

# Gestion de données réparties

1

## Architectures Réparties

- Motivation
- Définitions
- Conception
  - Fragmentation
  - Réplication
- Traitement des requêtes

2

## Motivations

- Limites des architectures centralisées :
  - Données des entreprises disséminées sur plusieurs sites (banques, sociétés, compagnies de transport,...)
  - Goulot d'étranglement (transfert de données + coûts de communications)
- Développement des réseaux
- Besoin de fédérer les bases de données

3

## Architectures Réparties

Avantages ..  
optimisation et autonomie  
solution naturelle à la gestion  
d'entreprises géographiquement réparties

4

## Approches

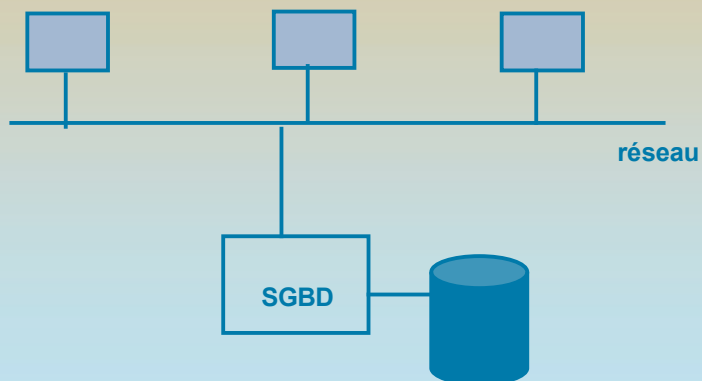
Diverses possibilités ..

Ne pas confondre ...

- Client/serveur : BD centralisée, seuls certains traitements (interface, p.ex.) sont locaux.
- Accès distant (Remote Data Access) : accès simultané à plusieurs bases de données locales
- Vues réparties : extension du mécanisme de vues pour définir des vues sur plusieurs sites.

5

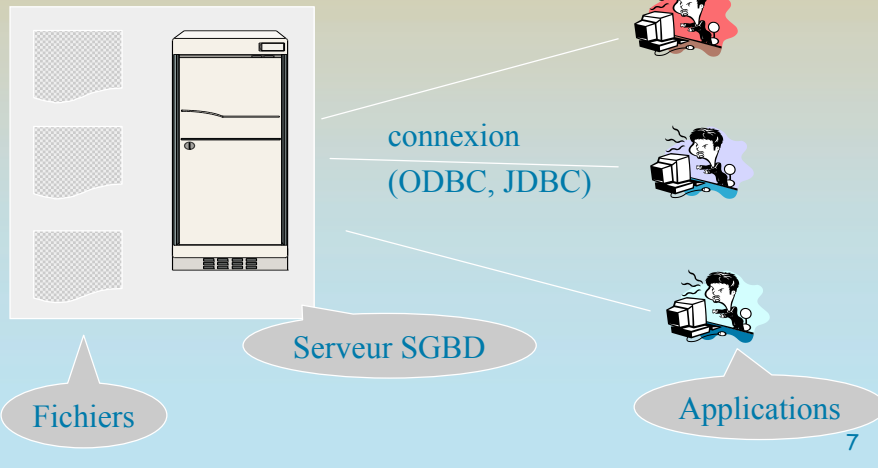
## Base de donnée distante



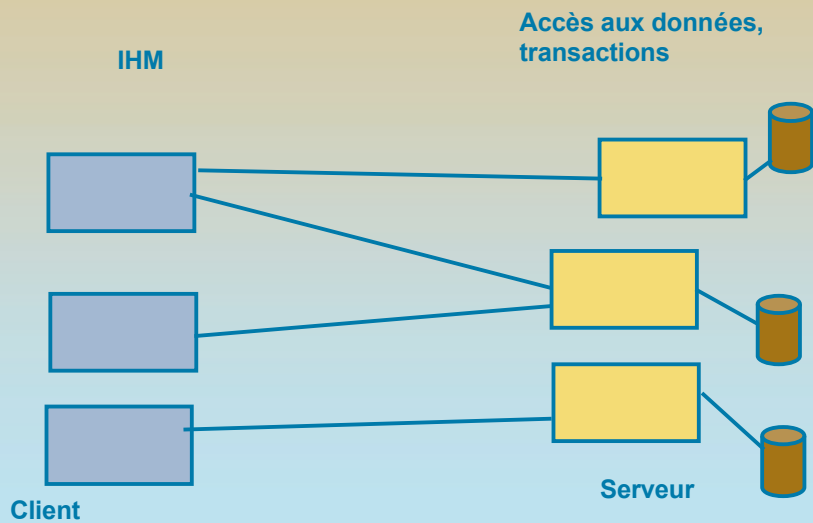
6

# Client-Serveur

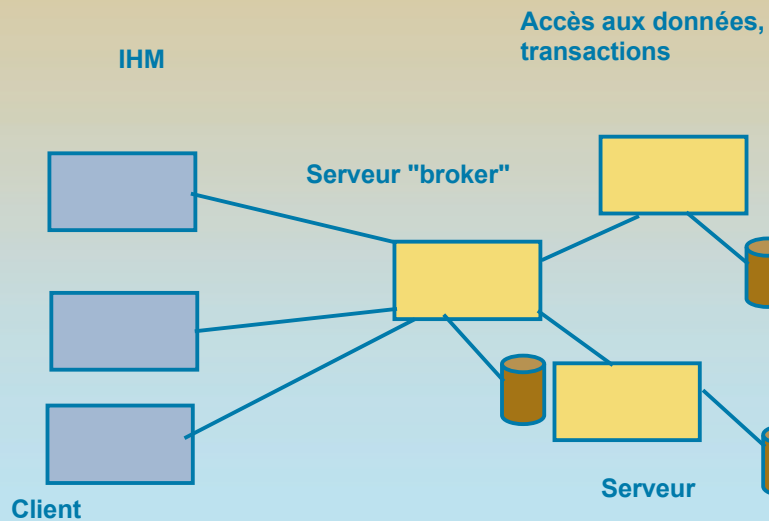
“2 tier system” ou “client-server”



