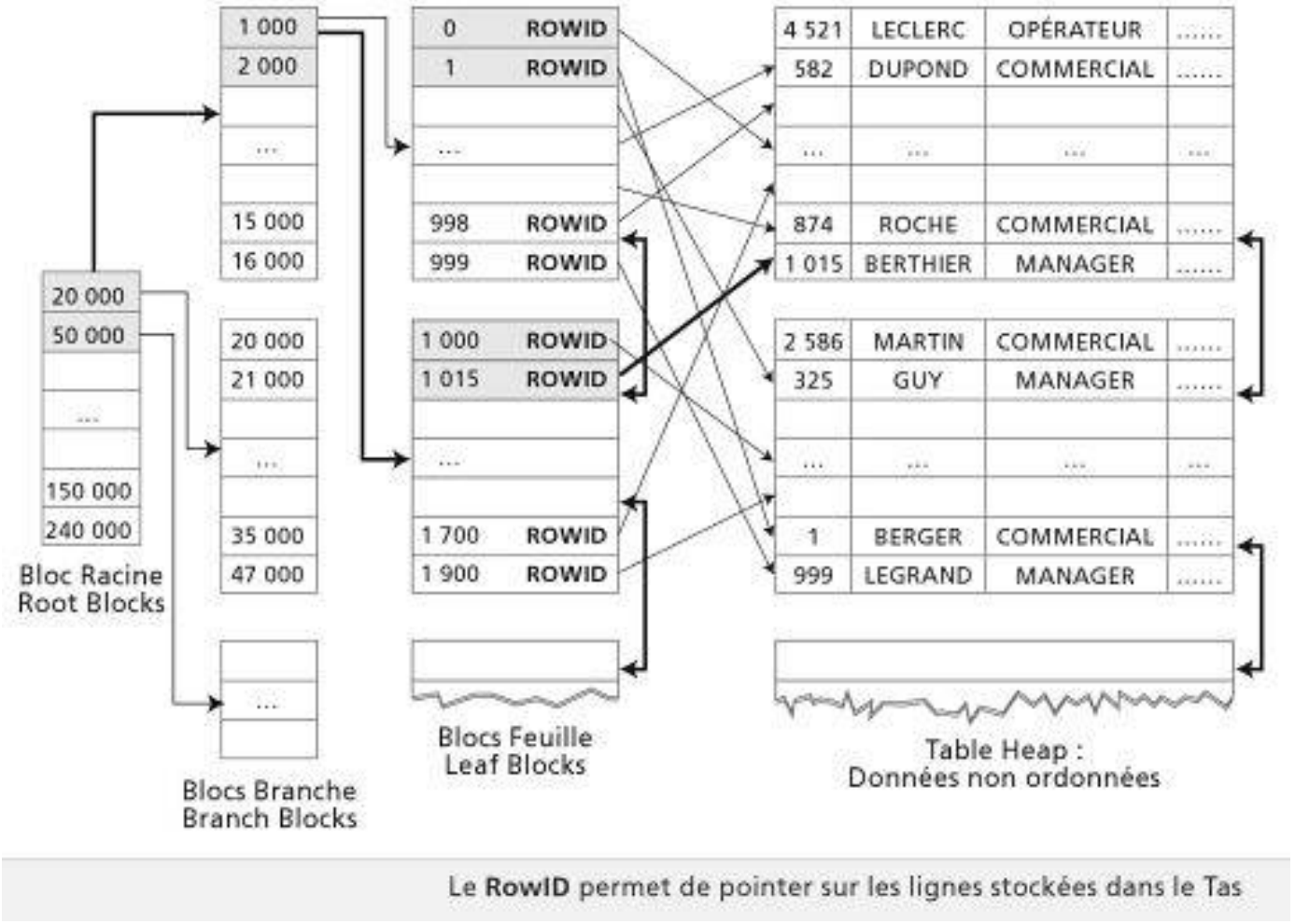
Index

Références

- Oracle 11g administration. Olivier Heurtel. Eni Eds. Collection Ressources Informatiques. Novembre 2008. (organisation du stockage sous Oracle)
- SQL - 4e édition 2012. Frédéric Brouard, Christian Soutou, Rudi Bruchez. Collection Synthex Informatique. Pearson. Juillet 2012
- Optimisation des bases de données - Mise en œuvre sous Oracle. Laurent Navarro. Pearson. Juin 2010
- SQL : Au cœur des performances. Markus Winand. 2013

