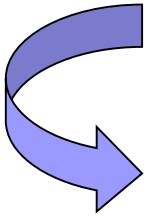
Chapitre
Bases de Données Réparties
Partie 1

Définitions (1/3)

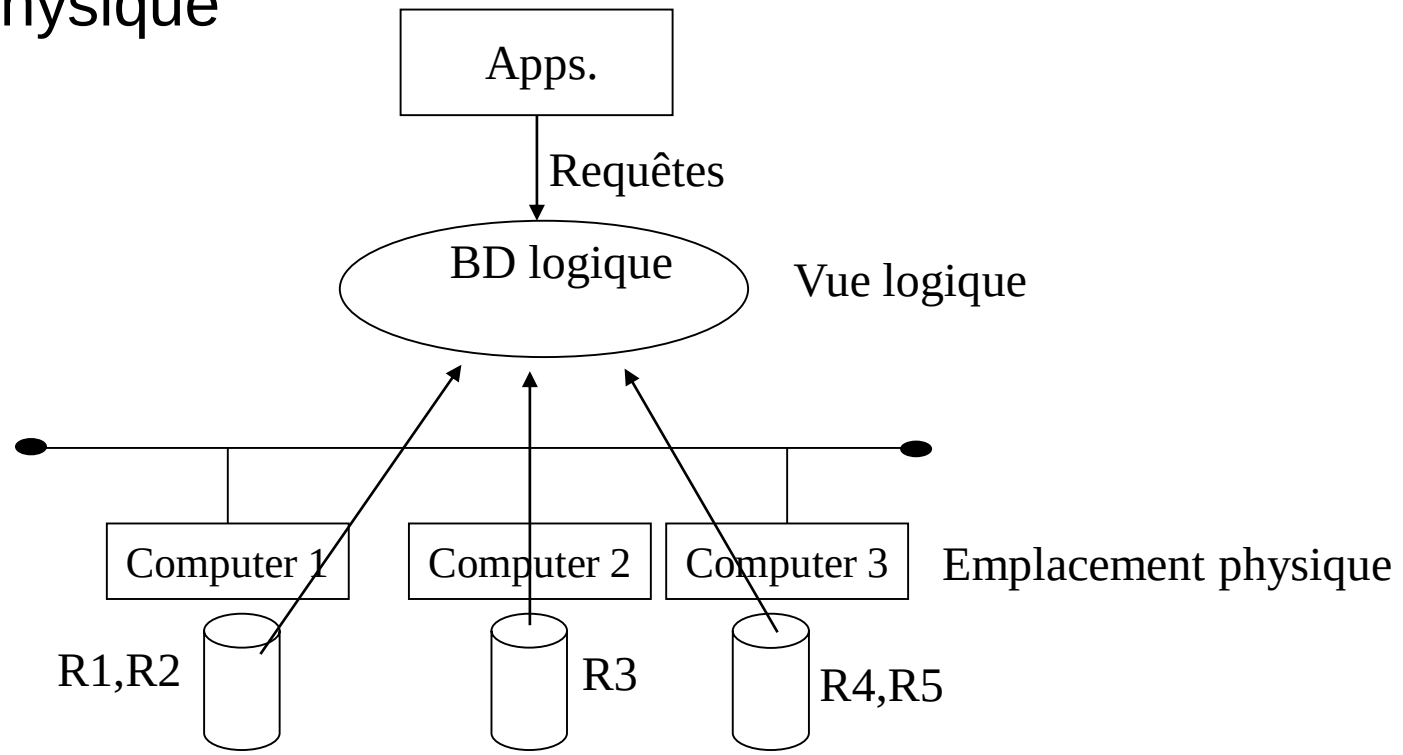
- Base de données reparties?
  - Une base de données répartie (distribuée) est une base de données logique dont les données sont distribuées sur plusieurs SGBD et visibles comme un tout.
  - Une BDR est une base de données dont les différentes parties sont stockées sur des sites (géographiquement distants), reliés par un réseau.



