le cnam

Bases de données distribuées et fédérées et Transactionnel Réparti

Plan

- Introduction
  - Contexte et définition
  - Exemple
  - Définitions complémentaires
  - Avantages et inconvénients
- Objectifs techniques des SGBD distribués
  - Multi clients multiserveurs
  - Transparence à la localisation des données
  - Meilleure disponibilité
  - Autonomie locale
  - Accès intégré à des données hétérogènes
- Architecture des SGBD distribués
  - Organisation des schémas
  - Architecture de référence
- Niveaux de transparence à la localisation
  - Client/Multibases :
    RDA, DRDA, SQL-CLI, UDA/ODBC
  - Vues distribuées
  - SGBD distribués

Plan

- Conception des SGBD distribués
  - Conception ascendante
    - Fédération de BD
  - Conception descendante
    - Techniques de fragmentation
    - Allocation des fragments
    - Allocation optimale
- Réplication dans les bases de données distribuées
  - Réplication : objectifs et problèmes
  - Réplication symétrique/asymétrique
  - Techniques de diffusion de mises à jour
- Optimisation des requêtes distribuées
  - Fonction de coût à optimiser
  - Schéma général de traitement et d'optimisation d'une requête distribuée
  - Exemple
- Un aperçu sur les SGBD du commerce
  - Un peu d'histoire : Ingres/STAR
  - IBM DB2, Informix, Oracle, Sybase
- Évaluation des SGBD distribués
- Bibliographie

Contexte

Contexte

Technologique :
Réseautique, des technologies de la communication de données, incarnées par l'Internet, l'informatique mobile, le « sans fil » et les périphériques intelligents

Organisationnel : décentralisation des activités

Financier : « downsizing » des matériels et logiciels

Bases de données distribuées

Une *base de données distribuée* permet de rassembler des ensembles de données plus au moins hétérogènes disséminés dans un réseau d'ordinateur sous forme d'une base de données globale homogène et intégrée

Définition et Exemple

Définition : BD DISTRIBUEE

ensemble de base de données logiquement reliées entre elles, gérées par des sites différents et apparaissant à l'utilisateur comme une base unique

Répartition possible de données

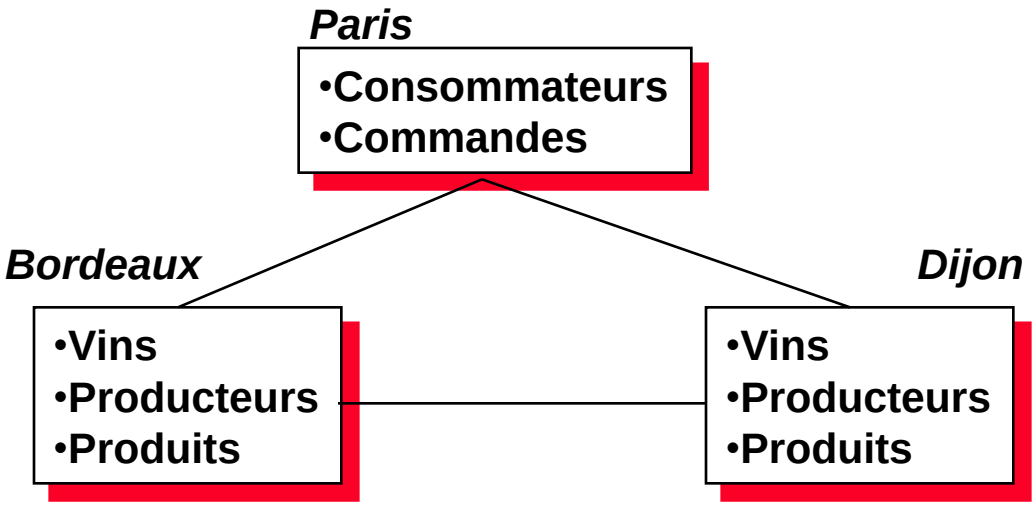


Schéma global

BUVEURS(NB, nom, prénom, ville)

COMANDES(NB,NV,date,qté)

VINS(NV,cru,année,degré)

PRODUCTEURS(NP,nom,région)

PRODUIT(NV,NP,qté)

schéma global
relationnel

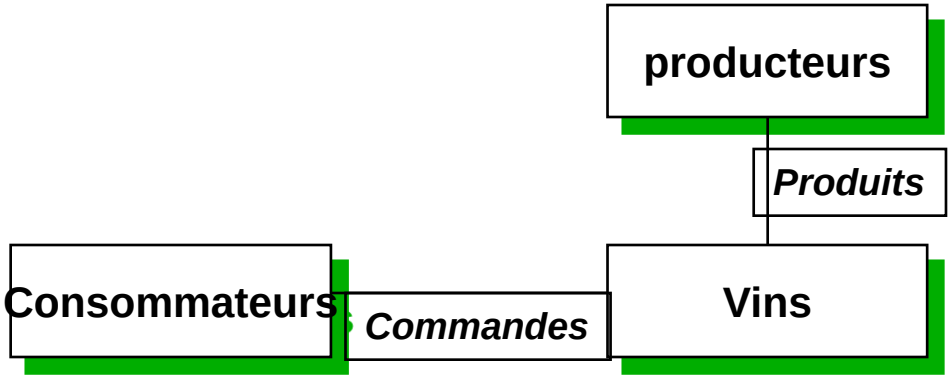


Schéma global
(entité association)

- schéma global :
  - définition de l'ensemble des types de données de la base
  - pas forcément matérialisé : chaque base locale implémente une partie, ces parties sont les seules matérialisée sur le disque

SGBD distribué

Définition :

SGBD distribué (Distributed DBMS) ou SGBD réparti : Système gérant une collection de BD logiquement reliées, distribuées sur différents sites en fournissant un moyen d'accès rendant la distribution transparente

Objectifs

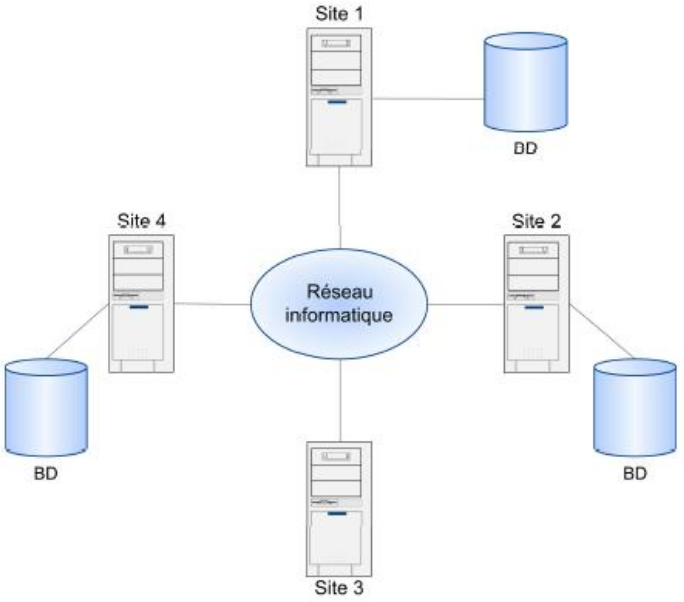
- Rend la répartition (ou distribution) transparente
  - Définition des données réparties : Cohérence des données

---> dictionnaire des données réparties

  - traitement des requêtes réparties

---> Requête distribuée : Requête émise par u client dont l'exécution nécessite l'exécutio de n sous

- requêtes sur n serveur (n > 1)
- gestion de transactions réparties
- gestion de la cohérence et de la sécurité
- Autonomie locale des sites
- Support de l'hétérogénéité



Topologie d'un système de gestion de base de Données distribuée

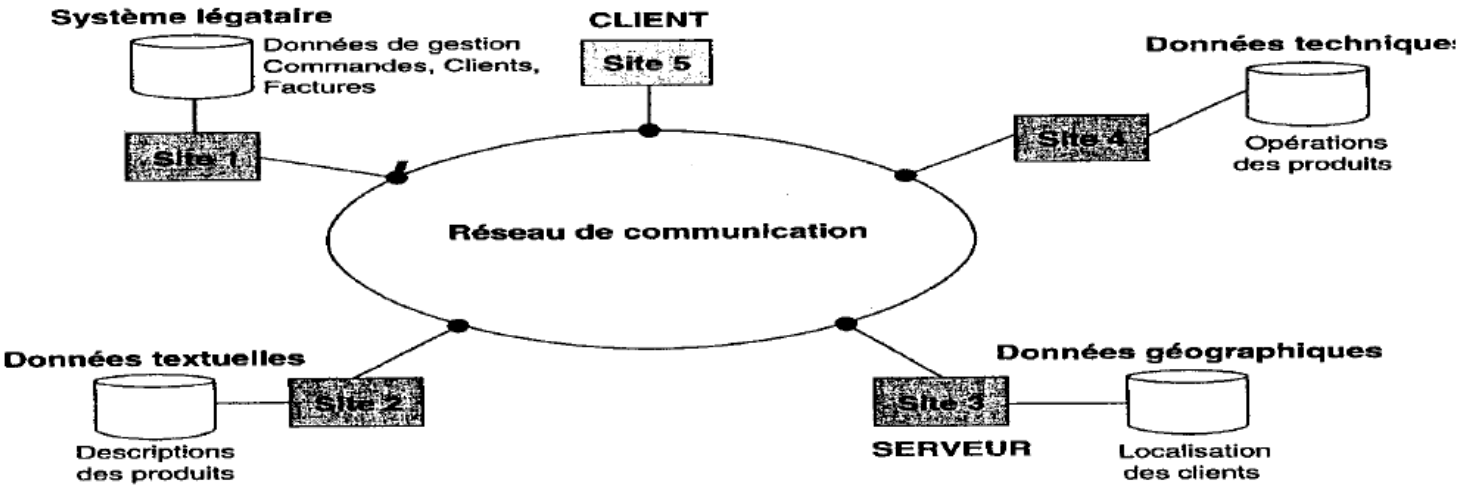
Note : SGBD distribué ou réparti sont des termes que nous considèrerons comme équivalents. On utilisera ici de préférence distribué.

Définitions complémentaires

- **BD Interopérables**
  - BD capable d'échanger des données en comprenant mutuellement ce qu'elles représentent
- **Base de donnée fédérée** - *a priori* hétérogène (Federated BD)
  - Plusieurs BD hétérogènes capables d'interopérer *via* une vue commune (modèle commun)
- **Multibase**
  - Plusieurs BD (hétérogènes ou non) capables d'interopérer sans une vue commune (absence de modèle commun)

Exemple de base de données fédérées

Exemple



Commentaires

- Système légataire : stocke les données de gestion de l'entreprise,
- Bases de données techniques : Cette base décrit les produits fabriqués et leurs composants, base de données textuelles : contient par exemple les manuels d'opérations.
- bases de données géographiques : Cette BD décrit la localisation des usines et des clients.

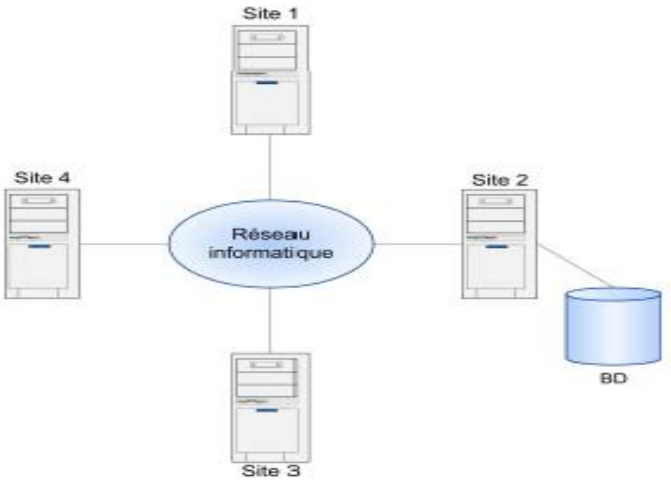
Le problème qui se pose est de faire collaborer toutes ces bases construites avec des systèmes, des modèles et des langages différents. Selon que l'on fournit seulement un langage d'accès commun, ou que l'on donne vision globale unifiée, on parlera de **multibase** ou de **base fédérée**

Définitions complémentaires

Traitement distribué

Il est essentiel de distinguer un SGBD distribué du traitement distribué

Traitement distribué : Une base de données centralisée accessible par le biais d'un réseau informatique.



SGBD parallèles: Nous marquons également une distinction entre un SGBD distribué et un SGBD parallèle.

SGBD parallèle: Un SGBD fonctionnant sur plusieurs processeurs et plusieurs disques, conçu pour exécuter des opérations autant en parallèle que possible, de façon à améliorer les performances.
Les SGBD parallèles associent plusieurs machines plus modestes pour atteindre le même débit qu'une seule machine plus ambitieuse, offrant en outre une meilleure évolutivité et une meilleure fiabilité que les SGBD monoprocesseurs.

Avantages des BD Distribuées

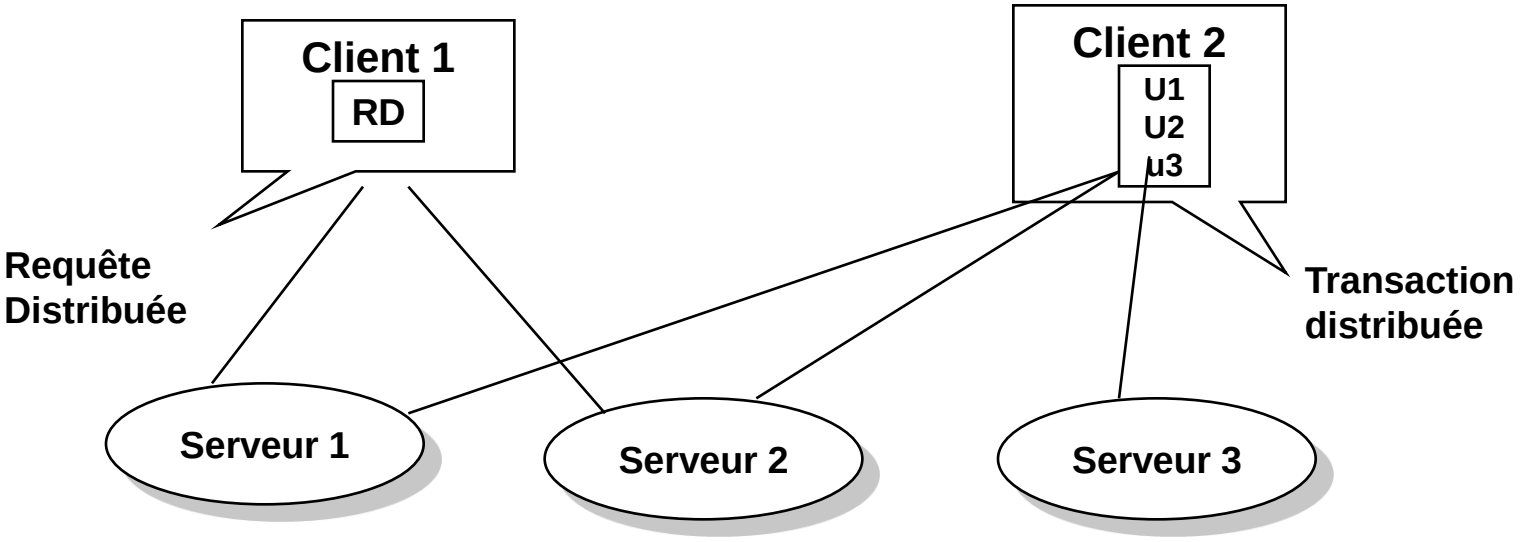
- **Reflète une structure organisationnelle** :Amélioration du partage et de l'autonomie locale
- **Disponibilité améliorée**: une panne dans un site d'un SGBDD ou une rupture de ligne de communication isolant un ou quelques sites n'immobilise pas l'ensemble du système
- **Performances améliorées:** les données se trouvent près du site de la plus grande demande/la concurrence pour les services de l'unité centrale et des entrées sorties se trouve nettement réduite par rapport à un SGBD centralisé.
- **Économie** : l'ajout de stations de travail à un réseau est nettement moins coûteux que l'extension d'un gros système centralisé.
- **Facilité d'accroissement (scalability)**

Inconvénients des BD Distribuées

- **Complexité**: un SGBDD masque sa nature répartie aux yeux des utilisateurs et fournit un niveau acceptable de performances, de fiabilité et de disponibilité
- **Sécurité**: Dans un système centralisé, l'accès aux données est d'un contrôle facile, tandis que dans un système distribué, non seulement il faut contrôler l'accès aux données dupliquées dans des emplacements multiples, mais le réseau lui-même doit être sécurisé.
- **Contrôle d'intégrité de données plus difficile**: L'intégrité de base de données fait référence à la validité et à la cohérence des données stockées.
- **Complexité plus grande du design de bases de données** : le design d'une base de données distribuée doit prendre en considération la fragmentation des données, l'allocation des fragments à des sites spécifiques et la réplication des données.
- **Coût** : la distribution entraîne des coûts supplémentaires en terme de communication, et en gestion des communications (hardware et software à installer pour gérer les communications et la distribution).

