

Conception de BDR et requêtes

Approche décomposition

Fragmentation

Allocation des fragments

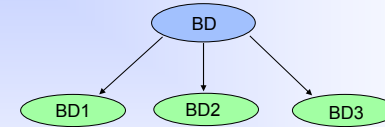
Fragmentation de requêtes

Optimisation de requêtes

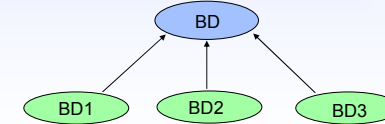
1

Migration vers une BDR

Décomposition en BD locales

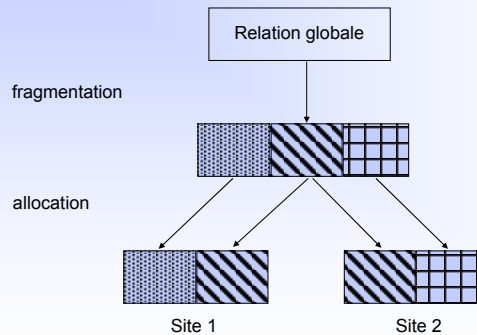


Intégration logique des BD locales existantes



2

Conception d'une BDR par Décomposition



3

Objectifs de la Décomposition

Fragmentation

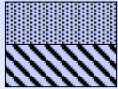
- trois types : horizontale, verticale, mixte
- performances en favorisant les accès locaux
- équilibrer la charge de travail entre les sites

Duplication (réplication)

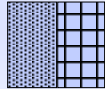
- favoriser les accès locaux
- augmenter la disponibilité des données

4

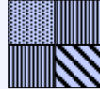
Types de Fragmentation



horizontale



verticale



mixte

5

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

6

Fragmentation Horizontale

Fragmentation horizontale de R sur m fragments

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

Fragmentation par round-robin

- Le tuple $t_i \in R_j$ avec $j = i \text{ mod } m$
- lecture séquentielle

Fragmentation par hachage sur l'attribut A

- $t_i \in R_j$ avec $j = \text{hash}(t_i.A)$
- sélection ($A=v$), équi-jointure ($A=B$)

Fragmentation par intervalle

- Fragmenter le domaine de l'attribut A en m intervalles
- Vecteur $\{a_1, \dots, a_{m-1}\}$
- $t_i \in R_j$ avec $a_{j-1} \leq T_i.A < a_j$
- sélection («A between x and y»)

7

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	Dupont	Paris
C 2	Martin	Lyon
C 3	Martin	Paris
C 4	Smith	Lille

Client1

nclient	nom	ville
C 1	Dupont	Paris
C 3	Martin	Paris

Client2

nclient	nom	ville
C 2	Martin	Lyon
C 4	Smith	Lille

8

Fragmentation Horizontale par sélection

- Fragmentation de R selon n prédicats
 - les prédicats $\{p_1, \dots, p_n\}$ 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 $\{R_1, \dots, R_k\}$
 - 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

9

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

10

Propriétés de la fragmentation horizontale dérivée

R: fragmentation horizontale

S: fragmentation horizontale dérivée

Complète

- Chaque tuple de S doit joindre avec au moins un tuple de R

Disjointe

- $\forall i, j, (S \triangleright R_i) \cap (S \triangleright R_j) = \emptyset$

Reconstructible

- $S = (S \triangleright R_1) \cup (S \triangleright R_2) \cup \dots \cup (S \triangleright R_n)$

⇒ contrainte d'intégrité référentielle

- A = clé de R
- S.A référence R.A

11

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

12

Propriétés de la fragmentation verticale

Fragmentation verticale de R

- $A_i \subseteq \text{attributs}(R)$
- Fragments: $R_i = \pi_{A_i}(R)$

Complète

- $\text{Attributs}(R) = A_1 \cup A_2 \cup \dots \cup A_n$

Disjointe

- $A_1 \cap A_2 \cap \dots \cap A_n = \text{clé}(R)$

Reconstructible

- $R = R_1 \bowtie R_2 \bowtie \dots \bowtie R_k$

13

Matrice d'affinité des attributs

Matrice A

a_{ij} = affinité de A_i avec A_j

ex: nb de requêtes qui accèdent A_i et A_j

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

14

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

15

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

16

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

17

Exemple d'Allocation de Fragments

Client1

nclient	nom	ville
C 1	Dupont	Paris
C 3	Martin	Paris

Client2

nclient	nom	ville
C 2	Martin	Lyon
C 4	Smith	Lille

Cde1

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

Site 1

Cde2

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

Site 2

18

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.

19

Transparence des requêtes

Schéma de fragmentation

- contient la définition des fragments

Schéma d'allocation

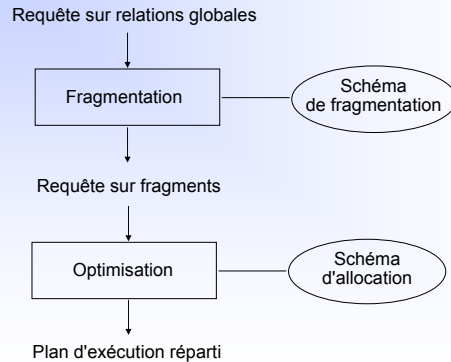
- contient la localisation de chaque réplique de fragment

Où se trouve le schéma de répartition ?