# Client-Serveur

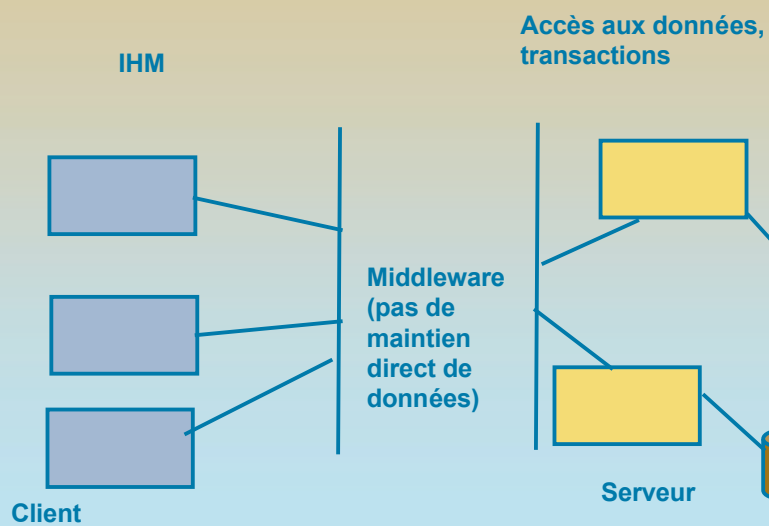


## Collaboration Server Systems MultiBase



9

## Middleware Systems



10

## Définitions

- Bases de données *réparties*
  - Plusieurs bases sur plusieurs sites, mais une seule BD « logique ».
  - Les ordinateurs (appelés sites) communiquent via le réseau et sont faiblement couplés.
  - Chaque site contient des données de la base, peut exécuter des transactions locales et participer à l'exécution de transactions globales
  - La répartition affecte les données, les traitements, les contrôles
  - La topologie des BD réparties est semblable à celles des réseaux : anneau, étoile, arbre, ...

11

## SGBD distribué

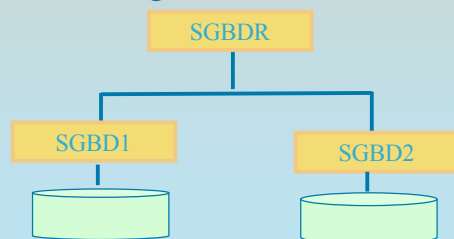
### Objectifs

- Indépendance à la localisation
- Indépendance à la fragmentation
- Indépendance à la duplication
- Indépendance aux SGBD
- *Autonomie des sites*

12

## SGBD distribué

- Même SGBD, plusieurs serveurs
  - Système distribué Homogène
- Divers SGBD
  - Système distribué Hétérogène



13

## SGBD distribué

Rend la distribution (ou répartition) des BD locales "transparente"

- catalogue des données réparties
- traitement des requêtes distribuées
- gestion de transactions distribuées
- maintien de la cohérence et de la sécurité

14

## Evaluation

### Paramètres à considérer

- Coût et temps de communication entre deux sites
- Fiabilité
  - fréquence des pannes des sites, du réseau
- Accessibilité aux données
  - accès aux données en cas de panne des sites, du réseau.
- Accès aux sites les moins encombrés, les plus puissants

15

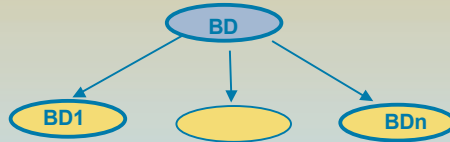
## Evaluation

- Avantages
  - extensibilité
  - partage des données hétérogènes et réparties
  - performances avec le parallélisme
  - disponibilité (tolérance aux pannes) avec la réplication
- Inconvénients
  - administration complexe
  - complexité de mise en oeuvre et de développement
  - distribution du contrôle
  - difficulté de migration
  - surcharge (l'échange de messages augmente le temps de calcul)

