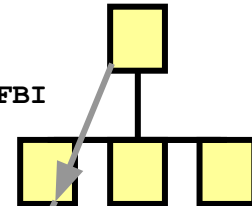


```
drop index I_Deptno;  
create index IX_D on EMP(deptno desc);  
select * from emp where deptno <30;
```

Script Output x Explain Plan x				
0.011 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			2	9
TABLE ACCESS	EMP	BY INDEX ROWID	2	9
INDEX	IX_D	RANGE SCAN	1	1
Access Predicates				
SYS_OP_DESCEND(DEPTNO)>HEXTORAW('3EE0FF')				
Filter Predicates				
SYS_OP_UNDESCEND(SYS_OP_DESCEND(DEPTNO))<30				

# Balayage d'intervalle d'index basé sur une fonction

Index Range SCAN IX\_FBI



```
create index IX_FBI on EMP(UPPER(ename));
```

```
select * from emp where upper(ENAME) like 'A%';
```



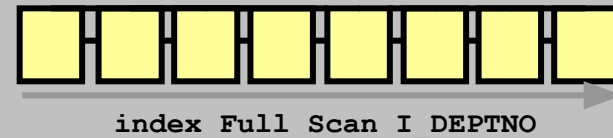
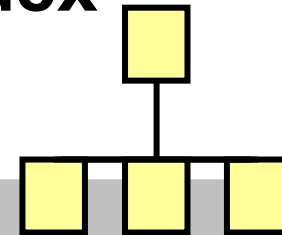
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			2	1
TABLE ACCESS	EMP	BY INDEX RO...	2	1
INDEX	IX_FBI	RANGE SCAN	1	1
Access Predicates				
UPPER(ENAME) LIKE 'A%'				
Filter Predicates				
UPPER(ENAME) LIKE 'A%'				



# Balayage complet d'index

```
create index I_DEPTNO on EMP(deptno);

select *
from emp
where sal > 1000 and deptno is not null
order by deptno;
```



Script Output x Explain Plan x				
0.012 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			2	13
TABLE ACCESS	EMP	BY INDEX RO...	2	13
Filter Predicates SAL>1000				
INDEX	I_DEPTNO	FULL SCAN	1	14
Filter Predicates DEPTNO IS NOT NULL				

# Balayage complet et rapide d'index

db\_file\_multiblock\_read\_count = 4

lecture multibloc

lecture multibloc

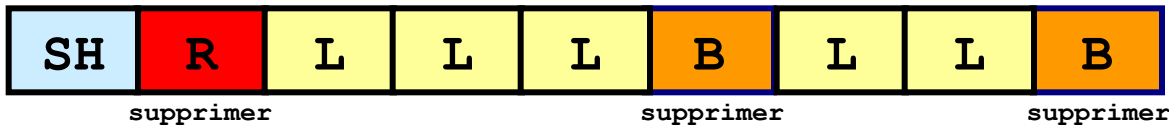
LEGENDE :

SH=en-tête de segment

R=bloc racine

B=bloc branche

L=bloc feuille

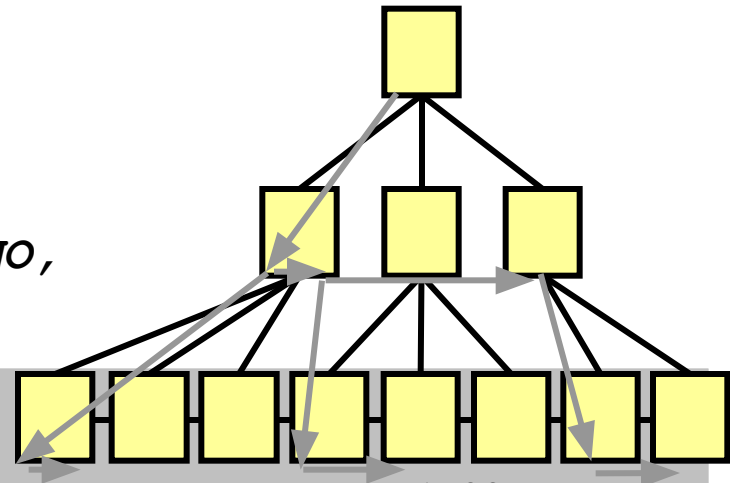


```
select /*+ INDEX_FFS(EMP I_DEPTNO) */ deptno from emp
where deptno is not null;
```

Script Output x Explain Plan x				
0.01 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			2	14
INDEX	I_DEPTNO	FAST FULL SCAN	2	14
Filter Predicates				
DEPTNO IS NOT NULL				

# Balayage par saut d'index : Exemple

*Index sur (DEPTNO,  
SAL)*



```
create index IX_SS on EMP(DEPTNO,SAL);
```

```
select /*+ index_ss(EMP IX_SS) */ * from emp where SAL < 1500;
```

Script Output x Explain Plan x				
0.01 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			4	2
TABLE ACCESS	EMP	BY INDEX ROWID	4	2
INDEX	IX_SS	SKIP SCAN	3	2
Access Predicates				
SAL<1500				
Filter Predicates				
SAL<1500				

## Balayage par saut d'index : Exemple

L'exemple de la diapositive ci-dessus recherche les employés qui ont un salaire inférieur à 1 500 en utilisant un balayage par saut d'index.

Il est supposé qu'il existe un index concaténé sur les colonnes `DEPTNO` et `SAL`.

Vous pouvez voir que l'interrogation ne comporte pas de prédicat sur la colonne de tête `DEPTNO`. Cette colonne de tête ne contient que des valeurs discrètes, à savoir 10, 20 et 30.

Le balayage par saut permet de diviser un index composé de manière logique en sous-index de taille plus petite. Le nombre de sous-index logiques est déterminé par le nombre de valeurs distinctes dans la colonne initiale.

Le système fait comme si l'index était en réalité composé de trois petites structures d'index masquées dans la grande. Dans cet exemple, il existe trois structures d'index :

```
where deptno = 10
```

```
where deptno = 20
```

```
where deptno = 30
```

Les résultats sont classés selon `DEPTNO`.

**Remarque :** Le balayage par saut est avantageux lorsqu'il existe peu de valeurs distinctes dans la colonne de tête de l'index composé et de nombreuses valeurs distinctes dans la clé secondaire.

# Balayage de jointure d'index

```
alter table emp modify (SAL not null, ENAME not null);  
create index I_ENAME on EMP(ename);  
create index I_SAL on EMP(sal);
```

select /\*+ INDEX\_JOIN(e) \*/ ename, sal from emp e;

Statement Output x Autotrace x

1.563 seconds

OPERATION	OBJECT_NAME	COST
SELECT STATEMENT		3
VIEW	index\$_join\$_001	3
type="db_version"		
11.2.0.1		
HASH JOIN		
Access Predicates		
ROWID=ROWID		
INDEX FAST FULL SCAN	I_ENAME	1
INDEX FAST FULL SCAN	I_SAL	1

# Index B\*-tree et valeurs NULL

Les index B\*-tree ne stockent pas les valeurs NULL

```
create table nulltest ( col1 number, col2 number not null);  
create index nullind1 on nulltest (col1);  
create index notnullind2 on nulltest (col2);
```

<pre>select /*+ index(t nullind1) */ col1 from nulltest t;</pre>				
Script Output x Explain Plan x				
0.864 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			2	1
TABLE ACCESS	NULLTEST	FULL	2	1

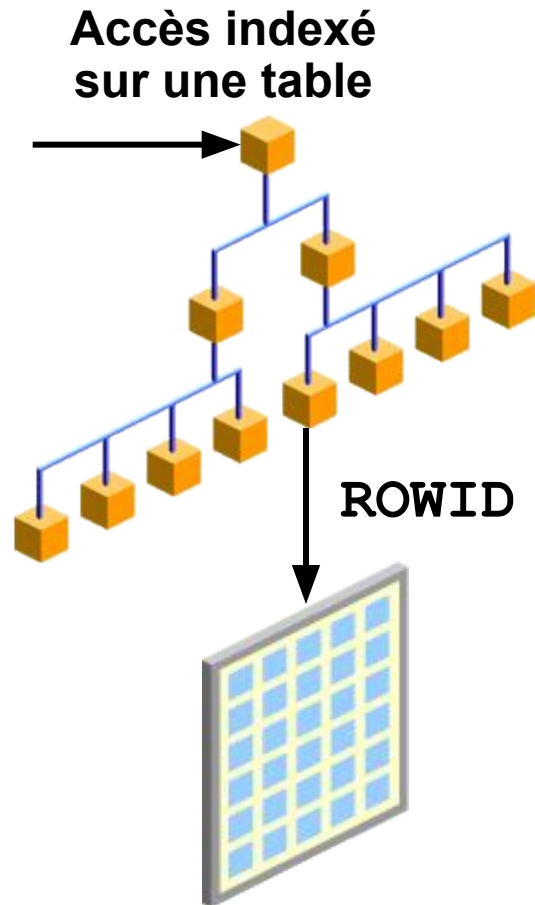
  

<pre>select col1 from nulltest t where col1=10;</pre>				
Script Output x Explain Plan x				
1.105 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			1	1
INDEX	NULLIND1	RANGE SCAN	1	1
Access Predicates COL1=10				