  - Une BDR est une BD stockée sur plusieurs sites (machine + BD locale) connectés par un réseau et :
    - Logiquement reliés
    - Physiquement distribués

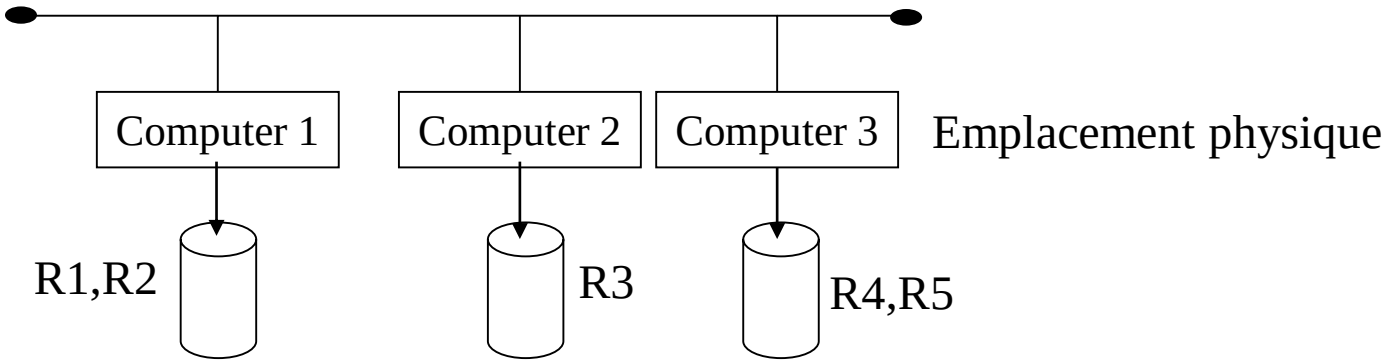
Définitions (2/3)

- Logiquement reliés
  - Les applications voient les données comme une seule BD indépendamment de son emplacement physique



Définitions (3/3)

- Physiquement distribués
  - La BD est placée sur différentes machines d'un réseau.



Objectifs (1/4)

- Permet aux utilisateurs de partager des données géographiquement reparties.
- Correspond à un besoin de décentralisation des organisations
- Avantages
  - Partage des données
  - Fiabilité, disponibilité des données
  - Accroissement de la vitesse de traitement
- Inconvénients
  - Complexité des SGBDs
  - Risque d'erreurs + important
  - Surcoût du traitement du à la communication inter-sites

Objectifs (2/4)

- Selon Christopher Date (Introduction to Database Systems) « To the user, a distributed system should look exactly like a nondistributed system. »
- Une BDR doit, donc, permettre :
  - Autonomie locale
  - Egalité entre sites (pas de site « central »)
  - Fonctionnement continu (pas d'interruption de service)
  - Localisation transparente
  - Fragmentation transparente
  - Indépendance à la réplication
  - Exécution de requêtes distribuées
  - Gestion de transactions réparties
  - Indépendance vis-à-vis du matériel
  - Indépendance vis-à-vis du Système d'Exploitation
  - Indépendance vis-à-vis du réseau
  - Indépendance vis-à-vis du SGBD

Objectifs (3/4)

- Autonomie locale
  - La BD locale est complète et autonome (intégrité, sécurité), elle peut évoluer indépendamment des autres
- Égalité entre sites
  - Un site en panne ne doit pas empêcher le fonctionnement des autres sites (même si certaines perturbations sont possibles)
- Fonctionnement continu
  - La distribution permet une résistance aux fautes et aux pannes
- Localisation transparente
  - Accès uniforme aux données quel que soit leur site de stockage
- Fragmentation transparente
  - Des données (d'une même table) éparpillées doivent être vues comme un tout
- Indépendance à la réplication
  - Les données répliquées doivent être maintenues en cohérence

Objectifs (4/4)

- Requêtes distribuées
  - L'exécution d'une requête peut être répartie (automatiquement) entre plusieurs sites
- Transactions réparties
  - Le mécanisme de transactions peut être réparti entre plusieurs sites
- Indépendance vis-à-vis du matériel
  - Le SGBD fonctionne sur les différentes plateformes utilisées
- Indépendance vis-à-vis du SE
  - Le SGBD fonctionne sur les différents SE
- Indépendance vis-à-vis du réseau
  - Le SGBD est accessible à travers les différents types de réseau utilisés
- Indépendance vis-à-vis du SGBD
  - La base peut être distribuée sur des SGBD hétérogènes

SGBD Reparti

- Un SGBD reparti assure la gestion d'une BD repartie
- Objectifs
  - Exécution des transactions
    - *locales* : accès aux données sur site
    - *globales* : accès sur plusieurs sites
  - Cohérence des données
  - Contrôle de concurrence
  - Reprise après panne
  - Optimisation de questions
  - Indépendance des applications
    - machine
    - système d'exploitation
    - protocole réseau

