Master d'Informatique – M1

4IN803 – COURS 5

Conception de BD réparties

Fragmentation

2020

Bases de Données Réparties

- Définition
- Conception
- Décomposition
- Fragmentation horizontale et verticale
- Outils d'interface SGBD
  - extracteurs, passerelles
- Réplication
- SGBD répartis hétérogènes

BD réparties (1)

- Principe
  - Un site héberge une BD : accès local rapide, interne au site.
  - Accès global possible à des BD situées sur des sites externes
- Plusieurs niveaux d'intégration :
  - Accès distant (Remote Data Access)
  - Vues réparties
    - Extension du mécanisme de vues
    - Définir des vues sur plusieurs sites.
  - Médiateurs
  - BD réparties/fédérées

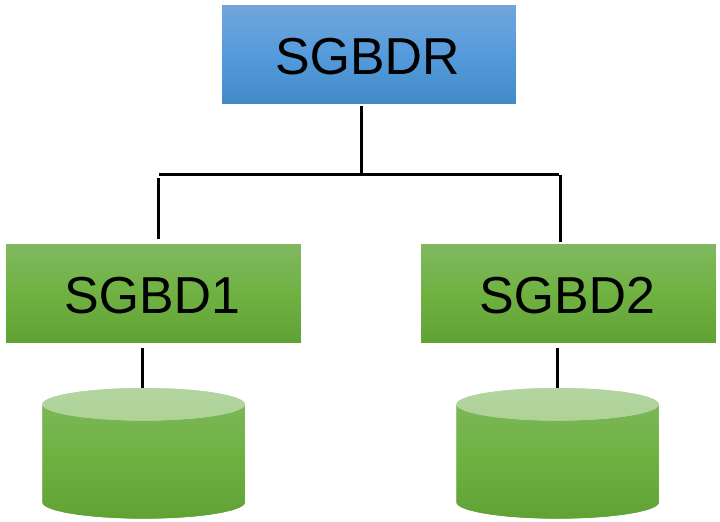
BD Réparties (2)

- BD réparties :
  - Plusieurs BD sur plusieurs sites, mais une seule BD « logique ».
  - Fédérée : intègre des bases et des schémas existants
  - Répartie « pur » : conçue répartie. Pas d'accès locaux

- Les ordinateurs (appelés sites) communiquent via le réseau et sont faiblement couplés
  - pas de partage de MC, disque, au contraire de *BD parallèles*

- Chaque site
  - contient des données de la base,
  - peut exécuter des transactions/requêtes locales et
  - participer à l'exécution de transactions/requêtes globales

SGBD réparti

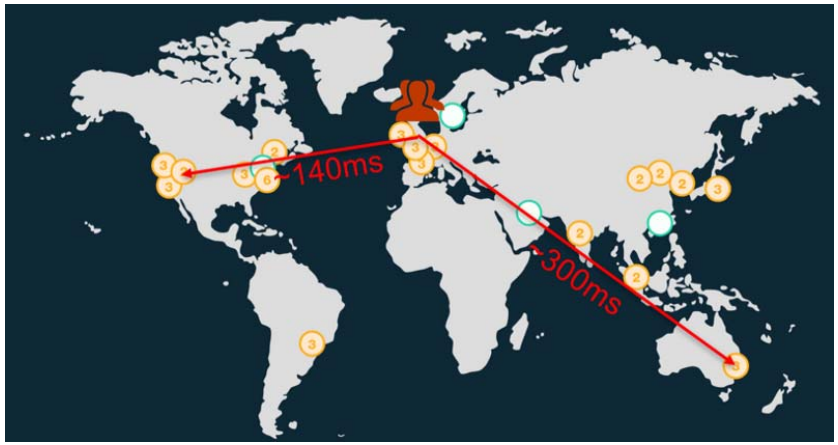


Rend la répartition (ou distribution) *transparente*

- dictionnaire des données (catalogue, métabase) réparties
- traitement des requêtes réparties
- gestion de transactions réparties
- gestion de la cohérence et de la sécurité

Paramètres à considérer

- Coût et temps de communication entre deux sites
  - Accès réseau (longue distance, WAN, MAN) beaucoup plus coûteux que accès disque
- Fiabilité : fréquence des pannes
  - des sites, du réseau (cf. P2P)
- Accessibilité aux données
  - accès aux données en cas de panne des sites, du réseau.
  - accès aux sites les moins encombrés, les plus puissants



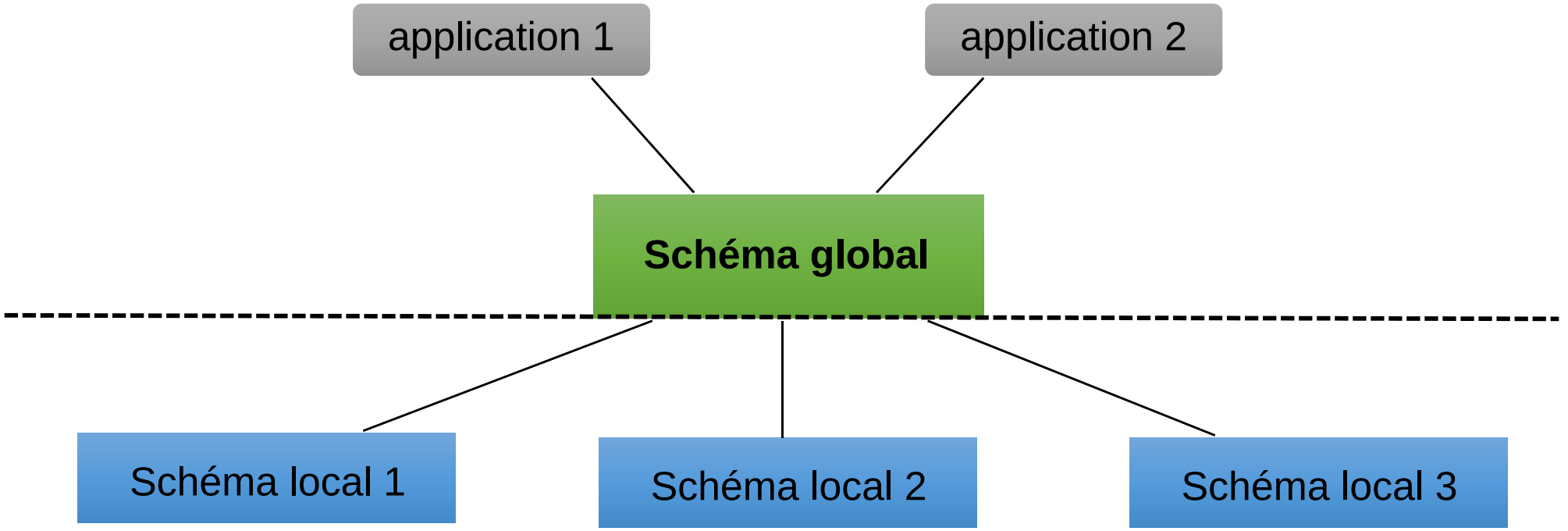
- A lire : https://read.acloud.guru/why-and-how-do-we-build-a-multi-region-active-active-architecture-6d81acb7d208

