Base de données distribuées (distributed database)

Dr I GAYE

Universite Alioune DIOP (UAD)

3 novembre 2025

Objectifs du Cours

- ☞ Comprendre les concepts de base des systèmes de bases de données réparties.
- ☞ Maîtriser les architectures et les modèles de données répartis.
- ☞ Apprendre les techniques de fragmentation, de réplication et de gestion des transactions.
- ☞ Identifier les défis liés à la cohérence, la disponibilité et la tolérance aux pannes.
- ☞ Explorer des systèmes de bases de données réparties modernes (NoSQL, NewSQL, etc.).

Références

- Livres
  - *Distributed Databases : Principles and Systems* ; Stefano Ceri et Giuseppe Pelagatti.
  - *Designing Data-Intensive Applications* ; Martin Kleppmann.
  - *Introduction aux BD Réparties* ; Bernard Espinasse
  - *Matthieu Exbrayat – BD Réparties* ; ULP Strasbourg - Décembre 2007
- Documentation technique des systèmes de bases de données réparties (Cassandra, MongoDB, etc.).

Outline

Généralité sur les BDD

C'est quoi une Base de Données Réparties (ou Distribiées)

- ☞ C'est une Base de Données qui n'est pas limitée à un seul système, elle est répartie sur différents sites, c'est-à-dire sur plusieurs ordinateurs ou sur un réseau d'ordinateurs.
- ☞ Un Système de Base de Données Réparties est situé sur différents sites qui ne partagent pas de composants physiques.

Généralité sur les BDD

C'est quoi une Base de Données Réparties (ou Distribiées)

- Comme toute BD, une BDR possède un schéma appelé schéma global qui permet de définir l'ensemble des types de données de la base.
- Distribution : elle est faite sans redondance des données : Base de données distribuées.
- Réplication : elles est une copie des informations d'une base à une autre.
  - **Objectif** : la disponibilité des données

Fonctonnalités

Les mêmes fonctionnalités classiques des Bases de Données avec quelques fonctionnalités additionnelles :

- ☞ Transparence de la localisation via des mécanismes de fragmentation ou d'agrégation ;
- ☞ Décomposition des requêtes et Agrégation des réponses aux sous-requêtes générées ;
- ☞ Cohérence des mises à jour multibase grâce au protocole de validation à deux phases (2PC pour Two Phases Commit Protocol)
- ☞ Réplication des données sur les sites pour optimiser leur accès.

Pourquoi BD Réparties

- Applications naturellement répartiés
  - Télécommunications
  - Applications critiques en temps de réponse : réservation arérienne, système bancaire, ...
- Evolution de la technologie : réseaux, puissance des machines, ...
- Vulnérabilités des systèmes centralisés
- Souplesse à l'évolution : ajout de périphériques, modificaion de configurations, ...

Intétêts BD Réparties

- Coopération : échange de données, exécution locale d'une partie de transaction, ...
- Vue intégrée des données : l'utilisateur ignore la localisation
- Disponibilité avec la réplication : plusieurs données identiques sur plusieurs sites
- Performance : temps d'accès aux données, files d'attente (dans le réseau, dans les serveurs, ...)
- Resistance aux pannes : moins vulnérable que le système centralisé
- Extensibilité
- Partage des données hétérogènes et réparties
- Performances avec le parallélisme

Difficultés et problèmes des BD Réparties

- Développement logiciel
  - complexité de mise en oeuvre et de développement
  - difficultés d'assurer des algorithmes pour des sites travaillant en parallèle
- Complexité : intégration des aspects réseaux (introduction de protocoles), gestion de la cohérence des données plus difficile
- Augmentation de l'overload (surcharge) : l'échange de messages augmente le temps de calcul
  - Surcharge du réseau dû aux échange de messages
  - traitements supplémentaires : pour la coordination entre sites
- Administration complexe

Architecture des BD réparties et réplication

Architecture des BD réparties

- Plusieurs niveaux d'intégration
  - Client/serveur : BD centralisée, seuls certains traitements sont locaux.
  - Vues réparties : extension du mécanisme de vues pour définir des vues sur plusieurs sites.
- BD réparties
  - Plusieurs BD sur plusieurs sites, mais une seule BD "logique".
  - Ordinateurs (sites) faiblement couplés, communiquent via le réseau.
  - Chaque site contient des données de la base, peut exécuter des transactions locales et participer à l'exécution de transactions globales.

