Conception des BD réparties

Conception de BDR et requêtes

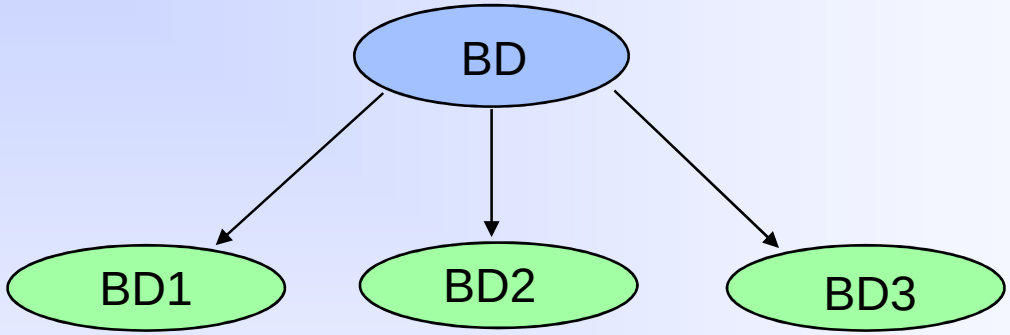
Approche décomposition

Fragmentation

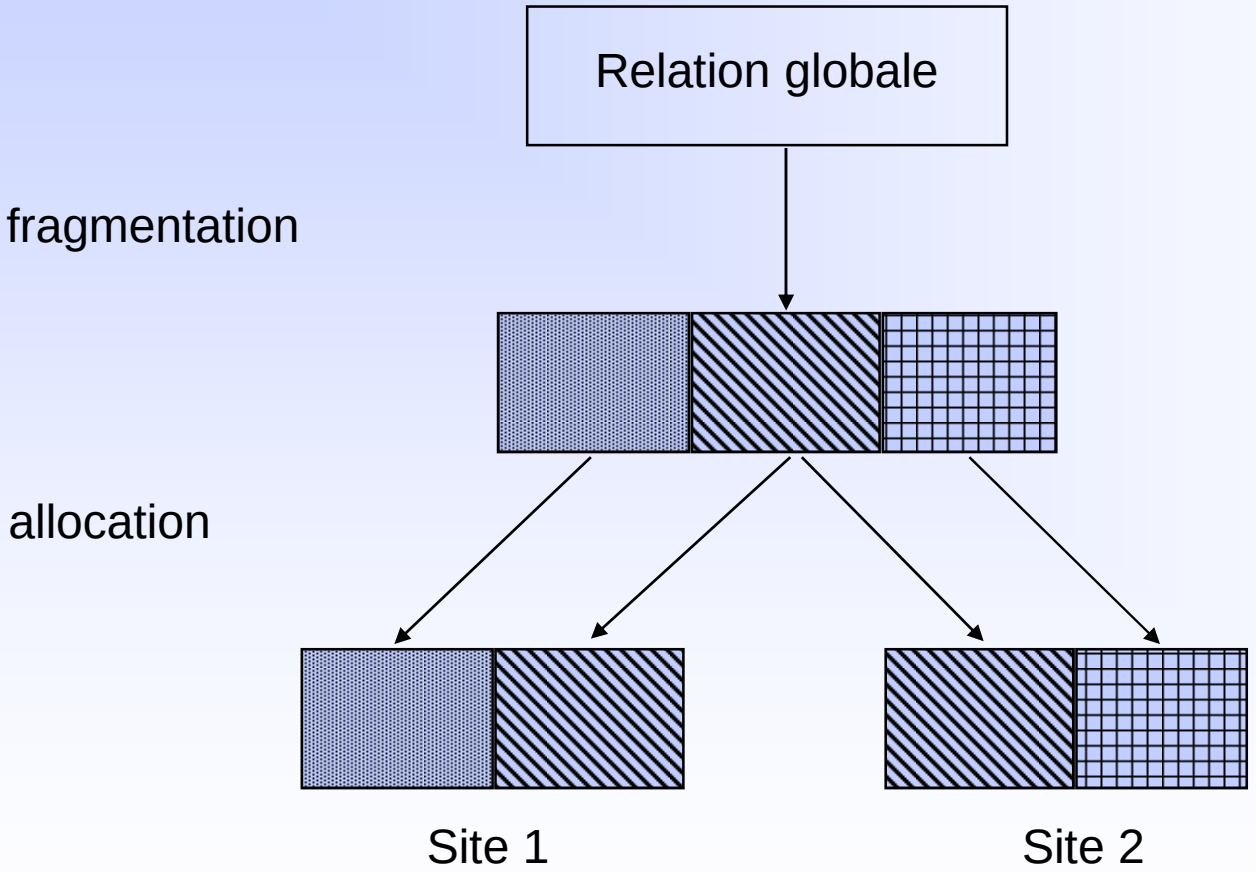
Allocation des fragments

Migration vers une BDR

Décomposition en BD locales



Conception d'une BDR par Décomposition



Objectifs de la Décomposition

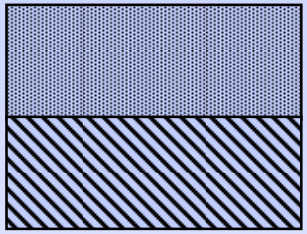
Fragmentation

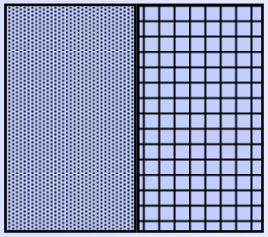
- deux types : horizontale, verticale,
  - possibilité de combiner des fragmentations
- performances en favorisant les accès locaux
- équilibrer la charge de travail entre les sites

Duplication (réplication)

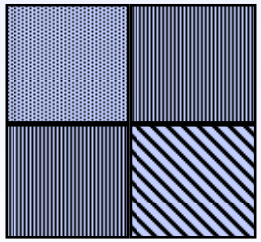
- favoriser les accès locaux
- augmenter la disponibilité des données

Types de Fragmentation





horizontale **verticale**



Mixte = horizontal + vertical

Fragmentation correcte

Complète

- chaque élément de R doit se trouver dans un fragment

Reconstructible

- on doit pouvoir recomposer R à partir de ses fragments

Disjointe

- chaque élément de R ne doit pas être dupliqué

Fragmentation Horizontale

Fragmentation horizontale de R sur *m* fragments

tuples de R: $\{t_1, \ldots, t_n\}$, fragments: $\{R_1, \ldots, R_m\}$

- **Fragmentation par round-robin**
  - **Le tuple $t_i \in R_j$ avec $j = i \bmod m$**
  - **☝ lecture séquentielle**
- **Fragmentation par hachage sur l'attribut *A***
  - **$t_i \in R_j$ avec $j = hash(t_i.A)$**
  - **☝ sélection (A=v), équi-jointure (A=B)**
- **Fragmentation par intervalle**
  - **Fragmenter le domaine de l'attribut A en m intervalles**
  - **Vecteur $\{a_1, \ldots, a_{m-1}\}$**
  - **$t_i \in R_j$ avec $a_{j-1} \leq T_i.A < a_j$**