Objectifs techniques des SGBD distribués

Multiclients multiserveurs



Fonctionnement multiclients multiserveurs

Requête distribuée : requête émise par un client dont l'exécution necessite l'exécution de N sous requêtes sur N serveurs avec N>1

Transaction distribuée: transaction mettant en oeuvre plusieurs serveurs

Objectifs techniques des SGBD distribués

Transparence à la localisation des données (*location transparency*)

Propriété d'un SGBD distribué permettant d'écrire des requêtes avec des noms d'objets référencés(les tables en relationnel) ne contenant pas la localisation des données

Exemple: **requête « noms et prénoms des buveurs parisiens ayant passé une commande de Volnay de degré >12 en quantité supérieure à 100 depuis le 1 JANVIER 1992**

Select Nom, prénom

From BUVEURS B, VINS V, COMMANDES C

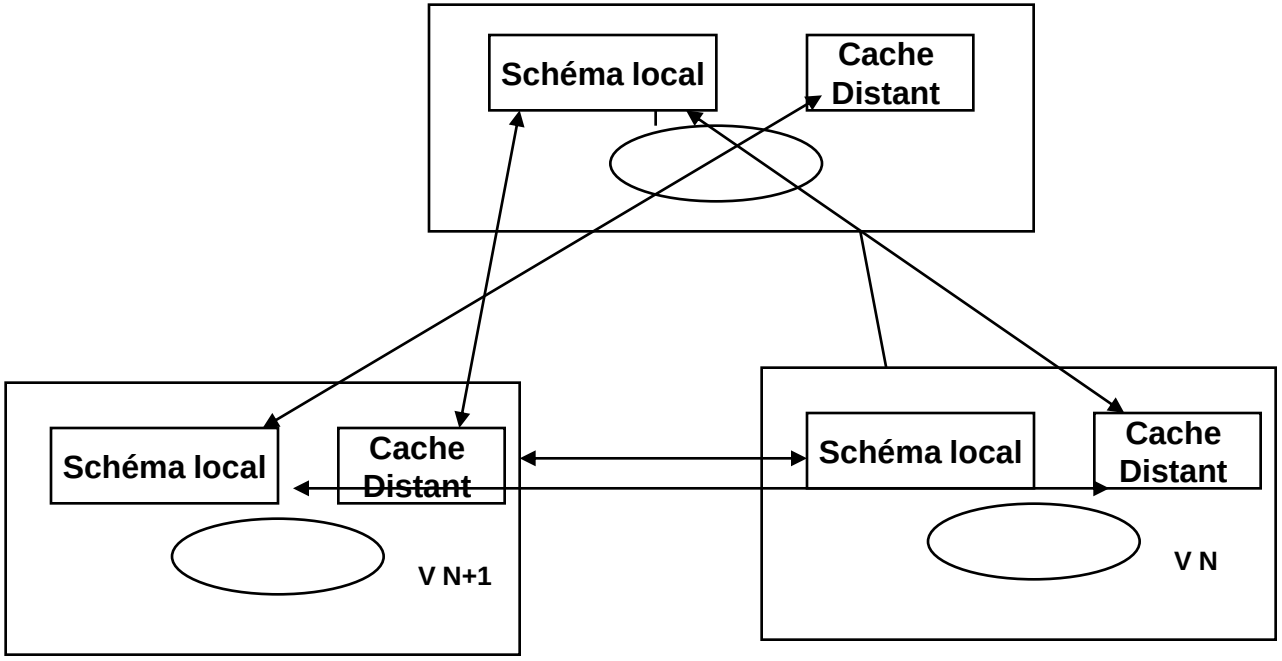
WHERE V.CRU= 'volnay' AND V.DEGRE>12 AND C.QTE>100 and C.NV=V.NV and C.DATE> 1/10/92 AND B.NB=C.NB

And B.VILLE='PARIS'

- **Même requête quelque soit la localisation des tables**
- **SGBDR recherche les sites capables de générer des éléments de réponse à une requête**

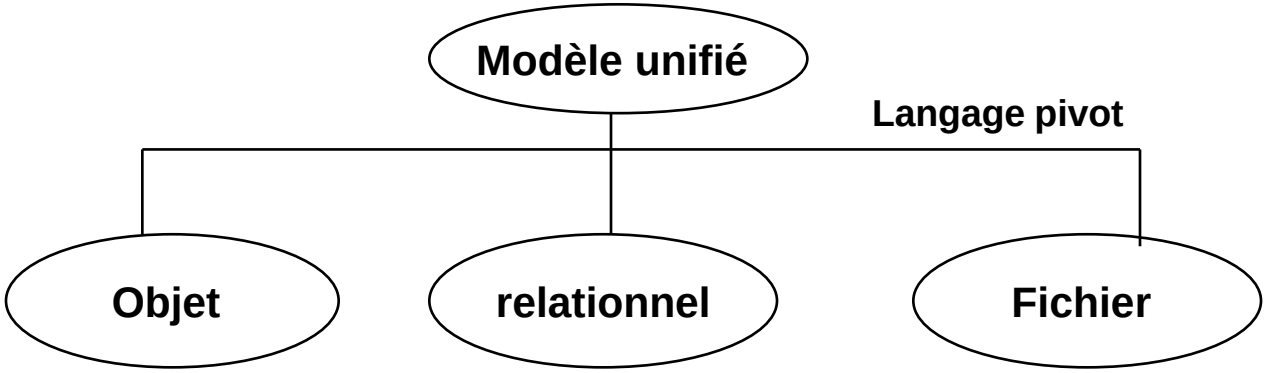
Objectifs techniques des SGBD distribués

- **Meilleure disponibilité**
  - Ne plus dépendre d'un site central unique
  - Gestion de copies : se replier sur une copie quand un site tombe en panne (notion de réplication)
- **Autonomie locale**



Objectifs techniques des SGBD distribués

- **Accès intégré à des données hétérogènes**



Unification des modèles et des langages

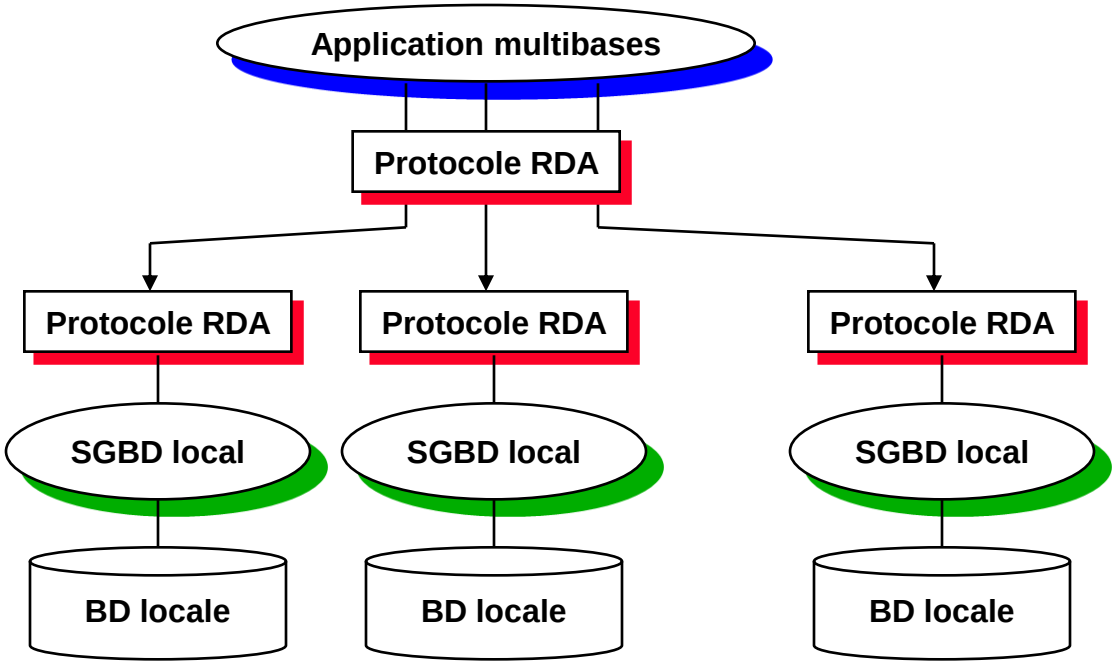
Intégration de schémas (*schema integration*):
Procédé consistant à rapprocher des schémas sémantiquement différents afin de construire un schéma unique équivalent

Niveaux de transparence à la localisation

- **Trois types d'accès :**
  - **Client / Multibases :**
    - RDA (Remote Data Access) - Standard ISO
    - DRDA (Distributed Relational Database Architecture) d'IBM
    - SQL-CLI (Call Level Interface) de l'Open Group
    - ODBC (Open Database Connectivity) de Microsoft
    - UDA Microsoft
    - JDBC (Java Database Connection) de SUN
  - **Vues distribuées (sur BD fédérées)**
  - **SGBD distribué**

RDA - Remote Data Access

- Les usagers connaissent la localisation
- Si une jointure est nécessaire, elle doit être réalisée par l'application



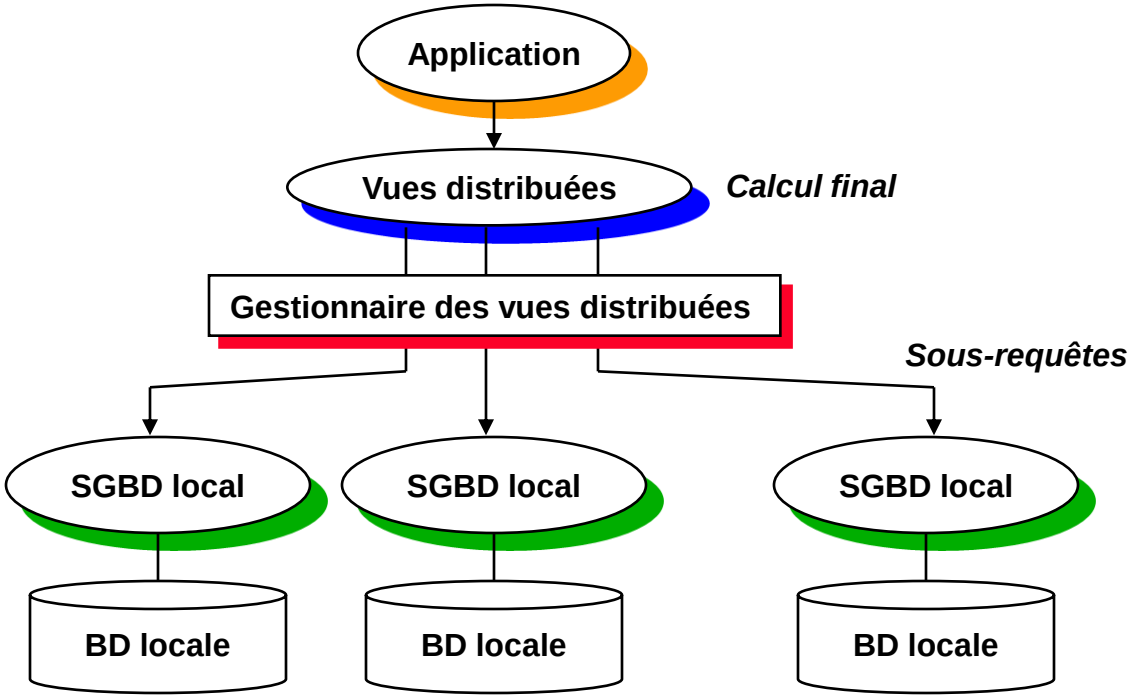
RDA - Remote Data Access (2)

- **Exemple**
  - Site 1 : Cartes grises
    Personne (N° personne, nom, prénom, adresse, …)
    Voiture (N° véhicule, marque, type, …)
    Conducteur (N° personne, N° véhicule, NB_accidents,..)
  - Site 2 : Base SAMU
    Accident (N° accident, date, departement, N° véhicule, N° personne, …)
    Blessé (N° accident, N° personne, gravité, ….)
  - Site 3 : requête
    Liste des blessés graves dans une voiture de marque xxx et de type yyy dans région parisienne
    - Requête en centralisé :
      - SELECT P.nom,P.prénom FROM Personne P, Blessé B, Accident A, Voiture V
        WHERE P.N° personne = B.N° personne
        AND B.gravité > « commotion »
        AND B.N° accident = A.N° accident
        AND A.N° véhicule = V.N° véhicule
        AND V. marque = « xxx »
        AND V. type = « yyy»
        AND A.département IN (75, 78 , 91, 92 ,93, 94, 95)
- **Solution RDA**
  - Requête sur site 1 : SELECT N° véhicule FROM Voiture WHERE marque = « xxx » AND type = « yyy » INTO temp1
  - Requête sur site 1 : SELECT * FROM Personne INTO temp2
  - Requête sur site 2 : SELECT B.N° personne, A.N° véhicule FROM Blessé B, Accident A
    WHERE B.gravité > « commotion » AND B.N° accident = A.N° accident
    AND A.département IN (75, 78 , 91, 92 ,93, 94, 95) INTO temp3
- **Conclusion**
  - Il est nécessaire d'envoyer 3 requêtes pour seulement 2 sites
  - La totalité de la relation Personne doit être transférée
  - L'intégration du résultat final doit être faite par l'application :

    SELECT P.nom, P.prénom FROM temp2 P, temp3 B, temp1 V
    WHERE P.N° personne = B.N° personne
    AND B.N° véhicule = V.N° véhicule
- **Difficultés : Programmation et adhésion de l'industrie des SGBD au protocole RDA**

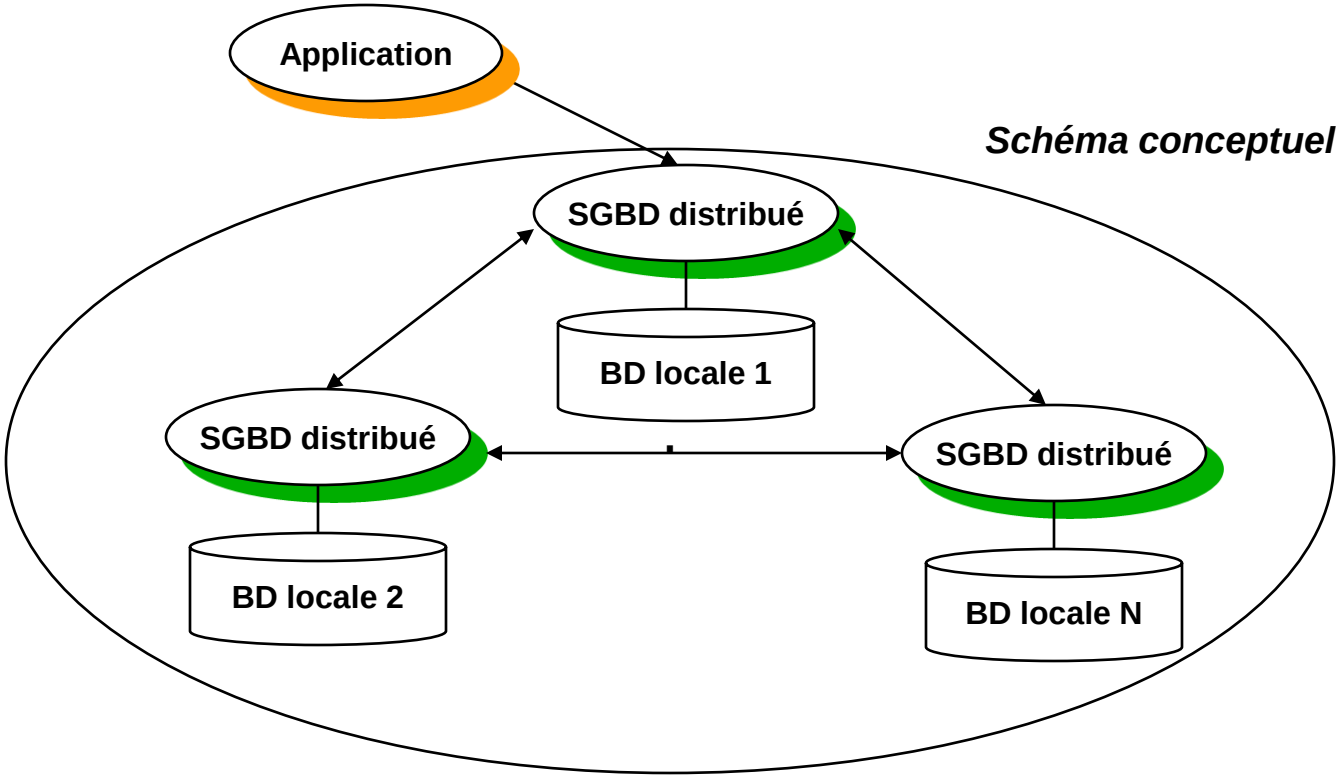
Vues distribuées

- La transparence à la localisation est assurée par la définition des vues distribuées
- Les jointures inter-bases sont exécutées par le système
- Les mises à jour sont supportées au moyen des vues distribuées
- Un protocole de validation à 2 phases est supporté