Evaluation de l'approche BDR

- avantages
  - extensibilité
  - partage des données hétérogènes et réparties
  - performances avec le parallélisme
  - Disponibilité et localité avec la réplication

- inconvénients
  - administration complexe
  - complexité de mise en œuvre
  - distribution du contrôle
  - surcharge (l’échange de messages augmente le temps de calcul)

Architecture de schémas



- indépendance applications / bases locales
- schéma global lourd à gérer

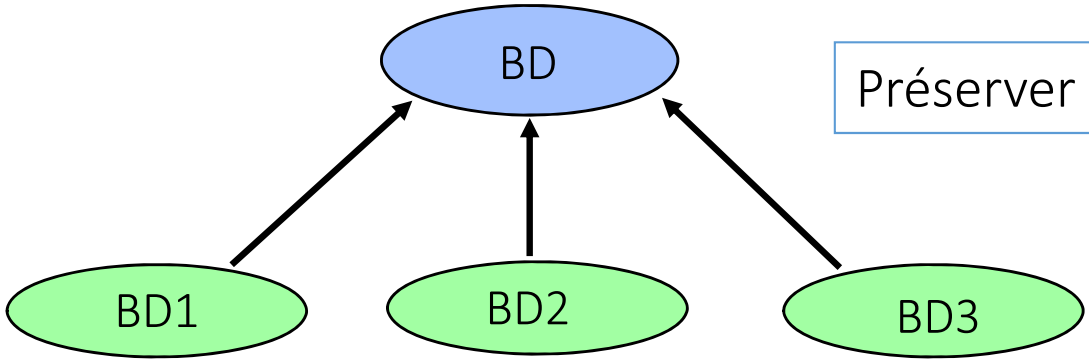
Schéma global

- Schéma conceptuel global
  - description globale et unifiée de toutes les données de la BDR
    - Nom des relations avec leurs attributs
  - Fournir l'indépendance à la répartition
- Schéma de placement
  - règles de correspondance avec les données locales
    - Vues globales définies sur les relations locales (cf. global as view)
  - Fournit l'indépendance à la localisation, la fragmentation et la duplication

- Le schéma global fait partie du dictionnaire de la BDR et peut être conçu comme une BDR (dupliqué ou fragmenté)

Migration vers une BDR : 2 approches

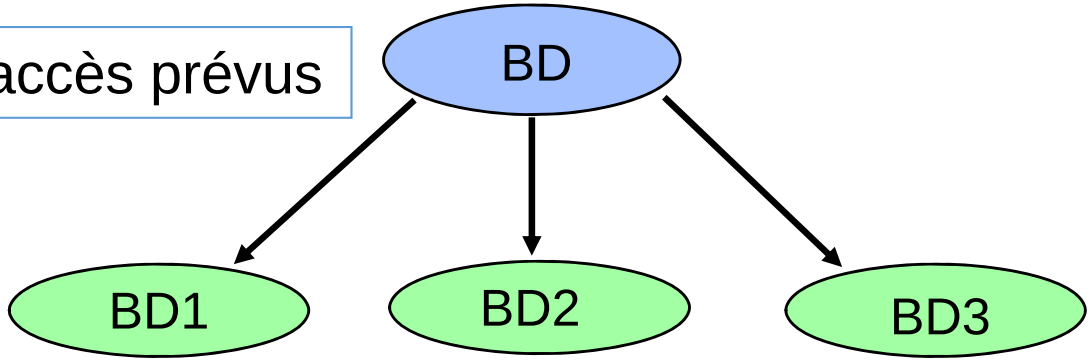
Intégration logique des BD locales existantes (fédérée, médiateur)



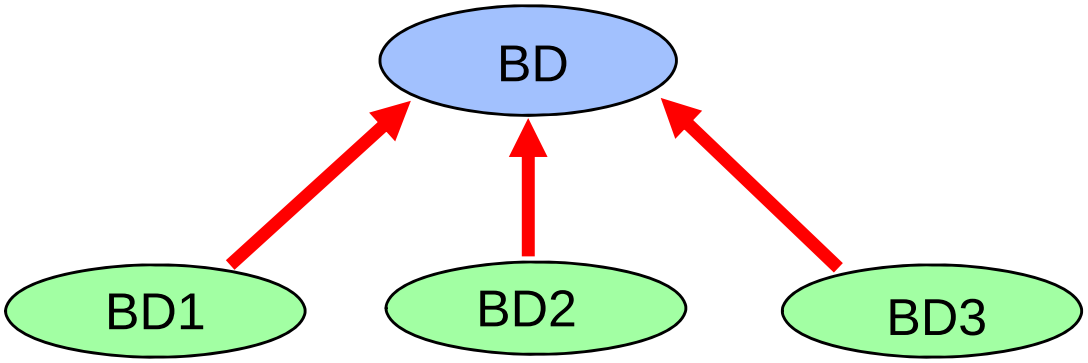
Préserver l'autonomie

Décomposition en BD locales : répartie « pur »