Architecture des BD réparties

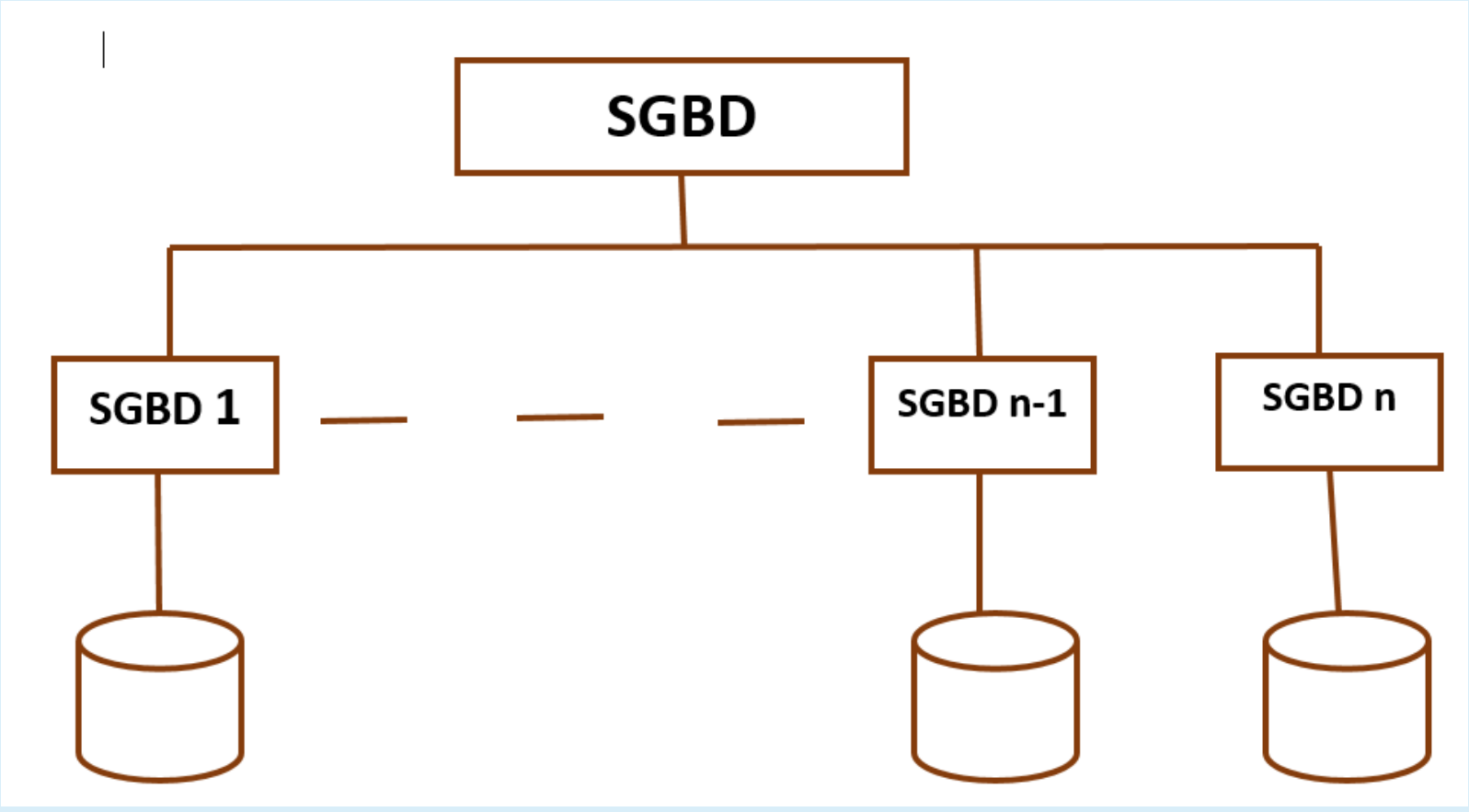


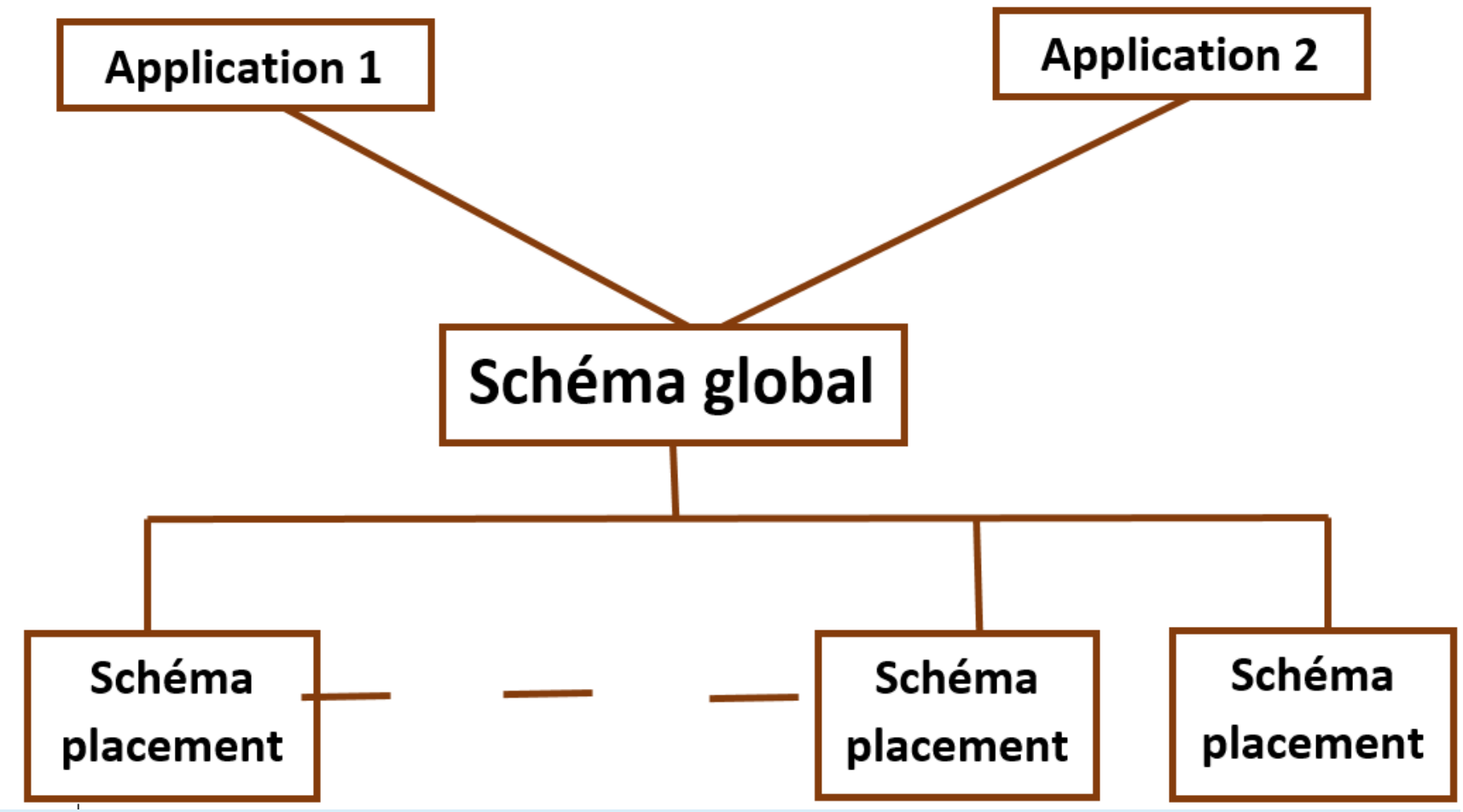
FIGURE: SGBD réparti

Architecture des BD réparties

Rend la répartition (ou distribution) transparente

- ☞ dictionnaire des données réparties
- ☞ traitement des requêtes réparties
- ☞ gestion de transactions réparties
- ☞ gestion de la cohérence et de la sécurité

Architecture des BD réparties



indépendance applications/BDR

Schéma global et Schéma de placement

- Schéma conceptuel global
  - Donne la description globale et unifiée de toutes les données de la BDR (ex : les relations globales)
  - Indépendant de la répartition de données
- Schéma de placement
  - règles de correspondance avec les données locales
  - Fournit une indépendance à la localisation, la fragmentation (horizontale et/ou verticale) et la duplication

Le schéma global fait partie du dictionnaire de la BDR.
Il peut être dupliqué ou fragmenté.

Architecture des BD réparties

Schéma global et Schéma de placement

- Schéma conceptuel global

  Client (nclient, nom, ville)
  Cde (ncde, nclient, produit, qté)

- Schéma de placement

  Client = Client1@Site1 $\bigcup$ Client1@Site2
  Cde = Cde@Site3

Schéma global et Schéma de placement

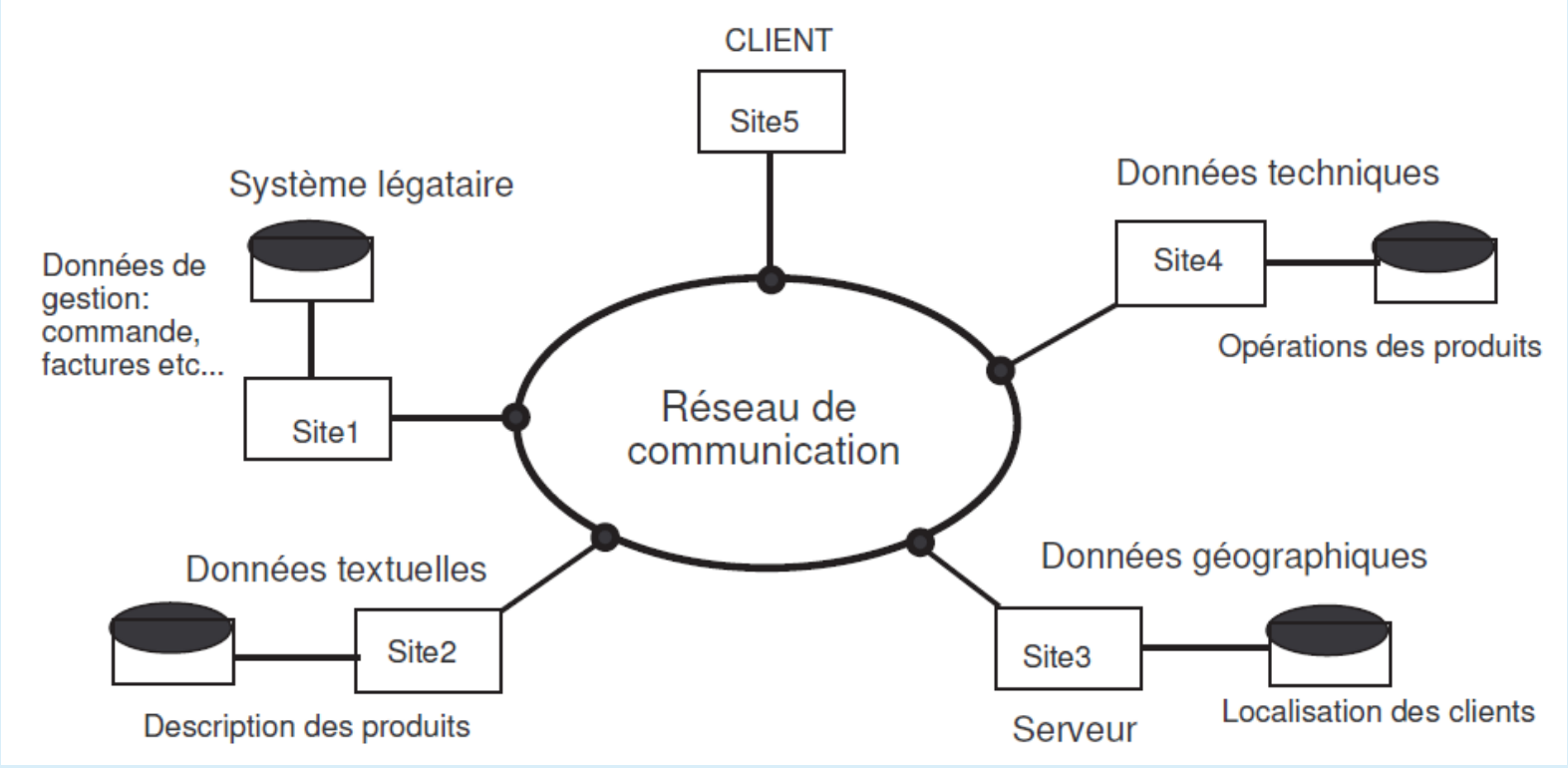


FIGURE: Un BD répartie sous 5 sites

Processeur de requêtes

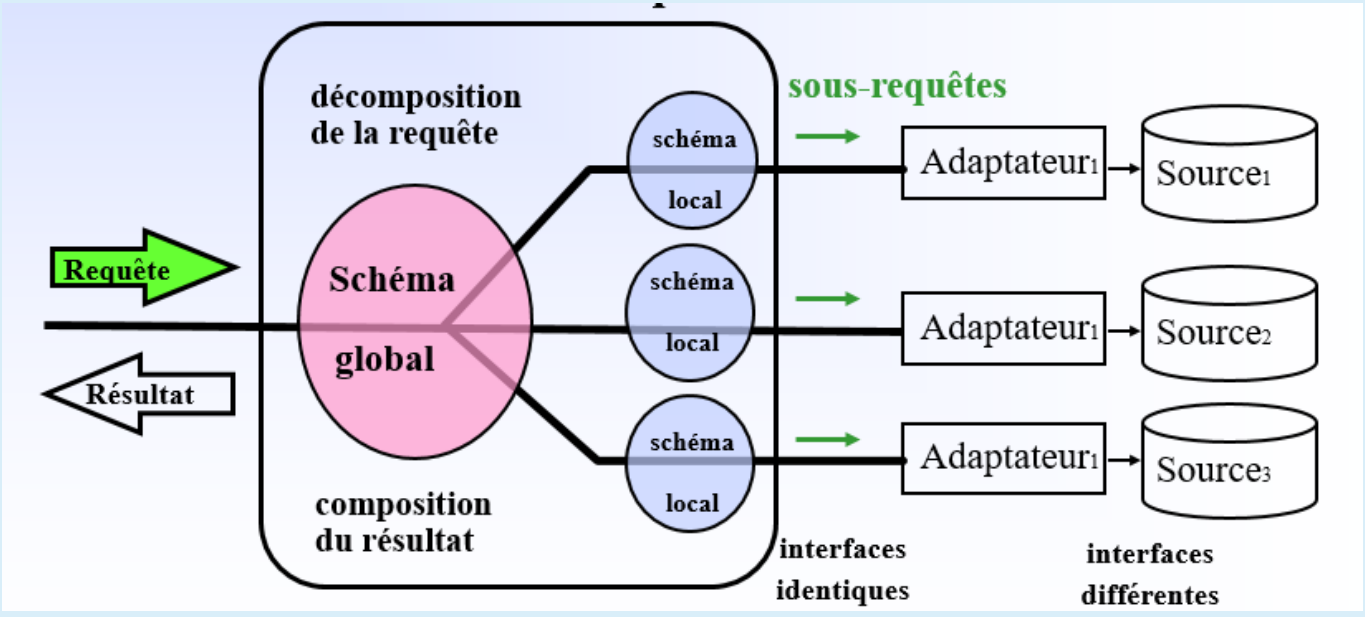


FIGURE: Processus des requêtes réparties

Offres d'adaptateurs : les produits

- SGBD relationnels
  - Oracle, DB2, SQL Server 2000, Sybase, Informix
- VirtualDB (Enterworks)
  - basé sur GemStone, vue objet des tables
  - Open Database Exchange (B2Systems)

Offres d'adaptateurs : Oracle

Oracle/Star

- SGBD Oracle
  - Gestion du dictionnaire de la Base de Données Réparies
- SQL*Net : SQL*Net (ou Net8) est le logiciel de mise en réseau d'Oracle qui permet l'accès à distance aux données entre les programmes et la base de données Oracle, ou entre plusieurs bases de données Oracle .
  - transparence au réseau
  - connexion client-serveur, login à distance automatique
  - évaluation de requêtes réparties
  - validation en deux étapes et réplication
- SQL*Connect : passerelle vers les bases non-Oracle

Offres d'adaptateurs : Oracle

Database link

- ☞ Lien à une table dans une BD distante specifié par :
- ☛ nom de lien
- ☛ nom de l'utilisateur et password
- ☛ chaîne de connexion SQL*Net (protocole réseau, nom de site, options, etc. . .)

