Base de Données Avancée : Concurrence

Thomas Gerald

March 17, 2026

Laboratoire Interdisciplinaire des Sciences du Numérique – LISN, CNRS
thomas.gerald@lisn.upsaclay.fr

[10pt,aspectratio=169]beamer

[]beamerthemelisn

appendixnumberbeamer hyperref soul booktabs [scale=2]ccicons amsfonts amssymb amsmath definition listings

multirow graphicx framed,lipsum sidecap algorithm2e xcolor listings xstring

inconsolata

forest

xspace

Base de Données Avancée : Concurence

Thomas Gerald

March 17, 2026

Laboratoire Interdisciplinaire des Sciences du Numérique – LISN, CNRS
thomas.gerald@lisn.upsaclay.fr

Objectifs

Une requête R, cas idéal

- R (ou un ensemble de requêtes) s'execute indépendamment de tout autre requête
- L'exécution de R se déroulera toujours intégralement

Mais...
Gestion de plusieurs utilisateurs

- Exécution correcte des applications concurrentes

Bonnes performances:

- **Paralléliser E/S et calcul CPU** : réduire les temps de non usage du disque ou du CPU
- **Eviter les blocages** : Ne pas bloquer les transactions courte par une transaction longue
  $\rightarrow$ contrôle du temps de réponse

Plusieurs problèmes

Problèmes

- Plusieurs utilisateurs accédant au mêmes données
- Une intéruption d'un utilisateur (déconnexion)
- Contraintes non respectés
- Pannes
- etc...

Comment faire ?

Controle de la concurence

- Reprise sur panne $\rightarrow$ revenir dans un états connu de la base de données
- Stocker les modifications non validées
- Vérrous sur les données $\rightarrow$ bloquer certaines ressources où des opérations “critiques” sont éffectuées
- Vérifier un bon ordonnancement des transactions (sinon revenir à un état connu)

Transaction
Une transaction T est une suite/séquence d'opérations de lecture, d'écriture ou de mise à jour sur une base de données. Elle se termine par l'une des instructions suivantes :

- **commit** : On valide les opérations de la transaction
- **rollback/abort** : On indique l'annulation de la transaction, on revient dans l'état post-transaction

En pratique
Une transaction est un programme (ou un sous-programme) Interrogeant la base de données.

Exemple : réservation d'un billet d'avion
Une transaction T est une suite/séquence d'opérations de lecture, d'écriture ou de mise à jour sur une base de données. Elle se termine par l'une des instructions suivantes :

- **commit** : On valide les opérations de la transaction
- **rollback/abort** : On indique l'annulation de la transaction, on revient dans l'état post-transaction

En pratique
Une transaction est un programme (ou un sous-programme) Interrogeant la base de données.

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Atomicité : Tout ou rien

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Atomicité : Tout ou rien

Découpage en atome/bloc d'exécution → transaction :

- Soit toute la transaction est exécutée (commit)
- Soit aucune action n'est exécutée (rollback)

→ Pas de gestion par l'utilisateur des intéruptions d'une transaction (erreur, panne, contraintes, ...)

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Cohérance: Preserver les contraintes d'intégrité

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Cohérance: Preserver les contraintes d'intégrité

Les transactions satisfont les contraintes, production d'une base de données cohérente

→ Dépend des programmes crées par les dévellopeurs (les triggers, procedure, etc..)

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Isolation : exécution comparable au mono-utilisateur

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Isolation : exécution comparable au mono-utilisateur

Le résultat d'une transaction doit être comparable ("équivalent") à celui produit dans un environnement mono-utilisateur.

$\rightarrow$ L'utilisateur ne doit pas avoir à prendre en compte l'interaction avec d'autres programmes pour écrire ses procédures de traitement

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Isolation : exécution comparable au mono-utilisateur

Le résultat d'une transaction doit être comparable ("équivalent") à celui produit dans un environnement mono-utilisateur.

→ L'utilisateur ne doit pas avoir à prendre en compte l'interaction avec d'autres programmes pour écrire ses procédures de traitement

→ Notion de Sérialisabilité

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Durabilité : données persistentes
Une fois terminée (après validation), les effets d'une transaction sur la base de données sont définitifs, doivent persister malgré pannes ou autres problèmes.

$\rightarrow$ L'utilisateur ne doit pas avoir à prendre gérer ce qui se passe après l'exécution d'une transaction

Garanties ACID

ACID
Atomicité, **C**ohérence, **I**solation, **D**urabilité

Durabilité : données persistentes
Une fois terminée (après validation), les effets d'une transaction sur la base de données sont définitifs, doivent persister malgré pannes ou autres problèmes.

→ L'utilisateur ne doit pas avoir à prendre gérer ce qui se passe après l'exécution d'une transaction

→ Reprise sur panne

Les transactions : entremêlement des actions

Transaction T_1	...	Transaction T_K	Transaction T_M
$A_1^1, A_1^2, \ldots A_1^n$	...	$A_K^1, A_K^2, \ldots A_K^k$	$A_M^1, A_M^2, \ldots A_M^m$

Pour éviter d'attendre la fin d'une transaction A avant de commencer le traitement d'une autre transaction, on execute les transactions de manière concurrente

$\rightarrow$ Suite d'actions des différentes transactions:

$A_j^1, A_1^1, A_N^1, A_j^2, \ldots A_1^n, A_j^7, A_2^9, \ldots$

- Chaque action d'une instruction apparait exactement une fois (A_i^j n'apparait qu'une seul fois dans l'exécution)
- L'ordre des instructions est préservé pour chaque transaction

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie

1. Insertion de la nouvelle note dans *rating*

INSERT INTO rating **VALUES** (5, 18.5, 'blabla')

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie

1. Insertion de la nouvelle note dans *rating*

   INSERT INTO rating **VALUES** (5, 18.5, 'blabla')

2. Retrouver le nombre de note et la moyenne du film dans *movie*

   SELECT nb_rate **FROM** movie **WHERE** mid=5 (il vaut 8)

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie

1. Insertion de la nouvelle note dans *rating*

   INSERT INTO rating **VALUES** (5, 18.5, 'blabla')

2. Retrouver le nombre de note et la moyenne du film dans *movie*

   SELECT nb_rate **FROM** movie **WHERE** mid=5 (il vaut 8)

3. Mettre à jour *movie*

   UPDATE movie **SET** nb_rate = 8+1 **WHERE** mid=5

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

1. Obtenir le *mid*

SELECT mid **FROM** rating **WHERE** rid=22 (il vaut 5)

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

1. Obtenir le *mid*

   SELECT mid **FROM** rating **WHERE** rid=22 (il vaut 5)

