

Master d'Informatique

BDR - 4I803 – COURS 5

Conception de BD réparties
Fragmentation et réplication

2017

Bases de Données Réparties

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

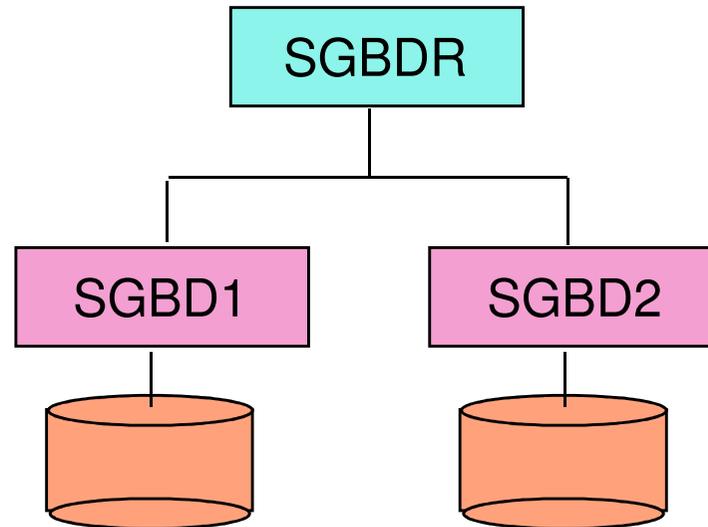
BD réparties (1)

- 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 (ordre croissant d'intégration) :
 1. Client/serveur : BD centralisée, seuls certains traitements (interface, p.ex.) sont locaux.
 2. Accès distant (Remote Data Access)
 3. Vues réparties : extension du mécanisme de vues pour définir des vues sur plusieurs sites.
 4. médiateurs
 5. BD réparties/fédérées

BD Réparties (2)

- BD réparties :
 - Plusieurs bases 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

