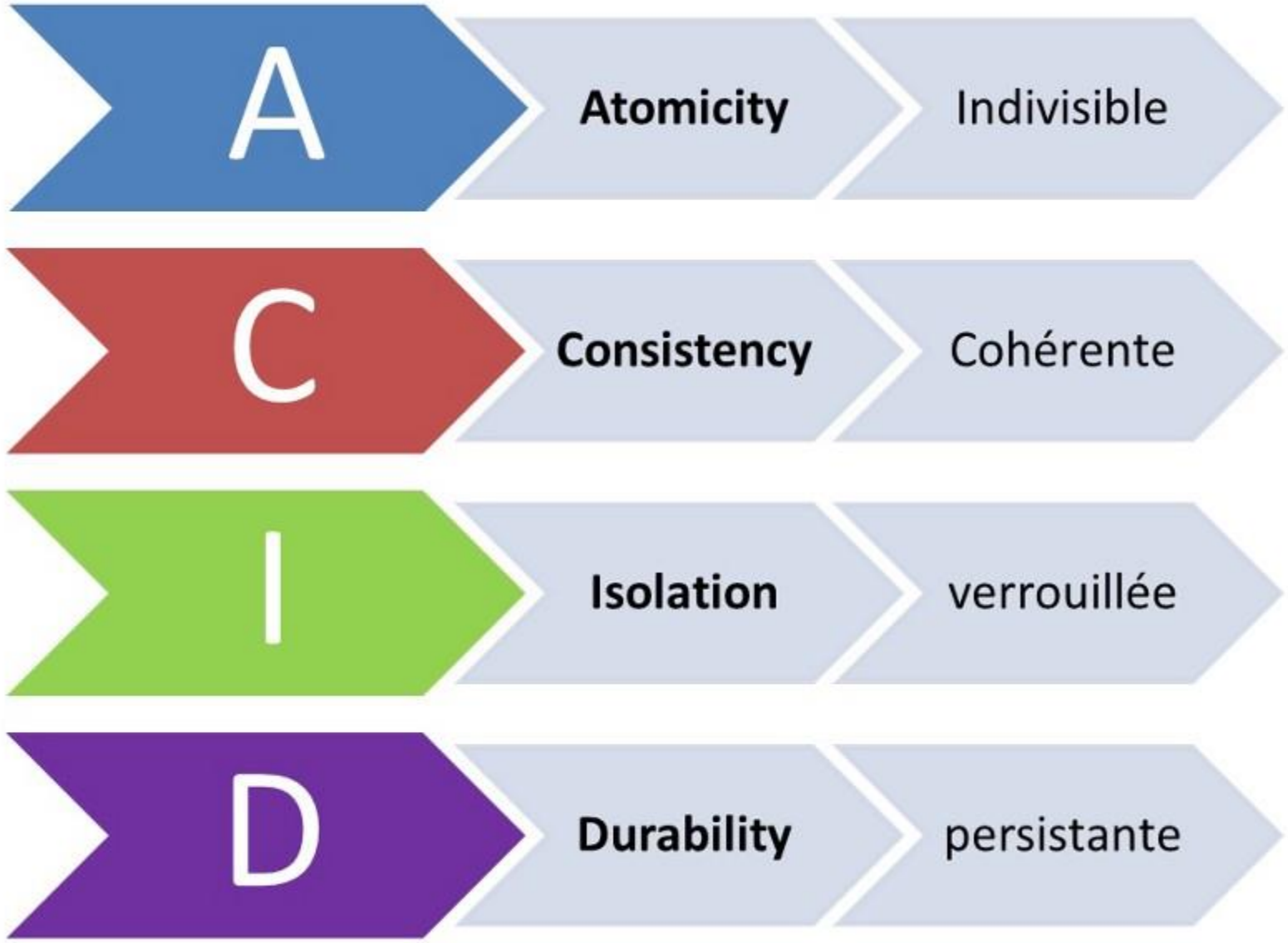
# Transaction, répartition, et cohérence

- **Propriétés ACID et gestion des conflits**
- **Architecture, Répartition des données et Réplication**
- **Niveaux de cohérence**
- **Théorème de CAP**

# Acronyme ACID



# Définition des propriétés ACID

- **Atomicité :** Tout ou rien
  - Une transaction effectue toutes ses actions ou aucune
  - En cas d'annulation, les modifications engagées doivent être défaites
- **Cohérence** : Intégrité des données

  Passage d'un état cohérent de la base à un autre état cohérent de la base de données
- **Isolation** : Pas d'interférence entre transactions

  Les résultats d'une transaction ne sont visibles par les autres transactions qu'après sa validation
- **Durabilité** : Journalisation des mises à jour

  Les modifications effectuées sont garanties même en cas de panne

# Atomicité et Cohérence

| **Before:** X : 500 | Y: 200 |
|---|---|
| Transaction T | |
| **T1** | **T2** |
| Read (X)<br>X: = X − 100<br>Write (X) | Read (Y)<br>Y: = Y + 100<br>Write (Y) |
| **After:** X : 400 | Y : 300 |

- Exécution complète de T1 et T2 => cohérence
- Panne après l'exécution de *write(X)* mais avant celle de *write(Y)* => incohérence

# Isolation et Cohérence

| T | T'' |
|---|---|
| (1) Read (X)<br>(2) X: = X*100<br>(3) Write (X)<br>(7) Read (Y)<br>(8) Y: = Y – 50<br>(10) Write (Y) | (4) Read (X)<br>(5) Read (Y)<br>(6) Z: = X + Y<br>(9) Write (Z) |

- Exécution complète de T suivie de celle de T'' => isolation et cohérence
- Entrelacement des op. de T'' et de T =>
  - T'' lit une valeur impropre de X (en cas d'annulation de T il faut annuler T'')
  - Etat incohérent de la base (≠ de celui obtenu si on avait tout T puis tout T'')

# Ordonnancement sérialisable

$T_1$: Solde A - x; Solde B + x;

$T_2$ : Solde A - y; Solde B + y;

$T_3$ : Solde A - z; Solde B + z;

Exécution séquentielle : $T_1$ $T_2$ $T_3$

Une exécution sérialisable équivalente à $T_1$ $T_2$ $T_3$ :

Solde A - x;

Solde A - y;

Solde B + x;

Solde A - z;

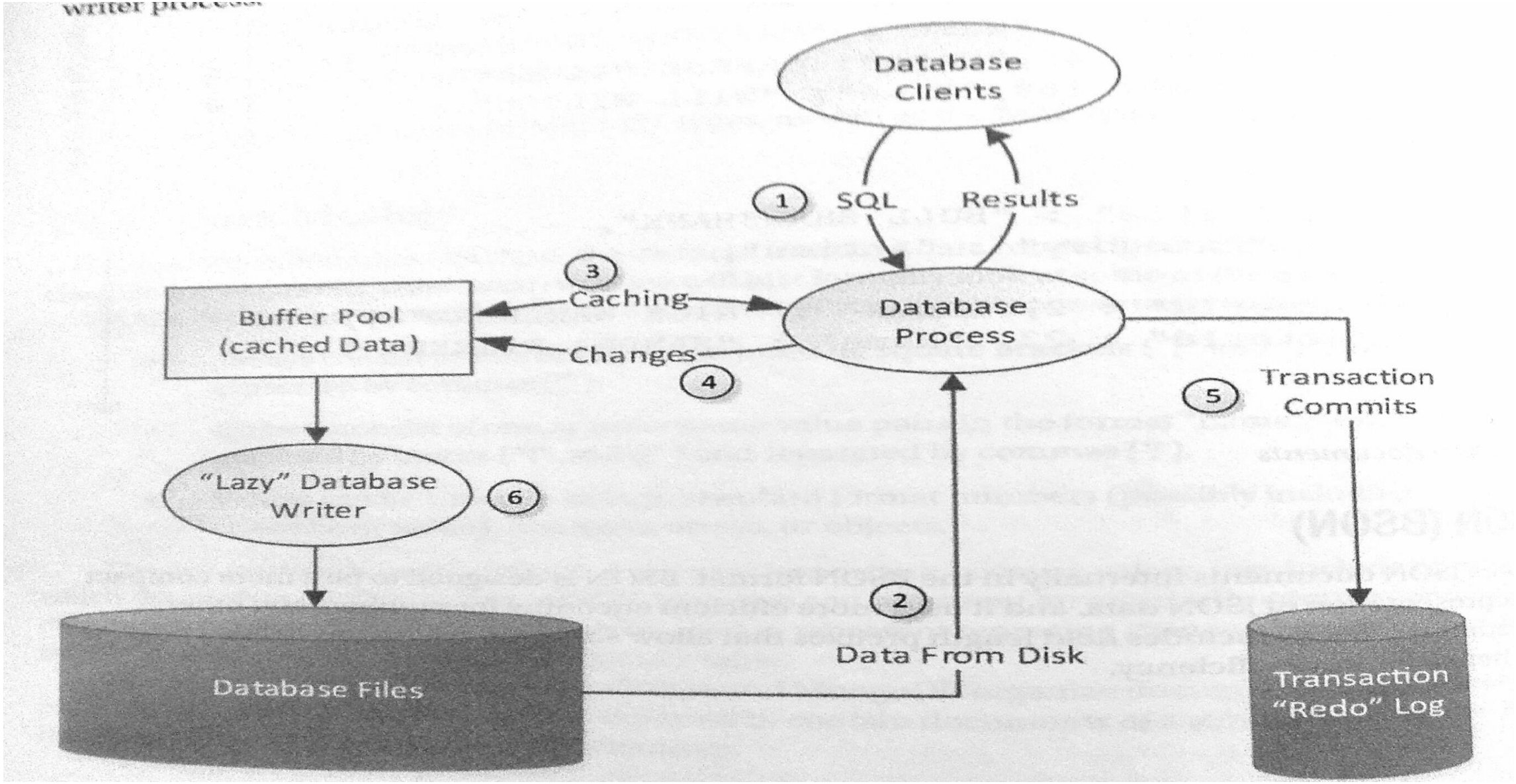
Solde B + y;

Solde B + z;

L'item A voit passer les transaction dans l'ordre
T1 T2 T3

L'item B voit passer les transaction dans l'ordre
T1 T2 T3

# Durabilité


Database Clients
1
SQL
Results
3
Caching
Buffer Pool (cached Data)
Changes
4
Database Process
5
Transaction Commits
"Lazy" Database Writer
6
2
Data From Disk
Database Files
Transaction "Redo" Log

# Conflits

- **Ecriture - Lecture :**
  - **Lecture impropre ou parasite (*dirty read*)**
  - **Placement momentanée de la base dans un état incohérent**
  - **Annulation en cascade de transactions**
- **Lecture - Ecriture :**
  - **Lecture non reproductible (*unrepeatable read*)**
  - **Incohérence**
- **Ecriture- Ecriture :**

  **Perte de mises à jour (*blind write*)**

# Conflits : perte de mise à jour

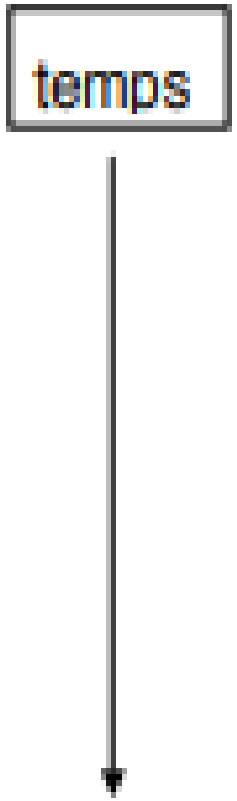
- T1 et T2 transactions concurrentes
- T1 : lire (x) ; x <- x + 10 ; écrire (x)
- T2 : lire (x) ; x <- x + 20; écrire (x)
- Au départ x = 50

| T1 | T2 |
|---|---|
| Lire (x) (T1 lit x = 50 dans BD) | |
| x <- x + 10 | |
| | Lire (x) (T1 lit x = 50 dans BD) |
| Ecrire x (dans BD : x = 60) | |
| | x <- x + 20 |
| | Ecrire x (dans BD : x = 70) |

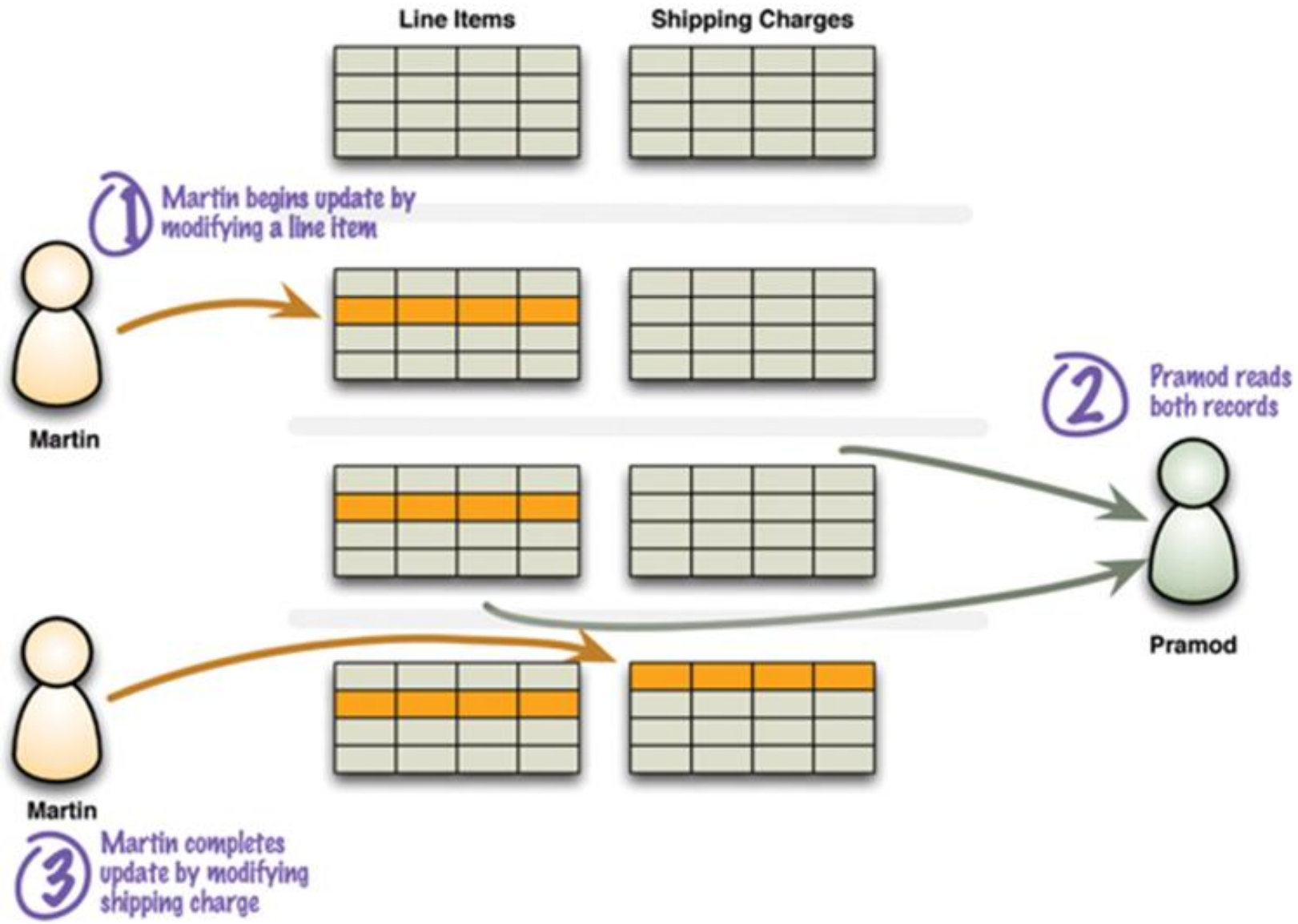
# Conflits : lecture impropre (1/3)

- **T1 et T2 transactions concurrentes**

| T1 | T2 |
|---|---|
| UPDATE comptes<br>SET solde = 25000<br>WHERE num_compte = '9' ; | |
| | SELECT solde<br>FROM comptes<br>WHERE num_compte = '9' ; |
| ROLLBACK | |



# Conflits : lecture impropre (2/3)



# Conflits : exemple de lecture impropre (3/3)

- **T1 et T2 transactions concurrentes**
- **T1 transfère 10 euros du compte '9' vers le compte '5'**
- **T2 affiche le solde total des 2 comptes**

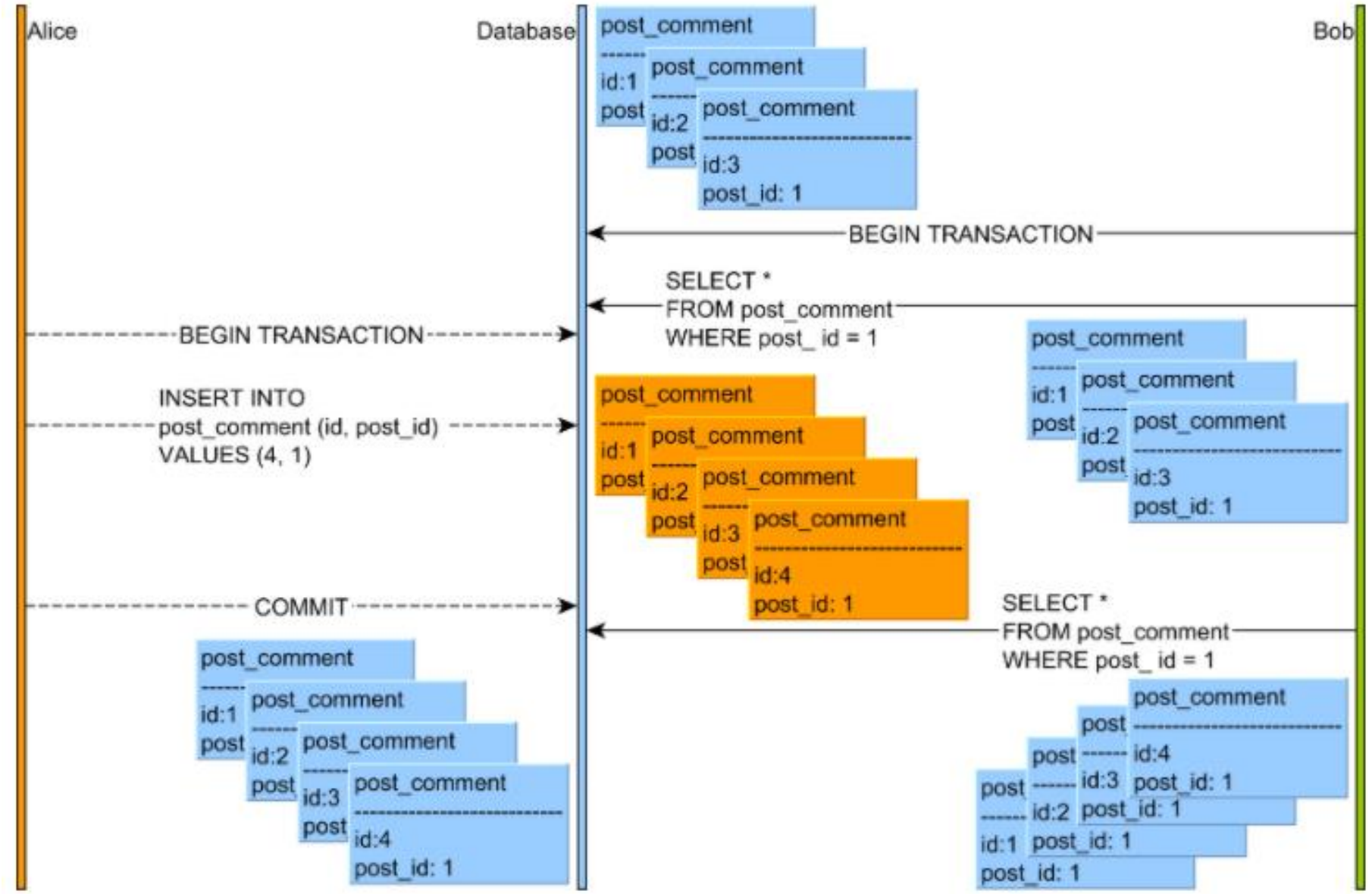
| T1 | T2 |
|---|---|
| UPDATE Comptes<br>SET solde = solde - 10<br>WHERE num_compte = '9' ; | |
| | SELECT SUM (solde)<br>FROM Comptes<br>WHERE num_compte in ('9', '5') ; |
| UPDATE Comptes<br>SET solde = solde + 10<br>WHERE num_compte = '5' ; | |
| COMMIT | |

# Conflits : lecture non reproductible

- **T1 et T2 transactions concurrentes**
- **T1 effectue plusieurs fois la lecture du même objet**


| T1 | T2 |
|---|---|
| SELECT points FROM Resultats<br>WHERE num_cours = 5 AND num_eleve = 7 ; | |
| | UPDATE Resultats<br>SET points = points * 1.1<br>WHERE num_cours = 5<br>AND nom_eleve = 7 ; |
| SELECT points FROM Resultats<br>WHERE num_cours = 5 AND num_eleve = 7 ; | |
| COMMIT | |

# Conflits : lecture fantôme



# Anomalies d'isolation

3 types d'anomalies d'isolation défini dans ANSI SQL92 :

- **Lectures sales ou impropres**

  Une transaction T1 lit des modifications non validées d'items effectuées par T2.

