Architecture et Administration des SGBDR

Application au SGBD Oracle

Nacer Boudjlida
http://www.loria.fr/~nacer

Nancy Université, Université Henri Poincaré Nancy 1
FST, Secteur scientifique MIAE et ESIAL
(Septembre 2010)

Architecture et Administration des SGBDR

Transparents tirés de :

- *Gestion et Administration des Bases de Données. Application à Sybase et Oracle.*
  N. Boudjlida, Dunod, Collection Sciences Sup.
  Octobre 2003.
- *Bases de données et systèmes d'informations. Le modèle relationnel:langages, systèmes et méthodes.*
  N. Boudjlida, Dunod, Collection Sciences Sup.
  Septembre 1999.

Sommaire

Chapitre I : Introduction

1. Généralités
2. Administration systèmes : Préalables

1- Généralités

- Administrer $\simeq$:
  - Installer (Serveur(s) SQL et Serveur(s) Back-up)
  - Gérer et contrôler les espaces (disque, mémoire)
  - Allouer rôles et privilèges aux utilisateurs
  - Sauvegarder/Restaurer les bases
  - Diagnostiquer les problèmes système
  - Configurer le système pour de meilleures performances

1- Généralités

- Rôles requis pour administrer : administrateur, officier de sécurité
- Administration par *isql* (Sybase)
- Administration sous *sqlplus* (Oracle)
- Environnements graphiques d'administration
  - Préalable : Maîtrise des concepts du système.

2- Administration système : préalables

1. Installation de serveur(s) et de produit(s)
2. Allocation de ressources
3. Sauvegardes et restoration
4. Maintenance
5. Historisation ("carnet de bord")

2.1- Installations (serveurs, produits)

- Consulter le *Release Bulletin* pour :
  - (In)Compatibilités éventuelles
  - Problèmes (*System Problem Reports*) et solutions (*Closed Problem Reports*)
- *Installation et upgrade*
  - Consulter les guides d'installation et de configuration
  - Consulter l'administrateur système : espaces, E/S asynchrones, etc.
  - [Ajout de protocoles réseaux]
- Configuration et test des connexions clients

2.2- Allocation de ressources physiques au serveur SQL

- Mémoire, Espace disque, CPU (cas multi-processeurs)
- Choix machine dédiée (au serveur SQL) vs partagée
- Planification de l'utilisation des ressources
  - Pour les reprises : données et logs sur des unités physiques différentes
  - Miroir des données importantes
- Configuration au niveau système d'exploitation
  - *Raw devices* (cf. administrateur Unix)
  - Nombre de connexions, d'unités, etc.

2.3- Sauvegardes et restorations

- Clé de voûte : dictionnaire, fichiers de configuration ; à sauvegarder après :
  - Création/suppression de base(s)
  - Initialisation de nouvelles unités bases de données
  - Ajout de nouvelles unités de sauvegarde
  - Utilisation d'une commande de *mirroring* (*multiplexage*)
  - Création/suppression/modification d'un segment
  - Ajout de nouveaux comptes

Sauvegardes et restorations

- + garder des copies *"off line"*
- Automatisation des procédures de sauvegardes (scripts ou utilitaires de déclenchement automatique)
- Vérifier la cohérence de la base avant sauvegarde
- Contrôler la taille du *log* ou utilisation de *thresholds*
- Sauvegarder périodiquement le *log* (de préférence avant la base)

2.4- Maintenance en exploitation

- Lancement du serveur en même temps que le système d'exploitation
- Examen périodique du fichier de trace et d'alert (Oracle) ou *error log* (Sybase)
- Raffraîchissement du fichier

2.5- Historisation : "carnet de bord"

- Changements, problèmes
- Référence des contacts (système, constructeur, etc.)
- Paramètres de configuration
- Commandes de création de bases, de comptes et d'utilisateurs
- Noms, localisations, tailles des unités initialisées
- Échéancier des contrôles de cohérence, sauvegardes, examen de l'*Audit*, recompilation de procédures, etc.
- Hardware et système d'exploitation
- Procédures de sauvegarde et de reprise

Chapitre II : Installation et lancement de serveurs

1. Installation de serveurs SQL

- Procédure d'installation dépendante du SGBD
- Tâches préalables :
  - *Installation Guide*, *Release Notes*
  - Machine(s) d'installation : serveur de données, outils clients
  - Paramètres importants de configuration/d'initialisation
    - *dynamiques* : prise en compte immédiate
    - *statiques* : relance du serveur
- 1ères expériences : maximum de paramètres par défaut
- Puis installations "personnalisées"
- Hypothèse : logiciel déjà copié de son support de livraison vers la machine d'accueil

II- Installation : Application à Oracle

- *Oracle Universal Installer* : environnement graphique
- Quelques concepts pour comprendre ses fonctionnalités :

1. *Serveur de données Oracle* : Base de données + Instance Oracle
2. *Instance* :
   (a) Ensemble de processus d'arrière-plan
   (b) Zone système globale (*System Global Area*, SGA)
3. *SGA* : Tampons partagés, dont tampon de données et tampon de journal
4. *Dictionnaire de données* : créé dans chaque base (pas de base des bases)

Installation : Application à Oracle

5. *Control file* : un (au moins) par base
   - nom, date de création, localisation des espaces
   - consulté à chaque lancement (*startup*)
   - mis à jour automatiquement si modification des caractéristiques de la base (ajout/suppression d'espaces, . . .)
6. *Init file* : Paramètres d'initialisation de la base
7. *Transactions et journalisation* :
   (a) Images *avant* : segments/fichiers d'annulation (*rollback segments/undo tablespaces*)
   (b) Images *après* : journaux de ré-exécution (*redo log files*)

Scénario typique d'installation (Oracle 10g)

1. Lancer Oracle Universal Installer (Exemple : D:\ORACLE-10G-SRC\database\setup)
2. Indiquer l'emplacement du logiciel à installer
3. Choisir un type d'installation :
   - Enterprise/Standard/Personal Edition
   - Personnalisé
4. Indiquer le répertoire cible de l'installation (*OraHome*)
5. Consulter les messages des pré-requis pour l'installation
6. Choisir : Installation avec/sans création initiale de base de données
7. Si installation avec création de base de données, en choisir le type :
   - Usage général

   - Traitements des transactions (*OLTP*)
   - Data Warehouse (*OLAP*)
   - Avancé : Configuration personnalisée
8. Lancement effectif de l'installation
9. Configuration de divers assistants : exemples
   - Oracle Net Configuration Assistant
   - Oracle Database Configuration Assistant
   - iSQL*Plus Configuration Assistant
10. Création éventuelle de la base de données
11. (Dé)verrouillage de comptes et changement éventuel de mots de passe des utilisateurs privilégiés (droits administrateurs) :
    - *sys*/*change_on_install*; *system*/*manager*
12. Fin de l'installation : Oracle et ses services dans le menu

Installation : Application à Oracle

La suite :

1. Installation
2. Eléments du dictionnaire
3. Processus d'arrière plan d'une instance Oracle

Le dictionnaire Oracle

- *Ensemble d'ensembles* de tables et de vues (*all*|*user*|*dba*|*v$*_xxx)
- Créé et rangé dans chaque base dans l'espace *SYSTEM*
- Propriétaire : Utilisateur privilégié *SYS*
- Synonymes publics des vues pour les désigner sans le préfixe de leur propriétaire
- Table *SYS.AUD$* : seule table modifiable (cf. *auditing*).

1. Vues *USER*_xxx :
   - Concernent les objets d'un utilisateur
   - Accessibles par cet utilisateur
   - Exemple : Vue *USER_TABLES* : tables d'un utilisateur

Le dictionnaire Oracle

2. Vues *ALL*_xxx :
   - Etendent les informations des vues *USER* par des informations sur les objets auxquels il a accès
   - Exemple : Vue *ALL_TABLES* : tables d'un utilisateur + tables auxquelles il a accès.

3. Vues *DBA*_xxx :
   - Plus de colonnes que les autres
   - Information sur les objets de tous les utilisateurs
   - Exemple : Vue *DBA_TABLES* : toutes les tables d'une base

Le dictionnaire Oracle

4. Vues *V$*xxx :
   - Tables de performance dynamiques
   - Concernent l'activité d'une base
   - Exemple : Vue *V$DATAFILE* : Information sur les fichiers physiques d'une base.

5. Vue *DICTIONARY* : Liste des tables et des vues du dictionnaire.

- *DESCRIBE* Nom de table ou de Vue : liste des colonnes

Le dictionnaire Oracle

```
SQL> desc dict
 Name                      Null?    Type
 ------------------------- -------- ------------------
 TABLE_NAME                         VARCHAR2(30)
 COMMENTS                           VARCHAR2(4000)

SQL> select count(*) from dict;

  COUNT(*)
----------
      1362
```


Le dictionnaire Oracle

- select * from dictionary : A EVITER
- PREFERER : Recherche avec *motif*

```
select table_name from dict
where upper(table_name) like upper('%TABLESPACE%');

TABLE_NAME
------------------------------
USER_TABLESPACES
   ...
V$TABLESPACE
   ...
```

Installation : Application à Oracle

La suite :

1. Installation
2. Eléments du dictionnaire
3. Processus d'arrière plan d'une instance Oracle

Les processus Oracle

1. *Processus utilisateurs*
   - Créés pour exécuter le code d'un outil Oracle (comme *Oracle Enterprise Manager*) ou d'un programme d'application (comme un programme *Pro*C/C++*)
   - Gèrent la communication avec les processus serveur grâce à l'interface de programme (*program interface*) : mécanismes
     - de formattage et de transmission de données
     - de conversion de données (cas machines hétérogènes)
2. *Processus Oracle* :
   (a) un processus serveur
   (b) un ensemble de processus d'arrière-plan (*background processes*)

Processus Oracle : (a) Processus serveur

- Créé pour gérer les demandes d'actions des utilisateurs
- Chargé de la communication avec le processus utilisateur et de l'interaction avec Oracle
- Si configuration de type *serveur dédié* : un processus serveur ne gère que les demandes d'un seul processus utilisateur
- Si configuration multitâches (*multithreaded*) : plusieurs processus utilisateurs partagent un nombre restreint de processus serveurs.

Processus Oracle : (b) Processus arrière-plan

Processus Oracle : (b) Processus arrière-plan

Exécution asynchrone des opérations d'entrées/sorties et de contrôle d'autres processus Oracle; Principaux processus :

- DBW_i (*Database Writer*$_i$)
  - Chargé des écritures physiques
  - Un par défaut (*DBW0*).
  - Sinon i, $i \in [0..9]$ ou $0 \leq i \leq db\_writer\_processes$)
  - *db_writer_processes* : paramètre d'initialisation d'une base

Processus Oracle : (b) Processus arrière-plan

- *LGWR (Log Writer)* : écriture
  - dans les fichiers *Redo Log*
  - dans leurs miroirs éventuels (cf. chapitre Sécurité).
- *CKPT* : informe le DBW de l'arrivée d'un point de contrôle (*checkpoint*)

Processus Oracle : (b) Processus arrière-plan

- *ARC_i* :
  - Archivage du journal si
    - base en mode *archivelog*
    - et archivage automatique autorisé (cf. chapitre Sécurité)
  - Peut, si la charge du serveur le nécessite, créer des *ARC_i*
    - $i \in [0..9]$ ou $0 \leq i \leq log\_archive\_max\_processes$
    - *log_archive_max_processes* : paramètre *dynamique*
  - *PMON (Process Monitor)* : chargé de
    - Reprise en cas d'échec d'un processus utilisateur,
    - Nettoyage de sa mémoire cache et Libération de ses ressources

Processus Oracle : (b) Processus arrière-plan

- *SMON (System Monitor)* réalise, entre autres :
  - la reprise, lors de la relance d'une instance (*startup*) ;
  - le "nettoyage" des segments d'annulation (*rollback segments*) qui ne sont plus utilisés (cf. chapitre Organisation et Stockage) ;
  - la reprise
    - de transactions "mortes" qui n'ont pas pu être prises en compte au moment de l'incident,
    - de l'instance Oracle du fait d'erreurs de lecture ou de mises hors ligne (*offline*) d'unités (reprise des transactions lors de la remise en ligne) ;

Processus Oracle : (b) Processus arrière-plan

- *RECO* :
  - Résoudre les transactions distribuées en suspens du fait de problèmes réseau ou système
  - Répétition de tentatives de connexion à la base pour annuler ou valider
- *Dnnn* (*Dispatcher*) : processus optionnel
  - existe dans une configuration *multithreaded* du serveur
  - un par protocole de communication
  - chargé du routage des demandes et des réponses entre le processus serveur et les processus utilisateurs
- *LCK0* : Verrouillage inter-instances dans *Oracle parallel server* (*Oracle parallel server* : accès à une base par plusieurs instances).

3. Installation de serveurs : Conclusion

- Installation = activité nécessitant une préparation minutieuse
- En coopération avec l'administrateur système et réseau
- Connaissance minimale
  - Concepts et structure du serveur (bases propres éventuelles, structures mémoire requises, etc.)
  - Paramètres initiaux de configuration
- Configuration : données quantitatives
  - Nombre de bases attendues et tailles estimées
  - Nombre d'utilisateurs locaux et/ou distants, etc.
  - "Bon réglage" (*tuning*)

Chapitre III : Organisation et stockage des données

Contenu du chapitre :

I- Organisation et Stockage des données

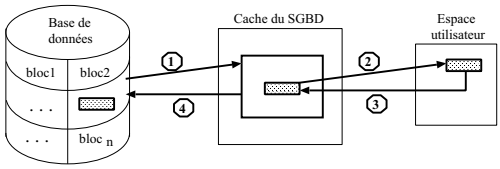
- Comprendre :
  1. Stockage et accès sans index
  2. Index : Organisation, structuration, accès
- Pour savoir :
  1. Gérer les espaces ;
  2. Forme à donner aux requêtes pour maximiser l'utilisation des index ?
  3. Utiliser les outils du serveur pour examiner les choix de son optimiseur.

