Bases de données réparties

J. Akoka - I. Wattiau

Contexte

- Technologique :
  - des solutions de communication efficace entre les machines
  - des SGBD assurent la transparence des données réparties
  - standardisation :SGBDR, SQL, middlewares,...

- Organisationnel :
  - décentralisation des activités
  - subsidiarité
- Financier :
  - « downsizing » des matériels et logiciels

SGBD distribué

- permet la création, l'accès et la manipulation de données inter-reliées, localisées sur différents sites d'un réseau informatique
- chaque site a une capacité de traitement local autonome
- chaque site participe à l'exécution d'au moins une application globale via le réseau

Intérêts et conséquences

- Améliorer l'accessibilité, le partage, la disponibilité et la performance des SGBD
- Tout en préservant les avantages d'une base de données centralisée
- L'optimisation globale des traitements va dépendre :
  - du coût de communication
  - du coût des traitements locaux
  - de la stratégie d'allocation des données
  - de la stratégie d'exécution des traitements

12 règles de Date

- Autonomie locale
- Pas de dépendance d'un site central
- Opère en continu
- Indépendance vis-à-vis de la localisation
- Indépendance vis-à-vis de la fragmentation
- Indépendance vis-à-vis de la réplication
- Traitement de requêtes distribuées
- Traitement de transactions distribuées
- 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

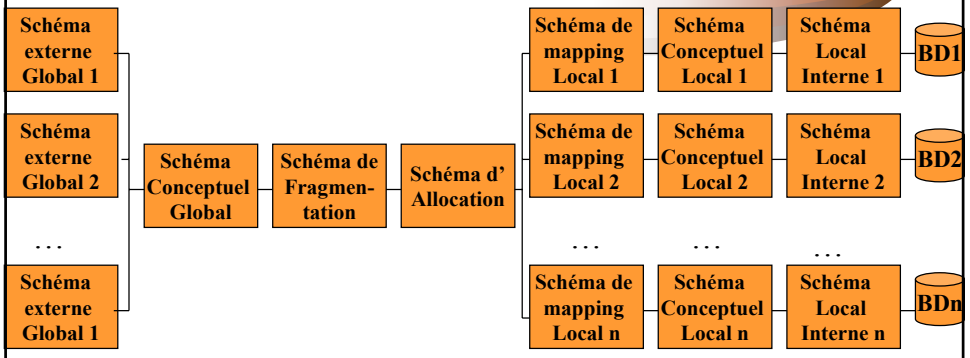
Typologie des BD distribuées

- Homogènes versus hétérogènes
- Semi-autonomes versus autonomes

Bases de données fédérées

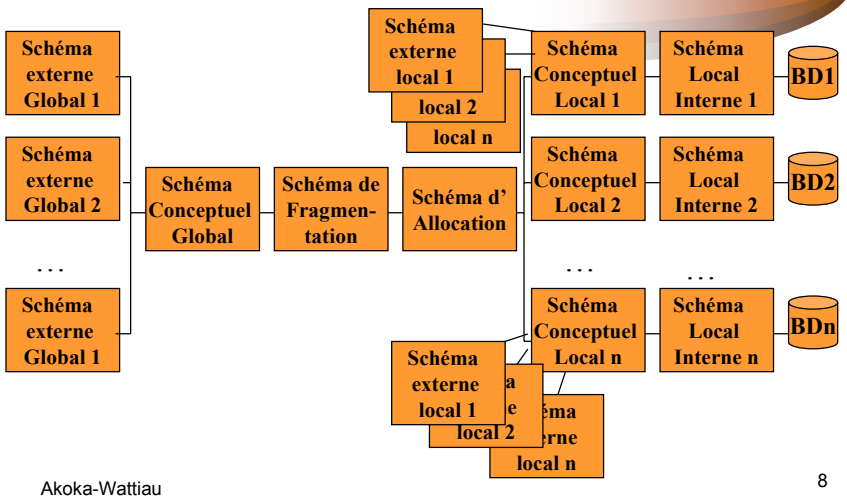
Systèmes multibases

Architecture d'un SGBD distribué



ANSI-SPARC

Architecture d'un SGBD fédéré



Objectifs de la conception répartie

- Contrôle de la redondance des données
- Indépendance des BD locales
- Optimisation globale
- Combinaison de la fragmentation et de l'allocation

