Chapitre 1

Bases de Données Distribuées

Sommaire

1.1 Résumé

Un gestionnaire complet de bases de données distribuées implique qu'une application particulière soit capable d'opérer de façon "transparente" sur des données réparties dans diverses bases de données, gérées par divers SGBD, s'exécutant sur diverses machines, prises en charge par différents systèmes d'exploitation et connectées par divers réseaux de communication.

Transparent, signifie que l'application, d'un point de vue logique, opère comme si les données étaient gérées par un seul SGBD s'exécutant sur une seul machine.

1.2 Préliminaire

Un système de BDD distribué est constitué d'un ensemble de sites, interconnectés par un réseau de communication, dans lequel :

— Chaque site est un système de BDD à part entière.

— Tous les sites travaillent ensemble afin qu'un utilisateur puisse accéder, depuis n'importe quel site, aux données se trouvant n'importe où dans le réseau, comme si les données sont centralisées.

Une BDD distribuée correspond alors à un objet virtuel, dans les composants sont physiquement enregistrés dans un certain nombre de BDD réelles sur des sites distincts. Il s'agit de l'union logique de ces bases de données réelles.

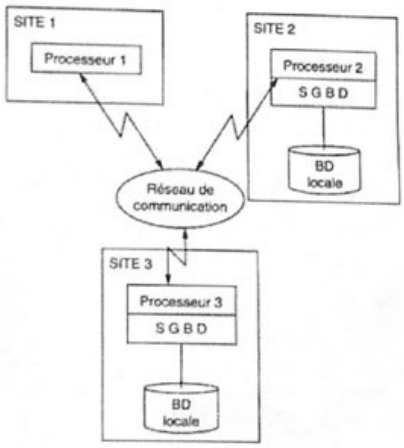

FIGURE 1.1 – Principe d'un SGBD distribué : système réparti + base de données distribuée (ensemble des bases de données locales)

1.2.1 Motivations

Pourquoi fait-on recours au BDD-Distribuées ?
- Le grand volume des données ;
- La taille des entreprises ;
- La puissance des micro-ordinateurs et des stations de travail ;
- La fiabilité et la souplesse des SGBD (notamment relationnels) ;
- Les performances des réseaux

Tout en préservant l'essentiel : la cohérence des bases de données (la difficulté).
La taille des entreprises : elles sont normalement déjà distribuées, au moins logiquement (en division, services, groupes de travail, ... etc.) et la même manière physiquement (en usines, ateliers laboratoire, ... etc.) d'où l'on déduire que les données sont déjà distribuées, car chaque unité organisationnelle dans l'entreprise doit nécessairement gérer les données pertinents pour son fonctionnement. Le capital informationnelle de l'entreprise est donc éclaté en ce que l'on appelle, (quelquefois), des îles d'information. Un système distribué permet une structuration de la base de données qui reflète la structure de l'entreprise : les données locales peuvent être conservées localement, là où les liens d'appartenance sont les plus forts alors que même temps, des données éloignées peuvent être accédées si nécessaire.

1.2.2 Objectifs

1. Autonomie locale : Les sites d'un système distribué doivent être autonomes. L'autonomie locale signifie que toutes les opérations d'un site donné sont contrôler par ce site. Le fait qu'une opération soit couronnée de succès sur un site X ne doit pas dépendre du site Y (i.e. le fait que le site Y tombe en panne ne doit pas entraîner le non-fonctionnement du site X). L'Autonomie locale implique également que les données locales soient possédées et gérées localement, avec une responsabilité locale ; La sécurité, l'intégrité et la représentation mémoire des données locales sont sous le contrôle du site local. L'objectif de l'autonomie locale n'est pas assuré à 100%, il existe certaines situation qui compromettent quelque peu cet objectif :
   - Les fragments individuels d'une relation fragmentée ne peuvent pas normalement être accédés directement, même depuis les sites où ils sont mémorisés (ce point nécessite une exemple de fragmentation.
   - Les copies individuelles d'une relation répliquée (ou fragment) ne peuvent pas normalement être accédées directement, même depuis les sites où ils sont mémorisés.
   - Soit R la copie répliquée d'une copie primaire P (P mémorisée sur le site X), alors chaque site accédant à R est dépendant de X (Une mise à jour de P nécessite la propagation des modifications à R).