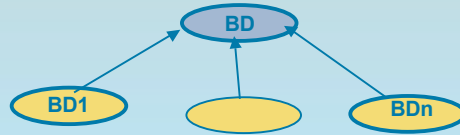
16

## Migration vers une BDR

- Décomposition physique en BD locales



- Intégration logique des BD locales existantes



17

## Architectures

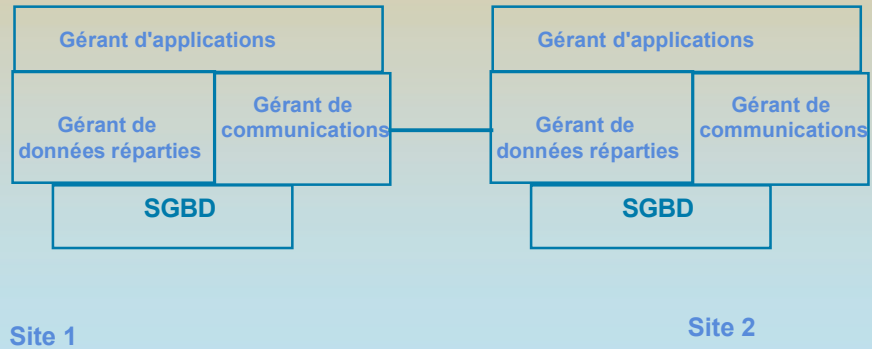
Diverses

Mais on présente

- une architecture de schémas
- une architecture fonctionnelle

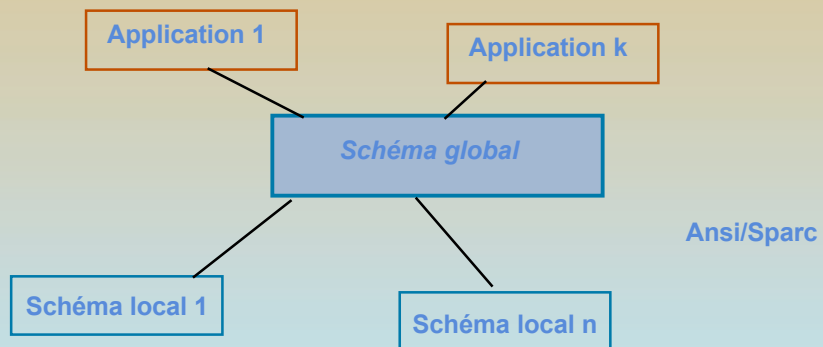
18

# Architectures



19

# Architecture de schémas



- Avantage : indépendance applications/BDR
- Inconvénient : schéma global à gérer

20

## Architecture de schémas

Schéma externe global 1

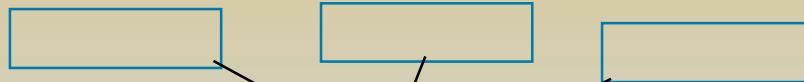


Schéma global

Schéma conceptuel global  
Schéma de placement

Schéma local 1

Schéma externe  
Schéma conceptuel  
Schéma Physique

Schéma local 2

Schéma externe  
Schéma conceptuel  
Schéma Physique

21

## Schéma global

### Schéma conceptuel global

Donne la description globale et unifiée de toutes les sources

Offres(emploi, ville, date), Demandes(nom, emploi)