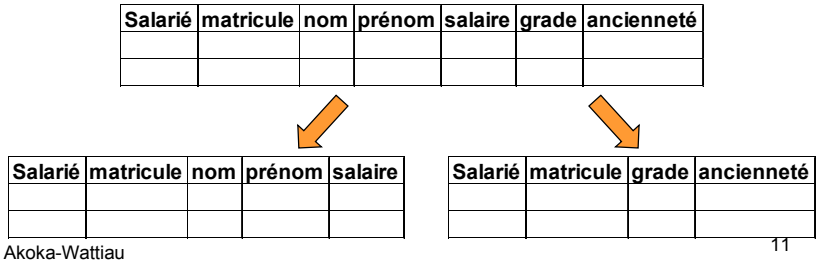

logique physique

Fragmentation

- Une table T est partitionnée en un nombre minimum de sous-tables disjointes $T_1, T_2, .., T_n$
- Chaque fragment T_i doit contenir assez d 'information pour reconstruire T
- Fragmentation verticale vs horizontale

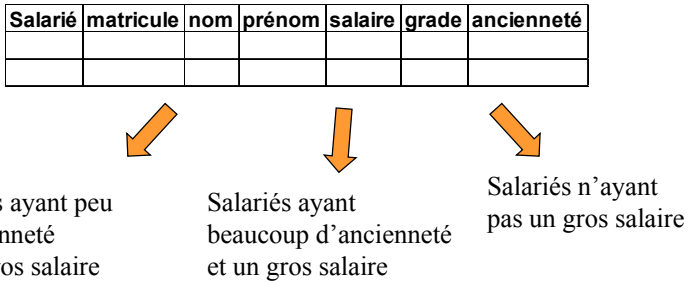
Fragmentation verticale

- Subdivision des attributs de T en groupes (en dupliquant une clé commune) et projection sur les groupes
- Réversible par jointure



Fragmentation horizontale

- Sélection selon un prédicat de qualification
- Réversible par union



Fragmentation (suite)

- Fragmentation mixte : horizontale et verticale
- Fragment le plus grand = une table
- Fragment le plus petit = un tuple
  - > trop de sensibilité aux mises à jour et trop de jointures

Allocation (1/2)

- Centralisée
  - taille base limitée par les capacités du site central
  - disponibilité faible
- Partitionnée
  - limite l'espace disque local
  - meilleur si la fiabilité de la solution centralisée n'est pas suffisante
  - intéressant s'il y a une partie importante de traitement local

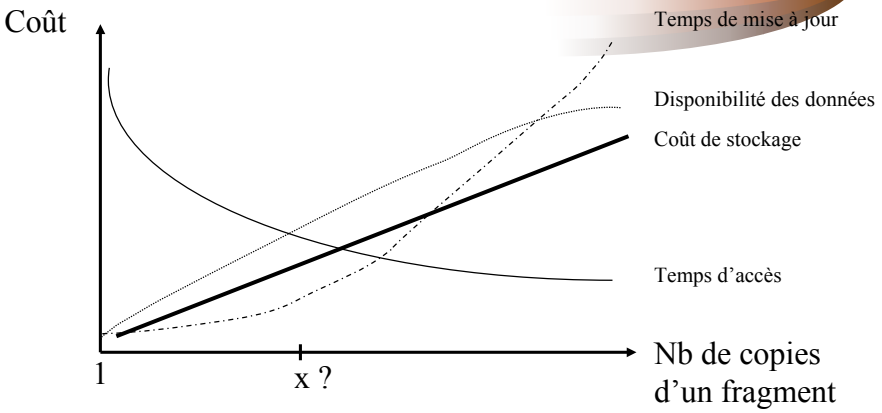
Allocation (2/2)

