Bases de données réparties

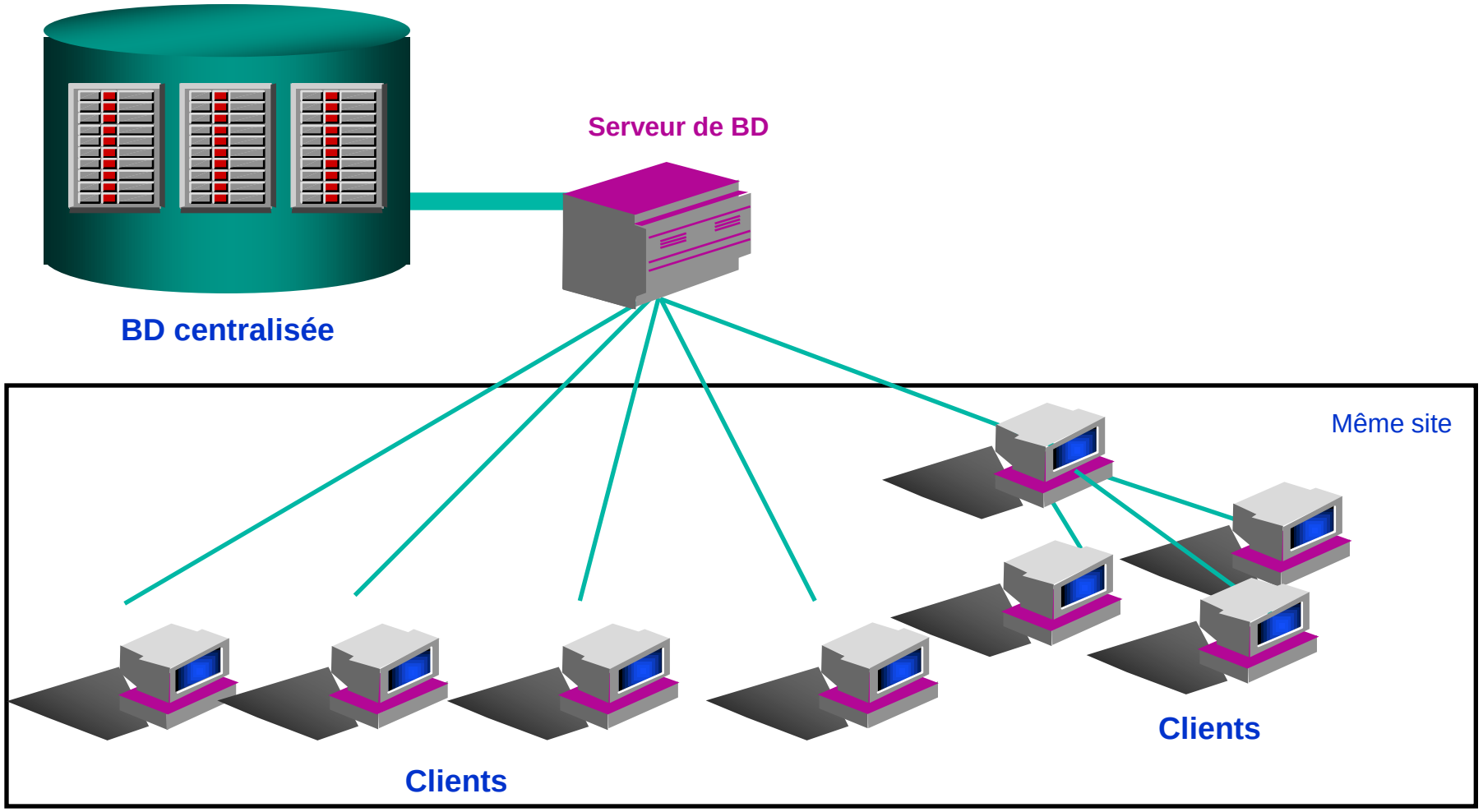
M. Nassar

Ecole Nationale Supérieure d'Informatique
et d'Analyse des Systèmes
- Rabat -

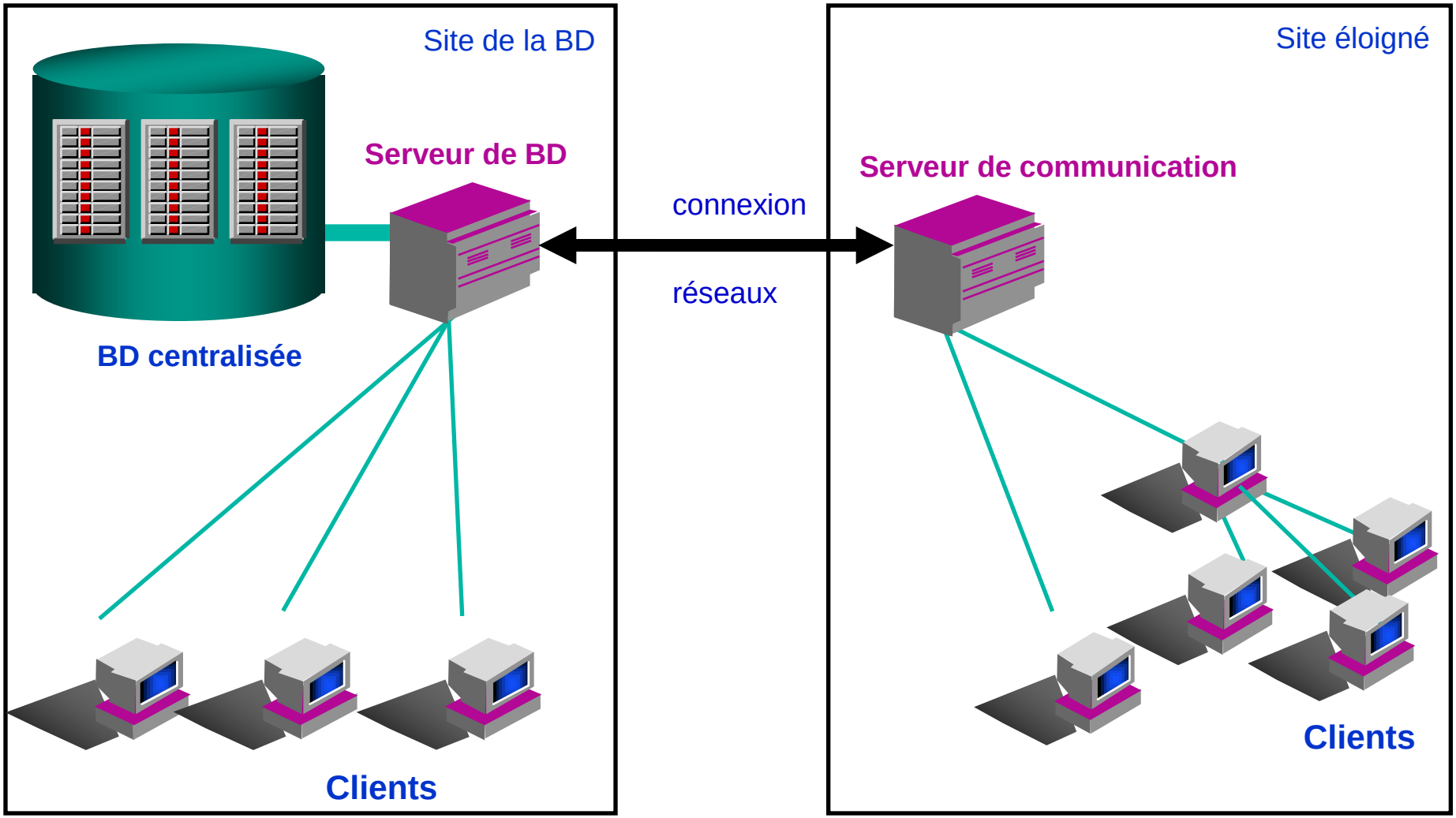
Plan

- Introduction
- Définitions et vocabulaire
- Objectifs des SGBDR
- Architecture des SGBDR
- Conception des bases de données réparties
- Gestion des transactions et contrôle de concurrence
- Duplication et Réplication des données réparties
- Cas d'Oracle
- Conclusion

Introduction
BD centralisée – sur 1 site : « locale »



Introduction
BD centralisée – sur n ≥ 2 sites : « distante »



Introduction
BD centralisée – avantages et inconvénients

Avantages :

- **Un seul SGBD**
- **Un seul administrateur de la BD**
- **La requête est calculée sur le serveur par le SGBD et les résultats sont acheminés vers le client distant**

Inconvénients :

- **Le réseau peut tomber en panne !**
- **Sur utilisation du réseau**

Introduction
Problème 1

Enoncé :

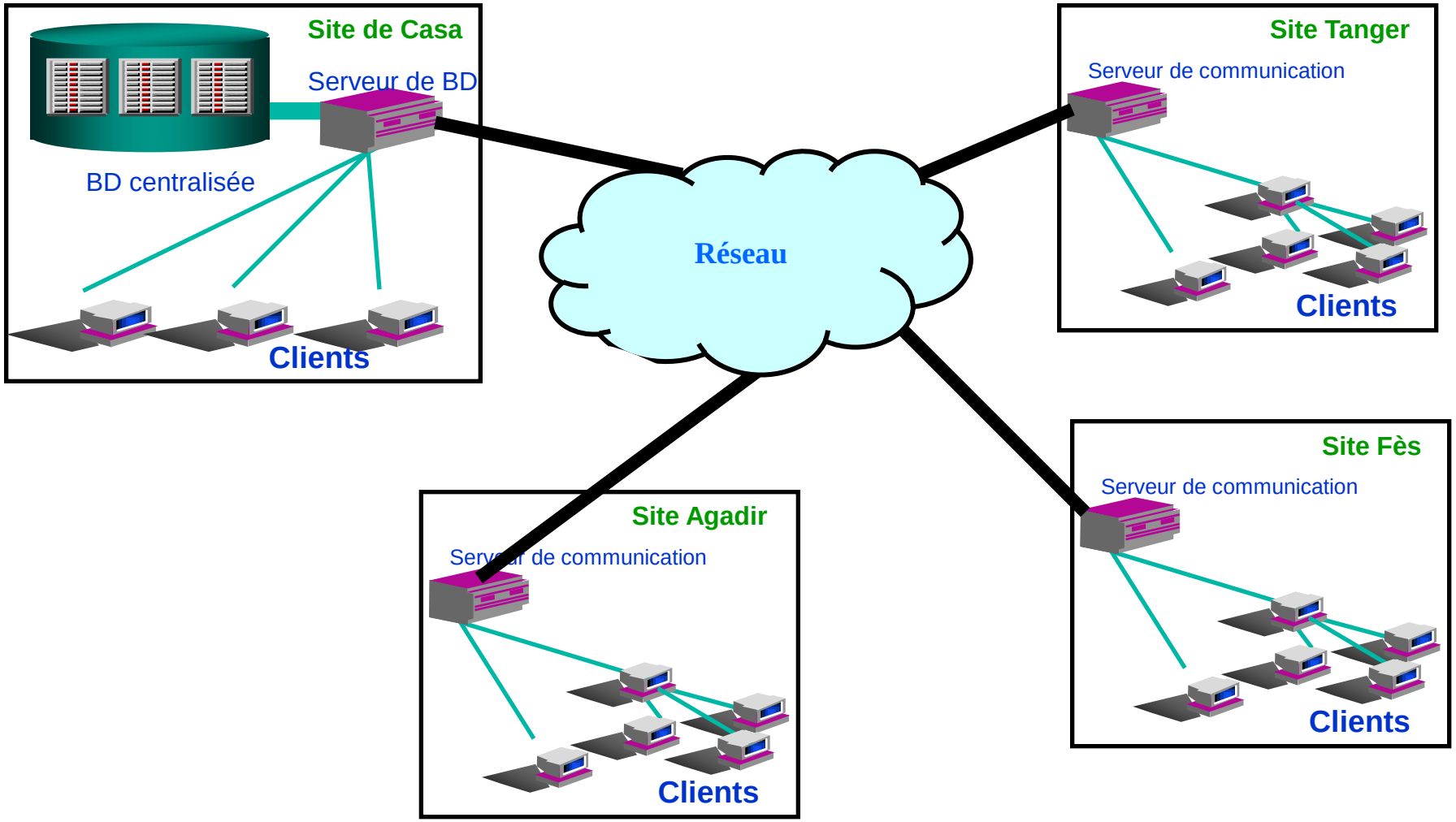
- La société X d'assurance santé est implantée sur l'ensemble du Maroc,
- Le siège est localisé à Casablanca,
- Dans un souci de service, la société crée de nouvelles filiales au Maroc,
- Chaque filiale gère les dossiers des adhérents de sa région,
- Chaque filiale peut accéder aux informations du siège ou d'une autre filiale.