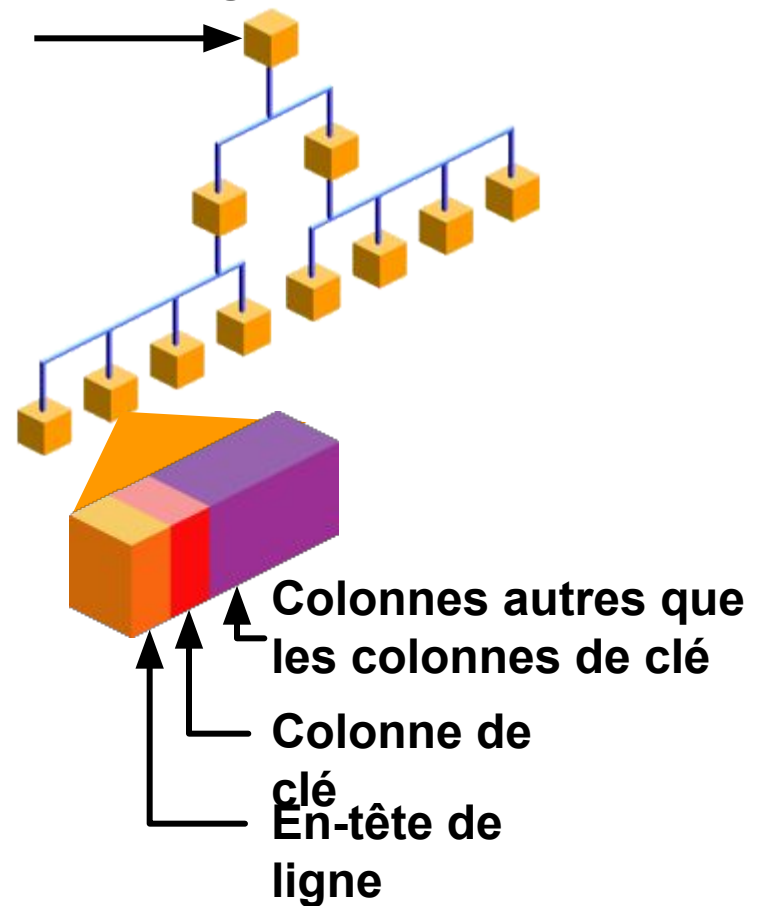
  

<pre>select /*+ index(t notnullind2) */ col2 from nulltest t;</pre>				
Script Output x Explain Plan x				
0.034 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			1	1
INDEX	NOTNULLIND2	FULL SCAN	1	1

# Tables organisées en index



**Accéder à la table organisée en index**



# Balayage des tables organisées en index

```
create table iotemp
  ( empno number(4) primary key, ename varchar2(10) not null,
    job varchar2(9), mgr number(4), hiredate date,
    sal number(7,2) not null, comm number(7,2), deptno
    number(2))
  organization index;
```

```
select * from iotemp where empno=9999;
```

Script Output x Explain Plan x

0.734 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			1	1
INDEX	SYS_IOT_TOP_7...	UNIQUE SCAN	1	1

Access Predicates  
EMPNO=9999

```
select * from iotemp where sal>1000;
```

Script Output x Explain Plan x

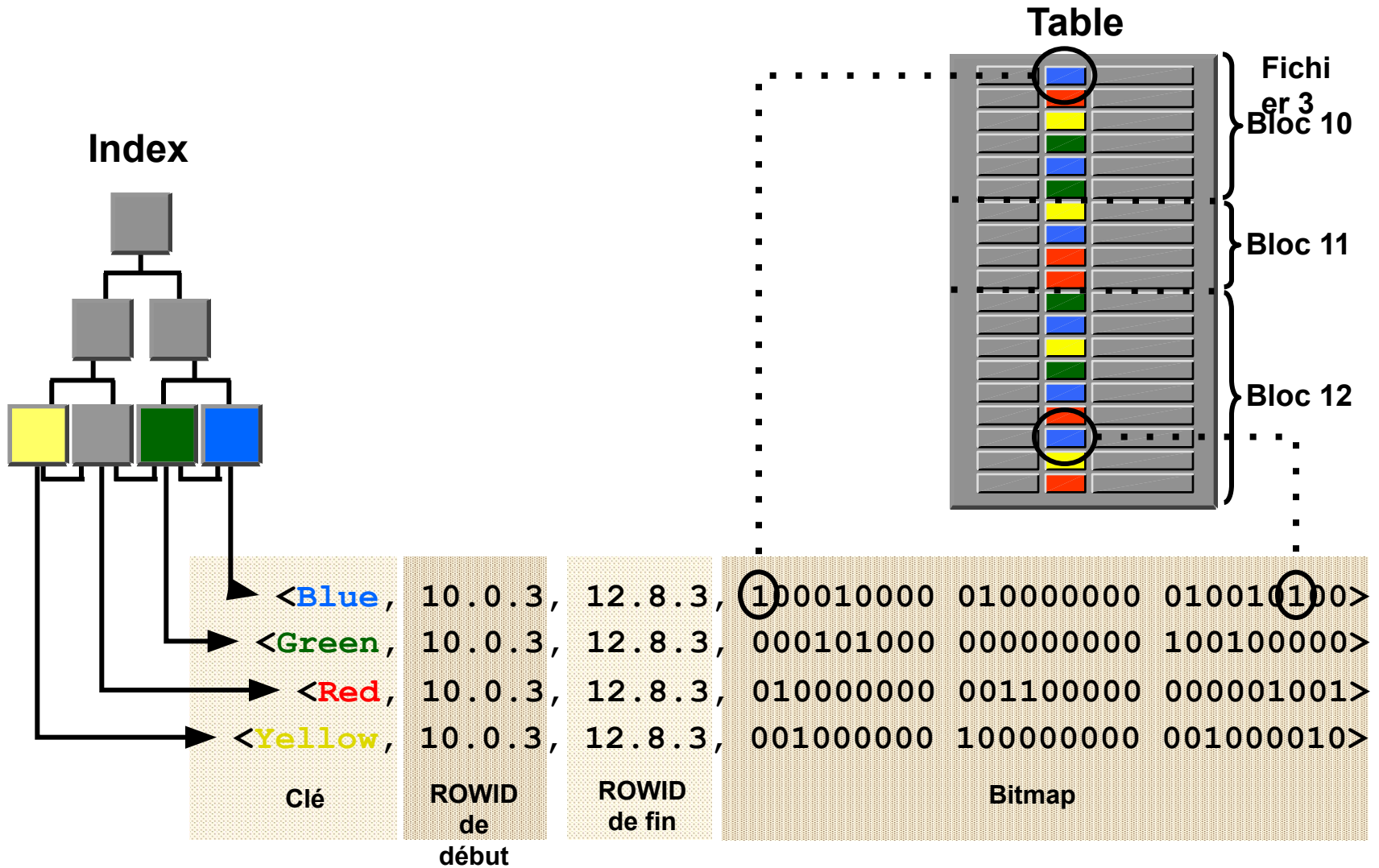
0.007 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			2	1
INDEX	SYS_IOT_TOP_7...	FAST FULL SCAN	2	1

Filter Predicates  
SAL>1000



# Index bitmap



# Accès aux index bitmap : Exemples

```
SELECT * FROM PERF_TEAM WHERE country='FR';
```

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	45
1	TABLE ACCESS BY INDEX ROWID	PERF_TEAM	1	45
2	<b>BITMAP CONVERSION TO ROWIDS</b>			
3	<b>BITMAP INDEX SINGLE VALUE</b>	IX_B2		

```
Predicate: 3 - access("COUNTRY"='FR')
```

```
SELECT * FROM PERF_TEAM WHERE country>'FR';
```

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	45
1	TABLE ACCESS BY INDEX ROWID	PERF_TEAM	1	45
2	<b>BITMAP CONVERSION TO ROWIDS</b>			
3	<b>BITMAP INDEX RANGE SCAN</b>	IX_B2		

```
Predicate: 3 - access("COUNTRY">'FR') filter("COUNTRY">'FR')
```

# Combinaison de chemins d'accès par index bitmap

```
SELECT * FROM PERF_TEAM WHERE country in ('FR','DE');
```

Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	45
1	INLIST ITERATOR			
2	TABLE ACCESS BY INDEX ROWID	PERF_TEAM	1	45
3	<b>BITMAP CONVERSION TO ROWIDS</b>			
4	<b>BITMAP INDEX SINGLE VALUE</b>	IX_B2		

Predicate: 4 - access("COUNTRY"='DE' OR "COUNTRY"='FR')

```
SELECT * FROM PERF_TEAM WHERE country='FR' and gender='M';
```

