Classe Master 1 en Systèmes d'information et aide à la décision

Entrepôts de données

D. Boukraâ – 2020/2021. Université de Jijel

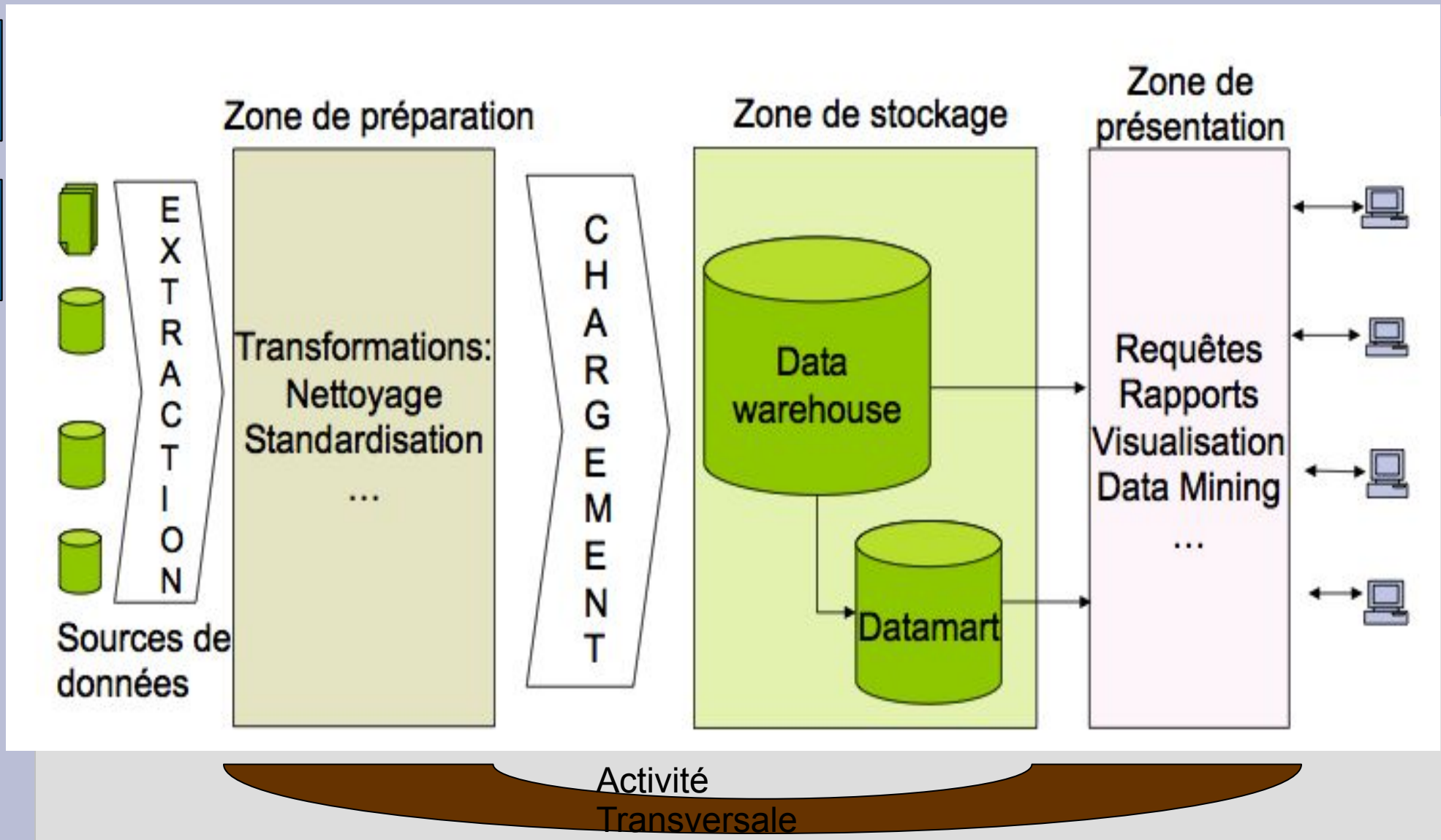
Classe Master 1 en Systèmes d'information et aide à la décision

Chapitre 5

Administration d'un entrepôt de données

Intégration de données

- Place de l'administration d'un ED dans le processus d'entreposage



Administration d'un entrepôt de données

- **Introduction**
  - L’administration d'un ED concerne la gestion de différents aspects
    - Vérification de consistance des données
    - Archivage des données
    - Développement et maintenance des fonctionnalités d'indexation
    - Gestion de l'évolution des schémas des sources et de l’entrepôt
    - Assurance de la qualité des données
    - Optimisation des performances