Organisation et Stockage des données : Sommaire

1. Organisation et stockage : pages et *extents*

- Hypothèse : données organisées et stockées “en vrac” (*en tas* ou *heap*)
- Espace (données, index, log) : blocs (*pages*) de même taille
- *Page* : Unité de lecture/écriture *physique*
- Lecture/écriture *via* des tampons (cache) du système
- *Lecture/Ecriture logique* : Lecture/écriture dans les pages des tampons
- *Défaut de page* : Absence d’une page dans le cache
- *Extent* :
  - Nombre, fixe ou variable, de pages contiguës
  - Attribué généralement à un seul objet

Organisation et stockage en pages



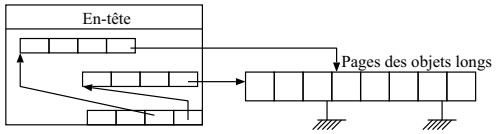
Structure générale des pages

1. *En-tête* :
   - Identification de l’objet contenu dans la page
   - Chaînage page suivante, page précédente de l’objet, etc. ;
2. *Corps* : Valeurs des tuples ;
3. *Table des déplacements* :
   - Localisation des tuples dans la page
   - Sybase : en fin de page
   - Oracle : En début de page

2. Représentation des tuples dans les pages

- En-tête :
  - identification du tuple,
  - nombre de colonnes à valeurs inconnues (*NULL*)
  - nombre de colonnes de longueur variable, etc.)
- Valeur effective du tuple :
  - Colonnes de type “ordinaire”
  - ou pointeurs vers les valeurs des objets longs

3. Stockage des objets de grande taille

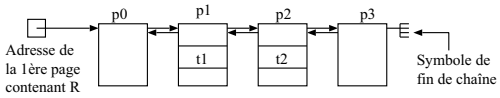


4. Organisation “en tas” (*heap*)

- Comportement :
  1. Recherche,
  2. Insertion,
  3. Suppression,
  4. Modification de tuples.

4.1- Organisation “en tas” et recherche de tuples

- Recherche de tuples de R
- Adresse de la première page contenant R : dans une des tables de la méta-base
- Parcours des pages de R



4.2- Organisation “en tas” et insertion

- Dans la dernière page de la relation, si espace disponible
- Sinon, dans une page vide de l’*extent* courant
- Si *extent* saturé :
  - Allocation d’un nouveau
  - Insertion dans sa première page
- Note : Adresse de la dernière page

4.3- Organisation “en tas” et suppression

- Localisation du tuple
- Effacement
- Décalage des tuples (éviter “l’effet gruyère”)
- Mise à jour des déplacements des tuples
- Cas page vide après suppression ?
- Cas extent vide après suppression ?

4.4- Organisation “en tas” et mise à jour

- Parcours des pages ;
- *Cas taille du tuple modifié* :
  1. = taille(ancien) : Modification du tuple ;
  2. < taille(ancien) :
     (a) Modification du tuple et “décalage” vers le haut ;
     (b) Modification des pointeurs de tuples.
  3. > taille(ancien) et place disponible dans la page : cf. cas 2, mais avec “décalage” vers le bas ;
  4. Sinon (*Migration de tuple*)
     (a) Suppression du tuple de la page ;
     (b) Insertion dans la dernière page du tas.

Organisation “en tas” et mise à jour

- Gérer au mieux les migrations de tuples
- Contrôle du remplissage *(densité)* des pages
  - Sybase : *max_rows_per_page*
  - Oracle : *PCTUSED, PCTFREE* (voir plus loin)

4. Organisation “en tas” : Conclusion

- Quand utiliser des “tas” ?
  - Petites relations utilisant peu de pages ;
  - Pas d’accès direct à un tuple ;
  - Pas d’ordre sur les ensembles résultats ;
  - Relation ayant des tuples non uniques + ce qui précède ;
  - Relations avec peu de modifications et d’insertions.
- Maintenance périodique

5. Gestion de la mémoire cache

- $\simeq$ mémoire paginée dans les systèmes d’exploitation
- Recherche d’une donnée
  1. Dans une page du cache ?
  2. Si absente (*défaut de page*) : page la contenant $\longrightarrow$ cache
  3. Si toutes les pages du cache sont occupées: en “vider” une
- Stratégies de choix de la page du cache à remplacer :
  1. Dernière page utilisée (*Most Recently Used*, MRU)
  2. La moins récemment utilisée (*Less Recently Used*, LRU)

Stratégies de “caching”

- Cache géré comme une chaîne de pages Most Recently Used/Less RU
- “Âge” d’une page : MRU $\longrightarrow$ LRU
- Quand des pages modifiées atteignent un point dans la chaîne (*wash marker*) : Ecriture asynchrone de la page [ou *checkpoint*] par le serveur (i.e. quand une page arrive en fin de chaîne, elle est “propre” et peut alors être ré-utilisée)
- *Stratégies de remplacement de pages* :
  1. LRU
  2. MRU (*fetch and discard*)

1- Stratégie LRU de remplacement de pages

- Lecture séquentielle de pages en tête de la chaîne MRU
- “Pousser” éventuellement des pages vers la LRU
- Quand une page modifiée “passe” le *wash marker* : écriture
- Si nouveau besoin/accès à cette page : la remettre en tête de MRU.
- $\Longrightarrow$ Recherche d’une page p_i et
  - p_i non modifiée et $p_i \in$ cache : p_i
  - p_i modifiée et $p_i \in$ cache : p_i en tête MRU
  - $p_i \neg \in$ cache : lecture disque (1 buffer) en tête de MRU.

2- Stratégie MRU de remplacement de pages

- Pages mises juste avant le *wash marker*
- Si $p_i \in$ cache : mettre p_i avant le marqueur
- Sinon :
  1. Lire p_i
  2. Mettre p_i avant le marqueur

II- Organisation et stockage des index

- Localisation des tuples
  - Parcours séquentiel de l’instance d’une relation
  - “Directement” via des index.
- Index : structure de données associant
  - valeur d’un attribut ou une liste d’attributs (*clé de l’index*)
  - et “adresses” des tuples contenant cette valeur

Typologie des index

1/4) *Index dense*

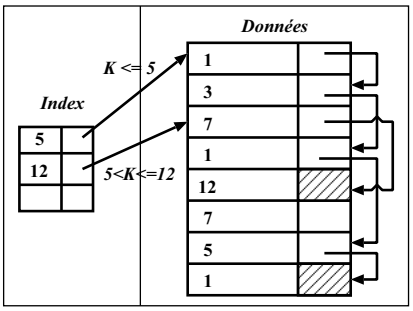
- Une entrée dans l'index par valeur de la clé
- Entrée : $< V_i,\ P_i >$ où
  - V_i : valeur d'une clé d'index C
  - P_i : tête d'une liste d'adresses de tuples
- Si Clé d'index = clé de la relation : pas de chaînage
- Rangement des tuples pas nécessairement contigu
- Exemple :

Typologie des index

2/4) *Index creux*

- Moins d'entrées que de valeurs de la clé
- Entrée : $< V_i,\ P_i >$ où
  - V_i : valeur d'une clé d'index C
  - P_i : tête d'une liste d'adresses de tuples t t.q. $\Pi_C(t) \leq V_i$

Typologie des index : Index Creux



Typologie des index

3/4) *Index primaire*

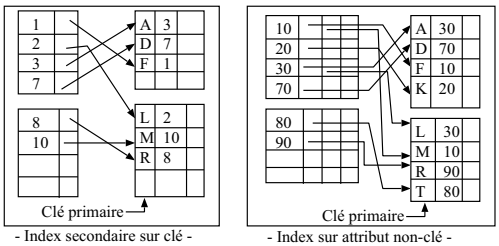
- Défini sur une clé primaire de relation
- Valeurs ordonnées dans la relation
- Ordre logique = ordre physique.
- Exemple :

Typologie des index

4/4) *Index secondaire*

- Construit sur un attribut dont les valeurs devraient être ordonnées.
- Mais relation déjà ordonnée sur sa clé primaire
- $\Longrightarrow$ ordre logique $\neq$ ordre physique

Typologie des index : Index secondaire



- Index secondaire sur clé -

- Index sur attribut non-clé -

Représentation des index

- Arbres équilibrés (*Balanced trees*, B-arbres)
- Feuilles toujours à égale distance de la racine
  1. Arbres B^+
  2. B-arbres

Représentation des index : (1) Arbres B^+

- Chaque nœud a entre n et $n/2$ nœuds fils
- n fixé pour un arbre donné
- *Nœud feuille* :
  - au plus $(n-1)$ valeurs et n pointeurs
  - Pas moins de $\lceil (n-1)/2 \rceil$ valeurs
  - Dernier pointeur : $\longrightarrow$ feuille suivante
- *Nœud racine et Nœuds internes* :
  - Entre $\lceil n/2 \rceil$ et n pointeurs
  - Premier Pointeur P_1 : clés $K_i,\ K_i < K_1$
  - Dernier pointeur P_p, $p \leq n$: clés K_j, $K_{i-1} \leq K_j < K_i$

Représentation des index : (1) Arbres B^+

- Recherche
  - $\simeq$ Recherche par dichotomie
  - longueur du chemin de la racine jusqu'aux feuilles $\leq \log_{\lceil n/2 \rceil} \mathcal{N}$
  - $\mathcal{N}$ étant le nombre de valeurs de la clé
  - Exemple : $n = 3$

Représentation des index : (1) Arbres B^+ et mise à jour

- Efficacité en recherche
- Activité supplémentaire en mise à jour :
  - Maintenir la propriété "B"
  - Garantir la structure des nœuds
- Insertion : cf. exemple de la page 63
  - Ajout d'un tuple de clé C.
- Suppression de la valeur P (sur l'arbre de la page 63)

Représentation des index : (2) B-arbres

- Similaires aux arbres B^+
- Différence : toutes les clés de l'index n'apparaissent pas aux feuilles
- Nombre de nœuds du B-arbre $<$ ceux du B^+
- Ajout de pointeurs dans les nœuds non-feuilles pour les valeurs de la clé de l'index qui n'apparaissent pas aux feuilles
- Exemple Correspondant à l'exemple d'arbre B^+, page 64

Organisation et stockage : III. Application à Oracle

- Espace de tables (*tablespace*) : ensemble d'unités logiques de stockage des objets d'une base
- Base : occupe une ou plusieurs *tablespaces*
- *Tablespace* : contenu rangé dans un plusieurs "fichiers de données" (*Datafile*)
- Un fichier physique : associé à une seule *tablespace* et à une seule base
- **Note :** *Datafile* n'accueille pas exclusivement des données

Application au SGBD Oracle

- Outre les *datafiles*, une base est dotée :
  - d'un fichier de paramètres de configuration (*control file*)
  - d'un fichier de paramètres d'initialisation (*init.ora*)
  - de fichiers de journalisation des transactions (*redo log* files, *rollback segments/undo tablespaces*)
- Gestion du *control file* et du fichier d'initialisation : cf. chapitre Bases de données et leurs objets
- Espaces physiques persistants (disque) structurés en :
  - Pages (appelés *blocs*),
  - *extents* et
  - *segments*

Application au SGBD Oracle

- Bloc = l'unité par défaut de lecture/écriture physique
- *Extent* : nombre déterminé de blocs contigus
- Segment : ensemble d'*extents* non nécessairement contigus
- La suite :
  1. Quelques notions sur les *tablespaces*
     - pour comprendre la gestion des blocs, des *extents* et des segments
     - Compléments dans le chapitre Bases de données et leurs objets
  2. Gestion des blocs et des *extents*
  3. Gestion des segments
  4. Gestion des espaces physiques non persistants (cache)

1. Introduction aux *tablespaces*

- *Tablespace SYSTEM*
  - automatiquement créée lors de la création de chaque base
  - contient, dans son premier *datafile* :
    - dictionnaire de la base
    - unités de programmes stockés (procédures, fonctions, paquetages et triggers)
- Recommandé : créer des *tablespaces* pour
  - contrôler le placement des objets
  - affecter des ressources aux utilisateurs

Introduction aux *tablespaces*

- Exemples de *tablespaces* "dédiés" :
  - Images avant modification (*undo tablespaces*)
  - Pour stockage exclusif des données ou des index
  - Espaces temporaires (pour les tris, par exemple)
- Durée de vie d'une *tablespace* : création (*create tablespace*) $\longrightarrow$ suppression
  - explicite (*drop tablespace*)
  - implicite : *tablespace temporaire* ($\longrightarrow$ fin de session de création)

Modes de gestion des *tablespaces*

- Espace des *tablespaces* découpé en blocs
- Blocs groupés en *extents*
- *Extents* (libres/occupés) gérés
  1. à l'aide du dictionnaire (*dictionary-managed*)
  2. localement à la *tablespace* (*locally-managed*)
- Mode de gestion (*dictionary/locally*)
  - choisi lors de la création (*create tablespace... extent management dictionary* ou *local*)
  - non modifiable

Modes de gestion des *tablespaces*

- Par défaut : *dictionary-managed*
  - Gestion d'une liste des *extents* libres dans des tables du dictionnaire
  - Journalisation des modifications de celles-ci
- Mode de gestion dit *local* (Versions $>$ Version 8.0) :
  - Matrice de positions binaires (*bitmap*) par *datafile* de la *tablespace*
  - Matrice mise à jour, sans journalisation, lors de chaque allocation ou libération d'*extent*
- Modes de gestion également applicables aux *tablespaces* temporaires
- Exemples :

Statuts et états d'une *tablespace*

1. Temporaire/permanente : changement par :
   alter tablespace . . . temporary/permanent
2. Accessible (*online"*)/Inacessible (*offline"*) :
   "alter tablespace Nom *online/offline"*
   - Pas de mise hors ligne de *SYSTEM*
   - Pas de mise hors ligne de *tablespaces* contenant des segments d'annulation (*rollback segments*) en cours d'utilisation
   - Pendant la mise hors ligne : données des transactions non achevées journalisées dans *segments d'annulation différée* (cf. chapitre Sécurité et reprise).

Etats et statuts d'une *tablespace*

- Mise hors ligne d'une *tablespace* ne contenant que des index : données indexées restent accessibles
- Mise hors ligne d'une *tablespace* par le système si incidents sur la *tablespace*
  - Exemple : échec de tentatives d'écriture par un des processus *DBWi*
- Si *tablespace* sur plusieurs fichiers physiques : tous les fichiers sont inaccessibles
- Mise en/hors ligne sélective de *datafiles* :
  "alter database . . . datafile . . . online/offline"

Etats et statuts d'une *tablespace*

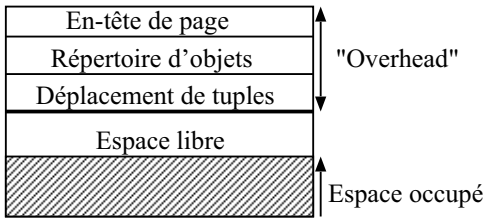
3. Lecture seule (*read only*) :
   - Par défaut, *tablespace* en lecture/écriture
   - Mise en lecture seule provisoire ou définitive
     - *alter tablespace . . . read only*
     - Changement d'état effectif à l'issue des transactions actives
     - Seules les opérations de consultation et de suppression d'objets (index, tables) restent autorisées.
- Exemples :
- *dba_tablespaces* : informations sur toutes les *tablespaces*
- *user_tablespaces* : sur celles accessibles par un utilisateur.

