Aspects Systèmes

S. Lopes, complété par L.Yeh

January 28, 2021

Content

SECTION Le matériel

- Sous-système d'E/S

Le matériel

- Principaux composants
  - Processeur : un ou plusieurs, différentes fréquences d'horloge
  - Mémoire : utilisée comme cache de données, évite les accès disque couteux
  - E/S : système d'E/S, contrôleur de disque, disques, différentes technologies
  - Réseau : largeur de bande, vitesse

Le matériel

- Principaux composants

  Processeur : un ou plusieurs, différentes fréquences d'horloge

  Mémoire : utilisée comme cache de données, évite les accès disque couteux

  E/S : système d'E/S, contrôleur de disque, disques, différentes technologies

  Réseau : largeur de bande, vitesse

- Pas indépendant du logiciel: une ressource matérielle surchargée ne nécessite pas forcément d'acheter plus de ce matériel. Il peut être préférable de réécrire une requête ou d'ajouter un index pour diminuer la charge.

Sous-système d'E/S

- Sous-système crucial dans le cas d'une BD
  - Prévention des pertes de données
  - Capacité adaptée et possibilité de passage à l'échelle
  - Accès efficace aux données
  - Tolérance aux pannes et réparations rapides
  - Ajout ou suppression de disques sans interruption
  - Bon rapport efficacité/coût

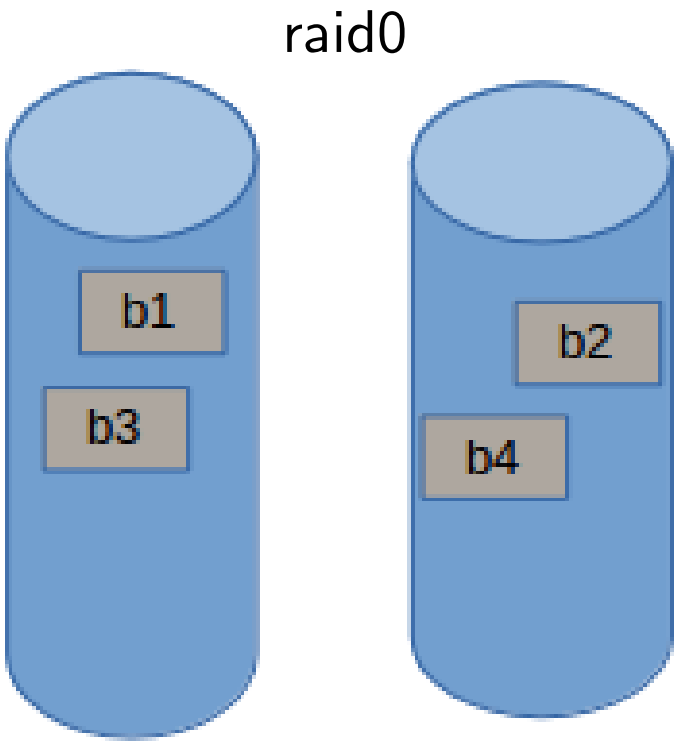
Sous-système d'E/S

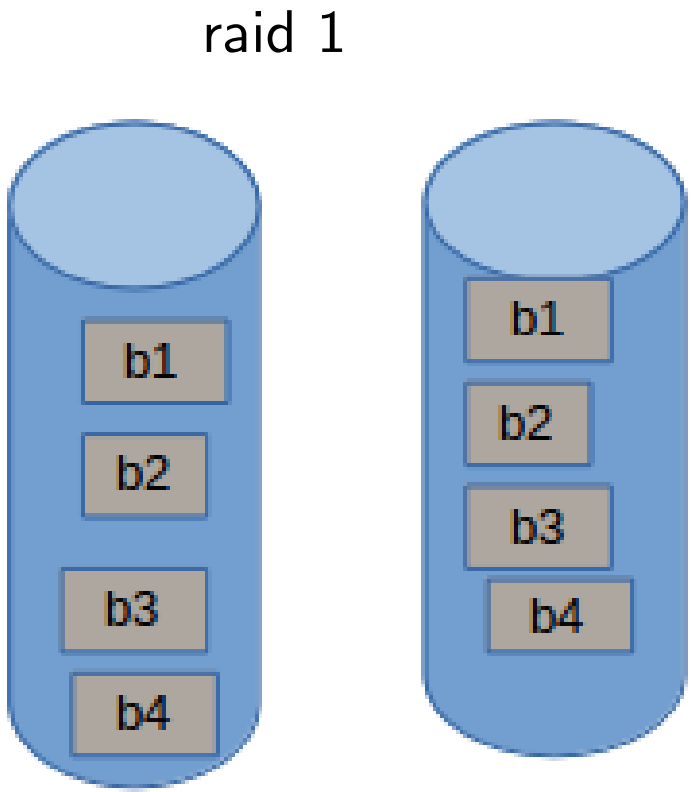
- Sous-système crucial dans le cas d'une BD
  - Prévention des pertes de données
  - Capacité adaptée et possibilité de passage à l'échelle
  - Accès efficace aux données
  - Tolérance aux pannes et réparations rapides
  - Ajout ou suppression de disques sans interruption
  - Bon rapport efficacité/coût
- Plusieurs étapes pour tuner les E/S
  1. Configuration des disques (niveau RAID)
  2. Utilisation du cache du controleur
  3. Planification et dimensionnement
  4. Configuration des volumes logiques: couche logiciel qui permet de regrouper plusieurs disques physiques en un disque logique.

Niveaux RAID

- RAID = Redundant Arrays of Inexpensive Disks
- Intérêt des *piles* (*grappes*) de disques (*disk array*)
  - tolérance aux pannes (redondance sur plusieurs disques)
  - rendement amélioré (parallélisme au niveau du contrôleur)

RAID 0 et Raid 1





RAID 0: Striping

- Données découpées en bloc et chaque bloc est placé sur un disque différents

Example 1 *8 disques, taille des blocs 1Ko. Le premier Ko de données se trouve sur le premier disque. Le deuxième Ko de données se trouve sur le deuxième disque. . . . Le huitème Ko de données se trouve sur le huitième disque. Le neuvième Ko de données se trouve sur le premier disque. . . .*

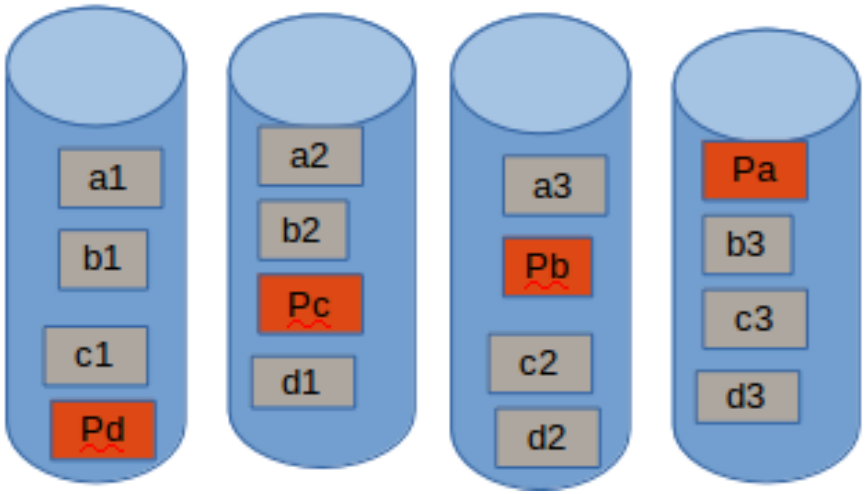
- La charge est répartie sur plusieurs disques et/ou contrôleur
- Les meilleurs performances sont obtenus si les données sont réparties entre différents contrôleurs qui ne gère chacun qu'un disque
- Une lecture/écriture = plusieurs lectures/écritures physiques en parallèle
- Pas de redondance (pas vraiment RAID) donc pas de tolérance aux pannes