— Une relation qui participe aux contrainte d'intégrité multisite ne peut être accédée, pour des besoins de mise à jour, dans le contexte local du site où elle est enregistrée, mais uniquement dans un contexte global de la BDD distribué dans laquelle la contrainte est définie.
— Un site qui intervient en tant qu'acteur dans le processus de commit à deux phase doit se conformer à la décision (commit ou annulation) du site coordinateur correspondant (Exemple de Transaction distribuée)

2. Pas de contrôle centralisé : l'autonomie locale implique que tous les sites doivent être considérés comme égaux ; ainsi, on ne fera pas confiance à un maître central pour certains services afin que l'ensemble du système ne dépende pas du site central (exemple : l'évaluation centraliser des requêtes, la gestion centralisée des transactions ou le service centralisé de nommage). Ce second objectif est donc un corollaire du premier. La centralisation n'est pas souhaitable pour deux raisons : *i*) le site central peut s'avéré un goulot d'étranglement, *ii*) le système peut être vulnérable en ca de panne du site central.
3. Continuité de service : l'un des avantages des systèmes distribuées est qu'ils devraient apporter une grande fiabilité et une plus grande disponibilité.
— La fiabilité [1] est améliorer car les systèmes distribués ne sont pas des solution tout à rien, ils peuvent continuer à fonction (même en mode dégradé) en cas de défaillance d'un compostant (ou d'un site).
— La disponibilité [2] est améliorée grâce à la réplication des données.
Ces deux considération s'appliquent au cas où un arrêt est imprévu (panne quelconque) (les imprévus sont indésirables mais inévitables et difficile à éliminer complètement). En revanche, les arrêts prévus ne devraient jamais être nécessaire ; c-à-d le système ne devrait pas nécessite d'arrêt pour effectuer une maintenance (ajout de service, de SGBD, de BDD ...).
4. Indépendance de l'emplacement (ou transparence d'emplacement ou de localisation) : les utilisateurs doivent pouvoir se comporter (au moins d'un point de vue logique) comme si les données étaient toute mémorisées sur leur propre site. L'objectif est de simplifier les programmes utilisateurs et de permettre une migration de données d'un site à un autre sans invalider ces programmes. Notant que la migration est utile et permet de déplacer les données à l'intérieur du réseau pour répondre à l'évolution des exigences de performance.
5. Indépendance de la fragmentation : Un système gère la fragmentation des données si une relation mémoire (i.e. Table) peut être divisée en plusieurs morceaux "ou fragment" pour les besoin de mémorisation physique. La fragmentation est souhaitable pour des raisons de performance : les données peuvent

1. La probabilité que le système fonctionne à n'importe quel moment
2. La probabilité que le système fonctionne en continu pendant une période spécifique

être mémorisées à l'endroit où elle sont le plus fréquemment utilisées, de sorte que la plupart des opérations sont entièrement locales et la circulation des données est réduite. .