Hypothèse :

- Aucun héritage informatique

Quelle architecture adopter ?

Introduction
Solution 1 : BD centralisée



Introduction
Solution 1, Critiques

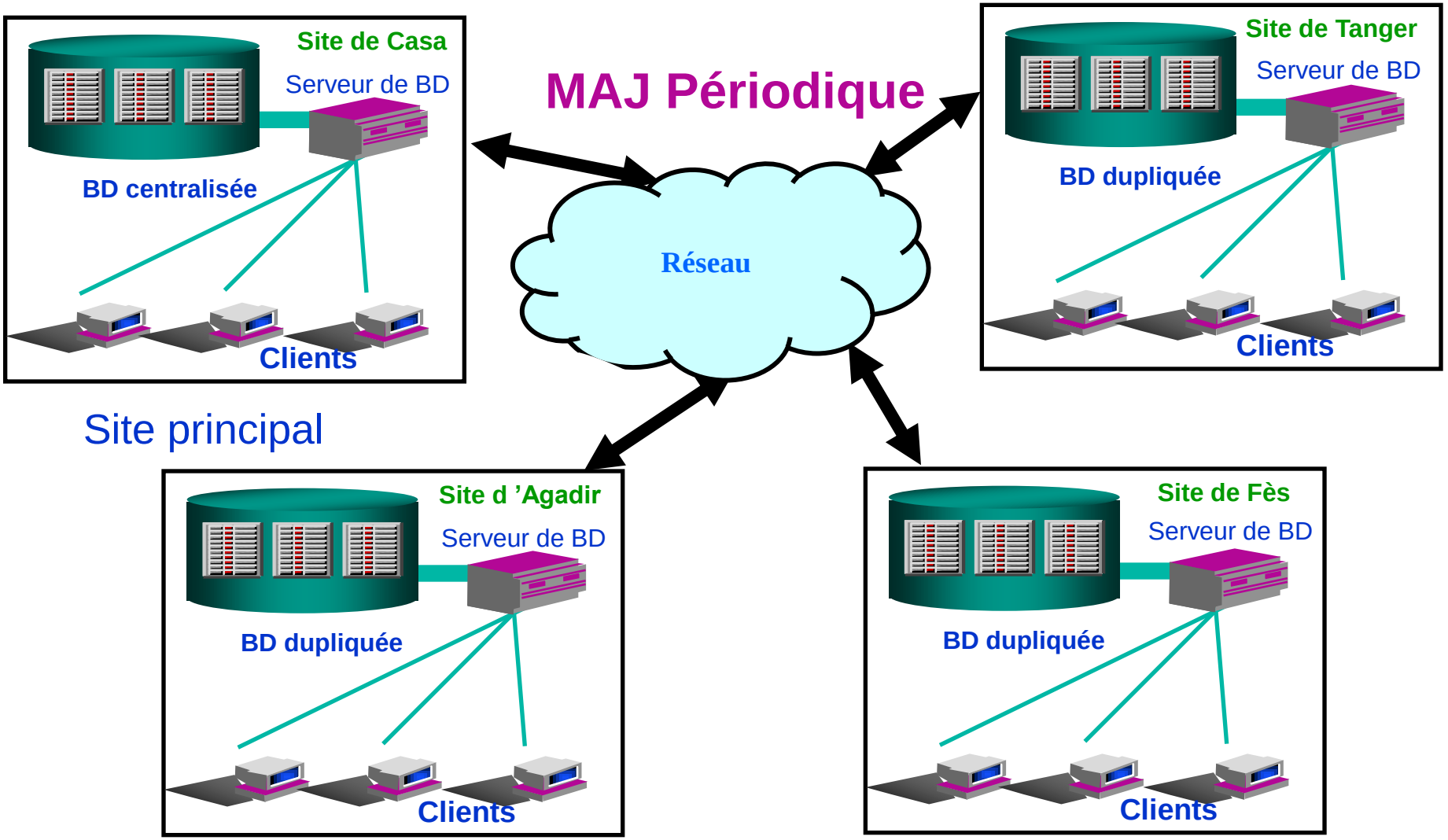
Avantages :

- **Un seul SGBD**
- **Un seul administrateur de la BD**
- **La requête est calculée sur le serveur par le SGBD et les résultats sont acheminés vers le client distant**

Inconvénients :

- **Si le site central tombe en panne ?!**
- **Si le réseau tombe en panne ?!**
- **Le coût de la communication et du transfert des données**
- **Charge de calcul concentrée sur le serveur central**

Introduction
Solution 2, BD dupliquée



Introduction
Solution 2, Critiques

Avantages :

- **Toute la base de données est localisée dans chaque site**
  - **La requête est calculée sur le serveur local par le SGBD local**
  - **Réduction du coût de la communication**

Inconvénients :

- **Plusieurs SGBDs**
  - **Problèmes de MAJ et d'incohérence de la base**
  - **Plusieurs administrateurs de la base**
- **Surcharge de la base par des données non nécessairement utiles en local**

Introduction
Question 1

Comment construire une architecture BD EFFICACE sur un compromis entre :

1. l'éloignement des sites ?
versus
2. la disponibilité des données ?

Introduction
Problème 2

Enoncé :

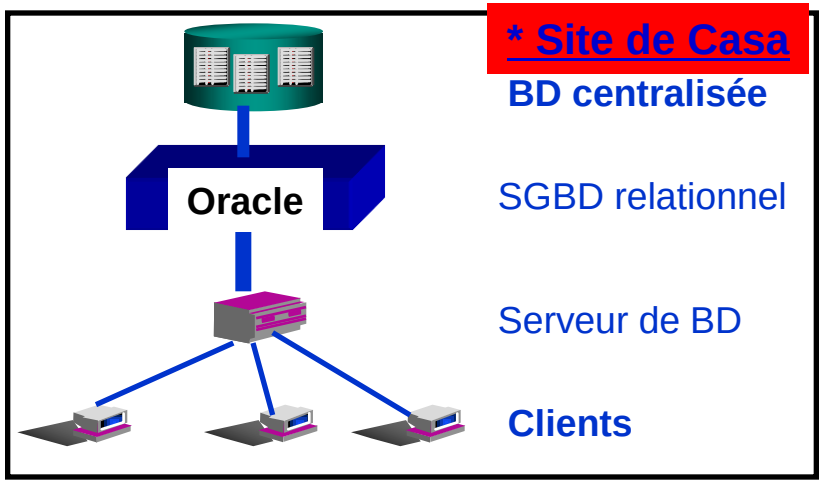
- La société X d'assurance santé est implantée sur l'ensemble du Maroc,
- Le siège est localisé à Casablanca,
- Dans un souci de service, la société crée de nouvelles filiales au Maroc,
- Chaque filiale gère les dossiers des adhérents de sa région,
- Chaque filiale peut accéder aux informations du siège ou d'une autre filiale.

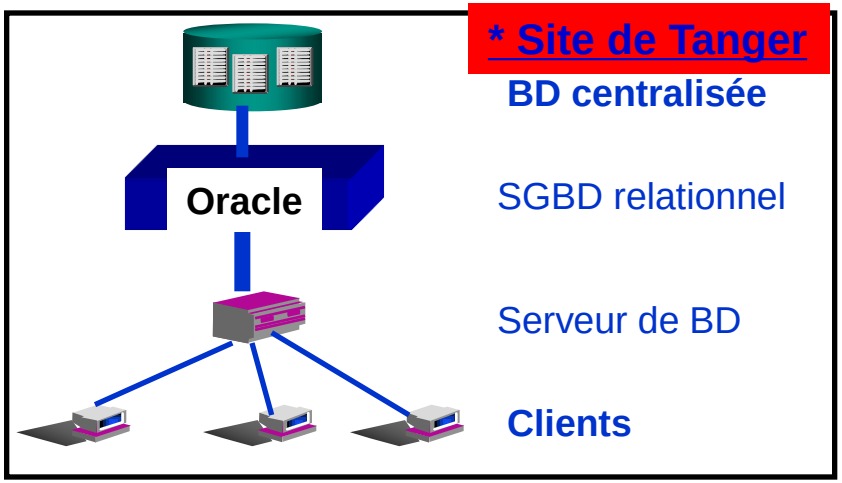
Hypothèse :

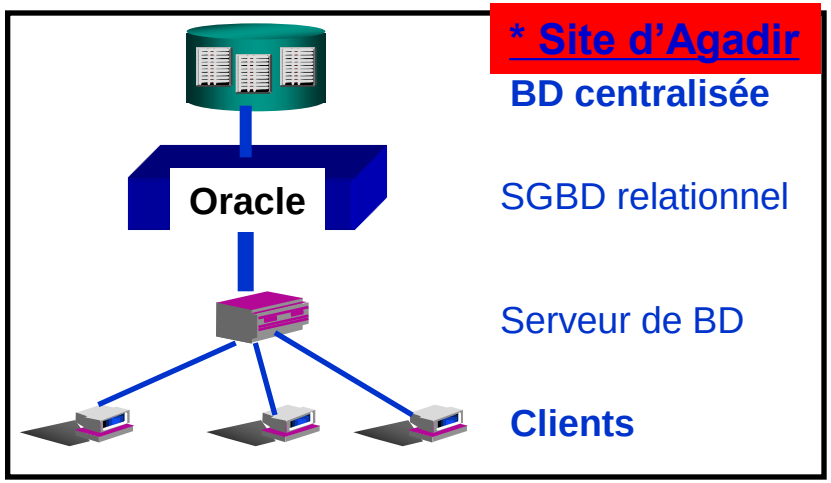
- Il existe un héritage informatique

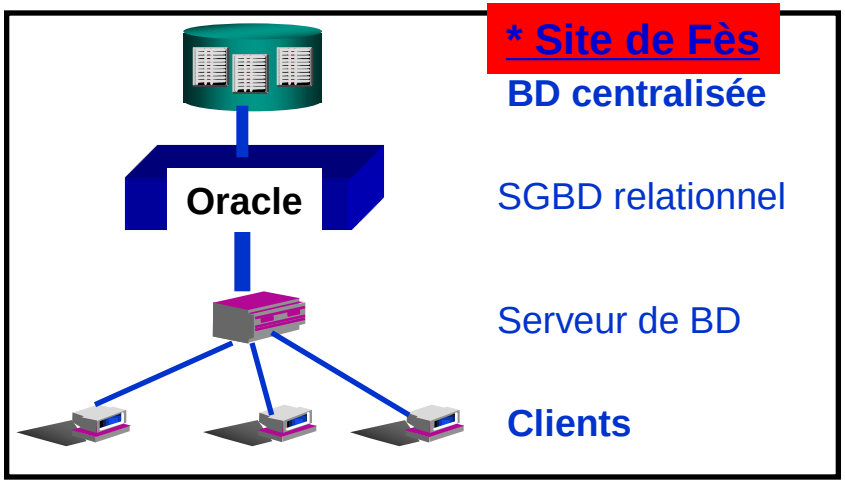
Quelle architecture adopter ?