2. Suppression dans *rating*

   DELETE FROM rating **WHERE** rid=22

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

1. Obtenir le *mid*

   SELECT mid **FROM** rating **WHERE** rid=22 (il vaut 5)

2. Suppression dans *rating*

   DELETE FROM rating **WHERE** rid=22

3. Retrouver le nombre de note et la moyenne du film dans *movie*

   SELECT nb_rate **FROM** movie **WHERE** mid=5 (il vaut 8)

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

1. Obtenir le *mid*

   SELECT mid **FROM** rating **WHERE** rid=22 (il vaut 5)

2. Suppression dans *rating*

   DELETE FROM rating **WHERE** rid=22

3. Retrouver le nombre de note et la moyenne du film dans *movie*

   SELECT nb_rate **FROM** movie **WHERE** mid=5 (il vaut 8)

4. Mettre à jour *movie*

   UPDATE movie **SET** nb_rate = 8-1 **WHERE** mid=5

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

Executons la transaction A puis B :

(A,1) → (A,2) → (A, 3)→(B, 1)→(B, 2)→(B, 3)→(B, 4) :

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

Executons la transaction A puis B :

(A,1) → (A,2) → (A, 3)→(B, 1)→(B, 2)→(B, 3)→(B, 4) :

nb_rate vaut 8 pour $mid = 5$

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

Et si les transactions A et B sont effectués en parallèles (exécution concurrente) ?

Exécutons les deux transactions dans l'ordre suivant :

(A,1) → (A,2)→(B, 1)→(B, 2)→(B, 3)→(B, 4) → (A, 3)

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

Et si les transactions A et B sont effectués en parallèles (exécution concurrente) ?

Exécutons les deux transactions dans l'ordre suivant :

(A,1) → (A,2)→(B, 1)→(B, 2)→(B, 3)→(B, 4) → (A, 3)

nb_rate vaut 9 pour *mid* = 5

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

Et si les transactions A et B sont effectués en parallèles (exécution concurrente) ?

Exécutons les deux transactions dans l'ordre suivant :

(A,1) → (A,2)→(B, 1)→(B, 2)→(B, 3)→(B, 4) → (A, 3)

nb_rate vaut 9 pour $mid = 5$

Exécutons maintenant les deux transactions dans l'ordre suivant :

(A,1) → (A,2)→(B, 1)→(B, 2)→(B, 3) → (A, 3) →(B, 4)

Les transactions : exemple d'une transaction

- **movie** : (**mid**: number, **title** : text, **nb_rate** : number)
- **rating** : (**rid**,**#mid**: number, **score** : number, **comment** : text)

Transaction A : Ajout d'une note → mettre à jour le nombre de note dans movie
Transaction B : Suppression d'une note → mettre à jour le nombre de note dans movie

Et si les transactions A et B sont effectués en parallèles (exécution concurrente) ?

Exécutons les deux transactions dans l'ordre suivant :

(A,1) → (A,2)→(B, 1)→(B, 2)→(B, 3)→(B, 4) → (A, 3)

nb_rate vaut 9 pour $mid = 5$

Exécutons maintenant les deux transactions dans l'ordre suivant :

(A,1) → (A,2)→(B, 1)→(B, 2)→(B, 3) → (A, 3) →(B, 4)

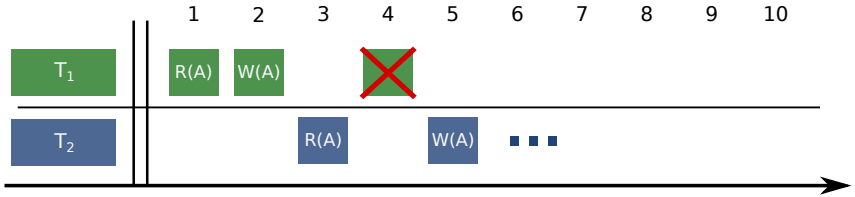
nb_rate vaut 7 pour $mid = 5$

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)

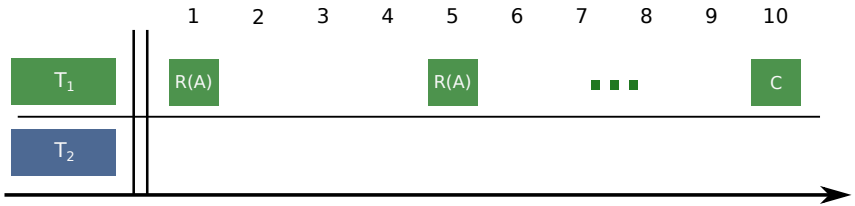
Par exemple suite à la suppression d'une valeur dans une transaction concurrente (ici le n-uplet représenté par A)



Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)

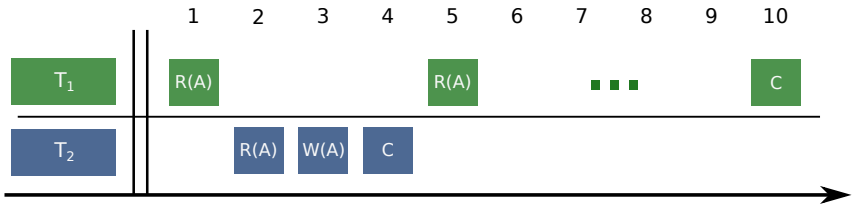


Changement de la valeur de A depuis la première lecture du n-uplet pour T_1, sans changements les valeurs sont differentes

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)

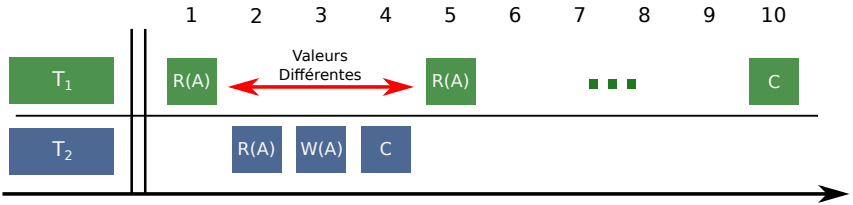


Changement de la valeur de A depuis la première lecture du n-uplet pour T_1, sans changements les valeurs sont differentes

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)



Changement de la valeur de A depuis la première lecture du n-uplet pour T_1, sans changements les valeurs sont differentes

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)
- Écriture incohérente

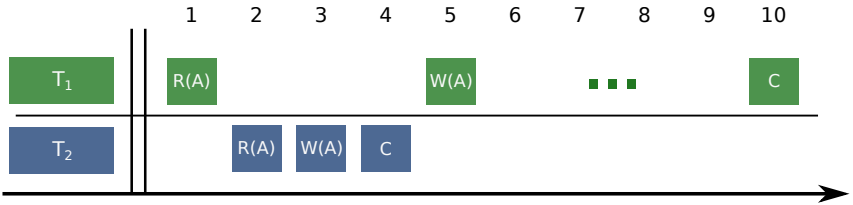
L'écriture dans T_1 de A dépend de la valeur lue de A précédente, celle-ci est entre temps modifiée par T_2