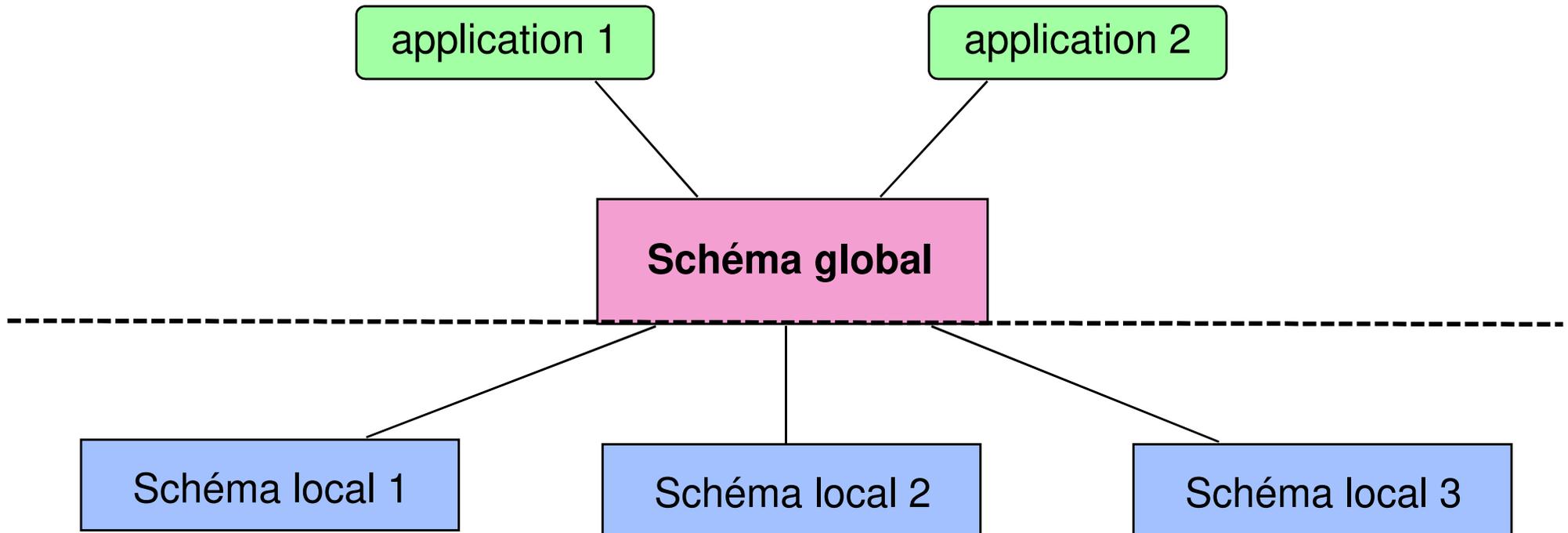
Paramètres à considérer

- 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

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



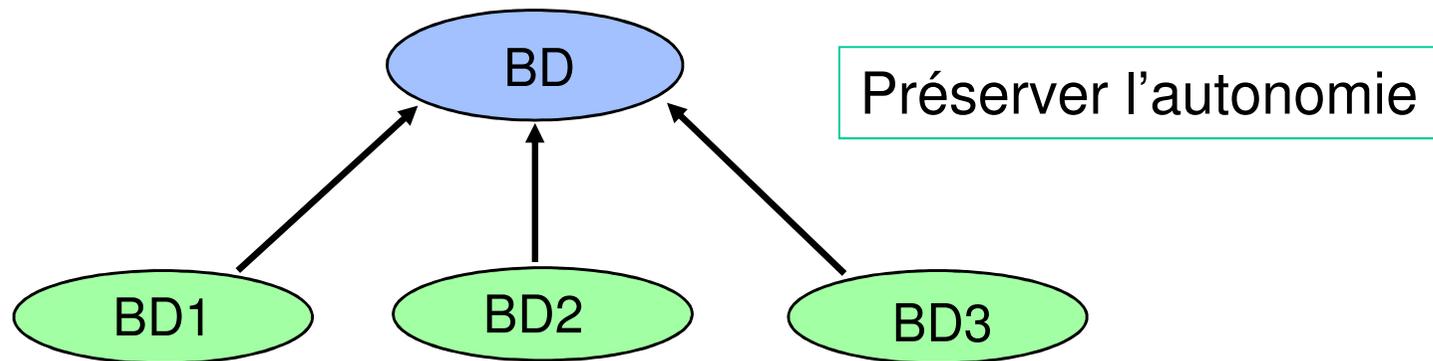
- indépendance applications / bases locales
- schéma global lourd à gérer

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)

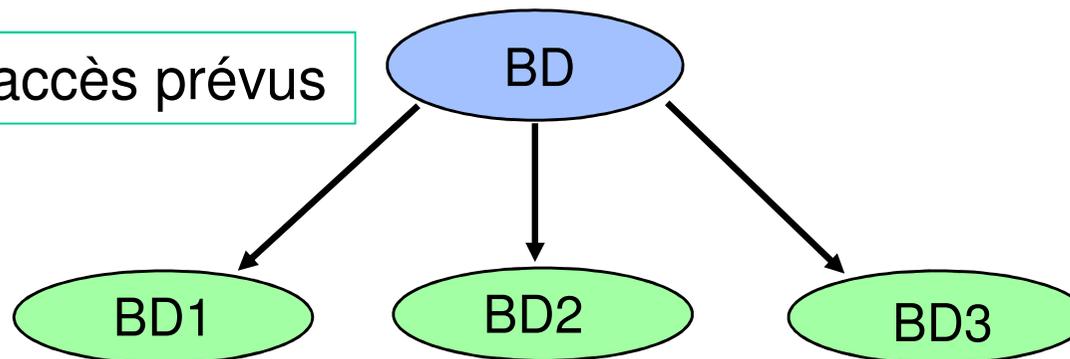
Migration vers une BDR : 2 approches

Intégration logique des BD locales existantes (fédérée, médiateur)

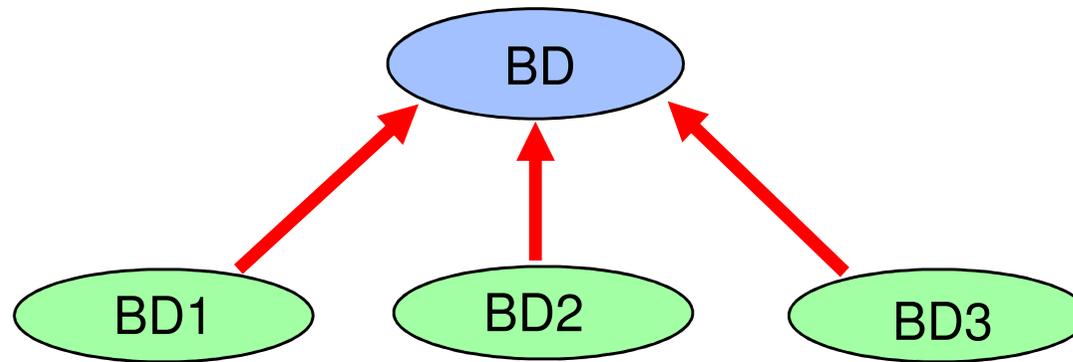


Décomposition en BD locales : répartie « pur »

