

## Implantation des SGBD relationnels

Université de Nice Sophia-Antipolis  
Version 3.4 – 25/11/06  
Richard Grin

### Plan de cette partie

- Nous allons étudier (très rapidement !) quelques éléments de solutions utilisés par Oracle pour
  - implantation des index
  - enregistrer les données sur les mémoires de masse
  - gérer les accès concurrents
  - effectuer les reprises après panne
  - optimiser les requêtes
  - valider les transactions dans les bases de données réparties

R. Grin

SGBD

2

## Index

R. Grin

SGBD

3

### Définition et fonctionnalités

- Un index est un objet informatique qui contient des clés
- Une clé d'un index d'une table permet d'accéder rapidement à une ou plusieurs lignes de la table
- Un index peut ainsi accélérer une jointure ou une recherche d'information dans la base

R. Grin

SGBD

4

### Idée principale pour l'implémentation d'un index

- Limiter le nombre des accès disque pour trouver une clé

R. Grin

SGBD

5

### Coûts pour retrouver des données

- Un accès disque = quelques millisecondes ( $10^{-3}$  secondes)
- Un accès mémoire = quelques nanosecondes ( $10^{-9}$  secondes)
- Donc un accès disque est en gros 1.000.000 de fois plus lent qu'un accès mémoire
- Aussi à retenir : en une milliseconde, un processeur moderne peut exécuter au moins 1.000.000 d'instructions machine

R. Grin

SGBD

6

- Les algorithmes d'utilisation des index doivent donc à tout prix limiter les accès disques, même si ça nécessite des calculs en mémoire centrale

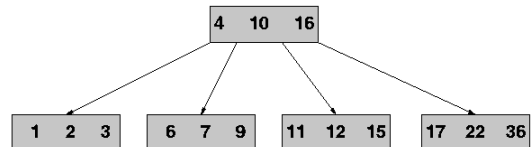
## Types d'index

- B-arbres (et ses variantes) ; ce sont les plus fréquemment utilisés
- Index bitmaps réservés aux tables modifiées rarement
- Index « table de hachage », presque plus utilisés

## B-arbres

- Ils permettent de réduire fortement le nombre d'accès disque
- On peut parcourir l'arbre dans l'ordre des clés
- Les caractéristiques physiques principales :
  - toutes les branches ont la même profondeur
  - chaque nœud a toujours un nombre minimum et un nombre maximum de clés

## Exemple de B-arbre



- Les nœuds qui ont m clés ont m + 1 fils
- 1er fils : clés inférieures à la 1ère clé
- Autres fils : clés supérieures à une clé et inférieures à la clé suivante (si elle existe)

## Implémentation des B-arbres

- Principe des arbres binaires mais chaque nœud a beaucoup plus que 2 fils (plusieurs dizaines le plus souvent)
- B-arbre d'ordre n : les nœuds ont au moins n et au plus 2 x n clés (sauf, la racine qui peut n'avoir qu'une seule clé)
- Nombre de fils d'un nœud = nombre de clés + 1
- Toutes les feuilles sont au même niveau

## Optimisation

- La taille des nœuds doit être un multiple de la taille du secteur du disque (lu en un seul accès disque)

## Ordre d'idées

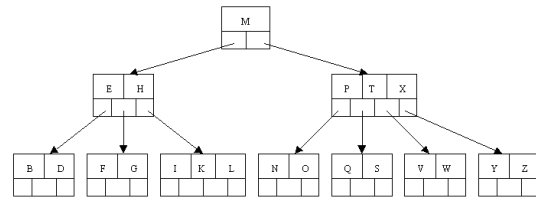
- Si un B-arbre est d'ordre 25, il suffira de 4 accès disques pour accéder à une clé parmi des millions
- En effet, la hauteur d'un B-arbre d'ordre 25 qui contient 10.000.000 de clés est inférieure à  $\log_{25}((10^7 + 1) / 2) \sim 4.8$  (en  $O(\log n)$ )
- 3 accès suffiront car la racine est le plus souvent en mémoire centrale (utilisation d'un cache)

R. Grin

SGBD

13

## Un exemple



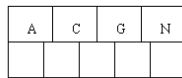
- Ordre 2 : maximum 4 clés et 5 pointeurs (2 clés au minimum)

R. Grin

SGBD

14

## Au départ

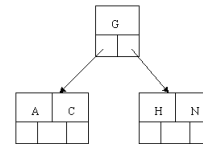


- Ajout de H

R. Grin

SGBD

15

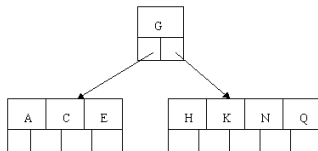


- L'insertion peut provoquer l'ajout de pages en cascade jusqu'à la création d'une nouvelle racine.
- Les B-arbres poussent par la racine !
- Ajout de E, K, Q