6. Indépendance de la réplication : Un système gère la réplication des données (ou des fragments) si une relation mémorisée peut être représentée par plusieurs copies ou *duplicatas* mémorisées sur des sites différents.
   Les utilisateurs n'ont pas à savoir si plusieurs copies d'une même information sont disponibles. C'est le principe de transparence de duplication. La conséquence directe est que lors de la modification d'une information, c'est le système qui doit se préoccuper de mettre à jour toutes les copies.
7. Evaluation des requêtes distribuées : *i*) Le système doit être en mesure d'envoyer (ou diffusion) des requêtes sur les sites et de récupérer les résultats. *ii*) L'optimisation : est plus importante dans un système distribué que dans un système centralisé. Le point essentiel est que dans une requête faisant intervenir plusieurs sites, il y'aura plusieurs façon de déplacer les données dans le réseau pour satisfaire la requête et il est crucial de trouver une stratégie efficace (exemple de requête d'intersection ou d'union entre deux sites).
8. Gestion des transaction distribuées : La gestion des transaction a deux aspects : le contrôle de la reprise après panne et le contrôle de la concurrence. Dans un système distribué, une seule transaction peut nécessiter l'exécution du code sur plusieurs sites (précisément, elle peut nécessité des mises à jour sur plusieurs sites. Par conséquence, on dit que chaque transaction est constituée de plusieurs agents, ou un agent est le processus exécuté pour le compte d'une transaction sur un site donné. Le système a besoin de savoir si deux agents font partie de la même transaction (généralement, deux agents d'une même transaction ne doivent se retrouver en interblocage sur un même site).
   Pour la reprise après panne : Pour garantir qu'une transaction est atomique (tout ou rien) dans un système distribué, le système doit donc garantir que l'ensemble des agents de cette transaction sont tous validés à l'unanimité ou bien tous annulés à l'unanimité (usage du protocole Commit à deux phases)
   Le contrôle de la concurrence : est en principe fondé sur le verrouillage.
9. Indépendance matérielle : des machines de différents constructeurs (IBM, PC, HP, ...) participent au système distribué.
   Il existe un besoin réel de pouvoir intégrer les données sur tous ces systèmes et présenter à l'utilisateur une "image système unique".
10. Indépendance des systèmes d'exploitation : cet objectif est en corollaire du précédent. Il est souhaitable de pouvoir exécuter le SGBD sur les différentes plates-formes.
11. Indépendance du réseau : le système doit pouvoir gérer un ensemble de réseaux de communication de grande diversité (i.e. composé d'éléments hétérogène ou

disparate).

12. Indépendance du SGBD : l'hypothèse l'homogénéité stricte est très forte ; ce qu'on a besoin est que les SGBD installer sur différents sites gèrent la même interface (SQL par exemple). Il y a la possibilité de prendre en charge les hétérogénéité à travers les passerelles ou les Middleware.
Une base de données répartie ne doit pas être dépendante des différents systèmes de gestion de bases de données. La relation globale doit pouvoir être exprimée dans un langage normalisé indépendant des constructeurs.

1.3 Les problèmes (ou les difficultés) des systèmes de distribués

Evaluation des requêtes : Le besoin de minimiser l'utilisation du réseau a pour conséquence que le processus d'optimisation des requêtes lui-même est distribué, de même que le processus d'exécution de requêtes. Autrement dit, le processus global d'optimisation consistera en un étape d'optimisation globale suivie par des étapes d'optimisation locale sur chaque site concerné.

— La gestion du catalogue : Dans un système distribué, le catalogue du système contiendra, nous seulement les donnée d'un catalogue classique concernant les relations de base, les vues, les index, les utilisateurs,... etc. mais également toutes les informations de contrôle nécessaire au système pour qu'il puisse assurer l'indépendance de l'emplacement, de la fragmentation et de la réplication. Cela pose le problème : où et comment le catalogue doit-il être mémorisé ?

1. Centralisé : le catalogue est mémorisé (une seule fois) sur un site central unique.
2. Réplication Totale : l'ensemble de catalogue est mémorisé sur chaque site.
3. Partitionné : Chaque site gère son propre catalogue. Le catalogue global est l'Union de tous ces catalogues locaux.
4. Combinaison de 1 et 3 : chaque site gère son catalogue local (comme dans 3), et un site central unique gère une copie unifiée de tous les catalogues locaux.

Chaque approche à ses problèmes :

— L'approche 1 viole l'objectif "pas de contrôle centralisé".
— L'approche 2 souffre d'une perte importante d'autonomie ; chaque mise à jour d'un catalogue doit être propagée vers tous les sites.
— L'approche 3 rend les opérations non locales très coûteuses (trouver un objet distant nécessite, en moyenne, un accès à la moitié des sites).
— L'approche 4, est plus efficace que la $3^{ème}$, mais viole de nouveau l'objectif "pas de contrôle centralisé".

