

Chapitre 3 : Optimisation : Index et plan d'exécution

INDEX :

B-arbre, Bitmap

Création d'index sous oracle

Définition

- Pour compléter le schéma d'une table (relation), on peut créer des **index**.
- Permet une **recherche rapide d'une information**.

Exemple : Index d'un livre

INDEX

A

acteur, 448
action, 269, 591
administrateur, 30
 d'application, 30
 d'entreprise, 30
 de bases de données, 30
 de données, 16, 30
adressage ouvert, 75
adresse relative, 66
affinement du schéma logique, 662
agrégat, 210
agrégation 21, 668, 369, 670
 composite, 668, 676
 indépendante, 668, 675
agrégats auto-maintenables, 295
algorithme
 d'unification, 169
 génétique, 501, 503

arbre

algébrique, 306
 B, 85
 B+, 88
 de segments, 136
 de traitement, 306
 de versions d'objet, 642
 ET/OU, 548
 linéaire droit ou gauche, 335
 ramifié, 335
 relationnel, 207
 relationnel, 306
architecture
 à deux strates, 47
 à trois strates, 48
 BD répartie, 50
 client-serveur, 45
 répartie, 50
article, 60, 61, 113
association, 21, 670
 bijective, 673

déductive, 154
 hiérarchique, 137
 logique, 154
blocage, 67
 permanent, 607
boucles imbriquées, 330

C

cache volatile, 621
calcul relationnel
 de domaine, 157
 de tuples, 166
capture des besoins, 662
cardinalités d'association, 665
catalogue, 68
 de base, 71
 hiérarchisé, 69
certification de transaction, 614
chafnage, 75
 arrière, 543
 avant, 542
 de déclencheur, 269
 de règle, 523
conflits de noms, 364
connecteurs logiques, 523
conséquence immédiate, 529
constante, 225, 523
 de Skolem, 152
constructeur, 448
 d'objet, 375
contexte d'un événement,

couverture minimale, 688
création d'objets, 572
croisement, 502
 curseur, 129, 235

D

DATALOG, 522
 avec double négation
 avec ensemble, 539
 avec fonction, 537, 538
 avec négation, 533
déclencheur, 38, 248, 264
décomposition, 682, 686
 préservant
 les dépendances, 693
 sans perte, 683
définition
 de classe, 407
 de littéral, 408
degré d'isolation, 600
dégroupage, 385, 386, 539
 d'une collection, 386
démarche objet, 678
dénormalisation, 705
densité d'un index, 83
dépendance
 algébrique, 703

diagramme
 de Bachman, 116
 UML, 663

dictionnaire
 de règles, 266
 des données, 30
différence, 191, 387
distribution des objets, 374
division, 199
domaine, 181
 de discours, 150
données privées, 357
durabilité, 588

E

éclatement, 201, 202
écriture des clauses, 153
effet
 domino, 610
 net, 275
élaboration
 du schéma conceptuel, 662
 du schéma physique, 662
élimination
 de duplicata, 387
 des implications, 152

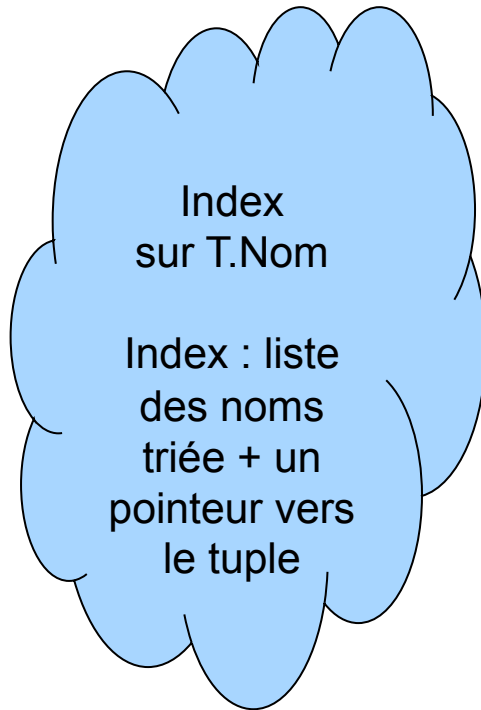
Index En action

```
Select *  
From Etudiant  
where Nom = "dupont"
```

- Un moyen pour récupérer la ou les lignes répondant à la clause nom='dupont' est de balayer toute la table (le fichier).
- Le temps de réponse sera prohibitif dès que la table dépasse quelques centaines de lignes.
- Afin d'optimiser ce type de requête, on pourra indexer l'attribut "nom".

	Nom	Pre	Adr
1	Martin	Léo	...
2	Deloin	Léa	...
3	Garcia	Luis	...
4	Fadle	Mac	...
5	Milka	Domi	...
6	dupont	Marco	...
7	Zara	Thé	...

Index (un attribut)