R. Grin

SGBD

16

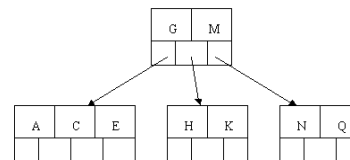


- Ajout de M

R. Grin

SGBD

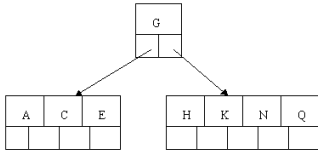
17



R. Grin

SGBD

18

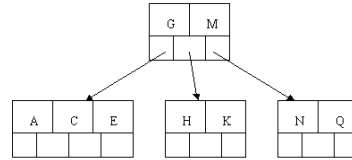


- On aurait aussi pu « équilibrer » : G passe à gauche, H passe en haut et M se place à droite
- Ajout de M

R. Grin

SGBD

19

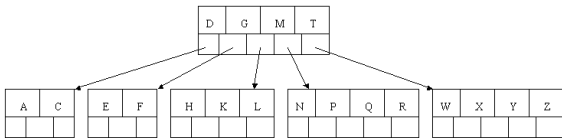


- Ajout de D, F, L, P, R, T, W, X, Y, Z

R. Grin

SGBD

20

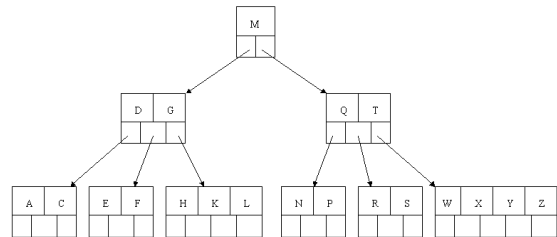


- Ajout de S

R. Grin

SGBD

21

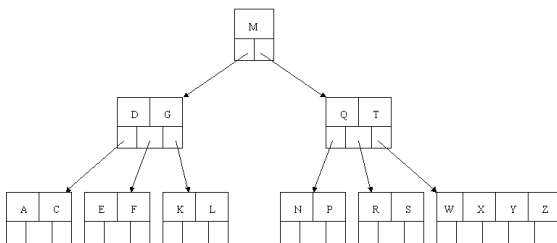


- Suppression de H (dans une feuille)

R. Grin

SGBD

22

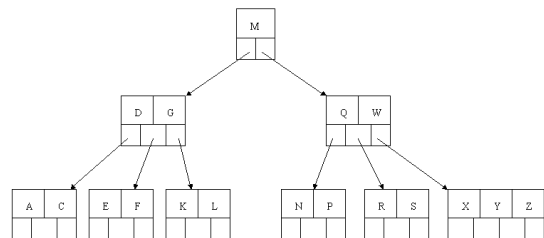


- Suppression de T (pas dans une feuille)

R. Grin

SGBD

23



- T est remplacé par son successeur
- Suppression de R (avec équilibrage)

R. Grin

SGBD

24

• On a équilibré avec la feuille de droite  
 • Suppression de E

R. Grin SGBD 25

• La suppression de E provoque la concaténation de 2 nœuds avec la clé mère  
 • Un des nœuds n'a pas assez de clés !

R. Grin SGBD 26

• On ne peut équilibrer le nœud qui n'a pas assez de clés avec ses voisins, donc on le concatène avec son voisin et la racine ; l'arbre a un niveau de moins

R. Grin SGBD 27

### Pour se tester

- <http://deptinfo.cnam.fr/Enseignement/Cycle A/SD/demonstration/B-Arbre.html>
- <http://slady.net/java/bt/>

R. Grin SGBD 28

### Variante B+ des B-arbres

- Les clés sont toutes rangées dans les feuilles et les feuilles sont chaînées
- Le parcours de toutes les clés dans l'ordre de l'index est ainsi très rapide (on parcourt la liste chaînée des feuilles)

R. Grin SGBD 29

### Index bitmaps

- Utiles lorsque les données de la table ne sont presque jamais modifiées
- Peuvent être très performants sur des combinaisons (« et », « ou ») de critères (reviennent à faire des opérations sur des tableaux de bits)

R. Grin SGBD 30

## OLAP

- Les index bitmaps sont le plus souvent utilisés dans les applications décisionnelles OLAP (*On Line Analytical Processing*)
- Les applications OLAP facilitent les prises de décisions liées à l'analyse des données conservées par une entreprise
- A rapprocher de OLTP (*Online Transaction Processing*) : mode d'utilisation pendant les TPs par exemple