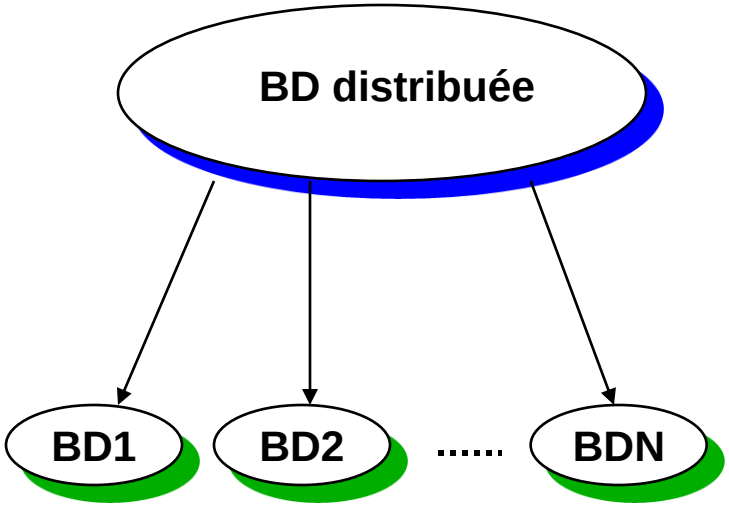
SGBD distribué

- La transparence à la localisation est assurée par la définition de la base distribuée
- Les différentes opérations sont prises en charge par les différents SGBD
- Un protocole de validation à 2 phases est supporté



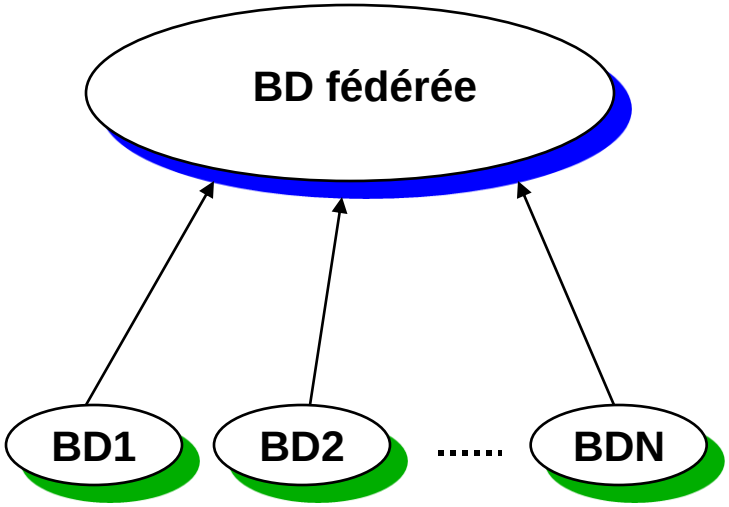
Méthodes de conception

Approche descendante



- Maîtrise de la complexité de la distribution (fragmentation, duplication, placement)
- Définition des schémas locaux à partir du schéma global

Approche ascendante



- Maîtrise de l'hétérogénéité sémantique (BD) et syntaxique (SGBD, communications,….)
- Maîtrise de l'intégration des schémas locaux pour créer un schéma global

Méthode ascendante

⇨ **Fédération** : Création d'un schéma unique partant de plusieurs schémas

- **Objectifs :**
  - **Donner aux utilisateurs une vue unique des données implémentées sur plusieurs systèmes *a priori* hétérogènes (plates-formes et SGBD)**
  - **Cas typique rencontré lors de la concentration d'entreprises : faire cohabiter les différents systèmes tout en leur permettant d'inter opérer**
  - **Procédure d'intégration :**
    - **Traitement de l'hétérogénéité sémantique**
    - **Traduction des schémas : traitement de l'hétérogénéité syntaxique**
    - **Intégration des schémas**

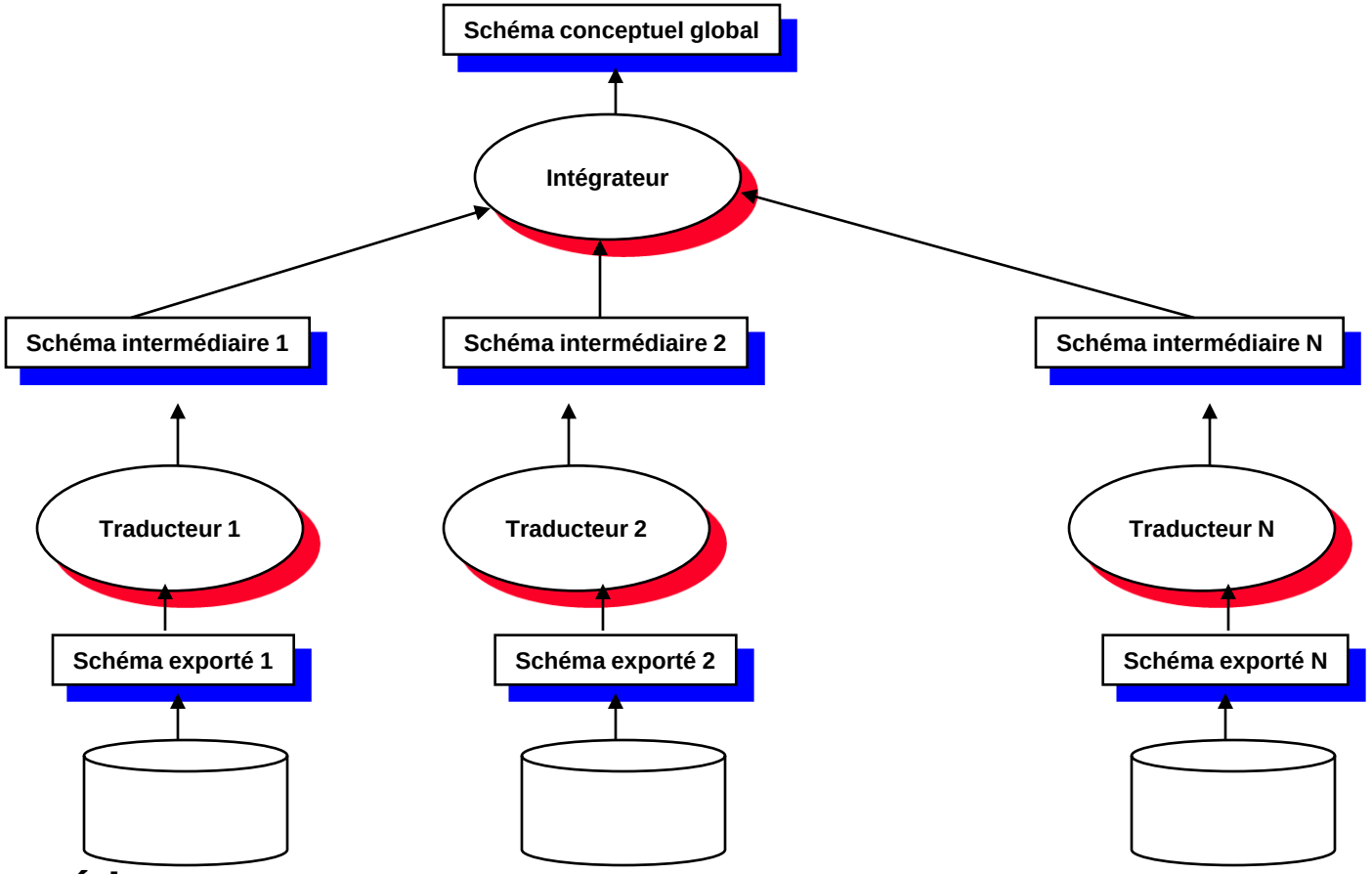
Fédération de BD

- Hétérogénéité sémantique
  - Origine : Résulte des conceptions indépendantes des différentes BD
  - Effet : Désaccord sur la signification des données
  - Solution : Analyse sémantique comparée des données préalable à la fédération souvent groupée avec la phase de traduction

- Traduction des schémas (résolution de l'hétérogénéité syntaxique)
  - Origine : utilisation de modèles différents dans les BD composantes
  - Effet : nécessite des traductions de tous les modèles vers tous les modèles
  - Solution : traduction de tous les schémas dans un modèle commun (dit canonique ou pivot)
  - Problématique :
    - Le modèle canonique doit avoir un pouvoir de modélisation ≥ à ceux des modèles des BD composantes
    - Nécessité de compléter sémantiquement des modèles de BD composantes qui seraient trop pauvres
  - Choix du modèle canonique :
    - Entité - Association et Relationnel
    - Objet

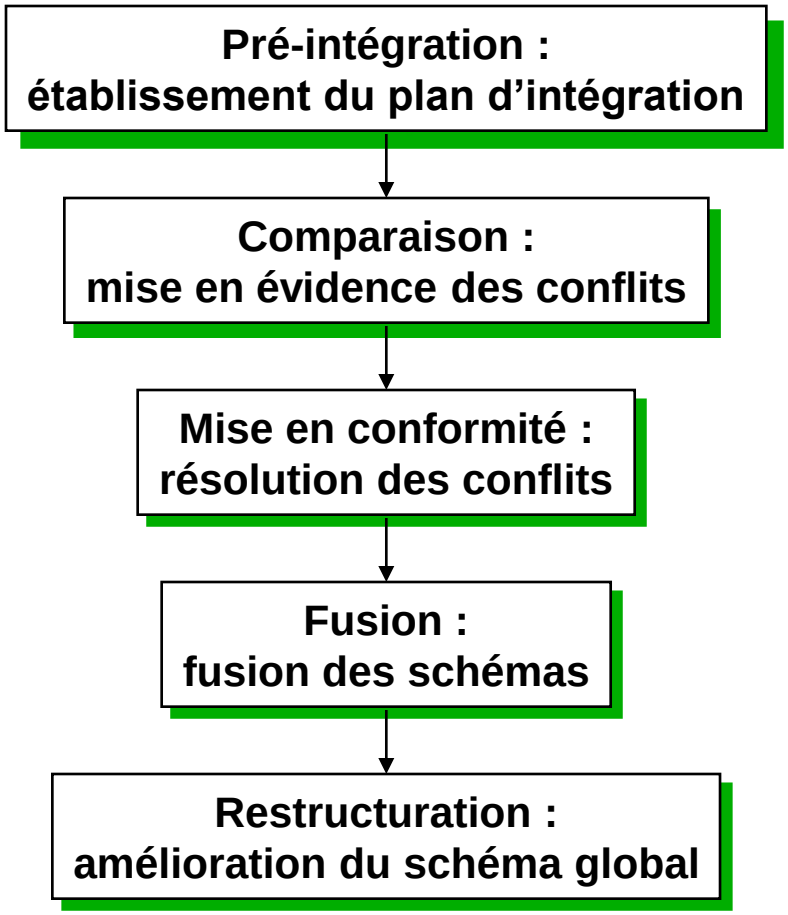
Fédération de BD

- **Intégration des schémas**

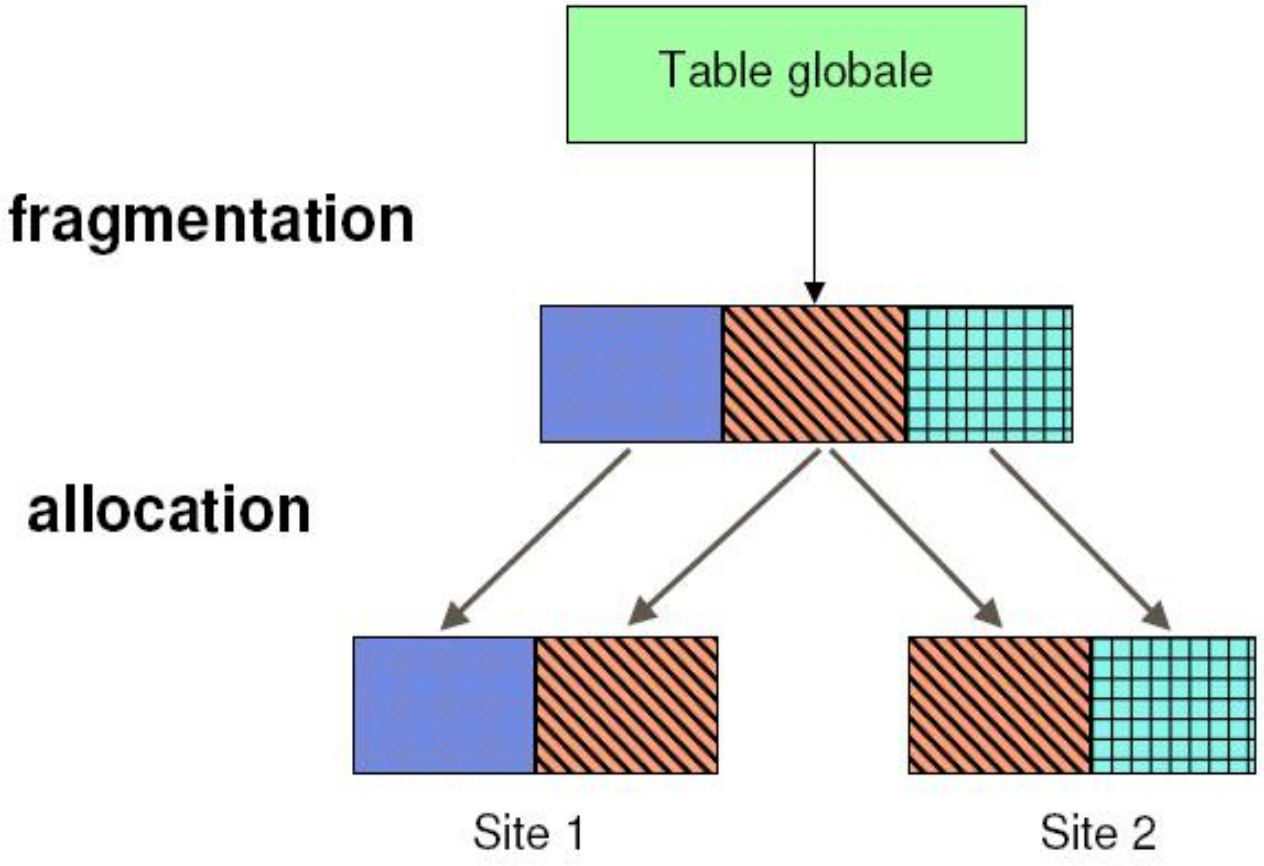


- **Procédure :**
  - **Identifier les éléments de base qui sont liés**
  - **Choisir la représentation la plus adéquate pour le schéma global**
  - **Intégrer les éléments des schémas intermédiaires**

Démarche d'intégration



Méthode descendante



Méthode descendante

- **La fragmentation**: Une relation est scindée en un certain nombre de sous-relations, appelées **fragments**, réparties ensuite. (exemple: la **fragmentation horizontale** et la **fragmentation verticale**. Les fragments horizontaux sont des sous-ensembles de tuples et les fragments verticaux sont des sous-ensembles d'attributs).
- **L'allocation**. Chaque fragment est stocké dans un site avec une répartition « optimale ».
- **La réplication**. Le SGBDD conserve une copie d'un même fragment dans plusieurs sites différents

Méthode descendante

- La définition et l'allocation des fragments dépendent nécessairement de la manière d'utiliser la base de données
- La conception de la BDD doit se fonder sur des informations tant quantitatives que qualitatives. Les informations quantitatives servent dans l'allocation, tandis que les informations qualitatives servent lors de la fragmentation.

Les informations quantitatives reprennent, notamment :

- La fréquence d'exécution d'une transaction.
- Le site à partir duquel une transaction est exécutée.
- Les critères de performance des transactions.

Les informations qualitatives peuvent inclure des détails relatifs aux transactions exécutées, tels que :

- Les relations, attributs et tuples qui reçoivent un accès.
- Le type d'accès (en lecture ou en écriture)

Fragmentation

- Intérêts
  - L'usage : dans le contexte de la répartition des données, il paraît approprié de travailler sur des sous-ensembles de relations, constituant l'unité de répartition
  - L'efficacité. Un stockage des données à proximité du lieu où elles sont le plus utilisées est essentiel et inversement, les données non nécessaires aux applications locales ne sont pas emmagasinées inutilement.
  - Le parallélisme: Lorsque le fragment constitue l'unité de répartition, une transaction peut être découpée en plusieurs sous-requêtes qui opèrent sur des fragments. Ceci a pour effet d'accroître le degré de simultanéité, c'est-à-dire le parallélisme, du point de vue du système dans sa totalité, ce qui permet aux transactions qui le peuvent, de s'exécuter en parallèle et en toute sécurité.
  - La sécurité. Les données qui ne sont pas indispensables aux applications locales ne sont pas présentes inutilement à des endroits à la portée des utilisateurs non autorisés

Fragmentation correcte

Trois règles régissent la fragmentation :

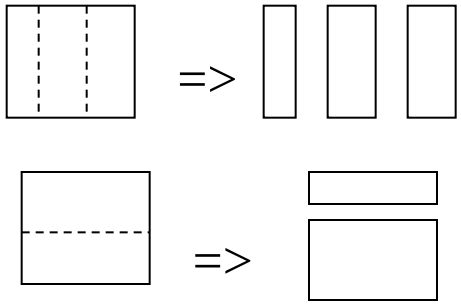
1) **L'aspect complet** . Si une instance de relation R est décomposée en fragments R1, R2, … R*n*, chaque donnée qui se trouve normalement dans R doit apparaître dans au moins un des fragments. Cette règle est indispensable pour interdire toute perte de donnée pendant la fragmentation.

2) **La reconstruction**. Il est toujours possible de définir une opération relationnelle qui permette de reconstruire la relation R à partir de ses fragments. Cette règle garantit la préservation des dépendances fonctionnelles.