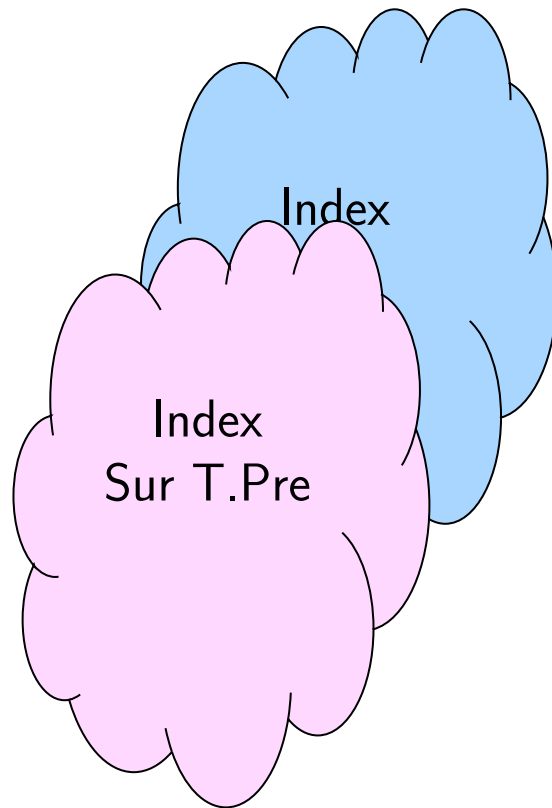


Nom	Ptr		Nom	Pre	Adr, ...
Deloin	2	1	Martin	Léo	...
Dupont	6	2	Deloin	Léa	...
Fadle	4	3	Garci	Luis	...
Garci	3	4	Fadle	Miche l	...
Martin	1	5	Milka	Domi	...
Milka	5	6	Dupont	Marco	...
Zara	7	7	Zara	Thé	...
...		

The table illustrates the mapping between the original data and an index. The index table (left) has columns 'Nom' and 'Ptr'. The original table (right) has columns 'Nom', 'Pre', and 'Adr, ...'. Arrows show the mapping: Deloin (Ptr 2) points to row 2, Dupont (Ptr 6) points to row 6, Fadle (Ptr 4) points to row 4, Garci (Ptr 3) points to row 3, Martin (Ptr 1) points to row 1, Milka (Ptr 5) points to row 5, and Zara (Ptr 7) points to row 7. The rows in the original table are sorted by name.

On peut effectuer une recherche dichotomique

Plusieurs index sur une même table

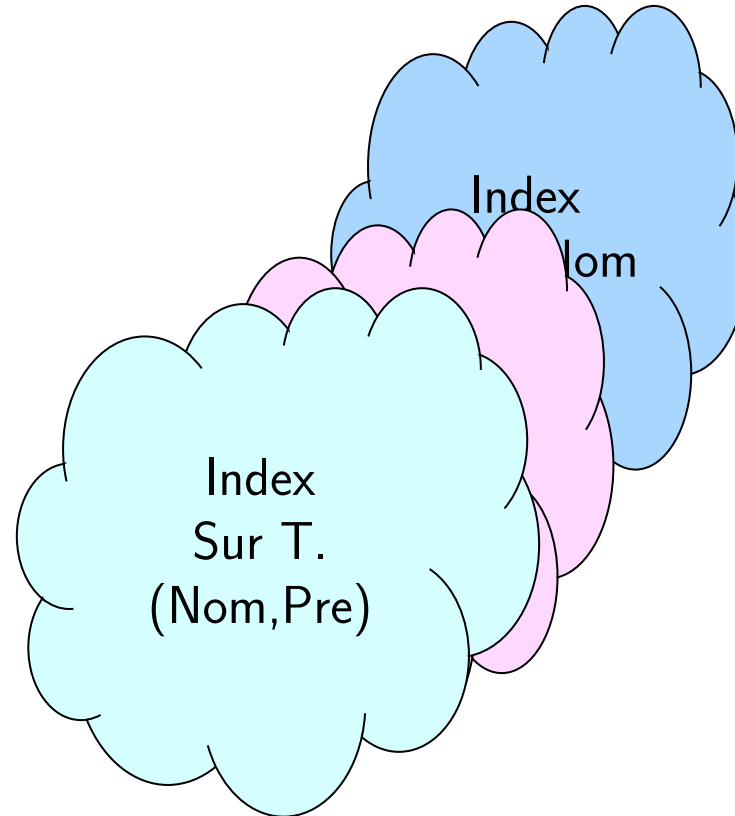


T

	Nom	Pre	C
1	Martin	Léo	...
2	Deloin	Léa	...
3	Garci	Luis	...
4	Fadle	Mich el	...
5	Milka	Domi	...
6	Dupont	Marc o	...
7	Zara	Thé	...

Possibilité de créer plusieurs index sur une même table.

Index composé



T

	Nom	Pre	Adr
1	Martin	Léo	...
2	Deloin	Léa	...
3	Garci	Luis	...
4	Fadle	Michel	...
5	Milka	Domi	...
6	Du pon t	Marco	...
7	Zara	Thé	...

Peut concerner plusieurs attributs d'une même table (index composé) (cas clé primaire composé).

Création index : syntaxe SQL

- Création d'un index

```
CREATE INDEX nom_index ON nom_table(colonne[,colonne2 ...]);
```

- Création index unique

```
CREATE UNIQUE INDEX nom_index ON nom_table(colonne[,colonne2 ...]);
```

- Exemple

```
CREATE INDEX IdxNomEtud On Etudiant (Nom)
```

- Suppression

```
DROP INDEX nom_index ;
```


Index par défaut

- Création **implicite** d'index :
 - chaque fois qu'une **clé primaire** ou
 - une **contrainte d'unicité** sur un attribut est définie pour une table.
- Pour vérifier l'existence d'index : consulter les vues (ORACLE)
 - USER_INDEXES, USER_IND_COLUMNS, ALL_INDEXES,
 - ALL_IND_COLUMNS ou DBA_INDEXES

