Gestion de Grandes Masses de Données

GGMD

NICOLAS LUMINEAU

département
Informatique

Université Claude Bernard Lyon 1

PARCOURS TIW / DS-INFO / BIO-INFO

nicolas.lumineau@univ-lyon1.fr

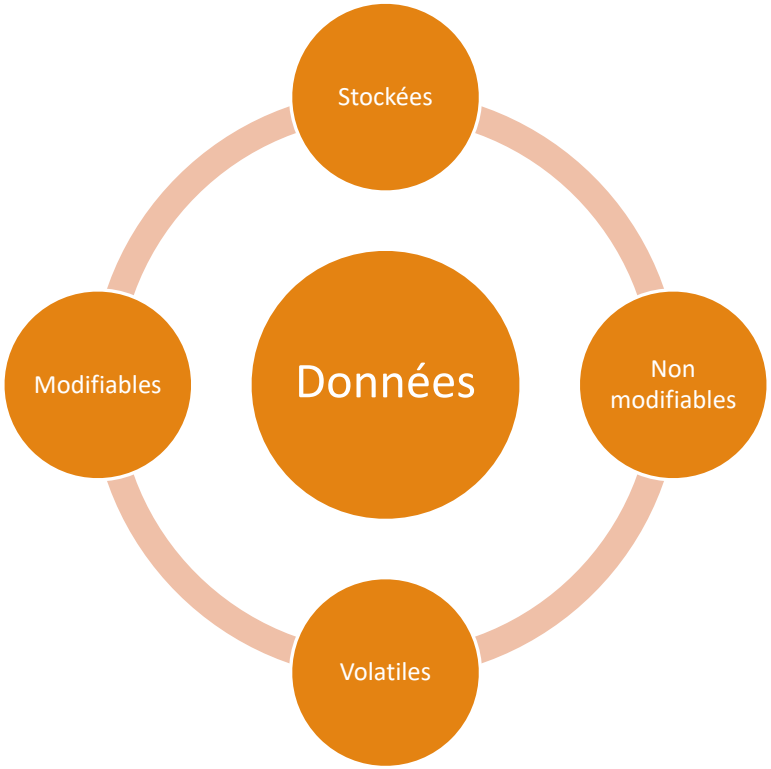
C’est une UE sur :

Gestion de Données …

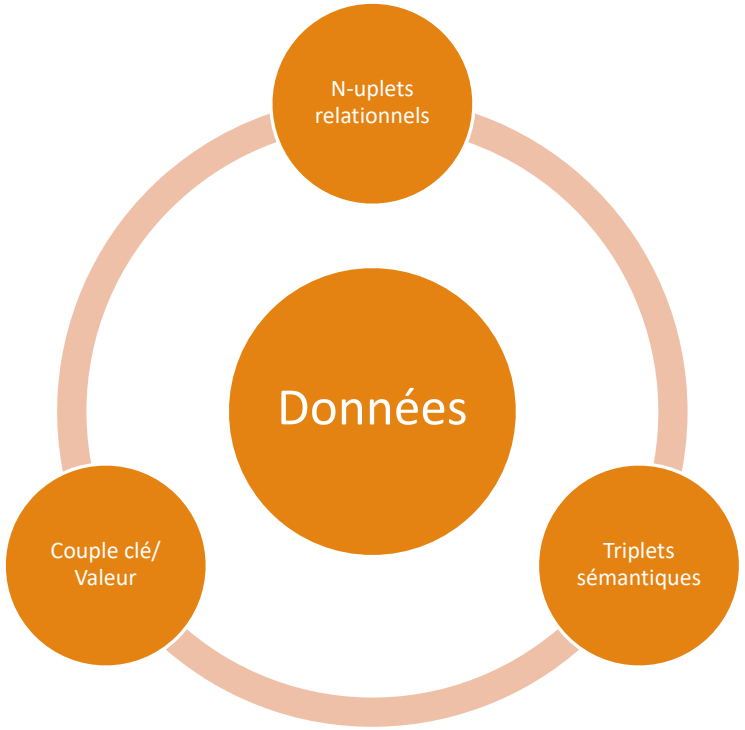
… et de comment faire
quand on a de Grandes Masses de Données à traiter

Gestion de Grandes Masses de Données

Quelle catégorie de données ?



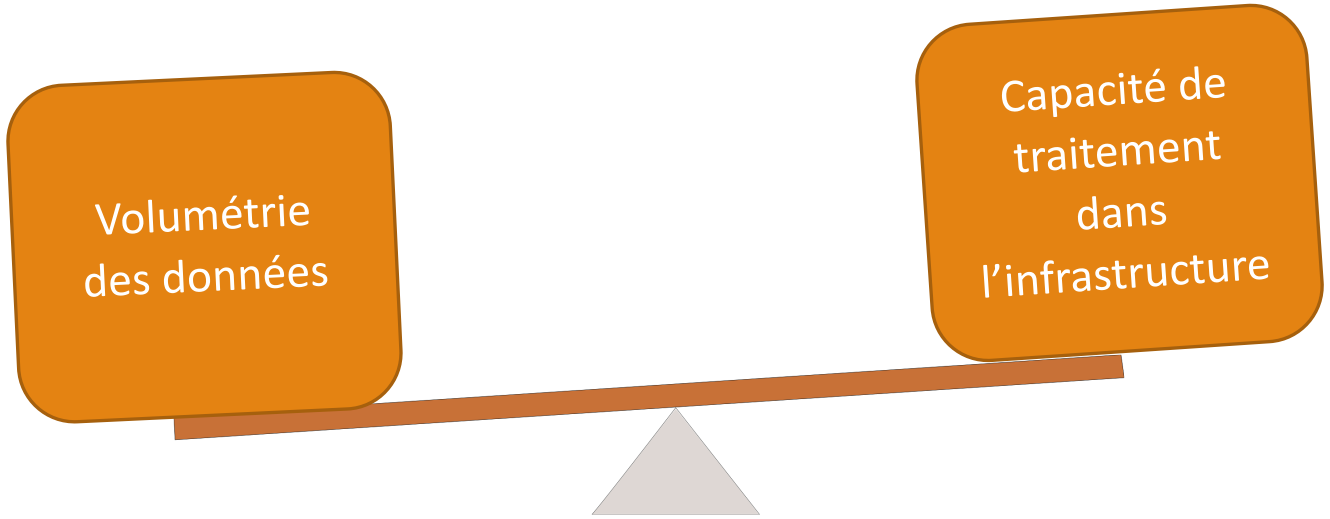
Quel type des données ?



Gestion de Grandes Masses de Données

Quand est-ce que l'on parle de Masse de Données ?

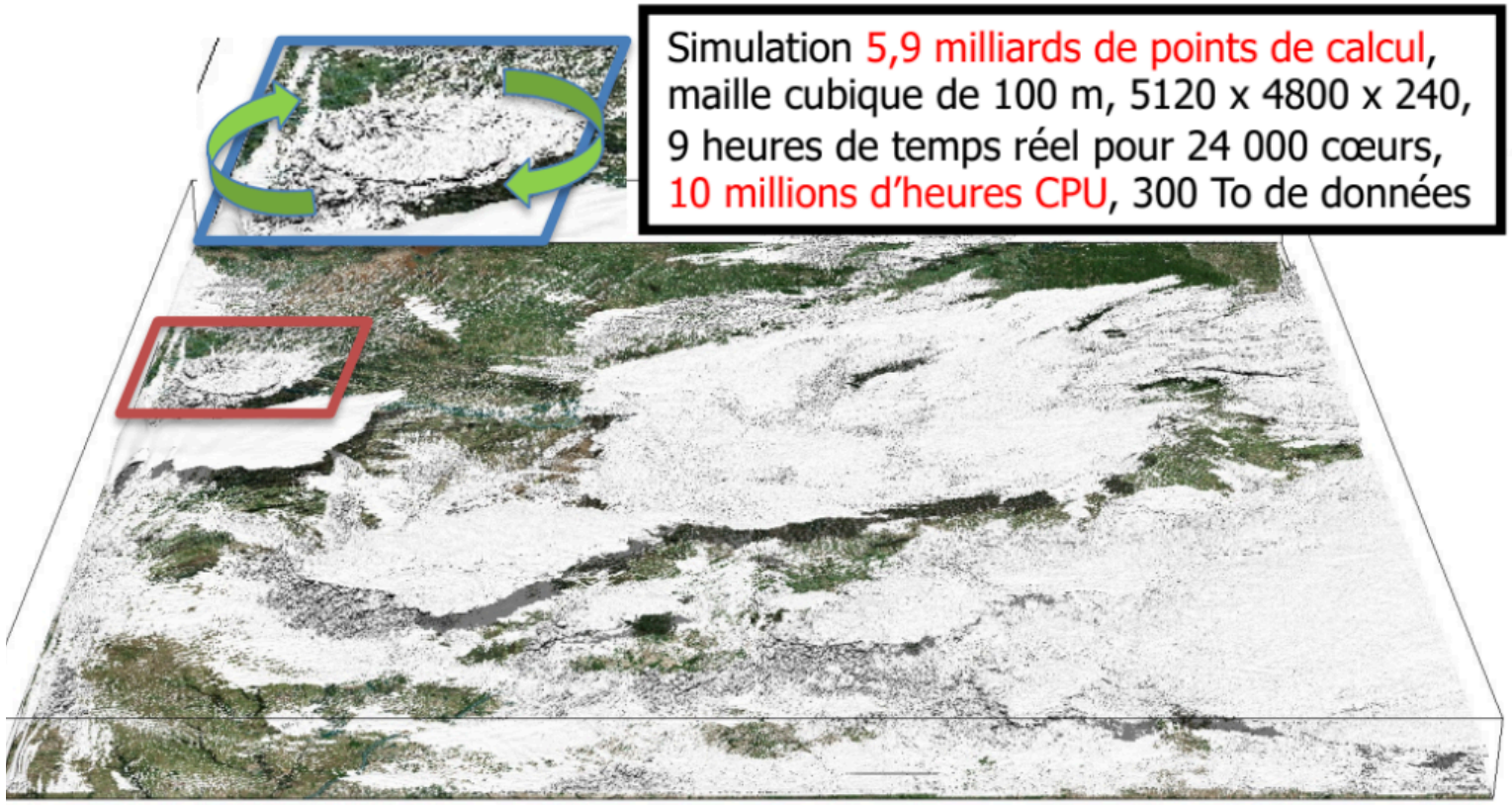
- Combien de To, Po ou Zo ?
  - A titre d'exemple, la NASA en 2021, c'est 40 Po (prévision à 245Po en 2025)*

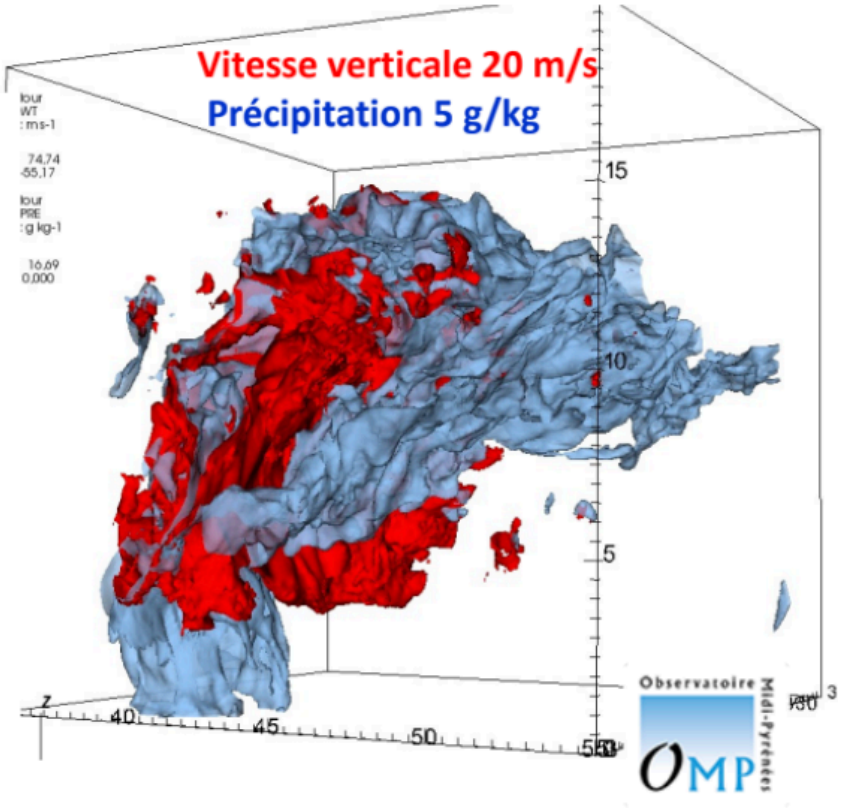


* Source TrustMyScience

Quelques exemples : en météorologie

Formation rafales et grêle dans un orage violent

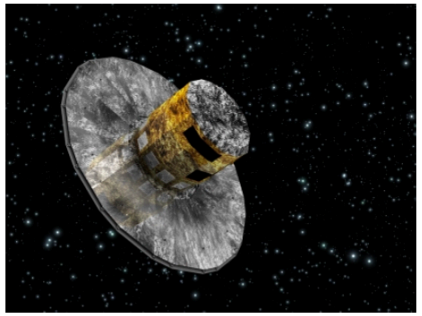




Présentation Sylvie Joussaume - ASNUM 2022

Quelques exemples : en astronomie

Vue d'artiste de SKA
Crédit: SKA Organisation



Vue d'artiste de Gaia
Crédit: ESA

Vue d'oiseau du Very Large Telescope
Crédit: J.L. Dauvergne & G. Hüdepohl (atacamaphoto.com)/ESO

Hubble, le télescope spatial
Crédit: NASA, 2002

XMM-Newton
Crédit: Image courtesy of ESA

Sous le charme des Nuages de Magellan, ALMA
Crédit: ESO/C. Malin

Remerciements André Schaaff du CDS

Vera C. Rubin

- Production de 20 To par nuit

SKAO

- Archivage de 300 Po par année

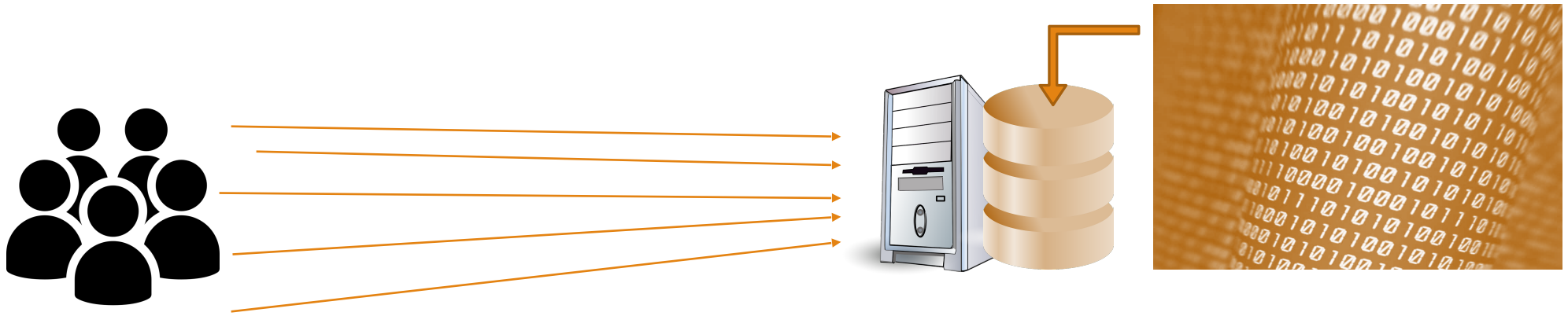
James Webb

- Production de 68 Go par 24h

Gestion de Grandes Masses de Données

Quel contexte pour gérer les données ?

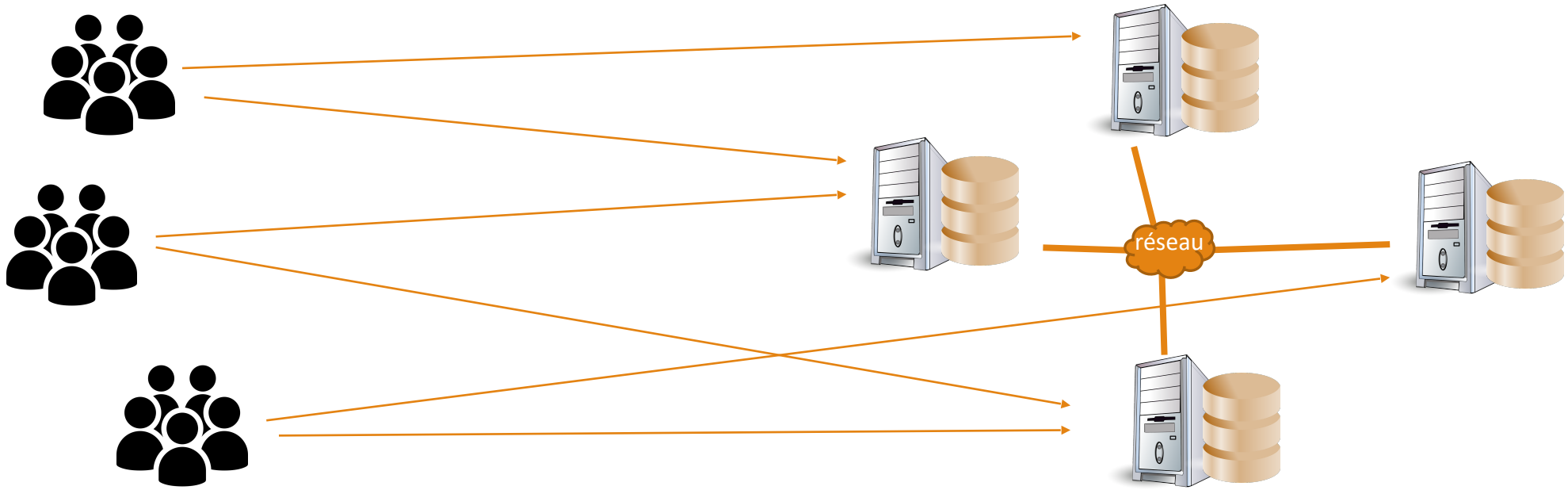
Des utilisateurs + Un serveur avec une forte capacité de stockage et de calcul



Gestion de Grandes Masses de Données

Quel contexte pour gérer les données ?

Des utilisateurs + Données et des calculs distribués sur quelques serveurs



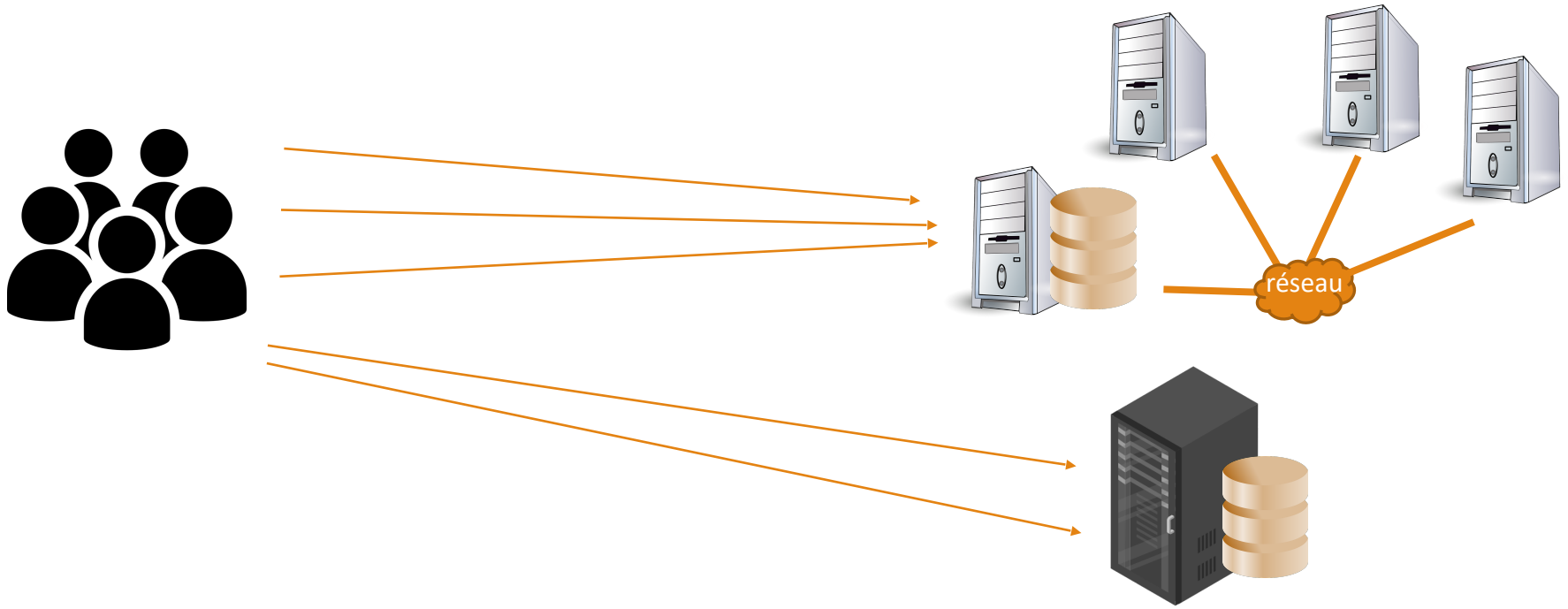
Gestion de Grandes Masses de Données

Quel contexte pour gérer les données ?

Des utilisateurs

+

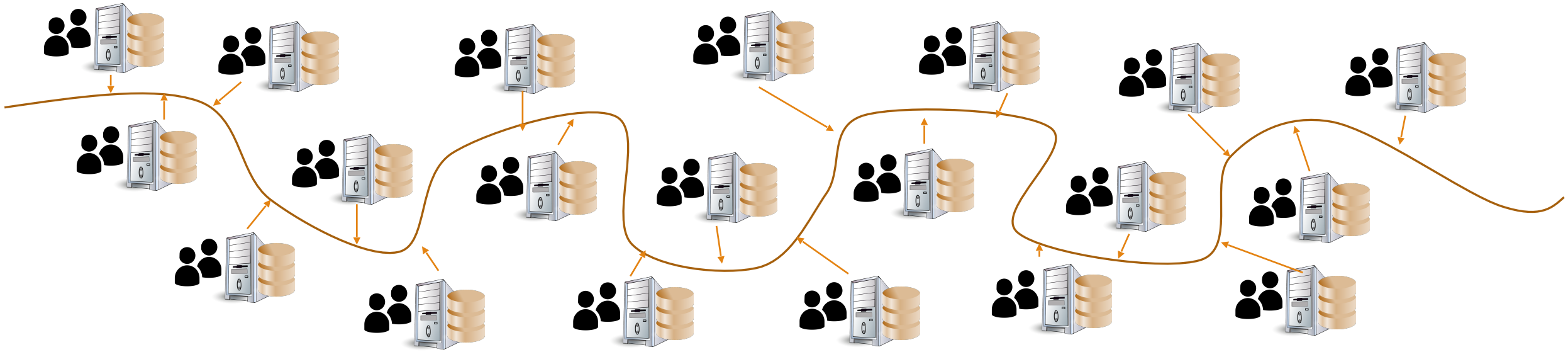
Des serveurs distribués dédiés au calcul



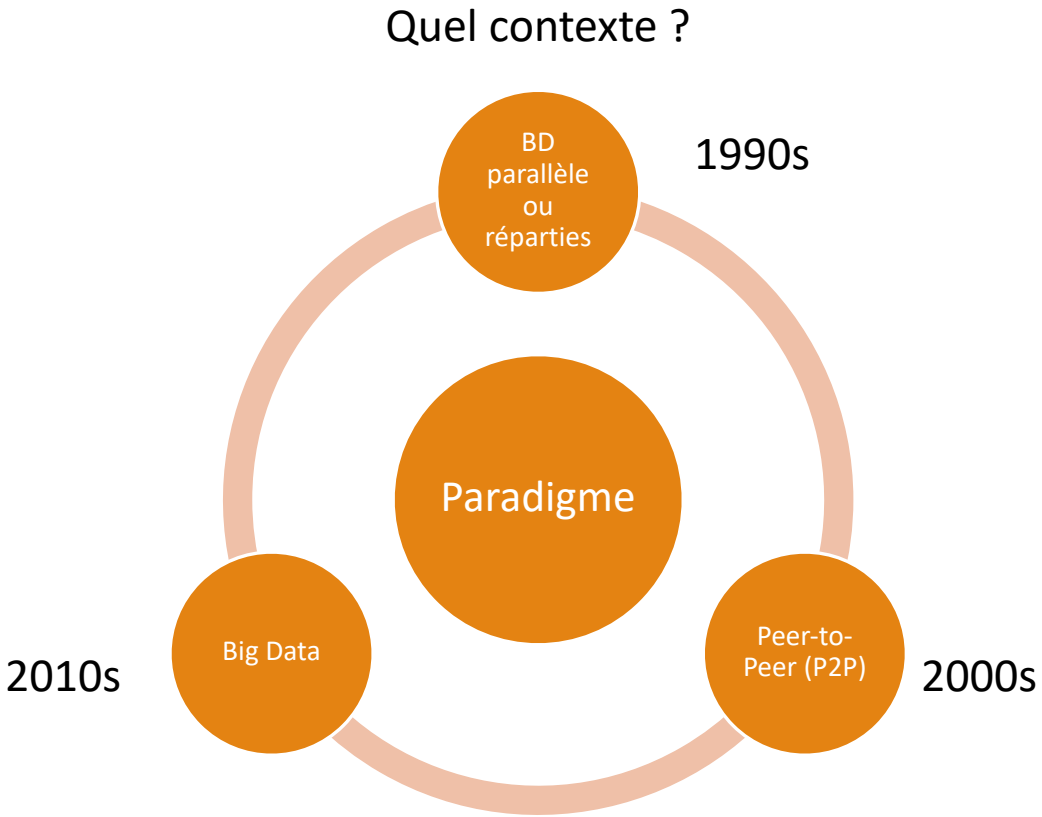
Gestion de Grandes Masses de Données

Quel contexte pour gérer les données ?

Un très grand nombre d'utilisateurs + Un très grand nombre de serveurs distribués dédiés au stockage et/ou au calcul



Gestion de Grandes Masses de Données



Gestion de Grandes Masses de Données

Quel objectif ?

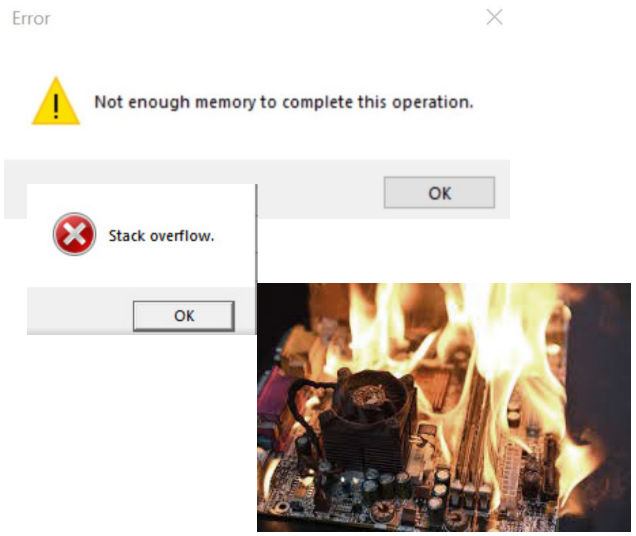
- **Offrir à l'utilisateur final un accès satisfaisant aux données selon le paradigme le plus adapté au besoin.**

Qu'est-ce qu'un accès satisfaisant ?