  - **☝ sélection («A between x and y»)**

Fragmentation Horizontale par sélection

Fragments définis par sélection

create table *Client1* as
select * from Client where ville = ‘Paris’

create table *Client2* as
select * from Client where ville <> ‘Paris’

Reconstruction

create view *Client* as
select * from *Client1*
union
select * from *Client2*;

Client

nclient	nom	ville
C 1 C 2 **C 3** C 4	Dupont Martin Martin Smith	Paris Lyon Paris Lille

Client1

nclient	nom	ville
C 1 C 3	Dupont Martin	Paris Paris

Client2

nclient	nom	ville
C 2 C 4	Martin Smith	Lyon Lille

Fragmentation Horizontale par sélection

- **Fragmentation de R selon *n* prédicats**
  - **les prédicats $\{p_1, \ldots, p_n\}$ ex: {a<10, a > 5, b='x', b='y'}**
- **L'ensemble *M* des prédicats de fragmentation est :**
  - **$M = \{ m \mid m = \bigwedge_{1 \leq k \leq n} p_k^* \}$ avec $p_k^* \in \{p_k, \neg p_k\}$**
  - **Eliminer les *m* de sélectivité nulle ex: $a>10 \wedge a < 5$**
  - **Simplifier: $a<10 \wedge a \leq 5 \wedge b='x' \wedge b \neq 'y'$ devient $a \leq 5 \wedge b='x'$**
- **Construire les fragments $\{R_1, \ldots, R_k\}$**
  - **Pour chaque m_i , $R_i = \sigma m_i (R)$**
- **Minimalité**
  - **Ne pas avoir 2 fragments toujours lus ensemble**
- **Choisir les p_i des requêtes les plus fréquentes**

Fragmentation Horizontale Dérivée

Fragments définis par semi-jointure

create table *Cde1* as
select * from Cde, Client1
where Cde.nclient = Client1.nclient

create table *Cde2* as
select * from Cde, Client2
where Cde.nclient = Client2.nclient

Reconstruction

Cde = Cde1 union Cde2

Cde

ncde	nclient	produit	qté
D 1	C 1	P 1	10
D 2	C 1	P 2	20
D 3	C 2	P 3	5
D 4	C 4	P 4	10