$\rightarrow$ C'est l'exemple précédent (incrementation)

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)
- Écriture incohérente



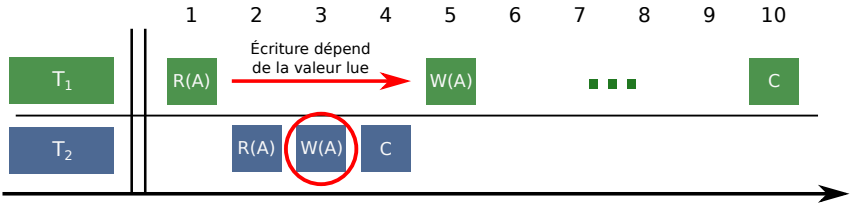
L'écriture dans T_1 de A dépend de la valeur lue de A précédente, celle-ci est entre temps modifiée par T_2

$\rightarrow$ C'est l'exemple précédent (incrementation)

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)
- Écriture incohérente



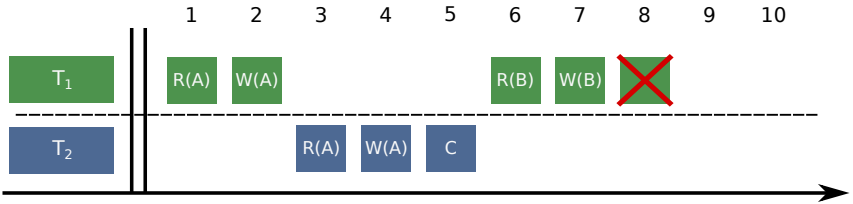
L'écriture dans T_1 de A dépend de la valeur lue de A précédente, celle-ci est entre temps modifiée par T_2

$\rightarrow$ C'est l'exemple précédent (incrementation)

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)
- Écriture incohérente
- Annulation impossible (garantie du commit)

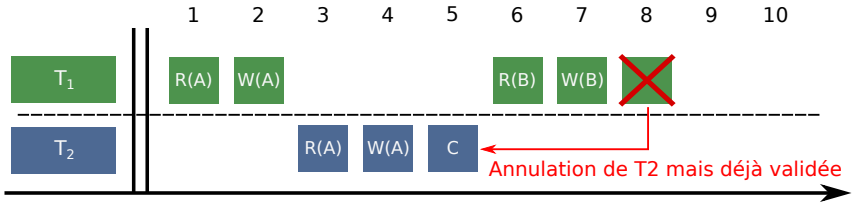


La transaction T_2 dépend de ce qui a été écrit dans T_1, T_1 est annulé par la suite
$\rightarrow$ La garantie de la validation de T_2 empêche de revenir en arrière

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)
- Écriture incohérente
- Annulation impossible (garantie du commit)



La transaction T_2 dépend de ce qui a été écrit dans T_1, T_1 est annulé par la suite
$\rightarrow$ La garantie de la validation de T_2 empêche de revenir en arrière

Transaction : Détection et correction d'anomalie

Quels types d'anomalies ?

- Lecture non valide (dirty read)
- Lecture non répétable (unrepeatable read)
- Écriture incohérente
- Annulation impossible (garantie du commit)

Anomalies
Un anomalie peut apparaitre lorsque (sur deux tranasactions concurrentes) :

- Une lecture puis une écriture sont éffectué sur le même objet (Read-Write)
- Une écriture puis une lecture est éffectué sur le même objet (Write-Read)
- Deux écritures sont faites sur le même objet (Write-Write)

Des transactions en serie
On considère les exécutions des transactions en série comme correcte. Un exécution en série correspond à l'exécution successive des transaction

Des transactions en serie
On considère les exécutions des transactions en série comme correcte. Un exécution en série correspond à l'exécution successive des transaction

Si on considère les transactions :

Transaction T_1	...	Transaction T_K	Transaction T_M
$A_1^1, A_1^2, \ldots A_1^n$	...	$A_K^1, A_K^2, \ldots A_K^k$	$A_M^1, A_M^2, \ldots A_M^m$

Un exécution en série correpond à l'exécution pour une transaction de toutes ses actions:

Des transactions en serie
On considère les exécutions des transactions en série comme correcte. Un exécution en série correspond à l'exécution successive des transaction

Si on considère les transactions :

Transaction T_1	...	Transaction T_K	Transaction T_M	Un exécution en série
$A_1^1, A_1^2, \dots A_1^n$	...	$A_K^1, A_K^2, \dots A_K^k$	$A_M^1, A_M^2, \dots A_M^m$	

correpond à l'exécution pour une transaction de toutes ses actions: Par exemple :
$\rightarrow A_1^1, A_1^2, \dots A_1^n, A_2^1, A_2^2, \dots, A_M^1, A_M^2, \dots A_M^m$

Des transactions en serie
On considère les exécutions des transactions en série comme correcte. Un exécution en série correspond à l'exécution successive des transaction

Si on considère les transactions :

Transaction T_1	...	Transaction T_K	Transaction T_M
$A_1^1, A_1^2, \dots A_1^n$	...	$A_K^1, A_K^2, \dots A_K^k$	$A_M^1, A_M^2, \dots A_M^m$

Un exécution en série correpond à l'exécution pour une transaction de toutes ses actions: Par exemple :
$\rightarrow A_1^1, A_1^2, \dots A_1^n, A_2^1, A_2^2, \dots, A_M^1, A_M^2, \dots A_M^m$
$\rightarrow A_K^1, A_K^2, \dots A_M^1, A_M^2, \dots A_M^m, \dots A_1^1, A_1^2, \dots A_1^n$

Ces deux exécution sont en serie

Des transactions en serie
On considère les exécutions des transactions en série comme correcte. Un exécution en série correspond à l'exécution successive des transaction

Exécution en série
Transactions exécutés en séries $\rightarrow$ Même état de la base de données ?

Des transactions en serie
On considère les exécutions des transactions en série comme correcte. Un exécution en série correspond à l'exécution successive des transaction

Exécution en série
Transactions exécutés en séries → Même état de la base de données ? **Non**

Tranasaction A

1. Lire l'objet

Des transactions en serie
On considère les exécution des transaction en série comme correcte. Un exécution en série correspond à l'exécution successive des transaction

Sérialisabilité
Soit H une exécution concurrente de n transactions $T_1 \dots T_n$. Cette exécution est sérialisable si et seulement si, quel que soit l'état initial de la base, il existe un ordonnancement H" de $T_1 \dots T_n$ tel que le résultat de l'exécution de H est équivalent à celui de l'exécution en série des transactions de H".