RAID 1: Mirroring

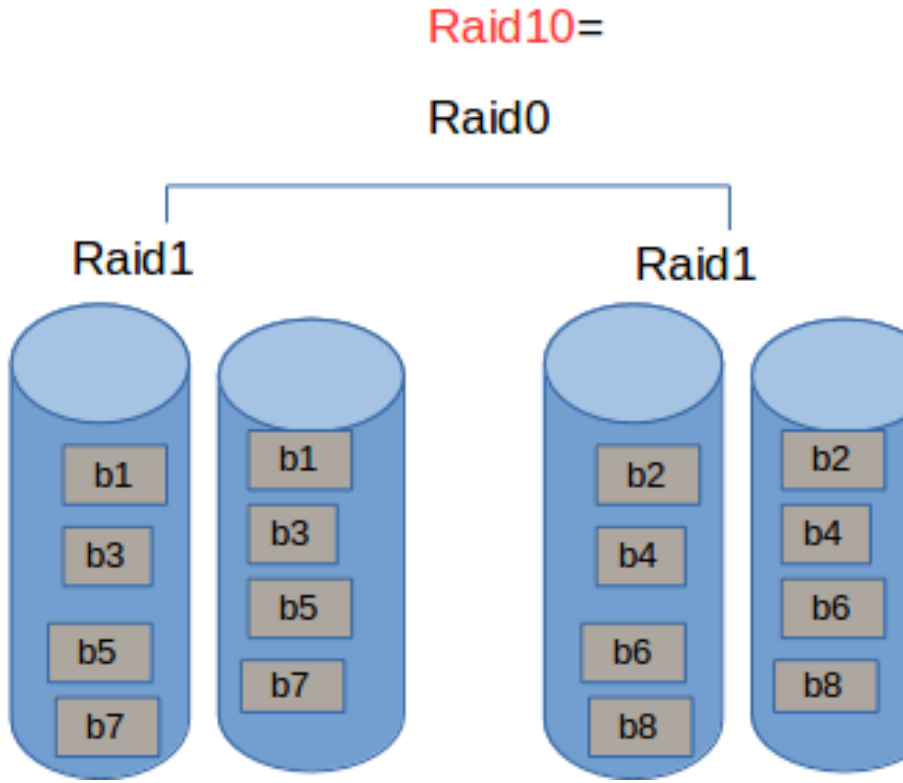
- Chaque disque est répliqué sur un autre
- Tolérance aux pannes si un disque est défectueux
- Permet de doubler le taux de lecture
- Une écriture = deux écritures physiques
- Une lecture = une lecture sur le disque le moins occupé ou deux lectures en parallèle (se termine quand le plus rapide répond)

RAID 5 et Raid 10

RAID 5



RAID 10



Niveaux RAID

RAID 5: Rotated parity striping

- Les données sont réparties sur plusieurs disques
- Les blocs ont des parités réparties également sur les disques (pas de disque dédié aux parités)
- Tolérance aux pannes basée sur la correction d'erreur
- Une écriture = deux lectures physiques (ancienne valeur et ancienne parité) + deux écritures physiques (nouvelle valeur et nouvelle parité) (PNew = (VOld xor VNew) xor POld)
- Le cache du contrôleur permet généralement de limiter cette inconvénient

RAID 10: Striped mirroring

- Pile de disques en RAID 0 dont chaque segment est lui-même en RAID 1
- Même tolérance aux pannes que RAID 1
- Bonnes performances dues au *striping*
- Nécessite beaucoup de disques

Choix du niveau RAID

- Taille d'un bloc RAID: même taille qu'une page de la BD (unité de transfert minimum $\Rightarrow$ accès à un seul disque)

Choix du niveau RAID

- Taille d'un bloc RAID: même taille qu'une page de la BD (unité de transfert minimum $\Rightarrow$ accès à un seul disque)
- Niveau RAID
  - **Fichiers de log** (et tablespace des segments de rollback sous Oracle): RAID 1
    - l'écriture des log étant séquentiel et synchrone, elle bénéficie peu du *striping*
    - RAID 10 si beaucoup d'écriture
    - éviter RAID 5 à cause des pénalités à l'écriture
  - **Fichiers temporaires**: RAID 0
    - le système peut généralement tolérer la perte des données
  - **Fichiers de données et index**: RAID 5
    - bonne tolérance aux pannes
    - prédominance des lecture par rapport aux écritures

Choix du niveau RAID

- Taille d'un bloc RAID: même taille qu'une page de la BD (unité de transfert minimum $\Rightarrow$ accès à un seul disque)
- Niveau RAID
  - **Fichiers de log** (et tablespace des segments de rollback sous Oracle): RAID 1
    - l'écriture des log étant séquentiel et synchrone, elle bénéficie peu du *striping*
    - RAID 10 si beaucoup d'écriture
    - éviter RAID 5 à cause des pénalités à l'écriture
  - **Fichiers temporaires**: RAID 0
    - le système peut généralement tolérer la perte des données
  - **Fichiers de données et index**: RAID 5
    - bonne tolérance aux pannes
    - prédominance des lecture par rapport aux écritures
- RAID matériel ou logiciel: lié au prix

Utilisation du Cache du Controleur

- Les contrôleurs de disque dispose de mémoire cache pouvant être utilisée en lecture et/ou en écriture
- Cache en lecture *read-ahead* :
  - place plus de données dans le cache que nécessaire
  - intéressant pour des lectures séquentielles de gros volumes de données
  - peu utile pour les accès aléatoire
  - **non recommandé dans le cas d'un SGBD car ce dernier fournit un équivalent plus performant**

Utilisation du Cache du Controleur: **Cache en écriture**

write-through (mode classique): l'écriture se termine quand les données sont écrites sur le disque

Utilisation du Cache du Controleur: **Cache en écriture**

write-through (mode classique): l'écriture se termine quand les données sont écrites sur le disque

write-back : l'écriture se termine quand les données sont dans le cache.

Utilisation du Cache du Controleur: **Cache en écriture**

write-through (mode classique): l'écriture se termine quand les données sont écrites sur le disque

write-back : l'écriture se termine quand les données sont dans le cache.

- le contrôleur doit assurer l'écriture des données sur le disque même en cas d'interruption de l'alimentation
- write-back semble être le plus performant (c'est le cas si le contrôleur est peu chargé)

Utilisation du Cache du Controleur: **Cache en écriture**

write-through (mode classique): l'écriture se termine quand les données sont écrites sur le disque

write-back : l'écriture se termine quand les données sont dans le cache.