Exemple :

```
CREATE DATABASE LINK testLINK
CONNECT TO Ibrahima
IDENTIFIEDBY monPW
USING Dakar.emp
```

Offres d'adaptateurs : Oracle

Database link

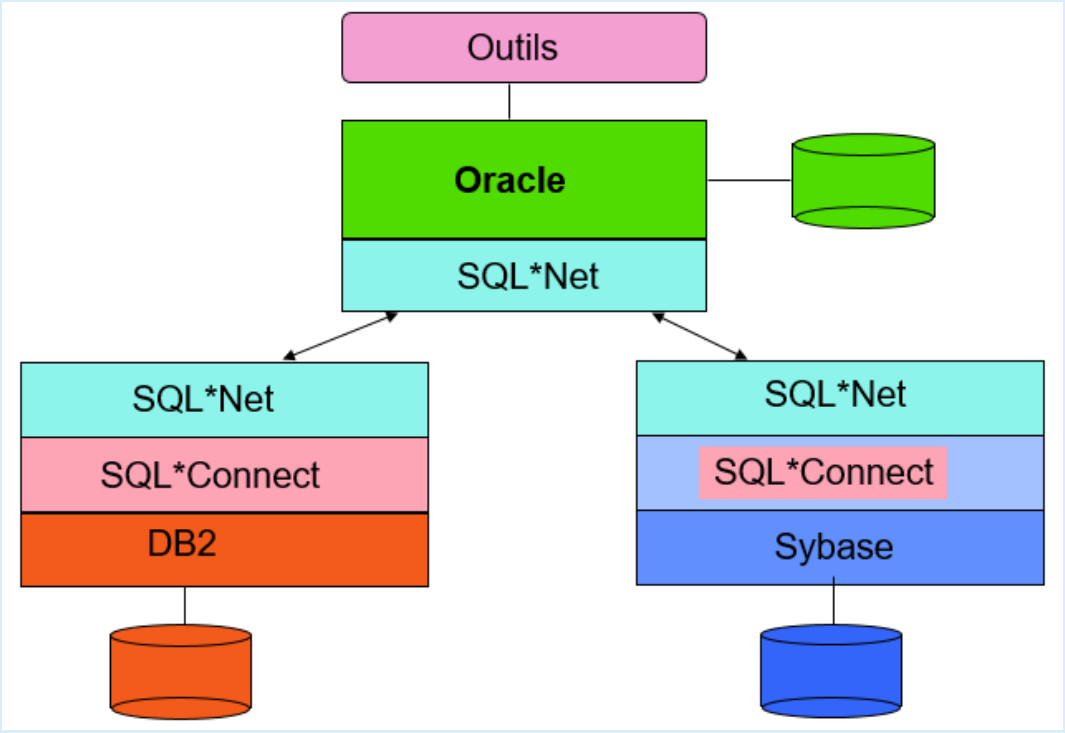


FIGURE: Processus des requêtes réparties

Architecture des BD réparties et réplication

La réplication

Définition

- L'ensemble de la relation est stocké de manière redondante sur des sites.
- Si le schéma est disponible sur tous les sites : Redondance entière.
- Réplication ou copie de données
  - Fragment horizontal ou vertical d'une table stockée dans une base de données qui est copiée et transféré vers une autre base de données
  - L'original est appelé la copie primaire et les copies sont appelées copies secondaires

La réplication

Définition

- Transparence
  - Les applications clientes croient à l'existence d'une seule copie des données qu'ils manipulent
    - soit « logique » dans le cas d'une vue
    - soit physique
- Inconvénient : les données doivent être constamment mises à jour ; toute modification apportée sur un site doit être enregistrée sur chaque site.

La réplication

- Base de données homogène
  - Tous les sites stockent la base de données de manière identique.
  - Le SE, le SGBD et les Schémas de Base de Données sont identiques sur tous les sites.

La réplication

- Base de données hétérogène
  - Les différents sites peuvent utiliser des schémas et des logiciels différents susceptibles d'entraîner des problèmes dans le traitement des requêtes et les transactions.
  - Un site particulier peut ignorer complètement les autres sites.
  - Des SE, des SGBD, des schémas sont différents.
  - conséquence : une intégration est nécessaire pour que les différents sites puissent communiquer.

La réplication

Avantage de la réplication

- Amélioration des performances
  - lecture de la copie la plus proche
  - évitement du goulot d'étranglement du serveur unique
- Amélioration de la disponibilité
  - lors d'une panne d'un serveur, on peut se replier sur l'autre
- Meilleure tolérance aux pannes
  - possibilité de détecter des pannes diffuses

Les problèmes de la réplication

- Convergence
  - les copies doivent être maintenues à jour
  - à un instant donné, elles peuvent être différentes
  - mais elles doivent converger vers un même état cohérent où toutes les mises à jour sont exécutées partout dans le même ordre
- Transparence : le SGBD doit assurer
  - la diffusion et la réconciliation des mises à jour
  - la résistance aux défaillances

La réplication

Diffusion synchrone

- Une transaction met à jour toutes les copies de toutes les données qu'elle modifie.
  - mise à jour en temps réel des données
  - Malheureusement trop coûteux pour la plupart des applications
  - Malheureusement pas de contrôle de l'instant de mise à jour

Diffusion synchrone

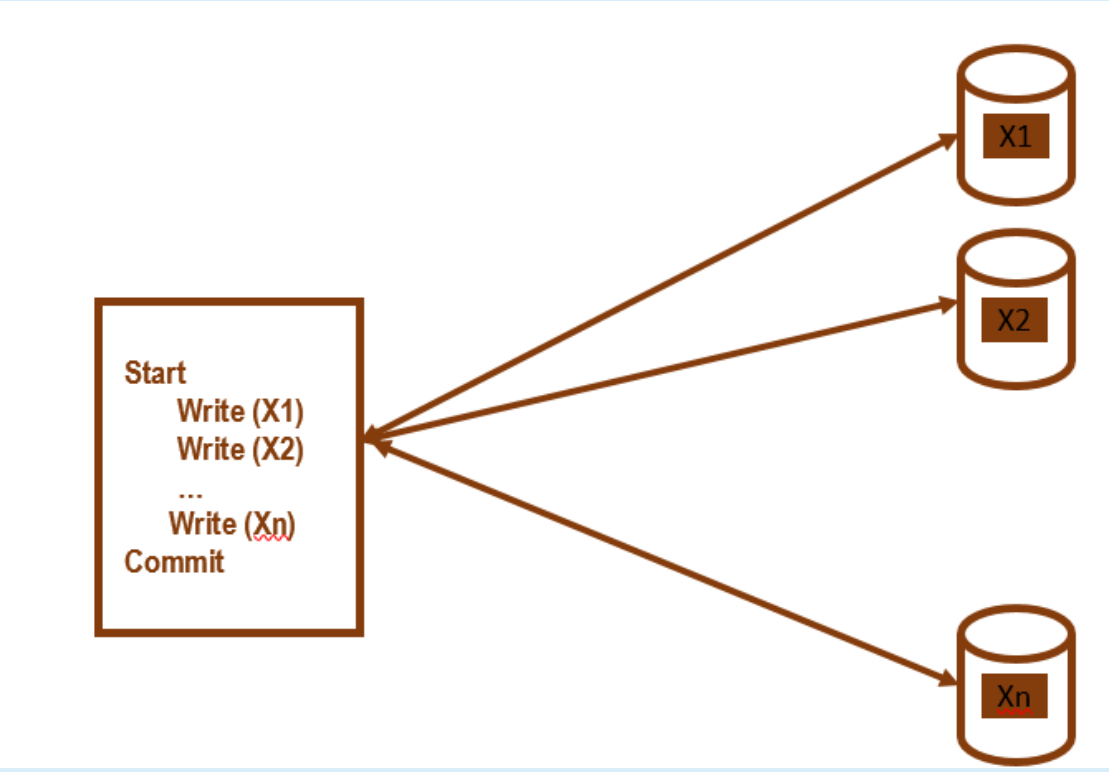


FIGURE: Réplication synchrone

La réplication

Diffusion asynchrone

- Chaque transaction met à jour une seule copie et la mise à jour des autres copies est différée (dans d'autres transactions)
- Réplication asymétrique : toutes les transactions mettent à jour la même copie
- Réplication symétrique : les transactions peuvent mettre à jour des copies différentes
  - mise à jour en temps choisi des données
  - accès aux versions anciennes puis nouvelles
  - Malheureusement l'accès à la dernière version n'est pas garanti

Validation normale d'une réplication

- Pour assurer qu'une transaction validée sur le serveur primaire, le soit aussi sur les serveurs secondaires, la réplication synchrone utilise le protocole de validation en deux phases (two-phase commit ou 2PC).
- Lorsque le serveur primaire reçoit un commit, il écrit les données sur le disque et indique qu'elles sont dans l'état "prêt", puis il effectue les opérations suivantes :
  - Demande à tous les réplicas s'ils sont prêts à valider la transaction (phase de vote).
  - Attends que tous les réplicas aient répondu ou que le délai soit dépassé.
  - Si tous les réplicas sont prêts, ordonne aux réplicas de valider la transaction, sinon ordonne aux réplicas de l'annuler.

