

A thick black L-shaped frame is positioned around the text. It starts with a vertical line on the left, a horizontal line at the top, and another vertical line on the right, with a horizontal line at the bottom. The text is centered within the frame.

# CHAPITRE 2 : BASES DE DONNÉES REPARTIES

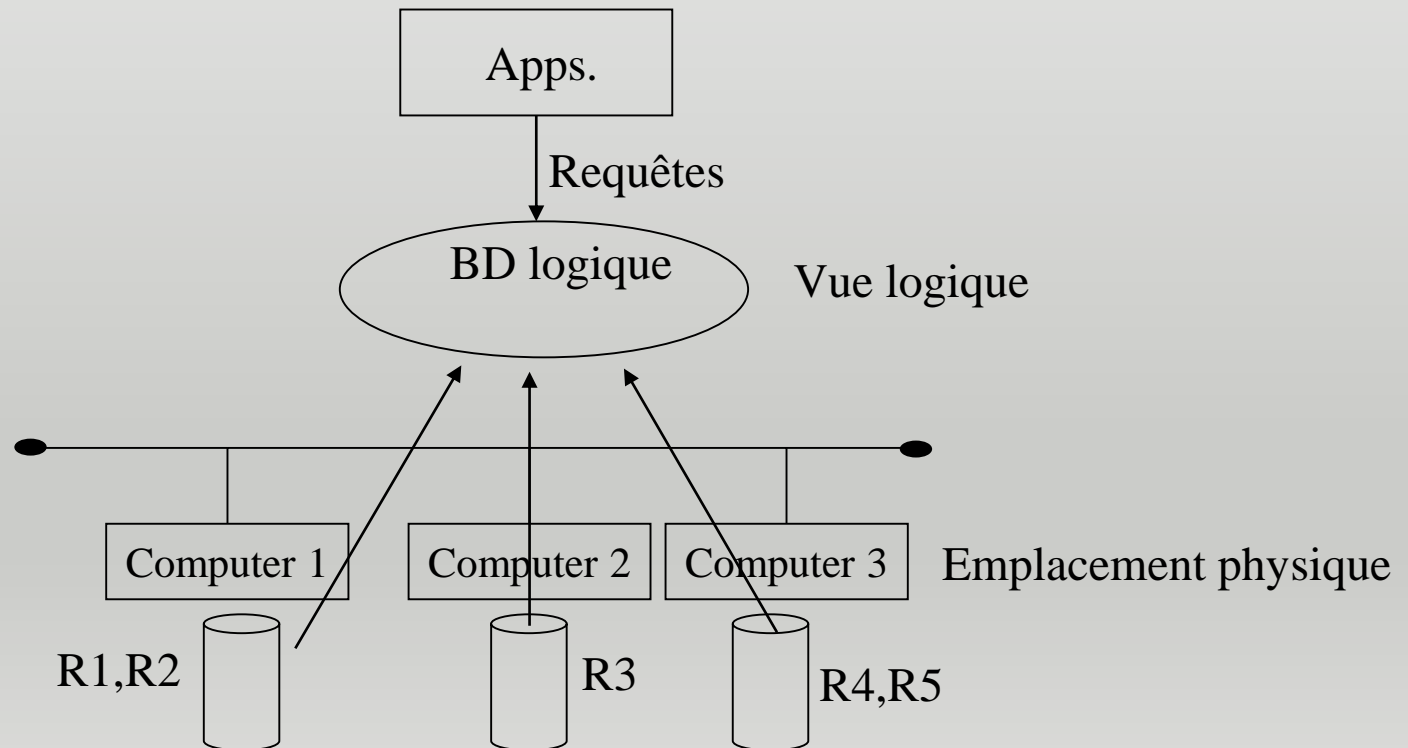
# INTRODUCTION

# Définition

- Base de données réparties?
  - *Est une BD stockée sur plusieurs sites(machine +BD locale) connectés par un réseau.*
    - Logiquement reliées
    - Physiquement distribuées

