Base de données distribuées

Babacar Diop

Dpt. d'Informatique

UFR des Sciences Appliquées et de Technologies

Université Gaston Berger de Saint-Louis

2018/2019

Base de données distribuées

Définition

- Plusieurs bases de données réparties géographiquement
- Une BD distribuée est un ensemble de plusieurs BDs interconnectées, réparties physiquement sur divers emplacements et communiquant via un réseau informatique
- Un SGBD distribué gère les BD distribuées de sorte qu'elle apparaisse sous la forme d'une base de données unique pour les utilisateurs

Base de données distribuées

Caractéristiques

- Les BDD sont logiquement liées les unes aux autres pour représenter une seule base de données logique
- Les données sont stockées physiquement sur plusieurs sites et gérées par un SGBD indépendant des autres sites
- Incorpore un traitement de transaction, mais ce n'est pas synonyme d'un système de traitement de transaction

Base de données distribuées

SGBDD

- Crée, récupère, met à jour et supprime les BDD
- Synchronise périodiquement la BD et fournit des mécanismes d'accès grâce auxquels la distribution devient transparente pour les utilisateurs
- Met à jour les données modifiées sur n'importe quel site universellement
- Il maintient la confidentialité et l'intégrité des données des bases de données

Base de données distribuées

Motivations

- **Nature distribuée des unités organisationnelles** - À l'heure actuelle, la plupart des organisations sont subdivisées en plusieurs unités physiquement réparties à travers le monde. Chaque unité nécessite son propre ensemble de données locales. Ainsi, la BD globale de l'organisation devient distribuée

- **Besoin de partage de données** - Les multiples unités organisationnelles doivent souvent communiquer entre elles et partager leurs données et leurs ressources. Cela nécessite des BDs communes ou répliquées qui doivent être utilisées de manière synchronisée

Base de données distribuées

Motivations

- **Prise en charge d'OLTP et OLAP** - le traitement des transactions en ligne (OLTP) et le traitement analytique en ligne (OLAP) fonctionnent sur des systèmes diversifiés pouvant contenir des données communes. Les systèmes de BDs distribués facilitent ces deux traitements en fournissant des données synchronisées
- **Récupération de base de données** - L'une des techniques couramment utilisées est la réplication de données sur différents sites. Ainsi, une défaillance de la BD peut devenir presque invisible pour les utilisateurs

Base de données distribuées

Motivations

- Prise en charge de plusieurs logiciels d'application –
  - La plupart des organisations utilisent divers logiciels d'application, chacun avec son support de base de données spécifique. Un SGBDD fournit une fonctionnalité uniforme pour utiliser les mêmes données sur différentes plates-formes

Base de données distribuées

Avantages

- **Développement modulaire** – l'extension à de nouvelles unités dans des systèmes de BDs centralisés nécessite des efforts importants et une perturbation du fonctionnement existant
- Dans les BDD, le travail consiste simplement à ajouter de nouveaux composants et des données locales au nouveau site et à les connecter au SD, sans interruption des fonctions actuelles

Base de données distribuées

Avantages

- **Plus fiable** - En cas de défaillance de la base de données, l'ensemble du système de BD centralisées s'arrête. Toutefois, dans les systèmes distribués, en cas de défaillance d'un composant, les performances du système peuvent continuer à diminuer. Par conséquent, un SGBDD est plus fiable.

Base de données distribuées

Avantages

- **Meilleure réponse** - Si les données sont distribuées de manière efficace, les demandes des utilisateurs peuvent être satisfaites à partir des données locales elles-mêmes, permettant ainsi une réponse plus rapide. En revanche, dans les systèmes centralisés, toutes les requêtes doivent passer par l'ordinateur central pour être traitées, ce qui augmente le temps de réponse.

Base de données distribuées

Avantages

- **Coût de communication réduit** - Dans les systèmes de BD distribués, si les données sont locales et utilisées, les coûts de communication pour la manipulation des données peuvent être minimisés. Ce n'est pas faisable dans les systèmes centralisés

Base de données distribuées

Possibles inconvénients

- Besoin de logiciels complexes et coûteux
- Traitement supplémentaire - Même les opérations les plus simples peuvent nécessiter un grand nombre de communications et des calculs supplémentaires pour uniformiser les données sur les sites

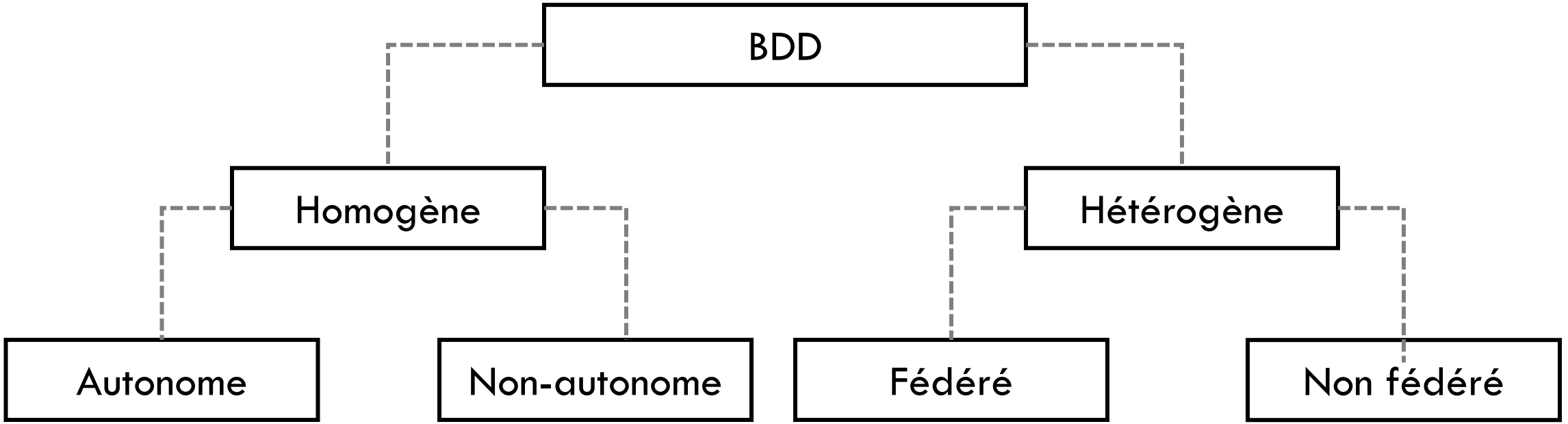
Base de données distribuées

Possibles inconvénients

- **Intégrité des données** - La nécessité de mettre à jour les données sur plusieurs sites pose des problèmes d'intégrité des données
- **Autres problèmes en cas de mauvaise distribution des données** - La réactivité des requêtes dépend en grande partie de la bonne répartition des données. Une mauvaise distribution des données entraîne souvent une réponse très lente

Base de données distribuées

Catégories de BDD



BDD homogènes

Concepts

- Tous les sites utilisent :
  - un SGDB et un OS identiques
  - des logiciels très similaires
  - des bases de données identiques
- Chaque site est informé de tous les autres sites et coopère avec d'autres sites pour traiter les demandes des utilisateurs
- La BD est accessible via une interface unique, comme s'il s'agissait d'une BD unique

BDD homogènes

Catégories

- Il existe deux types de BDD homogène
  - Autonome - Chaque BD est indépendante et fonctionne de manière autonome. Ellles sont intégrés par une application de contrôle et utilisent la transmission de messages pour partager les mises à jour des données
  - Non autonome - Les données sont réparties sur les nœuds homogènes et un SGBD central ou principal coordonne les mises à jour des données sur les sites

BDD hétérogènes

Concepts

- Dans une BDD hétérogène, différents sites ont des systèmes d'exploitation, des produits de SGBD et des modèles de données différents
- Différents sites utilisent des schémas et des logiciels différents (relationnel, réseau, hiérarchique, orienté objet, etc.)
- Traitement complexe de requêtes en raison de schémas différents
- Traitement complexe des transactions en raison de logiciels différents
- Un site peut ne pas être au courant d'autres sites et la coopération dans le traitement des demandes des utilisateurs est donc limitée

BDD hétérogènes

Catégories

- **Fédéré** - Les systèmes de BD hétérogènes sont de nature indépendante et intégrés ensemble, de sorte qu'ils fonctionnent comme un système de BD unique

- **Non fédéré** - Les systèmes de BD utilisent un module de coordination central permettant d'accéder aux BDs

Architecture

Architecture des BDD

Concepts

- Architectures développées en fonction de 3 paramètres
  - Distribution - Indique la distribution physique des données sur les différents sites
  - Autonomie - Indique la répartition du contrôle du système de base de données et le degré de fonctionnement indépendant de chaque SGBD constitutif
  - Hétérogénéité - Il s'agit de l'uniformité des modèles de données, des composants système et des bases de données