Choix des index

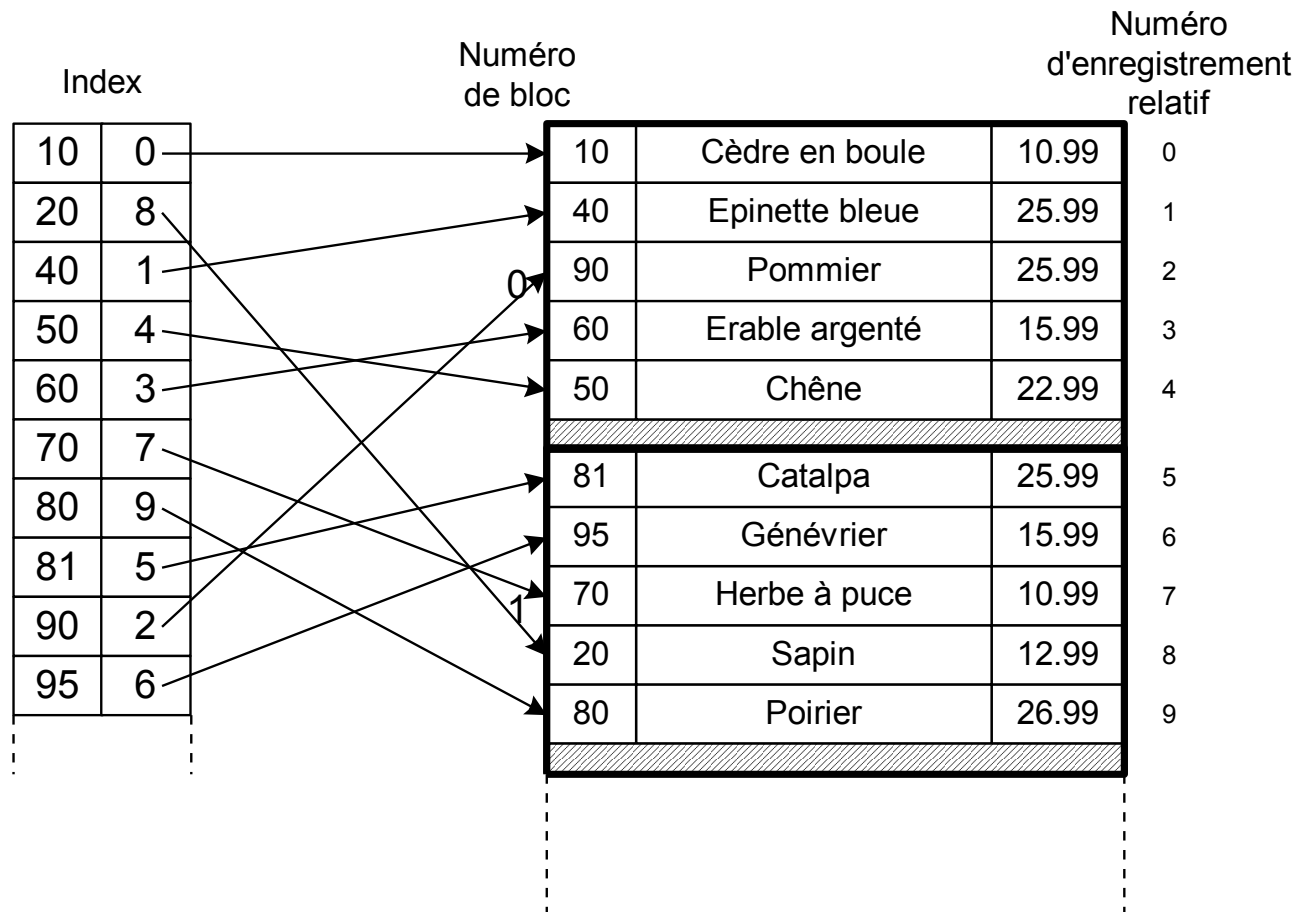
- On choisira de créer un index sur :
 - Les attributs utilisés comme **critère de jointure**,
 - Les attributs servant souvent de **critères de sélection**,
 - Sur une table de gros volume (d'autant plus intéressant si les requêtes sélectionnent peu de lignes).
- L'adjonction d'index accélère la recherche des lignes. Mais impact négatif sur la mise à jour de la table.
 - Impact négatif sur les commandes d'insertion et de suppression (car mise à jour de tous les index portant sur la table) → coût.