*Indépendance à la répartition*

### Schéma de placement

Précise la façon dont les schémas sont placés dans les diverse sites

Règles de correspondance avec les données locales

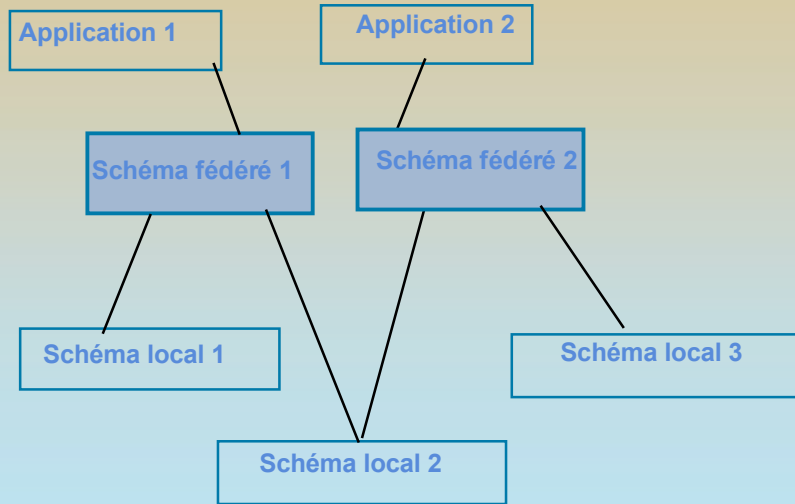
Offres= Offres@site1UOffres@site2

Demandes = Demandes@site1

*Indépendance à la localisation, la fragmentation, la duplication*

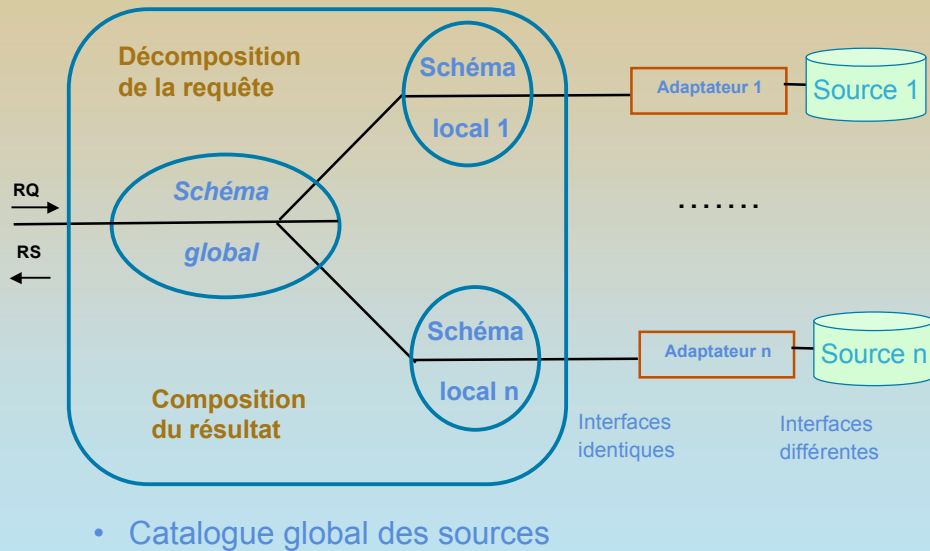
22

## Architecture fédérée



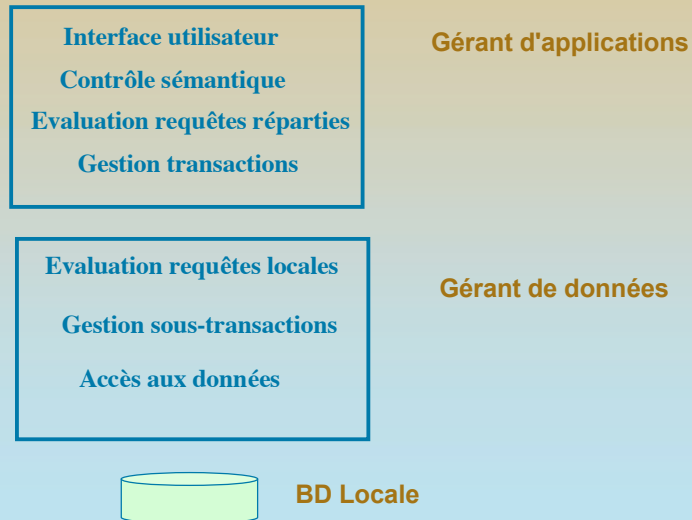
23

## Architecture



24

## Architecture fonctionnelle



25

## Architecture

- L'interface utilisateur reçoit la requête et restitue le résultat
- Contrôle sémantique des données
- Décomposition
- Gestion de transactions
  
- Evaluation des requêtes locales
- Gestion de sous transactions
- Accès aux données
  
- Si nécessaire conversion de requêtes (adaptateurs)

26

# Conception

Décomposition

Intégration

27

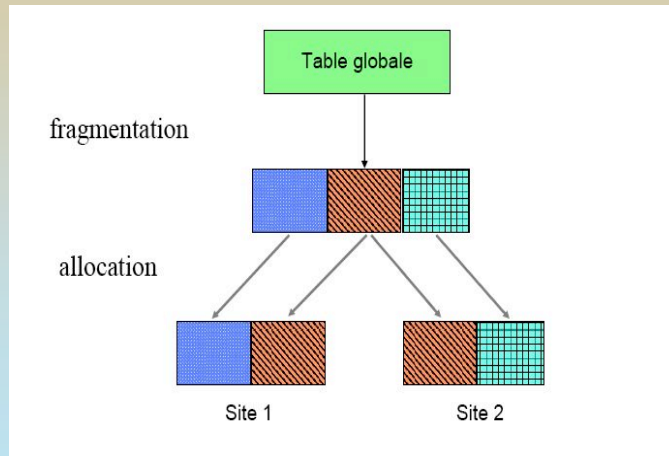
# Conception par décomposition

Fragmentation doit être

- 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

28

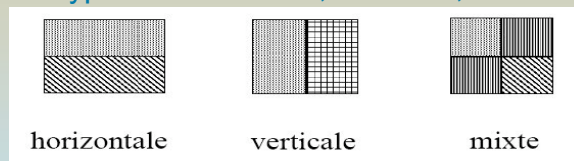
## Conception par décomposition



29

## Conception par décomposition

- fragmentation
- trois types : horizontale, verticale, mixte



- performances en favorisant les accès locaux
- équilibrer la charge de travail entre les sites (parallélisme)

30

## Fragmentation horizontale

Fragmentation horizontale de R sur m fragments  
N-uplets de R:  $\{t_1, \dots, t_n\}$  fragments :  $\{R_1, \dots, R_m\}$

Fragmentation par round-robin  
le n-uplet  $t_i \in R_j$  avec  $j=i \bmod m$   
*lecture séquentielle*

Fragmentation par hachage sur l'attribut A  
 $t_i \in R_j$  avec  $j=\text{hash}(t_i.A)$   
*sélection ( $A=v$ ), équijointure*

Fragmentation par intervalle  
Fragmenter le domaine de A en m intervalles  $(a_1, a_2, \dots, a_m)$   
 $t_i \in R_j$  avec  $a_{j-1} \leq t_i.A \leq a_j$   
*sélection ( $A$  between  $x$  and  $y$ )*

31

## Fragmentation horizontale par sélection (directe)

Fragments définis par sélection  
Client1 = Client where ville = 'Paris'  
Client2 = Client where ville  $\neq$  'Paris'

Reconstruction

Client = Client1 U Client2

nclient	nom	ville
C 1	Dupont	Paris
C 2	Martin	Lyon
C 3	Martin	Paris
C4	Smith	Lille

32

## Fragmentation horizontale dérivée (indirecte)

Fragments définis par semi-jointure

$Cde1 = Cde \bowtie Client1$

$Cde2 = Cde \bowtie Client2$

ncde	nclient	produit	qtte
D1	C1	P1	10
D2	C1	P2	20
D3	C2	P3	5
D4	C4	P4	10

Reconstruction

$Client = Cde1 \cup Cde2$

ncde	nclient	produit	qtte
D1	C1	P1	10
D2	C1	P2	20

ncde	nclient	produit	qtte
D3	C2	P3	5
D4	C4	P4	10

33

## Propriétés de la fragmentation horizontale dérivée (indirecte)

**R** fragmentation horizontale

**S** fragmentation horizontale dérivée

**Complète**

Chaque n-uplet de S doit joindre au moins un n-uplet de R

**Disjointe**

$\forall i, j \quad (S \bowtie R_i) \cap (S \bowtie R_j) = \emptyset$

**Reconstructible**

$S = \forall i, j \quad (S \bowtie R_i) \cup (S \bowtie R_2) \cup \dots \cup (S \bowtie R_n)$