En fonction des accès prévus

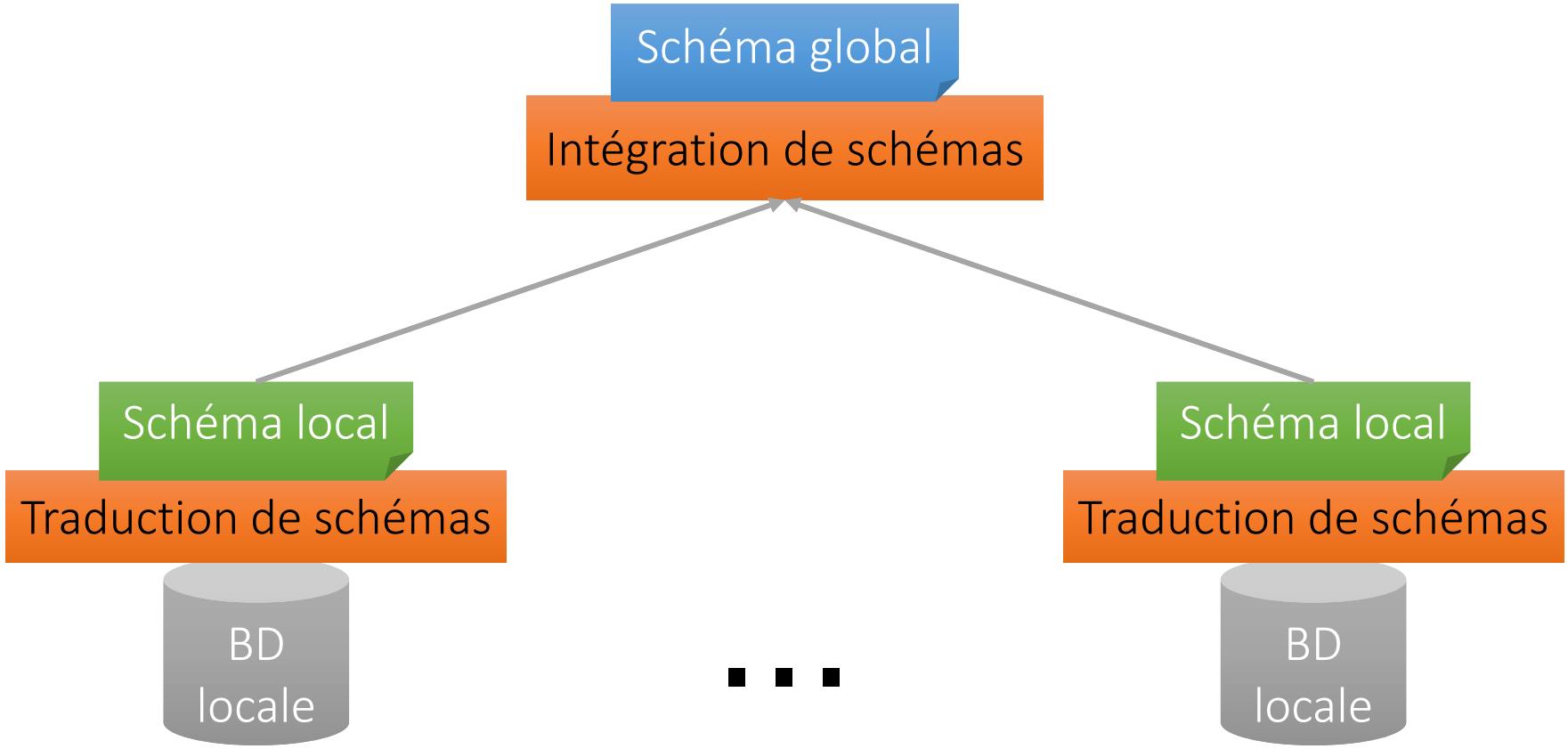


Intégration de BD existantes



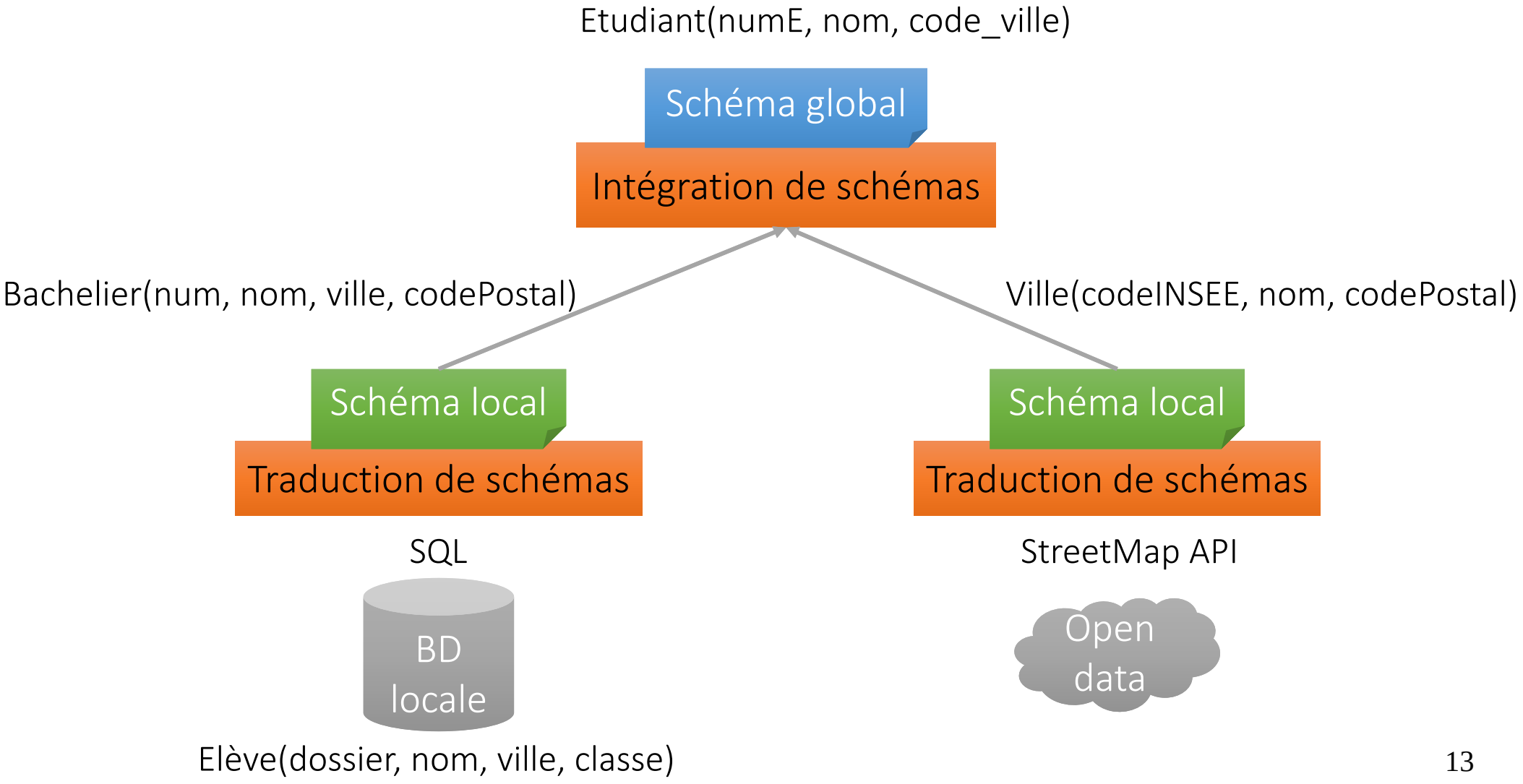
Conception d'une BDR par intégration

Approche médiateur



Conception d'une BDR par intégration

Exemple



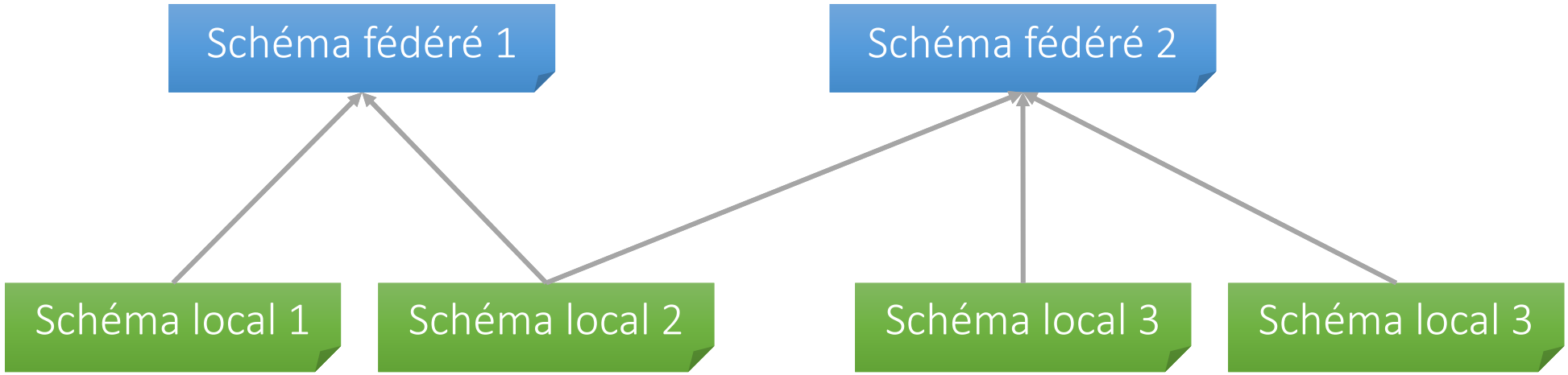
Intégration de schémas

- 1. pré-intégration
  - Les schémas sont transformés pour les rendre plus homogènes
  - identification des éléments reliés (e.g. domaines équivalents) et établissement des règles de conversion (e.g. 1 inch = 2,54 cm)
  - Pbs : hétérogénéité des modèles de données, des puissances d'expression, des modélisations

- 2. comparaison
  - identification des conflits de noms (synonymes et homonymes) et des conflits structurels (types, clés, dépendances)

- 3. conformance
  - résolution des conflits de noms (renommage) et des conflits structurels (changements de clés, tables d'équivalence)
  - Définition de règles de traduction entre le schéma intégré et les schémas initiaux.