Conflit entre opérations/actions d'une exécution concurrente
Deux actions/opérations $A_i(x)$ et $A_j(x)$ (x étant le n-uplet cible), provenant de deux transactions T_i et T_j ($i \neq j$), sont en conflit si A_i ou/et A_j est une écriture.

Transaction : Sérialisabilité et conflit

Conflit entre opérations/actions d'une exécution concurrente
Deux actions/opérations $A_i(x)$ et $A_j(x)$ (x étant le n-uplet cible), provenant de deux transactions T_i et T_j ($i \neq j$), sont en conflit si A_i ou/et A_j est une écriture.

Exemple
Soit $R_i(A)$ définissant une action de lecture sur le n-uplet A pour la transaction i
Soit $W_i(A)$ définissant une action d'écriture sur le n-uplet A pour la transaction i
Quelles sont les conflits pour l'exécution concurrente suivante :

$$R_1(A), W_1(A), W_2(B), R_2(A), R_2(B), W_2(C)$$

Conflit entre opérations/actions d'une exécution concurrente
Deux actions/opérations $A_i(x)$ et $A_j(x)$ (x étant le n-uplet cible), provenant de deux transactions T_i et T_j ($i \neq j$), sont en conflit si A_i ou/et A_j est une écriture.

Exemple
Soit $R_i(A)$ définissant une action de lecture sur le n-uplet A pour la transaction i
Soit $W_i(A)$ définissant une action d'écriture sur le n-uplet A pour la transaction i
Quelles sont les conflits pour l'exécution concurrente suivante :

$$R_1(A), W_1(A), W_2(B), R_2(A), R_2(B), W_2(C)$$

Conflits

- $(W_1(A),\ R_2(A))$

Conflit entre opérations/actions d'une exécution concurrente
Deux actions/opérations $A_i(x)$ et $A_j(x)$ (x étant le n-uplet cible), provenant de deux transactions T_i et T_j ($i \neq j$), sont en conflit si A_i ou/et A_j est une écriture.

Relation $T_i \lhd T_j$:

On définit $T_i \lhd T_j$ pour deux transactions T_i et T_j si il existe un conflit entre deux actions $A_i(x)$ et $A_j(x)$ et que A_i est exécuté avant A_j :

$$T_i \lhd T_j \Leftrightarrow \exists A_i \in T_i, A_j \in T_j \text{ où } A_i \text{ est en conflit avec } A_j \text{ et } A_i <_H A_j$$

Conflit entre opérations/actions d'une exécution concurrente
Deux actions/opérations $A_i(x)$ et $A_j(x)$ (x étant le n-uplet cible), provenant de deux transactions T_i et T_j ($i \neq j$), sont en conflit si A_i ou/et A_j est une écriture.

Relation $T_i \lhd T_j$:

On définit $T_i \lhd T_j$ pour deux transactions T_i et T_j si il existe un conflit entre deux actions $A_i(x)$ et $A_j(x)$ et que A_i est exécuté avant A_j :

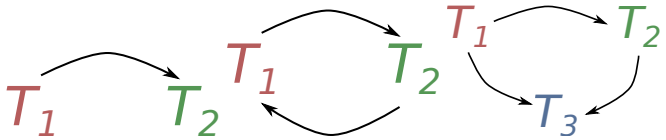
$$T_i \lhd T_j \Leftrightarrow \exists A_i \in T_i, A_j \in T_j \text{ où } A_i \text{ est en conflit avec } A_j \text{ et } A_i <_H A_j$$

Proposition sur la sérialisabilité (Sérialisablité conflictuelle) :

Soit H une exécution concurrente d'un ensemble de transactions $\mathcal{T} = T_1, T_2, \ldots, T_n$. Si le graphe engendré par les relations $\lhd$ $((\mathcal{T}, \lhd))$ est acyclique, alors H est sérialisable (et conflictuellement sérialisable). Sinon H est non-conflictuellement sérialisable.

Proposition sur la sérialisabilité (Sérialisablité conflictuelle) :

Soit H une exécution concurrente d'un ensemble de transactions $\mathcal{T} = T_1, T_2, \ldots, T_n$. Si le graphe engendré par les relations $\lhd$ $((\mathcal{T}, \lhd))$ est acyclique, alors H est sérialisable (et conflictuellement sérialisable). Sinon H est non-conflictuellement sérialisable.

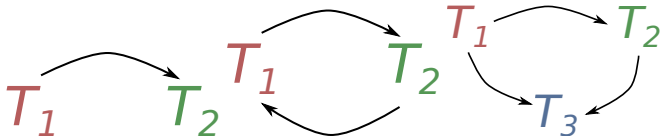


Quelles sont les exécutions sérialisables ?

1. Acyclique $\rightarrow$ Sérialisable
2. Cyclique $\rightarrow$ **On ne sait pas si l'exécution est sérialisable**
3. Acyclique $\rightarrow$ Sérialisable

Proposition sur la sérialisabilité (Sérialisablité conflictuelle) :

Soit H une exécution concurrente d'un ensemble de transactions $\mathcal{T} = T_1, T_2, \ldots, T_n$. Si le graphe engendré par les relations $\lhd$ $((\mathcal{T}, \lhd))$ est acyclique, alors H est sérialisable (et conflictuellement sérialisable). Sinon H est non-conflictuellement sérialisable.



Quelles sont les exécutions sérialisables ?

1. Acyclique $\rightarrow$ Sérialisable
2. Cyclique $\rightarrow$ **On ne sait pas si l'exécution est sérialisable**
3. Acyclique $\rightarrow$ Sérialisable

Sérialisabilité conflictuelle : Premier exemple

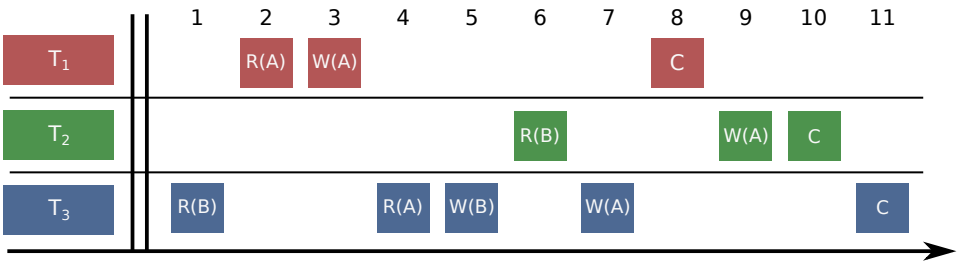
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?



Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

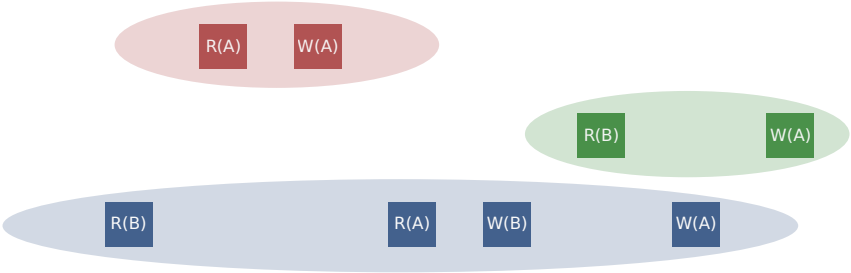
Sérialisable ?

Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

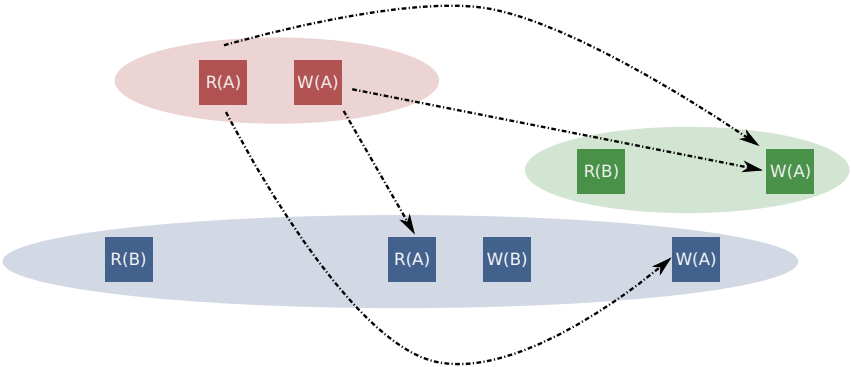
Sérialisable ?



Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ? Trouver $T_1 \lhd T_i$

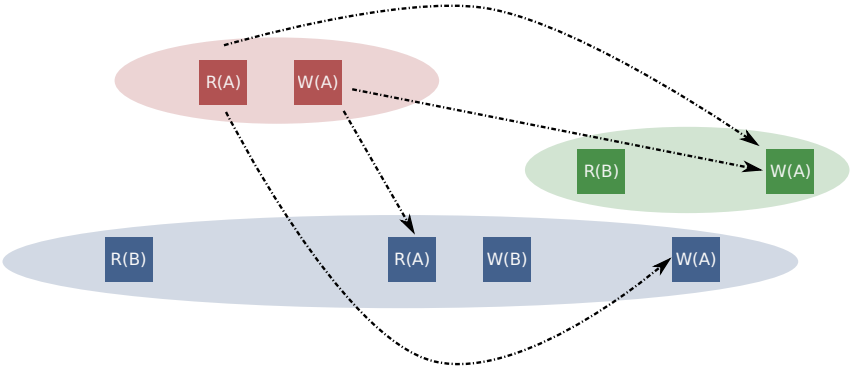


Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

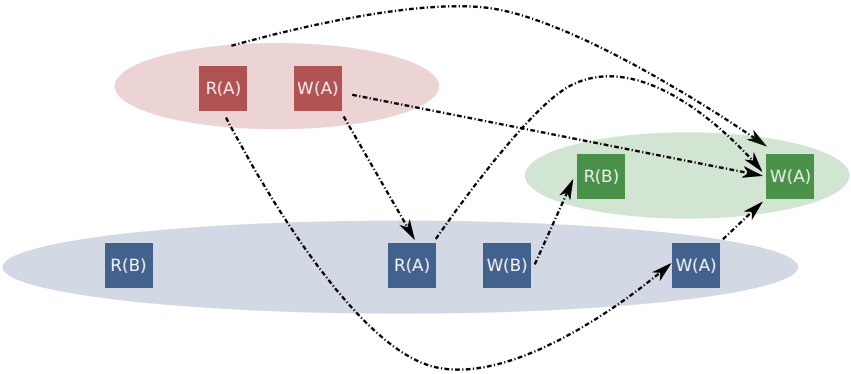
Sérialisable ? Trouver $T_2 \lhd T_i$



Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ? Trouver $T_3 \lhd T_i$

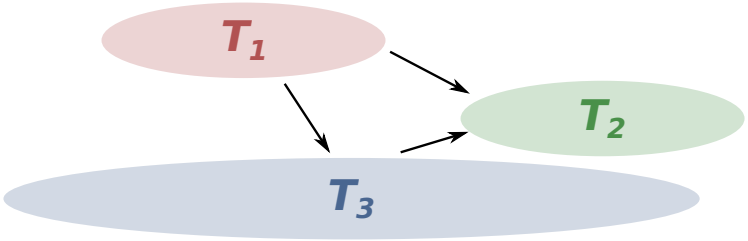


Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B)	R(A)	W(A)	R(A)	W(B)	R(B)	W(A)	C	W(A)	C	C

Sérialisable ?

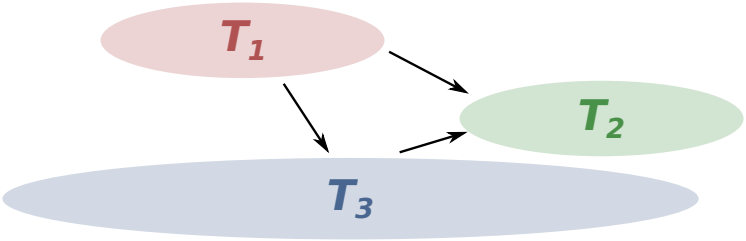


Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B)	R(A)	W(A)	R(A)	W(B)	R(B)	W(A)	C	W(A)	C	C

Sérialisable ?



Oui l'exécution est sérialisable !!!!

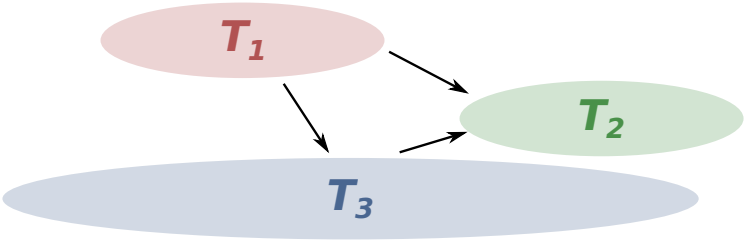
À quelle exécution en série ?

Sérialisabilité conflictuelle : Premier exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B)	R(A)	W(A)	R(A)	W(B)	R(B)	W(A)	C	W(A)	C	C

Sérialisable ?



Oui l'exécution est sérialisable !!!!