- le contrôleur doit assurer l'écriture des données sur le disque même en cas d'interruption de l'alimentation
- write-back semble être le plus performant (c'est le cas si le contrôleur est peu chargé)
- *write-through* peut être préférable dans le cas d'un contrôleur chargé connecté à plusieurs disques

Utilisation du Cache du Controleur: **Cache en écriture**

write-through (mode classique): l'écriture se termine quand les données sont écrites sur le disque

write-back : l'écriture se termine quand les données sont dans le cache.

- le contrôleur doit assurer l'écriture des données sur le disque même en cas d'interruption de l'alimentation
- write-back semble être le plus performant (c'est le cas si le contrôleur est peu chargé)
- *write-through* peut être préférable dans le cas d'un contrôleur chargé connecté à plusieurs disques
- *write-through* bénéficie du parallélisme pour l'accès aux disques alors que les écritures en mode write-back se font en série quand le cache est plein

Utilisation du Cache du Controleur: **Cache en écriture**

write-through (mode classique): l'écriture se termine quand les données sont écrites sur le disque

write-back : l'écriture se termine quand les données sont dans le cache.

- le contrôleur doit assurer l'écriture des données sur le disque même en cas d'interruption de l'alimentation
- write-back semble être le plus performant (c'est le cas si le contrôleur est peu chargé)
- *write-through* peut être préférable dans le cas d'un contrôleur chargé connecté à plusieurs disques
- *write-through* bénéficie du parallélisme pour l'accès aux disques alors que les écritures en mode write-back se font en série quand le cache est plein
- généralement le mode *write-back* n'est pas recommandé dans le cas d'un serveur de BD chargé

Dimensionnement du Système de Stockage

- Nombre de disques (nombre de disques logiques = ce que le SGBD voit)
  - un fichier de **log** (ou rollback segment) par disque (plus éventuellement la redondance)
  - les **index secondaires** ne devraient pas se trouver sur les mêmes disques que les données
  - le **catalogue** devrait se trouver sur un disque différents des données utilisateurs
  - on calcul ensuite la bande passante pour chaque disque pour voir si c'est suffisant (fournit le nombre minimum de disques)

Dimensionnement du Système de Stockage

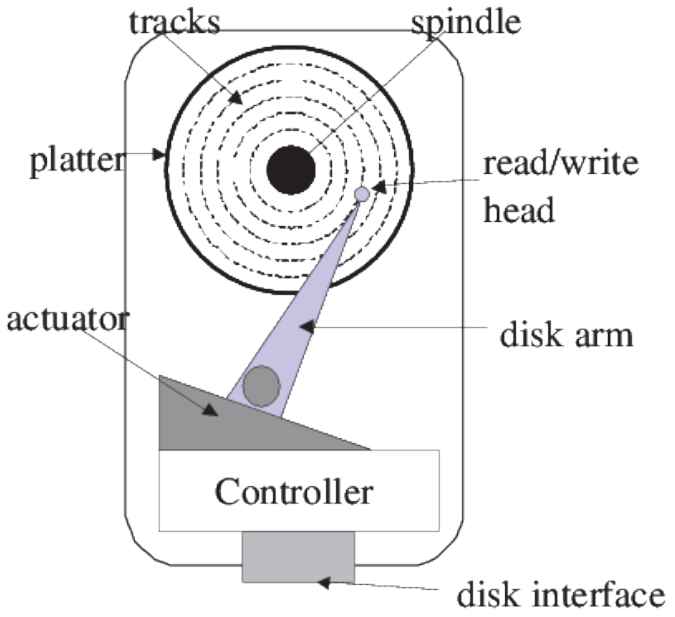
- Nombre de disques (nombre de disques logiques = ce que le SGBD voit)
  - un fichier de **log** (ou rollback segment) par disque (plus éventuellement la redondance)
  - les **index secondaires** ne devraient pas se trouver sur les mêmes disques que les données
  - le **catalogue** devrait se trouver sur un disque différents des données utilisateurs
  - on calcul ensuite la bande passante pour chaque disque pour voir si c'est suffisant (fournit le nombre minimum de disques)
- Caractéristiques des disques
  - les paramètres des disques (*seek time*, *rotational delay*, ...) influence évidemment les performances du sous-système d'E/S

Dimensionnement du Système de Stockage

- Caractéristiques des interconnexions
  - **interface SCSI**
    - très utilisées pour les SGBD
    - pas de tolérances aux pannes
    - limité en nombre de périphériques (au plus 16 par bus)
    - vitesse à un degré moindre
  - **Storage-Area Network (SAN)** comme *fiber channel*
    - solution de stockage réseau
    - jusqu'à 126 périphériques par boucles
    - jusqu'à 200 boucles
    - 106Mo/s
    - tolérance aux pannes
    - très bien adapté aux BD
  - **Network-Attached Storage (NAS)**
    - solution de stockage partagé
    - gestion simple
    - peu adapté dans le cas d'une BD

Caractéristiques des disques

- Un disque est une collection de plateaux (les uns au dessus des autres) en rotation autour d'un axe commun (*spindle*)
- Chaque plateau possède deux faces (sauf le premier et le dernier)
- Chaque face est divisée en pistes (*track*) disposées en cercles concentriques
- Chaque face est associée à une tête de lecture/écriture (*head*)
- L'accès à une donnée de la piste i nécessite le positionnement de la tête sur la piste i
- Toutes les têtes sont solidaires (cylindre i = ensemble des pistes i)



Amélioration de la Configuration Matérielle

Ajout de mémoire

- Peu coûteux
- Permet d'augmenter la taille des caches et donc de diminuer l'activité disque
- Possible d'ajouter de la mémoire cache au niveau du controleur de disque

Amélioration de la Configuration Matérielle

Ajout de mémoire

- Peu coûteux
- Permet d'augmenter la taille des caches et donc de diminuer l'activité disque
- Possible d'ajouter de la mémoire cache au niveau du controleur de disque

Ajout de disques

- Largeur de bande est généralement plus critique que la capacité
- Permet de placer le *log* sur un disque différent
- Permet de changer le niveau RAID (exemple, de RAID 5 à RAID 10)
- Permet de **partitionner** les données sur plusieurs disques
  - pour les applications orientées écriture (*write intensive*), séparer les index secondaires des données (car les modifications s'appliquent à la fois sur la table et sur les index)
  - pour les applications orientées lecture (*read intensive*), partitionner les tables fréquemment accédées

Amélioration de la configuration matérielle

Ajout de processeurs

- Peut être nécessaire pour gérer de grandes quantités de données
- Différentes architectures. Plusieurs processeurs:
  - une mémoire, un ensemble de disques (*tightly coupled*)
  - chacun avec disques et mémoire (*shared nothing*)
  - chacun sa mémoire, disques partagés (*shared disk*)
- Architecture la moins onéreuse: *shared nothing*
  - machines connectées par un réseau rapide (*cluster*)
  - adapté pour les situations suivantes
    - une partie de l'application (par exemple GUI) peut être séparé du reste
    - OLTP séparé du décisionnel (sur des données moins à jour)
    - partitionnement des données (une transaction longue sera répartie sur plusieurs sites)

SECTION Système d'Exploitation

- Introduction: Paramètres Impactant les Performances du SGBD
- Gestion des processus
- Gestion des Fichiers