En fonction des accès prévus

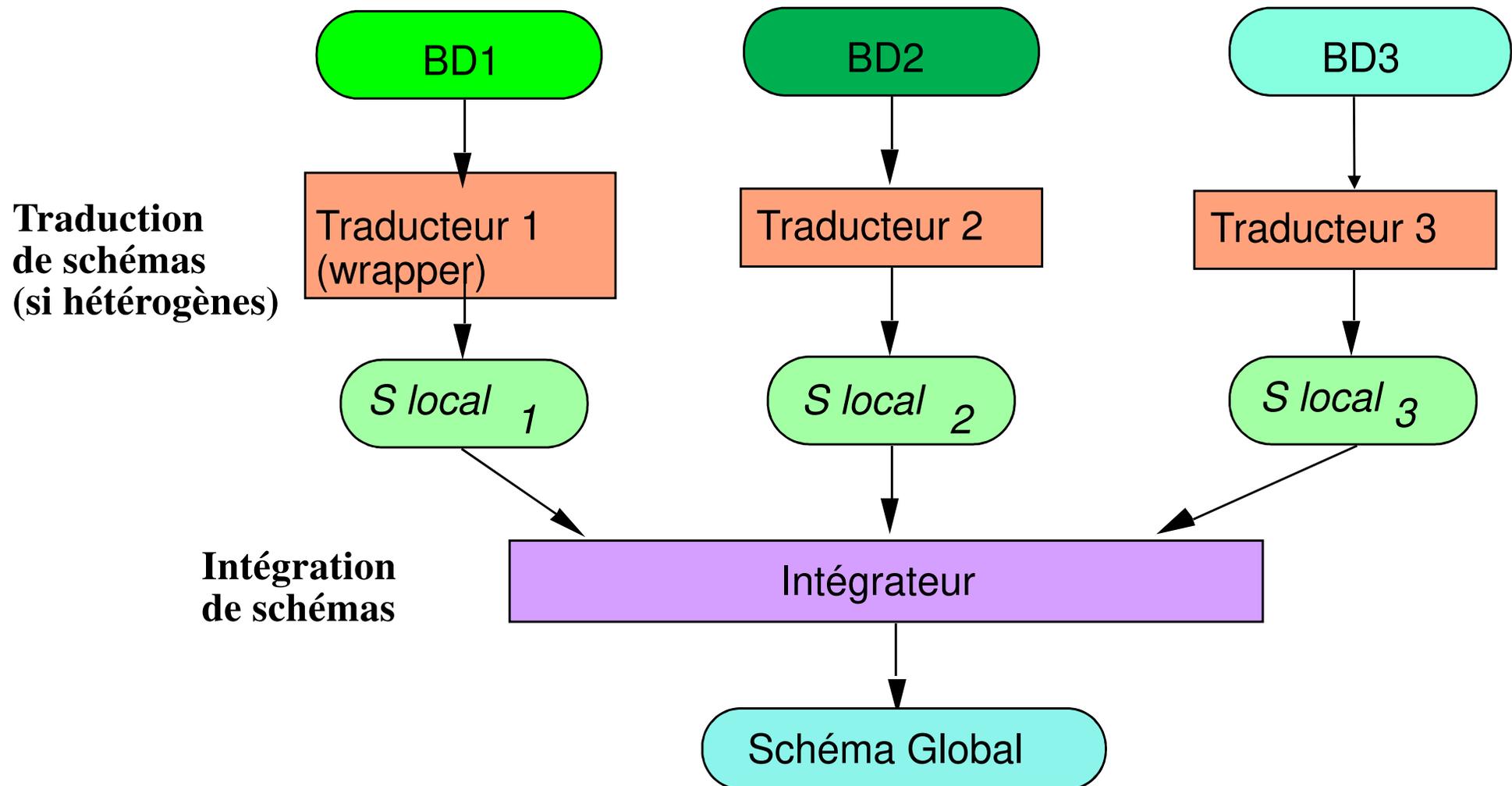


Intégration de BD existantes



Conception d'une BDR par intégration

Approche médiateur



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

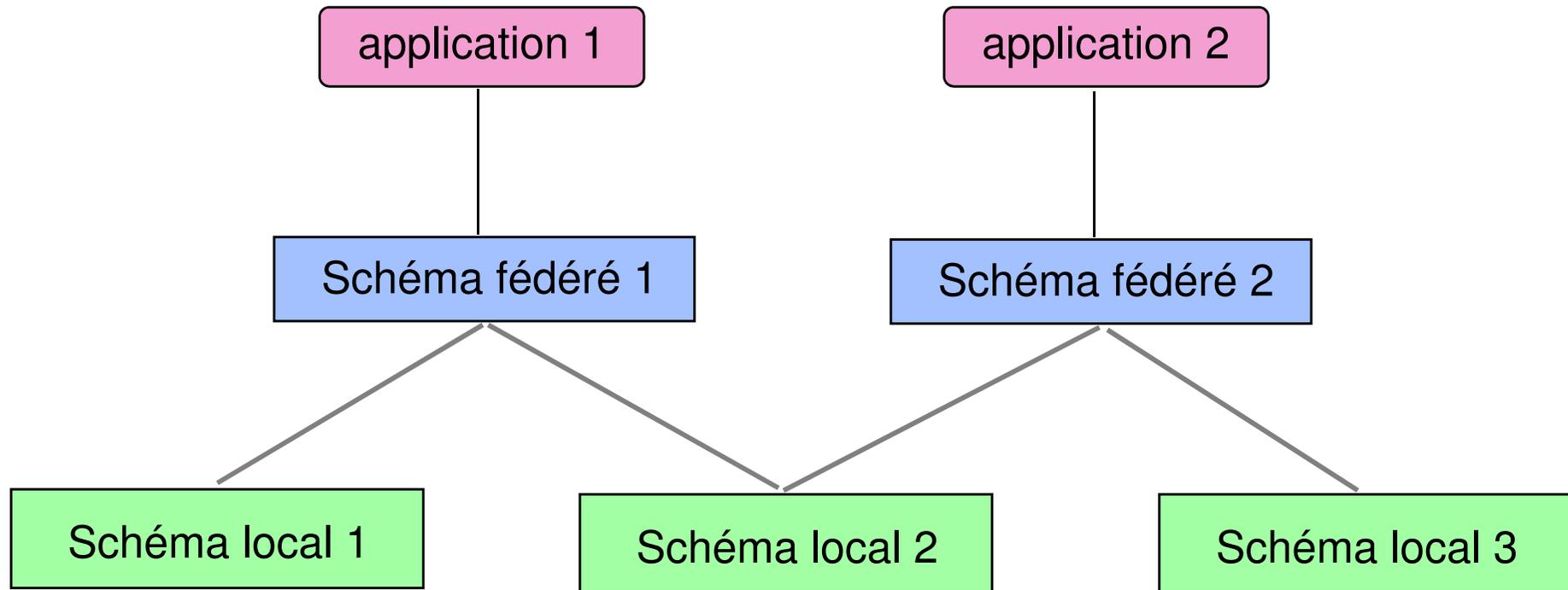
2. comparaison