Introduction
Problème 2.1 : Distribution géographique

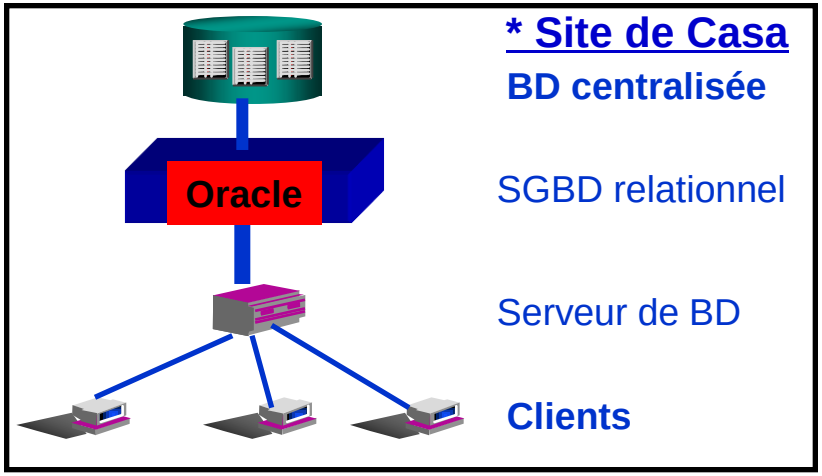


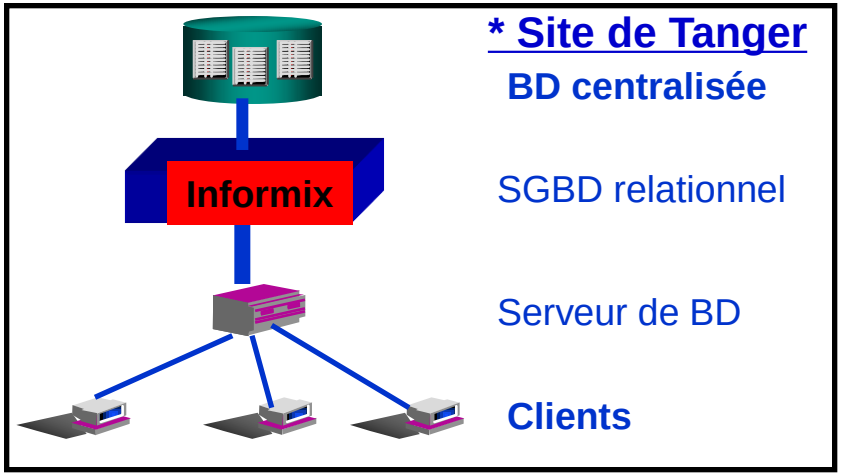


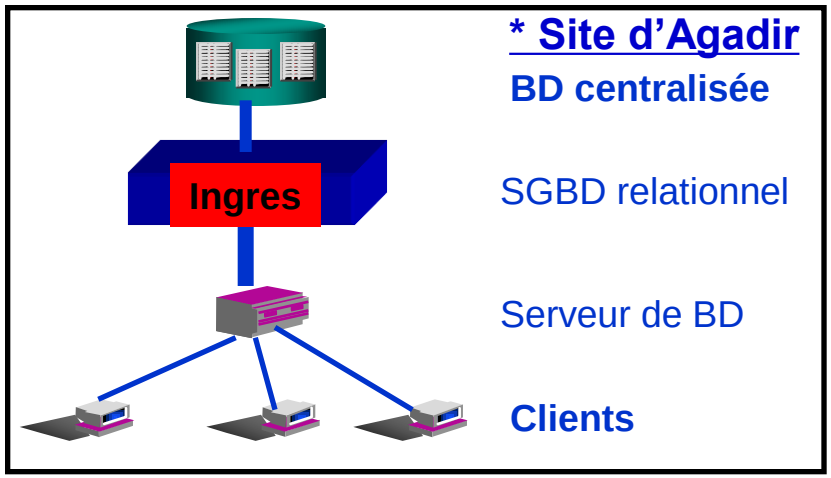


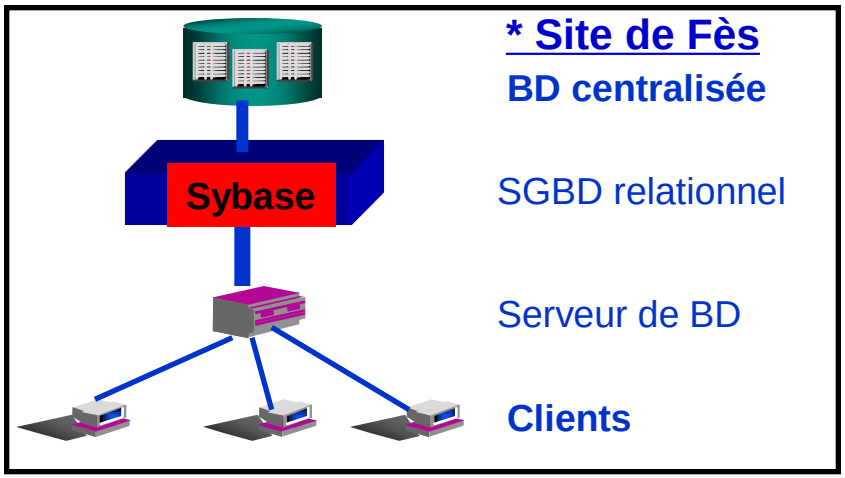


Introduction
Problème 2.2 : Hétérogénéité des SGBDs

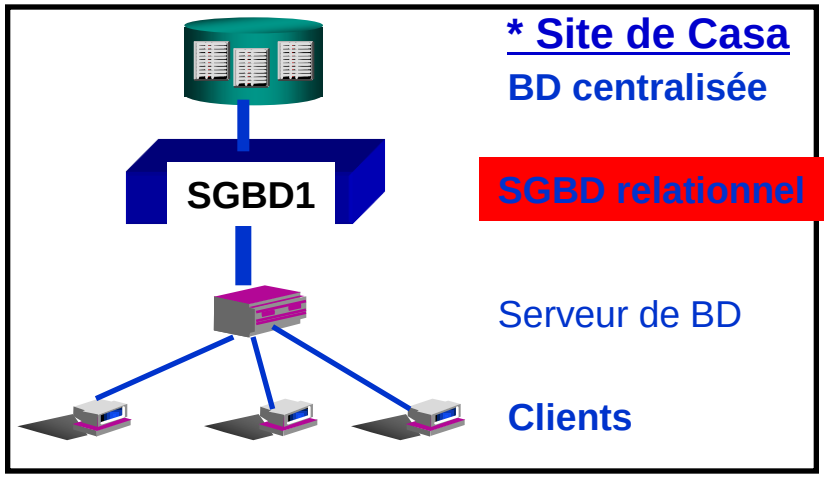


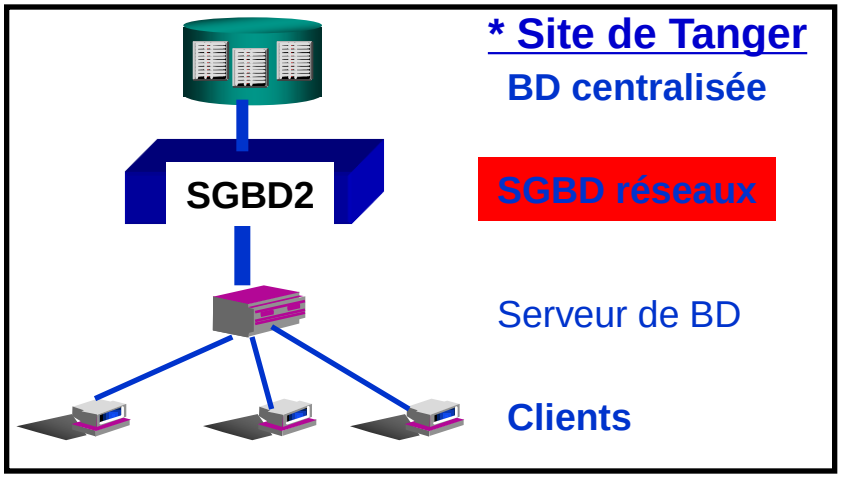


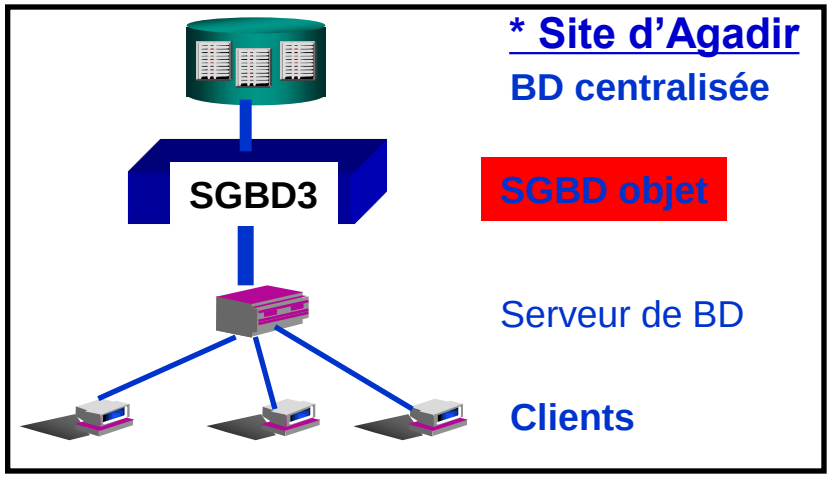


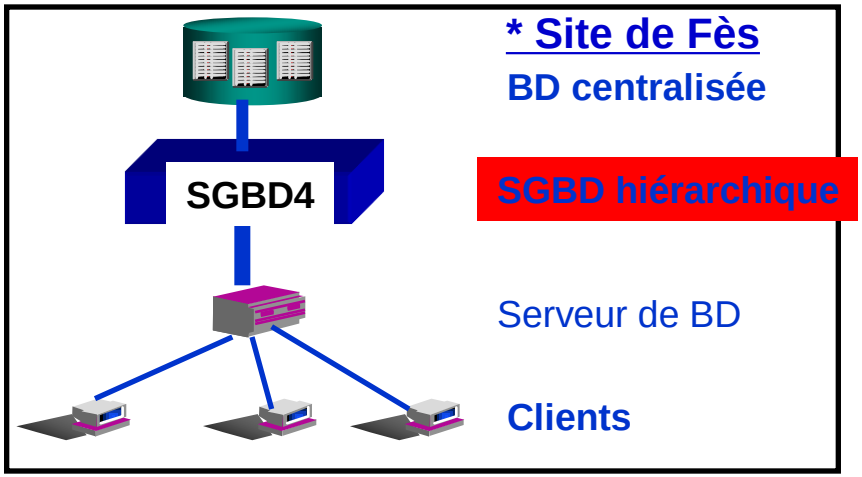


Introduction
Problème 2.3 : Hétérogénéité des modèles









Introduction
Question 2

Comment fusionner l'héritage pour construire une SEULE BASE DE DONNEES ?

