Gestion de données distribuées

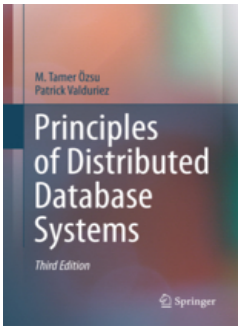
Patrick Valduriez
Inria, Montpellier

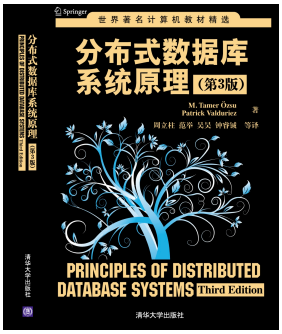
Plan du cours

1. Architecture de SGBD distribué
2. Conception par décomposition
3. Intégration de données hétérogènes
4. Fédérateurs de données
5. SGBD transactionnels
6. Analyse du théorème CAP

Bibliographie

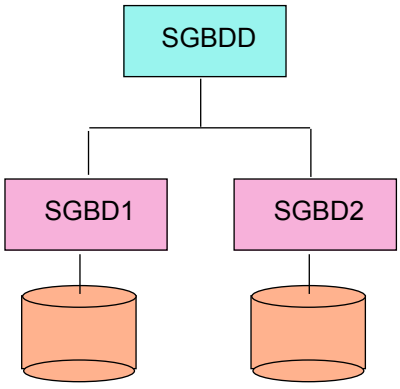
Principles of Distributed Database Systems
Tamer Özsu & Patrick Valduriez
Springer, 850 pages, 2011.



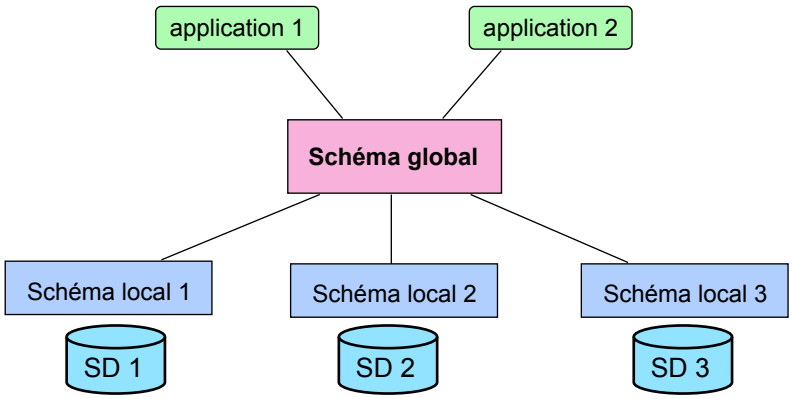


1. Architecture de SGBD distribué

- Rend la distribution transparente
  - catalogue des BD
  - admin. et sécurité
  - traitement des requêtes distribuées
  - gestion de transactions distribuées
  - réplication des données



Schémas



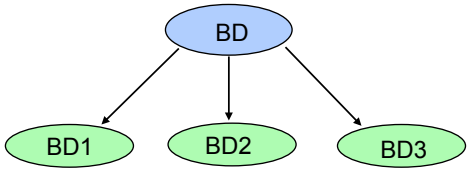
- Indépendance des applications aux sources
- Les sources vues comme une BD virtuelle

Schéma global

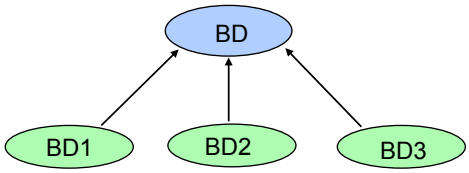
- Ensemble de vues
  - Donne la description globale et unifiée de toutes les sources de données (ex. des relations globales)
- Exemple
  - Offres (emploi, ville, date) =
    Offres@site1 U Offres@site2
  - Demandeurs (nom, emploi) =
    Demandeurs@site2 U Demandeurs@site3

Migration vers une BDR

Décomposition en BD locales: **SGBDD homogène**

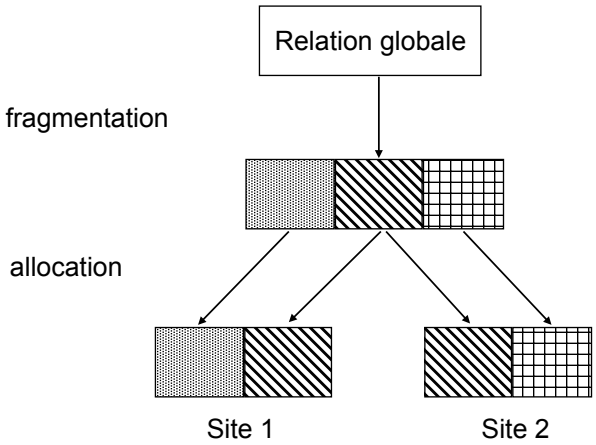


Intégration logique des BD existantes: **SGBDD hétérogène**



P. Valduriez - 7

2. Conception d'une BDR par décomposition

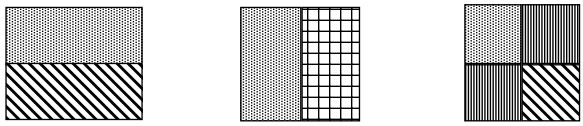


P. Valduriez - 8

Objectifs de la décomposition

- Fragmentation
  - performances en favorisant les accès locaux
  - équilibrer la charge de travail entre les sites
- Réplication
  - favoriser les accès locaux
  - augmenter la disponibilité des données

Types de fragmentation



horizontale verticale 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
- Disjointe
  - chaque élément de R ne doit pas être dupliqué

Fragmentation Horizontale

Fragments définis par sélection
- Client1 = Client where ville = Paris
- Client2 = Client where ville not= Paris

