

Eléments d'Architecture des SGBDR

Application au SGBD Oracle

Nacer Boudjlida
<http://www.loria.fr/~nacer>
Université Henri Poincaré Nancy 1
FST, UFR STMIA
(Avril 2006)

Eléments d'Architecture 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.

Eléments d'Architecture des SGBDR : Introduction

1. Fonctions d'un SGBD
2. Typologie des SGBD
3. Situation dans l'architecture des applications
4. Les langages des SGBD relationnels
5. Architecture des SGBD

1- Rôle/Fonctions d'un SGBD

- 1.
- 2.
- 3.
- 4.
- 5.

2- Typologie des SGBD

- Modèle de représentation et de manipulation de données → Classe de SGBD
 1. Hiérarchique, Réseau → CODASYL
 2. Relationnel → SGBD Relationnel
 3. A objets → SGBDOO
 4. [Relationnel "étendu" → "Objet-Relationnel]
 5. Logique → SGBD Déductif
 6. Non ou semi-structurées → SGBD pour XML (?)

3- Situation dans l'architecture des applications

3- Situation dans l'architecture des applications

4- Langages Relationnels

Type	Fondement
Algébrique	Théorie des ensembles
Prédicatifs	Logique du 1er ordre
a) à variable n-uplet	
b) à variable domaine	

- SQL \simeq "dialecte" fondé sur l'algèbre et le calcul prédicatif à variable n-uplet

4.1- Algèbre relationnelle

- Caractéristiques :
 - Opérande(s) : Relation(s)
 - Résultat : Relation
 - Opérateur : Opérateur du calcul relationnel
- Ensemble minimal d'opérateurs :
 - 1.
 - 2.
 - 3.
 - 4.
 - 5.
- Remarque : Jointure (\bowtie =

4.1- Algèbre relationnelle**4.1- Algèbre relationnelle****4.2- Syntaxe d'un langage algébrique**

- Si R est un nom de relation alors R est une expression algébrique (ea)
-

4.3- De l'algèbre à SQL

- $R \rightarrow$
- $\sigma_C(R) \rightarrow$
- $\Pi_L(R) \rightarrow$
- $R1 \times R2) \rightarrow$

- $R1 \setminus R2 \rightarrow$

- $R1 \cup R2 \rightarrow$

- En réalité, dans un SGBD : de SQL vers l'algèbre!

5- Architecture d'un SGBD**Eléments d'Architecture des SGBDR : Sommaire**

- Chapitre 1** Introduction (p. 3)
- Chapitre 2** Eléments d'architecture du SGBD Oracle (p. 16)
- Chapitre 3** Organisation et stockage de données (p. 36)
- Chapitre 4** Traitement des requêtes (p. 156)
- Bibliographie** (p. 262)

Chapitre II : Eléments d'architecture du SGBD Oracle

- Sommaire :
 1. Quelques concepts Oracle
 2. Eléments d'architecture du SGBD Oracle

I. Quelques concepts Oracle

- *Serveur de données Oracle* : Une base + une instance
- *Instance* :
 1. {Processus} d'arrière-plan
 2. *Zone Globale Système* (System Global Area, SGA) : tampons données, journal partagés
- *Dictionnaire* : {Tables} et {Vues}
 - Créées dans chaque base, dans l'espace *SYSTEM*
 - Propriétaire = *sys*, utilisateur privilégié
 - Non modifiables (sauf la table *sys.aud\$*)

I.1- Quelques concepts Oracle : Le dictionnaire

- Vues *USER_xxx* :
 - concernent les objets d'un utilisateur
 - accessibles par cet utilisateur
 - Exemple : Vue *USER_TABLES* = tables d'un utilisateur

Quelques concepts Oracle : Le dictionnaire

- Vues *ALL_xxx* :
 - étendent les informations des vues *USER* par des informations sur les objets auxquels il a accès
 - Vue *ALL_TABLES* : tables d'un utilisateur + tables auxquelles il a accès
- Vues *DBA_xxx* :
 - Plus de colonnes que les autres
 - Information sur les objets de tous les utilisateurs
 - Vue *DBA_TABLES* : toutes les tables d'une base

Quelques concepts Oracle : Le dictionnaire

- Vues *V\$_xxx* :
 - Tables de performance dynamiques
 - Propriété de *sys*
 - Concernent l'activité d'une base
 - Vue *V\$_DATAFILE* : Information sur les fichiers physiques d'une base
- Vue *DICTIONARY* : Liste des tables et des vues du dictionnaire
- *DESCRIBE* NomDeTable ou NomDeVue : liste des colonnes

II.2- Composants d'une base Oracle

- *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, ...)
- *Init file* : Paramètres d'initialisation de la base
- *Transactions et journalisation* :
 1. Images *avant* : segments/fichiers d'annulation (*rollback segments/undo tablespaces*)
 2. Images *après* : journaux de réexécution (*redo log files*)
- *Espaces de stockage* (données, index, etc.).

II.3- Oracle : éléments d'organisation de données et de stockage

- Espace de table (*tablespace*) : ensemble d'unités logiques de stockage des objets d'une base
- Base : découpée en 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
- Espaces physiques persistants (disque) structurés en :
 - Pages (ou *blocs*) = unité (par défaut) de lecture/écriture physique
 - *extents* : nombre déterminé de blocs contigus
 - *segments* : ensemble d'*extents* non nécessairement contigus

Oracle : organisation et stockage

Oracle : organisation et stockage

II.3- Concepts Oracle : Les processus Oracle

- *Processus utilisateurs et 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
 - mécanismes de conversion de données (cas machines hétérogènes)

Les processus Oracle

2. *Processus Oracle* :

- (a) un processus serveur
- (b) un ensemble de processus d'arrière-plan (*background processes*)
- *Note* : *Net8* = interface propriétaire avec les protocoles standard de communication sur un réseau

Processus serveur Oracle

- créé pour gérer les demandes d'actions des utilisateurs
- chargés 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 arrière-plan

Oracle : Processus arrière-plan

- Exécution asynchrone des opérations d'entrées/sorties et de contrôle d'autres processus Oracle
- Principaux processus :
 1. *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
 2. *LGWR* (*Log Writer*) : écriture
 - dans les fichiers *Redo Log*
 - dans leurs miroirs éventuels

Oracle : Processus arrière-plan

3. *CKPT* : informe le DBW de l'arrivée d'un point de contrôle (*checkpoint*)
4. *SMON* (*System Monitor*) réalise (entre autres) :
 - la reprise (*recovery*), lors de la relance d'une instance (*startup*) ;
 - le "nettoyage" des segments d'annulation (*rollback segments*) qui ne sont plus utilisés ;
 - le compactage d'espaces physiques (*extents*) dans les espaces de données (*tablespace*) gérés par le dictionnaire (*Dictionary-Managed Tablespace*)

Oracle : Processus arrière-plan

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

Oracle : Processus arrière-plan

7. *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
8. *LCK0* : en charge du verrouillage inter-instances dans *Oracle parallel server*, celui-ci permettant l'accès simultané à une base par plusieurs instances.

II.4- Concepts Oracle : Mémoires cache

- Structures mémoire d'une instance Oracle incluent :
 1. Zone globale système (SGA)
 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.

Oracle : encore et encore

- Création de bases de données (Bon courage!)
- Gestion des utilisateurs
- Concurrence et reprise
- Sécurité par duplication (*multiplexage*)
- Surveillance (*audit*)
- Traitement des requêtes
- Extensions objets
- Bases de données réparties
- etc.