Introduction: Paramètres Impactant les Performances du SGBD

- **Gestion des processus** (et des *threads*)
  - temps pour passer d'un processus à un autre (le plus faible possible)
  - quantum de temps pour les processus du SGBD (le plus grand possible)
  - priorités (les processus du SGBD doivent tous avoir la même priorité)
  - nb de processus utilisateurs pouvant accéder simultanément au SGBD
- **Gestion de la mémoire** (physique et virtuelle)
  - portion de la mémoire virtuelle partagée par les processus du SGBD
  - portion de la mémoire allouée pour le cache du SGBD
- **Gestion des fichiers**
  - fichiers réparties sur plusieurs disques (grosse BD)
  - fichiers formés de blocs contigus (performance des scans)
  - lecture en avant (*lookahead*) (performance des scans)
  - accés aux pages d'un gros fichier plus coûteux que l'accès aux pages d'un petit (nombre d'indirections)
- **Gestion du temps**
  - savoir si une application est limitée par les E/S ou le CPU

Changement de contexte

- Utilisation du processeur inutile pour l'application $\Rightarrow$ à minimiser
- Choix d'un SE adapté (*lightweight thread switching*)
- Choix d'un quantum de temps suffisament important
  - améliore le taux de sortie (*throughput*) au détriment du temps de réponse
  - un quantum d'une seconde est un bon compromis

Priorités

- Fixer une priorité inférieure pour les processus du SGBD par rapport aux autres applications est un mauvais choix évident
- Fixer des priorités différentes pour certains processus important peut avoir des conséquences néfastes (inversion de priorité, voir figure suivante)

Priorités (Inversion de priorité)

Exemple [Wikipedia]:

1. La tâche A (de haute priorité) est mise en attente sur un verrou X. Elle n'a donc plus accès à la ressource processeur.
2. La tâche B (de basse priorité) acquiert un verrou Y.
3. La tâche B est préemptée car la tâche A vient d'obtenir le verrou X.
4. La tâche A essaye d'obtenir le verrou Y ; comme il est déjà acquis par la tâche B, la tâche A est donc mise en attente.

- une solution existe sur certains systèmes (*héritage de priorité*): un processus qui obtient un verrou voit sa priorité augmentée au plus haut niveau de priorité des processus bloqués
- si système n'offre pas de protection, même priorité pour tous les processus
- sinon, possible de donner une priorité plus élevée pour OLTP et moins élevée pour les traitements batch

Nombre de Processus Concurrents

- Trop de processus concurrents provoque une dégradation des performances
  - la mémoire utilisée par les processus utilisateurs **dépasse la RAM disponible** (utilisation du disque par le SE (*swaping*))
  - **conflits de verrouillage** dus au nombre de transactions concurrentes

Nombre de Processus Concurrents

- Trop de processus concurrents provoque une dégradation des performances
  - la mémoire utilisée par les processus utilisateurs **dépasse la RAM disponible** (utilisation du disque par le SE (*swaping*))
  - **conflits de verrouillage** dus au nombre de transactions concurrentes
- Réglage
  1. fixer une borne minimum pour le nombre de transactions concurrentes
  2. augmenter cette borne
  3. si les performances augmentent, continuer

Nombre de Processus Concurrents

- Trop de processus concurrents provoque une dégradation des performances
  - la mémoire utilisée par les processus utilisateurs **dépasse la RAM disponible** (utilisation du disque par le SE (*swaping*))
  - **conflits de verrouillage** dus au nombre de transactions concurrentes
- Réglage
  1. fixer une borne minimum pour le nombre de transactions concurrentes
  2. augmenter cette borne
  3. si les performances augmentent, continuer
- Si le profil des transactions change au cours du temps, le nombre de processus doit être adapté en conséquence

Gestion des Fichiers

- **Taille d'un bloc**
  - plutôt grande pour l'historique (log)
  - plutôt petite pour les accès aléatoires (meilleur gestion de l'occupation)
- **Mise en cache en écriture** peut être gênante (risque de perte de données)
- **Nombre d'indirections** pour accéder à un bloc
  - sous UNIX, les accès à des blocs en début de fichier utilisent moins d'indirections que les accès à des blocs en fin de fichier
  - peut être confié au SGBD (gestion de fichiers propriétaire (*RAW*))

SECTION Le SGBD

- Cache de Données
- Contrôle de Concurrence
- Transactions sous Oracle
- Reprise

Cache de Données

- Cache de données minimise les accès disques .
- Cas extrème: si la BD tient en mémoire, possible de la charger au démarrage
- Terminologie: *buffer*, *database buffers*, *database cache*
- Contient des données partagées par les différentes transactions

Cache de Données

- Trois paramètres importants
  1. **lecture/écriture logique**
     - pages auxquel le SGBD accède par des appels systèmes.
     - certaines pages sont dans le cache, d'autres nécessitent un accès disque.
  2. **SGBD page replacement**
     - écriture physique quand une page doit être placé dans le buffer, que le buffer est plein et que les pages occupées ne sont pas encore sur le disque
     - doit arriver rarement: le disque doit être maintenu le plus à jour possible
  3. **SE paging** (*swapping*)
     - accès physique au disque quand une partie du cache se trouve en dehors de la RAM
     - doit impérativement être évité

Cache de Données

- Mesure du nombre d'accès logique par rapport au nombre d'accès physiques:

$$HitRatio = \frac{(NbAccesLog - NbAccesPhy)}{NbAccesLog}$$

Cache de Données

- Mesure du nombre d'accès logique par rapport au nombre d'accès physiques:

$$HitRatio = \frac{(NbAccesLog - NbAccesPhy)}{NbAccesLog}$$

- **Régler la taille du cache de données permet d'ajuster le hit ratio**
  - exécuter une charge typique pendant une certaine durée (une heure par exemple)
  - vérifier si le *hit ratio* est trop bas (90% si le système a beaucoup de mémoire, moins si la BD est de grande taille ou s'il y a beaucoup d'accès aléatoire)
  - la stratégie consiste à augmenter la taille du cache jusqu'à ce que le *hit ratio* se stabilise tout en gardant (2) et (3) bas (i.e. *SGBD page replacement*, *SE paging*)

Cache de Données

- Mesure du nombre d'accès logique par rapport au nombre d'accès physiques:

$$HitRatio = \frac{(NbAccesLog - NbAccesPhy)}{NbAccesLog}$$

- **Régler la taille du cache de données permet d'ajuster le hit ratio**
  - exécuter une charge typique pendant une certaine durée (une heure par exemple)
  - vérifier si le *hit ratio* est trop bas (90% si le système a beaucoup de mémoire, moins si la BD est de grande taille ou s'il y a beaucoup d'accès aléatoire)
  - la stratégie consiste à augmenter la taille du cache jusqu'à ce que le *hit ratio* se stabilise tout en gardant (2) et (3) bas (i.e. *SGBD page replacement*, *SE paging*)
- **Ajouter de la RAM devient utile si**
  - l'augmentation de la taille du cache avec la quantité de RAM actuelle fait augmenter le *paging*

Contrôle de Concurrence

- Une application de BD est un ensemble de transactions
- Hypothèse du programmeur: chaque transaction s'exécute en *isolation*
- Problème: ce n'est pas le cas !
- Choix (compromis entre performance et correction)
  - nombre de verrous par transaction (à minimiser)
  - type de verrous (en lecture de préférence)
  - durée pendant laquelle une transaction garde un verrou (à minimiser)