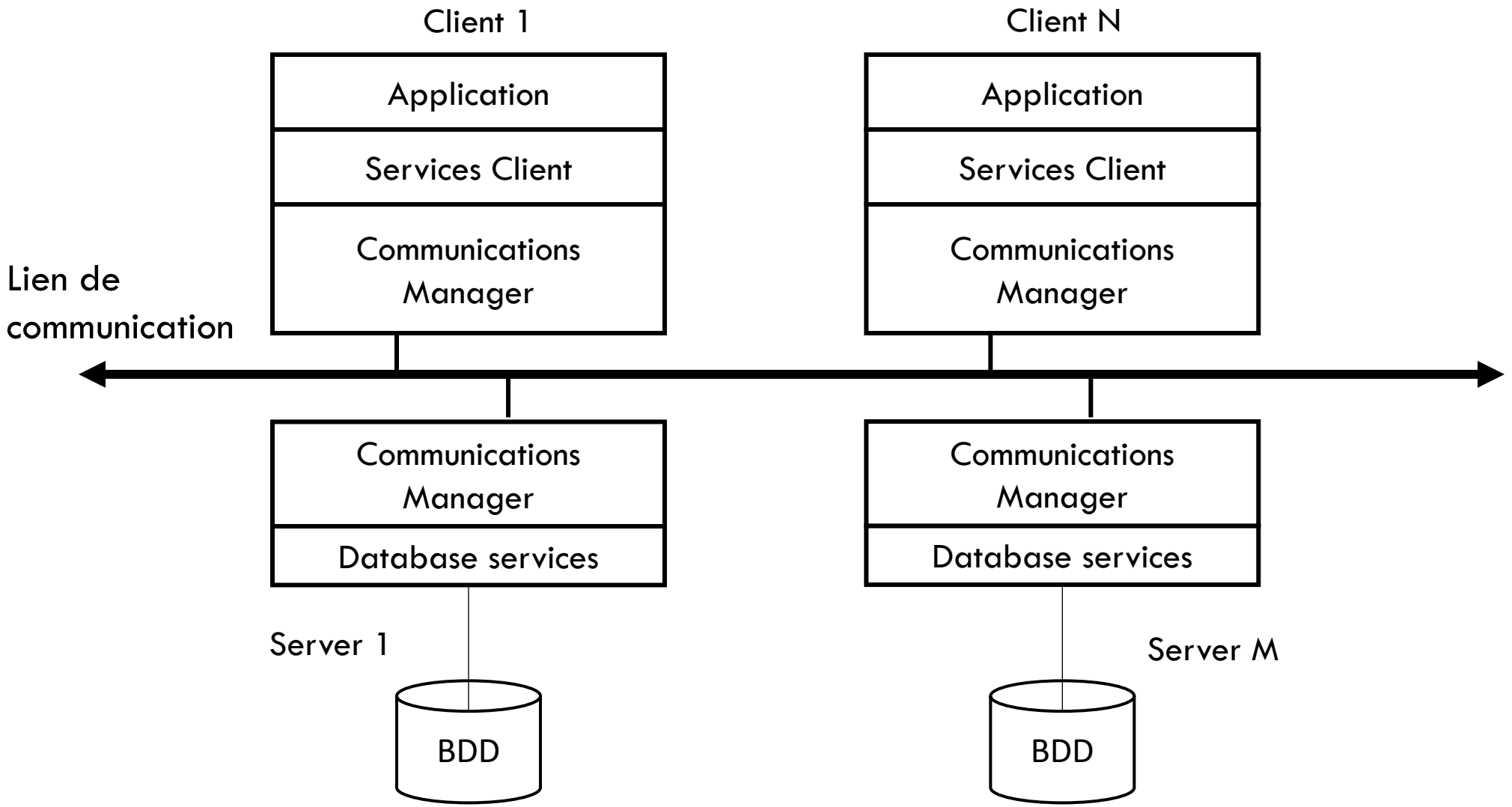
Architecture des BDD

Modèles

- Les modèles peuvent être :
  - Client - Server
  - Peer - to - Peer
  - Multi - SGBD Architecture

Architecture des BDD

Client-serveur



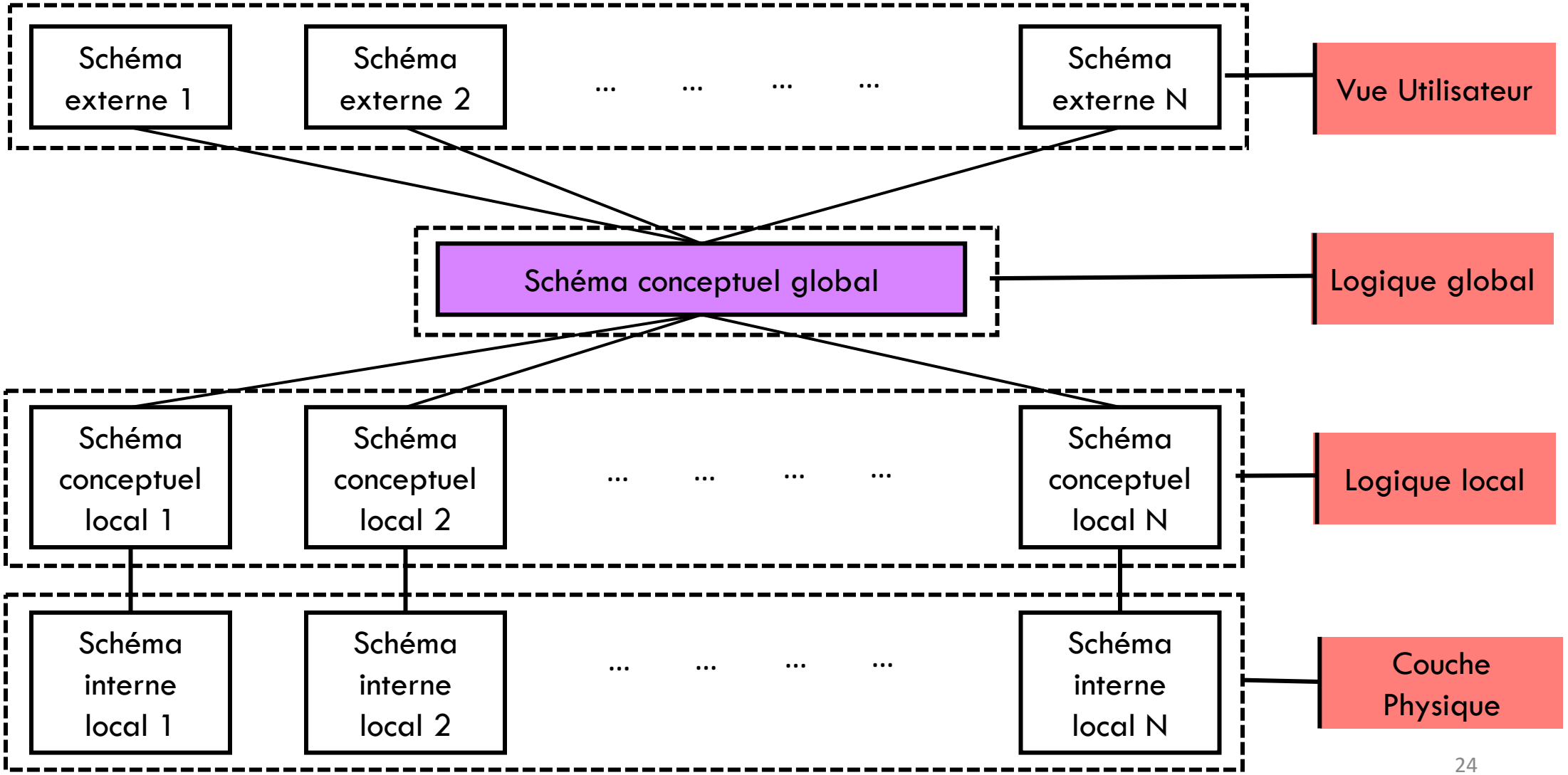
Architecture des BDD

Peer-to-peer

- Dans ces systèmes, chaque pair agit à la fois comme client et serveur
- Les pairs se partagent leurs ressources et coordonnent leurs activités
- Architecture généralement à quatre niveaux de schémas
  - Schéma conceptuel global représentant la vue logique globale des données
  - Schéma conceptuel local décrivant l'organisation des données logiques sur chaque site
  - Schéma interne local décrivant l'organisation des données physiques sur chaque site
  - Schéma externe décrivant la vue utilisateur des données

Architecture des BDD

Peer-to-peer



Architecture des BDD

Multi-SGBD

- Il s'agit d'un système de base de données intégré constitué d'un ensemble de deux ou plusieurs systèmes de base de données autonomes

Architecture des BDD

Multi-SGBD

- Trois niveaux de schémas globaux
  - Niveau de vue multi-SGBD- décrit plusieurs **vues** d'utilisateur comprenant des sous-ensembles de la base de données distribuée intégrée
  - Niveau conceptuel multi-SGBD- représente une multi-base de données intégrée comprenant des **définitions globales logiques** de structure de plusieurs bases de données
  - Niveau interne multi-SGBD - décrit la **répartition** des données sur différents sites et le **mappage** de plusieurs bases de données vers des données locales