  En cas de annulation de T2, T1 a lu des valeurs invalides

- **Lecture non reproductibles**

  T1 lit un item, T2 modifie ce même item, T1 relit ce item et obtient une valeur différente

- **Lectures fantômes**

  T1 lit un ensemble de nuplets, T2 ajoute/supprime des nuplets, T1 relit l'ensemble de nuplets et obtient un ensemble différent comme résultat

# Degrés d'isolation en SQL

| Degré | Lecture impropre | Lecture non reproductible | Références fantômes |
|---|---|---|---|
| READ UNCOMMITTED | OUI | OUI | OUI |
| READ COMMITTED | NON | OUI | OUI |
| REPEATABLE READ | NON | NON | OUI |
| SERIALIZABLE | NON | NON | NON |

# Degrés d'isolation et conflits

- **Degré 0**

  Résolution des pertes de mises à jour

- **Degré 1**

  Pas d'annulation en cascade + Degré 0

- **Degré 2**

  Pas de lecture impropre + Degré 1

- **Degré 3**

  Isolation totale - Mise en attente des transactions en conflit

# Gestion des conflits

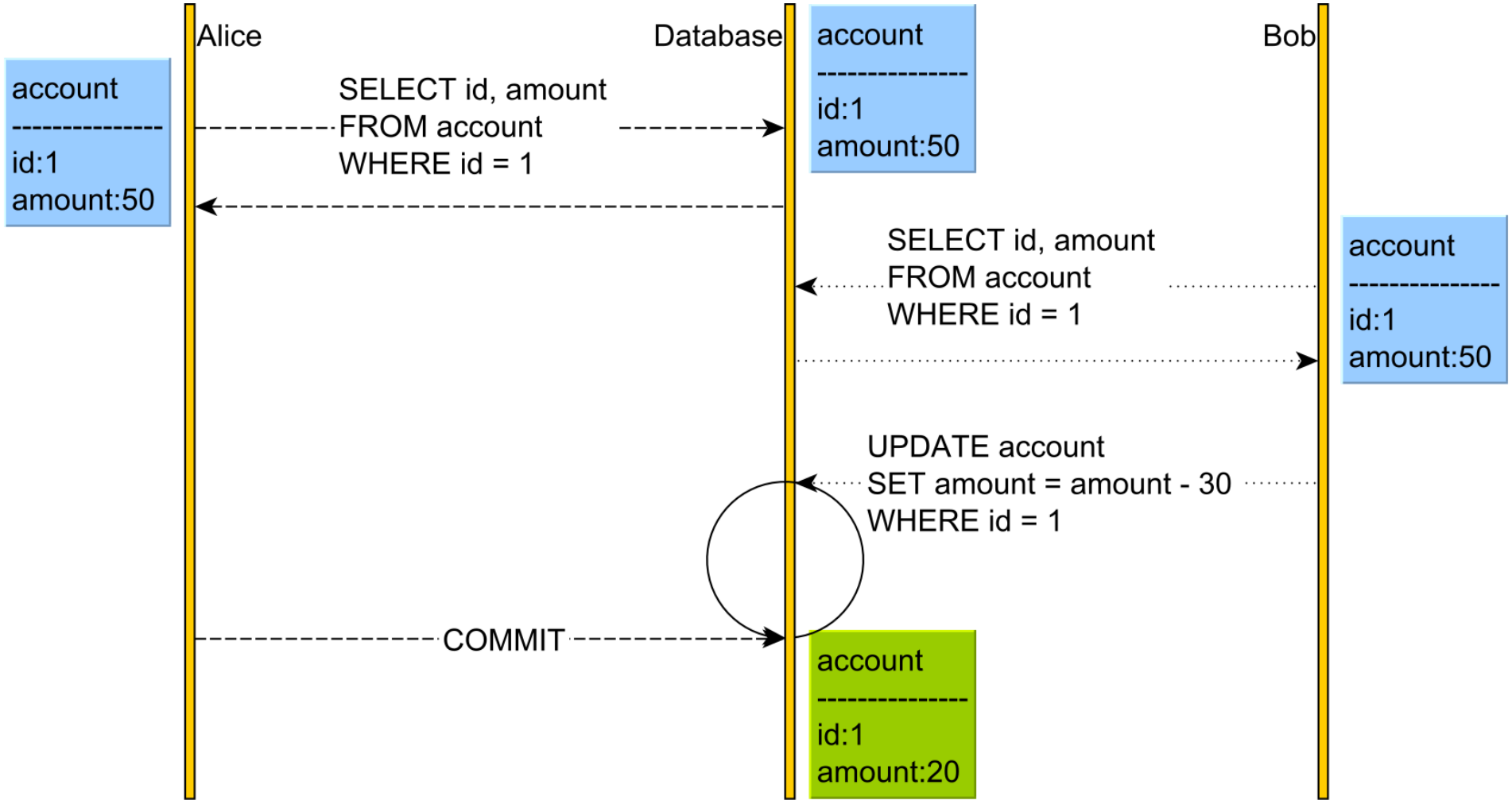
- Approche pessimiste : prévention des conflits par ex. en utilisant le verrouillage
- Approche optimiste : détection des conflits

# Pessimiste/Optimiste

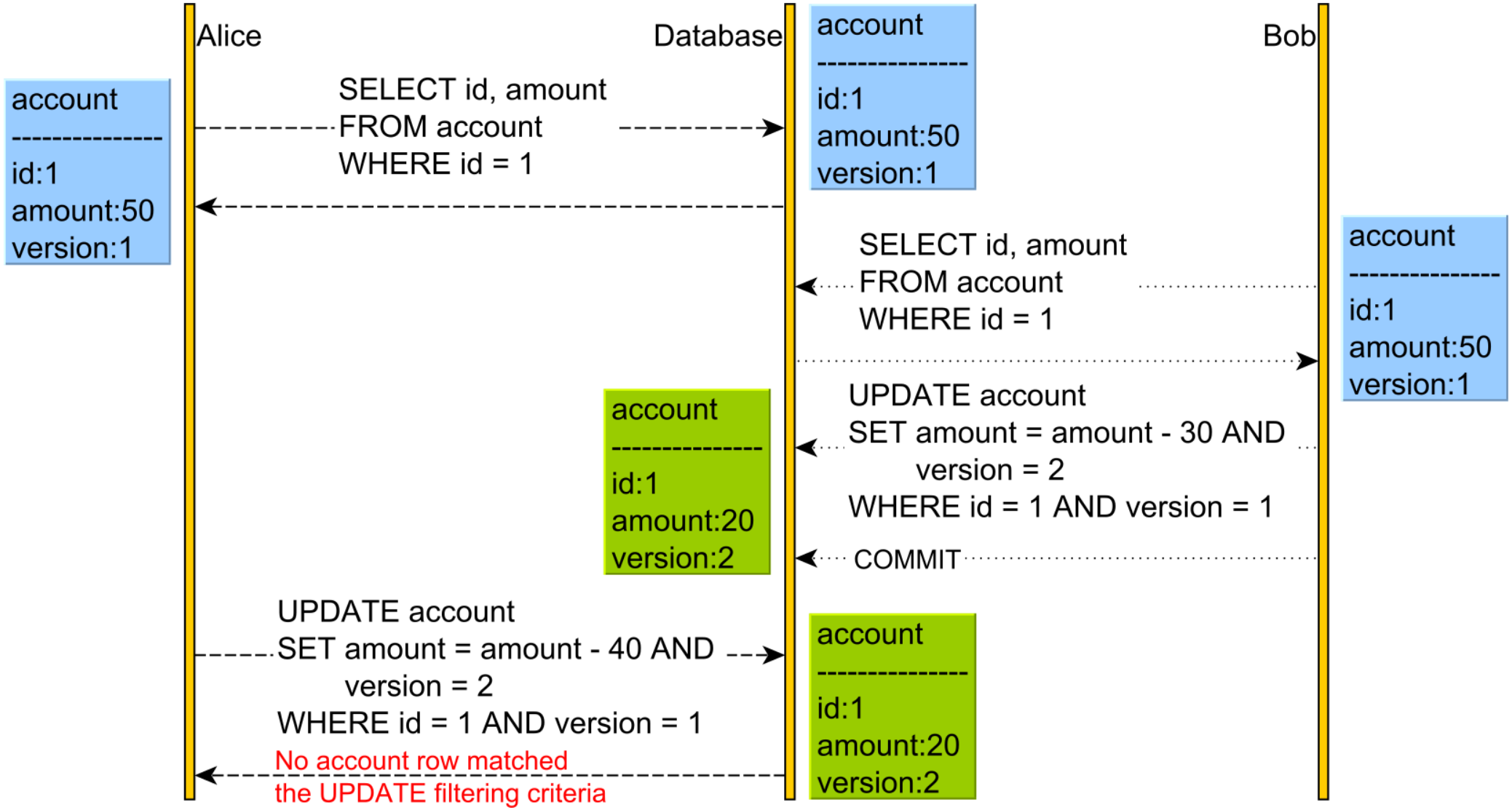
| | Niveau d'isolation verrouillage optimiste | | | |
|---|---|---|---|---|
| Anomalie transactionnelle | READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE |
| Lecture sale | Possible | Impossible | Impossible | Impossible |
| Lecture non répétable | Possible | Possible | Impossible | Impossible |
| Lecture fantôme | Possible | Possible | Possible | Impossible |
| Perte de mise à jour | Possible | Possible | Possible | Possible |

| | Niveau d'isolation verrouillage pessimiste | | | |
|---|---|---|---|---|
| Anomalie transactionnelle | READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE |
| Lecture sale | Possible | Impossible | Impossible | Impossible |
| Lecture non répétable | Possible | Possible | Impossible | Impossible |
| Lecture fantôme | Possible | Possible | Possible | Impossible |
| Perte de mise à jour | Impossible | Impossible | Impossible | Impossible |

# Verrouillage Pessimiste

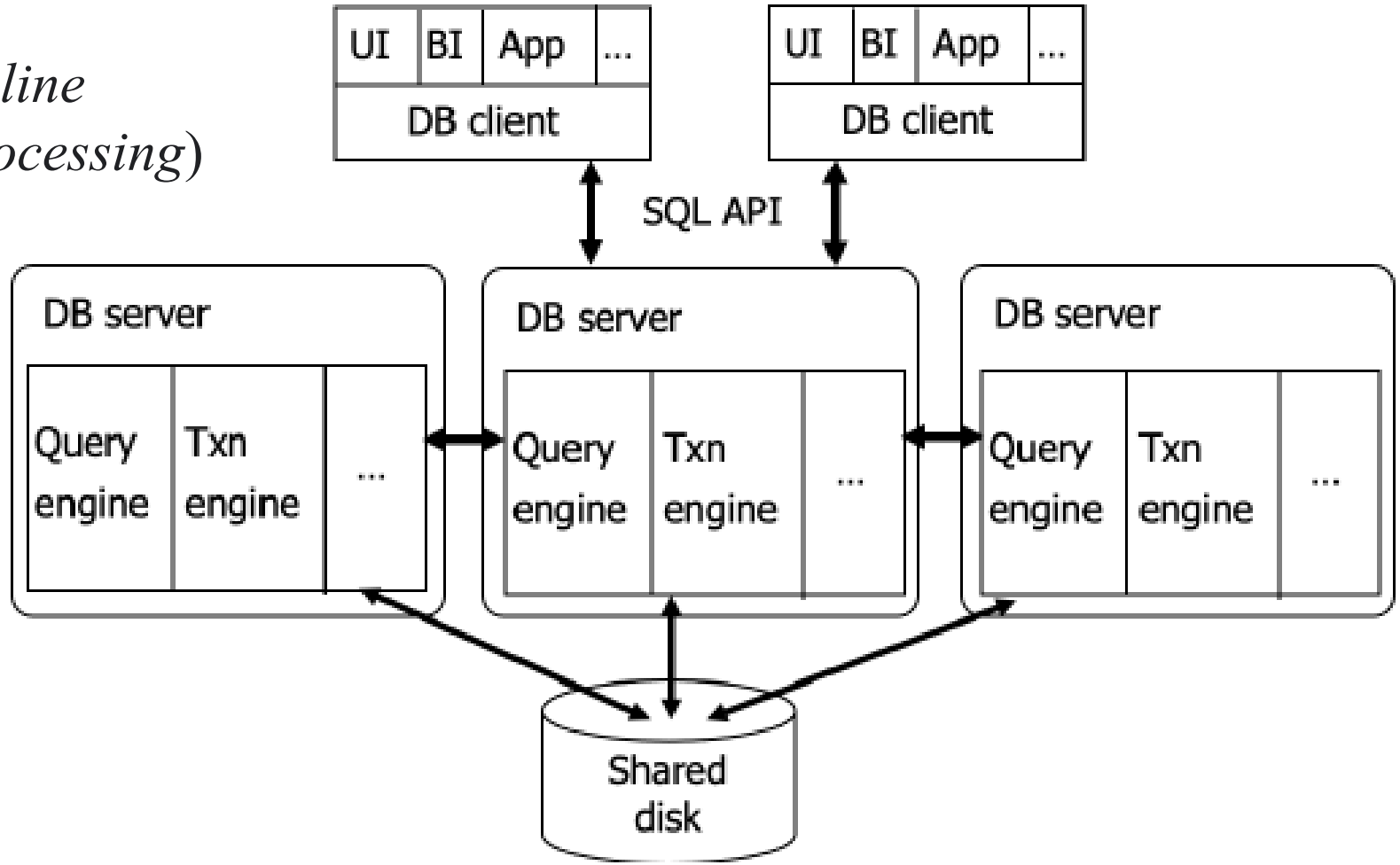


# Verrouillage Optimiste



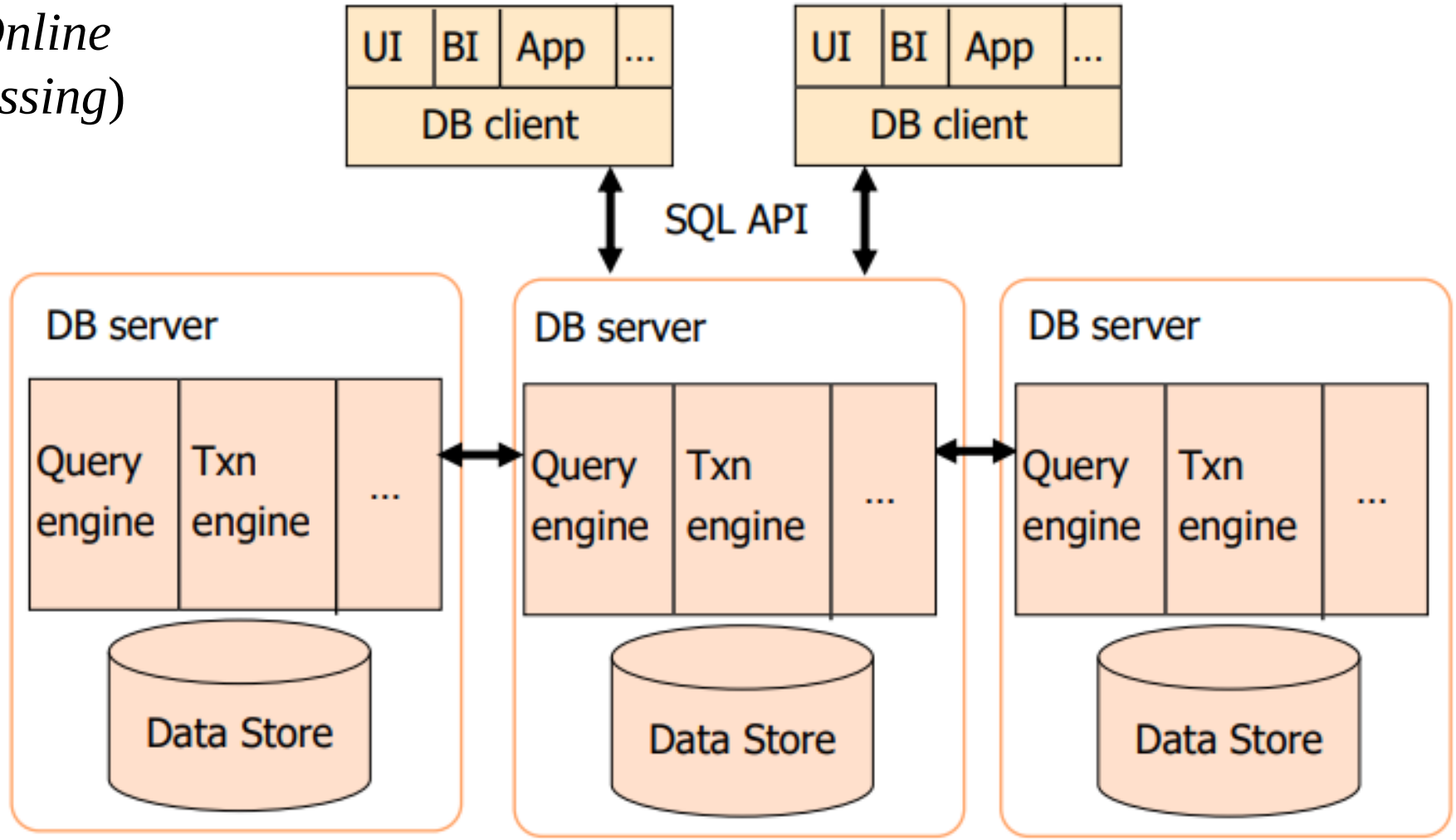
# *Shared-disk architecture*

Pour l'OLTP (*Online Transactional Processing*)

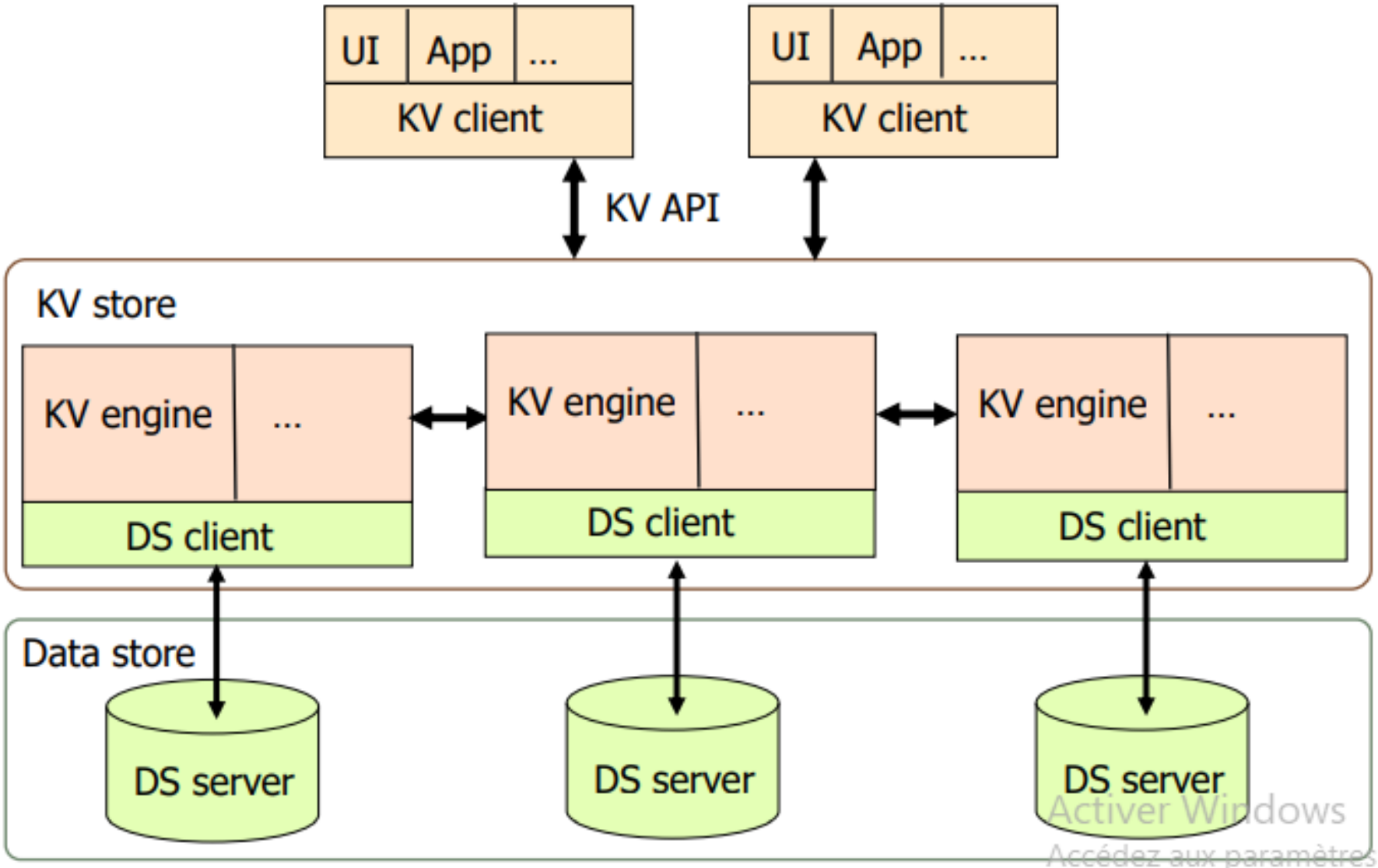


# *Shared-nothing* architecture

Pour l'**OLAP** (*Online Analytical Processing*)