Application au SGBD Oracle

1. Notions sur les *tablespaces*
   - Compléments dans le chapitre Bases de données et leurs objets
2. Gestion des blocs
3. Gestion des *extents*
4. Gestion des segments
5. Gestion des espaces physiques non persistants (cache)

2. Blocs Oracle

- Taille multiple de la taille des blocs du système hôte



Blocs Oracle : *Descripteur ("overhead")*

- 84 à 120 octets

1. *En-tête* : adresse, type du segment d'appartenance, etc.
2. *Répertoire de tables* : informations sur les objets contenus dans le bloc
3. *Table des déplacements* des tuples :
   - 2 octets par entrée
   - Entrées non libérées lors de la suppression de tuples
   - Ré-utilisables lors d'insertions dans le bloc

Blocs Oracle : *Contenu*

- Espace occupé : "croît de bas en haut"
- Espace libre : peut parfois contenir des informations sur des transactions (*transaction entries*)
- *Paramètres de stockage INITRANS* et *MAXTRANS* :
  - Nombre initial et nombre maximum de transactions concurrentes sur le bloc ;
  - Applicables aux index, tables et groupements de tables (*cluster*).

Gestion et contrôle du remplissage d'un bloc

- Paramètres de stockage : Contrôler le remplissage de blocs de tables, groupements de tables (*cluster*) et index :
  1. *PCTFREE* : pourcentage minimum d'espace libre dans un bloc
  2. *PCTUSED* : pourcentage maximum pouvant être occupé
- Spécifiés lors de la création ou de la modification d'une table, d'un *cluster* ou d'un index (*create/alter table, cluster* ou *index*)
- Si taux de remplissage d'un bloc $>=$ *PCTFREE* :
  - Insertions interdites dans le bloc ;
  - Suppressions et modifications autorisées ;
  - Reprise des insertions si taux de remplissage $<$ à *PCTUSED*.

PCTFREE, PCTUSED : Exemple

- *PCTFREE* = 30% et *PCTUSED* = 50% :
  1. Insertions autorisées dans le bloc jusqu'à ce qu'il n'y reste que 30% d'espace libre ;
  2. Seules les mises à jour sont alors autorisées dans le bloc ;
  3. Insertions de nouveau permises lorsque le taux d'occupation du bloc descend au-dessous de 50%.

3. *Extents* Oracle

- Groupe de blocs contigus,
- Unité d'allocation d'espace pour un objet dans un segment,
- *Segment* :
  - Un ou plusieurs *extents* ;
  - Bloc d'en-tête d'un segment : répertoire des *extents*.
- *Formattage des extents* : au fur et à mesure des besoins
- *Forcer le formattage dès l'allocation* : clause *ALLOCATE EXTENT* dans *alter table* (administrateur).

Extents Oracle : Paramètres de stockage

- Sans paramètres de stockage explicites :
  - Allocation d'un extent (*initial extent*) à la création d'un objet ;
  - Quand saturé : allocation d'un nouveau (*increment*).
- *Paramètres de stockage* :
  - Taille et nombre d'extents pour un objet ;
  - Par défaut : paramètres de la *tablespace* du segment.
  1. Cas des tablespaces gérées localement
  2. Cas des tablespaces gérées par le dictionnaire

a) Extents : Paramètres de stockage et gestion locale de *tablespaces*

- *Extents* alloués dans le premier fichier (*datafile*) de la *tablespace* ayant le nombre requis de blocs libres contigus
- Localisation des blocs libres : Matrices d'occupation des *datafiles*.
- Taille d'un *extent* :
  - Fixe ou variable
  - Déterminée par le système.

a) Extents : Paramètres de stockage et gestion locale de *tablespaces*

- Modes de choix de la taille d'un *extent* :
  - Arguments *UNIFORM* ou *AUTOALLOCATE* de *create tablespace*
  - *UNIFORM* : Taille fixe (spécifiée ou 1MO par défaut)
  - *AUTOALLOCATE* (option par défaut)
    - Taille *extent* choisie par système
    - Taille minimum : 64KO
    - Valeur par défaut pour les *tablespaces* permanentes : 64KO
- Exemple :
  - *T1* : *tablespace* : 5M; *extents* : 128K
  - *T2* : *tablespace* : 2M; *extents* : 64K, minimum

b) Extents : Paramètres de stockage et gestion par le dictionnaire

- *Clause de stockage* de la *tablespace* : spécification de la taille de l'*extent initial*, de l'incrément, des suivants, etc.
- *Clause de stockage* s'applique à la plupart des objets logiques ou physiques d'une base (*clusters*, tables, *rollback segments*, partitions, etc.).
- (Quelques) Arguments de la clause de stockage :
  - *INITIAL, NEXT*, *MINEXTENTS, MAXEXTENTS*,
  - *PCTINCREASE*,
  - *BUFFER_POOL*.
- Exemple :

b) Extents : Paramètres de stockage et gestion par le dictionnaire

1. *INITIAL* x*K* ou y*M* :
   - Taille du premier *extent* (xKO ou yMO) ;
   - Alloué à un objet dès la création de sa définition (ou schéma) ;
   - Valeur par défaut : 5*taille d'un bloc ;
   - Allocations de blocs : pour éviter la fragmentation des espaces :

   (a) Allocation de $\lceil (t/taille\,d'un\,bloc) \rceil$ blocs,
   - si taille *t* (*xK* ou *yM*) $\neq$ multiple de la taille des blocs
   - et si *t* requiert moins de cinq blocs.

   (b) Allocation du plus petit multiple de cinq blocs satisfaisant la demande si nombre de blocs requis par $t > 5$.

b) Extents : Paramètres de stockage et gestion par le dictionnaire

2. *NEXT* x*K* ou y*M* :
   - = Taille du deuxième *extent* ;
   - Minimum = Taille d'un bloc ;
   - Valeur par défaut = 5*taille d'un bloc.
3. *PCTINCREASE* :
   - = Pourcentage d'augmentation de la taille des *extents* alloués après le deuxième *extent*.
   - Taille t_i d'un $extent_i$, $i > 2$: $t_i = (1 + pctincrease/100) * t_{(i-1)}$.
   - t_i arrondi à la taille du plus proche multiple de la taille des blocs.
   - Par défaut, *PCTINCREASE* = 50%.

b) Extents : Paramètres de stockage et gestion par le dictionnaire

4. *MINEXTENTS* : nombre minimum d'*extents* à allouer sur un segment lors de sa création ;
5. *MAXEXTENTS* : nombre maximum d'*extents* allouables à un segment ;
6. *BUFFER_POOL* :
   - Affecter un cache par défaut à un objet autre qu'une *tablespace* ou qu'un segment d'annulation
   - Imposer une stratégie de caching pour l'objet
   - cf. Gestion des caches

Gestion des *extents* : Libération

- Généralement pas de libération physique avant suppression de l'objet contenu
- *Libération logique*, i.e. marquage
  - dans la matrice d'occupation
  - ou dans les tables du dictionnaire
- Forcer la libération des *extents* marqués libres (administrateur) :
  - option *deallocate unused*
  - dans *"alter table"*, *"alter index"* ou *"alter cluster"*.

4. Gestion des Segments Oracle

- Blocs des *extents* : contigus,
- *extents* d'un segment : pas nécessairement contigus
- Segment : alloué
  - à un objet de la base,
  - dans une *tablespace*
  - éventuellement dans des fichiers physiques différents.
- Quatre types importants de segments :
  1. Données (cf. chapitre bases de données)
  2. Index (cf. chapitre bases de données)
  3. Temporaires (ici)
  4. Annulation (cf. chapitre sécurité et reprise)

Oracle : Segments temporaires

- = Espaces disque
- Utilisés lors de l'analyse et de l'exécution
  - de *create index*,
  - de *select distinct* avec/sans *order by* et/ou *group by*
  - de requêtes avec $\cup$, $\cap$, $\setminus$
  - de jointures sans index.
- Utilisables pour des index construits pour des tables temporaires
- Localisation par défaut : *tablespace SYSTEM*
- Segments dans *tablespaces* temporaires : *create temporary tablespace*
- Affectation utilisateur-espace temporaire :
  '*create* ou *alter user* . . . *temporary tablespace* NomEspace"

5. Gestion des caches Oracle

- Structures mémoire d'une instance Oracle incluent :
  1. Zone globale système (SGA, cf. chapitre installation)
  2. Zones globales de programmes (*Program Global Area, PGA*) ou de processus (serveur et arrière-plan)
  3. Zones mémoire partageables pour code exécutable :
     (a) code de l'instance Oracle
     (b) code utilisateur
  4. Zones de tri, etc.

5. Mémoires cache Oracle

- Plan :
  1. SGA
     (a) Taille
     (b) Cache de données
     (c) Cache du *log*
  2. PGA

5.1.1 Mémoires cache : Taille de la SGA

- Déterminée lors du lancement d'une instance Oracle
- Affichée lors du lancement de *Oracle Enterprise Manager*
- Peut être obtenue par *SHOW SGA* (sous *SQL*Plus*) :

```
SQL> show sga;

Total System Global Area   73701404 bytes
Fixed Size                    75804 bytes
Variable Size              56770560 bytes
Database Buffers           16777216 bytes
Redo Buffers                  77824 bytes
```


5.1.1 Mémoires cache : Taille de la SGA

- Pour de bonnes performances :
  - Mémoire vive de la machine du serveur doit être suffisante pour accueillir la totalité de la SGA
  - Sinon risque de $\searrow$ des performances (combinaison gestion mémoire Oracle-gestion mémoire système hôte)
- Quelques paramètres d'initialisation de la base influant sur la taille :

1. *db_block_size* : taille d'un bloc (en octets) ;
2. *db_block_buffers* : nombre de tampons alloués à la SGA ;
3. *log_buffer* : nombre d'octets du tampon *redo log* ;
4. *shared_pool_size* : nombre d'octets des zones partagées SQL et PL/SQL.

5.1.2 Mémoires cache : cache de données

- Taille fonction de :
  - *db_block_size* (généralement 2 ou 4KO) et *db_block_buffers*
- Cache comportant :
  1. *write list* : liste des blocs modifiés en attente d'écriture physique ;
  2. liste *LRU* (*"Least Recently Used"*) :
     - blocs libres,
     - + blocs modifiés non encore transférés vers la *write list*
     - + blocs en cours d'utilisation (*"pinned blocks"*).
- Recherche d'un bloc libre dans le cache :

5.1.2 Cache de données : Remplacement de blocs

- Par défaut, parcours d'une table "en *fetch and discard"* (MRU)
- Contrôle de la stratégie :
  1. clause *"CACHE| NOCACHE"* de *create* ou *alter table* ou *index*
  2. Utilisation de pools de blocs :
     (a) Configurer les pools
     (b) Associer/affecter objets aux pools

5.1.2.a) Configuration de pools de blocs de données

- Pool = 1, n ensembles de buffers
- 3 types de pools : *KEEP*, *REUSE* et *DEFAULT*

1. *KEEP* : Taille spécifiée par *buffer_pool_keep* (paramètre de configuration)
2. *REUSE* : Taille spécifiée par *buffer_pool_reuse*
3. *DEFAULT* :
   - Emplacement, par défaut, des objets
   - Taille = *db_block_buffers* - (*buffer_pool_keep* + *buffer_pool_reuse*)

5.1.2.b) Association pools-objets

- Objets : tables, *clusters*, index, partitions
- Type de pool → Stratégie de remplacement de blocs
- Association pool d'objets-stratégie :
  - Dans {*alter* | *create*} {*table* | *index* | *cluster*}
  - Options *KEEP* et *RECYCLE* de la *clause de stockage*.
- *KEEP* : maintien des blocs en mémoire après utilisation ;
- *RECYCLE* : pas de maintien.

5.1.2 Configuration de caches de données : Exemple

create table R(x *int*, . . .) *storage (buffer_pool recycle)*;
create index Idx2R *on* R(x) *storage (buffer_pool keep)*;

- Gestion des blocs de *R* : *"fetch and discard"* ;
- Gestion des blocs de *Idx2R* : *LRU* ;
- Si dans le fichier de configuration de la base :
  - *db_block_buffers = 2048*
  - *buffer_pool_keep = (buffers: 300, . . .)*
  - *buffer_pool_reuse = 100*,
- Pool *KEEP* : 300 blocs ;
- Pool *REUSE* : 100 ;
- Pool par défaut : (2048 - (300 + 100)).

5.1.3 Mémoires cache du journal (*redo log*)

- Enregistrements du journal : *redo entries*
- Taille du cache du *log* :
  - Par défaut : 4*taille maximum d'une page du système hôte ;
  - ou *log_buffer* octets (paramètre de configuration de la base)
- *LGWR* : écritures physiques dans le fichier ou le groupe de fichiers *redo log actifs* (cf. chapitre sécurité et reprise) ;
- Gestion circulaire des blocs
- Exemple : Cache de 4 blocs

5.2- Mémoires cache : *Program Global Area*

- Aussi appelées zones globales de processus (*process global area*),
- Non partagées,
- Une par processus serveur ou d'arrière-plan,
- Allouées par Oracle lorsqu'un utilisateur se connecte à une base et qu'une session est créée,
- Taille fixe, dépendante du système hôte,
- Contenu varie selon que la session est sous un serveur dédié ou sous un serveur *multi-threaded*.

Organisation et stockage des données et des index : Conclusion

- Généralement transparente pour un utilisateur naïf (*indépendance données-programmes*) ;
- Compréhension nécessaire pour
  - Gérér les espaces des bases ;
  - Comprendre les choix de l'optimiseur ;
  - Configurer le serveur ;
  - . . .
  - Concevoir et implémenter des SGBD.

Chapitre IV : Les bases de données et leurs objets

- Différents types d'espaces de persistance
  - données, index, journal des transactions,
  - sauvegardes (données, journal),
  - sécurité par duplication (*mirroring, multiplexage*).
- Emplacements physiques par défaut exclusivement : non viable
- But du chapitre : définir et gérer
  - les espaces,
  - les objets dans ces espaces.

Bases de données Oracle

- Structures logiques d'une base :
  1. *tablespaces*, blocs, *extents*, segments (déjà vus)
  2. "objets schéma" (*schema objects*).
- Dans ce chapitre :
  1. Essentiellement *"schema objects"*,
  2. Compléments sur les autres structures.
- Terminologie Oracle : *schéma* détermine une collection d'objets d'une base accessibles à un utilisateur,
- *Objet schéma* : structures logiques se référant aux données de la base ;
- *"Schema objects"* : tables, vues, procédures, index, . . .

