Bases de données réparties : fragmentation et allocation

Notes de cours - C. L. Roncancio - C. Labbé

Références : T. Ozsu, P. Valduriez, Principles of Distributed DB Systems, (Prentice Hall) et notes de S. Abiteboul.

Déroulement de l'année

- ✓ Cyril.Labbe@imag.fr
- ✓ www-lsr.imag.fr/Les.Personnes/Cyril.Labbe/M2P
- ✓ Planning :
  - 28/09 : Cours, Fragmentation-Transactions (C.Labbé)
  - 5/10 : TD, Fragmentation (F.Jouanot)
  - 12/10 : Cours-TD, Transactions (C.Labbé-J.Jouanot)
  - 19/10,26/10,2/11,9/11,16/11: Cours-TP, Admin Oracle (G.Forestier)
  - 23/11,30/11: Cours-TD, Relationel-objet/semi-structuré (C.Labbé-J.Jouanot)

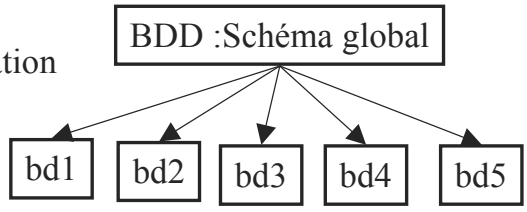
Pourquoi une Base de Données Distribuée

- ✓ Limiter le transfert d'information (nombre et volume)
- ✓ Répartition de charge
- ✓ Augmenter la fiabilité (duplication)
- ✓ Fusionner des systèmes d'informations

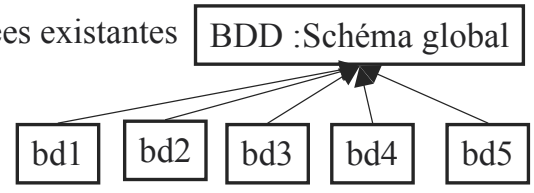
- ✓ Def : Une base de données distribuée est une base de données dont les différentes parties sont stockées sur des sites (géographiquement distants), reliés par un réseau. La réunion de ces parties forme la base de données distribuée.

Approches de conception d'une Base de Données Distribuée

- ✓ Approche descendante (décomposition)
  - Conception du schéma conceptuel global
  - Distribution pour obtenir des schémas conceptuels locaux
  - Fragmentation
  - Affectation aux sites - Allocation



- ✓ Approche ascendante
  - Intégration de bases de données existantes
  - Hétérogénéité

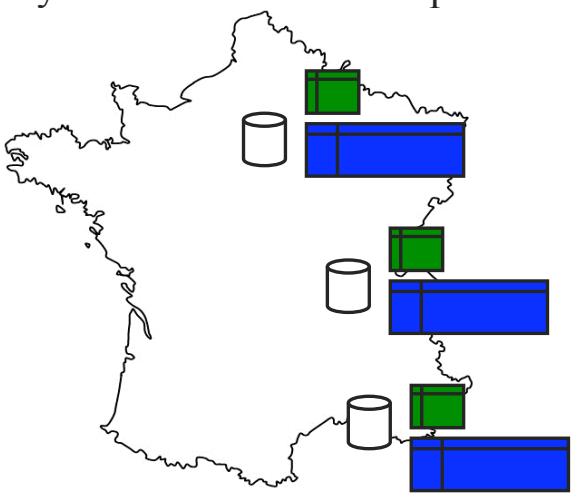


Exemple

- Relation Employé (nss, nom, loc, ...) 
- Relation Taux (pays, valeur, ...) 
- 12 bureaux d'environ la même taille
  - 6 à Paris, 4 à Marseille et 2 à Lyon
- 80% des requêtes dans une ville portent sur les employés de la ville
- 10% des requêtes dans une ville portent sur Taux

Exemple - suite

- Créer 3 bases de données : P, M, L
- Sur chaque base : les employés de la ville et une copie de Taux.
- Si trop cher :
  - fusionner M et L
  - ou maintenir Taux à M.