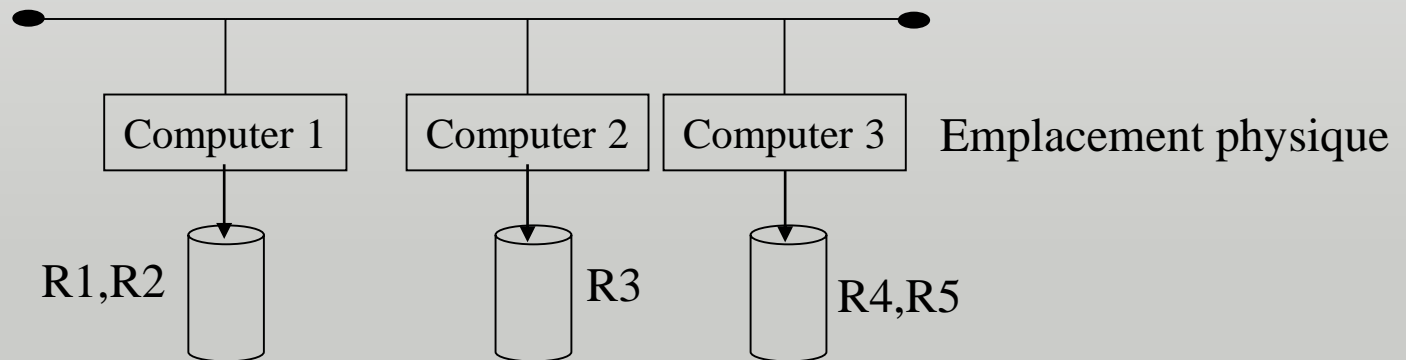
# Logiquement reliées

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



# Physiquement distribuées

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



# Problématique

- Les pressions pour la distribution :
  - *Il devient impératif de décentraliser l'information (cas des multinationales),*
  - *Augmentation du volume de l'information (14 fois de 1990 à 2000),*
  - *Augmentation du volume des transactions (10 fois dans les 5 prochaines années).*

- Pour améliorer le débit des E/Ss :
  - *Partitionnement des données,*
  - *Accès parallèle aux données,*
  - *Utiliser plusieurs noeuds (avec un bon coût/performance), et les faire communiquer par un réseau.*
- Les BDRs se sont développées, grâce au progrès technologiques réalisés au niveau de l'infrastructure réseau et des postes de travail.

# Objectifs

1. Transparence pour l'utilisateur
2. Autonomie de chaque site
3. Absence de site privilégié
4. Continuité de service
5. Transparence vis à vis de la localisation des données
6. Transparence vis à vis de la fragmentation
7. Transparence vis à vis de la réplication
8. Traitement des requêtes distribuées
9. Indépendance vis à vis du matériel
10. Indépendance vis à vis du système d'exploitation
11. Indépendance vis à vis du réseau
12. indépendance vis à vis du SGBD



# Problèmes à surmonter


1. **Coût** : la distribution entraîne des coûts supplémentaires en terme de communication, et en gestion des communications (-hardware et software à installer pour gérer les communications et la distribution).
2. **Problème de concurrence.**
3. **Sécurité** : la sécurité est un problème plus complexe dans le cas des bases de données réparties que dans le cas des bases de données centralisées.

# Objectifs

- Permet aux utilisateurs de partager des données géographiquement réparties.
- Besoin de décentralisation des organisations, critères économiques
- **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 dû à la communication inter-sites

# 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
  - *Définition : données locales/réparties*
  - *Cohérence des données*
  - *Contrôle de concurrence*
  - *Reprise après panne*
  - *Optimisation de question*
  - *Indépendance des applications*
    - machine
    - système d'exploitation
    - protocole réseau
    - système de gestion de bases de données
    - localisation des données