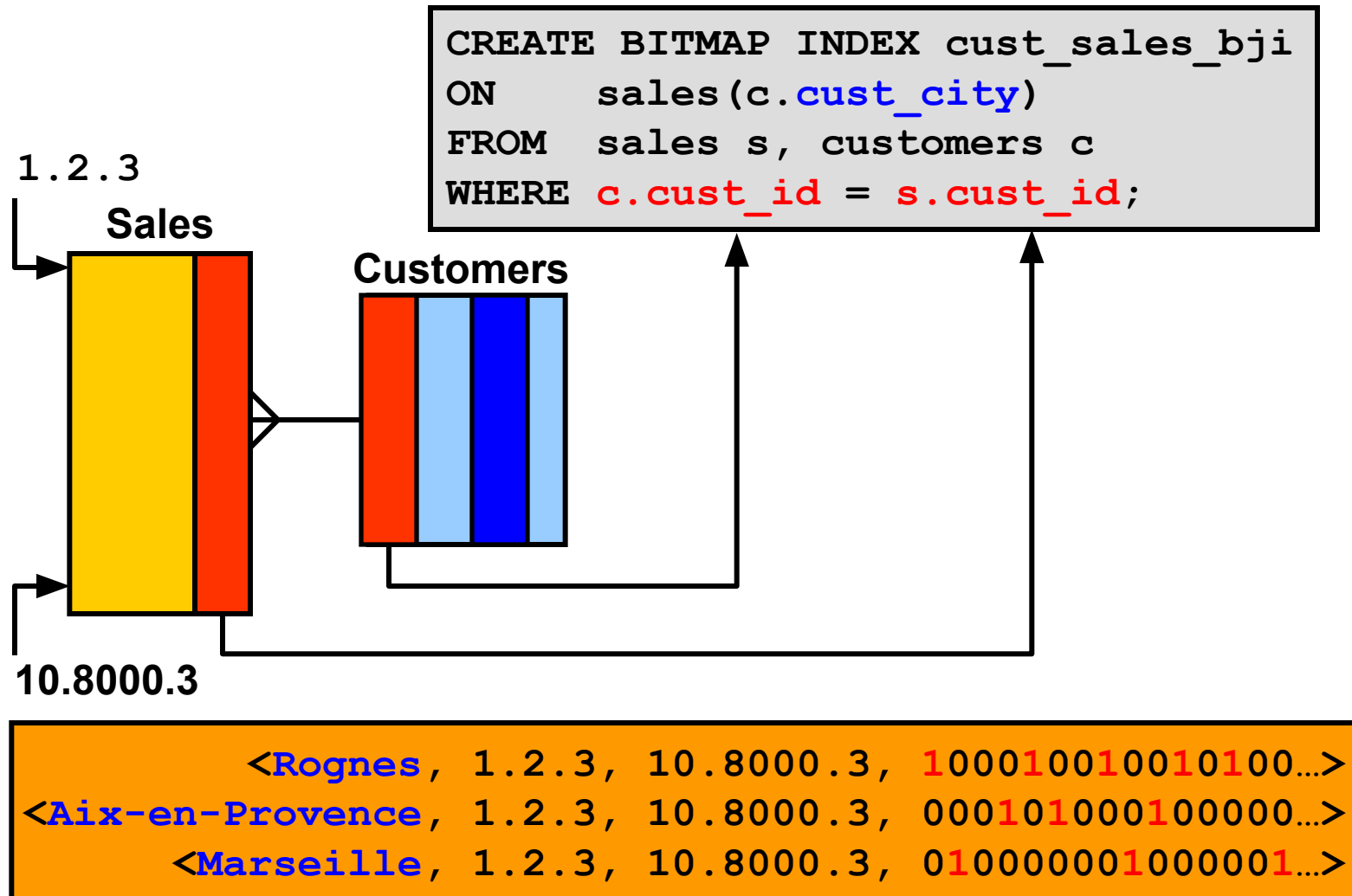
Id	Operation	Name	Rows	Bytes
0	SELECT STATEMENT		1	45
1	TABLE ACCESS BY INDEX ROWID	PERF_TEAM	1	45
2	<b>BITMAP CONVERSION TO ROWIDS</b>			
3	<b>BITMAP AND</b>			
4	<b>BITMAP INDEX SINGLE VALUE</b>	IX_B1		
5	<b>BITMAP INDEX SINGLE VALUE</b>	IX_B2		

Predicate: 4 - access("GENDER"='M') 5 - access("COUNTRY"='FR')

# Opérations sur les index bitmap

- **BITMAP CONVERSION :**
  - TO ROWIDS
  - FROM ROWIDS
  - COUNT
- **BITMAP INDEX :**
  - SINGLE VALUE
  - RANGE SCAN
  - FULL SCAN
- **BITMAP MERGE**
- **BITMAP AND/OR**
- **BITMAP MINUS**
- **BITMAP KEY ITERATION**

# Index de jointure bitmap (1/2)



## Index de jointure bitmap (2/2)

Outre un index bitmap sur une table unique, vous pouvez créer un index de jointure bitmap. Il s'agit d'un index bitmap qui sert à la jointure de deux tables ou plus. Un tel index constitue une méthode peu consommatrice d'espace qui permet de réduire le volume des données à joindre grâce à l'exécution préalable de la jointure. **Remarque** : En termes de stockage, les index de jointure bitmap sont beaucoup plus efficaces que les vues matérialisées de jointure. Reportez-vous au chapitre intitulé "Etude de cas : Transformation en étoile" pour un exemple de row source.

Dans l'exemple de la diapositive ci-dessus, vous créez un index de jointure bitmap nommé `cust_sales_bji` sur la table `SALES`. La clé de cet index est la colonne `CUST_CITY` de la table `CUSTOMERS`. Il est supposé qu'une contrainte de clé primaire est appliquée à la table `CUSTOMERS`, afin de garantir que les éléments stockés dans le bitmap correspondent aux données présentes dans les tables. La colonne `CUST_ID` est la clé primaire de `CUSTOMERS`, mais elle est également une clé étrangère de `SALES`, même si elle n'est pas obligatoire.

Les clauses `FROM` et `WHERE` de l'instruction `CREATE` permettent au système d'assurer le lien entre les deux tables. Elles représentent la condition de jointure entre ces tables. La partie médiane de la diapositive présente l'implémentation théorique de l'index de jointure bitmap. Chaque entrée ou clé de l'index représente une ville qu'il est possible de trouver dans la table `CUSTOMERS`. Un bitmap est ensuite associé à chaque clé. Chaque bit du bitmap correspond à une ligne de la table `SALES`. Dans la première clé de la diapositive (Rognes), vous pouvez voir que la première ligne de la table `SALES` correspond à un produit vendu à un client de Rognes. Le stockage du résultat d'une jointure permet d'éviter complètement la jointure pour les instructions SQL qui utilisent un index de jointure bitmap.

# Index composés

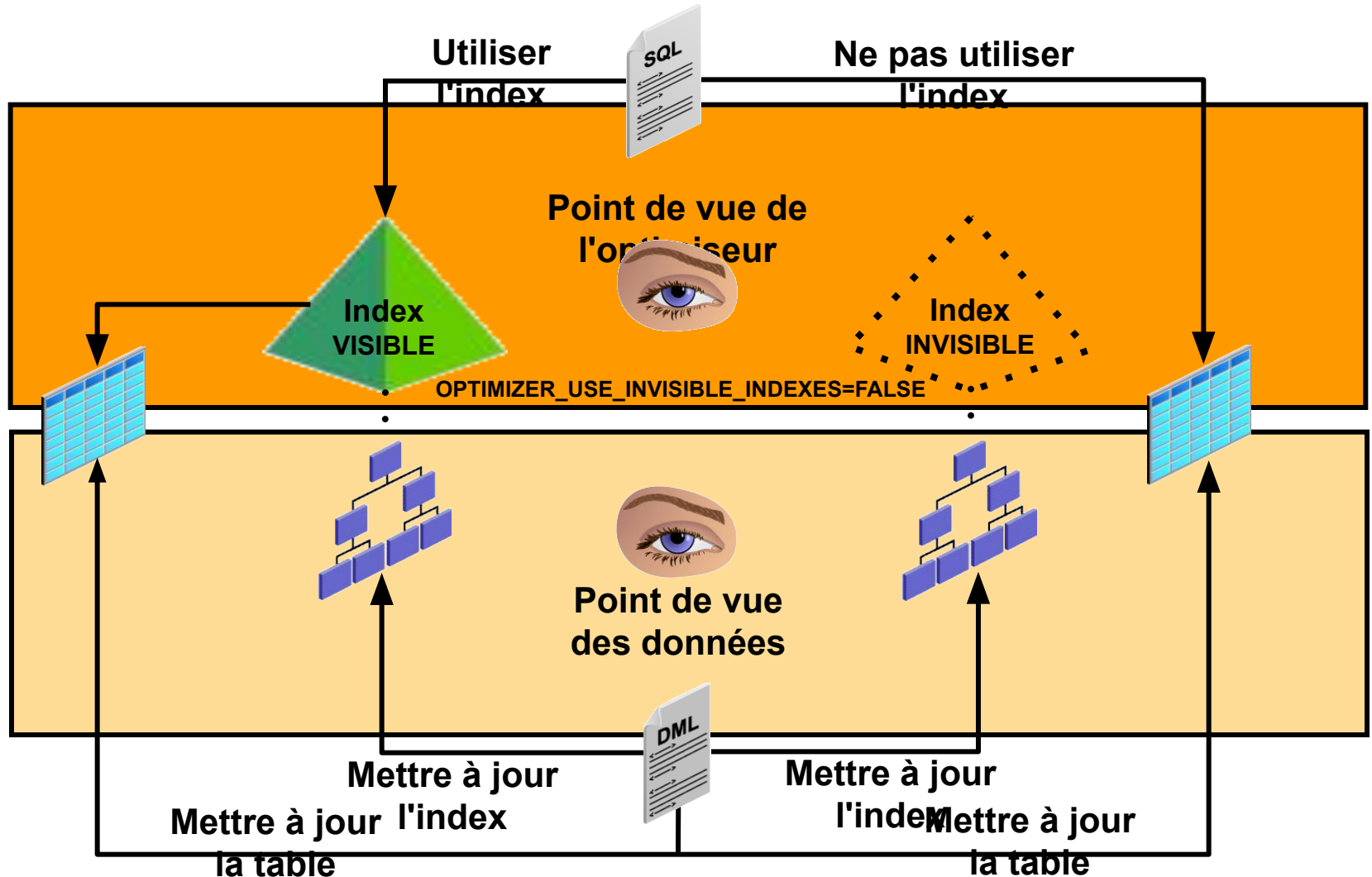


```
create index cars_make_model_idx on cars(make, model);
```

```
select *  
from cars  
where make = 'CITROËN' and model = '2CV';
```

Id	Operation	Name
0	SELECT STATEMENT	
1	TABLE ACCESS BY INDEX ROWID	CUSTOMERS
* 2	INDEX RANGE SCAN	CARS_MAKE_MODEL_IDX

# Index invisible : Présentation





# Index invisibles : Exemples

- L'index est modifié pour ne plus être visible par l'optimiseur :

```
ALTER INDEX ind1 INVISIBLE;
```