Structure d'un index B-tree



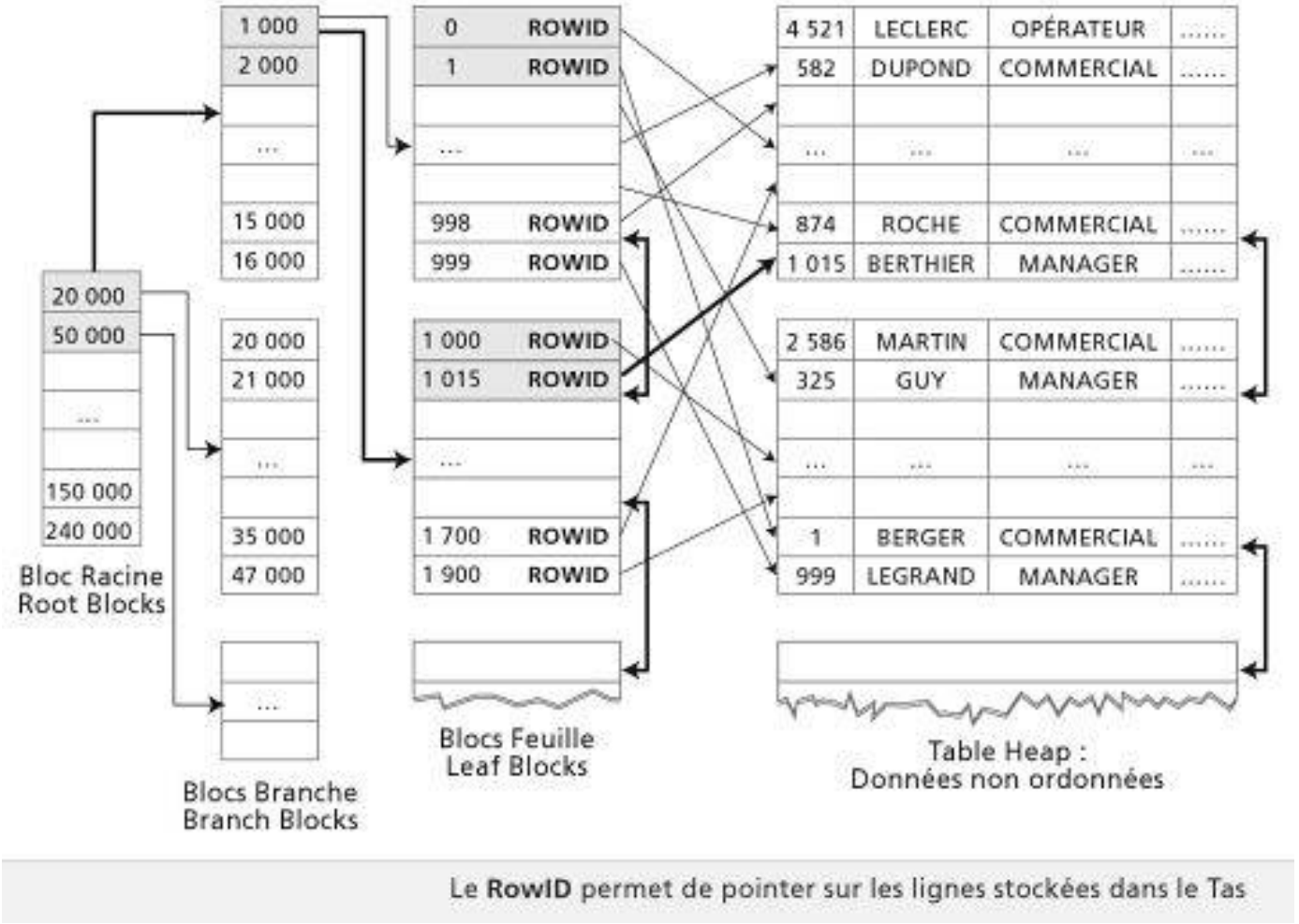
Extrait du livre « Optimisation de bases de données » - Laurent Navarro

Obtention d'une ligne

Obtenir la ligne ayant pour clé 1015 dans l'index B-Tree :

