Ingénierie des systèmes d'information

Concevoir un système d'information

Manon Ansart

ESIREM, LEAD

2023

La dernière fois

Propriétés souhaitées pour un système d'information de bonne qualité :

- **Cohérence :** pas des réponses contradictoires
- **Disponibilité :** vitesse de chargement, latence
- **Robustesse :** persistance des données dans le temps même face aux erreurs utilisateurs
- **Confidentialité, sécurité :** accès non autorisé impossible

Plan

Rappel : index

Définition

Un index est un fichier structuré (table de correspondance) contenant pour chaque **clé** l'adresse de l'enregistrement correspondant

Propriétés :

- accélère la recherche (select) et donc les jointures
- insertion et modification potentiellement plus lentes
- il faut trouver un équilibre !

Exemples d'index

Exemple 1 : Index sur les attributs

- index des termes techniques utilisés dans un livre ou d'ingrédients pour un livre de recette
- l'attribut (mot clé, ingrédient) est clé de l'index
- **index dense** : tous les enregistrements sont présents dans l'index

Exemples d'index

Exemple 2 : Index sur des clés

- index des numéros étudiants, ou des mots dans le dictionnaire
- le fichier (base des étudiants, dictionnaire) est **ordonné** selon la clé
- la clé de la table (numéro étudiant, mot) est clé de l'index
- index non-dense : seulement certains enregistrements sont présents dans l'index (un par bloc), et permettent de retrouver les autres plus rapidement dans le fichier ordonné

Types d'index

- Index dense : contient toutes les valeurs de la clé
- Index non-dense : ne contient qu'une clé par bloc

Choix des index

- Il est possible de faire un index sur plusieurs colonnes, mais c'est au dela du cadre de ce cours
- Toujours se demander si ça vaut le coup : recherche vs insertion / modification
- Les clés des tables sont déjà indexées
- Si une clé est composée de plusieurs attributs, ils sont généralement indexés ensemble et non individuellement

Choix des colonnes :

- Where
- Jointures

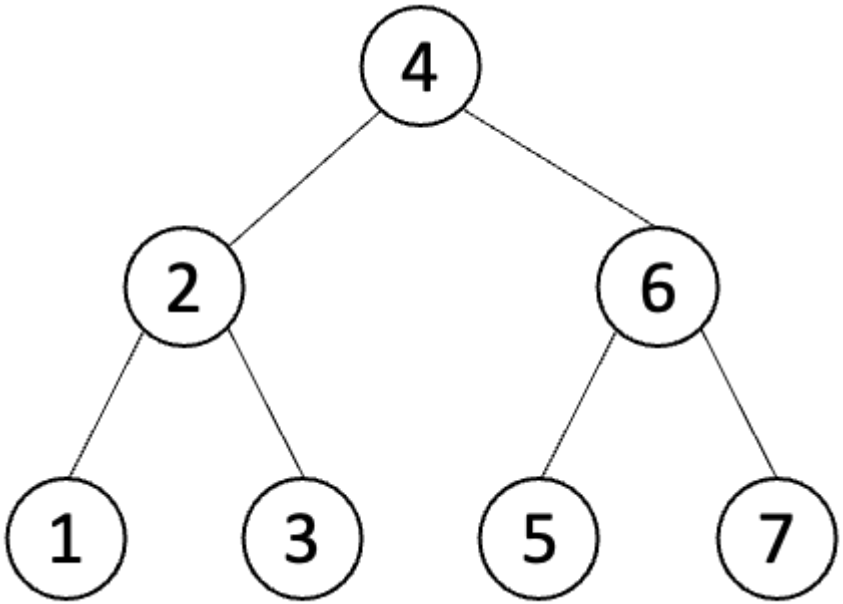
Activité

Comment rechercher dans un index ?

- Comment effectuez-vous une recherche dans un fichier ordonné (dictionnaire) ?
-
- N lignes : nécessite en moyenne
-