- Les requêtes/traitements aboutissent à un résultat avec une latence acceptable pour l'usage

Qu'est-ce que le paradigme le plus adapté au besoin ?

- Le paradigme qui offrira les meilleurs performances, avec le niveau de fiabilité (des résultats ou du système) souhaité par l'utilisateur, pour effectuer les traitements



Objectifs de l'UE

- Comprendre les grands challenges liés au traitement de grandes masses de données.
- Comprendre les paradigmes et leur différences
- Prendre en main différents outils répondant à ces paradigmes

Hypercore Protocol

Pourquoi est-ce si important de maîtriser ces paradigmes ?

Pour avoir du recul par rapport à l'**effet buzz** !

Retours d'expériences en entreprise.

A l'ère du **Big Data**, il y a des solutions de **Big Data** pour répondre à des problèmes de **Big Data**

A l'ère du **Big Data**, il y a des solutions de **BD Relationnelles** pour répondre à des problèmes de **BD relationnelles**

Organisation de l'UE

- Créneaux
  - Mercredi matin 8h-13h

- Intervenants
  - CM : N. Lumineau (Parties BD Réparties/P2P/Flux) et M-S. Hacid (Partie BigData)
  - TD + TP : N. Lumineau, E. Coquery

- Evaluations
  - CC *via* rendus/questionnaires de TP (40%)
  - CCF le 29 novembre 2023 (60%)

- Communications
  - chat.info.univ-lyon1.fr. => M2-Info-GGMD

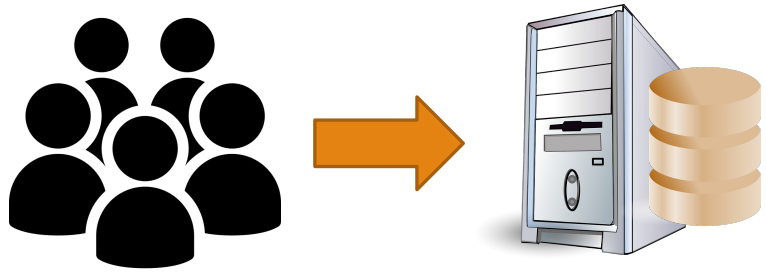
Plan

- Introduction générale
  - Contexte et objectifs
  - Informations sur l'UE
- Focus sur le paradigme SGBD parallèles/réparties
  - Principe de la parallélisation des traitements de requêtes
  - Principe de fragmentation des données en BDR
  - Performances des traitements
- Focus sur le paradigme P2P
  - Principe de la localisation des données
- Focus sur le paradigme Big Data (Mohand-Saïd Hacid)

Contexte initial

Centralisation et séquentialité

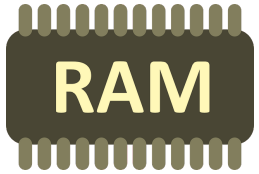
- Les données sont stockées sur un seul serveur
- Les opérations de traitements sont gérées séquentiellement



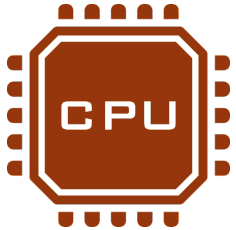
Observations

Plantages ou latences de traitement des requêtes trop importantes pour une utilisation satisfaisante

Problèmes potentiels

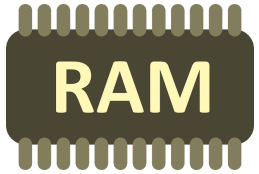


- Pas assez de mémoire pour le chargement et la manipulation des données
  - RAM du serveur insuffisante



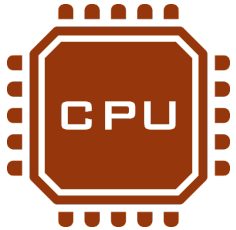
- Pas assez de puissance de calcul pour des requêtes trop coûteuses (trop de jointures…)
  - CPU (en nombre et puissance) pas assez suffisant

Solutions potentielles



- Ajout physique de RAM
- Création d'un fichier de SWAP

Infère de la latence



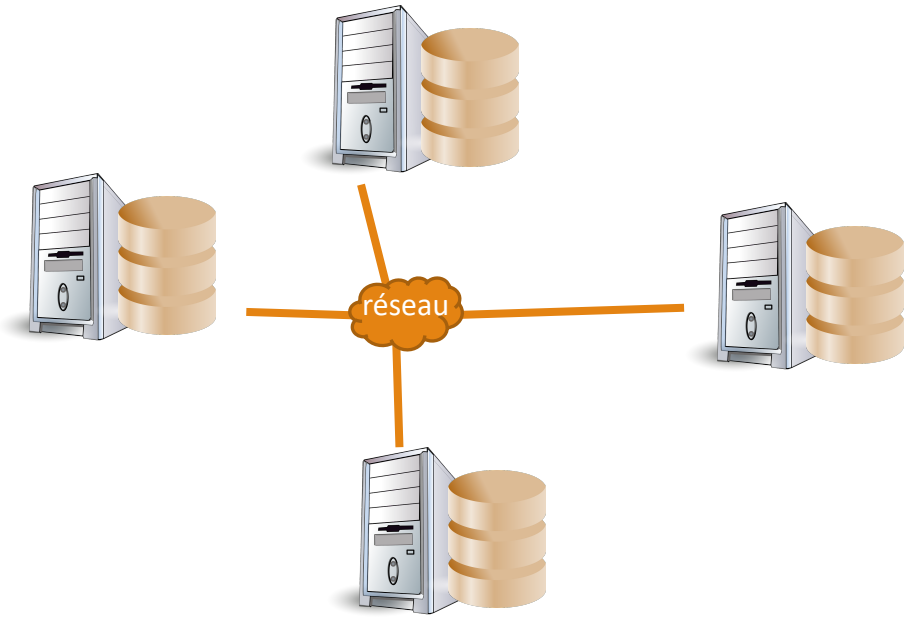
- Distribuer et paralléliser les calculs sur plusieurs CPU

Comment faire ?

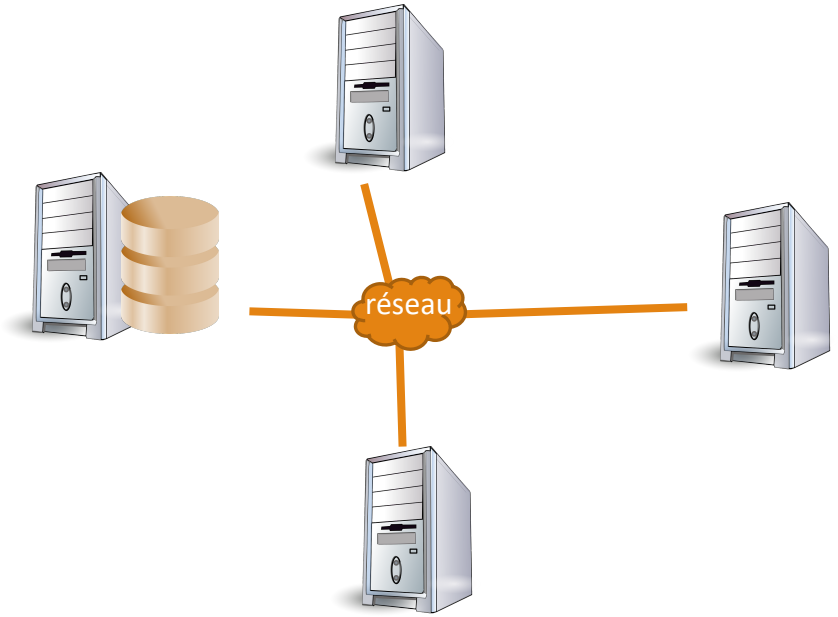
Vers une solution distribuée

Mais pour quel contexte ?

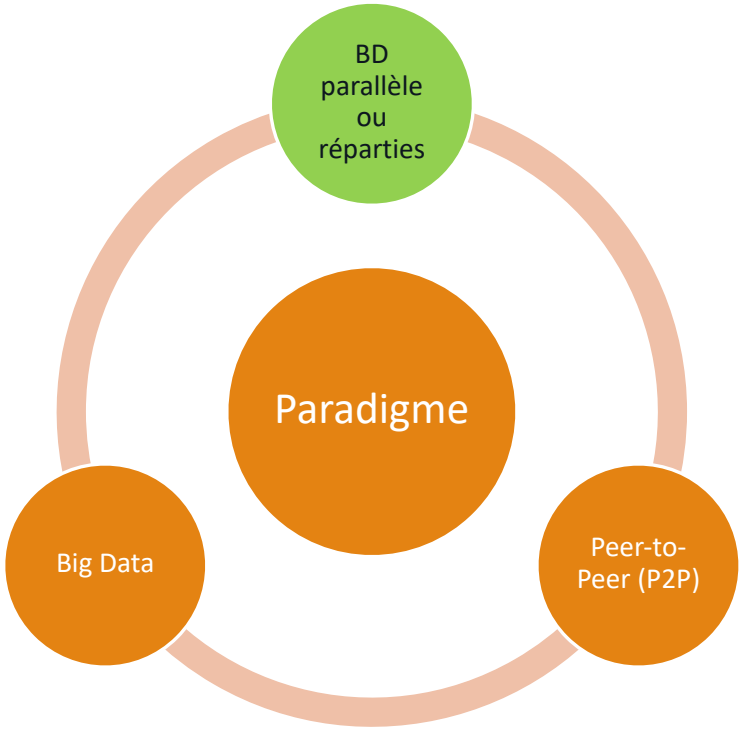
Distribution du stockage des données
et distribution des calculs



Distribution uniquement des calculs
(et stockage des résultats intermédiaires)



Focus sur le paradigme des SGBD parallèles/répartis



Les bases de données parallèles

Principe

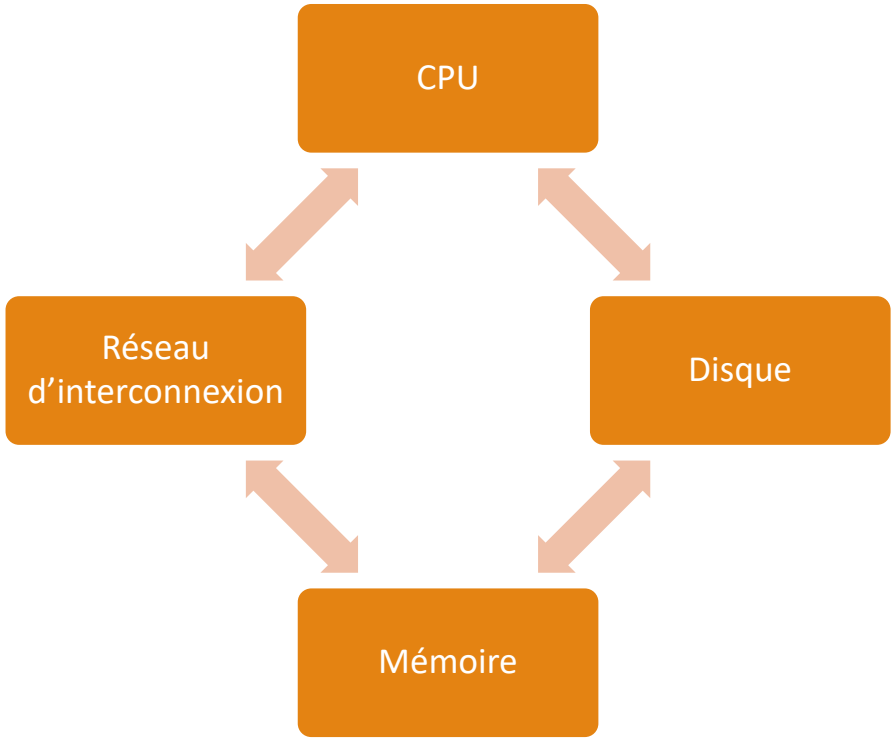
- Orchestration des calculs d'une requête à travers plusieurs CPUs et disques avec une gestion de la mémoire adaptée.

Verrous

- Gestion de l'équilibrage de charge
- Complexité

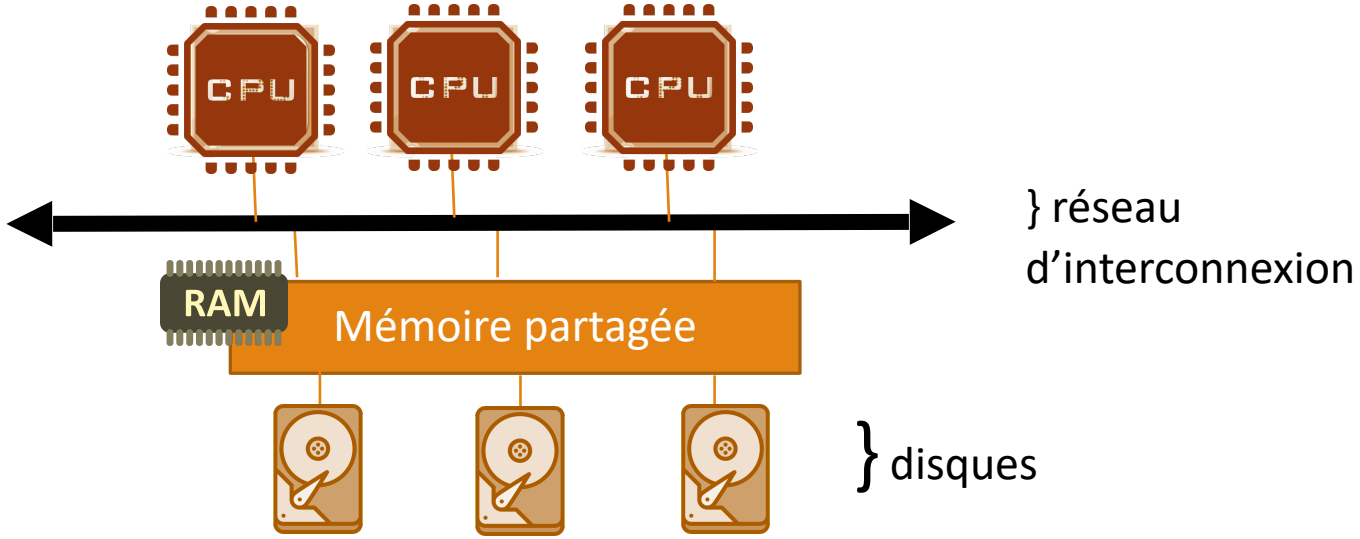
Différents types d'architectures :

- Architecture à Mémoire Partagée
- Architecture à Disque Partagé
- Architecture à Mémoire Distribuée



Base de données parallèles

Architecture à Mémoire Partagée (Shared Memory)



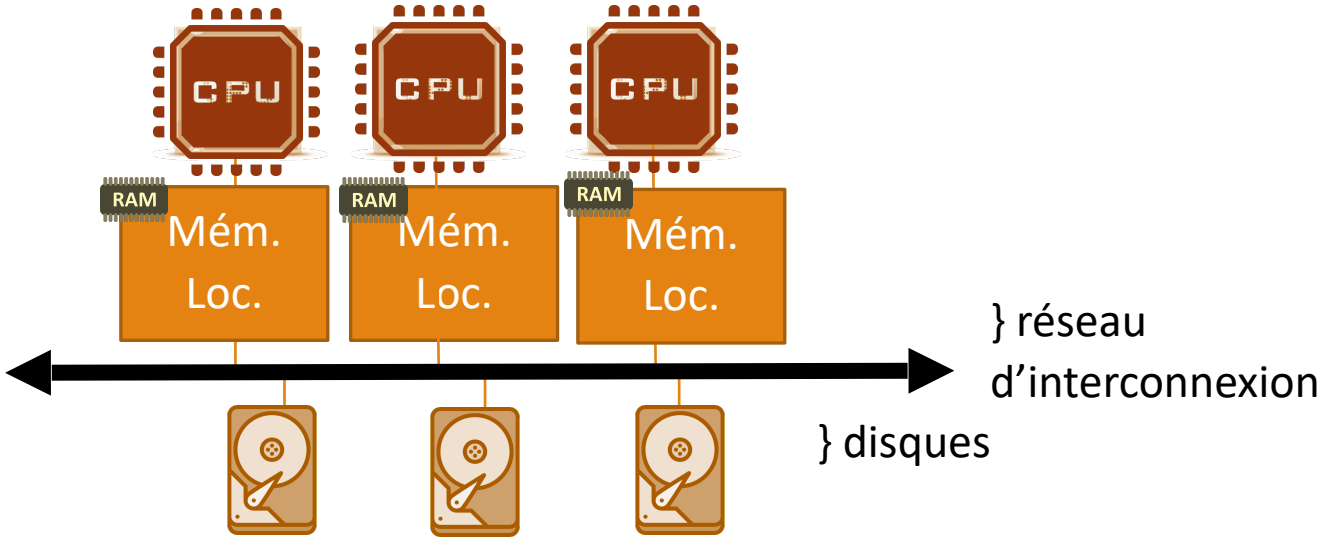
Modèle de programmation:
- mémoire partagée (multiprocessus, multithreading)

Avantages :
- simplicité, équilibrage de la charge

Inconvénients :
- Coût du Réseau (bus parallèle ou multibus), faible extensivité

Base de données parallèles

Architecture à Disque Partagé (Shared Disk)



Avantages :

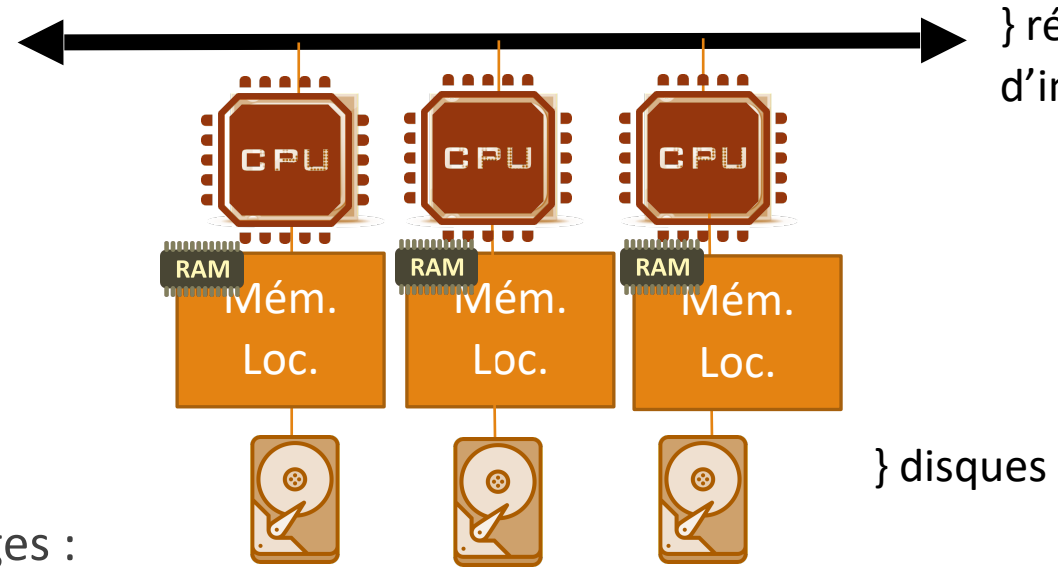
- Coût du Réseau, extensibilité, migration depuis les monoprocesseurs, sûreté de fonctionnement

Inconvénients :

- complexité

Base de données parallèles

Architecture à Mémoire Distribuée (Shared Nothing)



} réseau d'interconnexion

} disques

Avantages :

- Coût, extensibilité, disponibilité

Inconvénients :

- complexité, équilibrage de charge

Base de données parallèles

Performance

En termes de passage à l'échelle

- Mémoire partagée: ~10
- Disques partagés: ~100
- Mémoire distribuée: ~1000

Possibilité de solutions hybrides combinant Mémoire partagée (pour l'équilibrage de charge) et le Mémoire distribuée (pour l'extensibilité)

Vers la conception de bases de données réparties

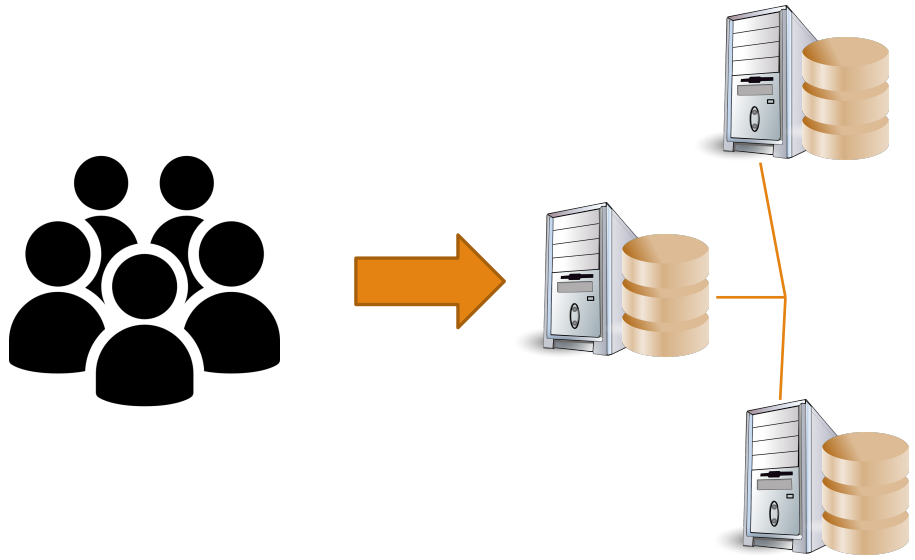
Focus sur le paradigme des SGBD répartis

Distribution et parallélisme

Le données sont des données relationnelles

Les données sont distribuées sur plusieurs SGBD hébergés sur des serveurs interconnectés

On dispose d'un réseau performant



Remarques

Infrastructure distribuée multi-processeurs avec mémoire distribuée

Base de données répartie

Définition

- C’est une base de données logique dont les données sont stockées dans différents SGBD interconnectés

Caractéristiques

- Une base de données répartie dispose d’un schéma global et d’un schéma d’allocation
- La distribution des données est transparente pour l’utilisateur

Base de données répartie

Exemple

personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

commande

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
4	3	Cahier A4 PC	1
5	4	Gomme	25
6	6	Cahier Spirale GC	3

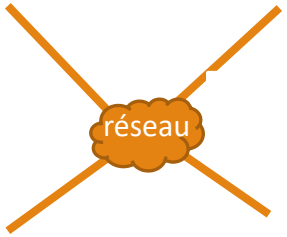
LOGIQUE

personne@Site1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

commande@Site3

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
5	4	Gomme	25



personne@Site2

IdP	NOM	Prénom	Secteur
3	NAULT	Pia	S
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

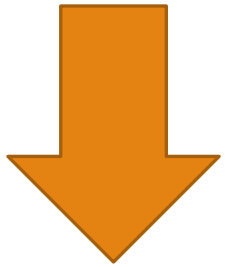
commande@Site4

IdC	IdP	Produit	Qte
4	3	Cahier A4 PC	1
6	6	Cahier Spirale GC	3

PHYSIQUE

Remarque importante

Besoin de formaliser la les éléments de la base de données répartie



FOCUS SUR LE PRINCIPE DE DECOMPOSITION D'UNE BASE DE DONNEES CENTRALISEE

Décomposition

Comment découper logiquement une base pour répartir les n-uplets sur différents sites?