Chapitre III : Organisation et stockage des données

Contenu du chapitre :

1. Organisation et stockage des données (p. 37)
2. Organisation et stockage des index (p. 65)
3. Application à Oracle (p. 85)

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

- 80% de l'amélioration des performances obtenus par \searrow des lectures
 - Sans index : parcours de toute la table (i.e. parcours de *toutes les pages* contenant des tuples de la table)
 - Possibilité d'estimer *a priori* le nombre d'E/S pour des σ ou des \bowtie
- Selon l'existence ou non d'index et leur type, l'optimiseur décide :
 1. *Table Scan* : Lecture de toutes les pages
 2. *Index Access* : Lecture *via* les index
 3. *Index Covering* : Valeurs aux feuilles suffisent pour satisfaire la requête

Organisation et Stockage des données : Sommaire

1. Organisation en pages
2. Représentation des tuples
3. Cas des "objets longs" : texte et image
4. Organisation "en tas" (*heap*)
5. La mémoire cache

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

Structure générale des pages

Organisation et Stockage des données : Sommaire

1. Organisation en pages
2. Représentation des tuples
3. Cas des "objets longs" : texte et image
4. Organisation "en tas" (*heap*)
5. La mémoire cache

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”
 - Pointeurs vers les valeurs des objets longs

3. Stockage des objets de grande taille

Organisation et Stockage des données : Sommaire

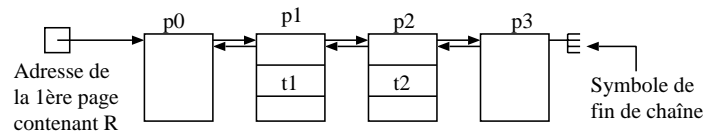
1. Organisation en pages
2. Représentation des tuples
3. Cas des “objets longs” : texte et image
4. Organisation “en tas” (*heap*)
5. La mémoire cache

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 (“effet gruyère”)
- Mise à jour des déplacements des tuples
- Cas page vide après suppression ?
- Cas extent vide après suppression ?

Organisation “en tas” et 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

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*

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

Organisation et Stockage des données : Sommaire

1. Organisation en pages
2. Représentation des tuples
3. Cas des “objets longs” : texte et image
4. Organisation “en tas” (*heap*)
5. [La mémoire cache](#)

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 \rightarrow cache
 3. Si toutes les pages du cache sont occupées: en “vider” une
- Stratégies de choix de la page du cache à remplacer (à “vider”) :
 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 buffers Most Recently Used/Less RU
- “Âge” d'un buffer : MRU \rightarrow 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.

1- Stratégie LRU de remplacement de pages

1- Stratégie LRU de remplacement de pages (fin)

- \implies Recherche d'une page p_i et
 - p_i non modifiée et $p_i \in \text{cache}$: p_i
 - p_i modifiée et $p_i \in \text{cache}$: p_i en tête MRU
 - $p_i \notin \text{cache}$: lecture disque (1 buffer) en tête de MRU.

2- Stratégie MRU de remplacement de pages

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

Chapitre III : Organisation et stockage des données

1. Organisation et stockage des données (p. 37)
2. [Organisation et stockage des index](#) (p. 65)
3. Application à Oracle (p. 85)

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 : $\langle V_i, P_i \rangle$ 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

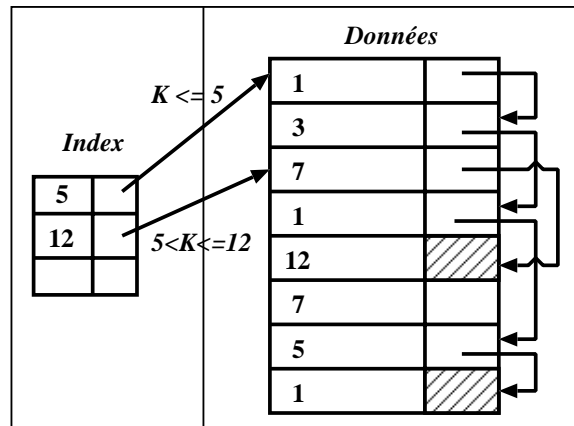
Typologie des index : *Index dense*

Typologie des index

2/4) *Index creux*

- Moins d'entrées que de valeurs de la clé
- Entrée : $\langle V_i, P_i \rangle$ 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



©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Typologie des index

3/4) Index primaire

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

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Typologie des index : Index Primaire

Typologie des index

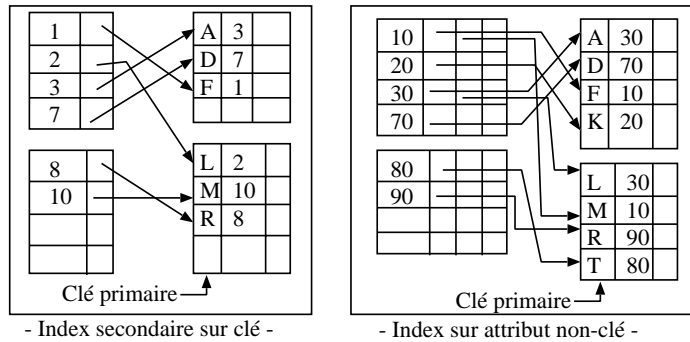
4/4) Index secondaire

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

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Typologie des index : Index secondaire



©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Représentation des index

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

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

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 : \rightarrow 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 $K_j, K_{i-1} \leq K_j < K_i$

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

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

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Représentation des index : (1) Arbres B^+

- 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

Arbres B^+ et mise à jour : avant

- cf. exemple de la page 77

Arbres B^+ après insertion

- Ajout d'un tuple de clé C

Arbres B^+ après suppression

- Suppression de la valeur P (sur l'arbre de la page 77)

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

Représentation des index : (2) B -arbres

- Exemple Correspondant à l'exemple d'arbre B^+ , page 80

Chapitre III : Organisation et stockage des données

Contenu du chapitre :

1. Organisation et stockage des données (p. 37)
2. Organisation et stockage des index (p. 65)
3. [Application à Oracle](#) (p. 85)

Organisation et stockage : Application à Oracle

1. (Quelques) Objets logiques (*schema objects*)
 - (a) Types de tables
 - (b) Types d'index
2. Organisation et stockage

Application à Oracle : Types de tables

1. Tables "classiques" (*create table*)
2. Tables objets (*create ... as object*)
3. Tables organisées en index (*create table ... index organized*)
4. Tables partitionnées (*create table ... partition*)
5. Groupement de tables (*cluster*).

Application à Oracle : Types d'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
 - *Mais éviter de manipuler une grande partie de la table pendant la création ou la reconstruction de l'index.*

Application à Oracle : Types d'index

- Index automatiquement créé sur attributs *UNIQUE* ou *PRIMARY KEY* :
 - \Rightarrow Allouer un espace suffisant pour les tables ;
 - Suppression des contraintes \rightarrow Suppression des index.
- 5 types d'index :
 1. "Ordinaires" (*create index*)
 2. Matrices de positions binaires (*create bitmap index*)
 3. Basés sur des fonctions (*create index ... on Fonction*)
 4. Spécifiques à un domaine d'application (cf. *Oracle Data Cartridge Interface*)
 5. Partitionnés (*create index ... partition*)

Application à Oracle : Organisation et stockage physique

- Espace de tables (*tablespace*) : ensemble d'unités logiques de stockage des objets d'une base
- Base : découpée en 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 (rappels)

- 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*)
- Espaces physiques persistants (disque) structurés en :
 - Pages (*blocs*) : unité de lecture/écriture physique
 - *extents* : nombre déterminé de blocs contigus
 - *segments* : ensemble d'*extents* non nécessairement contigus

Application au SGBD Oracle

- 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)
 - peut suffire pour une base de petite taille
- **MAIS**, (recommandé) créer des *tablespaces* pour
 - contrôler le placement des objets
 - affecter des ressources aux utilisateurs

Introduction aux tablespaces

- Exemples de tablespaces “dédiés” à :
 - Images avant modification (*undo tablespaces*)
 - Stockage exclusif des données ou des index
 - Tables des outils du système
 - Espaces temporaires (pour les tris, par exemple)
- Durée de vie d'une *tablespace* : création (*create tablespace*) → suppression
 - explicite (*drop tablespace*)
 - implicite : *tablespace temporaire* (→ 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

1. *dictionary-managed* : Mode par défaut
 - Gestion d'une liste des *extents* libres dans des tables du dictionnaire
 - Journalisation des modifications de celles-ci
2. *locally managed* :
 - Sur les 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 des tablespaces

- Modes de gestion également applicables aux *tablespaces* temporaires
 - “*create tablespace Nom-de-la-tablespace temporary ...*” : *tablespace* temporaire avec gestion par dictionnaire
 - “*create temporary tablespace ...tempfile ...*” : avec gestion locale.