34

## Fragmentation verticale

Fragments définis par projection

$Cde1 = Cde [ncde, nclient]$

$Cde2 = Cde [ncde, produit, qtte]$

ncde	nclient
D1	C1
D2	C1
D3	C2
D4	C4

ncde	nclient	produit	qtte
D1	C1	P1	10
D2	C1	P2	20
D3	C2	P3	5
D4	C4	P4	10

ncde	produit	qtte
D1	P1	10
D2	P2	20
D3	P3	5
D4	P4	10

Reconstruction

$Client = Cde1 \text{ jointure } Cde2$

35

## Propriétés de la fragmentation verticale

Fragmentation verticale de R

$A_i \subseteq \text{attributs}(R)$ , Fragments  $R_i = \prod_{A_i}(R)$

Complète

$\text{Attributs}(R) = A_1 \cup A_2 \cup A_3 \dots \cup A_n$

Disjointe

$A_1 \cap A_2 \cap A_3 \dots \cap A_n = \text{clé}(R)$

Reconstructible

$R = R_1 \text{ join } R_2 \text{ join } \dots \text{ join } R_n$

36

## Fragmentation mixte

Combinaison des deux fragmentations précédentes

Partition d'une relation

En sous-ensembles de n-uplets

Les n-uplets sont définis par fragmentation verticale

Les sous-ensembles par fragmentation horizontale

NV	cru	millésime	degré	prix
V1	J1	2000	12	15
V2	J2	2004	15	14
V3	J3	2003	13	9
V4	J1	2004	12	10

**VIN1 = Vin [NV, Degré, Prix]**

**VIN2 = sélection mill<2001 (VIN[NV, Cru, Millésime])**

**VIN3 = sélection mill>=2001 (VIN[NV, Cru, Millésime])** 37

## Allocation des fragments

Chaque fragment est alloué sur un site

**F un ensemble de fragments**

**S un ensemble de sites**

**Q un ensemble d'applications**

**Trouver une distribution optimale**

**Le plus près possible des applications qui les utilisent**

38

## Techniques de répartition avancées

### Duplication

Certains fragments sont dupliqués sur plusieurs sites  
Amélioration performances et disponibilité  
Difficultés MAJ

### Allocation dynamique

Changement de l'allocation d'un fragment en cours d'utilisation  
Efficace pour MAJ mais exige maintien du schéma d'allocation et schémas locaux