- Conception
  - Décomposition d'une base
- Interrogation
  - Optimisation algébrique et physique des requêtes
  - Modèle transactionnel

Rappel sur la conception d'un SGBD centralisé

Analyse des besoins

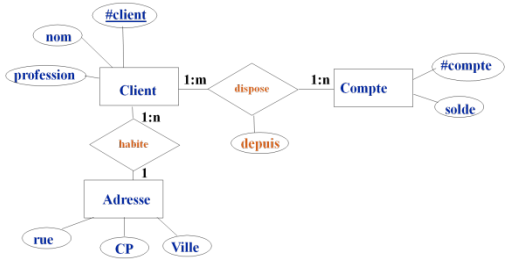


Schéma E/A

Modèle conceptuel de données

Modèle relationnel

Acteur

Nom	Prénom	Nationalité
Cruz	Penelope	ESP
Maura	Carmen	ESP
Johansson	Scarlett	USA
Villeret	Jacques	FR
Lhermite	Thierry	FR
Downey	Robert	UK
Allen	Woody	USA

Artiste = Acteur ∪ Réalisateur

Nom	Prénom	Nationalité
Cruz	Penelope	ESP
Maura	Carmen	ESP
Johansson	Scarlett	USA
Villeret	Jacques	FR
Lhermite	Thierry	FR
Downey	Robert	UK
Almodovar	Pedro	ESP
Allen	Woody	USA
Veber	Francis	FR
Favreau	John	USA

Réalisateur

Nom	Prénom	Pays
Almodovar	Pedro	ESP
Allen	Woody	USA
Veber	Francis	FR
Favreau	John	USA

Modèle relationnel (logique)

Implémentation (physique)

CREATE TABLE Client …
ALTER TABLE Client …
CREATE TABLE Commande …

Dans un contexte distribué

Analyse des besoins

Modèle conceptuel de données

Modèle relationnel (logique)

Implémentation de vues globales (physique)

Processus de conception de BD centralisé

Construction du schéma global

Fragmentation/ Réplication / Migration (physique)

Allocation (Placement)

Distribution des données

Schéma global

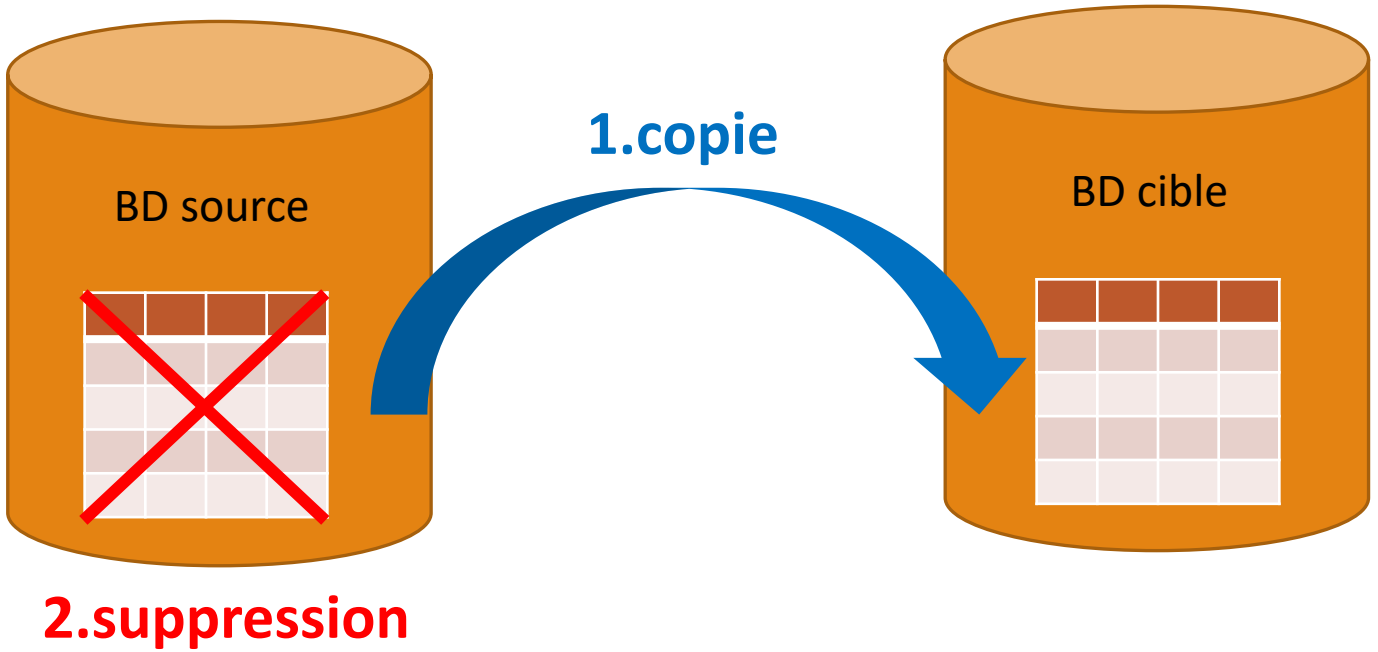
Le schéma global est constitué :

- d'un schéma conceptuel global
  - Contenant la description globale et unifiée de toutes les données de la BDR
  - Indépendant de la répartition des données

- d'un schéma de placement (d'allocation)
  - Contenant les règles de correspondance avec les données locales
  - indépendant à la fragmentation et à la réplication

Distribution des données: Migration

Transfère d'une relation complète sur un site distant



- **Intérêt**
  - Rapprocher les données **du** besoin

Concrètement

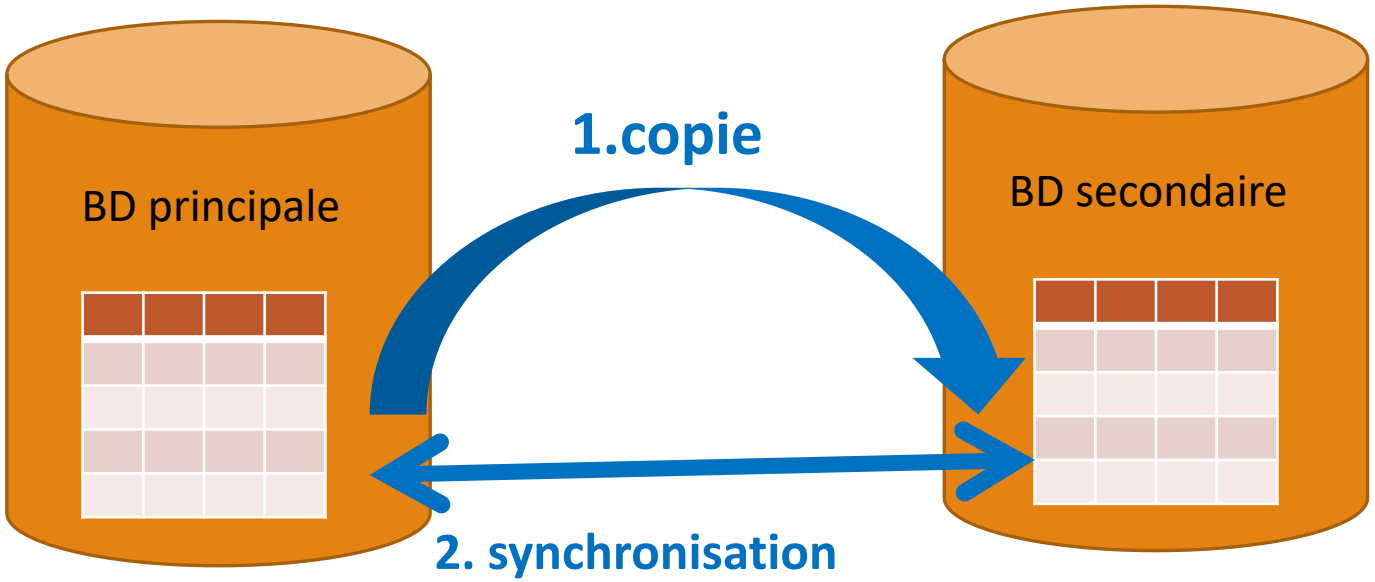
Utilisation de pg_dump

ou

(version 9+)
Création d'un *wrapper* et d'une 'foreign table' pour pouvoir faire en local une création de table à partir d'une requête

Distribution des données: Réplication/Duplication

Création d'une copie conforme d'une table (ou ensemble de tuples) sur un site distant.
La copie doit rester cohérente avec les données sources



Concrètement

Utilisation d'un processus de Publication/ Suscription

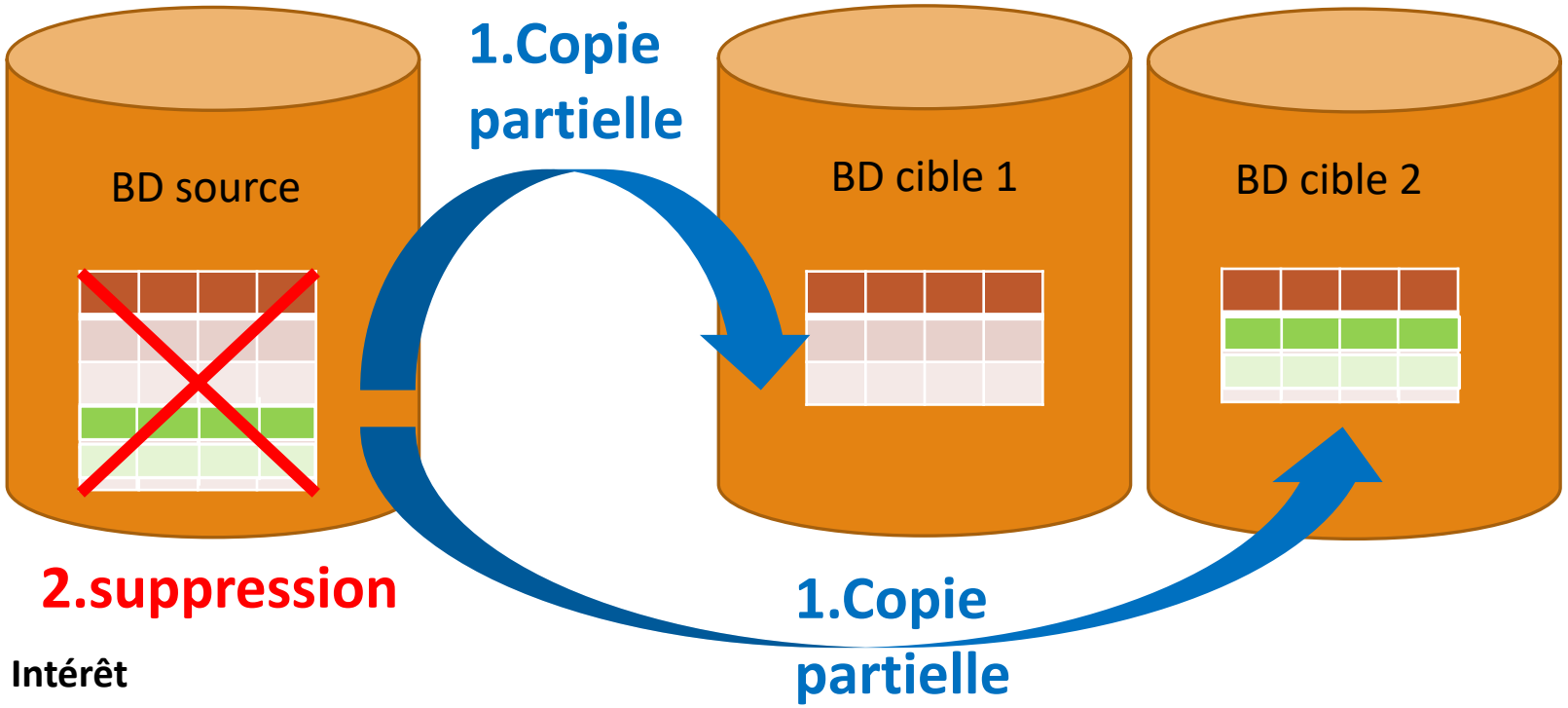
ou

Utilisation des vues matérialisées

- **Intérêt**
  - Rapprocher les données **des** besoins : avoir les mêmes données sur différents sites

Distribution des données: Fragmentation

Décomposition d'une relation en plusieurs fragments qui sont transférés sur différents sites distants



- **Intérêt**
  - Découper logiquement les données

Concrètement

PostgreSQL

Création d'un *wrapper* et d'une 'foreign table' pour pouvoir faire en local une création de table à partir d'une requête intégrant les contraintes de sélection

Différentes fragmentations

- Fragmentation horizontale
- Fragmentation horizontale dérivée

→ Un fragment est une **sélection** d'un sous-ensemble de n-uplets

- Fragmentation Verticale

→ Un fragment est une **projection** sur un sous-ensemble d'attributs

- Fragmentation Mixte

→ Un fragment est une **composition** d'une fragmentation horizontale (dérivée) avec une fragmentation verticale ou inversement

Fragmentation Horizontale

- **Schéma global** : personne (idP, nom, prenom, secteur)
  avec *secteur* ∈ { 'N', 'S' }

- **Opérateur de fragmentation** : $\sigma_{\text{<conditions>}}(\text{relation})$

  ➔ Fragments définis par **sélection**

- **Exemple :** *Fragmentation horizontale selon les secteurs*

$$\text{personne1} = \sigma_{\text{secteur = 'N'}}(\text{Personne})$$

$$\text{Personne2} = \sigma_{\text{secteur = 'S'}}(\text{Personne})$$

personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

personne1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

personne2

IdP	NOM	Prénom	Secteur
3	NAULT	Pia	S
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

Fragmentation Horizontale

- **Schémas locaux** : personne1 (idP, nom, prenom, secteur)
  personne2 (idP, nom, prenom, secteur)

- **Opérateur de reconstruction:** $\bigcup_{i=1}^{n} Fragment_i$

  ➔ Reconstruction par **union**

- **Exemple :** *Reconstruction des personnes*

`Personne = personne1` $\bigcup$ `Personne2`

personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

personne1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

personne2

IdP	NOM	Prénom	Secteur
3	NAULT	Pia	S
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

Fragmentation Horizontale Dérivée

personne1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

personne2

IdP	NOM	Prénom	Secteur
3	NAULT	Pia	S
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

commande

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
4	3	Cahier A4 PC	1
5	4	Gomme	25
6	6	Cahier Spirale GC	3

- **Schéma global** :
  - personne (idP, nom, prenom, secteur)
    avec *secteur* ∈ { 'N', 'S' }
  - commande (idC, #idp, produit, qte)
- **Opérateur de fragmentation** :

relation_à_fragmenter ⋉ fragment_relation_fragmentée

➔ Fragments définis par **semi-jointure**

- **Exemple :** *Fragmentation horizontale de commande dérivée de personne*

commande1 = commande ⋉ (idP) personne1

commande2 = commande ⋉ (idP) personne2

commande1

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
5	4	Gomme	25

commande2

IdC	IdP	Produit	Qte
4	3	Cahier A4 PC	1
6	6	Cahier Spirale GC	3

Fragmentation Horizontale Dérivée

- **Schémas locaux** : personne1 (idP, nom, prenom, secteur)
  commande1 (idC, #idp, produit, qte)
  personne2 (idP, nom, prenom, secteur)
  commande2 (idC, #idp, produit, qte)

- **Opérateur de reconstruction:** $\bigcup_{i=1}^{n} Fragment_i$

  ➔ Reconstruction par **union**

- **Exemple :** *Reconstruction des commandes*

`commande = commande1` $\bigcup$ `commande2`

commande

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
4	3	Cahier A4 PC	1
5	4	Gomme	25
6	6	Cahier Spirale GC	3

commande1

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
5	4	Gomme	25

commande2

IdC	IdP	Produit	Qte
4	3	Cahier A4 PC	1
6	6	Cahier Spirale GC	3

Fragmentation Verticale

- **Schéma global** : compte (numC, #idP, dcreation, typeC, IBAN)
  avec *typeC* ∈ { 'Premium', 'Base', 'Gold' }
- **Opérateur de fragmentation** : $\Pi_{<attributs>}(relation)$
  - ➔ Fragments définis par **projection**
  - ➔ *Contrainte* : la clé primaire doit appartenir à chaque fragment
- **Exemple :** *Fragmentation verticale de compte pour isoler l'IBAN*

$$compte1 = \Pi_{numC,\ IBAN}(compte)$$

$$compte2 = \Pi_{numC,\ idP,\ dCreation,\ typeC}(compte)$$

compte

numC	idP	dCreation	typeC	IBAN
59001	1	2022-09-23	Premium	FR123456
75003	2	2021-12-02	Base	FR654321
13005	3	2022-05-21	Base	FR415263
29012	4	2019-09-21	Premium	FR142536
06008	5	2022-04-25	Premium	FR613452
83198	6	2019-11-02	Base	FR164325

compte1

numC	IBAN
59001	FR123456
75003	FR654321
13005	FR415263
29012	FR142536
06008	FR613452
83198	FR164325

compte2

numC	idP	dCreation	typeC
59001	1	2022-09-23	Premium
75003	2	2021-12-02	Base
13005	3	2022-05-21	Base
29012	4	2019-09-21	Premium
06008	5	2022-04-25	Premium
83198	6	2019-11-02	Base

Fragmentation Verticale

- **Schémas locaux** : compte1 (numC, IBAN)
  compte2 (numC, #idp, dCreation, typeC)

- **Opérateur de reconstruction:** $\overset{\textbf{pk}}{\bowtie}_{i=1}^{n} Fragment_i$

  ➔ Reconstruction par **jointure**

- **Exemple :** *Reconstruction des comptes*

  compte = compte1 $\overset{\textbf{numC}}{\bowtie}$ compte2

Sans la clé primaire dans chaque fragment vertical, reconstruction impossible !

compte1' (numC, IBAN)

compte2' (#idp, dCreation, typeC)

compte

numC	idP	dCreation	typeC	IBAN
59001	1	2022-09-23	Premium	FR123456
75003	2	2021-12-02	Base	FR654321
13005	3	2022-05-21	Base	FR415263
29012	4	2019-09-21	Premium	FR142536
06008	5	2022-04-25	Premium	FR613452
83198	6	2019-11-02	Base	FR164325

compte1

numC	IBAN
59001	FR123456
75003	FR654321
13005	FR415263
29012	FR142536
06008	FR613452
83198	FR164325

compte2

numC	idP	dCreation	typeC
59001	1	2022-09-23	Premium
75003	2	2021-12-02	Base
13005	3	2022-05-21	Base
29012	4	2019-09-21	Premium
06008	5	2022-04-25	Premium
83198	6	2019-11-02	Base

Fragmentation Mixte

Fragments définis par combinaison de **sélection** et de **projection**

client

IdC	nom	prénom	Secteur	email	adresse	typeC
1	TARTINE	Kimberley	N	tk@toto.fr	12 Rue du P'tit dej …	Premium
2	DHONC	Medhi	N	dm@titi.fr	1 Av. des étonnés	Base
3	DIHIEAU	Kelly	S	dk@tata.fr	25 Rue de la Sottise	Premium
4	LAIZAUTRE	Pacôme	N	lp@toto.fr	NULL	Base
5	DABITUDE	Côme	S	dc@tutu.fr	2 Rue Myway	Base
6	HERE	Axel	S	ha@toto.fr	3 Rue des pressés	Premium

Exemple : *Fragmentation horizontale suivie d'une fragmentation verticale de chaque fragment*

$$\text{Client11} = \Pi_{\text{idc, nom, prénom, secteur}} (\sigma_{\text{secteur = 'N'}}(\text{client}))$$

$$\text{Client12} = \Pi_{\text{idc, email, adresse, typec}} (\sigma_{\text{secteur = 'N'}}(\text{client}))$$

$$\text{Client21} = \Pi_{\text{idc, nom, prénom, secteur}} (\sigma_{\text{secteur = 'S'}}(\text{client}))$$

$$\text{Client22} = \Pi_{\text{idc, email, adresse, typec}} (\sigma_{\text{secteur = 'S'}}(\text{client}))$$

Fragmentation Mixte

Fragments de l'exemple :

client

IdC	nom	prénom	Secteur	email	adresse	typeC
1	TARTINE	Kimberley	N	tk@toto.fr	12 Rue du P'tit dej …	Premium
2	DHONC	Medhi	N	dm@titi.fr	1 Av. des étonnés	Base
3	DIHEAU	Kelly	S	dk@tata.fr	25 Rue de la Sottise	Premium
4	LAIZAUTRE	Pacôme	N	lp@toto.fr	NULL	Base
5	DABITUDE	Côme	S	dc@tutu.fr	2 Rue Myway	Base
6	HERE	Axel	S	ha@toto.fr	3 Rue des pressés	Premium

Frag. horizontale

client1

IdC	nom	prénom	Secteur	email	adresse	typeC
1	TARTINE	Kimberley	N	tk@toto.fr	12 Rue du P'tit dej …	Premium
2	DHONC	Medhi	N	dm@titi.fr	1 Av. des étonnés	Base
4	LAIZAUTRE	Pacôme	N	lp@toto.fr	NULL	Base

Frag. verticale

client11

IdC	nom	prénom	Secteur
1	TARTINE	Kimberley	N
2	DHONC	Medhi	N
4	LAIZAUTRE	Pacôme	N

client12

IdC	email	adresse	typeC
1	tk@toto.fr	12 Rue du P'tit dej …	Premium
2	dm@titi.fr	1 Av. des étonnés	Base
4	lp@toto.fr	NULL	Base

Frag. horizontale

client2

IdC	nom	prénom	Secteur	email	adresse	typeC
3	DIHEAU	Kelly	S	dk@tata.fr	25 Rue de la Sottise	Premium
5	DABITUDE	Côme	S	dc@tutu.fr	2 Rue Myway	Base
6	HERE	Axel	S	ha@toto.fr	3 Rue des pressés	Premium

Frag. verticale

client21

IdC	nom	prénom	Secteur
3	DIHEAU	Kelly	S
5	DABITUDE	Côme	S
6	HERE	Axel	S

client22

IdC	email	adresse	typeC
3	dk@tata.fr	25 Rue de la Sottise	Premium
5	dc@tutu.fr	2 Rue Myway	Base
6	ha@toto.fr	3 Rue des pressés	Premium

Fragmentation correcte

Soit R=(A1,…,Ak) une relation globale. Soit $\mathcal{C}p$ l'ensemble des attributs composant la clé primaire de R

La fragmentation *f* telle que *f*(R) = $\{F_j\}$ est dite *correcte* si et seulement si elle est :

- **Complète**

  ➔ $\forall A_i \in R$, $\exists F_j \in f(R)$ tel que $A_i \in F_j$

  i.e., chaque élément de R doit se trouver dans un fragment

- **Reconstructible**

  ➔ $\exists h$ tel que $h(\{F_j\}) = R$

  i.e., on doit pouvoir recomposer R à partir de ses fragments

- **Disjointe**

  ➔ $\forall F_j , \forall F_k \in f(R) , F_j \cap F_k \neq \varnothing \implies F_j \cap F_k = \mathcal{C}p$

  i.e., chaque élément de R (hormis les clés) ne doit pas être dupliqué

Question

client

IdC	nom	prénom	Secteur	email	adresse	typeC
1	TARTINE	Kimberley	N	tk@toto.fr	12 Rue du P'tit dej …	Premium
2	DHONC	Medhi	N	dm@titi.fr	1 Av. des étonnés	Base
3	DIHEAU	Kelly	S	dk@tata.fr	25 Rue de la Sottise	Premium
4	LAIZAUTRE	Pacôme	N	lp@toto.fr	NULL	Base
5	DABITUDE	Côme	S	dc@tutu.fr	2 Rue Myway	Base
6	HERE	Axel	S	ha@toto.fr	3 Rue des pressés	Gold

Les fragmentations suivantes sont-elles correctes ?

Fragemtation 1: Non complète (gold?)

$$Client1 = \sigma_{typeC = \text{'Premium'}}(client) \; ; \quad Client2 = \sigma_{typeC = \text{'Base'}}(client)$$

Fragemtation 2: Non reconstructible (clé dans client2 ?)

$$Client1 = \Pi_{idc,\ nom,\ prénom,\ secteur}(client) \; ; \; Client2 = \Pi_{nom,\ email,\ adresse,\ typec}(client)$$

Fragemtation 3: Non disjointe (clients 'Premium' du secteur 'S' dupliqués)

$$Client1 = \sigma_{secteur = \text{'N'} \vee typeC = \text{'Premium'}}(client) \; ; \quad Client2 = \sigma_{secteur = \text{'S'}}(client)$$

Retour sur l'exemple initial

personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

commande

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
4	3	Cahier A4 PC	1
5	4	Gomme	25
6	6	Cahier Spirale GC	3

Distribution des données sur plusieurs sites

personne@Site1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

personne@Site2

IdP	NOM	Prénom	Secteur
3	NAULT	Pia	S
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

commande@Site3

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
5	4	Gomme	25

commande@Site4

IdC	IdP	Produit	Qte
4	3	Cahier A4 PC	1
6	6	Cahier Spirale GC	3

Retour sur l'exemple initial

personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

commande

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
4	3	Cahier A4 PC	1
5	4	Gomme	25
6	6	Cahier Spirale GC	3

$$personne1 = \sigma_{secteur = 'N'}(Personne)$$

$$Personne2 = \sigma_{secteur = 'S'}(Personne)$$

$$commande1 = commande \underset{idP}{\bowtie} personne1$$

$$commande2 = commande \underset{idP}{\bowtie} personne2$$

personne@Site1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

personne@Site2

IdP	NOM	Prénom	Secteur
3	NAULT	Pia	S
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

commande@Site3

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
5	4	Gomme	25

commande@Site4

IdC	IdP	Produit	Qte
4	3	Cahier A4 PC	1
6	6	Cahier Spirale GC	3

Création d'un fragment

Concrètement avec PostgreSQL

personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

Traitement de personne1

$$personne1 = \sigma_{secteur = 'N'}(Personne)$$

personne@Site1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

1. Déclaration du serveur bd0 sur bd1
   ➔ *CREATE SERVER servBD0 FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host '192.168.1.**' port '5432' dbname 'bd0');*
2. Déclaration de l'utilisateur distant sur bd1
   ➔ *CREATE USER MAPPING FOR 'userloc' SERVER servBD0 OPTIONS(user 'userdist' password '******');*
3. Création de la table distante personne sur bd1
   ➔ *CREATE FOREIGN TABLE remote_personne(idP integer,….) SERVER servBD0;*
4. Création du fragment local personne sur bd1
   ➔ *CREATE TABLE personne SELECT * FROM remote_personne WHERE secteur = 'N';*
5. Suppression de la table distante sur bd1
   ➔ *DROP FOREIGN TABLE remote_personne;*

Création de la vue globale

Concrètement avec PostgreSQL

personne1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

personne2

IdP	NOM	Prénom	Secteur
3	NAULT	Pia	S
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

Traitement de la reconstruction de personne

`personne = Personne1 U personne2`

v_personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

1. Déclaration des serveurs bd1 et bd2 sur bd0
   - ➔ *CREATE SERVER servBD1 FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host '192.168.1.**' port '5432' dbname 'bd1');*
   - ➔ *CREATE SERVER servBD2 FOREIGN DATA WRAPPER postgres_fdw OPTIONS (host '192.168.1.**' port '5432' dbname 'bd2');*
2. Déclaration de l'utilisateur distant sur bd0
   - ➔ *CREATE USER MAPPING FOR 'userloc' SERVER servBD1 OPTIONS(…);*
   - ➔ *CREATE USER MAPPING FOR 'userloc' SERVER servBD2 OPTIONS(…);*
3. Création des tables distantes personne1 et personne2 sur bd0
   - ➔ *CREATE FOREIGN TABLE remote_personne1(idP integer,….) SERVER servBD1;*
   - ➔ *CREATE FOREIGN TABLE remote_personne2(idP integer,….) SERVER servBD2;*
4. Création de la vue globale v_personne sur bd0
   - ➔ *CREATE VIEW v_personne AS SELECT * FROM remote_personne1*

Quid des contraintes d'intégrité ?

- Pas de possibilité d'exprimer des contraintes globales (trop dépendant des connexions réseaux)
  - Nécessité de définir des fonctions / triggers permettant de s'assurer que l'intégrité est conservée à l'ajout, la mise à jour et la suppression de tuples

Application

Exercices applicatifs:

➔ BDR : Fragmentation d'une base de données

TP:

➔ Déploiement d'une base de données répartie

En résumé

Dans le cas où vous avez besoin de distribuer vos données sur différents sites, vous disposez des outils pour réaliser la fragmentation d'une base de données réparties

Quel impact sur le traitement des requêtes ?