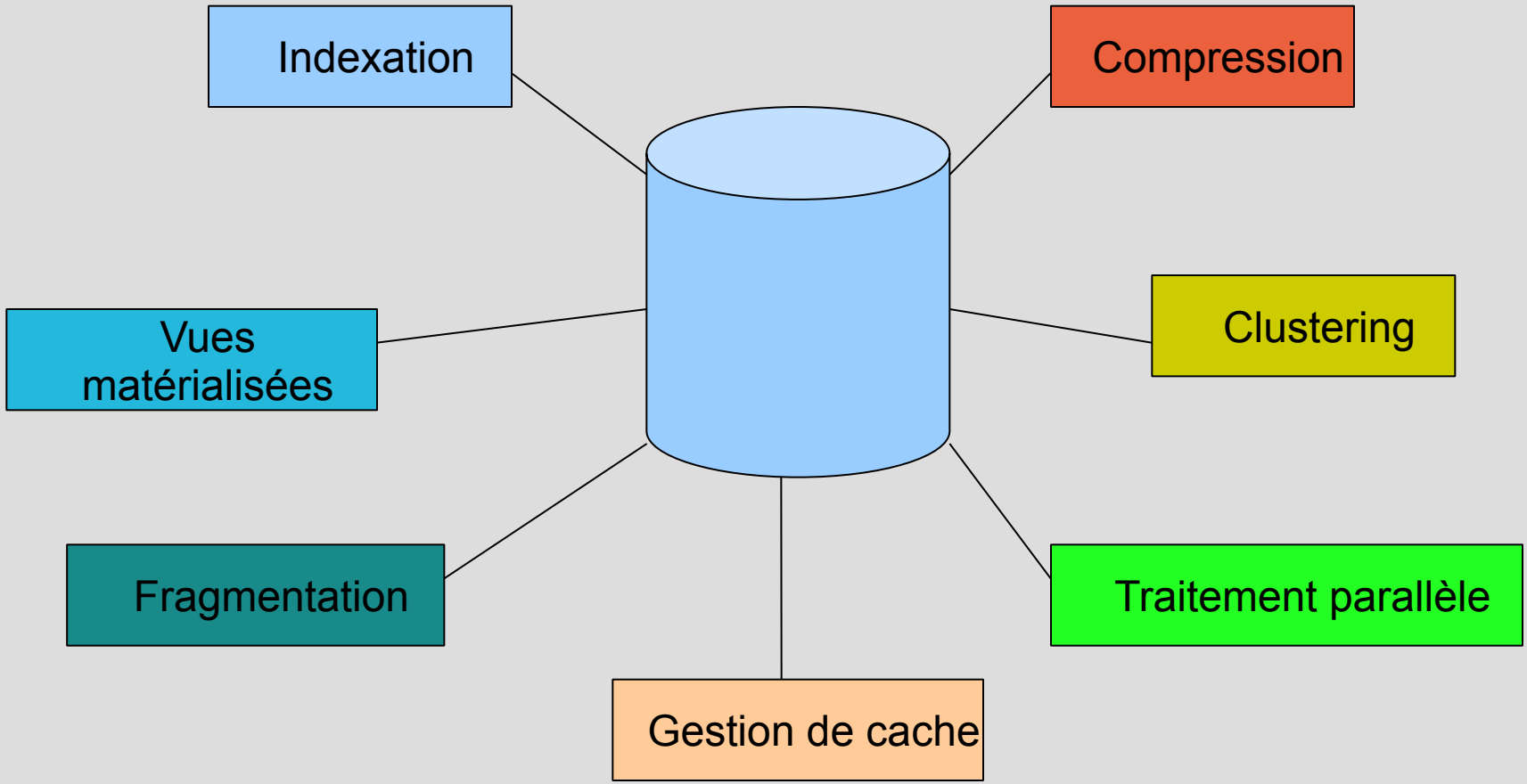
Opportunité de l'optimisation des performances

- Grand volume de données + Traitement analytique des données (OLAP) → Temps de réponses prohibitifs (longs) (Malgré les efforts de dénormalisation dans le schéma en étoile)