### Clichés

Copie figée de fragment

39

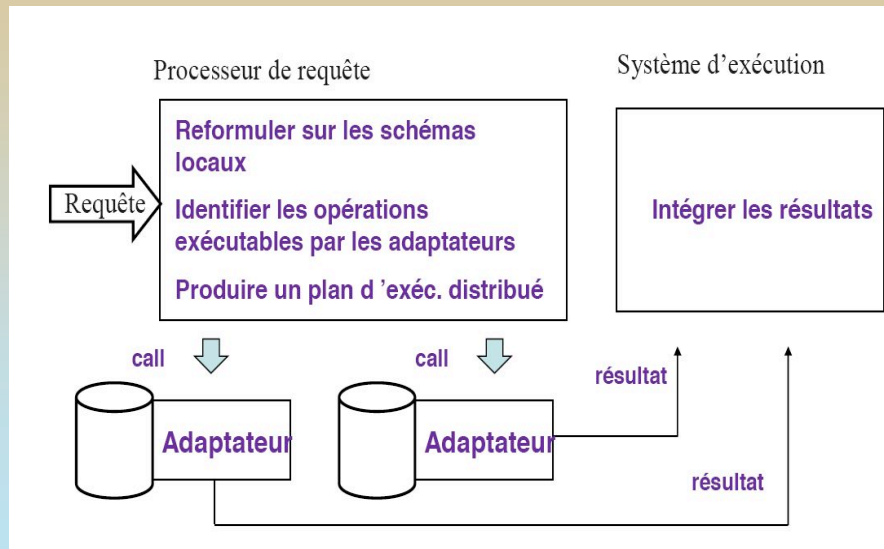
## Transparence des requêtes

Schéma de fragmentation  
définition des fragments

Schéma d'allocation  
contient la localisation

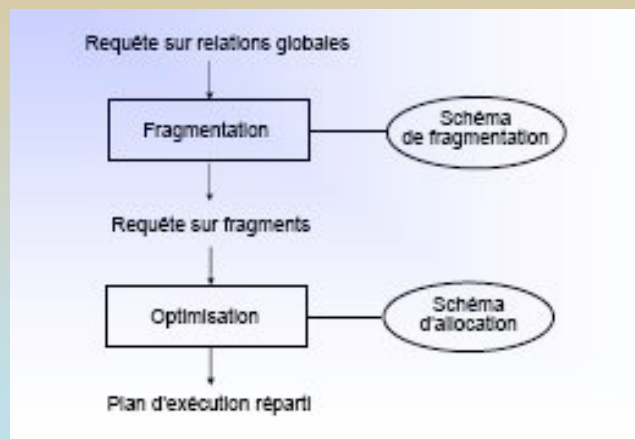
40

## Traitement de requête distribuée



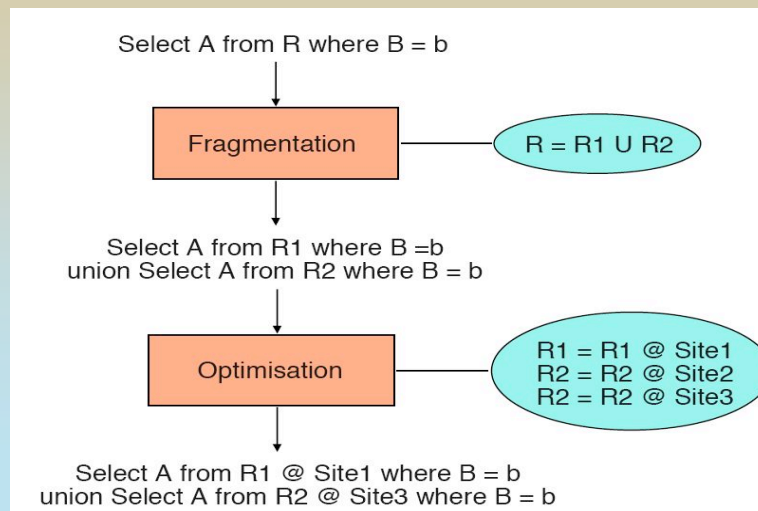
41

## Traitement de requête distribuée



42

## Exemple de traitement de requête simple



43

## Fragmentation

### Réécriture

mettre la requête sous la forme d'un arbre algébrique (feuille relation, nœud op relationnel P, J, S)

### Reconstruction

remplacer chaque feuille par le pgme de reconstruction de la relation globale

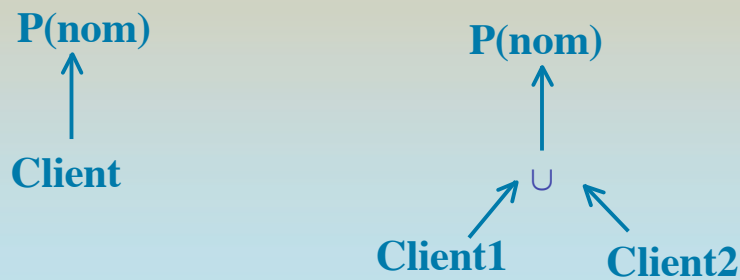
### Transformation

Appliquer des techniques de réduction pour éliminer les opérations inutiles

44

## Reconstruction

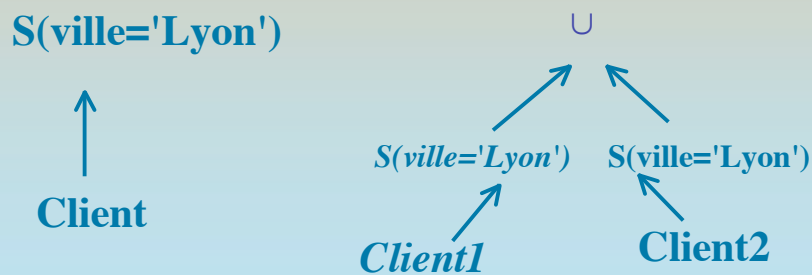
Select nom from Client



45

## Réduction pour la fragmentation horizontale

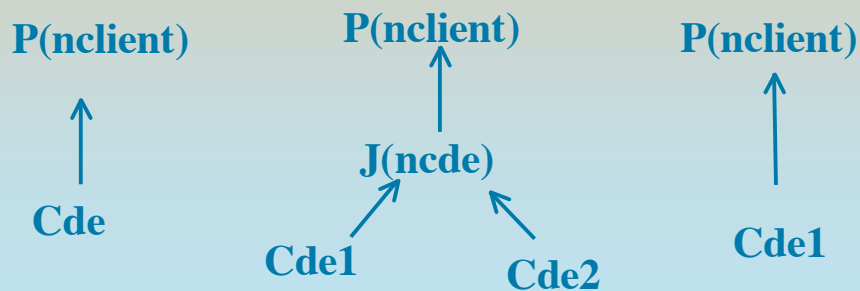
Éliminer les fragments inutiles  
*Select \* from Client where ville = 'Lyon'*



46

## Réduction pour la fragmentation verticale

Éliminer les accès aux relations de base qui n'ont pas d'attributs utiles pour le résultat final  
*Select nclient from Cde*



47

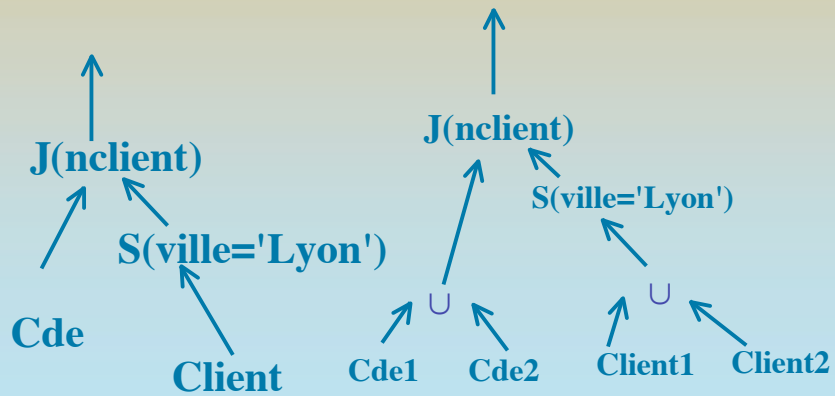
## Réduction pour la fragmentation horizontale dérivée