Reconstruction

Client =Client1 U Client2

Client

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

Client1

nclient	nom	ville
C 1 C 3	Dupont Martin	Paris Paris

Client2

nclient	nom	ville
C 2 C 4	Martin Smith	Lyon Lille

Fragmentation Horizontale Dérivée

Fragments définis par jointure
- Cde1 = Cde where
  Cde.nclient = Client1.nclient
- Cde2 = Cde where
  Cde.nclient = Client2.nclient

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

Cde = Cde1 U Cde2

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 Verticale

Fragments définis par projection
- Cde1 = Cde (ncde, nclient)
- Cde2 = Cde (ncde, produit, qté)

Reconstruction

Cde = [ncde, nclient, produit, qté]
where Cde1.ncde = Cde2.ncde

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
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

Allocation des fragments aux sites

- Non-répliquée
  - partitionnée : chaque fragment réside sur un seul site
- Répliquée
  - chaque fragment sur un ou plusieurs sites
  - maintien de la cohérence des copies multiples
  - Règle :
    - si 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

Trouver la distribution "optimale" de *F* sur *S*
- Optimum
  - coût minimal de com., 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

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

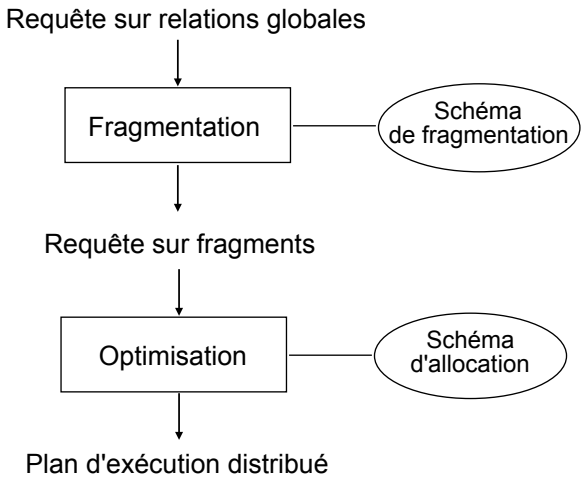
Site 1

Cde2

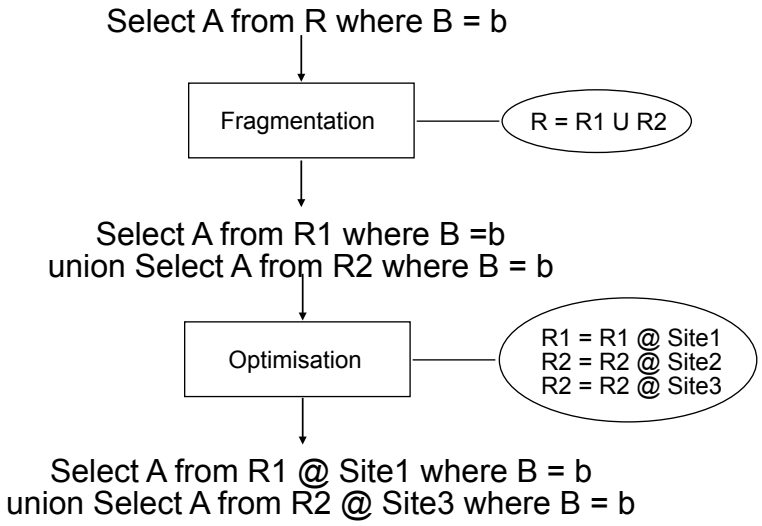
ncde	client	produit	qté
D 3 D 4	C 2 C 4	P 3 P 4	5 10

Site 2

3. Evaluation de requêtes distribuées



Exemple d'évaluation simple

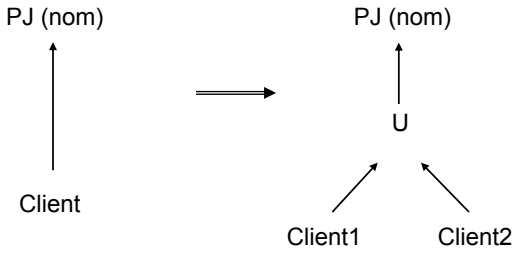


Fragmentation de requête

- Réécriture
  - mettre la requête sous forme d'un arbre algébrique (feuille = relation, noeud = op. relationnel)
- Reconstruction
  - remplacer chaque feuille par le programme de reconstruction de la relation globale
- Transformation
  - appliquer des techniques de réduction pour éliminer les opérations inutiles
- Notations utilisées :
  - SL : select
  - JN : join
  - PJ : project

Reconstruction

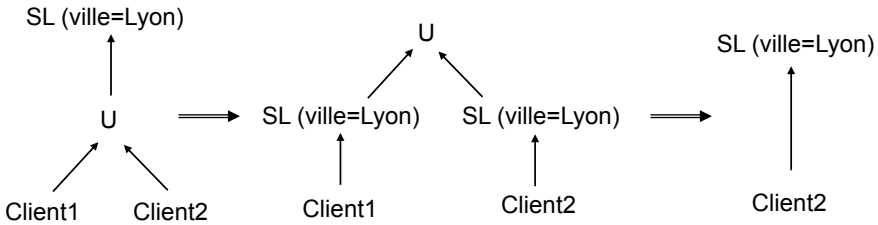
Select nom from Client
Client1 = Client where ville = Paris
Client2 = Client where ville not= Paris



Réduction pour fragmentation horizontale

- Règle: éliminer l'accès aux fragments inutiles

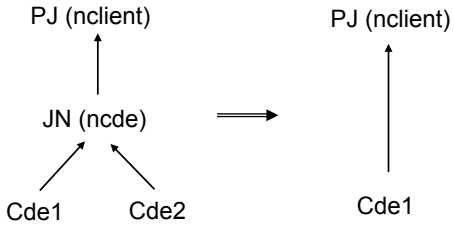
Client1 = Client where ville = Paris
Client2 = Client where ville not= Paris
Select * from Client where ville=Lyon