Tablespace : Exemples

- *Tablespace Espace1* : deux fichiers physiques (*datafiles*)
- *Tablespaces Espace2* et *EspaceTemp* : un seul espace physique

Statuts et états d'une tablespace

1. Temporaire/permanente : changement par :
alter tablespace ... temporary/permanent
2. Accessible (online) / Inaccessible (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
 - 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.
- Dans *dba_tablespaces* : informations sur toutes les *tablespaces*
- Dans *user_tablespaces* : sur celles accessibles par un utilisateur.

Etats et statuts d'une tablespace : Exemples

```
alter tablespace Espace2 offline temporary;
create tablespace Espace3 datafile 'D:\Fichier3.dat' size 30M offline;
alter tablespace Espace3 read only;
alter tablespace Espace3 online;
```

- Mise hors ligne temporaire de l'espace *Espace2*
- Création *Espace3* hors ligne puis mise en lecture seule puis remise en ligne

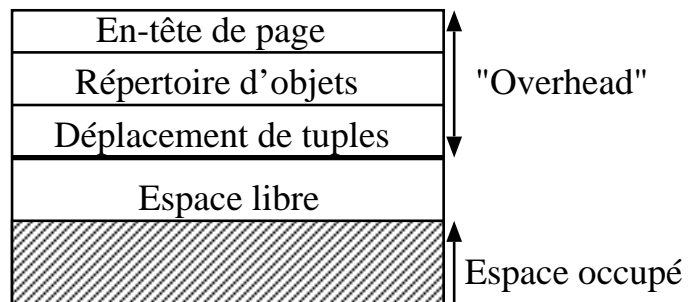
Application au SGBD Oracle

1. Notions sur les *tablespaces*
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
- Structure :
 1. *Descripteur*
 2. *Contenu*

Blocs Oracle : Structure



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
 - information stockée dans des *transaction entries*
 - Une entrée par opération *insert, update, delete, select for update*
 - environ 230 octets/entrée
- Paramètres de stockage *INITRANS* et *MAXTRANS* :
 - Nombre initial et nombre maximum de transactions concurrentes sur le bloc ;
 - Index, tables et des 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 ayant leurs propres segments de stockage :
 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 $\geq PCTFREE$:
 - Insertions interdites dans le bloc ;
 - Suppressions et modifications autorisées ;
 - Reprise des insertions si taux de remplissage $< PCTUSED$.

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%.

PCTFREE, PCTUSED : Exemple

Gestion et contrôle du remplissage d'un bloc

- Si taille d'un tuple > capacité d'un bloc (par exemple, types *LONG* et *LONG RAW*) : Rangement dans une chaîne de blocs.
- Pas de *compactage* (i.e. élimination "effet gruyère", *coalesce*) systématique du bloc après libération de place ;
- Compactage lorsque, pour une insertion ou une modification :
 1. espace contigu insuffisant ;
 2. mais somme d'espaces non contigus suffisante.

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 caches

3. Extents Oracle

- *Extent* =
 - 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

Modes de gestion des tablespaces : Rappel

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

Modes de gestion des tablespaces : Rappel

1. Par défaut : *dictionary-managed*
 - Gestion d'une liste des extents libres dans des tables du dictionnaire avec journalisation des modifications de ces tables
2. Mode de gestion dit *local* :
 - Sur les 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

a) 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) Paramètres de stockage et gestion locale de tablespaces

- Choix de la taille d'un extent :
 - Arguments *UNIFORM* ou *AUTOALLOCATE* de *create tablespace*
 - *UNIFORM* : Taille fixe (celle 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
 - * Possibilité de spécifier la taille de l'extent initial.

Paramètres de stockage et extents : exemple

```
create tablespace T1 datafile "F:\Data\Fichier2T1.dat" size 5 M
  extent management local uniform size 128K;
create tablespace T2 datafile "D:\Data\Fichier2T2.dat" size 2M
  extent management local uniform;
```

- T1 : tablespace : 5M; extents : 128K
- T2 : tablespace : 2M; extents : 64K, minimum

Paramètres de stockage et extents : exemple**b) 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.
 - 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*,
 - *PCTINCREASE*,
 - *MINEXTENTS*, *MAXEXTENTS*,
 - *BUFFER_POOL*.

b) Stockage et gestion par le dictionnaire : exemple

- Table avec au plus 30 *extents*,
- *Extent* initial : 100K
- Deuxième *extent* : 50K
- Pourcentage de croissance des suivants : 5
- Troisième *extent* :
 - Taille "théorique" : $(50 + (50 * 5 / 100)) = 52.5K$
 - Si blocs de 4K : allocation de 13 blocs (52K = plus proche multiple de la taille des blocs).

b) Stockage et gestion par le dictionnaire : exemple

b) Paramètres de stockage et gestion par le dictionnaire

1. INITIAL xK ou yM :

- 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 $\lceil t/\text{taille d'un bloc} \rceil$ 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) Paramètres de stockage et gestion par le dictionnaire

2. NEXT xK ou yM :

- = 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* $t_i, i > 2 : t_i = (1 + \text{pctincrease}/100) * t_{(i-1)}$.
- t_i arrondi à la taille du plus proche multiple de la taille des blocs.
- Par défaut, PCTINCREASE = 50%.

b) 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
 - cf. Gestion des caches

b) Stockage et gestion par le dictionnaire (suite)

- Rappel : 1er *extent* alloué dès la création de la définition d'un objet :
- Allocation des autres *extents* :

b) Stockage et gestion par le dictionnaire

- **Note** : compactages périodiques par le processus d'arrière-plan *SMON*.

Gestion des *extents* (fin) : Libération

- Généralement pas de libération physique avant suppression de l'objet contenu
- Libération logique : 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*”.