Conception d'une BDD

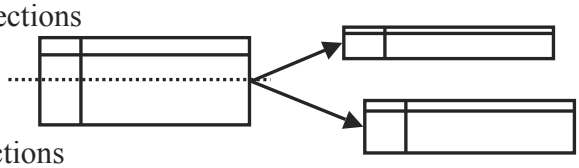
- De la conception en centralisé :
  - schéma conceptuel global
    - Attributs, Domaines, Tables, Vue,…
  - schéma physique
    - Stockage, index, pages,…
- Nouveau en distribué :
  - Définition des fragments
    - Unité de distribution logique
  - Conception « physique » : placement des fragments, stockage, chemins d'accès.

Pour quoi fragmenter ?

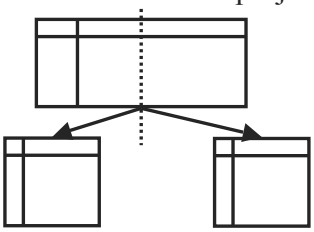
- Pas de vraie raison d'un point de vue « distribution de données ».
- Avantages de la distribution : performances, disponibilité, tolérance aux pannes, localité...
- Grain : une relation entière est une unité de distribution trop grande
  - cas des vues, faible concurrence,

Comment fragmenter ?

- Grain / degré de fragmentation
  - Trop peu de fragments - faible concurrence
  - Trop de fragments - surcoût dans la reconstruction des relations
- Possibilités de fragmentation d'une relation
  - Horizontale - basée sur des sélections

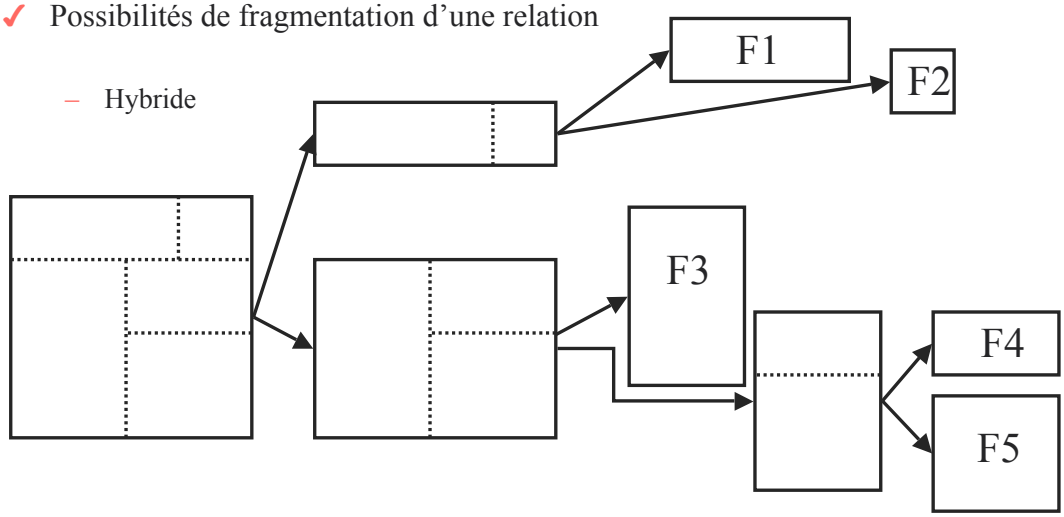


  - Verticale - basée sur des projections



Comment fragmenter (2) ?

- Possibilités de fragmentation d'une relation
  - Hybride



Et le placement des fragments ?

- ✓ Chaque fragment sur un seul site
  - copie unique, BD partitionnée
- ✓ Duplication de fragments
  - (+) performances des requêtes et disponibilité
  - (-) coût des mises à jour et contrôle de concurrence plus complexe
- ✓ Fréquemment : duplication partielle
- ✓ Applications pour la duplication totale

Objectifs généraux

- ✓ Fragmentation
  - Favoriser les accès locaux
  - Équilibrer la charge de travail entre les sites
- ✓ Duplication
  - Favoriser les accès locaux
  - Augmenter la disponibilité des données

Informations utile...