Réduction pour fragmentation verticale

- Règle : éliminer les accès aux relations de base qui n'ont pas d'attributs utiles pour le résultat final

Cde1 = Cde (ncode, nclient)
Cde2 = Cde (ncode, produit, qté)
Select nclient from Cde



Réduction pour fragm. hor. dérivée

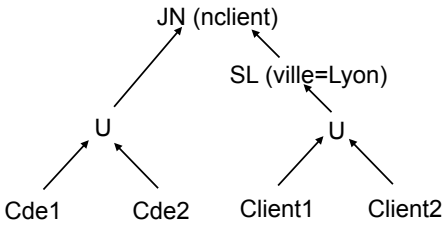
- Règle : distribuer les jointures par rapport aux unions et appliquer les réductions pour la fragmentation horizontale

Client1 = Client where ville=Paris
Client2 = Client where ville not=Paris
Cde1 = Cde where Cde.nclient=Client1.nclient
Cde2 = Cde where Cde.nclient=Client2.nclient

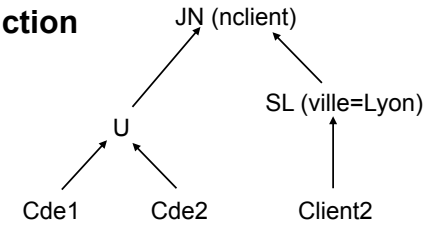
Select all from Client, Cde
where Client.nclient = Cde.nclient and ville=Lyon

Exemple

Requête canonique

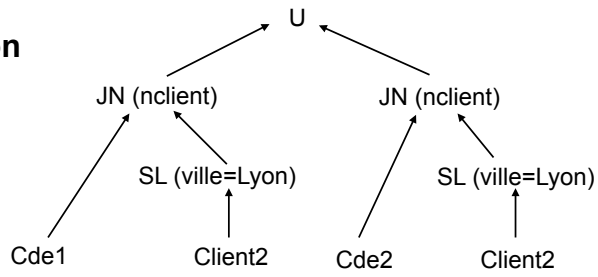


Après 1ère réduction

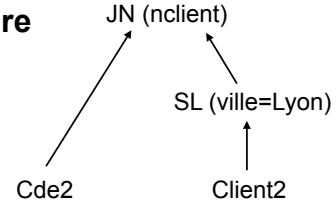


Exemple (suite)

Distribution



Elimination du sous-arbre inutile



Optimisation de requêtes distribuées

- Entrée : une requête simplifiée exprimée sur des fragments
- Sortie : un plan d'exécution optimal

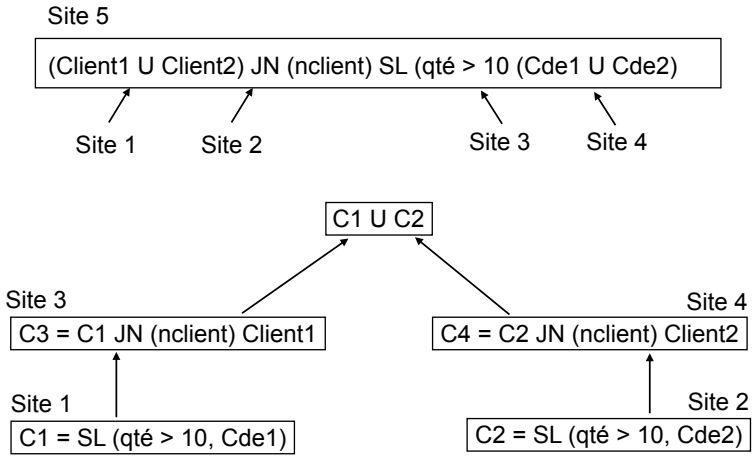
- Objectifs
  - choisir la meilleure localisation des fragments
  - minimiser une fonction de coût: *f* (I/O, CPU, com.)
  - exploiter le parallélisme
  - exprimer les transferts inter-sites
- Solution
  - examiner le coût de chaque plan possible (par transformation) et prendre le meilleur

Exemple

- Site 1 : Client1 = Client where ville = Paris
- Site 2 : Client2 = Client where ville not= Paris
- Site 3 : Cde1 = Cde where Cde.nclient = Client1.nclient
- Site 4 : Cde2 = Cde where Cde.nclient = Client2.nclient
- Site 5 : résultat

```
Select all from Client, Cde
where Client.nclient = Cde.nclient
and qté > 10
```

Choix de la solution



Coût des Solutions

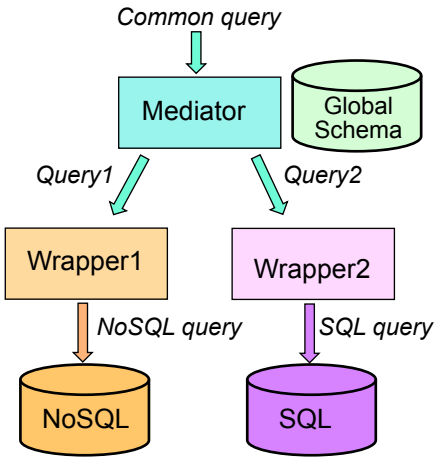
- Supposons
  - taille (Cde1) = taille (Cde2) = 10 000
  - taille (Client1) = taille (Client2) = 2 000
  - coût de transfert de 1n-uplet = 1
- Stratégie 1
  - transfert de Cde1 + Cde2 = 20 000
  - transfert de Client1 + Client2 = 4000
- Stratégie 2
  - transfert de C1 + C2 = 200
  - transfert de C3 + C4 = 200