Conception des BD Réparties

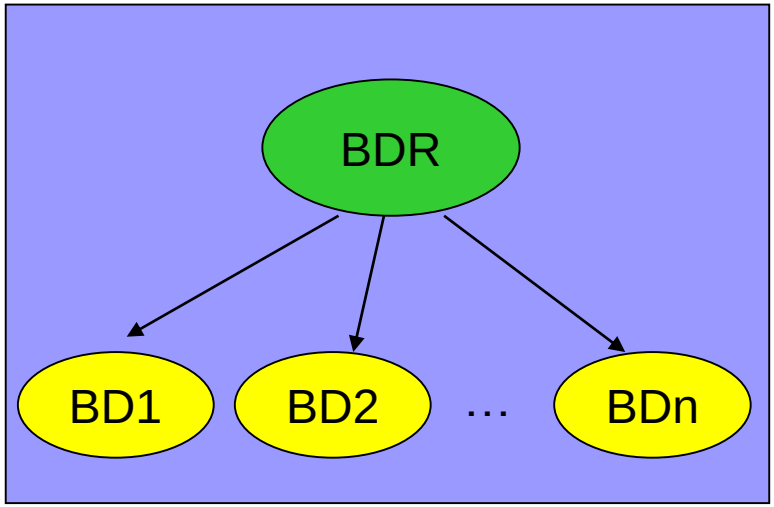
- Deux approches
  - Descendante : Top-down (du centralisé au distribué)
  - Ascendante : Bottom-up (à partir de SGBDs existants vers des vues intégrées)

Conception des BD Réparties

- **Approche descendante (Top-down)**
  - On commence par définir un schéma global de la base de données répartie (description globale et unifiée de toutes les données de la BDR). Puis, on le distribue sur les différents sites en des schémas locaux.
  - Pour déterminer les schémas locaux, on peut utiliser plusieurs méthodes:

    - Stocker une relation sur un site
    - La réplication
    - La fragmentation
    - Réplication + fragmentation

  - L'approche *top down* est intéressante quand on part du néant.
    Si les BDs existent déjà,
    la méthode *bottom up* est utilisée.



Conception des BD Réparties

Réplication

- Copie de chaque relation sur plusieurs sites.
- Réplication complète= copie sur tous les sites.
  - Avantages
    - Disponibilité des données.
    - Augmentation du parallélisme
    - Diminution du coût imposé par les transmissions
  - Inconvénients
    - Difficulté d'assurer la cohérence des différentes copies.
    - Propagation des mises à jour.

Conception des BD Réparties

Fragmentation

- Elle consiste à découper les relations en sous-relations appelées fragments.
- La répartition se fait donc en deux étapes: la fragmentation et l'allocation de ces fragments aux sites intéressés.
- Pourquoi fragmenter?
  - Généralement les applications utilisent des sous-ensembles de relations.
  - Une relation entière peut représenter une unité de distribution très grande
  - Utilisation de petits fragments permet de faire tourner plus d'un processus simultanément.
- Comment fragmenter?
  - On distingue trois possibilités de fragmentation:
    - Fragmentation Horizontale, Verticale et Hybride

Conception des BD Réparties

- Fragmentation correcte?
  - **Complétude:** R fragmentée en R_1 R_2 ,…,R_n chaque élément se trouvant dans R doit figurer dans au moins un fragment R_i
    - Évite les pertes de données pendant la fragmentation
  - **Reconstruction:** soit la relation *R* , *F*= {R_1, R_2,., R_n}
    - Il est toujours possible de reconstruire *R* en appliquant des opérations sur F
  - **Disjonction:** fragments de *R* contient des sous ensembles de R. $R_i \cap R_j = \varnothing$.
    - Garantit l'absence de redondance

Conception des BD Réparties

Fragmentation horizontale

- La fragmentation horizontale concerne les données.
- Chaque fragment représente un ensemble de tuples.
- Pour fragmenter, on a besoin d'information sur la BD (schéma global,…) et les applications (requêtes utilisées,…).
- Les fragments sont définis par des opérations de sélection sur les relations.

Conception des BD Réparties

Exemple

Proj	PNO	PNom	Budget	Location
	P1	Bioinfo	500000	Lausanne
	P2	ELearn.	300000	Rio
	P3	Plasma	100000	Geneva
	P4	Aircraft	150000	Lausanne

Conception des BD Réparties

Exemple (suite)

Proj 1	PNO	PNom	Budget	Location
	P1	Bioinfo	500000	Lausanne
	P2	ELearn.	300000	Rio

Proj 2	PNO	PNom	Budget	Location
	P3	Plasma	100000	Genève
	P4	Aircraft	150000	Lausanne

Relations avec BUDGET > 200.000 va dans Proj1 et le reste va dans Proj2.
Proj1= σ(budget>200.000) Proj
Proj2= σ(budget $\leq$ 200.000) Proj

Conception des BD Réparties

- L'opérateur de partitionnement est la *sélection* (σ)
- L'opérateur de recomposition est l'*union* (∪)
  - La reconstruction de la relation est, donc, définie par l'union des fragments.