Cde1

ncde	nclient	produit	qté
D 1	C 1	P 1	10
D 2	C 1	P 2	20

Cde2

ncde	nclient	produit	qté
D 3	C 2	P 3	5
D 4	C 4	P 4	10

Propriétés de la fragmentation horizontale dérivée

R: fragmentation horizontale → fragments R_i

S: fragmentation horizontale dérivée → fragments $S_i = S \ltimes_A R_i$

Complète

- **Chaque tuple de S doit joindre avec au moins un tuple de R**
  - $\forall s \in S, \exists t \in S_i, s = t$

Disjointe

- $\forall i,j \text{ tq } i \neq j , S_i \cap S_j = (S \ltimes R_i) \cap (S \ltimes R_j) = S \ltimes (R_i \cap R_j) = \varnothing$
  - **Rappel: R1 ∩ R2 ⇔ R1 ⋉ R2**

Reconstructible

- $\cup_i S_i = (S \ltimes R_1) \cup (S \ltimes R_2) \cup \ldots \cup (S \ltimes R_n) = S \ltimes (\cup_i R_i) = S$

⇒ contrainte d'intégrité référentielle

- **A = clé de R**
- **S.A référence R.A**
  - $\forall s \in S, \exists r \in R, s.A=y.A$

Fragmentation Verticale

Fragments définis par projection

Cde1 = Cde (ncde, nclient)

Cde2 = Cde (ncde, produit, qté)

Reconstruction

Cde = [ncde, nclient, produit, qté]

where Cde1.ncde = Cde2.ncde

Cde

ncde	nclient	produit	qté
D 1	C 1	P 1	10
D 2	C 1	P 2	20
D 3	C 2	P 3	5
D 4	C 4	P 4	10

Cde1

ncde	nclient
D 1	C 1
D 2	C 1
D 3	C 2
D 4	C 4

Cde2

ncde	produit	qté
D 1	P 1	10
D 2	P 2	20
D 3	P 3	5
D 4	P 4	10

Propriétés de la fragmentation verticale

Fragmentation verticale de R

- $A_i \subseteq$ attributs(R)
- Fragments: Ri = π_{Ai}(R)

Complète

- Attributs(R) = $A_1 \cup A_2 \cup \ldots \cup A_n$

Disjointe

- $\forall i,j$ tq $i \neq j$, $A_i \cap A_j$ = clé(R)

Reconstructible

- R = R1 $\bowtie$ R2 $\bowtie$ … $\bowtie$ Rk

Matrice d’affinité des attributs

Matrice A

aij = affinité de Ai avec Aj

ex: nb de requêtes qui accèdent Ai et Aj

	A1	A2	A3	A4
A1	45	0	45	0
A2		80	5	75
A3			53	3
A4				78

Matrice d'affinité regroupement des attributs

Matrice A

	A1	A3	A2	A4
A1	45	45	0	0
A3		53	5	3
A2			80	75
A4				78

Allocation des Fragments aux Sites

Non-dupliquée

- **partitionnée : chaque fragment réside sur un seul site**

Dupliquée

- **chaque fragment sur un ou plusieurs sites**
- **maintien de la cohérence des copies multiples**
- Règle **:**
  - **si le ratio Lectures/màj est > 1, la duplication est avantageuse**

Allocation de Fragments

Problème: Soit

***F* un ensemble de fragments**

***S* un ensemble de sites**

***Q* un ensemble d'applications et leurs caractéristiques**

trouver la distribution "optimale" de *F* sur *S*

Optimum

- **coût minimal de comm, stockage et traitement**
- **Performance = temps de réponse ou débit**

Solution

- **allouer une copie de fragment là où le bénéfice est supérieur au coût**

Exemple d'Allocation de Fragments

Client1

nclient	nom	ville
C 1 C 3	Dupont Martin	Paris Paris

Client2

nclient	nom	ville
C 2 C 4	Martin Smith	Lyon Lille

Cde1

ncde	client	produit	qté
D 1 D 2	C 1 C 1	P 1 P 2	10 20

Site 1

Cde2

ncde	client	produit	qté
D 3 D 4	C 2 C 4	P 3 P 4	5 10

Site 2

Transparence des requêtes

Transparence de la répartition : degré d'intégration du schéma

- **Requête indépendante de la fragmentation : haut niveau de transparence**
- **bas niveau de transparence : l'utilisateur doit spécifier les fragments qu'il manipule => pb de nommage non ambigu.**

Le système doit optimiser les lectures/écritures,

- **effectuer automatiquement les mises à jour des répliques,**
- **optimiser les requêtes.**

Conclusion

Fragmentation

- Nécessaire pour le passage à l ’échelle
- Paralléliser les requêtes accédant à des fragments distincts.

Limites

- Problème des données fortement reliées les unes aux autres
- Gérer les données partagées par les requêtes concurrentes.