3. Intégration des données hétérogènes

1. Architecture médiateur - adaptateur
2. Intégration de schémas
3. Modèles de données pour l'intégration

3.1. Architecture médiateur - adaptateur

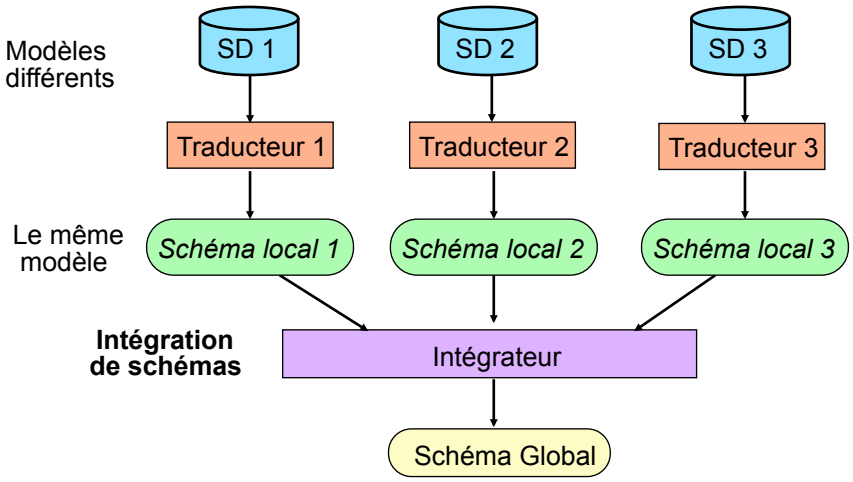
- Médiateur (mediator)
  - Centralise l'information fournie par les adaptateurs dans un schéma global
  - Transforme les requêtes exprimées dans un langage commun en requêtes pour les adaptateurs
  - Intègre les résultats des requêtes
- Adaptateur (wrapper)
  - Exporte les informations sur la source de données
    - Schéma, fonctionnalité
  - Transforme les requêtes exprimées dans le langage commun en requêtes pour les sources
  - Transforme les résultats de requêtes dans le modèle de données commmun



Les deux variantes principales

- Médiateurs (ou fédérateurs) de données
  - Pour des sources de données Web
    - Documents non structurés, structurés (XML, JSON, ..)
    - Des milliers de sources
  - En lecture seule
    - Pas de transactions
- SGBD transactionnels
  - Pour des sources de données d'entreprise
    - Des dizaines de sources
  - Lecture et mise à jour
    - Transactions

3.2. Intégration de schémas

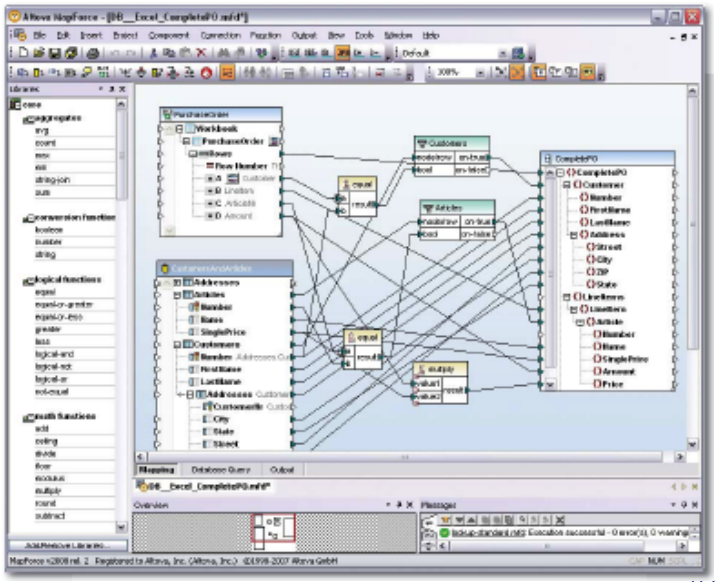


Intégration de schémas: étapes

1. Pré-intégration
   - *identification* des éléments reliés et établissement des règles de conversion
     - ex. 1 pouce = 2,54 cm, 1€=x$
2. Comparaison
   - *identification* des conflits de noms (ex. synonymes, homonymes) et des conflits structurels (ex. types, clés)
3. Mise en conformité
   - *résolution des* conflits (changements de types, de clés, etc.)
4. Création du schéma intégré
   - *fusion* des schémas intermédiaires

- Outils pour l'intégration
  - Dictionnaires, thesaurus, ontologies
  - Outils de mapping, ex. MapForce

MapForce d'Altova



3.3. Modèles de données pour l'intégration

- Modèle relationnel
  - Structures de données simples et régulières
- Modèle objet
  - Structures de données complexes et régulières
- Modèle semi-structuré (XML, JSON)
  - Structures de données complexes et irrégulières
  - Schéma (DTD, Xshema) optionnel

Intégration en relationnel

Emp = Emp@Site1 U Emp@Site2

prenom	nom	ville	tel.
null Anne null Jean	P. Dupont Martin A. Martin Smith	Paris Nantes Nantes Lille	0140... null 0235… null

Emp@Site1

nom	ville	tel.
P. Dupont A. Martin	Paris Nantes	0140... 0235...

Emp@Site2

prenom	nomF	ville
Anne Jean	Martin Smith	Nantes Lille

- Renommage et introduction de valeurs nulles

Interrogation en relationnel (SQL)

```
select prenom, nom, tel.
from Emp
where ville = "Nantes"
```

prenom	nom	ville	tel.
null Anne null Jean	P. Dupont Martin A. Martin Smith	Paris Nantes Nantes Lille	0140... null 0235… null

⇨

prenom	nom	tel.
Anne null	Martin A. Martin	null 0235…

Intégration en XML

```
<!DOCTYPE db [
<!ELEMENT db (Emp)*>
<!ELEMENT emp (nom,ville,(tel.)?)>
<!ELEMENT nom (#PCDATA | (prenom, nomF))>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT nomF (#PCDATA)>
<!ELEMENT tel. (#PCDATA)> ]>
```

```
<emp>
<nom> A. Martin </nom>
<ville> Nantes </ville>
<tel.> 0235... </tel.>
</emp>
```

```
<emp>
<nom>
  <prenom> Anne </prenom>
  <nomF> Martin </nomF>
</nom>
<ville> Nantes </ville>
</emp>
```