Conception des BD Réparties

Fragmentation Verticale

- La fragmentation concerne le schéma.
- Les fragments sont définis par des opérations de projection.
- La reconstruction est définie par des jointures.
- La clé doit être répétée dans chaque fragment.
- Pour appliquer la fragmentation verticale, il existe deux possibilités:
  - **Clustering:** affecter chaque attribut à un fragment, puis à chaque étape fusionner certains fragments et s'arrêter lors de la satisfaction de certaines conditions.
  - **Splitting:** on part de la relations, puis on décide de la partitionner en des fragments en se basant sur des informations concernant les applications et les attributs.

Conception des BD Réparties

- **Propriétés de la fragmentation verticale**
  - Fragmentation verticale de R
    - Ai ⊆ attributs(R)
    - Fragments: Ri = πAi(R)
  - Complète
    - Attributs(R) = A1 ∪ A2 ∪ … ∪ An
  - Disjointe
    - A1 ∩ A2 ∩ … ∩ An = clé(R)
  - Reconstructible
    - R = R1 ►◄R2 ►◄ … ►◄ Rk

Conception des BD Réparties

Exemple

Proj 1	PNO	Budget
	P1	500000
	P2	300000
	P3	1000000
	P4	150000

Proj 2	PNO	PNom	Location
	P1	Bioinfo	Lausanne
	P2	ELearn.	Rio
	P3	Plasma	Geneva
	P4	Aircraft	Lausanne

Conception des BD Réparties

Matrice d'utilisation

- Soit la relation Projet (Pno, Pname, Budget, Loc) et soit l'ensemble de requêtes:

  q_1= budget d'un projet étant donnée son numéro.

  q_2= nom et budget de tous les projets.

  q_3= nom des projets d'une ville.

  q_4= budget total des projets d'une ville

- La matrice d'utilisation est définie comme suit:

  $Ut(q_i, A_j)$=1 si la requête q_i utilise l'attribut A_j

  $Ut(q_i, A_j)$= 0 sinon

Conception des BD Réparties

q1 = SELECT budget
FROM proj WHERE pno=value ;

q2 = SELECT pname,budget
FROM proj ;

q3 = SELECT pname
FROM proj WHERE loc=value ;

q4 = SELECT sum (budget)
FROM proj WHERE loc=value ;

Conception des BD Réparties

A1 = pno
A2 = pname
A3 = budget
A4 = loc

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

Conception des BD Réparties

Matrice d'affinité

- **$Ref_s(q_k)$** pour deux attributs **(A_i, A_j)** = nombre d'accès effectués par une exécution de **q_k** (sur le site **s**) aux attributs **A_i** et **A_j**.
- **$Acc_s(q_k)$** est la fréquence d'exécution de **q_k** sur le site **s** (mesurée pendant une certaine période).
- La matrice d'affinité est définie comme suit:

$$Aff(A_i,A_j) = \sum_{K\ tq\ Ut(qk,Ai)=1\ et\ Ut(qk,Aj)=1} \sum_{s} Ref_s(q_k) * Acc_s(q_K)$$

Conception des BD Réparties

Exemple

- Supposons : $Ref_s(\ qk\ ) = 1,\ \forall$ s, k
- Les accès suivants (3 sites):
  - *Acc1*(*q1*)=15 *Acc2*(*q1*) = 20 *Acc3*(*q1*) = 10
  - *Acc1*(*q2*)=5 *Acc2*(*q2*) = 0 *Acc3*(*q2*) = 0
  - *Acc1*(*q3*)=25 *Acc2*(*q3*) = 25 *Acc3*(*q3*) = 25
  - *Acc1*(*q4*)= 3 *Acc2*(*q4*) = 0 *Acc3*(*q4*) = 0

Conception des BD Réparties

- $Aff(A1, A3) = \sum_{K=1} \sum_{s} Acc_s(q_k)$

  $= Acc1(q1) + Acc2(q1) + Acc3(q1) = 45$

- La matrice :

	A_1	A_2	A_3	A_4
A_1	45	0	45	0
A_2	0	80	5	75
A_3	45	5	53	3
A_4	0	75	3	78

Conception des BD Réparties

Algorithme BEA (Bound Energy Algorithm)