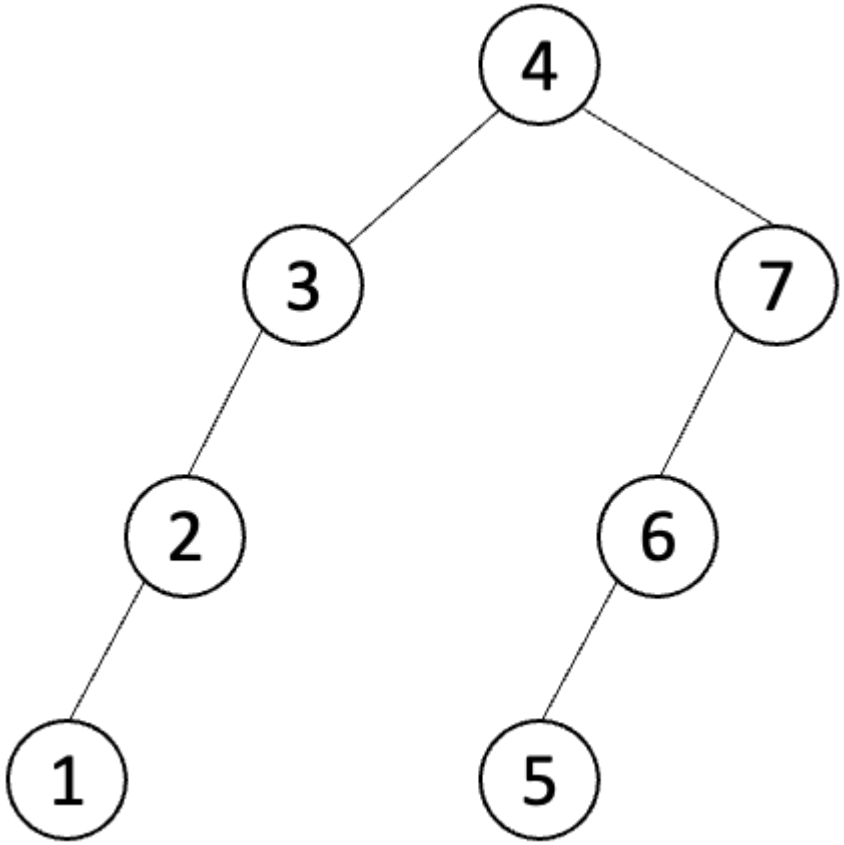
Peut-on faire mieux ? Oui !

Arbre binaire



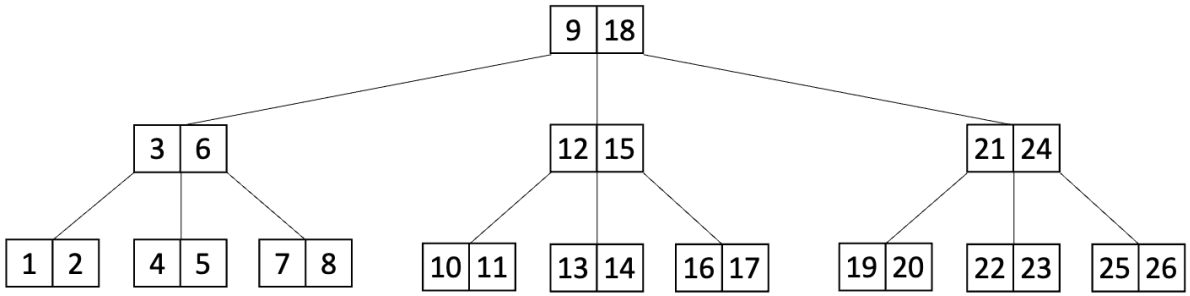
Équivalent à une recherche par dichotomie ($2^3 = 8$, $log_2 7 \approx 3$)

Arbre binaire non équilibré



Recherche moins efficace -> les arbres doivent être équilibrés !

Arbres 2-3



Recherche plus efficace que dans un arbre binaire ($3^3 = 27$, $log_3 26 \approx 3$)

Arbre 2-3

Définition : arbre 2-3 (variante de l'arbre-B)

- Chaque nœud a plusieurs étiquettes (clés) ordonnées notées e_i : au moins 1 étiquettes, au plus 2
- Chaque nœud, sauf la racine et feuilles, possède au moins 2 enfants, au plus 3 (noté T_i).
- Si un nœud ($\neq$ feuille) a k étiquettes
  - Si un nœud a 1 étiquette, il a enfants
  - Si un nœud a 2 étiquettes, il a enfants

Arbre 2-3

Pour un noeud ayant k étiquettes e_i (donc k+1 enfants), chaque étiquette $c_{j,i}$ du sous-arbre enfant T_j respecte les propriétés suivantes :

- Les étiquettes du premier enfant sont inférieures à la première étiquette du nœud ($c_{1,i} < e_1$).
- Les étiquettes du dernier enfant sont supérieures à la dernière étiquette du nœud ($c_{k+1,i} > e_k$).
- Si il y a 3 enfants (donc 2 étiquettes dans le nœud), les etiquettes du 2ème enfant sont comprises entre les 2 étiquettes du nœud. ($e_{j-1} < c_{j,i} < e_j$)