- Admet la différence d'éléments

Interrogation en Xquery

```
for $e in document(db)/emp
where $e/ville = "Nantes"
return <r>     <nom> $e/nom </nom>
               <tel.> $e/tel </tel>
       </r>
```

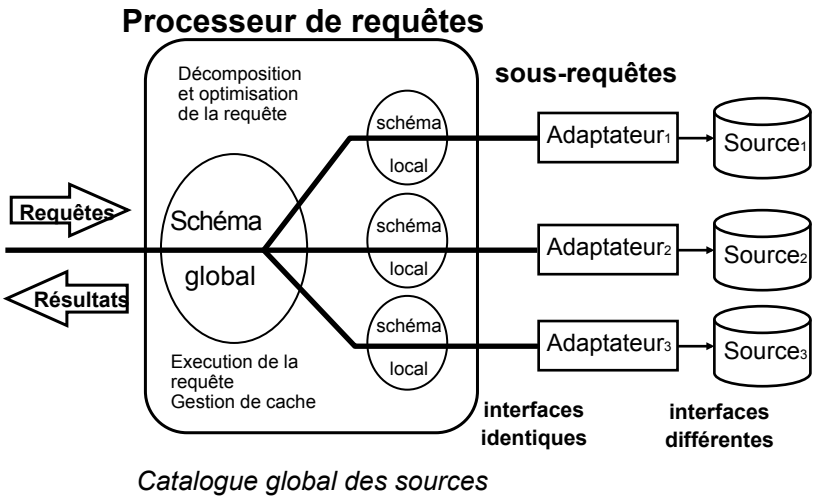
```
<r>
<nom> A. Martin </nom>
<tel.> 0235... </tel.>
</r>
```

```
<r>
<nom>
  <prenom> Anne </prenom>
  <nomF> Martin </nomF>
</nom>
</emp>
```


4. Fédérateurs de données

1. Traitement et optimisation de requêtes
2. Produits principaux

4.1. Traitement de requêtes



Catalogue global des sources

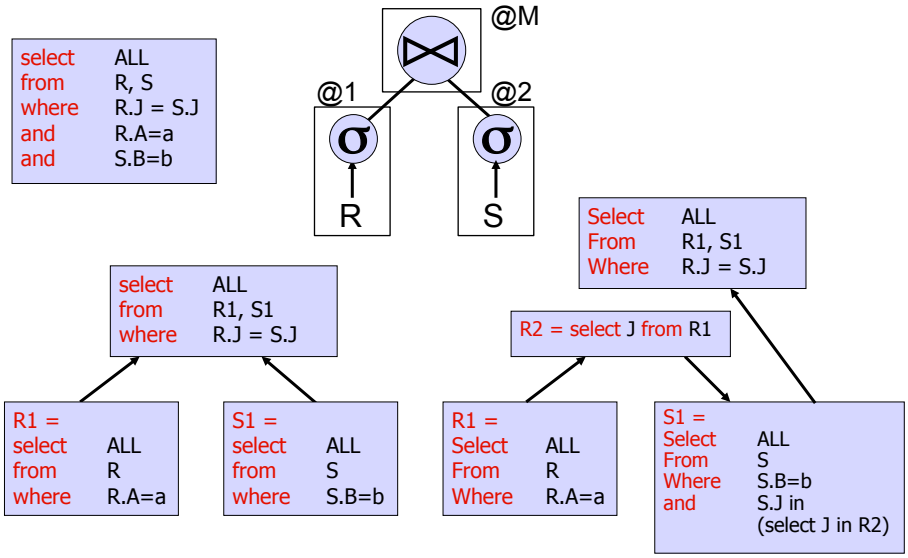
Adaptateurs

- Interfaces standards
  - ODBC, JDBC, ADO .NET
- Editeurs de SGBD
  - Passerelles (gateways) entre le SGBD et les sources
  - Oracle, IBM, Microsoft, SAP, etc.
- Fournisseurs indépendants
  - Extracteurs (ETL) entre sources et outils cibles
  - Information Builders, Evolutionary Technology Inc. (ETI), Talend (Open Source), etc.
- Médiateurs de données
  - Web services et API REST
  - Boite à outils pour adaptateurs

Optimisation de requête distribuées

- Comme pour SGBDD homogène
  - Espace de recherche, modèle de coût
  - Mais avec localisation des traitements au niveau
    - Source de données
    - Médiateur
  - Et de nouvelles transformations pour réduire les transferts de données
    - Ex. join => bindjoin