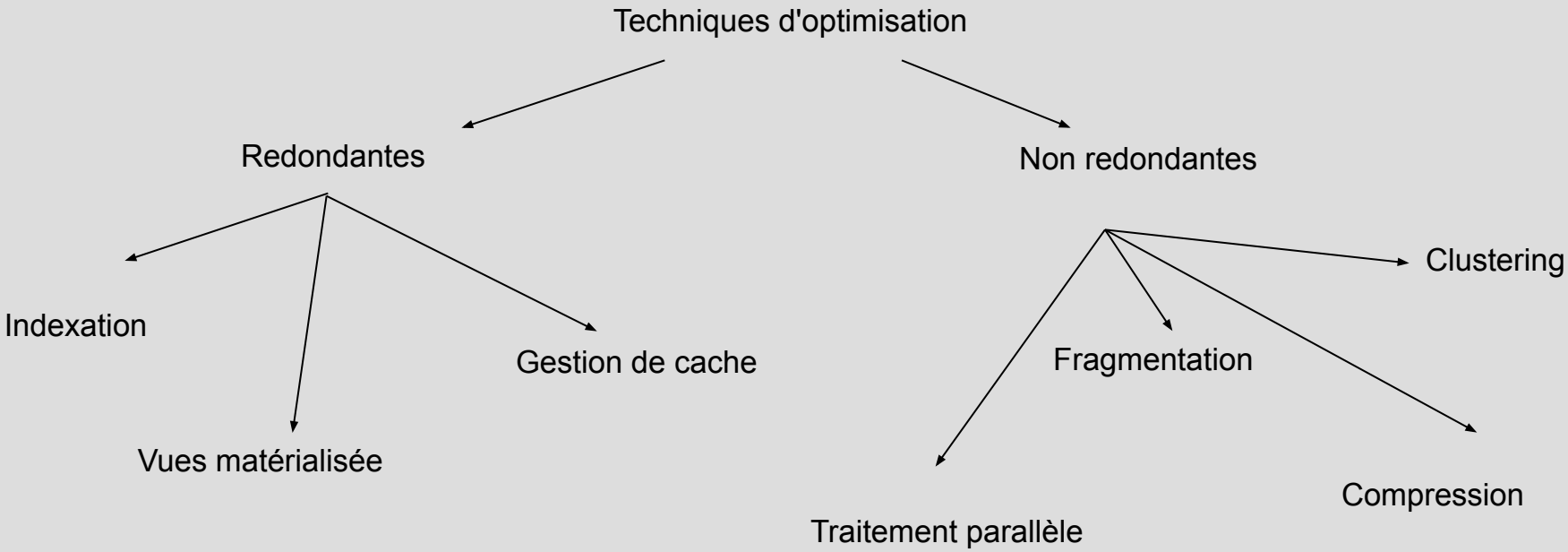
- Les techniques d'optimisation dans les BdD (ex : index,) doivent être adaptés aux entrepôts.

- L'optimisation des performances d'un ED vise à fournir des techniques pour aboutir, entre autres, à des temps de réponses meilleurs.

- **Techniques d'optimisation des performances**



- **Classement des techniques selon la redondance**



- **Survol des techniques :**

  - **Indexation :** structure auxiliaire facilitant l'accès aux données à travers des couples (valeur, RowID).
  - **Vue matérialisée :** requête nommée dont les données sont physiquement sauvegardées (matérialisées).
  - **Fragmentation :** découpage verticale ou horizontal d'une table.
  - **Gestion de cache :** garder les résultats des requêtes dans le cache pour la réutilisation.

- **Survol des techniques :**

  - **Traitement parallèle :** lancer différents traitements en parallèles, notamment pour l'analyse et chargement des données

  - **Clustering :** regrouper physiquement les tables qui sont utilisées co-accédées fréquemment

  - **Compression :** réduire la taille des tables rarement utilisées et / ou qui présentent des taux de redondance élevée.

1. Les vues matérialisées

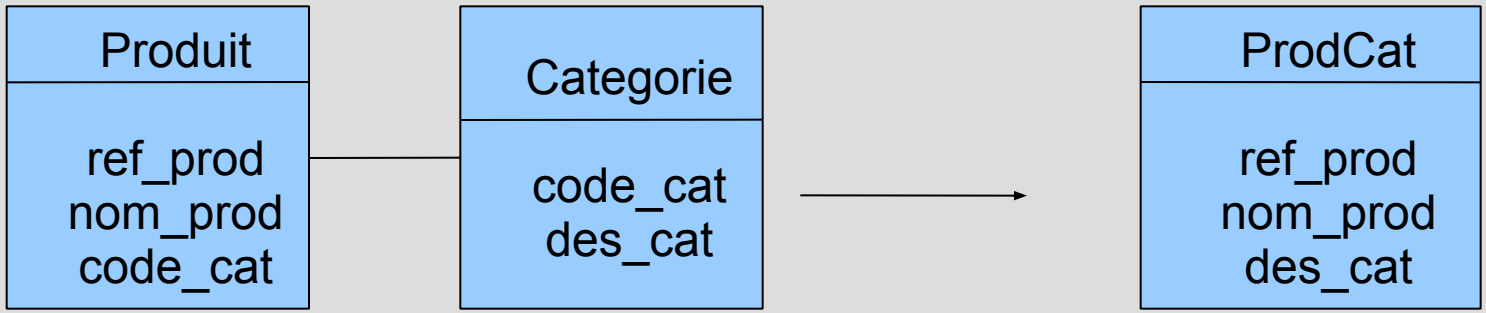
- **Principe :** une vue simple permet d'assurer la confidentialité, la sécurité et la simplification d'écriture des requêtes.

- Mais : une vue simple n'est qu'une requête → elle doit s'exécuter à chaque fois qu'elle est appelée.

- Une vue matérialisée ajoute le stockage (matérialisation) des données.

1. Les vues matérialisées

- **Exemple :**



Code de la VM :

```
Create Materialized View Ma_vue_mat AS
SELECT P.ref_prod, P.nom_prod, C.des_cat
FROM produit P, Categorie C
WHERE P.code_cat = C.code_cat
```

1. Les vues matérialisées

- **Dans le contexte des entrepôts de données**
  - Permettre de simplifier la jointure entre la table de faits et les tables de dimensions
  - Même chose pour les différents membres d'une hiérarchies dans un schéma en flocon de neige
  - Les vues agrégées (cuboïdes) peuvent être matérialisés à travers des VM
  - L'entrepôt de données peut être vu comme une grande vue matérialisée des données sources.