Contrôle de Concurrence: Correction

- Deux transactions sont *concurrentes* si elles se recouvrent dans le temps

Contrôle de Concurrence: Correction

- *Contrôle de concurrence*: **contrôler les interactions entre transactions concurrentes**, i.e. l'exécution d'un ensemble de transaction doit être équivalente (réarrangement, lit et écrit les mêmes valeurs) à une exécution en série (*sérialisabilité*)

Contrôle de Concurrence: Correction

- Moyen pour y parvenir: l'*exclusion mutuelle* (obtenue par exemple avec des *sémaphores* dans les SE)

Contrôle de Concurrence: Correction

- Moyen pour y parvenir: l'*exclusion mutuelle* (obtenue par exemple avec des *sémaphores* dans les SE)
- Cas extrême: un unique sémaphore pour toute la BD (correct mais peu performant), pas de sémaphore (incorrect mais très performant)

Contrôle de Concurrence: Correction

- Moyen pour y parvenir: l'*exclusion mutuelle* (obtenue par exemple avec des *sémaphores* dans les SE)
- Cas extrême: un unique sémaphore pour toute la BD (correct mais peu performant), pas de sémaphore (incorrect mais très performant)
- Un compromis peut être trouvé: le *verrouillage*

Contrôle de Concurrence: Correction

- Deux types de verrous: en lecture (partagé) et en écriture (exclusif)

Contrôle de Concurrence: Correction

- Deux types de verrous: en lecture (partagé) et en écriture (exclusif)
- Acquisition et libération de verrous: algorithme de verrouillage à 2 phases.
  - une transaction doit posséder un verrou sur x avant d'accéder à x, et il ne doit pas acquérir de nouveaux verrous après en avoir libéré un

Contrôle de Concurrence. Exemple Problème

Example 2

- *Trois comptes: A (4000), B (100), C (0)*
- *T_1 effectue un virement de 1000 de A vers C,*
- *T_2 calcule la somme des soldes des comptes*

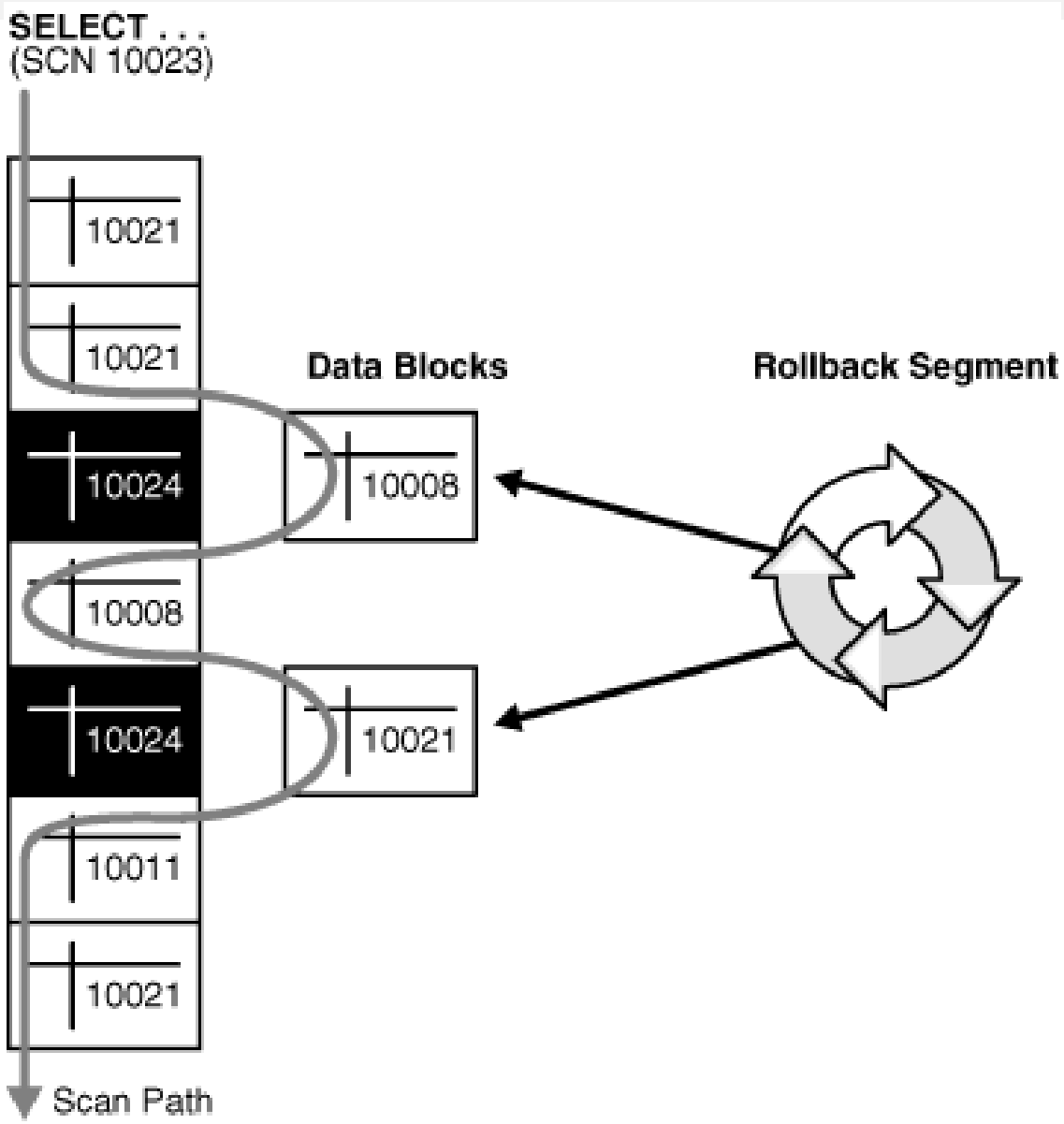
1. *T_2 obtient un verrou sur A, lit 4000 et relâche le verrou*
2. *T_1 obtient un verrou sur A, soustrait 1000, obtient un verrou sur C et ajoute 1000*
3. *T_2 obtient un verrou sur B, lit 100, obtient un verrou sur C et lit 1000*

$\Rightarrow$ *La somme totale est alors fausse (5100 au lieu de 4100)*

Attention: le comportement par défaut de la plupart des systèmes est de relacher les verrous en lecture immédiatement après la lecture (isolation de niveau 2).