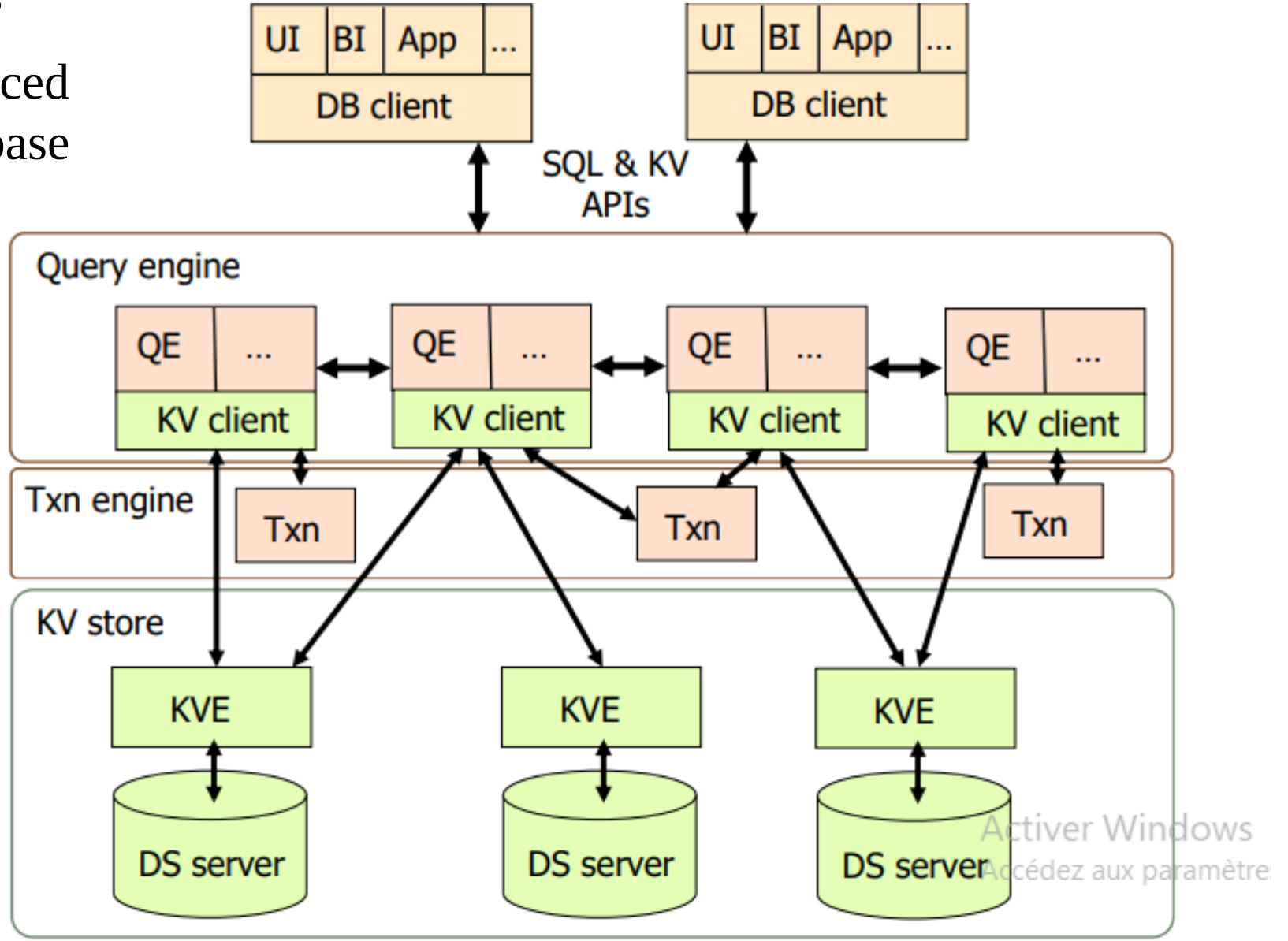


# *Key-value Store Distributed Architecture*

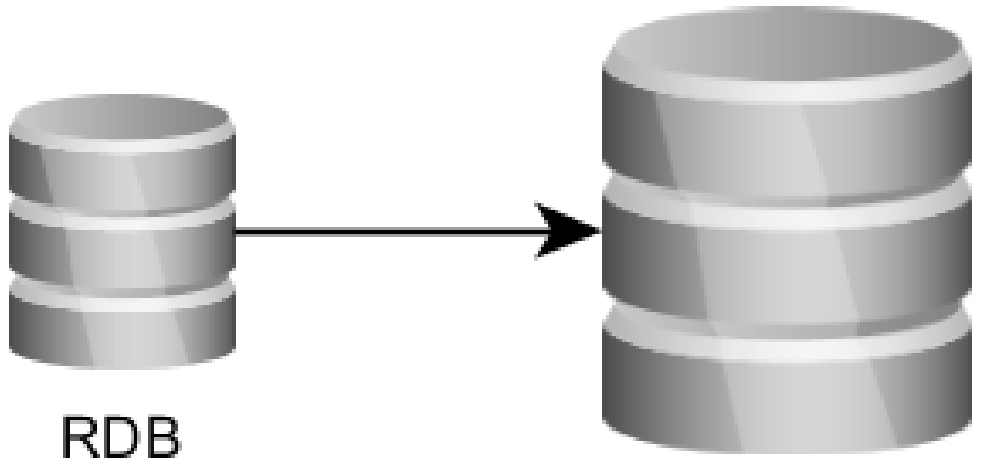


# *NewSQL Distributed Architecture*

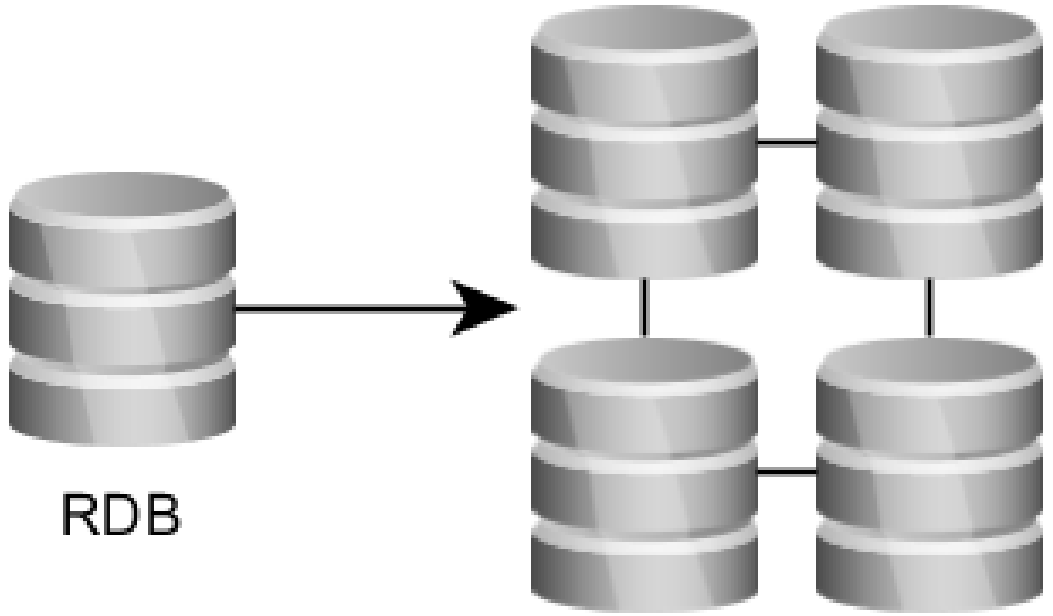
Architecture of the most advanced NewSQL database systems



# Passage à l'échelle (*Scalability/Elasticity*)




**Vertical**




**Horizontal**

# Répartition des données et réplication

## Différentes approches de répartition :

- **Partitionnement (*sharding*)**
  Différentes données réparties sur différents nœuds

- **Maître-esclave**
  Copie d'une même donnée sur différents nœuds

# *Sharding*

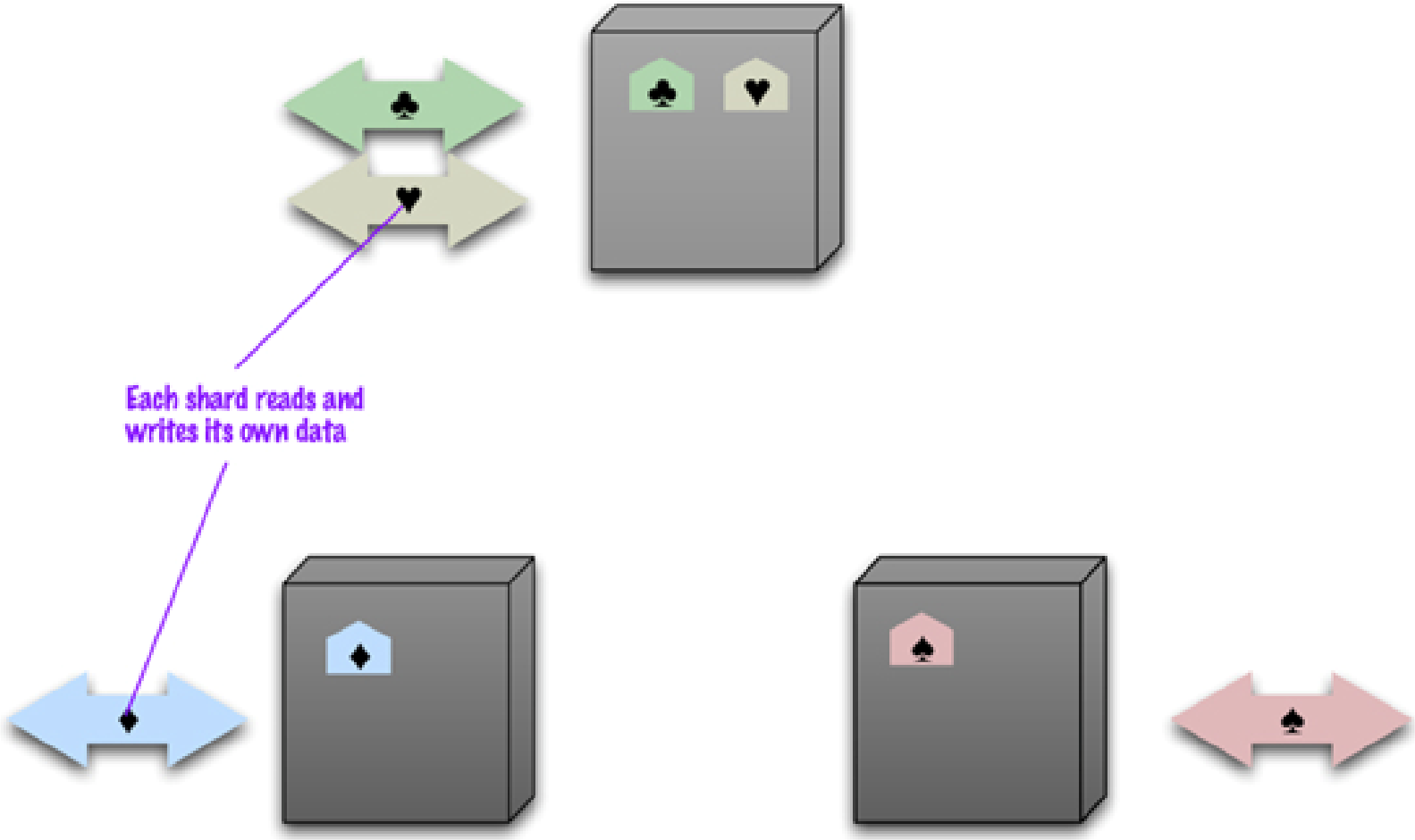
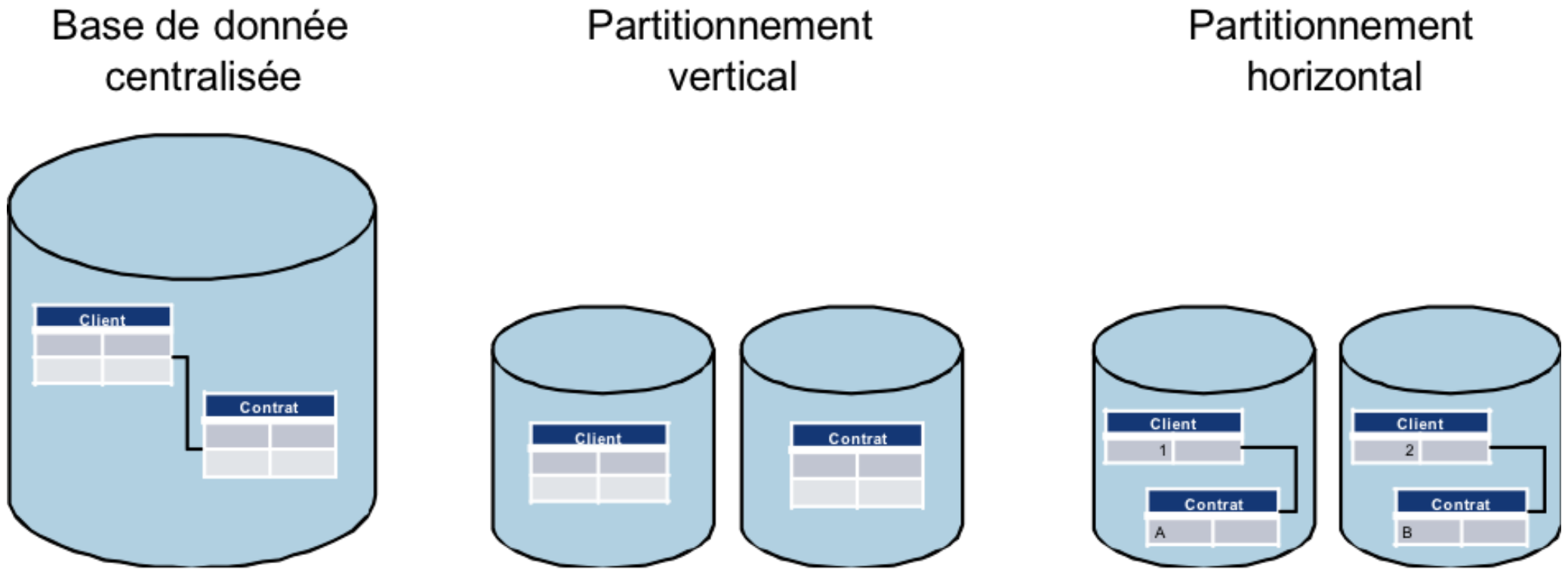


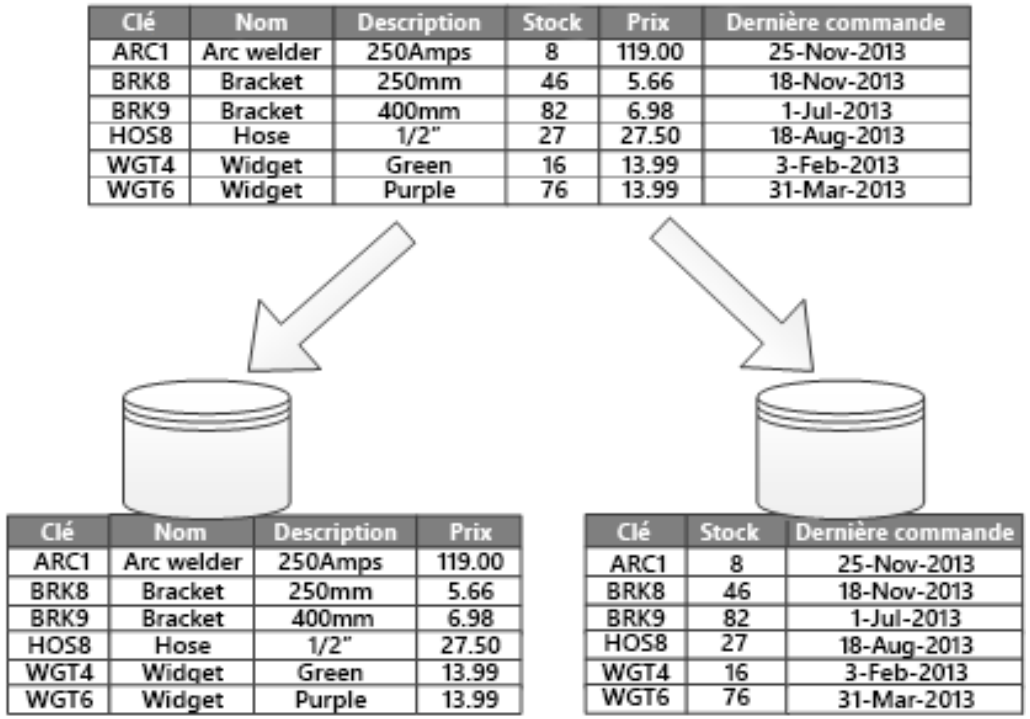

**Figure 4.1. Sharding puts different data on separate nodes, each of which does its own reads and writes.**

# Partitionnement

- **Vertical :** différentes tables ou colonnes réparties sur différents serveur
- **Horizontal** : chaque serveur est responsable d'un sous ensemble de données (identifié par un intervalle de clés)



# Partitionnement horizontale / vertical

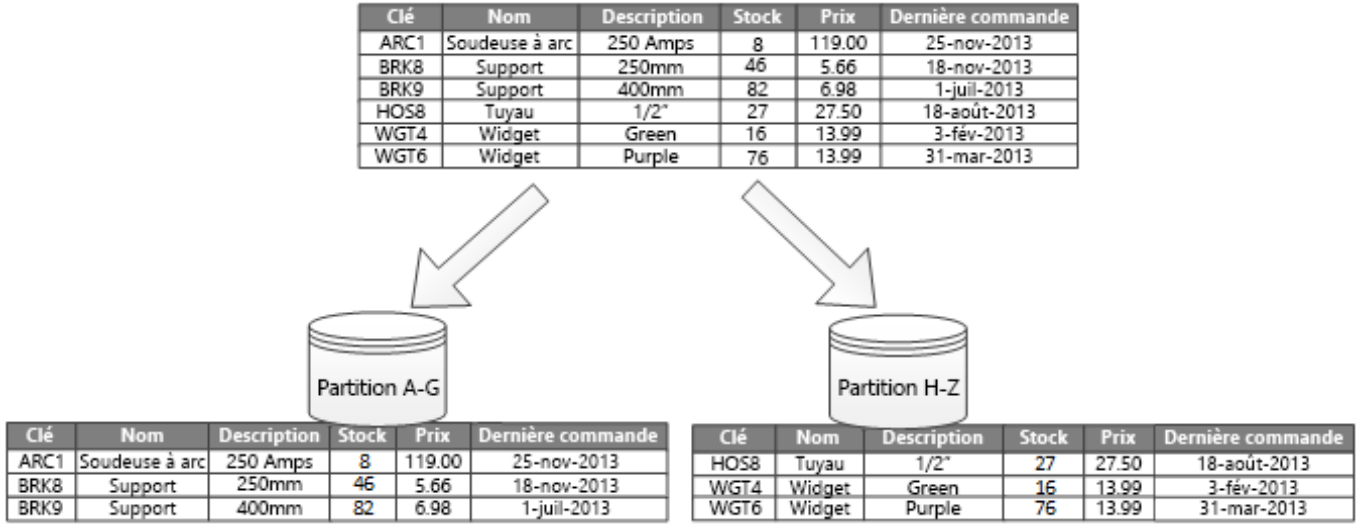


| Clé | Nom | Description | Stock | Prix | Dernière commande |
|---|---|---|---|---|---|
| ARC1 | Arc welder | 250Amps | 8 | 119.00 | 25-Nov-2013 |
| BRK8 | Bracket | 250mm | 46 | 5.66 | 18-Nov-2013 |
| BRK9 | Bracket | 400mm | 82 | 6.98 | 1-Jul-2013 |
| HOS8 | Hose | 1/2" | 27 | 27.50 | 18-Aug-2013 |
| WGT4 | Widget | Green | 16 | 13.99 | 3-Feb-2013 |
| WGT6 | Widget | Purple | 76 | 13.99 | 31-Mar-2013 |

| Clé | Nom | Description | Prix |
|---|---|---|---|
| ARC1 | Arc welder | 250Amps | 119.00 |
| BRK8 | Bracket | 250mm | 5.66 |
| BRK9 | Bracket | 400mm | 6.98 |
| HOS8 | Hose | 1/2" | 27.50 |
| WGT4 | Widget | Green | 13.99 |
| WGT6 | Widget | Purple | 13.99 |

| Clé | Stock | Dernière commande |
|---|---|---|
| ARC1 | 8 | 25-Nov-2013 |
| BRK8 | 46 | 18-Nov-2013 |
| BRK9 | 82 | 1-Jul-2013 |
| HOS8 | 27 | 18-Aug-2013 |
| WGT4 | 16 | 3-Feb-2013 |
| WGT6 | 76 | 31-Mar-2013 |