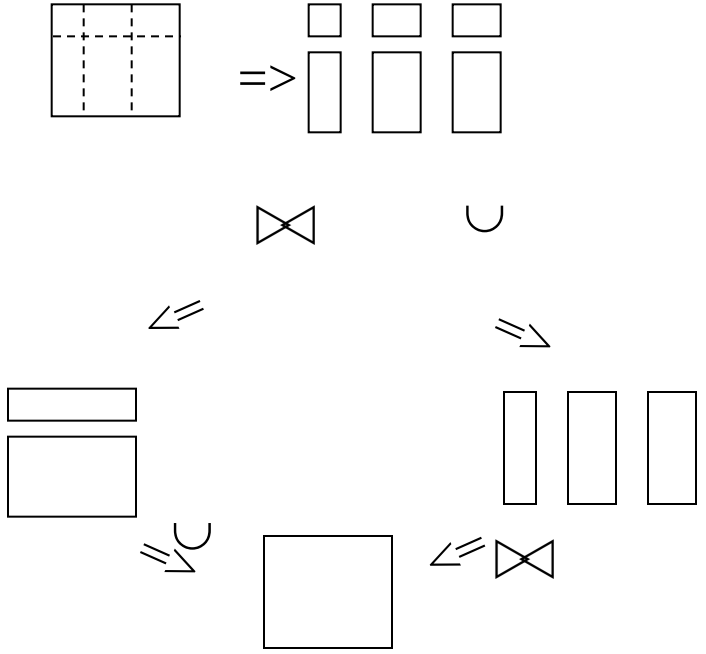
3) **La dis-jointure**. Si une donnée d*i* apparaît dans le fragment R*i*, alors il ne peut apparaître dans aucun autre fragment. La fragmentation verticale constitue une exception à cette règle, puisque les attributs de clé primaire doivent être répétés pour permettre la reconstruction. Cette règle garantit la redondance minimale des données.

Techniques de fragmentation

- **2 niveaux de fragmentation :**
  - **Schéma : fragmentation *verticale*,**
    Schéma découpé en sous-relations.
  - **Instance : fragmentation *horizontale*,**
    N-uplets d'une relation affectés à des fragments.

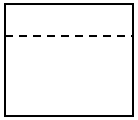


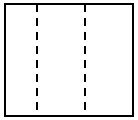
- **Ces 2 niveaux sont combinables: fragmentation mixte**
- **Recomposition :**
  - **Jointure(s)**
  - **Union(s)**

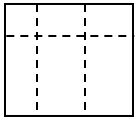


Unités de fragmentation

- Classe,
  - Table en relationnel
- Occurrence,
  - Fragmentation horizontale
    - Lien de localisation
- Attribut,
  - Fragmentation verticale
- Valeurs,
  - Combinaison des fragmentations horizontale et verticale







Répartition des classes

- Les relations (entières) sont disséminées sur les sites (sous-schémas).
- Exemple :
  - 3 relations : compte, client et agence,
  - 2 sites : site 1 et site 2,

  Compte et client sur site 1,

  Agence sur site 2.

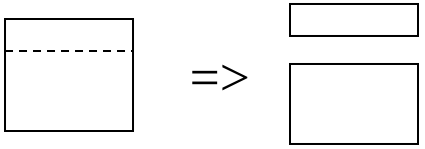
Fragmentation horizontale

- **Relations divisées en sous-relations, chacune contenant des n-uplets d'1 relation d'origine (tout n-uplet doit être dans un fragment),**

=> Fragmentation par une sélection,

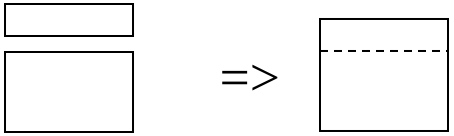
$$\text{Compte1} = \sigma_{(\text{TypeCompte} = \text{`courant'})}\text{Compte},$$

$$\text{Compte2} = \sigma_{(\text{TypeCompte} = \text{`dépôt'})}\text{Compte},$$



=> Recomposition par union,

$$\text{Compte} = \text{Compte1} \cup \text{Compte2},$$



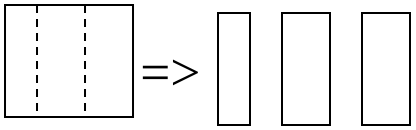
Fragmentation horizontale

Relation Compte

NoClient	Agence	TypeCompte	Somme	
174 723	**Lausanne**	**courant**	**123 345.89**	→ Site 1
177 498	**Genève**	**courant**	**34 564.00**	
201 639	**Lausanne**	**courant**	**45 102.50**	
201 639	**Lausanne**	**dépôt**	**325 100.00**	→ Site 2
203 446	**Genève**	**courant**	**274 882.95**	→ Site 1

Fragmentation verticale

Fragmentation verticale,



- Fragmentation par projection,

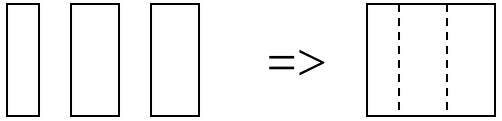
  Client1 = $\pi_{(NoClient,\ NomClient)}$Client

  Client2 = $\pi_{(NoClient,\ Prénom,Age)}$Client

- Recomposition par jointure,

  Client = Client1 $\bowtie$ Client2

- Attention à garder la clef (sinon perte d 'information).



Fragmentation verticale

Site 2
Site 1

Site 1

Site 2

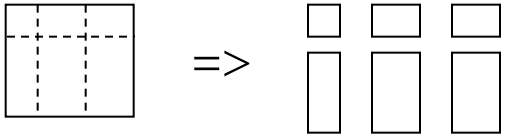
Relation Client

NoClient	NomClient	*Prénom*	*Age*
174 723	**Villard**	***Jean***	***29***
177 498	**Cattell**	***Blaise***	***38***
201 639	**Tsellis**	***Alan***	***51***
203 446	**Kowalsky**	***Vladimir***	***36***

Fragmentation mixte

- **combinaisons des précédentes : fragmentation horizontale ou verticale d'une relation ou d'un fragment obtenu antérieurement.**

⇒ **Sélections et projections.**



Exemple :

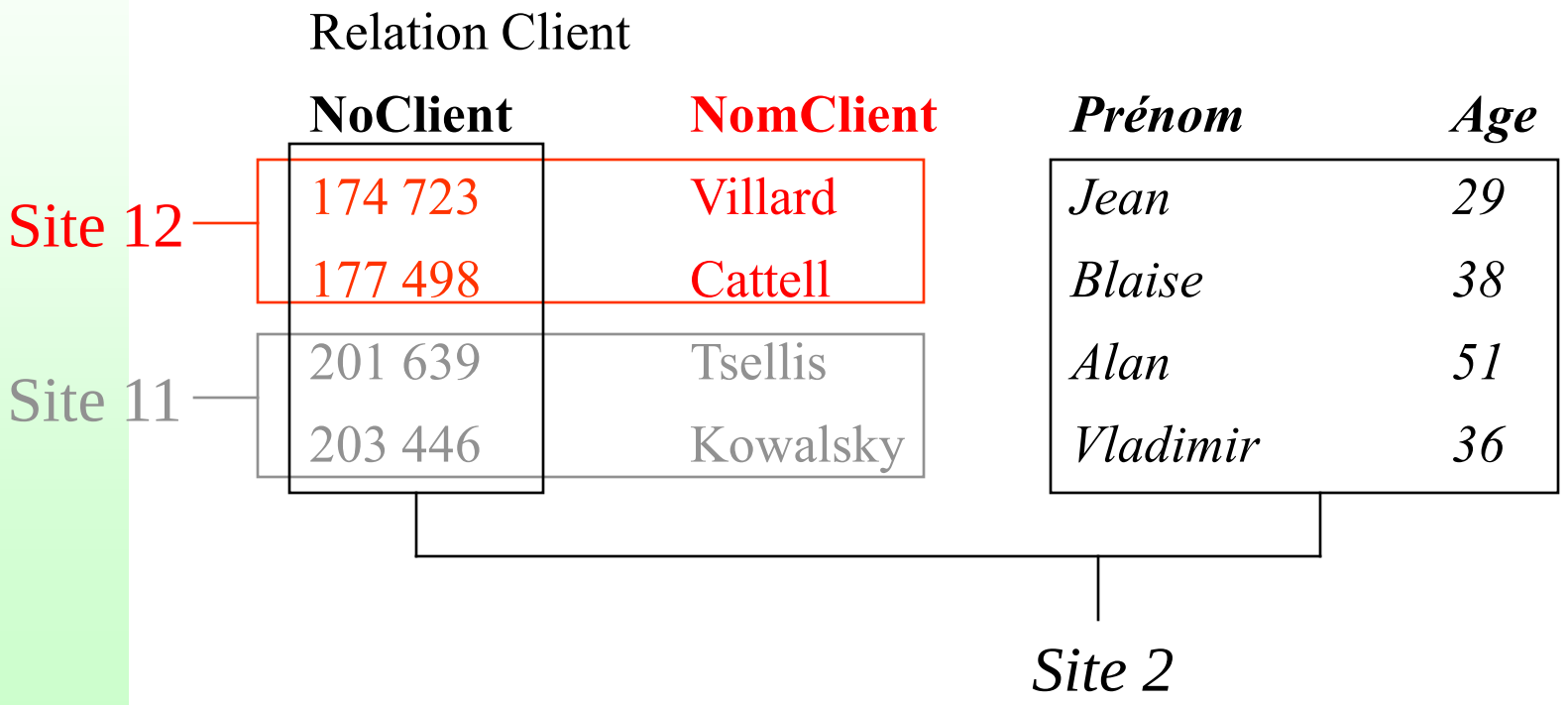
Compte11 = $\pi_{(NoClient, Agence)}$ $\sigma_{(NoClient > 200\ 000)}$Compte sur le site 11

Compte12 = $\pi_{(NoClient, Agence)}$ $\sigma_{(NoClient <= 200\ 000)}$Compte sur le site 12

Compte2 = $\pi_{(NoClient, Prénom, Age)}$Compte sur le site 2

Compte = (Compte11 U Compte12) ⋈ Compte2

Fragmentation mixte



Allocation de données

Quatre stratégies d'allocation de données

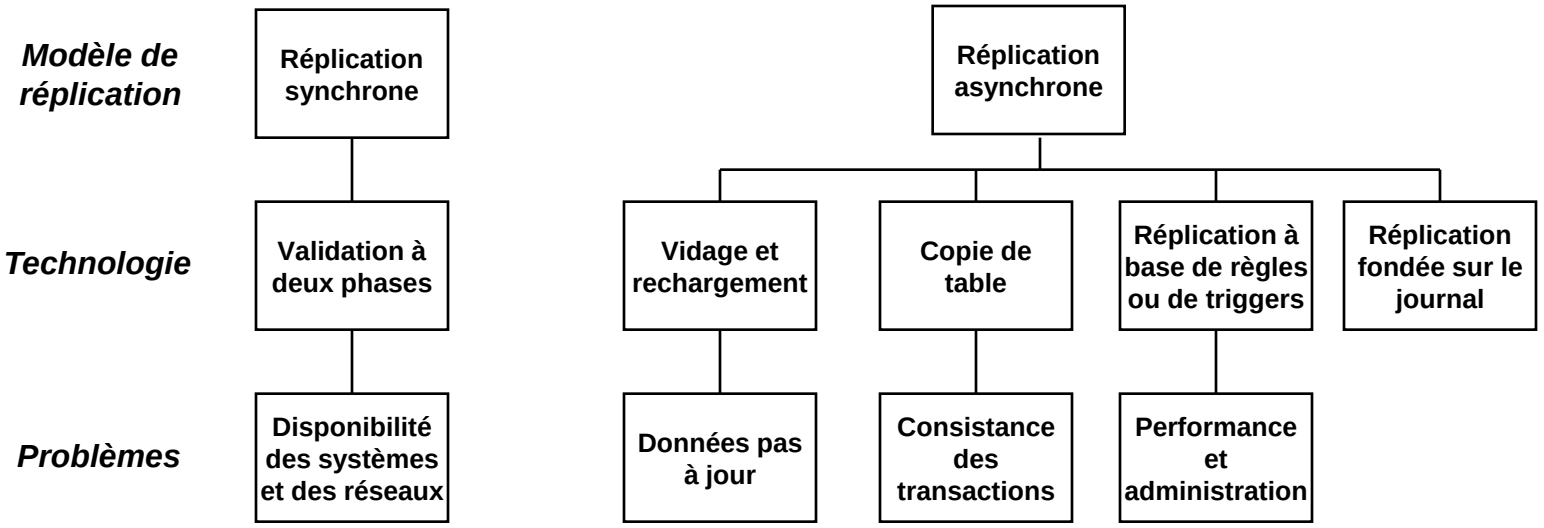
- **Allocation centralisée**: Cette stratégie consiste à établir une seule base de données et un seul SGBD sur un site et à répartir les utilisateurs sur le réseau. Nous avons qualifié précédemment cette approche de traitement centralisé.
- **Allocation fragmentée ou partitionnée** : Cette stratégie partitionne, découpe la base de données en des fragments disjoints attribués chacun à un seul site
- **Réplication complète**: Cette stratégie entretient une copie de la totalité de la base de données dans chaque site
- **Réplication sélective**: Cette stratégie combine la fragmentation, la réplication et la centralisation. Certaines données sont fragmentées pour obtenir la meilleure proximité des références et d'autres, utilisées par de nombreux sites et rarement modifiées, sont dupliquées. Sinon, les données demeurent centralisées

Réplication dans les BD

- PRINCIPE
  - Copie de chaque relation sur plusieurs sites
  - Réplication complète = copie sur tous les sites
- Objectifs de la réplication :
  - Amélioration de la disponibilité des données
  - Amélioration des performances
  - Autonomie locale
- Difficulté principale de la réplication :
  - Synchronisation des copies
- Mise à jour synchrone et asynchrone
  - Synchrone :
    - + Maintien de toutes les copies en cohérence
    - - Perte de performance du fait de la mise en œuvre de la validation à deux phases
  - Asynchrone : mise à jour différées des copies
    - + Incidence minime sur les performances
    - - Nécessité de mise à niveau de la copie ou des copies en cas de reprise

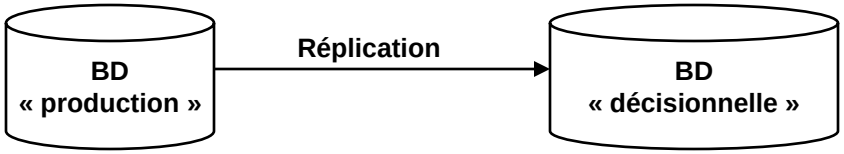
Réplication des données

- **Options de réplication de données :**

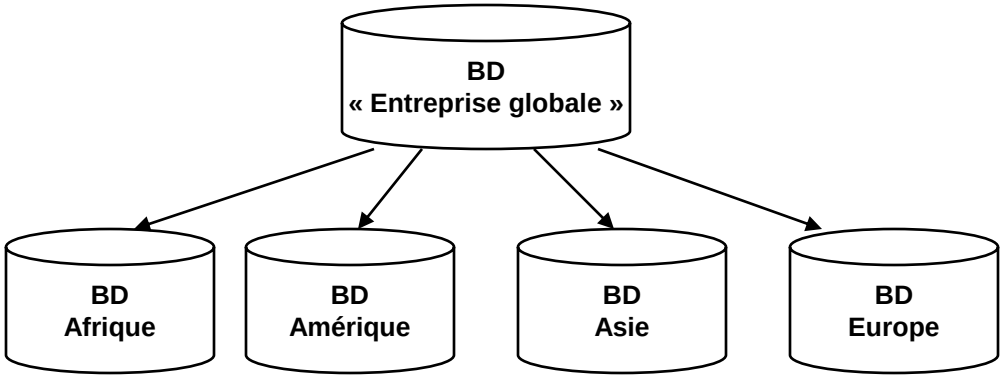


Réplication des données

- Quelques exemples d'utilisation
  - Mouvement d'information (OLTP ⇒ DSS)