1. Les vues matérialisées

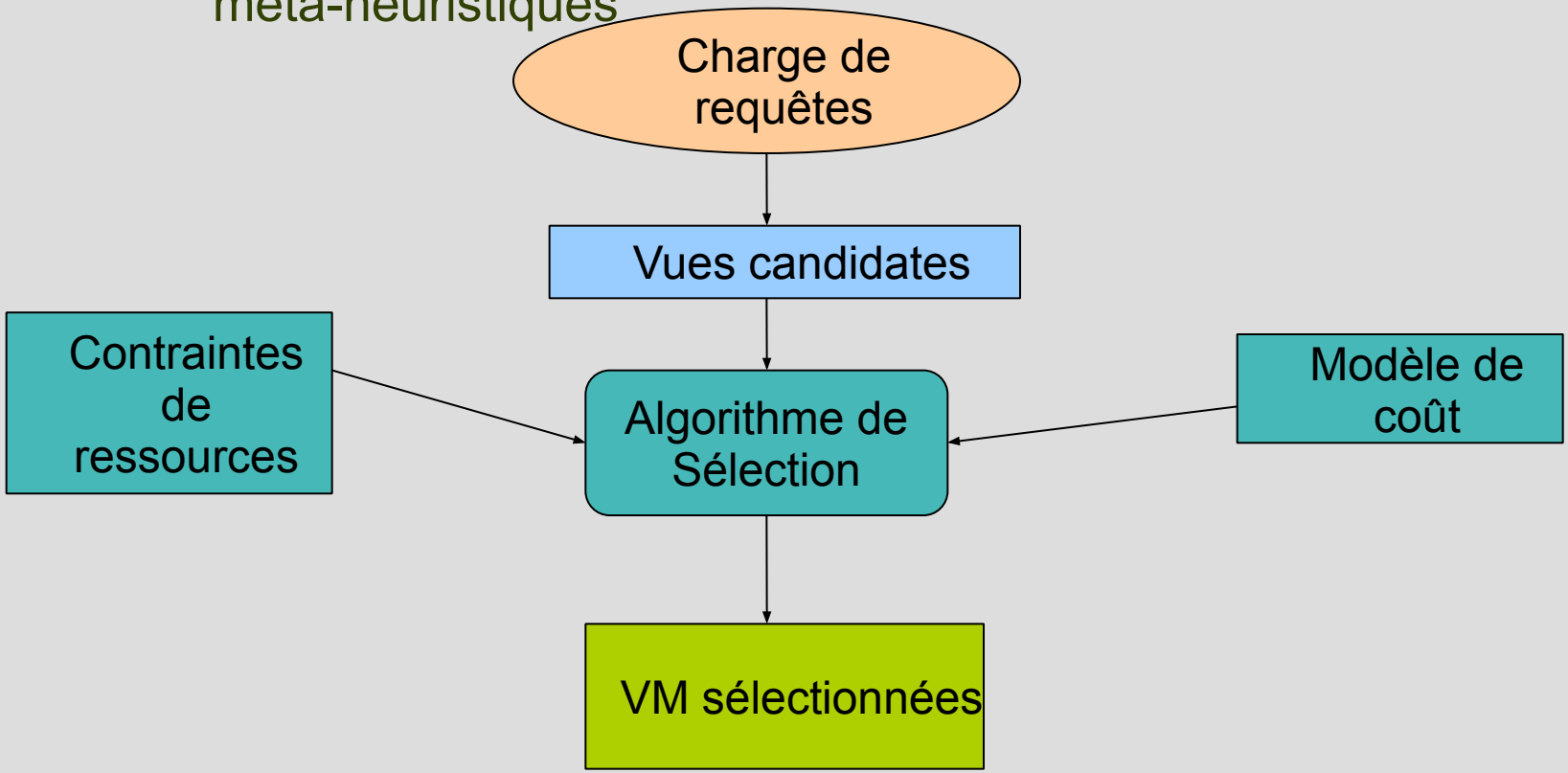
- Problème des vues matérialisées : deux problèmes majeurs
  - Sélection des vues
    - Quels sont les vues à matérialiser ? Problème d'espace.
  - Maintenance des vues
    - Comment maintenir les données des vues à jours (lorsque les données de l'entrepôt changent).