A propos des requêtes

Exemple

Nom et prénom des personnes ayant au moins une commande d'un produit en plus de 10 exemplaires

```
SELECT p.nom, p.prénom
FROM personne p
  JOIN commande c  ON p.idP = c.idP
WHERE c.qte > 10;
```

personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

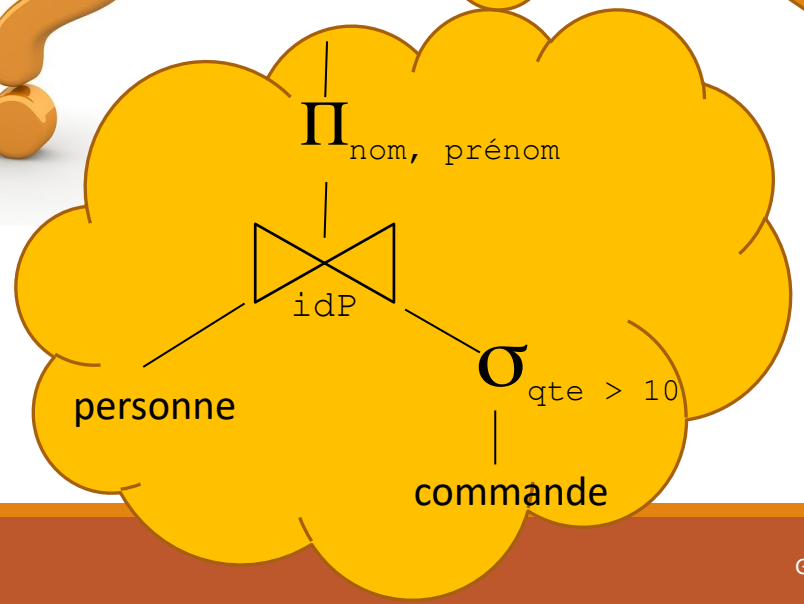
commande

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
4	3	Cahier A4 PC	1
5	4	Gomme	25
6	6	Cahier Spirale GC	3

LOGIQUE

A propos des requêtes

```
SELECT p.nom, p.prénom
FROM personne p
  JOIN commande c  ON p.idP = c.idP
WHERE c.qte > 10;
```



Traitement avec jointure inter-site

personne@Site1

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
4	THARE	Guy	N

personne@Site2

IdP	NOM	Prénom	Secteur
3	NAULT	Pia	S
5	HONETTE	Marie	S
6	GNOLLE	Guy	S

commande@Site3

IdC	IdP	Produit	Qte
1	1	Classeur	10
2	1	Ciseaux	5
3	2	Calculatrice	2
5	4	Gomme	25

commande@Site4

IdC	IdP	Produit	Qte
4	3	Cahier A4 PC	1
6	6	Cahier Spirale GC	3

Traitement de jointures inter-site

Jointures intersites

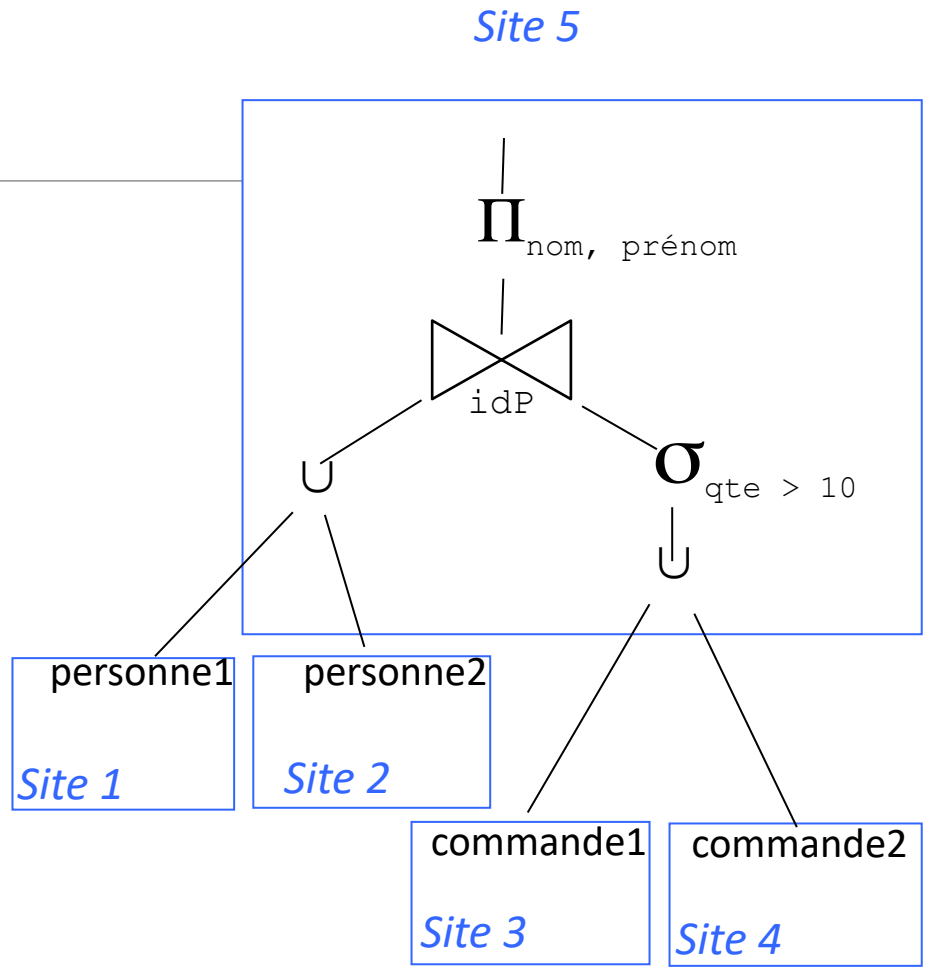
- C’est une jointure interrogeant des données distribuées sur différents sites
  - ➔ Coût de transfert des données dans la résolution des requêtes

Coût de transfert dépendant du plan d’exécution !

Un plan d'exécution possible pour une jointure inter-site

Plan d'exécution P1

- Transfert de personne1 du Site 1 vers le Site 5
- Transfert de personne2 du Site 2 vers le Site 5
- Transfert de commande1 du Site 3 vers le Site 5
- Transfert de commande2 du Site 4 vers le Site 5
- Sur site 5: Union de personne1 et personne2 ➔ on obtient T1
- Sur site 5: Union de commande1 et commande2 ➔ on obtient T2
- Sur site 5: Sélection sur T2 des n-uplets dont la valeur de qte > 10 : on obtient T3
- Sur site 5: Jointure entre T1 et T3
- Sur site 5: Projection sur le nom et le prénom ➔ on obtient le résultat



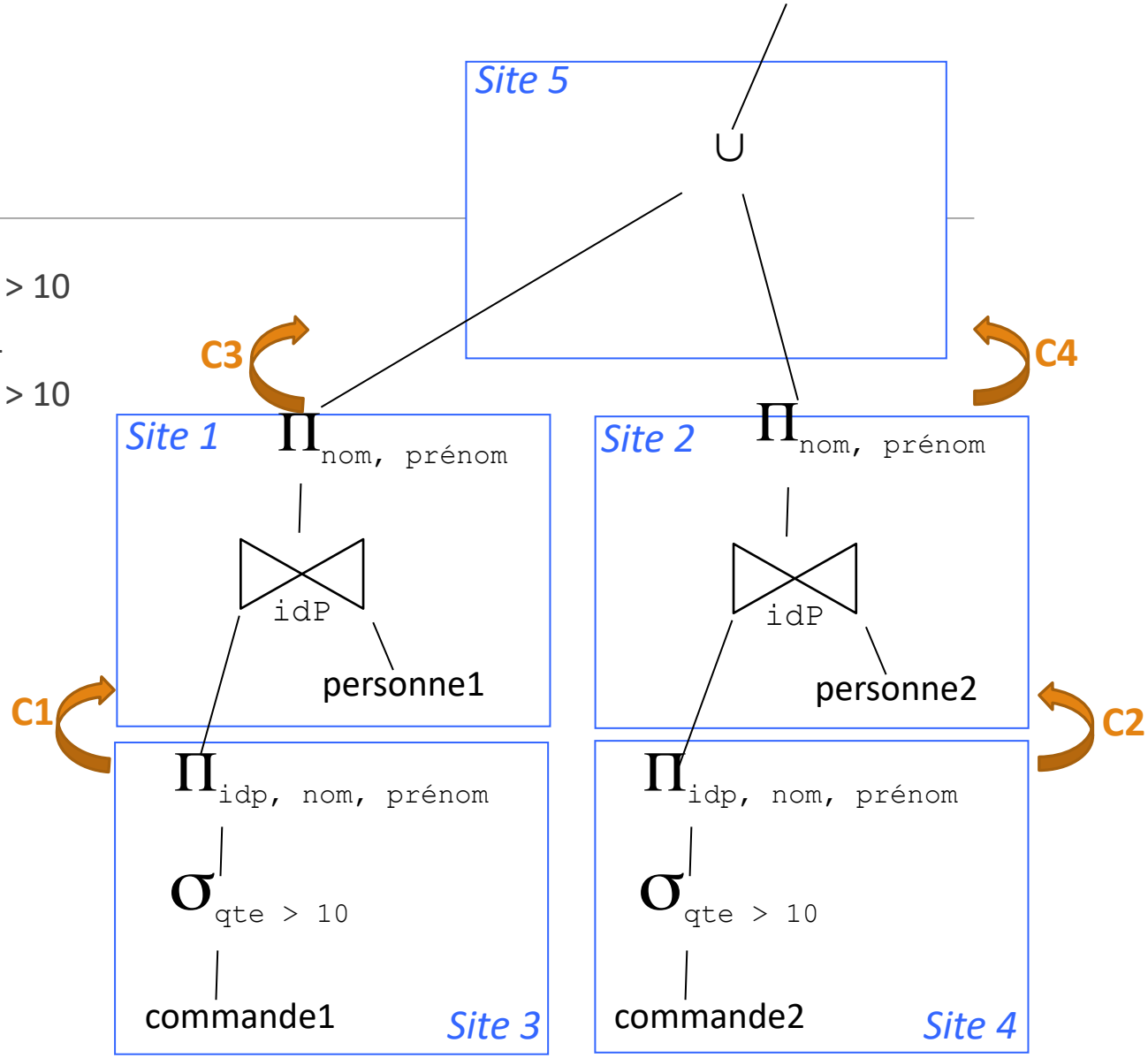
stratégie

Celui qui demande, récupère tout et c'est lui qui fait les calculs.

Un autre plan d'exécution possible pour une jointure inter-site

Plan d'exécution P2

- Sur site 3 : Sélection des commande de commande1 de quantité > 10
- Sur site 3 : Projection sur idp, npm et prénom ➔ on obtient C1
- Sur site 4 : Sélection des commande de commande2 de quantité > 10
- Sur site 4 : Projection sur idp, nom prénom ➔ on obtient C2
- Transfert de C1 du Site 3 vers le Site 1
- Transfert de C2 du Site 4 vers le Site 2
- Sur site 1: Jointure de C1 avec personne1
- Sur site 1 : projection sur nom et prénom ➔ on obtient C3
- Sur site 2: Jointure de C2 avec personne2
- Sur site 2 : Projection sur nom et prénom ➔ on obtient C4
- Transfert de C3 du Site 1 vers le Site 5
- Transfert de C4 du Site 2 vers le Site 5
- Sur site 5: Union de C3 avec C4 ➔ on obtient le résultat



stratégie **Collaborative**

Comparatif des transferts de n-uplets

Supposons
- |commande1| = |commande2 | = 10 000 n-uplets
- |personne1 | = |personne2| = 2 000 n-uplets
- Coût de transfert de 1 n-uplet = 1
- Facteurs de sélectivité de l'attribut *qte*

qte = 1	qte > 1	qte > 5	qte > 10	qte > 20
80 %	20%	5%	1%	0,2%

Plan d'exécution P1
- Transfert de (commande1 + commande2) = 10 000 + 10 000 = 20 000 n-uplets
- Transfert de (personne1 + personne2) = 2 000 + 2 000 = 4 000 n-uplets

} 24 000 n-uplets

x30

Plan d'exécution P2
- Transfert de C1 + C2 = (0,01x2 000) + (0,01x2 000) = 400 n-uplets
- Transfert de C3 + C4 = 400 n-uplets

} 800 n-uplets

car jointure sur clé étrangère (ce qui ne change pas le nombre mais la taille des n-uplets)

Comparatif du coût des transferts

Supposons

- |commande1| = |commande2 | = 10 000 n-uplets
- |personne1 | = |personne2| = 2 000 n-uplets
- Un attribut codé sur 1 Ko
- Coût de transfert de 1 n-uplet = 1
- Facteurs de sélectivité de l'attribut qte

qte = 1	qte > 1	qte > 5	qte > 10	qte > 20
80 %	20%	5%	1%	0,2%

Plan d'exécution 1

- Transfert de personne1 + personne2 = 10 000 x (4 x 1Ko) + 10 000 x (4 x 1Ko) = 80 000 Ko
- Transfert de commande1 + commande2 = 2 000 (4 x 1Ko) + 2 000 (4 x 1Ko) = 16 000 Ko

93,75 Mo

x48

Plan d'exécution 2

- Transfert de C1 + C2 = 200 x (3 x 1Ko) + 200 x (3 x 1Ko) = 1 200 Ko
- Transfert de C3 + C4 = 200 x (2 x 1Ko) + 200 x (2 x 1Ko) = 800 Ko

1,95 Mo

Exercices applicatifs

Exercices applicatifs

- BDR : Evaluation de requêtes réparties

Moralité

« Si tu veux que ta jointure inter-site aboutisse en un temps raisonnable,

il vaut mieux réduire les échanges de données entre les sites »

Comment orchestrer les échanges de données pour le traitement des jointures inter-site ?

Quel algorithme de jointure choisir ?

Comment améliorer les performances de traitement ?

- Technique d'exécution distribuée
- Orchestration du traitement des requêtes
- Choix d'algorithme pour les jointures intersites
- Paramétrage du SGBD
- Réplication logique des données
- Sharding

Techniques d'exécution distribuée

En tenant compte des propriétés du réseau

- Row blocking : transfert de données par batch plutôt qu'individuellement
- Multicast : dans le cas de transfert des mêmes données sur plusieurs sites, valoriser les chemins les moins coûteux
- Multithreading: parallélisation de certains opérateurs
  - Problème de communication synchronisée inter-thread

Orchestration d'une jointure inter-site

- Principe de base:
  - Transférer la plus petite des deux relations (si on n'a pas le choix)
- Solutions possibles
  - Algorithme des semi-jointures
  - Jointures parallèles
    - Jointure parallèle avec partitionnement des données sur attribut de jointure
    - Jointure parallèle sans partitionnement des données sur attribut de jointure

Algorithme des semi-jointures

- Objectif:
  - réduire la quantité d'information transférée pour exécuter une jointure quand peu de n-uplets participent à la jointure

- Constat:

  Soient R(A, B, C, D) sur Site 1 et S(A, D, E, F) sur Site2

  On a:

$$R \underset{A}{\bowtie} S \Leftrightarrow (R \underset{A}{\bowtie} \Pi_A(S)) \underset{A}{\bowtie} S \Leftrightarrow R \underset{A}{\bowtie} (S \underset{A}{\bowtie} \Pi_A(R))$$

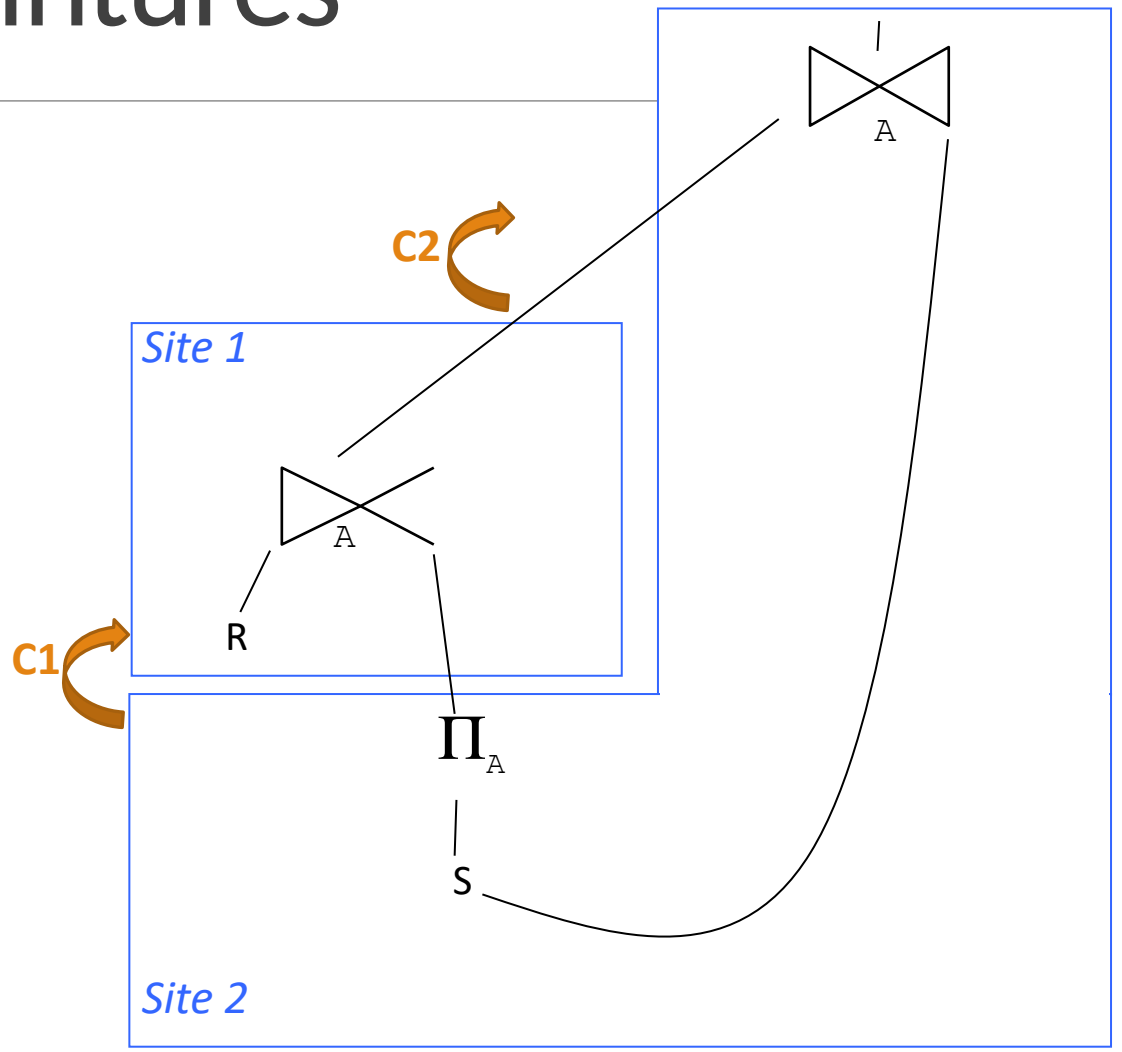
isole les n-uplets de R
qui joignent avec S

Algorithme des semis-jointures

Plan d'exécution pour exécuter la jointure de R avec S sur le site 2

- Sur site 2 : Projection de S sur A ➔ on obtient C1
- Transfert de C1 du site 2 vers le site 1
- Sur site 1 : Calcul de la semi-jointure $R \ltimes_A \Pi_A(S)$ ➔ on obtient C2
- Transfert de C2 du site 1 vers le site 2
- Sur site 2 : Calcul de la jointure C2 $\bowtie_A$ S ➔ on obtient le résultat

Remarque :
l'algorithme est particulièrement intéressant quand la jointure à un facteur de sélectivité élevé (peu de valeurs de A dans R correspondent à des valeurs de A dans S)



Question

Quel différence entre ces deux requêtes SQL si est R est sur le Site 1 et S sur le Site 2 et que les requêtes sont exécutées sur le Site 2?

```
SELECT *
FROM R JOIN S ON R.A = S.A;
```

Je laisse l'optimiseur choisir

```
SELECT *
FROM (SELECT R.*
        FROM R
          JOIN (SELECT A FROM S) P
          ON R.A = P.A
        ) T
        JOIN S ON T.A = S.A;
```

J'influence l'optimiseur

Jointures parallèles

- Contexte:
  - Partitionnement des données (fragmentation horizontale + horizontale dérivée)
- Objectif:
  - Distribuer le traitement d'une jointure sur plusieurs machines, en tenant compte du partitionnement

- Constat

  Soient R(A, B, C, D) sur Site 1 et S(A, D, E, F) sur Site2.

  Soit $A = \{A_1, A_2\}$ une partition du domaine de définition de l'attribut A

  On notera Ri les tuples de R pour lesquels A $\in A_i$

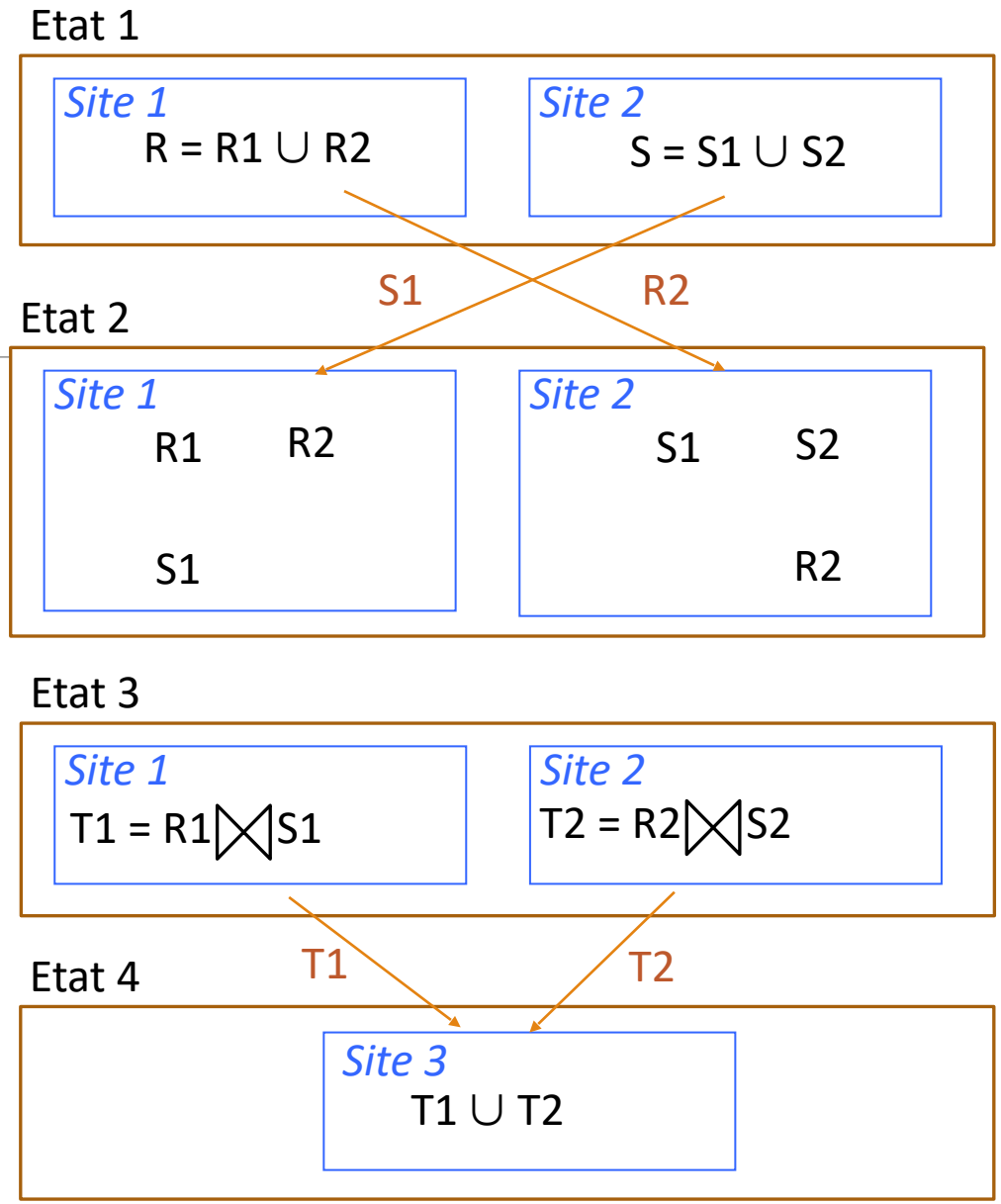
  On a:

$$R \bowtie_A S \Leftrightarrow R1 \bowtie_A S1 \cup R2 \bowtie_A S2$$

Jointures parallèles

Plan d'exécution pour une requête initiée par site 3

- Transfert de R1 sur Site 2
- Transfert de S2 sur Site 1.
- Faire jointure de R1 et S1 sur Site 1 ➔ on obtient T1
- Faire jointure de S2 et R2 sur Site 2 ➔ on obtient T2
- Transfert de T1 sur site 3
- Transfert de T2 sur site 3
- Fusion de T1 et T2 sur site 3



Jointures parallèles

- Contexte
  - Pas de partitionnement des données
- Objectif:
  - Eviter de rassembler l'ensemble des fragments horizontaux sur un même nœud.

- Constat
  - Soient :
    - R(A, B, C, D) fragmenté horizontalement en R1 et R2, respectivement sur Site 1 et Site 2 ➔ $R = R1 \cup R2$
    - S(A, D, E, F) fragmenté horizontalement en S1 et S2, respectivement sur Site 3 et Site 4 ➔ $S = S3 \cup S4$

On a:

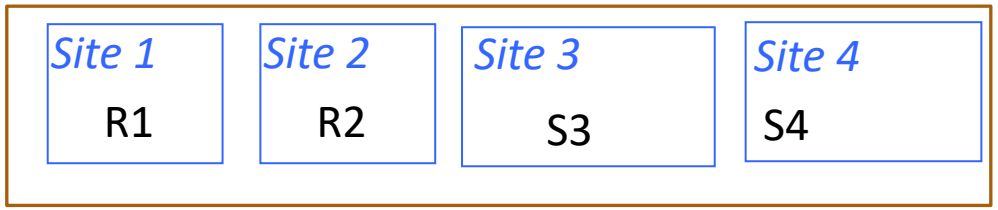
$$R \bowtie_A S \Leftrightarrow (R1 \cup R2) \bowtie_A (S3 \cup S4) \Leftrightarrow ((R1 \cup R2) \bowtie_A S3) \cup ((R1 \cup R2) \bowtie_A S4$$

Jointures parallèles

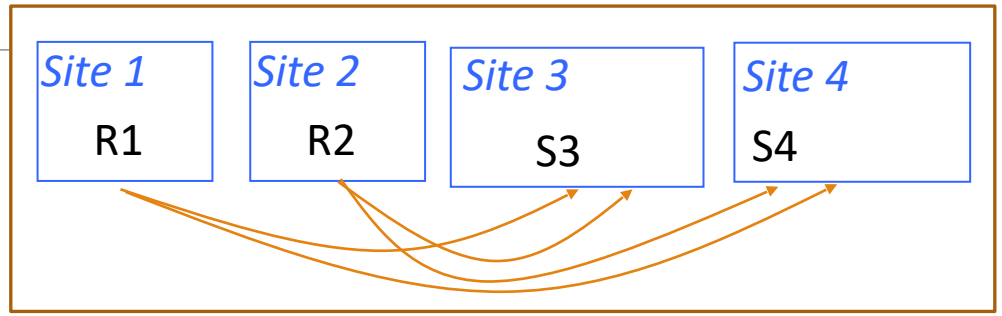
Plan d'exécution de la requête initiée par site 4

- Transférer R1 sur Site 3 et Site 4
- Transférer R2 sur Site 3 et Site 4
- Faire l'union de R1 et de R2 sur Site 3 puis faire la jointure avec S3 ➔ on obtient T1
- Faire l'union de R1 et de R2 sur Site 4 puis faire la jointure avec S4 ➔ on obtient T2
- Transférer T2 sur Site 2
- Faire union de T1 et T2 ➔ on obtient le résultat

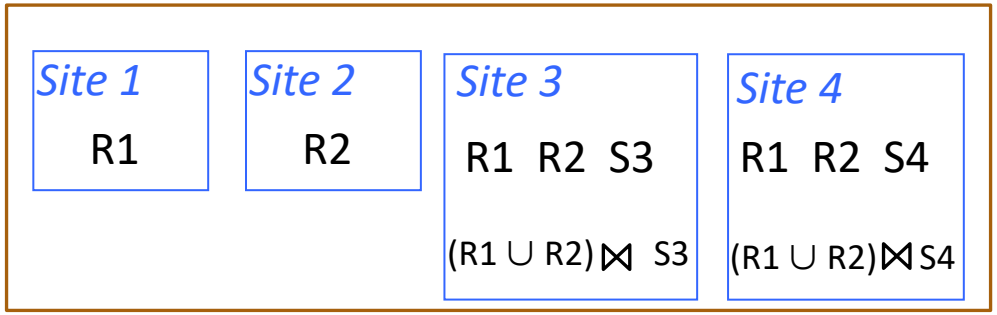
Etat 1



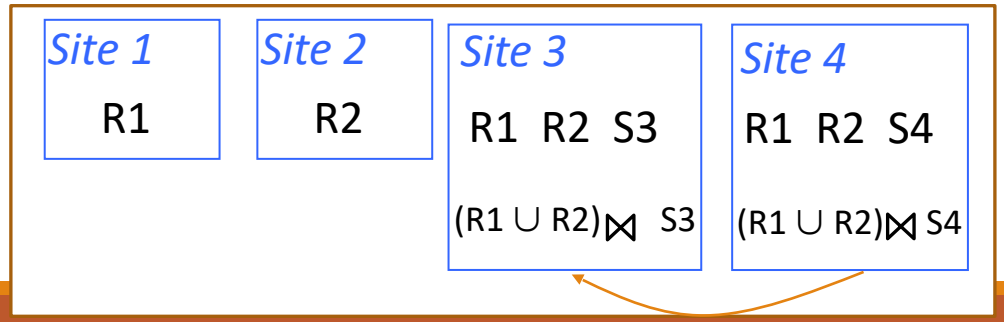
Etat 2



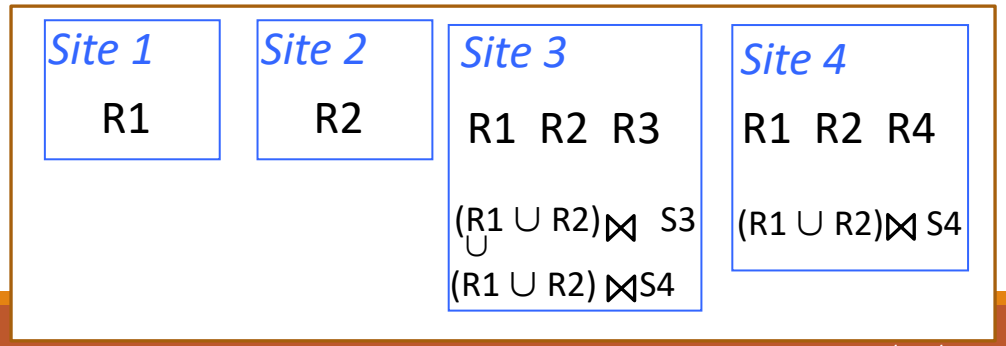
Etat 3



Etat 4



Etat 5



Et au niveau algorithme de jointure?