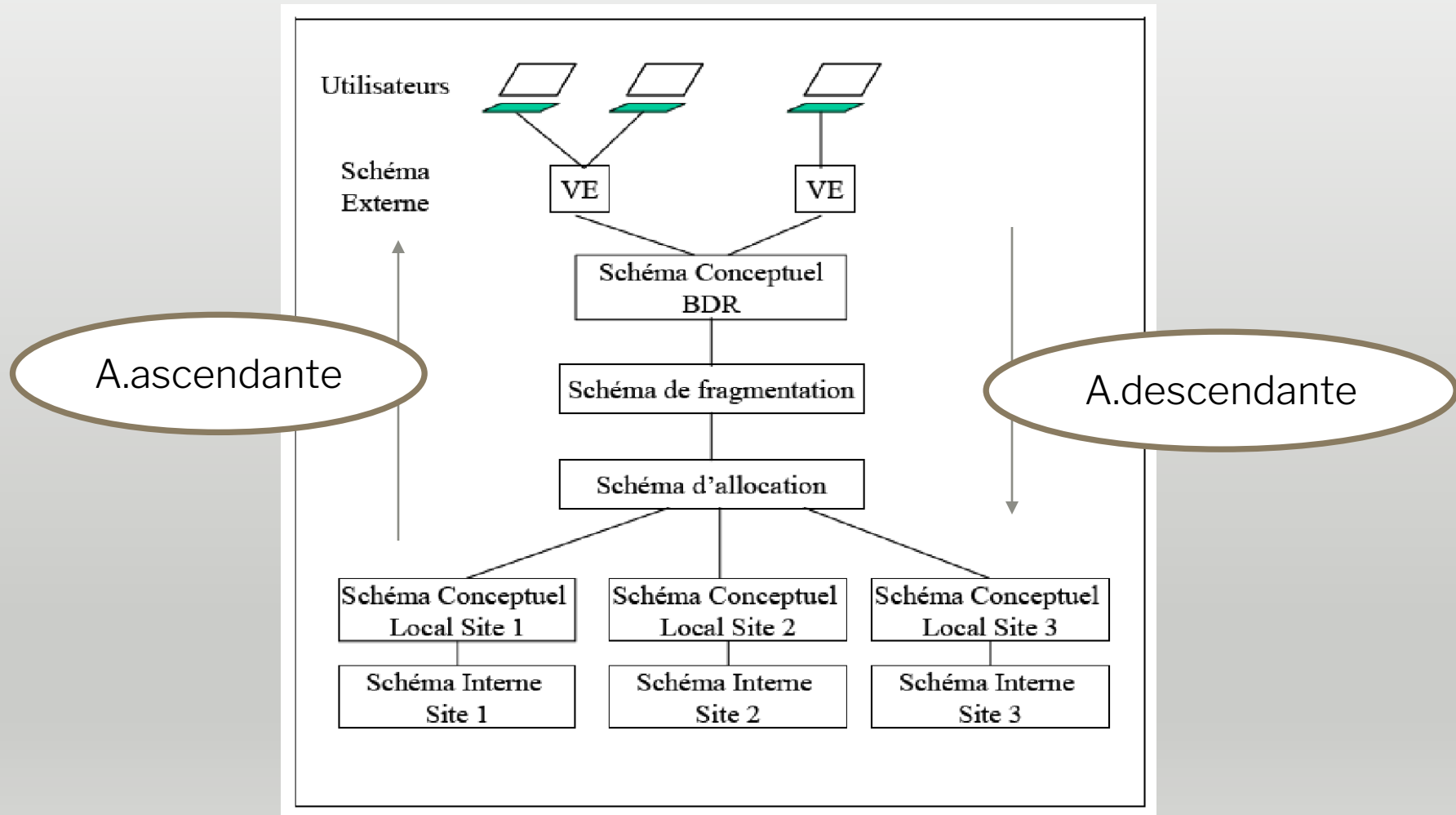
A thick black L-shaped frame is positioned around the text. It starts at the top left, goes right, then down, then right again, forming a large 'L' shape that frames the text on the left and bottom.

# CONCEPTION DES BDS REPARTIES

# Deux approches

- *Descendante : Top-down (du centralisée au distribuée)*
- *Ascendante Bottom-up ( a partir de SGBDs existants vers des vues intégrées)*

# Architecture d'une BDR



# Architecture d'une BDR

- La répartition d'une base de donnée intervient dans les trois niveaux de son architecture en plus de la répartition physique des données :
- **Niveau externe:** les vues sont distribuées sur les *sites utilisateurs*.
- **Niveau conceptuel:** le schéma conceptuel des données est associé, par l'intermédiaire du schéma de répartition (lui même décomposé en un schéma de fragmentation et un schéma d'allocation), aux schémas locaux qui sont réparties sur plusieurs sites, les *sites physiques*.
- **Niveau interne:** le schéma interne global n'a pas d'existence réelle mais fait place à des schémas internes locaux répartis sur différents sites.