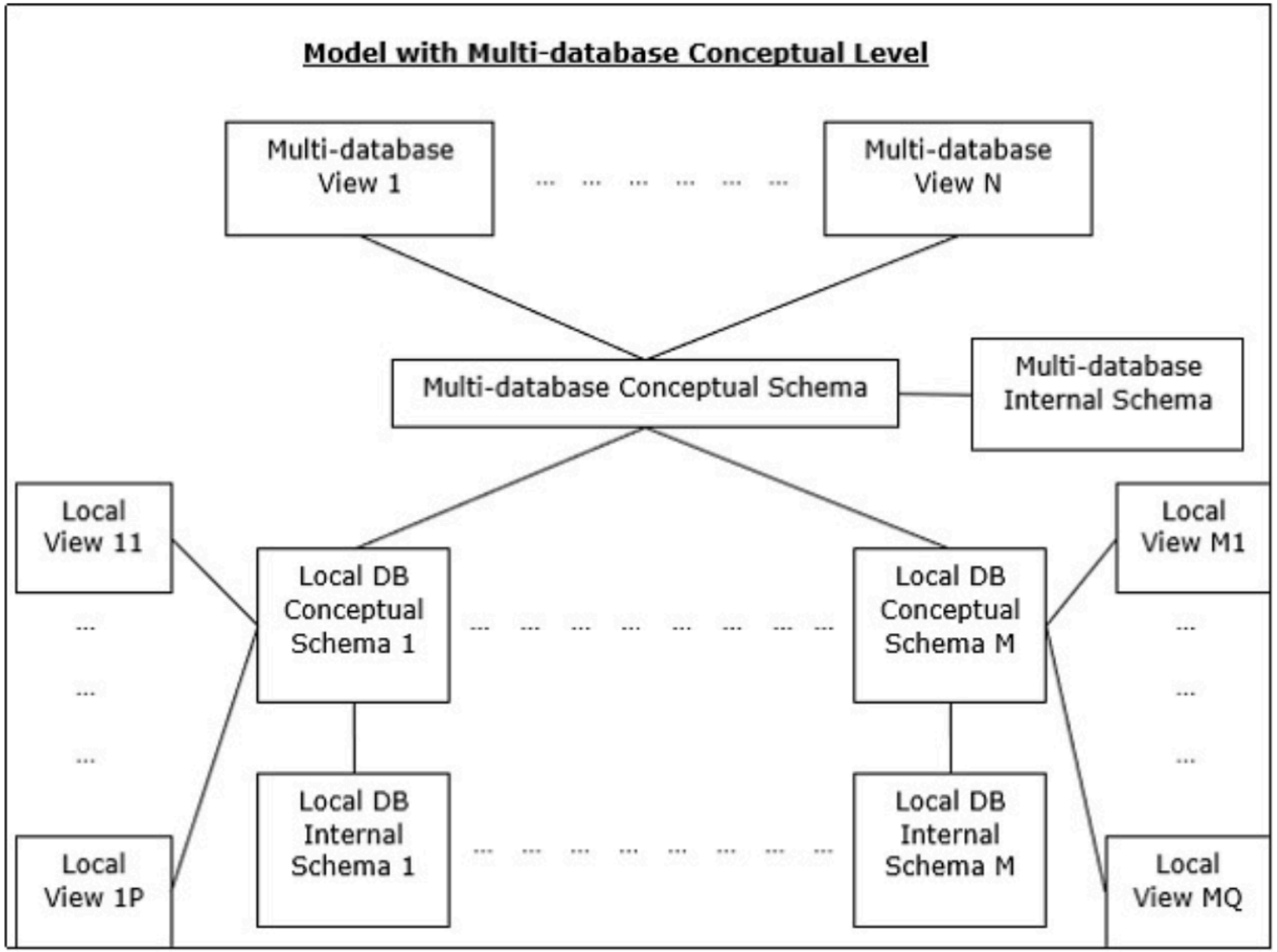
Architecture des BDD

Multi-SGBD

- Trois niveaux de schémas locaux
  - **Niveau de la base de données locale** - décrit la vue publique des données locales
  - **Niveau conceptuel de la base de données locale** - décrit l'organisation des données locales sur chaque site
  - **Base de données locale Niveau interne** - décrit l'organisation des données physiques sur chaque site

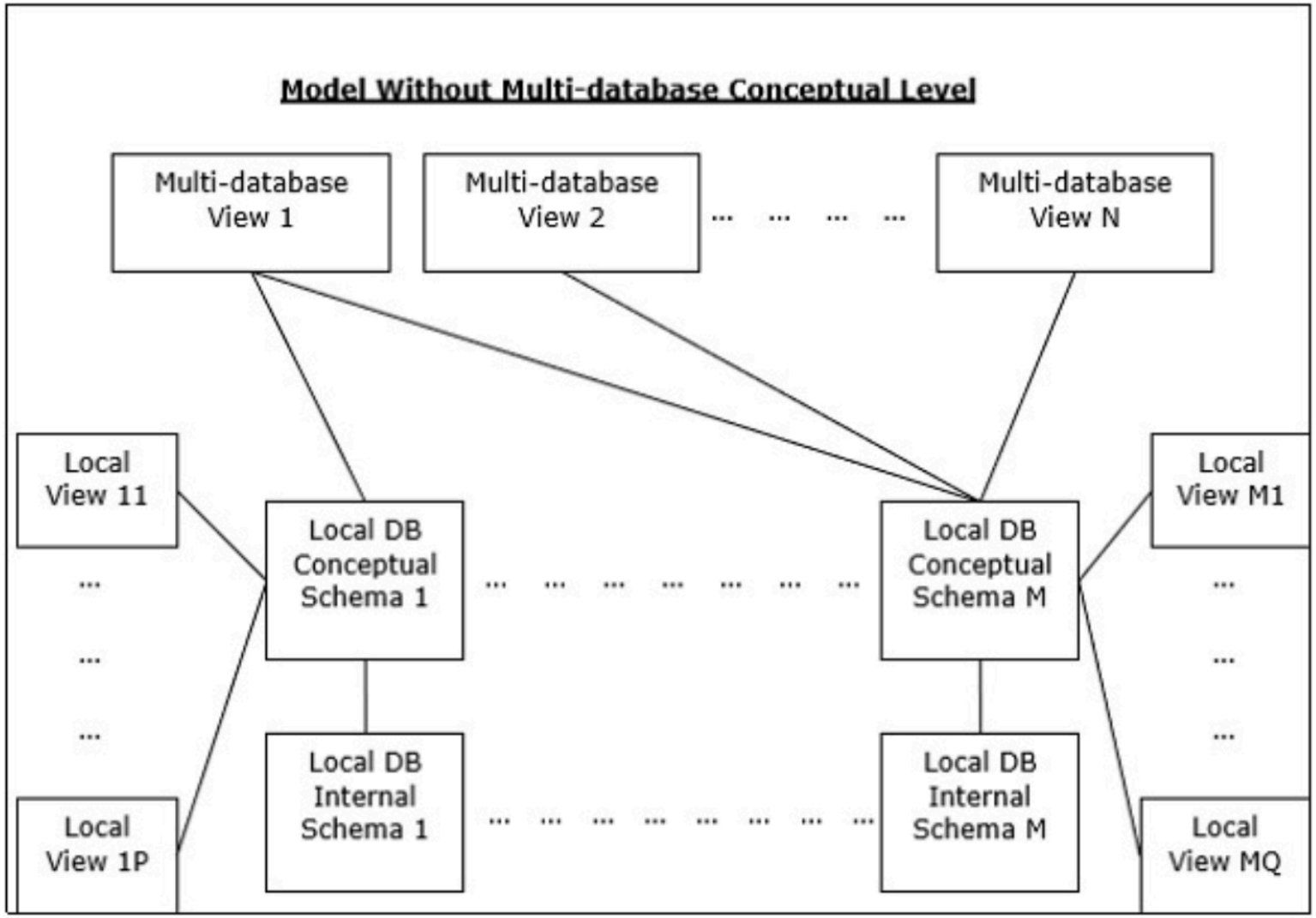
Architecture des BDD

Multi-SGBD avec niveau conceptuel



Architecture des BDD

Multi-SGBD sans niveau conceptuel



Architecture des BDD
D'autres architectures …

- Les alternatives de conception de la distribution pour les tables d'un SGBDR sont les suivantes:
  - Non répliqué et non fragmenté
  - Entièrement répliqué
  - Partiellement répliqué
  - Fragmenté
  - Mixte

Architecture des BDD

Non répliqué et non fragmenté

- Différentes tables sont placées sur différents sites de manière à ce qu'elles se trouvent à proximité du site où elles sont le plus utilisées
- Convient mieux aux SGBD où le pourcentage de requêtes nécessaires pour joindre des informations dans des tables placées sur différents sites est faible
- Si une stratégie de distribution appropriée est adoptée, cette alternative de conception permet de réduire les coûts de communication lors du traitement des données

Architecture des BDD

Entièrement répliqué

- Sur chaque site, une copie de toutes les tables de la base de données est stockée
- Les requêtes deviennent très rapides et ne nécessitent qu'un coût de communication négligeable
- Au contraire, la redondance massive des données nécessite des coûts énormes lors des opérations de mise à jour
- Par conséquent, cela convient aux systèmes où un grand nombre de requêtes doit être traité avec un nombre de mises à jour faible

Architecture des BDD

Partiellement répliqué

- Des copies de tables ou des portions de tables sont stockées sur différents sites
- La distribution des tables est faite en fonction de la fréquence d'accès
- Cela tient compte du fait que la fréquence d'accès aux tables varie considérablement d'un site à l'autre
- Le nombre de copies des tables (ou parties) dépend de la fréquence d'exécution des requêtes d'accès et du site qui génère les requêtes d'accès

Architecture des BDD

Fragmenté

- Une table est divisée en deux ou plusieurs pièces appelées fragments ou partitions, et chaque fragment peut être stocké sur un site différent
- Cela tient compte du fait qu'il arrive rarement que toutes les données stockées dans une table soient requises sur un site donné
- De plus, la fragmentation augmente le parallélisme et permet une meilleure reprise après panne

Architecture des BDD

Fragmenté

- Les trois techniques de fragmentation sont
  - Fragmentation verticale
  - Fragmentation horizontale
  - Fragmentation hybride

Architecture des BDD

Distribution mixte

- Combinaison de fragmentation et de réplications partielles
- Les tables sont initialement fragmentées sous n'importe quelle forme (horizontale ou verticale), puis ces fragments sont partiellement répliqués sur les différents sites en fonction de la fréquence d'accès aux fragments

Architecture des BDD

Réplication

- La réplication de données consiste à stocker des copies distinctes de la base de données sur deux sites ou plus. C'est une technique populaire pour pallier à la tolérance aux pannes des bases de données

Architecture des BDD

Avantages de la réplication

- Fiabilité en cas de défaillance d'un site
- Réduction de la charge du réseau
- Temps de réponse plus rapide
- Transactions simplifiées avec moins de jointures de tables situées sur différents sites

Architecture des BDD

Inconvénients de la réplication

- Augmentation des exigences de stockage en termes de coûts
- Augmentation du coût et de la complexité de la mise à jour des données
- Couplage entre applications et base de données –
  - si des mécanismes de mise à jour complexes ne sont pas utilisés, la suppression des incohérences de données nécessite une coordination complexe au niveau de l'application

Architecture des BDD

Techniques de réplication

- Certaines techniques de réplication couramment utilisées sont
  - Réplication instantané
  - Réplication en temps quasi réel
  - Pull replication

Architecture des BDD

Fragmentation

- La fragmentation consiste à diviser une table en un ensemble de tables plus petites appelées fragments.
- La fragmentation peut être de trois types:
  - horizontale,
  - verticale,
  - hybride (combinaison horizontale et verticale).