- ✓ BD : taille des relations, CI, etc.
- ✓ Applications : FAQ et où
- ✓ Sites : capacités de stockage / traitement.
- ✓ Réseaux et système
- ✓ Les deux dernières servent essentiellement dans l'allocation

Fragmentation Horizontale

- ✓ Fragments définis par sélection

Ex : Clients(NClient, Nom, Ville)

Client1 = $\sigma_{\text{Ville = Paris}}$ (Client)

Client2 = $\sigma_{\text{Ville != Paris}}$ (Client)

- ✓ Reconstruction par union des fragments

Ex : Client = Client1 U Client2

Fragmentation Horizontale – Exemple

Relation Client

NoClient	Nom	Ville
C1	Dupont	Paris
C2	Martin	Grenoble
C3	Martin	Paris
C4	Talon	Lille

Fragmentation Horizontale - Suite exemple

NoClient	Nom	Ville
C1	Dupont	Paris
C3	Martin	Paris

Client1

NoClient	Nom	Ville
C4	Talon	Lille
C2	Martin	Grenoble

Client2

Fragmentation Horizontale Dérivée

- ✓ Fragments définis par (semi) jointure
- ✓ Ex : Commande(NC, NClient, Produit,Qté)

Commande1 = Commande α Client1

Commande2 = Commande α Client2

- ✓ Reconstruction par union des fragments

Ex : Commande = Commande1 U Commande2

Fragmentation Horizontale Dérivée - Exemple

Relation Commande

NC	NClient	Produit	Qte
Co1	C1	P1	10
Co2	C1	P2	200
Co3	C2	P3	30
Co4	C4	P4	5

Fragmentation Horizontale Dérivée – Suite exemple

Relation Commande1

NC	NClient	Produit	Qte
Co1	C1	P1	10
Co2	C1	P2	200

Relation Commande2

NC	NClient	Produit	Qte
Co3	C2	P3	30
Co4	C4	P4	5

Fragmentation Verticale

- ✓ Fragments définis par projection
- ✓ Ex : Commande(NC, NClient, Produit,Qté)

CommandeA = $\pi_{\text{NC, NClient}}$ (Commande)

CommandeB = $\pi_{\text{NC, Produit, Qté}}$ (Commande)

- ✓ Reconstruction par jointure

Ex : Commande = CommandeA * CommandeB

- ✓ Utile si forte affinité des attributs

Fragmentation Verticale - Exemple

Relation Commande

NC	NClient	Produit	Qte
Co1	C1	P1	10
Co2	C1	P2	200
Co3	C2	P3	30
Co4	C4	P4	5

Fragmentation Verticale – Suite Exemple

Relation CommandeA

NC	NClient
Co1	C1
Co2	C1
Co3	C2
Co4	C4

Relation CommandeB

NC	Produit	Qte
Co1	P1	10
Co2	P2	200
Co3	P3	30
Co4	P4	5

Propriétés de la fragmentation

- ✓ Reconstruction
  - Possible avec les opérateurs de l'algèbre relationnelle
  - Critères de fragmentation horizontale simples ou complexes
- ✓ Complétude
  - Horizontale : chaque n-uplet de R est dans un fragment
  - Verticale : chaque attribut est dans un fragment
  - Pas de perte d'information
  - Similaire à la normalisation

Exemple perte d'information (1)

✓ RA

a	b	c
1	2	3
4	5	6

✓ RA1

a	b
1	2
4	5

RA2

b	c
2	3
5	6

Exemple perte d'information (2)

✓ RA

a	b	c
1	2	3
4	5	6

RA1 * RA2 =

a	b	c
1	2	3
4	5	6

✓ RA1

a	b
1	2
4	5

RA2

b	c
2	3
5	6

Exemple perte d'information (3)

✓ RA

a	b	c
1	2	3
4	5	6
7	2	8

✓ RA1

a	b
1	2
4	5
7	2

RA2

b	c
2	3
5	6
2	8

Exemple perte d'information (4)

✓ RA

a	b	c
1	2	3
4	5	6
7	2	8

✓ RA1

a	b
1	2
4	5
7	2

RA2

b	c
2	3
5	6
2	8

✓ RA1 * RA2

a	b	c
1	2	3
1	2	8
4	5	6
7	2	3
7	2	8

Propriétés de la fragmentation (2)

- ✓ Fragments disjoints
  - Une « entité » est dans un seul fragment
- ✓ Exemple FAQ : $\sigma_{sal < 30}(R)$ et $\sigma_{sal > 20}(R)$
- ✓ Fragments candidats : $\sigma_{sal < 30}(R)$ et $\sigma_{sal > 20}(R)$
  - Non disjoint
- ✓ Alternative / disjoint :

$\sigma_{sal <= 20}(R)$ et $\sigma_{20 < sal < 30}(R)$ et $\sigma_{sal >= 30}(R)$

Disjoint vs. Non Disjoint

- ✓ Disjoint

Bonne propriété, facilite les mises à jour

- ✓ Non Disjoint

Forme de duplication, accélère certaines requêtes

Comment ???