- L'optimiseur ne prend pas en compte cet index :

```
SELECT /*+ index(TAB1 IND1) */ COL1 FROM TAB1 WHERE ...;
```

- L'optimiseur prend en compte cet index :

```
ALTER INDEX ind1 VISIBLE;
```

- Vous pouvez créer un index défini comme invisible :

```
CREATE INDEX IND1 ON TAB1(COL1) INVISIBLE;
```

# Règles relatives à la gestion des index

- **Créer les index après l'insertion des données dans la table.**
- **Indexer les tables et les colonnes appropriées.**
- **Ordonner les colonnes d'index pour améliorer les performances.**
- **Limiter le nombre d'index de chaque table.**
- **Supprimer les index qui ne sont plus nécessaires.**
- **Indiquer le tablespace de chaque index.**
- **Envisager la création d'index en parallèle.**
- **Envisager la création d'index avec NOLOGGING.**
- **Prendre en compte les coûts et les avantages de la fusion ou de la reconstruction des index.**
- **Etudier le coût avant de désactiver ou de supprimer des contraintes.**

# Etudier les cas de non-utilisation des index

Il existe de nombreuses raisons qui font qu'un index n'est pas utilisé :

- Des fonctions sont appliquées sur la colonne indexée!  
ex: `Select ... where upper(emane)='AHMED'`
- Il existe une disparité de type de données:  
L'index n'est pas utilisé en cas de disparité de type de données entre la colonne indexée et la valeur comparée, à cause de la conversion implicite de type. Par exemple, si la colonne `SSN` est de type `VARCHAR2`, l'instruction suivante n'utilise pas l'index sur `SSN` :  

```
SELECT * FROM person WHERE SSN = 123456789
```
- Les statistiques sont anciennes.
- La colonne accepte les valeurs `NULL`.
- Les opérations seraient plus lentes en utilisant l'index qu'en ne l'utilisant pas.

# **Chapitre 05**

## **Optimiseur : Opérateurs de jointure**

# Objectifs

**A la fin de ce chapitre, vous pourrez :**

- **décrire les opérateurs SQL pour les jointures**
- **répertorier les chemins d'accès possibles**

# Méthodes de jointure

Un row source est un ensemble des données accessibles dans une interrogation

Une jointure :

- Définit les relations entre deux <<row sources>>
- est une méthode permettant de combiner les données de deux sources et de les faire correspondre entre eux
- est contrôlée par des prédicats de jointure qui définissent les relations entre les objets
- Méthodes de jointure :
  - Boucles imbriquées
  - Jointure de type tri-fusion (sort-merge)

```
SELECT e.ename, d.dname  
FROM dept d JOIN emp e USING (deptno)  
WHERE e.job = 'ANALYST' OR e.empno = 9999;
```

◀ Prédicat de jointure

◀ Prédicat de non-jointure

```
SELECT e.ename, d.dname  
FROM emp e, dept d  
WHERE e.deptno = d.deptno AND  
      (e.job = 'ANALYST' OR e.empno = 9999);
```

◀ Prédicat de jointure

◀ Prédicat de non-jointure

# Jointure en boucle imbriquée (Nested Loop join)

- Une des deux tables est définie en tant que table externe ou table directrice, l'autre est appelée table interne ou table de droite.
- Pour chaque ligne de la table externe (directrice) qui correspond aux prédicats de la table simple, toutes les lignes de la table interne qui satisfont au prédicat de jointure sont extradites

Le code permettant l'émulation d'une jointure en boucle imbriquée peut s'écrire de la manière suivante :

```
for r1 in (select rows from EMP that match single table predicate) loop
    for r2 in (select rows from DEPT that match current row from EMP) loop
        output values from current row of EMP and current row of DEPT
```

```
select ename, e.deptno, d.deptno, d.dname
from emp e, dept d
where e.deptno = d.deptno and ename like 'A%';
```

-----					
Id		Operation	Name	Rows	Cost
-----					
0		SELECT STATEMENT		2	4
1		NESTED LOOPS		2	4
2		TABLE ACCESS FULL	EMP	2	2
3		TABLE ACCESS BY INDEX ROWID	DEPT	1	1
4		INDEX UNIQUE SCAN	PK_DEPT	1	
-----					

```
2 - filter("E"."ENAME" LIKE 'A%')
```

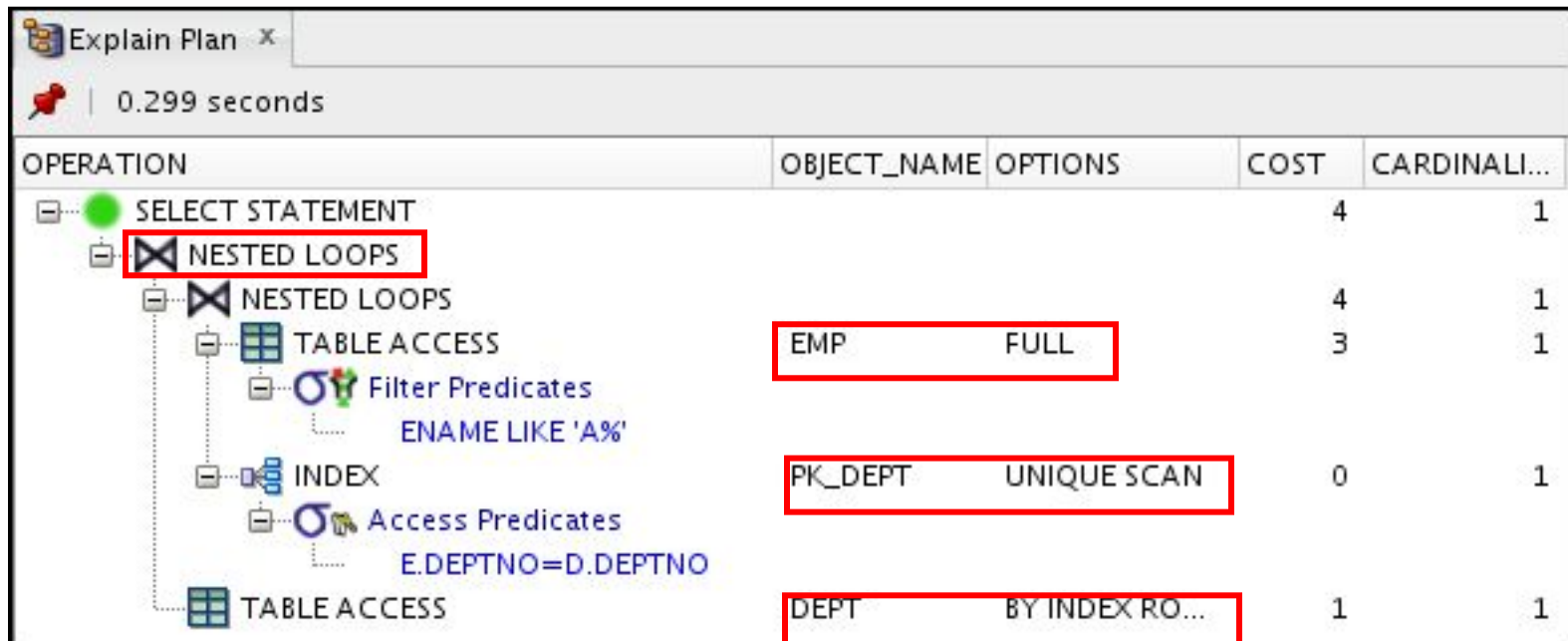
```
4 - access("E"."DEPTNO"="D"."DEPTNO")
```

# Jointure en boucle imbriquée

## Nested Loop avec Index

```
select ename, e.deptno, d.deptno, d.dname
from emp e, dept d
where e.deptno = d.deptno and ename like 'A%';
```

le système commence par effectuer une jointure NESTED LOOPS entre l'autre table et l'index.  
Au lieu d'accéder à la table pour chaque ROWID renvoyé par la première jointure NESTED LOOPS, le système crée des lots de ROWID et exécute une seconde jointure NESTED LOOPS entre les ROWID et la table



The image shows an Oracle Explain Plan for the provided SQL query. The plan is visualized with icons and text, and also includes a table with columns: OPERATION, OBJECT\_NAME, OPTIONS, COST, and CARDINALITY. The plan shows a NESTED LOOPS join between the EMP table and the DEPT table. The EMP table is accessed via a FULL scan, and the DEPT table is accessed via a UNIQUE SCAN on the PK\_DEPT index. The join condition is E.DEPTNO=D.DEPTNO. The filter predicate is ENAME LIKE 'A%'. The total cost is 4, and the cardinality is 1.

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			4	1
NESTED LOOPS			4	1
TABLE ACCESS	EMP	FULL	3	1
Filter Predicates ENAME LIKE 'A%'				
INDEX	PK_DEPT	UNIQUE SCAN	0	1
Access Predicates E.DEPTNO=D.DEPTNO				
TABLE ACCESS	DEPT	BY INDEX ROWID	1	1



# Jointure de type tri-fusion (sort-merge)

```
select /*+ USE_MERGE(d e) NO_INDEX(d) */  
  ename, e.deptno, d.deptno, dname  
from emp e, dept d  
where e.deptno = d.deptno and ename > 'A'
```

Le concept de table directrice n'est pas utilisé dans une jointure de type sort-merge:

1. Le premier jeu de données est extrait, en utilisant les prédicats d'accès et de filtrage, et les données sont triées selon les colonnes de jointure.
2. Le second jeu de données est extrait, en utilisant les prédicats d'accès et de filtrage, et les données sont triées selon les colonnes de jointure.
3. Pour chaque ligne du premier jeu de données, le système recherche le point de départ dans le second jeu et effectue un balayage jusqu'à une ligne qui ne peut être jointe.