# Approche descendante

- 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 seul site
  - La réplication
  - La fragmentation
  - Réplication + fragmentation



# Réplication

- Copie de chaque relation sur plusieurs site.
- 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.*

# 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.*

# Fragmentation

➤ Comment fragmenter?

On distingue trois possibilité de fragmentation:

- Fragmentation Horizontale
- Fragmentation Verticale
- Fragmentation Hybride

# Fragmentations correctes?

- Complétude:  $R$  fragmentée en  $R_1, R_2, \dots, R_n$  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  $R$ ,  $F = \{R_1, R_2, \dots, R_n\}$ 
  - *il est toujours possible de reconstruire  $R$  en appliquant des opérations sur  $F$*
- Disjonction – fragments de  $R$  contiennent des sous ensembles de  $R$ .  $R_i \cap R_j = \emptyset$ .
  - *Garantit l'absence de redondance*

# 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.

# Example

				ALLOC	ENO	PNO	RESP	DUR
EMP	ENO	ENom	TITLE		E1	P1	Mngr	12
	E1	M.Ahmed	Elec. Ing.		E2	P1	Anal.	24
	E2	M. Ali	Syst. Anal.		E2	P2	Anal.	6
	E3	T.Salim	Mech. Ing.		E3	P3	Cons.	10
	E4	S.Riad	Programmeur		E3	P4	Eng.	48
	E5	K.Rafik	Syst. Anal.		E4	P2	Prog.	18
	E6	L.Mohamed	Elect. Ing.		E5	P2	Mngr	24
	E7	R. Davis	Mech. Ing.		E6	P4	Mngr	48
	E8	J. Omar	Syst. Anal.					

# Exemple

Proj	PNO	PNom	Budget	Lieu
	P1	Bioinfo	500000	Alger
	P2	ELearn.	300000	Temouchent
	P3	Plasma	100000	Oran
	P4	Aircraft	150000	Alger

Sal	Titre	Salaire
	Elec. Eng.	40000
	Syst. Anal.	34000
	Mech. Eng.	27000
	Programmer	24000

# Exemple de fragmentation horizontale

Proj <sub>1</sub>	PNO	PNom	Budget	Lieu
	P1	Bioinfo	500000	Alger
	P2	ELearn.	300000	Temouchent

Proj <sub>2</sub>	PNO	PName	Budget	Lieu
	P3	Plasma	100000	Oran
	P4	téléphones	150000	Alger

tuples avec BUDGET > 200.000 vont à Proj1 et le reste à Proj2.

Proj1 =  $\sigma(\text{budget} > 200.000) \text{ Proj}$

Proj2 =  $\sigma(\text{budget} \leq 200.000) \text{ Proj}$



# Fragmentation horizontale

- La reconstruction de la relation est définie par l'union des fragments.
- Les requêtes de l'utilisateur incluent des prédicats plus complexes qui sont des combinaisons des prédicats simples. Puisqu'on peut toujours transformer une expression booléenne en une forme normale conjonctive, on se base sur les prédicats conjonctives pour fragmenter

# Comment?

- Comment obtenir des fragments disjoints et assurer la reconstruction ?
  - *Génération automatique*
  - *Souvent « manuelle » par l'administrateur.*
  - *Différentes méthodologies selon le type de fragmentation.*

# Analyse fragmentation horizontale

- Commencer avec les conditions de sélection fréquentes
  - Extraire des requêtes les conditions de **sélections**:
    - $A < 10$ ,  $A > 5$ , Ville = Temouchent, Ville = Oran...
  - On obtient un ensemble :  $CC = \{c1, c2, \dots, cn\}$  des conditions élémentaires (ce).