Type d'index

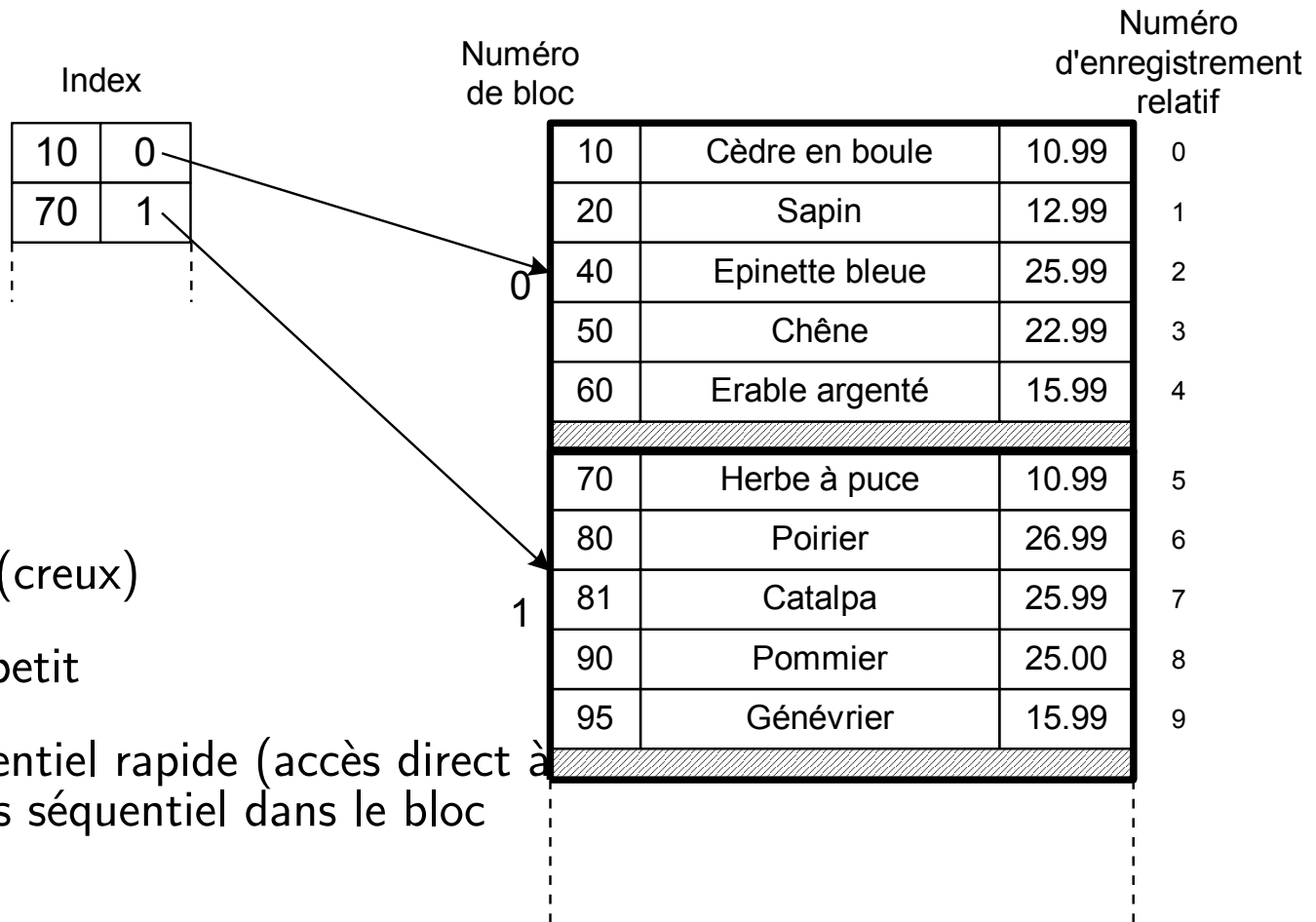
- Organisation de l'index
 - Séquentiel (index dense)
 - Séquentiel indexé (index creux)
 - Arbres B+ (par défaut dans ORACLE)
 - Bitmap
 - Hashage

Type d'index: Index séquentiel (dense)

- Toutes les clés de recherche sont présentes dans l'index



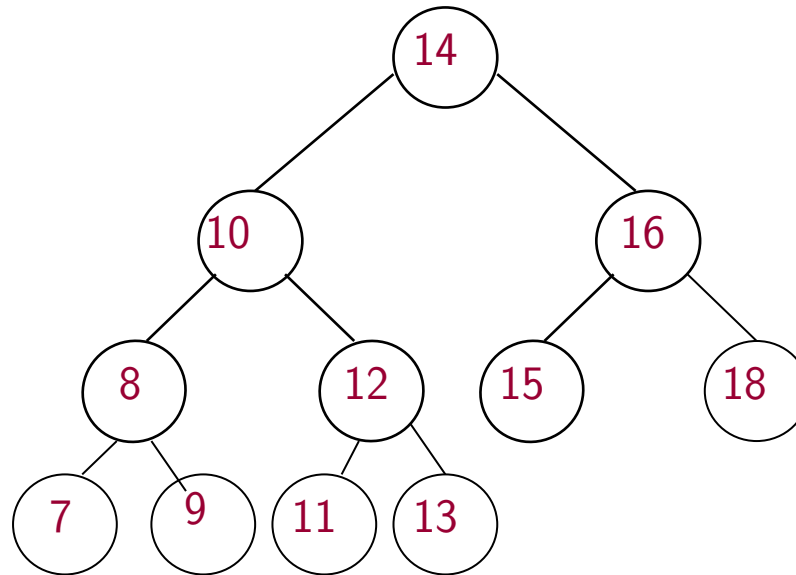
Type d'index: Index creux/épars (séquentiel indexé)



- Non dense (creux)
- Index plus petit
- Accès séquentiel rapide (accès direct à un bloc puis séquentiel dans le bloc)
- Primaire

Type index : Index par Arbre-B

- Rappel arbre binaire (idem à indexation séquentielle sur des clés triées (recherche dichotomique))