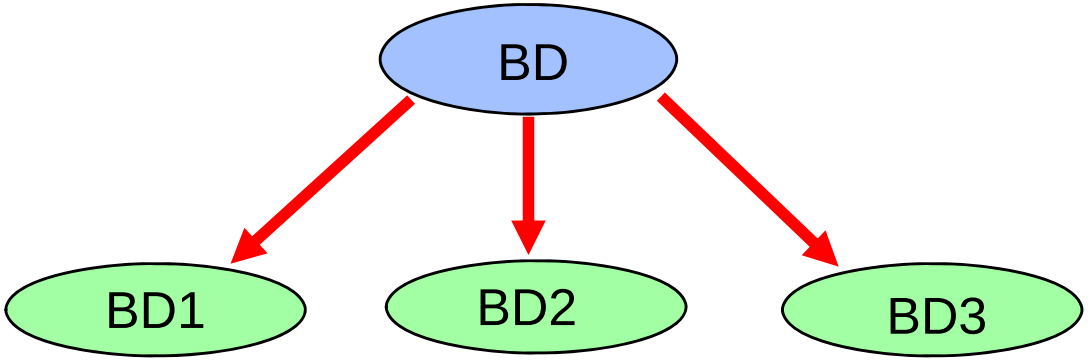
Architecture fédérée



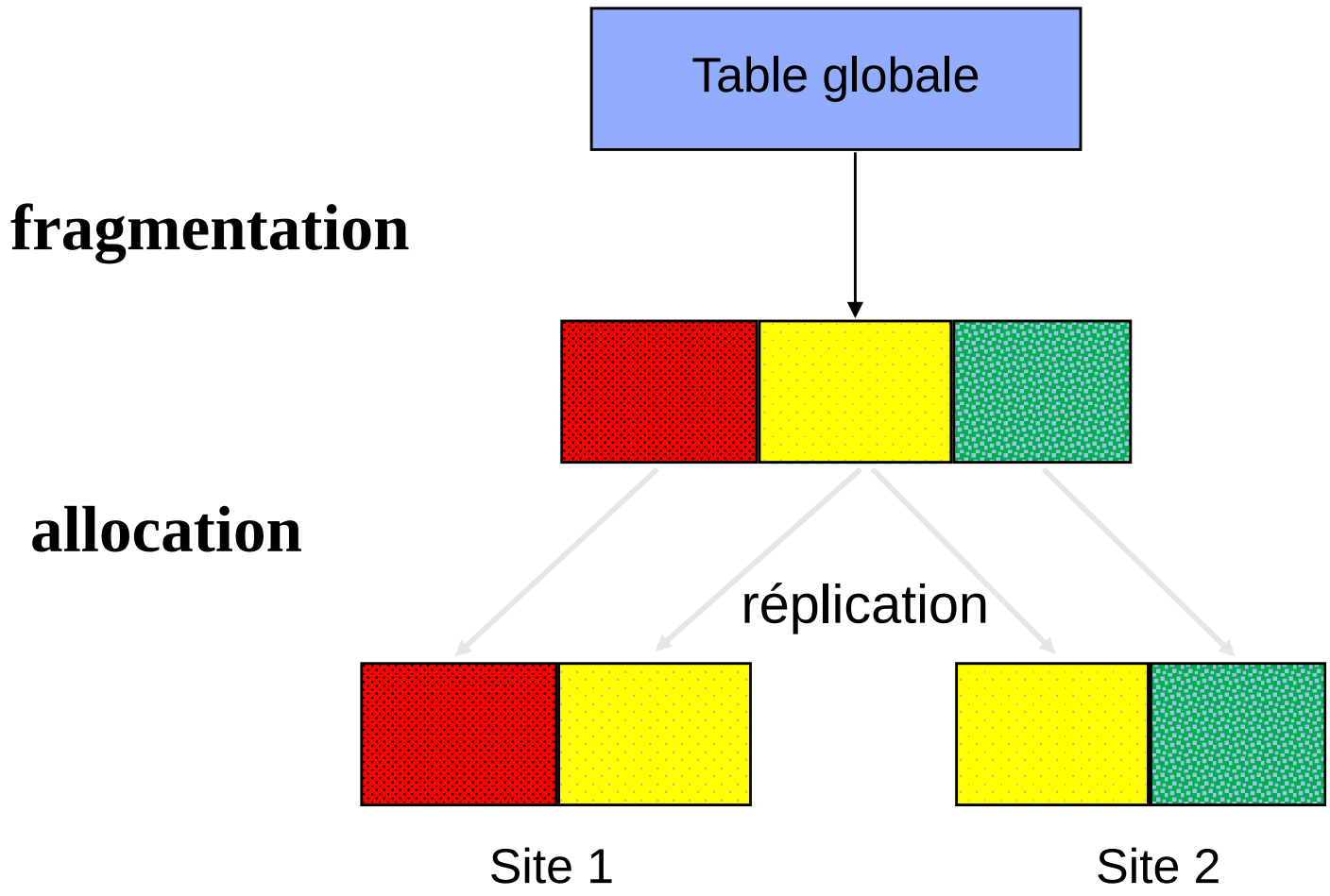
Moyen contrôlé de migration

Plusieurs niveaux possibles

Décomposition



Conception par décomposition



Objectifs de la décomposition

Fragmentation

- Trois types : horizontale, horizontale dérivée, verticale
  - Possibilité de composer plusieurs fragmentations: mixte
- Performances en favorisant les accès (et traitements) locaux
- Equilibrer la charge de travail entre les sites (parallélisme)
- Contrôle de concurrence plus simple pour les accès à un seul fragment

Trop fragmenter : BD éclatée, nombreuses jointures réparties

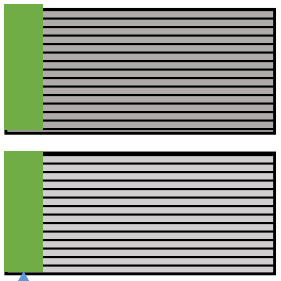
Duplication (ou réplication)

- favoriser les accès locaux
- augmenter la disponibilité des données

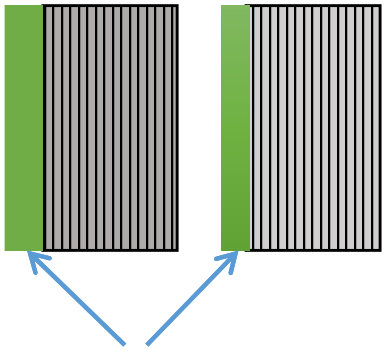
Trop répliquer : surcoût de maintenir cohérence des répliques