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

3. conformance

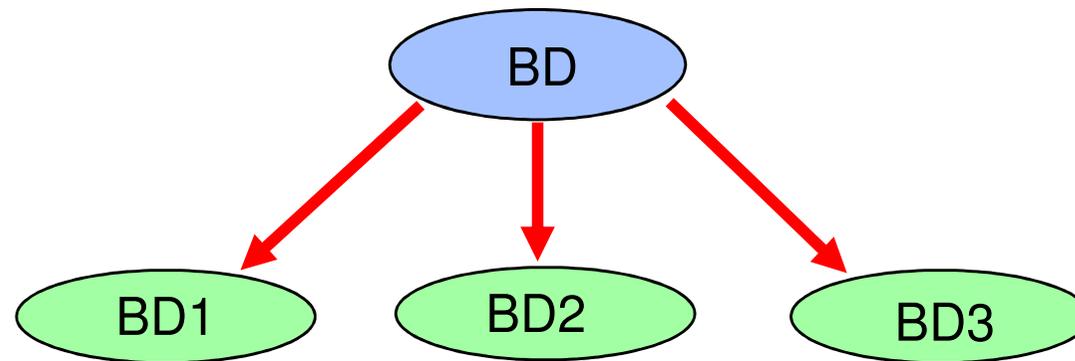
- *résolution des* conflits de noms (renommage) et des conflits structurels (changements de clés, tables d'équivalence)
- Définition de règles de traduction entre le schéma intégré et les schémas initiaux.