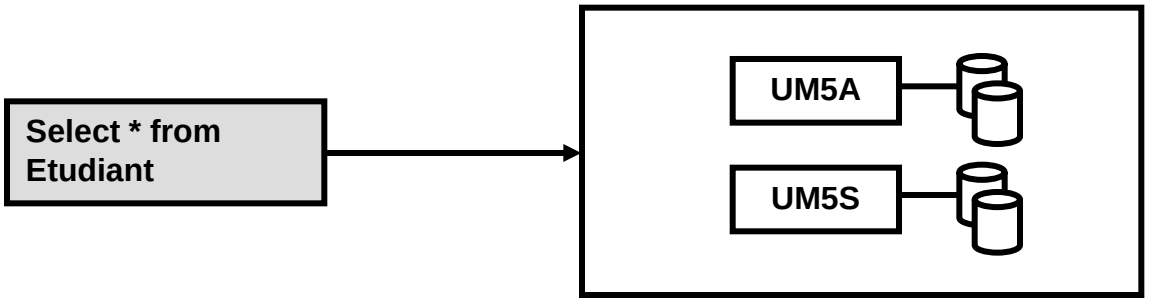
Les bases de données réparties

Définitions

- **Base de données répartie (BDR)**

Une BDR est un ensemble de bases de données gérées par des sites différents et apparaissant à l'utilisateur comme une base unique.

Exemple : les 2 universités de Rabat



- **SGBD réparti (SGBDR)**

Un SGBDR est un système qui gère une collection de BDs logiquement reliées, distribuées sur un réseau, en fournissant un mécanisme d'accès qui rend la répartition transparente aux utilisateurs.

Les bases de données réparties

Avantages et Inconvénients

Avantages :

- **Prendre en compte la répartition géographique des données**
- **Prendre en compte la distribution fonctionnelle des données**
- **Une meilleure disponibilité des données en présence de panne ou de dysfonctionnement des applications**
- **Une plus grande flexibilité afin d'assurer le partage des données hétérogènes et réparties**
- **Meilleures performances (données réparties sur plusieurs bases gérées par des serveurs différents mieux adaptés)**

Inconvénients :

- **Complexité des SGBDRs**
- **Problèmes de cohérence dus à la distribution du contrôle de données entre plusieurs sites**
- **Difficulté de changement (ex. intégration d'un nouveau type de SGBD)**

Les bases de données réparties

Objectifs des SGBDRs (1/4)

- **Multiclients multiserveurs**
- **Transparence à la localisation des données**
- **Meilleure disponibilité des données**
- **Autonomie locale des sites**
- **Support de l'Hétérogénéité**

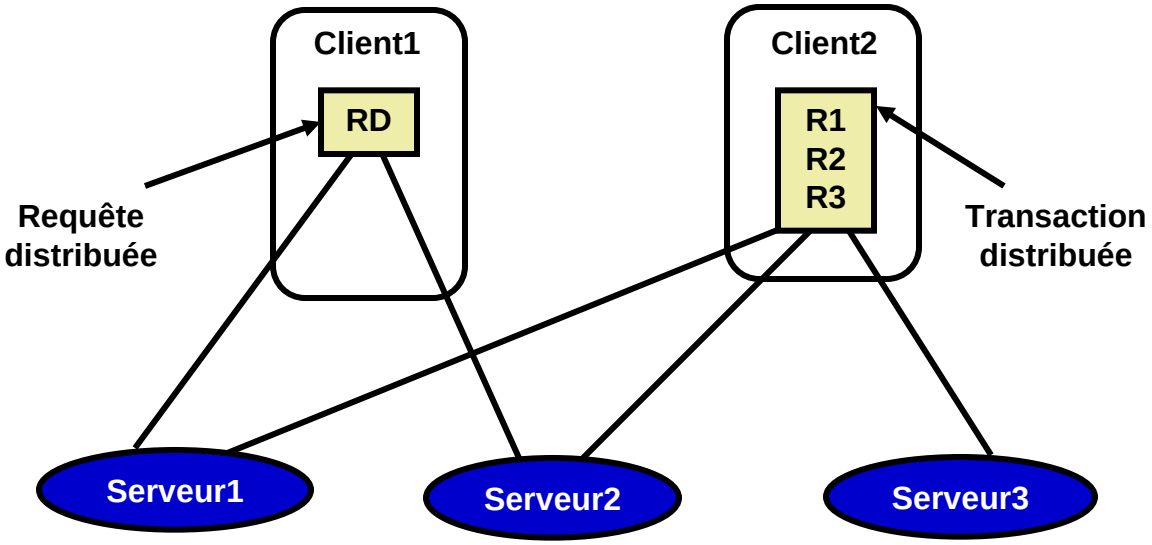
Les bases de données réparties

Objectifs des SGBDRs (2/4)

- **Multiclients multiserveurs**
  - **Fournir un mécanisme de contrôle des accès concurrents adapté**
  - **Garantir que l'effet de l'exécution simultanée des transactions est le même que celui d'une exécution séquentielle (sérialisabilité des transactions)**
  - **Permettre l'exécution des requêtes distribuées**

Une requête distribuée est une requête émise par un client dont l'exécution nécessite l'exécution de N sous-requêtes sur N serveurs avec N>1

Une transaction distribuée est une transaction qui met en œuvre plusieurs serveurs



Les bases de données réparties

Objectifs des SGBDRs (3/4)

- **Transparence à la localisation des données**

Propriété d'un SGBDR permettant d'écrire des requêtes avec des noms d'objets ne contenant pas la localisation des données

Avantages :

- **Simplifier la vue utilisateur et l'écriture de ses requêtes**
- **Introduire la possibilité de déplacer les objets sans modifier les requêtes**

Inconvénient :

- **Contraindre le SGBDR à rechercher les sites capables de générer des éléments de réponse à une requête pour l'exécuter (fonction pas évidente)**

Solution :

- **Utilisation d'un nom hiérarchique pour les objets : <objet>@<base>**
- **Utilisation d'un alias pour retrouver l'indépendance à la localisation**

Les bases de données réparties

Objectifs des SGBDRs (4/4)

- **Meilleure disponibilité**
  - **Disponibilité des données : une des justifications essentielles des SGBDR**
  - **La répartition permet de ne plus dépendre d'un site central**
  - **Gestion des copies : se replier sur une copie lorsqu'une autre est indisponible (site en panne) ===> Réplication**
  - **Assurer une meilleure disponibilité, c'est aussi garantir l'Atomicité des transactions**

- **Autonomie locale**

Garder une administration locale séparée et indépendante pour chaque serveur participant à la base de données répartie (pas d'administration globale)

===> les reprises après panne et les mises à niveau des logiciels doivent être accomplies localement et ne pas impacter les autres sites

- **Hétérogénéité**
  - **Capacité d'unifier des modèles et langages afin de générer des bases fédérées.**
  - **Intégration sémantique des bases**

Les bases de données réparties

Architecture des SGBDR (1/5)

Fonctions d'un SGBDR

Traitement des requêtes référençant des objets d'une BDR

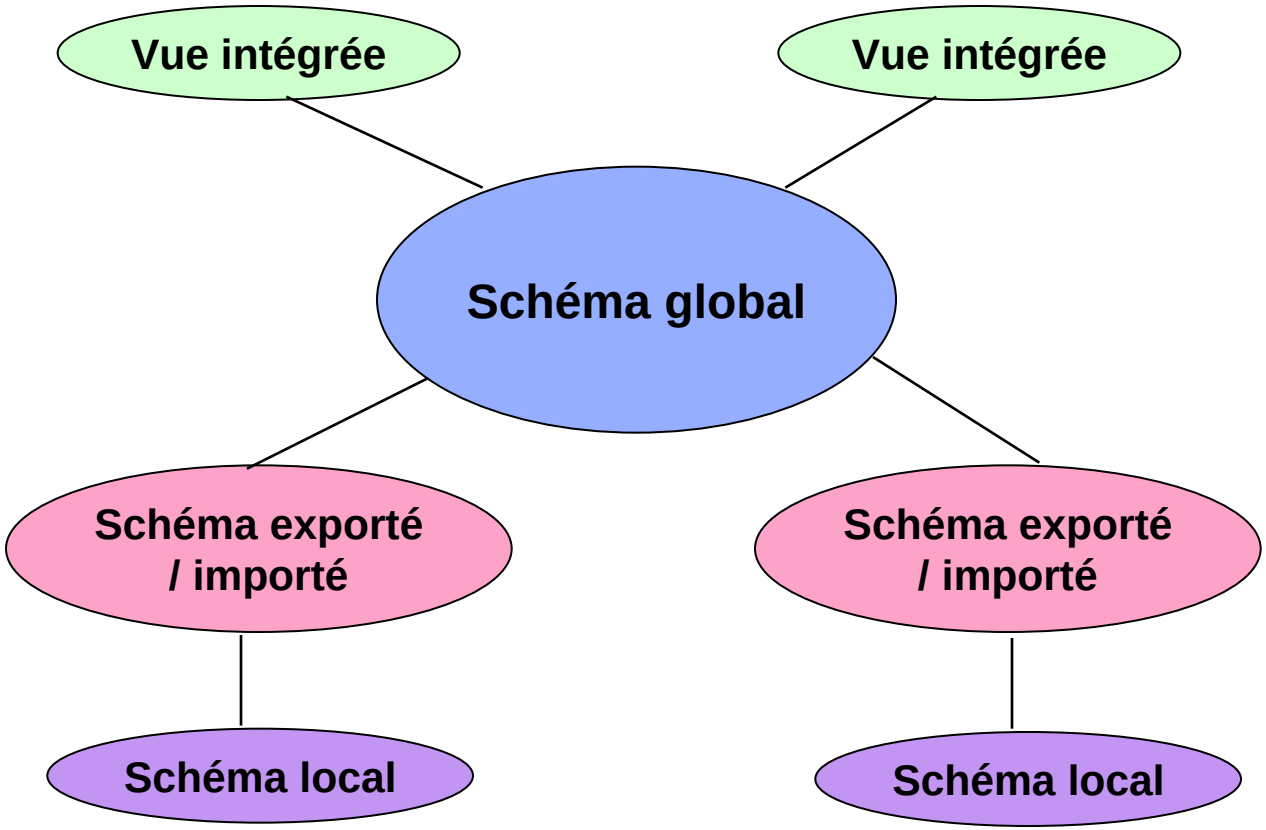
- **Assurer la décomposition de la requête distribuée en sous-requêtes locales envoyées à chaque site.**
- **Prendre en compte les règles de localisation lors de la décomposition.**
- **Evaluer globalement la requête distribuée en minimisant le transfert de données et en maximisant le parallélisme.**
- **Traduire les requêtes exprimées dans un langage Pivot (ex. SQL) en requêtes compréhensibles par le SGBD local (dans un contexte de BD hétérogènes).**
- **Router les mises à jour vers les sites concernés en assurant la gestion des transactions réparties (vérification des règles d'intégrité, contrôle des accès concurrents, gestion de l'atomicité des transactions distribuées).**

Les bases de données réparties

Architecture des SGBDR (2/5)

Organisation des schémas

Une BDR est décrite par différents niveaux de schémas



Les bases de données réparties

Architecture des SGBDR (3/5)

Organisation des schémas

Schéma local : Schéma décrivant les données d'une BD locale gérée par le SGBD local

Schéma exporté : Schéma décrivant les données exportées par un site vers les sites clientss

Schéma importé : Schéma exporté reçu par un site client

Schéma global : L'ensemble des schémas exportés par tous ou certains sites clients exprimé dans le modèle de référence du SGBDR et décrivant globalement la base répartie (le schéma global est non matérialisée dans les sites clients).