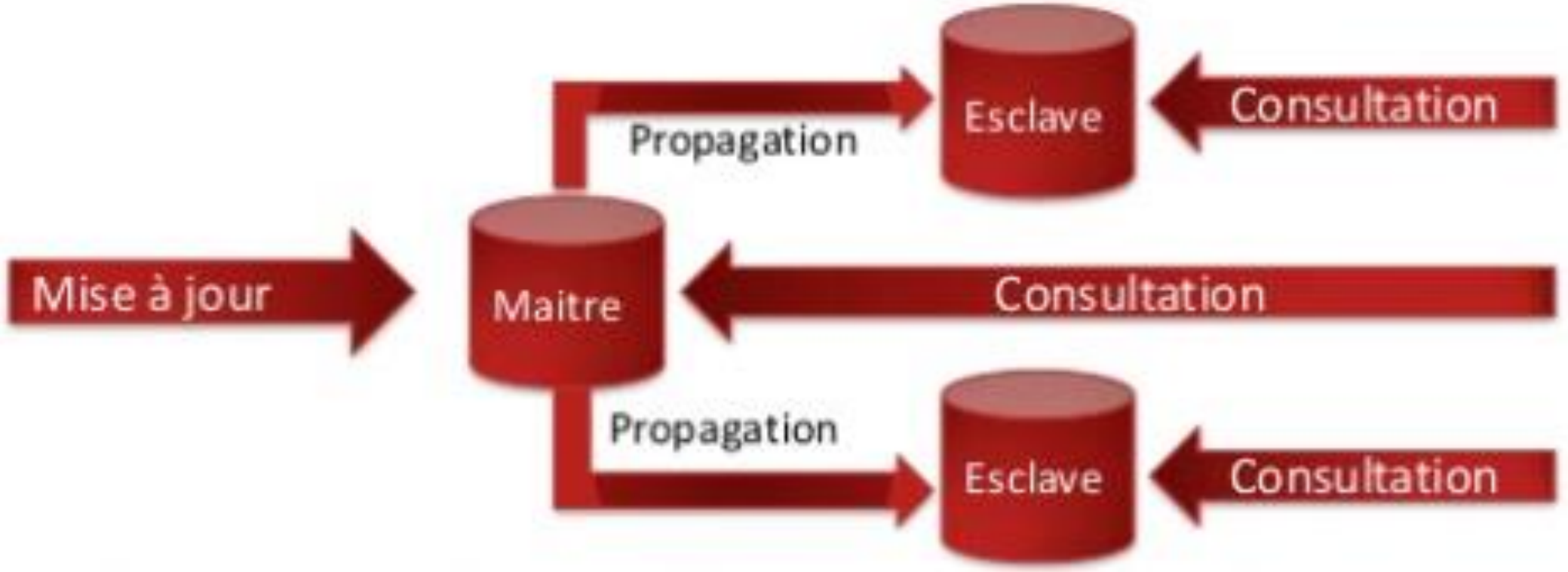
**Partitionnement vertical**



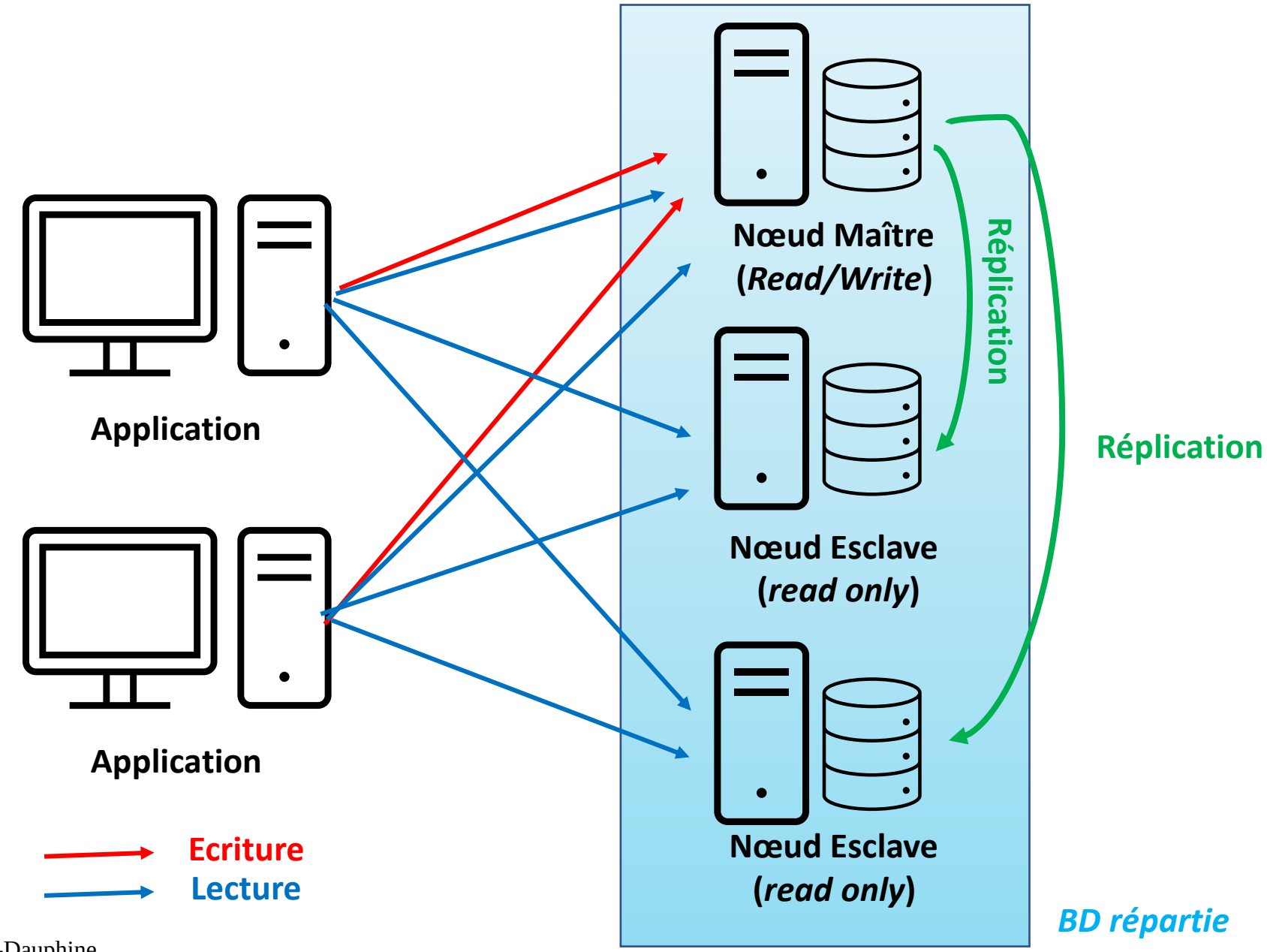

| Clé | Nom | Description | Stock | Prix | Dernière commande |
|---|---|---|---|---|---|
| ARC1 | Soudeuse à arc | 250 Amps | 8 | 119.00 | 25-nov-2013 |
| BRK8 | Support | 250mm | 46 | 5.66 | 18-nov-2013 |
| BRK9 | Support | 400mm | 82 | 6.98 | 1-juil-2013 |
| HOS8 | Tuyau | 1/2" | 27 | 27.50 | 18-août-2013 |
| WGT4 | Widget | Green | 16 | 13.99 | 3-fév-2013 |
| WGT6 | Widget | Purple | 76 | 13.99 | 31-mar-2013 |

**Partitionnement horizontal**

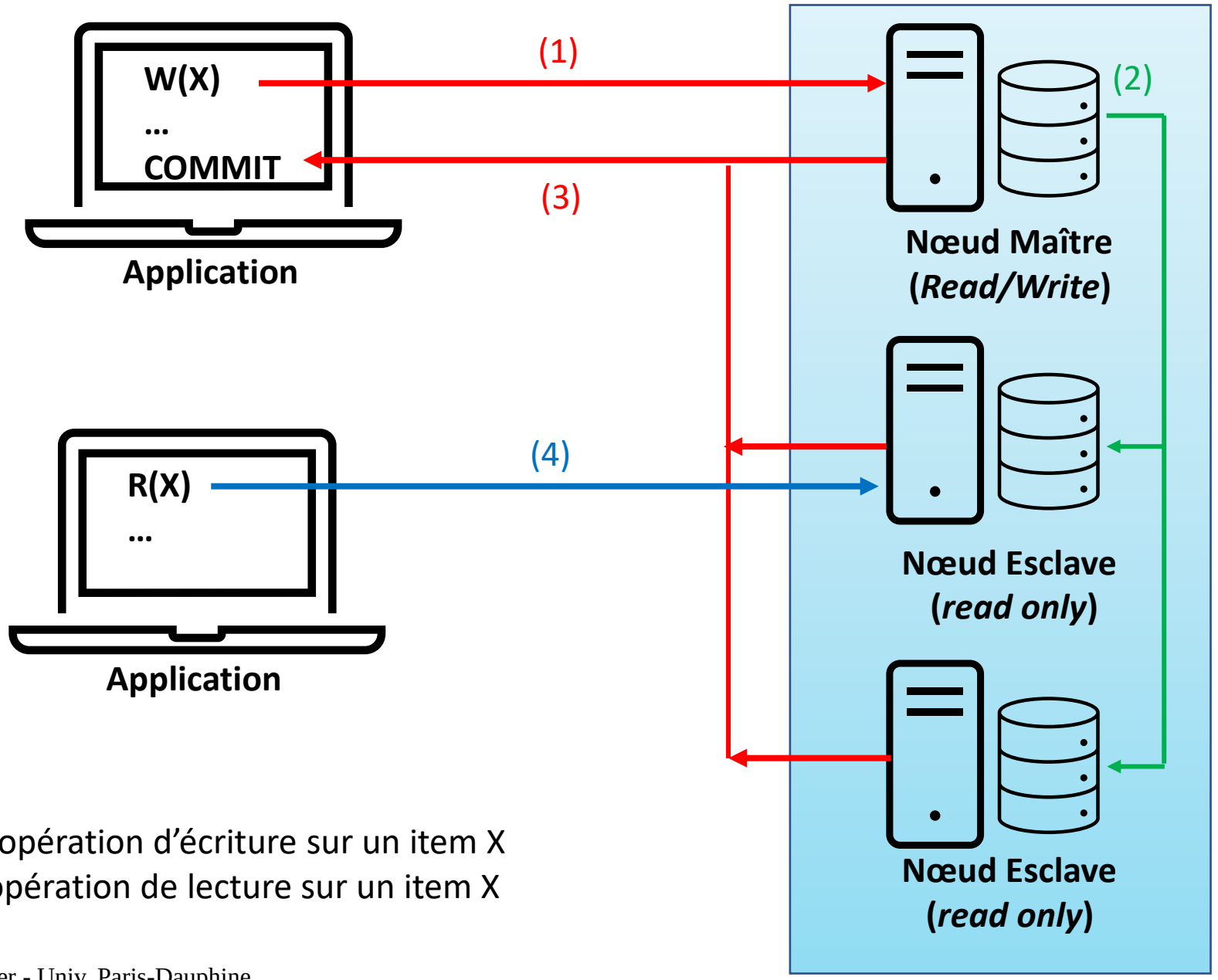
# Maître - Esclaves



# Architecture Maître – Esclave (1/4)



# Architecture Maître - Esclave (2/4)




(1) Ecriture sur le Maître

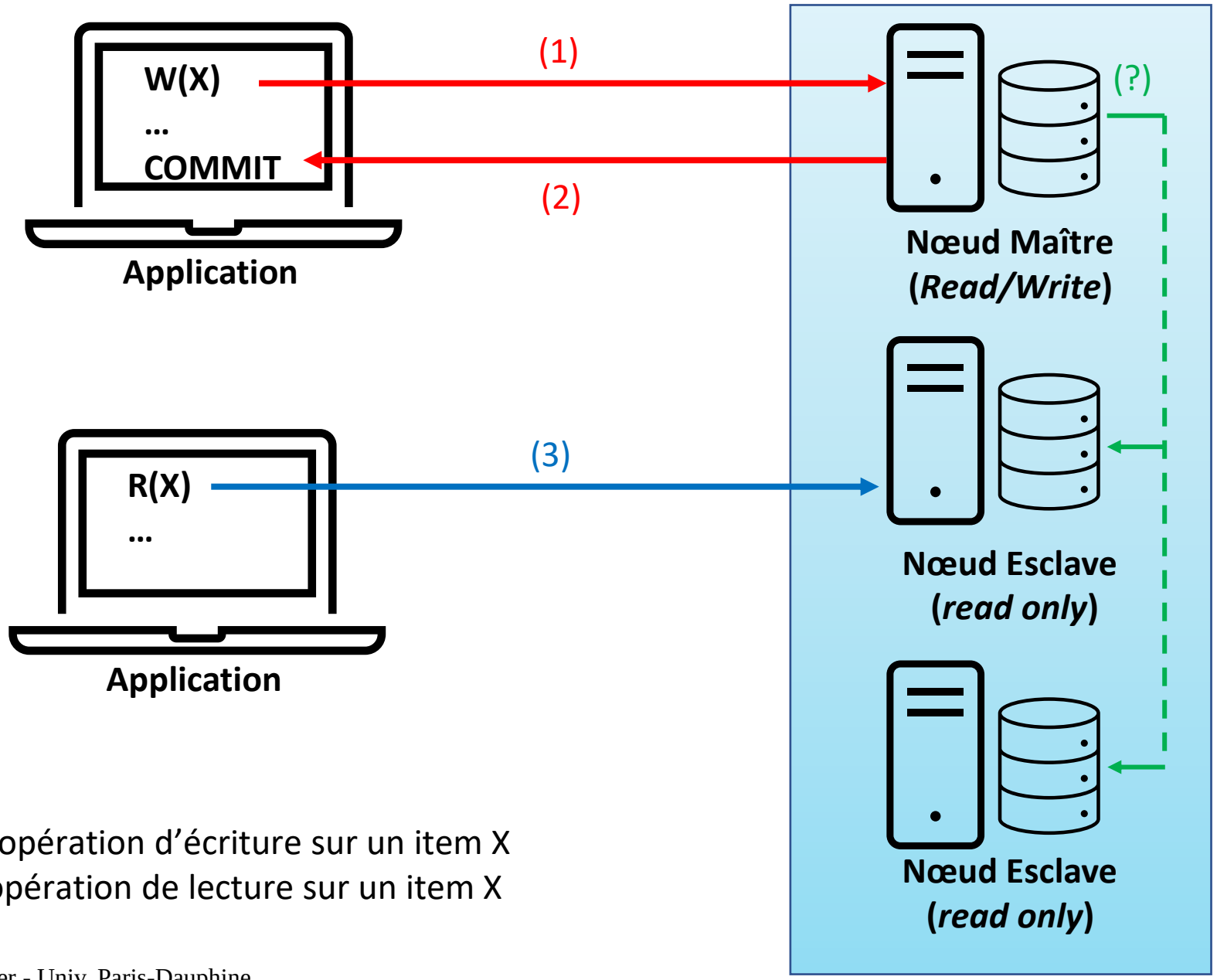
(2) Réplication synchrone de l'écriture sur les esclaves

(3) Validation de l'écriture et acquittement au client

(4) Lecture depuis n'importe quel nœud

W(X) : opération d'écriture sur un item X
R(X) : opération de lecture sur un item X

# Architecture Maître - Esclave (3/4)




(1) Ecriture sur le Maître

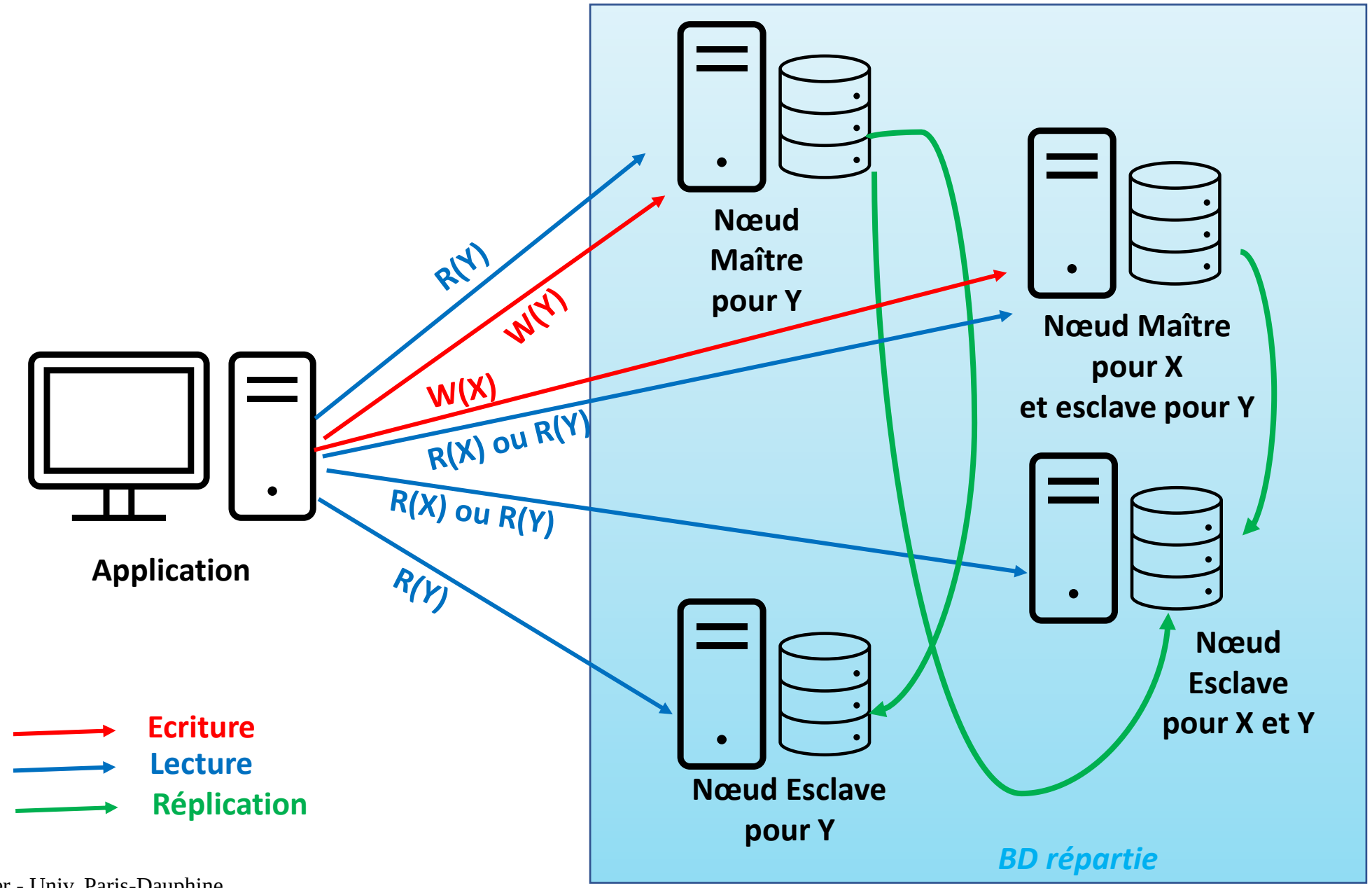
(2) Validation de l'écriture et acquittement au client