1. Les vues matérialisées

- Problème de sélection des VM
  - Dans un entrepôt de données, les utilisateurs expriment plusieurs requêtes
  - Certaines requêtes peuvent être satisfaites à partir d'une même VM.
  - Problème : Quelle vues faut il matérialiser ?
    - *Ne rien matérialiser → On n'a rien fait*
    - *Tout matérialiser →* problème d'espace, VM qui se chevauchent, et maintenance chère
    - *Sélectionner les vue à matérialiser →* Lesquelles ?
    - Problème connu sous le nom de PSV (Problème de sélection de vues)

1. Les vues matérialisées

- Problème de sélection des VM : Utiliser des méta-heuristiques



1. Les vues matérialisées

- Problème de maintenance des VM
  - Similaire au rafraîchissement d'un ED (processus ETL)
  - Détecter les changements dans les sources et mettre à jour les VM
  - Peut nécessiter le re-calcul de toute la VM ou d'une partie

2. L'indexation

- **Principe :** Structure de table ou de plusieurs tables associant les valeurs clés d'enregistrements à leur adresses relatives dans les fichiers (Gardarin).
- Les index classiques (Arbre B) ne sont pas adaptés aux entrepôts de données
  - Cause : une table de fait est scanné presqu'en totalité → l'accès aux index devient une charge supplémentaire.
- D'autres types index sont requis
  - Index binaire
  - Index de jointure
  - Index de jointure en étoile
  - Index de jointure binaire
  - Index de jointure de dimensions

2. L'indexation

- 2. 1. Index binaire : (BitMap index)
  - S'adaptent à des colonnes ayant une faible cardinalité (peu de valeurs différentes)
  - Principe :

  Pour chaque colonne à indexer
    - Chaque valeur distincte de la colonne est associée à un vecteur binaire (0 ou 1)
    - La longueur du vecteur est égale au nombre d'enregistrement de la table
    - Chaque position du vecteur correspond à un enregistrement
    - Un bit du vecteur correspond à 1 si la valeur de la colonne correspond à la valeur indexée ; 0 sinon

2. L'indexation

- 2. 1. Index binaire : (BitMap index)
  - **Exemple** : on veut indexer les valeurs distinctes des colonnes

Customer	City	Car
c1	Detroit	Ford
c2	Chicago	Honda
c3	Detroit	Honda
c4	Poznan	Ford
c5	Paris	BMW
c6	Paris	Nissan

Customer	City	Car
c1	Detroit	Ford
c2	Chicago	Honda
c3	Detroit	Honda
c4	Poznan	Ford
c5	Paris	BMW
c6	Paris	Nissan

ec1	Chicago	Detroit	Paris	Poznan
1	0	1	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	0	1
5	0	0	1	0
6	0	0	1	0

ec1	BMW	Ford	Honda	Nissan
1	0	1	0	0
2	0	0	1	0
3	0	0	1	0
4	0	1	0	0
5	1	0	0	0
6	0	0	0	1

2. L'indexation

- 2. 1. Index binaire : (BitMap index)
  - Répondre à une requête avec un bitmap

- Quel est le nombre clients qui ont une voiture Honda ?

→ C'est le nombre des 1 au vecteur « Honda » = 2.

- Quel est le nombre clients qui ont une voiture Honda à Detroit ?

→ C'est le AND entre les vecteurs Honda et Detroit: (011000 AND 101000) = 001000 Le nombre = 1

2. L'indexation

- 2. 2. Index de jointure

  - La jointure est l'une des opérations les plus utilisées dans les ED : jointure entre le fait et les dimensions ou jointure entre les membres d'une hiérarchie,

  - La jointure est une opération couteuse → dégradation rapide des performances.

  - Index de jointure → dédié aux opérations de jointure.

2. L'indexation

- 2. 2. Index de jointure

Principe : *structure qui pré-calcule la jointure entre les tables, et stocke les références des tuples qui joignent les tables permettant ainsi l'optimisation des requêtes de jointure.*