R. Grin

SGBD

31

## Index bitmaps et Oracle

- Il faut la version « Entreprise » d'Oracle pour disposer des index bitmaps

R. Grin

SGBD

32

## Implémentation des index bitmaps

- Très différente de celle des B-arbres
- Tableau de bits avec autant de colonnes que de valeurs distinctes de la colonne indexée et autant de lignes que la table
- Un bit est à 1 si la ligne de la table a la valeur correspondant à la colonne
- Il ne faut pas trop de valeurs distinctes ni trop de modifications

R. Grin

SGBD

33

## Exemple d'index bitmap

Index sur le sexe des employés

nb colonnes =  
nb valeurs distinctes

- Pour trouver les employées, il suffit de parcourir la colonne F et de récupérer les 1

Identificateur	F	M
8999887	0	1
8999885	0	1
8999878	1	0
8999865	0	1
8999889	1	0
8999899	0	1

R. Grin

SGBD

34

## Utilisation des index bitmaps

- Il est facile de combiner différents tableaux de bits pour répondre aux sélections de type « **col1=valeur1 and col2=valeur2** » où **col1** et **col2** sont indexées par un index bitmap
- Lorsque les index bitmap sont utilisés par l'optimiseur ils peuvent fournir de très bonnes performances (opérations de type « et » sur des tableaux de bits)

R. Grin

SGBD

35

## Utilisation des index bitmaps

- Au contraire, ils ne sont pas performants s'ils comportent trop de valeurs distinctes ou s'ils doivent être modifiés souvent

R. Grin

SGBD

36

## Index join bitmaps

- Nouveau type d'index bitmap pour accélérer les jointures

## Enregistrement des données Gestion des accès concurrents

## Processus clients et serveurs

- Les SGBD s'appuient sur le mode client-serveur :
  - les interfaces du SGBD avec l'utilisateur ou avec les applications sont clientes de la partie serveur du SGBD : elles lui envoient des requêtes et le serveur renvoie les données résultats
  - les parties clientes et serveur du SGBD peuvent être sur des machines différentes ; elles utilisent un protocole réseau propriétaire pour communiquer

## Fichiers de la base

- Les données de la base sont enregistrées dans de très gros fichiers du système d'exploitation
- Le SGBD gère lui-même ses données ; il enregistre les tables, vues, index,... dans ces gros fichiers
- Le SGBD peut même se passer complètement du système d'exploitation hôte pour l'enregistrement des données (fichiers "raw", hors système de fichiers)

## Image « avant »

- Image « avant » : contient les informations pour remettre la base dans un état antérieur à une modification
- Peut être enregistré dans des fichiers système ou un emplacement spécial de la base (chez Oracle, segments de *rollback* écrits dans la base)

## Images « après »

- Images « après » : contient les informations pour refaire une modification à partir d'un état antérieur de la base
- Enregistré le plus souvent dans des fichiers système (fichiers *redo log* chez Oracle)

## Archivage des images « après »

- La place disque réservée aux images « avant » et « après » est limitée ; elle est recyclée pour enregistrer les dernières images
- Il peut être intéressant d'archiver sur des bandes magnétiques (ou autre) les images « après » pour les réutiliser en cas de panne
- On peut ainsi refaire toutes les dernières actions effectuées par les utilisateurs depuis la dernière sauvegarde

R. Grin

SGBD

43

- Les différents SGBD peuvent avoir des implantations très différentes sur certains points
- Ces différences sont souvent dues à la façon d'implémenter le traitement des accès concurrents
- C'est l'implantation d'Oracle (depuis la version 7) qui est décrite ici

R. Grin

SGBD

44

## Écriture dans la base

- Les lectures/écriture dans la base, utilisent des *buffers* en mémoire centrale
- Écriture asynchrone des données dans la base :
  - pour des raisons de performances, même un *commit* peut ne pas provoquer l'écriture dans la base de données placées dans les *buffers*
  - si les *buffers* sont pleins, des données non validées peuvent être enregistrées dans la base

R. Grin

SGBD

45

## Écriture dans la base

- L'écriture est effectuée par un seul processus (en version monoprocesseur)
- Quand une modification est effectuée par une transaction, les données de la base sont modifiées tout de suite (dans les *buffers* ou les tables de la base), sans attendre un *commit*

R. Grin

SGBD

46

## Segments de *rollback*

- Dans Oracle, les images « avant » sont enregistrées dans des segments de *rollback*
- Ces segments de *rollback* sont enregistrés dans une table spéciale de la base