Bases de données Oracle : Structure de la présentation

1. Types de tables Oracle (survol)

- Outre les tables "classiques" :
  1. Tables objets (ORACLE : SGBD objet-relationnel)
  2. Tables organisées en index ou *index organized* : Feuilles de l'index = Pages de données
  3. Groupement de tables (*Cluster*)
  4. Tables partitionnées

2. Les groupements de tables ou *clusters*

- Création : *create cluster*
- Effet : ranger dans un même bloc des tuples de différentes relations
- Proximité physique de tuples fonction de la *clé du cluster*
- *Clé du cluster* : Colonnes communes aux relations
  - 16 colonnes, au plus,
  - Pas de colonne de type *LONG* ou *RAW*.
- Tuples ayant le même valeur de la clé : Rangement dans le même bloc.
- Accès au *cluster* :
  1. par une fonction de "hachage"
  2. via un index

Clusters et index de *cluster*

- *Impératif* : Créer un index pour le cluster
  - A définir sur la clé du *cluster*,
  - *après* création du *cluster*,
  - *avant* toute insertion dans les tables du *cluster*.
- Tables du *cluster* NON manipulables avant création de l'index ;
- Données inaccessibles en cas de *suppression de l'index* ;
- Re-création ⇒ Données de nouveau accessibles.

Clusters et index de *cluster* : Exemple

create cluster MonCluster (NoEq *number*(2)) *size* 400;

create table Equipe(Eq# *number*(2), Nom *varchar*(30), . . .)
cluster MonCluster(Eq#);

create table Personne(Pers# *number*(2), Nom *varchar*(30), . . . ,
Eq# *number*(2) *not null*) *cluster* MonCluster(Eq#);

create index IdxDuCluster *on cluster* MonCluster;

- Un bloc du *cluster* {*Equipe*, *Personne*} de clé *NoEq* :

3. Oracle : Les index

- *Clé d'un index* : au plus 32 colonnes ;
- Recommandation : créer et instancier les tables avant les index ;
- Option *"on line"* :
  - *"create index . . . on line"* ;
  - *"alter index . . . rebuild on line"* ;
  - Création ou reconstruction sans interdire l'accès à la table ;
  - Commandes de manipulation de données autorisées ;
  - Commandes de définition de données interdites

3. Les index

- Index automatiquement créé sur attributs *UNIQUE* ou *PRIMARY KEY* :
  - ⇒ Allouer un espace suffisant pour les tables ;
  - Suppression des contraintes ⟶ Suppression des index.
- 5 types d'index, Outre ceux créés automatiquement :
  1. "Ordinaires" (p. 114),
  2. Matrices de positions binaires (p. 115),
  3. Basés sur des fonctions (p. 150),
  4. Spécifiques à un domaine d'application (p. 117),
  5. Partitionnés (cf. §4, Partitionnement, p. 118).

3.1- Index "Ordinaires"

- Gérés comme des arbres équilibrés (*B-Tree*).
- *Exemple* :
  - Sur *Commande.Mois* et *Commande.Jour* ;
  - Ordre ↗ sur Mois, ↘ sur Jours ;
  - Collecte de statistiques à la création (cf. §5, Placement de objets, p. 120).

create index Idx_Mois_Jour
on Commande(Mois *asc*, Jour *desc*) *compute statistics*;

3.2- Index *bitmap*

- Pour les transactions de type décisionnel (*OnLine Analytical Processing ou OLAP*) ;
- Liste de positions binaires associée à chaque valeur de la clé de l'index :
  - K_j : valeur de la clé de l'index,
  - L_j : liste de positions binaires associée,
  - $L_j[i] = 1 \longleftrightarrow$ bloc de numéro i contient des tuples de clé K_j.
- cf. *create bitmap index*

3.3- Index sur fonction

- Valeurs des clés de l'index : produites par une fonction pré-définie ou par une fonction définie par l'utilisateur.
- Exemple :
  - *create index* IdxSurFonction *on* Personne(*UPPER*(Nom)) ;
  - Index sur *Personne.Nom* codé en lettres capitales (*UPPER*) ;
- Pas totalement *transparent* : commandes SQL doivent tenir compte de la fonction associée à l'index

3.4- Index de domaine d'application

- Définir ses propres mécanismes de gestion d'index ;
- Par exemple, pour images, documents, vidéo, etc. ;
- Index de domaine défini sur
  - Données d'une table ou
  - Attributs d'un type d'objet (cf. extensions objet).
- $\longrightarrow$ Définir et implanter ses propres fonctions de gestion de l'index (cf. *Oracle Data Cartridge Interface* pour fonctions à implanter)

4. Tables et index partitionnés

- *Partitionnement* : applicable aux index et aux tables,
- Décomposition index ou tables en sous-ensembles (*partitions*),
- Partitions :
  - de nouveau "partionables"
  - physiquement dans des espaces distincts (ou pas).
- Variété des opérations de gestion des partitions :
  - ajout, suppression, modification, fusion, ...
  - dépend du mode de partitionnement ;
  - Non présentées ici.

4. Tables et index partitionnés

- Intérêt des partitions d'objets :
  - Espaces distincts : accès simultanés ;
  - Sauvegarder/Restaurer partiellement un objet partitionné.
- *Clé de partitionnement* : Attribut ou liste d'attributs
- *Méthode de partitionnement* (Indiquée lors de la création de l'objet) :
  1. *range partitioning* : Par intervalles de valeurs
  2. *hash partitioning* : Par fonction de "hachage"
  3. *composite partitioning* : Composition *range* + *hash*

5- Analyse et gestion des espaces

- Un processus de gestion des espaces :
  1. Collecte de statistiques sur les objets ;
  2. Validation de l'intégrité des structures ;
  3. Analyse des informations collectées ;
  4. Correction des insuffisances ou défauts constatés.

5.1- Analyse des espaces

- Calculer/estimer (*compute/estimate statistics*) des statistiques sur les espaces ;
- *Estimation* : sur un échantillon spécifié en nombre de lignes (1064 par défaut) ou en pourcentage de la taille de l'objet.
- Clauses de portée de la commande d'analyse (*analyze*) :
  - *for table* : informations sur la table,
  - *for columns* : une liste de ses colonnes
- Stats sur tables : cf. vues *user_tables, all_tables* et *dba_tables*
- Sur colonnes : {*user_* | *all_* | *dba_*}*tab_columns*
- Sur clusters : (en partie) vues {*all_* | *user_* | *dba_*}*clusters*.
- Exemples :

5.1- Analyse des espaces : (1/3) Stats sur des tables

Nom de colonne	Nature de l'information
NUM_ROWS	Nombre de tuples
BLOCKS	Nombre de blocs de données
EMPTY_BLOCKS	Nombre de blocs qui n'ont jamais été occupés
AVG_SPACE	Taux moyen d'espace libre par bloc
CHAIN_COUNT	Nombre de tuples ayant subi une migration
AVG_ROW_LEN	Taille moyenne d'un tuple
etc.	

5.4.1- Analyse des espaces : (2/3) Stats sur des Index

Nom de colonne	Nature de l'information
BLEVEL	Nombre de niveaux d'index
LEAF_BLOCKS	Nombre de blocs feuilles
DISTINCT_KEYS	Nombre de valeurs distinctes de la clé
*AVG_LEAF_BLOCKS_*	Nombre moyen de blocs feuilles par valeur
PER_KEY	de la clé de l'index
*AVG_DATA_BLOCKS_*	Idem pour les blocs de données
PER_KEY	
CLUSTERING_Factor	Facteur de groupement des clés de l'index
etc.	

5.4.1- Analyse des espaces : (2/3) Stats sur des *Clusters*

- Stats sur un *cluster* incluent des statistiques sur
  1. ses tables,
  2. leurs index,
  3. l'index du *cluster*,
  4. + d'autres données (exemple : nombre moyen de blocs de données par valeur de la clé du *cluster* : *avg_blocks_per_key*)

5.2- Validation des structures

- *But* : Examiner l'intégrité des espaces occupés par une table (persistante, temporaire ou partitionnée), un index ou un *cluster*.
- Commande *"analyze ... validate structure ... "*
- Option *"cascade"* $\longrightarrow$ Valider une structure et celles des objets qui en dépendent.
- Résultats de l'analyse NON utilisés par l'optimiseur ;
- Résultats de *compute/estimate statistics* : utilisés par l'optimiseur ;
- Exemple : Cas structure à valider = table partitionnée :
  - Table *invalid_rows* $\supset$ *RowID* des tuples non rangés dans la "bonne partition" ;

5.3- Analyse des informations collectées

- Pour envisager des actions correctives.
- Outre les vues de données des statistiques, rangement des résultats de certaines analyses dans des tables.
- Exemple :
  - *"analyze table, index ou cluster"* + option *"list chained rows"*
  - Résultat = ID des tuples chaînés et de ceux ayant migré dans les espaces des tables et des *clusters* ;
  - Par défaut, résultat dans la table *chained_rows* ;

5.4- Correction des insuffisances ou défauts constatés

- Actions fonction des résultats de l'analyse
- Exemples :
  - Accroître/Réduire la taille des espaces alloués (cf. *deallocate unused* ou *drop storage*) ;
  - Reconstruire/Compacter (*rebuild*/*coalesce*) un index, etc.

6. Gestion des bases de données

- Création d'une base : maîtrise d'une partie des concepts introduits (ici + chapitres installation et stockage) ;
- Création assistée (*DataBase Configuration Assistant (DBCA)*) ou pas.
- *DBCA* :
  - Création avec paramètres et options par défaut ;
  - ou "personnalisation" de la création.
- Création non assistée : complexe ;
- S'appuyer sur *DBCA* pour les premières expériences ;
- Tenter la création personnalisée plus tard.

6. Gestion des bases de données

- La suite :
  1. Rappel : Constituants d'une base (p. 130)
  2. Processus de création d'une base (p. 135)
  3. Mise en exploitation (*montage, ouverture*) (p. 140)
  4. Suppression d'une base (p. 144)

6.1- Constituants d'une base de données Oracle

1. un *fichier d'initialisation* (*init file*),
2. un ou plusieurs fichiers de contrôle (*control file*),
3. divers espaces physiques (données, journal, index, etc.),
4. Espaces dupliqués, dictionnaire, etc.

6.1.1- Constituants d'une base : *Init file*

- Nom *init.ora* ou *init*NomDeLaBase.*ora* ;
- Contenu :
  - Nom de la base,
  - Paramètres de configuration, comme :
    - Taille des blocs de données (*db_block_size*),
    - Taille de certains pools,
    - Emplacements des fichiers de contrôle, etc.
- Modèle de fichier fourni avec le système,
- Modifiable par un administrateur et par Oracle,
- Consulté lors de l'activation de l'instance Oracle associée à la base.

6.1.1- Constituants d'une base : *Init file*, à partir de Oracle 9i

- Extension du nombre de paramètres d'initialisation *dynamiques*
- *spfile* : fichier d'initialisation binaire persistant
  - créé par :
    create spfile = 'Chemin . . . *spfile*Nom.*ora*'
    from 'Chemin vers le fichier texte *init.ora*';
  - en tant que *sys*/Mot de passe *as sysdba* ;
  - localisé par *show parameters spfile* ;
  - création version *texte* : *create pfile* = '. . . ' *from spfile* = '. . . '

6.1.2- Constituants d'une base : *Control file*

- Fichier binaire contenant des informations sur :
  - Nom, date de création de la base,
  - Noms et emplacements de ses fichiers (données et *redo log* en ligne),
  - Espaces de données (*tablespace*),
  - Fichiers *redo log* archivés, Fichiers de sauvegarde, etc.
- Paramètre *control_files* dans *init file* : Emplacement des *control files* ;
- Gestion par le système : mise à jour après opérations du type ajout, suppression ou renommage de fichiers de données ou de *log*.
- Recommandation : Plusieurs copies par base,
  - de préférence sur des unités physiques différentes.

6.1.3- Constituants d'une base : *Espaces physiques*

- Pour maîtriser gestion et placement des objets, créer des espaces
  1. Temporaires,
  2. [D'annulation (*Rollback segments, Undo tablespaces*),]
  3. Données,
  4. Index, etc.

6.2- Processus de création d'une base de données Oracle

- *DBCA* : Type de la base en fonction des types de traitements :
  1. Base pour transactions ou programmes "ordinaires" (*OLTP*),
  2. Base de type entrepôt de données (*Warehouse*) (applications de type décisionnel : *On Line Analytical Processing* ou *Decision Support System*),
  3. Mélange des deux types précédents (*Multipurpose*).
- *DBCA* : Choix de la taille (petite, moyenne, grande) de la base.

6.2- Création assistée d'une base de données

1. Lancer DBCA
2. Personnalisation éventuelle (option *Custom* de *DBCA*),
3. Conserver le script engendré (xxx.*bat*)
4. (Examiner le script pour voir la complexité du processus)

6.2- Script de création : Extrait du fichier *init.ora*

```
# Copyright (c) 1991, 2000 by Oracle Corporation
###################################################
# Example INIT.ORA file
# This file  is provided by  Oracle...
###################################################

db_name = "HandMade"
instance_name = HandMade
service_names = HandMade
db_files = 1024
control_files =
     ("F:\ORACLE\oradata\HandMade\control01.ctl",
      "F:\ORACLE\oradata\HandMade\control02.ctl",
      "F:\ORACLE\oradata\HandMade\control03.ctl")
```



```
open_cursors = 300
max_enabled_roles = 30
db_file_multiblock_read_count = 8
db_block_buffers = 4503
shared_pool_size = 6328422
large_pool_size = 11796480
log_checkpoint_interval = 1000
log_checkpoint_timeout = 180
processes = 150
parallel_max_servers = 5
log_buffer = 32768

#audit_trail = true  # if you want auditing
max_dump_file_size = 10240 # limit trace file size to 5M each
```

```
##### For archiving if archiving is enabled #####
log_archive_start = true
log_archive_dest_1 = "location=E:\HandMade\archive"
log_archive_format = %%ORACLE_SID%%T%TS%S.ARC
# If using private rollback segments, place lines of the
# following form in each of your instance-specific
# init.ora files:
#rollback_segments = ( RBS0, RBS1, RBS2 )
...
# define directories to store trace and alert files
background_dump_dest = F:\ORACLE\admin\HandMade\bdump
user_dump_dest = F:\ORACLE\admin\HandMade\udump
db_block_size = 4096
```


6.3- Mise en exploitation d'une base

- Base utilisable si *montée* et *ouverte* ;
- Au démarrage (*startup*) :
  1. Création d'une instance de la base
  2. Choix du mode de démarrage.
- Mode base "montée" :
  - Associer base-instance Oracle,
  - Mais base NON disponible à des utilisateurs,
- Mode base "montée + ouverte" : disponible pour les utilisateurs autorisés.