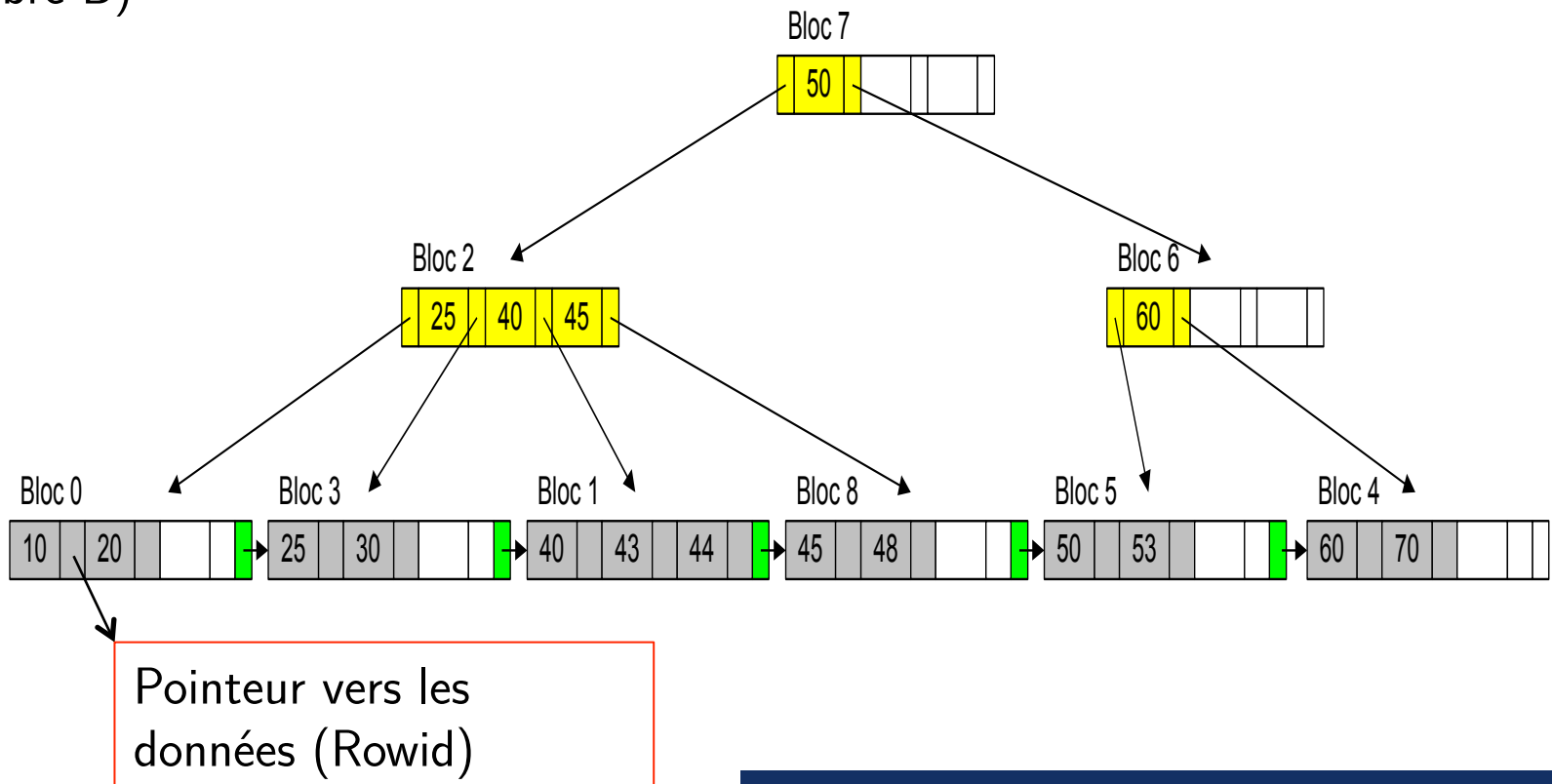
Parcours de l'arbre A:

1	2	3	4	5	6	7	8	9	10	11
14	10	16	8		12	15	18	7	9	11
13										

Cet index est équivalent à une recherche dans une table triée

Type index : Index par Arbre-B+

- Au lieu d'une clé par nœud, chaque nœud est un bloc contenant k clés triées et k+1 fils
- Les B-arbres sont équilibrés (toutes les feuilles à la même profondeur).
- Les feuilles contiennent les valeurs des clés et le Rowid
- Arbre B+ Les clés sont répétées au niveau des feuilles (mais pas dans un arbre B)