Algorithmes de jointures : RAPPELS

- **En centralisé**
  - Approche par boucle (LOOP)
    - ➔ Algorithme des boucles imbriquées (avec ou sans index)
  - Approche par tri-fusion (SORT-MERGE)
    - ➔ Algorithme du *sort-merge join*
  - Approche par hachage (HASH)
    - ➔ Algorithme du hash join

Pour chaque tuple de R, on parcourt S

On trie R et S sur les attributs de jointure, puis scan en parallèle pour permettre un seul parcours de chaque relation

Transfert de la plus petite des relations* dans une structure de hachage avec les attributs de jointure comme clé, pour accéder efficacement aux valeurs

* Dans postgresql, il s'agit de la relation de droite

Sort-merge distribué

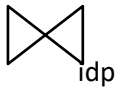
- Partitionnement des données en amont du tri
  - Facilite la parallélisation
  - Généralement, nécessite la matérialisation les fragments ➔ latence supplémentaire
- Plusieurs niveau de fusion en aval du tri

➢ différentes implémentations basées sur des approches de tri (réseaux de tri…) et des stratégies de fusions (M-Way, M-PASS, Massively Parallel)

Exemple de Sort-Merge Parallèle

compte

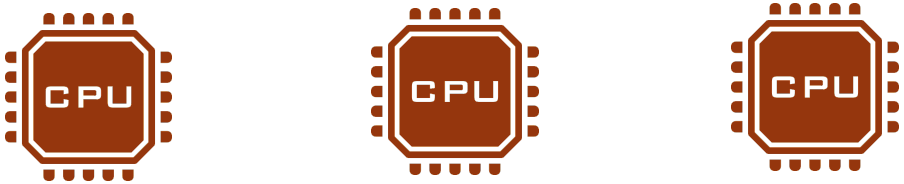
numC	idP	dCreation	typeC	IBAN
59001	10	2022-09-23	Premium	FR123456
…	…	…	…	…
75003	52	2021-12-02	Base	FR654321
13005	31	2022-05-21	Base	FR415263
…	…	…	…	…
29012	24	2019-09-21	Premium	FR142536
06008	57	2022-04-25	Premium	FR613452
…	…	…	…	…
83198	16	2019-11-02	Base	FR164325



personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S
…	….	…	…

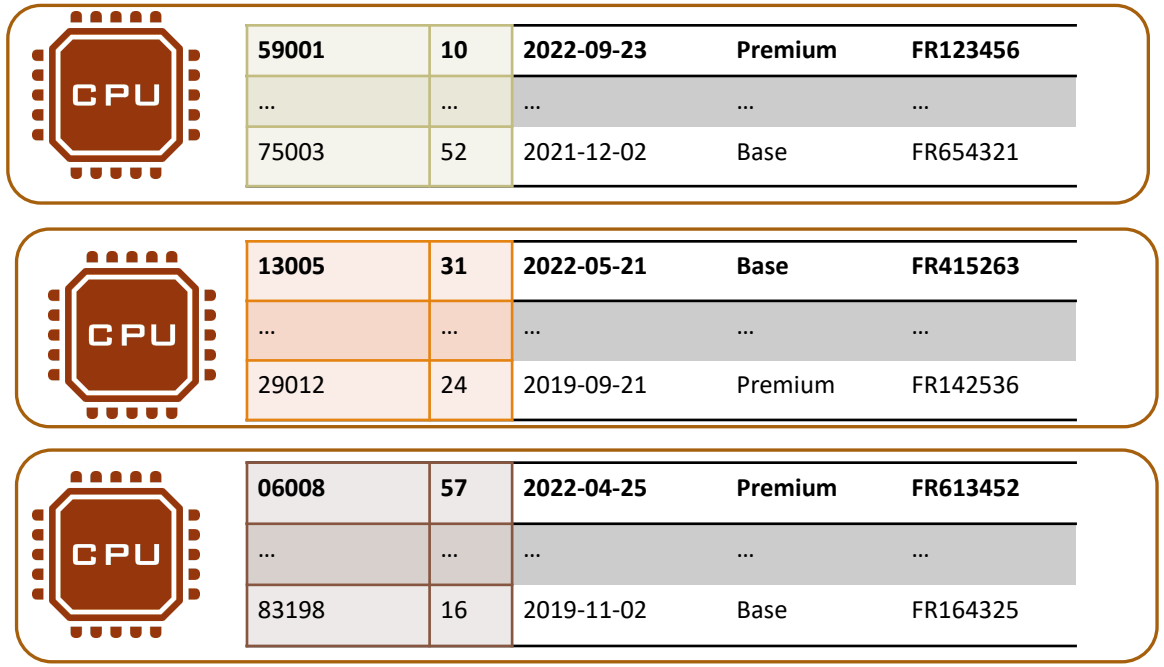
Sur :



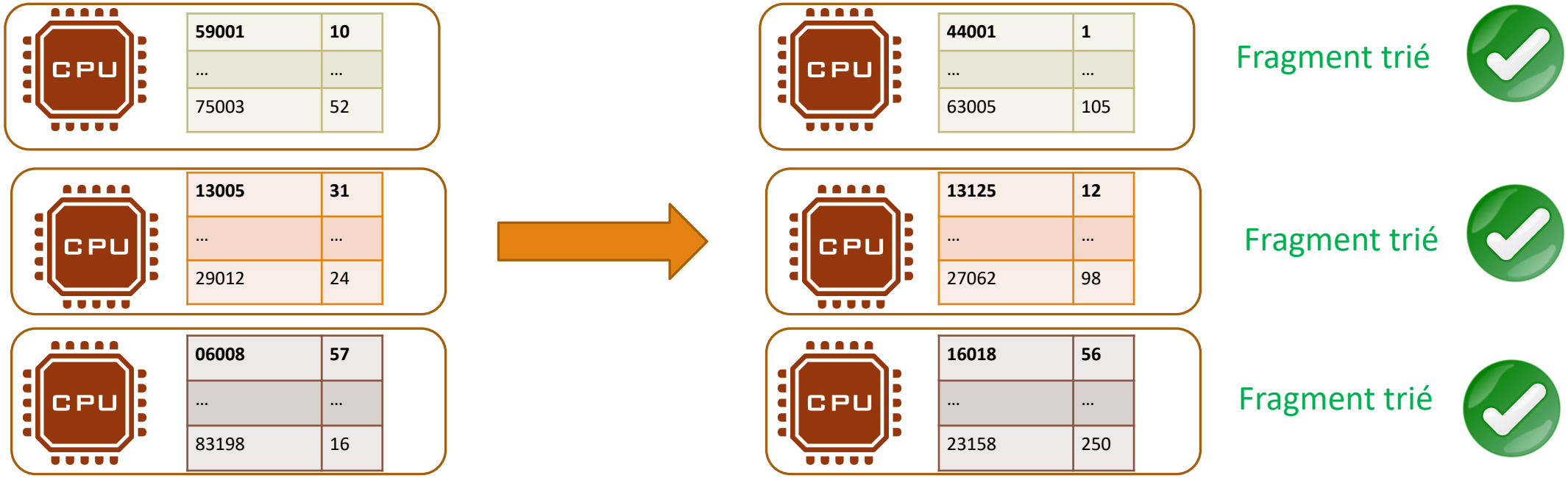
Exemple de Sort-Merge Parallèle : 1. Partitionnement

compte

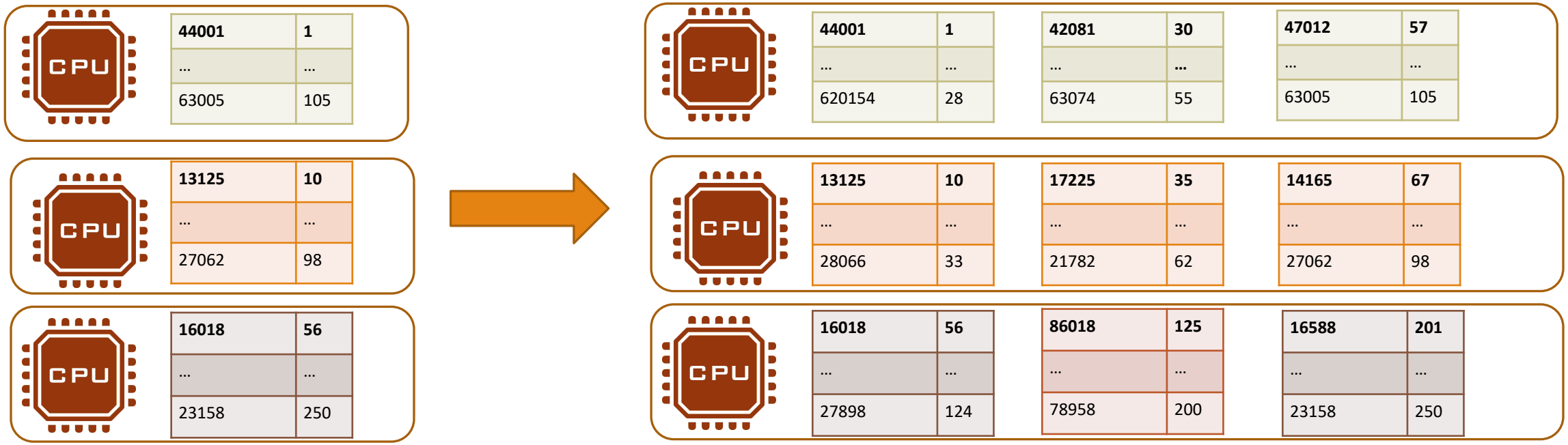
numC	idP	dCreation	typeC	IBAN
59001	10	2022-09-23	Premium	FR123456
…	…	…	…	…
75003	52	2021-12-02	Base	FR654321
13005	31	2022-05-21	Base	FR415263
…	…	…	…	…
29012	24	2019-09-21	Premium	FR142536
06008	57	2022-04-25	Premium	FR613452
…	…	…	…	…
83198	16	2019-11-02	Base	FR164325



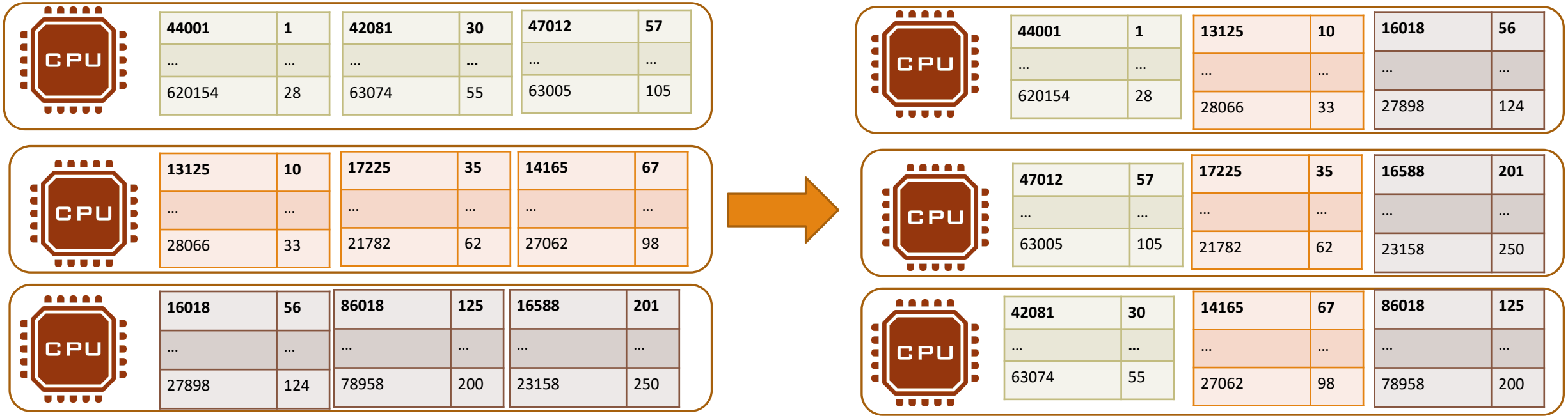
Exemple de Sort-Merge Parallèle : 2. Tri local



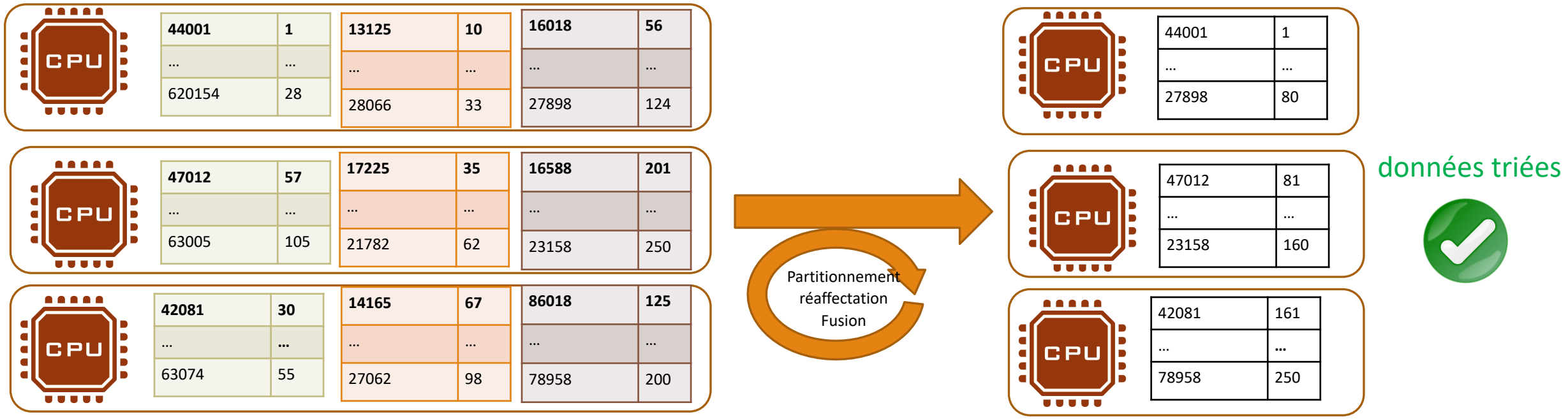
Exemple de Sort-Merge Parallèle : 3. Partitionnement



Exemple de Sort-Merge Parallèle : 4. Réaffectation



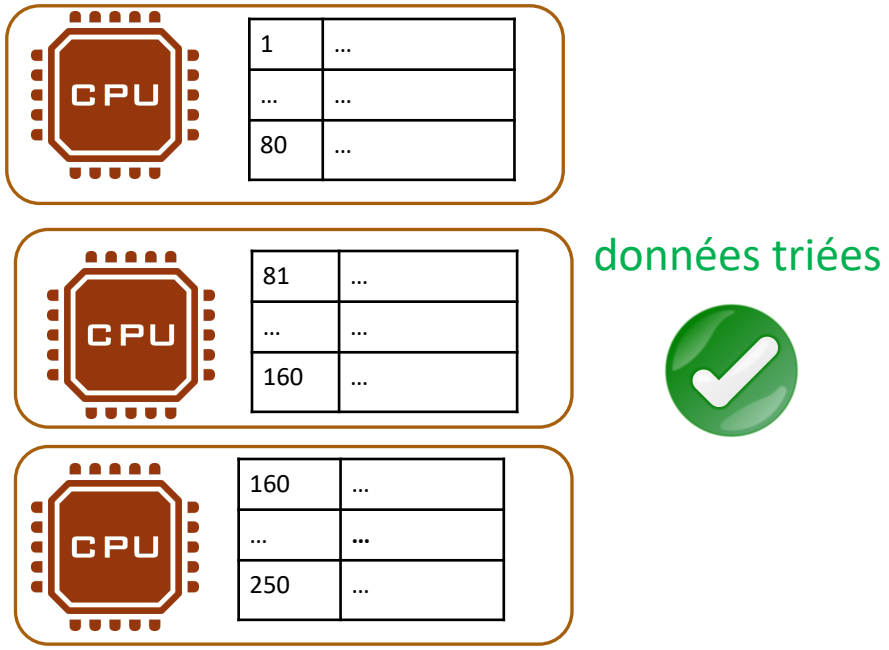
Exemple de Sort-Merge Parallèle : 5. Fusion



Exemple de Sort-Merge Parallèle : 6. tri de l'autre relation

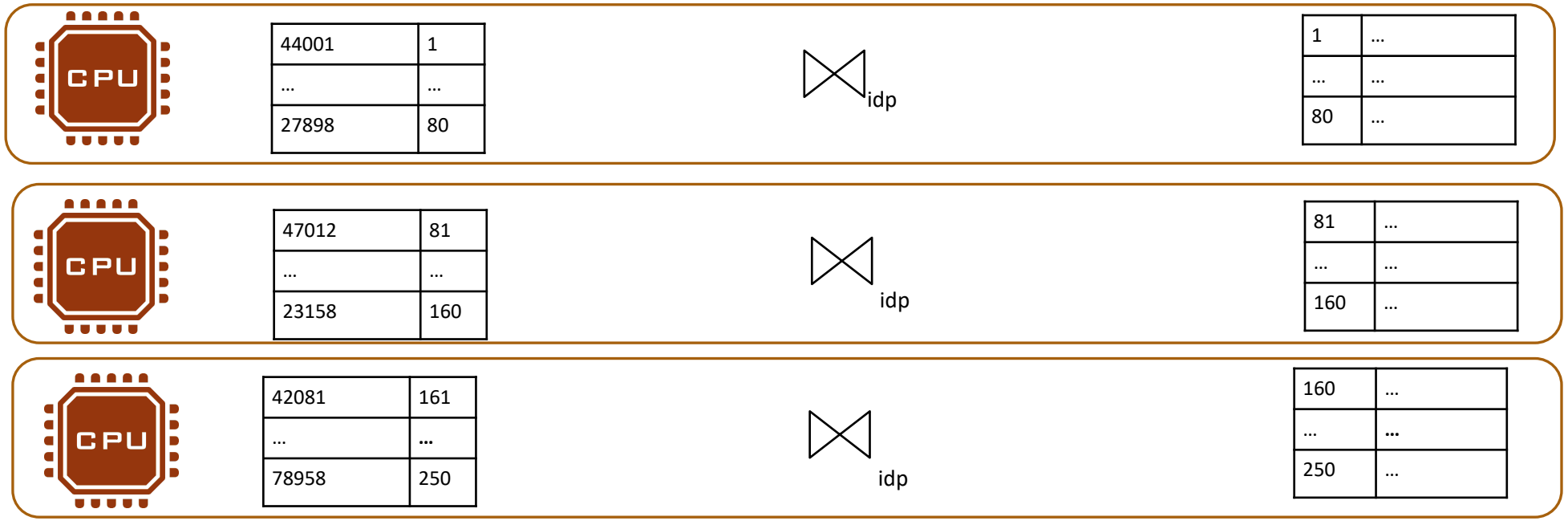
personne

IdP	NOM	Prénom	Secteur
1	BON	Jean	N
2	HETTE	Rose	N
3	NAULT	Pia	S
4	THARE	Guy	N
5	HONETTE	Marie	S
6	GNOLLE	Guy	S
…	….	…	…



données triées

Exemple de Sort-Merge Parallèle : 7. Jointure



Hash-join distribué

- Partitionnement des données en amont
  - Facilite la parallélisation
  - Création de plusieurs structure de hachage

  - Différentes implémentations basées sur les types de partitionnement (Partition partagée ou privée, RADIX…) et de structuration des table de hachage (hachage chaîné, adressage ouvert…)

Hash-Join versus Sort-Merge Join

Le débat est encore ouvert !

Evolution des architectures

Evolution des réseaux

Evolution des processeurs

Réplication logique

- Solution pour éviter des transferts est la réplication logique des certaines données
  - Rapprocher les données des besoins

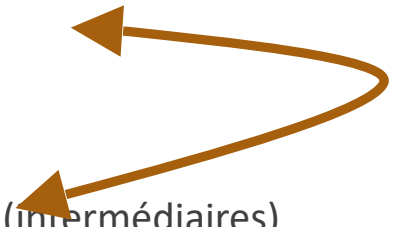
- Important :
  - Prendre en compte a fréquence de mise à jour des données répliquées

Réécriture de requête à partir des fragments

- Certaines requêtes n'impliquent pas forcément tous les fragments théoriquement utiles pour l'exécution de la requête
  - Appliquer des simplifications pour réduire les échanges inter-sites. (Cf exercices applicatifs 1)

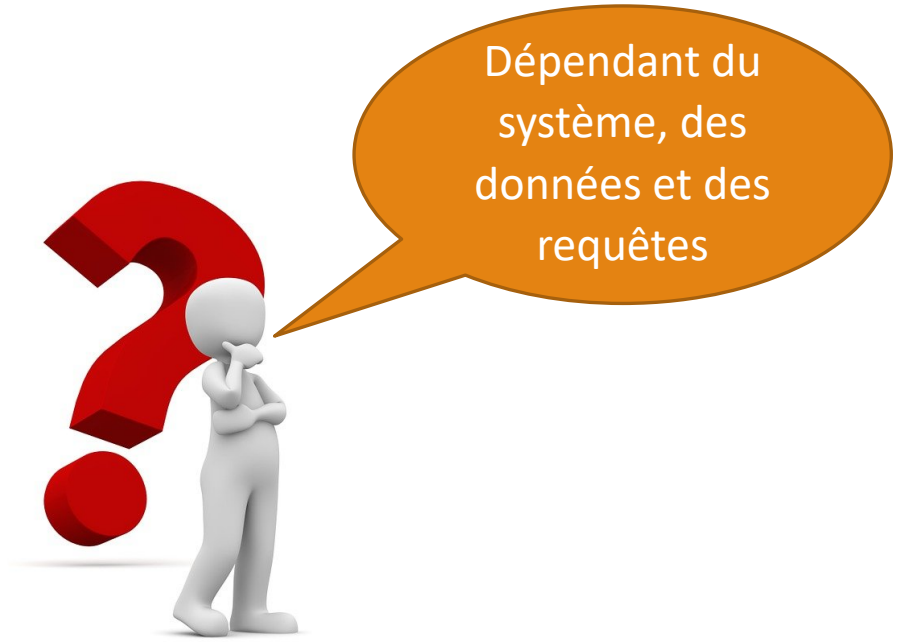
Ce qui peut aussi impacter

- Au niveau configuration
  - La taille des buffers de manipulation des données partagés
  - La taille des buffers de maintient de l'intégrité des données
  - La taille du cache
  - La mémoire utilisable pour le traitement d'opérateurs complexes tels que les opérateurs de tris
  - La mémoire utilisable pour les processus de maintenance (libération de mémoire, gestion index…)
  - Le nombre de connexions simultanées
  - La gestion du verrous dans le modèle transactionnel (en cas d'un niveau de concurrence élevée)
  - Le coût des accès disques
- Au niveau des données
  - L'exploitation des contraintes d'intégrités
  - Le partitionnement des tables
- Au niveau des requêtes
  - L'utilisation d'index
  - La parallélisation des calculs
  - La matérialisation de résultats (intermédiaires)



La meilleure configuration sous PostgreSQL

- Le nombre de connexions simultanées
  max_connections =
- La taille des buffers de manipulation des données partagés
  ➔ *shared_buffers* =
- La taille des buffers de maintient de l'intégrité des données
  ➔ *wal_buffers* =
- La taille du cache
  ➔ *effective_cache_size* =
- La mémoire utilisable pour le traitement d'opérateurs complexes
  ➔ *work_mem* =
- La mémoire utilisable pour les processus de maintenance
  ➔ *maintenance_work_mem* =