Comment obtenir des fragments disjoints et assurer la reconstruction ?

- ✓ Génération automatique
- ✓ Souvent « manuelle » par l 'administrateur.

- ✓ Différentes méthodologies selon le type de fragmentation.

Analyse fragmentation horizontale

- Commencer avec les conditions de sélection fréquentes (FAQ)
- Règle de Wiederhold : 80 / 20
- Extraire des requêtes les conditions de sélections :
  - A < 10, A > 5, Ville = Paris, Ville = Lyon
- On obtient un ensemble : C={c1, c2,...cn} des conditions élémentaires (ce).

Fragmentation horizon

- Construire l'ensemble des conjonctions de ce (cc) suivant :

$$CC = \left\{ \bigwedge_{i=1,n} C_i^* \quad ou\ C_i^*\ est\ soit\ c_i\ soit\ \neg c_i \right\}$$

$$|CC| = ??$$

- Simplifier chaque cc
- Supprimer les cc tjr fausses

Exemple

A < 10, A > 5, Ville = Paris, Ville = Lyon
A < 10, A > 5, Ville = Paris, Ville != Lyon
A < 10, A > 5, Ville != Paris, Ville = Lyon
A < 10, A > 5, Ville != Paris, Ville != Lyon
A < 10, A <= 5, Ville = Paris, Ville = Lyon
A < 10, A <= 5, Ville = Paris, Ville != Lyon
A < 10, A <= 5, Ville != Paris, Ville = Lyon
A < 10, A <= 5, Ville != Paris, Ville != Lyon
A >= 10, A > 5, Ville = Paris, Ville = Lyon
A >=10, A > 5, Ville = Paris, Ville != Lyon
A >= 10, A > 5, Ville != Paris, Ville = Lyon
A >= 10, A > 5, Ville != Paris, Ville !=Lyon
A >= 10, A <= 5, Ville = Paris, Ville =Lyon
A >=10, A <= 5, Ville = Paris, Ville !=Lyon
A >= 10, A <= 5, Ville != Paris, Ville = Lyon
A >= 10, A <= 5, Ville != Paris, Ville != Lyon

Éliminer les inutiles ...

- Conditions non satisfiables
- Connaissance des contraintes d'intégrité, e.g; ville soit Paris, soit Lyon.

A < 10, A > 5, Ville = Paris, Ville != Lyon
A < 10, A > 5, Ville != Paris, Ville = Lyon
A < 10, A <= 5, Ville = Paris, Ville != Lyon
A < 10, A <= 5, Ville != Paris, Ville = Lyon
A >=10, A > 5, Ville = Paris, Ville != Lyon
A >= 10, A > 5, Ville != Paris, Ville = Lyon

Fragments finaux

- Regrouper les conditions sur un même attribut

  5 < A < 10, Ville = Paris

  5 < A < 10, Ville = Lyon

  A <= 5, Ville = Paris

  A <= 5, Ville = Lyon

  A >=10, Ville = Paris

  A >= 10, Ville = Lyon

Propriétés de la fragmentation ?

- Complétude :

  tout n-uplet t vérifie une des ce et nous avons éliminé seulement les cc impossibles

- Disjonction :

  Supposons t valide deux cc (cc1 et cc2) alors :

$$\exists\, ci\ tq\ ci \in cc1\ et\ \neg ci \in cc2$$

  et donc t satisfit ci et ⎤ci , contradiction !!

Bonne propriétés

- Obtenir une partition de l'ensemble de n-uplets telle que : pour tout B de la partition, deux n-uplets de B aient la même probabilité d'être accédés par toute application/requête importante

Bonne propriétés - exemple

Emp(Enum, Nom, Ville,Sal, …)

- FAQ :
  - select * from Emp where Ville = Paris
  - select * from Emp where Ville = Lyon
- Ensemble vide de conditions : partition P1 = {B1}
- Mauvais : certains n-uplets de B1 ont 0 chances de répondre à la première requête, d'autres non

Ne pas oublier les règles – suite exemple

La partition ne doit pas être un raffinement d'une autre qui est déjà bonne

- ✓ Partition P2 avec conditions :
  Ville = Paris et Ville = Lyon
- ✓ Partition P3 avec conditions :
  (Ville = Paris et Sal < 10000) … et (Ville = Lyon et Sal >= 10000)
- ✓ P3 n'est pas minimale : il vaut mieux commencer par une minimale et distribuer plus si nécessaire.