- Parcours du bloc racine jusqu'à rencontrer la valeur supérieure
- Aller sur le bloc pointé par la valeur qui précède
- Répéter l'opération dans le bloc branche
- Dans le bloc feuille, rechercher la valeur (arrêt à la valeur supérieure si la valeur n'est pas présente), l'entrée pointée permet d'obtenir le ROWID de la ligne
- Le ROWID est obtenu dans le « tas »

Obtention d'une ligne



Extrait du livre « Optimisation de bases de données » - Laurent Navarro

=> seulement 3 blocs de données à lire

Ne pas utiliser d'index quand

- La majorité des requêtes ramènent plus de 5% des lignes d'une table
- Énormément de mises à jour par rapport au nombre de lectures
- Colonne pas utilisée dans les clauses WHERE des requêtes et dans les JOIN
- Un nombre de lignes faible dans votre table

Quels index créer ?

- Clé primaire : la plus courte possible, le moins de colonnes possible, stable (sans sémantique)
- Clé étangères
- Les colonnes les plus recherchées

Utilité d'un prédicat

Utilisation d'un index de manière efficace, si la condition de recherche permet d'utiliser l'index.

Exemple : index sur `ename` de `emp`

Conditions de la clause WHERE entraînant une utilisation efficace de l'index (Extrait du livre SQL - 4e édition 2012. Brouard, Soutou, Bruchez. Pearson. Juillet 2012)

Condition	**Méthode de recherche**
ename = 'DUPONT'	Recherche directe
ename < 'DUPONT'	Va à l'emplacement de DUPONT et parcourt toutes les occurrences en arrière
ename LIKE 'DUP%'	Va au premier nom commençant par DUP et parcourt en avant les suivants ayant même caractéristique
ename LIKE 'DU%T'	Va au premier nom commençant par DU et parcourt en avant les suivants ayant même caractéristique pour ne retenir que ceux finissant par T
ename LIKE 'DU %' AND ename LIKE '%T'	Même méthode qu'au dessus

Utilité d'un prédicat

Conditions du WHERE qui ne vont pas utiliser l'index

(Extrait du livre SQL - 4e édition 2012. Brouard, Soutou, Bruchez. Pearson. Juillet 2012)

Condition	**Méthode de recherche**
ename <> 'DUPONT'	Lecture séquentielle ne prenant pas en compte les DUPONT
ename NOT LIKE 'DUP %'	Lecture séquentielle ne prenant pas en compte les noms commençant par DUP
ename LIKE '%T'	Lecture séquentielle prenant en compte les noms se terminant par T
ename LIKE ' %DU%'	Lecture séquentielle prenant en compte les noms contenant DU
ename IN ('DUPONT','DURANT')	Lecture séquentielle prenant en compte les noms de la liste
length(ename)=6	Lecture séquentielle du nom, application de la fonction et vérification d'égalité
ename || ', ' || job='Dupont François'	Lecture séquentielle du nom et du job, puis concaténation et vérification d'égalité

Qualité d'un index

Extrait du livre SQL - 4e édition 2012. Frédéric Brouard, Christian Soutou, Rudi Bruchez. Collection. Synthex Informatique. Pearson. Juillet 2012

- Index 1 étoile : rend le filtre WHERE de la requête cherchable
- Index 2 étoiles : index 1 étoile qui couvre la clause SELECT
- Index 3 étoiles : index 2 étoiles qui assure le tri
- Index 4 étoiles : index 3 étoiles qui assure le regroupement (clause GROUP BY)

Exemple index 1 étoile

```
SELECT *
FROM bigemp
WHERE ename='DUVAL'
   AND job='ANALYST'
```

Index :

```
CREATE INDEX idx_emp_enamejob
   ON emp(ename, job);
```

Plan d'exécution

Plan d'exécution

- Liste des opérations établie par le SGBDR pour répondre à une requête de la manière estimée la plus efficace (un arbre algébrique)
- Estimation basée sur des statistiques
- Réduire les physical reads et les sorts (disks)
- Possibilité d'imposer un autre chemin : les hints (à utiliser avec beaucoup de précautions !)