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 caches

4. Segments Oracle

- Blocs des *extents* : contigus,
- *Extents* d'un segment : pas nécessairement contigus
- Allocation d'un segment :
 - à un objet de la base,
 - dans une *tablespace*
 - éventuellement dans des fichiers physiques différents.
- Quelques types de segments :
 1. Données
 2. Index
 3. Annulation
 4. Temporaires

Oracle : Exemple des segments temporaires

- = Espaces disque
- Utilisés lors de l'analyse et de l'exécution de requêtes
 - *create index*,
 - *select distinct* avec/sans *order by* et/ou *group by*
 - avec \cup , \cap , \setminus
 - de jointures sans index.
- Utilisables pour des index construits pour des tables temporaires

Oracle : Segments temporaires

- 0 segment requis si un tri est réalisable :
 - en mémoire vive (cf. gestion des caches)
 - ou en n'utilisant que les index
- 2 segments parfois pour des requêtes avec *order by*, *distinct* et *group by*
- Localisation par défaut : *tablespace SYSTEM*
- Segments dans *tablespaces* temporaires : *create temporary tablespace*
- Association utilisateur-espace temporaire :
 - “*create* ou *alter user ... temporary tablespace* NomEspace”
 - cf. chapitre gestion des utilisateurs

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 caches

5. Mémoires cache 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
 3. Zones de tri

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 ↘ 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. Mémoires cache Oracle

- Plan :
 1. SGA
 - (a) Taille
 - (b) [Cache de données](#)
 - (c) Cache du *log*
 2. PGA
 3. Zones de tri

5.1.2 Mémoires cache : cache de données

- Taille fonction de :
 - *db_block_size* (généralement 2 ou 4KO)
 - *db_block_buffers*.
- Comporte :
 1. *write list* : liste des blocs modifiés en attente d'écriture physique ;
 2. liste *LRU* ("*Least Recently Used*") :
 - blocs libres,
 - \cup blocs modifiés non encore transférés vers la *write list*
 - \cup blocs en cours d'utilisation ("*pinned blocks*").

5.1.2 Mémoires cache : cache de données

- 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*"
- 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 $\{alter \mid create\} \{table \mid index \mid cluster\}$
 - 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 :
 - $db_block_buffers = 2048$
 - $buffer_pool_keep = (buffers: 300, \dots)$
 - $buffer_pool_reuse = 100$,
- Pool *KEEP* : 300 blocs ;
- Pool *REUSE* : 100 ;
- Pool par défaut : $(2048 - (300 + 100))$.

5. Mémoires cache Oracle

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

5.1.3 Mémoires cache : le 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

5.1.3 Gestion des blocs *redo log* : Exemple

- Cache de 4 blocs et 3 premiers saturés :
 - Journalisations dans le 4ème,
 - *en même temps* : écriture physique du 1er ;
- 4ème bloc saturé :
 - Journalisations dans le 1er
 - + écriture physique du 2ème.
- etc.

5.1.3 Gestion des blocs *redo log* : Exemple

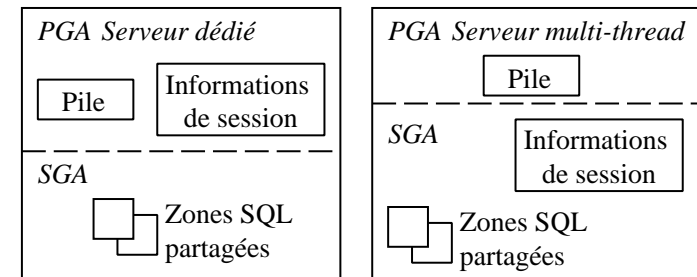
5. Mémoires cache Oracle

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

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*.

5.2- Mémoires cache : *Program Global Area*



5. Mémoires cache Oracle

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

5.3- Mémoires cache : Zone de tri

- Opérations de tri : Utilisation
 - de segments temporaires (disque)
 - d'une partie de la *PGA* (mémoire) du processus serveur d'Oracle qui réalise le tri pour le compte d'un processus utilisateur
 - + une partie de la zone de tri existe dans la zone *runtime* de la zone SQL privée du processus utilisateur.

5.3- Mémoires cache : Zone de tri

- Durant un tri,
 - Taille de la zone peut croître jusqu'à *sort_area_size*,
 - Désallocation de parties si le système a besoin d'espace pour d'autres tâches,
 - Contenu de l'espace libéré écrit dans les segments temporaires,
 - Désallocations ne doivent pas réduire la zone à une zone de taille $T_z, T_z < sort_area_retained_size$.
- Par défaut, $sort_area_retained_size = sort_area_size$.

Organisation et stockage des données et des index : Conclusion

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

Chapitre IV : Traitement des requêtes

- Contenu du chapitre :
 1. [Traitement des requêtes dans les SGBDR](#) (p. 157)
 2. Application à Oracle (p. 206)

Traitement des requêtes dans les SGBD relationnels

- Evaluation des requêtes :
 - Données de la base vers la mémoire centrale
 - Traitement
 - “Retour” dans le cas de mises à jour
- Peut nécessiter la création de tables intermédiaires
- Volume des transferts et des tables de travail fonction des tailles des relations mises en jeu
- Objectif d'un optimiseur : Réduction des volumes

Traitement des requêtes : Besoins d'optimiser ?

- Temps d'exécution d'une requête “anormal” % taille des relations, existence d'index
- Une requête s'exécute plus lentement que des requêtes similaires
- ↗ temps d'exécution d'une requête
- Temps d'exécution d'une requête en tant que procédure > celui de son exécution en tant que requête
- Plan d'exécution utilisant un parcours de relation au lieu d'utiliser un index

Traitement des requêtes : Origines des mauvaises performances ?

- “Vieilles” statistiques sur la distribution des données
- Inexistence d'index appropriés
- Index en cours d'utilisation pour accéder à une table volumineuse
- Clause *where* conduisant au choix d'une mauvaise stratégie
- Procédure non re-compilée après changements significatifs
- etc.

Traitement des requêtes : Les méthodes

1. Syntaxique :
 - Fondement : Heuristiques + Propriétés de l'algèbre
 - Transformation d'arbres (algèbre) ou de graphes (calcul relationnel)
2. Estimation de coûts d'exécution de diverses stratégies d'évaluation
3. Sémantique : Exploitation des contraintes d'intégrité
 - Méthodes non exclusives : 1 et 2 souvent combinées

Traitement des requêtes : Plan

1. Transformation d'arbres algériques
2. Transformation de graphes de requêtes
3. Optimisation à base de coûts
4. Optimisation Sémantique
5. Processus Général d'Optimisation

I. Transformation d'arbres algériques

- Heuristiques : \searrow taille des opérandes des opérations les plus coûteuses
 1. "Descente" des sélections : \searrow les relations "en hauteur"
 2. "Descente" des projections : \searrow les relations "en largeur"
- [Phrase SQL \longrightarrow] Arbre algébrique :
 - *Feuilles* : Relations
 - *Racine* : Résultat de la requête
 - *Nœuds internes* : Opérations de l'algèbre
 - *Arcs "entrants"* : Opérandes
 - *Arcs "Sortants"* : Résultat de l'évaluation d'un nœud
- Evaluation de "bas en haut" (gauche-droite ou droite-gauche)

I. Transformation d'arbres algériques : Exemple

- *Personne*(id#, nom, prenom, date_naissance, adresse, equ#)
- *Equipe*(equ#, nom_equ, resp_equ#)
- *Projet*(proj#, nom_proj, lieu_proj, equ#)
- *Travaille_sur*(id#, proj#, taux)
 - *Personne.equ#* : Equipe de rattachement
 - *Projet.equ#* : Equipe en charge du projet
 - *Travaille_sur.taux* : Part de temps sur un projet
- *Requête* :
Personnes nées après 1962 et travaillant sur le projet "BDD"

I. Transformation d'arbres algériques : Exemple

- En SQL :


```
SELECT nom
FROM   Projet, Travaille_sur, Personne
WHERE  Projet.nom_proj = 'BDD'
AND    Personne.id# = Travaille_sur.id#
AND    Personne.date_naissance > 'dec-31-1962'
AND    Travaille_sur.proj# = Projet.proj#
```
- *Représentation dite canonique* :
 - *Relations de la clause FROM* \longrightarrow Produit cartésien (X)
 - *WHERE* \longrightarrow Sélection (σ), nœud père du sous-arbre X
 - *SELECT* \longrightarrow Projection (Π) en racine de l'arbre

Transformation d'arbres algériques : Arbre initial (0)**Transformation d'arbres : Arbre 1**

- "Descente" des sélections

Transformation d'arbres : Arbre 2

- Permutation de Personne et de Projet (cf. notion de sélectivité)

Transformation d'arbres : Arbre 3

- Introduction de jointures

Transformation d'arbres : Arbre 4

- Introduction de projections : ne "faire remonter" que les attributs utiles

Transformation d'arbres : Sur l'arbre initial (Arbre 0)

- *Projet* : 20 tuples de 100 caractères
- *Travaille_sur* : 100 tuples de 50 caractères
- *Personne* : 500 tuples de 100 caractères
- Deux produits cartésiens : 10^6 tuples de 250 caractères !