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

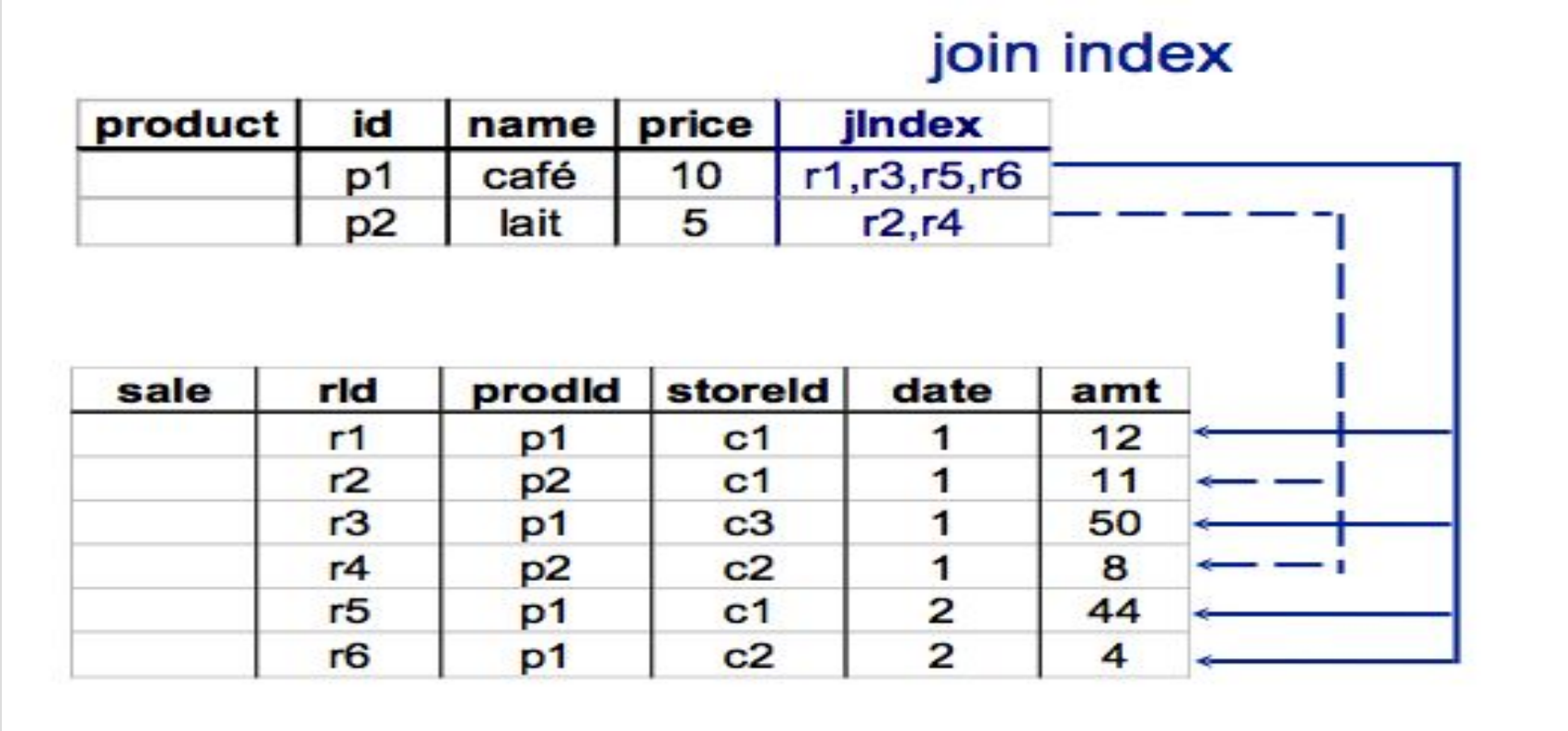
product	id	name	price
	p1	café	10
	p2	lait	5

	p1	café	10	c1	1	12
	p2	lait	5	c1	1	11
	p1	café	10	c3	1	50
	p2	lait	5	c2	1	8
	p1	café	10	c1	2	44
	p1	café	10	c2	2	4

2. L'indexation

- 2. 2. Index de jointure

Exemple : stocker dans une table les RowID des tuples qui lui correspondent



2. L'indexation

- 2. 3. Index de jointure en étoile

Principe : Un **index de jointure en étoile (IJE)** contient toutes les combinaisons possibles entre l'identifiant de la table des faits et les clés étrangères des tables de dimension.

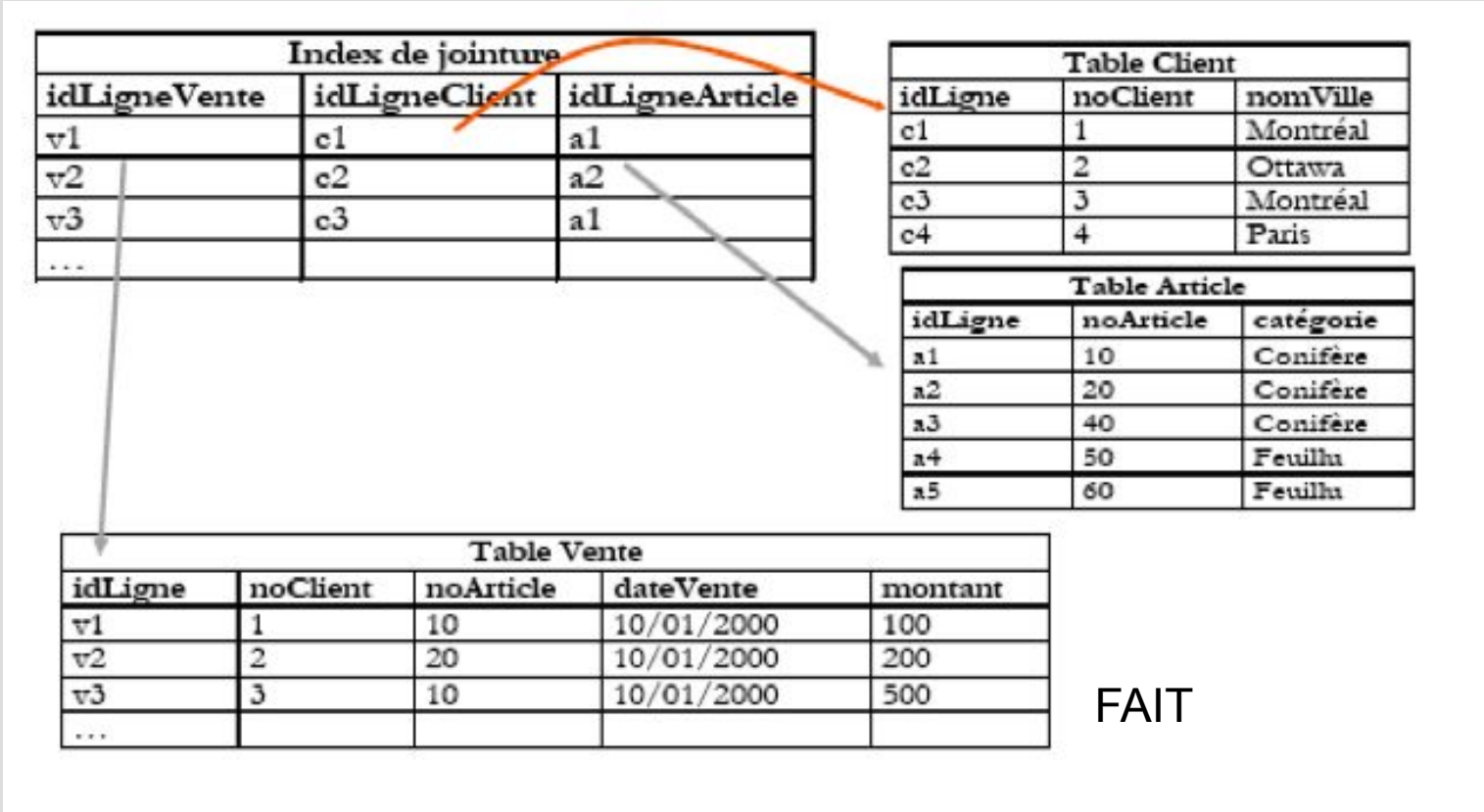
- Un IJE est **complet** s'il est construit en joignant toutes les tables de dimension avec la table des faits.