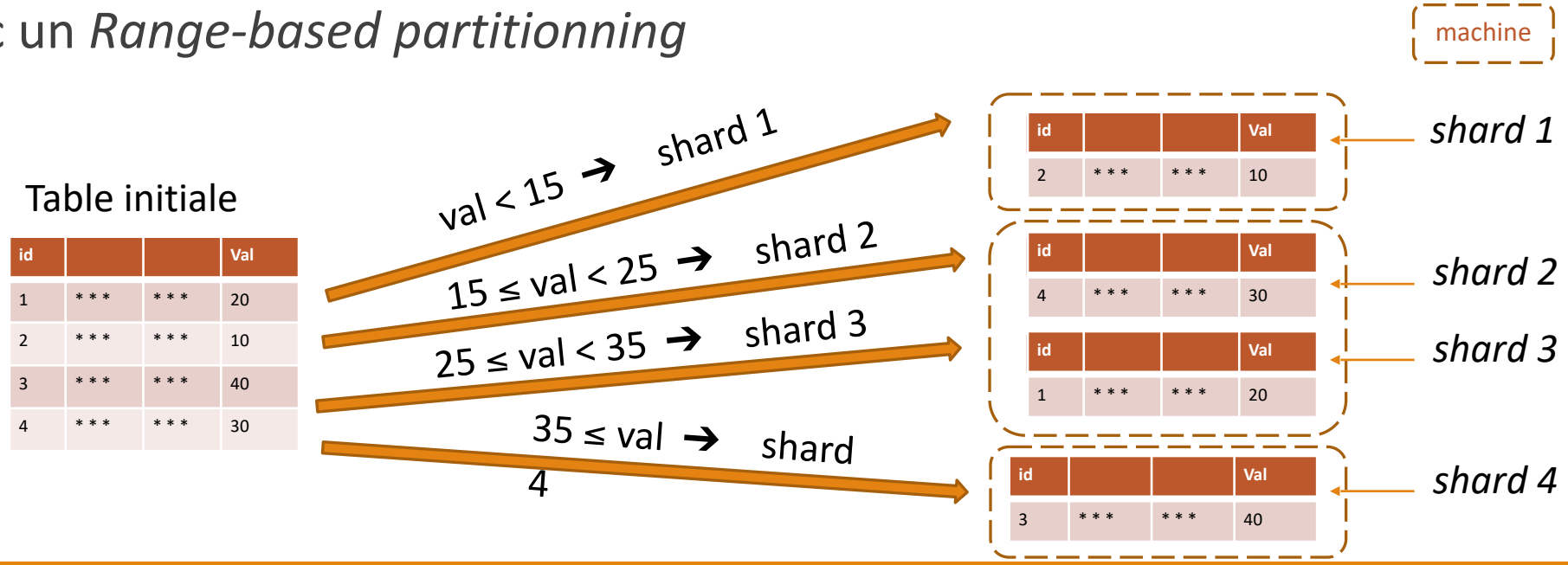


Partitionnement et parallélisation : Le ‘sharding’

- Technique du ‘Sharding’
  - Principe:
    - Fragmentation horizontale garantissant un partitionnement des données pour un traitement en parallèle des fragment
      - Choix de fragmentation :
        - Basé sur des plages de valeur de la clé de ‘sharding’ (Range-based partionning)
        - Basé sur une valeur de hachage de la clé de ‘sharding’ (Hash partionnong)
          - Ex : numero_shard = hash(id) % nombre_shard

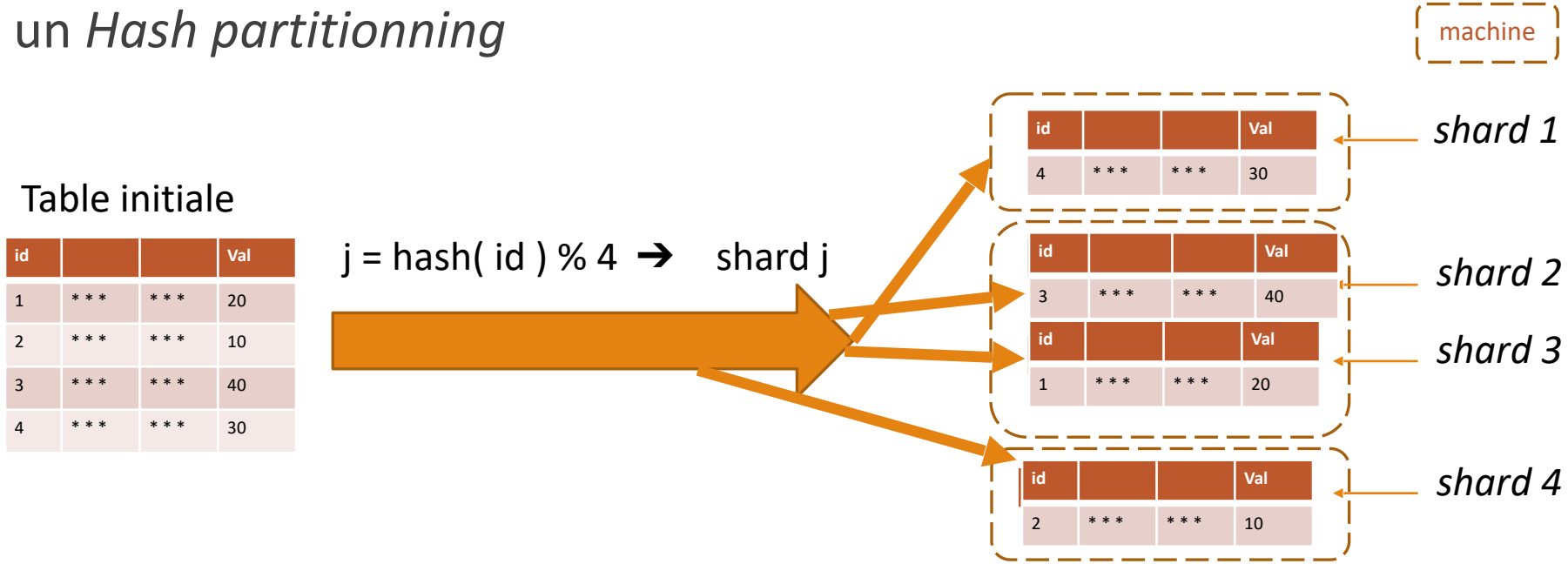
Partitionnement et parallélisation : Le ‘sharding’

- Mise en place du ‘sharding’ en deux étapes
  - Allocation des shards aux machines
    - Plusieurs *shards* possibles par machine
  - Affectation des tuples à leur shard
- Exemple avec un *Range-based partitionning*



Partitionnement et parallélisation : Le ‘sharding’

- Mise en place du ‘sharding’ en deux étapes
  - Allocation des shards aux machines
    - Plusieurs *shards* possibles par machine
  - Affectation des tuples à leur shard
- Exemple avec un *Hash partitionning*



Partitionnement et parallélisation : Le ‘sharding’

- Avantages:
  - Gain de temps de traitement
    - Traitement en parallèle (sur différentes machine CPU, RAM) des ‘shard’

      ➔ *Possibilités d’adapter la voilure de l’infrastructure (nombre de CPU et RAM) en fonction de la quantité de données à traiter*
  - Evite la centralisation des données sur un seul et unique serveur (sécurité)

- Inconvénients
  - Une mauvaise stratégie de ‘sharding’ (choix de la clé de sharding) peuvent rendre contre productif le processus à cause de ‘shard’ de taille trop hétérogène
    - Rééquilibrage complexe
  - Sensible à la performance des réseaux
  - Sensible à la disponibilité et l’évolution du cluster de machine

Application

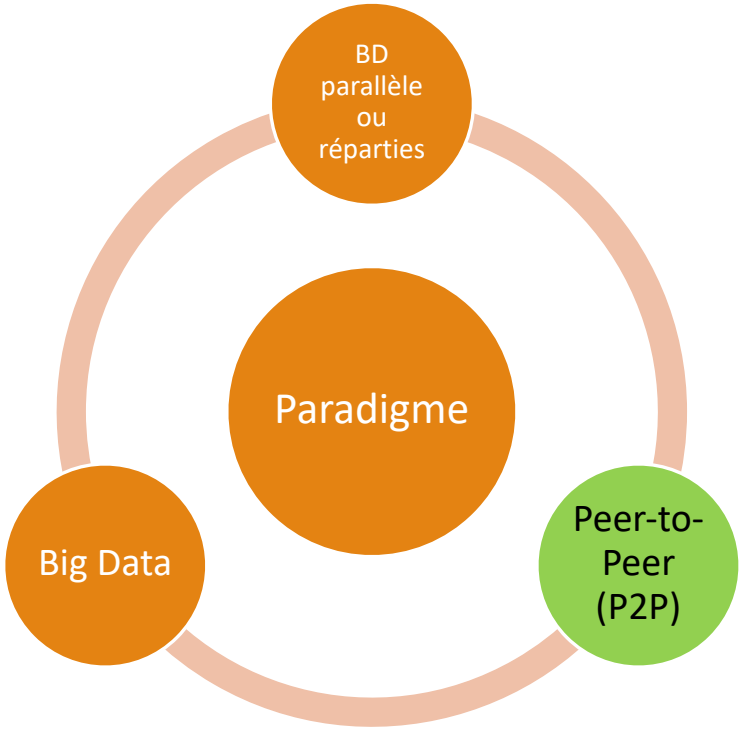
TP

- Performances et Tuning

Plan

- Introduction générale
  - Contexte et objectifs
  - Informations sur l'UE
- Focus sur le paradigme SGBD parallèles/réparties
  - Principe de la parallélisation des traitements de requêtes
  - Principe de fragmentation des données en BDR
  - Performances des traitements
- Focus sur le paradigme P2P
  - Principe de la localisation des données
- Focus sur le paradigme Big Data (Mohand-Saïd Hacid)

Focus sur le paradigme P2P



Achitectures Pair à Pair (P2P)

Le paradigme P2P

Les architectures P2P

- Architectures non structurées
- Architectures hiérarchiques
- Architectures structurés

Motivation

Limitations de l'architecture client/serveur dans le cadre du Web

- Concentration des ressources individuelles sur un petit nombre de nœuds
- Goulet d'étranglement

Besoin de répartir la charge de traitement et de la bande passante à travers tous les nœuds du système d'information réparti

Historique du P2P

Internet des débuts était un système P2P

- N'importe quel couple d'ordinateurs pouvait s'envoyer des paquets :
  - Pas de firewalls, pas de communications asymétriques
- Les machines sont indifféremment clients ou serveurs
- Approche coopérative

Renaissance du P2P

- Succès incontestable à l'époque des systèmes de partage de fichiers musicaux (eMule, eDonkey, KaAzA …)

Domaines applicatifs

- Partage de fichiers
- Moteur de recherche
- Gestion de données
- Travail collaboratif
- Messageries
- Calcul distribué
- Jeux en réseau
- Réseaux mobiles

Le paradigme P2P

Définition intuitive donnée par Clay Shirkey :

«*Peer-to-peer is a class of applications that take advantage of resources storage, cycles, content, human presence - available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, peer-to-peer nodes must have significant or total autonomy of central servers. »*

Ce qu'il faut retenir:

- Echelle de l'Internet,
- Contribution par simple présence
- Totale décentralisation
- Réseau dynamique
- Autonomie des pairs

Principe fondamental du P2P

- Chaque nœud est à la fois *client* et *serveur*
- Aucune construction de connaissance globale du réseau
  - *Aucun nœud n'est fiable*
- Absence de coordination centralisée
  - *Le comportement global du réseau émerge des interactions locales entre les pairs*
- Coopération pour le partage de ressources (fichiers ou puissance de calcul)
  - *Utilisation des ressources des autres et mise à disposition des ressources locales*
- Chaque nœud dispose d'un petit nombre de voisins
  - *Exploitation du principe du Milgram*

Théorème de Milgram

«Tout être humain peut être relié à un autre être humain par une chaîne constituée d'au plus 6 personnes ».

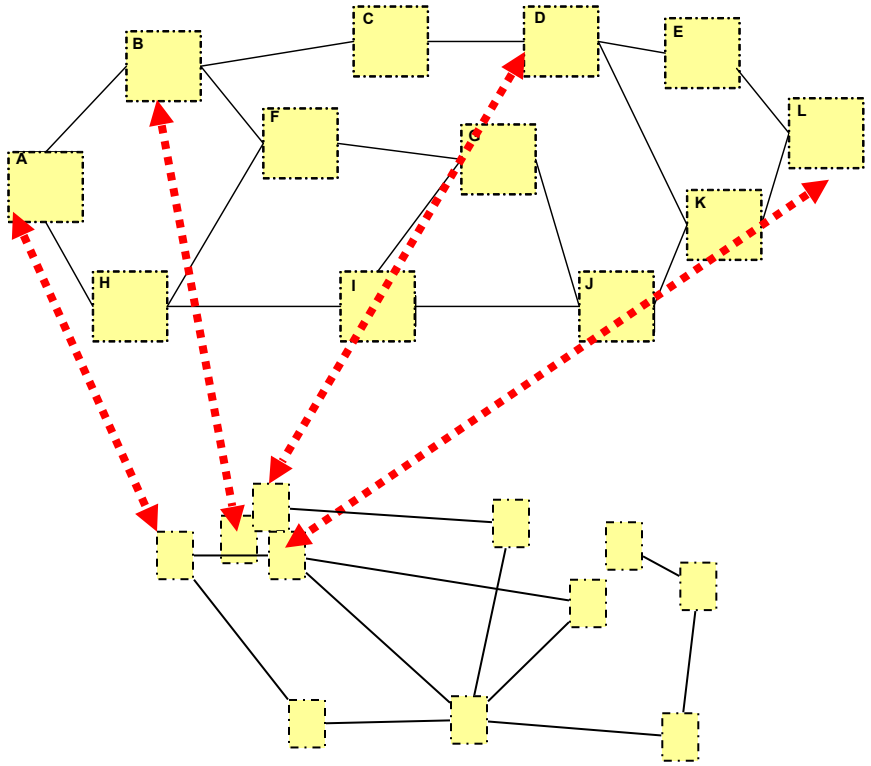
➔ Effet petit monde

1967 : envoi de lettres sans l'adresse du destinataire mais des informations sur son lieu de travail (Boston) et sa profession à partir du Nebraska

Représentation logique du réseau P2P

Réseau logique

Réseau physique



Voisins logiques

≠

Proximité physique des nœuds

Organisation du réseau logique

Les différents types d'architectures P2P

- Les architectures P2P non structurées
  - Les réseaux « pur » pair à pair
- Les architectures P2P hiérarchiques
  - Les réseaux super-peer
- Les architectures P2P structurées
  - Les réseau à base de *Distributed HashTable* (DHT)

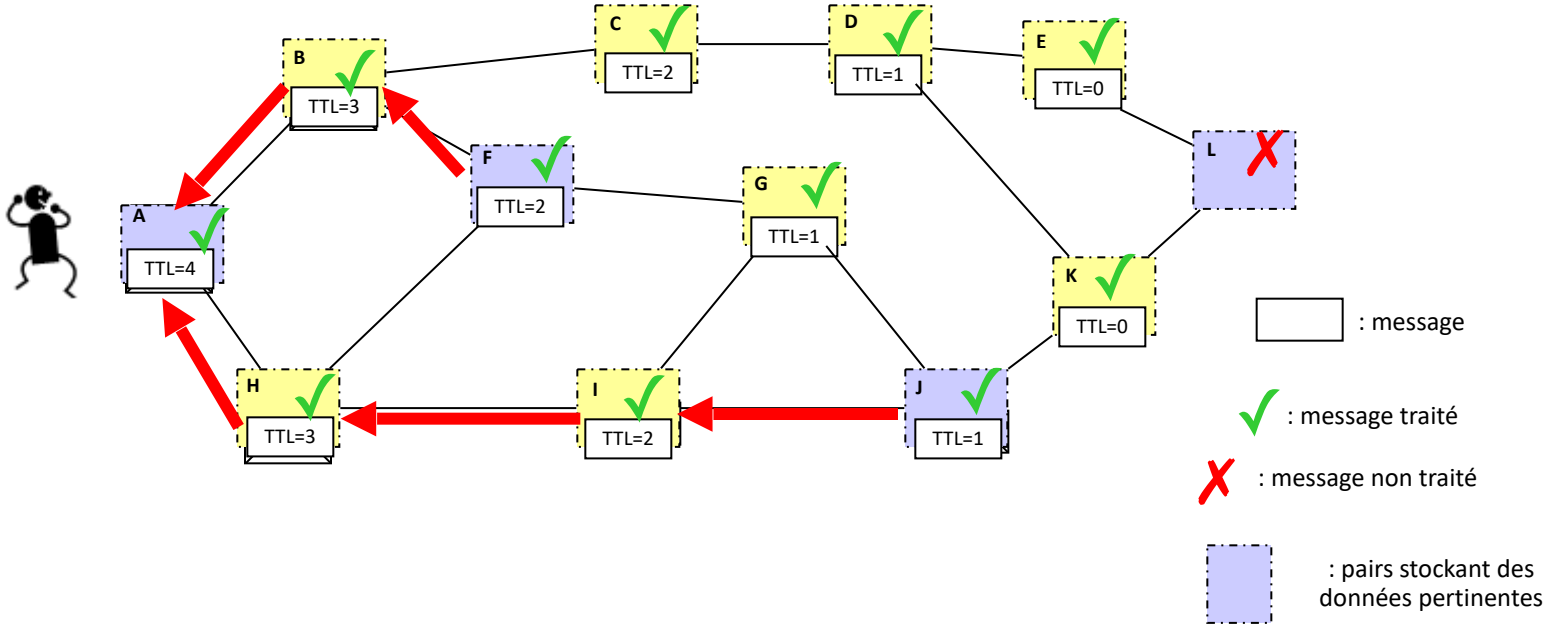
le P2P non structuré

- Une application directe du paradigme du P2P
- Chaque pair :
  - À la fois client et serveur
  - Est autonome en terme de stockage des ressources
  - A uniquement connaissance de son voisinage logique

Problèmes :

- Nombreuses limitations dues
  - Au nombre de messages échangés
  - Efficacité du processus de localisation
    - Approche à l'aveugle …

Illustration du processus d'inondation



Caractéristiques du P2P non structuré

- Principe d'égalité entre les nœuds
  - Même capacité (puissance, bande passante, …)
  - Même comportement (également client et serveur) et bon comportement (pas de « mensonge »)
- Principe de requêtes « populaires »
  - Les ressources très demandées sont très répliquées
  - Les requêtes concernent principalement peu de ressources
- Principe de topologie dynamique du réseau
  - Graphe entre pairs variant au fur et à mesure des connexions et des déconnexions

Question

Le paradigme P2P est-il réaliste ?

- Les nœuds d'un réseau sont-ils vraiment tous égaux ?
- Est-il raisonnable qu'au nom de l'autonomie, les ressources ne soient pas indexées ?

Hiérarchisation des pairs

Levée de la contrainte d'égalité des pairs

Objectif :

- Donner à certains pairs un rôle spécifique
  - Rôle de représentation
  - Rôle de routage

- Retirer du processus de localisation les pairs ayant une faible bande passante : charger d'autres pairs de les représenter

Stratégies

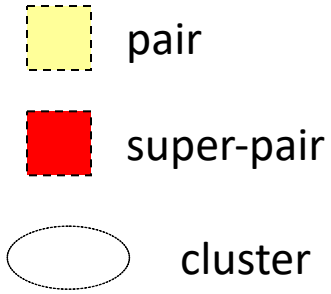
- Création d'un index des données des pairs sur certains pairs

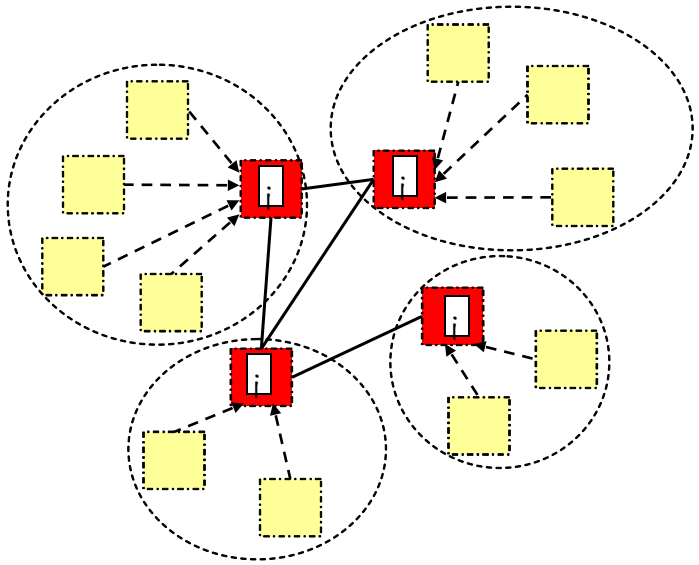
  => Super-Pair

Réseau hiérarchique

Les Super-Pairs sont organisés en **P2P**.

Les Pairs sont rattachés en mode **client/serveur** à un super-pair





Index des ressources du cluster

Super-pairs

Super-pair / Pair

- Super-Pair : Les nœuds avec une bonne bande passante
- Pair : Les nœuds avec faible bande passante sont regroupés autour d'un super-pair

Rôle du super-pair:

- Se substituer aux pairs dans le processus de localisation

Indexation

- Les super-pairs disposent d'un index de ressources des pairs qu'ils représentent.

Remarque

- Napster : 1 super-pair / n pairs
- KaZaA : m super-pairs / n pairs

Caractéristiques du P2P hiérarchique

Avantages

- Retrait des pairs aux faibles capacités (bande passante, processeur) du processus de localisation de l'information
- Localisation plus rapide des ressources

Inconvénients

- Création d'un index sur les super-pair susceptibles de résoudre les requêtes.
- Une centralisation, même locale, rend le système moins robuste aux défaillances des super-pairs.
- Redondance des super-pairs coûteuse

Besoin

Valoriser l'efficacité du processus de localisation

⇒Indexation des données.

- Garantir de trouver une donnée disponible sur le réseau

Besoin d'un index décentralisé !

Réseaux structurés

Caractéristiques

- Les données ou les références de données sont placées sur des pairs spécifiques
- Le placement se fait grâce à une fonction de hachage
- La recherche s'effectue à partir d'une clé

Challenge

- trouver rapidement un objet à partir de sa clé sur un grand réseau sans structure centralisée

Distributed Hash Table (DHT)

- Mécanisme distribué et décentralisé permettant d'associer des valeurs de clés à un contenu par hachage de la clé
  - Chaque participant gère une partie de la table de hachage.
  - Performance en recherche de l'ordre de log(N)
  - Reconfigurable lorsqu'un site se connecte ou se déconnecte

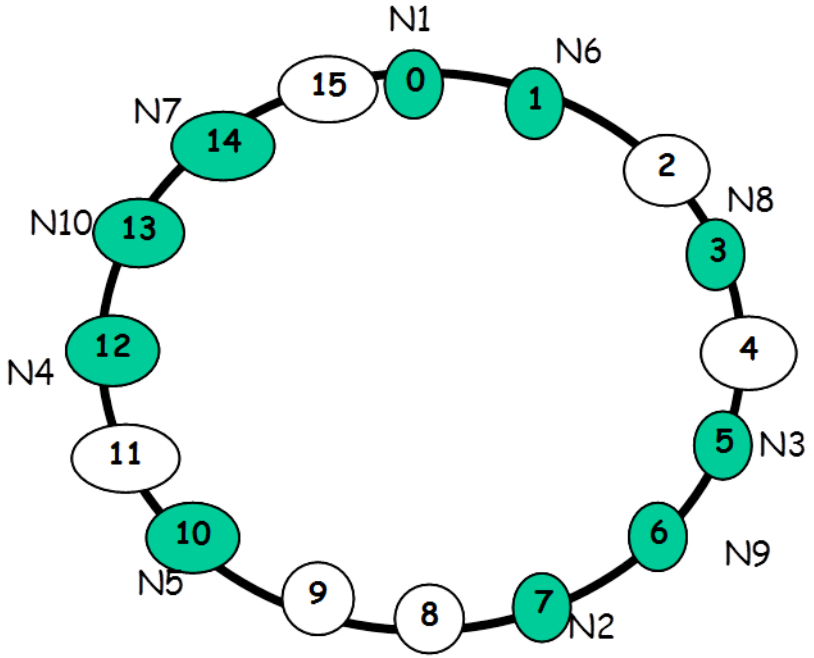
- Mécanisme général pour la localisation de ressources distribuées identifiées par des clés
  - Indexation de fichiers (clé = nom, contenu=adresse ou contenu)

- Consistent hashing
  - Le *consistent hashing* garantit avec une probabilité élevée que :
    - Les ressources sont distribuées uniformément sur l'ensemble des nœuds
    - Le retrait et l'ajout du n^{ième} nœud n'oblige à déplacer que o(1/n) ressources

- Structure de distribution/ Protocole
  - Anneau (Chord), B-Tree (Kadmelia), Hypercube (CAN)…

Chord

- DHT basée sur un anneau permettant de stocker 2^m nœuds
- Placement des nœuds
  - Hachage de l'IP du nœud :
    - Chaque nœud est alloué sur l'anneau en fonction du hachage de son IP.