- La propagation des mises à jour : le problème de base avec la réplication est qu'une mise à jour d'un objet logique doit être propagée à toutes les copies de cet objet. Il apparaît alors le problème d'inaccessibilité de certains sites détenant des copies d'objet en question (à cause d'une défaillance des sites ou du réseau). Une des stratégie pour traiter ce problème est la stratégie appelée "copie primaire", qui fonctionne de la façon suivante :
  - Une des copies d'un objet répliqué est considérée comme copie primaire et les autres comme copies secondaires.
  - Les copies primaires d'objets différentes sont sur des sites différents (à la mesure du possible).
  - Une mise à jour est censée logiquement achevée (ou complète) dès que la copie primaire a été mise à jour. Le site détenant la copie primaire se charge (et il est responsable) de la propagation de la mise à jour vers les copies secondaires à une date ultérieure. (il est important que cette date doit précédée l'execution de COMMIT d'une transaction distribuée pour préserver la propriété ACID).
- Le contrôle de reprise : il est typiquement fondé sur le protocole de validation à deux phases, qui est nécessaire dans toute environnement où une transaction peut interagir avec plusieurs SGBD autonomes.
  En cas de panne, et après reprise d'un participant, il doit consulter le coordinateur sur la décision à prendre (commit ou rollback)
- Le contrôle de concurrence : il est fondé (dans la plupart des systèmes distribués) sur le verrouillage. Cependant, les requêtes destinées à tester, positionner ou abandonner des verrous devient des messages, les messages impliquent une dégradation des performances.
  Par exemple, pour n sites participants :
  - n requêtes de verrouillage
  - n attributions de verrous (message envoyé par un participant au coordinateur)
  - n messages de mise à jour
  - n accusés de réception
  - n message d'abandon de verrou

  Un autre problème avec le verrouillage dans un système distribué est qu'il peut conduire à un blocage global qui fait intervenir deux ou plusieurs sites.
  Exemple

  1. L'agent de la transaction T_2 sur le site A attend que l'agent de la transaction T_1 sur le site A abandonne un verrou.
  2. L'agent de la transaction T_1 sur le site A attend que l'agent de la transaction T_1 sur le site B ait terminé.
  3. L'agent de la transaction T_1 sur le site B attend que l'agent de la transac-

tion T_2 sur le site A abandonne un verrou.

4. L'agent de la transaction T_2 sur le site B attend que l'agent de la transaction T_2 sur le site A ait terminé.

Le problème posé par ce type de blocage est que aucun site ne peut le détecter en se basant uniquement sur les informations internes, (i.e. il n'y a pas de cycle dans le graphe d'attente local, mais le cycle apparaît lorsque deux graphes locaux sont combinés en un graphe d'attente global. Il s'ensuit que la détection des blocages globaux fait courir le risque d'une charge de communication supplémentaire pour réunir les graphes locaux. Tous les blocage ne peuvent être détecter, solution : mécanisme de timeout (ou péremption) [3].

1.4 Architecture

Dans un SGBD Réparti, il apparaît une dualité (coexistence) entre le niveau global qui ouvre l'accès à la BD répartie, et le niveau local où interviennent les différents SGBD qui manipulent directement les bases de données locales.

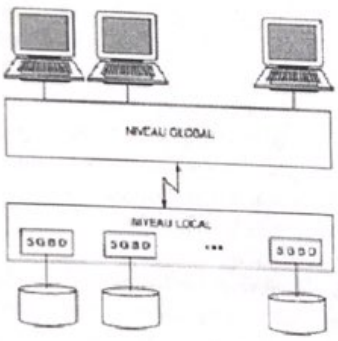

FIGURE 1.2 – Niveaux local et global d'un SGBD Réparti

1.4.1 Interface d'une BDD Répartie

Une BDR est décrite dans un dictionnaire de données sous la forme de schémas globaux distincts conformément à l'architecture ANSI/SPARC :

3. Conditions d'Apparition d'Interblocage : Exclusion mutuelle, Possession et attente,Sans péremption, Attente circulaire.

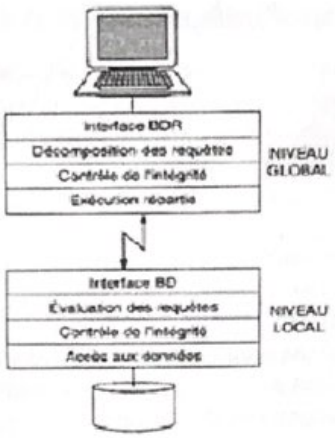

FIGURE 1.3 – Décomposition fonctionnelle d'un SGBD Réparti

- **Les schémas externes** où les données sont décrites sous forme de vues, chacune d'elles étant adaptée à une classe particulière d'utilisateurs ; un schéma externe, élaboré à partir du schéma conceptuel, peut mixer des données stockées dans différentes bases ;
- **Le schéma conceptuel** où les données sont représentées sans prendre en compte des contraintes techniques ou de mise en forme ; toutes les données sont décrites (dans ce schéma) indépendamment de leur localisation ;
  Le schéma interne où sont spécifiées la fragmentation des données et la localisation de ces fragments ; les données y sont décrites en fonction de l'architecture du système réparti et des spécificités techniques du système informatique (diversité des matériels et des modèles de données, architecture du réseau,...).