- Il est **partiel** s'il est construit en joignant certaines tables de dimension avec la table des faits.

2. L'indexation

- 2. 3. Index de jointure en étoile

Exemple



Index de jointure

idLigneVente	idLigneClient	idLigneArticle
v1	c1	a1
v2	c2	a2
v3	c3	a1
…		

Table Client

idLigne	noClient	nomVille
c1	1	Montréal
c2	2	Ottawa
c3	3	Montréal
c4	4	Paris

Table Article

idLigne	noArticle	catégorie
a1	10	Conifère
a2	20	Conifère
a3	40	Conifère
a4	50	Feuillu
a5	60	Feuillu

Table Vente

idLigne	noClient	noArticle	dateVente	montant
v1	1	10	10/01/2000	100
v2	2	20	10/01/2000	200
v3	3	10	10/01/2000	500
…				

FAIT

2. L'indexation

- 2. 4. Index de jointure binaire

Un index de jointure binaire (IJB) ou **Bitmap Join Index** est une extension de l'index bitmap. Un IJB créé un index binaire sur une colonne d'une table A et l'associe aux tuples d'une autre table B qui est jointe à A. Les requêtes sur la table B qui nécessite de tester la colonne indexée dans A n'auront pas à accéder à cette dernière.

Exemple: soit les deux tables

Ventes (clé_produit, clé_client, ...) et

Clients (clé_client, situation_familale...).

Soit la requête SELECT V.date_vente, V.clé_client, C.situation_familiale FROM ventes V, clients C where V.clé_client = C.clé_client and situation_familiale = 'marié'.

Dans ce cas, on peut créer un index sur la colonne situation_familiale pour éviter de référencer la table client.

2. L'indexation

- 2. 5. Index de jointure de dimension

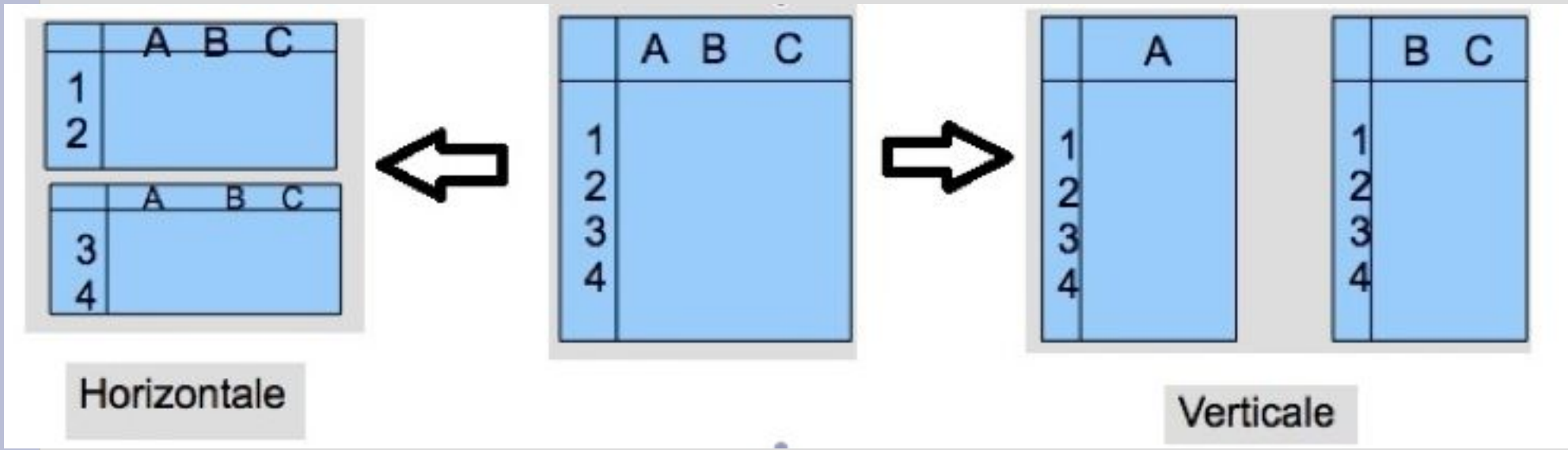
Un index de jointure de dimensions est une structure binaire adéquate à la jointure entre la table de fait et une table de dimension appartenant à une hiérarchie dans un schéma en flocon de neige.

