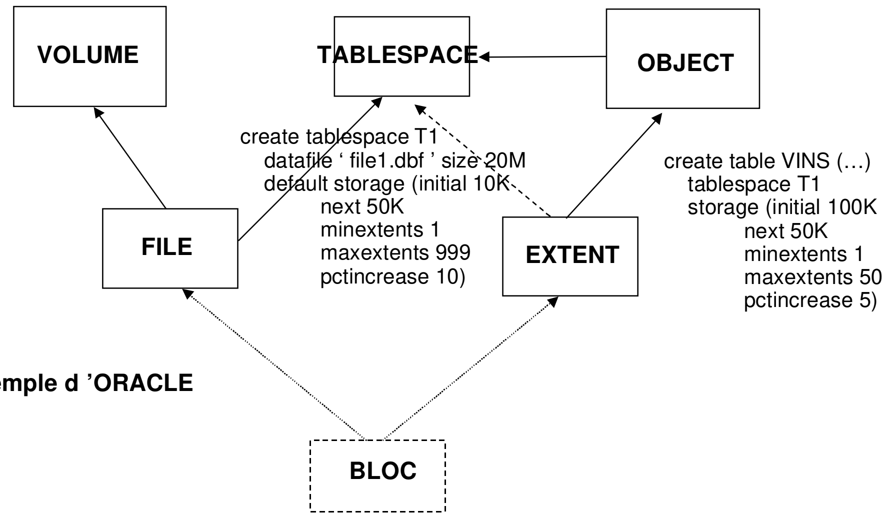
Le placement et les méthodes d'accès aux données:
le stockage physique

J-M. SAGLIO

Fichiers de blocs

- disque et mémoire volatile
    - 3D (Cylindre, Piste, Secteur-mot) / 1D (mot)
    - accès en millisecondes / en microsecondes
    - le bloc (ou la "page") unité d'échange et tampon
    - adressage physique des blocs du disque
- fichiers de blocs
    - contigüité totale ou par segments (ou "extents")
    - en-tête et adressage logique des blocs du fichier
    - primitives:
        - Open, Close
        - Find(Retreive), FindNext
        - Read(Get), Write(Put or Update)
        - Append(Insert), Delete

Mapping objets logiques /fichiers de blocs



L'exemple d 'ORACLE

Fichiers d'articles

- Représentation physique d'un article (ou "record" ou "structure" ou 't-uple")

- **- format fixe v. variable, avec séparateurs**
- **- le "facteur de blocage" = nombre d'articles par bloc**
- **- seuils mini et maxi pour l'insert d'un nouvel article dans un bloc**

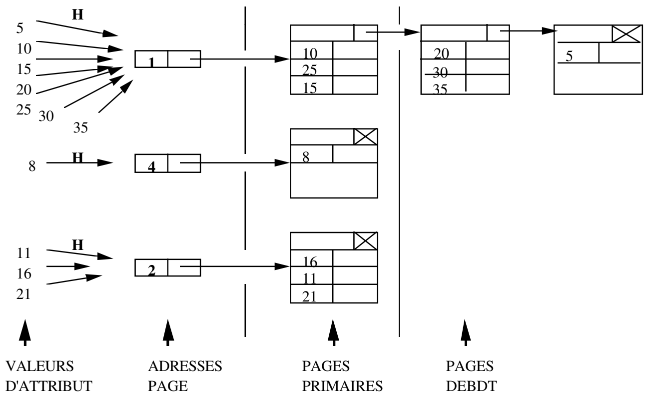
Oracle: PCTFREE et PCTUSED

- Placement

- **- en "tas": recherche par balayage (linéaire), mais insertion simple (append)**
- **- trié sur un attribut "unique" (en général la clef): recherche dichotomique (log_2 blocs), mais insertion ou destruction => réorganisation**
- **- par calcul sur un attribut unique ("hachage"): recherche directe (même calcul pour l'accès comme pour le placement), mais débordements possibles (solutions: redirection, multiples calculs)**

critère: un placement le plus uniforme possible

Organisation aléatoire



H = FONCTION DE HACHAGE [(RESTE MOD 5) + 1]

Principe et choix de la fonction de hachage

Principe

* Le fichier (la relation) est divisé en P paquets de taille fixe (1 BLOC)
* La clé est convertie en numéro de paquet à l'aide d'une fonction (aléatoire)
* A l'intérieur d'un paquet, les articles ne sont pas triés.