6.3- Monter et ouvrir une base

- Arrêt : *shutdown*.
- Opérations réalisables :
  1. sous *sqlplus*,
  2. sous le gérant de reprise (*Recovery Manager*),
  3. dans l'environnement *Oracle Enterprise Manager*.

6.3- Monter/ouvrir une base sous *sqlplus*

1. Connexion : *"sqlplus; connect* Nom/MotDePasse *as sysdba"* ;
2. Quelques possibilités :
   (a) *startup nomount* : Démarrage d'une instance sans montage de base ;
   (b) *startup mount* : Démarrage d'une instance avec montage d'une base ;
   (c) *startup* : Démarrage d'une instance avec montage et ouverture ;
   (d) *startup restrict* : Démarrage avec montage et ouverture en limitant l'accès.

- Exemples :

6.3- Monter/ouvrir une base

- Démarrage avec "montage seul" : Pour des tâches d'administration
  - Renommage d'espaces (*datafiles* et *redo log files*),
  - Ajout ou suppression d'espaces,
  - Reprise "à froid" de la base, etc.
- *startup . . . restrict* : accès limité à
  - Administrateurs,
  - Utilisateurs ayant conjointement les droits *create session* et *restricted session*.

6.4- Supprimer une base

- Suppression manuelle ou avec *DBCA*.
- Supprimer tous ses fichiers (données, journaux, contrôle, initialisation, archives, etc.).
- Noms des fichiers : voir *v$datafile* et *v$logfile* ;

Bases de données et leurs objets : Conclusion

- Gestion *naïve* des bases de données : abstraction de l'organisation et de la gestion physiques des données ;
- $\neq$ Pour sécurité de fonctionnement et performances ;
- *Sécurité* : Dissocier les unités physiques de stockage (données, journal, unités dupliquées, ...) ;
- Performances : Gain lors de lectures/écritures physiques par
  - Dissociation des unités de stockage (tables, index) ;
  - Fragmentation de tables volumineuses sur disques différents ;
  - Multiplication des mémoires-cache, etc. ;
  - cf. Chapitre Traitement des requêtes et chapitre *Tuning*.

Chapitre V : Gestion des utilisateurs

- *Sous-système d'autorisation* : protéger une base contre des accès non autorisés
- Principales techniques :
  - Numéro de compte/mot de passe
  - Allocation/révocation de droits
  - Limitation des ressources
- Plan :
  1. Rappels : Utilisateurs, privilèges et rôles (p. 147)
  2. Application à Oracle (p. 150)
  3. Conclusion (p. 167)

1. Utilisateurs, privilèges et rôles (Rappels)

- Utilisateur : nécessairement identifié par le SGBD
- Administrateur : utilisateur/compte privilégié
- Privilèges : Type d'actions autorisées
  - Niveau compte
  - Niveau objets (relations, procédures, index, etc.)
- Privilèges :
  - Liste de privilèges ou nom de liste (*rôle*)
  - Parfois transmissibles (*grant ... with* {*grant* | *admin*} *option*)

Privilèges (Rappels)

1. Niveau compte :
   - Droits de créer/supprimer/modifier des définitions d'objets
   - Allocation par
     - Administrateur,
     - Propriétaire de la base,
     - Utilisateur ayant les "bons" droits.
2. Niveau objet :
   - Type des commandes sur un objet
   - Allocation par le propriétaire de l'objet.

- *Propriétaire* : Compte/Utilisateur ayant créé l'objet

Rôles (Rappels)

- Rôle = Ensemble nommé de privilèges
- Allocation à un utilisateur et/ou à un autre rôle
- "Cycle de vie" :
  1. Création (*create role*)
  2. Allocation de privilèges (*grant*)
  3. Allocation/révocation du rôle (*grant/revoke role*)
  4. ...
  5. Suppression (*drop role*)
- Voir exemple plus loin.

2. Oracle : Gestion des utilisateurs

- Versions < 7 : gestion des utilisateurs et des privilèges : *grant* et *revoke* ;
- Versions ≥ 7 :
  - Notion de rôle ;
  - Commandes explicites de gestion des utilisateurs.
- La suite :
  1. Gestion des comptes et des utilisateurs ;
  2. Ressources et profils ;
  3. Gestion des rôles ;
  4. Gestion des privilèges.

2.1. Gestion des comptes et des utilisateurs

- Six types d'utilisateurs Oracle :
  1. Administrateurs (rôle *DBA*),
  2. Responsables sécurité (rôle *OPER*),
  3. Développeurs d'applications,
  4. Administrateurs d'applications,
  5. Utilisateurs d'une base,
  6. Administrateurs réseau.

2.1.1- Les administrateurs

- AU moins deux comptes, par défaut, créés lors de la création d'une base : *sys* et *system*
- *dba* : rôle prédéfini
  - Créé lors de la création d'une base ;
  - Octroyable à d'autres utilisateurs par *sys* ou *system*.
- Authentification des utilisateurs bénéficiant de ces rôles :
  - Fichier de mots de passe ou
  - Système d'exploitation.
- Exemple :

2.1.2- Les utilisateurs d'une base

- *licence_max_sessions* (*init.ora*) : Nombre maximum de sessions simultanées
- *licence_max_users* : Nombre maximum d'utilisateurs qu'on peut créer dans une base ;
- Création utilisateur :
  - dans la base,
  - droit administrateur ou droit *create user*,
  - Nom, mot de passe
- + Autorisation de connexion : privilège *"create session"*
- Modification des caractéristiques d'un utilisateur : *"alter user"*

2.1.2- Les utilisateurs d'une base

- Espace par défaut et espace temporaire : *system*
- Affectation d'espace par défaut :
  - *create user* ou *alter user* ;
  - *default tablespace* ;
  - *temporary tablespace*
- Suppression : *drop user*
- *drop user... "cascade"* : Suppression de l'utilisateur et de ses objets
- Exemple :

2.2. Limitations de ressources et notion de profil

- *Profil* : ensemble nommé de limites de ressources,
- Exemples de ressources :
  - Nombre de sessions (*sessions_per_user*),
  - Temps unité centrale (*cpu_per_session*, *cpu_per_call*),
  - Temps de connexion (*connect_time*) par session,
  - Nombre de lectures logiques (*logical_reads_per_session* ou *per_call*).
- Autres éléments possibles d'un *profil* :
  - durée de vie d'un mot de passe (*password_life_time*)
  - Nombre de tentatives de connexion infructueuses au delà duquel le compte est verrouillé (*failed_login_attempts*).

2.2. Limitations de ressources et notion de profil

- Permettre la limitation de ressources :
  - *resource_limit = true* (*init.ora*)
  - ou pendant la session :
    alter system set resource_limit = true
- *Profil par défaut* :
  - Dans chaque base,
  - Ressources illimitées (*unlimited*),
  - Hérité par tout utilisateur sans profil explicite.

  $\Longrightarrow$ définir des profils avec les "bonnes limites".

2.2. Limitations de ressources et notion de profil

- Opérations sur les profils :
  - Créer, modifier, supprimer : *create, alter, drop profile* ;
  - *Droits* : administrateur ou *create, alter, drop profile*.
- Affectation de profils :
  - {*create, alter*} *user . . . profile. . .*
  - Limites du profil remplacent celles du profil par défaut ;
  - Limites non spécifiées par le profil gardent leurs valeurs par défaut!
- Suppression profil $\longrightarrow$ Profil par défaut, à la prochaine connexion.
- Exemple :

2.3. Gestion des rôles

- Rôles prédéfinis : *sysdba* (mais *connect, dba, resource* persistent pour la compatibilité avec les versions $<$ V8)
- Autres rôles prédéfinis installables (Scripts fournis) :
  - Rôles *exp_full_database* et *imp_full_database* : Import/Export données
  - Script *catexp.sql*
- Création possible de rôles (*create role*) avec/sans mot de passe
- Affectation à des rôles et/ou à des utilisateurs (*grant role*)
- Exemple :

2.3. Gestion des rôles

- Affectation rôle par défaut : *alter user . . . default role . . .*
- A la connexion, privilèges = privilèges octroyés $\cup$ ceux du rôle par défaut
- Bénéficier des privilèges d'un rôle : *set role*
- Nombre maximum de rôles endossables pendant une session $\leq$ *max_enabled_roles* (*init.ora*).

2.4. Gestion des privilèges sous Oracle

- Types de privilèges :
  1. Objets
  2. Système

2.4.1- Les privilèges sur des objets

- Qui : propriétaire objet
- Quoi :
  - *select*, *update, delete, insert* : vues ou relations,
  - Utiliser une relation dans une contrainte (*references*),
  - Exécuter (*execute*) des procédures, des fonctions ou des paquetages.
  - privilège *update* sur certaines colonnes.
- Droit de propager des privilèges acquis : *grant . . . with admin option*
- Révocation en cascade : *revoke . . . cascade constraints*.

2.4.1- Privilèges sur des objets : Exemple

grant all on Produit *to* Toi *with admin option*
/* Toi peut attribuer tous les privilèges à d'autres utilisateurs */
grant select on Produit *to PUBLIC*
/* Tout utilisateur peut consulter la relation Produit */
grant select, update(qte) *on* Stock *to* Lui
/* Seul l'attribut Stock.qte est modifiable par Lui */
revoke all on Produit *from PUBLIC*
/* Retrait de tous les privilèges sur Produit */
revoke update on Stock *from* Lui
/* Lui ne peut plus mettre à jour Produit */

2.4.2- Privilèges système

- Une centaine ! (cf. Manuel de référence)
- Créer/supprimer/modifier objets système ou base de données (session, user, profils, rôles, index, tables, vues, triggers, etc.)
- Agir sur :
  - Base : *alter database*
  - Système : *alter system*
  - *Audit* : *audit system*
- Egalement transmissibles.

2.4.2- Privilèges système : Exemple

grant create role to LUI *with admin option*;
grant drop any role to LUI *with admin option*;
grant create table to public;
grant grant any privileges to Elle;
grant audit system to Paul;

2.4. Gestion des utilisateurs : Eléments du dictionnaire

- *all_users*, *user_users*, *dba_users*,
- *user_ts_quotas*, *dba_ts_quotas*,
- *user_password_limits*, *user_resource_limits*,
- *dba_profiles*,
- *v$session*, *v$sesstat*, *session_roles*,
- *session_privs*,
- *all_tab_privs_recd*, *user_tab_privs_recd* (privilèges reçus),
- *all* et *user_tab_privs_made* (privilèges octroyés),
- *all* et *user_col_privs_recd* (ou *made*),
- *user* et *dba_role_privs*, ...

2.4. Gestion des utilisateurs : Eléments du dictionnaire

- *"select * from session_privs"* : Privilèges en vigueur pour la session courante ;
- *select table_name, grantee, grantor, privilege from user_tab_privs_made*
  - noms des objets (*table_name*),
  - nom du bénéficiaire (*grantee*),
  - désignation du privilège (*privilege*)
  - nom de l'utilisateur qui l'a octroyé (*grantor*).

3. Gestion des utilisateurs : Conclusion

- Numéros de compte, mots de passe,
- Privilèges, rôles,
- Rôles prédéfinis ou pas,
- Ressources, profils,
- Sans oublier aussi les vues.

Chapitre VI : Sécurité de fonctionnement et reprise

1. Sécurité et reprise : Généralités et rappels

- Sécurité : Serveur(s) et Bases
- Compléments : Mécanismes/procédures sécurité réseau/système
- Mécanismes :
  1. Duplication (*Multiplexage, Mirroring*),
  2. Sauvegardes et Reprises en cas d'incident,
  3. Surveillance (*audit*)

1.1. Sécurité par réplication (*Multiplexage, Mirroring*)

- Synchroniser le contenu d'un espace avec un réplicat,
- Modifications "en double",
- Répliquer au moins le *log* des bases de données "sensibles".
- !!! Espace et *"Overhead"* induits.

1.2. Sauvegardes et reprises

- *Transactions* : rappels
  - A.C.I.D.
  - Sérialisabilité
  - *Log, Checkpoints*
  - *Two Phase Locking Protocol* (et interblocage)
  - *Write Ahead Log Protocol*
  - *Commit/Rollback*
- Reprise automatique ("à chaud")
- Reprise "à froid"
  - Sauvegardes (données, *Log*s)
  - Restorations et reprises (*Roll forward*)

1.3. Surveillance (*Audit*)

- Processus :
  1. Installer/Activer le mécanisme de surveillance
  2. Spécifier les "événements" à surveiller
  3. Analyser les observations.
- !!! *"Overhead"* induit.

2. Sécurité : Application à Oracle

2.1. Multiplexage sous Oracle

- Duplication = *multiplexage*
- Ecritures par le processus d'arrière-plan *LGWR*
- Pas de continuité de service : erreur d'entrée-sortie sur une des copies peut nécessiter une relance de l'instance Oracle concernée,
- Mécanisme à combiner avec services de mise en miroir (*"mirroring"*) d'unités ou de fichiers si disponibles sur le système d'exploitation hôte
- Multiplexer (au moins) *control file* et *redo logs*
- *Groupe* et *membre* :
  - Membre = fichier d'un groupe
  - Membres d'un même groupe = copies les uns des autres

2.1. Multiplexage sous Oracle

- Recommandations :
  - Même nombre de membres dans les groupes
  - Même taille pour les membres d'un groupe
- Minimum deux *groupes* d'au moins un *membre* par base
- *Redo log* :
  - Nombre maximum de groupes : *maxlogfiles* (*create database*)
  - Nombre maximum de membres : *maxlogmembers*
- Ajout/suppression de groupes et de membres pendant la durée de vie de la base (*alter database . . . add/drop logfile group/member*)

2.1. Multiplexage sous Oracle : Exemple

create database MaBase
control file reuse
logfile group 1 ('/u1/log11.log', '/u2/log12.log') *size* 80K,
group 2 ('/u1/log21.log', '/u2/log22.log') *size* 80K,
maxlogfiles 6 . . .
alter database MaBase
add logfile group 3 ('/u1/log31.log', '/u2/log32.log') *size* 80K;
alter database MaBase
add logfile member '/u3/log33.log' *to group* 3;

Multiplexage sous Oracle

- Ecriture dans les membres d'un même groupe
- Si saturation des membres d'un groupe :
  - Archivage
  - Journalisation dans les membres d'un autre groupe.
- Si échec d'une écriture dans un membre d'un groupe :
  - Marquage du membre
  - Message d'erreur dans le fichier de trace du *LGWR* et dans celui de la base,
  - Poursuite des écritures dans les autres membres.