La réplication

Validation normale d'une réplication

Ce protocole fonctionne à condition qu'il n'y ait pas de panne du coordinateur (dans notre cas le serveur primaire) ou de l'un participant (un réplica) entre la phase de vote et la validation ou l'annulation

Validation normale d'une réplication

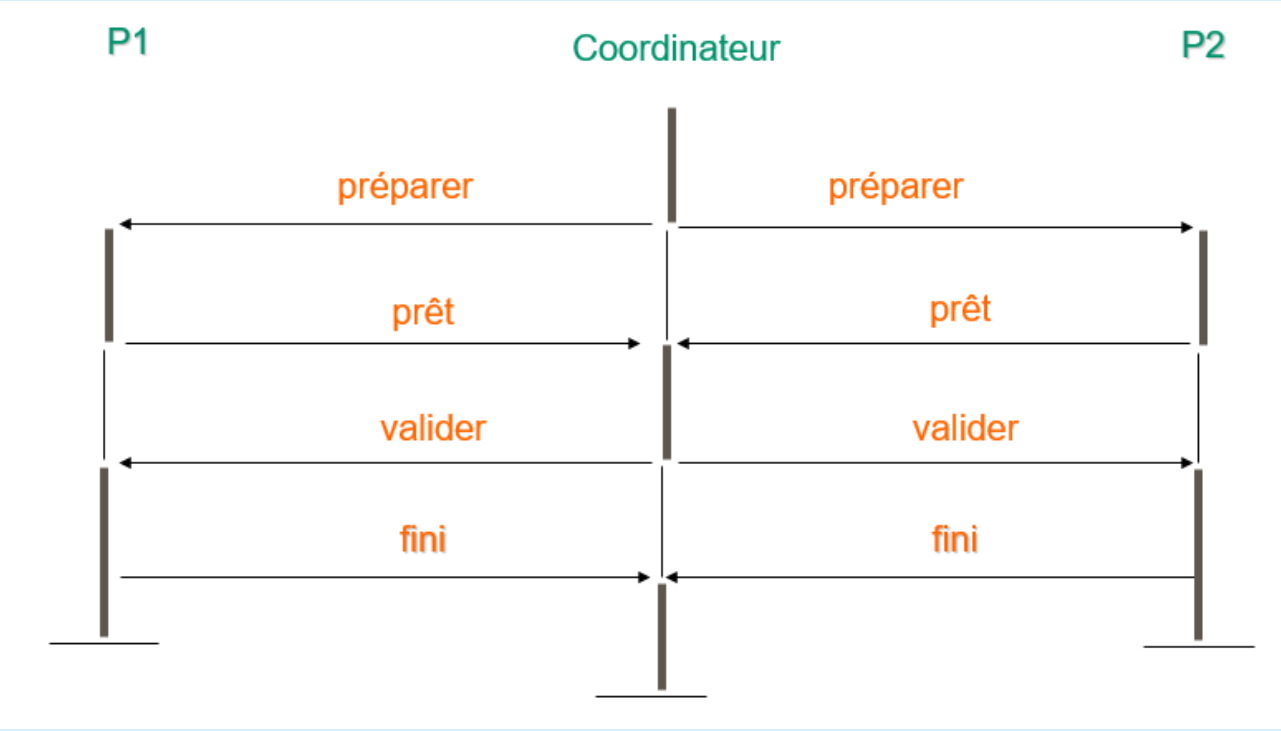


FIGURE: protocole de validation en deux phases (two-phase commit ou 2PC)

Conception des BD réparties et requêtes

Migration vers une BDR : intégration

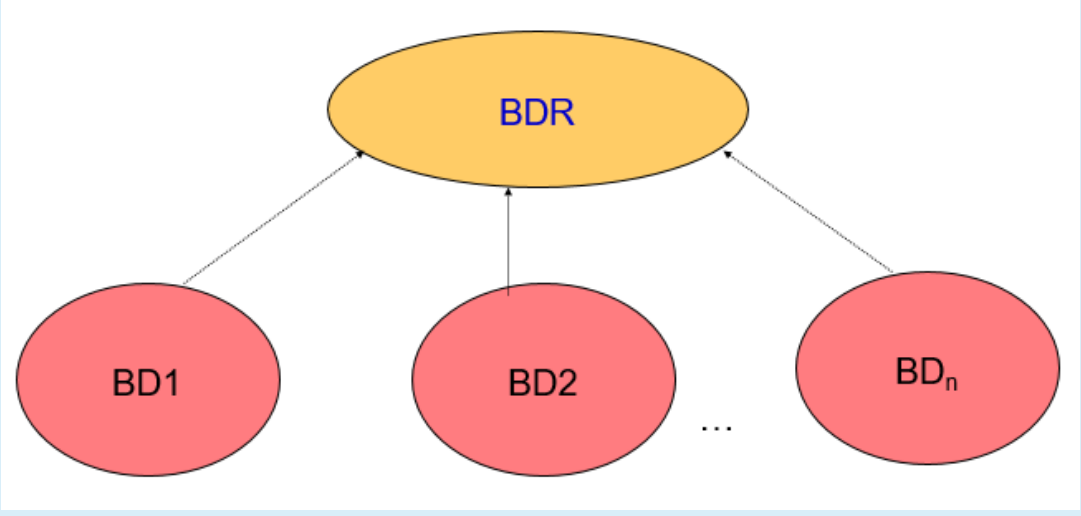


FIGURE: Conception par intégration

Migration vers une BDR : décompsition

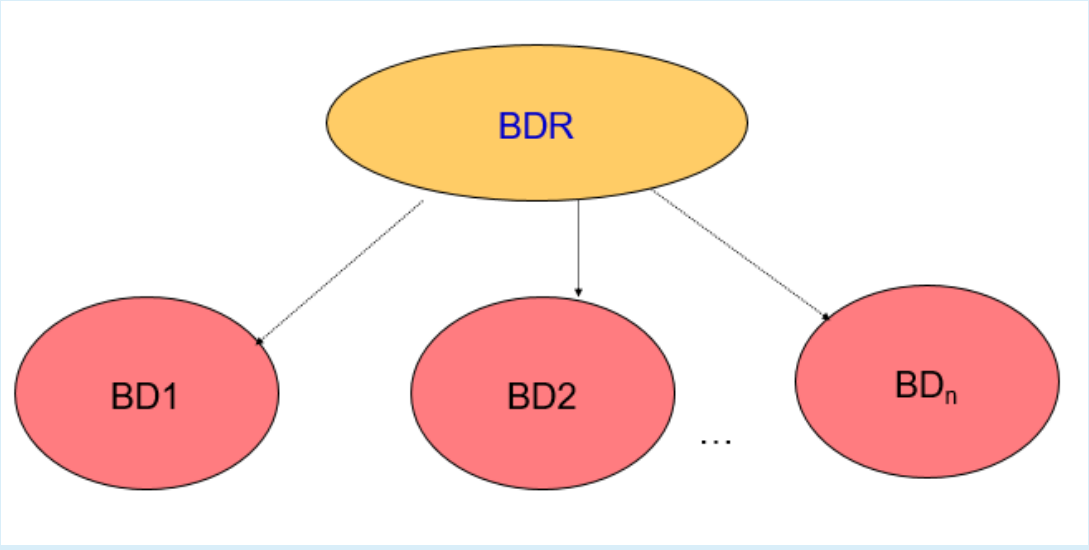


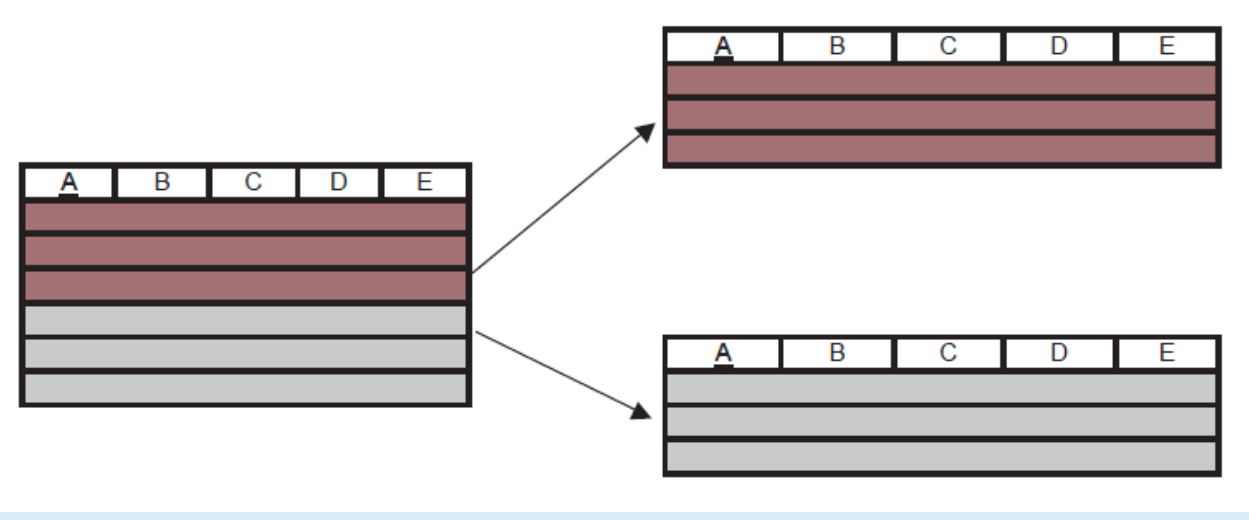
FIGURE: Conception par décomposition

Conception par décomposition

- Fragmentation
  - Les relations sont fragmentées et chacun des fragments est stocké dans différents sites.
  - Il faut s'assurer que les fragments sont tels qu'ils peuvent être utilisés pour reconstruire la relation originale.
  - Avantage : ne crée pas de copies des données ; la cohérence n'est pas un problème.
  - La fragmentation des relations peut se faire de trois manières :
    - Fragmentation horizontale ;
    - Fragmentation verticale ;
    - Fragmentation mixte.