BEA (Entrée : MA , Sortie : MAC) {
MAC(.,1)← MA(.,1) ;
MAC(.,2)← MA(.,2) ;
index ← 3 ;
n, index, loc, i, j : Entier,;
Tant que index<= n Faire {
    Pour i de 1 à index Faire
        Calculer cont(C_{i-1}, C_{index}, C_i)
    Fin Pour
    Calculer cont($C_{index-1}$, C_{index}, $C_{index+1}$)
    loc ← Max cont ;
    Pour j de index à loc [-1] Faire
        MAC(.,j)← MAC(.,j-1) ;
    Fin Pour
    MAC(.,loc)← MA(.,index) ;
    index← index + 1 }
Permuter les lignes comme les colonnes résultantes }

L'algorithme BEA est une permutation de lignes et de colonnes pour maximiser les affinités des prédicats. On fixe au début les deux premières colonnes puis on ajoute colonne par colonne en maximisant les affinités.

Les attributs sont regroupés selon leurs affinités en utilisant l'algorithme BEA qui fourni en sortie des classes d'attributs. Ces classes seront par la suite, utilisées pour générer des fragments verticaux.

Conception des BD Réparties

Algorithme BEA (Bound Energy Algorithm)

On considère les colonnes C1 et C2,

On analyse l'effet d'insérer C3 en terme de contribution (Cont).

Pour une composition de (C1, C3, C2), nous avons:

Cont(C1,C3,C2) = 2BOND(C1,C3) + 2BOND(C3,C2) – 2BOND(C1,C2)

Où BOND(Cx,Cy) = Σ i,j=1..n Aff(Ai,Ax)*Aff(Aj,Ay)

avec aff(A0,Aj)=aff(Ai,A0)=aff(An+1,Aj)=aff(Ai,An+1)=0

Voici le calcul de Cont (C0,C3,C1) :

- Cont(C0,C3,C1)=2(bond(C0,C3)+bond(C3,C1)-bond(C0,C1))
- bond(C0,C3)=aff(A1,A0)*aff(A1,A3)+aff(A2,A0)*aff(A2,A3)+ aff(A3,A0)*aff(A3,A3)+aff(A4,A0)*aff(A4,A3)= 0 + 0 + 0+ 0=0
- bond(C3,C1)=aff(A1,A3)*aff(A1,A1)+aff(A2,A3)*aff(A2,A1)+ aff(A3,A3)*aff(A3,A1)+aff(A4,A3)*aff(A4,A1)=45*45+5*0+53*45+3*0=4410
- bond(C0,C1)=0
- Cont(C0,C3,C1)=2*4410=8820

Conception des BD Réparties
Algorithme BEA (Bound Energy Algorithm)

- Cont(C1,C3,C2) = 10150
- Cont(C2,C3,C4) = 1780
- On trouve que la meilleure composition est (C1,C3,C2) qui donne le maximum d'affinité des prédicats.
- Nous permutons alors la 2ème et la 3ème colonne et nous faisons de même pour les lignes :

	A_1	A_3	A_2	A_4
A_1	45	45	0	0
A_3	45	53	5	3
A_2	0	5	80	75
A_4	0	75	3	78

Conception des BD Réparties
Algorithme BEA (Bound Energy Algorithm)

- De même pour la 4ème colonne, la contribution (C3,C2,C4) est la meilleure composition qui donne le maximum d'affinité des prédicats.
- Donc l'ordre des colonnes et des lignes est maintenu :

➢ **Regroupement des attributs ayant une haute affinité**

	A_1	A_3	A_2	A_4
A_1	45	45	0	0
A_3	45	53	5	3
A_2	0	5	80	75
A_4	0	75	3	78

Conception des BD Réparties

Partitionnement

- Consiste à trouver un point dans la matrice pour créer deux ensembles d'attributs: AttrHaut(AH) et AttrBas(AB).
- Pour savoir quelle requête accède à quel ensemble, on définit:
  - **$AR(q_i)=\{A_j/Ut(q_i, A_j)=1\}$**
  - **$RH= \{q_i/AR(q_i)\subseteq AH\}$**
  - **$RB= \{q_i/AR(q_i)\subseteq AB\}$**
  - **$Autres = Q- \{RH \cup RB\}$**
- D'après notre exemple, on obtient:
  - **$Q =\{q_1, q_2, q_3, q_4\}$**
  - **$AH=\{A_1, A_3\}$, $AB=\{A_2, A_4\}$**
  - **$RH= \{q_1\}$, $RB= \{q_3\}$**
  - **$Autres= \{q_2, q_4\}$**