1. Chaque site a une copie de tout le schéma :
 - avantage : lectures rapides
 - inconvénient : modifications de schémas
2. Chaque site a un schéma local
 - il faut retrouver les informations sur les autres sites
3. Solutions mixtes : copie partielle du schéma sur certains sites

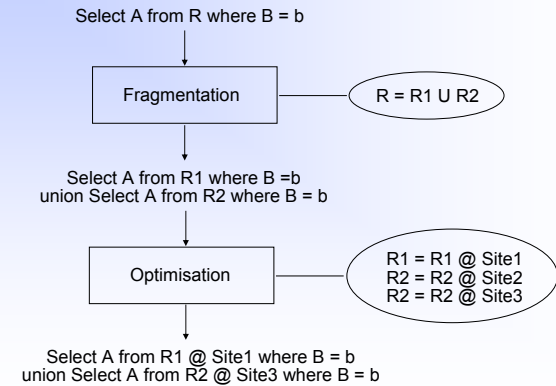
20

Evaluation de Requêtes Réparties



21

Exemple d'Evaluation Simple



22

Fragmentation

Réécriture

- mettre la requête sous forme d'un arbre algébrique (feuille = relation, noeud = op. relationnel)

Reconstruction

- remplacer chaque feuille par le programme de reconstruction de la relation globale

Transformation

- appliquer des techniques de réduction pour éliminer les opérations inutiles

Notations utilisées :

- S : select
- J : join
- P : project

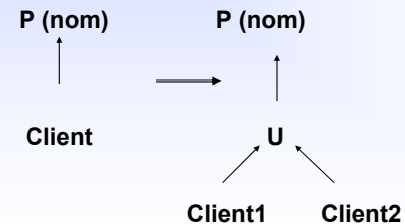
23

Reconstruction

Select nom from Client

Client1 : Client where ville = 'Paris'

Client2 : Client where ville not 'Paris'



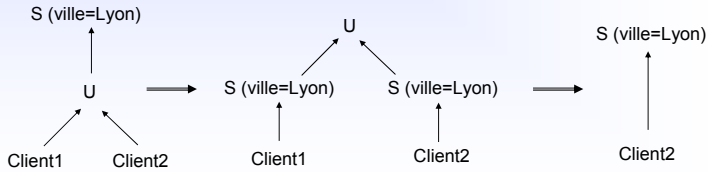
24

Réduction pour la fragmentation horizontale

Règle : éliminer l'accès aux fragments inutiles

Exemple :

Client1 : Client where ville = 'Paris'
Client2 : Client where ville not 'Paris'
Select * from Client where ville = 'Lyon'



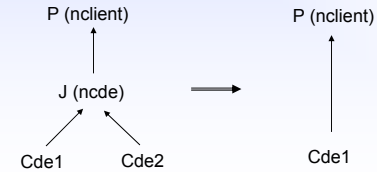
25

Réduction pour la Fragmentation Verticale

Règle : éliminer les accès aux relations de base qui n'ont pas d'attributs utiles pour le résultat final

Exemple

Cde1 : Cde (ncode, nclient)
Cde2 : Cde (ncode, produit, qté)
Select nclient from Cde



26

Réduction pour la Fragn. Hor. Dérivée

Règle : distribuer les jointures par rapport aux unions et appliquer les réductions pour la fragmentation horizontale

Exemple

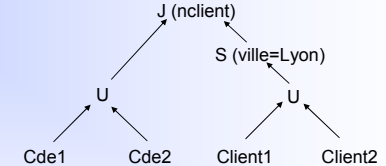
Client1 : Client where ville= 'Paris'
Client2 : Client where ville not 'Paris'
Cde1 : Cde where Cde.nclient=Client1.nclient
Cde2 : Cde where Cde.nclient=Client2.nclient

Select * from Client, Cde
where Client.nclient = Cde.nclient
and ville= 'Lyon'

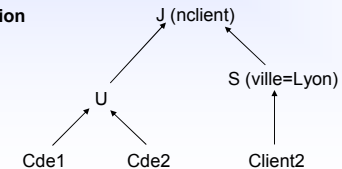
27

Exemple

Requête canonique



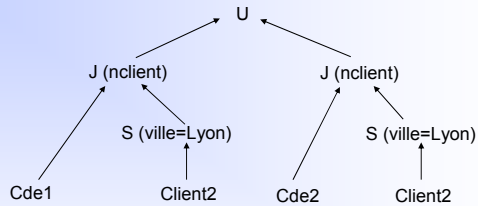
Après 1ère réduction



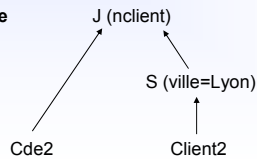
28

Exemple (suite)