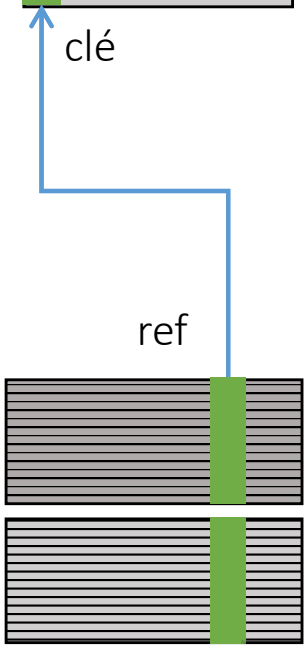
Types de Fragmentation

horizontale



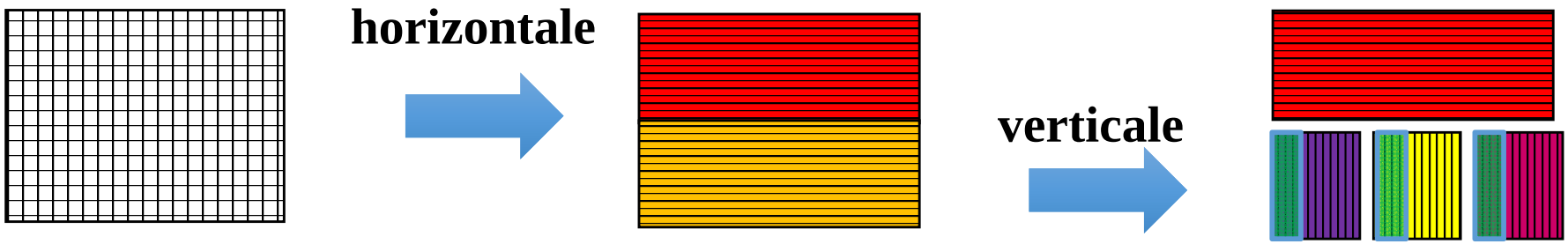
verticale

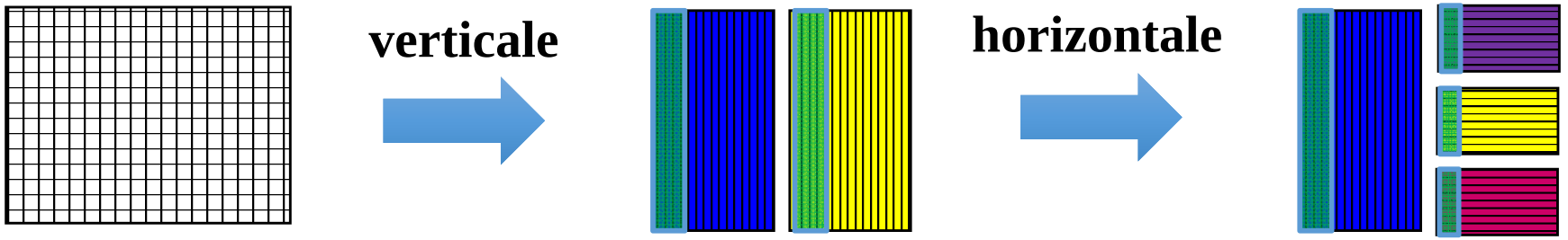




horizontale dérivée

Fragmentation Mixte





Fragmentation correcte

Complète

- chaque élément de R doit se trouver dans un fragment

Reconstructible

- on doit pouvoir recomposer R à partir de ses fragments (ressemble à décomposition de schéma vue en Li341 pour fragmentation verticale)

[Disjointe] /*si on veut éviter réplication pour cohérence */

- chaque élément de R ne doit pas être dupliqué (sauf clé en cas de fragmentation verticale)

En pratique : seulement Complète et Reconstructible, mais pas disjointe.

Fragmentation Horizontale

Fragments définis par **sélection**

$Client_1 = \sigma_{ville = 'Paris'} Client$

$Client_2 = \sigma_{ville \neq 'Paris'} Client$

Inférence : correcte

Reconstruction par union

$Client = Client_1 \cup Client_2$

En SQL :

```
create view Client as
   select * from Client1
   union
   select * from Client2
```

Client

nclient	nom	ville
C 1	Dupont	Paris
C 2	Martin	Lyon
C 3	Martin	Paris
C 4	Smith	Lille

Client1

nclient	nom	ville
C 1	Dupont	Paris
C 3	Martin	Paris

Client2

nclient	nom	ville
C 2	Martin	Lyon
C 4	Smith	Lille

Fragmentation Horizontale : Exemples

Fragments définis par **sélection d'intervalles**

Ville(numV, nom, cp, population)

$V_1 = \sigma_{\text{population} < 10K}\ \text{Ville}$
$V_2 = \sigma_{10K \leq \text{population} < 50K}\ \text{Ville}$
$V_3 = \sigma_{50K \leq \text{population}}\ \text{Ville}$

Fragments définis par **sélection sur plusieurs attributs**

Personne(numP, nom, prenom, age, statut)

$P_1 = \sigma_{\text{age} < 18 \text{ and statut} = \text{'élève'}}\text{Personne}$
$P_2 = \sigma_{\text{age} < 18 \text{ and statut} = \text{'employé'}}\text{Personne}$
$P_3 = \sigma_{\text{age} \geq 18}\ \text{Personne}$