Exemple avec bindjoin

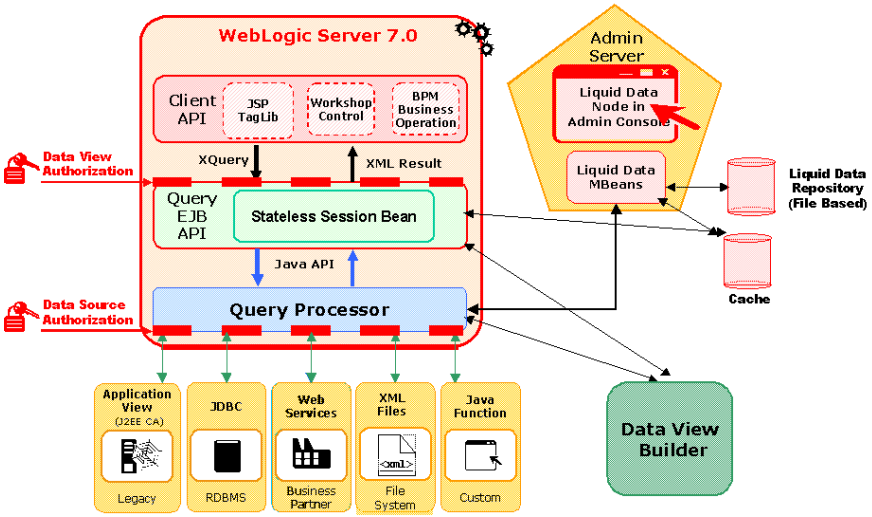


4.2. Produits principaux

Editeur	Produit	Remarques
Denodo	Denodo Platform	Intégration SQL et XML
Actuate	BIRT iHub	XML, rachat de la startup Nimble
Google	Fusion table	Web service pour visualiser des sources web
IBM	Infoshere Information Integrator	Hybride relationnel/XML avec SQL/XML
Microsoft	ADO .NET	Intégration SQL et XML
Oracle	Data Service Integrator	Issu de BEA Liquid Data
SAP/BO	Data Federator	Relationnel, avec API XML Rachat de la startup Médience (INRIA)
Talend	Data Services	Logiciel libre
Xquare	Xquare Bridge, Xquare Fusion	Logiciel libre XML issu de Xlive (Univ. Versailles)

Exemple de médiateur XML

- BEA Liquid Data (aujourd'hui Oracle Data Service Integrator)



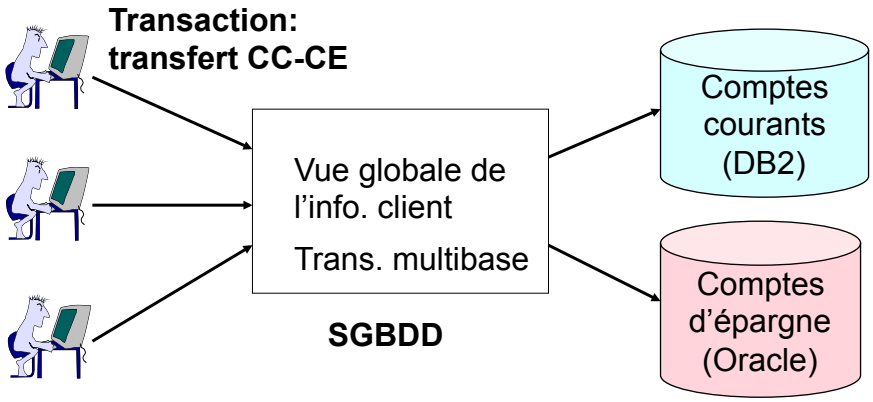
Etude de cas: Kelkoo

- Médiateur avec modèle relationnel et SQL simple
  - Basé sur le logiciel *DISCO (Distributed Information Search Component)* développé par Bull et INRIA
  - Modèle d'intégration relationnel
    - Requêtes SQL simples avec mots-clés
  - Classification des sources de données
    - Livres, voyages, vins, etc.
    - Permet de passer à l'échelle
  - Boîte à outils pour adaptateurs
    - Besoin de modifier un adaptateur quand l'interface de la source change
  - Gestion d'un cache sur disque au niveau du médiateur
    - Accès immédiat aux données très demandées

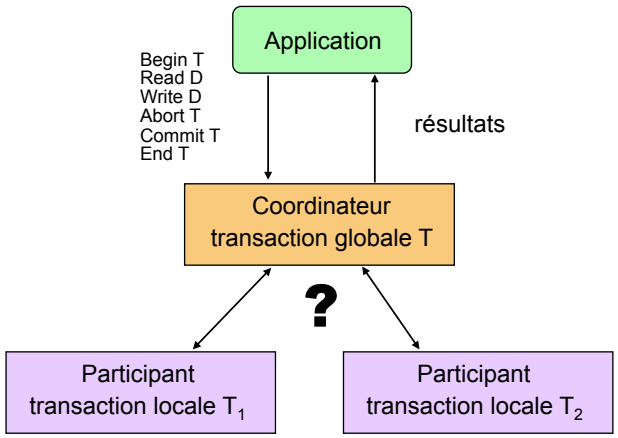
5. SGBD transactionnels

1. Transactions distribuées
   - Protocole de validation 2PC
2. Réplication des données
   - Etude de cas
3. Principaux produits

5.1. Transaction distribuée



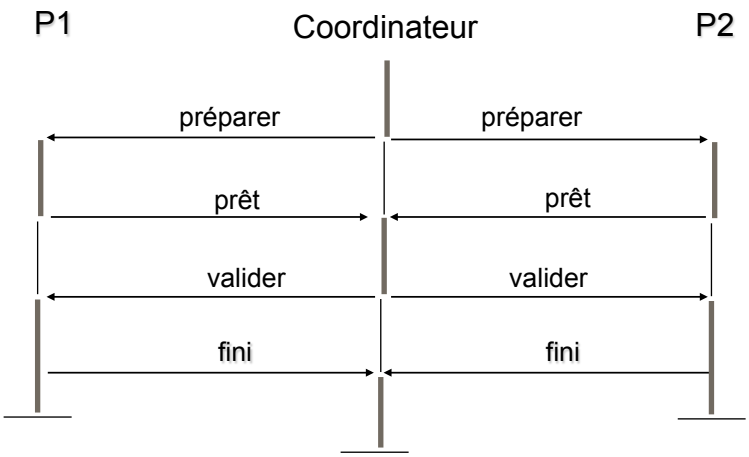
Architecture



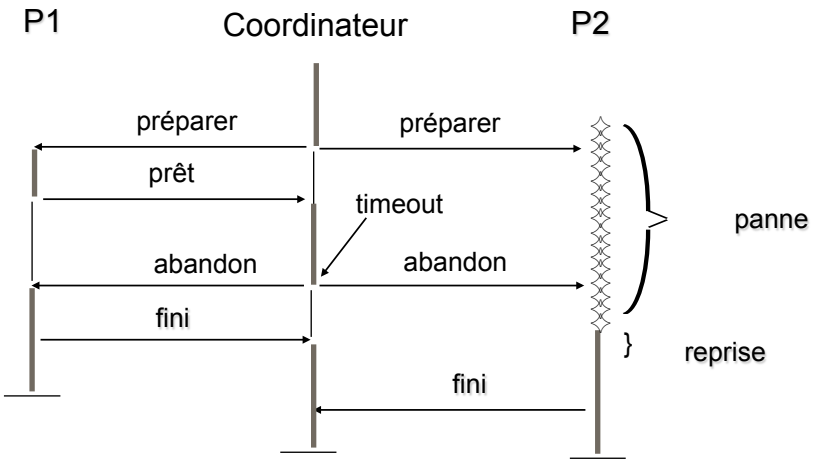
Le 2PC (2 phase commit)