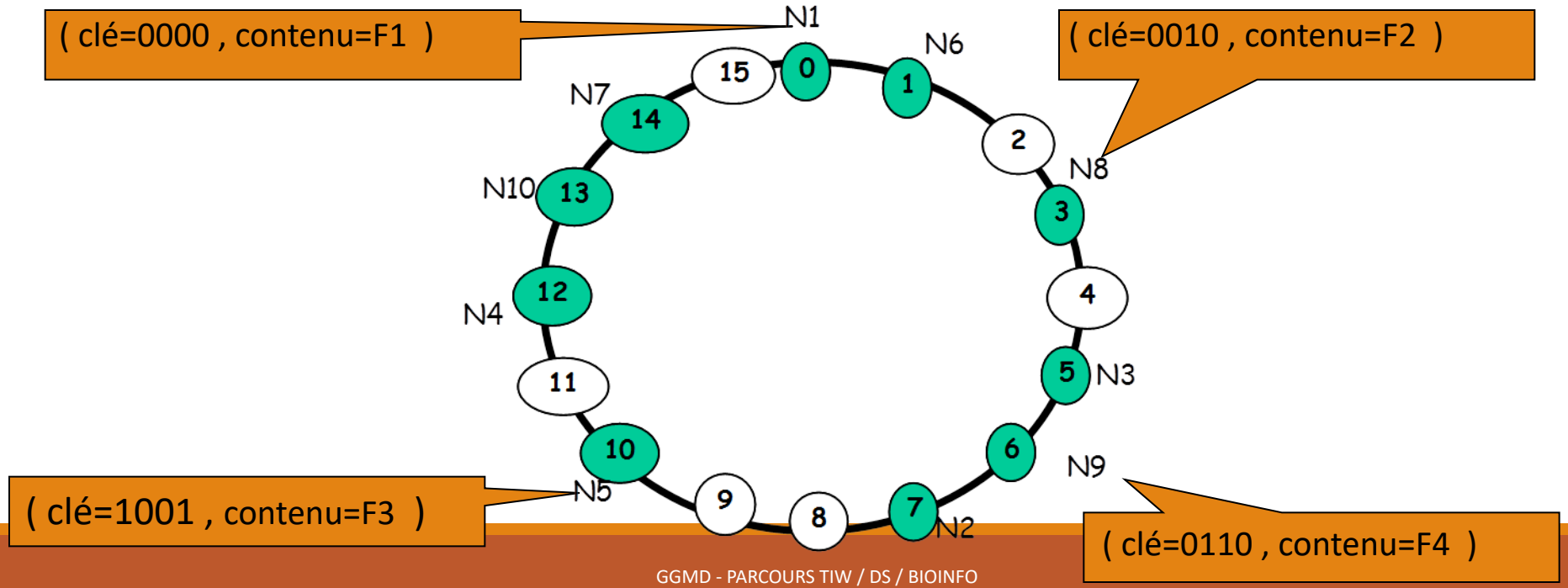


m = 4

➔ 16 emplacements

Placement des ressources

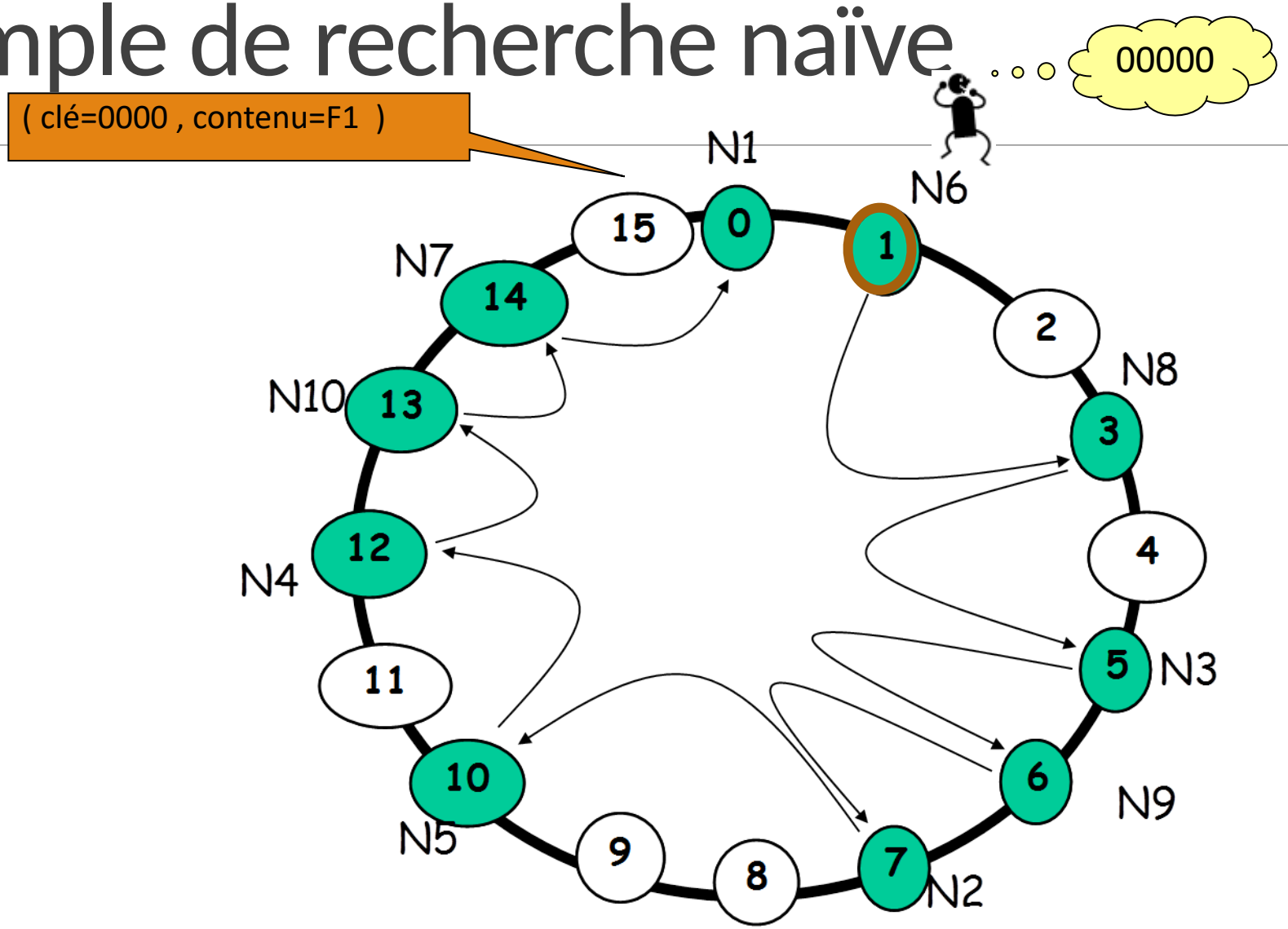
- Hachage des ressources :
  - H(ressource) = clé
  - La clé est placée sur le nœud d'IP haché immédiatement supérieur ou égal à la valeur de la clé



Recherche naïve d'une ressource

- Chaque nœud stocke l'identifiant de son successeur
- Pour traiter une requête, un pair propage la requête à son successeur.
- Le résultat suit le chemin dans l'ordre inverse
- Recherche linéaire en nombre de nœuds

Exemple de recherche naïve



Amélioration de la recherche

Avoir une table de routage plus complète:

➔ Table des *finger nodes*

- Utilisation de successeurs autres que les successeurs sur l'anneau

Principe

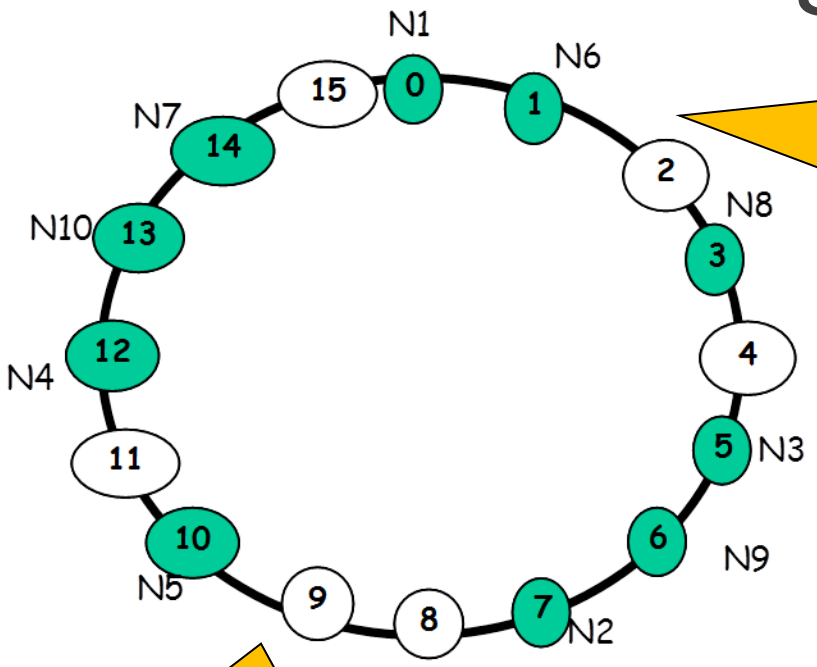
- Pour chaque nœud Ni de valeur de hachage j :

  Succ[k]=premier nœud sur l'anneau qui vérifie :

  $$(j + 2^{k-1}) \bmod 2^m,\ 1 <= k <= m$$

- Utilisation de prédécesseurs pour la maintenance de l'anneau

Table des « finger nodes »



$1+2^0=2$	N8
$1+2^1=3$	N8
$1+2^2=5$	N3
$1+2^3=9$	N5

[2,3 [
[3,5 [
[5,9 [
[9,1 [

$10+2^0=11$	N4
$10+2^1=12$	N4
$10+2^2=14$	N7
$10+2^3=2$	N8

[11,12 [
[12,14 [
[14,2 [
[2,10 [

Algorithme de recherche

Rechercher si la clé existe localement.

- Si oui on renvoie la valeur associée
- Sinon,

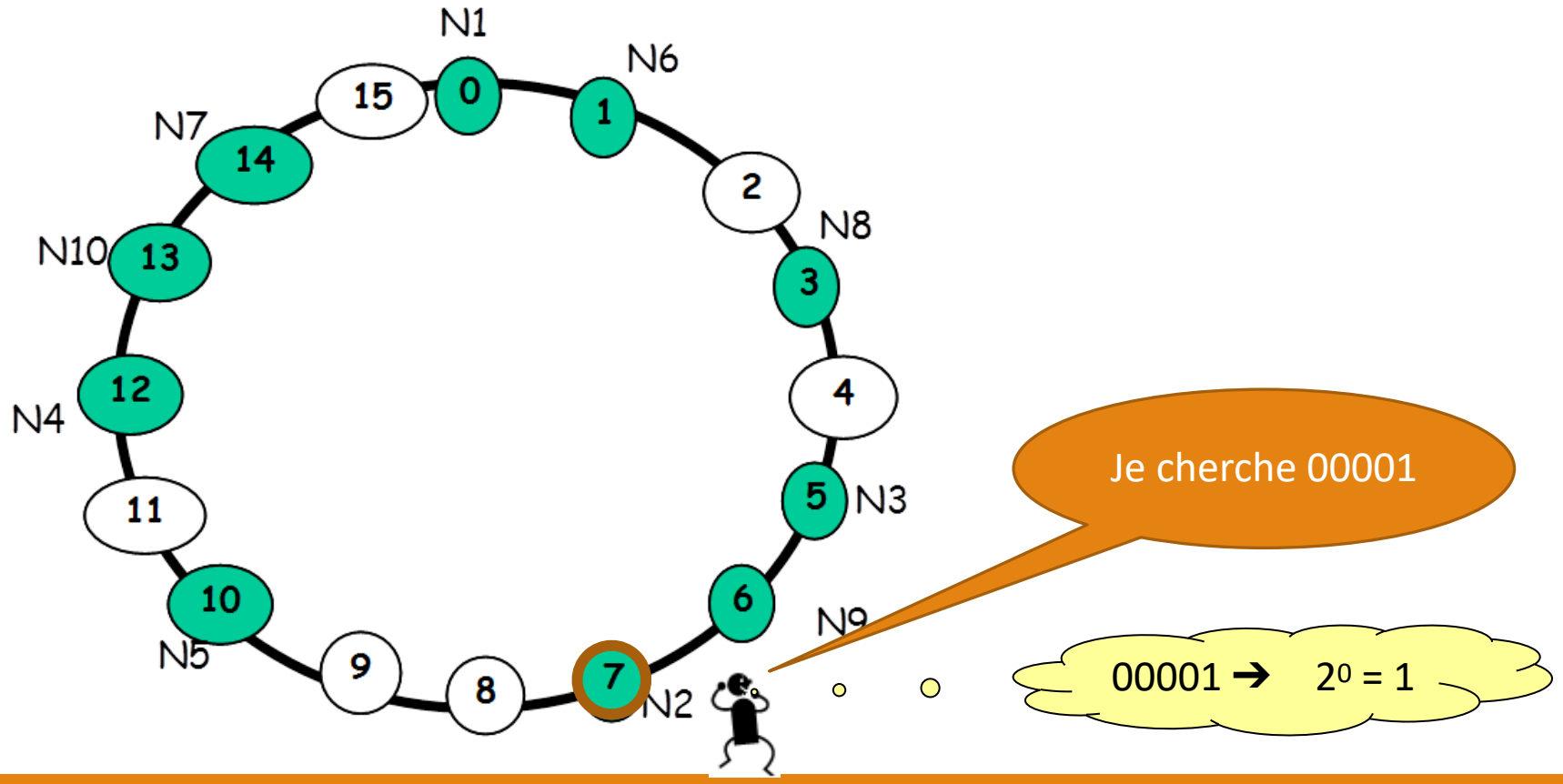
  Rechercher dans la table routage un nœud avec la plus grande valeur inférieure ou égale à la clé cherchée

  Transmettre la requête au nœud sélectionné et appliquer récursivement

➔Nombre de sauts moyen : $O(\log_2(N))$

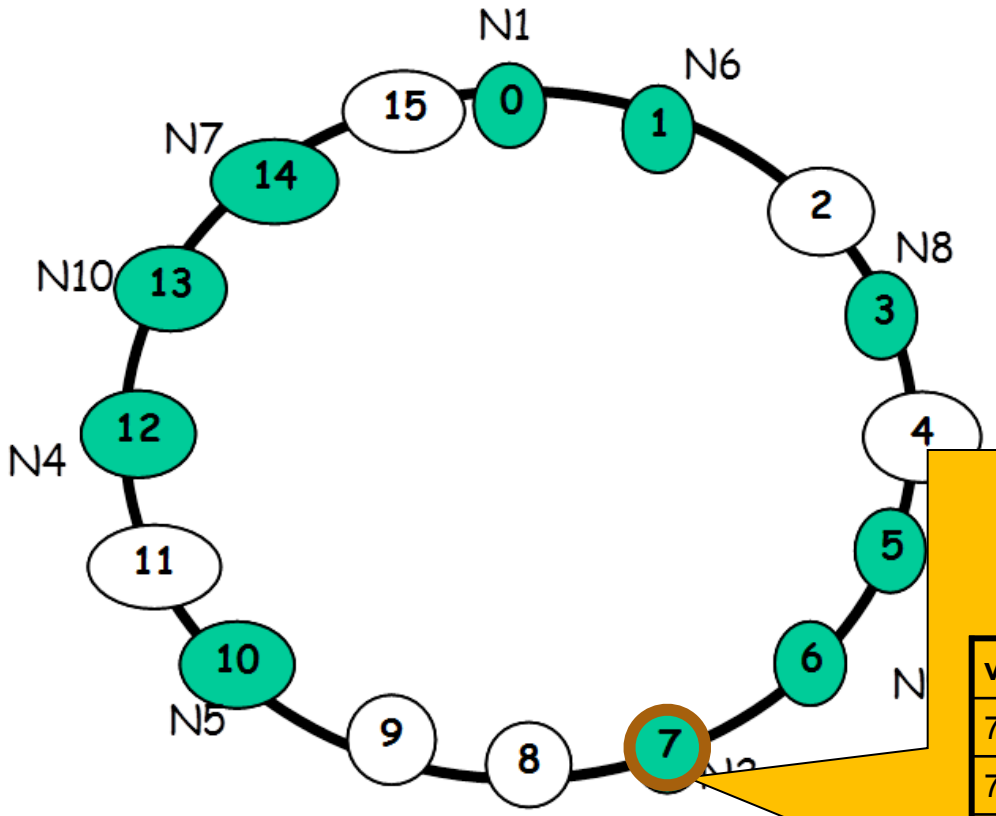
Exemple de recherche dans Chord

Recherche de la clef 00001 à partir du pair N2



Exemple de recherche dans Chord

Recherche de la clef 00001 à partir du pair N2



Clés stockées

{00111,10111}

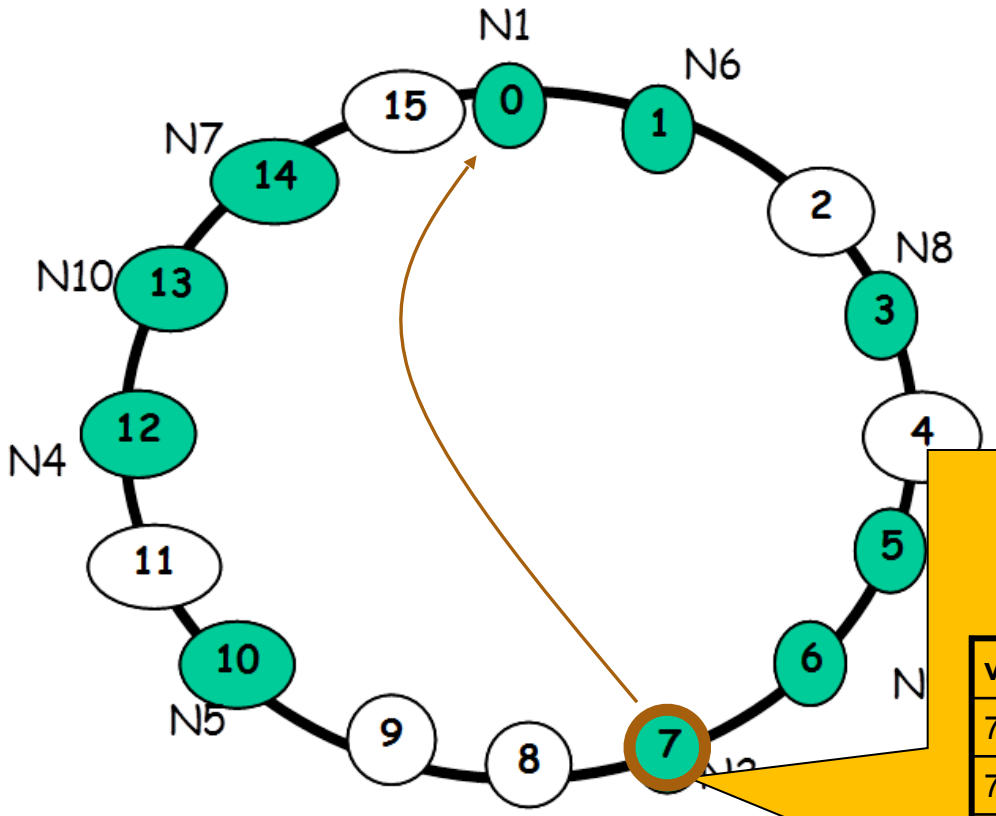
Table de routage (Finger nodes)

valeur	Resp.	Nœud
$7 + 2^0 = 8$	[8,9[	N5
$7 + 2^1 = 9$	[9,11[	N5
$7 + 2^2 = 11$	[11,15[	N4
$7 + 2^3 = 15$	[15,7[	N1

$1 \in [15, 7[$

Exemple de recherche dans Chord

Recherche de la clef 00001 à partir du pair N2



Clés stockées

{00111,10111}

Table de routage (Finger nodes)

valeur	Resp.	Nœud
$7 + 2^0 = 8$	[8,9[	N5
$7 + 2^1 = 9$	[9,11[	N5
$7 + 2^2 = 11$	[11,15[	N4
$7 + 2^3 = 15$	[15,7[	N1

$1 \in [15, 7[$

Recherche dans Chord

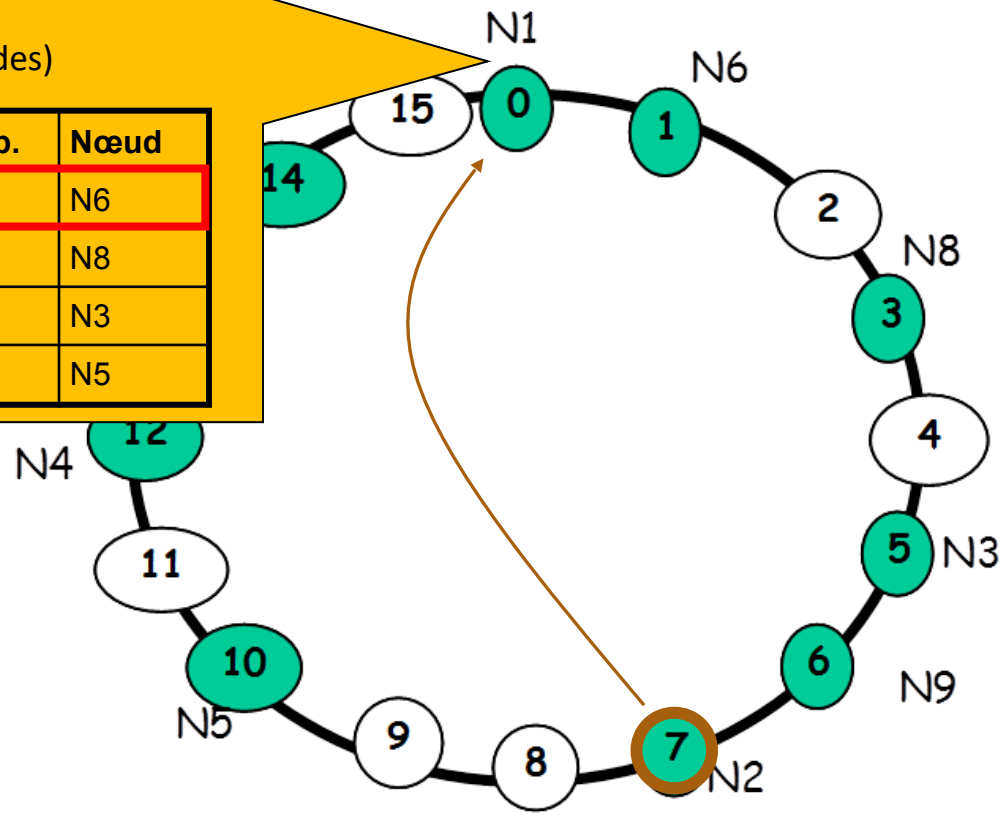
Clés stockées

{00000,01111,10000,11111}

Table de routage (Finger nodes)

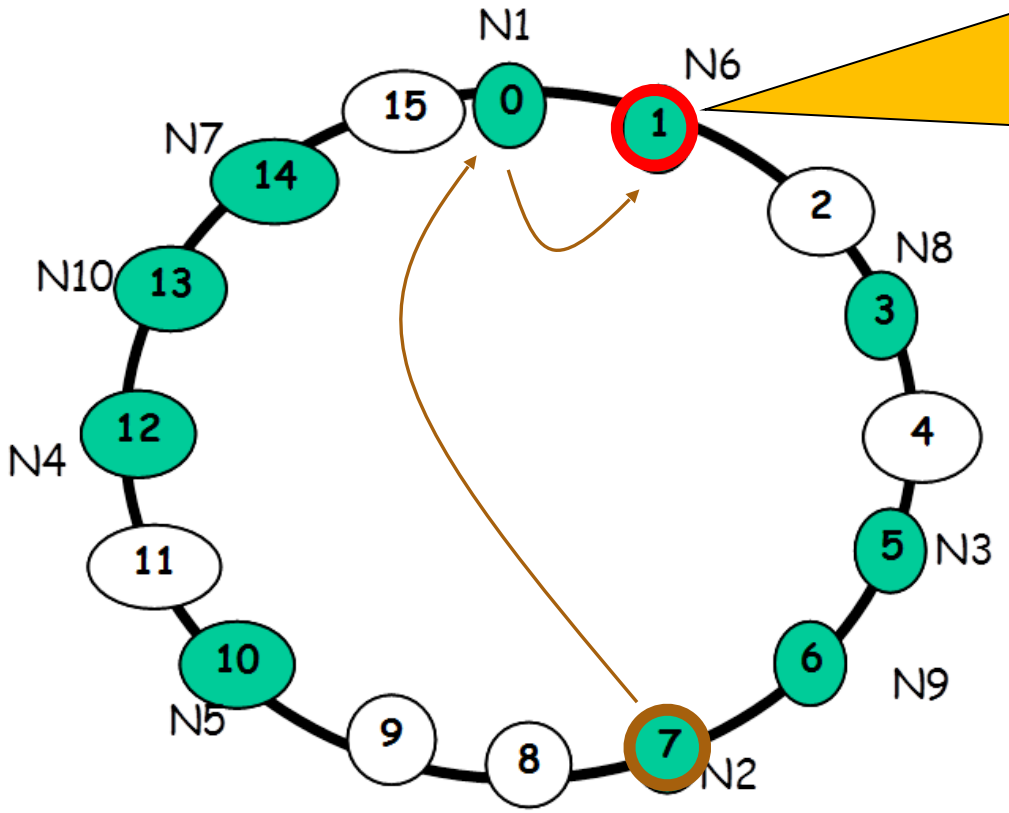
$1 \in [1, 2[$

valeur	Resp.	Nœud
$0 + 2^0 = 1$	[1,2[	N6
$0 + 2^1 = 2$	[2,4[	N8
$0 + 2^2 = 4$	[4,8[	N3
$0 + 2^3 = 8$	[8,0[	N5



Recherche dans Chord

Recherche de la clef 00001 à partir du pair N2



Clés stockées

{00001, 10001}

Table de routage (Finger nodes)

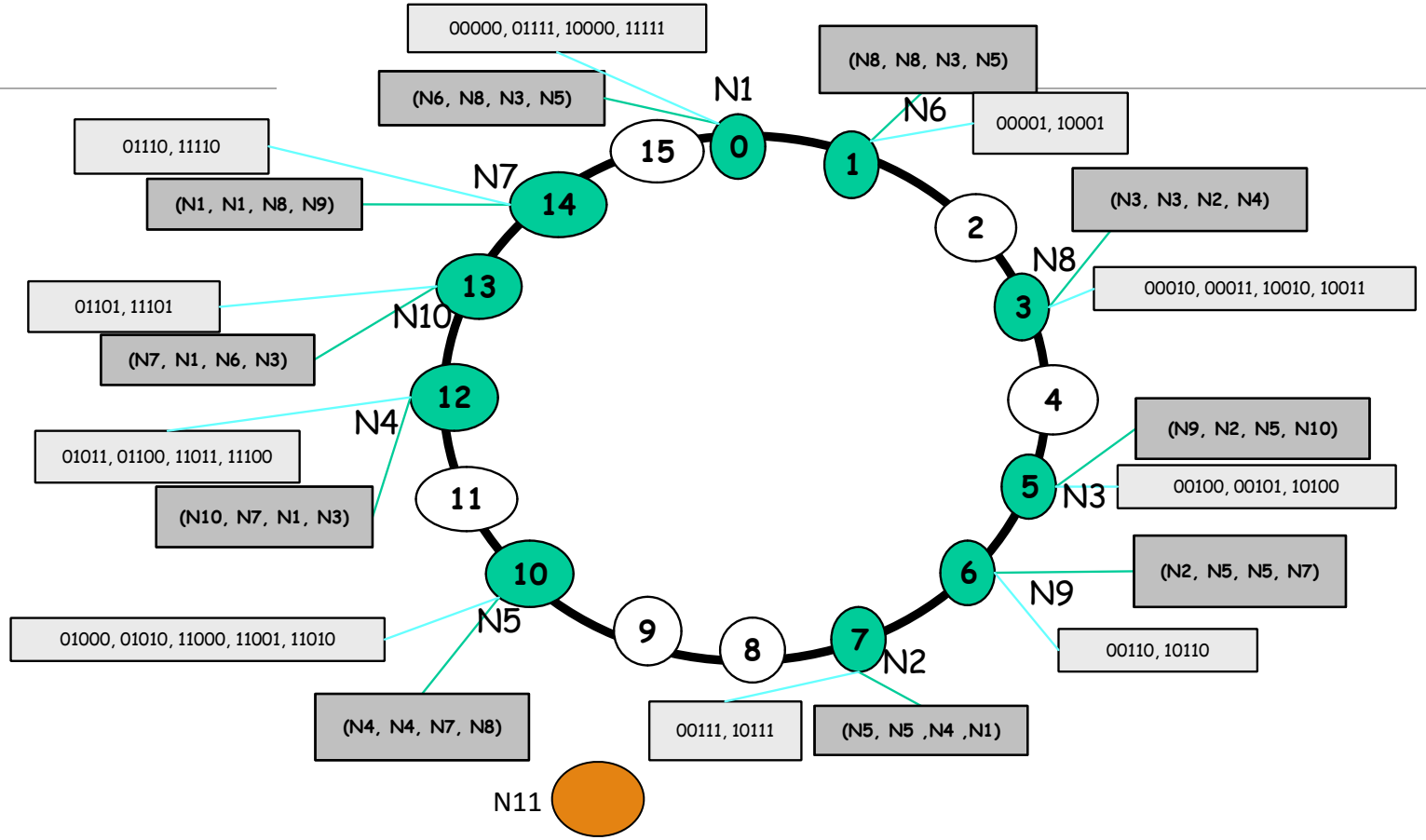
Valeur	Resp.	Nœud
$1 + 2^0 = 2$	[2,3[	N8
$1 + 2^1 = 3$	[3,5[	N8
$1 + 2^2 = 5$	[5,9[	N3
$1 + 2^3 = 9$	[9,1[	N5