Analyse Fragmentation Horizontale Dérivée

- ✓ R est fragmentée en R1,…Rk
- ✓ S est fragmentée en S α R1,…,S α Rk
- ✓ Conditions pour que ça marche :

Reconstruction S = U (S α Ri)

Disjonction (i != j) (S α Ri)∩(S α Rj) = ∅

Conception : association 1-n entre entités R et S

Intégrité : S a une clé étrangère de R

… Condition suffisante pour reconstruction et disjonction

Analyse Fragmentation Verticale

Emp(Enum, Nom, Ville,Sal)
Emp1(Enum, Nom, Ville) Emp2(Enum,Sal)

- ✓ Complétude et reconstruction

Jointure sans perte d'information

Condition suffisante : répéter la clé dans chaque fragment

Autre choix : utiliser une dépendance fonctionnelle pour la décomposition (cf. normalisation)

- ✓ Disjonction : seulement répéter la clé

Comment fragmenter verticalement ?

- ✓ Utiliser le sens commun...essayer d'automatiser
- ✓ Heuristiques :
- ➡ *Clustering* - rapprochement en partant d'un attribut par fragment
- ➡ *Splitting* - décomposition en partant d'un fragment unique pour obtenir une partition selon l'affinité des attributs

Outil - Matrice d'affinité (1)

Exemple Projet(Pnum, Pnom, Budget, Loc)

	A1	A2	A3	A4

q1: budget d'un projet étant donné son numéro

q2: nom et budget de tous les projets

q3 : nom des projets d'une ville

q4 : budget total des projets d'une ville

Matrice d 'utilisation

- $$Ut(q_i, A_j) = \begin{cases} 1 \; si \; q_i \; utilise \; A_j \\ 0 \; sinon \end{cases}$$

- Example

	A1	A2	A3	A4
q1	1	0	1	0
q2	0	1	1	0
q3	0	1	0	1
q4	0	0	1	1

Matrice d'affinité - Définition

- $Ref_s(q_k)$, pour deux attributs (A_i, A_k) = nombre d'accès faits par une exécution de q_k (sur le site s) à A_i et A_k
- $Acc_s(q_k)$ = fréquence de q_k sur le site s (mesurée pendant une certaine période)
- Matrice d'affinité

$$Aff(A_i, A_j) = \sum_{\substack{k\ tq \\ Ut(q_k, A_i)=1 \\ \wedge Ut(q_k, A_j)=1}} \sum_{s} Ref_s(q_k) Acc_s(q_k)$$

Matrice d'affinité - *Exemple*

- Supposons : $Ref_s(q_k) = 1\ \forall s, k$
- Les accès suivants (3 sites):

$Acc_1(q_1)=15$ $Acc_2(q_1)=20$ $Acc_3(q_1)=10$

$Acc_1(q_2)=5$ $Acc_2(q_2)=0$ $Acc_3(q_2)=0$

$Acc_1(q_3)=25$ $Acc_2(q_3)=25$ $Acc_3(q_3)=25$

$Acc_1(q4)=3$ $Acc_2(q_4)=0$ $Acc_3(q_4)=0$

Matrice d'affinité - *Exemple* (2)

- $Aff(\text{A1}, \text{A3}) = \Sigma_{K=1} \Sigma_s \, Acc_s(q_k)$
  $= Acc_1(q_1) + Acc_2(q_1) + Acc_3(q_1) = 45$
- La matrice :

	A1	A2	A3	A4
A1	45	0	45	0
A2	0	80	5	75
A3	45	5	53	3
A4	0	75	3	78

Regroupement des attributs

- Regroupement des attributs ayant une haute affinité
- Exemple :

	A1	A3	A2	A4
A1	45	45	0	0
A3	45	53	5	3
A2	0	5	80	75
A4	0	3	75	78

Partitionnement

- Trouver un point dans la matrice pour créer deux ensembles d'attributs : AttrHaut (AH) et AttrBas (AB)
- Quelle requête accède quel ensemble ?
  - $AR(q_i) = \{A_i / Ut(q_i,A_i) = 1 \}$
  - $RH = \{ q_i / AR(q_i) \subseteq AH \}$
  - $RB = \{ q_i / AR(q_i) \subseteq AB \}$
  - $Autres = Q - \{ RH \cup RB \}$