À quelle exécution en série ? $T_1 \rightarrow T_3 \rightarrow T_2$

Sérialisabilité conflictuelle : Second exemple

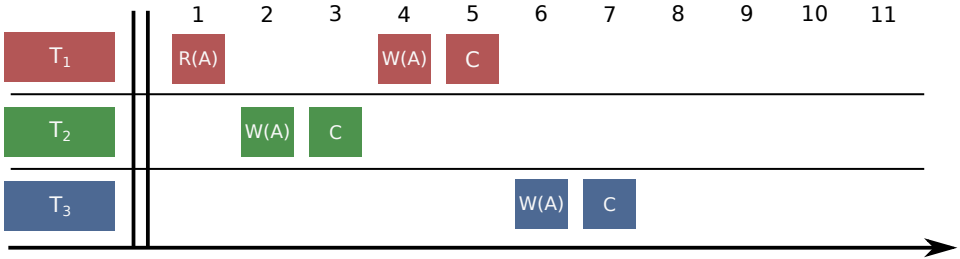
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

Sérialisabilité conflictuelle : Second exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?



Sérialisabilité conflictuelle : Second exemple

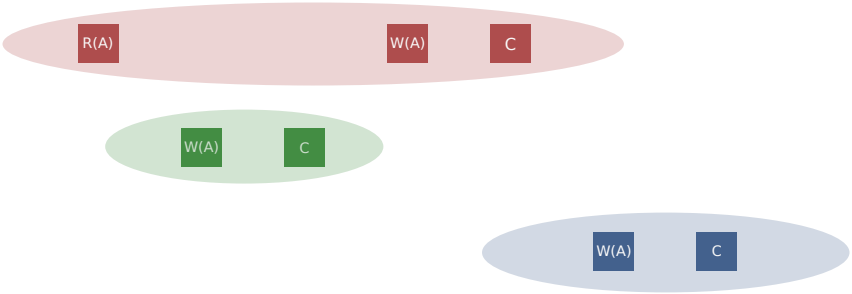
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

Sérialisabilité conflictuelle : Second exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?



Sérialisabilité conflictuelle : Second exemple

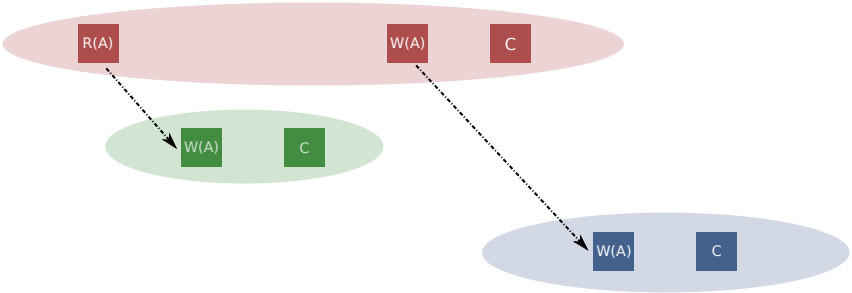
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ? Trouver $T_1 \lhd T_i$

Sérialisabilité conflictuelle : Second exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ? Trouver $T_1 \lhd T_i$



Sérialisabilité conflictuelle : Second exemple

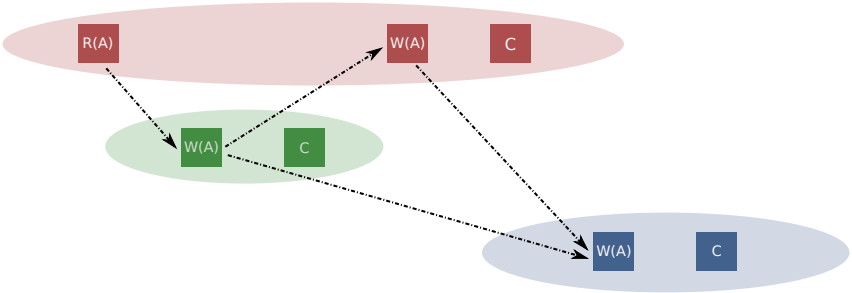
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ? Trouver $T_2 \lhd T_i$

Sérialisabilité conflictuelle : Second exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ? Trouver $T_2 \lhd T_i$



Sérialisabilité conflictuelle : Second exemple

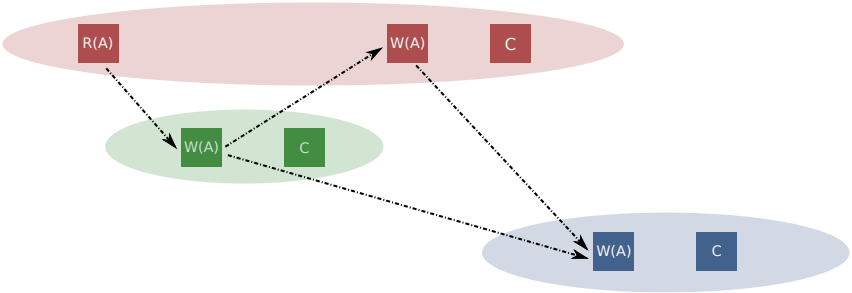
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ? Trouver $T_3 \lhd T_i$

Sérialisabilité conflictuelle : Second exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

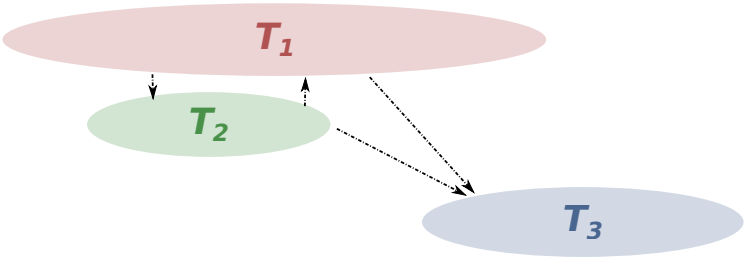
Sérialisable ? Trouver $T_3 \lhd T_i$



Sérialisabilité conflictuelle : Second exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

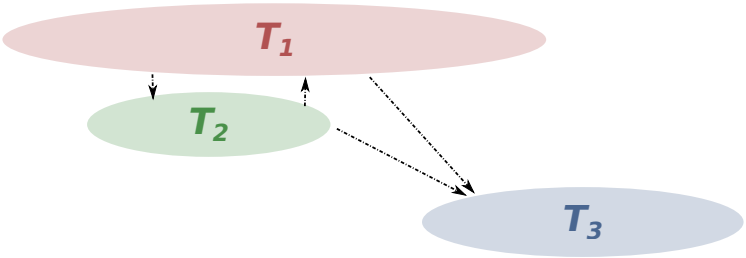
Sérialisable ?



Sérialisabilité conflictuelle : Second exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?



Pas conflictuellement sérialisable !!!!

Sérialisabilité conflictuelle : Second exemple

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

- Dans tous les cas, A aura le résultat de donnée par T_3 !!!!