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALI...
SELECT STATEMENT			8	14
MERGE JOIN			8	14
SORT		JOIN	4	4
TABLE ACCESS	DEPT	FULL	3	4
SORT		JOIN	4	14
Access Predicates				
AND				
E.DEPTNO=D.DEPTNO				
SYS_OP_DESCEND(E.DEPT				
Filter Predicates				
E.DEPTNO=D.DEPTNO				
TABLE ACCESS	EMP	FULL	3	14
Filter Predicates				
ENAME>'A'				

# Jointure de hachage

## Pour effectuer une jointure de hachage (hash join) entre deux row sources:

Le système lit le premier jeu de données et crée un tableau de "hash buckets" en mémoire.

Un "hash bucket" est un emplacement qui sert de point de départ pour une liste liée de lignes de la table créée.

Une ligne appartient à un "hash bucket" si le numéro de celui-ci correspond au résultat obtenu par le système en exécutant la fonction de hachage interne sur la ou les colonnes de jointure de la ligne.

Le système commence à lire le second jeu de lignes en utilisant le mécanisme d'accès le mieux adapté pour récupérer les lignes, et il utilise la même fonction de hachage sur la ou les colonnes de jointure pour calculer le numéro du "hash bucket" pertinent.

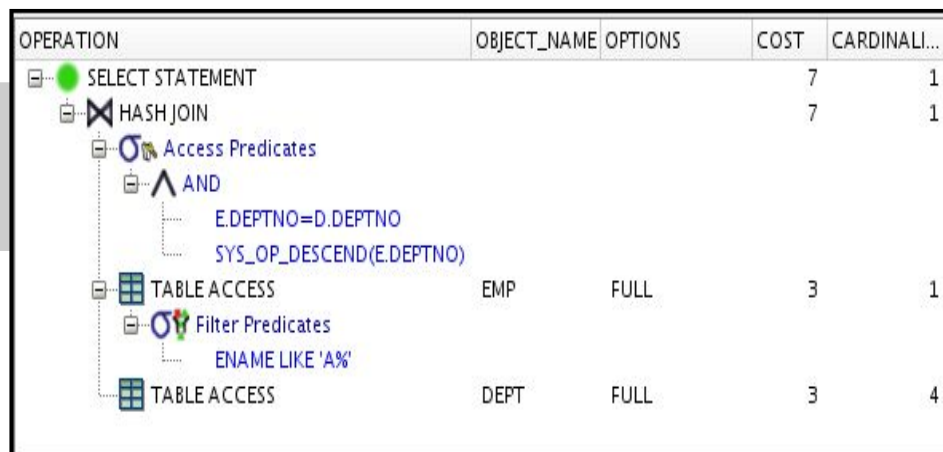
Il vérifie ensuite s'il existe des lignes dans ce "hash bucket". Cette opération est appelée test de la table de hachage.

S'il n'existe pas de lignes dans le "hash bucket" considéré, le système peut immédiatement supprimer la ligne de la table de test.

Dans le cas contraire, le système effectue une vérification de correspondance exacte sur la ou les colonnes de jointure. S'il trouve des lignes présentant une correspondance exacte, il peut les extraire immédiatement (ou les transmettre à l'étape suivante du plan d'exécution). Ainsi, lorsque vous effectuez une jointure de hachage, vous devez extraire toutes les lignes du plus petit row source avant de renvoyer la première ligne à l'opération suivante.

**Remarque : Les jointures de hachage ne sont appliquées qu'aux équijointures. Elles sont surtout utiles pour joindre d'importants volumes de données.**

```
select /*+ USE_HASH(e d) */
  ename, e.deptno, d.deptno, dname
from emp e, dept d
where e.deptno = d.deptno and ename like 'A%'
```

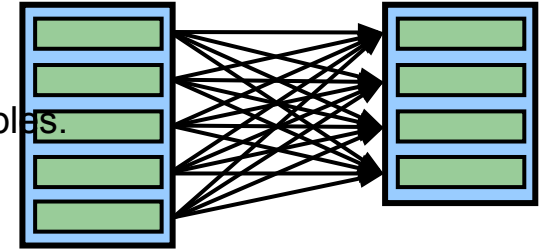


# Jointure cartésienne

Les jointures cartésiennes sont utilisées lorsqu'une ou plusieurs des tables concernées n'ont pas de conditions de jointure avec d'autres tables de l'instruction.

L'optimiseur joint toutes les lignes d'une table à toutes les lignes de l'autre table

**Remarque :** Les jointures cartésiennes ne sont généralement pas souhaitables.



```
select ename, e.deptno, d.deptno, dname
from emp e, dept d where ename like 'A%';
```

Explain Plan x				
0.295 seconds				
OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALI...
SELECT STATEMENT			6	4
MERGE JOIN		CARTESIAN	6	4
TABLE ACCESS	EMP	FULL	3	1
Filter Predicates				
ENAME LIKE 'A%'				
BUFFER		SORT	3	4
TABLE ACCESS	DEPT	FULL	3	4

# Types de jointure

- Une opération de jointure combine la sortie de deux row sources et renvoie un seul row source.
- Les types d'opération de jointure sont les suivants :
  - Jointure (équijointure/naturelle, non-équijointure)
  - Jointure externe (complète, gauche et droite)
  - Semijointure : sous-interrogation EXISTS
  - Antijointure : sous-interrogation NOT IN
  - Interrogation en étoile (optimisation)

# Equijointures et non-équijointures

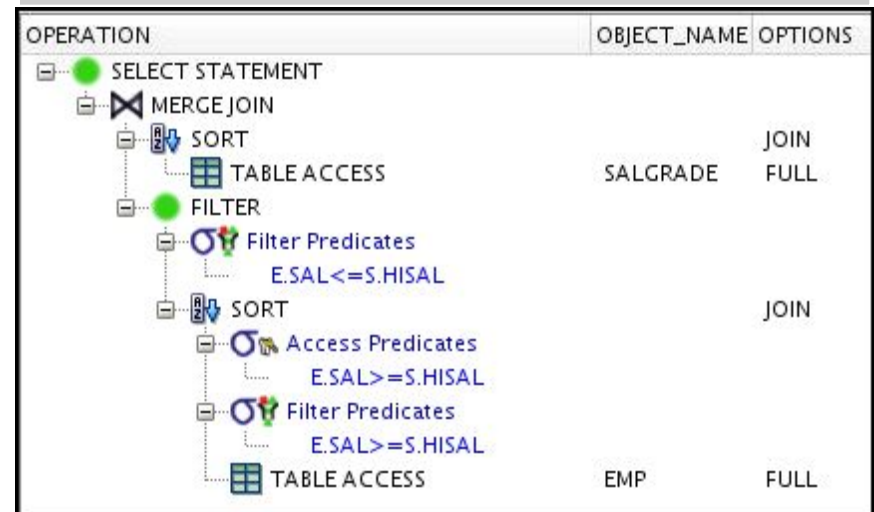
```
SELECT e.ename, e.sal, s.grade
FROM   emp e ,salgrade s
WHERE  e.sal = s.hisal;
```

**Equijointure:**  
utilise opérateur d'égalité



Non-équijointure

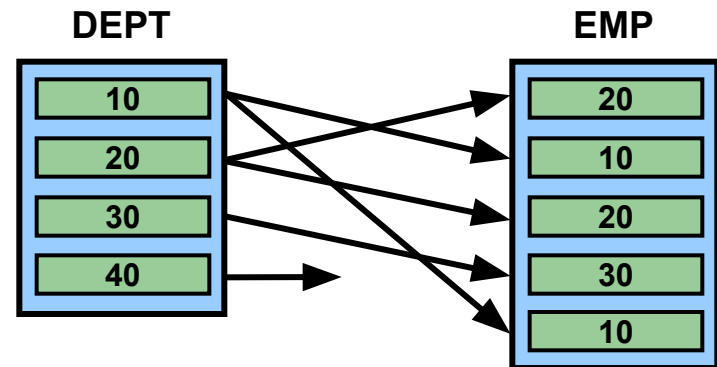
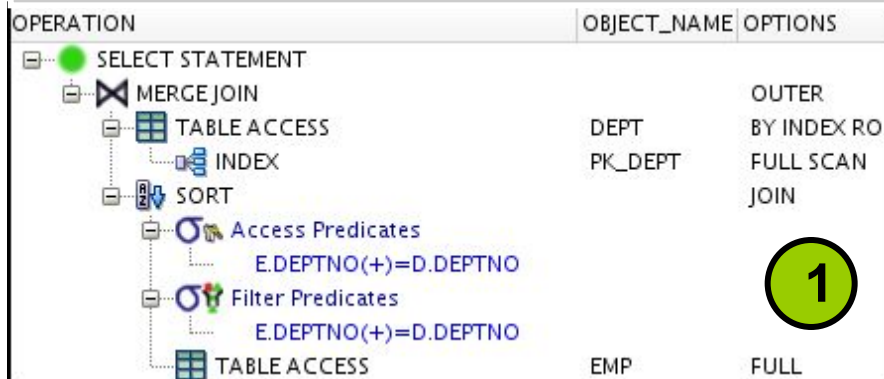
```
SELECT e.ename, e.sal, s.grade
FROM   emp e ,salgrade s
WHERE  e.sal between s.hisal and s.hisal;
```



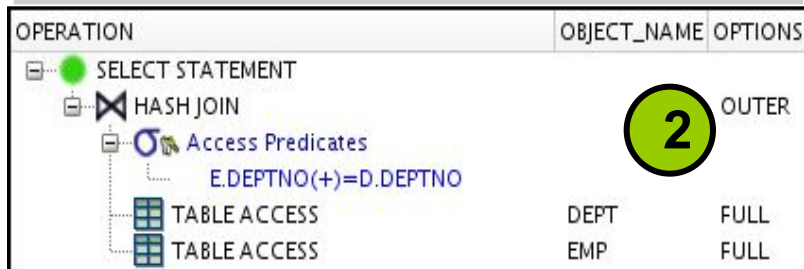
# Jointures externes

Une jointure externe renvoie une ligne même si aucune correspondance n'est trouvée.