Architecture des BDD

Fragmentation

- La fragmentation horizontale peut en outre être classée en deux techniques:
  - la fragmentation horizontale primaire,
  - la fragmentation horizontale dérivée
- La fragmentation doit être effectuée de manière à ce que la table originale puisse être reconstruite à partir des fragments. Cette exigence est appelée "capacité de reconstruction"

Architecture des BDD

Avantages de la fragmentation

- Les données étant stockées à proximité du site d'utilisation, l'efficacité du système de base de données est accrue
- Les techniques d'optimisation des requêtes locales sont suffisantes pour la plupart des requêtes car les données sont disponibles localement
- Les données non pertinentes n'étant pas disponibles sur les sites, la sécurité et la confidentialité du système de base de données peuvent être préservées

Architecture des BDD

Inconvénients de la fragmentation

- Lorsque des données provenant de différents fragments sont requises, les vitesses d'accès peuvent être très élevées
- En cas de fragmentation récursive, le travail de reconstruction nécessitera des techniques coûteuses
- L'absence de copies de sauvegarde des données sur différents sites peut rendre la base de données inefficace en cas de défaillance d'un site

Architecture des BDD

Fragmentation verticale

- Dans la fragmentation verticale, les champs ou les colonnes d'une table sont regroupés en fragments
- Afin de maintenir la capacité de reconstruction, chaque fragment doit contenir le ou les champs de clé primaire de la table
- Par exemple, considérons qu'une base de données d'une université conserve les enregistrements de tous les étudiants inscrits dans une table d'étudiants ayant le schéma suivant:

Regd_No	Name	Course	Address	Semester	Fees	Marks

Architecture des BDD

Fragmentation verticale

Dans ce cas, le concepteur pourrait fragmenter la base de données comme suit avec SQL:

```
CREATE TABLE STD_FEES AS
        SELECT Regd_No, Fees
        FROM STUDENT;
```

Regd_No	Name	Course	Address	Semester	Fees	Marks

Architecture des BDD

Fragmentation horizontale

- La fragmentation horizontale regroupe les n-uplets d'une table en fonction des valeurs d'un ou de plusieurs champs
- La fragmentation horizontale devrait également confirmer la règle de la capacité de reconstruction. Chaque fragment horizontal doit avoir toutes les colonnes de la table de base d'origine

Architecture des BDD

Fragmentation horizontale

- Par exemple, dans le schéma de l'étudiant, si les détails de tous les étudiants du cours d'informatique doivent être conservés à la School of Computer Science, le concepteur fragmentera la base de données horizontalement comme suit:

```
CREATE COMP_STD AS
        SELECT *
        FROM STUDENT
        WHERE COURSE = "Computer Science";
```

Architecture des BDD

Fragmentation hybride

- Dans la fragmentation hybride, une combinaison de techniques de fragmentation horizontale et verticale est utilisée
- C'est la technique de fragmentation la plus flexible car elle génère des fragments avec un minimum d'informations superflues
- Cependant, la reconstruction de la table originale est souvent une tâche coûteuse

Architecture des BDD

Fragmentation hybride

- La fragmentation hybride peut être réalisée de deux manières:
  - Au début, générer un ensemble de fragments horizontaux; puis générer des fragments verticaux à partir d'un ou plusieurs des fragments horizontaux
  - Dans un premier temps, générer un ensemble de fragments verticaux; puis générer des fragments horizontaux à partir d'un ou plusieurs des fragments verticaux

Transparence et Contrôle

Transparence dans les BDD

Les 3 dimensions …

- La transparence de la distribution permet d'entrevoir la base de données distribuée facile à utiliser comme toute base de données centralisée
- Les trois dimensions de la transparence de la distribution sont les suivantes:
  - Transparence de localisation
  - Transparence de la fragmentation
  - Transparence de réplication

Contrôle dans les BDD

Concept

- Le contrôle de base de données fait référence à l'application de contraintes d'intégrité afin de fournir des données correctes aux utilisateurs authentiques et aux applications d'une base de données
- En outre, les données doivent être protégées des utilisateurs non autorisés afin de préserver la sécurité et la confidentialité de la base de données
- Le contrôle de la base de données est l'une des tâches principales de l'administrateur de base de données (DBA)

Contrôle dans les BDD

Les 3 dimensions

- Les trois dimensions du contrôle de base de données sont –
  - Authentification
  - Droits d'accès
  - Contraintes d'intégrité

Contrôle dans les BDD

Authentification

- Dans un système de base de données distribuée, l'authentification est le processus par lequel seuls les utilisateurs légitimes peuvent accéder aux ressources de la BDD

Contrôle dans les BDD

Authentification

- L'authentification peut être appliquée à deux niveaux
  - Contrôle de l'accès à l'ordinateur client: [@exemple_ combinaison nom d'utilisateur/mot de passe, biométrique pour les données hautement sécurisées]
  - Contrôle de l'accès au logiciel de base de données - Le logiciel/admin attribue des informations d'identification à l'utilisateur [@exemple_ créer un compte de connexion au sein du serveur de base de données]

Contrôle dans les BDD

Droits d'accès

- Les droits d'accès d'un utilisateur font référence aux privilèges qui lui sont accordés en ce qui concerne les opérations du SGBD, tels que les droits de créer une table, de supprimer une table, d'ajouter/de supprimer/de mettre à jour des n-uplets dans une table ou d'interroger la table
- Dans les environnements distribués, en raison du grand nombre de tables et du nombre d'utilisateurs, il est impossible d'attribuer des droits d'accès individuels aux utilisateurs. D'où la nécessité de définir certains rôles

Contrôle dans les BDD

Droits d'accès: rôles

- Un rôle est une construction avec certains privilèges dans un système de base de données
- Une fois les différents rôles définis, chaque utilisateur se voit attribuer l'un de ces rôles
- Une hiérarchie de rôles est souvent définie en fonction de la hiérarchie d'autorité et de responsabilité de l'organisation

Contrôle dans les BDD

Droits d'accès: rôles

- Par exemple, les instructions SQL suivantes créent un rôle "Accountant", puis attribuent ce rôle à l'utilisateur "ABC"

```
CREATE ROLE ACCOUNTANT;
GRANT SELECT, INSERT, UPDATE ON EMP_SAL TO ACCOUNTANT;
GRANT INSERT, UPDATE, DELETE ON TENDER TO ACCOUNTANT;
GRANT INSERT, SELECT ON EXPENSE TO ACCOUNTANT;
COMMIT;
GRANT ACCOUNTANT TO ABC;
COMMIT;
```

Contrôle dans les BDD

Intégrité sémantique

- Le contrôle d'intégrité sémantique définit et applique les contraintes d'intégrité du système de base de données
- Les contraintes d'intégrité sont les suivantes :
  - Contrainte d'intégrité de type de données
  - Contrainte d'intégrité d'entité
  - Contrainte d'intégrité référentielle

Contrôle dans les BDD

Intégrité de types de données

- Une contrainte de type de données limite la plage de valeurs et le type d'opérations pouvant être appliquées au champ avec le type de données spécifié
- Par exemple, considérons qu'une table "HOSTEL" comporte trois champs: le numéro de l'auberge, le nom de l'auberge et la capacité. Le numéro d'auberge doit commencer par la lettre majuscule "H" et ne peut pas être NULL. La capacité ne doit pas dépasser 150 (Exemple: voir diapo suivant)

Contrôle dans les BDD

Intégrité de types de données

- La commande SQL suivante peut être utilisée pour la définition des données -

```
CREATE TABLE HOSTEL (
    H_NO VARCHAR2(5) NOT NULL,
    H_NAME VARCHAR2(15),
    CAPACITY INTEGER,
    CHECK ( H_NO LIKE 'H%'),
    CHECK ( CAPACITY <= 150)
);
```

Contrôle dans les BDD

Intégrité d'entités

- Le contrôle d'intégrité d'entité applique les règles afin que chaque tuple puisse être identifié de manière unique à partir d'autres tuples. Pour cela, une clé primaire est définie
- La contrainte d'intégrité d'entité stipule qu'aucun des n-uplets d'une table ne peuvent avoir des valeurs identiques pour les clés primaires et qu'aucun champ faisant partie de la clé primaire ne peut avoir la valeur NULL

Contrôle dans les BDD

Intégrité d'entités

- Par exemple, dans la table d'hôte ci-dessus, le numéro d'hôte peut être attribué comme clé primaire via l'instruction SQL suivante (en ignorant les vérifications)

```
CREATE TABLE HOSTEL (
    H_NO VARCHAR2(5) PRIMARY KEY,
    H_NAME VARCHAR2(15),
    CAPACITY INTEGER
);
```

Contrôle dans les BDD

Intégrité référentielle

- La contrainte d'intégrité référentielle définit les règles des clés étrangères
- Une clé étrangère est un champ dans une table de données qui est la clé primaire d'une table liée
- La contrainte d'intégrité référentielle définit la règle selon laquelle la valeur du champ de clé étrangère doit figurer parmi les valeurs de la clé primaire de la table référencée ou être entièrement NULL