(3) Lecture depuis n'importe quel nœud

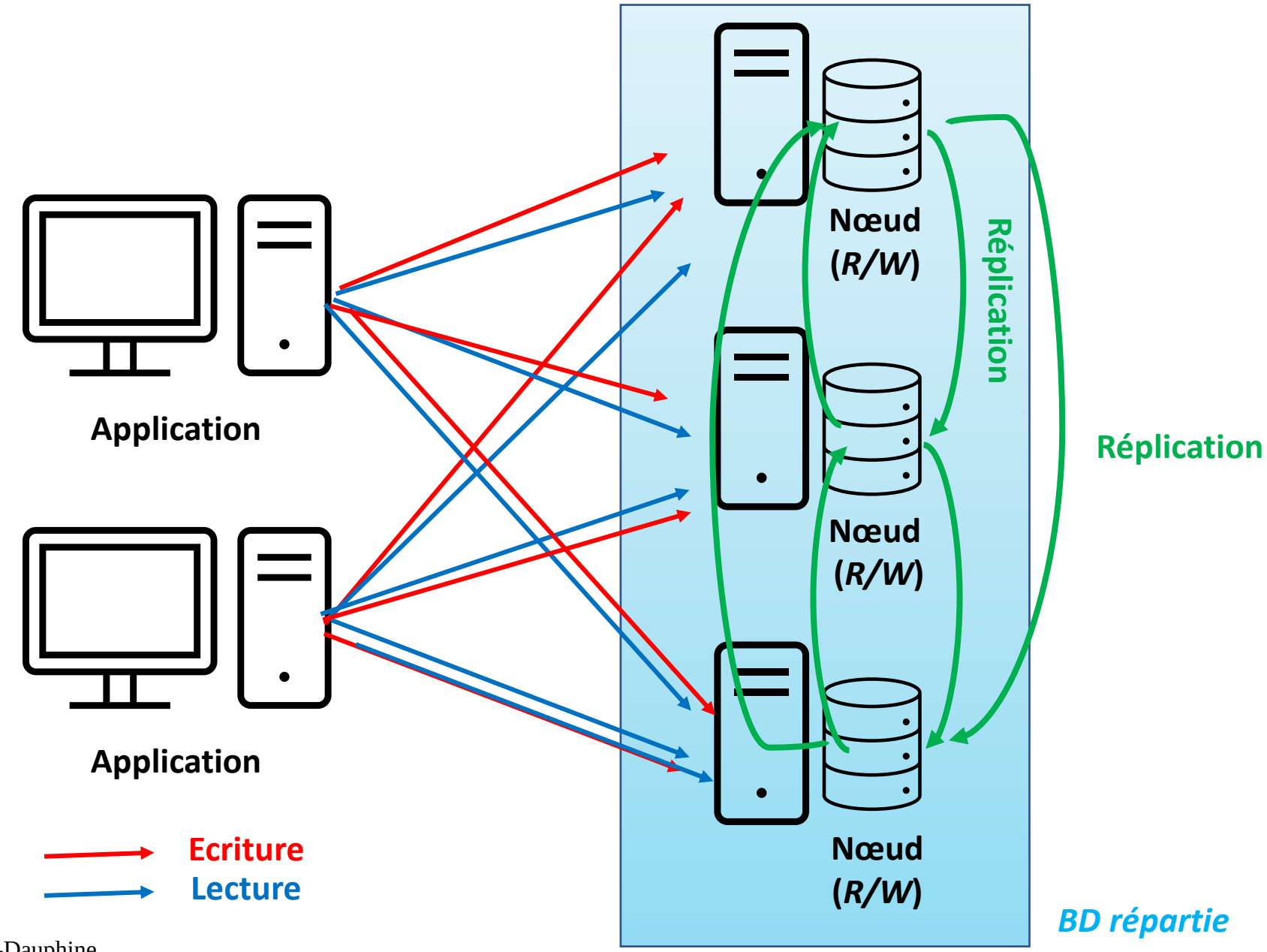
(?) Réplication asynchrone de l'écriture sur les esclaves (cohérence à terme)

W(X) : opération d'écriture sur un item X
R(X) : opération de lecture sur un item X

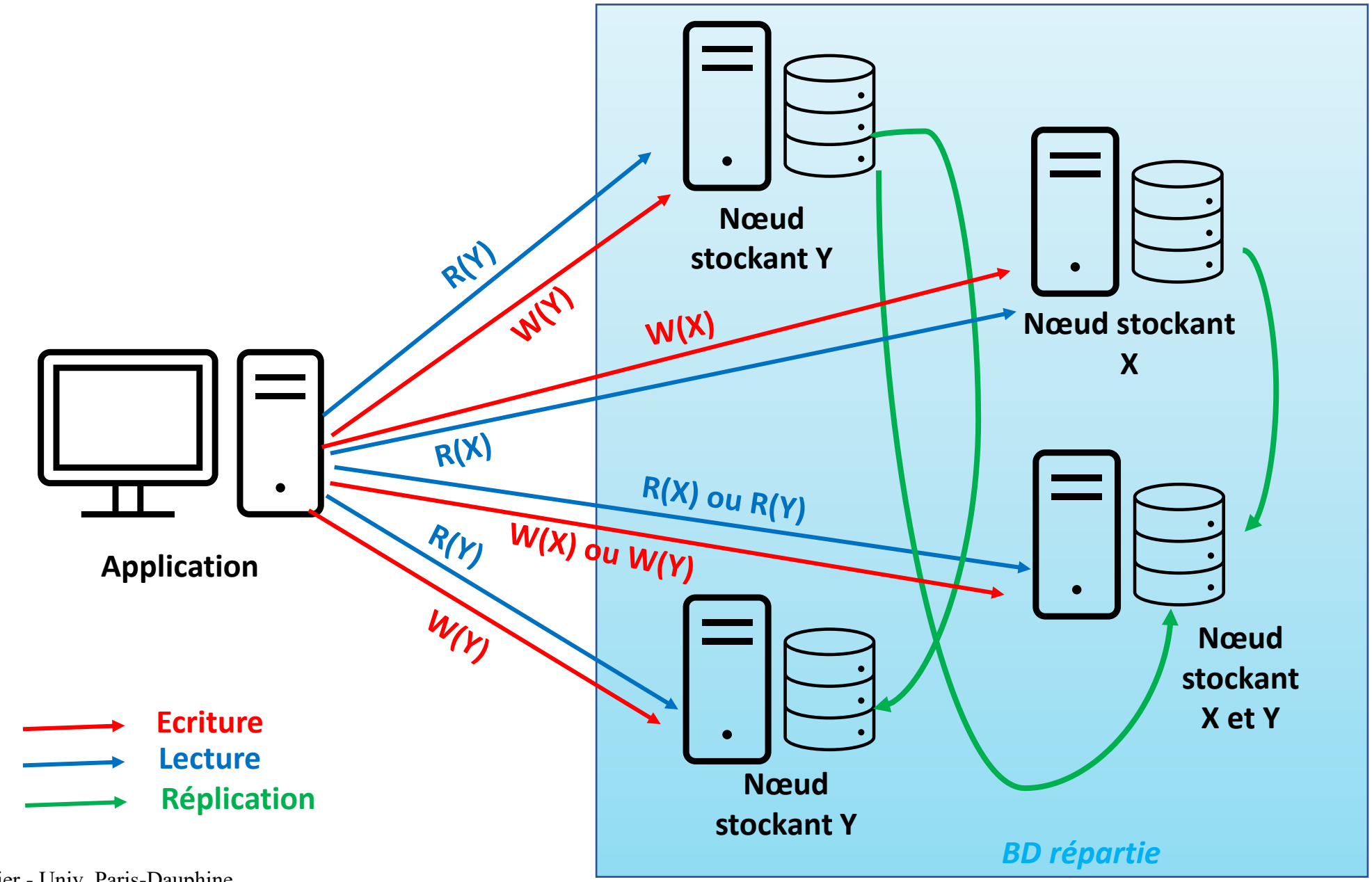
# Architecture Maître – Esclave (4/4)



# Architecture Pair à Pair (1/2)



# Architecture Pair à Pair(2/2)



# Maitre – Esclaves et réplication

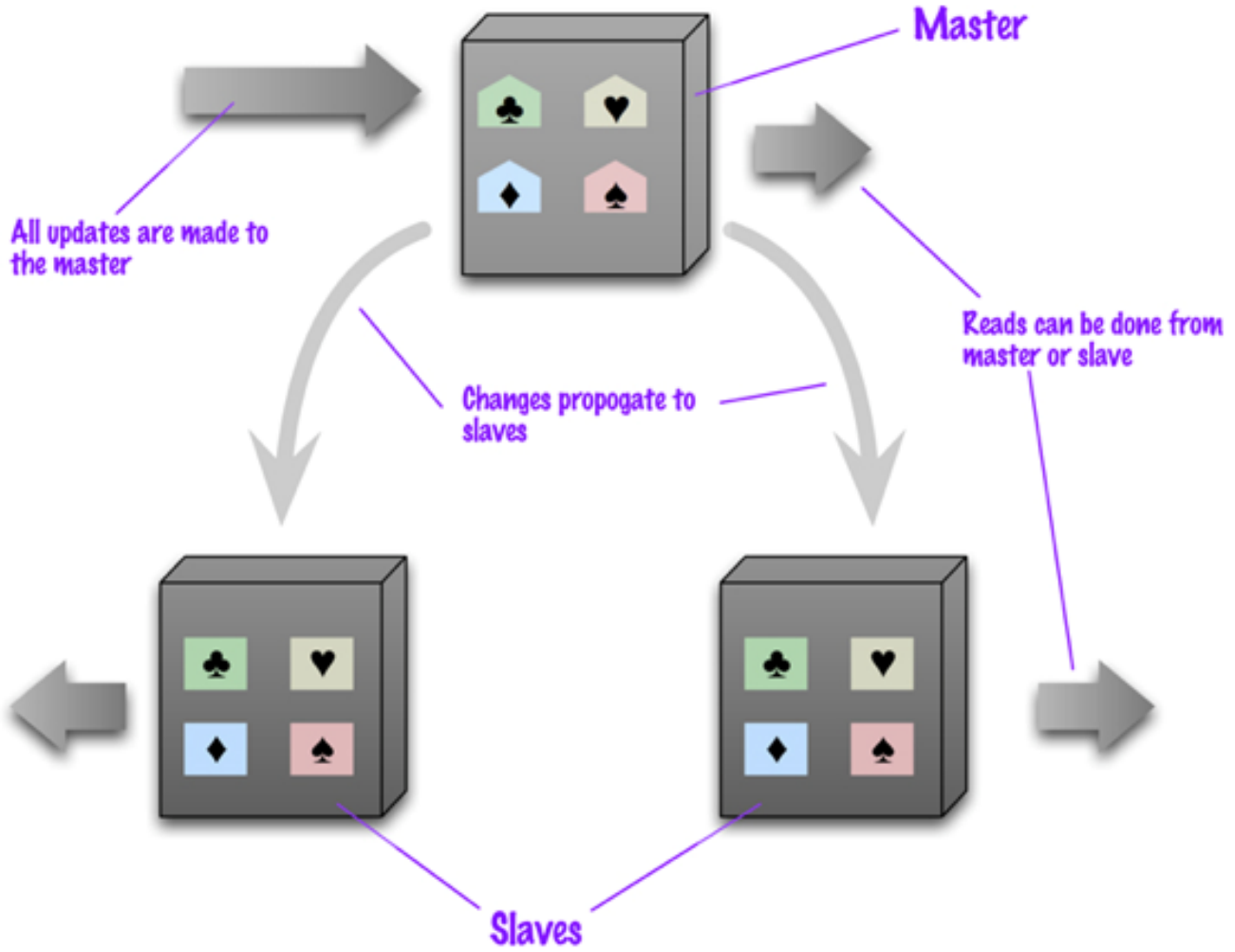



Figure 4.2. Data is replicated from master to slaves. The master services all writes; reads may come from either master or slaves.

# Réplication *peer-to-peer*

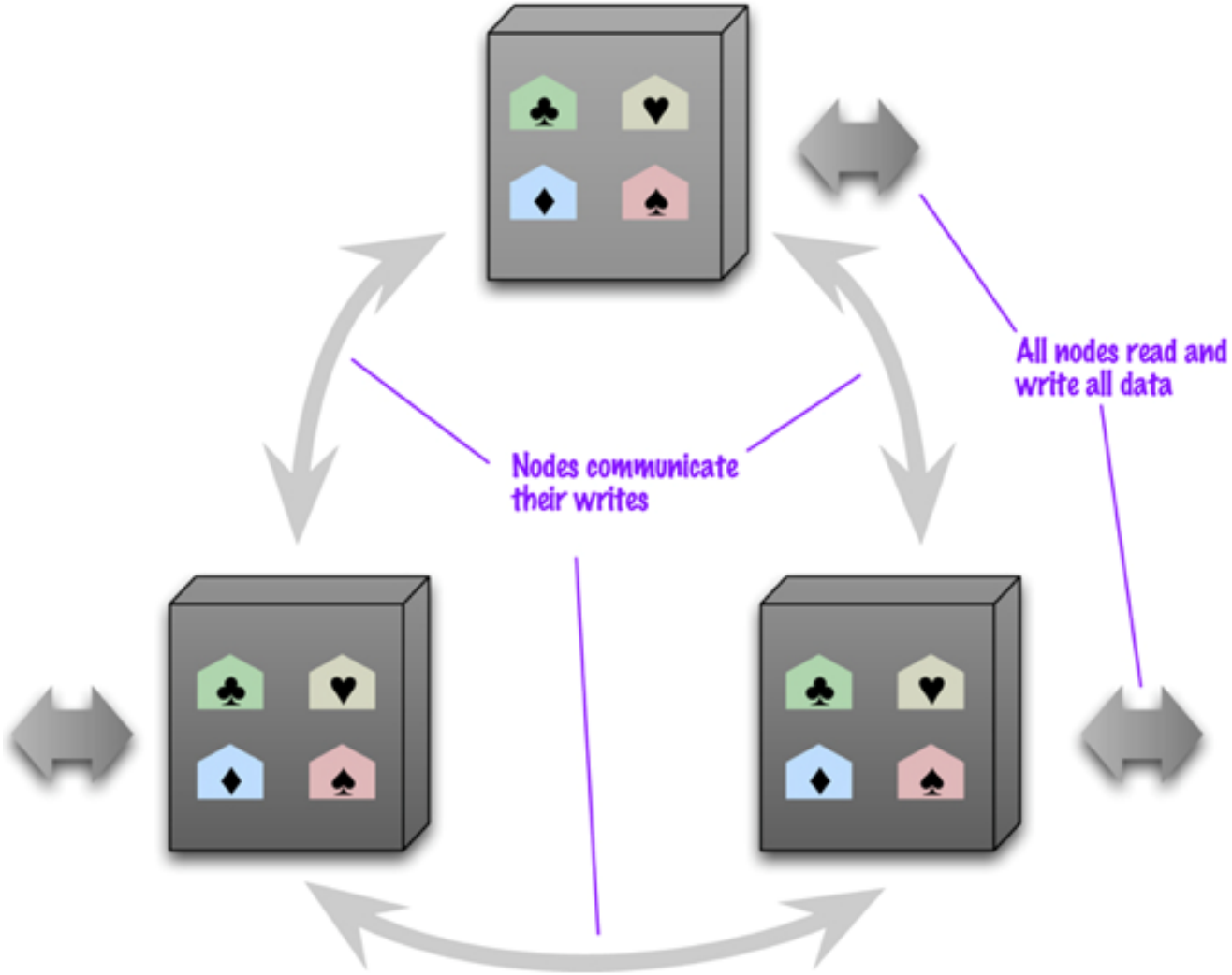



Figure 4.3. Peer-to-peer replication has all nodes applying reads and writes to all the data.