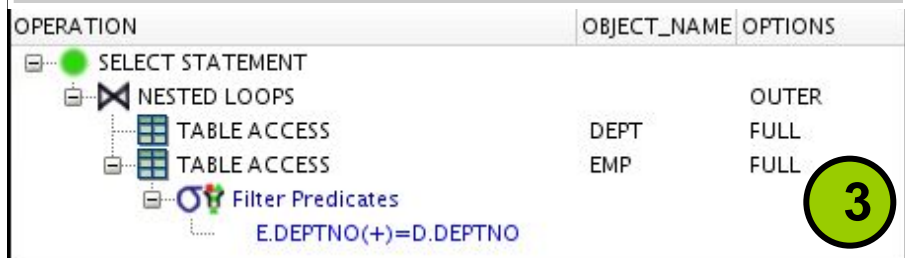
```
SELECT d.deptno,d.dname,e.empno,e.ename
FROM emp e, dept d
WHERE e.deptno(+) = d.deptno;
```



```
SELECT /*+ USE_HASH(e d) */
d.deptno,d.dname,e.empno,e.ename
FROM emp e, dept d
WHERE e.deptno(+) = d.deptno;
```

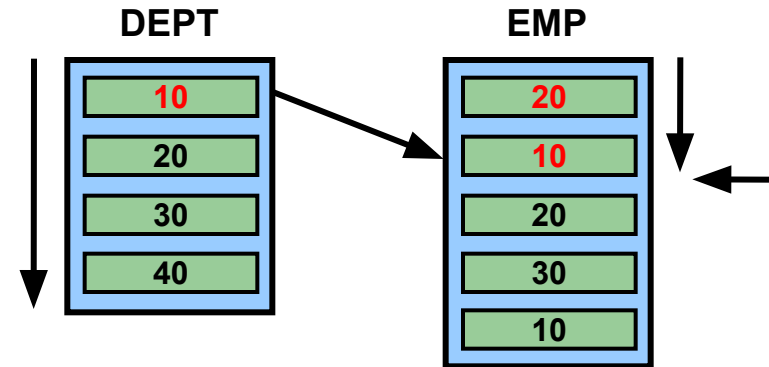


```
SELECT /*+ USE_NL(e d) */
d.deptno,d.dname,e.empno,e.ename
FROM emp e, dept d
WHERE e.deptno(+) = d.deptno;
```



# Semijointures

Les semijointures recherchent uniquement la première correspondance.



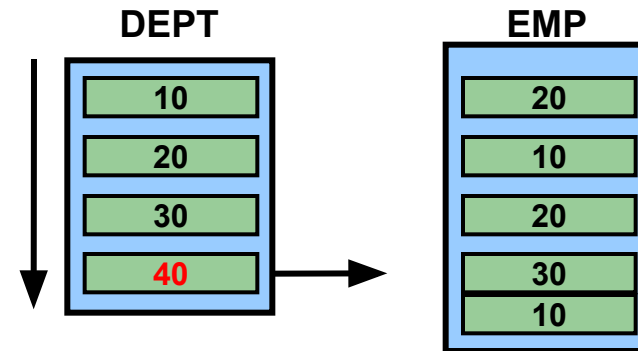
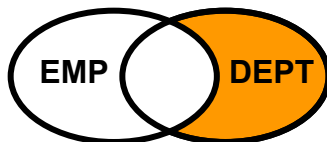
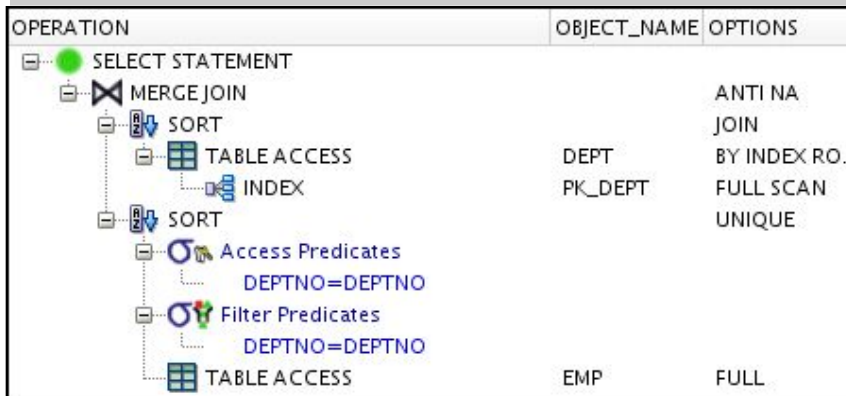
```
SELECT deptno, dname
FROM   dept
WHERE  EXISTS (SELECT 1 FROM emp WHERE emp.deptno=dept.deptno) ;
```

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
MERGE JOIN		SEMI
TABLE ACCESS	DEPT	BY INDEX RO.
INDEX	PK_DEPT	FULL SCAN
SORT		UNIQUE
Access Predicates		
EMP.DEPTNO=DEPT.DEPTNO		
Filter Predicates		
EMP.DEPTNO=DEPT.DEPTNO		
TABLE ACCESS	EMP	FULL

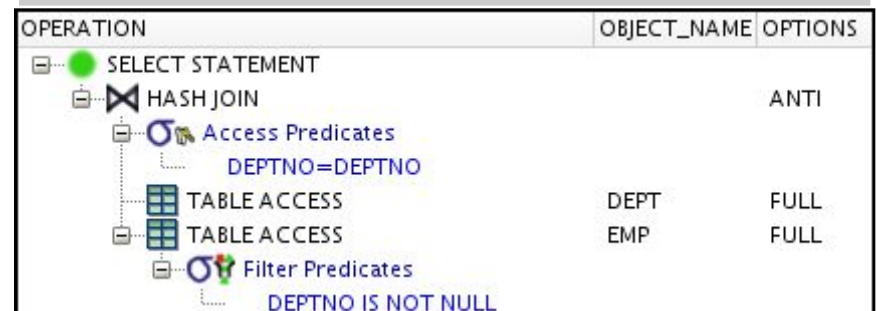
# Antijointures

Inversion des résultats qui auraient été renvoyés par une jointure

```
SELECT deptno, dname
FROM dept
WHERE deptno not in
(SELECT deptno FROM emp);
```



```
SELECT deptno, dname FROM dept
WHERE deptno IS NOT NULL AND
deptno NOT IN
(SELECT /*+ HASH_AJ */ deptno FROM
emp WHERE deptno IS NOT NULL);
```





# **Chapitre 6**

## **Diagnostiquer le cache de tampons**

# Comportement du processus serveur pour le traitement des blocks

Pour lire le bloc dont il a besoin, le processus serveur suit les étapes:

1. Le serveur commence par vérifier s'il trouve le bloc requis dans le cache de tampons.
  - ✓ Si c'est le cas, il l'éloigne de l'extrémité LRU « devient le plus souvent utilisé »
  - ✓ Si le bloc est introuvable dans le cache de tampons, le processus serveur doit lire le bloc dans le fichier de données.
    - a. Il recherche d'abord une mémoire tampon disponible
      - S'il l'a trouvé : Lit le bloc à partir du fichier de données
      - Sinon : Le serveur signale au  $DBW_n$  de vider les tampons "dirty" puis, lit le bloc à partir du fichier de données
    - b. Une fois que le bloc est disponible dans le cache, il est mis à l'extrémité LRU
2. Si le bloc n'est pas cohérent (La valeur a été modifiée mais non encore validée), le serveur reconstitue une version antérieure à partir du bloc en cours et des segments d'annulation (rollback segments).

# Objectifs et techniques de réglage

- **Objectifs du réglage :**
  - Blocs disponibles en mémoire pour les processus serveurs pour éviter les E/S physiques
  - Pas d'attente au niveau du cache de tampons
- **Mesures de diagnostic**
  - Vérifier les taux de succès en mémoire cache
  - Utiliser la fonction de conseil disponible dans la vue V\$DB\_CACHE\_ADVICE
  - Vérifier les événements d'attentes « Wait » au niveau du cache
- **Techniques de réglage :**
  - Réduire le nombre de blocs requis par les instructions SQL
  - Augmenter la taille du cache de tampons
  - Utiliser plusieurs pools de tampons
  - Mettre les tables en mémoire cache
  - Eviter d'utiliser le cache de tampons pour les opérations de tri et de lecture en parallèle

# Mesurer le taux de succès en mémoire cache

## Le taux de succès en mémoire Cache:

- ✓ *Indique le pourcentage de trouver un bloc dans le cache et ne pas devoir le lire à partir du fichier de données*
- ✓ *il est conseillé de le consulter dans des conditions de charge globale plutôt qu'au démarrage*
- ✓ *Sa valeur ne doit pas trop descendre au-delà de 98%*
- ✓ *Il est calculé à partir de la vue V\$SYSSTAT :*

```
SQL> SELECT 1 - (phy.value - lob.value - dir.value)
        / ses.value "CACHE HIT RATIO"
2  FROM    v$sysstat ses, v$sysstat lob,
3          v$sysstat dir, v$sysstat phy
3  WHERE   ses.name = 'session logical reads'
4  AND     dir.name = 'physical reads direct'
5  AND     lob.name = 'physical reads direct (lob)'
6  AND     phy.name = 'physical reads';
```

- ✓ *On peut, aussi, le lire à partir du rapport AWR:*

```
Instance Efficiency Percentages (Target 100%)
  Buffer Nowait %:  99.97
    Buffer Hit  %:  93.87
```

# **Conseils sur l'utilisation du taux de succès en mémoire cache**

- ✓ **Le taux de succès est affecté par les méthodes d'accès aux données :**
  - Balayages complets de table
  - Conception des données ou de l'application
  - Table volumineuse avec accès aléatoire
  - Répartition irrégulière des succès en mémoire cache
  
- ✓ **Si le taux de succès est faible (<< 98%), il faut prévoir une augmentation de la taille du DB\_CACHE\_SIZE dans les cas :**
  - Les éventuels événements d'attents ont été réglés.
  - Les instructions SQL ont été réglées.
  