2.2. Transactions, sauvegardes et reprises

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

2.2. Transactions et reprises sous Oracle : Plan

2.2.1- Journaux ou *redo logs*

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

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

2.2.1.a- Archivage automatique des *redo logs*

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

2.2.1.a- Activation/Désactivation archivage

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

2.2.1.a- Principe de l'archivage

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

2.2.1.a- Principe de l'archivage

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

2.2.1.b- Gestion des redo logs archivés

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

2.2.2- Gestion des *rollback segments*

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

2.2.2- *Rollback segments* et transactions

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

Rollback segments et transactions

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

Types de *Rollback segments*

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

Types de *Rollback segments*

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

Rollback segments vs *Undo tablespaces*

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

Undo tablespaces : Exemple

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

Rollback segments vs *Undo tablespaces*

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

2.2.3- Validation et annulation de transactions

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

Validation de transactions : Les effets

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

2.2.3- Cas Echec des transactions

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

2.2.4.a- Isolation, verrouillage et points de contrôle

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

2.2.4.b- Isolation, Verrouillage et points de contrôle

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

2.2.4.b- Isolation, Verrouillage et points de contrôle

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

2.2.4.c- Isolation, Verrouillage et Points de contrôle

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

2.2.4.c- Isolation, Verrouillage et Points de contrôle

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

2.2.5- Les sauvegardes

- Avec *Recovery Manager (rman)* ou au niveau système hôte

1. Sauvegarde totale :
   - Tous les fichiers (data et control file)
   - Sur base ouverte : risque d'incohérence
   - Sur instance arrêtée "proprement" (pas *shutdown abort*)
   - Cas base en mode *archivelog* : sauvegarde totale = seule moyen de se prémunir contre une détérioration physique des supports.

2.2.5- Les sauvegardes

2. Sauvegardes partielles :
   - *tablespace(s), datafile(s), control file*
   - Par précaution, quand base en mode *archivelog* car en cas de problème, une reprise permettrait d'obtenir des états cohérents des objets sauvegardés.
- La suite :
  1. Sauvegardes niveau système d'exploitation
  2. Utilisation de *rman*

2.2.5.a- Sauvegardes niveau système d'exploitation

- *v$tablespace* : *tablespaces* existantes
- *v$datafile* : *datafiles* existants
- *v$logfile* : *log files*
- *v$controlfile* ou *init.ora* : fichiers de contrôle

1. Sauvegarde totale :
   (a)
   (b)
   (c)

2.2.5.a- Sauvegardes niveau système d'exploitation

2. Sauvegardes partielles :
   (a)
   (b)
   (c)
   ou
   (a)
   (b)
   (c)

2.2.5.a- Sauvegardes niveau système d'exploitation

3. Cas du control file : *"alter database backup controlfile to* 'destination' *reuse"*

- **Note :** Vérification
  1. *dbverify* ou *dbv*
  2. ou Instancier une base d'essai et y faire une reprise

2.2.5.b- Sauvegardes avec *rman*

- Mode ligne de commande ou graphique
- A besoin d'une base "cible" (*target*)
- Utilise un catalogue ou le fichier de contrôle (option *nocatalog*)
- Utilise une notion de canal (*channel*)
- Des exemples :

2.5.2.c- Autres possibilités de sauvegarde des *redo logs*

- Tous ceux archivés : *backup archivelog all*
- Un sous-ensemble défini par un intervalle :
  1. de temps : *"backup archivelog from time = . . . "* [*"until time = . . . "*] ou *"backup archivelog until time = . . . "*,
  2. de numéros de séquence : *"backup archivelog from logseq = . . . "*, *"backup archivelog until logseq = . . . "* ou *"backup archivelog from logseq = . . . until logseq = . . . "*,
  3. ou de numéros de modification (*from SCN =*, *until SCN =* ou *from SCN = . . . until SCN = . . .*).

2.2.6- La restauration à partir de sauvegardes

- *rman* ou commandes du système hôte
- Dans l'emplacement d'origine ou pas
- Tout "objet" de stockage
- Cas tablespace : tous ses datafiles ou pas
- La suite : Niveau SE (pour *rman* cf. § reprise)
  1. Cas des datafiles
  2. Cas des archives de journaux

2.2.6.a- Restaurer des datafiles

- Sous SE : commande de copie (*cp*)
- Si restauration dans un nouvel emplacement : créer d'abord un datafile vide (*alter database create datafile. . .*)
- Exercice : Comment restaurer un datafile sans en avoir une sauvegarde ?

2.2.6.b- Restaurer des archives de journaux

1. Emplacement d'origine : cp (cf. *log_archive_dest*)
2. Nouvel emplacement :
   (a) cp dans le nouvel emplacement,
   (b) Indiquer l'emplacement lors d'une reprise.
      (1) *cp* /disk1/LogBackup/*.arc /disk2/tmp/Archives/.
      (2) *set logsource* /disk2/tmp/Archives; *recover;*
      (3) *alter database recover from* /disk2/tmp/Archives *database;*

- *v$log_history* : liste de logs archivés
- *v$recover_log* : logs nécessaires pour une reprise.

2.2.7- La reprise en cas d'incident

1. *Rollforward* des modifications journlisées + journalisation
2. *Rollbackward* : Annuler transaction non validées

2.2.7- La reprise en cas d'incident

1. Reprise automatique :
   - *shutdown abort*, panne système hôte, etc.
   - utilisation des *redo logs* actifs
2. Reprise sur support (*media recovery*) :
   (a) *recover* sous SqlPlus
   (b) *alter database recover*
   (c) ou *rman*

2.2.7- La reprise en cas d'incident : Plan

2.2.7.a- Objets d'une reprise

- *recover database* : reprise, après restauration, de tous les fichiers en ligne qui le nécessitent. La base doit être montée en mode *exclusive* et non ouverte.
- *recover tablespace* : reprise sur tous les fichiers de données (*datafiles*) de la ou des *tablespaces* spécifiés comme arguments de la commande. La ou les *tablespaces* concernées doivent être *hors ligne*.
- *recover datafile* : reprise sur un ou plusieurs fichiers. Si la base est *ouverte*, seuls les *fichiers hors ligne* sont pris en compte.

2.2.7.b- Quels journaux pour une reprise ?

- En donner la liste ou se laisser guider par Oracle
- Exemple 1 : Oracle choisit les *redo logs*
- Exemple 2 : idem exemple 1
- Exemple 3 : Application manuelle de *redo logs*
   1) *alter tablespace* users *offline*;
   2) *alter database recover tablespace* users;
   3) //– Oracle suggère un nom de fichier *log* (f_i.log, par exemple).
   4) *alter database recover logfile* "f_i.log";
   5) *alter tablespace* users *online*;

2.2.7.c- Reprise sur une base fermée

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

2.2.7.d- Reprise sur une base ouverte

- Reprise sur des fichiers de données, base en exploitation
  1. Mettre hors ligne les espaces de stockage concernés (*alter tablespace . . . offline temporary*).
  2. Restaurer les seuls fichiers endommagés (pas les journaux ni le fichier de contrôle).
  3. Reprise sur les espaces de stockage (*recover tablespace*) contenant des fichiers endommagés
  4. Remettre en ligne les espaces de stockage.

2.2.7.e- Remise en exploitation après reprise

- Cas reprise partielle ou reprise avec une sauvegarde du fichier de contrôle (*recover database . . . using backup controlfile*)
- Les journaux archivés ne sont généralement plus valides
- Nécessité de réinitialiser ceux en ligne
- *resetlogs* $\longrightarrow$ Numéro de séquence = 1.

  alter database open resetlogs (ou *noresetlogs*)

2.2.7.f- Restauration et reprise avec *rman*

- Base ouverte :

```
run { allocate channel C1 type disk;
      sql "alter tablespace TabSp1 offline immediate";
      restore tablespace TabSp1;
      recover tablespace TabSp1;
      sql "alter tablespace TabSp1 online"; }
```


2.2.7.f- Restauration et reprise avec *rman*

- Avec un nouvel emplacement :

```
run { allocate channel C1 type disk;
      allocate channel C2 type disk;
      sql "alter tablespace TabSp2 offline immediate";
      set newname for datafile 'Lancien' to 'LeNouveau';
      restore tablespace TabSp2;
      switch datafile all;
      recover tablespace TabSp2;
      sql "alter tablespace TabSp2 online"; }
```


2.3. La surveillance (*audit*)

- Quoi = Options de l'audit
- Résultats où : *Audit trail* (table *sys.aud$*)
- Comment :
  1. Activer l'auditing : paramètre statique (*audit_trail = true*)
  2. Spécifier les options : instruction *audit* (arrêt : *noaudit*)
  3. Examiner périodiquement l'*audit trail*
- *Audit* par session (*per session*)/par accès (*by access*)
- *Audit* si succès/si échec (*whenever* [*not*] *successful*)

2.3.1- La surveillance : les options

1. Sur instructions SQL
2. Sur commandes liées aux privilèges système
3. Sur types d'actions sur des objets

- Exemples :

2.3.2- La surveillance : Gestion de l'*audit trail*

- *sys.aud$* créée par *catalog.sql*
- Contrôle du remplissage :
  - Archivage (select into) et vidage périodique (truncate)
  - ou "purger" les enregistrements inutiles

2.3.3- La surveillance : Vues et tables du dictionnaire

- *stmt_audit_option_map* : associe un code à un type d'option ; par exemple, 3 : *"alter system"*, 65 : *"create table"*, etc.
- *audit_actions* : code des actions (par exemple, 1 pour *"create table"*, 2 pour *"insert"*, etc.) ;
- *all_def_audit_opts* : options d'*audit* par défaut ;
- *dba_stmt_audit_opts* : options d'*audit* pour le système et par utilisateur ;
- *dba_priv_audit_opts* : options sur les privilèges système ;
- *dba_* et *user_obj_audit_opts* : options sur tous les objets (vue *dba*) et sur ceux possédés par l'utilisateur courant (vue *user*)
- *dba* et *user_audit_trail* : liste des entrées de l'*audit trail* ;

2.3.3- La surveillance : Vues et tables du dictionnaire

- *dba* et *user_audit_object* : contient les enregistrements de surveillance de tous les objets (vue *dba*) ou de ceux accessibles à l'utilisateur (vue *user*) ;
- *dba* et *user_audit_session* : idem pour les connexions et déconnexions ;
- *dba* et *user_audit_statement* : idem pour les instructions *grant, revoke, audit, noaudit* et *alter system* concernant toute la base (vue *dba*) ou un utilisateur (vue *user*) ;
- *dba_audit_exists* : entrées de l'*audit trail* produites par *"audit exists"* et *"audit not exists"*.

Sécurité de fonctionnement : conclusion

- Journalisation, Sauvegarde, Chargement
- Autres mécanismes :
  - Bases : Duplication (*miroir*)
  - Serveurs : *Audit*, *Back-up*
- Tâches pas uniquement "techniques"
- Organisation, discipline de travail

Chapitre VII : Traitement des requêtes

- Méthodes :
  1. Syntaxique
  2. Par estimation de coûts
  3. Sémantique
- Implémentations : Combinaison des 2 premières
- Rappels détaillés : cf. Annexes

Traitement des requêtes sous Oracle

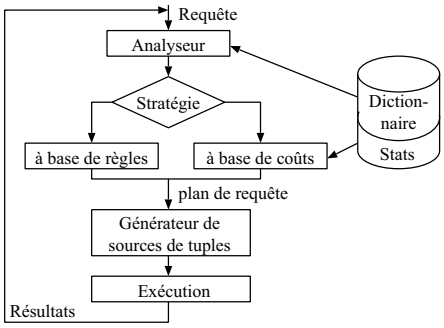
- Etapes “classiques” dans le processus de traitement des requêtes :
  1. Analyse,
  2. Optimisation,
  3. Génération de plans d’exécution,
  4. Exécution.

Processus de Traitement des requêtes sous Oracle

- Deux stratégies d’optimisation :
  1. A base de règles (*Rule Based Optimization*, *RBO*) : $\simeq$ optimisation d’arbres algébriques
  2. A base de coûts : *Cost Based Optimization* (*CBO*).
- Choix de la stratégie : automatique ou imposé pour une requête, une session ou une instance Oracle,
- Influencer la génération de plans : Directives d’optimisation dans les requêtes,
- Ré-utiliser des plans : cf. *outlines* (non traité ici).

Traitement des requêtes sous Oracle : Plan

1/5. Architecture de l’optimiseur et plans d’exécution



1/5. Architecture de l’optimiseur et plans d’exécution

1. *Analyseur* : décomposition de la requêtes en un ensemble de composants emboîtés ou reliés (*blocs de la requête*)
   - Vue, requête imbriquée $\longrightarrow$ blocs distincts de celui de la requête (ou pas)
2. *Générateur de sources de tuples* : Engendrer un plan d’exécution étant donné le plan optimal ($\simeq$ arbre algébrique) rendu par l’optimiseur :
   - Noeuds opérations “internes” à Oracle ($\neq$ opérateurs algébriques) :
     - Provenance = mode d’accès aux tuples et type d’algorithme de jointure choisis
     - Exemples : tri (*sort*), fusion (*merge*)
   - Source de tuples $\simeq$ feuille d’un arbre ou relation résultant de l’évaluation d’un sous-arbre.

1/5. Exemple de plan d’exécution

- Hypothèse : Aucun index
- Requête :

```
explain plan for (
select  p.libelle, s.qte from produit p, stock s
where  p.prod# = s.prod#);
```

- Schématisation du plan d’exécution :

1/5. Plans d'exécution

- Demande de génération : *explain plan . . . for* instruction SQL
- Plan engendré :
  - par défaut, dans *plan_table*
  - ou dans une table de même schéma (*explain plan . . . into . . .*)
- *plan_table* créée par le script *utlxplan.sql* (généralement sous *$OraHome\rdbms\admin*).
- Examen du plan :
  1. *select . . . from plan_table where . . .*
  2. utilitaire *utlxpls* : traitements en série
  3. utilitaire *utlxplp* : exécutions parallèles

1/5. Plans d'exécution : Des colonnes de *plan_table*

- *id :* numéro d'étape du plan ;
- *parent_id :* numéro de l'étape qui utilise les résultats de l'étape *id* ;
- *position :* rang d'une étape parmi les étapes "filles" d'une même étape "mère" ;
- *cardinality :* nombre estimé de tuples accédés par l'opération ;
- *operation :* nom de l'opération interne réalisée dans l'étape ;
- *object_name :* objet concerné par l'opération ;
- *cost :* coût estimé de l'opération (si optimisation basée sur les coûts) ;
- *options :* "variante" d'opération utilisée pour exécuter *operation*.