Vue intégrée : Schéma décrivant dans le modèle du SGBDR les données de la BDR accédées par une application.

Les bases de données réparties

Architecture des SGBDR (4/5)

Architecture de référence

Architecture s'articulant autour de 3 niveaux de fonctionnalités

Niveau local : présent sur chaque serveur permet d'exporter les données locales selon le modèle pivot du SGBDR.

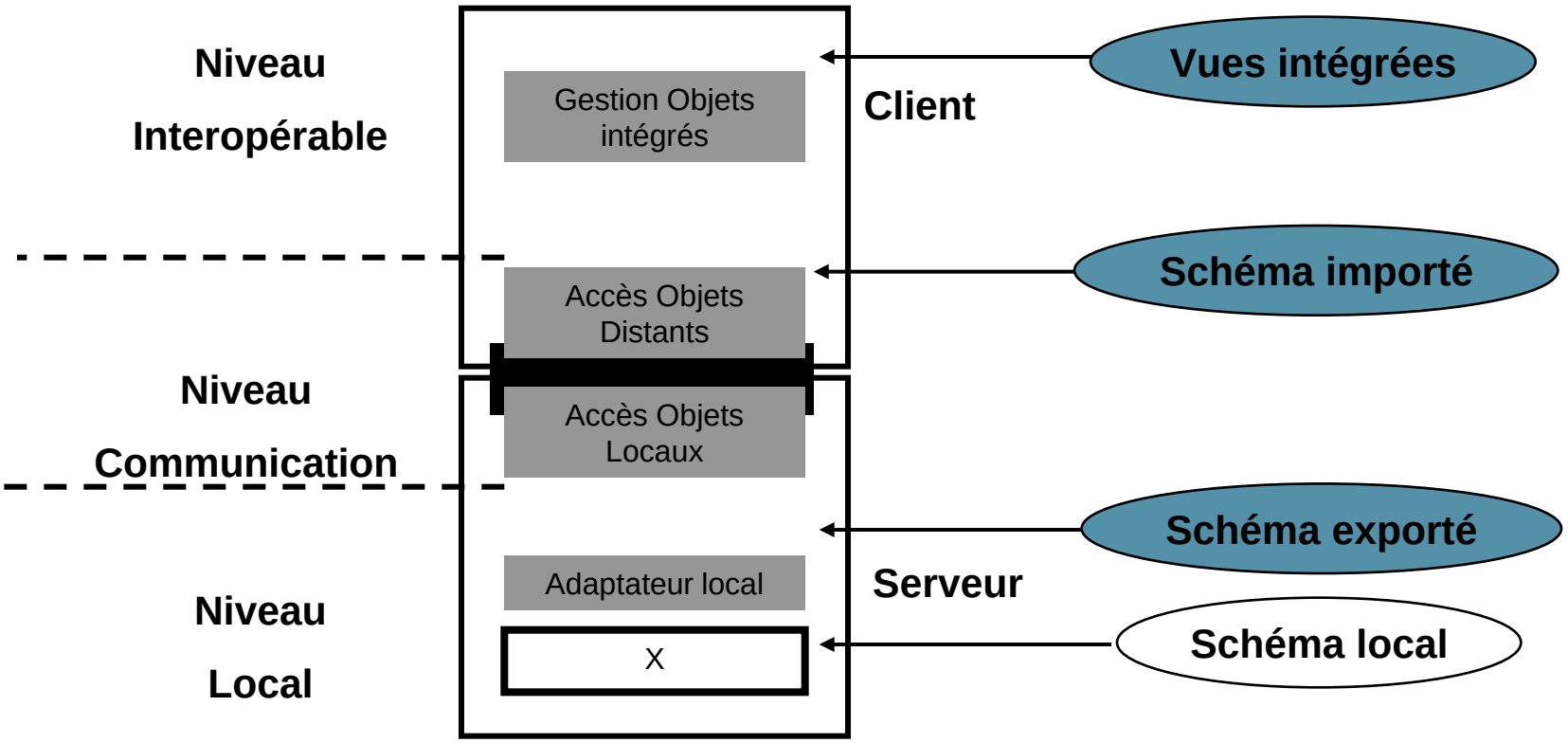
Niveau communication : permet de transmettre les sous-requêtes en provenance d'un site client au serveur dans le langage pivot d'échange. Ces sous-requêtes référencent le schéma exporté vers ce site client.

Niveau interopérabilité : permet de formuler des requêtes mettant en jeu des vues intégrées de la base. Il assure la décomposition des requêtes en sous-requêtes et le passage des vues intégrées aux différents schémas importés

Les bases de données réparties

Architecture des SGBDR (5/5)

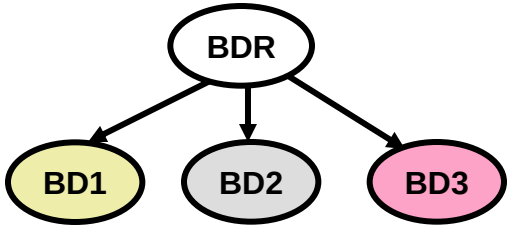
Architecture de référence



Les bases de données réparties

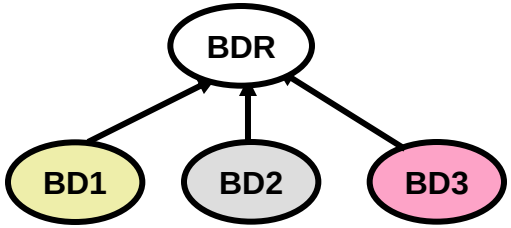
Conception des BDR

- **Conception descendante**
  - **Utilisée lors de la constitution de nouvelles bases de données**
  - **Un schéma conceptuel est tout d'abord élaboré puis les diverses entités de ce schéma sont distribuées sur les sites ===> définition des schémas locaux**



- **Conception ascendante**
  - **Intégration des BD locales existantes dans une base fédérée**

===> intégration des schémas locaux existants afin de constituer un schéma global



Les bases de données réparties

Les 12 règles d'un SGBD

- (1) Autonomie locale
- (2) Pas de site fédérateur
- (3) Exploitation en continue
- (4) Indépendance à la localisation
- (5) Règles de fragmentation
- (6) Duplications multiples
- (7) Requêtes distribuées
- (8) Transactions distribuées
- (9) Indépendance du matériel
- (10) Indépendance des systèmes d'exploitation
- (11) Indépendance du réseau
- (12) Indépendance du SGBD

Les bases de données réparties

1 – Autonomie locale

- Les données locales sont gérées localement
- Administration locale des données locales
- Site autonome pour ses propres opérations
- L'administration de la BDR est décentralisée
  - Administrateurs locaux coordonnés
  - Pas d'administration globale
  - Possibilité d'autoriser ou non un accès réparti à une base locale
  - Accès local possible simultanément aux accès répartis

Les bases de données réparties

2 – Pas de site fédérateur

- Pas de dépendance d'un site par rapport à un autre
- Avantages :
  - Architecture non vulnérable en cas de panne
- Inconvénients :
  - Pas de contrôle centralisé des accès concurrents
  - Pas de dictionnaire central
  - Pas de 'recovery' central
  - Pas d'exécution des requêtes centralisée

Les bases de données réparties

3 – Exploitation en continue

- Pas d'arrêt de la BDR pour la modification d'un site
  - Modification de la structure d'une BD locale (LDD)
  - Modification d'un SGBD (Release, Upgrade, ...)
- Extensibilité
  - Ajout, suppression ou modification d'un site
  - Opération locale et non globale (règle 1)
  - Propriété importante : évolution permanente

Les bases de données réparties

4 – Indépendance à la localisation

- Objectif le plus important : changement de la localisation des données sans changer les programmes
- Nécessité d'un dictionnaire réparti
- Avantages attendus :
  - Illusion, pour l'utilisateur, de travailler sur un seul site avec BD centralisée
  - Évolution des règles de localisation sans impact sur l'application
  - Invisibilité du ou des réseaux (règle 11)
- Inconvénient majeur :
  - Attention aux performances après un changement de localisation de données

Les bases de données réparties

5- Règles de fragmentation

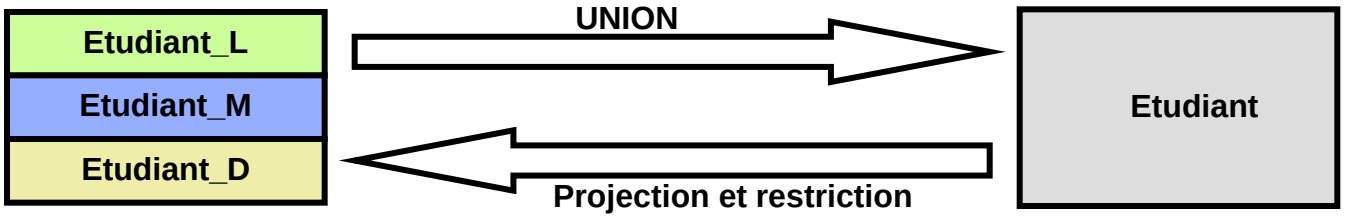
- Découpage d'une relation pour des raisons fonctionnelles
- Transparent pour l'utilisateur
- Fragment : sous-table obtenue par sélection de lignes et de colonnes à partir d'une table globale, localisée sur un site unique
- Trois types de fragmentations : Horizontal, Vertical et Mixte
- Opérateurs SQL utilisés
  - Projection et restriction pour fragmenter
  - UNION et JOINTURE pour reconstituer

- La conception ascendante d'une BD conduit à distribuer sur différents sites des fragments
- Pour la conception descendante, une table globale étant aussi divisée en fragments

Les bases de données réparties

Fragmentation horizontale

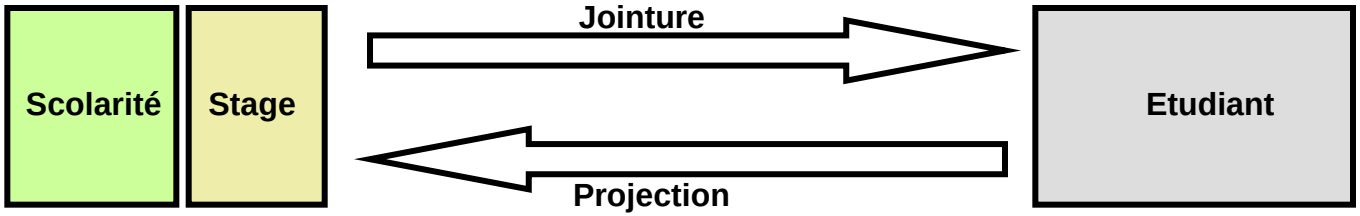
Découpage d'une table en sous-table par utilisation de prédicats permettant de sélectionner les lignes appartenant à chaque fragment