Distribuer les jointures par rapport aux unions et appliquer les réductions pour la fragmentation horizontale

```
Select * from Client, Cde  
Where Client.nclient=Cde.nclient  
And ville='Lyon'
```

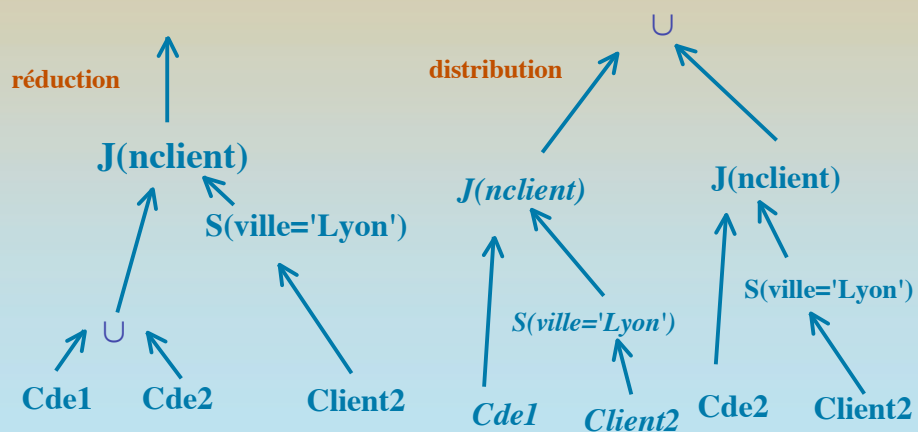
48

## Réduction pour la fragmentation horizontale dérivée



49

## Réduction pour la fragmentation horizontale dérivée



50

## Transactions réparties

- OBJECTIF
  - Garantir que toutes les mises à jour d'une transaction sont exécutées sur tous les sites ou qu'aucune ne l'est.
- EXEMPLE
  - Transfert de la somme X du compte A vers le compte B
  - DEBUT
    - site 1:  $A = A - X$
    - site 2:  $B = B + X$       PANNE --> INCOHERENCE  
DONNEES
  - FIN
- PROBLEME
  - Le contrôle est réparti : chaque site peut décider de valider ou d'annuler ...

51

## Transactions réparties

- PROBLEME

1 transaction globale T  
=> N transactions locales  $T_i$   
nécessité d'une coordination entre les sites locaux  $S_i$
- SOLUTION

un site coordonnateur C  
un protocole de validation en 2 étapes

52

## Commit en 2 phases

- Principe
  - Diviser la commande COMMIT en deux phases
  - *Phase 1* :
    - Préparer à écrire les résultats des mises à jour dans la BD
    - Centralisation du contrôle
  - *Phase 2* :
    - Écrire ces résultats dans la BD
- Coordinateur :
  - Le composant système d'un site qui applique le protocole
- Participant :
  - Le composant système d'un autre site qui participe dans l'exécution de la transaction

53

## Protocole C/S

- .PREPARER
  - Le coordinateur C demande aux autres sites Si s'ils sont prêts à commettre leurs mises à jour.
- 2a. SUCCES : COMMETTRE
  - Tous les participants effectuent leur validation sur ordre du coordinateur.
- 2b. ECHEC : ABORT
  - Si un participant n'est pas prêt, le coordinateur demande à tout les autres sites de défaire la transaction.
- REMARQUE
  - Le protocole nécessite la journalisation des mises à jour préparées et des états des transactions dans un journal local à chaque participant.

54

- **PHASE 1**

- > Le site C ajoute <T préparée> dans son journal
- > Le site C envoie "prêt à commettre T" à tous les sites Si
- > Chaque site Si prend une décision :
  - s'il valide Ti , alors :
    - . Le site Si écrit <T prête à commettre> dans son journal
    - . Le site Si envoie "T prête" au site C
  - s'il ne valide pas Ti , alors :
    - . Le site Si écrit <non T> dans son journal
    - . Le site Si envoie "abort T" au site C

55

- **PHASE 2**

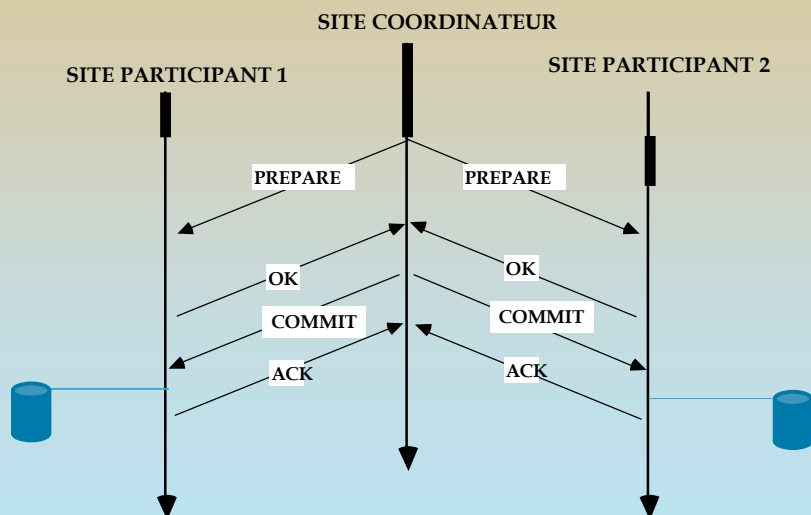
- > A partir des messages reçus des sites Si et au terme d'un laps de temps déterminé, le site C prend une décision :
  - il valide T s'il a reçu "T prête" de tous les sites Si :
    - . il ajoute <T validée> à son journal
    - . il envoie "valider T" à tous les sites
  - il avorte T s'il a reçu au moins un message "abort T" d'un site Si (ou au bout du laps de temps spécifié) :
    - . il ajoute <T abortée> à son journal
    - . il envoie <abort T> à tous les sites Si