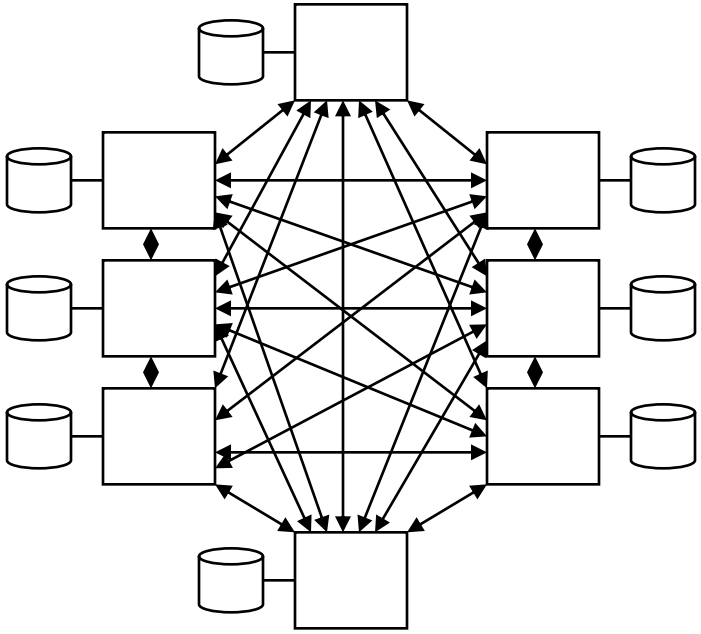


  - Distribution d'information

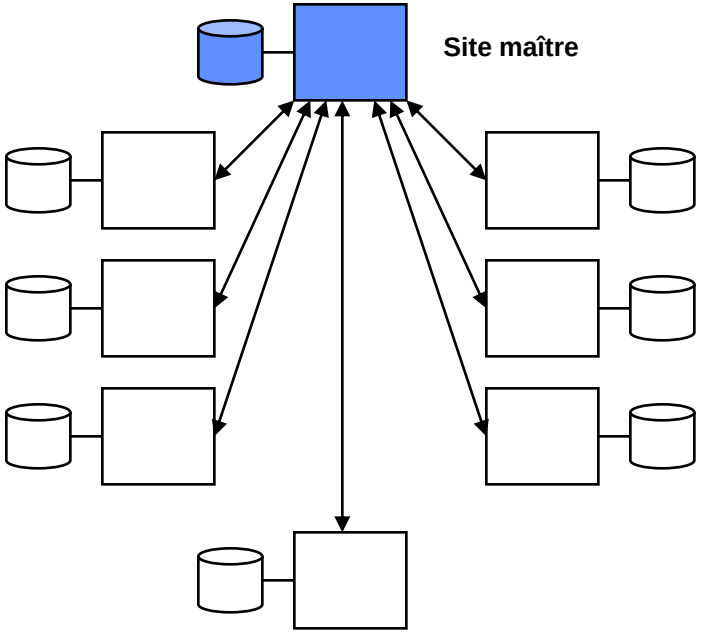


Réplication des données

Illustration de stratégies de mise à jour



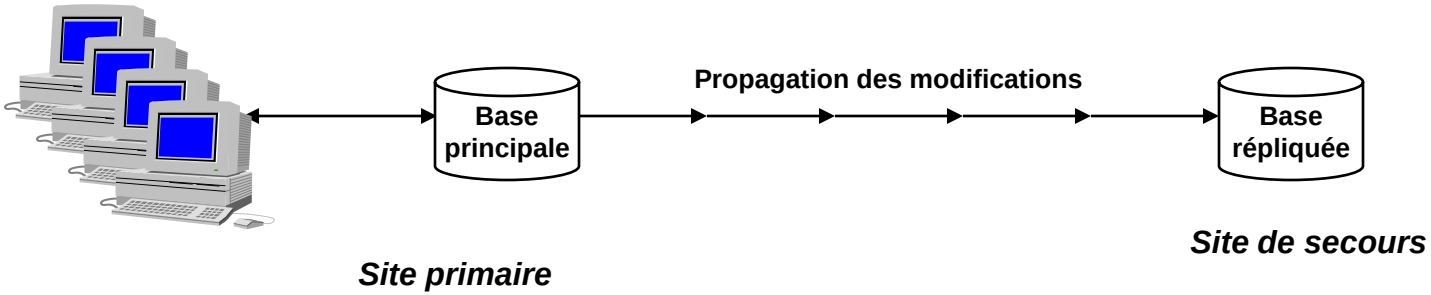
- Difficulté de conception
- Difficulté de reprise après panne
- Impact des mises à jour sur le fonctionnement des nœuds (overhead)



- Mises à jour asynchrones à partir d'un site maître
- Un seul point de référence
- Faible impact des mises à jour sur le fonctionnement des nœuds

Réplication des données

- **Réplication en vue de la résistance aux défaillances (Disaster Recovery)**



  - **La réplication peut être partielle ou totale**
  - **Elle peut se fonder sur une sauvegarde totale périodique (e.g. hebdomadaire) et la copie du journal des transactions à intervalles réguliers. La base répliquée est alors régénérée à partir de la dernière sauvegarde totale et du journal**

Optimisation des requêtes distribuées

- **Établissement d'un plan d'évaluation optimal**
- **Optimisation d'une fonction de coût ou de temps de réponse de la forme :**

  coût global = a x coût(E/S) + b x coût(Processeur) + c x coût(Communication)
  + d x coût(Transfert des données)

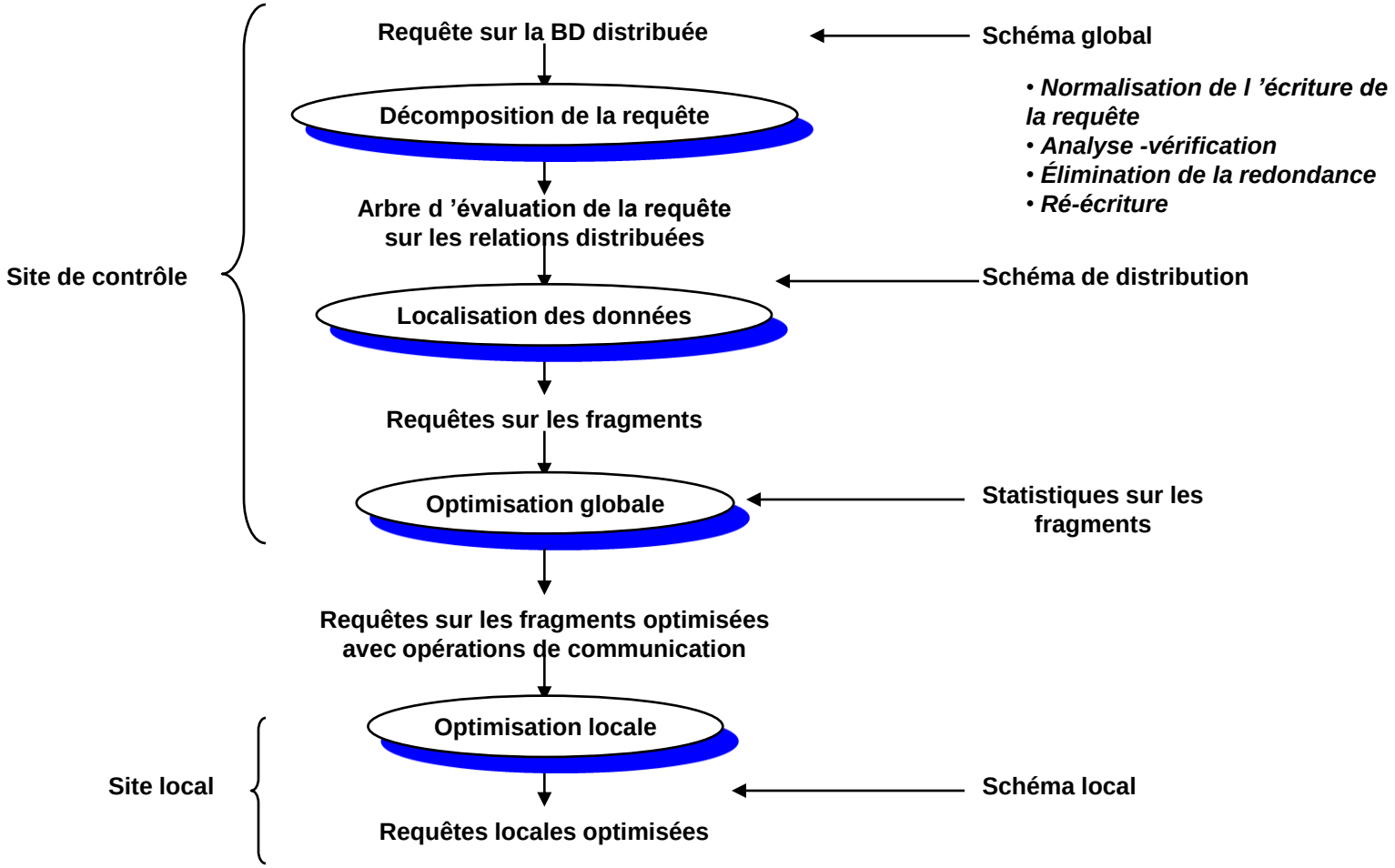
- **Rappel : la compilation d'une requête SQL produit un arbre d'évaluation composé d'un certain nombre d'opérateurs de base :**
  - **Projection : Π_X R projection de la relation R sur la liste d 'attributs X**
  - **Sélection : σ_P R sélection des tuples de R vérifiant le prédicat P**
  - **Équijointure : R1 $\bowtie_A$ R2 jointure des relations R1 et R2 selon l'attribut A (R1.A=R2.A)**
  - **Produit cartésien : x produit de deux relations**
  - **Union : $\cup$ union de 2 relations**
  - **Intersection : $\cap$ intersection de deux relations**
- **L'optimisation vise à transformer l'arbre d'évaluation en un arbre optimal**

Optimisation de requêtes distribuées

- **Requête distribuée : Requête émise par un client dont l'exécution nécessite l'exécution de n sous requêtes sur n serveur (n > 1)**
- **Objectifs de l'optimisation: un optimiseur de SGBD reparti doit élaborer un plan d'exécution reparti optimal de la requête composée de sous requêtes et de transferts**

Plan d'exécution reparti (*distributed execution plan*)
Arbre d'opérations dont certaines correspondent à des opérations locales à un site et d'autre à des transferts de données depuis un site vers un autre site

Schéma général de traitement et d'optimisation d'une requête distribuée



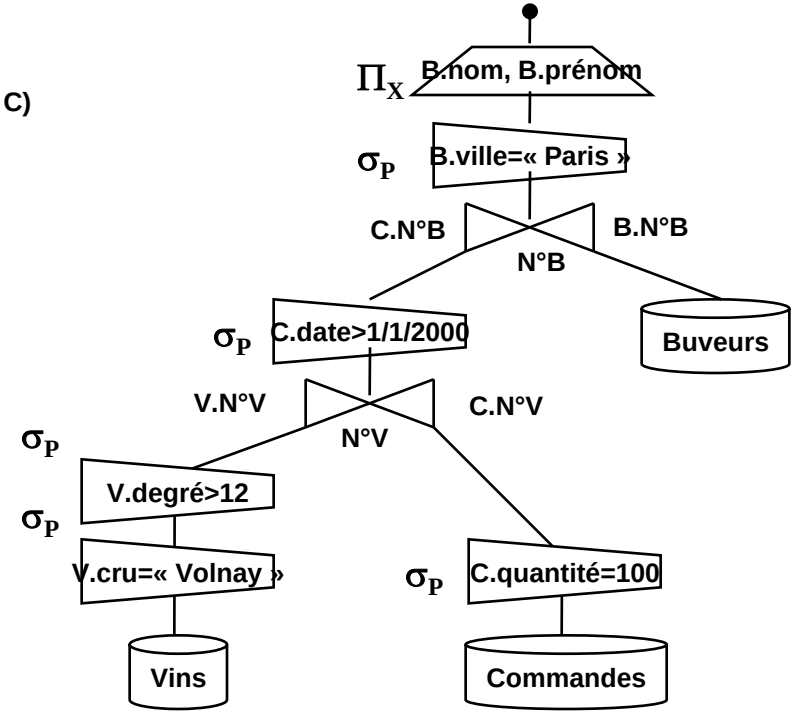
Optimisation d'une requête distribuée

- **Décomposition de requête: Cette couche prend une requête exprimée en termes de relations globales et effectue une première optimisation partielle . Le résultat de cette première optimisation est un arbre d'algèbre relationnelle fondé sur les relations globales.**
- **Localisation des données. Cette couche prend en ligne de compte la répartition des données sur le système. Une itération plus poussée de l'optimisation est effectuée en remplaçant les relations globales des feuilles de l'arbre d'algèbre relationnelle par leurs algorithmes de reconstruction (ce que l'on appelle parfois les programmes de localisation de données), c'est-à-dire les opérations d'algèbre relationnelle qui reconstituent les relations globales à partir des fragments constitutifs.**
- **Optimisation globale. Cette couche prend en considération des informations statistiques pour trouver un plan d'exécution proche de l'optimum. Le résultat de cette couche est une stratégie d'exécution basée sur des fragments, où viennent s'ajouter des primitives de communication qui envoient les parties de la requête aux SGBD locaux pour qu'ils les exécutent, et qui permettent ensuite de recevoir les résultats.**
- **Optimisation locale. Tandis que les trois premières couches s'exécutent sur le site de contrôle (généralement le site que a lancé la requête), cette couche particulière s'exécute sur chacun des sites locaux impliqués dans la requête. Chaque SGBD local effectue ses propres optimisations**

exemple

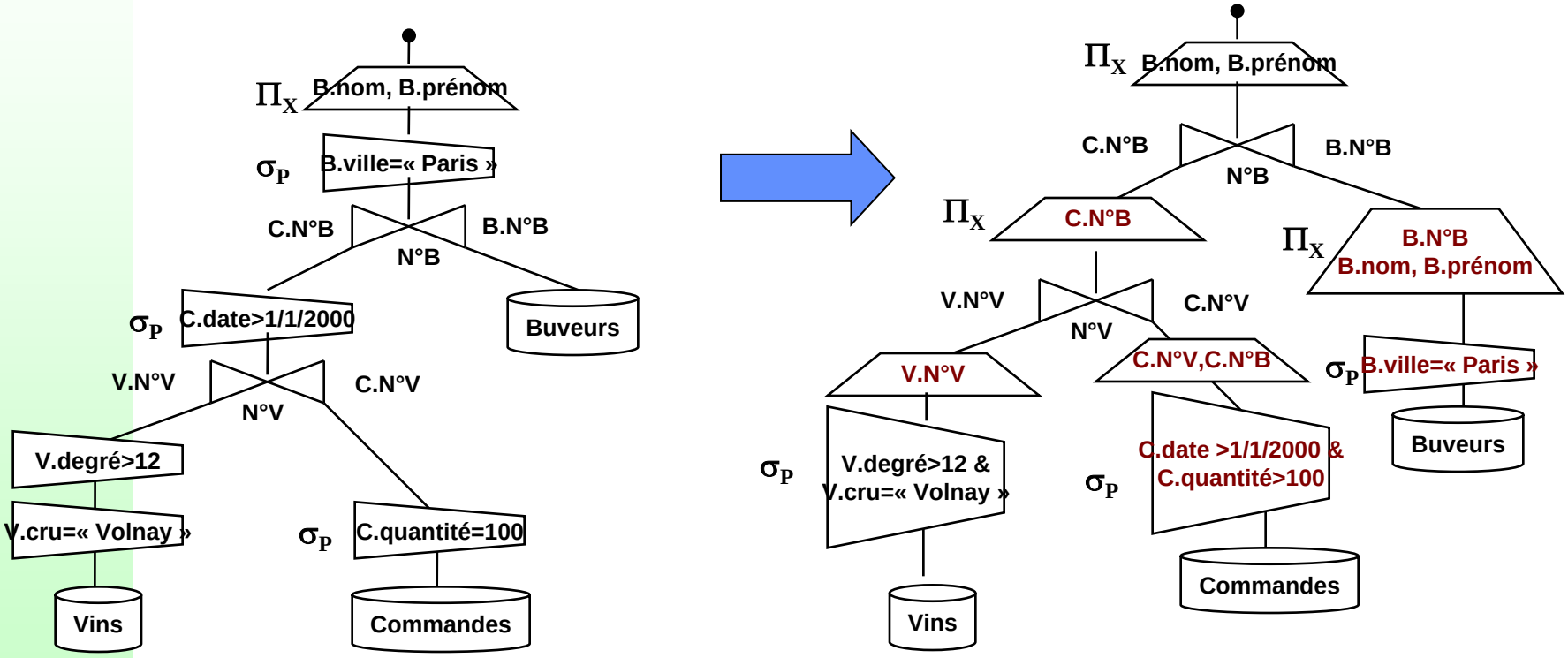
- ***Arbre résultant de la traduction directe de la requête en opérations algébriques***

```
Select nom, prenom
From buveurs(b),vins (v), commandes ( C)
Where v.cru='volnay'
And v.degré>12
And c.qté>100
And c.nv=v.nv
And c.date>1/10/88
And b.nb=c.nb
And b.ville="paris"
```



exemple

- ***Arbre optimisé par restructuration algébrique***



Optimisation des requêtes distribuées

- **Hypothèse de volume :**
  - Buveurs (B) : 10 000 tuples
  - Vins (V) : 1 000 tuples
  - Commandes : 200 000 tuples
- **Hypothèse de distribution des BD :**
  - Paris : Buveurs (B)
  - Dijon : Vins 1 (V1) restriction N°V<= 400
    Commandes 1 (C1) restriction N°V<= 400
  - Bordeaux : Vins 2 (V2) restriction N°V > 400
    Commandes 2 (C2) restriction N°V>400
- **Requête émise sur le site de Paris :**
  « Noms des buveurs parisiens n'ayant pas commandé en décembre 2000 »
  Sélectivités supposées : 20% de parisiens et 1% n'ayant pas commandé
- **Stratégies :**
  - Simpliste : transférer C1 et C2 vers Paris (200 000 tuples) et faire C = C1 ∪ C2 et évaluer SELECT B.nom FROM Buveurs (B) WHERE B.ville = « Paris » AND B.N°B NOT IN (SELECT N°B FROM C WHERE C.date>1/12/2000 AND C.date<1/1/2001)
  - Améliorée : Transférer vers Dijon et Bordeaux Buveurs.N°B des seuls parisiens (= 2 x 2 000 petits tuples). Évaluer sur les sites de Dijon et Bordeaux : Buveurs.N°B NOT IN (SELECT N°B FROM Ci WHERE Ci.date>1/12/2000 AND Ci.date<1/1/2001)
    Transférer les résultats vers Paris (= 2 x 20 petits tuples) et faire l'intersection des résultats

Résumé

- Une **base de données distribuée** est une collection logiquement interconnectée de données partagées (et une description de ces données) réparties physiquement sur un réseau informatique. Le **système de gestion de base de données distribuée (SGBDD)** est le logiciel qui gère la base de données distribuée et assure la transparence de la distribution vis-à-vis des utilisateurs.
- Un SGBDD se distingue du **traitement distribué**, où un SGBD centralisé reçoit des accès par l'entremise d'un réseau. Il se distingue également du **SGBD parallèle**, qui s'exécute sur plusieurs processeurs et disques, et qui est conçu pour évaluer des opérations en parallèle chaque fois que cela est possible, de manière à améliorer les performances.
- Les avantages d'un SGBDD sont qu'il reflète la structure organisationnelle, améliore la partage des données, améliore la fiabilité, la disponibilité et les performances; il est également plus économique, il facilite l'expansion grâce à sa modularité,il facilite l'intégration et aide à maintenir la compétitivité des organisations. Ses principaux inconvénients sont son coût, sa complexité, le manque de standards et d'expérience dans l'industrie.
- Un SGBDD est considéré comme homogène ou hétérogène. Dans un **système homogène**, tous les sites utilisent le même produit de SGBD, tandis que, dans un **système hétérogène**, les sites exploitent des produits de SGBD différents, qui ne suivent pas nécessairement le même modèle de données sous-jacent, de sorte que le système peut être composé de SGBDrelationnels, en réseau, hiérarchiques ou orientés objet.
- Un **système multibase de données** (SMBD) est un SGBD distribué dont chaque site garde sa propre autonomie..