Les bases de données réparties

Fragmentation verticale

Découpage d'une table en sous-table par projections en permettant de sélectionner les colonnes composant chaque fragment



Duplication de la clé primaire sur chaque fragment

Les bases de données réparties

Fragmentation mixte

Résulte de l'application successive d'opérations de fragmentation horizontale FH et verticale FV sur une relation globale

Les bases de données réparties

6 – Duplication multiple

- Duplication des données sur des sites distants
- Gain de temps pour les accès en lecture (interdit en modification) → augmentation des performances
- Mises à jour par rafraîchissements périodiques à partir du site qui possède le fragment initial
- Inconvénient : pas de mise à jour immédiate
- Intéressant pour les données stables

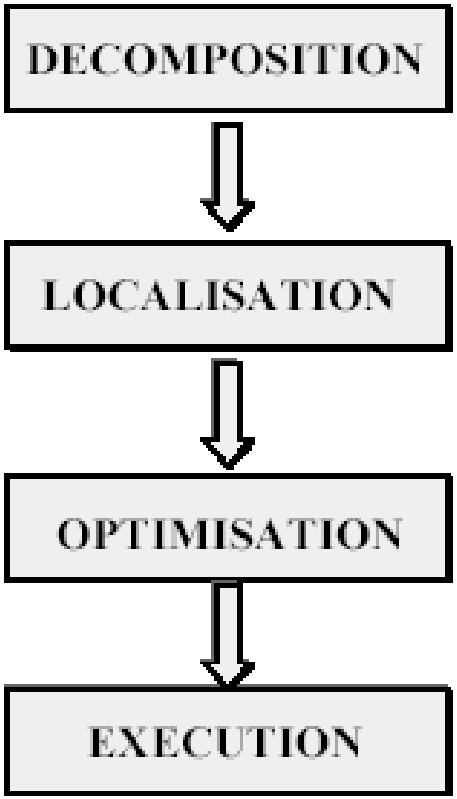
Les bases de données réparties

7 – Requêtes distribuées

- Évaluation des requêtes distribuées ou réparties → plan d'exécution réparti
- Le SGBD comprend que la requête traite avec des objets physiques distants
- La requête est reconstruite en tenant compte de la localisation des objets
- Les opérateurs de restriction (sélection, projection) qui réduisent la taille sont appliqués au plus tôt
- Utilisation du parallélisme

Les bases de données réparties

Evaluation d'une requête distribuée



SQL → ALGEBRE

Reconstruction de la requête avec le dictionnaire réparti

Ordonnancement et algorithme d'accès :

→règles de localisation

→ choix du site de transfert

→ semi - jointure

Les bases de données réparties

8- Transactions distribuées

- **Propriétés des transactions**

- **Atomicité : Une transaction doit effectuer toutes ses mises à jour ou ne rien faire du tout**

- **Cohérence : Une transaction doit faire passer la BD d'un état cohérent à un autre**

- **Isolation : Les résultats d'une transaction ne doivent être visibles aux autres transactions qu'une fois la transaction validée**

- **Durabilité : Dès qu'une transaction valide ses modifications, le système doit garantir que ces modifications seront conservées en cas de panne**

Dans le cadre d'un système réparti, chaque site peut décider de valider ou d'annuler une transaction ====> il faut donc coordonner les validations

Les bases de données réparties

Gestion des transactions

- **Validation en deux phases**

Principe : Diviser la validation en deux phases :

- **Phase 1 : réalise la préparation de l'écriture des résultats des mises à jour dans les BD et la centralisation du contrôle (sous la direction d'un site appelé coordinateur : site initiateur de la transaction)**
- **Phase 2 : réalisée seulement en cas de succès de la phase 1, intègre effectivement les résultats des mises à jour dans la BD de données répartie**

Tous les cas d'erreur sont traités grâce aux journaux sur chaque site

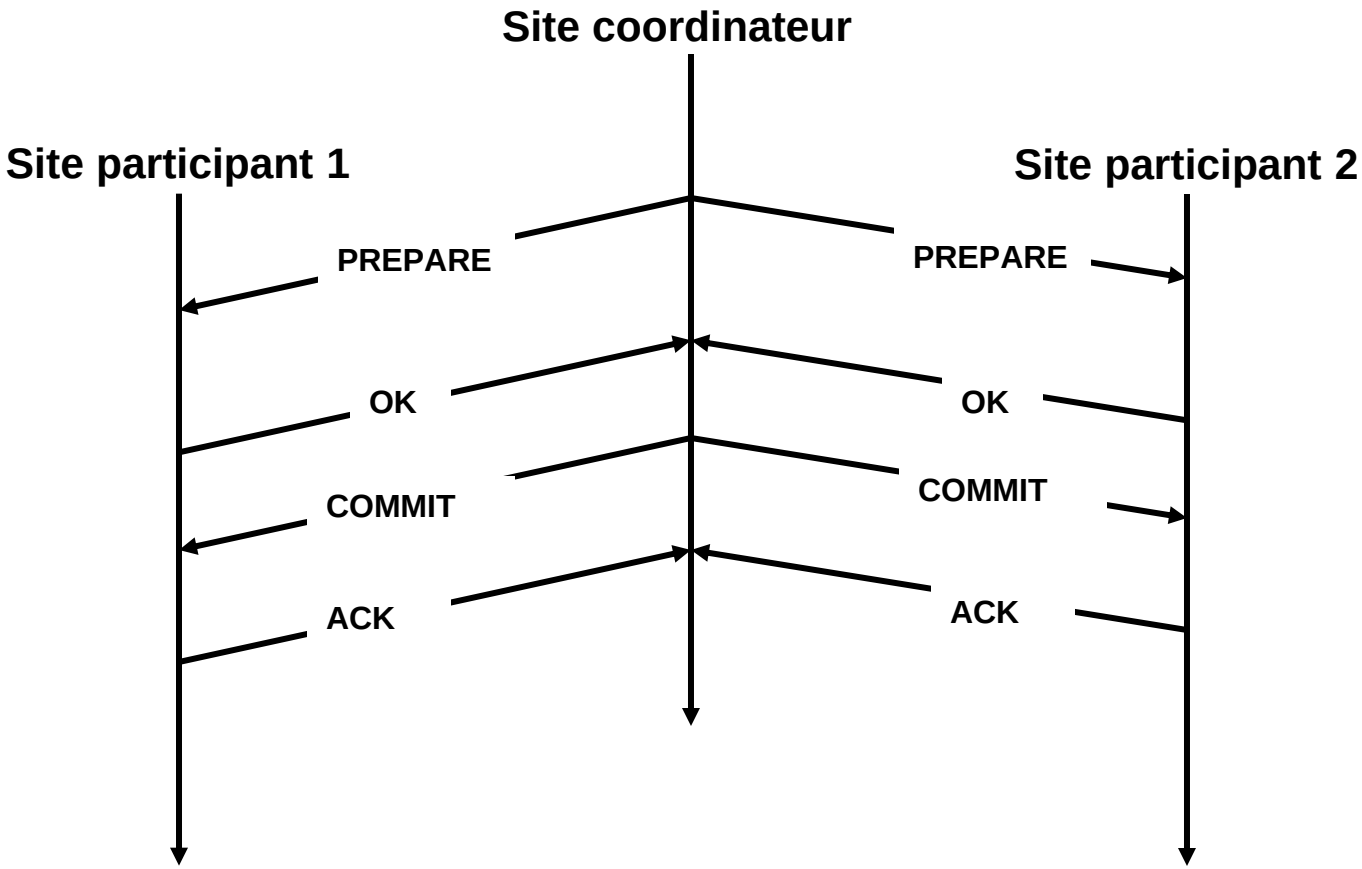
Concepts :

- **Protocole de validation en deux étapes : Protocole permettant de garantir l'atomicité des transactions dans un système réparti, composé d'une préparation de la validation et de centralisation du contrôle, et d'une étape d'exécution.**
- **Coordinateur de validation : Nœud d'un système réparti qui dirige le protocole en centralisant le contrôle.**
- **Participant à validation : Nœud d'un système réparti qui exécute des mises à jour de la transaction et obéit aux commandes des préparations, validation ou annulation du coordinateur.**

Les bases de données réparties

Gestion des transactions

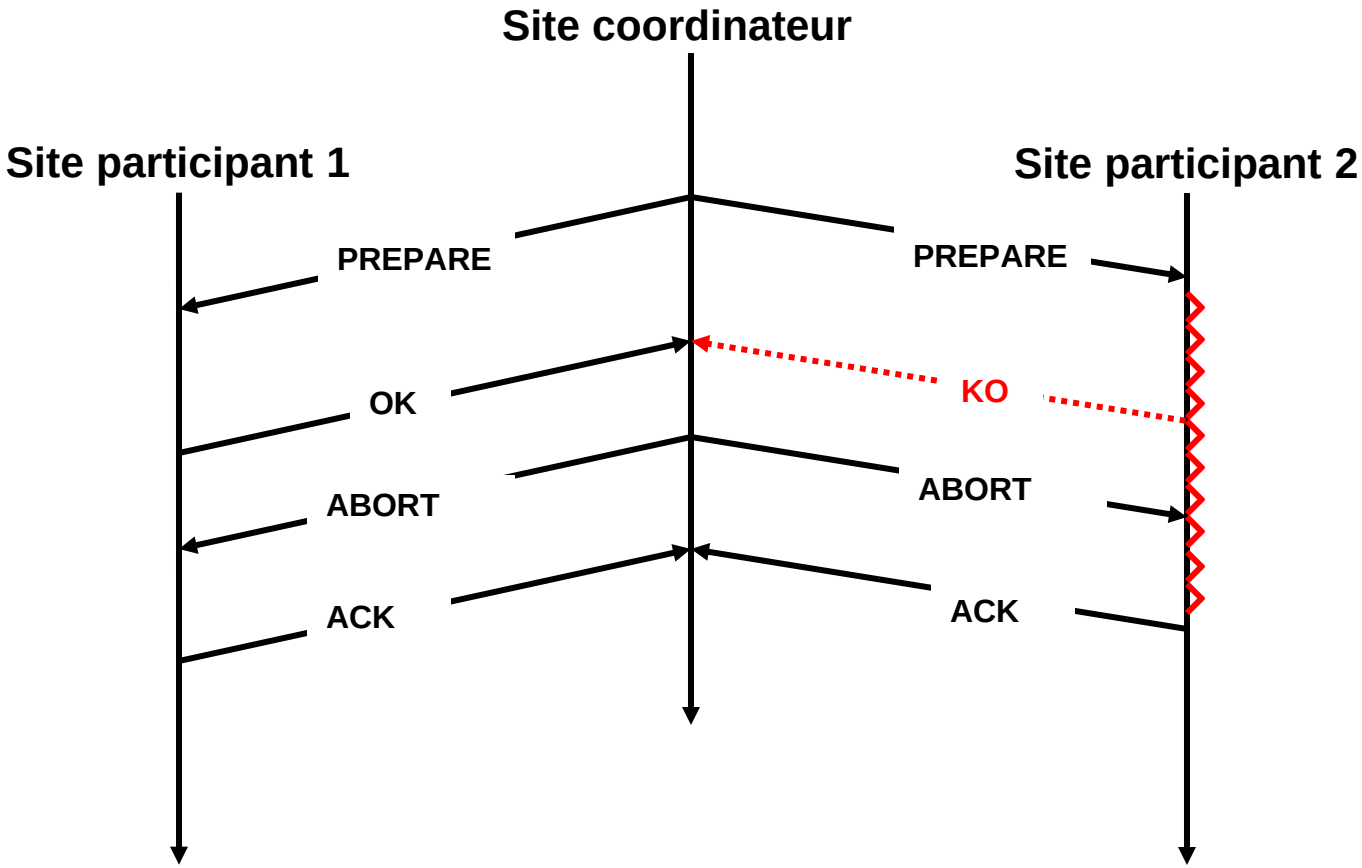
- **Validation en deux phases avec succès**