Décomposition horizontale

- Chaque fragment $\Re_i = \sigma_{Q_i}(\Re)$ est localisée dans le site *i*
- le recomposition de $\Re = \bigcup \Re_i$



Décomposition horizontale

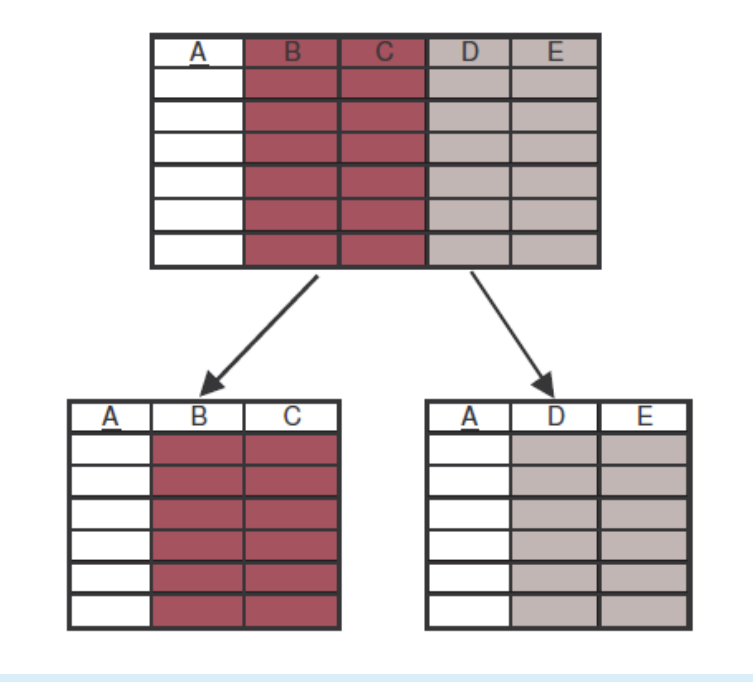
Soit la relation commande (ncde, nclient, produit, qté)

ncde	nclient	produit	qté
cmd1	cl1	Savon	200
cmd12	cl1	lait	10
cmd23	cl12	savon	20
cmd24	cl12	lait	40
cmd45	cl1	Fromage	3

☞ Faire une partition horizontale des commandes suivant le client (en algèbre relationnelle et en SQL)

Décomposition verticale

- Chaque fragment $\Re_i = \Pi_{\Re_i}(\Re)$ est localisée dans le site *i*
- le recomposition de $\Re = \bowtie \Re_i$



Conception d'une BD réparties

Décomposition horizontale

Soit la relation commande (ncde, nclient, produit, qté)

ncde	nclient	produit	qté
cmd1	cl1	Savon	200
cmd12	cl1	lait	10
cmd23	cl12	savon	20
cmd24	cl12	lait	40
cmd45	cl1	Fromage	3

☞ Faire une partition verticale en regroupant les info sur les articles et sue les clients (en algèbre relationnelle et en SQL)

Conception d'une BD réparties

Facteur d'affinité

- Soit $D = D_1, D_2, ..., D_m$ un ensemble de m données
- Soit $s = S_1, S_2, ..., S_n$ un ensemble de n sites contenant ces données.
- Une matrice réelle $A(mXn)$: fréquences des données aux sites.

Facteur d'affinité

- Affinité d'une données D_i

$$F(D_i) = \sum_{j=1}^{n} A_{i,j} \tag{1}$$

- Affinité mutuelle de la donnée D_i et de la source S_j notée $F(D_i, S_j)$ est égale à $A_{i,j}$
- Facteur d'affinité

$$AF(D_i, S_j) = \frac{F(D_i, S_j)}{F(D_i) + F(S_j)} \tag{2}$$

Facteur d'affinité

☞ Règle de palcemet des données dans les sités

La donnée D_i seera localisée sur le site S_j si et seulement si :

$$\forall l \in [1..n] et l \neq j AF(D_i, S_j) > AF(D_i, S_l) \quad (3)$$

On cherche à placer les données là où elles sont plus sollicitées
Attention à la disponibilité : l'affinité pure n'accepte pas de duplication. Une donnée ne sera donc disponible que sur un et un seul site.

Facteur d'affinité

☞ Exemple : soit quatre données et cinq sites

D/S	S_1	S_2	S_3	S_4	S_5
D_1	5	7	3	1	8
D_2	6	4	9	10	2
D_3	6	3	8	9	12
D_4	12	7	4	1	3

TABLE: Fréquence d'accès des données aux sites

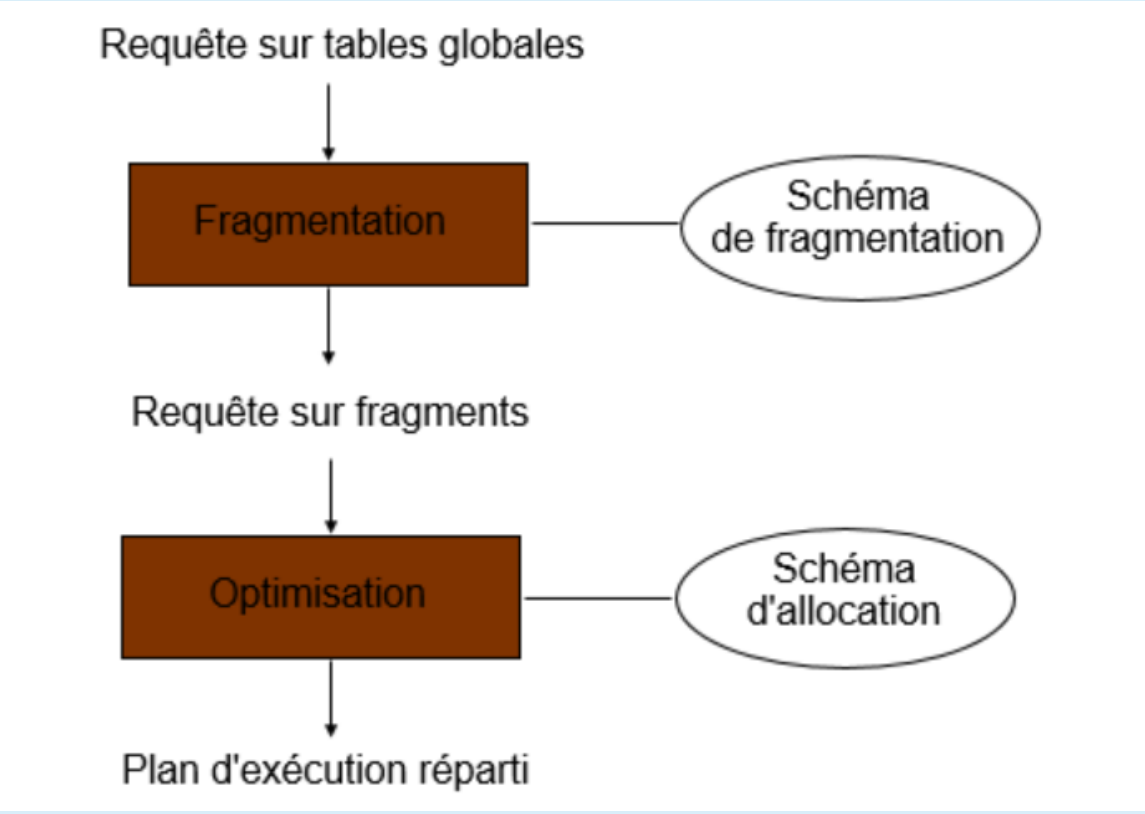
Question : Calculer les facteurs d'affinités et placer les données dans les sites.

Conception des BD réparties et requêtes

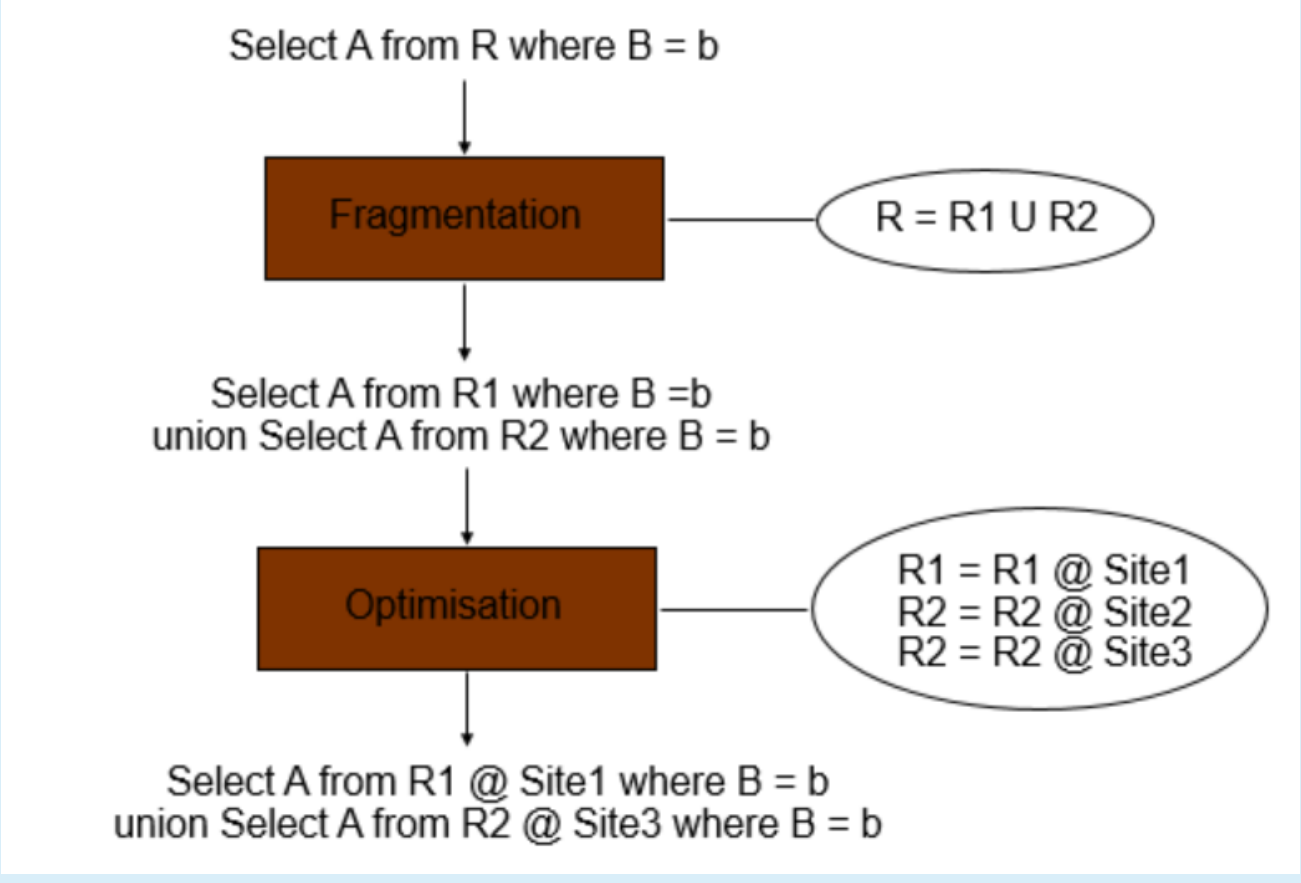
Optimisation de requêtes dans le cas centralisé

- Rappels
  - Commutavité des jointures et des produits cartésiens : $R1 \bowtie R2 = R2 \bowtie R1$ et $R1 * R2 = R2 * R1$
  - Cascade des séléctions : $\sigma_{F1}(\sigma_{F2}(E)) = \sigma_{F1 \wedge F2}(E)$
  - Commutativité des sélections et projections : $\Pi_{A_1,A,2,...,A_n}\sigma_F(R) = \sigma_F(\Pi_{A_1,A,2,...,A_n}(R)$
  - Commutativité des sélections et produits cartésiens : $\sigma_F(R1 * R2) = \sigma_F(R1) * \sigma_F(R2)$
  - Commutativité de la sélection et union : $\sigma_F(R1 \bigcup R2) = \sigma_F(R1) \bigcup \sigma_F(R2)$
  - etc.