Résumé

- Une relation peut être scindée en un certain nombre de sous-relations, appelées **fragments**, **allouées** (attribuées) à un ou plusieurs sites. Les fragments sont éventuellement **dupliqués** (copiés tels quels) pour offrir une meilleure disponibilité et des performances accrues.
- Nous connaissons deux types de fragmentations : **horizontale** et **verticale**. Les fragments horizontaux sont des sous ensembles de tuples et les fragments verticaux sont des sous-ensembles d'attributs de relations. La fragmentation se présente aussi sous un autre type : **mixte**.
- La définition de l'allocation des fragments est menée de façon stratégique, de manière à garantir la localité des références, une fiabilité et une disponibilité améliorées, des performances acceptables, un équilibre des capacités de stockage et des coûts, ainsi que des coûts de communication réduits. Les trois règles dites de « correction » de la fragmentation sont l'aspect complet, la reconstruction et la disjointure.
- Quatre stratégies d'allocation existent, en fonction de la disposition des données : **centralisée** (une seule base de données centralisée), **fragmentée** (des fragments sont attribués à un site), de **réplication complète** (une copie complète de la base de données est entretenue dans chaque site) et de **réplication sélective** (une combinaison des trois autres).
- Un SGBDD doit apparaître comme un SGBD centralisé, en assurant une série de transparences. La **transparence de distribution** fait que les utilisateurs ignorent la réplication ou la fragmentation des données. La **transparence de transaction** garantit la transparence de la base de données globale quand des utilisateurs accèdent simultanément à la base de données et quand des pannes se produisent. La **transparence de performances** garantit que le système gère efficacement les requêtes faisant référence à des données de plus d'un site. La **transparence de SGBD** permet l'usage de plusieurs SGBD différents dans le système, sans que l'utilisateur ne s'en rende compte

le cnam

Transactionnel et transactionnel réparti

Plan

- Introduction
- Concept de transaction - Propriétés ACID
- Caractéristiques du transactionnel
- Transactionnel réparti
- Moniteur transactionnel
- Modèle X/Open
- Exemple de moniteur transactionnel: Tuxedo
- Moniteurs transactionnels et SGBDs

Introduction

- Le transactionnel est une dimension essentielle des systèmes d'information des entreprises
- Un système transactionnel (OLTP On Line Transaction Processing) fournit un cadre pour les applications critiques, il est fiable et à haute performance;
- Le *transactionnel* réfère à un mode d'exploitation de données tourné vers la saisie, le stockage, la mise à jour, la sécurité et l'intégrité des données
- Par exemple, les systèmes de gestion des transactions boursières ou bancaires, dont les guichets automatiques ou les systèmes d'inventaire dans les magasins

Concept de transaction

Une transaction est une collection d'actions qui transforment la BD depuis un état cohérent en un autre état cohérent

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

Définition

- **Une transaction est un ensemble d'opérations menées sur une BD,Ces opérations peuvent être en lecture et/ou écriture,**
- **Une opération est atomique, c'est donc une unité indivisible de traitement,**
- **Une transaction est soit validée par un *commit*, soit annulée par un *rollback*,soit interrompue par un *abort*,**

- **Une transaction a une marque de début (Begin Of Transaction BOT), et une marque de fin (End Of Transaction EOT).**

- **La cohérence et la fiabilité d'une transaction sont garanties par 4 propriétés :**
- **l'Atomicité, la Cohérence, l'Isolation, la Durabilité qui font l'*ACID*ité d'une transaction.**

Exemple

- **Transfert d'argent entre les comptes:**

  UPDATE Compte1

  Val = Val -100

  UPDATE Compte2

  Val = Val + 100

- **Si seulement une de ces requêtes est exécutée, la BD perd sa consistance**

Propriétés ACID

- on appelle transaction une séquence d'actions sur l'état physique et logique d'une application qui respecte les propriétés suivantes dites ACID (Atomicity, Consistency, Isolation, Durability):

- **Atomicité:** Les changements opérés par une transaction sur l'état sont atomiques: ils sont tous exécutés ou bien aucun ne l'est;
- **Consistance**: Une transaction est une transformation correcte de l'état. L'ensemble des actions accomplies par la transaction ne viole pas les contraintes associées avec l'état. Ceci implique que la transaction soit un programme correct;
- **Isolation**: les résultats d'une transaction ne doivent être visibles aux autres transactions qu'une fois la transaction validée, ceci afin d 'éviter les interférences avec les autres transactions
- **Durabilité**: Lorsqu'une transaction se termine avec succès (commitement), le changement qu'elle a provoqué sur l'état doit survivre aux défaillances.

Caractéristiques du transactionnel

- Partage:
  - en Lecture et Écriture
  - par l'ensemble des utilisateurs
  - Propriétés ACID
- Flux de requêtes irrégulier
- Travail répétitif
  - Répertoire de fonctions pré-défini typiquement O(100) fonctions
- Fonctions simples
  - Fonctions peu complexes (typiquement de 10^5 à 10^7 instructions et 10 E/S)
- Possibilité de traitement de type batch (avec respect des propriétés ACID)
- Grand nombre de terminaux (1000-10000)
- Clients intelligents (stations, PC, autres systèmes, terminaux)
- Haute disponibilité requise
  - Recouvrement effectué par le système
  - Fondé sur les propriétés ACID
- Taille des bases de données
  - Proportionnelle à l'activité de la Société
- Peu de données "touchées" par une transaction
- Équilibrage de charge automatique
- Recherche de la performance au moyen du parallélisme inter-requête
- Performance : haut débit et temps de réponse garanti
- Scalabilité : exigence typique

Commit et Abort

- **INTRODUCTION D'ACTIONS ATOMIQUES**
  - **Commit (fin avec succes) et Abort (fin avec echec)**
  - **Ces actions s'effectuent en fin de transaction**
- **COMMIT**
  - **Validation de la transaction**
  - **Rend effectives toutes les mises à jour de la transaction**
- **ABORT**
  - **Annulation de la transaction**
  - **Défait toutes les mises à jour de la transaction**

Schéma de transaction simple

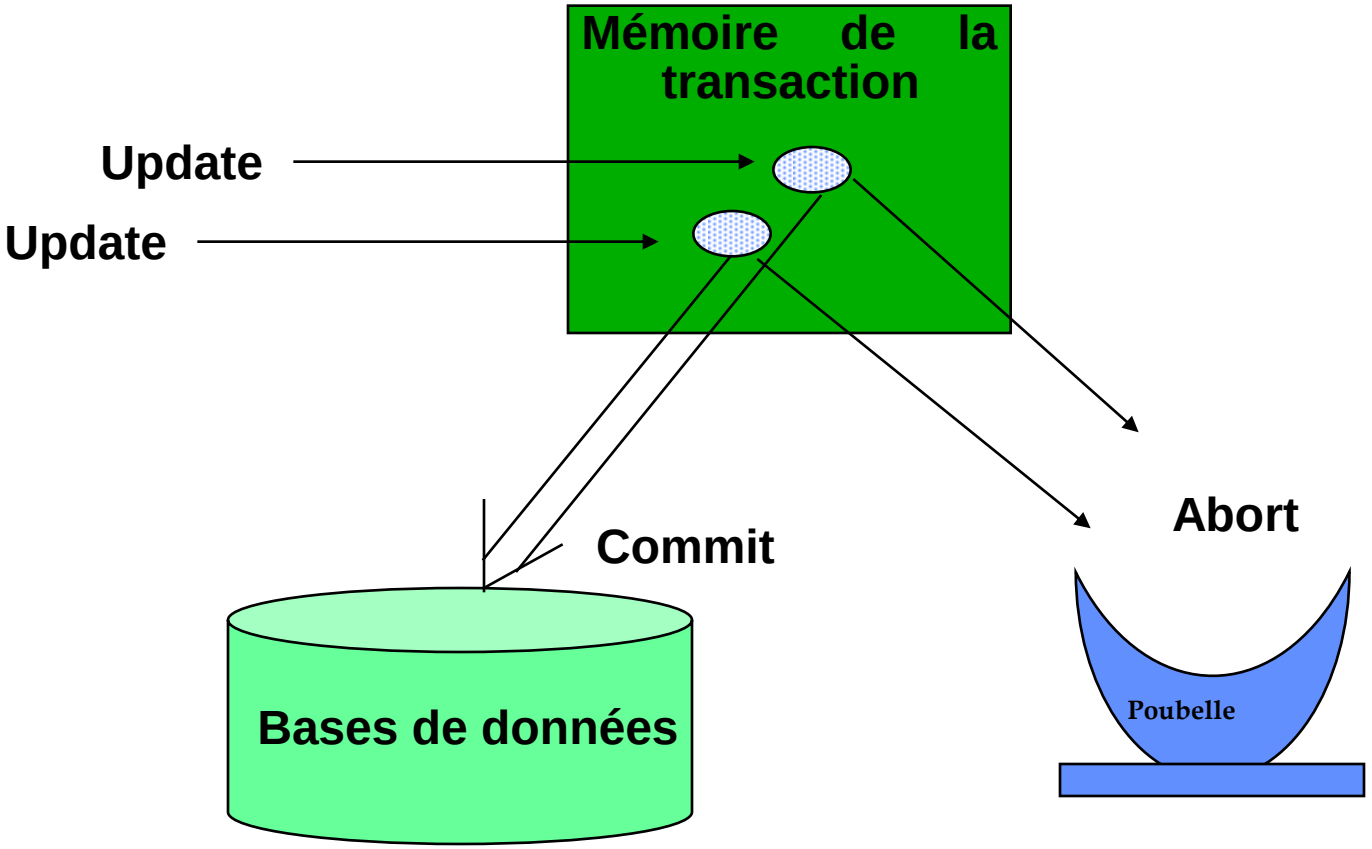
- Fin avec succès ou échec

- Begin_Transaction
  - update
  - update
  - .....
  - Commit ou Abort

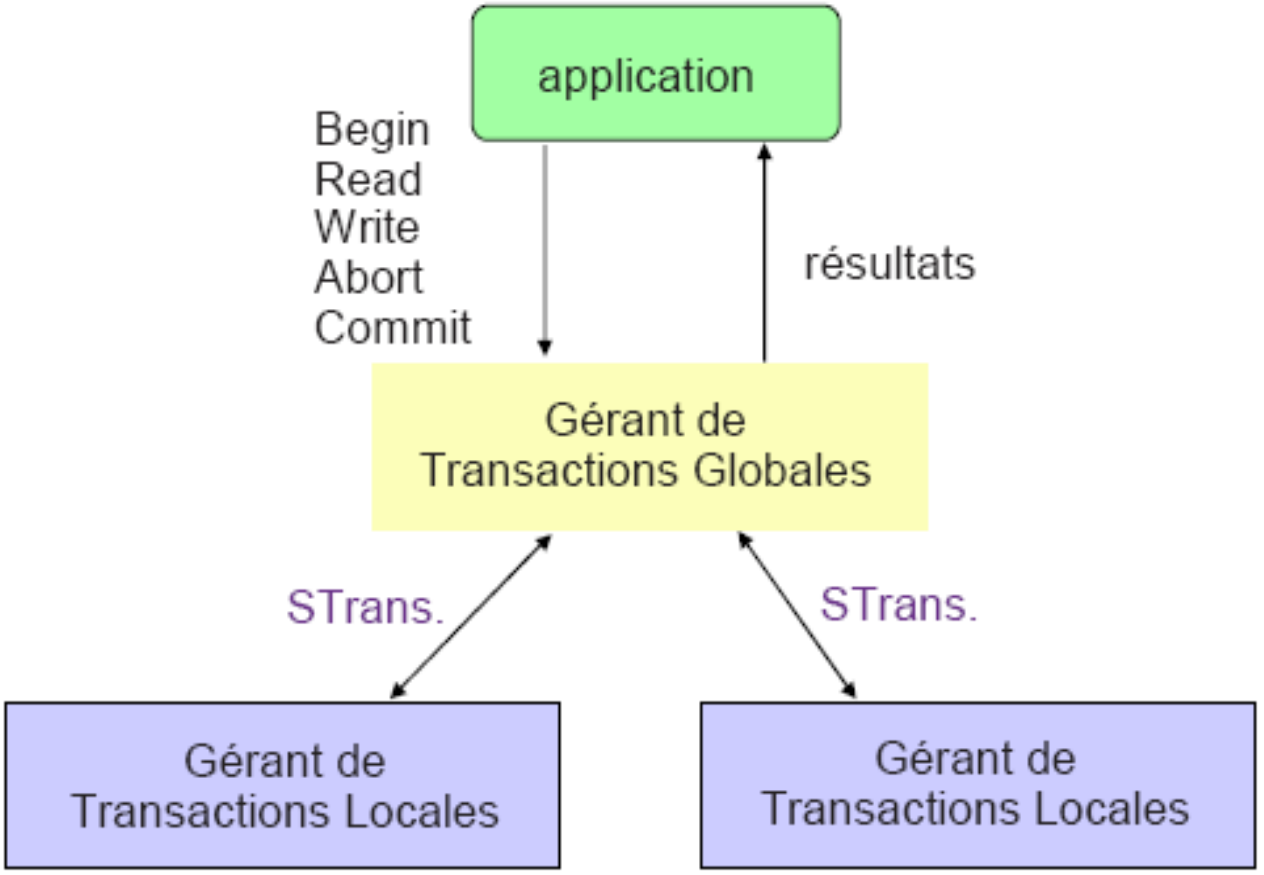
- Provoque l'intégration réelle des mises à jour dans la base
- Relâche les verrous

- Provoque l'annulation des mises à jour
- Relâche les verrous
- Reprend la transaction

Effet logique



Gestion de transactions reparties



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 ...