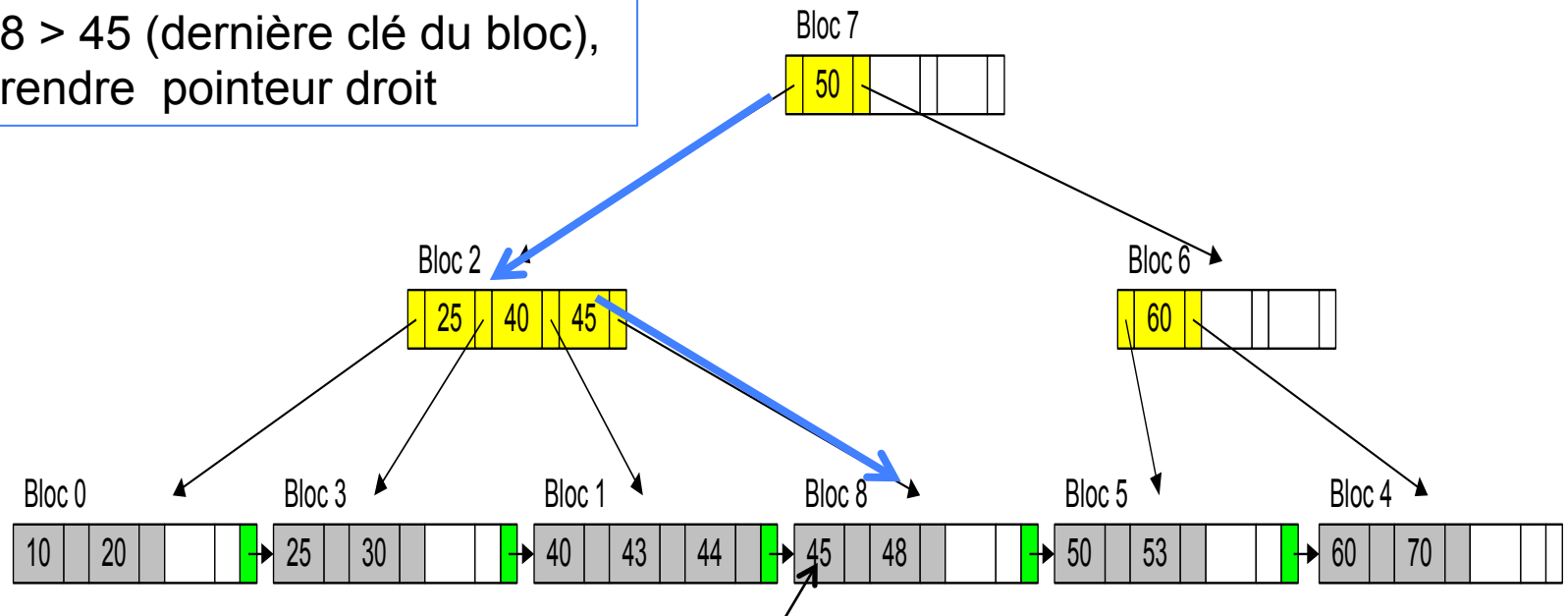


Type d'index : Index par Arbre-B

- Exemple : recherche clé 48

Commencer par la racine
 $48 < 50$, prendre pointeur gauche

$48 > 45$ (dernière clé du bloc),
prendre pointeur droit



Lire Bloc 8
Comparer séquentiellement 48 aux autres clés du Bloc

Type d'index: Index bitmap

	Titre	Genre
1	Vertigo	Suspense
2	Brazil	Science-fiction
3	Twin Peaks	Fantastique
4	Underground	Drame
5	Easy Rider	Drame
6	Psychose	Drame
7	Greystoke	Aventures
8	Shining	Fantastique
9	Annie Hall	Comédie
10	Jurassic Park	Science-fiction
11	Metropolis	Science-fiction
12	Manhattan	Comédie
13	Reservoir Dogs	Policier
14	Impitoyable	Western
15	Casablanca	Drame
16	Smoke	Comédie

Un index bitmap considère toutes les valeurs possibles pour un attribut. Pour chaque valeur, on stocke un tableau de bits (dit bitmap) avec autant de bits qu'il y a de lignes dans la table.

Très utile pour les colonnes qui ne possèdent que quelques valeurs distinctes.

Index sur genre

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Drame	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	0
Science-fiction	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0
Comédie	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1

Index bitmap

Rechercher les films de type "drame"

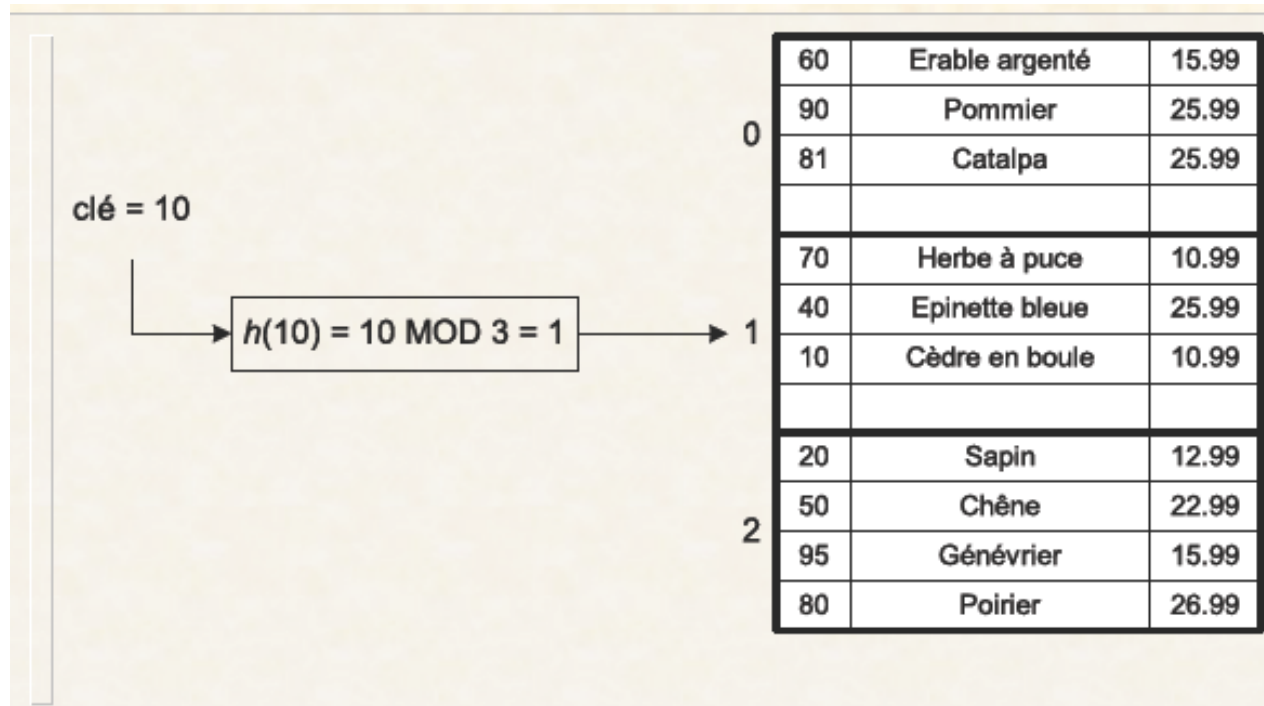
1- Sélectionner dans la table les numéros de ligne (RowId) ayant Drame="1"

RowId	Drame	Science-fiction	Comédie
1	0	0	0
2	0	1	0
3	0	0	0
4	1	0	0
5	1	0	0
6	1	0	0
7	0	0	0
8	0	0	0
9	0	0	1
10	0	1	0
11	0	1	0
12	0	0	1
13	0	0	0
14	0	0	0
15	1	0	0
16	0	0	1

	Titre	Genre
1	Vertigo	Suspense
2	Brazil	Science-fiction
3	Twin Peaks	Fantastique
4	Underground	Drame
5	Easy Rider	Drame
6	Psychose	Drame
7	Greystoke	Aventures
8	Shining	Fantastique
9	Annie Hall	Comédie
10	Jurassic Park	Science-fiction
11	Metropolis	Science-fiction
12	Manhattan	Comédie
13	Reservoir Dogs	Policier
14	Impitoyable	Western
15	Casablanca	Drame
16	Smoke	Comédie