- ✓ **Si l'augmentation du cache de tampons n'a pas été efficace utiliser la technique des pools de tampons multiples (Pool : KEEP & RECYCLE )**

# Autres indicateurs de performances du cache de tampons

A partir de la vue `V$SYSTEM_EVENT` :

- **Buffer Busy Waits**

Cet événement indique que plusieurs processus tentent d'accéder simultanément à certains tampons dans le cache de tampons. Les classes de tampons courantes pour lesquelles surviennent des événements Buffer Busy Waits incluent le **bloc de données, l'en-tête d'annulation**

- **Free Buffer Waits**

Cet événement Wait indique qu'un processus serveur n'a pas été en mesure de trouver une mémoire tampon disponible et a chargé le Database Writer de libérer des mémoires tampon en supprimant des tampons, il peut arriver que DBWn n'enregistre pas les tampons "dirty" assez rapidement sur le disque

```
SQL> SELECT event, total_waits
2  FROM    v$system_event
3  WHERE   event in
4          ('free buffer waits', 'buffer busy waits');
```

EVENT	TOTAL_WAITS
-----	-----
free buffer waits	337
buffer busy waits	3466

# Régler les problèmes d'attentes au niveau du cache de tampon de la BD

Pour réduire les attentes « Waits » :

## □ Buffer Busy Waits :

- ✓ Utiliser La gestion automatique de l'espace libre dans les tablespaces:  
via l'option : "SEGMENT SPACE MANAGEMENT AUTO"
- ✓ Utiliser la gestion automatique des segment d'annulation

## □ Free Buffer Waits

- ✓ Utiliser des unités de disques d'ur plus rapides
- ✓ Utiliser plusieurs processus DBWRn en agissant sur le paramètre:  
`DB_WRITER_PROCESSES=1, . . . , 9`

# Utilisation de la fonction de conseil sur le cache de tampons

- La fonction de conseil sur le cache de tampons active et désactive la collecte des statistiques pour prévoir ce qui peut se passer avec les différentes tailles de cache.
- Ces statistiques permettent aux DBA de choisir la taille de cache de tampons la mieux adaptée à une charge globale donnée.
- La fonction de conseil sur le cache de tampons est activée à l'aide du paramètre d'initialisation

**DB\_CACHE\_ADVICE :**

Ce paramètre est dynamique et peut être modifié à l'aide de  
**ALTER SYSTEM set DB\_CACHE\_ADVICE=ON | ready | off ;**

Où

**OFF** : la fonction de conseil est désactivée

**READY** : la fonction de conseil est désactivée mais une quantité de mémoire lui est allouée.

**ON** : la fonction de conseil est activée et les statistiques sont calculées dans v\$DB\_CACHE\_ADVICE



## Vue V\$DB\_CACHE\_ADVICE

La vue contient différentes lignes qui prévoient le nombre de lectures physiques pour les différentes tailles de cache

```
SELECT size_for_estimate , estd_physical_reads
FROM   V$DB_CACHE_ADVICE
WHERE  name = 'DEFAULT'
      AND block_size = <db_block_size>;
```

Cache Size (MB)	Phys Reads
30	192,317,943
182	25,668,196
212	17,850,847
243	13,720,149
273	11,583,180
304	10,282,475
334	9,515,878
364	8,909,026
395	8,495,039

*La sortie suivante montre que si le cache faisait 364 Mo, au lieu de la taille actuelle de 304 Mo, le nombre estimé de lectures physiques serait de 9.5 millions. L'augmentation de la taille du cache au-delà de sa taille actuelle n'offrirait donc aucun avantage significatif.*

# Utiliser la fonctionnalité d'information sur le cache de tampons avec Enterprise Manager

Database Instance: orcl.oracle.com > Memory Parameters

## Memory Parameters

Page Refreshed September 30, 2005 7:51

Show

SGA **PGA**

The System Global Area (SGA) is a group of shared memory structures that contains data and control information for the Oracle database. The SGA is allocated in memory when an Oracle database instance is started.

Automatic Shared Memory Management **Disabled**

Shared Pool  MB

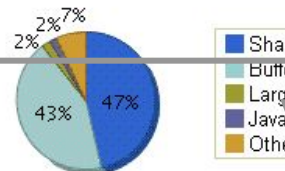
Buffer Cache  MB

Large Pool  MB

Java Pool  MB

Other (MB) **14**

Total SGA (MB) **214**



### Maximum SGA Size

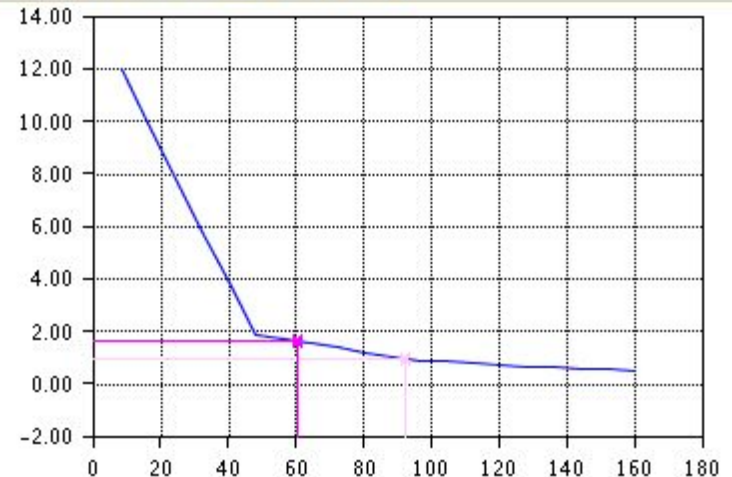
The Maximum SGA Size specifies the maximum memory that the database may allocate. If you specify a value for Maximum SGA Size, you can later dynamically change SGA component sizes (provided the total SGA size does not exceed the Maximum SGA Size).

Maximum SGA Size\* (MB)

The database must be restarted before any changes to this value take effect.

## Buffer Cache Size Advice

Relative change in physical reads



Cache Size (MB)

- Change in physical reads for various cache sizes
- Current cache size

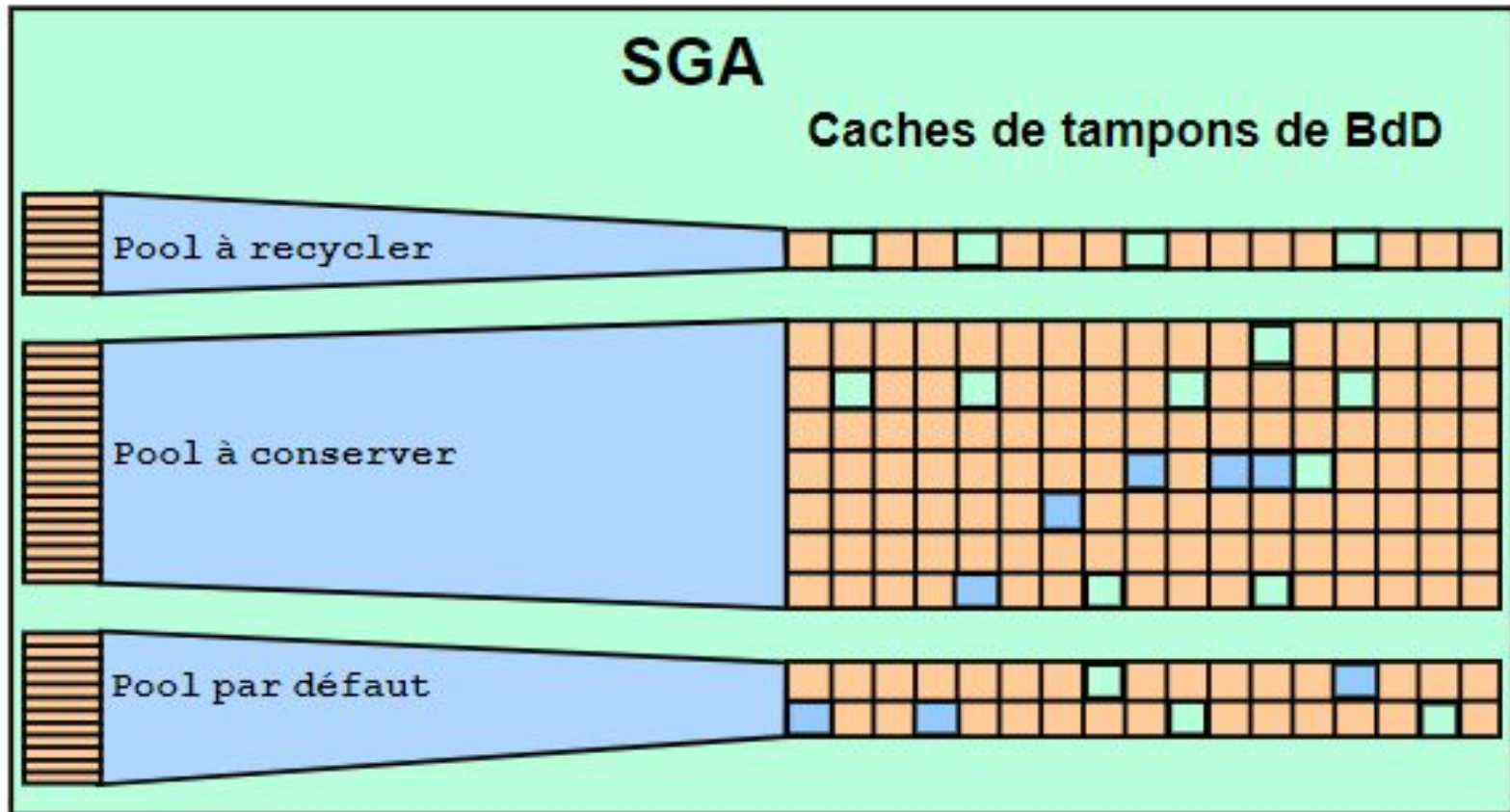
Cache Size (MB) **60**

☒ **TIP** You can click on the curve in the graph to set new value.