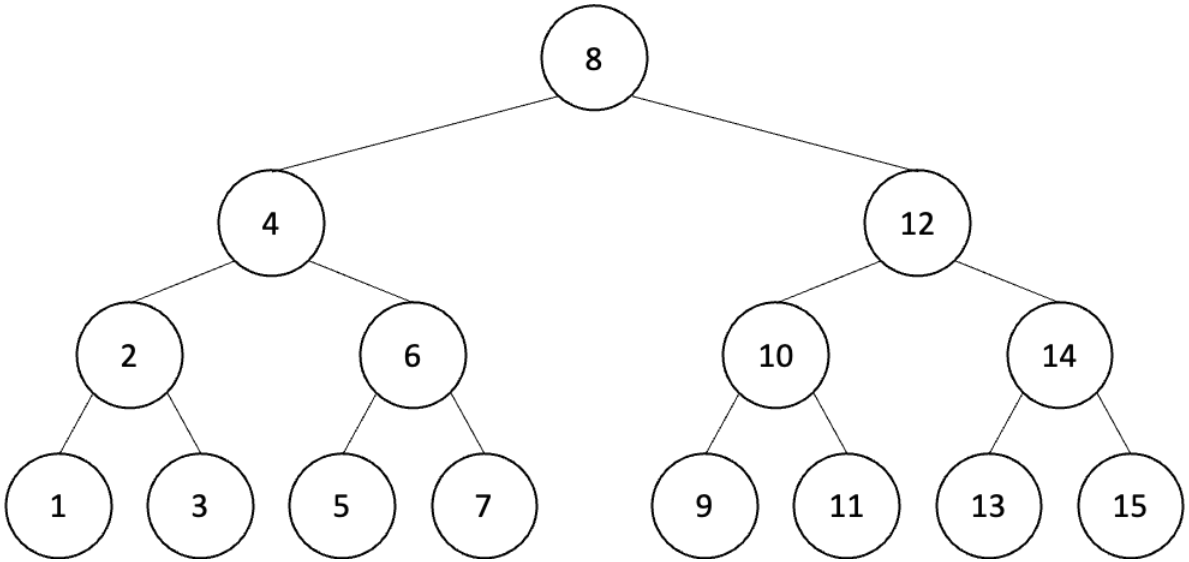
Arbre 2-3

Définition : arbre 2-3

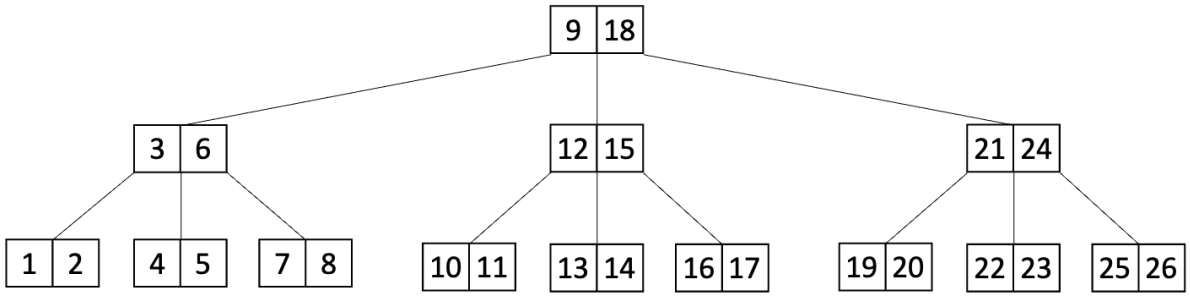
- Un arbre B est toujours équilibré
  - Quand vous contruisez un arbre, vérifiez toujours qu'il est équilibré !
- En plus des étiquettes on peut aussi stocker la charge utile (adresse).
- Exemples au tableau (équilibré ou non, nombre d'étiquettes / parents)

Activité

Le pire des cas



Le meilleur des cas



Activité

Le pire des cas (arbre 2-3)

1. Combien chaque noeud a-t-il d'enfants ? **2**
2. Combien chaque noeud a-t-il d'étiquette ? **1**

Étage	Nombre de noeuds	Nombre d'étiquettes	Nombre d'enfants
0 (racine)	1	1	2
1	2	2	4
2	4	4	8
i	2^i	2^i	2^{i+1}

Le meilleur des cas (arbre 2-3)