Type d'index: Hashage

Hashage : utilise une fonction(h) qui appliquée à la clé h(clé) donne l'adresse ou se trouve le tuples (l'enregistrement)



Problème avec le hashage → conflits entre clés

→ deux attributs qui ont la même valeur de hashage

→ prévoir des algos pour résoudre les conflits

Index dans Oracle

- Plusieurs types d'index dans Oracle :
 - Arbres équilibrés (arbre B+)(par défaut)
 - Bitmap
 - Hasahage

```
CREATE [UNIQUE | BITMAP] INDEX  
[<schema>.]<nom index>  
  ON <nom de table>  
  (<nom de colonne> [ASC|DESC]  
  [,<nom de colonne> [ASC|DESC]]  
  , ...)
```

- Unique = index sans doublons

Table organisée en index

- Table organisée en index
 - Fusion entre la table (son contenu) et son index
 - Toutes les valeurs de la table sont au sein d'un index B-arbre
 - Utile uniquement si la clé primaire constitue l'essentiel des attributs (manipulés)

Ordre Oracle

```
CREATE TABLE (  
  id  NUMBER NOT NULL PRIMARY KEY,  
  [...]  
) ORGANIZATION INDEX;
```

Conclusion

- Indexation processus important pour l'accès rapide aux données

Optimisation des Requêtes
Plan d'exécution
EXPLAIN PLAN

Outils d'optimisation d'oracle

- Outils
 - EXPLAIN: visualisation des plans d'exécution
 - ANALYSE: production de statistiques
 - TKPROF: mesure du temps d'exécution

Plan d'exécution

- Oracle conserve la trace du chemin d'accès d'une requête dans une table appelée « **plan_table** »
- Tous les graphes des requêtes du LID et LMD (**SELECT**, **UPDATE**, **INSERT** et **DELETE**) sont conservées
- Exemple

Plan d'exécution

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		14	518	2
1	TABLE ACCESS FULL	ARTISTE	14	518	2

Statistiques

Plan d'exécution

- Id : Identifiant de l'opérateur ;
- Operation : type d'opération utilisée
- Name : Nom de la relation utilisée ;
- Cost: coût estimé par oracle;
- Rows : Le nombre de lignes qu'Oracle pense transférer. ,
- Bytes : Nombre d'octets qu'oracle pense transférer.

Type d'opération

- Chaque opération sur un objet est notée avec :
 - L'ordre d'exécution (ordre des opérateurs)
 - Le chemin d'accès aux objets consultés (table, index, cluster, view, ...)
 - Le type d'opération (opérateurs physiques)

Type d'opération : chemin d'accès

- Parcours séquentiel (balayage complet)
 - TABLE ACCESS FULL
- Accès direct par adresse
 - TABLE ACCESS BY (INDEX|USER|...) ROWID
- Accès par index
 - INDEX (UNIQUE|RANGE|...) SCAN
- Accès par hachage
 - TABLE ACCESS HASH
- Accès par cluster
 - TABLE ACCESS CLUSTER

Type d'opération : Opérateurs physiques

- Pour la jointure
 - Boucles imbriquées: NESTED LOOPS
 - Tri-fusion: SORT JOIN, MERGE JOIN
 - Hachage: HASH JOIN
- Autres opérations
 - Union d'ensembles d'articles: CONCATENATION, UNION
 - Intersection d'ensembles d'articles: INTERSECTION
 - Différence d'ensembles d'articles: MINUS
 - Filtrage d'articles d'une table basé sur une autre table: FILTER
 - Intersection d'ensembles de ROWID: AND-EQUAL
 - ...
- Tous les données d'explain Plan sont stockées dans la table PLAN_TABLE

OPERATIONS	OPTIONS	SIGNIFICATION
AGGREGATE	GROUP BY	Une recherche d'une seule ligne qui est le résultat de l'application d'une fonction de group à un groupe de lignes sélectionnées.
AND-EQUAL		Une opération qui a en entrée des ensembles de rowids et retourne l'intersection de ces ensembles en éliminant les doublants. Cette opération est utilisée par le chemin d'accès par index.
CONNECT BY		Recherche de ligne dans un ordre hiérarchique
COUNTING		Opération qui compte le nombre de lignes sélectionnées.
FILTER		Accepte un ensemble de ligne, appliqué un filter pour en éliminer quelque unes et retourne le reste.
FIRST ROW		Recherché de la première ligne seulement.
FOR UPDATE		Opération qui recherche et verrouille les lignes pour une mise à jour
INDEX	UNIQUE SCAN	Recherche d'une seule valeur ROWID d'un index.
INDEX	RANGE SCAN	Recherche d'une ou plusieurs valeurs ROWID d'un index. L'index est parcouru dans un ordre croissant.
INDEX	RANGE SCAN DESCENDING	Recherche d'un ou plusieurs ROWID d'un index. L'index est parcouru dans un ordre décroissant.
INTERSECTION		Opération qui accepte deux ensembles et retourne l'intersection en éliminant les doublons.
MARGE JOIN+		Accepte deux ensembles de lignes (chacun est trié selon un critère), combine chaque ligne du premier ensemble avec ses correspondants du deuxième et retourne le résultat.
MARGE JOIN+	OUTER	MARGE JOIN pour effectuer une jointure externe
MINUS		Différence de deux ensembles de lignes.
NESTED LOOPS		Opération qui accepte deux ensembles, l'un externe et l'autre interne. Oracle compare chaque ligne de l'ensemble externe avec chaque ligne de l'ensemble interne et retourne celle qui satisfait une condition.
NESTED LOOPS	OUTER	Une boucle imbriquée pour effectuer une jointure externe.
PROJECTION		Opération interne
REMOTE		Recherche de données d'une base distante.
SEQUENCE		Opération nécessitant l'accès à des valeurs du séquenceur
SORT	UNIQUE	Tri d'un ensemble de lignes pour éliminer les doublons.
SORT	GROUP BY	Opération qui fait le tri à l'intérieur de groupes
SORT	JOIN	Tri avant la jointure (MERGE-JOIN).
SORT	ORDER BY	Tri pour un ORDER BY.
TABLE ACCESS	FULL	Obtention de toutes lignes d'une table.
TABLE ACCESS	CLUSTER	Obtention des lignes selon la valeur de la clé d'un cluster indexé.
TABLE ACCESS	HASH	Obtention des lignes selon la valeur de la clé d'un hash cluster
TABLE ACCESS	BY ROW ID	Obtention des lignes on se basant sur les valeurs ROWID.
UNION		Union de deu
VIEW		Opération qui