Transformation d'arbres : Sur l'arbre final (Arbre 4)

- 1ère jointure :
 - Une relation à une colonne et probablement à un n-uplet
 - Relation *Travaille_sur* : deux colonnes
- 2ème jointure :
 - Une relation à une colonne (sous-arbre gauche)
 - Une relation à 2 colonnes (sous-arbre droit) réduite aux seuls n-uplets satisfaisant la sélection

Transformation d'arbres : Règles de transformation

- **(R1)** Décomposition des expressions de sélection.

$$Select(R, C_1 \wedge C_2 \wedge \dots \wedge C_{n-1} \wedge C_n) \longrightarrow$$

$$Select(Select(\dots Select(Select(R, C_n), C_{n-1}), \dots), C_2), C_1)$$
- **(R2)** Commutativité de la sélection.

$$Select(Select(R, C_2), C_1) \longrightarrow Select(Select(R, C_1), C_2).$$
- **(R3)** Restriction de la liste de projections.

$$Proj_{L1}(Proj_{L2}(\dots (Proj_{Ln}(R)) \dots)) \longrightarrow Proj_{L1}(R).$$
- **(R4)** Commutation de la sélection et de la projection.
 Si C ne porte que sur les A_i :

$$Proj_L(Select(R, C)) \longrightarrow Select(Proj_L(R), C)$$

Transformation d'arbres : Règles de transformation

- **(R5)** Commutativité de la jointure (et du produit cartésien).
 $Join_C(R, S) \rightarrow Join_C(S, R).$
- **(R6)** Commutation sélection-jointure (ou produit cartésien).
 - **(R61)** Si les attributs de la condition C de sélection n'appartiennent qu'à une des relations de la jointure, par exemple R , alors :
 $Select(Join_E(R, S), C) \rightarrow Join_E>Select(R, C), S).$
 - **(R62)** Sinon, si la condition C peut se ré-écrire en $C_1 \wedge C_2$ où :
 - C_1 ne porte que sur des attributs de R
 - $C_1 C_2$ ne porte que sur ceux de S , alors :
 $Select(Join_E(R, S), C) \rightarrow Join_E>Select(R, C_1), Select(S, C_2))$

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Transformation d'arbres : Règles de transformation

- **(R7)** Commutation projection-jointure (produit cartésien).
 - **(R71)** $Proj_L(Join_C(R, S)) \rightarrow Join_C(Proj_{L_1}(R), Proj_{L_2}(S))$
 1. Si attributs de C = attributs de L
 2. Si $L = L_1 || L_2$ avec $L_1 = Liste(A_i)$, A_i : attributs de R et $L_2 = Liste(B_j)$, B_j : attributs de S
 - **(R72)** Si la condition de jointure comporte des sous-listes L_3 d'attributs de R et L_4 d'attributs de S n'appartenant pas à L :
 1. Les ajouter aux projections "internes"
 2. Appliquer une projection "externe" sur L
 $Proj_L(Join_C(R, S)) \rightarrow Proj_L(Join_C(Proj_{L_1, L_3}(R), Proj_{L_2, L_4}(S)))$.

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Transformation d'arbres : Règles de transformation

- **(R8)** Commutativité des opérations ensemblistes (\cup , \cap).
- **(R9)** Associativité : jointure, produit cartésien, union et intersection.
 $((R op_i S) op_j T) \rightarrow (R op_j (S op_i S))$ où R, S, T sont des relations et op_i, op_j des opérateurs parmi ceux cités
- **(R10)** Commutation de la sélection et des opérations ensemblistes.
 $(Select((R op S), C)) \rightarrow (Select(R, C) op Select(S, C))$ où op est l'opérateur d'union, d'intersection ou de différence

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Transformation d'arbres : Règles de transformation

- **(R11)** Commutation de la projection et des opérations ensemblistes.
 $(Proj_L(R op S)) \rightarrow ((Proj_L(R)) op (Proj_L(S)))$.
- Autres :
 - Equivalences logiques
 - Autres propriétés des opérateurs algébriques

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

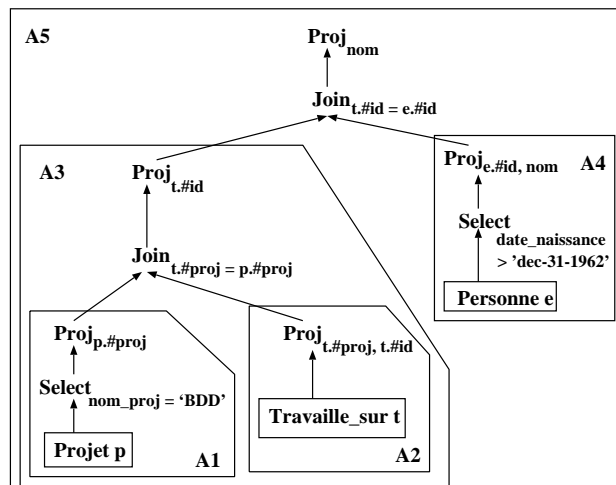
Transformation d'arbres : Ebauche d'algorithme

1. Transformer toute expression de sélection conjonctive en une "cascade" de sélections (Règle R1)
2. "Descendre" les sélections le plus bas possible ((R2), (R4), (R6), (R10)).
3. Ordonner les feuilles de l'arbre de sorte que les sélections les plus restrictives soient évaluées en premier ((R8), (R9))
 - "Plus restrictives" = produisant les plus petites relations
 - cf. distribution des valeurs, index (dictionnaire)
 - cf. notion de sélectivité, plus loin

Transformation d'arbres : Ebauche d'algorithme

4. (Produit cartésien; Sélection) \rightarrow Jointure où la condition de jointure est la condition de sélection.
5. "Fragmenter" et "faire descendre" le plus bas possible les listes de projection, en créant de nouvelles projection, si besoin est (Règles (R3), (R4), (R7), (R11)),
6. Identifier les sous-arbres qui représentent des groupes d'opérations pouvant être exécutés par une seule routine du SGBD

Transformation d'arbres : Exemples de plan (Etape 6)



Traitement des requêtes : Plan

1. Transformation d'arbres algébriques
2. Transformation de graphes de requêtes
3. Optimisation à base de coûts
4. Optimisation Sémantique
5. Processus Général d'optimisation

II. Optimisation de graphes de requêtes

- Technique dite de décomposition de requêtes
- Introduite dans Ingres/QUEL (Variables n-uplets)
- Grappe de requêtes :
 - Sur les arcs : Conditions de jointure des nœuds reliés
 - Nœuds : Variables de tuples ou constantes de la requête
 - Arcs "conditions de sélection" : relie des nœuds de constantes aux nœuds des variables impliquées dans les conditions.
- Heuristique (sélection, projection, jointure seulement) : exécuter les sélections avant les jointures (ou le produit cartésien) en identifiant des sous-requêtes à une seule variable et une condition de sélection