R. Grin

SGBD

47

## Utilisation des segments de *rollback*

- Annuler les transactions
- Cacher aux transactions les données modifiées par les autres transactions non validées
- Assurer une lecture cohérente de la base aux requêtes et aux transactions « *read only* »

R. Grin

SGBD

48

## Données multi-versions

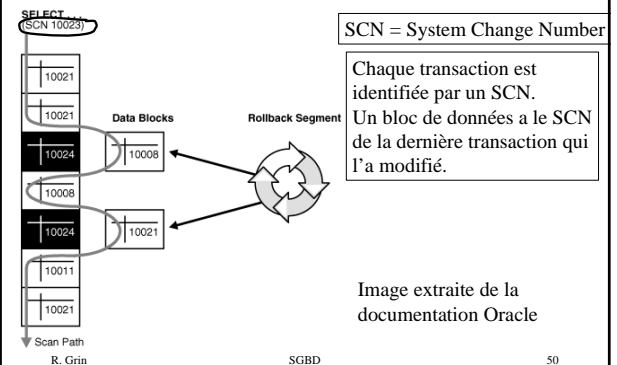
- Les segments de *rollback* permettent de fournir aux transactions différentes « versions » des données
- Si une transaction souhaite lire des données qui sont en cours de modification par une autre transaction, Oracle lui fournit une ancienne version de ces données

R. Grin

SGBD

49

## Multi-version



SGBD

50

## Inconvénient du multi-version

- En cas de modifications très nombreuses, ou de transaction « *read only* » très longue, Oracle peut manquer de segments de *rollback*, ce qui provoque l'abandon de transactions
- L'administrateur de la base peut augmenter les segments de *rollback*

R. Grin

SGBD

51

## Blocage des données lues

- La plupart des SGBD bloquent les écritures les données lues par une autre transaction (par un select), pour éviter les mises à jour perdues
- Oracle ne bloque pas les données lues par un select, sauf si on utilise la clause « *for update* »

R. Grin

SGBD

52

## Oracle version 6

- Avec Oracle version 6, les modifications effectuées par les utilisateurs n'étaient enregistrées dans les tables de la base qu'après un commit
- En attendant le commit, les modifications effectuées par un utilisateur étaient enregistrées à part dans la base

R. Grin

SGBD

53

## Fichiers *redo log*

- Les images « après » sont enregistrées dans des fichiers *redo log* (à partir des buffers *redo log*)
  - avant que les données ne soient vraiment modifiées dans les tables de la base
  - et au moins à chaque validation de transaction ; c'est cet enregistrement du *commit* qui « fait foi » pour savoir si une transaction a été validée
- L'écriture est séquentielle, donc rapide

R. Grin

SGBD

54

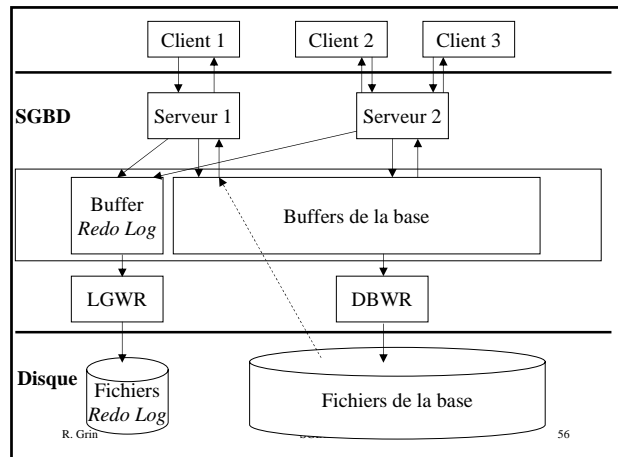
## Archivage des fichiers *redo log*

- L'écriture dans les fichiers *redo log* est circulaire : les premières données sont écrasées par les dernières quand les fichiers sont pleins
- On peut demander à Oracle d'archiver automatiquement les fichiers *redo log* avant qu'ils ne soient pleins
- Cet archivage pourra être utilisé en cas de panne

R. Grin

SGBD

55



R. Grin

56

## Reprise après panne

R. Grin

SGBD

57

## Types de pannes

- logicielles
- matérielles
- dues aux réseaux
  
- Les fichiers de la base peuvent avoir été endommagés ou pas

R. Grin

SGBD

58

## Reprise après panne quand les fichiers sont corrects

- Automatiquement, quand il redémarre, le SGBD
  - termine les transactions validées qui n'ont pu être enregistrées complètement dans la base
  - annule les transactions qui n'ont pas été validées (rollback, ou panne avant le commit)
- Pour cela, le SGBD utilise les images « après » et « avant »

R. Grin

SGBD

59

## Reprise après panne quand les fichiers sont endommagés

- L'administrateur doit commencer par recharger la dernière sauvegarde complète de la base
- Ensuite 2 cas :
  - si les images « après » de la base ont été archivées, on redéroule les actions enregistrées depuis la dernière sauvegarde
  - sinon, il faut relancer à la main toutes les commandes perdues (si on peut !)