Contrôle dans les BDD

Intégrité référentielle

- Par exemple, considérons une table d'étudiants sur laquelle un étudiant peut choisir de vivre dans un **Hostel**. Pour l'inclure, la clé primaire de la table d'hôte doit être incluse en tant que clé étrangère dans la table des étudiants. L'instruction SQL suivante incorpore ceci-

```
CREATE TABLE STUDENT (
    S_ROLL INTEGER PRIMARY KEY,
    S_NAME VARCHAR2(25) NOT NULL,
    S_COURSE VARCHAR2(10),
    S_HOSTEL VARCHAR2(5) REFERENCES HOSTEL
);
```

Gestion de requêtes dans une BD centralisée

Requêtes

Concept

- Lorsqu'une requête est écrite, elle est d'abord analysée syntaxiquement, sémantiquement et validée
- Une représentation interne de la requête est ensuite créée, telle qu'un arbre ou un graphe de requête
- Des stratégies d'exécution alternatives sont ensuite conçues pour extraire les résultats des tables de la base de données
- Le processus de choix de la stratégie d'exécution la plus appropriée pour le traitement des requêtes est appelé **optimisation des requêtes**

Requêtes

Optimisation

- Dans un SGBDD, l'optimisation des requêtes est cruciale
- La complexité est élevée car le nombre de stratégies alternatives peut augmenter de manière exponentielle en raison des facteurs suivants:
  - La présence d'un certain nombre de fragments,
  - La répartition des fragments ou des tableaux sur différents sites
  - La vitesse des liens de communication
  - La disparité dans les capacités de traitement local

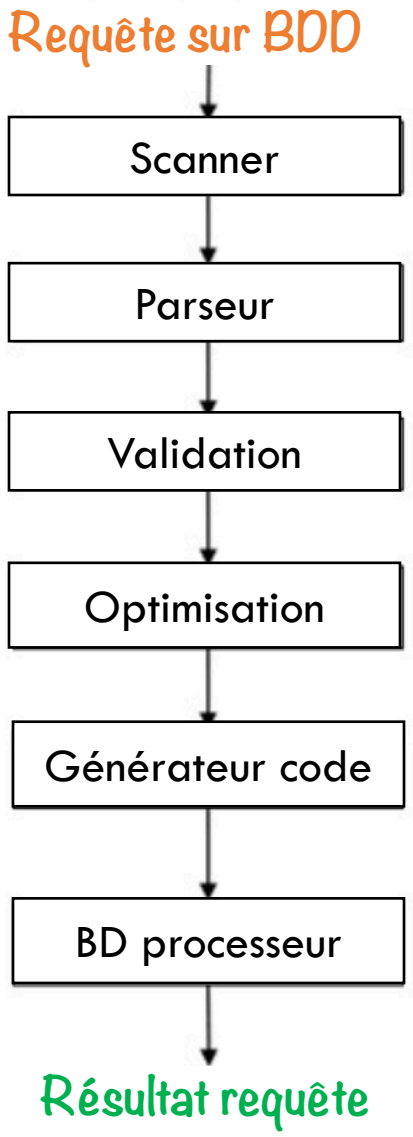
Requêtes
Optimisation

- Par conséquent, dans un système distribué, l'objectif est souvent de trouver une **bonne** stratégie d'exécution pour le traitement des requêtes plutôt que la **meilleure**
- Le temps d'exécution d'une requête est la somme des éléments suivants:
  - temps pour communiquer des requêtes aux bases de données
  - temps pour exécuter les fragments de requête locaux
  - temps pour rassembler les données provenant de différents sites
  - temps pour afficher les résultats dans l'application

Requêtes

Traitement

- Le traitement des requêtes est l'ensemble de toutes les activités allant du placement de la requête à l'affichage des résultats de la requête
- Les étapes sont comme indiquées dans le diagramme suivant (à droite)



Requêtes

Optimisation dans une BD centralisée

- Le chemin d'accès optimal est déterminé à partir des chemins d'accès produits par calcul algébrique relationnel
- Dans un système centralisé, le traitement des requêtes est effectué dans le but suivant:
  - minimiser le temps de réponse de la requête
  - maximiser le débit du système
  - réduire la quantité de mémoire et de stockage requise pour le traitement
  - augmenter le parallélisme

Requêtes

Optimisation dans une BD centralisée

- Initialement, la requête SQL est analysée pour rechercher les erreurs syntaxiques et l'exactitude des types de données. Si la requête réussit cette étape, la requête est décomposée en blocs de requête plus petits
- Chaque bloc est ensuite traduit en une expression d'algèbre relationnelle équivalente

Étapes pour l'optimisation de la requête

- L'optimisation des requêtes implique trois étapes, à savoir :
  - la génération d'arborescence de requêtes
  - la génération de plans
  - la génération de codes de plans de requêtes

Optimisation dans une BD centralisée

- Une arborescence de requête est une structure de données arborescente représentant une expression d'algèbre relationnelle
- Les tables de la requête sont représentées sous forme de nœuds feuille
- Les opérations d'algèbre relationnelle sont représentées en tant que nœuds internes
- La racine représente la requête dans son ensemble

Optimisation dans une BD centralisée

- Pendant l'exécution, un nœud interne est exécuté chaque fois que ses tables d'opérandes sont disponibles
- Le nœud est ensuite remplacé par la table de résultats
- Ce processus se poursuit pour tous les nœuds internes jusqu'à ce que le nœud racine soit exécuté et remplacé par la table de résultats