# Partitionnement et réplication : maître -esclaves

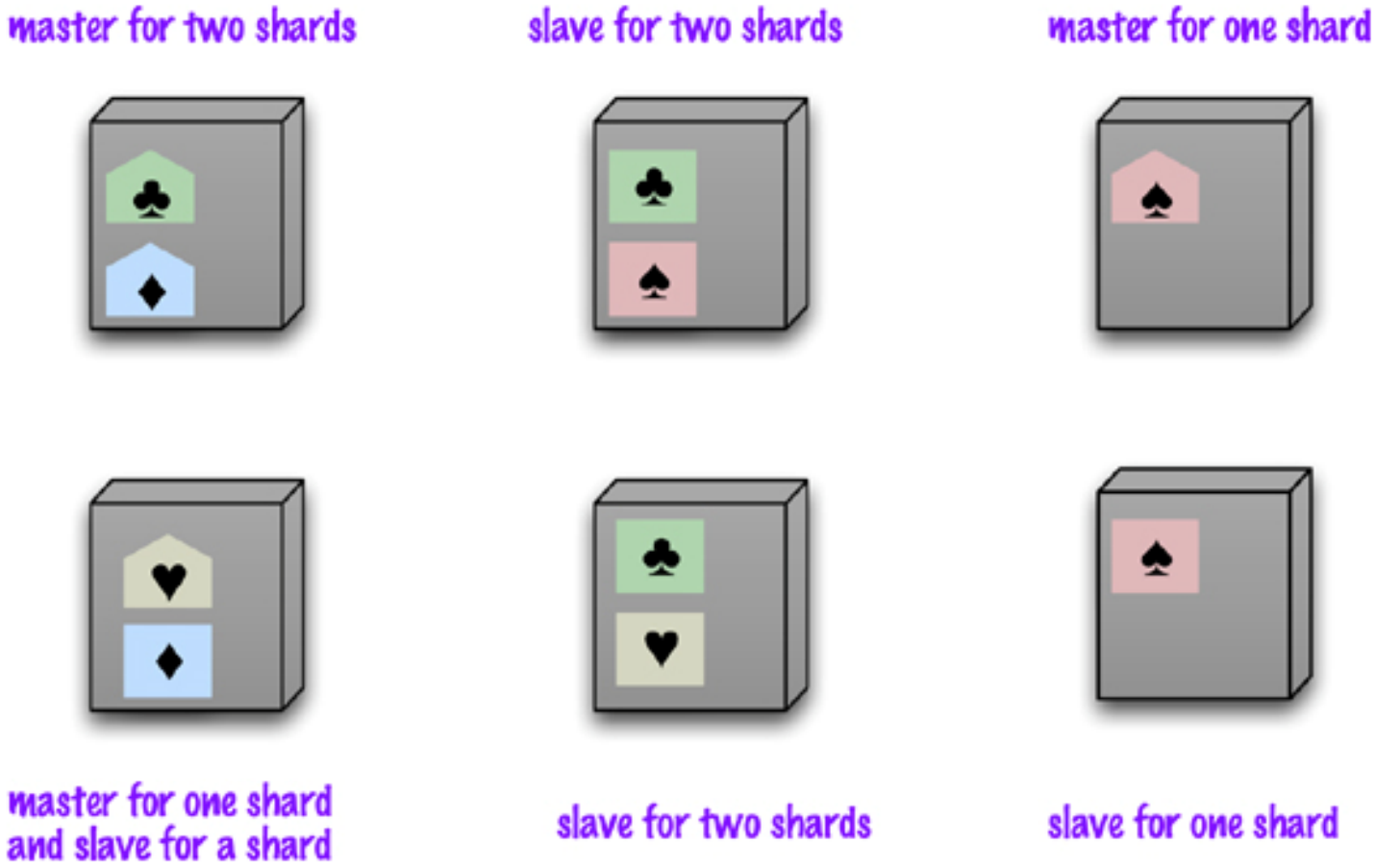



Figure 4.4. Using master-slave replication together with sharding

# Partitionnement et réplication : *peer-to-peer*

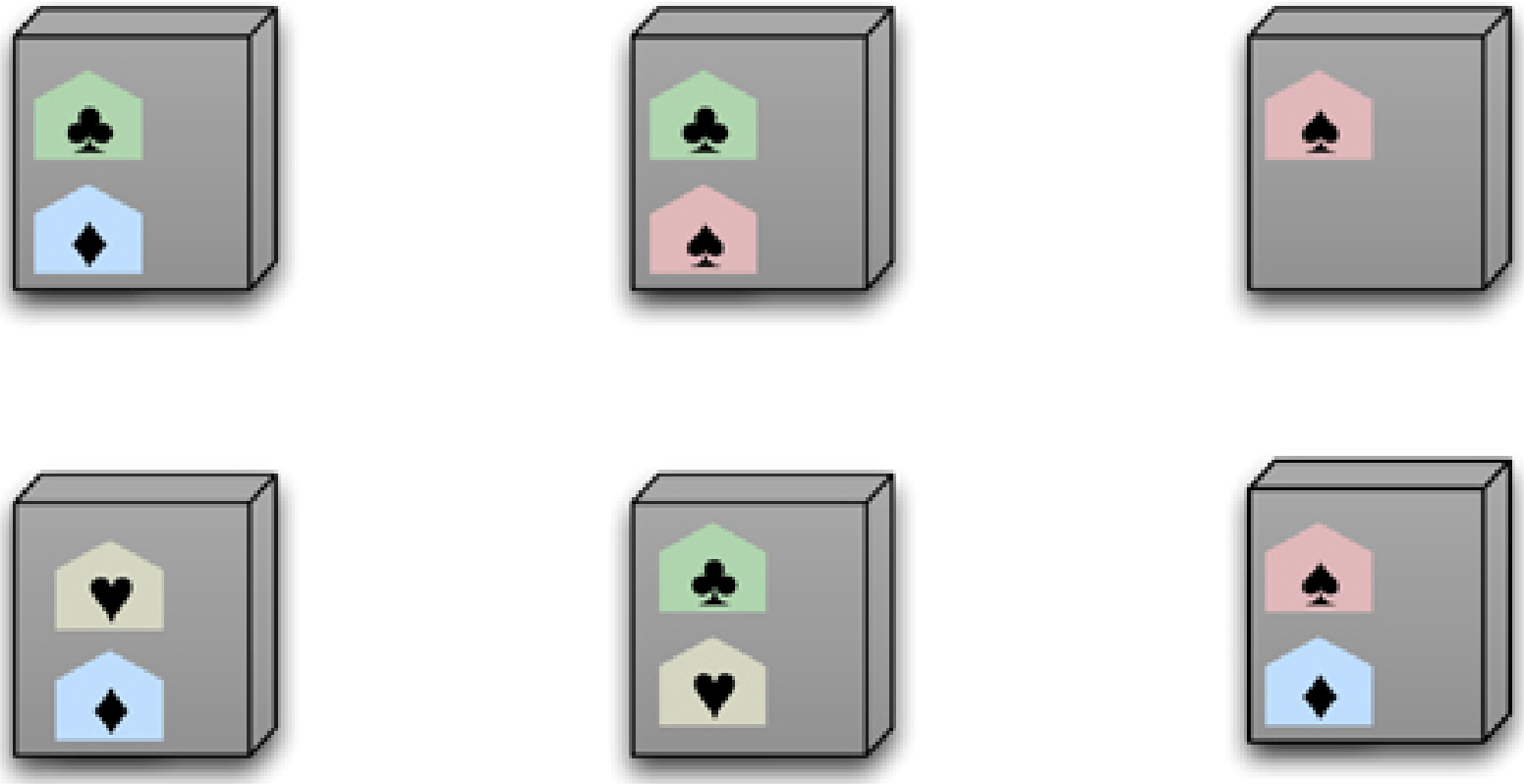


Figure 4.5. Using peer-to-peer replication together with sharding

# Réplication : une source d'incohérence

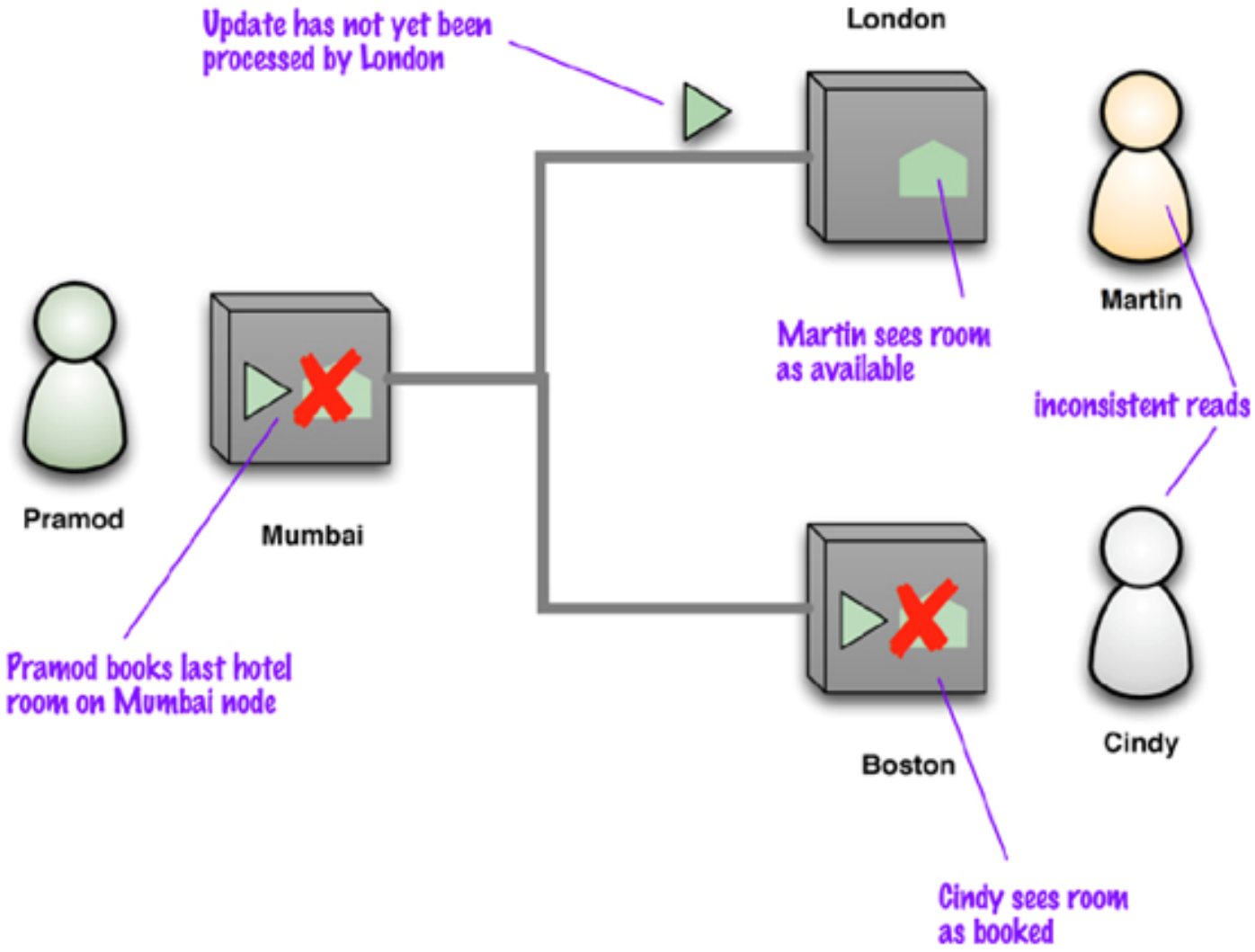
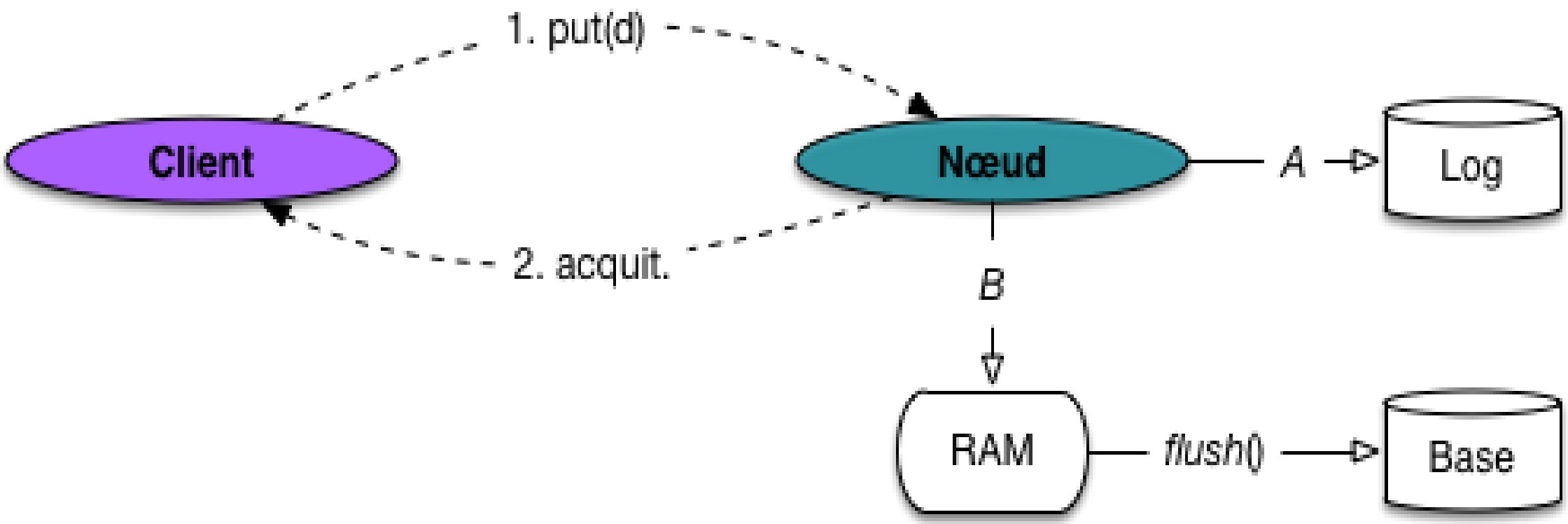