1.4.2 Décomposition des requêtes

Un traitement réparti fait appel à des données gérées par des SGBD distincts. Un traitement réparti contient donc des requêtes formulées à partir d'un schéma externe global ; ces requêtes correspondent à un ensemble d'opérations de recherches et de mises à jour sur des données de la BD Répartie. Le SGBD Réparti contrôle et analyse chaque requête et la décompose en opérations locales (plan d'exécution réparti) qui seront soumises pour exécution aux SGBD concernés (cet aspect sera développé dans le chapitre Traitement des requêtes).

1.4.3 Contrôle de l'intégrité

Le contrôle de l'intégrité des données permet d'assurer que tout traitement (ou transaction) sur la base de données fait passer celle-ci d'un état cohérent à un autre état cohérent sans garantir la cohérence des états intermédiaires. Les sources pouvant engendrées des anomalies sont Nombreuses :

- Absence d'expression de certaines contraintes d'intégrité dans le schéma des données.
- Perte d'opérations suite à un enchevêtrement (Extrême complication, désordre, confusion) de mises à jour concurrentes.
- Panne du réseau, etc.

Ce type de problème existe aussi dans les SGBD centralisés qui proposent des solutions adaptées notamment pour gérer les accès concurrents aux données (techniques d'évitement ou de détection des incohérences).

1.5 Systèmes de gestion de bases de données répartis (SGBD Réparti)

En général, un SGBD réparti doit être capable d'offrir les mêmes services qu'un SGBD centralisé, en déchargeant les utilisateurs de tous les problèmes de concurrence, de fiabilité et d'optimisation de requêtes. Ainsi, un SGBD réparti doit disposer d'un *Dictionnaire de données réparties*, et il doit pouvoir :

- Traiter de requêtes réparties.
- Gérer des transactions réparties.
- Assurer la communication de les échanges de données inter-site.
- Gérer de la cohérence.
- Gérer la sécurité.

Le SGBD réparti reçoit des requêtes référençant des tables d'une base de données réparties. Il assure la réécriture des requêtes distribuées en plusieurs sous−requêtes locales envoyées à chaque site. La réécriture de requête est une décomposition qui prend en compte les règles de localisation.

Pour ce qui concerne les mises à jour, le SGBD Réoparti doit assurer la gestion des transactions réparties, en prenant en compte la vérification des règles d'intégrité multi-bases, le contrôle des accès concurrents et surtout la gestion de l'atomicité des transactions distribuées. Le SGBD réparti peut utiliser les fonctions locales de gestion de transactions pour accomplir les fonctions globales.

1.6 Conception des bases de données réparties

L'existence de SGBD Réparti, aussi sophistiqué soit-il, ne dispense pas l'utilisateur (l'administrateur des données) de concevoir la BDD Réparti, c'est-à-dire de définir la structure de la base de données et les opérations qui lui sont applicables. Le problème de conception est cependant différent selon que l'on crée de toute pièce une BDD Répartie (démarche descendante) ou bien que l'on constitue une BDD Répartie par agrégation de bases de données existantes (démarche ascendante).

1.6.1 Démarche descendante

Aux niveaux conceptuel et externe, la BDD Répartie est perçue comme une base de données centralisée ; les processus de conception classiques pour bases de données centralisées s'appliquent donc aux niveaux externe, global et conceptuel global. Toute la difficulté réside donc dans le niveau interne global où, considérant la BDD Répartie comme un ensemble de relations, on spécifie :

1. La fragmentation des relations en unités de localisation ;
2. La localisation de ces fragments dans le réseau.

L'ensemble des fragments stockés sur un site donné correspond à une base de données locale. La figure présente l'architecture d'une BDD Répartie. On constate la présence des trois niveaux de l'architecture ANSI/SPARC (externe, conceptuel et interne) à la fois au niveau global (celui de la BDD Répartie) et au niveau local (celui des BDD) (voir figure 1.4.