1. Combien chaque noeud a-t-il d'enfants ? **3**
2. Combien chaque noeud a-t-il d'étiquette ? **2**

Étage	Nombre de noeuds	Nombre d'étiquettes	Nombre d'enfants
0 (racine)	1	2	3
1	3	6	9
2	9	18	27
i	3^i	2×3^i	3^{i+1}

Complexité : le pire des cas

- Nombre d'éléments pour un arbre à h niveaux : somme des étiquettes pour les h niveaux

$$N = \sum_{i=0}^{h} 2^i = \frac{2^{h+1} - 1}{2 - 1} = 2^{h+1} - 1$$

- Nombre d'étages h pour stocker N éléments :

$$2^{h+1} - 1 \geq N$$

$$2^{h+1} \geq N + 1$$

$$h + 1 \geq log_2(N + 1)$$

$$h \geq log_2(N + 1) - 1$$

Complexité : le meilleur des cas

- Nombre d'éléments pour un arbre à h niveaux : somme des étiquettes pour les h niveaux

$$N = \sum_{i=0}^{h} 2 \times 3^i = 2\frac{3^{h+1} - 1}{3 - 1} = 3^{h+1} - 1$$

- Nombre d'étages h pour stocker N éléments :

$$3^{h+1} - 1 \geq N$$

$$3^{h+1} \geq N + 1$$

$$h + 1 \geq log_3(N + 1)$$

$$h \geq log_3(N + 1) - 1$$

Recherche dans l'arbre B : exemple

Recherche dans l'arbre B

Entrée : k, une clé que nous recherchons dans l'arbre.
Sortie : v, la valeur de la charge utile associée à la clé k s'il existe, NULL sinon.

1. Prendre la racine. La racine est notre *nœud courant*.
2. Si nœud courant est vide, on retourne NULL.
3. Si le nœud courant contient la clé k, alors on termine en retournant la valeur de la charge utile associée.
4. Sinon on trouve un enfant susceptible de contenir la clé k. Cet enfant devient le nœud courant.
5. On revient vers 2.

Insertion dans l'arbre B : exemple

Insertion dans l'arbre B

1. Trouver une feuille où la nouvelle clé k devrait être insérée. Cette feuille est notre *nœud courant* A.
2. Insérer la clé dans A et terminer l'algorithme si A possède un nombre acceptable de clés.
3. Sinon
   1. Trouver une médiane m parmi les clés de A et la nouvelle clé.
   2. Transformer A en deux nœuds séparés par la médiane m.
   3. Répéter l'insertion récursivement, c'est-à-dire
      - $A \leftarrow$ parent de A
      - Répéter les étapes à partir de 2.

Si la clé est insérée dans la racine, qui contient déjà le nombre maximal d'éléments, une nouvelle racine avec un élément sera créée.

Suppression

https://en.wikipedia.org/wiki/B-tree#Deletion

Variantes

Arbre B+

L'arbre B+ diffère légèrement de l'arbre B, en ceci que toutes les données sont stockées exclusivement dans des feuilles, et celles-ci sont reliées entre elles.

Arbre B* garantit que les nœuds (sauf la racine) soient remplis au moins aux 2/3 au lieu de 1/2 (Knuth 1998, The Art of Computer Programming, p. 488).

Bilan des outils vus

Activité

Plan

RGPD

Règlement Général sur la Protection des Données (RGPD)

Règlement européen relatif à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données

- Donnée personnelle : « toute information se rapportant à une personne physique identifiée ou identifiable »
  - Identification directe ou indirecte
  - à partir d'une donnée ou d'un croisement
- Qui est concerné ? Tout organisme quels que soient sa taille, son pays d'implantation et son activité

Recommandations CNIL

1. Ne collectez que les données vraiment nécessaires pour atteindre votre objectif
2. Soyez transparent : informez les individus de l'utilisation qui sera faite de leurs données
3. Organisez et facilitez l'exercice des droits des personnes (de consultation, de rectification ou de suppression)
4. Fixez des durées de conservation
5. Sécurisez les données et identifiez les risques

Chiffrement

Granularité

- Full disk
- Transparent data encryption
- Column-level encryption

Temporalité

- Chiffrement du réseau
- Data at rest : chiffrement des données stockées

Chiffrement

Idéalement

- Chiffrement du réseau et data at rest
- Granularité : équilibre entre sécurité et disponibilité selon les usages
- Hachage des mots de passe

Géré par le moteur de BD

- Attention ! Pas disponible avec SQLite

Droits d'accès

Différents droits :

- interrogation
- mise à jour : insertion, modification suppression
- administration : modification et suppression de tables, contraintes d'intégrités

Niveau :

- utilisateur : les droits sont attribués à chaque personne individuellement
- rôle : création de rôle (CREATE ROLE) puis de droit pour chaque rôle

Droits d'accès

Approches :

- centralisée : seul l'administrateur peut attribuer des droits
- décentralisée : un individu peut transmettre ses droits

Comment ?