1/5. Plans d'exécution : Exemple

```
Id P_id    Operation       Objet    Options
-- ---- ----------------- -------- ---------
 0       SELECT STATEMENT
 1    0  MERGE JOIN
 2    1  SORT                        JOIN
 3    2  TABLE ACCESS       STOCK    FULL
 4    1  SORT                        JOIN
 5    4  TABLE ACCESS       PRODUIT  FULL
6 ligne(s) sélectionnée(s).
```

- cf. figure exemple p. 234
- Obtenu par :

1/5. Plans d'exécution

- *plan_table* : représentation tabulaire des plans d'exécution et des choix de l'optimiseur :
  - mode de parcours des objets
  - type d'algorithme de jointure, etc.
- Choix explicités dans *plan_table.operation* et *plan_table.options*

Opération	**Options**	**Commentaires**
table access	*full, by rowid, hash, . . .*	Type d'accès à une table.
sort	*unique*	Tri en éliminant les doublons.
	join	Tri avant jointure par fusion.
	order by	Tri requis par la requête.
nested loops		Jointure par boucles imbriquées.
	outer	Idem pour une jointure externe.
merge join		Jointure par tri-fusion.
	outer	Idem pour une jointure externe.
	semi	Idem pour une semi-jointure.
view		Interrogation d'une vue.

1/5. Plans d'exécution : autres outils Oracle

1. *analyze . . . compute | estimate statistics* : Collecte de statistiques (cf. chapitre Bases de données, p. 120)
2. *SQL Trace* :
   - Autoriser son utilisation : *sql_trace* (paramètre statique d'initialisation)
   - Effets : Génération de statistiques sur
     - Durées d'analyse et d'exécution,
     - Temps total et temps unité centrale,
     - Nombre de lectures et écritures physiques,
     - Nombre de tuples traités, etc.

1/5. Plans d'exécution : autres outils Oracle

- Résultats de *SQL Trace* : dans des *fichiers de trace*
  - Localisation par défaut : *user_dump_dest* (*init.ora*)
  - Indication dynamique de leur emplacement : *alter system set user_dump_dest* = chemin vers un répertoire)
  - *max_dump_file_size* : taille maximum des fichiers (paramètre dynamique)

3. *tkprof* : formattage des fichiers produits par *SQL Trace*
   - Production d'un fichier par fichier de trace,
   - Fichier formatté en fonction de paramètres et options d'exécution

2/5. Les étapes de l'optimisation

1. Évaluation d'expressions de constantes et simplification d'expressions logiques ;
2. Transformation éventuelle de la requête ;
3. Intégration éventuelle des définitions de vues ;
4. Choix de la stratégie d'optimisation (à base de coûts ou de règles) et de l'objectif de l'optimisation (globale ou temps de réponse) ;
5. Choix du mode d'accès et de parcours de chaque table ;
6. Choix de l'ordre des jointures ;
7. Choix de l'algorithme de jointure pour chaque couple de relations.

2.1/5- Simplification et évaluation d'expressions

- cf. Traitement des requêtes dans les SGBDR
- *Simplification d'expressions logiques* : éliminer des sous-expressions communes par la mise d'expressions composées sous forme normale conjonctive $[(p_1 \vee \cdots \vee p_n) \wedge \cdots \wedge (q_1 \vee \cdots \vee q_m)]$.
- *Transformation d'expressions logiques* : règles de réécriture syntaxique et équivalences logiques.

1. Expression de constante évaluable statiquement remplacée par sa valeur (Exemple : *"X > 550/10"* $\longrightarrow$ *"X > 55"*).
2. *X like 'ABCD'* $\longrightarrow$ *X = 'ABCD'*, si X est de type chaîne de caractères à longueur variable.

2.1/5- Simplification et évaluation d'expressions

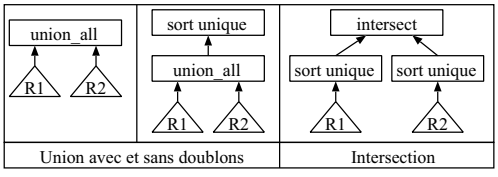
3. *"x between v1 and v2"* $\longrightarrow$ *"x >= v1 and x <= v2"*.
4. *"x in (V_1, ..., V_n)"* $\longrightarrow$ (*"x = V_1 or ... or x = V_n"*).
5. *"x > all (V_1, ..., V_n)"* $\longrightarrow$ (*"x > V_1 and ... and x > V_n"*)
6. *"x > all (select y ... where condition)"* $\longrightarrow$ *"not exists (select y ... where condition and x <= y)"*.
7. Idem pour des expressions comprenant *any* ou *some* :
   Exemple1 : *"x > any (y, z)"* $\longrightarrow$ *"x > y or x > z"*
   Exemple2 : *"x > any (select y ... where Condition)"* $\longrightarrow$ *"exists(select y ... where Condition and x > y)"*.
8. etc.

2.2/5- Réécriture de requêtes complexes

- Transformations décidées en fonction de la complexité de la requête, de la présence de vue ou d'index, etc.
- Types de transformation :
  1. Requête simple $\longrightarrow$ Requête composée,
  2. Désimbriquer des sous-requêtes,
  3. Incorporer des définitions de vues dans une requête
  4. Introduire des prédicats de la requête dans une vue.

2.2/5- Cas de requêtes composées

- Cas requête avec opérateurs ensemblistes ($\cup, \cap, \setminus$) :
  1. Un plan d'exécution pour chaque membre de l'opérateur
  2. Composition du résultat avec l'opérateur ensembliste
- Exemples : (triangle $\simeq$ sous-arbre)



2.3/5- Traitement des requêtes avec vues

(a) Intégrer la définition de la vue à la requête

(b) Etendre la définition de la vue par des prédicats de la requête.

Conditions d'application peu claires

2/5. Les étapes de l'optimisation : Plan

1. Simplification et Evaluation d'expressions (p. 243)
2. Réécriture de requêtes complexes (p. 245)
3. Réécriture de requêtes avec vues (p. 247)
4. Choix de la stratégie d'optimisation (p. 249)
5. Traitement des jointures (p. 254)

2.4/5- Stratégie et but de l'optimisation

- Stratégie (rappel) : Base de coûts/de règles
- But de l'optimisation : globale ou temps de réponse
- *Optimisation globale* : minimiser la quantité totale de ressources nécessaires au traitement de tous les tuples concernés par une instruction
  - But par défaut
  - Adaptée pour les traitements en mode différé
- *Optimisation du temps de réponse* : minimiser les ressources pour accéder au premier tuple concerné par l'instruction.
  - Adaptée pour les applications conversationnelles

2.4/5- Stratégie et but de l'optimisation : Paramétrisation

1. Niveau instance Oracle : paramètre d'initialisation *optimizer_mode*
2. Niveau session : argument *optimizer_goal* de *alter session*
3. Niveau instruction : directives d'optimisation *optimizer_mode* et *optimizer_goal*

- Si paramères utilisés comme directive d'optimisation : portée = l'instruction qui les contient.
- Si *optimizer_goal* spécifié pendant une session : substitution à la valeur de *optimizer_mode* pour la durée de la session.

2.4/5- Paramétrisation : Valeurs de *optimizer_mode* et *optimizer_goal*

.

2.4/5- Paramétrisation : Valeurs de *optimizer_mode* et *optimizer_goal*

- *choose, rule, all_rows, first_rows*

1. *choose* et présence de statistiques sur au moins une table
   - Stratégie : à base de coûts
   - But : optimisation globale
2. *choose* et absence de statistiques : stratégie à base de règles.
3. *rule :*
   - = Valeur par défaut
   - Stratégie : à base de règles (avec ou sans statistiques)
   - But : optimisation globale

2.4/5- Paramétrisation : Valeurs de *optimizer_mode* et *optimizer_goal*

4. *all_rows :* même en l'absence de statistiques,
   - Stratégie : à base de coûts
   - But : optimisation globale
5. *first_rows :*
   - Stratégie : à base de règles
   - But : minimisation du temps de réponse

2.5/5- Traitement des jointures

- Combinaison des facteurs :
  - **(a)** Type d'algorithme de jointure
  - **(b)** Ordre des jointures
  - **(c)** Chemins d'accès
    - Dans l'optimisation à base de coûts : page 263
    - Dans l'optimisation à base de règles : page 279

2.5.a/5- Choix de l'algorithme de jointure

- Méthodes ou algorithmes de jointure :
  1. Boucles imbriquées (*nested loops*),
  2. Tri-fusion (*sort-merge*),
  3. Hachage (*hash join*),
  4. Groupement (*cluster join*).
- Critères de choix : cf. pages 256 à pages 258

2.5.a/5- Choix de l'algorithme de jointure

1. Boucles imbriquées :
   - Utilisable par les deux stratégies d'optimisation
   - Choisi si taille du résultat > 10000 tuples
   - Coût = $C\_R\_ext + (C\_R\_int * Card(R\_ext))$
   - *C_R_ext* : coût de l'accès à la relation externe
   - *C_R_int* : coût de l'accès à la relation interne

2.5.a/5- Choix de l'algorithme de jointure

2. Tri fusion :
   - Utilisable par les deux stratégies d'optimisation
   - Le plus efficace si
     (i) optimisation à base de règles
     (ii) résultat de grande taille
   - Coût : Coûts d'accès aux relations [+ Coûts de leur tri]

2.5.a/5- Choix de l'algorithme de jointure

3. Hachage :
   - Tables partitionnées
   - Non utilisable dans une optimisation à base de règles
   - Le plus efficace si résultat de grande taille
   - Coût : $C\_R2 + (C\_R1 * Nb\_Partition\_R2)$,
     - *C_R1, C_R2* : coûts des accès aux relations
     - *Nb_Partition_R2* : nombre de partitions de hachage de *R2*
4. Groupement : utilisable dans une équi-jointure :
   - de tables d'un même *cluster*
   - sur la clé du cluster

2.5.b/5- Ordre des jointures

- Quelle que soit la stratégie d'optimisation :
  1. En tête de l'ordre : Table dont la jointure rend un seul tuple
  2. Poursuite de la recherche de l'ordre (variable selon les stratégies)
- cf. détails pages 260 à 261

2.5.b/5- Ordre des jointures et

1. Si optimisation à base de coûts : Engendrer et évaluer le coût des plans en prenant en compte :
   - les chemins d'accès disponibles
   - l'ordre des jointures
   - le type d'algorithme de jointure.
2. Si optimisation à base de règles :
   - Engendrer un ensemble de plans d'exécution des jointures
   - En choisir un en se laissant guider par un objectif :
     (a) de maximisation du nombre de jointures par boucles imbriquées
     (b) où la table interne est accessible *via* un index (*index scan*).

2.5.b/5- Génération des ordres des jointures

1. Engendrer des ordres, chacun ayant une table différente en tête
2. Pour chaque ordre engendré :
   (a) Le compléter en classant les tables dans l'ordre décroissant d'une pondération affectée en fonction du poids des chemins d'accès ;
   (b) Le doter d'un algorithme de jointure de chaque table (ou sous-arbre) R_i avec sa suivante R_{i+1} dans un ordre de jointure,
- Choix de l'algorithme de jointure :

  si le poids du chemin d'accès à R_i est ≤ 11
  alors algorithme des boucles imbriquées
  sinon si équi-jointure
  alors sort-merge
  sinon nested loops avec R_{i+1} en table externe

Traitement des requêtes sous Oracle : Plan

1. Architecture du compilateur de requêtes et structure de stockage des plans d'exécution (p. 232)
2. Processus et principales étapes de traitement des requêtes
3. Optimisation à base de coûts (p. 263)
4. Optimisation à base de règles (p. 279)
5. Directives d'optimisation (p. 282)

3/5. Optimisation à base de coûts

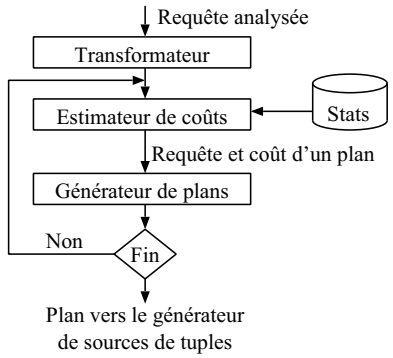
- Conditions de mise en œuvre :
  Disponibilité de statistiques + Paramètre d'initialisation :
  – *optimiser_mode = choose, first_rows* ou *all_rows*
- *Note* : Exemple de consultation des paramètres

  select name, value from v$parameter
  where upper(name) like upper('optimizer)%'
- Stratégie généralement requise pour les instructions qui utilisent :
  – des tables partitionnées et/ou organisées en index,
  – des index basés sur des fonctions.

3/5. Optimisation à base de coûts

- Processus d'optimisation :
  1. Engendrer un ensemble de plans : se fonder sur
     – les chemins d'accès existants
     – les éventuelles directives d'optimisation
  2. Estimer le coût de chaque plan : utiliser les statistiques sur
     – la distribution des données
     – les caractéristiques de stockage des objets (tables, index, partitions) impliqués
  3. Choisir le plan le moins coûteux.

3/5. Optimisation à base de coûts : Architecture



3/5. Optimisation à base de coûts

- La suite :
  1. Chemins d'accès (p. 267)
  2. Transformation de requêtes (p. 271)
  3. Estimation de coûts (p. 274)
  4. Génération de plans (p. 277)

3.1/5. Optimisation à base de coûts : Chemins d'accès

- (Terminologie d'Oracle) Chemin d'accès (*access path*) :
  1. Mode d'accès
  2. Mode de parcours éventuel des tuples d'un objet.
- Quelques types de chemins d'accès :
  - *Full table scans* : parcours de toute la table ;
  - *Sample table scans* : parcours d'un échantillon de tuples ;
  - *Table access by RowId* : accès à l'aide d'un identifiant de tuple ;
  - *Index scans* : mode de parcours d'index avec des variantes dont :
    - *unique scan* : lorsqu'un identifiant de tuple est rendu ;
    - *range scan* : étant donné un intervalle de valeurs de la clé ;
    - *full scan* : parcours de la totalité d'un index ;
    - *fast full scan* : l'index suffit pour satisfaire la requête.

3.1/5. Optimisation à base de coûts : Chemins d'accès

- Des facteurs influant sur le *choix d'un chemin d'accès* :
  - Liste des chemins disponibles,
  - Estimation du gain potentiel en utilisant tout ou partie des chemins,
  - *Sélectivité* des attributs et de la jointure,
  - Nombre de valeurs de chaque colonne d'une table (dans *user_tab_columns_num_distinct*),
  - Nombre de tuples d'une table (*user_tab_columns_num_rows*),
  - Statistiques disponibles (*user_tab_col_statistics* et *user_tab_columns*).