Donc l'exécution est équivalente à :

$$T_1, T_2, T_3 \text{ et } T_2, T_1, T_3$$

Donc l'exécution est sérialisable !!!

Sérialisabilité : Conflictuellement, vue et globalement sérialisable

Une exécution concurrente S est XXX-sérialisable
S'il existe une exécution en série de $T_1, T_2, \ldots, T_n$ XXX-équivalente à S.

$\rightarrow$ Le nombre d'exécution en série de $T_1, T_2, \ldots, T_n$ est le nombre de permutation de $T_1, T_2, \ldots, T_n$

$\rightarrow n!$ (compléxité du test)

Les équivalences

- **Conflits** : La méthode précédente, mais toutes exécution non conflictuellement sérialisable ne sont pas selon la définition de sérialisabilité
- **Vue** : Observation du flux de données d'entrées/sorties et des lectures
- **Globale** : Observation du flux d'entrée et sortie

Sérialisabilité : Explication au tableau

Vue équivalence
Deux éxécution S_1 et S_2 sont dites vue équivalentes ssi :

1. Si T_i lit la valeur initiale de A dans S_1 alors T_i doit lire la valeur initiale de A dans S_2
2. Si T_i lit la valeur de A écrite par T_j dans S_1 alors T_i doit lire la valeur de A écrite par T_j dans S_2
3. Si T_i écrit la valeur finale A dans S_1 alors T_i doit écrire la valeur finale dans S_2

- Toute exécution conflictuellement sérialisable est aussi vue-sérialisable
- Une exécution vue-sérialisable n'est pas forcémment conflictuellement sérialisable

Modélisation du flux d'entrée sortie par un graphe

1. Construction du Graphe $G(S)$ pour l'exécution concurrente S : les noeuds sont des actions
   - Ajouter les arcs intra-transaction : $A_i^k \rightarrow A_i^p$, tel que les deux actions A_i^k est une lecture et A_i^p est une écriture et que A_i^k précéde A_i^p

Modélisation du flux d'entrée sortie par un graphe

1. Construction du Graphe $G(S)$ pour l'exécution concurrente S : les noeuds sont des actions
   - Ajouter les arcs intra-transaction : $A_i^k \rightarrow A_i^p$, tel que les deux actions A_i^k est une lecture et A_i^p est une écriture et que A_i^k précéde A_i^p
   - Ajouter les arcs inter-transaction : $A_i^k \rightarrow A_j^p$ tel que A_i^k portent sur le même élément, A_i^k est une ecriture et A_j^p une lecture, A_i^k précéde A_j^p.

Modélisation du flux d'entrée sortie par un graphe

1. Construction du Graphe $G(S)$ pour l'exécution concurrente S : les noeuds sont des actions
   - Ajouter les arcs intra-transaction : $A_i^k \rightarrow A_i^p$, tel que les deux actions A_i^k est une lecture et A_i^p est une écriture et que A_i^k précéde A_i^p
   - Ajouter les arcs inter-transaction : $A_i^k \rightarrow A_j^p$ tel que A_i^k portent sur le même élément, A_i^k est une ecriture et A_j^p une lecture, A_i^k précéde A_j^p.
2. Connecter un noeud S (source) à toutes les lectures initiales et un noeud P (puit) à toutes les dernières écritures

Vue Sérialisabilité

Modélisation du flux d'entrée sortie par un graphe

1. Construction du Graphe $G(S)$ pour l'exécution concurrente S : les noeuds sont des actions
   - Ajouter les arcs intra-transaction : $A_i^k \rightarrow A_i^p$, tel que les deux actions A_i^k est une lecture et A_i^p est une écriture et que A_i^k précéde A_i^p
   - Ajouter les arcs inter-transaction : $A_i^k \rightarrow A_j^p$ tel que A_i^k portent sur le même élément, A_i^k est une ecriture et A_j^p une lecture, A_i^k précéde A_j^p.
2. Connecter un noeud S (source) à toutes les lectures initiales et un noeud P (puit) à toutes les dernières écritures

S est Vue sérialisable si il existe un graphe d'une exécution en série qui est équivalent

Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

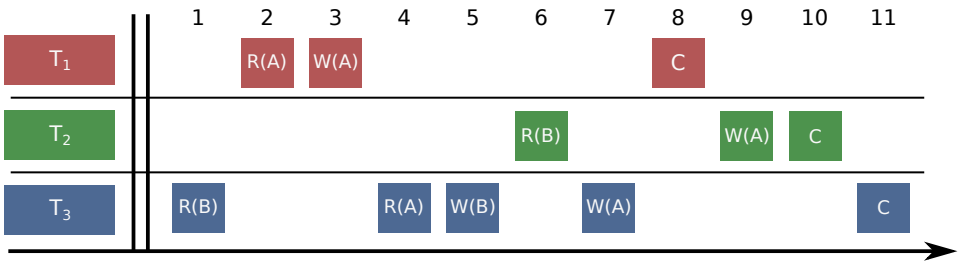
Sérialisable ?

Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

Sérialisable ?

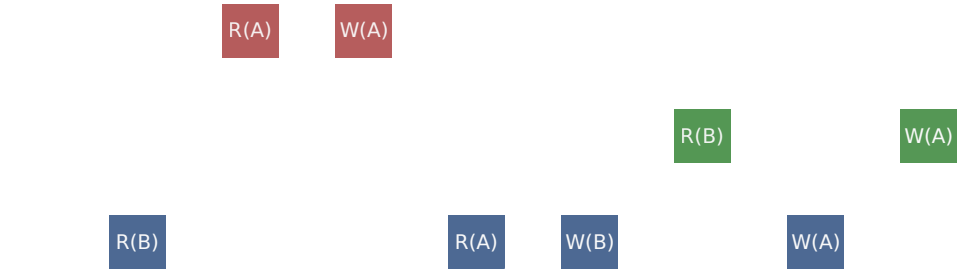


Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

Sérialisable ?

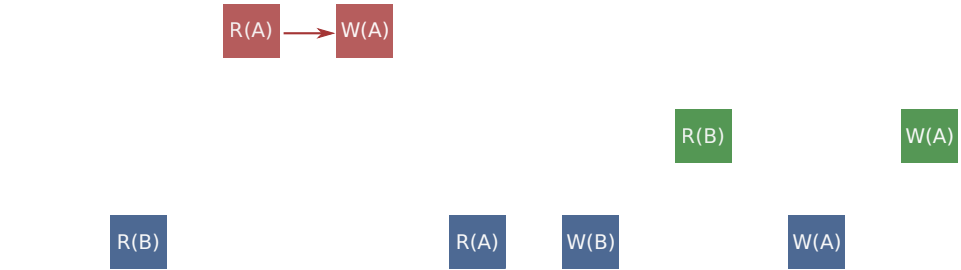


Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

Sérialisable ?