II. Optimisation de graphes de requêtes

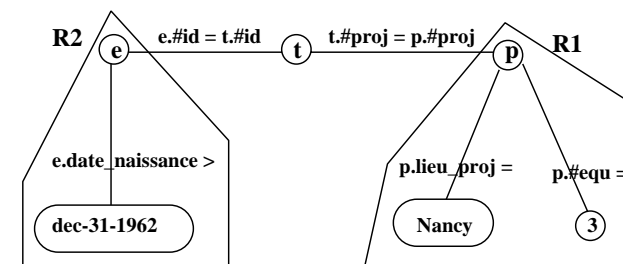
- Technique : Requête à plusieurs variables décomposée en sous-requêtes à une variable, par *détachement* de sous-graphes et *substitution de tuples*.
- 1. Détachement :
 - Identifier des sous-requête ayant une seule variable en commun avec le reste de la requête
 - Les détacher du graphe
- 2. Substitution de tuples :
 - Evaluation de sous-requête
 - Substituer, un tuple à la fois, de valeurs aux variables de la requête.
- Requête à n variables \rightarrow m requêtes, plus simples, à (n-1) variables

II. Optimisation de graphes de requêtes : Exemple

"Personnes nées après 1962 et travaillant sur des projets se déroulant à Nancy dans l'équipe numéro 3"

RANGE OF p IS Projet, t IS Travaile_sur, e IS Personne
 RETRIEVE e.nom
 WHERE p.lieu_proj = 'Nancy' AND p.equ# = 3
 AND p.proj# = t.proj# AND t.id# = e.id#
 AND e.date_naissance > 'dec-31-62'

II. Optimisation de graphes de requêtes : Exemple



Traitement des requêtes : Plan

1. Transformation d'arbres algébriques
2. Transformation de graphes de requêtes
3. Optimisation à base de coûts
4. Optimisation Sémantique
5. Processus Général d'Optimisation

III. Optimisation par estimation de coûts

- Minimiser une fonction coût, avec comme facteurs :
 - Accès disque,
 - Coût du traitement en mémoire centrale,
 - Taille des n-uplets, des relations,
 - Index : Existence, nombre de niveaux,
 - Coût de communication (requête, résultats) entre sites (architectures client/serveur, par exemple),
 - etc.
- Utilisation effective (souvent) des coûts des accès

III. Optimisation par estimation de coûts

- Informations du dictionnaire
 - Nombre exact ou estimé de tuples et de blocs physiques par relation ;
 - Nombre de niveaux d'index ;
 - Nombre de valeurs distinctes par attribut, dans les cas où un index ou un cluster existe : permet d'estimer le nombre moyen d'enregistrements qui satisfont une sélection (sélectivité ou cardinal de sélection d'un attribut A).
- * A est clé : $s = 1$
- * A n'est pas clé : $s = Card(R)/\text{Nombre de valeurs de A}$.

III. Optimisation par estimation de coûts

- Sélectivité de la jointure : $sj = Card((R \bowtie_{Cond} S))/Card(R \times S)$
 - $sj = 1$ si pas de condition de jointure
 - $sj = 0$ si aucun tuple ne vérifie *Cond*.
- Cas de l'équi-jointure : (Condition du type $R.A = S.B$) Cas particuliers :
 1. A est clé de R : $Card(R \bowtie S) \leq Card(S)$
 - $sj \leq Card(S)/Card(R) \times Card(S)$
 - $sj \leq 1/Card(R)$
 2. De même, si B est clé de S : $sj \leq 1/Card(S)$.
- D'où la taille estimée de $R \bowtie S$: $sj \times Card(R) \times Card(S)$.

Exemples d'estimation de l'équi-jointure

- br : nombre de blocs de R,
- bs : nombre de blocs de S
- k : facteur de blocage de la relation résultat

1. Jointure par boucles imbriquées

- R dans la boucle extérieure ; sj donné ; 2 buffers
- Coût estimé : $br + (br * bs) + ((sj * Cardinal(R) * Cardinal(S))/k)$
- Dernier facteur : Coût de l'écriture du résultat.

Exemples d'estimation de l'équi-jointure

2. Jointure par tri-fusion

- Si relations déjà triées : $Coût = br + bs$
- Sinon, $Coût = br + bs + l * (br * \log_2 br + bs * \log_2 bs)$.
 - $(l * b * \log_2 b)$: Approximation du tri
 - b : nombre de blocs
 - l : facteur constant.

IV. Optimisation sémantique

- Contrainte d'intégrité Formule logique CI
- Expression de sélection E
- Pas d'accès à la base si $\neg(E \wedge CI)$
- Exemple :
 - CI : Tous les vols long courrier partent de Roissy
 - E : Vols long courrier partant d'Orly ?
- Méthode non implantée (démonstration automatique)

Optimisation de requêtes : SGBD Sybase

- Génération de plans avec/sans exécution de requête
- Exemple : Visualisation d'un plan sans exécution


```
SET SHOWPLAN ON
SET NOEXEC ON
Requête dont on veut examiner le plan.
```
- Plans d'exécution et procédures stockées :
 - Exécution avec *WITH RECOMPILE* : MàJ du plan stocké
 - Création avec *WITH RECOMPILE* : Génération systématique
- Mise à jour des informations sur la distribution des valeurs des clés des index :

UPDATE STATISTICS NomDeRelation [NomIndex]

Optimisation de requêtes : SGBD Oracle

- Commande *EXPLAIN PLAN* : obtenir le plan d'exécution
- Plan rangé dans *PLAN_TABLE* ou dans une relation créée préalablement à l'exécution de *EXPLAIN PLAN* (clause *INTO* de *EXPLAIN PLAN*)

EXPLAIN PLAN

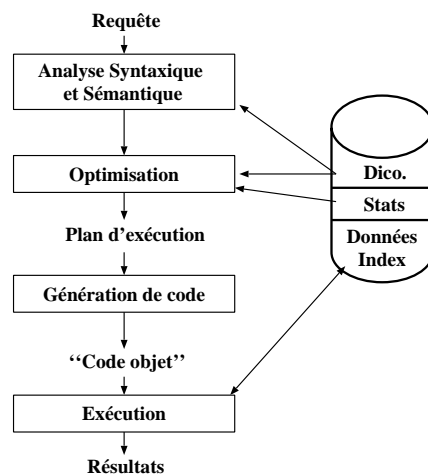
[*SET STATEMENT_ID* = identification_plan]

[*INTO* [nom_utilisateur.]nom_de_relation] *FOR* requête_SQL

Traitement des requêtes dans les SGBDR : Plan

1. Transformation d'arbres algébriques
2. Transformation de graphes de requêtes
3. Optimisation à base de coûts
4. Optimisation Sémantique
5. Processus Général d'Optimisation

V. Optimisation : Processus général



Traitement des requêtes : Processus général

1. Normalisation des prédicats
2. Analyse sémantique
3. Simplification des formules logiques
4. Mise sous forme d'arbre relationnel

1/4) Normalisation des prédicats

- Forme normale conjonctive : conjonction de disjonctions
 $(p_1 \vee p_2 \vee \dots \vee p_n) \wedge \dots \wedge (q_1 \vee \dots \vee q_m)$
- Forme normale disjonctive : disjonction de conjonctions
 $(p_1 \wedge p_2 \wedge \dots \wedge p_n) \vee \dots \vee (q_1 \wedge \dots \wedge q_m)$
- Formules sans quantificateurs :
 - Commutativité, Associativité de la conjonction et de la disjonction
 - Distributivité (\vee , \wedge), (\wedge , \vee)
 - $\neg(p_1 \wedge p_2) \longleftrightarrow \neg p_1 \vee \neg p_2$
 - $\neg(p_1 \vee p_2) \longleftrightarrow \neg p_1 \wedge \neg p_2$
 - $\neg(\neg p) \longleftrightarrow p$
- Formules avec quantificateurs : mise sous forme *prenex*