Transactions sous Oracle: Estampillage multi-version

- Le contrôle de concurrence sous Oracle est basé sur un estampillage multi-version et des techniques de verrouillage



Transactions sous Oracle

- Une transaction débute avec la première instruction SQL et se termine explicitement par `COMMIT` ou `ROLLBACK` ou implicitement lors d'un ordre DDL
- Oracle gère l'annulation au niveau de la requête, i.e. une requête qui échoue n'a aucun effet
- Chaque transaction est associée à un segment rollback
- Terminaison d'une transaction
  - `COMMIT` ou `ROLLBACK` sans `SAVEPOINT`
  - ordre DDL (valide les requêtes précédentes, exécute et valide l'ordre DDL)
  - déconnexion = validation
  - terminaison anormale = annulation
- La première instruction suivant la fin d'une transaction en commence une autre

Transactions sous Oracle

- **Pendant l'exécution d'une transaction**
  1. des entrées d'annulation dans le cache des *segments rollback* sont allouées
  2. des entrées *redo log* dans le cache *redo log*
  3. changements dans le cache de données

Transactions sous Oracle

- **Pendant l'exécution d'une transaction**
  1. des entrées d'annulation dans le cache des *segments rollback* sont allouées
  2. des entrées *redo log* dans le cache *redo log*
  3. changements dans le cache de données
- **Validation**
  1. le processus LGWR écrit les entrées du buffer *redo log* sur disque
  2. libération des verrous
  3. marquer la transaction comme terminée

Transactions sous Oracle

- **Pendant l'exécution d'une transaction**
  1. des entrées d'annulation dans le cache des *segments rollback* sont allouées
  2. des entrées *redo log* dans le cache *redo log*
  3. changements dans le cache de données
- **Validation**
  1. le processus LGWR écrit les entrées du buffer *redo log* sur disque
  2. libération des verrous
  3. marquer la transaction comme terminée
- **Annulation**
  1. annulation des changements de chaque ordre SQL à l'aide des segments de *rollback*
  2. libération des verrous
  3. terminaison de la transaction

Transactions sous Oracle

- SAVEPOINT: Point de contrôle
  - divise une transaction longue en morceaux
  - possibilité d'annulation des requêtes SQL après un point de contrôle sans annuler complètement la transaction
  - perte des points de contrôle après le point de contrôle
  - libération des verrous après le point de contrôle
  - la transaction reste active

Example 3

```
SAVEPOINT pt1;
...
ROLLBACK TO SAVEPOINT pt1;
```

Transactions sous Oracle

- SAVEPOINT: Point de contrôle
  - divise une transaction longue en morceaux
  - possibilité d'annulation des requêtes SQL après un point de contrôle sans annuler complètement la transaction
  - perte des points de contrôle après le point de contrôle
  - libération des verrous après le point de contrôle
  - la transaction reste active

Example 3

```
SAVEPOINT pt1;
...
ROLLBACK TO SAVEPOINT pt1;
```

- Oracle supporte les niveaux d'isolation 2 (*read committed* par défaut) et 3 (*serializable*) plus un mode lecture seule

Transactions sous Oracle

- SAVEPOINT: Point de contrôle
  - divise une transaction longue en morceaux
  - possibilité d'annulation des requêtes SQL après un point de contrôle sans annuler complètement la transaction
  - perte des points de contrôle après le point de contrôle
  - libération des verrous après le point de contrôle
  - la transaction reste active

Example 3

```
SAVEPOINT pt1;
...
ROLLBACK TO SAVEPOINT pt1;
```

- Oracle supporte les niveaux d'isolation 2 (*read committed* par défaut) et 3 (*serializable*) plus un mode lecture seule
- Oracle assure la cohérence des lectures au niveau de la requête, i.e. les données lues par une requête proviennent du même point dans le temps (début de la requête)

Transactions sous Oracle

- Oracle assure la cohérence des lectures au niveau de la transaction en mode sérialisable

Transactions sous Oracle

- Oracle assure la cohérence des lectures au niveau de la transaction en mode sérialisable
- On peut fixer le mode d'isolation en début de transaction

Example 4

```
-- Niveau 2 d'isolation
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
-- Niveau 3 d'isolation
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
-- Cohérence des lectures au niveau de la transaction
SET TRANSACTION READ ONLY;
-- Cohérence des lectures au niveau de l'instruction
SET TRANSACTION READ WRITE;
```

Transactions sous Oracle

- Oracle assure la cohérence des lectures au niveau de la transaction en mode sérialisable
- On peut fixer le mode d'isolation en début de transaction

Example 4

```
-- Niveau 2 d'isolation
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
-- Niveau 3 d'isolation
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
-- Cohérence des lectures au niveau de la transaction
SET TRANSACTION READ ONLY;
-- Cohérence des lectures au niveau de l'instruction
SET TRANSACTION READ WRITE;
```

- ou pour une session

Example 5

```
ALTER SESSION SET ISOLATION LEVEL READ COMMITTED;
ALTER SESSION SET ISOLATION LEVEL SERIALIZABLE;
```

Transactions sous Oracle: Verrouillage

- deux modes de verrouillage (partagé et exclusif)
- les verrous sont acquis pour toute la durée de la transaction
- détection des *deadlocks* puis annulation de l'une des requêtes impliquées
- pas de verrous en lecture
- granularité au niveau du tuple ou de la table
- pas de changement automatique du niveau de granularité

Transactions sous Oracle: Types de verrous (DML, DDL, interne/latch, ...)

- **Row lock** (TX)
  - `INSERT, UPDATE, DELETE, SELECT FOR UPDATE`: `SELECT name FROM CUSTOMER FOR UPDATE OF CUSTOMER;`
  - exclusif
  - demande aussi un verrou de type table
- **Table lock** (TM)
  - `INSERT, UPDATE, DELETE, SELECT FOR UPDATE, LOCK TABLE`
  - du moins contraignant au plus contraignant
  - *row share* (RS): `SELECT FOR UPDATE`, verrouillage du tuples avec intention d'écriture. `LOCK TABLE machin IN ROW SHARE MODE`
  - *row exclusive* (RX): `INSERT, UPDATE, DELETE`, un ou plusieurs tuples ont été modifiés. `LOCK TABLE machin IN ROW EXCLUSIVE MODE`
  - *share* (S): `LOCK TABLE machin IN SHARE MODE`
  - *share row exclusive* (SRX): `LOCK TABLE machin IN SHARE ROW EXCLUSIVE MODE`
  - *exclusive* (X): `LOCK TABLE machin IN EXCLUSIVE MODE`

Différentes Actions Possibles

Les points suivants vont être détaillés:

1. Utiliser les fonctionnalités spécifiques des SGBD pour les transactions longues en lecture
2. Éviter le verrouillage si possible (cf après)
3. Morceler les transactions si possible (cf après)
4. Diminuer le niveau d'isolation si l'application le permet (cf après)
5. Choisir le granule de verrouillage approprié (cf après)
6. Exécuter les ordres DDL lors de périodes creuses
7. Partitionner
8. Eliminer les points chauds
9. Paramétrer l'interval de *deadlock*

Attention: les choix 2, 3 et 4 peuvent faire disparaître la guarantie d'isolation

Eliminer les Verrouillages Inutiles

- On peut trouver des cas où le verrouillage est inutile
  - une seule transaction à la fois (par exemple lors du chargement de la BD)
  - uniquement des transactions en lecture (par exemple pour du décisionnel)

Découpage de Transactions

- La longueur des transactions est un paramètre important
  - Plus une transaction demande de verrous, plus elle risque d'attendre qu'une autre transaction libère un verrou
  - Plus une transaction T est longue, plus une autre transaction devra attendre un verrou détenu par T

  $\Rightarrow$ Les transactions courtes sont préférables
- Il est parfois possible de découper une transaction sans perdre les garanties d'isolation
- Le fait qu'une transaction T puisse être découpée dépend de ce qui est concurrent avec T

Découpage de Transactions

- Deux questions
  - Est-ce que les transactions concurrentes avec T font produire à T un résultat incohérent ou voir une valeur incohérente si T est découpée ?
  - Est-ce que les transactions concurrentes avec T deviendront incohérente si T est découpée ?