Evaluation de requêtes réparties



Exemple d'évaluation simple



Fragmentation

- Réécriture
  - Traduire la requête SQL en un arbre algébrique
    - Noeuds feuilles sont les relations
    - Autres noeuds sont les opérateurs relationnel
- Reconstruction
  - Substituer chaque relations globale par sa définition en fonction des fragments
- Transformation
  - Appliquer des techniques de réduction pour éliminer les opérations inutiles

Exemple de fragementation

- Schéma conceptuel global

  Client(nclient, nom, ville)
  Com (ncde, nclient, produit, qté)

- Schéma de placement

  Client = Client1@SiteDakar $\bigcup$ Client1@SiteAutre
  Com = Com@SiteDakar $\bigcup$ *Com@SiteAutre*

- Les clients sont répartis dans deux sites (Dakar : les clients de Dakar ; Autre : les clients qui ne sont pas de Dakar)
- Certaines colonnes des commandes sont stockées dans le site de Dakar et d'autres dans le site Autre

Requêtes réparties

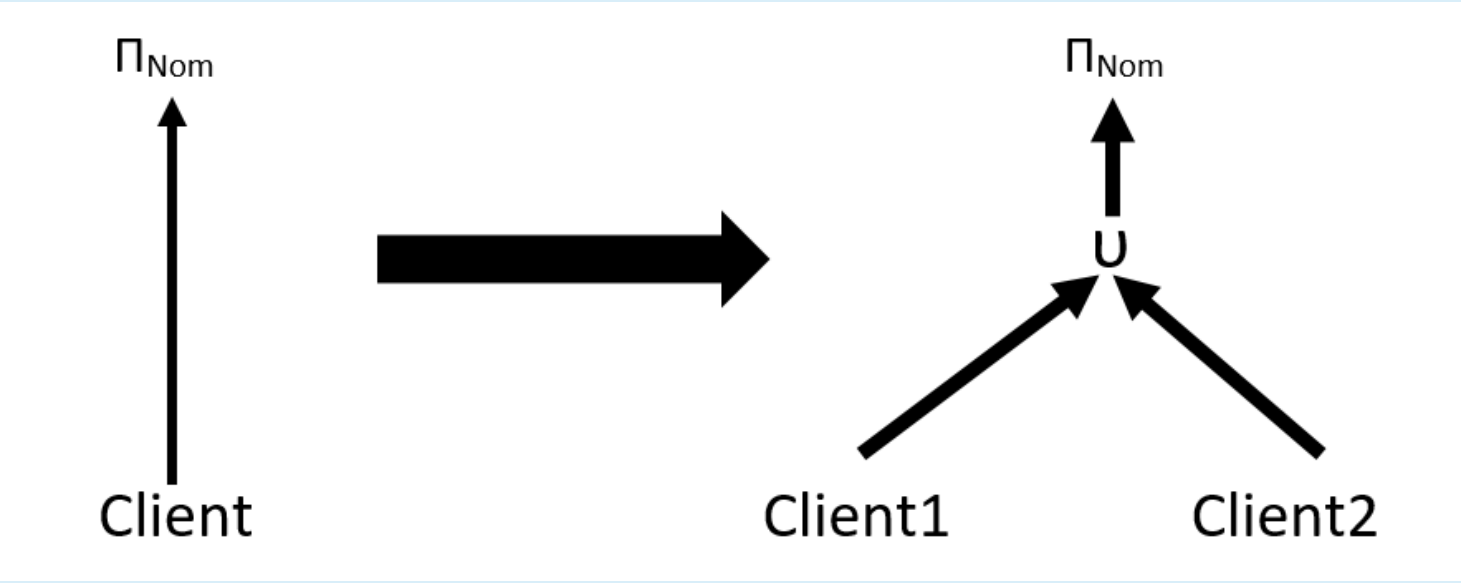
Reconstruction

☞ Requête : **Select distinct nom From Client**

$$\left\{ \begin{array}{l} Client1 = \sigma_{ville='Dakar'} Client \\ Client1 = \sigma_{ville\neq'Dakar'} Client \end{array} \right\} Client = Client1 \bigcup Client2$$

Requêtes réparties

Reconstruction



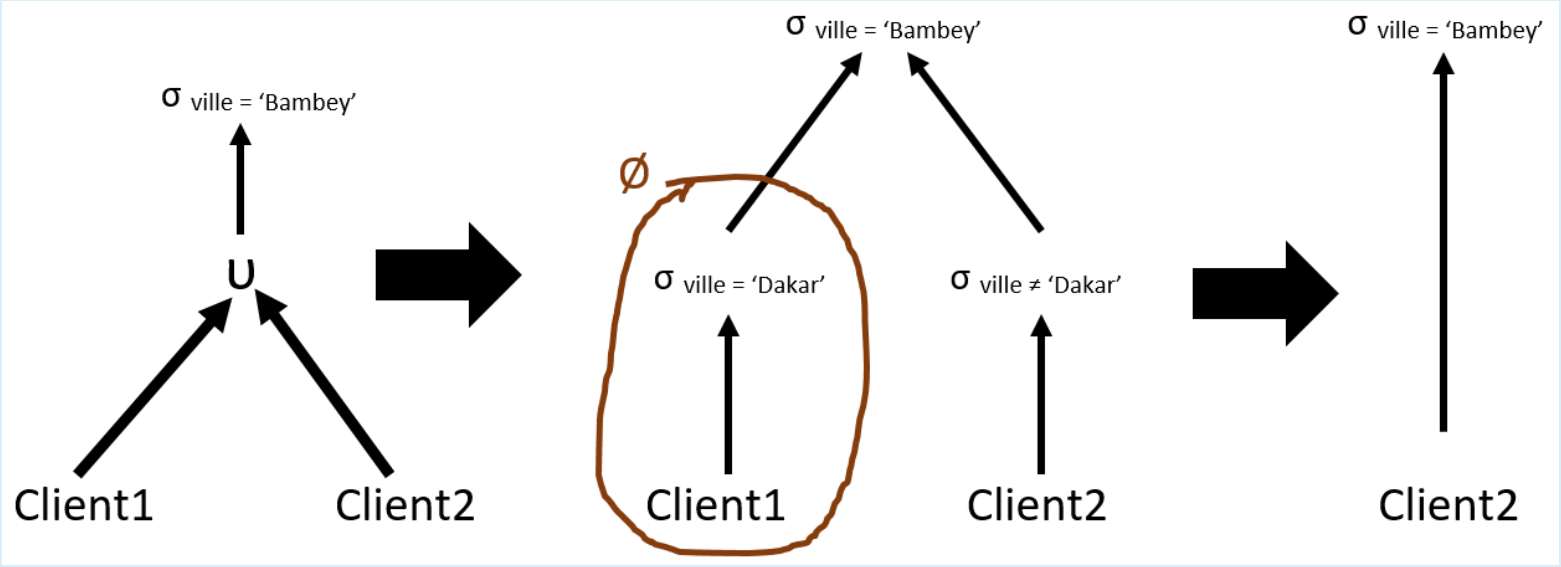
Requêtes réparties

Réduction pour la fragmentation horizontale

- ☞ Règle : éliminer l'accès aux fragments inutiles
- ☞ Requête : $\sigma_{ville='Bambey'}$

$$\left\{ \begin{array}{l} Client1 = \sigma_{ville='Dakar'} Client \\ Client1 = \sigma_{ville\neq'Dakar'} Client \end{array} \right\} Client = Client1 \bigcup Client2$$

Réduction pour la fragmentation horizontale

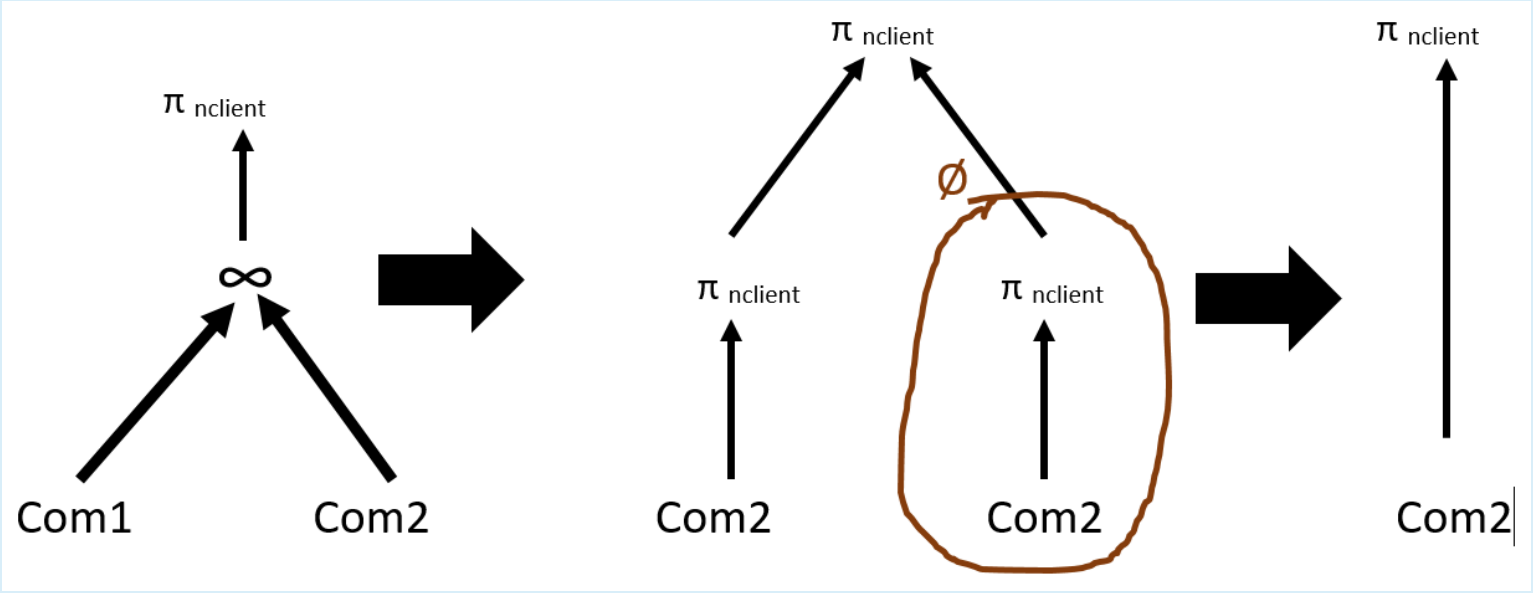


Réduction pour la fragmentation verticale

- ☞ Règle : éliminer les accès aux relations de base qui n'ont pas d'attributs utiles pour le résultat final
- ☞ Requête : **Select distinct nclient from Com**