Architecture fédérée

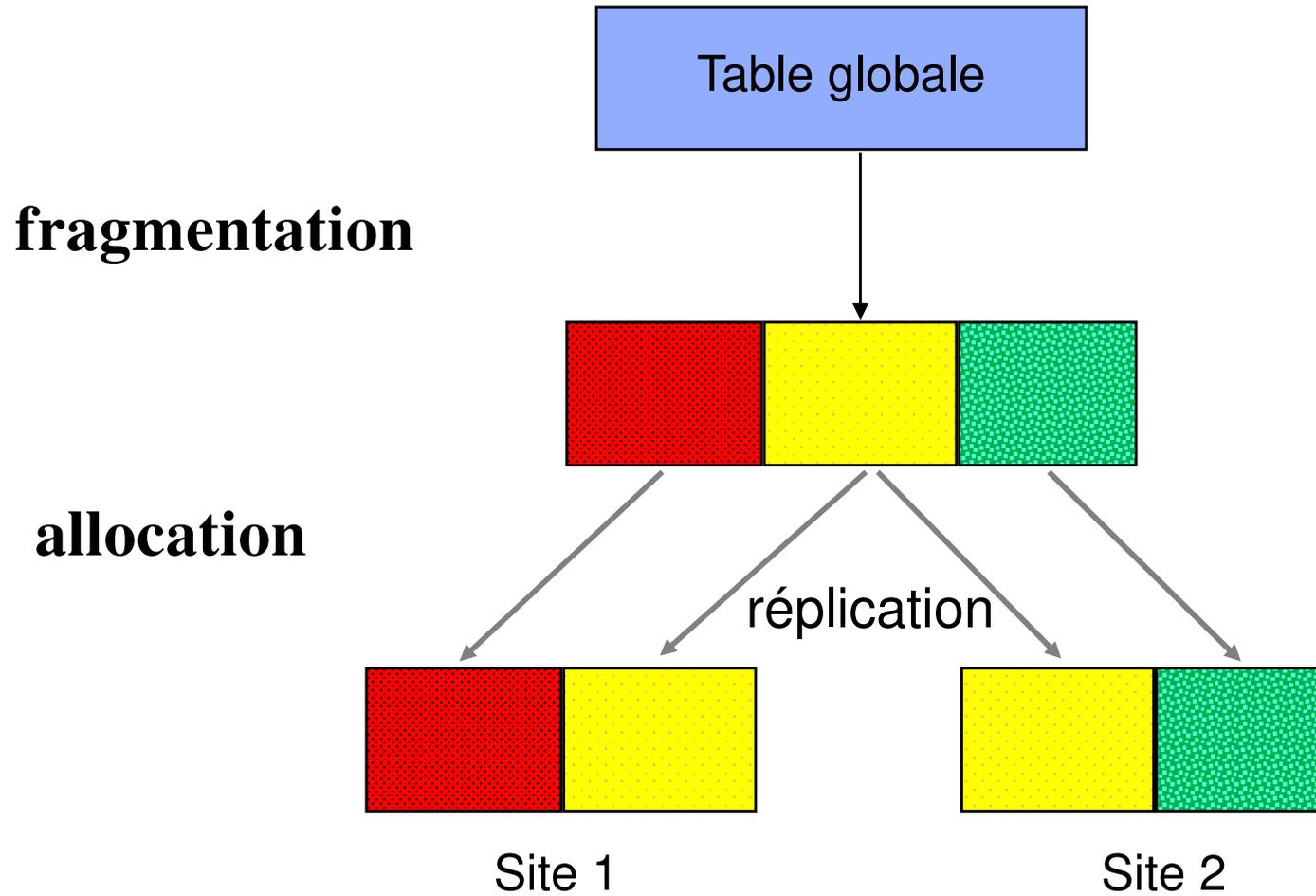


moyen contrôlé de migration

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

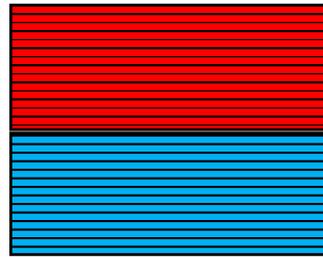
Duplication (ou réplique)

- favoriser les accès locaux
- augmenter la disponibilité des données

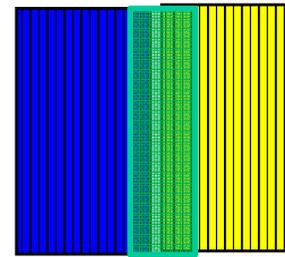
Trop répliquer : surcoût de maintenir cohérence des répliques

Types de Fragmentation

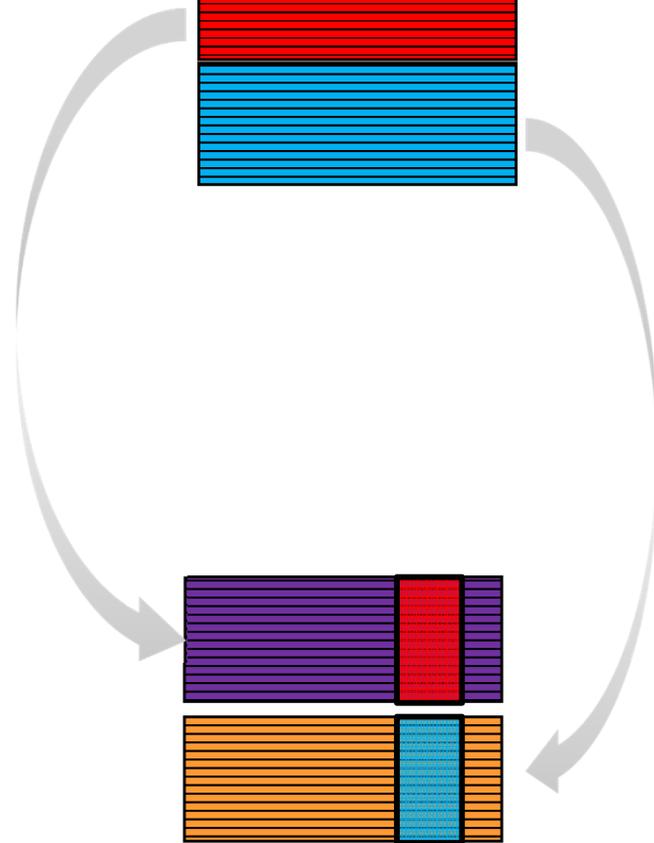
horizontale



verticale

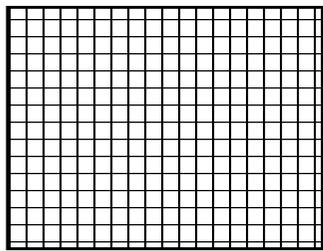


Clé = attribut commun

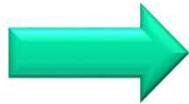


horizontale dérivée

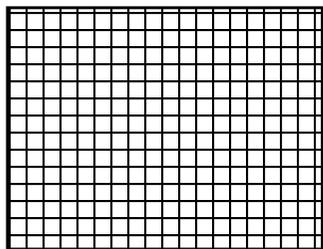
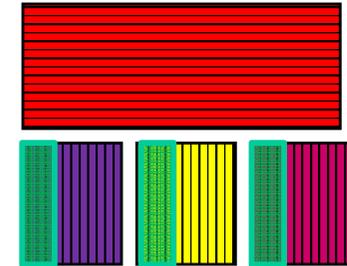
Fragmentation Mixte



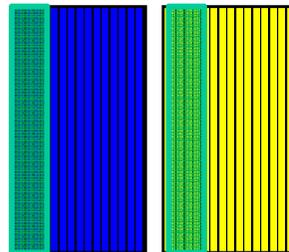
horizontale



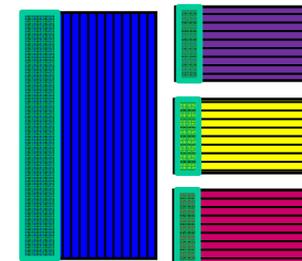
verticale



verticale



horizontale



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)