Exemple : soit un schéma en flocon de neige composé d'une table de fait Ventes, et de plusieurs dimensions dont une dimension appelée Client hiérarchisée comme suit : Client → Ville → Wilaya → Pays.

Pour les requêtes nécessitant d'afficher la désignation des pays des clients avec les données de la table de fait, on peut créer un index de jointure binaire sur la colonne Pays.designation en effectuant la jointure entre les tables Ventes, Client, Ville, Wilaya et Pays.

3. La fragmentation

- **Principe de la fragmentation :** la fragmentation repose sur la supposition ou l'observation que les données d'une tables ne sont pas accessibles toutes à la fois soit en termes de colonnes ou de tuples, à cause par exemple de la répartition géographique des tuples (ventes est, ventes ouest...) ou la nature des requêtes (accès fréquent à un groupe de colonnes ou de tuples).



3. La fragmentation

- **Principe de la fragmentation :**
  - Découper une table relationnelle horizontalement ou verticalement en fragments
  - Chaque fragment étant plus petit que la table d'origine, on accède à MOINS de données
  - La fragmentation doit remplir trois conditions :
    - Complétude
    - Disjonction
    - Réversibilité (reconstruction)

3. La fragmentation

- **Exemple de la fragmentation horizontale:**

Car_ID	Manufacturer	Color	Owner_ID
1	Renault	Red	100
2	Ford	Blue	101
3	Peugeot	White	102
4	Ford	Blue	103
5	Peugeot	White	104
6	Renault	Red	105
7	Suzuki	Red	106
8	Suzuki	White	107

Cars

Car_ID	Manufacturer	Color	Owner_ID
1	Renault	Red	100
6	Renault	Red	105
7	Suzuki	Red	106

Red Cars

Car_ID	Manufacturer	Color	Owner_ID
3	Peugeot	White	102
5	Peugeot	White	104
8	Suzuki	White	107

White Cars

Car_ID	Manufacturer	Color	Owner_ID
2	Ford	Blue	101
4	Ford	Blue	103

Blue Cars

3. La fragmentation

- **Exemple de la fragmentation verticale:**

Car_ID	Manufacturer	Color	Owner_ID
1	Renault	Red	100
2	Ford	Blue	101
3	Peugeot	White	102
4	Ford	Blue	103
5	Peugeot	White	104
6	Renault	Red	105
7	Suzuki	Red	106
8	Suzuki	White	107

Cars

Car_ID	Manufacturer	Owner_ID
1	Renault	100
2	Ford	101
3	Peugeot	102
4	Ford	103
5	Peugeot	104
6	Renault	105
7	Suzuki	106
8	Suzuki	107

Manufacturer

Car_ID	Color	Owner_ID
1	Red	100
2	Blue	101
3	White	102
4	Blue	103
5	White	104
6	Red	105
7	Red	106
8	White	107

Color

3. La fragmentation

- **Complétude**
  - Fragmentation horizontale : chaque tuple de la table d'origine doit faire partie d'AU MOINS un fragment.
  - Fragmentation verticale: chaque colonne de la table d'origine doit faire partie d'AU MOINS un fragment.

3. La fragmentation

- **Disjonction**
  - Fragmentation horizontale : chaque tuple de la table d'origine doit appartenir à un et un seul fragment. L'intersection de deux fragments doit donner l'ensemble vide.
  - Fragmentation verticale: chaque colonne de la table d'origine doit appartenir à un et un seul fragment. Ne s'applique pas au clés primaires et aux clés étrangères.

3. La fragmentation horizontale

- **Réversibilité**
  - Fragmentation horizontale : la construction par UNION des fragments doit donner la table d'origine.
  - Fragmentation verticale: la construction par JOINTURE des fragments doit donner la table d'origine.

3. 2. La fragmentation horizontale dans les ED

- La FH est bénéfique pour les sélections

- La sélection porte fréquemment sur les tables de dimensions

→ Fragmenter d'abord la table de dimension selon des prédicats (fragmentation primaire)

→ Fragmenter ensuite la table de fait selon les fragments de la dimension (fragmentation dérivée)

3. 2. La fragmentation horizontale dans les ED

- Exemple de FH