Exemple de plan d'exécution

```
SELECT empno
FROM bigemp
WHERE empno=100000;
```

Nombre d'octets qu'Oracle pense transférer

Estimation du nombre de lignes retournées

Estimation du nombre de blocs accédés. N'est pas un facteur fiable de coût réel

```
-------------------------------------------------------------------------------
| Id  | Operation         | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |            |     1 |     6 |     2   (0)| 00:00:01 |
|*  1 |  INDEX UNIQUE SCAN| PK_BIGEMP  |     1 |     6 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("EMPNO"=100000)
```

Liste des opérations, à lire de bas en haut

Les opérations notées avec * sont précisées ici

Sous SQL*Plus

- Mesurer le temps d'exécution :

```
set timing on;
```

- Afficher les statistiques et le plan d'exécution d'une requête :

  activation du mode autotrace

```
set autotrace on;
```

  Ou

```
explain plan for (requete);
select plan_table_output from
table(dbms_xplan.display());
```

AUTOTRACE

```
SQL> set autotrace traceonly stat;

SQL> select * from bigemp where empno>2000000;

520098 ligne(s) sélectionnée(s).

Statistiques

----------------------------------------------------------
          18  recursive calls
           0  db block gets
       37865  consistent gets
           0  physical reads
           0  redo size
    26916354  bytes sent via SQL*Net to client
      381767  bytes received via SQL*Net from client
       34675  SQL*Net roundtrips to/from client
           0  sorts (memory)
           0  sorts (disk)
      520098  rows processed

SQL>
```

Statistiques réelles du plan d'exécution

- `recursive calls` : nb d'instructions pour mettre à jour les tables internes (comprend construction du plan, allocations mémoire pour les tris, …)
- `db block gets` : nb de lectures de blocs en mémoire en mode « courant » (l'état du bloc est le plus à jour) (`update` et `delete` ont besoin d'y accéder pour effectuer les mises à jour)
- `consistent gets` : nb de lectures en mémoire en mode « consistant ». Représentatif du nombre de pages disque lues lorsque toute les données sont sur disque et que le cache est vide
  L'addition de consistent gets et db block gets donne le nombre de blocs logiques lus
- `physical reads` : nb de lectures (en nb de blocs) effectives sur le disque
- `redo size` : nb d'octets générés pour le journal, permettant de refaire l'opération

Statistiques réelles du plan d'exécution

- `bytes sent via SQL*Net to client` : octets envoyés sur le réseau du serveur au client
- `bytes received via SQL*Net from client` : octets envoyés sur le réseau du client au serveur
- `SQL*Net roundtrips to/from client` : nb d'échanges réseau entre le client et le serveur
- `sorts (memory)/(disk)` : nb de tris effectués complètement en mémoire/ayant requis au moins un accès disque
- `rows processed` : nb de lignes retournées

Accès aux tables et aux index

```
SELECT empno
FROM bigemp
WHERE empno=100000;
```

```
-------------------------------------------------------------------------------
| Id  | Operation         | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |            |     1 |     6 |     2   (0)| 00:00:01 |
|*  1 |  INDEX UNIQUE SCAN| PK_BIGEMP  |     1 |     6 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("EMPNO"=100000)
```

Des opérations du plan d'exécution

Opération	Option	Description
table access		type d'accès à une table
	full	obtention de toutes les lignes
	by rowid	obtention des lignes en se basant sur les valeurs ROWID
index		utilisation de l'index
	unique	parcourt l'arborescence de l'index pour localiser une clé unique
	range scan	parcourt une partie de l'index de façon ordonnée : accès à des entrées adjacentes de l'index
	full scan	parcourt l'index en entier en respectant l'ordre de l'index
	fast full scan	parcourt l'index en entier sans respecter l'ordre des données

Accès à la table

- Full Table Scan : toutes les lignes de la table sont lues en mémoires et sont filtrées en fonction des restrictions de la clause WHERE → coûteux pour une grosse table car nombreux accès disque
  Choisi si
  - une large portion des lignes de la table doit être accédée
  - aucun index n'existe
  - les index présents ne peuvent être utilisés
  - le coût en utilisant un index est plus important

Accès à la table

- Index Scan/Table Access by Rowid : utilisation de l'index pour décider quelles lignes sont lues en mémoire, puis ces lignes sont récupérées en utilisant le Rowid de l'index

Utilisation de l'index

- Index Unique Scan : une seule ligne sera retournée par le parcours d'un index unique
  - Choisi en cas de condition d'égalité sur un index unique ou un index issu de la création d'une clé primaire