1.6.1.1 Fragmentation

Grâce à sa simplicité, à l'universalité de ses concepts et surtout à l'algèbre qui lui est associée, le modèle relationnel se prête bien à l'étude de la répartition des données. Si bien que l'on a qualifié ce modèle de pivot en lui faisant jouer un rôle central : les bases de données conçues avec d'autres modèles (hiérarchique et réseau) sont alors traduites en termes relationnels pour les intégrer dans le système réparti. Pour fragmenter une relation globale sans perte d'information, il suffit d'appliquer à cette relation l'opération algébrique de restriction (fragmentation horizontale) ou celle de projection (fragmentation verticale). Les opérations de jointure et d'union permettent ensuite de reconstituer la relation initiale.

1.6.1.2 Localisation des fragments

Dans la démarche descendante, la localisation des fragments est transparente pour l'utilisateur. Ainsi, la répartition des fragments dans les différents sites du réseau répond à des critères essentiellement techniques tels que le coût de stockage, le coût

de mise à jour, la performance d'accès. Par exemple, chaque fois qu'un fragment est dupliqué dans le réseau, les applications peuvent accéder plus aisément à ce fragment mais, en contrepartie, le coût de mise à jour est plus élevé (propagation des modifications aux différentes copies). Les SGBD Répartis proposent généralement des modèles d'allocation des données qui sont paramétrables en fonction des contraintes liées à l'application (taux de mise à jour, temps de réponse,...).

1.6.2 Démarche ascendante

L'intérêt de cette démarche est de constituer une BDD Répartie à partir de BDD existantes. Évidemment, c'est dans le cadre d'une telle démarche que se pose plus particulièrement le problème de la coopération de SGBD hétérogènes (et éventuellement des BDD hétérogènes). L'architecture de la BDD Répartie, si elle suit les niveaux proposés par l'ANSI/SPARC, n'en est pas moins distincte de celle de la démarche descendante. En effet, les bases de données locales sont considérées comme des entités connues des utilisateurs ; le SGBD Réparti leur offre la possibilité d'opérer sur des données appartenant à plusieurs bases distinctes (on parle aussi de coopération de BDD ou de multibase). Ainsi, au niveau conceptuel, la BDD Répartie est définie comme un ensemble de BDD entre lesquelles peuvent être définies des associations et diverses contraintes d'intégrité. À ce niveau, la localisation des BDD reste inconnue. Au niveau externe, les vues peuvent faire apparaître, ou non, la multiplicité des BDD, selon le souhait de l'utilisateur.
Le niveau interne n'autorise pas une allocation des données aussi fine que dans l'architecture précédente. En effet, les bases de données locales étant existantes, (généralement) seule est permise une duplication de la totalité d'une base de données (voir figure 1.5.