Découpage de Transactions

- Deux questions
  - Est-ce que les transactions concurrentes avec T font produire à T un résultat incohérent ou voir une valeur incohérente si T est découpée ?
  - Est-ce que les transactions concurrentes avec T deviendront incohérente si T est découpée ?
- **Principe**: si T accède à X et Y mais T' accède au plus à l'un des deux et à rien d'autre alors T peut être divisée en deux transactions, l'une accédant à X, l'autre à Y.

Example 6 *Différentes agences d'une banque, T_1 calcule la somme des soldes des comptes, T_2 accède aux comptes d'une agence. T_1 peut être découpée par agence car T_2 n'accède qu'à une agence.*

Diminuer le Degré d'Isolation: Types de conflits

- **Conflit écriture/lecture (WR) (lecture sale)**
  - T_1 transfère une somme de A vers B, T_2 incrémente A et B de 10%
  - $R_1(A), W_1(A)$(état incohérent), $R_2(A)$, $W_2(A), R_2(B), W_2(B)$, $Commit_2, R_1(B), W_1(B), Commit_1$

Diminuer le Degré d'Isolation: Types de conflits

- **Conflit écriture/lecture (WR) (lecture sale)**
  - T_1 transfère une somme de A vers B, T_2 incrémente A et B de 10%
  - $R_1(A), W_1(A)$(état incohérent), $R_2(A)$, $W_2(A), R_2(B), W_2(B)$, $Commit_2, R_1(B), W_1(B), Commit_1$
- **Conflit lecture/écriture (RW) (lectures non reproductibles)**
  - A est le nombre de copie d'un livre, T_1 et T_2 commandent un livre
  - $R_1(A = 1), R_2(A = 1), W_2(A = 0), Commit_2$, (a) $W_1(A = 0)$, $Commit_1$. Si en (a) $R_1(A)$, il ne trouve pas la même valeur.

Diminuer le Degré d'Isolation: Types de conflits

- **Conflit écriture/lecture (WR) (lecture sale)**
  - T_1 transfère une somme de A vers B, T_2 incrémente A et B de 10%
  - $R_1(A), W_1(A)$(état incohérent), $R_2(A)$, $W_2(A), R_2(B), W_2(B)$, $Commit_2, R_1(B), W_1(B), Commit_1$
- **Conflit lecture/écriture (RW) (lectures non reproductibles)**
  - A est le nombre de copie d'un livre, T_1 et T_2 commandent un livre
  - $R_1(A = 1), R_2(A = 1), W_2(A = 0), Commit_2$, (a) $W_1(A = 0)$, $Commit_1$. Si en (a) $R_1(A)$, il ne trouve pas la même valeur.
- **Conflit écriture/écriture (WW) (écriture aveugle)**
  - Contrainte: A et B doivent être égaux
  - $W_1(A = 10), W_2(B = 20)$, $W_1(B = 10)$, $W_2(A = 20), Commit_2, Commit_1$

Diminuer le Degré d'Isolation: Degrés d'isolation de SQL

1. **Degré 0**
   - les lectures peuvent être sales, i.e. lire des données écrites par des transactions non validées
   - les lectures ne sont pas reproductibles
   - les écritures peuvent remplacer des données sales
   - une transaction possède un verrou en écriture juste pendant l'écriture

Diminuer le Degré d'Isolation: Degrés d'isolation de SQL

1. **Degré 0**
   - les lectures peuvent être sales, i.e. lire des données écrites par des transactions non validées
   - les lectures ne sont pas reproductibles
   - les écritures peuvent remplacer des données sales
   - une transaction possède un verrou en écriture juste pendant l'écriture
2. **Degré 1** (*read uncommitted*)
   - les lectures peuvent être sales
   - les lectures ne sont pas reproductibles
   - les écritures ne peuvent pas remplacer des données sales

Diminuer le Degré d'Isolation: Degrés d'isolation de SQL

1. **Degré 2** (*read committed*)
   - les lectures ne peuvent lire que des données valides
   - les lectures ne sont pas reproductibles
   - le verrouillage en écriture est en 2PL
   - les verrous en lecture sont relachés juste après la lecture

Diminuer le Degré d'Isolation: Degrés d'isolation de SQL

1. **Degré 2** (*read committed*)
   - les lectures ne peuvent lire que des données valides
   - les lectures ne sont pas reproductibles
   - le verrouillage en écriture est en 2PL
   - les verrous en lecture sont relachés juste après la lecture
2. **Degré 3** (*serializable*)
   - les lectures ne peuvent lire que des données valides
   - les lectures sont reproductibles
   - les écritures ne peuvent pas remplacer des données sales

   $\Rightarrow$ les transactions s'exécutent en isolation

Diminuer le degré d'isolation: Utilisation

- Certaines transactions ne nécessitent pas forcément une réponse exacte (par exemple statistiques): dans ce cas, degré 2 ou même 1
- Marche à suivre
  1. Commencer au degré d'isolation le plus élevé
  2. Si une transaction souffre de *deadlocks* ou cause des blocages, considérer la baisse du niveau d'isolation (**Attention, le résultat peut devenir incorrect**)

Granularité des Verrous

- Différents niveaux (du plus fin au plus gros)
  - tuple
  - page
  - table
- Le plus adapté est généralement le verrou au niveau du tuple

Granularité des Verrous

- Différents niveaux (du plus fin au plus gros)
  - tuple
  - page
  - table
- Le plus adapté est généralement le verrou au niveau du tuple
- Intérêt du niveau table
  - évite le blocage de transaction longue
  - évite les *deadlocks*
  - réduit le coût dû au verrouillage

Granularité des verrous

- Principe
  - une transaction longue (qui accède à une grosse portion d'une table) utilise un verrou au niveau de la table pour éviter les blocages
  - une transaction courte utilise un verrou au niveau du tuple pour augmenter la concurrence

Granularité des verrous

- Principe
  - une transaction longue (qui accède à une grosse portion d'une table) utilise un verrou au niveau de la table pour éviter les blocages
  - une transaction courte utilise un verrou au niveau du tuple pour augmenter la concurrence
- Paramètres
  - contrôle explicite de la granularité des verrous dans une transaction ou globalement (par table par exemple)
  - définir le point d'escalade: choisir un seuil suffisamment élevé pour éviter l'escalade dans un environnement de transactions courtes (taille de la table des verrous)

Ordres DDL

- Les ordres DDL manipulent les métadonnées
- Le catalogue est accédé par toutes les requêtes mais un ordre DDL le verrouille en écriture
- Le catalogue est un point chaud pour le SGBD
- Il faut donc éviter les mises à jours du catalogue lors des périodes d'intense activité

Partitionnement

- Un problème se pose par exemple si toutes les insertions se font au même endroit (goulot d'étranglement)

Partitionnement

- Un problème se pose par exemple si toutes les insertions se font au même endroit (goulot d'étranglement)
- **Stratégie**: partitionnement des points d'insertion
  1. Fixer plusieurs points d'insertion
  2. Mettre en place un index primaire sur un attribut non corrélé avec le moment de l'insertion

Partitionnement

- Un problème se pose par exemple si toutes les insertions se font au même endroit (goulot d'étranglement)
- **Stratégie**: partitionnement des points d'insertion
  1. Fixer plusieurs points d'insertion
  2. Mettre en place un index primaire sur un attribut non corrélé avec le moment de l'insertion
- Si n est le nombre maximum de transactions concurrentes, alors on définit $\frac{n}{4}$ points d'insertion