Structure de PLAN_TABLE

STATEMENT_ID	VARCHAR2(30)	id choisi
TIMESTAMP	DATE	date d'exécution
REMARKS	VARCHAR2(80)	
OPERATION	VARCHAR2(30)	opération (plus loin)
OPTIONS	VARCHAR2(30)	option de l'opération
OBJECT_NODE	VARCHAR2(128)	pour le réparti
OBJECT_OWNER	VARCHAR2(30)	propriétaire
OBJECT_NAME	VARCHAR2(30)	nom objet (table, index,
OBJECT_INSTANCE	NUMBER(38)	
OBJECT_TYPE	VARCHAR2(30)	(unique, ...)
OPTIMIZER	VARCHAR2(255)	
SEARCH_COLUMNS	NUMBER	
ID	NUMBER(38)	n° noeud courant
PARENT_ID	NUMBER(38)	n° noeud parent
POSITION	NUMBER(38)	1 ou 2 (gauche ou droite)
COST	NUMBER(38)	
CARDINALITY	NUMBER(38)	
BYTES	NUMBER(38)	
OTHER_TAG	VARCHAR2(255)	
PARTITION_START	VARCHAR2(255)	
PARTITION_STOP	VARCHAR2(255)	
PARTITION_ID	NUMBER(38)	
OTHER	LONG	

Mise en oeuvre

- Calcul du plan d'exécution remplissage dans PLAN_TABLE

```
EXPLAIN PLAN
SET statement_id = 'p1' FOR
select noemp, nomemp
from EMPLOYE
where noemp > 3;
```

Visualisation EXPLAIN PLAN command & dbms_xplan.display function

```
SELECT plan_table_output FROM
table(dbms_xplan.display('plan_table',null,'basic'));
```

```
Plan hash value: 1519211061
```

```
-----
| Id | Operation                | Name |
-----
| 0 | SELECT STATEMENT         |      |
| 1 | TABLE ACCESS FULL      | EMPLOYE |
-----
```

```
8 rows selected
```

Mise en oeuvre

- Calcul du plan d'exécution remplissage dans PLAN_TABLE

```
EXPLAIN PLAN
SET statement_id = 'p1' FOR
select noemp, nomemp
from EMPLOYE
where noemp > 3;
```

```
column operation format a18
column options format a15
column object_name format a13
column id format 99
column parent_id format 99
column position format 99
select operation, options, object_name, id, parent_id, position
from plan_table
where statement_id='p1'
```

Mise en œuvre

```
EXPLAIN PLAN
SET statement_id = 'p1' FOR
select nomemp,nomdept, salaire
from employe, departement
where employe.noddept = departement.noddept
and salaire >5000
```

OPERATION	OPTIONS	OBJECT_NAME	ID	PARENT_ID	POSITION
SELECT STATEMENT			0		6
MERGE JOIN			1	0	1
TABLE ACCESS	BY INDEX ROWID	DEPARTEMENT	2	1	1
INDEX	FULL SCAN	PK_DEPARTEMENT	3	2	1
SORT	JOIN		4	1	2
TABLE ACCESS	FULL	EMPLOYE	5	4	1

6 lignes sélectionnées

Mise en œuvre

Vous pourrez aussi exécuter

exécuter « set autotrace on »

```
set autotrace on
select noemp, nomempfrom
EMPLOYE where noemp > 3;
```

Plan hash value: 1519211061


```
-----
| Id | Operation      | Name  | Rows | Bytes | Cost (%CPU)| Time |
|-----|-----|-----|-----|-----|-----|-----|
|  0 | SELECT STATEMENT |      |    11 |  275 |    3 (0)| 00:00:01 |
|*  1 | TABLE ACCESS FULL| EMPLOYE |    11 |  275 |    3 (0)| 00:00:01 |
|-----|-----|-----|-----|-----|-----|
```

Predicate Information (identified by operation

id):-----

1 - filter("NOEMP">3)

Mise en œuvre

SQL Developer permet d'obtenir un plan d'exécution pour une requête grâce à l'icône . 

```
select noemp,  
departement.nomdept from EMPLOYE,  
Departement where  
Employe.NoDept=Departement.NoDept  
and salaire >2000;
```

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			7
HASH JOIN			7
Access Predicates EMPLOYE.NODEPT=DEPARTEMENT.NODEPT			
TABLE ACCESS	DEPARTEMENT	FULL	3
TABLE ACCESS	EMPLOYE	FULL	3
Filter Predicates SALAIRE>2000			

Fin