Partitionnement - *Exemple*

- Q = { q1, q2, q3, q4 }
- AH = { A1, A3 }
- AB = { A2, A4}
- RH = { q1 }
- RB = { q3 }
- Autres = { q2, q4 }

Partitionnement (suite)

- Objectif :
  - Maximiser les accès à un seul fragment
  - Minimiser les accès aux deux fragments

- TotalAccès = $\sum_{qi \in Q} \sum_{\forall site\ s} Ref_s(q_i) Acc_s(q_i)$
  $= 128$
- TotalRH = $\sum_{qi \in RH} \sum_{\forall site\ s} Ref_s(q_i) Acc_s(q_i)$
  $= 45$

Partitionnement (suite 2)

- TotalRB = $\sum_{qi \in RB} \sum_{\forall site\ s} Refs(qi) Accs(qi)$
  $= 75$
- TotalAutres
  $= \sum_{qi \in Autres} \sum_{\forall site\ s} Refs(qi) Accs(qi)$
  $= 8$

Partitionnement (suite 3)

- Trouver un point dans la matrice pour créer deux ensembles d'attributs AH et AB tels que :

  $z = (TotalRH * TotalRB) - TotalAutres^2$

  est maximisé

Fin exemple !

- La relation Projet est fragmentée en

  Projet1(Num, Budget)

  Projet2(Num, Nom, Loc)

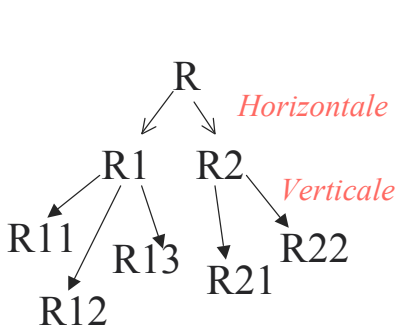
- Dans l'exemple (simple) la clé est dans l'ensemble d'attributs considéré
- Duplication de la clé pour garantir la reconstruction

Partitionnement (fin)

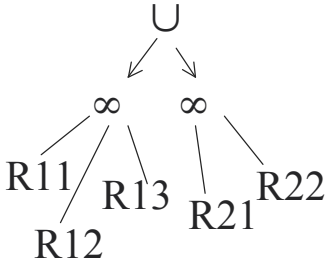
- ✓ Si beaucoup d'attributs alors il est peut être nécessaire de trouver plusieurs points dans la matrice.
- ✓ Utiliser plutôt une approche récursive : partir de deux groups AH et AB et raffiner.

Fragmentation Hybride

- ✓ Fragmentation horizontale suivit de verticale ou vice-versa



Reconstruction



Allocation

- ✓ Problème d'optimisation très difficile
- ✓ Minimiser coûts: stockage, traitement, communication
- ✓ Maximiser performances: temps de réponse, débit du système
- ✓ Modèle coût / performances
- ✓ Utiliser modèle simple / sens commun

  e.g: considérer les com. / Accès

Exemple Client + Commandes

- ✓ Client1 = $\sigma_{\text{Ville} = \text{Paris}}$ (Client)

  Client2 = $\sigma_{\text{Ville} != \text{Paris}}$ (Client)

  Commande1 = Commande α Client1

  Commande2 = Commande α Client2
- ✓ Allocation

  @Site1 : Client1, Commande1

  @Site2 : Client2, Commande2

Duplication

- ✓ Redondance
- ✓ Compromis requêtes (plus vite) / mises à jour (plus lent)
  Quelques fois ralentissement des requêtes…
- ✓ Bons candidats : peu de modifications, beaucoup de consultations e.g. « style annuaire »
- ✓ Approche itérative : pour chaque fragment potentiellement duplicable, Bénéfice ? Coût ?
  Ok si (Bénef – Coût) > 0 ou Maximale

Conclusion

- ✓ Fragmentation des relations
  - Décomposition / reconstruction
  - Horizontale simple / dérivée, verticale, hybride
  - Propriétés : reconstruction, complétude, disjonction
- ✓ Allocation / duplication
  - Modèle de coût
  - Approche itérative