•**Choix de la fonction**

Pliage de la clé, Conversion, Modulo P

Une bonne fonction de hachage doit satisfaire deux exigences :

- Son calcul doit être très rapide
- Elle doit déterminer une distribution uniforme et éviter de saturer des paquets.

Exemple

H(K) = K modulo M

avec M nombre de paquets

Gestion des débordements

* **ADRESSAGE OUVERT** :
  Recherche du premier paquet non-saturé en séquence

* **CHAINAGE** :
  Ajout d'un paquet de débordement chaîné au paquet saturé

* **REHACHAGE :**
  Application d'une nouvelle fonction avec des paquets de débordement

Techniques de hachage pour les fichiers évolutifs

- Le hachage "dynamic" (Larson 1978)
utilise un arbre binaire de hauteur variable

- Le hachage "extensible"(Fagin 1979)
utilise une table d'adresses de longueur variable

- Le hachage "linéaire" (Litwin 1980)
utilise un curseur et suite logique de fonctions de hachage

Types d'Index

- à un seul niveau

(l'index est lui même un fichier ordonné => recherche dichotomique)

- "primaire" = sur clé primaire ordonnant totalement le fichier indexé

(en général non dense, mais peut être dense)

- "clustering" = sur un attribut ordonnant partiellement le fichier indexé

- "secondaire" = sur un attribut quelconque du fichier indexé

(dense si sur clé secondaire)

- à plusieurs niveaux

(temps de recherche en log $_{fbi}$ - au lieu de log$_2$ - blocs du 1er niveau)

fbi ***= facteur de blocage ou de "ventilation" de l'index***

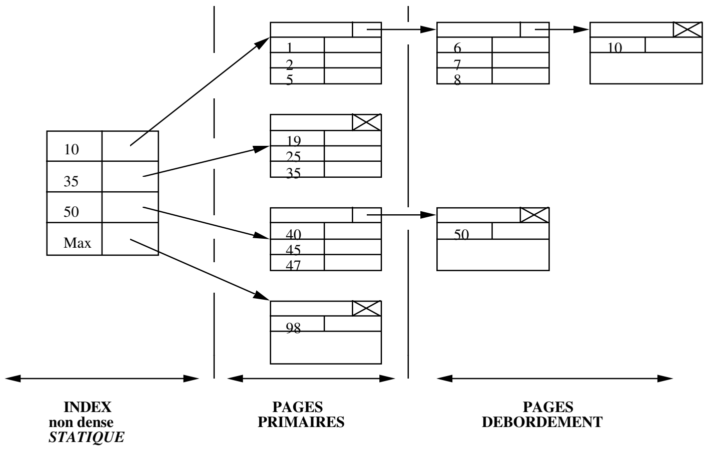
- statique

- dynamique

arbres de recherche d'ordre prédéfini

arbres "balancés" (B trees et B+ trees)

Index statiques



Index dynamiques

Dans un bloc d'index, on place au plus 2m-1 valeurs de clés et 2m pointeurs

(m est l' "ordre" de l'index, il s'apparente au "facteur de blocage" ou "fan-out")

- Arbre simple:

< P1, **K1,** P2, **K2,** P3, **.. , Ki-1,** Pi, **Ki, .., Kq-1,** Pq >

(avec q =< 2m)

- Arbre B+ (« balancé ») :

< P1, **K1,** P2, **K2,** P3, **.. , Ki-1,** Pi, **Ki, .., Kq-1,** Pq >

(avec q =< 2m et q>= m - *exception faite de la racine et des feuilles* **)**

pour tout bloc intermédiaire pointé par Pi **: Ki-1 < K =< Ki**

et pour les blocs-feuilles:

< *Pr1*, **K1,** *Pr2*, **K2,.. ,** *Pri-1*, **Ki-1,** *Pri*, **Ki,..,** *Prq-1*, **Kq-1,** *Pnext bloc* >