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 par sélection

- Fragmentation de R selon n prédicats
 - les prédicats $\{p_1, \dots, p_n\}$ ex: $\{a < 10, a > 5, b = 'x', b = 'y'\}$
- L'ensemble M des prédicats de fragmentation est :
 - $M = \{ m \mid m = \bigwedge_{1 \leq k \leq n} p_k^* \}$ avec $p_k^* \in \{p_k, \neg p_k\}$
 - Eliminer les m de sélectivité nulle ex: $a > 10 \wedge a < 5$
 - Simplifier: $a < 10 \wedge a \leq 5 \wedge b = 'x' \wedge b \neq 'y'$ devient $a \leq 5 \wedge b = 'x'$
- Construire les fragments $\{R_1, \dots, R_k\}$
 - 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
```

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

Reconstruction par **union**

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

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

Propriétés de la fragmentation horizontale dérivée

R: fragmentation horizontale \rightarrow fragments R_i

S: fragmentation horizontale dérivée \rightarrow fragments $S_i = S \bowtie_A R_i$

- **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 $i \neq j$, $S_i \cap S_j = (S \bowtie R_i) \cap (S \bowtie R_j) = S \bowtie (R_i \cap R_j) = \emptyset$
 - Rappel: $R_1 \cap R_2 \Leftrightarrow R_1 \bowtie R_2$
- **Reconstructible**
 - $\bigcup_i S_i = (S \bowtie R_1) \cup (S \bowtie R_2) \cup \dots \cup (S \bowtie R_n) = S \bowtie (\bigcup_i R_i) = S$
- \Rightarrow contrainte d'intégrité référentielle
 - A = 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**

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

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

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**

$$Cde = Cde1 \bowtie Cde2$$

En SQL :

create view Cde as

select * from Cde₁, Cde₂

where Cde₁.ncde = Cde₂.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

a_{ij} = affinité de A_i avec A_j

ex: nb de requêtes qui accèdent A_i et A_j

	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

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	Dupont	Paris
C 3	Martin	Paris

Client2

nclient	nom	ville
C 2	Martin	Lyon
C 4	Smith	Lille

Cde1 = Cde >< Client1

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

Site 1

Cde2

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

Site 2

Exemple

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 entraînement. 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**.

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;



Vue
répartie

Outils d'interface SGBD

Passerelle



Répliqueur



Extracteur



Passerelles

- Fonctions
 - définition des procédures de transformation (dictionnaire) et exécution dans l'environnement cible
 - conversion de formats et de valeurs
 - filtrage et fusion de fichiers ou de tables
 - données calculées et résumés

La réplication

Plan:

Objectifs

Fonctions

Modèles d'appartenance

- fixe, dynamique ou partagé

Détection des modifications

Objectifs de la réplication

- + Accès simplifié, plus performant pour les lectures
 - + Résistance aux pannes
 - + Parallélisme accru
 - + Evite des transferts
-
- Overhead en mise à jour
 - Cohérence des données