Distribution



Elimination du sous-arbre inutile



29

Optimisation de Requêtes Réparties

entrée : une requête simplifiée exprimée sur des fragments

sortie : un plan d'exécution réparti optimal

Objectifs

- choisir la meilleure localisation des fragments
- minimiser une fonction de coût: $f(I/O, CPU, Comm.)$
- exploiter le parallélisme
- exprimer les transferts inter-sites

Solution

- examiner le coût de chaque plan possible (par transformation) et prendre le meilleur

30

Exemple

Site 1 : Client1 = Client where ville= 'Paris'

Site 2 : Client2 = Client where ville not 'Paris'

Site 3 : Cde1 = Cde where Cde.nclient=Client1.nclient

Site 4 : Cde2 = Cde where Cde.nclient=Client2.nclient

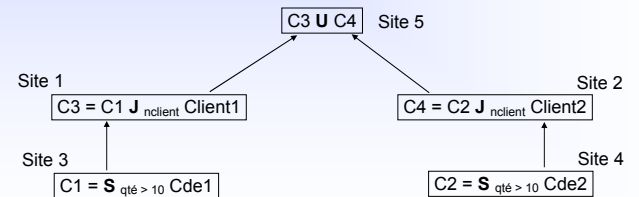
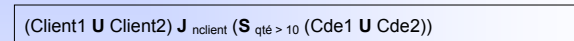
Site 5 : résultat

Select * from Client, Cde
 where Client.nclient = Cde.nclient
 and qté > 10

31

Choix de la Solution

Site 5



32

Coût des Solutions

Supposons

- taille (Cde1) = taille (Cde2) = 10 000
- taille (Client1) = taille (Client2) = 2 000
- coût de transfert d'un n-uplet = 1
- Sélectivité($qté > 10$) = 1%

Stratégie 1

- transfert de Cde1 + Cde2 = 20 000
- transfert de Client1 + Client2 = 4000

Stratégie 2

- transfert de C1 + C2 = 1%*10000 + 1%* 10000 = 200
- transfert de C3 + C4 = 200

33

Jointure

R sur S1, S sur S2, T sur S3.

Requête demandée sur le site S0 : $R \bowtie S \bowtie T$

Plusieurs possibilités :

- Copier tout sur S0 et faire les jointures sur S0
- Copier R sur S2, et joindre R et S sur S2
Copier le résultat sur S3, et faire la jointure avec T sur S3
Copier le résultat sur S0

Tenir compte des index, de la taille des relations à transférer, de la taille des relations intermédiaires, de la charge des sites, de la vitesse de transmission, etc.

On peut paralléliser les jointures : grand nombre de stratégies

34

Semi-jointure

Traiter la jointure entre deux relations réparties sur 2 sites:

R1 sur S1, R2 sur S2 Rappel: $R1 \bowtie R2 = \Pi_{R1}(R1 \bowtie R2)$

Requête $R1 \bowtie R2 = R1 \bowtie (R2 \bowtie R1)$ et résultat sur S1

- T1 = $\Pi_A(R1)$ sur S1, puis envoi de T1 sur S2
- T2 = $R2 \bowtie T1$ sur S2, puis envoi de T2 sur S1
- Calcul de $R1 \bowtie T2$ sur S1

Requête $R1 \bowtie R2 = (R1 \bowtie R2) \bowtie (R2 \bowtie R1)$ et résultat sur S0

- Transférer T1 = $\Pi_A(R1)$ de S1 → S2
- Transférer T2 = $\Pi_A(R2)$ de S2 → S1
- Transférer T3 = $R1 \bowtie R2$ de S1 → S0
- Transférer T4 = $R2 \bowtie R1$ de S2 → S0
- Sur S0, T3 \bowtie T4

35

Semi jointure: réduction des transferts

Réduire la taille des transferts inter site

T1 = $\Pi_A(R)$ à transférer de S1 vers S2

Vecteur de bits $V = \{b1, \dots, bM\}$,

Domaine de l'attribut R.A: $\{a1, \dots, a_n\}$,

$V_i = 1$ si $a_i \in \Pi_A(R)$, sinon $V_i = 0$

Limite: si le domaine est grand ($n \gg 1$), taille vecteur

Bloom filter : vecteur de taille fixe (=M)

- Vecteur $V = \{b1, \dots, bM\}$,
- fonction de hachage $h() : \{a1, \dots, a_n\} \rightarrow \{1, M\}$
- $V_j = 1$ si $h(a_i) = j$ et $a_i \in \Pi_A(R)$, sinon $V_i = 0$
- Inconvénient: $a \in \Pi_A(R)$, $b \notin \Pi_A(R)$, $h(a) = h(b)$
 - Les tuples de R tq R.A=b sont transférés inutilement.

36

Conclusion

Importance de la fragmentation horizontale pour augmenter le parallélisme inter- et intra-requête

Utiliser la fragmentation verticale si forte affinité d'attributs et beaucoup d'attributs

L'optimisation reposant sur un modèle de coût est critique s'il y a beaucoup de sites