- Créé en SQL, géré par le moteur de BD
- Principe de moindre privilège : un utilisateur ne doit bénéficier que du minimum de droits nécessaire

Attention : SQLite ne gère pas les droits d'accès, ça doit être géré côté application

Syntaxe SQL

```
GRANT SELECT, INSERT, UPDATE
ON EMPLOYES_UB
TO Martin
WITH GRANT OPTION
```

```
REVOKE DELETE
ON EMPLOYES_UB
FROM dev
```

Déploiement

- Un serveur surchargé est plus lent
  - Taille de la BD
  - Taille du serveur

- Scalabilité verticale Augmenter les performances du serveur (ajout de processeurs, RAM, disques...)
- Scalabilité horizontale Ajout de serveurs

En relationnel :

- Scalabilité verticale facile à mettre en place
- Scalabilité horizontale moins commune et plus limitée

Opérations de maintenance

Opérations de maintenance :

- Réindexation : indexation n'est pas faite au fur et à mesure ou de manière imprécise
- Défragmentation : réorganise le contenu du disque pour augmenter la vitesse de lecture
- Ralentissement pendant que ces tâches sont faites en arrière plan

Réplication de données

- Sauvegarde : enregistrer régulièrement l'état de la base de données pour pouvoir la récupérer plus tard en cas de problème
  - Erreurs de développement, attaque
- Archivage : garder des données inactives dans un objectif administratif ou légal
- Réplication des données pour la redondance : RAID

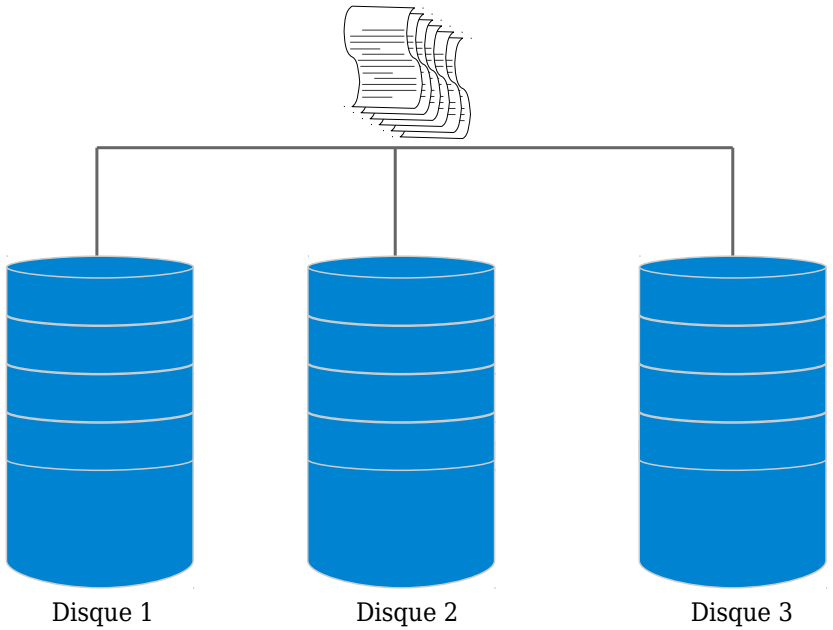
RAID

Redundant Array of Independant Disks (RAID)