Les bases de données réparties

Gestion des transactions

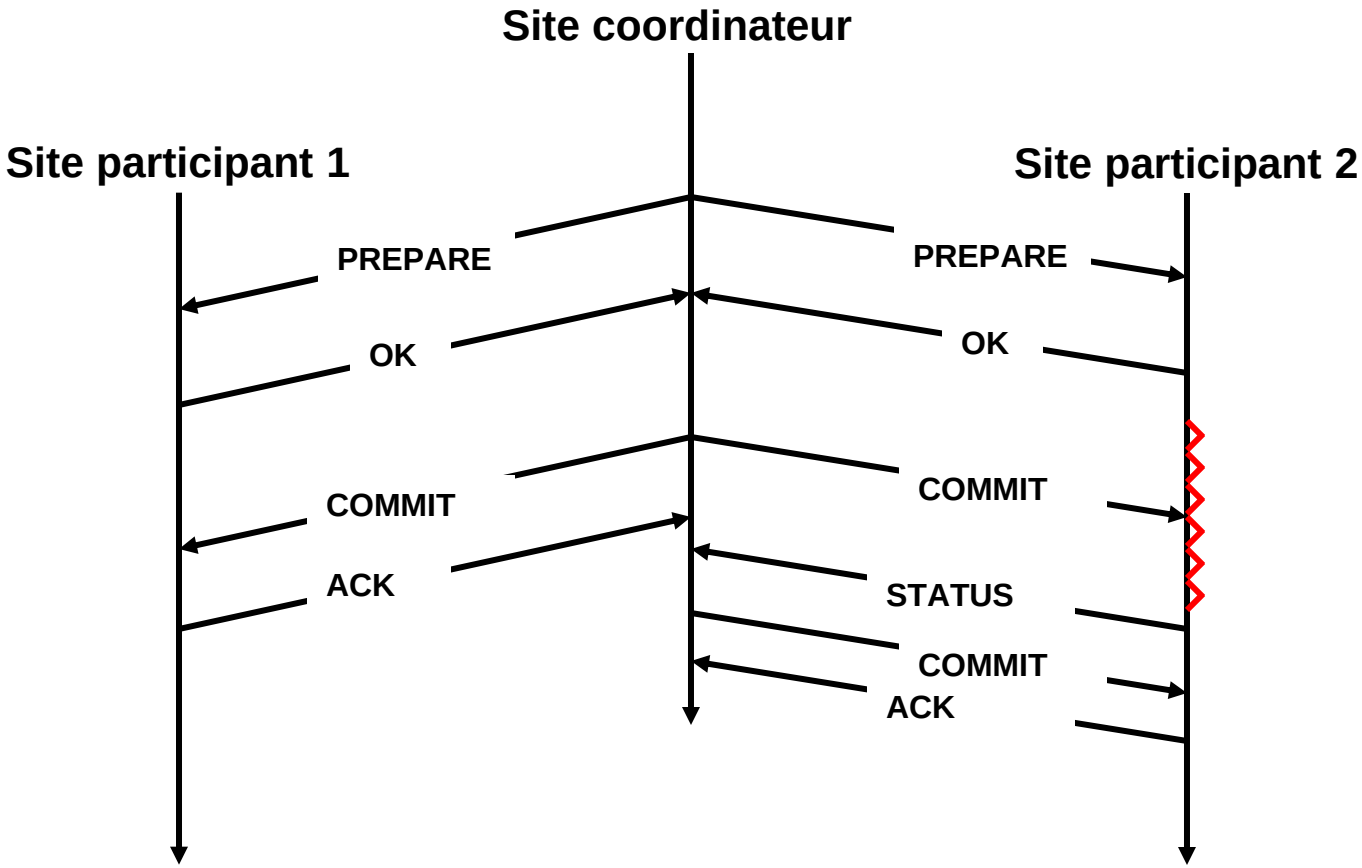
- **Validation en deux phases avec panne totale d'un participant**



Les bases de données réparties

Gestion des transactions

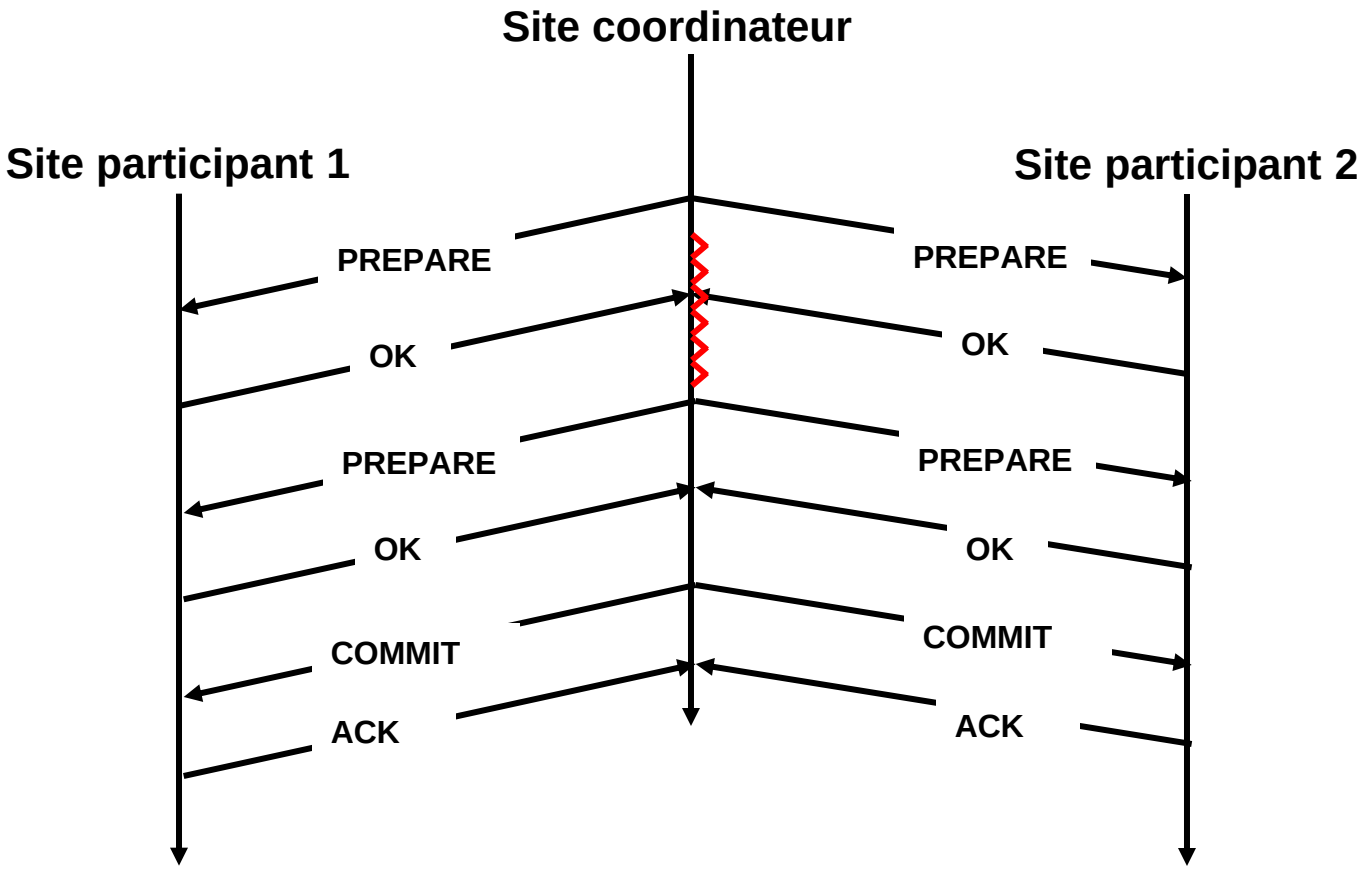
- **Validation en deux phases avec panne partielle d'un participant**



Les bases de données réparties

Gestion des transactions

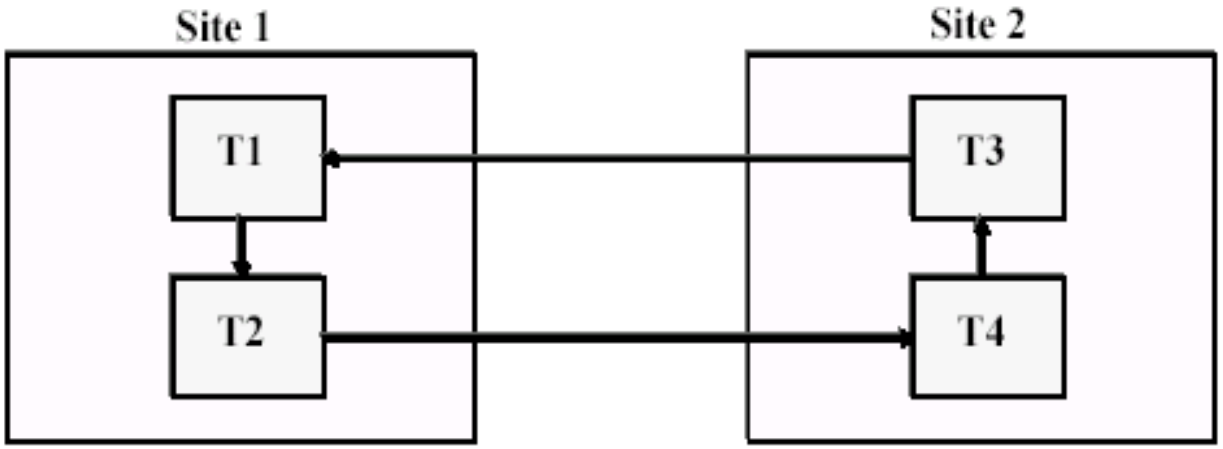
- **Validation en deux phases avec panne du coordinateur**



Les bases de données réparties

Problème des accès concurrents – verrou mortel global

- Pose de verrous sur des objets répartis
- Gestion d'un graphe d'attente réparti
  - Graphe Qui Attend Quoi (QAQ) réparti
  - Chaque site gère son propre graphe local et communique l'information sur ses propres transactions bloquées



Les bases de données réparties

9-10-11- Indépendance du matériel, des systèmes d'exploitation et du réseau

- Une base est "cliente" d'une autre base et vice – versa : architecture client - serveur
- C'est le "connect string" ou "chaîne hôte" qui permet cette indépendance
- Contenu du "connect string"
  - Protocole réseau
  - Serveur hôte (n° port)
  - Nom de l'instance de la base

Les bases de données réparties

12 – Indépendance du SGBD

- Très difficile d'arriver à cette indépendance
- Interfaces communes minimum :
  - Protocole d'échange
  - Lecture et traduction des dictionnaires, types de données ….
  - Fonctions réciproques client – serveur
  - Système de mise à jour cohérent
  - Verrouillage cohérent, …..
- Noyau standard avec SQL ANSI