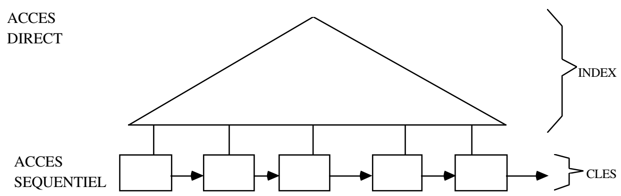
N.B. Chaque chemin de la racine vers une feuille est de la même longueur H appelée hauteur, H est de l 'ordre de Log $_m$ (nombre de clés)

Index en Arbre B+

Index organisé en arbre B

\+

Ensemble de feuilles contenant les clés



Dynamique

* Eclatement d'une feuille : Une copie de la clé médiane est remontée

* Suppression d'une clé : uniquement dans une feuille

=> Recherche jusqu'aux feuilles

Variations techniques

Arbres B*:

facteur de remplissage aux 2/3 ou réglable

Index hachés

au lieu d'être un fichier ordonné au 1er niveau, le fichier des couples <Ki, Pi> est lui-même haché (donc accédé par calcul sur la valeur de clef)

Index (secondaires) logiques

au lieu des couples <Ki, Pi> on range des couples <Ki, KPi> où KPi désigne une valeur de la clef primaire

Fichier inversé

fichier "en tas" avec autant d'index secondaires B+ (physiques) que d'attributs

Fichier séquentiel indexé

fichier ordonné (séquentiel) avec index incorporé, physique - avec adressage très orienté disque - (ISAM) ou logique(VSAM)

Méthode d'accès et placement

=> **ACCES PRIMAIRE - Plaçant**

Pour accélérer l'accès à un large ensemble de tuples

=>**ACCES SECONDAIRE - non plaçant**

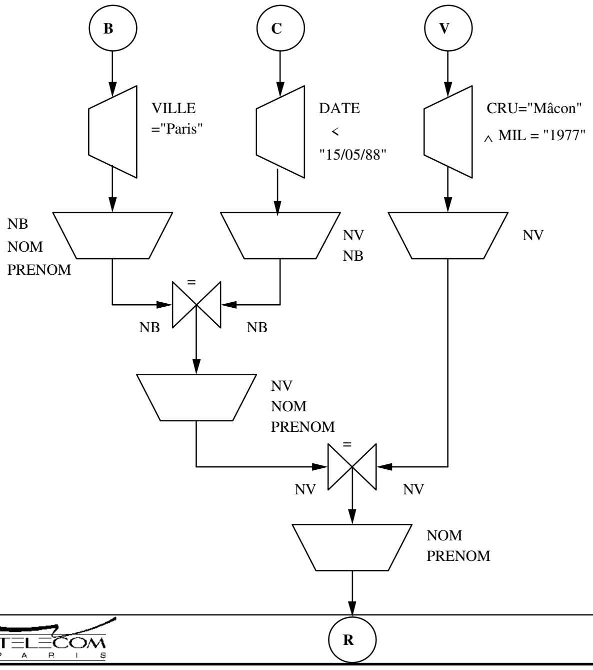
Pour accélérer l'accès à un petit ensemble de tuples, en complément d'un type de placement

UNE RELATION

-> UN SEUL TYPE DE PLACEMENT

-> N INDEX ASSOCIES

Arbre d 'exécution



Exécution des jointures

<u>BOUCLES IMBRIQUEES</u>

POUR CHAQUE PAGE DE R1
POUR CHAQUE PAGE DE R2
JOINDRE (PAGE.R1, PAGE.R2)

COUT = |R1| * |R2| + |R1|

<u>TRI FUSION</u>

TRIER R1
TRIER R2
FUSIONNER (R1, R2)

COUT = 2|R1| LOG|R1| + 2 |R2|LOG|R2| + |R1|+|R2|

UTILISATION POSSIBLE DES INDEX

Optimisation

.le SGBD doit exécuter un arbre algébrique optimisé

.optimiser

-> le nombre d'E/S

-> le temps UC

-> ...

.facteurs déterminants

-> ordre d'exécution des opérations algébriques

-> algos implantant les opérations algébriques

-> placement des données sur disque

-> taille des relations intermédiaires

Architecture d'un interprêteur