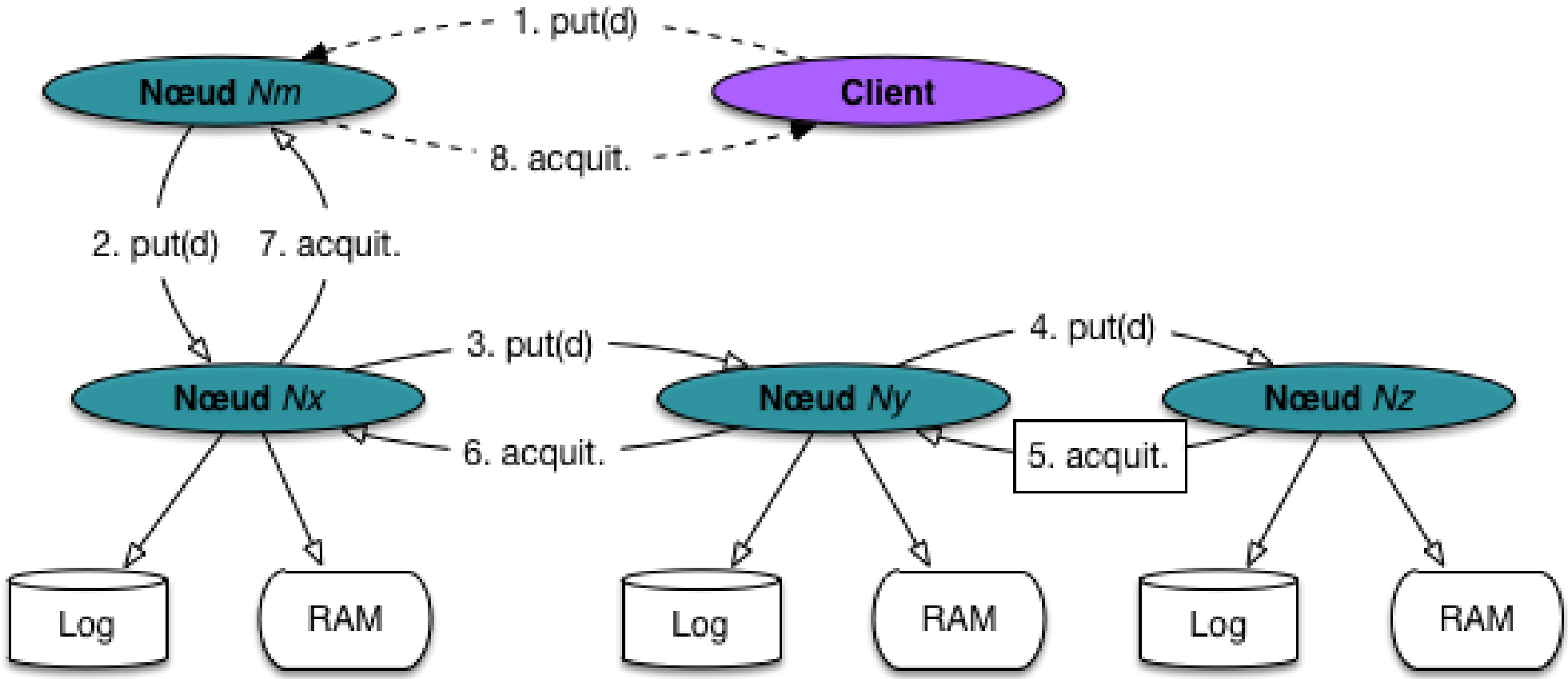

Figure 5.2. An example of replication inconsistency

# Réplication synchrone / asynchrone

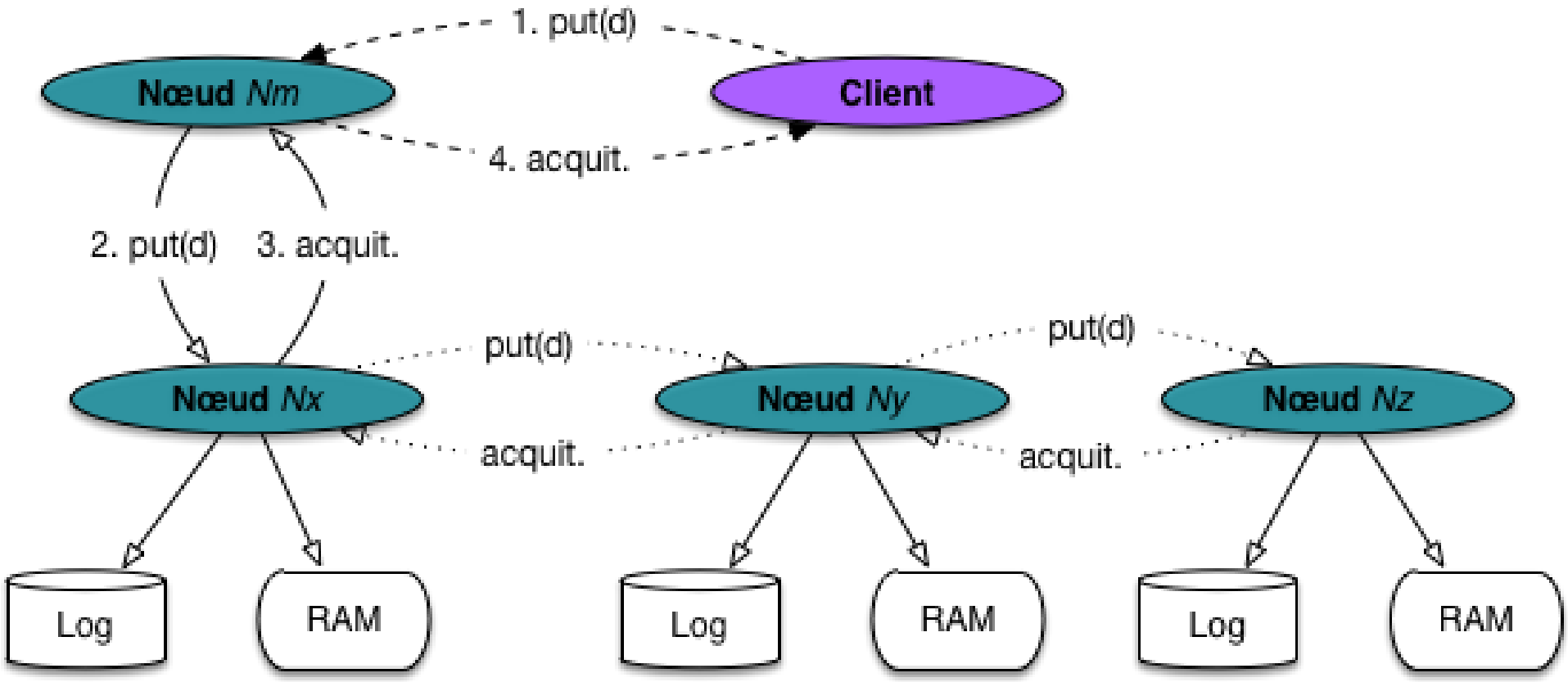



**Ecriture avec journalisation**

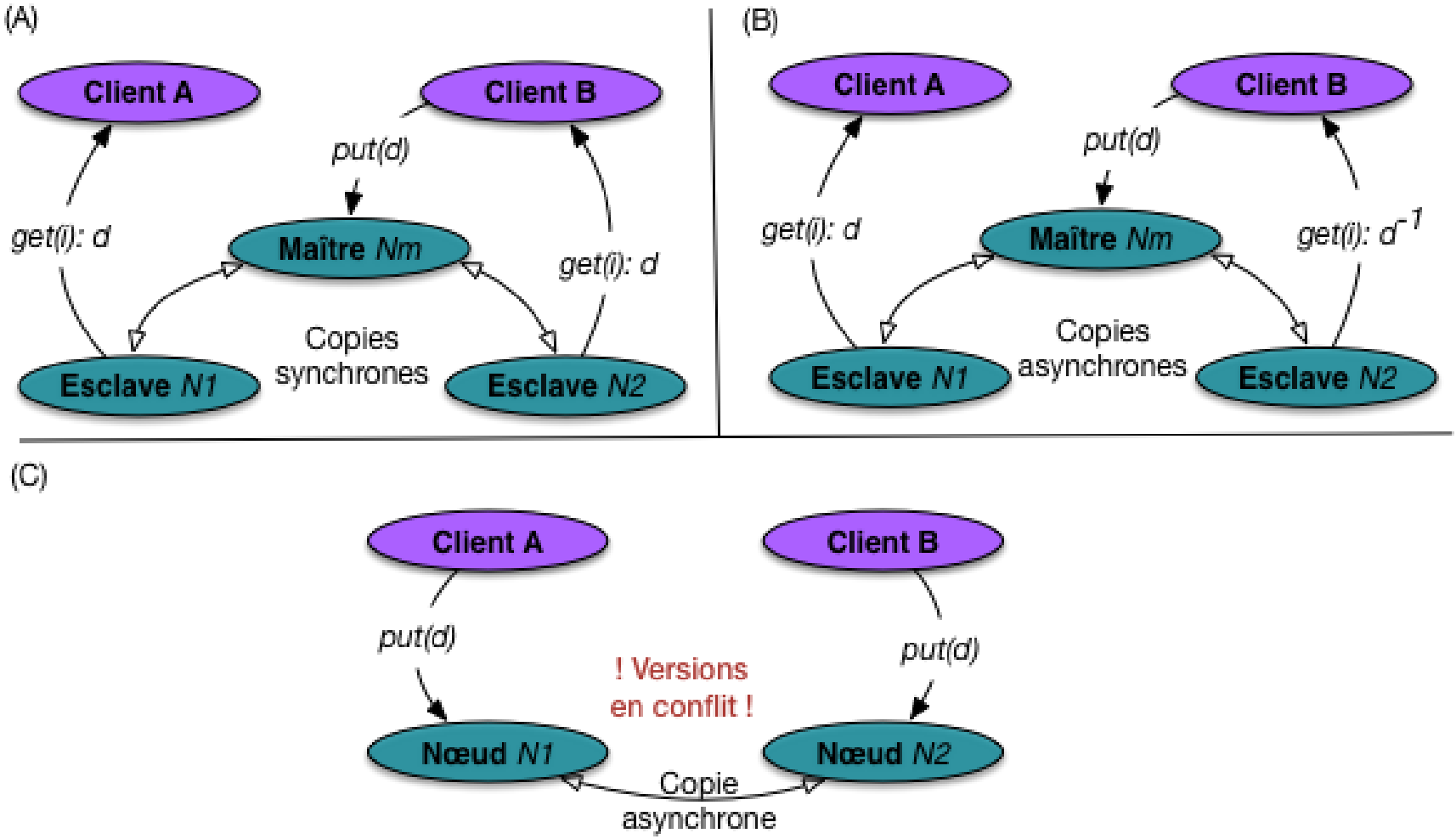
# Réplication synchrone



# Réplication asynchrone



# Réplication et cohérence

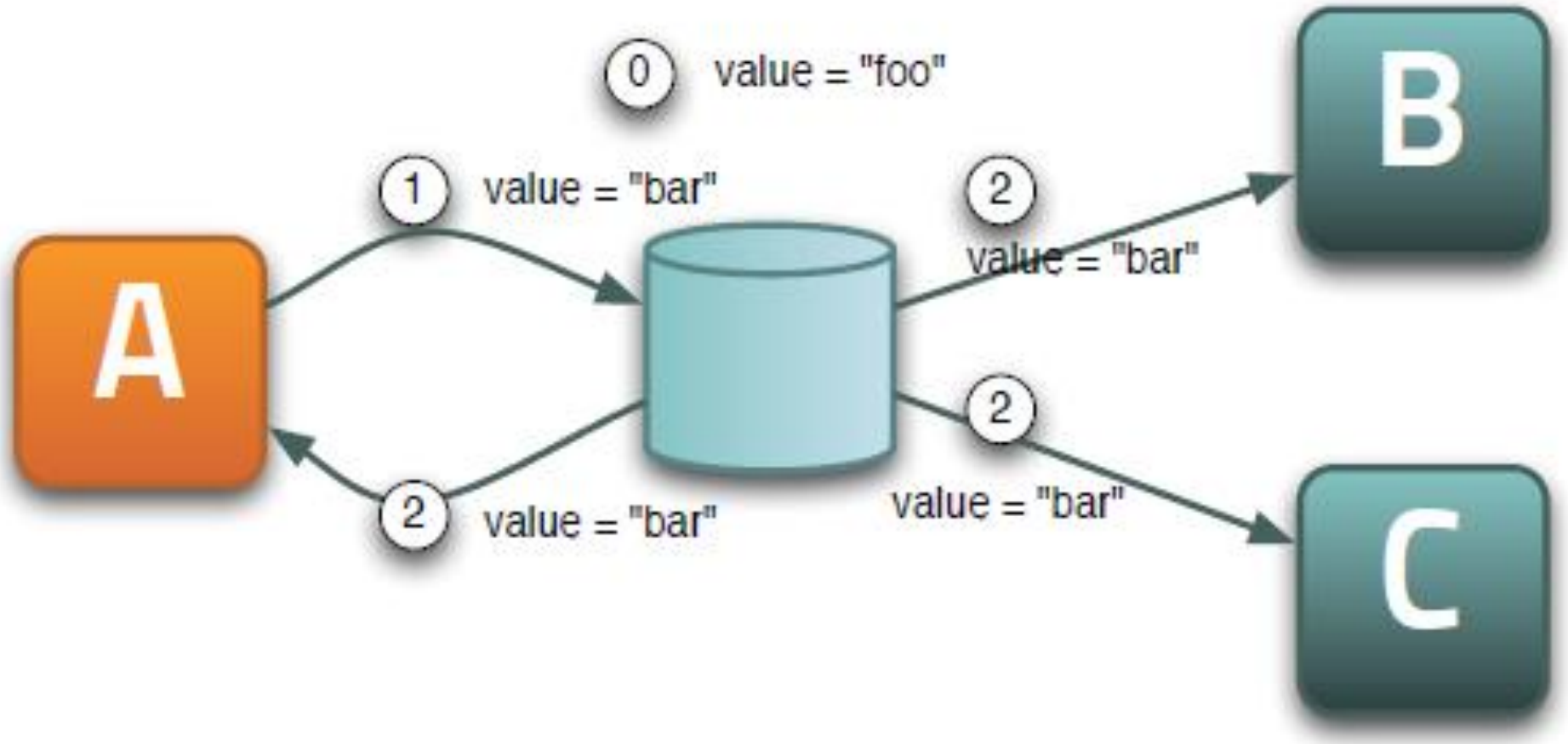


# Cohérence

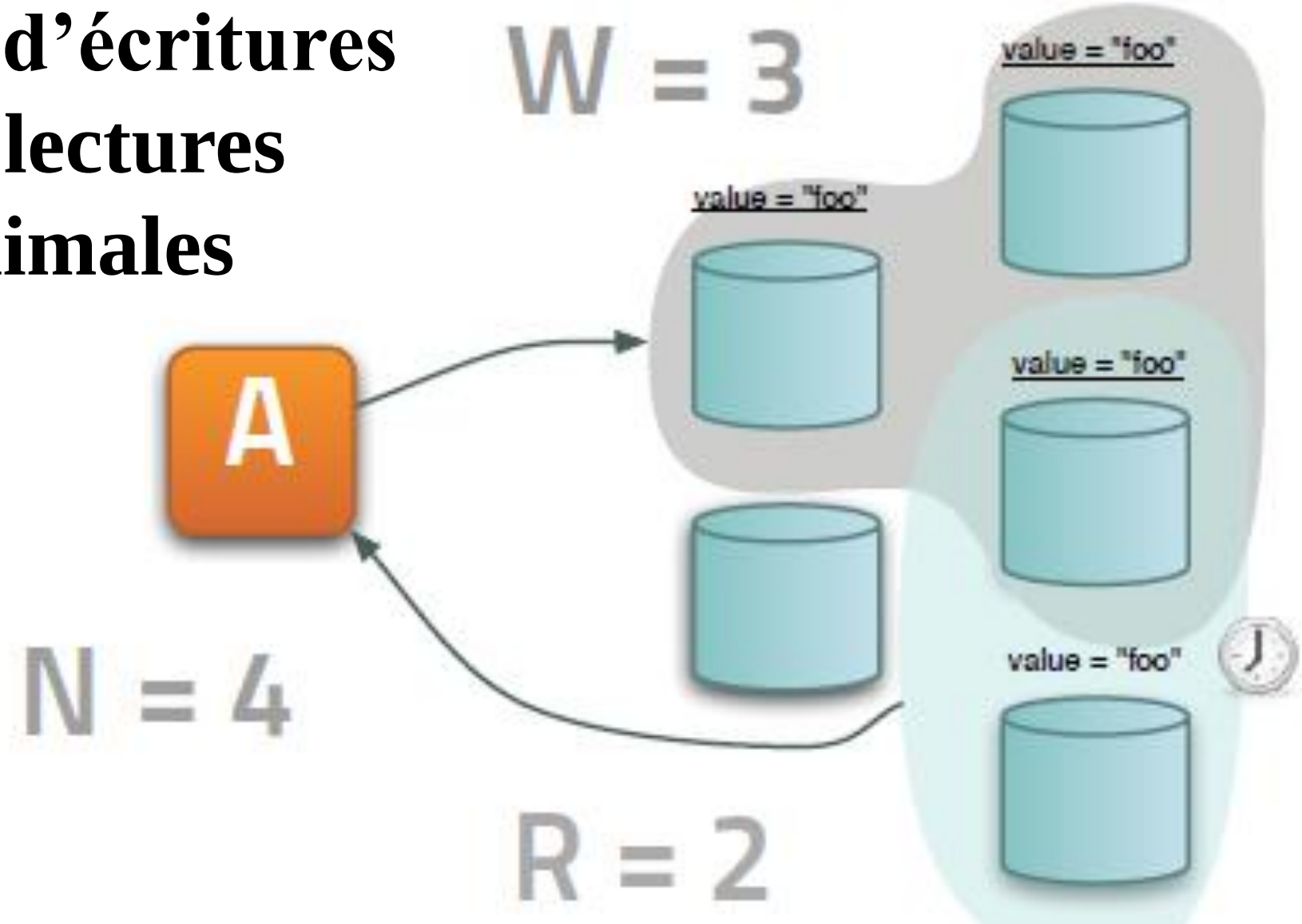
## 3 niveaux de cohérence :

- **Forte (propriétés ACID)** : toutes les copies sont toujours en phase, le prix à payer étant un délai pour chaque écriture;
- **Faible**: les copies ne sont pas forcément en phase, et rien ne garantit qu'elles le seront;
- **A terme (*Eventual consistency*) :** niveau de cohérence typique des systèmes NoSQL ; les copies ne sont pas immédiatement en phase, mais le système garantit qu'elles le seront « à terme »

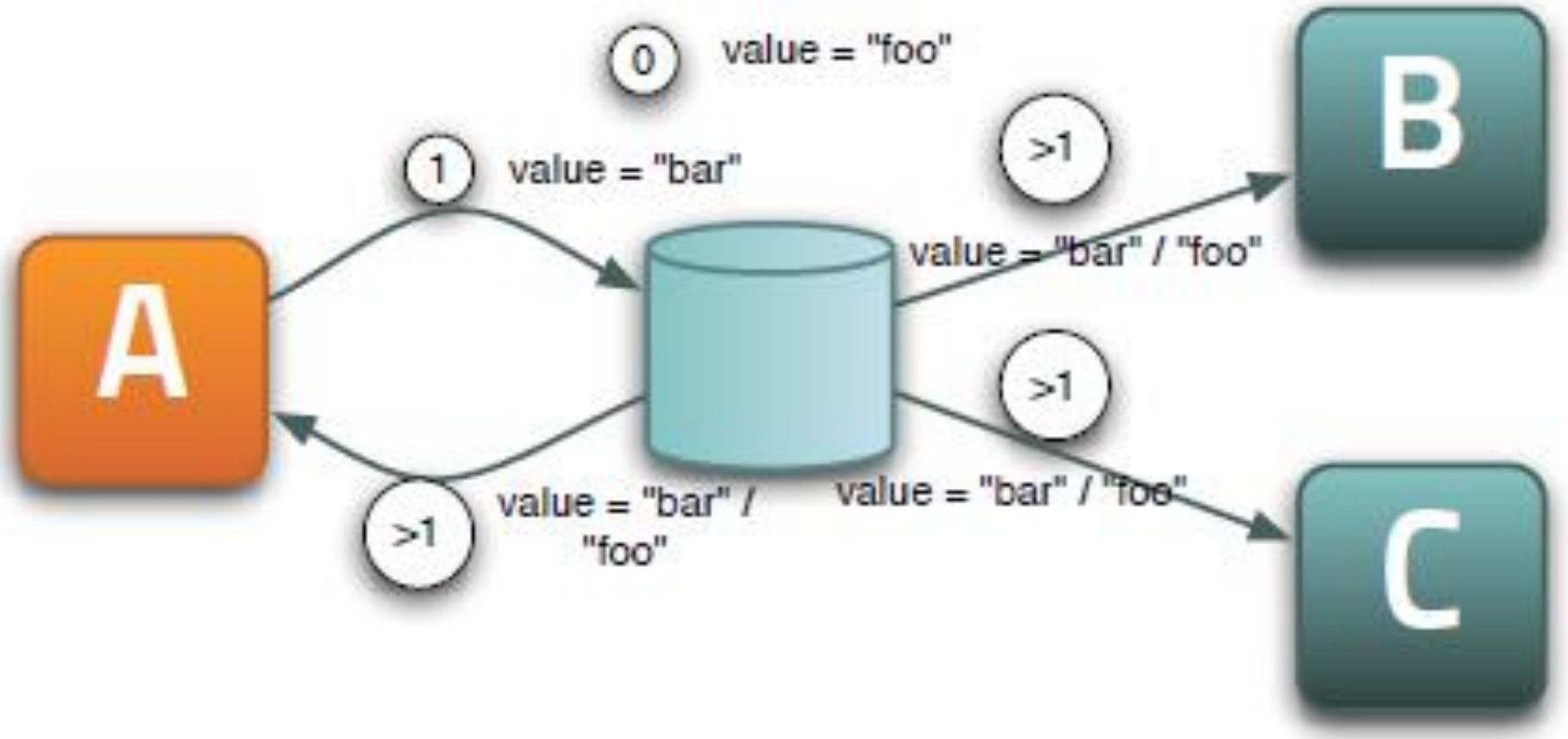
# Cohérence forte



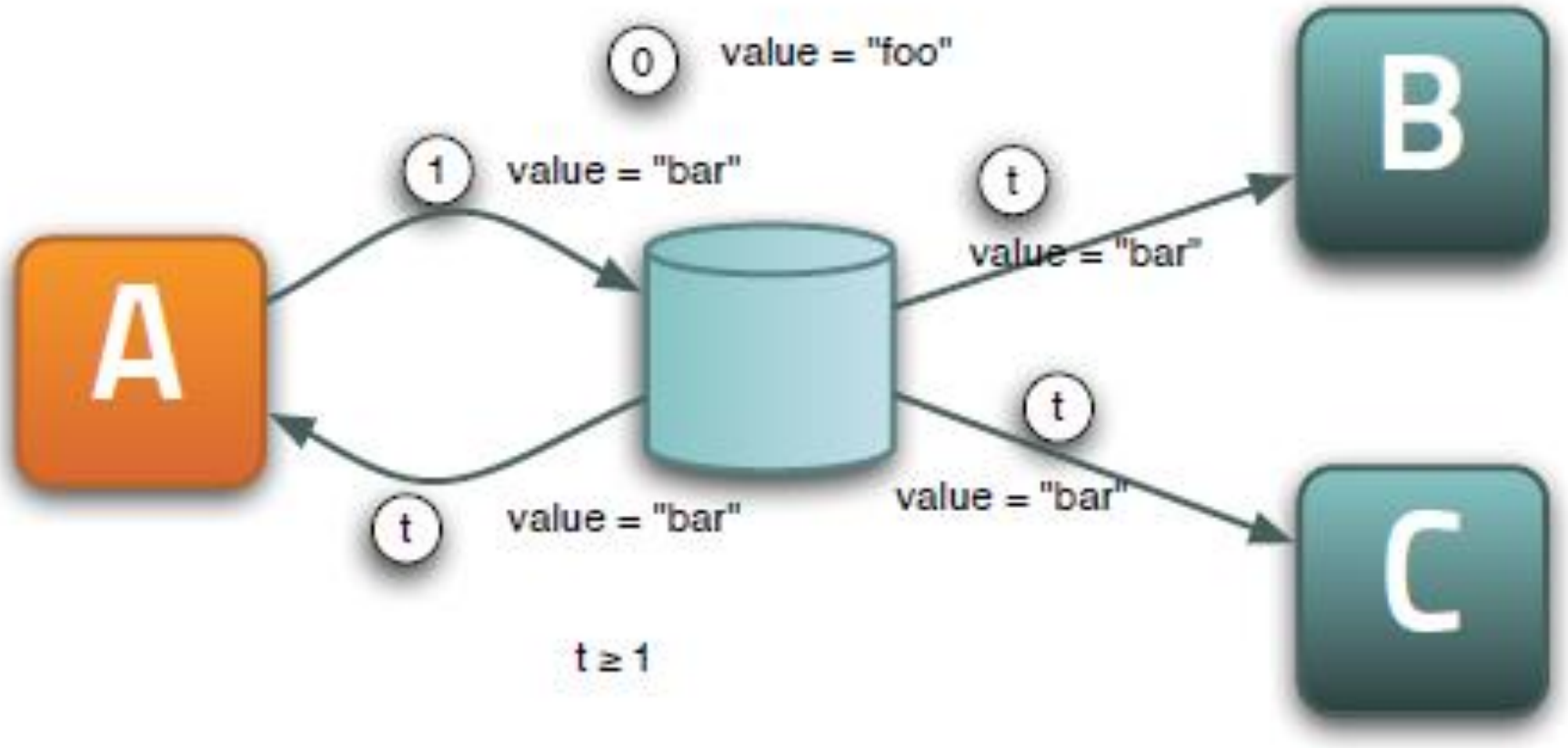
# Cohérence forte : nombre d'écritures et de lectures minimales



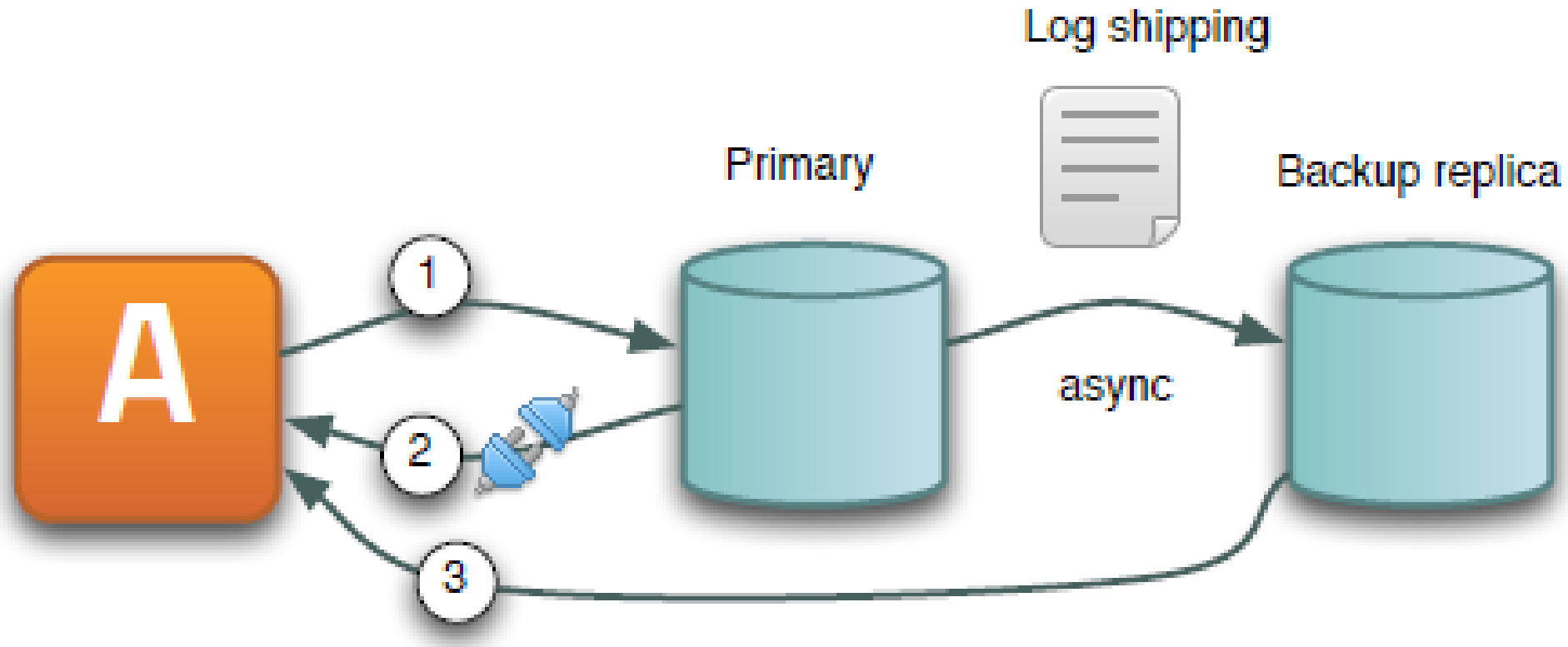
# Cohérence faible



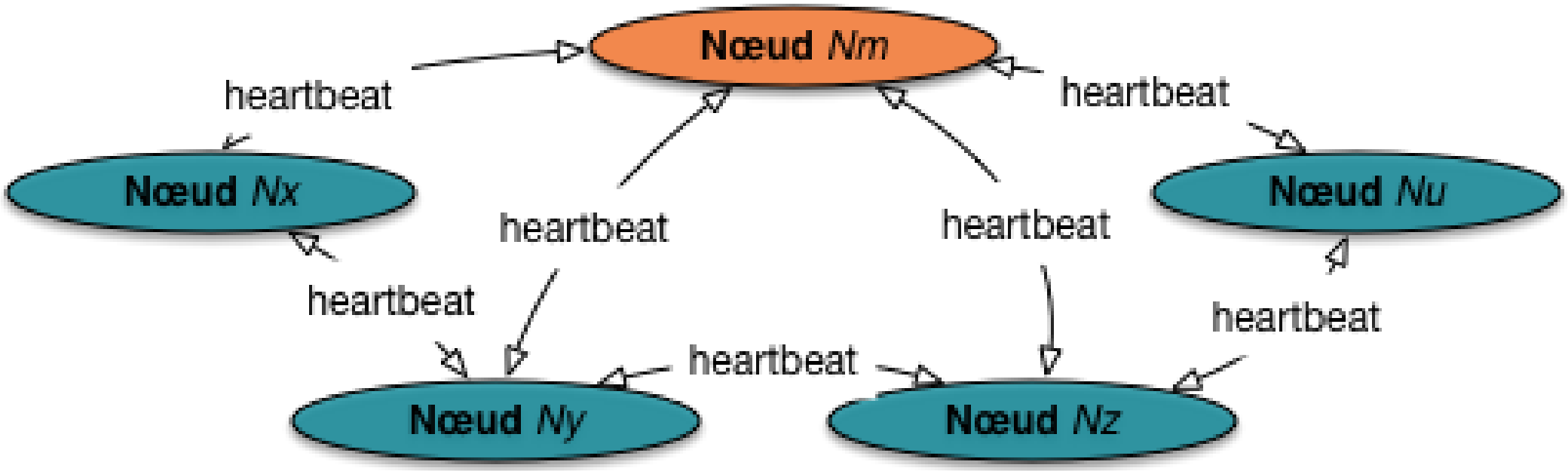
# Cohérence à termes (*Eventual Consistency*)