Validation d'une Transaction

- **Transactions Centralisés**
  - **l'application et les données sont sur la même machine**
  - **Panne facile à traiter**
  - **Validation à 1 phase**
- **transactions Distribuées**
  - **l'application et les données sont sur 2 à N machines**
  - **Panne (partielle) difficile à traiter**
  - ***Validation à 2 phases (2PC : Two Phases Commit)***
  - ***Validation à 3 phases (3PC : Three Phases Commit)***

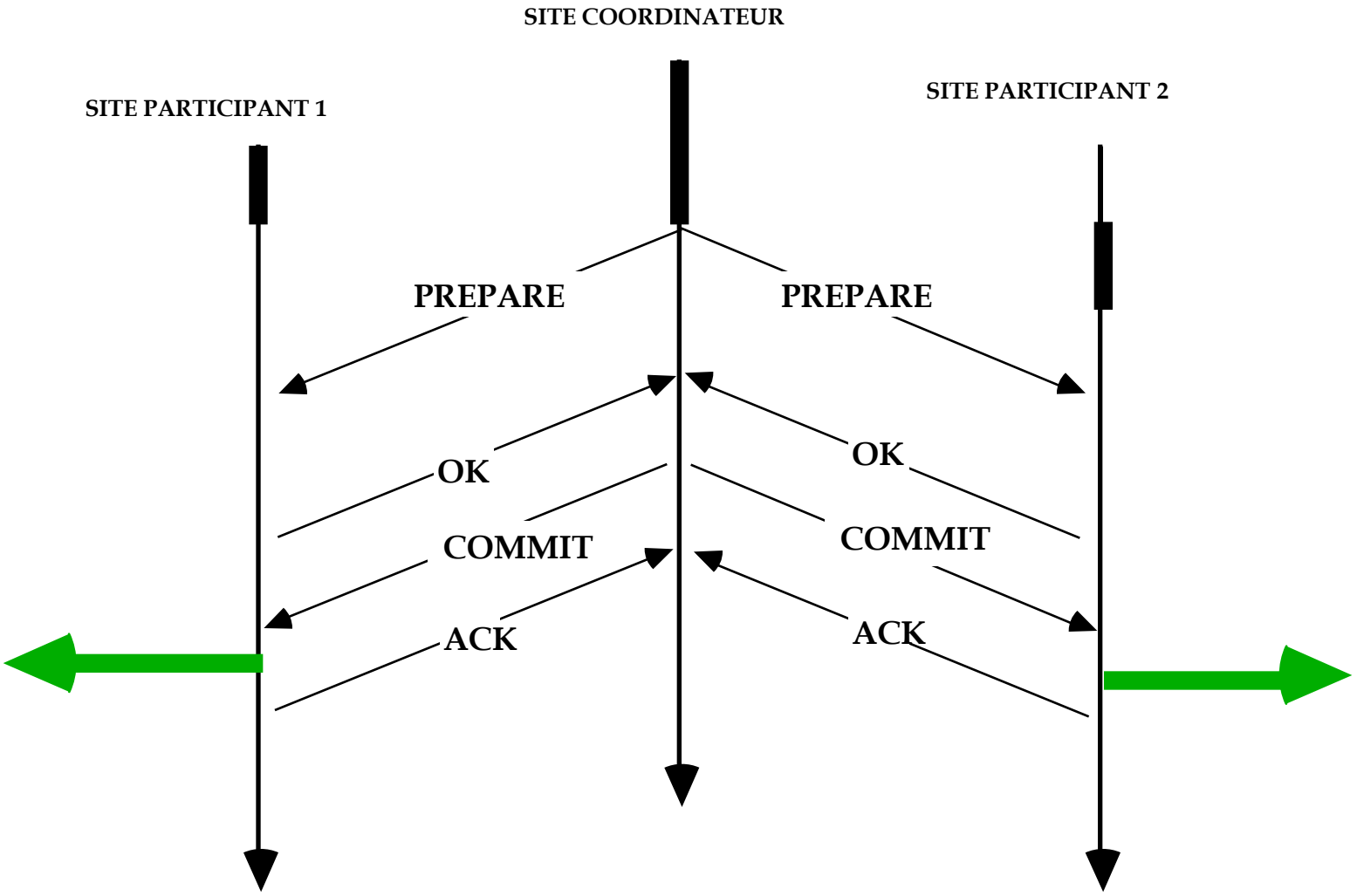
Commit en 2 Phases

- **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

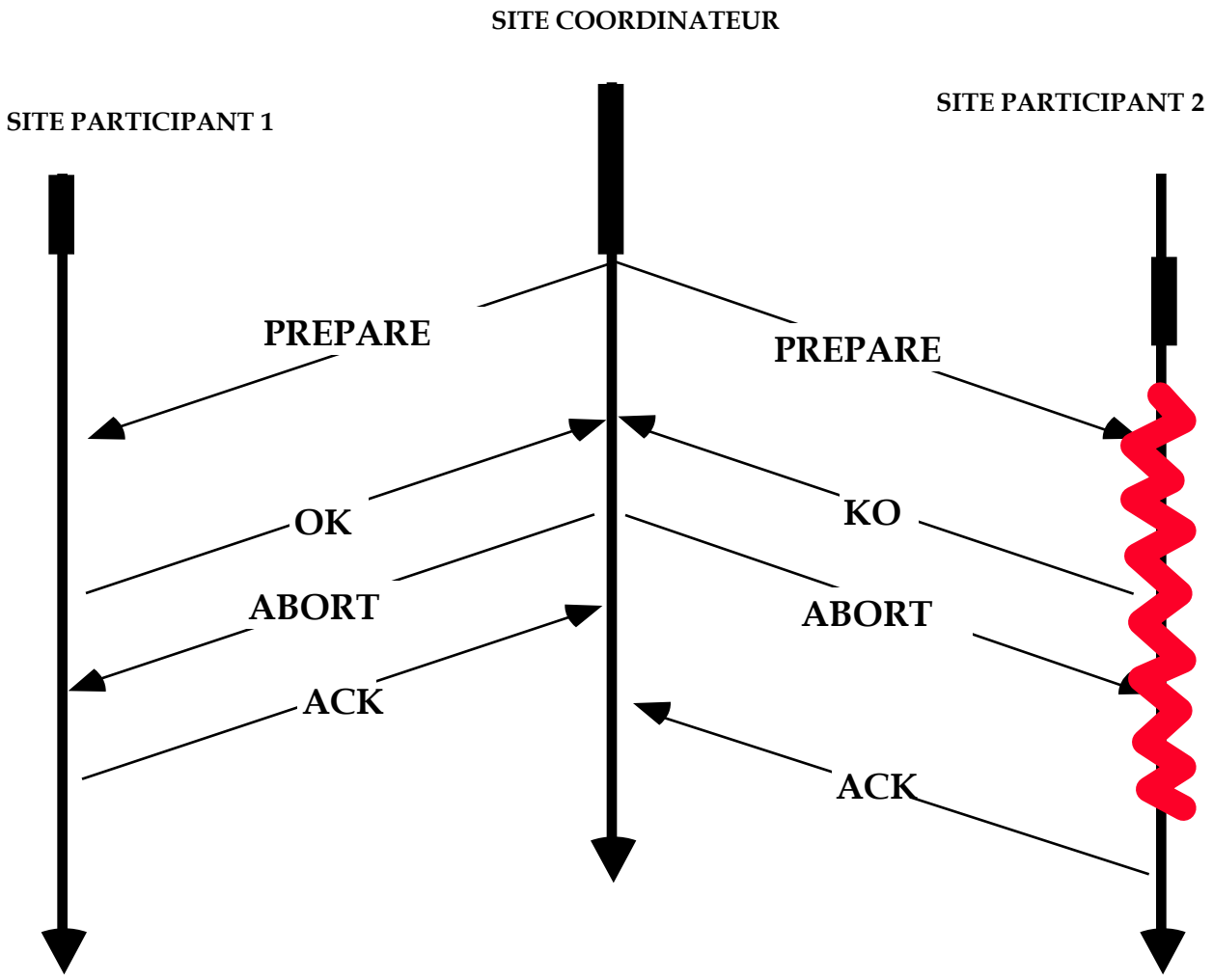
Protocole C/S

- 1. PREPARER
  - Le coordinateur demande aux autres sites s'ils sont prêts à commettre leurs mises à jour.
- 2a. SUCCES : COMMETTRE
  - Tous les participants effectuent leur validation sur ordre du client.
- 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.

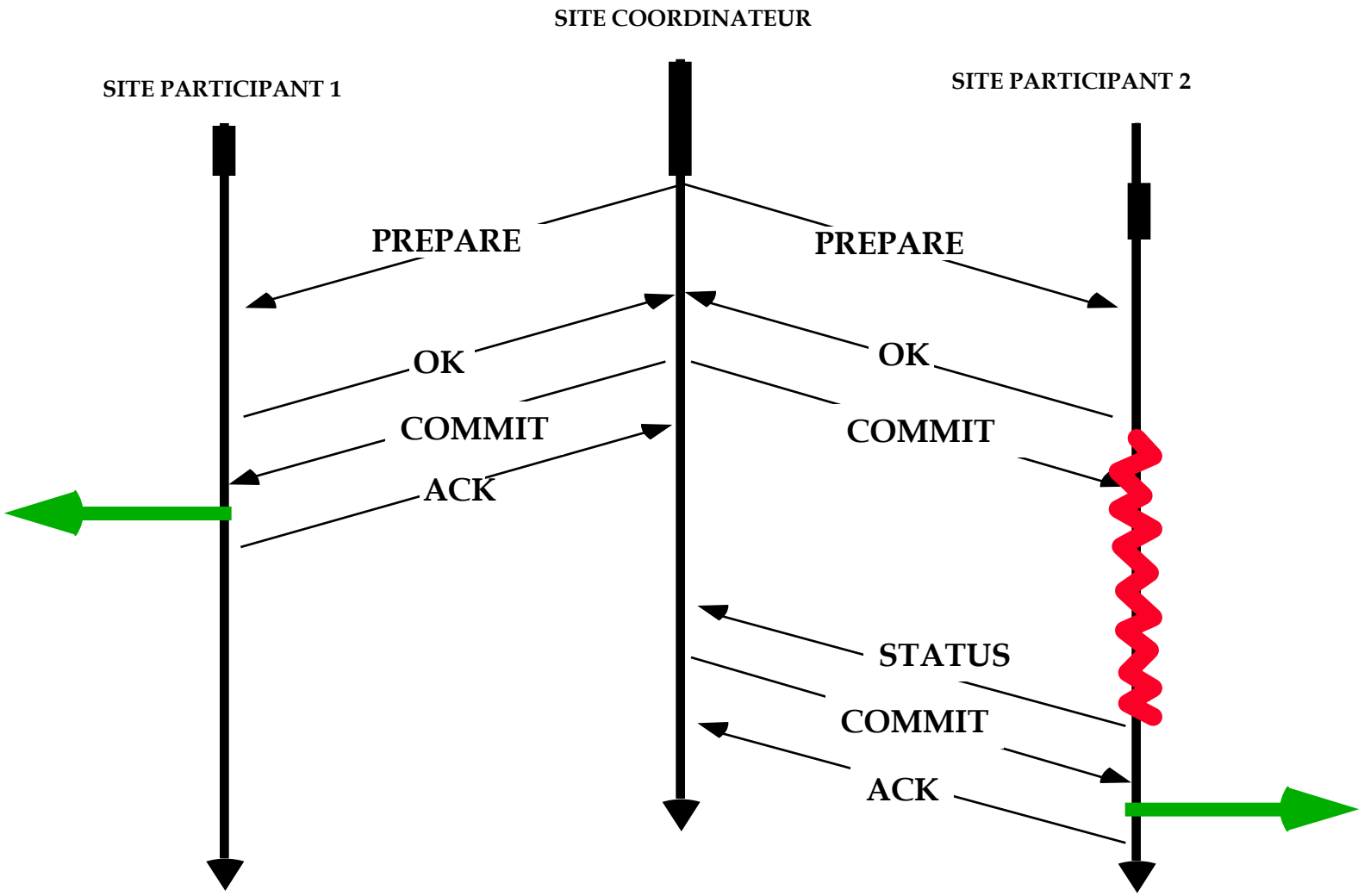
Cas Favorable



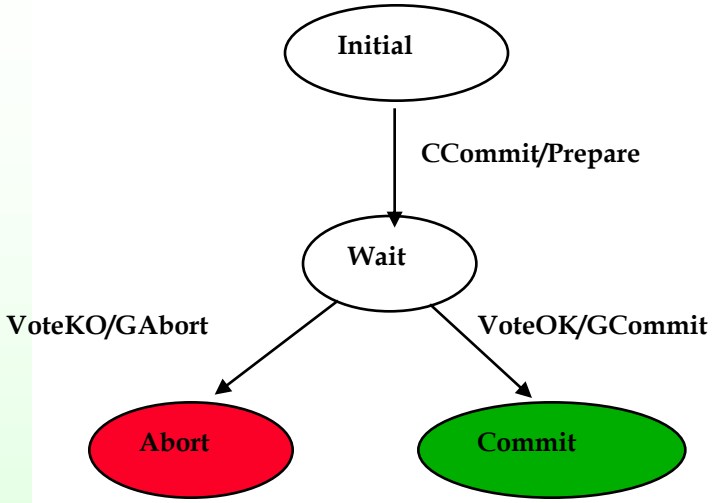
Cas Défavorable (1)



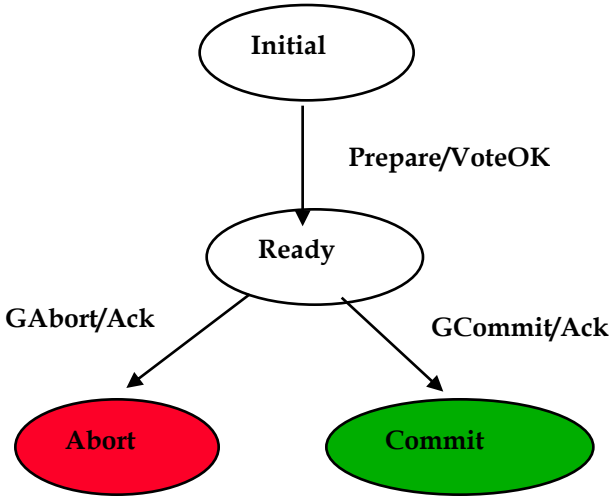
Cas Défavorable (2)



Transitions d'Etats

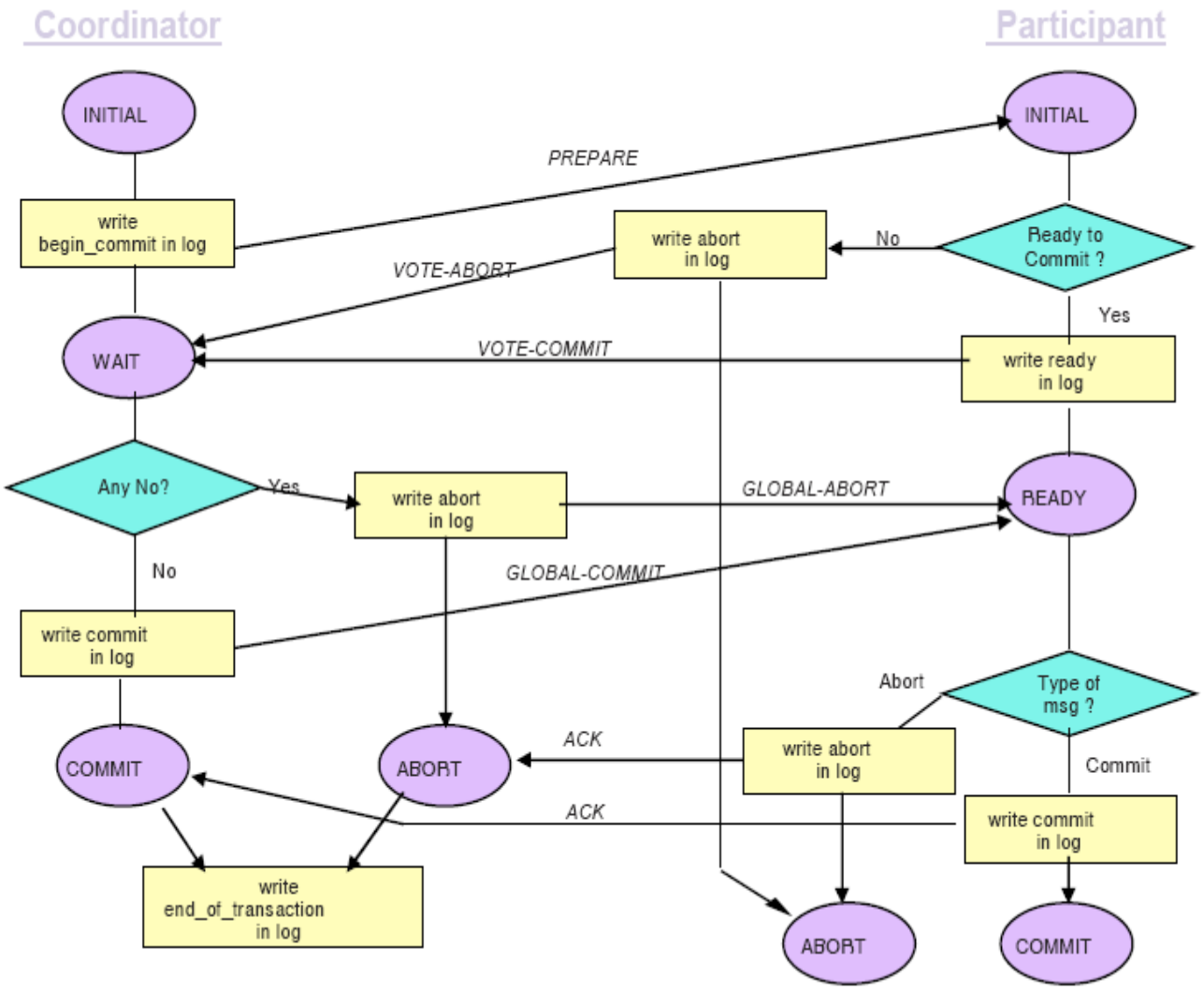


COORDINATEUR



PARTICIPANT

Actions du protocole

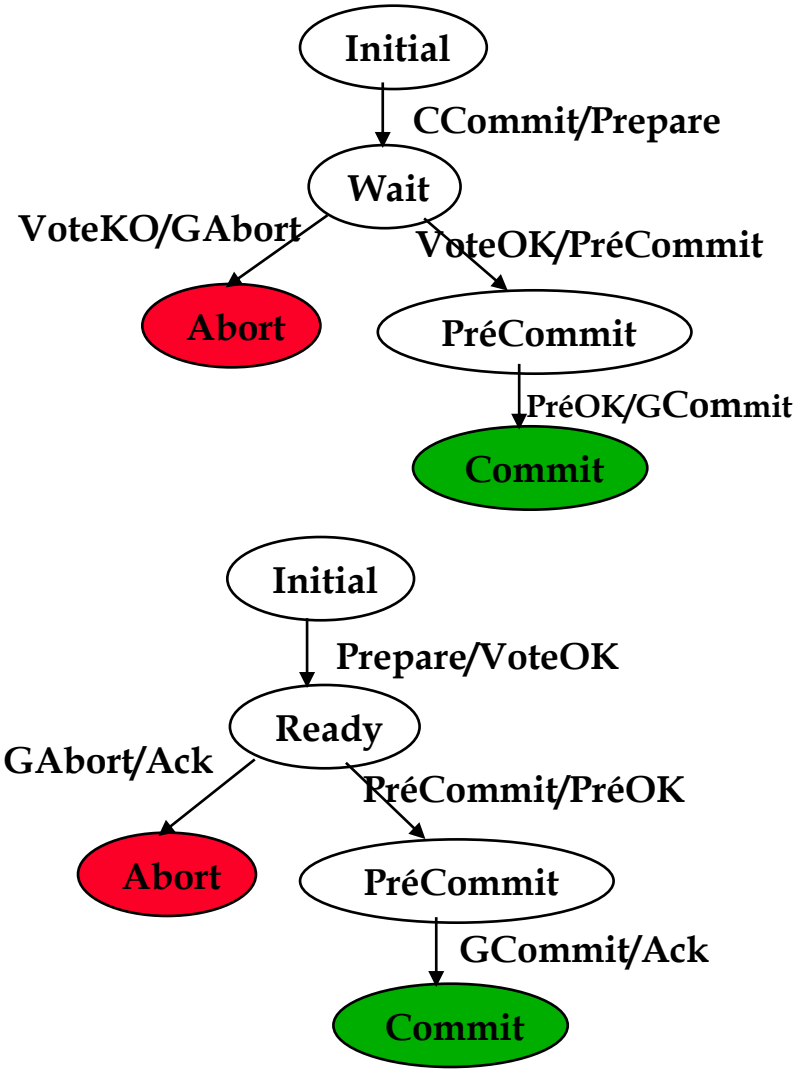


Transactions bloquées

- Que faire en cas de doute ?
  - Demander l'état aux autres transactions : STATUS
    - conservation des états nécessaires
    - message supplémentaire
  - Forcer la transaction locale : ABORT
    - toute transaction annulée peut être ignorée
    - cohérence garantie avec un réseau sans perte de message
  - Forcer la transaction locale : COMMIT
    - toute transaction commise peut être ignorée
    - non garantie de cohérence avec le coordinateur