Fragmentation Horizontale par sélection

- Fragmentation de R selon *n* prédicats
  - les prédicats $\{p_1, \ldots, p_n\}$ ex: {a<10, a > 5, b='x', b='y'}
- L'ensemble *M* des prédicats de fragmentation est :
  - $M = \{ m \mid m = \bigwedge_{1 \le k \le n} p_k^* \}$ avec $p_k^* \in \{p_k, \neg p_k\}$
  - Eliminer les *m* de sélectivité nulle ex: $a>10 \wedge a < 5$
  - Simplifier: $a<10 \wedge a \le 5 \wedge b='x' \wedge b \neq 'y'$ devient $a \le 5 \wedge b='x'$
- Construire les fragments $\{R_1, \ldots, R_k\}$
  - Pour chaque m_i , $R_i = \sigma m_i (R)$
- Minimalité
  - Ne pas avoir 2 fragments toujours lus ensemble
- Choisir les p_i des requêtes les plus fréquentes

- Ref biblio récente (2015) sur la fragmentation et le choix des fragments optimaux. Regroupement hiérarchique d'ensembles de nuplets issus de requêtes fréquentes.
  - Waterloo Univ : G. Aluç, M. T. Özsu, K. Daudjee and O. Hartig.
    - "Executing Queries over Schemaless RDF Databases", ICDE 2015 (Int'l Conf. on Data Engineering)

Fragmentation Horizontale Dérivée

Fragments définis par semi jointure

Cde1 = select Cde.*
from Cde, Client 1
where Cde.nclient = Client1.nclient

$Cde_i = Cde \ltimes Client_i$ pour *i* dans{1; 2}

Reconstruction par **union**

$Cde = Cde_1 \cup Cde_2 = \cup_i Cde_i$

Cde

ncde	nclient	produit	qté
D 1	C 1	P 1	10
D 2	C 1	P 2	20
D 3	C 2	P 3	5
D 4	C 4	P 4	10

Cde1

ncde	nclient	produit	qté
D 1	C 1	P 1	10
D 2	C 1	P 2	20

Cde2

ncde	nclient	produit	qté
D 3	C 2	P 3	5
D 4	C 4	P 4	10

Fragmentation Horizontale Dérivée
Méthode générale

- Commencer par recenser les fragments horizontaux déjà définis
- Repérer des clés étrangères dans la relation à fragmenter
- Semi-jointure(s)

Exemple avec 2 semi-jointures

- Personne(<u>numP</u>, nom, premon)
  - 4 fragments pour a=[0,18,25,60,100]
  - $P_i = \sigma_{a_{i-1} \leq age < a_i}$ Personne
- Sport(<u>numS</u>, type)
  - 2 fragments pour t=[indiv, collectif]
  - $S_j = \sigma_{type = t_j}$ Sport

- Inscrit(numP*, numS*, date)
  - 4*2 = 8 fragments: $Inscrit_{i\,j} = Inscrit \ltimes P_i \ltimes S_j$

Propriétés de la fragmentation horizontale dérivée

R: fragmentation horizontale → fragments R_i

S: fragmentation horizontale dérivée → fragments $S_i = S \ltimes_A R_i$

- Complète
  - Chaque tuple de S doit joindre avec au moins un tuple de R
    - $\forall s \in S, \exists t \in S_i, s = t$
- Disjointe
  - $\forall i,j$ tq $i \neq j$, $S_i \cap S_j = (S \ltimes R_i) \cap (S \ltimes R_j) = S \ltimes (R_i \cap R_j) = \varnothing$
    - Rappel: $R1 \cap R2 \Leftrightarrow R1 \ltimes R2$
- Reconstructible
  - $\cup_i S_i = (S \ltimes R_1) \cup (S \ltimes R_2) \cup \ldots \cup (S \ltimes R_n) = S \ltimes (\cup_i R_i) = S$

  $\Rightarrow$ contrainte d'intégrité référentielle
  - A est une clé de R
  - S.A référence R.A $\forall s \in S, \exists r \in R, s.A = r.A$

Fragmentation Verticale

Fragments définis par projection

Cde1 = $\pi_{\text{ncde, nclient}}$ Cde

Cde2 = $\pi_{\text{ncde, produit, qté}}$ Cde

Cde

ncde	nclient	produit	qté
D 1	C 1	P 1	10
D 2	C 1	P 2	20
D 3	C 2	P 3	5
D 4	C 4	P 4	10

Reconstruction par jointure

Cde = Cde1 ⋈ Cde2

En SQL :

create view Cde as

select * from Cde_1 ,Cde_2

where Cde_1.ncde = Cde_2.ncde

Cde1

ncde	nclient
D 1	C 1
D 2	C 1
D 3	C 2
D 4	C 4

Cde2

ncde	produit	qté
D 1	P 1	10
D 2	P 2	20
D 3	P 3	5
D 4	P 4	10

Fragmentation Verticale

Comment définir une fragmentation verticale ?

- Affinité des attributs : mesure la proximité sémantique des attributs (combien « ils vont ensembles »)
  - Soit par connaissance de l'application,
  - Soit par analyse des requêtes (on mesure combien de fois deux attributs donnés ont été interrogé ensembles)
  - Résultat sous forme de matrice d'affinité
- 2 approches : regroupement, partitionnement (grouping – splitting)
  - Idem que pour optimisation de schéma relationnel SPI, SPD
- Algorithme de regroupement des attributs bien adapté
  - BEA : bond energy algorithm (Mc Cormick et al. 72) : $O(n^2)$
  - insensible à l'ordre de départ des attributs
  - part des attributs individuels et effectue des regroupements de groupes
- Algorithme de partitionnement
  - Part d'une relation et observe le bénéfice qu'on peut tirer à partitionner

Matrice d’affinité des attributs

Matrice A

aij = affinité de Ai avec Aj

ex: nb de requêtes qui accèdent Ai et Aj

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

Matrice d’affinité
regroupement des attributs

Matrice A

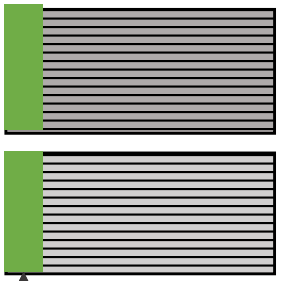
	A1	A3	A2	A4
A1	45	45	0	0
A3		53	5	3
A2			80	75
A4				78