$$\left\{ \begin{array}{l} Com1 = \prod_{ncde,nclient} Com \\ Com2 = \prod_{ncde,produit,qt} Com \end{array} \right\} Com = Com1 \bowtie Com2$$

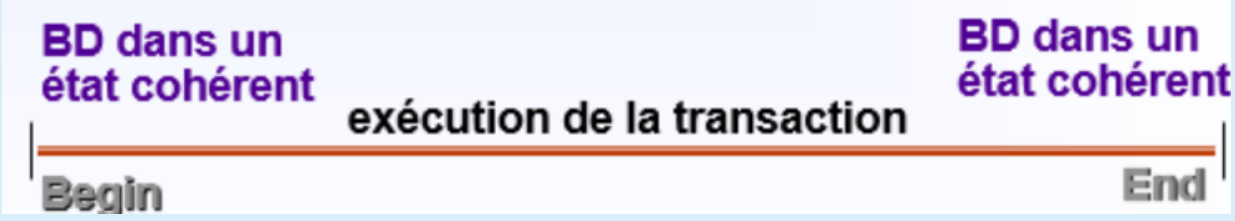
Réduction pour la fragmentation verticale



Concept de transaction

Une transaction est une collection d'actions qui transforment la BD (ou des fichiers) depuis un état cohérent en un autre état cohérent

- ☞ BD cohérente (garantie par le système)
- ☞ transaction cohérente (garantie par le programmeur)



Concept de transaction

- ☞ Réduire la cde n10 de 5 unités et les reporter à la cde n12

Transaction Report-qté
begin

exec sql UPDATE Cde

SET qte = qté - 5
WHERE ncde = 10 ;

exec sql UPDATE Cde

SET qté = qté + 5
WHERE ncde = 12 ;

exec sql COMMIT WORK ;

end.

Transactions réparties

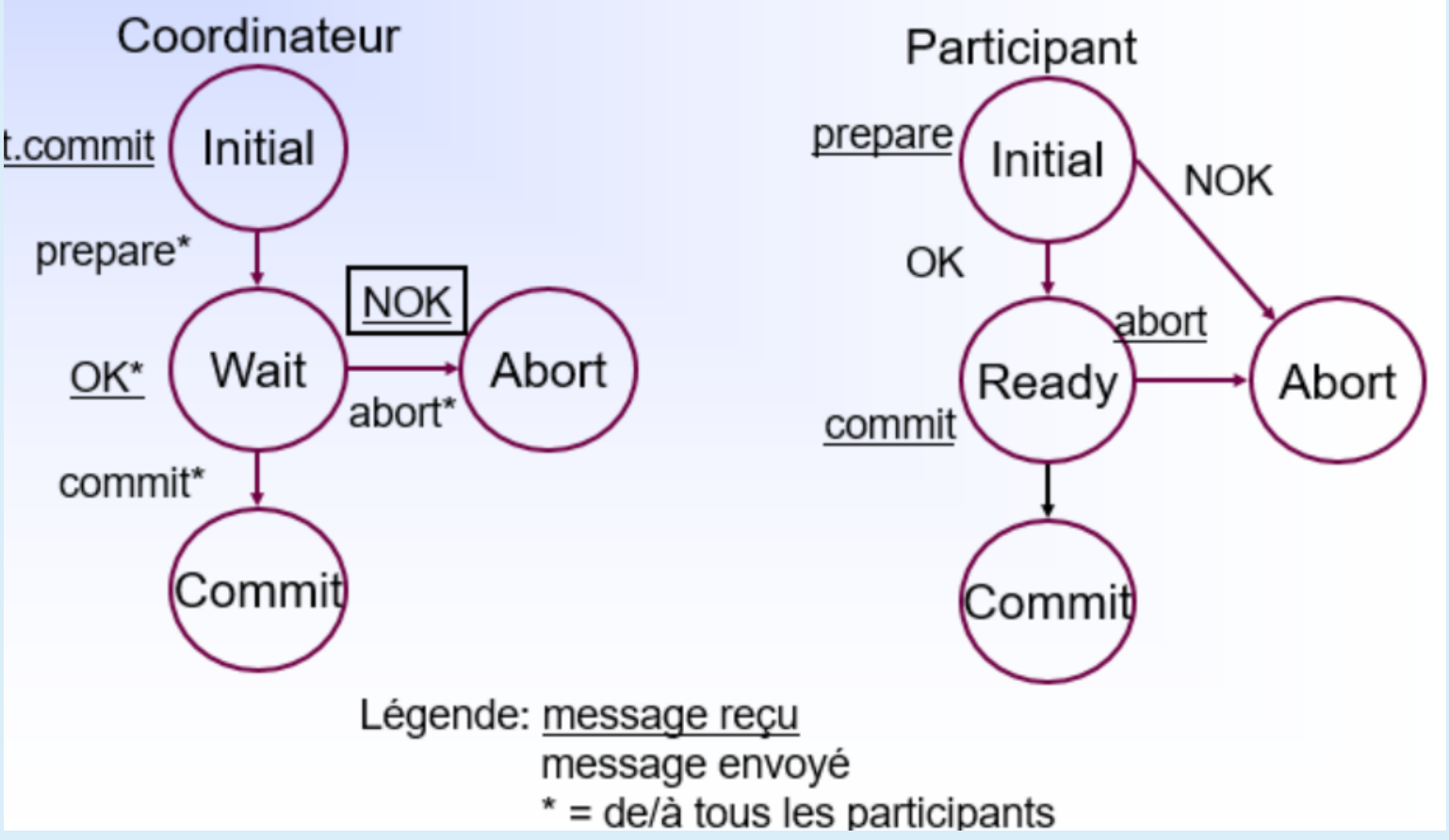
Propriétés des transactions (ACIDité)

- ATOMICITE
  - tout (COMMIT) ou rien (ABORT)
- COHERENCE
  - les contraintes d'intégrité sont respectées
- ISOLATION
  - les mises à jour concurrentes sont invisibles
- DURABILITE
  - les mises à jour validées persistent

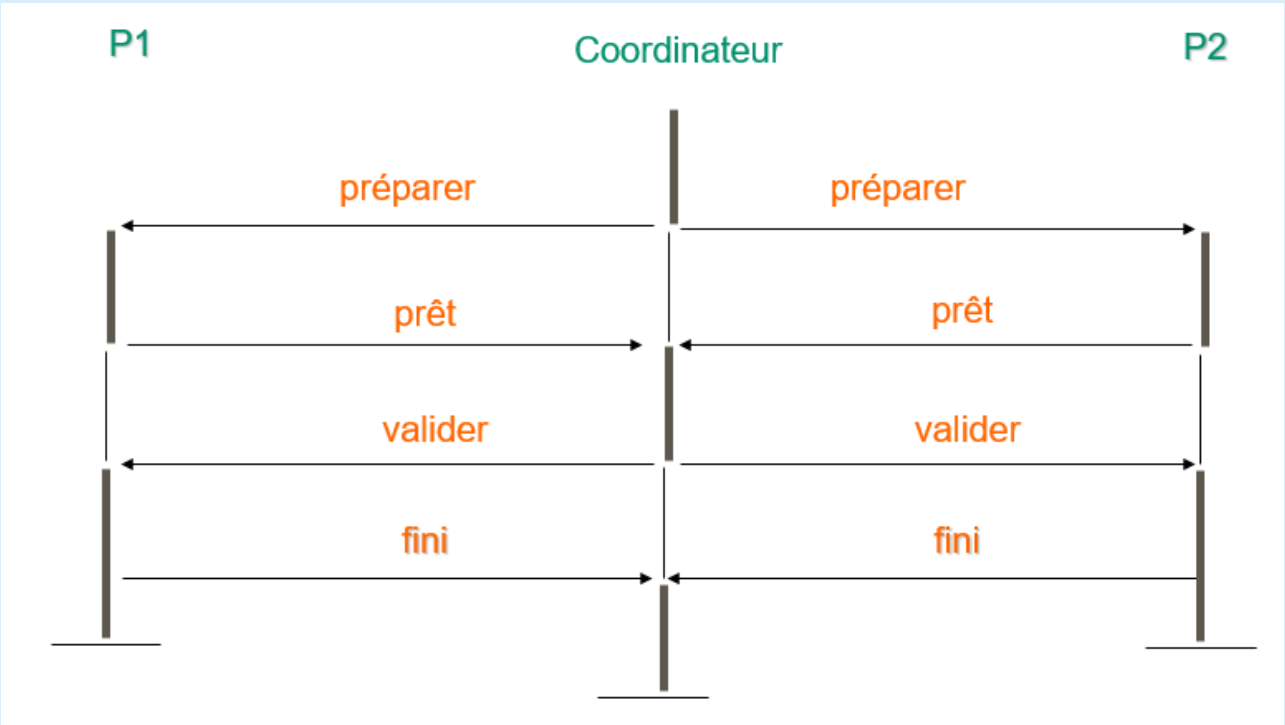
Protocole de validation en 2 étapes

- **Objectif** : Exécuter la commande COMMIT pour une transaction répartie
  - **Phase 1** : Préparer à écrire les résultats des mises-à-jour dans la BD
  - **Phase 2** : Ecrire ces résultats dans la BD
- Coordinateur : composant système d'un site qui applique le protocole
- Participant : composant système d'un autre site qui participe dans l'exécution de la transaction