3.1/5. Optimisation à base de coûts : Chemins d'accès

- Des paramètres influant sur le *choix des plans d'exécution* : *db_file_multiblock_read_count*, *optimizer_features_enabled*, *optimizer_mode*, *sort_area_size*, *hash_area_size*.
- Des paramètres influant sur le *choix des index* :
  - *optimizer_index_caching* : agit sur la stratégie de gestion des index en mémoire cache (voir le chapitre Organisation et stockage des données, page 35)
  - *optimizer_index_cost_adj* : encourage l'usage intensif des index sans tenir compte de leur *sélectivité*.

3/5. Optimisation à base de coûts

- La suite :
  1. Chemins d'accès (p. 267)
  2. Transformation de requêtes (p. 271)
  3. Estimation de coûts (p. 274)
  4. Génération de plans (p. 277)

3.2/5- Optimisation à base de coûts : Transformation de requêtes

- Objectif du transformateur de requêtes : est-il avantageux d'opérer des réécritures de requêtes ?
- Evaluation de l'opportunité
  1. d'intégrer les définitions de vues à la requête,
  2. de désimbriquer des requêtes,
  3. d'utiliser des *vues matérialisées*.
- **Note** : *Vue matérialisée* = Vue instanciée ($\simeq$ Table)

3.2/5- Optimisation à base de coûts : Transformation de requêtes

1. Intégration des vues
   - Cas vue *fusionnable* avec l'instruction : Génération d'un seul plan
   - Cas vue *non fusionnable* : Génération d'un sous-plan pour la vue (dans laquelle des prédicats de la requête ont éventuellement été incorporés).
2. Sous-requêtes :
   - Certaines peuvent être désimbriquées et d'autres pas
   - Celles non désimbriquées $\longrightarrow$ Plans séparés + Ordre entre les plans.
3. Vues matérialisées : si une partie de la requête correspond à la définition d'une vue matérialisée, la remplacer par le nom de la vue.

3/5. Optimisation à base de coûts

- La suite :
  1. Chemins d'accès (p. 267)
  2. Transformation de requêtes (p. 271)
  3. Estimation des coûts (p. 274)
  4. Génération de plans (p. 277)

3.3/5- Optimisation à base de coûts : Estimations

- Facteurs :
  1. *Sélectivité*,
  2. Cardinal des ensembles
  3. Coût des ressources utilisées : principale unité = nombre d'entrées-sorties physiques (blocs de données et d'index).
- *Sélectivité*, sans disponibilité de statistiques :
  - Utilisation d'une valeur interne est utilisée
  - Exemple : Sélectivité d'une d'une expression du type *"attribut = valeur"* est estimée meilleure que celle d'une expression du type *"attribut > valeur"*.

3.3/5- Optimisation à base de coûts : Estimations

- *Sélectivité* et présence de statistiques :
  - Calcul de la sélectivité
  - Exemple : Sélectivité d'une expression *"attribut = valeur"* = (1/nombre de valeurs distinctes de l'attribut).
- *Cardinal des ensembles* :
  - Tables, Vues,
  - Résultats de jointures et de requêtes avec *group by*
  - Calcul d'un *cardinal effectif* en utilisant la sélectivité
  - Si défaut de statistiques : Estimation d'un *cardinal de base* en fonction du nombre d'*extents* occupés.

3/5. Optimisation à base de coûts

- La suite :
  1. Chemins d'accès (p. 267)
  2. Transformation de requêtes (p. 271)
  3. Estimation des coûts (p. 274)
  4. Génération de plans (p. 277)

3.4/5- Optimisation à base de coûts : Génération de plans

- Un sous-plan :
  - pour chaque sous-requête désimbriquée
  - pour chaque vue non intégrée à la requête.
- Génération et optimisation du plan global de bas en haut (du bloc de requête le plus interne vers la requête)
- Arrêt lorsque l'évaluateur estime que le coût du dernier plan engendré est "acceptable"
  (i.e. rapport gain escompté-coût examen exhaustif)
- Optimisation à base de coûts extensible :
  - Si fonctions ou index de domaines créés
  - Obligation de fournir des fonctions de calcul des statistiques, de la sélectivité et des coûts.

Traitement des requêtes sous Oracle : Plan

1. Architecture du compilateur de requêtes et structure de stockage des plans d'exécution (p. 232)
2. Processus et principales étapes de traitement des requêtes
3. Optimisation à base de coûts (p. 263)
4. Optimisation à base de règles (p. 279)
5. Directives d'optimisation (p. 282)

4/5. Optimisation à base de règles

- Choix du plan d'exécution en fonction
  1. des chemins d'accès existants
  2. d'un poids associé à chaque type de chemins,
- Meilleur chemin : celui de plus faible poids
- Quinze types de chemins disponibles
- Leur choix dépend :
  1. de la forme de la requête
  2. de la structure des objets concernés

4/5. Optimisation à base de règles : Exemples de choix de chemins d'accès

Chemin d'accès	**Conditions et forme de la requête**
Single Row by RowId	*where rowid = '...'*
Single Row by cluster join	*where* R1.A = R2.A *and* R1.B = valeur et R1, R2 groupées (*cluster*) sur A et B est clé primaire de R1.
single row by unique or primary key	La clause *where* porte sur tous les attributs d'une clé primaire ou d'un index *unique*.
bounded range search or index columns	*where* A = expression ou *where* A > [=] expression1 *and* A > [=] expression2.

Traitement des requêtes sous Oracle : Plan

1. Architecture du compilateur de requêtes et structure de stockage des plans d'exécution (p. 232)
2. Processus et principales étapes de traitement des requêtes
3. Optimisation à base de coûts (p. 263)
4. Optimisation à base de règles (p. 279)
5. Directives d'optimisation (p. 282)

5/5. Directives d'optimisation

- Pour forcer certains choix de l'optimiseur
- Sous la forme : "... /* + *directives* */ ..." dans une instruction
- Exemples :

5/5. Directives d'optimisation et leurs objets

1. Méthode et but de l'optimiseur : *all_rows*, *first_rows*, *choose*, *rule*.
   - *all_rows*, *first_rows*, *choose* $\longrightarrow$ invoquer l'optimiseur à base de coûts
2. Méthode d'accès : *full, rowid, index, cluster, index_join, rewrite,* etc.
3. Ordre des jointures : *ordered.*
4. Algorithme de jointure :
   - *use_nl* : boucles imbriquées,
   - *use_merge* : tri-fusion),
   - *use_hash* : fonction de hachage
   - *use_nl*, *use_merge* recommandés conjointement avec *ordered*

5/5. Directives d'optimisation et leurs objets

5. Fusion (ou pas) de vues :
   - *merge, no_merge*,
   - *push_pred*, *no_push_pred.*
6. Désimbrication (ou non) de sous-requêtes : *unnest, no_unnest*.
7. Stratégie de gestion en mémoire cache : *cache, nocache.*
8. Exécution parallèle d'instructions : *parallel, noparallel, parallel_index* et *noparallel_index*.
9. etc.

Traitement des requêtes : Conclusion

- Vision non naïve du traitement des requêtes
- Traitement fondé sur :
  - Propriétés formelles des opérateurs algébriques
  - Facteurs de coût (logique et physique)
- A connaître pour $\nearrow$ performances

Chapitre VIII : Adaptation de serveurs et performances

Adaptation (tuning) d'un serveur

- Tuning = optimisation des performances
- Partie essentielle du tuning : réduire les conflits sur les ressources système (caches, verrouillage de pages, CPU, etc.)
- Niveaux d'adaptation :
  - Applications
  - Bases de données
  - Serveurs
  - Unités de stockage
  - Réseau
  - Matériel (Hardware)
  - Système d'exploitation

I- Processus d'adaptation/optimisation

- Identifier les origines des problèmes de performances
- Leur apporter une solution

1. Collecte de données sur les performances : utilisation d'outils t.q.
   - Bancs d'essais standards ou "maison"
   - Outils du système (exemple sybase : *sp_sysmon* ou SQL Server Monitor), etc.
2. Analyse des données
   - Symptômes ?
   - Problèmes touchant un/plusieurs utilisateurs/applications/bases de données ?
   - Problème "fugitif" ou permanent ?

I- Processus d'adaptation/optimisation

3. Analyse des applications
   - Examen des relations, index, transactions, etc.

4. Diagnostics (hypothèses) : d'où peut provenir le problème ?
   - Valeur des paramètres de configuration ?
   - Mauvaise conception ?
   - Mauvaise affectation de ressources, etc. ?

I- Processus d'adaptation/optimisation

5. Test et validation des hypothèses : "remèdes" fonction des diagnostics
   - Test de solution ;
   - Vérification par collecte des mêmes types de données qu'en 1 ;
   - Itérations éventuelles (sur les remèdes alternatifs).

6. Concrétisation des hypothèses validées
   - Attention aux optimisations "locales".

II.1- Applications et tuning

- Essentiellement, optimisation des requêtes et des transactions

1. Facteurs pouvant affecter les performances
   - Types de transactions (OLTP vs OLAP) peuvent nécessiter des stratégies différentes ;
   - Durée des transactions : longues transactions et durée du verrouillage ;
   - Contraintes d'intégrité référentielle (nécessitent des jointures) ;
   - Index et mises à jour ;
   - Audit et diminution des performances.

II.1- Applications et tuning

2. Eléments de décision
   - Dénormalisation, redondance, fragmentation, ajout d'attributs, etc. (cf. Niveau logique de la démarche de conception) ;
   - Eviter les attributs dérivés (calculés) ;
   - Traitement à distance et/ou réplication peuvent alléger la machine OLTP ;
   - Minimiser les niveaux de verrouillage ;
   - Utiliser de procédures stockées (réduction temps de compilation et activité réseau) ;
   - Optimiser les transactions/requêtes coûteuses.

II.2. Bases de données et tuning

1. Facteurs
   - Politique de sauvegarde ;
   - Distribution de données sur des unités ;
   - Surveillance (*Auditing*) ;
   - Activités de maintenance, comme :
     - Création de bases, d'index ;
     - Sauvegarde/chargement ;
     - Chargement en masse ;
     - Test de la cohérence ;
     - Mise à jour de statistiques.

II.2. Bases de données et tuning

2. Eléments de décision
   - Automatiser les sauvegardes (avec bornage des espaces ou *thresholds*) pour éviter les débordements ;
   - Sauvegardes en dehors des heures d'activité ;
   - Bornage des espaces sur les segments de données ;
   - Partitionnement de relations : accélération chargement de données ;
   - Bon placement des objets pour $\searrow$ concurrence sur les accès disque :
     - Unités physiques différentes (si possible) ;
     - Au minimum : segments différents.
   - Configuration des caches pour les tables et les index importants ;
   - Périodicité des points de contrôle : ni trop long ni trop court.

II.3. Serveur de données et tuning

1. Facteurs
   - Types d'application : OLTP et/ou OLAP
   - Nombre d'utilisateurs
   - Charge réseau
   - Existence de réplicats
   - Transactions réparties

II.3. Serveur de données et tuning

2. Eléments de décision
   - Taille mémoire, cache, tampons d'E/S, nombre de connexions, etc.
   - Serveurs dédiés OLTP/OLAP
   - Traitements différés (batch)
   - Déporter des applications au niveau des clients
   - Machine multi-processeur

II.4. Unités de stockage et tuning

1. Facteurs
   - Quels miroirs ?
   - Répartition (données, logs, miroirs) sur les unités ;
   - Partitionnement de relations.
2. Eléments de décision
   - Unités de taille moyenne au lieu de très grandes unités ;
   - Répartir les bases, les relations et les index.

II.5. Réseau et tuning

1. Facteurs
   - Débit ;
   - Charge ;
   - Taux erreurs réseau ;
   - Identification et localisation des goulots d'étranglement :
     - Quels processus/programmes accèdent à de grands volumes de données ?
     - Transactions utilisant SQL % celles utilisant des procédures stockées ?
     - Transactions distribuées utilisant le 2pc ?
     - Services de réplicats actifs ?
     - Part de la charge réseau non dûe au serveur SQL ?

II.5. Réseau et tuning

2. Eléments de décision
   - Réduction des communications au niveau des applications ;
     - Procédures stockées ;
     - Ne lire que les seules données nécessaires :
       - *select X, Y, . . .* au lieu de *select* *
       - *curseurs* : lire un ensemble de tuples, au lieu d'un à la fois.
   - Configuration de sous-réseaux ;
   - Isolation des utilisations qui chargent le réseau (dans un sous-réseau ; une carte réseau par serveur SQL) ;
   - Configuration de la taille des paquets.

II.6. Matériel et tuning

1. Facteurs
   - Capacité CPU
   - Disques et controleurs
   - Taille mémoire
2. Eléments de décision
   - Proceseurs multiples
   - Conception d'applications multi-processeurs
   - Configuration caches multiples
3. Procédures, paramètres de configuration, commandes
   - cf. configuation des caches
   - cf. Installation sur machine multi-processeurs

II.7. Système d'exploitation et tuning

1. Facteurs
   - SGF utilisé par d'autres logiciels ou applications ?
   - Ressources pour d'autres logiciels ou types d'application ?
   - Nombre de processeurs affectés au serveur SQL ?
2. Eléments de décision
   - Accroître la taille mémoire ;
   - Choisir entre fichiers et raw devices ;
   - Dédier la machine au serveur SQL ;
   - Affecter plusieurs processeurs au serveur SQL.

III- Tuning : Conclusion

- Une des fonctions les plus importantes de l'administrateur
- Compétences multiples

References

[1] Besancenot (J.) et al. – *Les systèmes transactionnels: concepts, normes et prouits*. – Paris, Editions Hermes, 1997, *Collection informatique*.

[2] Boudjlida (N.). – *Bases de données et systèmes d'informations. Le modèle relationnel: langages, systèmes et méthodes (Databases and Information Systems. The relational model: Languages, Systems and Design)*. – Dunod, Paris, 1999. Cours et exercices corrigés. Collection Sciences Sup. (*in French*).

[3] Boudjlida (N.). – *Gestion et Administration des Bases de Données: Application à Sybase et Oracle*. – Dunod, Paris, 2003.

[4] Brown (P.G.). – *Object-Relational Database Development*. – Prentice-Hall, 2000. ISBN 0130194603.

[5] Oracle Corp.– *Administrator's Guide*, 10g Release 2 (10.2) B14231-02, May 2006

[6] Oracle Corp.– *Concepts*, 10g Release 2 (10.2) B14220-02, October 2005

[7] Oracle Corp.– *Performance Tuning Guide*, 10g Release 2 (10.2) B14211-01, June 2005

[8] Oracle Corp.– *Reference*, 10g Release 2 (10.2) B14237-02, November 2005

[9] Oracle Corp.– *SQL Reference*, 10g Release 2 (10.2) B14200-02, December 2005