R. Grin

SGBD

60

## Optimisation des requêtes

R. Grin

SGBD

61

## Traitement d'une requête

1. Analyse syntaxique (grammaire du langage) et sémantique (utilisation de la description des données de la base) de la requête
2. Contrôle des droits d'accès et des contraintes d'intégrité
3. Optimisation des requêtes : élaboration d'un plan pour répondre à la requête en utilisant le coût minimum (utilisation des index par exemple)
4. Exécution du plan

R. Grin

SGBD

62

## Optimiseur de requêtes

- SQL est un langage non procédural : on ne décrit pas comment obtenir le résultat
- L'optimiseur du SGBD va concevoir un plan pour aller rechercher les données de la manière la plus efficace
- Il doit comparer tous les plans possibles, ou une partie des plans sélectionnée par heuristique

R. Grin

SGBD

63

## Optimisation

- L'optimiseur prend en compte
  - des considérations logiques, comme de commencer par réduire la taille des données traitées (faire d'abord des sélections ou projections) avant de faire des jointures
  - l'implantation physique des données comme l'existence des index (et des clusters)
  - des statistiques sur les données contenues dans les tables (nombre de lignes, de valeurs différentes, etc.)

R. Grin

SGBD

64

## Bases de données réparties

R. Grin

SGBD

65

## Bases de données réparties

- Les processus qui fonctionnent dans un SGBD, sont souvent réparties sur plusieurs machines
- De plus en plus fréquemment les données sont elles-mêmes réparties sur plusieurs sites
- Cette répartition doit être le plus transparente possible pour l'utilisateur

R. Grin

SGBD

66

## COMMIT « distribués »

- L'implantation d'un COMMIT est complexe si la transaction porte sur des données réparties sur plusieurs sites distants
- Un problème de réseau peut survenir quand un COMMIT est lancé, et une machine distante peut ne jamais recevoir l'avis de COMMIT
- Le COMMIT à 2 phases permet de conserver les propriétés des transactions, même sur des bases réparties (ou des bases multiprocesseurs)

R. Grin

SGBD

67

## COMMIT à 2 phases

- Un des sites (le plus souvent celui qui a reçu l'ordre COMMIT) est le coordinateur de la manœuvre
- Le COMMIT va se dérouler en 2 étapes bien distinctes
- En fonctionnement normal, ce COMMIT à 2 phases est transparent pour l'utilisateur

R. Grin

SGBD

68

## COMMIT à 2 phases ; étape 1

1. Préparation du COMMIT : le coordinateur ordonne à tous les autres sites de préparer sa part du travail  
Si un site indique qu'il est prêt, il assure qu'il peut faire les modifications, même s'il tombe en panne avant de recevoir l'ordre final du coordinateur  
Pour avoir cette assurance, les sites enregistrent à ce moment les modifications à faire dans les images « après »

R. Grin

SGBD

69

## COMMIT à 2 phases ; étape 2

2. Exécution du COMMIT (ou du ROLLBACK) :
  - si tous les sites répondent qu'ils sont prêts pour le COMMIT, le coordinateur
    - a. enregistre le COMMIT dans ses images « après »
    - b. lance à tous les sites l'ordre d'effectuer le COMMIT
  - si un des sites ne répond pas ou indique qu'il ne pourra effectuer le COMMIT, le coordinateur ordonne un ROLLBACK à tous les sites

R. Grin

SGBD

70

## COMMIT à 2 phases ; traitement des pannes

- Si un des sites qui a indiqué qu'il était prêt, tombe en panne avant qu'il ne reçoive l'avis de validation ou d'annulation du coordinateur,
- Lors de la reprise après la panne, ce site va
  - s'apercevoir qu'il n'a pas reçu l'ordre final du coordinateur
  - s'informer auprès du coordinateur de sa décision
  - effectuer un COMMIT ou un ROLLBACK

R. Grin

SGBD

71

## Réplication des données

- Une difficulté des bases réparties est la perte de performance due à la lenteur des réseaux
- Pour les sites confrontés à ce problème, il est possible de ne gérer en temps réel que les données locales, en utilisant des copies des données distantes
- On peut automatiser la réplication de ces données distantes à intervalles réguliers

R. Grin

SGBD

72