Chapitre III : Organisation et stockage des données

Contenu du chapitre :

I- Organisation et Stockage des données

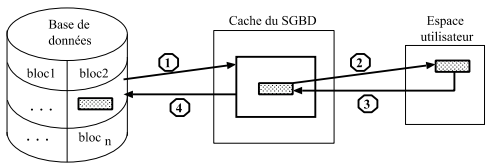
- Comprendre :
  1. Stockage et accès sans index
  2. Index : Organisation, structuration, accès
- Pour savoir :
  1. Gérer les espaces ;
  2. Forme à donner aux requêtes pour maximiser l'utilisation des index ?
  3. Utiliser les outils du serveur pour examiner les choix de son optimiseur.

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



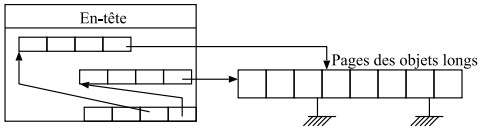
Structure générale des pages

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

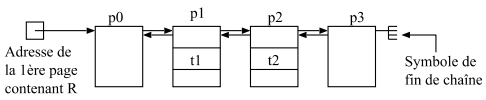


4. Organisation “en tas” (*heap*)

- 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. $\prec$ taille(ancien) :
     (a) Modification du tuple et “décalage” vers le haut ;
     (b) Modification des pointeurs de tuples.
  3. $\succ$ 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*

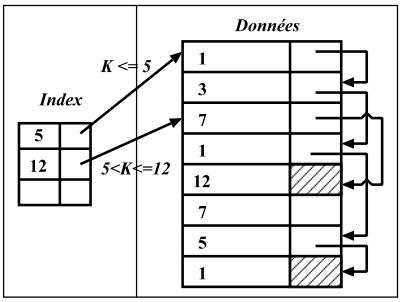
- Une entrée dans l'index par valeur de la clé
- Entrée : $< V_i,\ P_i >$ où
  - V_i : valeur d'une clé d'index C
  - P_i : tête d'une liste d'adresses de tuples
- Si Clé d'index = clé de la relation : pas de chaînage
- Rangement des tuples pas nécessairement contigu
- Exemple :

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*

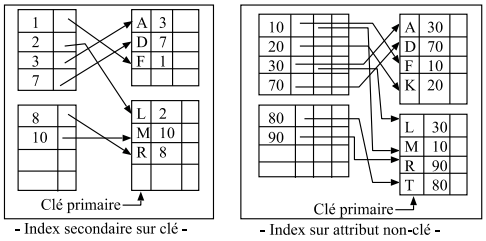
- Défini sur une clé primaire de relation
- Valeurs ordonnées dans la relation
- Ordre logique = ordre physique.
- Exemple :

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 : › 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*)
- [redacted]
- 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

[redacted]

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)

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*

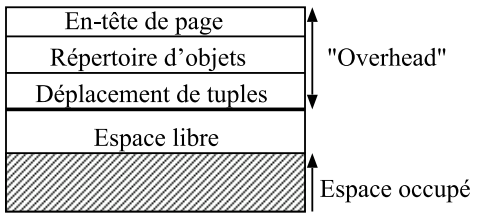
3. Lecture seule (*read only*) :
   - Par défaut, *tablespace* en lecture/écriture
   - Mise en lecture seule provisoire ou définitive
     - *alter tablespace . . . read only*
     - Changement d'état effectif à l'issue des transactions actives
     - Seules les opérations de consultation et de suppression 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

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*.

[redacted]

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 [redacted]
  2. Index [redacted]
  3. Temporaires [redacted]
  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

[redacted]

- 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*

[redacted]

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
```


5.1.2 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"*).

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

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.