Optimisation dans une BD centralisée

- Par exemple, considérons les schémas suivants et la requête ci-dessous -

EMPLOYEE

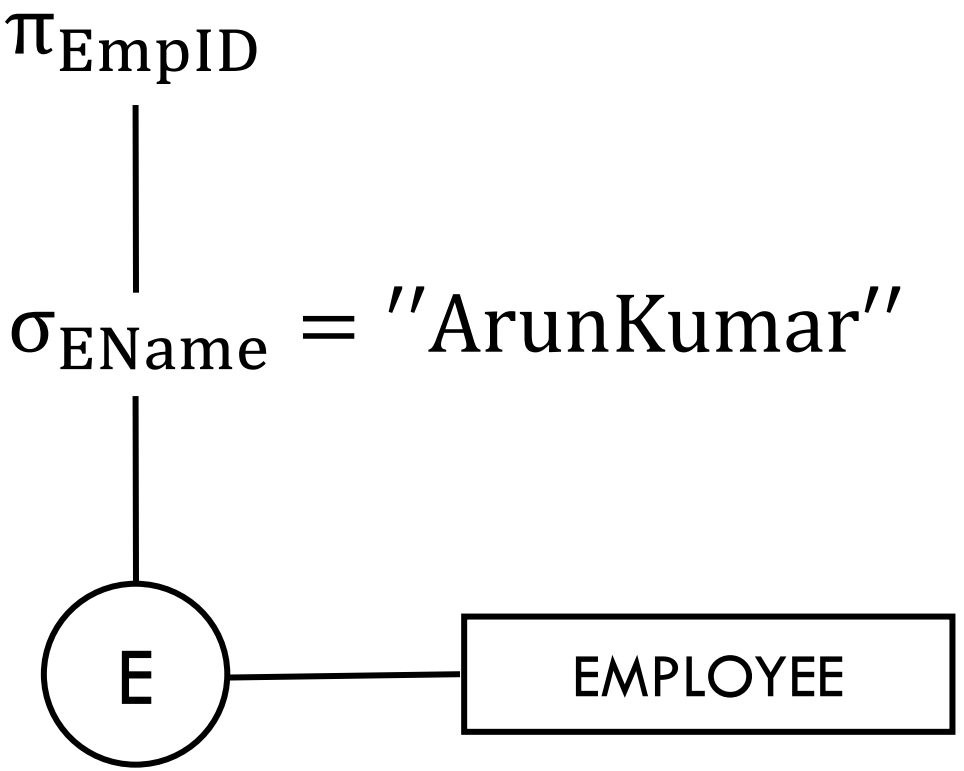
EmpID	EName	Salary	DeptNo	DateOfJoining

DEPARTMENT

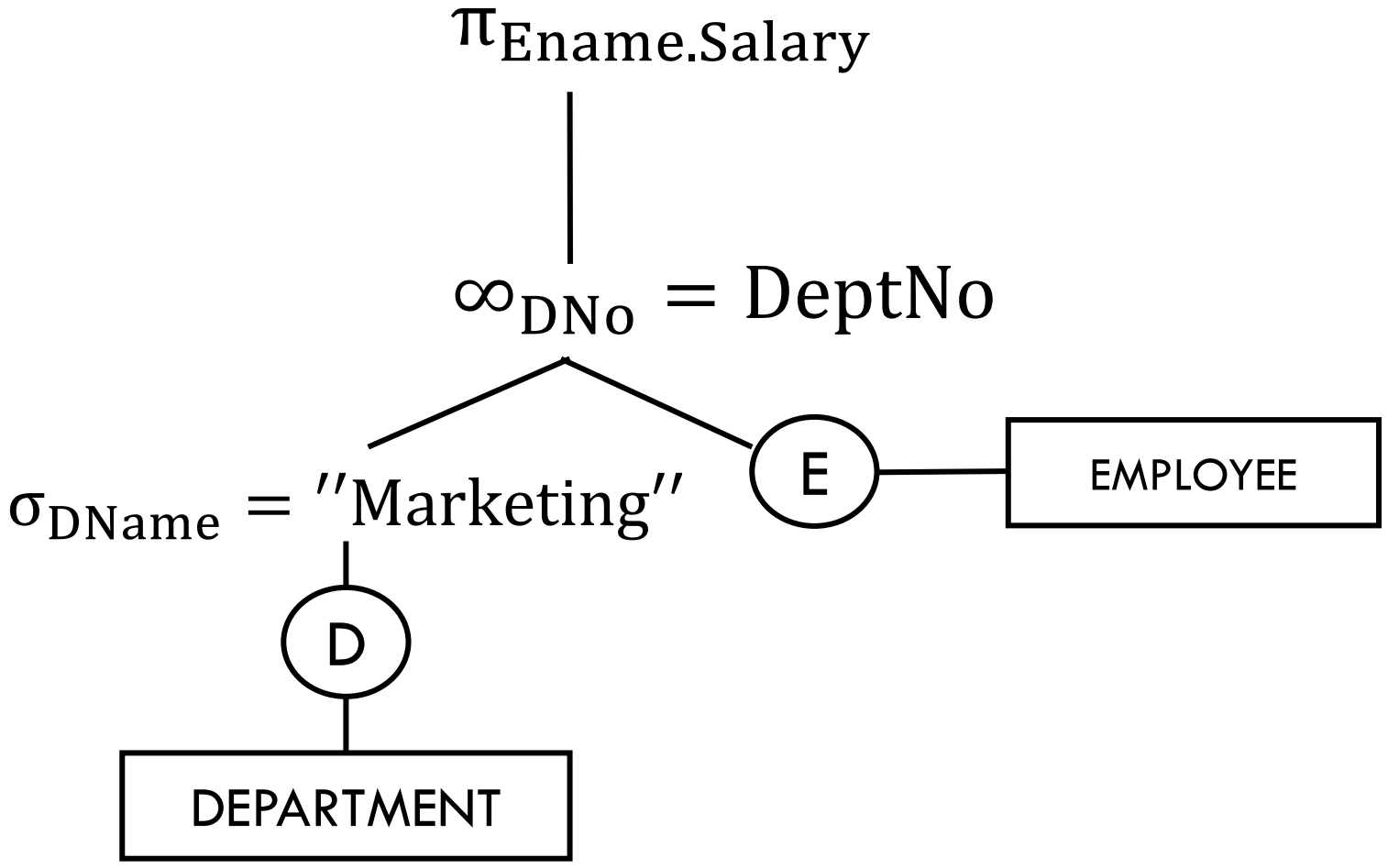
DNo	DName	Location

$$\pi_{EmpID}(\sigma_{EName="ArunKumar"}(EMPLOYEE))$$

$$\pi_{EmpID}(\sigma_{EName="ArunKumar"}(EMPLOYEE))$$



$$\pi_{EName,Salary}(\sigma_{DName="Marketing"}(DEPARTMENT)) \bowtie_{DNo=DeptNo} (EMPLOYEE)$$



Optimisation dans une BD centralisée

- Une fois l'arborescence de requêtes générée, un plan de requête est créé
- Un plan de requête est une arborescence de requête étendue qui inclut des chemins d'accès pour toutes les opérations de l'arborescence de requête
- Les chemins d'accès spécifient comment les opérations relationnelles dans l'arborescence doivent être effectuées
- En outre, un plan de requête indique également comment les tables intermédiaires doivent être transmises d'un opérateur à l'autre, comment les tables temporaires doivent être utilisées et les opérations doivent être combinées

Optimisation dans une BD centralisée

- La génération de code est la dernière étape de l'optimisation de requêtes
- C'est la forme exécutable de la requête, dont la forme dépend du type de système d'exploitation sous-jacent
- Une fois le code de requête généré, le gestionnaire d'exécution l'exécute et génère les résultats

Requêtes

Optimisation dans une BD centralisée

- Différentes familles d'approches:
  - Recherche exhaustive
  - Recherche heuristique (approximative)

Requêtes

Optimisation dans une BD centralisée

Recherche exhaustive

- Pour une requête, tous les plans de requête possibles sont initialement générés, puis le meilleur plan est sélectionné
- Bien que cette technique constitue la meilleure solution, la complexité est exponentielle dans le temps et dans l'espace en raison de la grandeur de l'espace de solution

Optimisation dans une BD centralisée

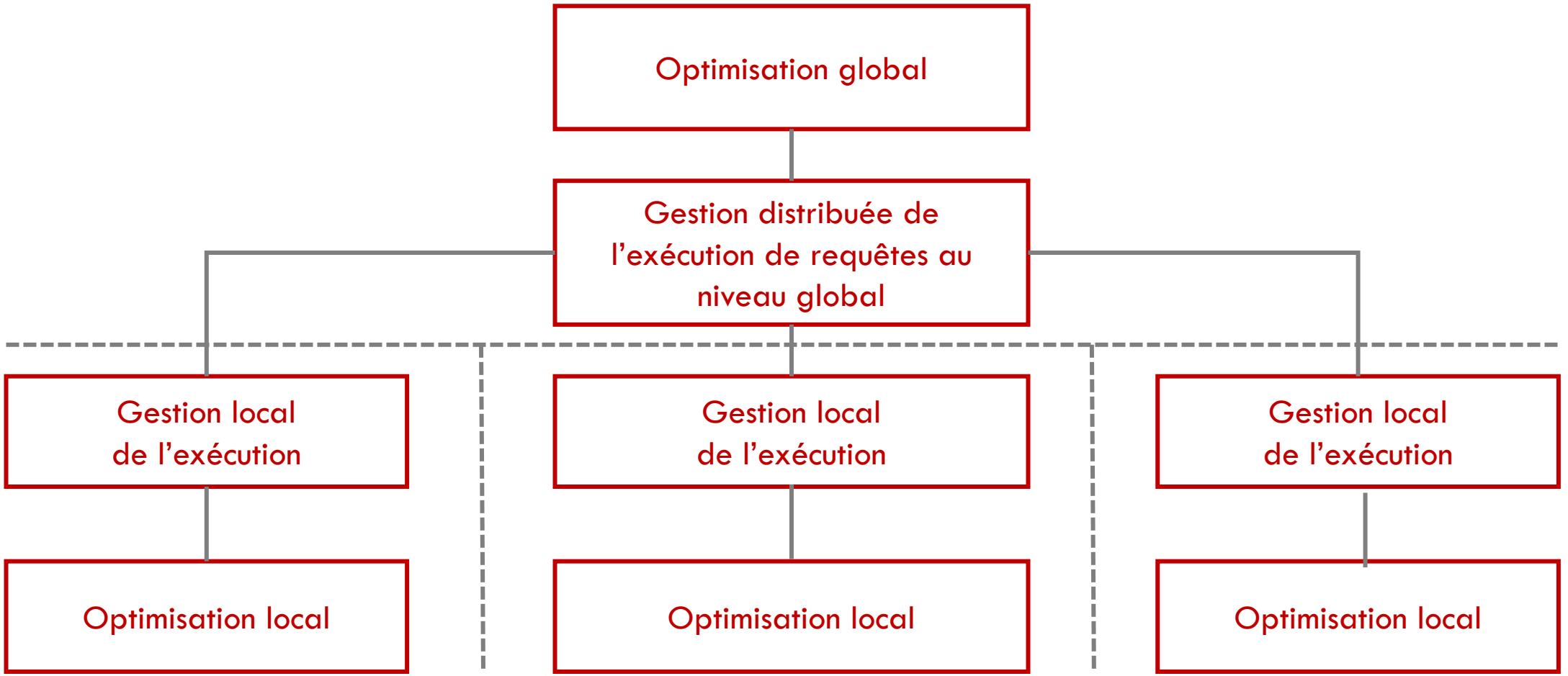
- Les heuristiques ont une complexité temporelle et spatiale polynomiale, inférieure à la complexité exponentielle des algorithmes exhaustifs basés sur la recherche dans un espace de solutions
- Certaines des règles heuristiques communes sont -
  - Effectuez les opérations de sélection et de projection avant les opérations de jointure. Pour ce faire, déplacez les opérations SELECT et PROJECT dans l'arborescence de la requête. Cela réduit le nombre de tuples disponibles pour la jointure
  - Effectuez les opérations de sélection/projection les plus restrictives en premier avant les autres opérations
  - Évitez les opérations entre produits car elles génèrent des tables intermédiaires de très grande taille

Gestion de requêtes dans une BD ditribuée

Traitement de requêtes
dans une BD distribuée

- Dans un SGBDD, le traitement d'une requête comprend une optimisation à la fois au niveau global et au niveau local
- La requête entre dans le SGBDD sur le client ou sur le site de contrôle
- L'utilisateur est validé, la requête est vérifiée, traduite et optimisée au niveau global

Traitement de requêtes
dans une BD distribuée



Traitement de requêtes
dans une BD distribuée

- Mapping de requêtes globales aux requêtes locales
- Le processus de mapping des requêtes globales aux requêtes locales peut être réalisé comme suit:
  - Les tables requises dans une requête globale ont des fragments répartis sur plusieurs sites
  - Les bases de données locales contiennent uniquement des informations sur les données locales
  - Le site de contrôle utilise le dictionnaire de données global pour rassembler des informations sur la distribution et reconstruire la vue globale à partir des fragments

Traitement de requêtes
dans une BD distribuée

- En l'absence de réplication, l'optimiseur global exécute des requêtes locales sur les sites où les fragments sont stockés
- S'il y a réplication, l'optimiseur global sélectionne le site en fonction des coûts de communication, de la charge de travail et de la vitesse du serveur
- L'optimiseur global génère un plan d'exécution réparti de manière à minimiser le transfert de données sur les sites
- Le plan indique l'emplacement des fragments, l'ordre dans lequel les étapes de la requête doivent être exécutées et les processus impliqués dans le transfert des résultats intermédiaires

Traitement de requêtes
dans une BD distribuée

- Par exemple, considérons que le schéma de projet suivant est fragmenté horizontalement selon les villes, New Delhi, Kolkata et Hyderabad.

PROJECT

PId	City	Department	Status

Traitement de requêtes
dans une BD distribuée

Supposons qu'il existe une requête pour extraire les détails de tous les projets dont le statut est «En cours».