- Garantit l'atomicité de la transaction (tout ou rien)
  - Le coordinateur applique le protocole
  - Chaque participant obéit et répond au coordinateur
  - Coordinateur et participants enregistrent l'état de la trans. dans leur journal
- Phase 1 : préparation
  - Déclenchement par le coordinateur
  - Préparation des participants à écrire dans la BD
    - Vérification des contraintes d'intégrité
    - Ecriture dans le journal
  - Envoi du vote ok (ou non ok) au coordinateur
- Phase 2 : décision globale du coordinateur
  - Si tous ok, COMMIT; sinon ABORT

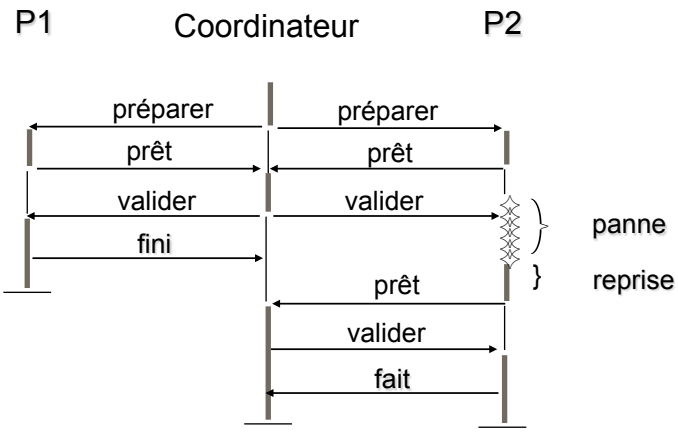
Validation normale



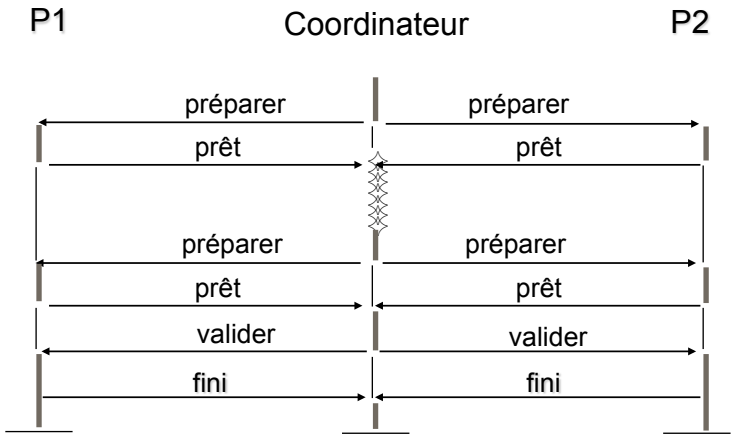
Panne d'un participant avant d'être prêt



Panne d'un participant après être prêt



Panne du coordinateur



Critique du 2PC

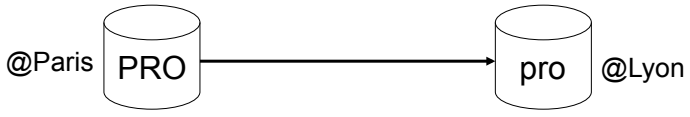
- Avantages
  - standard
    - DTP de l'OpenGroup (TX et XA)
    - OTS de l'OMG, JTS de Sun
  - robuste: résiste à tout type de panne
- Inconvénients
  - cher en communications et écritures forcées dans les journaux
  - bloquant en cas de panne du coordinateur
    - alternative = réplication

5.2. Réplication de données

1. Objectifs
2. Modes de réplication
3. Etude de cas: magasins franchisés

Objectifs de la réplication

- Duplication des données distribuées partagées
  - depuis un site primaire vers des sites secondaires
  - *augmente les performances et la disponibilité*
- Fonction de base d'un SGBDD



Définition de la réplication

- Définition des objets répliqués
  - Copie secondaire (en lecture seule) = vue horizontale et/ou verticale d 'une ou p tables
  - Exemple

    CREATE SNAPSHOT Emp AS
    SELECT * FROM EMP@hq.com WHERE ...
- Définition du rafraichissement
  - Quand? périodiquement, immédiat, événement, etc
  - Comment? *complet* ou *partiel* (snapshot log)
  - Quels sites? Group refresh
  - Quel mode?
    - Push: le site primaire diffuse aux sites secondaires
    - Pull: le site secondaire demande aux sites primaires

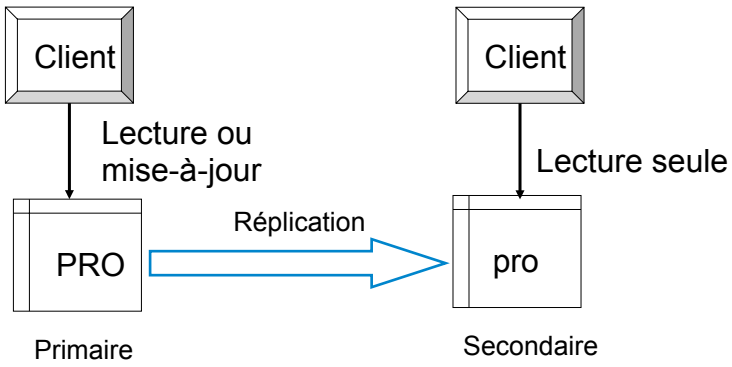
Réplication synchrone

- Lorsqu'une transaction met à jour une copie primaire (ex: EMP), toutes ses copies secondaires (ex: Emp) sont mises à jour dans la même transaction (en 2PC)
  - copies primaire et secondaires toujours égales
  - problèmes liés au 2PC : bloquant et cher
  - dégradation des performances si beaucoup de copies secondaires