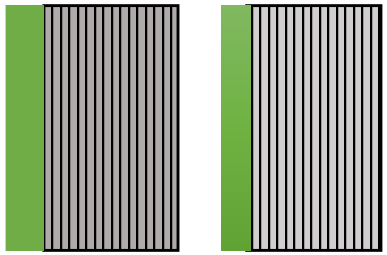
Résumé des Fragmentations

HORIZONTALE

Fragments = sélection

Globale = UNION





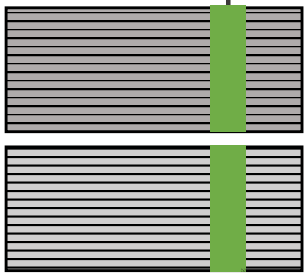
VERTICALE

Fragments = Projection contenant la clé

Globale = Jointure

clé

ref



HORIZONTALE Dérivée

Fragments = semi-jointure avec des fragments horizontaux

Globale = UNION

Allocation des Fragments aux Sites

Non-répliquée

- partitionnée : chaque fragment réside sur un seul site

Dupliquée

- chaque fragment sur un ou plusieurs sites
- maintien de la cohérence des copies multiples : coûteux
- **(le fameux) Compromis Lecture/écriture**:

  + le ratio Lectures/màj est > 1, + la duplication est avantageuse

Allocation de Fragments

Problème: Soit

F un ensemble de fragments

S un ensemble de sites

Q un ensemble d'applications et leurs caractéristiques

trouver la distribution "optimale" de *F* sur *S*

Optimum

- coût minimal de communication, stockage et traitement
- Performance = temps de réponse ou débit

Solution

- allouer une copie de fragment là où le bénéfice est supérieur au coût

Exemple d'Allocation de Fragments

Client1

nclient	nom	ville
C 1 C 3	Dupont Martin	Paris Paris

Client2

nclient	nom	ville
C 2 C 4	Martin Smith	Lyon Lille

Cde1 = Cde ⋉ Client1

ncde	client	produit	qté
D 1 D 2	C 1 C 1	P 1 P 2	10 20

Site 1

Cde2

ncde	client	produit	qté
D 3 D 4	C 2 C 4	P 3 P 4	5 10

Site 2

Exemple : Sport universitaire

Trois universités parisiennes (Jussieu, Sorbonne, Dauphine) ont décidé de mutualiser leurs équipements sportifs (locaux) et les entraîneurs. La gestion commune est effectuée par une base de données répartie, dont le schéma **global** est le suivant :

PROF (Idprof, nom, adresse, tél, affectation, salaire)

ETUDIANT (Idetu, nom, adresse, assurance, police, université, équipe)

LOCAUX (Idlocal, adresse, université)

EQUIPE (équipe, sport, niveau)

HORAIRE (Idlocal, équipe, jour, heure_début, heure_fin, prof)

- Chaque université rémunère ses profs en envoyant un chèque à leur adresse, mais aussi elle doit pouvoir contacter tout prof qui utilise ses locaux.
- Chaque équipe correspond à un sport. La plupart des équipes ont droit à un (seul) créneau (jour, heure) dans un des locaux communs pour leur entrainement. Cependant, pour le sport «cyclisme », il n'y pas besoin de locaux.
- Chaque université gère évidemment ses propres étudiants, ainsi que ses locaux et les créneaux correspondants.
- Les équipes ne sont associées à aucune université en particulier. Cependant, pour des questions d'assurance, chaque université doit aussi gérer les étudiants qui utilisent ses locaux. Pour le cyclisme , c'est Dauphine qui en a la charge.
- Les relations globales sont fragmentées et réparties sur les différents sites.

Exemple : Sport universitaire
Une solution

- a=[Jussieu, Sorbonne, Dauphine] i dans {1 ; 2; 3}
- $Locaux_i = \sigma_{univ= ai}(Locaux)$
- $Horaire_i = Horaire \ltimes Locaux_i$

Fragmentation mixte pour les profs :

- $ProfPaye_i = \pi_{idProf, nom, adresse, salaire}(\sigma_{affectation = ai}(Prof))$
- $ProfContact_i = \pi_{idProf, nom, tel}(Prof \ltimes Horaire_i)$

Etudiant fragmenté par université et on rajoute les étudiant qui utilisent les équipements

- $Etu_i = (\sigma_{affectation = ai}(Etudiant)) \cup (Etudiant \ltimes (Horaire \ltimes Locaux_i)$
- Pour i dans {1 ; 2} on a $Etudiant_i = Etu_i$
- Pour i=3, il faut rajouter $EtuCycle = Etu \ltimes (\sigma_{sport = "cyclisme"} Equipe)$
- $Etudiant_3 = Etu_3 \cup EtuCycle$

- Equipe est répliquée partout (petite et très rarement mise à jour)

Données réparties avec Oracle : Database link

Lien à une table dans une BD distante spécifié par :

- nom de lien
- nom de l'utilisateur et password
- Infos de connexion (protocole client-serveur d'oracle)

Exemple de syntaxe :

```
create database link Site2
connect to E1234  identified by "E1234" using 'ora10';
```

Create synonym Emp2 for Emp@Site2;

Ou

Create view Emp2 as select * from Emp@Site2;

Conclusions et perspectives

Applications classiques

- décisionnel (data warehouse)
- transactionnel

Applications à l'échelle du web

  - grand nombre de sources
  - hétérogénéité très forte
- intégration des données semistructurées
- intégration de la recherche documentaire
- intégration de services Web (ex. agence de voyage)