 - Toujours bien si on privilégie les lectures et/ou si peu de conflit entre mäj

Objectifs de la réplication

Problème : comment partager des données entre p sites ?

- Solution 1 : sans duplication
 - stockage sur un site et accès réseau depuis les autres sites
 - problèmes de performances et de disponibilité
- Solution 2 : duplication synchrone
 - propagation des mises à jour d'un site vers les autres par une transaction multi-site avec validation 2PC ou communication de groupe
 - problèmes liés au 2PC : bloquant et cher. Communication de groupe seulement pour LAN
- Solution 3 : réplication asynchrone
 - Non bloquant mais uniquement cohérence à terme
 - Contrôle de la fraîcheur, traitement spécifique des requêtes read-only

Aussi mono-maître, multi-maîtres

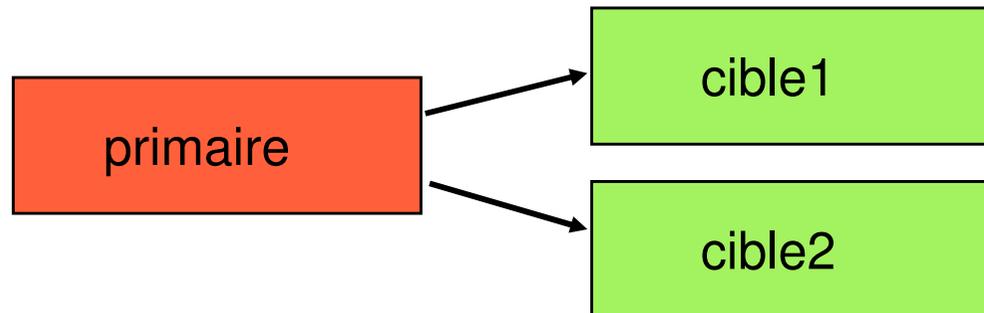
Fonctions d'un répliqueur

- Définition des objets répliqués
 - table cible = sous-ensemble horizontal et/ou vertical d'une ou plusieurs tables
- Définition de la fréquence de rafraîchissement
 - immédiat (après mise à jour des tables primaires)
 - à intervalles réguliers (heure, jour, etc.)
 - à partir d'un événement produit par l'application
- Rafraîchissement
 - complet ou partiel (propagation des modifications)
 - push (primaire -> cibles) ou pull (cible -> primaire)

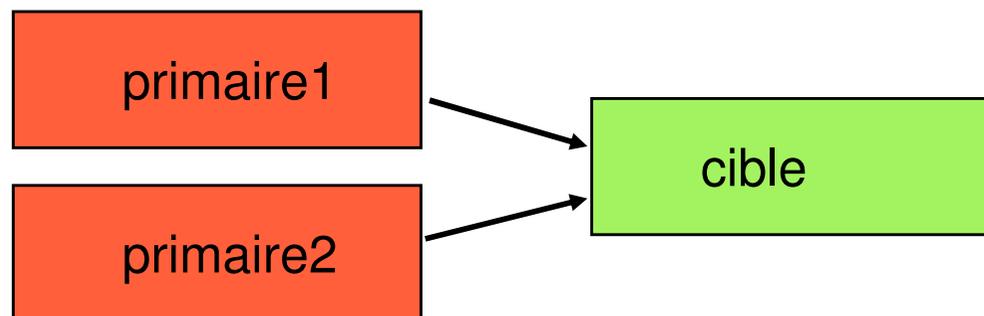
Réplication Monomaître

- Seul le **site primaire** peut recevoir les transactions des applications
 - insert, update, delete
- les **sites cibles** ne reçoivent que des requêtes en lecture seule
 - select

Diffusion



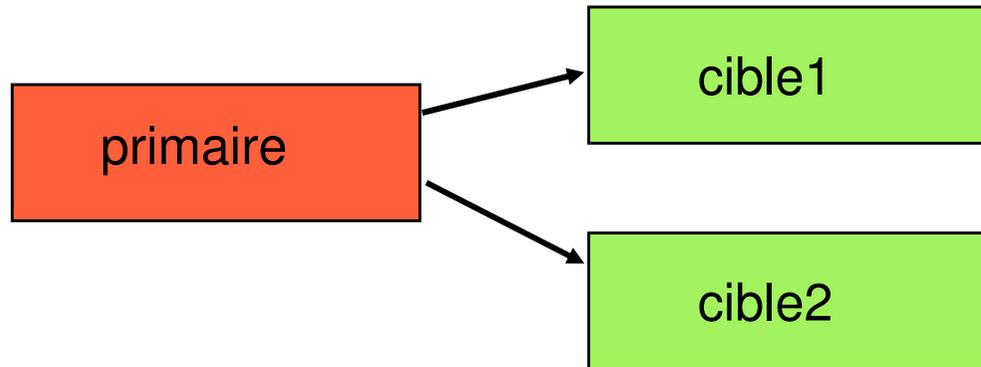
Consolidation



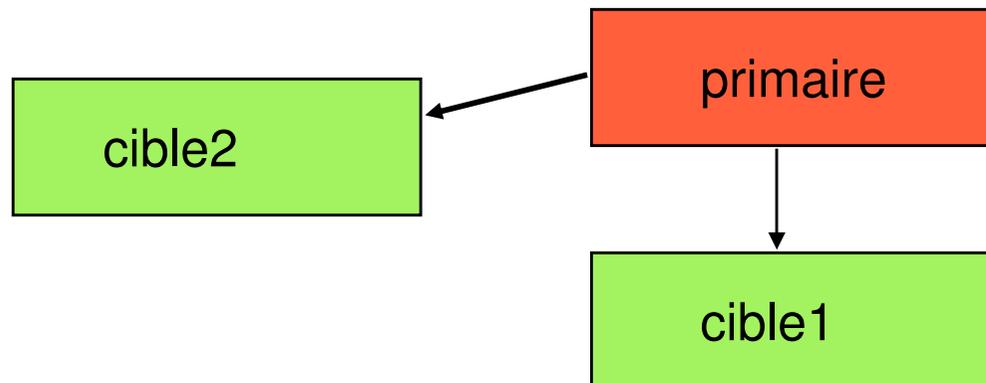
Monomaitre avec appartenance dynamique

Le site **primaire** peut être différent au cours du temps, en fonction d'événements: panne d'un site, état de la données, etc.

Appartenance à l'instant t1



Appartenance à l'instant t2



Propagation des mises à jour

- Propagation
 - gérée par le SGBD réparti
 - Synchronisée, ou non, avec la transaction

Propagation synchrone

- Avant la validation de la transaction
- L'application obtient une réponse **APRES** la propagation
 - 1) transaction locale → 2) propagation → 3) validation → 4) réponse
- Propagation : nb de messages échangés entre sites
 - Immédiate après chaque opération (L,E)
 - 1 message par opération (L/E): interaction linéaire
 - Différée juste avant la fin de la transaction
 - 1 message par transaction : interaction constante
 - Contenu du message: SQL ou Log
- Validation: décision prise
 - par plusieurs sites (vote, ex 2PC)
 - par chaque site séparément (sans vote)

Propagation asynchrone

- L'application obtient une réponse **AVANT** la propagation

– 1) transaction locale → 2) validation locale → 3) réponse

– 4) propagation → 5) validation

Etapes (1,2,3) indépendantes de (4,5)

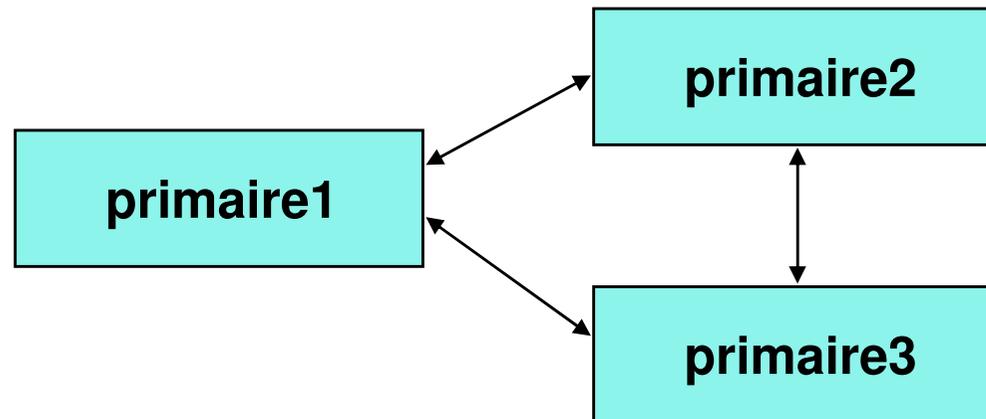
Réplication multi-maîtres

- Plusieurs maîtres pour une donnée (R1, R2)
- Mises à jour sur R1 et R2
- Pb: propagation des mise à jour:
 - Les mises à jour de R sont réparties sur **plusieurs** maîtres R1, ... Rn
 - Une réplique (r) doit recevoir toutes les mises à jour reçues par **les maîtres**

Réplication Multimaîtres