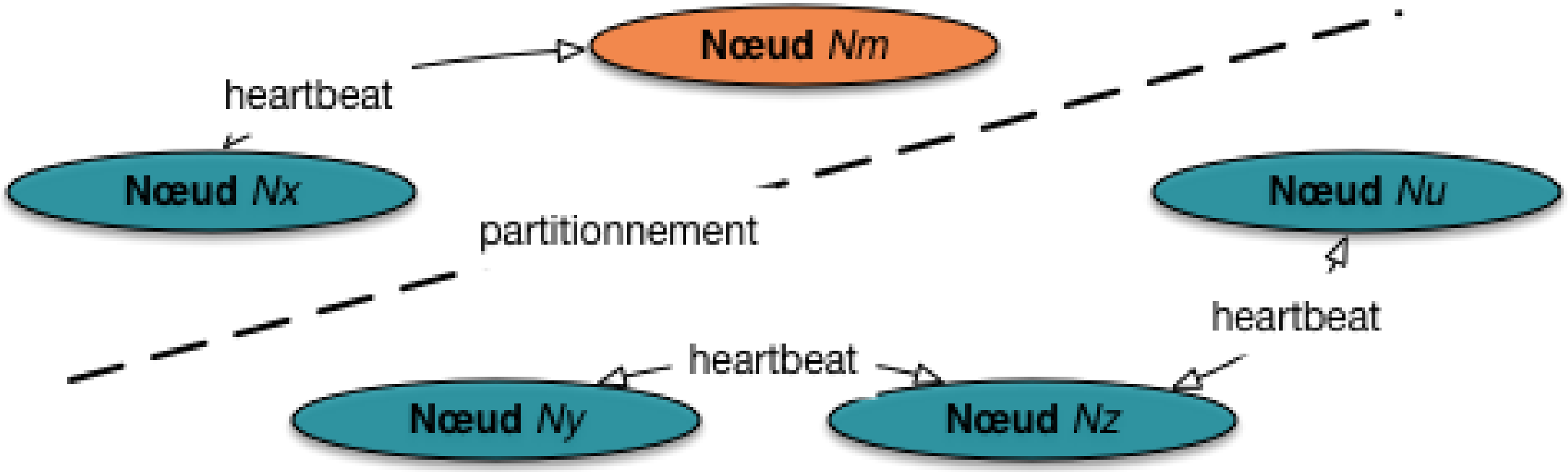
# Cohérence à termes (*Eventual Consistency*) et réplication asynchrone



# Réplication et reprise sur panne : communication entre le maître et ses esclaves

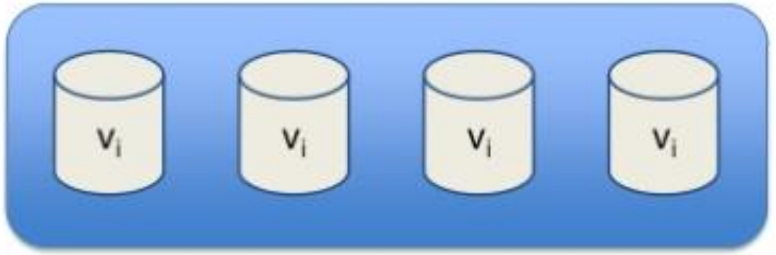


# Réplication et reprise sur panne : élection d'un nouveau maître



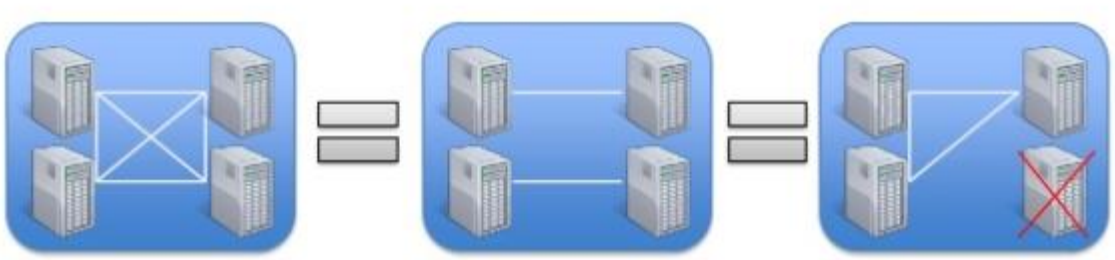
# Théorème de CAP : 3 proporiétés

- ***Consistency*** **(Cohérence)** :




- ***Availability*** **(Disponibilité)** :


- ***Partition Tolerance*** **(Distribution ou Tolérance au partitionnement)** :

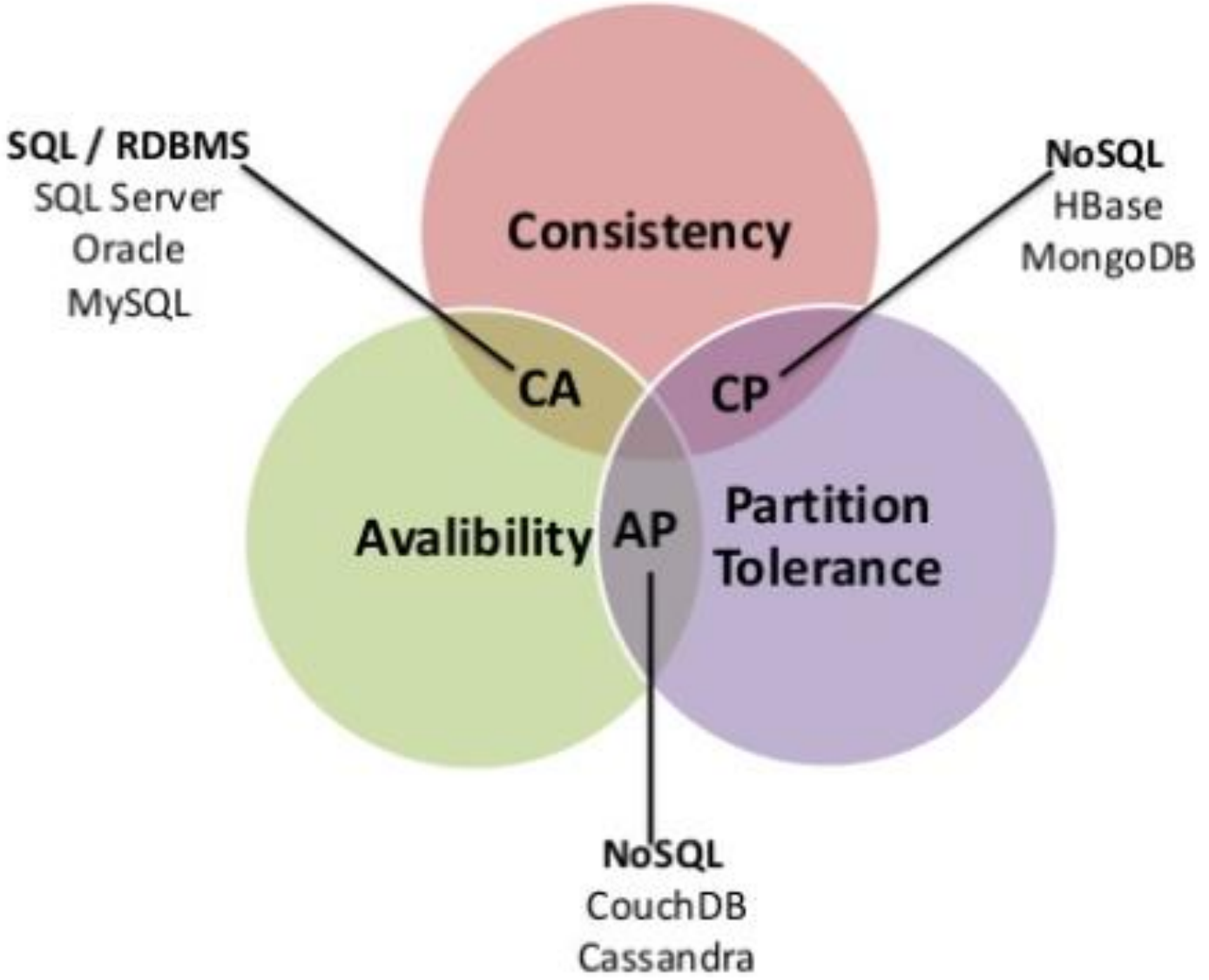


# Théorème de CAP : détails des propriétés

- Proposé par Eric A. Brewer en 2000
- Définition de **3 propriétés** :
  - ***Consistency*** **(Cohérence)** : Une donnée n'a qu'un seul état visible quel que soit le nombre de réplicas - tous les nœuds du système voient exactement les mêmes données au même moment
  - ***Availability*** **(Disponibilité)** : Tant que le système tourne (distribué ou non), la donnée doit être disponible - garantie que toutes les requêtes reçoivent une réponse
  - ***Partition Tolerance*** **(Distribution ou Tolérance au partitionnement)** : Quel que soit le nombre de serveurs, toute requête doit fournir un résultat correct - aucune panne moins importante qu'une coupure totale du réseau ne doit empêcher le système de répondre correctement (ou encore : en cas de morcellement en sous-réseaux, chacun doit pouvoir fonctionner de manière autonome)

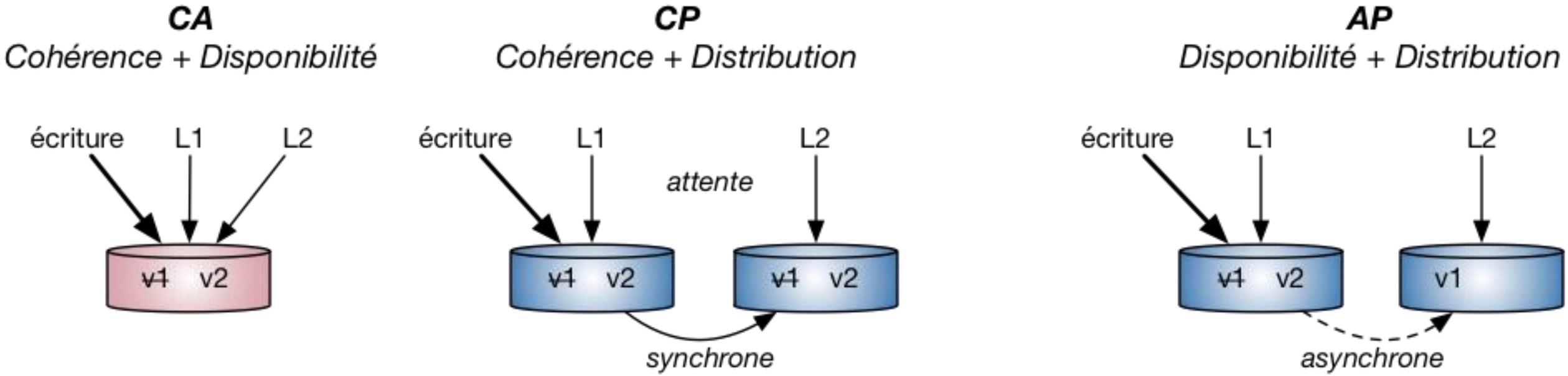
Repris er adapté de https://openclassrooms.com/en/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462471-maitrisez-le-theoreme-de-cap et wikipedia

# Positionnement SQL/NoSQL selon CAP



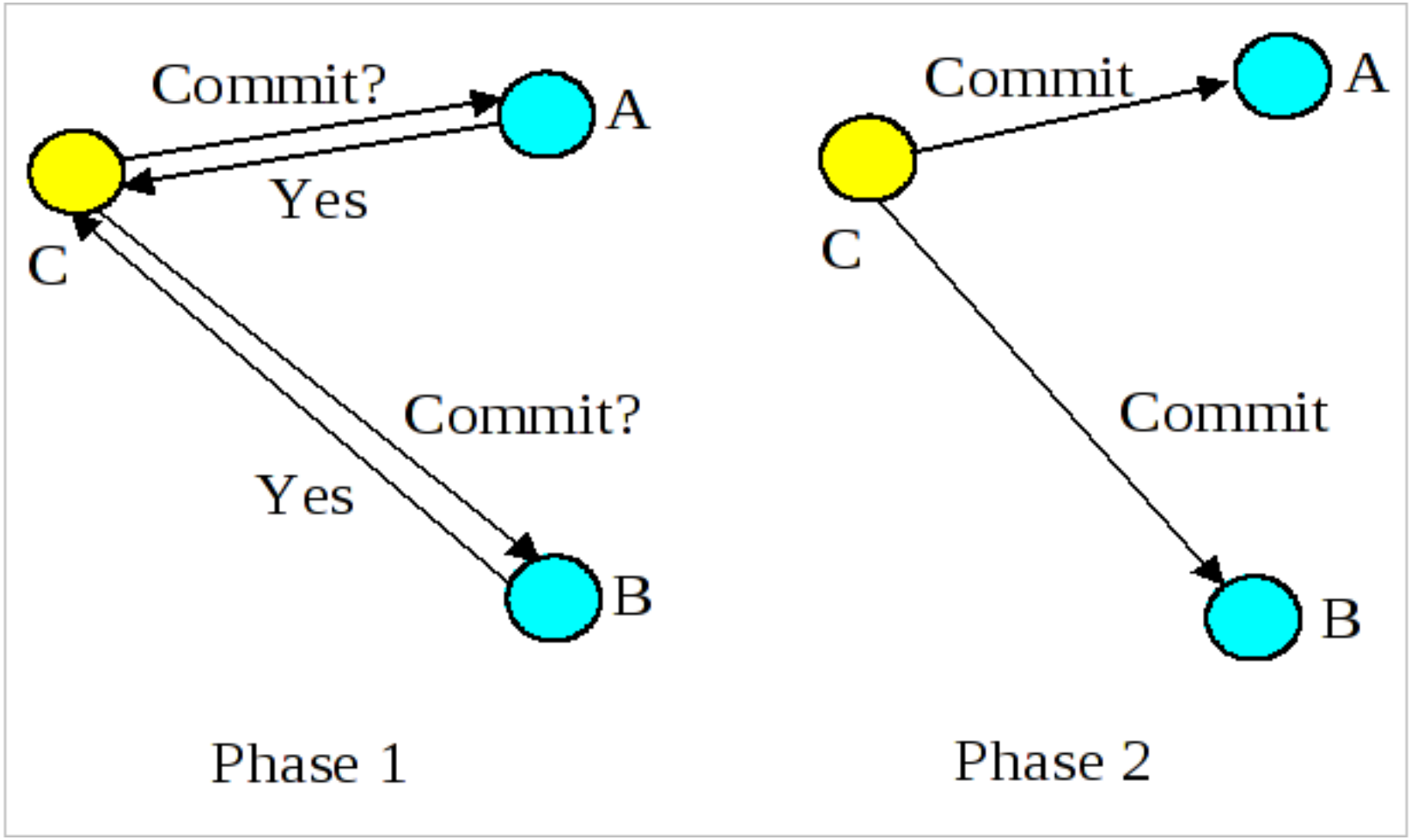
# Théorème de CAP : exemples

"Dans toute base de données, on ne peut respecter au plus que 2 propriétés parmi la cohérence, la disponibilité et la distribution "




Repris er adapté de https://openclassrooms.com/en/courses/4462426-maitrisez-les-bases-de-donnees-nosql/4462471-maitrisez-le-theoreme-de-cap

# C + A : Validation à 2 phases (PC)



# C + P : verrouillage pessimiste (ex. *Two-Phases Locking*)

- **Principe**
  - **Phase 1 : Verrouillage des items / Phase ascendante**
  - **Phase 2 : Déverrouillage des items / Phase descendante**
- **Théorème**

  **Un ordonnancement obtenu par le protocole V2P est sérialisable**
- **Inconvénients**

  **Risque d'annulation de transactions en cascade et d'inter-blocages**
- **Protocole V2P strict**

  **Les verrous sont conservés par une transaction jusqu'à la fin**

# Protocole de verrouillage en deux phases

**Soit l'ordonnancement : $W_1(x), R_2(y), R_1(y), R_2(x), C_1, C_2$**

**Graphe de précédence :**


Pas de cycle dans le graphe de précédence ⇒ ordonnancement sérialisable

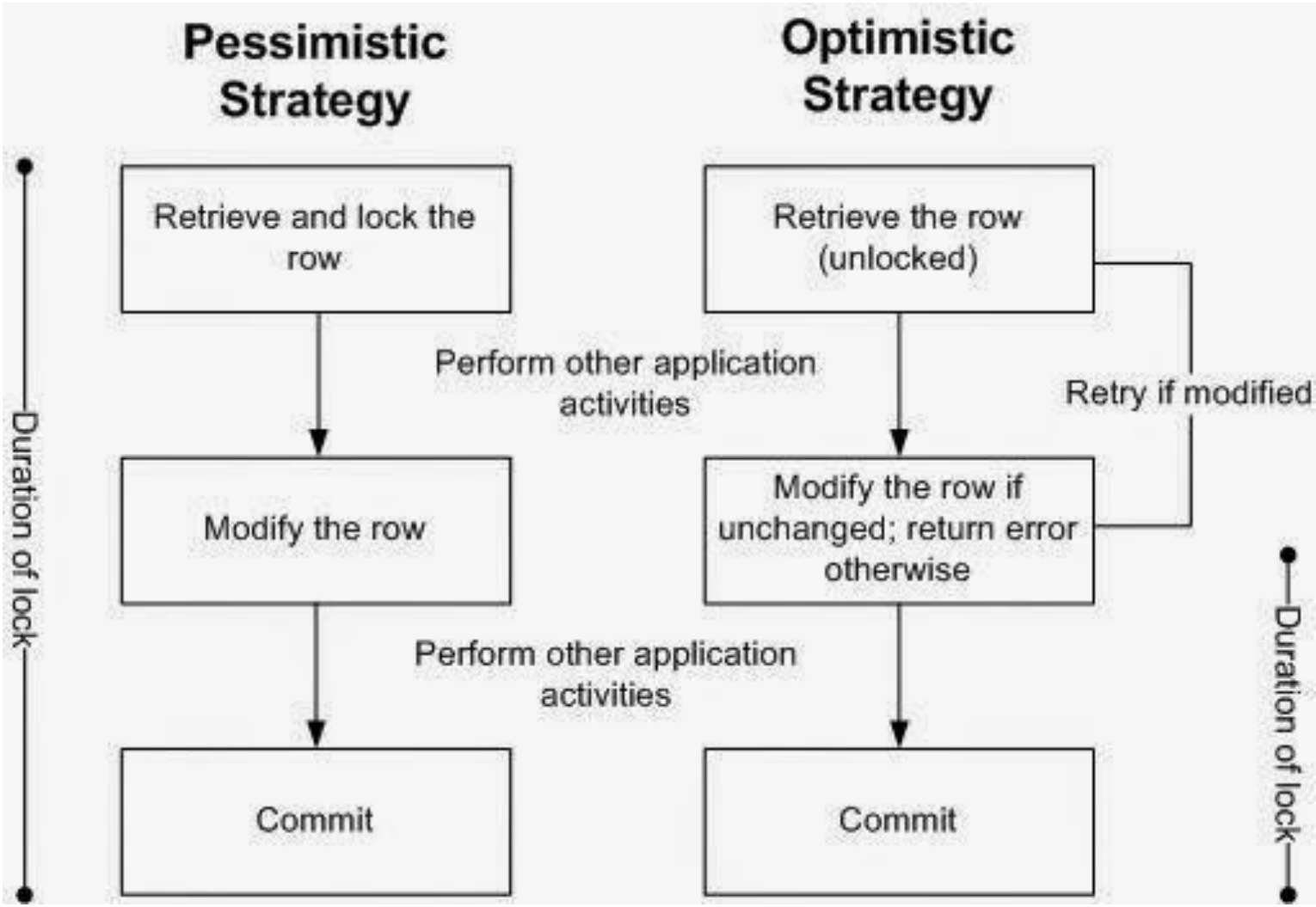
**Protocole V2P strict :**

| $T_1$ | $T_2$ |
|---|---|
| $VX_1(x)$ $W_1(x)$ | |
| | $VP_2(y)$ $R_2(y)$ |
| $VP_1(y)$ $R_1(y)$ | |
| | ~~$VP_2(x)$ $R_2(x)$~~ |

**Protocole V2P non strict :**

| $T_1$ | $T_2$ |
|---|---|
| $VX_1(x)$ $W_1(x)$ | |
| | $VP_2(y)$ $R_2(y)$ |
| $VP_1(y)$ $R_1(y)$ | |
| $D_1(x)$ | |
| | $VP_2(x)$ $R_2(x)$ |

# P + A : Stratégie optimiste (ex. MVCC)



# *Multi-Versioned Concurrency Control* (MVCC)