Requête globale	$\sigma_{status = \text{" ongoing "}}(PROJECT)$
Requête sur new Dheli	$\sigma_{status = \text{" ongoing "}}(NewD\_PROJECT)$
Requête sur new Kolkata	$\sigma_{status = \text{" ongoing "}}(Kol\_PROJECT)$
Requête sur new Hyderabad	$\sigma_{status = \text{" ongoing "}}(Hyd\_PROJECT)$

Traitement de requêtes
dans une BD distribuée

- Pour obtenir le résultat global, nous devons regrouper les résultats des trois requêtes de la manière suivante:

$$\sigma_{status = \text{" ongoing "}}(NewD\_PROJECT) \cup \sigma_{status = \text{" ongoing "}}(kol\_PROJECT)$$

$$\cup \sigma_{status = \text{" ongoing "}}(Hyd\_PROJECT)$$

Optimisation de requêtes
dans une BD distribuée

- L'optimisation de requête distribuée nécessite l'évaluation d'un grand nombre d'arbres de requête produisant chacun les résultats requis d'une requête
- Ceci est principalement dû à la présence d'une grande quantité de données répliquées et fragmentées
- Par conséquent, l'objectif est de trouver une solution optimale au lieu de la meilleure

Optimisation de requêtes
dans une BD distribuée

- Les principaux problèmes liés à l'optimisation des requêtes distribuées sont les suivants:
  - utilisation optimale des ressources dans le SD,
  - requête trading,
  - réduction de l'espace de solution de la requête.

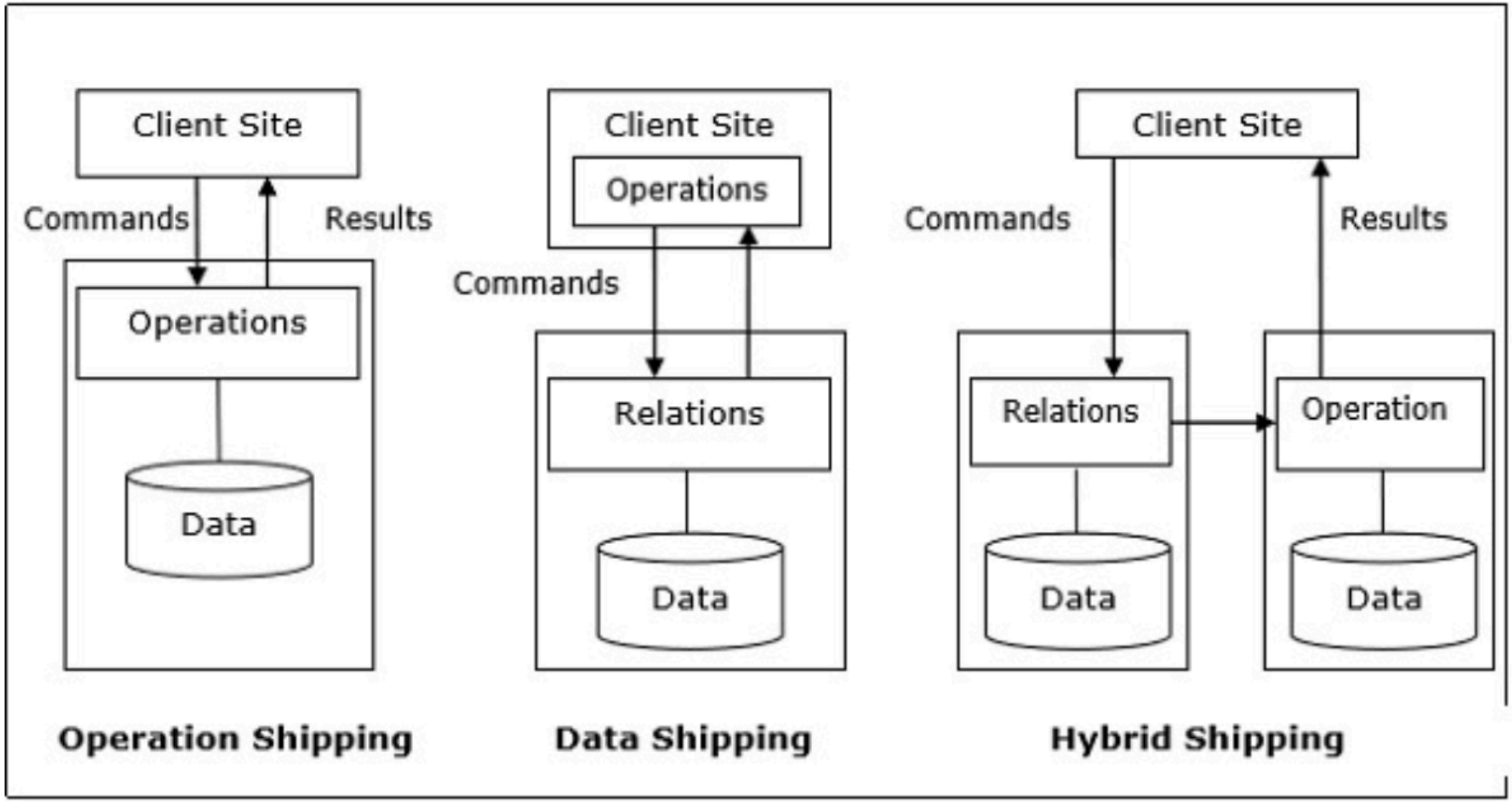
Optimisation de requêtes dans une BD distribuée

Utilisation optimale des ressources dans le SD

- Un système distribué dispose d'un certain nombre de serveurs de base de données dans les divers sites pour effectuer les opérations relatives à une requête
- Les approches d'utilisation optimale de ressources sont:
  - Operation shipping
  - Data shipping
  - Hybrid shipping

Optimisation de requêtes
dans une BD distribuée

Utilisation optimale des ressources dans le SD



Operation shipping

L'opération est exécutée sur le site où les données sont stockées et non sur le site client. Les résultats sont ensuite transférés sur le site client. Ceci est approprié pour les opérations où les opérandes sont disponibles sur le même site

Exemple: Opérations Select et Project

Data shipping

Lors de l'expédition des données, les fragments de données sont transférés vers le serveur de base de données, où les opérations sont exécutées. Ceci est utilisé dans les opérations où les opérandes sont distribués sur différents sites. Ceci est également approprié dans les systèmes où les coûts en communication sont bas et où les processeurs locaux sont beaucoup plus lents que le serveur client

Exemple: Opérations Select et Project

Hybrid shipping

Il s'agit d'une combinaison de data/opération shipping. Ici, les fragments de données sont transférés aux processeurs à grande vitesse, où l'opération est exécutée. Les résultats sont ensuite envoyés au site client

Optimisation de requêtes dans une BD distribuée

Requête trading

Dans l'algorithme de requête trading pour les systèmes de bases de données distribuées, le site de contrôle/client pour une requête distribuée est appelé le **demandeur** et les sites d'exécution des requêtes locales sont appelés **traders**. Le **demandeur** formule un certain nombre d'alternatives pour choisir les **traders** et reconstruire les résultats globaux.

L'objectif du **demandeur** est d'atteindre le coût optimal.

L'algorithme commence avec le **demandeur** qui attribue des sous-requêtes aux sites du **trader**. Le plan optimal est créé à partir des plans de requête optimisés locaux proposés par les **traders**, combinés au coût de la communication pour reconstruire le résultat final. Une fois que le plan optimal global est formulé, la requête est exécutée.

Optimisation de requêtes
dans une BD distribuée

Réduction de l'espace de solution de la requête

Une solution optimale implique généralement une réduction de l'espace de la solution, de sorte que le coût des requêtes et du transfert de données est réduit. Cela peut être réalisé grâce à un ensemble de règles heuristiques, tout comme les heuristiques dans les systèmes centralisés.

Optimisation de requêtes
dans une BD distribuée

Réduction de l'espace de solution de la requête

- Voici quelques règles
  - Effectuer les opérations de sélection et de projection le plus tôt possible pour réduire le flux de données sur le réseau de communication
  - Simplifier les opérations sur les fragments horizontaux en éliminant les conditions de sélection non pertinentes pour un site particulier
  - Dans le cas d'opérations de jointure et d'union comprenant des fragments situés sur plusieurs sites, transférer les données fragmentées vers le site où la plupart des données sont présentes et effectuer l'opération à cet endroit

Optimisation de requêtes
dans une BD distribuée

Réduction de l'espace de solution de la requête

- Utilisez l'opération de semi-jointure pour qualifier les n-uplets à joindre. Cela réduit la quantité de transfert de données, ce qui réduit les coûts de communication

- Fusionnez les feuilles et les sous-arbres communs dans une arborescence de requêtes distribuée

Transaction

Transaction

Concept

- Une transaction est un programme comprenant une collection d'opérations sur une base de données, exécutée comme une unité logique de traitement de données
- Les opérations effectuées dans une transaction incluent une ou plusieurs opérations de base de données telles que l'insertion, la suppression, la mise à jour ou la récupération de données.
- Il s'agit d'un processus **atomique** dont l'achèvement est soit complet ou n'est pas exécuté du tout. Une transaction impliquant uniquement la récupération de données sans aucune mise à jour de données est appelée transaction en lecture seule

Transaction

Concept

- Chaque opération de haut niveau peut être divisée en plusieurs tâches ou opérations de bas niveau
- Par exemple, une opération de mise à jour des données peut être divisée en trois tâches:
  - read_item () - lire une donnée de la mémoire dans une mémoire principale
  - modify_item () - changer la valeur de l'item dans la mémoire principale
  - write_item () - écrire la valeur modifiée de la mémoire principale dans la mémoire

Transaction

Concept

Les opérations de bas niveau effectuées dans une transaction sont les suivantes:

- begin_transaction - Marqueur qui spécifie le début de l'exécution de la transaction
- read_item ou write_item - Opérations de base
- end_transaction - Marqueur qui spécifie la fin de la transaction
- commit - Signal pour spécifier que la transaction a été complétée avec succès
- rollback - Signal indiquant que la transaction a échoué et que toutes les modifications temporaires dans la base de données sont annulées

NB: Une transaction validée ne peut pas être annulée

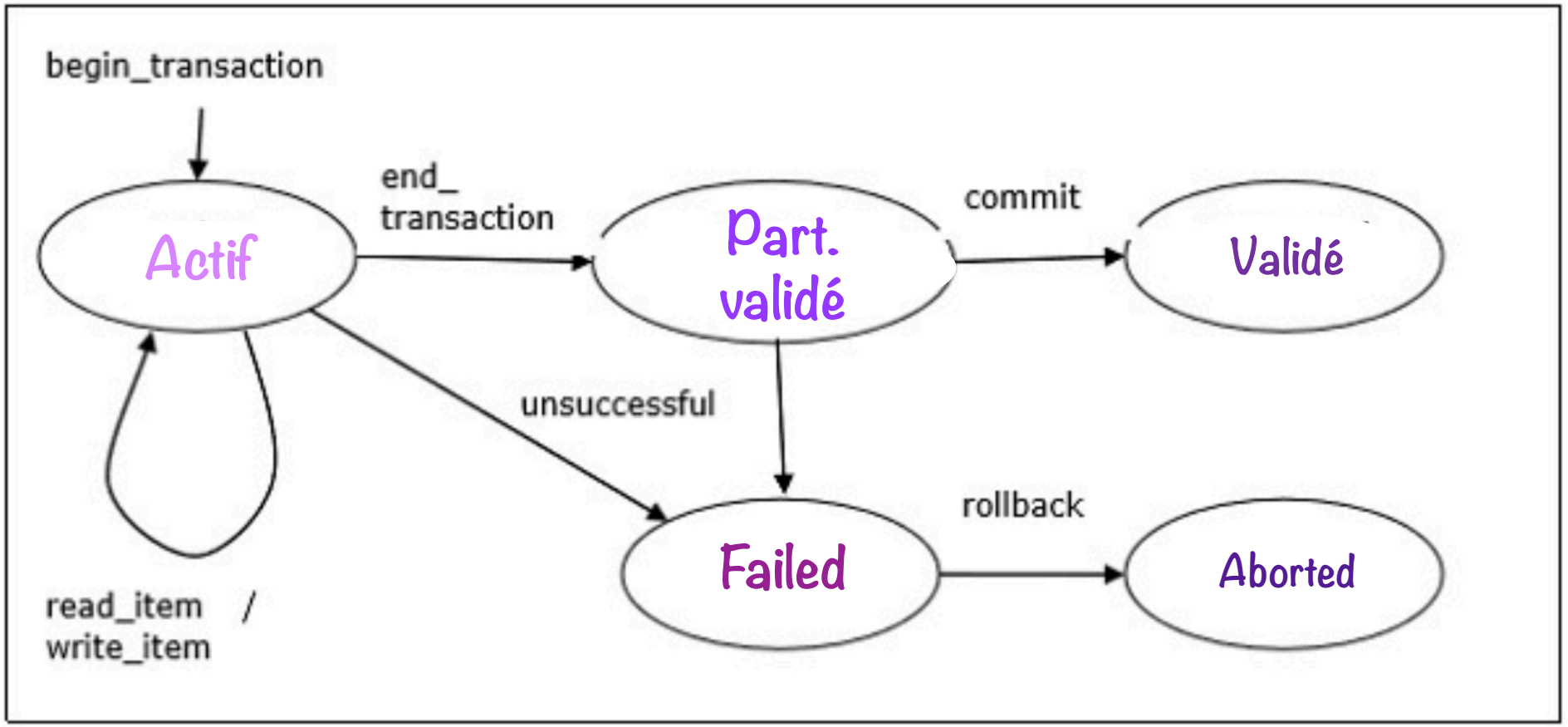
Transaction

États

- Une transaction peut passer par cinq états:
  - Actif - L'état initial et d'exécution des opérations de lecture, d'écriture
  - Partiellement validée- État après l'exécution du dernier relevé de transaction
  - Validé - Une fois la transaction terminée et le signal de validation émis
  - Failed (Échec) – cas d'échec d'échec ou d'exécution anormale
  - Aborted – État d'annulation de la transaction après échec et rétablissement de la base de données à son état antérieur

Transaction

États



Transaction

Propriétés

- Toute transaction doit conserver les propriétés ACID
  - Atomicité – Une transaction est une unité de traitement atomique, c'est-à-dire qu'elle est exécutée dans sa totalité ou pas du tout. Aucune mise à jour partielle
  - Cohérence - Une transaction doit faire passer la base de données d'un état cohérent à un autre état cohérent, i.e. ne devrait pas affecter les données de la base de données
  - Isolation - Une transaction doit être exécutée comme si elle était la seule du système. Il ne devrait y avoir aucune interférence des autres transactions simultanées en cours d'exécution
  - Durabilité - Si une transaction validée entraîne une modification, celle-ci doit être durable dans la base de données et non perdue en cas de défaillance

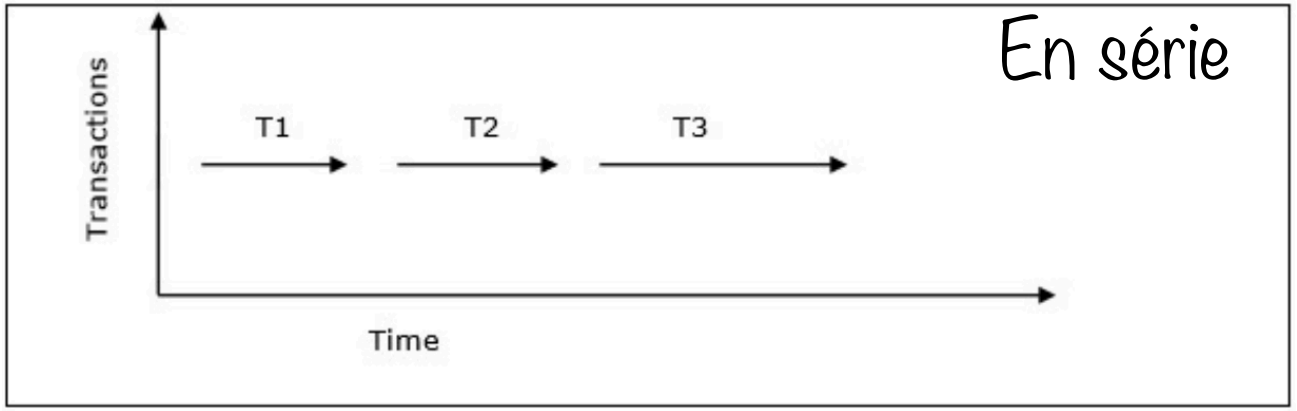
Transaction

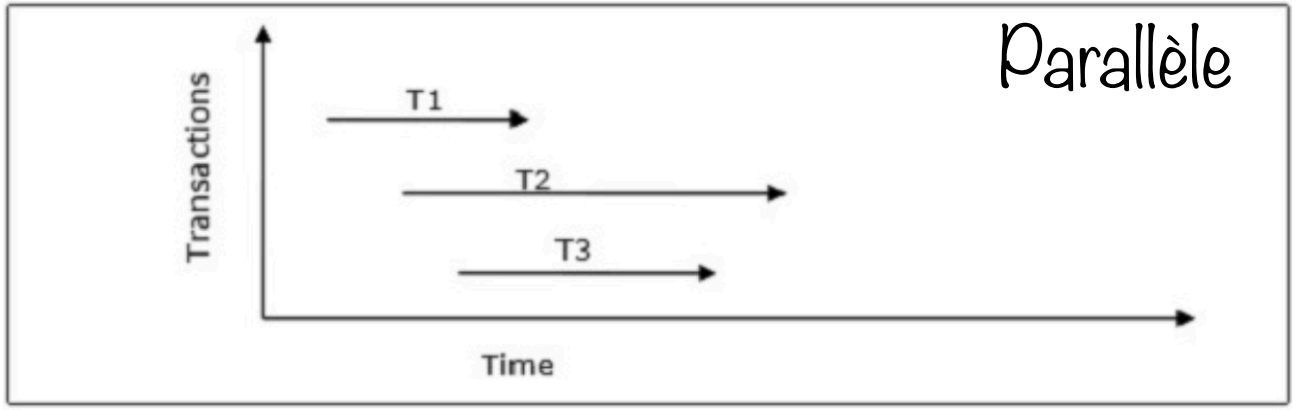
Ordonnancement et concurrence

- Dans un système comportant plusieurs transactions simultanées, une planification correspond à l'ordre total d'exécution des opérations
- Soit une annexe S comprenant n transactions, soit T_1, T_2, T_3 , …, T_n; pour toute transaction T_i, les opérations dans T_i doivent être exécutées conformément à l'annexe S

Transaction

Ordonnancement et concurrence





Transaction

Ordonnancement et concurrence

- Dans une planification comprenant plusieurs transactions, un conflit se produit lorsque deux transactions actives effectuent des opérations non compatibles.
- Deux opérations sont considérées en conflit lorsque les trois conditions suivantes sont simultanément réunies:
  - Les deux opérations font partie de transactions différentes
  - Les deux opérations accèdent au même emplacement de données
  - Au moins une des opérations est une opération write_item (), c'est-à-dire qu'elle tente de modifier l'élément de données

Transaction

Ordonnancement et concurrence

Un programme sérialisable de "n" transactions est un programme parallèle qui équivaut à un programme sériel comprenant les mêmes "n" transactions. Un programme sérialisable contient l'exactitude du programme en série tout en assurant une meilleure utilisation du processeur par le programme en parallèle

Fin