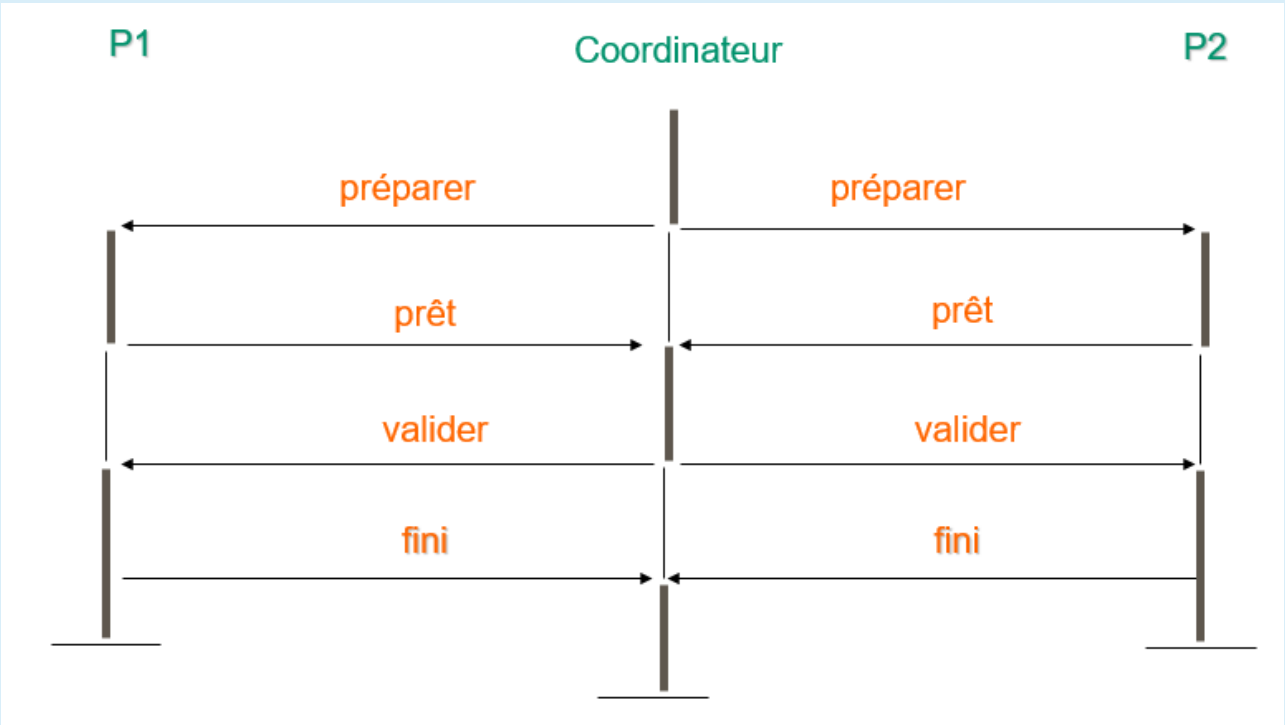
Protocole 2PC : messages échangés



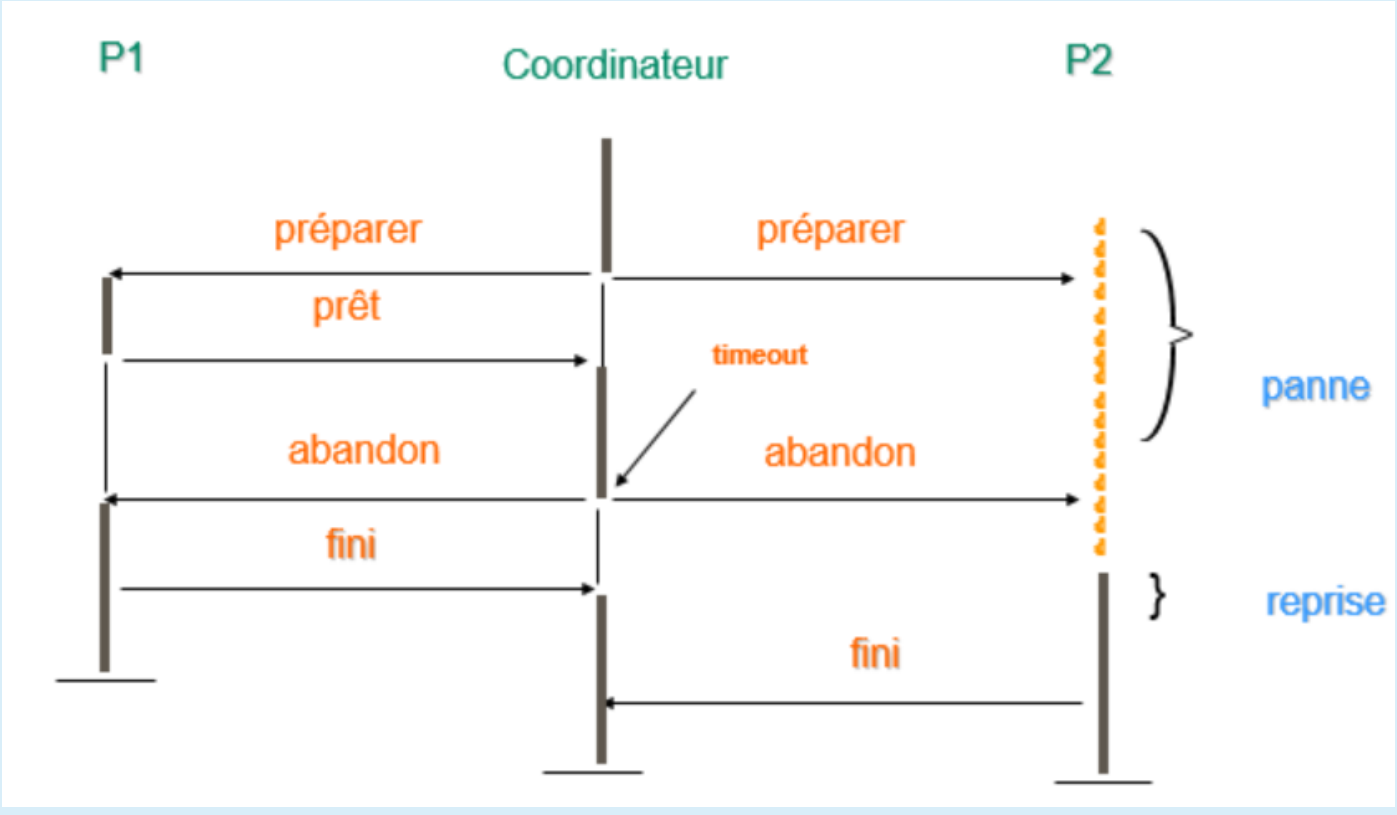
Validation normale



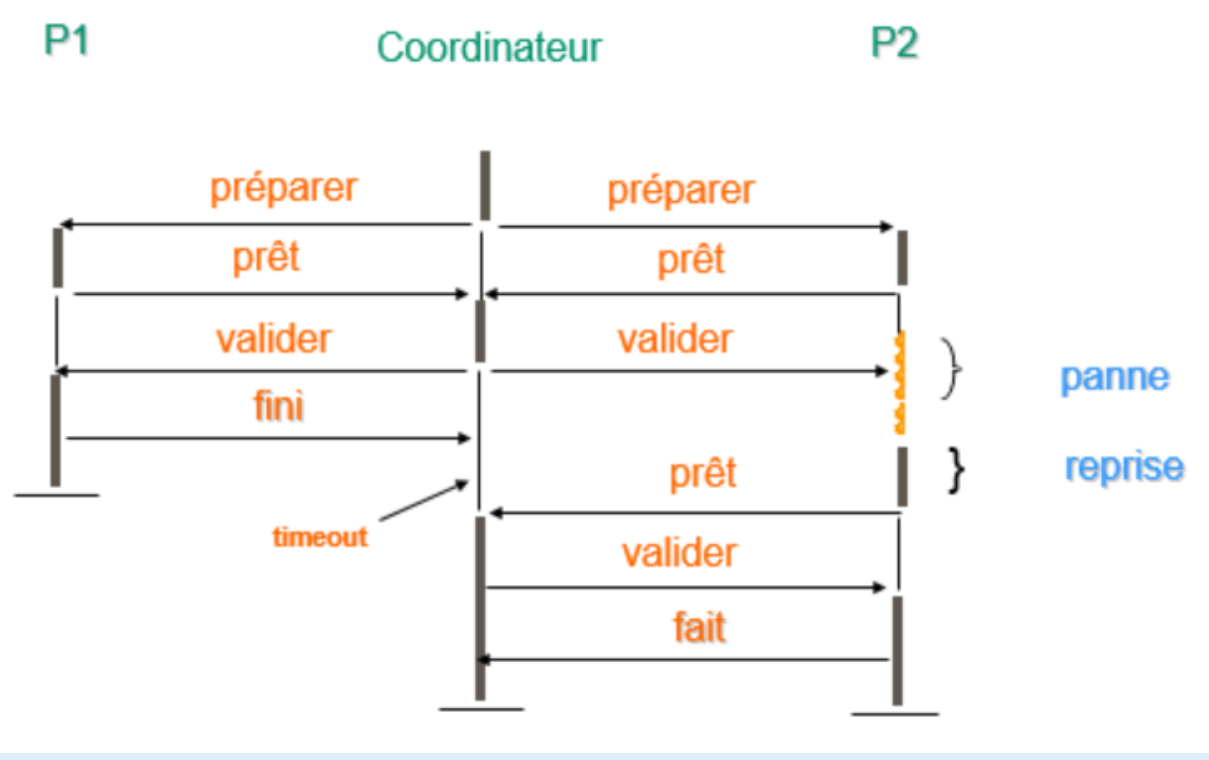
Validation normale



Panne d'un participant avant Prêt



Panne d'un participant après Prêt



Panne du coordinateur