- Répliquée
  - une copie de la base sur chaque site
  - maximise la fiabilité
  - possible si la base est petite et si l'inefficacité des mises à jour est tolérable
- Réplication sélective
  - les fragments critiques sont répliqués
  - les fragments non critiques sont localisés sur un seul site
  - grande flexibilité

Réplication

- Synchrone : mise à jour immédiate, protocole de validation à deux phases
- Asynchrone : mise à jour différée
- Mécanismes de réplication :
  - Maître – esclave
  - « workflow »
  - symétrique

Estimation des coûts de la réplication



Min (coût_stockage + coût_communic + coût_traitlocal)
sous contraintes de temps de réponse et de disponibilité

Méthode « best fit »

- Permet de déterminer le site sur lequel stocker le fragment, là où l'on maximise les traitements locaux
- n'intègre pas le problème de la réplication

Exemple

Fragments	temps d'accès local (ms)	temps mise à jour local	temps accès distant	temps m.a.j. distant
R1	100	150	500	600
R2	150	200	650	700
R3	200	250	1000	1100

Transactions	sites	fréquence	accès tables (L= lecture, E = écriture)
T1	S1,S4,S5	1	4 accès R1 (3L, 1E) 2 accès R2 (2L)
T2	S2,S4	2	2 accès R1 (2L) 4 accès R3 (3L, 1E)
T3	S3,S5	3	4 accès R2 (3L, 1E) 2 accès R3 (2L)

Exemple Application Best fit

Fragment	Site	Accès T1 * Fréquence	Accès T2 * Fréquence	Accès T3 * Fréquence	Total pondéré
R1	S1	4*1	0	0	4
	S2	0	2*2	0	4
	S3	0	0	0	0
	S4	4*1	2*2	0	8
	S5	4*1	0	0	4
R2	S1	2*1	0	0	2
	S2	0	0	0	0
	S3	0	0	4*3	12
	S4	2*1	0	0	2
	S5	2*1	0	4*3	14
R3	S1	0	0	0	0
	S2	0	4*2	0	8
	S3	0	0	2*3	6
	S4	0	4*2	0	8
	S5	0	0	2*3	6

On stocke R1 sur S4

On stocke R2 sur S5

On stocke R3 sur S2 ou S4 (S2 pour mieux répartir)

Méthode Best Fit - Remarques

- Calculs simples
- n'intègre pas la réplication
- nb de références < > temps d'accès

Méthode ABS

- « All Beneficial Sites »
- Principe : stocker un fragment sur tous les sites où le bénéfice du stockage est supérieur à son coût
- bénéfice = (tps trait distant - tps trait local) * fréquence
- coût stockage copie = coût de la mise à jour locale + coût des mises à jour distantes

Exemple - Application ABS

Fragment	Site	M.a.j. locales	M.a.j. distantes	Coût m.a.j.	Interrogations	Interr. locales	Interr. distantes	bénéfice copie	bénéfice - coût
R1	S1	T1	T1 de S4 et S5	1*1*150+2*1*1*600 = 1350	T1 de S1	3*1*100	3*1*500	1200	<0
	S2		T1 de S1,S4 et S5	3*1*1*600 = 1800	T2 de S2	2*2*100	2*2*500	1600	<0
	S3		T1 de S1,S4 et S5	3*1*1*600 = 1800				0	<0
	S4	T1	T1 de S1 et S5	1*1*150+2*1*1*600 = 1350	T1 et T2 de S4	100+2*2	1*500+ 2*2*5	2800	>0
	S5	T1	T1 de S1 et S4	1*1*150+2*1*1*600 = 1350	T1 de S5	3*1*100	3*1*500	1200	<0
R2	S1		T3 de S3 et S5	2*1*3*700 = 4200	T1 de S1	2*1*150	2*1*650	1000	<0
	S2		T3 de S3 et S5	2*1*3*700 = 4200				0	<0
	S3	T3	T3 de S5	1*1*3*200+1*1*3*700 = 2700	T3 de S3	3*3*150	3*3*650	4500	>0
	S4		T3 de S3 et S5	2*1*3*700 = 4200	T1 de S4	2*1*150	2*1*650	1000	<0
	S5	T3	T3 de S3	1*1*3*200+1*1*3*700 = 2700	T1 et T3 de S5	150 + 2*1	3*650+2*1*6	5500	>0
R3	S1		T2 de S2 et S4	2*1*2*1100 = 4400				0	<0
	S2	T2	T2 de S4	1*1*2*250 + 1*1*2*1100 = 2700	T2 de S2	2*3*200	2*3*1000	4800	>0
	S3		T2 de S2 et S4	2*1*2*1100 = 4400	T3 de S3	3*2*200	3*2*1000	4800	>0
	S4	T2	T2 de S2	1*1*2*250 + 1*1*2*1100 = 2700	T2 de S4	2*3*200	2*3*1000	4800	>0
	S5		T2 de S2 et S4	2*1*2*1100 = 4400	T3 de S5	3*2*200	3*2*1000	4800	>0