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

- Forme disjonctive :
 - Requête traitée comme une union de sous-requêtes
 - Risque de calcul redondant
- Forme conjonctive : généralement plus de "ET" que de "OU"
- Exemple :


```
select libelle from produit p, stock s
where p.prod# = s.prod# and s.adr = "Nancy"
and (s.qte = 1000 or s.qte=200)
```

 - Forme disjonctive :
 $(p.prod\# = s.prod\# \wedge s.adr = \text{"Nancy"} \wedge s.qte = 1000) \vee (p.prod\# = s.prod\# \wedge s.adr = \text{"Nancy"} \wedge s.qte = 2000)$
 - Forme conjonctive :
 $p.prod\# = s.prod\# \wedge s.adr = \text{"Nancy"} \wedge (s.qte = 1000 \vee s.qte=2000)$

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

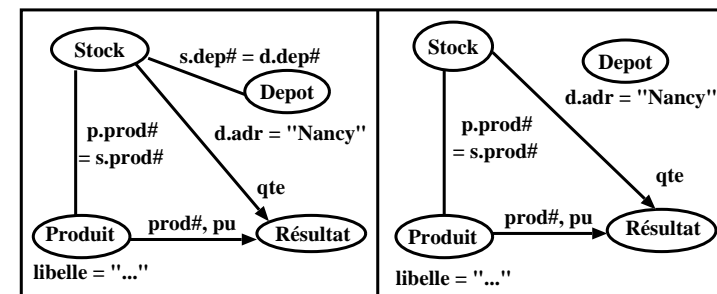
2/4) Analyse

- Dictionnaire : Relations, attributs connus
- Typage des expressions
- Correction sémantique : pas de sous-requête isolée
 - Calcul relationnel : impossible à déterminer
 - Requêtes sans \vee ni \neg : graphe connexe
- Graphe de requête :
 - Nœud : résultat ou opérande
 - Arcs entre nœuds non résultats : Jointure
 - Arcs d'extrémité nœud résultat : Projection
 - Nœud non résultat peut être labellé par une expression de sélection ou une auto-jointure

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

- Exemple :


```
select p.prod#, p.pu, s.qte
from produit p, stock s, depot d
where p.prod# = s.prod# and s.dep# = d.dep#
and s.qte >= 0 and d.adr = "Nancy" and libelle = "..."
```



©Nacer.Boudjlida@loria.fr, UHP Nancy 1

3/4) Simplification d'expressions logiques

- *Cas de l'utilisation de vues*

create view V

as select * from produit
where pu > 100.0 and pu < 200.0

- *Requête :*

select * from V
where pu < 200.0

- *Après ré-écriture :*

select * from produit
where pu > 100.0
and pu < 200.0 and pu < 200.0

3/4) Simplification d'expressions logiques (suite)

- *Elimination de la redondance* : règles d'idempotence

$$- p \wedge p \longleftrightarrow p; p \vee p \longleftrightarrow p$$

$$- p \wedge \text{Vrai} \longleftrightarrow p; p \wedge \text{Faux} \longleftrightarrow \text{Faux}$$

$$- p \vee \text{Vrai} \longleftrightarrow \text{Vrai}; p \vee \text{Faux} \longleftrightarrow p$$

$$- p \wedge \neg p \longleftrightarrow \text{Faux}; p \vee \neg p \longleftrightarrow \text{Vrai}$$

$$- p_1 \wedge (p_1 \vee p_2) \longleftrightarrow p_1$$

$$- p_1 \vee (p_1 \wedge p_2) \longleftrightarrow p_1$$

3/4) Simplification d'expressions logiques (suite et fin)

- *Exemple :*

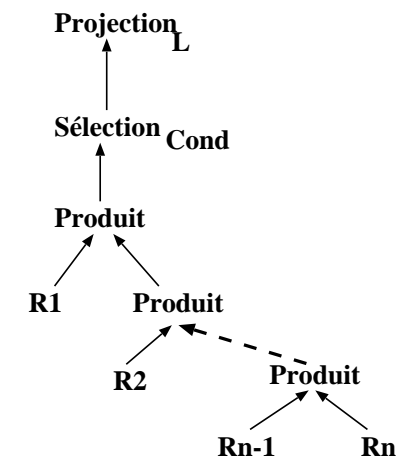
select libelle from produit
where NOT libelle = "K7Cr"
and (libelle = "K7Cr" or pu = 20.0)
and NOT pu = 20.0
and prod# = 4

se simplifie en :

select libelle from produit
where prod# = 4

4/4) Mise sous forme d'arbre algébrique

SELECT L
FROM R1, R2, ..., Rn
WHERE Cond



Chapitre IV : Traitement des requêtes

- Contenu du chapitre :
 1. Traitement des requêtes dans les SGBDR (p. 157)
 2. [Application à Oracle](#) (p. 206)

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. [Architecture du compilateur de requêtes et structure de stockage des plans d'exécution](#) (p. 209)
2. Processus et principales étapes de traitement des requêtes (p. 221)
3. Optimisation à base de coûts (p. 239)
4. Optimisation à base de règles (p. 252)
5. Directives d'optimisation (p. 255)

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 \rightarrow 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 :


```
select p.libelle, s.qte from produit p, stock s
where p.prod# = s.prod#;
```
- 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 ... 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

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. 211

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

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

- analyze ... compute | estimate statistics* : Collecte de statistiques
- SQL Trace* :
 - Autoriser son utilisation : *sql_trace* (paramètre statique d'initialisation)
 - Pour une instance : *sql_trace = true* dans le fichier d'initialisation
 - Pour une session : *alter session set sql_trace = true*
 - 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)
- tkprof* : formatage des fichiers produits par *SQL Trace*
 - Production d'un fichier par fichier de trace,
 - Fichier formaté en fonction de paramètres et options d'exécution
 - Conservation éventuelle de statistiques dans *tkprof.table* (créée à la demande).

Traitement des requêtes sous Oracle : Plan

- Architecture du compilateur de requêtes et structure de stockage des plans d'exécution (p. 209)
- Processus et principales étapes de traitement des requêtes (p. 221)
- Optimisation à base de coûts (p. 239)
- Optimisation à base de règles (p. 252)
- Directives d'optimisation (p. 255).

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

1. Simplification et Evaluation d'expressions (cf. Processus général)
2. Réécriture de requêtes complexes (p. 223)
3. Réécriture de requêtes avec vues (p. 228)
4. Choix de la stratégie d'optimisation (p. 229)
5. Traitement des jointures (p. 236)

2.1/5- 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 \rightarrow 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.

2.1/5.- Réécriture de requêtes et index (Exemple)

select ... where A = Constante1 or B = Constante2

réécrite en :

(select ... where A = Constante1)

union (select ... where B = Constante2)

Si index sur A et B.