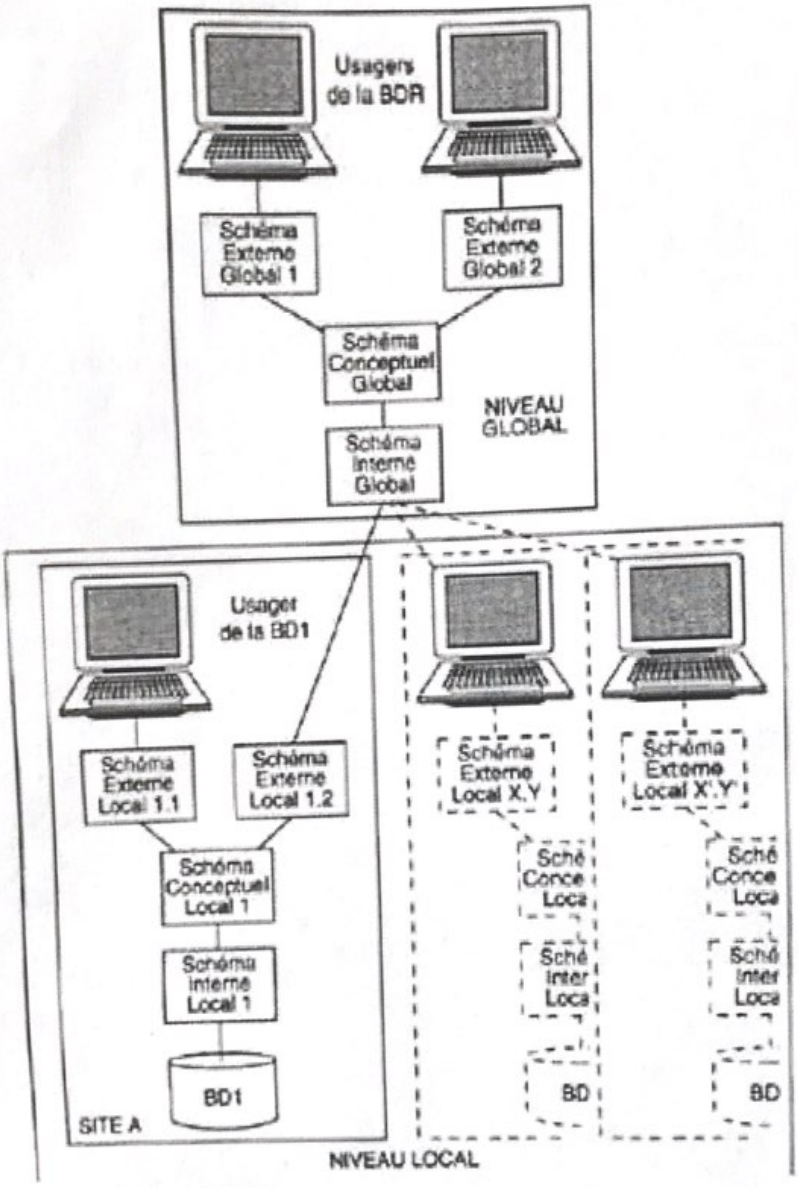

FIGURE 1.4 – Architecture d'une BDD Répartie selon la démarche descendante

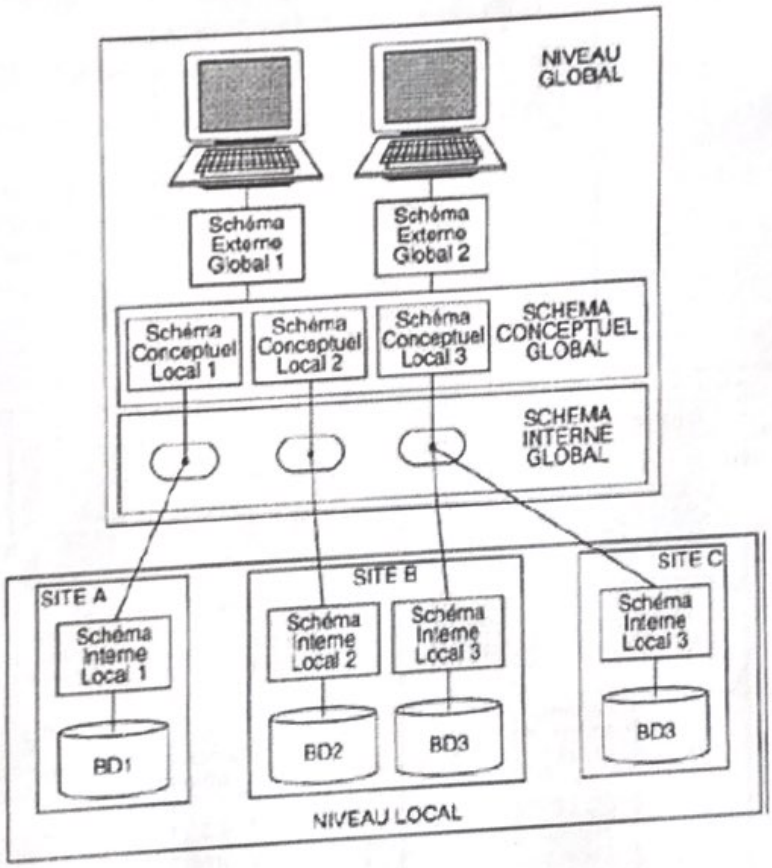

FIGURE 1.5 – Architecture d'une BDDR selon la démarche ascendante