56

- > Chaque site  $S_i$  :
  - inscrit dans son journal le message envoyé par le site C ( <T validée> ou <T abortée> )
  - envoie un accusé de réception au site C
- > Quand le site C a reçu tous les accusés des sites  $S_i$  , il ajoute : <T finie> à son journal

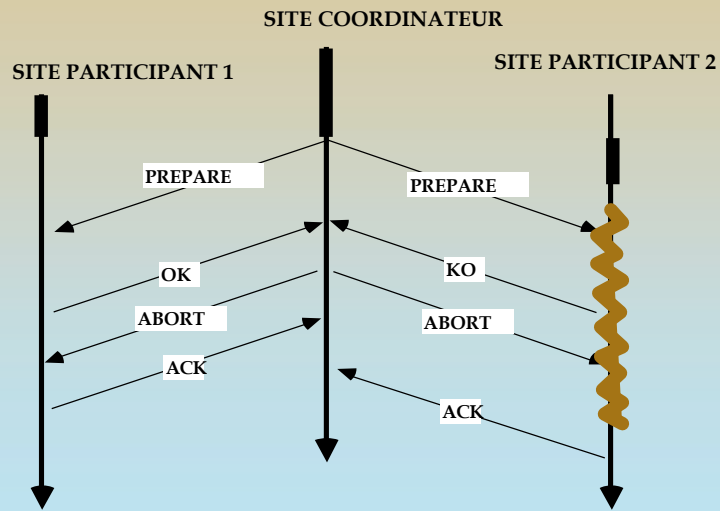
57

## Cas favorable



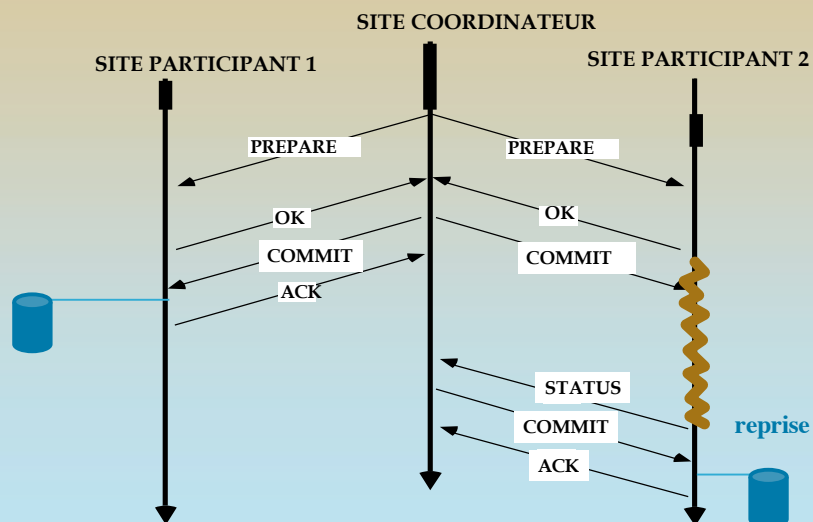
58

## Cas défavorable



59

## Cas panne d'un participant



60

## Panne d'un site participant

- Un site participant Si tombe en panne  
==> REPRISE A FROID DE Si
- Analyse du journal du site Si :
  - <T validée> => refaire T
  - <T abortée> => défaire T
  - <T prête à commettre>
    - => consulter le site C pour avoir l'état de T :
    - C répond => Si a connaissance de l'état de T
    - C ne répond pas => Si envoie une demande de l'état de à tous les sites
  - Rien concernant T => Si sait que C a aborté T

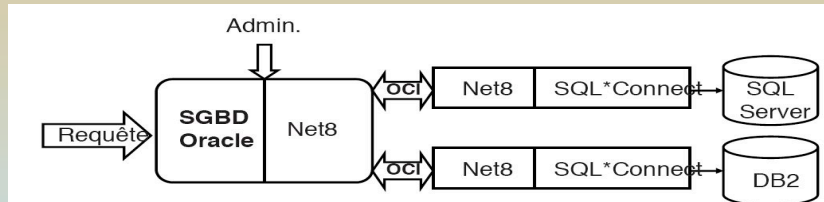
61

## SGBD distribués

- SGBD relationnels
  - Oracle, Ingres, Sybase, DB2, Informix
- DataJoiner (IBM)
  - basé sur DB2
- VirtualDB (Enterworks)
  - basé sur GemStone, vue objet des tables
- Open Database Exchange (B2Systems)

62

## Oracle et la distribution des données



- SGBD Oracle
  - gestion du catalogue de la BDR
- SQL\*Net
  - connexion client-serveur, login à distance automatique
  - requêtes distribuées, transactions distribuées, réplication
- SQL\*Connect : passerelle vers les bases non-Oracle

63

## Database link

- Lien à une table dans une BD distante spécifié par :
  - nom de lien
  - nom de l'utilisateur et password
  - chaîne de connexion Net8 (protocole réseau, nom de site, options, etc...)
- Exemple

```
CREATE DATABASE LINK empParis
CONNECT TO anne IDENTIFIEDBY monPW
USING Paris.emp
```

64