Mise en grappe de plusieurs disques

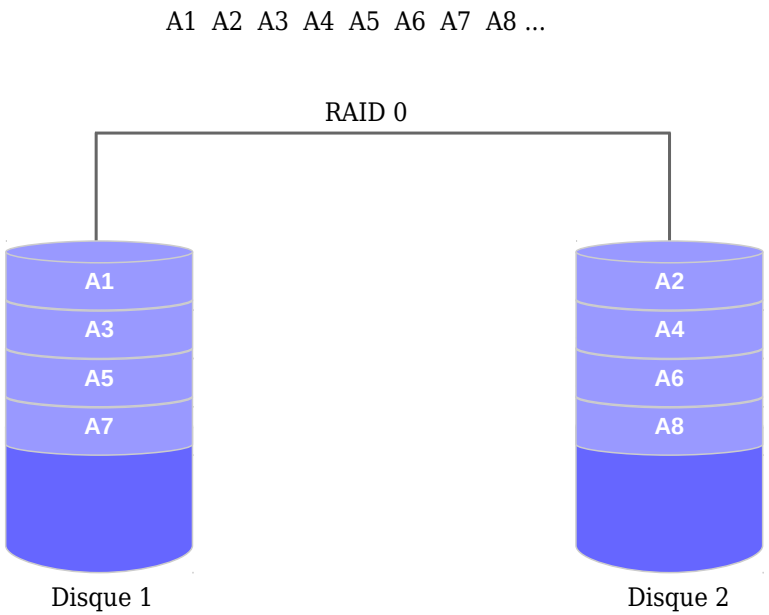
- Transparent pour l'utilisateur : est vu comme une seule partition logique

RAID



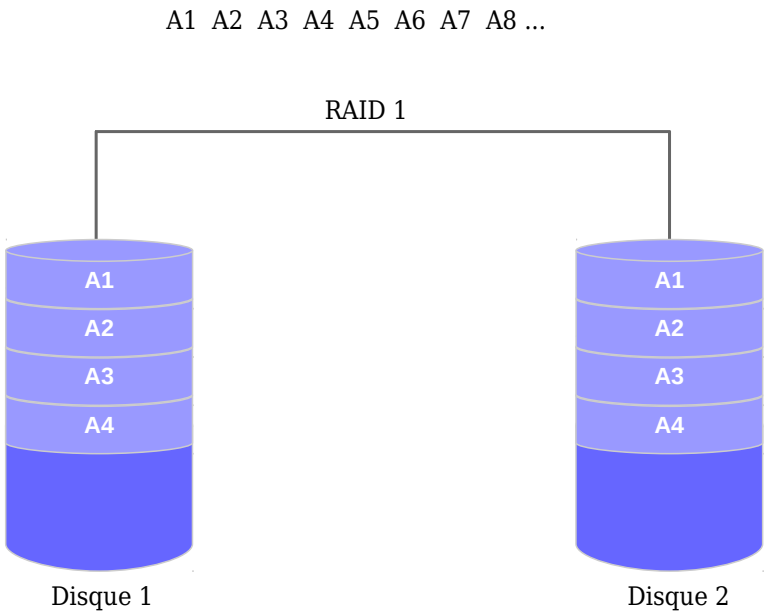
- Vitesse : écriture en parallèle sur les disques
- Redondance : possibilité de récupérer une donnée ou de la reconstruire
- Tolérance aux pannes : la perte d'un disque n'entraîne pas forcément la perte des données.

RAID 0



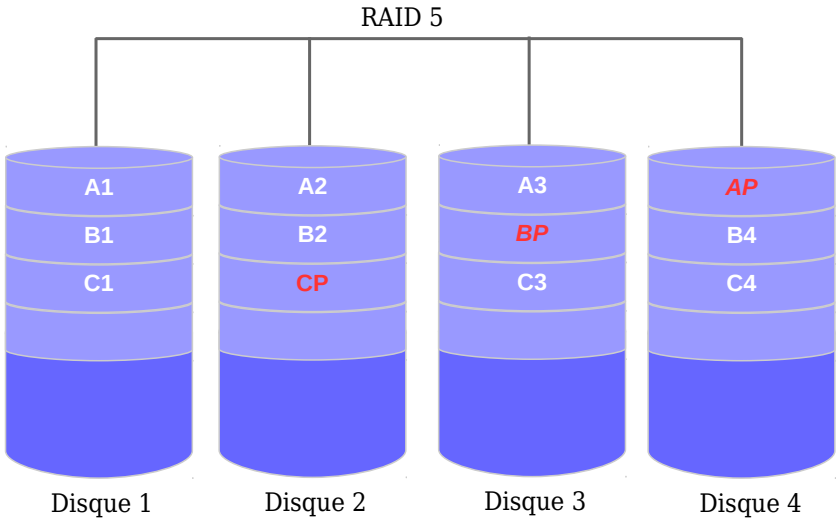
- Avantages : Augmente stockage et vitesse d'écriture
- Inconvénient : Pas de tolérance aux pannes. Un disque mort = toutes les données sont perdues !