- Index Range Scan : parcourt une partie de l'index de façon ordonnée : accès à des entrées adjacentes de l'index
  - Choisi pour des requêtes sélectives, en cas de condition d'égalité sur un index non unique ou de non égalité/inférieur/supérieur/like sur un index unique

Utilisation de l'index

- Full Index Scan : parcourt l'index en entier en respectant l'ordre de l'index
  - Choisi quand toutes les colonnes nécessaires pour la requête sont dans l'index (toutes les colonnes (ou éventuellement une partie) d'un order by/group by sont dans l'index et dans le même ordre, …)
- Fast Full Index Scan : parcourt l'index en entier sans respecter l'ordre des données (par exemple avec `mod(empno,2)=0`)
- Plus de détails :
  http://docs.oracle.com/database/121/TGSQL/tgsql_optop.htm#TGSQL95143

Correction requête N°3 TP1

Nombre total de livres commandés en 2007 par pays

- Base de données Librairie :

clients(noclient, nom, prenom, adresse1, adresse2, codepostal, ville, pays, tel, email)

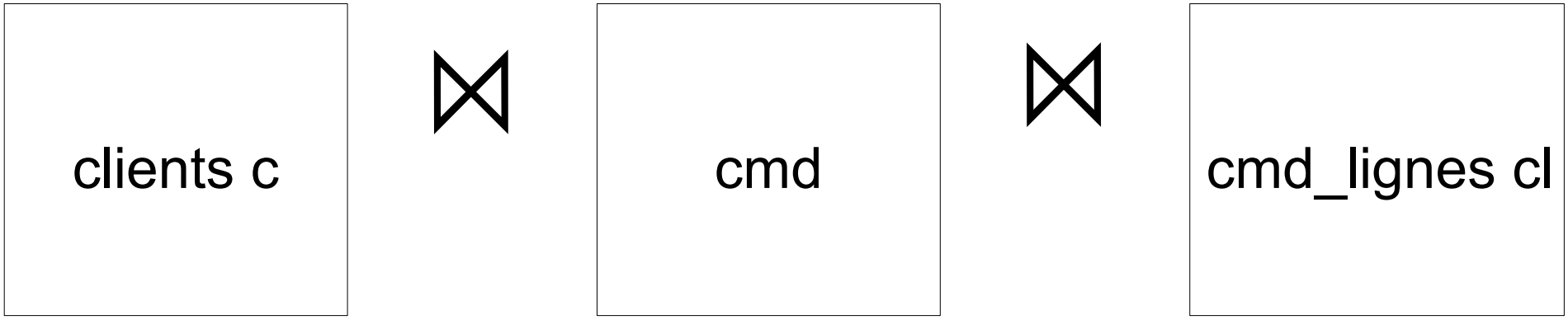
cmd(nocmd, noclient, datecommande, etatcommande)

livres(nolivre, titre, auteur, editeur, collection, isbn, prix)

cmd_lignes(nocmd, nolivre, quantite, remise, montant)

Correction requête N°3 TP1

Nombre total de livres commandés en 2007 par pays



Correction requête N°3 TP1

```
SELECT sum(cl.quantite)
FROM clients c JOIN cmd
       ON c.noclient=cmd.noclient
     JOIN cmd_lignes cl ON cmd.nocmd=cl.nocmd
WHERE extract(YEAR FROM cmd.datecommande)=2007 ;
```

=> On obtient le nombre total tous pays confondus
Rappel : impossible donc de mettre c.pays dans le SELECT

Correction requête N°3 TP1

Pour obtenir le nombre total par pays, il faut faire des groupes sur les pays

Résultat de la jointure, par groupes

Puis faire le calcul du nombre total sur chacun de ces groupes

Correction requête N°3 TP1

```
SELECT initcap(c.pays), sum(cl.quantite)
FROM clients c JOIN cmd ON
        c.noclient=cmd.noclient
     JOIN cmd_lignes cl ON cmd.nocmd=cl.nocmd
WHERE extract(YEAR FROM cmd.datecommande)=2007
GROUP BY initcap(c.pays);
```

NB :
- initcap(c.pays) permet de regrouper les pays entre eux même s'ils ont été saisis avec une casse différente
- tous les noms de colonnes apparaissant dans le SELECT doivent être des critères de formation des groupes, et doivent donc apparaître dans le GROUP BY tel quel (pas de c.pays dans le SELECT si initcap(c.pays) dans le GROUP BY)