Une donnée appartient à plusieurs sites, qui peuvent chacun mettre à jour et diffuser aux autres sites

- **augmente la disponibilité**
- **Optimiste : peut produire des conflits, qui doivent être détectés et résolus**
- **Préventif : éviter les conflits**



Classification des solutions

- Monomaître / Multimaîtres
- Asynchrone / Synchrone
 - Synchrone : avec interaction constante / linéaire
 - Asynchrone : optimiste / pessimiste
- Validation: avec vote / sans vote
- Rmq
 - Primary copy = Monomaître
 - Update Everywhere = Multi-maître
 - Eager Replication = Repl. avec propagation synchrone
 - Lazy Replication = Repl. avec propagation asynchrone

Détection des modifications

- Solution 1 : utilisation du journal
 - les transactions qui modifient écrivent une marque spéciale dans le journal
 - détection périodique en lisant le journal, indépendamment de la transaction qui a modifié
 - modification de la gestion du journal
- Solution 2 : utilisation de triggers
 - la modification d'une donnée répliquée déclenche un trigger
 - mécanisme général et extensible
 - la détection fait partie de la transaction et la ralentit
- Solution 3: détecter les conflits potentiels (a priori)
 - Parser le code (pas possible pour transactions interactives)
 - Graphe d'ordonnancement global

Réplication totale vs. partielle

- Evaluation de requêtes, gestion du répertoire (catalogue)
 - Le plus facile est réplication totale
 - Réplication partielle ou fragmentation: difficulté. équiv.
- Contrôle de concurrence
 - Fragmentation > répli. totale > répli. Partielle (+difficile)
- Fiabilité
 - Répli. totale > répli. partielle >> fragmentation
- Réalisme : le plus réaliste est réplication partielle

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 semiestructurées
- intégration de la recherche documentaire
- intégration de services Web (ex. agence de voyage)

Complément



Exercice 2016

- **Personne** (idpers, nom, prenom, age, couleurcheveux, tél)
- **Salon** (idsalon, marque, ville, surface, jourdefermeture)
- **Coiffeur** (idpers, idsalon, niveau) travaille dans un seul salon
- **Client** (idpers, coupefavorite)
- **RDV** (idcoiffeur, idclient, jour, heure, coupe) rv passés et futurs
- **Tarif** (coupe, marque, prixbase)
- **Promo** (heure, coef) ex: 10% à 14h

Enoncé

- Chaque marque possède un site informatique dans chaque ville
 - stocker les données nécessaires aux salons de cette marque dans la ville. Fragments selon la marque et la ville.

Personne_{m,v} stockées sur le site S_{m,v}

Salon_{m,v} Coiffeur_{m,v} Client_{m,v} Rdv_{m,v} Tarif_{m,v} Promo_{m,v}

Solution

- $\text{Personne}_{m,v} = (\text{Personne} \times \text{Client}_{m,v}) \cup (\text{Personne} \times \text{Coiffeur}_{m,v})$
- $\text{Salon}_{m,v} = \sigma_{\text{marque}=m \wedge \text{ville}=v} \text{Salon}$
- $\text{Coiffeur}_{m,v} = (\text{Coiffeur} \times \sigma_{\text{ville}=v} \text{Salon})$
- $\text{Client}_{m,v} = \text{Client} \times \text{RDV}_{m,v}$
- $\text{RDV}_{m,v} = \text{RDV} \times (\text{Coiffeur} \times \text{Salon}_{m,v})$
- $\text{Tarif}_{m,v} = \sigma_{\text{marque}=m} \text{Tarif}$
- $\text{Promo}_{m,v} = \text{Promo}$