RAID 1



- Avantage : Tolérance aux pannes
- Inconvénients : N'augmente pas la capacité de stockage ni la vitesse d'écriture

RAID 5



Utilisation de blocs de parité

- Un bloc de parité par bande
- Tolérance aux pannes comme RAID1, stockage plus important
- Inconvénients : Ralentit à l'écriture

RAID 01 (0+1)

RAID 10 (1 + 0)

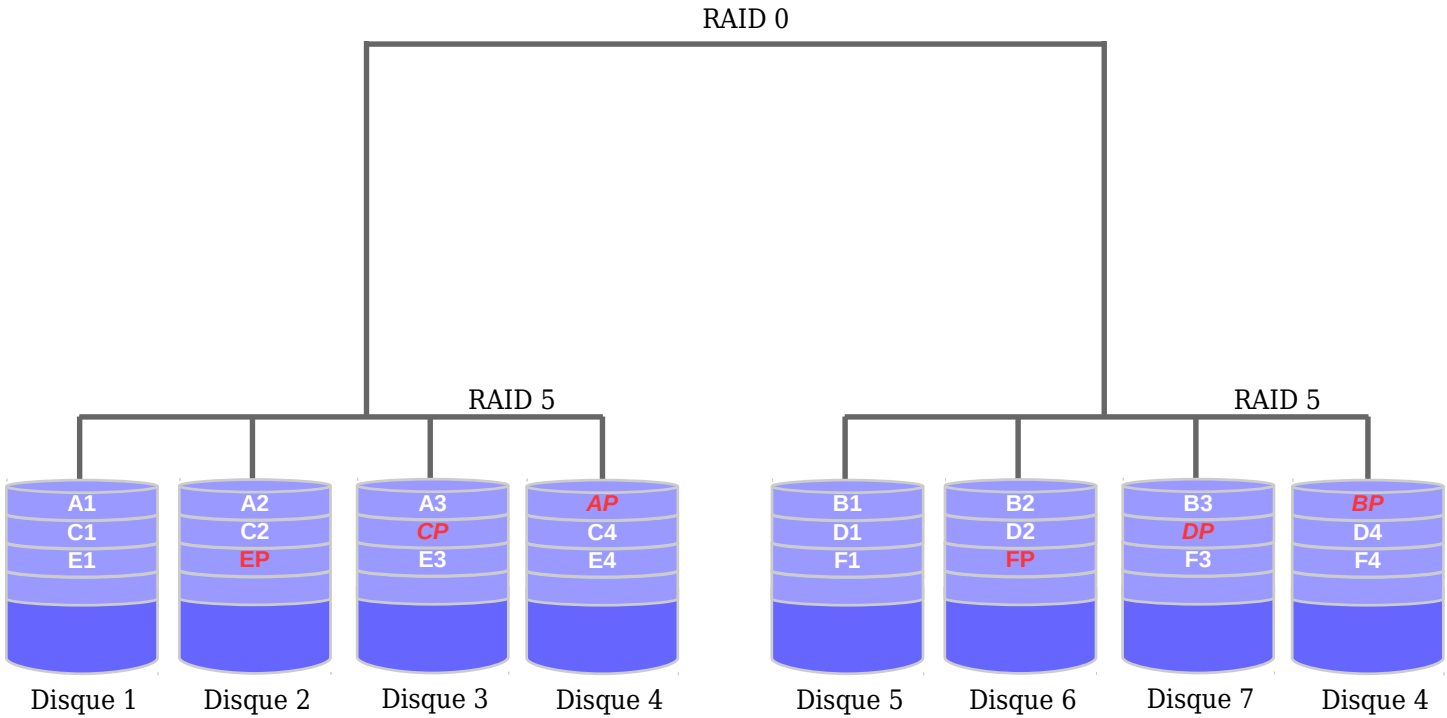
Activité

RAID 01 vs RAID 10

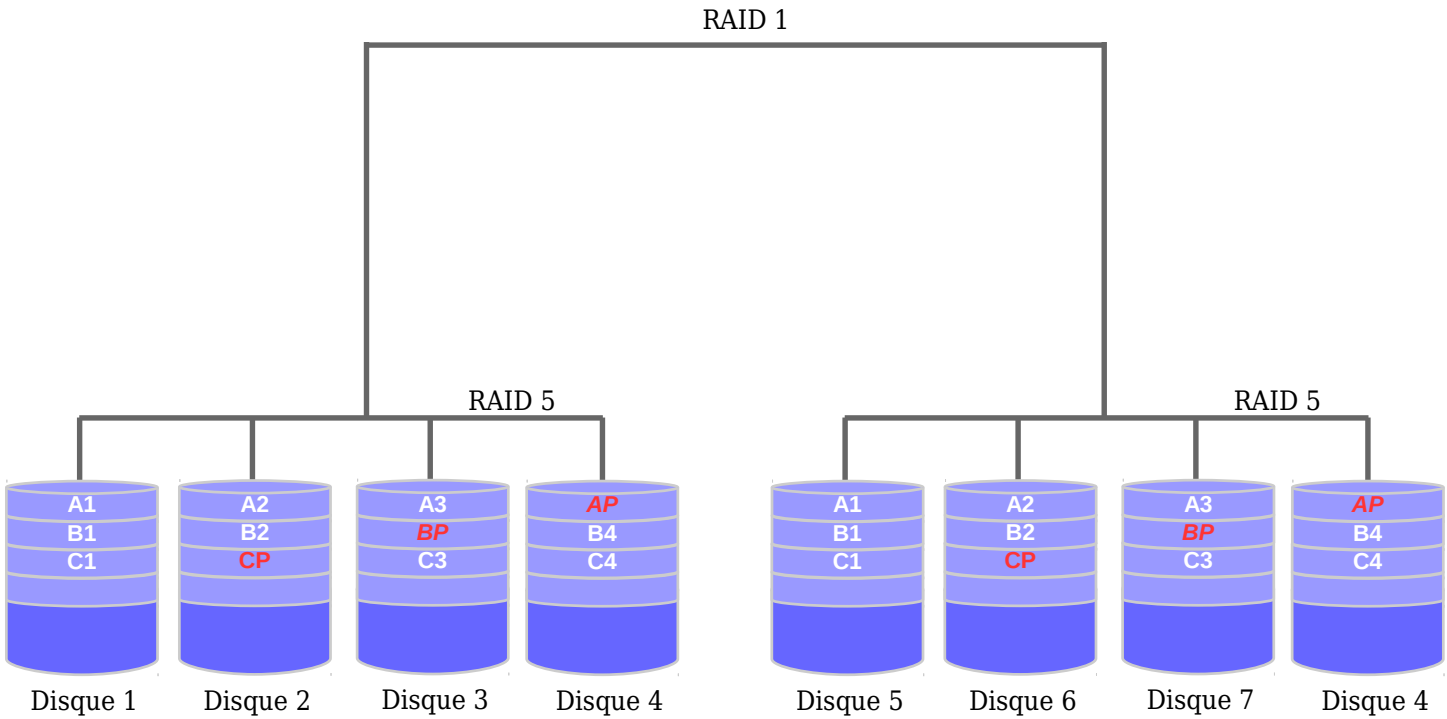
Activité

Encore une activité

RAID 50 (5 + 0)


RAID 0
RAID 5
RAID 5
A1
C1
E1
A2
C2
EP
A3
CP
E3
AP
C4
E4
B1
D1
F1
B2
D2
FP
B3
DP
F3
BP
D4
F4
Disque 1
Disque 2
Disque 3
Disque 4
Disque 5
Disque 6
Disque 7
Disque 4
Utilisation de blocs de parité
Utilisation de blocs de parité

RAID 51 (5 + 1)


RAID 1
RAID 5
RAID 5
A1
B1
C1
A2
B2
CP
A3
BP
C3
AP
B4
C4
A1
B1
C1
A2
B2
CP
A3
BP
C3
AP
B4
C4
Disque 1
Disque 2
Disque 3
Disque 4
Disque 5
Disque 6
Disque 7
Disque 4
Utilisation de blocs de parité
Utilisation de blocs de parité

Plan

Bilan des outils vus

Activité

Assurer la cohérence

Il est possible d'intervenir à différents niveaux :

- À la conception : formes normales
- Lors de la création de tables : **contraintes d'intégrités**
- Autre, coté BD : **trigger**
- Autres, coté appli : dans le code

Assurer la disponibilité

- Organisation physique des données et opérations de maintenance
  - Index, arbres B
- Déploiement (taille du serveur, scalabilité), opérations de maintenance
- Conception et implémentation : verrous, trigger, vues, optimisation des requêtes (jointures inutiles), cache et RAM

Assurer la robustesse

- Robustesse aux erreurs utilisateurs
  - trigger
  - contraintes d'intégrités
  - formes normales
- Sauvegardes et réplication des données / redondance : RAID

Assurer la confidentialité et la sécurité

- Vues : ne donner que l'information nécessaire à chaque utilisateur
- Droits d'accès
- Chiffrement et hachage

Sources

- Sergey Kirgizov - Ingénierie des systèmes d'information (cours ESIREM 2021)
- `http://sys.bdpedia.fr/arbreb.html`