2.1/5.- Réécriture de requêtes et Sous-requêtes (Exemple)

- Traiter la requête sous sa forme initiale
- ou désimbriquer la sous-requête en introduisant des jointures dans la requête initiale.
- Exemple : Si contrainte de clé primaire ou d'unicité sur $R2.x$

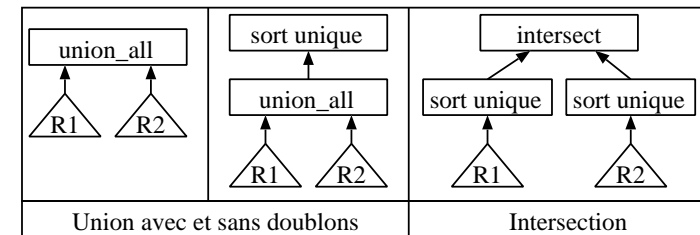
```
select ... from R1
where x in (select x from R2 where condition);
```

réécrite en :

```
select ... from R1, R2
where condition and R1.x = R2.x.
```

2.1/5- Requêtes composées (Exemple)

- 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)

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

1. Simplification et Evaluation d'expressions (cf. Processus général)
2. Réécriture de requêtes complexes (p. 223)
3. Réécriture de requêtes avec vues (p. 228)
4. Choix de la stratégie d'optimisation (p. 229)
5. Traitement des jointures (p. 236)

2.2/5- Traitement des requêtes avec vues

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

“select x1 from V where Condition2”

et “create view V as (select x1, ... from R where Condition1)”

→ “select x1 from R where Condition1 and Condition2”

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

- Quand échec de l'intégration de la vue à la requête.

2.3/5- 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

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

2.3/5- 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*

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

2.3/5- Paramétrisation : Valeurs de *optimizer_mode* et *optimizer_goal***2.3/5- 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

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

2.3/5- 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

Rappels : Statistiques

- Collecte par :
 - *analyze ... {compute | estimate} statistics*
 - *{create | alter} index ... "compute statistics"*
 - Paquetage *dbms_stats*
- Mises à jour périodiques explicites
- Si absence totale de statistiques : utilisation possible de données quantitatives sur l'encombrement des espaces (nombre de blocs occupés, nombre de niveaux d'index, etc.)

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

1. Simplification et Evaluation d'expressions (cf.processus général)
2. Réécriture de requêtes complexes (p. 223)
3. Réécriture de requêtes avec vues (p. 228)
4. Choix stratégie d'optimisation ? (p. 229)
5. Traitement des jointures (p. 236)

2.5/5- Traitement des jointures

- Combinaison des facteurs :
 - (a) Type d'algorithme de jointure
 - (b) Ordre des jointures
 - (c) Chemins d'accès

2.5/5- Algorithme et ordre 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*).
- Ordre de jointures (\forall 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)

Traitement des requêtes sous Oracle : Plan

1. Architecture du compilateur de requêtes et structure de stockage des plans d'exécution (p. 209)
2. Processus et principales étapes de traitement des requêtes
3. Optimisation à base de coûts (p. 239)
4. Optimisation à base de règles (p. 252)
5. Directives d'optimisation (p. 255)

3/5. Optimisation à base de coûts

- Conditions de mise en œuvre :
 1. Disponibilité de statistiques
 2. Paramètres d'initialisation positionnés :
 - (a) *optimiser_mode = choose, first_rows* ou *all_rows*
 - (b) *optimiser_features_enable = 8.1.6*
 - (c) *compatible = 8.1.6*
- *Note* : Consulter les paramètres en vigueur dans *v\$parameter*
- Exemple :


```
select name, value from v$parameter
where name like 'optimizer%'
```

3/5. Optimisation à base de coûts

- Stratégie généralement requise pour les instructions qui utilisent :
 - des tables partitionnées et/ou organisées en index,
 - des index basés sur des fonctions.
- 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. 243)
 2. Transformation de requêtes (p. 245)
 3. Estimation de coûts (p. 248)
 4. Génération de plans (p. 250)

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.

3/5. Optimisation à base de coûts

- La suite :
 1. Chemins d'accès (p. 243)
 2. Transformation de requêtes (p. 245)
 3. Estimation de coûts (p. 248)
 4. Génération de plans (p. 250)

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 \rightarrow 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. 243)
 2. Transformation de requêtes (p. 245)
 3. Estimation des coûts (p. 248)
 4. Génération de plans (p. 250)

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*".

3/5. Optimisation à base de coûts

- La suite :
 1. Chemins d'accès (p. 243)
 2. Transformation de requêtes (p. 245)
 3. Estimation des coûts (p. 248)
 4. Génération de plans (p. 250)

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

1. Un sous-plan :
 - pour chaque sous-requête désimbriquée
 - pour chaque vue non intégrée à la requête.
2. Génération et optimisation du plan global de bas en haut (du bloc de requête le plus interne vers la requête)
3. 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. 209)
2. Processus et principales étapes de traitement des requêtes
3. Optimisation à base de coûts (p. 239)
4. Optimisation à base de règles (p. 252)
5. Directives d'optimisation (p. 255)

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

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Traitement des requêtes sous Oracle : Plan

1. Architecture du compilateur de requêtes et structure de stockage des plans d'exécution (p. 209)
2. Processus et principales étapes de traitement des requêtes
3. Optimisation à base de coûts (p. 239)
4. Optimisation à base de règles (p. 252)
5. [Directives d'optimisation](#) (p. 255)

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

5/5. Directives d'optimisation

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

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

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* → 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*

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

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 ↗ performances

Eléments d'Architecture des SGBDR : Conclusion

- Administrer \simeq :
 - Installer (Serveur(s) SQL et Serveur(s) Back-up)
 - Gérer et contrôler les espaces (disque, mémoire) et les connexions
 - Allouer rôles et privilèges aux utilisateurs
 - Sauvegarder/Restaurer les bases
 - Diagnostiquer les problèmes système
 - Configurer le système pour de meilleures performances
- Possible interférence avec les concepteurs pour normalisation
 - définition de données,
 - politique de maintien de l'intégrité, etc.

Eléments d'Architecture des SGBDR : Conclusion

- 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
- En savoir plus :
 - Ouvrages référencés
 - Master 2, Informatique, Spécialité Ingénierie du logiciel

Architecture et Administration des SGBDR

- Chapitre 1** Installation et lancement de serveurs
- Chapitre 2** Organisation et stockage de données
- Chapitre 3** Création de bases de données
- Chapitre 4** Gestion des utilisateurs
- Chapitre 5** Sécurité de fonctionnement et reprises
- Chapitre 6** Traitement des requêtes
- Chapitre 7** Adaptation de serveurs et performances

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

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. – *Oracle 8i, Administrator's guide*, December 1999. Release 2(8.1.6), Part No A76956-01.
- [6] Oracle-Corp. – *Oracle 8i, Backup and Recovery Guide*, December 1999.

©Nacer.Boudjlida@loria.fr, UHP Nancy 1

Release 2(8.1.6), Part No A76993-01.

- [7] Oracle-Corp. – *Oracle 8i Concepts*, December 1999. Release 2(8.1.6), Part No A76965-01.
- [8] Oracle-Corp. – *Oracle 8i, Designing and Tuning for performance*, December 1999. Release 2(8.1.6), Part No A76992-01.
- [9] Oracle-Corp. – *Oracle 8i, Recovery Manager User's Guide*, December 1999. Release 2(8.1.6), Part No A76990-01.
- [10] Oracle-Corp. – *Oracle 8i Reference*, December 1999. Release 2(8.1.6), Part No A76961-01.
- [11] Oracle-Corp. – *Oracle^(R) Universal Installer Concepts Guide*, October 1999. Release 1.7.
- [12] Oracle-Corp. – *Oracle 8i, SQL Reference*, September 2000. Release 3(8.1.7), Part No A85397-01.
- [13] Oracle-Corp. – *SQL*Plus User's Guide and Reference*, September 2000. Release 8.1.7, Part No A82950-01.

©Nacer.Boudjlida@loria.fr, UHP Nancy 1