Les bases de données réparties

Création des fragments dans les bases distantes

- **SQL-ANSI propose l'ordre CREATE FRAGMENT permettant de créer des tables distantes (cet ordre n'est pas encore implémenté)**
- **Oracle propose l'ordre COPY permettant de dupliquer un fragment d'une base vers une autre en utilisant les chaînes de connexion (connect string)**
  - **Syntaxe de la commande :**

```
COPY FROM spécification_base1
         TO  spécification_base2
{APPEND|CREATE|REPLACE|INSERT}
fragment [(Colonnes)]
USING SELECT …
```

  - **APPEND : [CREATE] + INSERT**
  - **CREATE : CREATE + INSERT**
  - **REPLACE : [DROP] + CREATE + INSERT**
  - **INSERT : INSERT**

Les bases de données réparties

Exemples de création des fragments

- **Création ou replacement du fragment (fragmentation horizontale)**

```
COPY FROM naciri/naciri@alias_base1
        TO naciri/naciri@alias_base2
REPLACE Client_Rabat
USING Select * FROM Client
            Where Ville='Rabat' ;
```

- **Création d'un fragment (fragmentation verticale)**

```
COPY FROM naciri/naciri@alias_base1
        TO naciri/naciri@alias_base2
CREATE Client_Infos (NumCli, Nom, Prenom, Ville)
USING Select NumCli, Nom, Prenom, Ville FROM Client;
```

Les bases de données réparties

Contraintes des fragments

- La commande COPY n'exporte pas les contraintes (sauf NOT NULL)
- Il faut les recréer (clés primaires, clés étrangères, contraintes de domaine, …)
- Il est impossible d'utiliser les DB LINKS pour les clés étrangères distantes
- Solution : Créer deux TRIGGERS :
  - Sur le fragment fils : le père doit exister
  - Sur le fragment père : suppression impossible si des fils sont présents

Les bases de données réparties

Travail de compte à compte

- **Ne pas travailler avec les véritables comptes propriétaires des données**
- **Chaque site distant doit créer un compte ayant accès aux objets répartis locaux**
- **Ces comptes ‘miroir’ sont créés par l’administrateur et reçoivent les droits d’accès par les propriétaires des données réparties**
- **Chaque responsable local de la base de données réparties ne connaît que les mots de passe des comptes ‘miroir’ distants**

Les bases de données réparties

Les liens inter-bases : DATABASE LINK

- Lien défini par un utilisateur pour relier deux bases
- Il faut connaître le login et le mot de passe du compte miroir distant
- Utilisation de la chaîne de connexion du serveur distant

```
CREATE DATABASE LINK lien_base2
CONNECT TO Nom_util_distant IDENTIFIED BY mot_passe_util_distant
USING 'base2';
```

Nom du lien

User/Password du compte miroir distant

Chaîne de connexion du serveur distant

- Suppression d'un lien

```
DROP DATABASE LINK lien_base2
```

- Dictionnaire de données : USER_DB_LINKS
- Utilisation d'un lien : Sur la base de données 'Base1'
  - Select * from client@lien_base2; -- liste la table distante Client
  - Update client@lien_base2 set Ville='Fès'; -- modification de la table distante Client

Les bases de données réparties

Indépendance à la localisation

- **Les synonymes :**
  - **Création d'un synonyme**

    CREATE SYNONYM Client FOR Client@lien_Base2;
  - **Suppression d'un synonyme**

    DROP SYNONYM Client;
  - **Synonyme d'une séquence distante :**

    CREATE SYNONYM Sequence_Client

    FOR Sequence_Client@lien_Base2;

- **Les objets virtuels :**
  - **Reconstitution d'une table fragmentée : VIEW**

    CREATE VIEW Client (NumCli, Nom, Ville) AS

    SELECT NumCli, Nom, 'Casablanca' FROM Client@lien_BaseCasa

    UNION

    SELECT NumCli, Nom, 'Rabat' FROM Client@lien_BaseRabat;

Les bases de données réparties

Indépendance à la localisation

- **Les procédures stockées :**
  - **Les données réparties sont encapsulées et ne sont pas accessibles directement aux développeurs clients**
  - **Les règles de fragmentation sont dans les procédures**
  - **Transparence de la localisation des données aux utilisateurs**

Les bases de données réparties

Les triggers INSTEAD OF

- Ces triggers s'appliquent sur les vues
- Les clients connaissent les objets virtuels et exécutent les ordres du LMD
- Les triggers INSTEAD OF prennent la main et font les mises à jour sur les fragments distants

- **Exemple :**

```
CREATE TRIGGER InseretClient
INSTEAD OF INSERT ON Client
FOR EACH ROW
BEGIN
IF :NEW.Ville='Rabat' THEN
INSERT INTO Client_Rabat@vers_BaseRabat VALUES(:NEW.NumCli, :NEW.Nom, :NEW.Adr) ;
ELSIF :NEW.Ville='Casablanca' THEN
INSERT INTO Client_Casa@vers_BaseCasa VALUES(:NEW.NumCli, :NEW.Nom, :NEW.Adr) ;
ELSE RAISE_APPLICATION_ERROR (-20455,'Entrer Rabat ou Casablanca');
END IF;
END;
/
```

Les bases de données réparties

Duplication et Réplication des données réparties

- **Distribution**
  - **Bases de données distribuées ou réparties**
  - **Sans redondance**

- **Duplication**
  - **Duplication locale d'un fragment éloigné maître**
  - **Fragment locale en lecture seule**
  - **Utilisation de la notion de cliché**
  - **Duplication synchrone ou asynchrone**

Les bases de données réparties

Duplication et Réplication des données réparties

- **Réplication**
  - **Pas de fragment maître**
  - **Réplication synchrone : utilisation de jetons**
  - **Réplication asynchrone : problèmes de cohérence**

Avantages :

- **Améliorer les performances**

  Utilisation de la copie la plus proche du client => évite des transferts inutiles

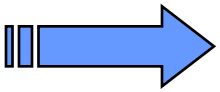
- **Augmenter la disponibilité des données**

  Lors d'une panne d'un serveur, on peut se replier sur un autre disposant d'une copie des données

  Avec N copies sur des serveurs différents => Disponibilité = 1 – probabilité_panne ^{N}

Inconvénients :

- **Assurer la convergence des copies**
- **Offrir une transparence de gestion aux clients : les clients doivent croire à l'existence d'une seule copie**



Le SGBD doit assurer la diffusion des mises à jour aux copies et le choix de la meilleure copie lors des accès

Les bases de données réparties

Duplication des données : différentes possibilités

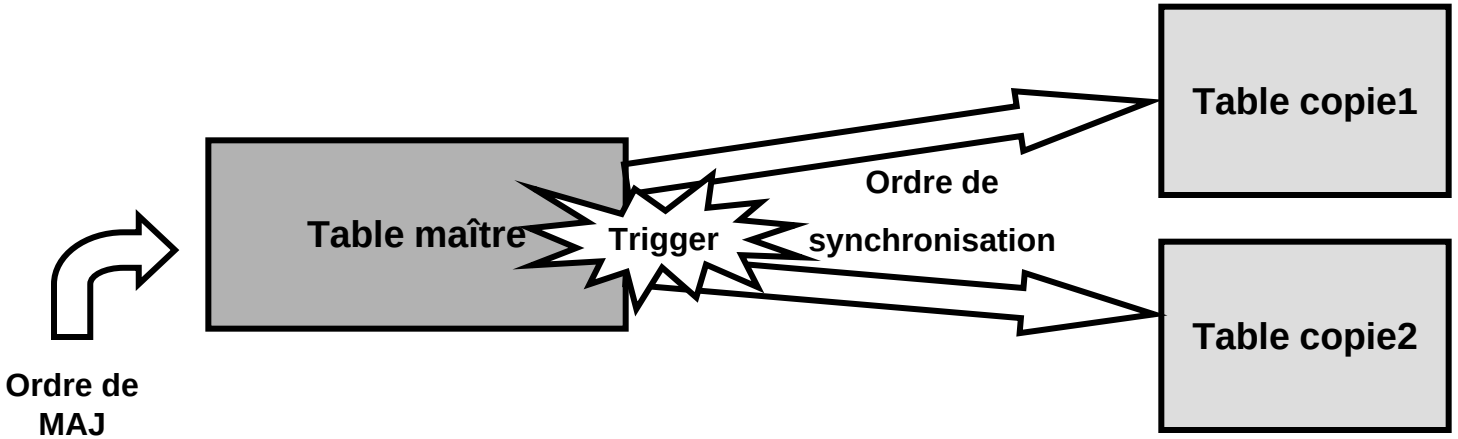
- **Duplication d'une base de données entière : Utilisation des utilitaires Export/Import**
- **Duplication d'une table : Utilisation de COPY ou CREATE**
  - **Exemple : Create table Copie as select * from table_maitre@dblink**
- **Duplication synchrone : propager immédiatement les modifications apportées aux données sources vers les copies ➔ Utilisation des Trigger ou Trigger instead of**
- **Duplication asynchrone : propager les modifications apportées aux données sources vers les copies à des intervalles prédéfinis.**
  - **Utilisation d'un programmateur**
  - **Oracle utilise la notion de SNAPSHOT ou vues matérialisées**

Les bases de données réparties

Duplication synchrone des données

- **Mise à jour instantanée de la copie pour toute modification de la table maître**

- **Utilisation des trigger de type 'before' qui propage la mise à jour sur la table image**



Les bases de données réparties

Duplication asynchrone par Oracle : SNAPSHOT ou les vues matérialisées

- **Un Snapshot (cliché) est un fragment en lecture seule**

- **Un cliché est constitué par une requête SQL**
  - **Image simple : utilisation d'une seule table, pas d'opérations ensemblistes, pas de GROUP BY.**
  - **Image complexe : autrement**

- **Le cliché est rafraîchi à intervalles réguliers (refresh) ou à la demande (manuellement)**
  - **Rafraîchissement rapide (uniquement pour les images simples) : uniquement les modifications sont propagées et non toute la table**
  - **Rafraîchissement complet : régénérer complètement l'image**

Les bases de données réparties

Principe

- **Création d'un cliché avec les méthodes de rafraîchissement et le contenu choisis**
- **Pour chaque table maître qui alimente un cliché, il faut créer un journal d'images (SNAPSHOT LOG)**
- **Le journal contient les mises à jour différées.**
- **Une table maître (même journal) peut alimenter plusieurs fragments dupliqués**

Les bases de données réparties

Création d'un cliché

CREATE SNAPSHOT [nom_schéma.]Nom_image

[Spécification de stockage]

[REFRESH [FAST|COMPLETE|FORCE][START WITH Date1]

[NEXT date2]

AS Requête;

- REFRESH : mode est fréquence de rafraîchissement

- **FAST : rapide**
- **COMPLETE : complet**
- **FORCE : laisser le choix à Oracle**

Premier rafraîchissement : date1

Puis rafraîchissement toutes les date2

Si REFRESH est omis =➔ aucune rafraîchissement

Les bases de données réparties

Exemple

Création d'un cliché avec rafraîchissement toutes les 10 jours

```
CREATE SNAPSHOT Image_client
REFRESH FAST
START WITH SYSDATE
NEXT SYSDATE + 10
AS Select * from client@dblink;
```

Création d'un journal image (créé sur la base source)

```
CREATE SNAPSHOT LOG ON Client
```