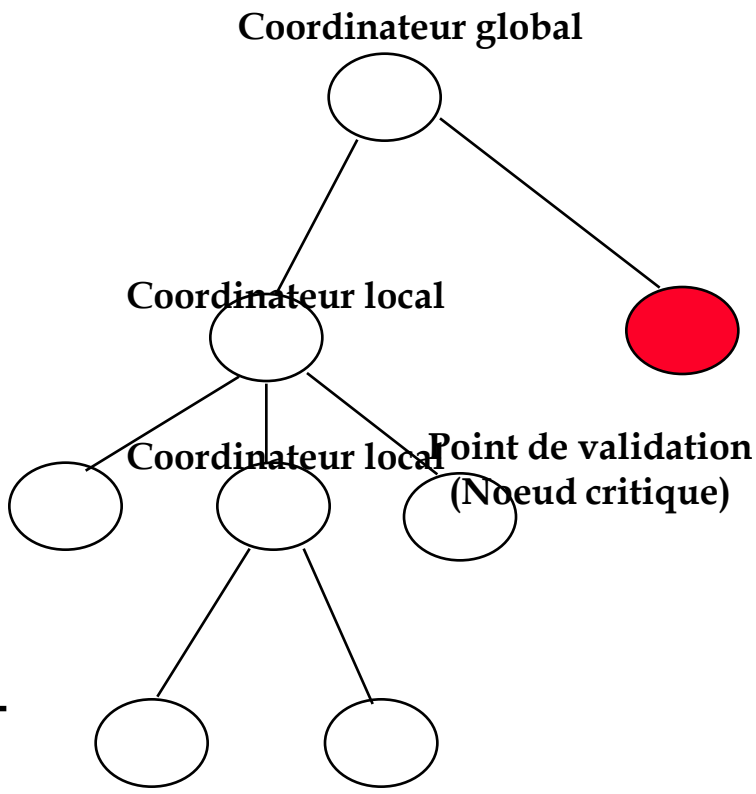
Commit en 3 Phases

- **Inconvénient du commit à 2 phases**
  - en cas de time-out en état “Prêt”, le participant est bloqué
  - le commit à 3 phases permet d’éviter les blocages
- **Messages du commit à 3 phases**
  - Prepare,
  - Prepare to Commit,
  - Global-Commit,
  - Global-Abort.



Protocole arborescent TP

- TP est le standard proposé par l'ISO dans le cadre OSI
- Protocole arborescent
  - Tout participant peut déclancher une sous-transaction
  - Un responsable de la validation est choisi
  - Un coordinateur est responsable de ses participants pour la phase 1
    - collecte les PREPARE
    - demande la validation
  - Le point de validation est responsable de la phase 2
    - envoie les COMMIT



Moniteur transactionnel

- Logiciel assurant le contrôle de la faisabilité d'une transaction commerciale en temps réel et l'équilibre de la charge de traitement entre les différents serveurs.
- Vendre des produits ou proposer des services en ligne implique l'interaction de nombreux processus. Pour la réservation de billets, par exemple, il faut notamment vérifier la disponibilité du service, débiter le compte du client et créditer celui de l'entreprise.
- Dans ce contexte, le recours à un moniteur transactionnel garantit à la fois la fiabilité et la robustesse des applications commerciales.
- Le moniteur transactionnel surveille ainsi toutes les requêtes du client et appelle les fonctions correspondantes sur le serveur.

Moniteur transactionnel

- Peu de systèmes d'exploitation ont été conçus dans l'optique du transactionnel
- Le support d'un grand nombre d 'utilisateurs et d 'un flux important de transactions (plusieurs milliers par seconde) provoque un effondrement des systèmes
- Le rôle d'un moniteur transactionnel est de :
  - Gérer des processus comprenant le lancement des applications, le contrôle de leur déroulement et l 'équilibrage de charge (on peut parler de multiplexage des requêtes sur les ressources du système)
  - Gérer des transactions (respect des propriétés ACID) dans un contexte, éventuellement distribué, mettant en jeu plusieurs gestionnaires de données)

Moniteur transactionnel

- Support des transactions ACID
- Accès continu aux données
- Reprise rapide du système en cas de panne
- Sécurité d'accès
- Performances optimisées
- partage des connexions
- réutilisation de transactions
- Partage de charge
- distribution de transactions sur les serveurs
- Support de fichier et bases de données hétérogènes
- API standardisées

Fonctions transactionnelles

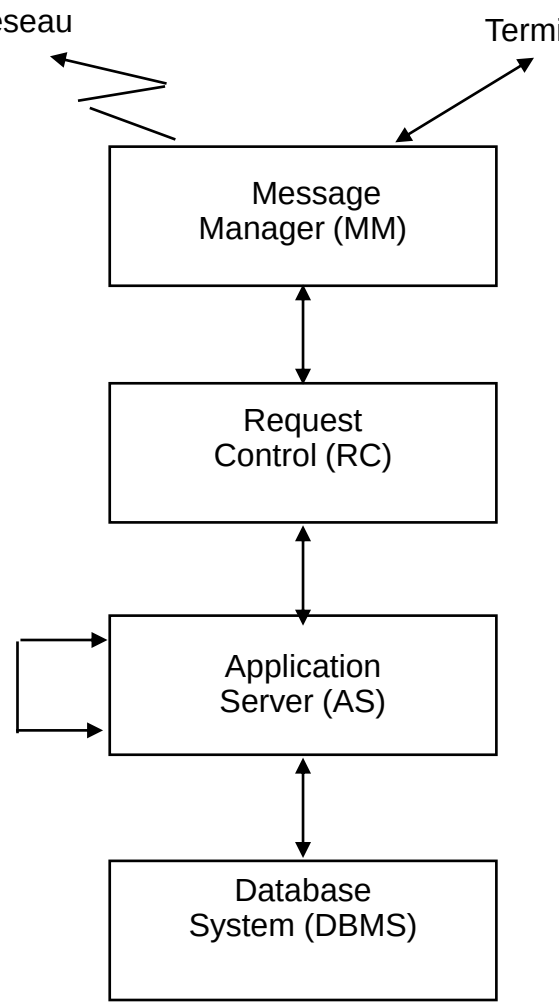
Etapes du traitement transactionnel

- traduire la requête utilisateur en format interne au moniteur
- déterminer le type de transaction et le programme nécessaire à son exécution
- débuter la transaction et lancer son exécution
- valider, ou abandonner, la transaction
- retourner le résultat à l'émetteur de la requête

Un moniteur assure aussi :

- équilibrage de charge entre requêtes et serveurs
- résistance aux pannes (client, serveur)
- gestion de configuration (processus, sessions, ...)
- contrôle de performances et réglage
- sécurité

Modèle de moniteur transactionnel



. Collecte les entrées des transactions (gestion de formes)
. Construit un format standard d'entrée des requêtes
. Envoie les résultats (gestion de formes)

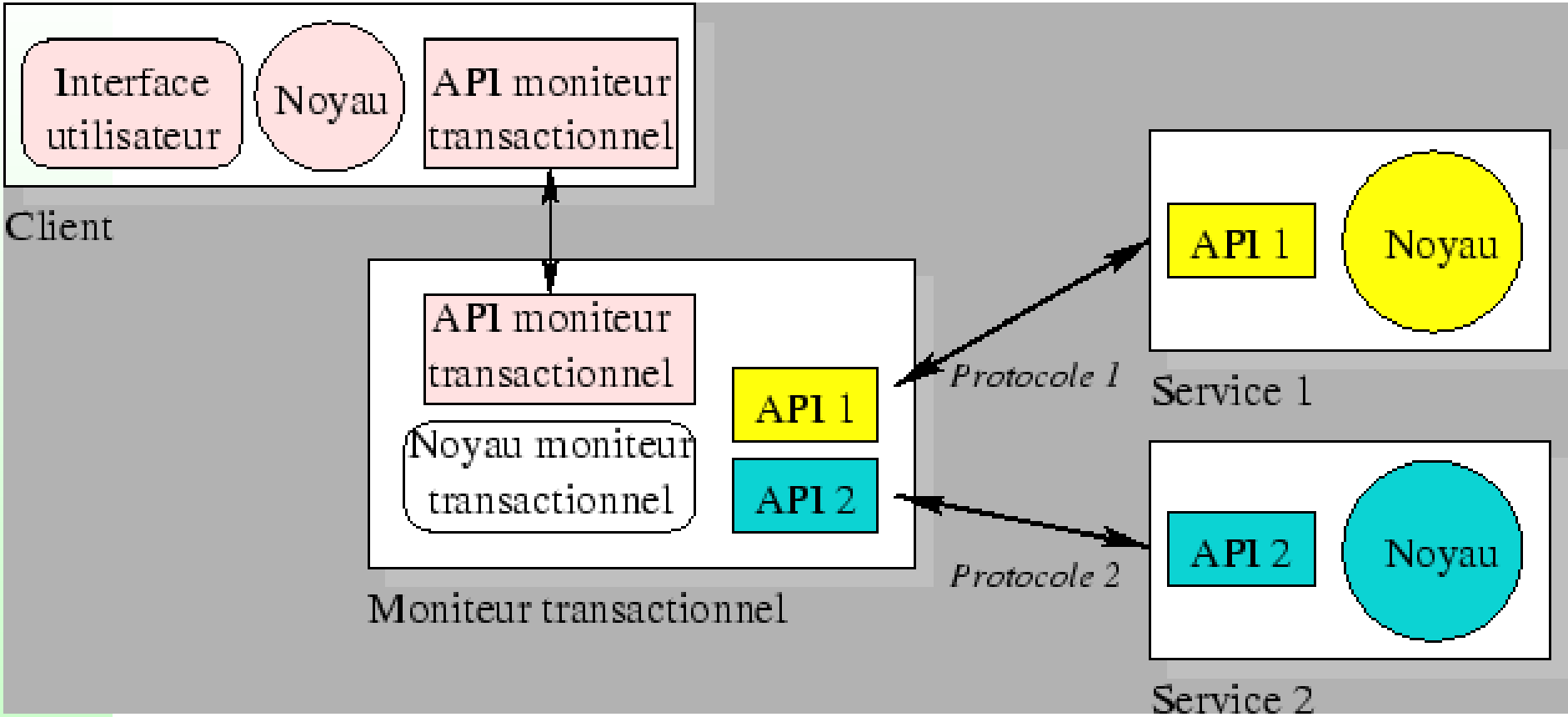
. Débute et termine les transactions
. Détermine le type des requêtes
. Dirige les requêtes vers les applications appropriées

. Exécute les programmes d'application

. Gère les données partagées

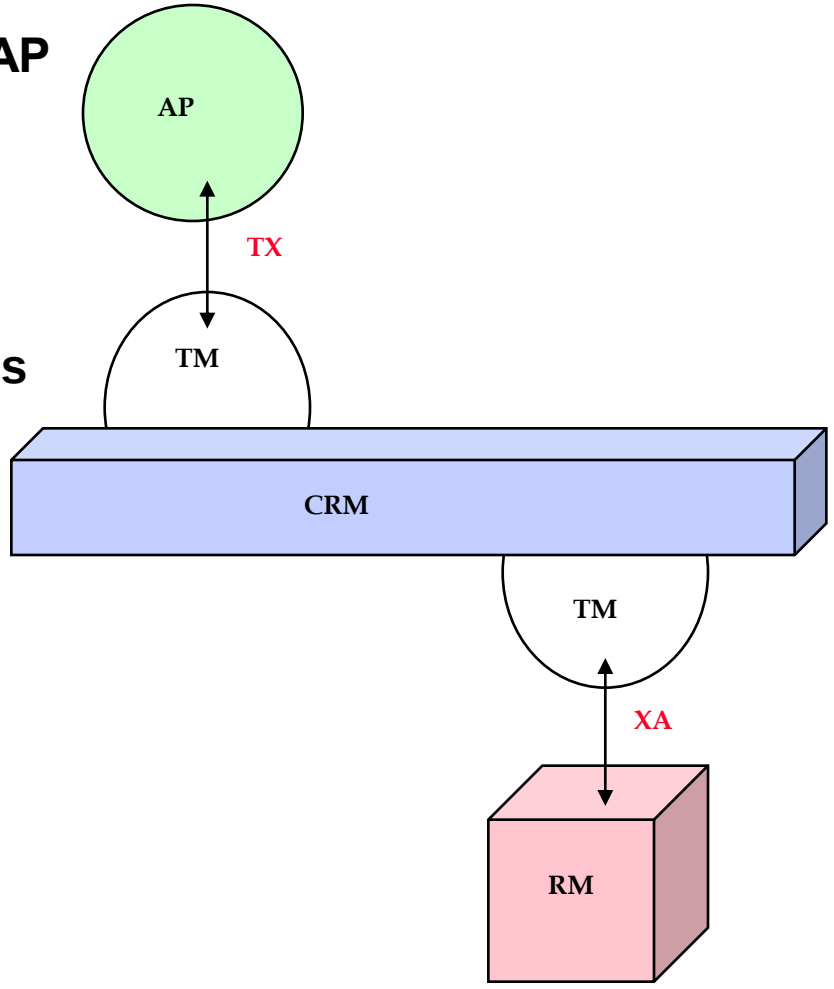
Interface utilisateur

Utilisation d'un API unique
Masquage de la complexité de l'organisation des services



Moniteur transactionnel : Modèle X/OPEN

- **Modèle DTP de l'X/OPEN**
  - Programme d'application AP
  - Gestionnaire de transactions TM
  - Gestionnaire de communication CRM
  - Gestionnaire de ressources RM
- **Interfaces standards**
  - TX = interface du TM
  - XA = interface du RM
  - intégration de TP
- **Types de RM**
  - gestionnaire de fichiers
  - SGBD
  - périphérique



Interface applicative TX

- tx_open
  - ordonne au TM d'initialiser la communication avec tous les RM dont les librairies d'accès ont été liées à l'application.
- tx_begin
  - ordonne au TM de demander aux RM de débuter une transaction.
- tx_commit ou tx_rollback
  - ordonne au TM de coordonner soit la validation soit l'abandon de la transaction sur tous les RM impliqués.
- tx_set_transaction_timeout
  - positionne un " timeout " sur les transactions
- tx_info
  - permet d'obtenir des informations sur le statut de la transaction.

Interface ressource XA

- xa_open
  - ouvre un contexte pour l'application.
- xa_start
  - débute une transaction.
- xa_end
  - indique au RM qu'il n'y aura plus de requêtes pour le compte de la transaction courante.
- xa_prepare
  - lance l'étape de préparation du commit à deux phases.
- xa_commit
  - valide la transaction.
- xa_rollback
  - abandonne la transaction.

Moniteurs transactionnels et SGBD

- Il y a deux possibilités pour la programmation d'un système transactionnel:
  - Programmation Client/Serveur, sans moniteur transactionnel, en relation avec les possibilités offertes par les SGBDs. Ceci est appelé "TP Lite" ou transactionnel léger
  - Utilisation d'un moniteur transactionnel qui fournit le cadre architectural des applications et utilisation des services fournis par différents composants logiciels (e.g. SGBDs). Ceci est appelé "TP Heavy" ou transactionnel lourd
- Pour le choix entre ces deux approches, différents éléments rentrent en ligne tels que:
  - Transactionnel léger : dépendance vis à vis du fournisseur de SGBD, limitations vis à vis de la programmation des transactions (la validation est faite au niveau du SGBD), problèmes de performance,...
  - Pour TP Heavy: limitation potentielle dans les progiciels que l'on peut intégrer, pérennité du moniteur, complexité de la programmation,....