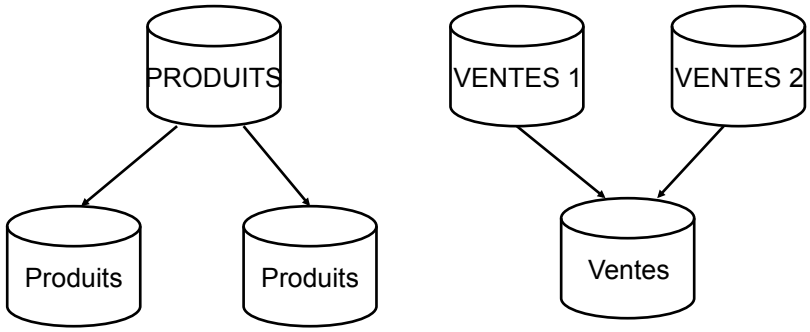
Réplication asynchrone

- Chaque transaction est validée sur le site primaire, puis chaque copie secondaire est mise à jour (rafraîchie) dans une transaction séparée
  - copie primaire et secondaires peuvent ne pas être égales
  - performant: on peut définir le bon moment pour rafraîchir (immédiat, périodique, etc.)
  - ne bloque pas en cas de panne de site

Réplication asymétrique

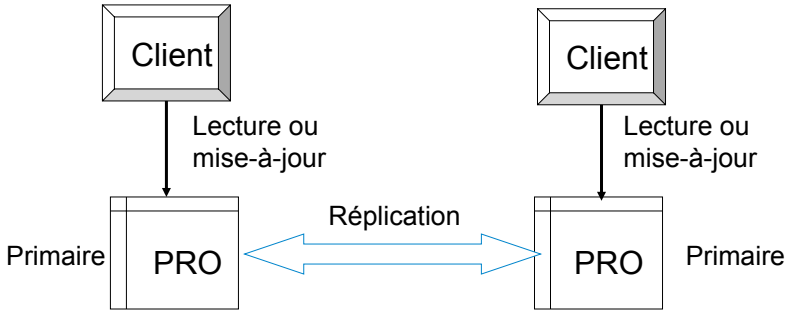


Exemple de réplication asymétrique



Diffusion **Consolidation**

Réplication symétrique (multi-maître)



Critique

- Augmente la disponibilité
- En asynchrone, peut produire des conflits => détection et résolution

Etude de cas: magasins franchisés

- Application: consolidation des ventes journalières des magasins au niveau du site central
  - Des centaines de magasins
  - Contraintes
    - Remontée et intégration < qqs heures
    - Minimisation des coûts réseau et matériel
- Choix techniques
  - Une BD locale par magasin et une BD globale au siège
    - Autonomie des magasins, coûts réseaux réduits
  - Réplication en mode pull : le siège contrôle l'accès aux magasins pour la remontée des ventes
    - Réduction du coût réseau
    - Evite le goulot d'étranglement de la BD siège (en mode push)

5.3. SGBD transactionnels

- SGBD relationnels
  - Tous les grands produits ont une version distribuée
  - IBM DB2, Microsoft SQL Server, Oracle, Ingres (Actian Corp., anc. Ingres Corp), SAP Sybase
- Websphere Information Integrator (IBM)
  - Accès en lecture et mise-à-jour à des sources de données distribuées
    - Relationnelles: DB2, Informix, SQL Server, Oracle, Sybase, Teradata, sources ODBC, etc.
    - Contenus : Websphere, LotusNotes, moteurs de recherche Web, Services Web, XML, Excel, etc.

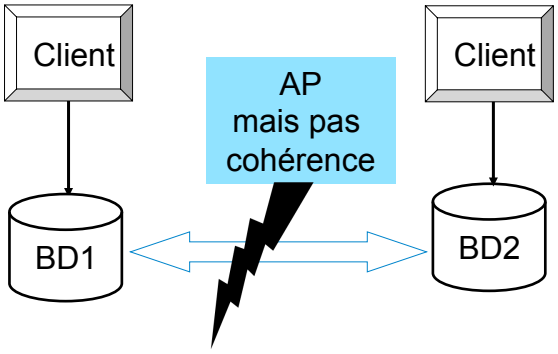
6. Analyse du théorème CAP

- Sujet polémique
  - "Une base de données ne peut fournir cohérence *et* disponibilité de service pendant une panne partielle du réseau"
- Deux points de vue différents
  - Bases de données (relationnelles)
    - Cohérence des données essentielle
      - Transactions ACID
  - Systèmes distribués
    - Disponibilité de service essentielle
      - Incohérence tolérée par l'utilisateur, ex. cache web
- Argument repris par le NoSQL
  - Relâchement de la cohérence au profit des autres propriétés

Le théorème CAP

- Les propriétés désirables d'un système distribué
  - ***C****onsistency* (cohérence): tous les nœuds voient les mêmes données au même moment
  - ***A****vailability* (disponibilité): toutes les requêtes reçoivent une réponse
  - ***P****artition tolerance* (résistance au morcellement): le système continue de fonctionner en cas de panne partielle
- Historique
  - A la conf. PODC 2000, Brewer (UC Berkeley) émet la conjecture qu'on ne peut avoir que deux propriétés en même temps
  - En 2002, Gilbert et Lynch (MIT) donnent une preuve de cette conjecture

Exemple avec réplication symétrique



Mais cohérence à terme (*eventual consistency*)

- Après reconnexion (et résolution des conflits), la cohérence peut être atteinte