=> R1 sur S4, R2 sur S3 et S5, R3 sur S2, S3, S4 et S5

Méthode ABS - Remarques

- Si coût = bénéfice, étudier l'évolution dans le temps
- Si coût > bénéfice sur tous les sites, choisir le site qui minimise la différence

Méthode d'allocation progressive

- Alloue la première copie d'un fragment au site qui maximise B - C
- puis, calcule B - C connaissant cette première copie :
  - C ne change pas
  - B peut changer car le temps d 'accès distant peut diminuer

Exemple - allocation progressive

- 2 fragments F1 et F2 - 2 sites S1 et S2

	C	B
F1 sur S1	150	200
F1 sur S2	170	175
F2 sur S1	60	30
F2 sur S2	50	45

- ABS : F1 a deux copies sur S1 et S2, F2 est sur S2
- Allocation progressive : F1 sur S2 et F2 sur S2
- si, après cette première allocation on a :

	C	B
F1 sur S2	170	165

=> on ne fait pas de copie de F1 sur S2

Remarques sur les méthodes ABS et allocation progressive

- ne tiennent pas compte de la topologie du réseau
- difficulté à estimer les temps moyens
- parfois on se concentre sur les transactions critiques et non sur l'ensemble des transactions

Niveaux de transparence

- Transparence de fragmentation
- Transparence d'allocation
- Transparence de langage
- Pas de transparence

Transparence (exemple)

- Fournisseur(num, nom, ville)
- Fragmentation horizontale selon la ville (Paris ou Marseille) fparis et fmarseille
- Le fragment de Marseille est dupliqué (Marseille1 ou Marseille2) fmarseille1 et fmarseille2
- Recherche d'un fournisseur à partir de son numéro

Niveau de transparence

Transparence de fragmentation :

```
procedure requete1(:num,:nom);
select nom into :nom
from fournisseur
where num=:num;
```

Transparence d'allocation :

```
procedure requete2(:num,:nom);
select nom into :nom
from fparis
where num=:num;
if :empty then
select nom into :nom
from fmarseille;
end procedure;
```

Transparence de langage :

```
procedure requete3(:num,:nom);
select nom into :nom
from fparis
where num=:num;
if :empty then
select nom into :nom
from fmarseille1;
end procedure;
```

Le programmeur choisit la copie à utiliser

Distribution dans Oracle

- Ne gère pas la fragmentation
- Assure la connectivité avec des bases Oracle et non Oracle
- Permet la définition d'une base de données globale (un nom)
- Permet les transactions à distance, les transactions distribuées (PL/SQL)
- Pas d'intégrité référentielle distribuée

Réplication dans Oracle

- « Snapshots » en lecture : une table maître recopiée et mise à jour à la demande de l'esclave
- « Snapshots » modifiables : sont modifiables en direct aussi
- Réplication multi-maître « poussée » par le maître
- Réplication procédurale : par package