Eliminer les Points Chauds

- Un point chaud est une données accédée par plusieurs transactions (mise à jour par certaines)
  - Il forme un goulot d'étranglement car les transactions de mise à jour doivent terminer avant qu'une autre transaction obtienne un verrou

Eliminer les Points Chauds

- Un point chaud est une données accédée par plusieurs transactions (mise à jour par certaines)
  - Il forme un goulot d'étranglement car les transactions de mise à jour doivent terminer avant qu'une autre transaction obtienne un verrou
- **Trois techniques**
  1. Partitionner
  2. Accéder au point chaud le plus tard possible dans la transaction pour minimiser le temps de verrouillage du point chaud
  3. Utiliser des fonctionnalités spécifiques du SGBD (par exemple, les séquences sous Oracle utilisent un contrôle d'accès non basé sur le verrouillage)

Reprise

- Du point de vue des performances, le sous-système de reprise est un surcoût pur et simple
- Deux types de pannes sont prises en compte
  - panne processeurs (échec-arrêt (*fail-stop*)) et perte de la RAM
  - panne disque (échec-arrêt) à condition qu'il y ait suffisamment de redondance

Principe

- L'unité de reprise est la transaction

Principe

- L'unité de reprise est la transaction
- Les effets d'une transaction validée sont permanents (*durabilité*)
  - $\Rightarrow$ les données des transactions validées doivent être placées sur un stockage stable, i.e. disques (panne d'alimentation) puis répliquées pour la redondance (panne disque)

Principe

- L'unité de reprise est la transaction
- Les effets d'une transaction validée sont permanents (*durabilité*)
  - ⇒ les données des transactions validées doivent être placées sur un stockage stable, i.e. disques (panne d'alimentation) puis répliquées pour la redondance (panne disque)
- Une transaction doit être *atomique*
  - ⇒ avant qu'une transaction ne valide, ses effets doivent pouvoir être annulés donc une image avant des données modifiées doit être placée sur stockage stable
  - ⇒ lors de la validation, les modifications doivent pouvoir être installées donc une image après doit aussi être conservée

Principe

⇒ Il faut prévoir un mécanisme supplémentaire: le journal (*log*)

- Algorithme de journalisation: état courant = état disque + log **(i)**
- Points de contrôle (*checkpoints*): périodiquement, le système copie les mises à jour validées sur le disque pour éviter au log de trop grossir. L'intervalle entre points de contrôle est un paramètre de tuning.
- *Database dump*: sauvegarde cohérente de la BD
- état courant = dump + log **(ii)**
- (i) et (ii)
  - ⇒ le log est très important donc à répliquer
  - ⇒ il faut séparer le dump de la BD opérationnelle

Tuning du Système de Reprise

- Placer le log sur un disque dédié
- Retarder l'écriture des mises à jour sur le disque le plus longtemps possible
- Régler les intervalles de checkpoint et de dump
- Réduire la taille des grandes transactions de mise à jours

Placer le log sur un disque dédié

- La gestion du log génère des écritures disque ce qui est mauvais pour les performances
- Les écritures se font séquentiellement et par "gros" blocs ce qui est bon pour les performances
- On place le log sur un disque dédié pour éviter les déplacements des têtes de lecture/écriture (taux d'E/S élevé)
- Différents log doivent se trouver sur des disques différents

Tuning du Système de Reprise

Retarder les écritures disques

- Lors de la validation, les images après sont écrites dans le log donc le log et le cache contienne la même information (pas la BD)
- Forcer les écritures est mauvais pour les performances (déplacements des têtes)
- De plus, certaines écritures ne sont pas utiles (si deux transactions écrivent un même objet successivement par exemple)
- C'est toutefois nécessaire de temps en temps
- Il faut donc choisir le bon moment pour les écritures
- Paramètres
  - Moment pour réaliser les écritures (`DB_BLOCK_MAX_DIRTY_TARGET` sous Oracle)
  - Fréquence des checkpoints

Tuning du Système de Reprise

Régler les intervalles de dump et de checkpoints

- Compromis entre temps de reprise et performance
- Généralement, le dump a lieu une ou deux fois par jour
- Avantages du dump
  - récupération des données en cas de panne disque (ne protège pas contre les pertes du log)
  - peut être utilisé pour le décisionnel
- Inconvénients
  - augmentation du temps de réponse lorsqu'il se produit
  - nécessite de la place
- Propriétés du checkpoint
  - réduit les temps de reprise (perte de RAM) et l'encombrement du log
  - ne réduit rien en cas de panne disque
  - dégrade moins les performances qu'un dump

SECTION Exercices

Exercices

Exercice 1 **Matériel - E/S** *Le système comporte deux disques. La BD est répartie sur les deux disques, le log se trouve sur le deuxième. Le taux d'activité du deuxième disque est très supérieur au taux d'activité du premier. Le client envisage d'acheter un autre disque. Comment l'utiliser au mieux ?*

Exercices

Exercice 2 **Matériel - E/S** *Le taux d'utilisation des disques est élevé. L'application est essentiellement en lecture seule et nécessite de nombreux parcours de tables. Chaque parcours provoque de nombreux déplacements des têtes de lecture. Il n'y a pas de possibilités pour acheter du matériel. Le log est sur un disque à part.*

Exercices

Exercice 3 Matériel - E/S *Le sous-système d'E/S utilise un contrôleur RAID 5. Les fichiers de données et les index se trouvent sur un disque, les fichiers temporaires et le log sur un deuxième. Les taux d'utilisation des deux disques restent très élevés. Que préconisez-vous ?*

Exercices

Exercice 4 **SGBD - Cache** *Le sous-système d'E/S est assez chargé. La mémoire du système est assez peu utilisée. Comment faire diminuer l'activité disque ?*

Exercices

Exercice 5 Matériel - Mémoire

Suite à l'intervention précédente, l'activité du disque contenant le SE a augmentée. Les performances de la BD se sont dégradées. Que s'est-il passé et comment y remédier ?

Exercices

Exercice 6 SGBD - CC *Les temps de réponse varient beaucoup au cours d'une journée. L'utilisation classique de la BD comporte des transactions courtes ainsi que des créations de tables indépendantes du reste de l'activité (par des développeurs par exemple). Quelle est la cause des variations de temps de réponse ?*

Exercices

Exercice 7 SGBD - CC *L'entreprise enregistre ses clients avec la transaction suivante:*

1. *Obtenir un numéro de client à partir d'un compteur global*
2. *Demander un certain nombre d'informations au client (adresse, . . .)*
3. *Insérer le client dans la table* `clients`

Le système ne peut pas supporter le volume de transaction. Que faire et pourquoi ?

Exercices

Exercice 8 SGBD - CC *Le système supporte au cours de la journée les transactions classiques de l'entreprise ainsi que les requêtes d'analyse (data mining) du service marketing. Les utilisateurs (classiques et analystes) sont mécontents des temps de réponse. Que faire ?*

Exercices

Exercice 9 SGBD - CC *Un transaction effectue des mises à jour sur la plupart des tuples d'une table. Elle s'exécute de façon concurrente avec des transactions qui mettent à jour quelques tuples de cette même table. On constate que les blocages et les deadlocks sont nombreux. Proposer un scénario où se produit un deadlock. Comment y remédier ?*