# Fragmentation Horizontale

- Construire l'ensemble des conjonctions de **ce** (cc) suivant :

$$CC = \left\{ \bigwedge_{i=1,n} C_i^* \quad \text{ou } C_i^* \text{ est soit } c_i \text{ soit } \neg c_i \right\}$$
$$|CC| = ??$$

- *Simplifier chaque cc*
- *Supprimer les cc qui sont toujours fausses*

# Exemple

- $A < 10, A > 5, \text{Ville} = \text{Temouchent}, \text{Ville} = \text{Oran}$
  - $A < 10, A > 5, \text{Ville} = \text{Temouchent}, \text{Ville} \neq \text{Oran}$
  - $A < 10, A > 5, \text{Ville} \neq \text{Temouchent}, \text{Ville} = \text{Oran}$
  - $A < 10, A > 5, \text{Ville} \neq \text{Temouchent}, \text{Ville} \neq \text{Oran}$
  - $A < 10, A \leq 5, \text{Ville} = \text{Temouchent}, \text{Ville} = \text{Oran}$
  - $A < 10, A \leq 5, \text{Ville} = \text{Temouchent}, \text{Ville} \neq \text{Oran}$
  - $A < 10, A \leq 5, \text{Ville} \neq \text{Temouchent}, \text{Ville} = \text{Oran}$
  - $A < 10, A \leq 5, \text{Ville} \neq \text{Temouchent}, \text{Ville} \neq \text{Oran}$
- 
- $A \geq 10, A > 5, \text{Ville} = \text{Temouchent}, \text{Ville} = \text{Oran}$
  - $A \geq 10, A > 5, \text{Ville} = \text{Temouchent}, \text{Ville} \neq \text{Oran}$
  - $A \geq 10, A > 5, \text{Ville} \neq \text{Temouchent}, \text{Ville} = \text{Oran}$
  - $A \geq 10, A > 5, \text{Ville} \neq \text{Temouchent}, \text{Ville} \neq \text{Oran}$
  - $A \geq 10, A \leq 5, \text{Ville} = \text{Temouchent}, \text{Ville} = \text{Oran}$
  - $A \geq 10, A \leq 5, \text{Ville} = \text{Temouchent}, \text{Ville} \neq \text{Oran}$
  - $A \geq 10, A \leq 5, \text{Ville} \neq \text{Temouchent}, \text{Ville} = \text{Oran}$
  - $A \geq 10, A \leq 5, \text{Ville} \neq \text{Temouchent}, \text{Ville} \neq \text{Oran}$

# Éliminer les inutiles ...

- Conditions non satisfiables
- Connaissance des contraintes d'intégrité, e.g; ville soit Temouchent, soit Oran
  - $A < 10, A > 5, Ville = Temouchent, Ville \neq Oran$
  - $A < 10, A > 5, Ville \neq Temouchent, Ville = Oran$
  - $A < 10, A \leq 5, Ville = Temouchent, Ville \neq Oran$
  - $A < 10, A \leq 5, Ville \neq Temouchent, Ville = Oran$
  - $A \geq 10, A > 5, Ville = Temouchent, Ville \neq Oran$
  - $A \geq 10, A > 5, Ville \neq Temouchent, Ville = Oran.$

# Fragments Finaux

- Regrouper les conditions sur un même attribut
  - $5 < A < 10$ , Ville = Temouchent
  - $5 < A < 10$ , Ville = Oran
  - $A \leq 5$ , Ville = Temouchent
  - $A \leq 5$ , Ville = Oran
  - $A \geq 10$ , Ville = Temouchent
  - $A \geq 10$ , Ville = Oran

# Propriétés de la fragmentation

- **Complétude :**

- *tout  $n$ -uplet  $t$  vérifie une des cc et les cc qui ont été éliminé sont seulement les cc impossibles*

- **Disjonction :**

- L'INTERSECTION entre deux CC est l'ensemble vide



# Fragmentation Horizontale Dérivée

- Fragments définis par (semi) jointure !
- Exemple de la relation
  - *ALLOC(ENO,PNO,RESP,DUR)*
  - ...
- Reconstruction par union des fragments

ALLOC	ENO	PNO	RESP	DUR
	E1	P1	Mngr	12
	E2	P1	Anal.	24
	E2	P2	Anal.	6
	E3	P3	Cons.	10
	E3	P4	Eng.	48
	E4	P2	Prog.	18
	E5	P2	Mngr	24
	E6	P4	Mngr	48

# Fragmentation verticale

- La fragmentation concerne le schéma.
- Les fragments sont définis par des opérations de projection.
- La reconstruction est définie par jointure.
- La clé doit être répétée dans chaque fragment.