Cancel

OK

# Utilisation de pools de tampons multiples



# Utilisation des pools de tampons multiples

- **Pool DEFAULT**

- Obligatoire, géré par LRU
- Définit par le paramètre `DB_CACHE_SIZE`
- Contient les blocs qui ne sont déstinés ni au Pool « keep » ni « recycle »

- **Pool à conserver « KEEP »**

- Les blocs contenus dans ce pool vont y être conserver et ne seront jamais supprimés par DBWRn
- Le pool keep n'est pas géré par LRU
- Choisir les segments dont les blocs sont utilisés de manière répétée.
- La taille des segments est inférieure de 10 % à celle du pool de tampons par défaut.
- Pour utiliser le pool KEEP, définir `DB_KEEP_CACHE_SIZE`

- **Pool à recycler**

- Les blocs contenus dans ce pool vont être immédiatement supprimer une fois la transaction terminée
- Le pool keep n'est pas géré par LRU
- Les blocs ne sont pas réutilisés hors de la transaction.
- La taille des segments est plus de deux fois supérieure à celle du pool de tampons par défaut.
- Pour utiliser le pool RECYCLE, définir `DB_RECYCLE_CACHE_SIZE`

# Activer des pools de tampons multiples

```
CREATE INDEX cust_idx ...  
    STORAGE (BUFFER_POOL KEEP ...) ;  
  
ALTER TABLE customer  
    STORAGE (BUFFER_POOL RECYCLE) ;  
  
ALTER INDEX cust_name_idx  
    STORAGE (BUFFER_POOL KEEP) ;
```

# Calculer le taux de succès pour plusieurs pools

```
SQL>
SQL> SELECT name,
            1 - (physical_reads /
            (db_block_gets +
            consistent_gets)) "HIT_RATIO"
  2   FROM    sys.v$dbuffer_pool_statistics
  3   WHERE   db_block_gets + consistent_gets > 0;
```

NAME	HIT_RATIO
KEEP	.983520845
RECYCLE	.503866235
DEFAULT	.790350047

# Vues du dictionnaire reflétant les pools de tampons

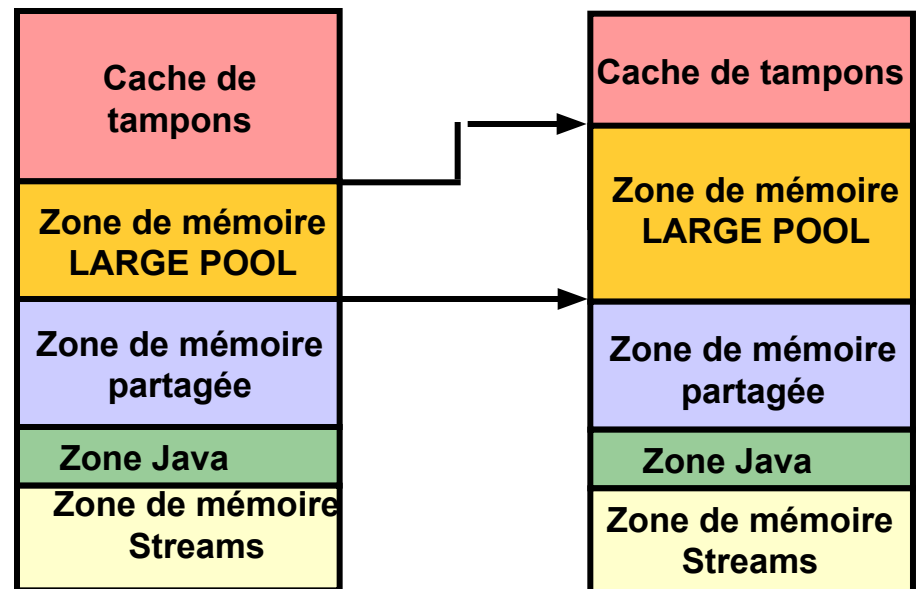
```
SQL> select id, name, block_size, buffers  
2 from v$buffer_pool;
```

ID	NAME	block_size	BUFFERS
---	-----	-----	-----
1	KEEP	4096	14000
2	RECYCLE	4096	2000
3	DEFAULT	4096	4000

# Gestion automatique de la mémoire partagée : Présentation

Gestion automatique de la mémoire permet :

- une adaptation automatique à l'évolution de la charge globale
- Utilisation de la mémoire plus efficace
- Elimination des erreurs dues à une mémoire insuffisante
- Si un fichier SPFILE est utilisé, les dernières tailles utilisées sont conservées après un arrêt





# Paramètres de dimensionnement de la mémoire SGA : Présentation

- Avec ASMM, cinq composants importants de la mémoire SGA peuvent être dimensionnés automatiquement.
- Les pools de tampons autres que le pool de tampons par défaut ne sont pas réglés automatiquement.
- Le tampon de journalisation n'est pas un composant dynamique, mais sa taille par défaut convient en général.

## Paramètres à réglage automatique

SHARED\_POOL\_SIZE  
DB\_CACHE\_SIZE  
LARGE\_POOL\_SIZE  
JAVA\_POOL\_SIZE  
STREAMS\_POOL\_SIZE

## Paramètres à réglage manuel dynamique

DB\_KEEP\_CACHE\_SIZE

DB\_RECYCLE\_CACHE\_SIZE

DB\_nK\_CACHE\_SIZE

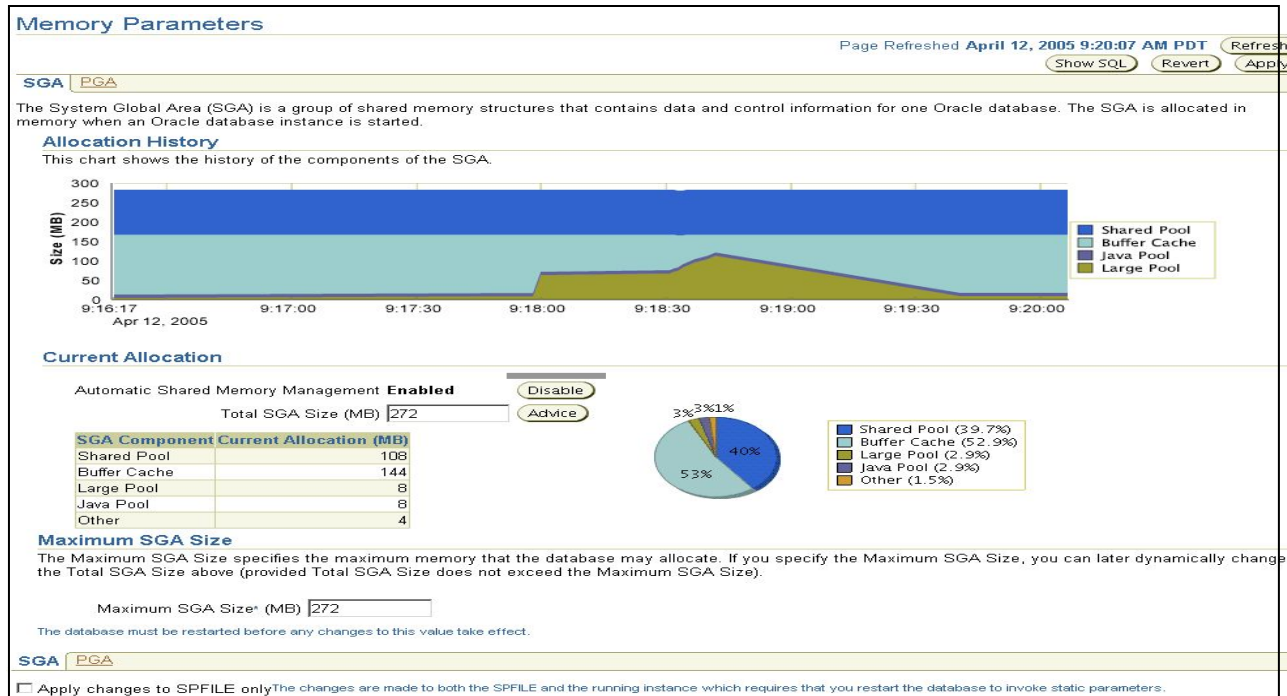
SGA\_TARGET

## Paramètres à réglage manuel statique

LOG\_BUFFER\_SIZE

SGA\_MAX\_SIZE

# Activer la gestion automatique de la mémoire (ASSM : Automatic Shared Memory Management)



**Gestion Automatique : SGA\_TARGET <> 0**  
**Gestion Manuel : SGA\_TARGET = 0**