Ajout d'un nœud dans CHORD

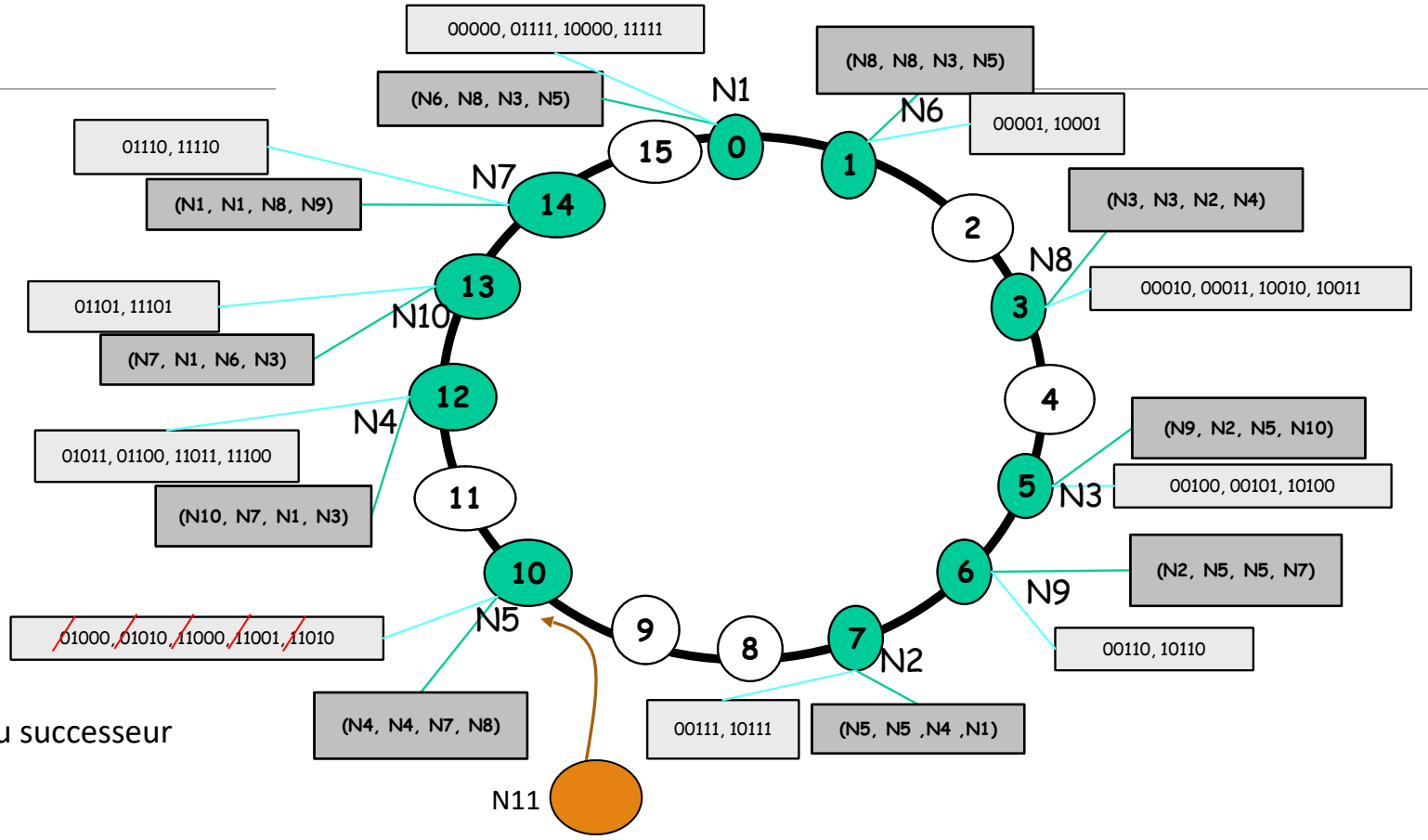
Repose sur la combinaison d'opérations élémentaires :

- N.JOIN(N')
  - nœud N annonce au nœud N' qu'il rentre dans le réseau et lui demande de lui fournir son successeur
- N.STABILIZE()
  - lancé périodiquement, permet à N et à son successeur de vérifier qu'ils forment un couple correct (il n'y a pas de nouveaux nœuds entre les 2)
- N.MAJENTRY() :
  - lancé périodiquement, permet d'initialiser la table de routage pour les nouveaux nœuds ou de la mettre à jour pour les nœuds existants
- N.TESTPREC() :
  - lancé périodiquement, vérifie que le prédécesseur est toujours là

Insérer le pair N11 à la position 9

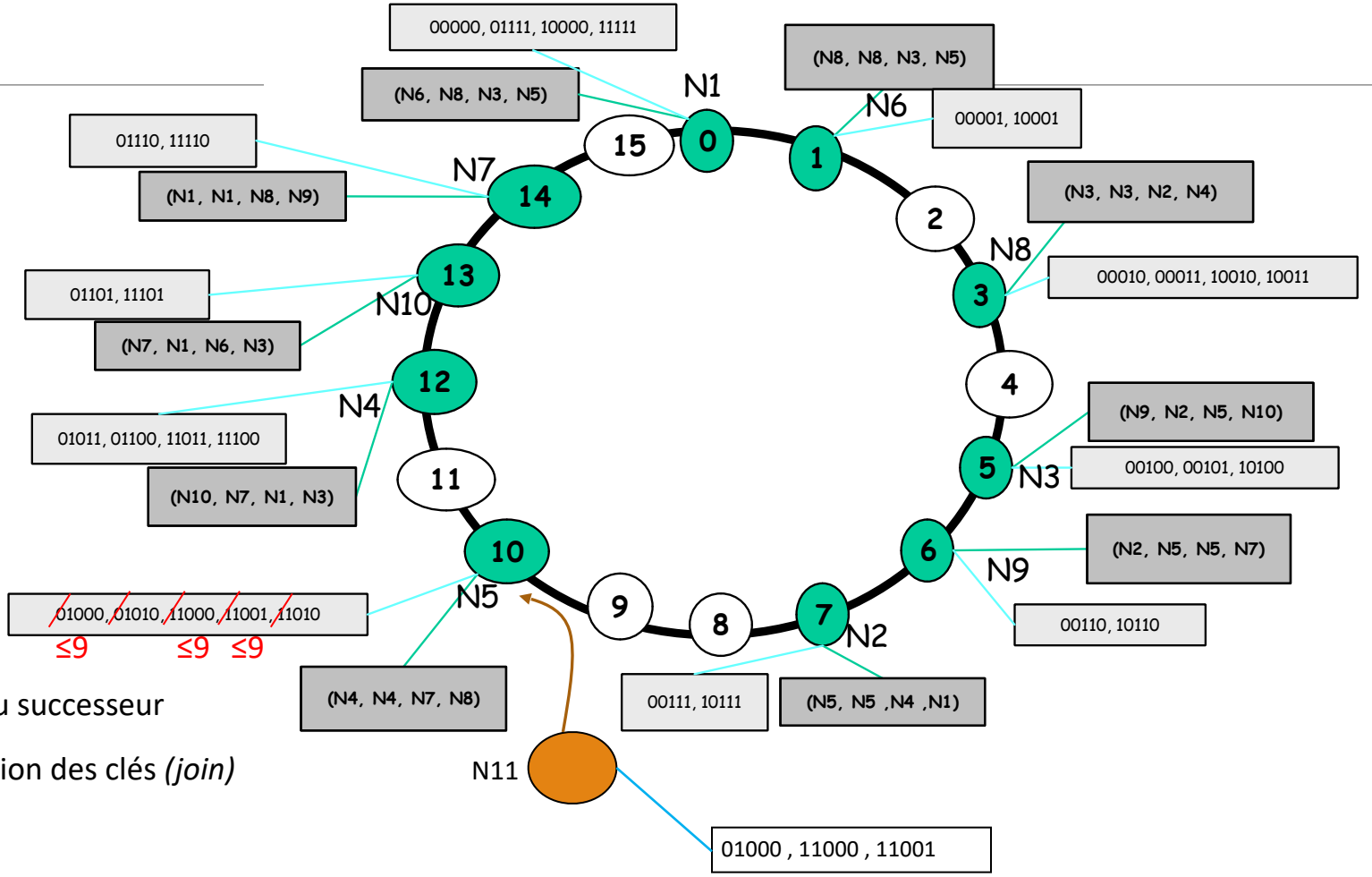


Insérer le pair N11 à la position 9



1) Contact du successeur

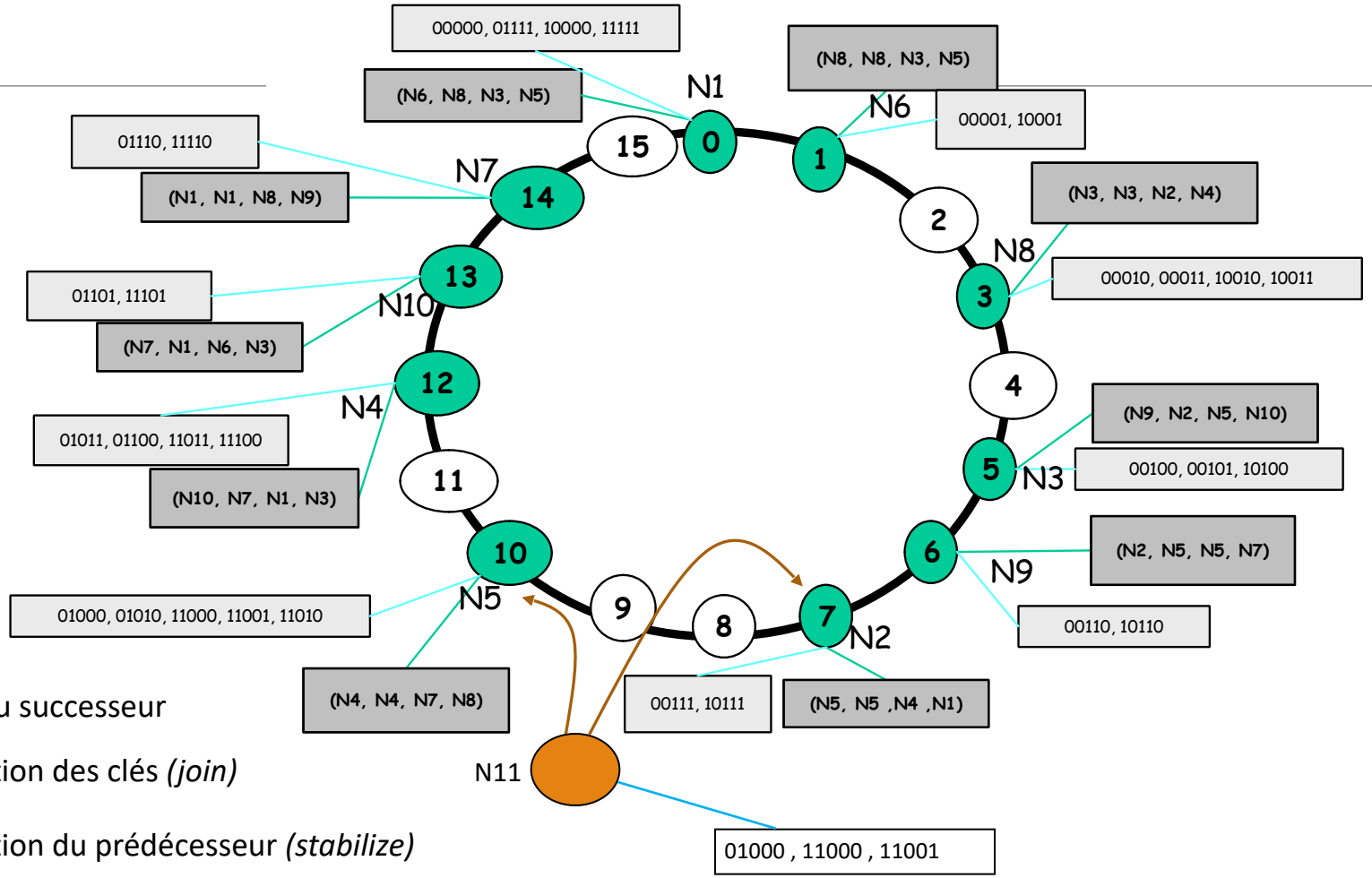
Insérer le pair N11 à la position 9



1) Contact du successeur

2) Récupération des clés *(join)*

Insérer le pair N11 à la position 9

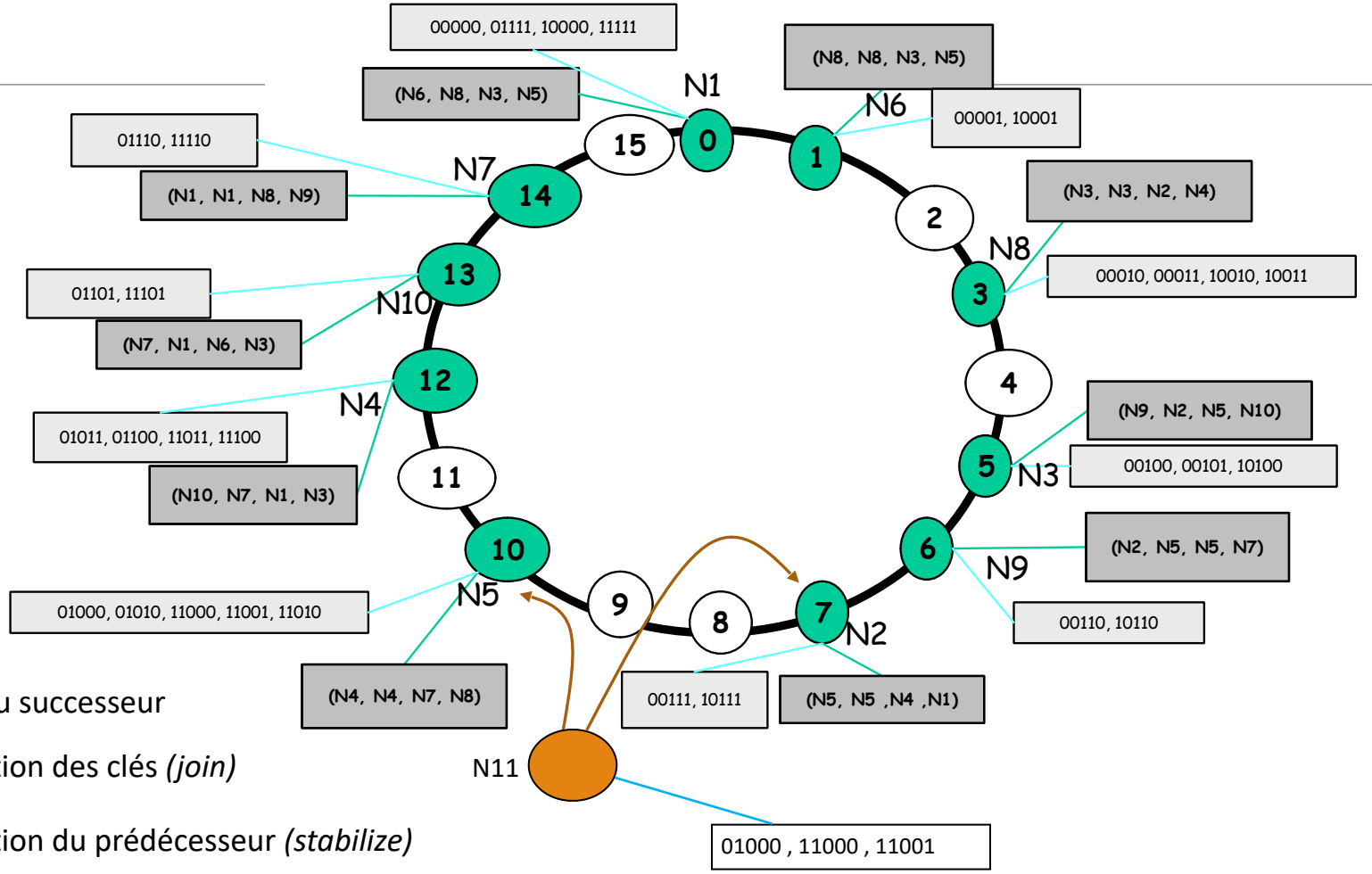


1) Contact du successeur

2) Récupération des clés *(join)*

3) Récupération du prédécesseur *(stabilize)*

Insérer le pair N11 à la position 9



1) Contact du successeur

2) Récupération des clés *(join)*

3) Récupération du prédécesseur *(stabilize)*

4) Mise à jour des tables de routages

Propriétés de l'algorithme d'ajout

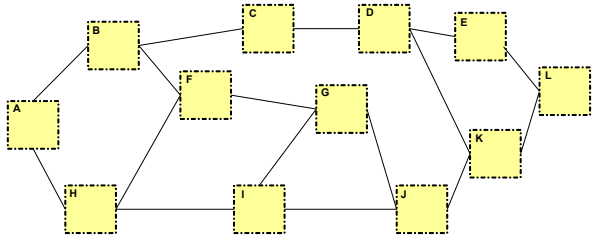
L'algorithme garantit que n'importe quelle séquence de *join* entrelacée avec des *stabilize* converge vers un état stable (tous les nœuds restent atteignables)

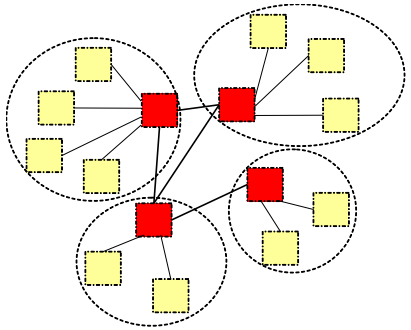
Même en présence d'un ajout (limité) de nœud, le coût de la recherche reste en O(log(N))

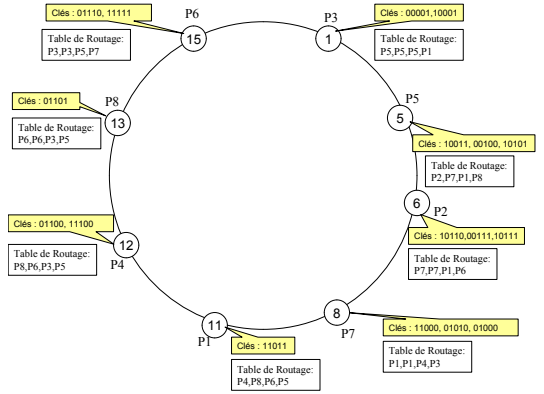
Une recherche peut échouer si le réseau n'est pas complètement stabilisé (il faut relancer la recherche un peu plus tard)

Comparatif

P2P non structuré / P2P hybride / P2P structuré







	Non structuré	hybride	structuré
autonomie	Forte	Moyenne	Faible
efficacité	Faible	Moyenne	Forte
Expressivité du langage	Forte	Moyenne	Faible
Tolérance aux pannes	Forte	Faible	Forte

Conclusion P2P

On distingue 3 types d'architectures Pair à Pair qui respecte plus ou moins le paradigme P2P:

- Non structurées (respecte le paradigme)
- Hybride (lève le principe d'égalité des pairs)
- Structurés (lève le principe d'autonomie et permet l'indexation des données)

Le choix d'une architecture se fait selon :

- le contexte applicatif
- Les contraintes
- …

Challenges

- Améliorer le processus de localisation, gérer des langages de requête de plus haut niveau, sécuriser les interactions…

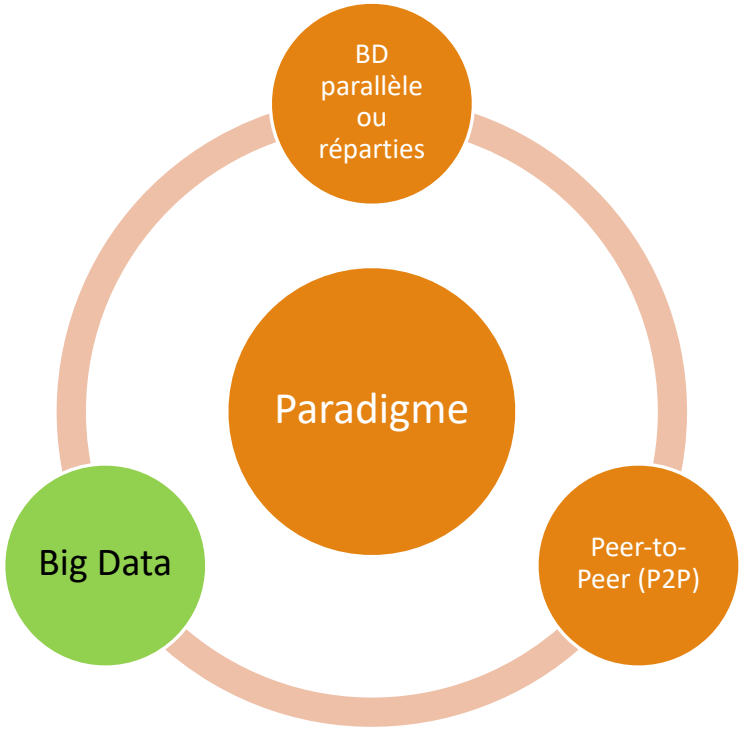
Application

- Partage de données en P2P

Plan

- Introduction générale
  - Contexte et objectifs
  - Informations sur l'UE
- Focus sur le paradigme SGBD parallèles/réparties
  - Principe de la parallélisation des traitements de requêtes
  - Principe de fragmentation des données en BDR
  - Performances des traitements
- Focus sur le paradigme P2P
  - Principe de la localisation des données
- Focus sur le paradigme Big Data (Mohand-Saïd Hacid)

Focus sur le paradigme Big Data



avec Mohand-Saïd Hacid

Quelques références

État de l’art : The state of the art in distributed query processing de D. Kossman

Environnement pour l'UE

Les jeux de données utilisés

- Issus de l'open data:
  - Données INSEE :
    - Décès de français répertoriés depuis 1970
    - Organisation administrative du territoire français en 2022

24 916 669 tuples (2,3Go)

https://www.insee.fr/fr/information/4190491
https://www.insee.fr/fr/information/6051727

  - Données DATA-GOUV:
    - Informations sur les mairies françaises

https://www.data.gouv.fr/fr/datasets/mairies-1/

  - Données DBLP :
    - Publications d'articles scientifiques

https://blog.dblp.org/2022/03/02/dblp-in-rdf/

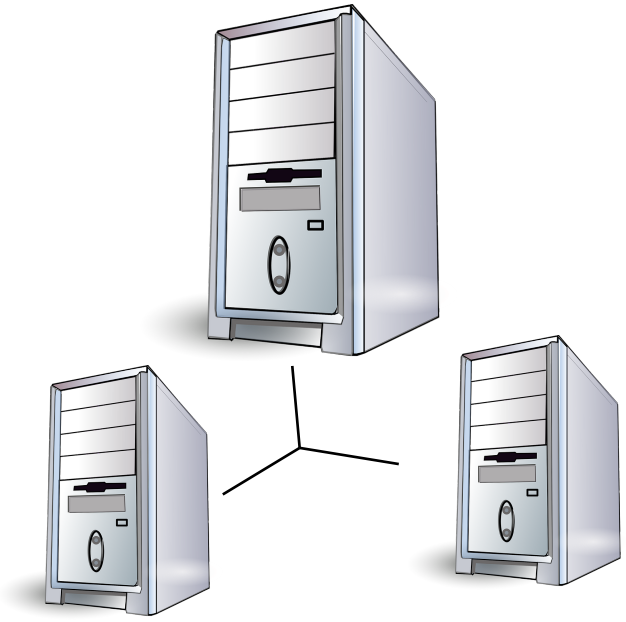
252 573 199 triplets (2,6Go)

- Issus de simulation
  - Flux de capteurs

L’infrastructure utilisée (pendant les TPs)

VCPUs : 1
RAM : 2Go

VCPUs : 4
RAM : 8Go



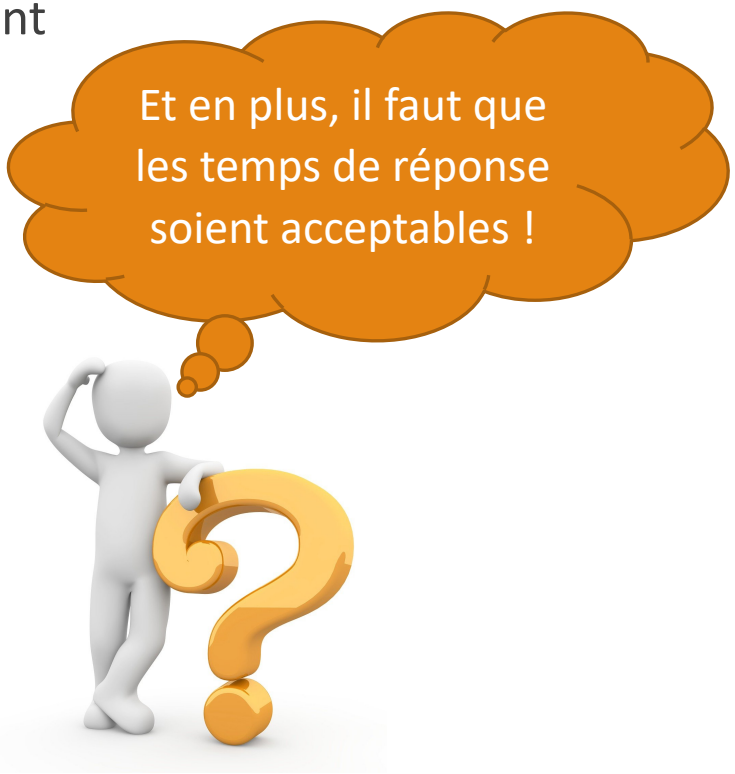
VCPUs : 4
RAM : 8Go

VCPUs : 2
RAM : 2Go

VCPUs : 1
RAM : 2Go

Première mission… en TP

Vous devez déployer les données INSEE (2,3 Go) sur le serveur de production *ggmd-01 (1 cœur, 2Go de RAM)* pour pouvoir traiter la requête retournant les homonymes (même nom et même premier prénom) qui sont nés la même année dans des communes différentes.



Objectif

Etudier les solutions permettant de répartir efficacement le traitement des données sur plusieurs CPU

Plan

Introduction générale
- Contexte centralisé

Focus sur le paradigme SGBD parallèles/réparties
- Principe de la parallélisation des traitements de requêtes
- Principe de la répartition des données
- Tuning de SGBD

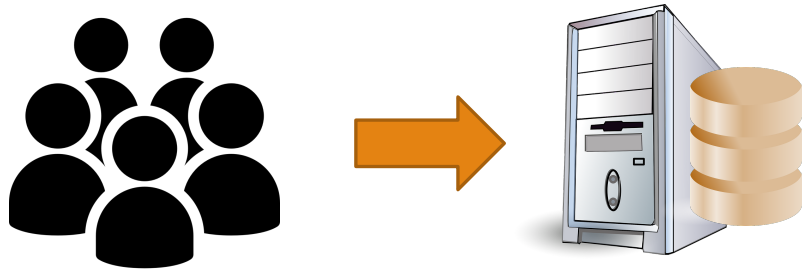
Focus sur le paradigme P2P
- Principe de la localisation des données

Focus sur le paradigme Big Data

Paradigme des SGBD parallèles

Contexte initial (Ci2)

- Le données sont des données relationnelles
  - Les données sont stockées dans un SGBD hébergé sur un seul serveur
    - Les opérations de traitements peuvent s'exécuter de manière concurente



Observations

Plantage ou latences de traitement des requêtes trop importantes (performances CPU) pour une utilisation satisfaisante de la couche applicative au dessus du SGBD