# Exemple de fragmentation verticale

Proj <sub>1</sub>	PNO	Budget
	P1	500000
	P2	300000
	P3	1000000
	P4	150000

Proj <sub>2</sub>	PNO	PNom	lieu
	P1	Bioinfo	Alger
	P2	ELearn.	Tlemcen
	P3	Plasma	Ain Temouchent
	P4	Téléphones	Oran

# Matrice d'utilisation

- Soit la relation  $\text{Projet}(\text{Pnum}, \text{Pnom}, \text{Budget}, \text{Lieu})$  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:
  - $t(q_i, A_j) = 1$  si la requête  $q_i$  utilise l'attribut  $A_j$  et
  - $Ut(q_i, A_j) = 0$  sinon.

# Exemple

- **Matrice d'utilisation**

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

# Matrice d'affinité

- Une matrice triangulaire permettant d'exprimer l'affinité entre les attributs pour les regrouper.

The diagram illustrates an affinity matrix as a square grid enclosed in large square brackets. The columns are labeled at the top as A1, A2, ..., A<sub>i</sub>, ..., A<sub>j</sub>, ..., A<sub>n</sub>. The rows are labeled on the left as A1, A2, ..., A<sub>i</sub>, ..., A<sub>j</sub>, ..., A<sub>n</sub>. A horizontal dotted line extends from the left bracket to the cell at the intersection of row A<sub>i</sub> and column A<sub>j</sub>, which contains the value a<sub>ij</sub>. A vertical dotted line extends from the top bracket to the same cell. This visualizes the matrix as lower triangular, where the value a<sub>ij</sub> represents the affinity between attributes A<sub>i</sub> and A<sub>j</sub>.

L'affinité  $a_{ij}$  entre les attributs  $A_i$  et  $A_j$  correspond au nombre de requêtes qui accèdent à la fois à l'attribut  $A_i$  et à l'attribut  $A_j$ .

# Exemple

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

Regroupement des attributs

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



# Matrice d'affinité

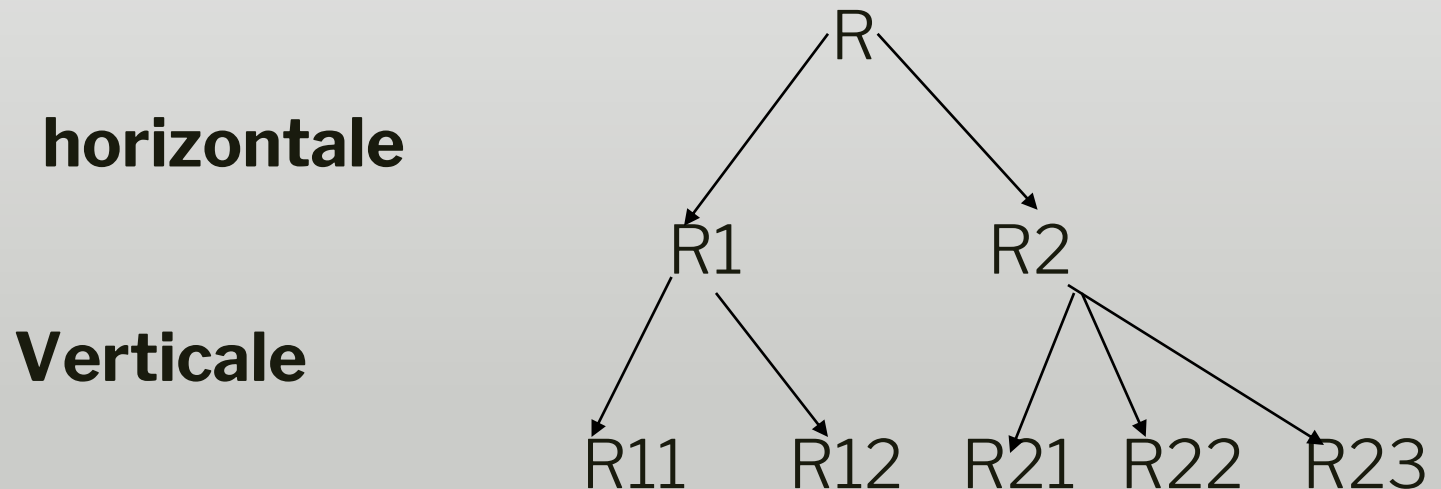
- La matrice d'affinité est utilisée pour guider la génération des fragments verticaux
- les attributs ayant une grande affinité seront groupés pour former un fragment vertical

# Objectifs

- Maximiser les accès à un seul fragment.
- Minimiser les accès aux deux fragments.
- La relation projet est partitionné en deux fragments:  
  
Projet1(Pnum, Budget).  
  
Projet2(Pnum, Pnom, Loc).
- La duplication de la clé permet de garantir la reconstruction.

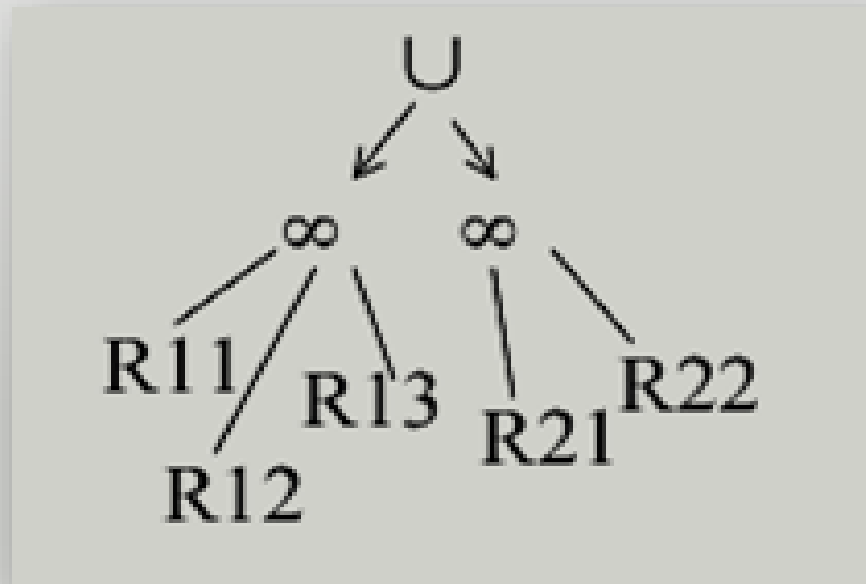
# Fragmentation Hybride

- Fragmentation horizontale suivit de la fragmentation verticale ou vice-versa



# Fragmentation Hybride

## Reconstruction



# Allocation

- Supposons qu'on dispose d'un ensemble de fragments  $F=\{F_1, F_2, \dots F_n\}$  et d'un réseau constitué d'un ensemble de sites  $S=\{S_1, S_2, \dots S_n\}$ .

- Une distribution optimale de  $F$  sur  $S$  est définie en considérant deux mesures:

- **Un coût minimal**

La fonction coût est une combinaison d'un ensemble de coûts:

- Coût de stockage de chaque fragment  $F_i$  sur un site  $S_j$ .
- Coût de modification de  $F_i$  sur un site  $S_j$ .
- Coût d'interrogation de  $F_i$  sur un site  $S_j$ .
- Coût de communication.

- **Une meilleure performance**

- Minimiser le temps de réponse.

## QUELQUES COMMANDES UTILES SUR ORACLE

- Création de lien logique entre les bases  
Create database link ...
- Création de fragments  
Create table ... (Copy from ...)
- Création de répliques  
Create materialized view ...

# DATABASE LINK

Créer un lien avec une table dans une BD distante

```
CREATE DATABASE LINK <nomLien>  
CONNECT TO<loginUser> IDENTIFIED BY <pwd>  
USING '<nomBD>'
```

# CREATE MATERIALIZED VIEW

Permet de créer des fragments à partir d'une base existante :

```
CREATE MATERIALIZED VIEW <nomRépl>  
REFRESH [on commit | on demand | start with ... next ...]  
[ complete | fast | force ]  
AS ( SELECT ... FROM ... WHERE ... );
```

Exemple de commande exécutée sur la base **orapeda2** pour créer une réplique sur la base **orapeda1** :

```
CREATE MATERIALIZED VIEW joueurs  
REFRESH ON DEMAND COMPLETE  
AS (Select nujoueur, nom, prenom, nation, annais  
      From ious@lienora2versora1);
```



# Résumé(approche descendante)