Vue sérialisabilité

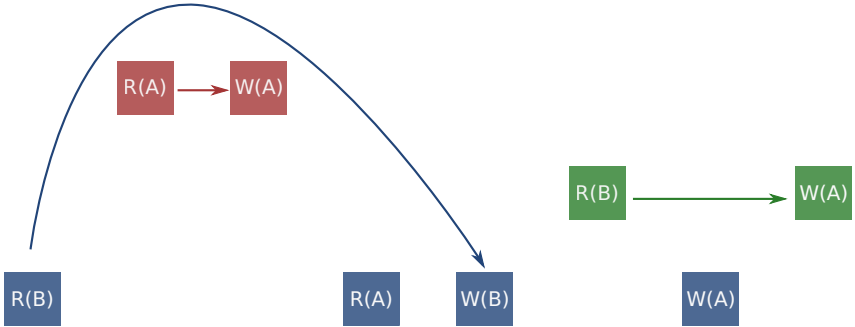
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

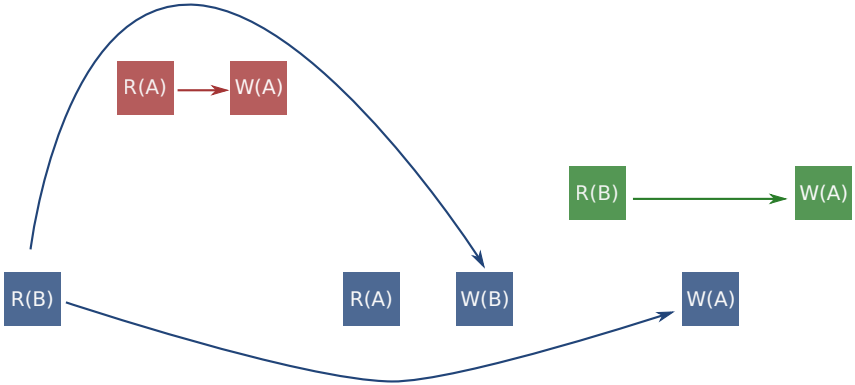
Sérialisable ?



Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

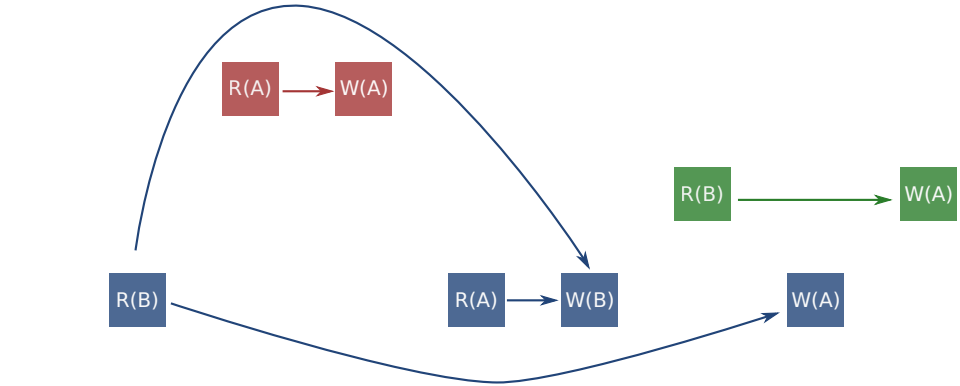


Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

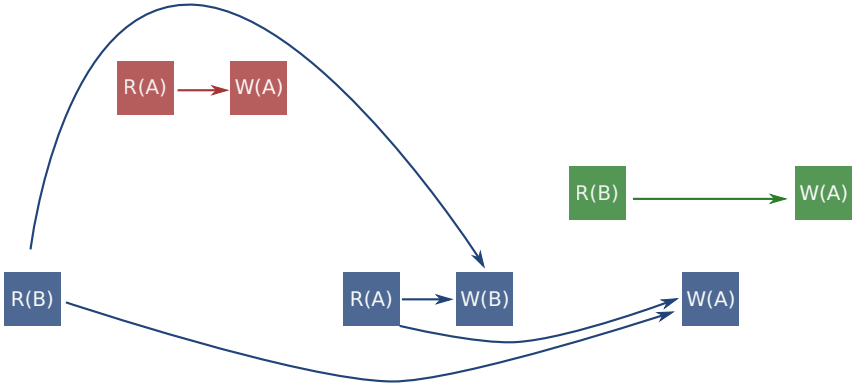
Sérialisable ?



Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

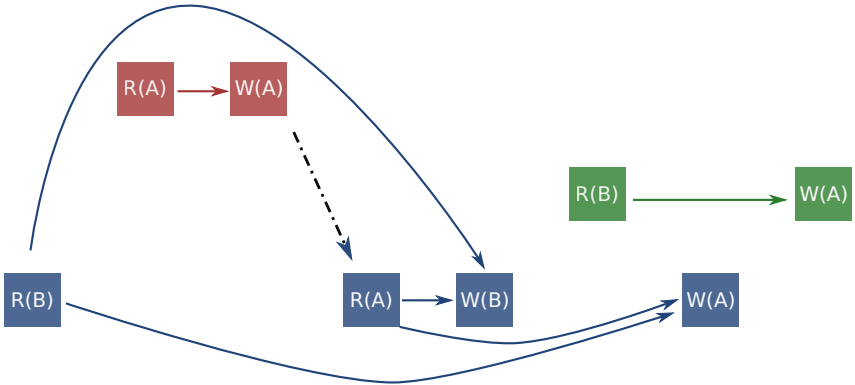


Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

Sérialisable ?

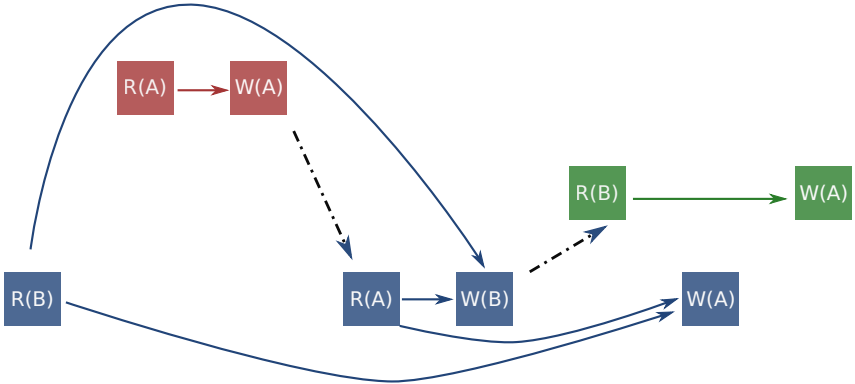


Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

Sérialisable ?

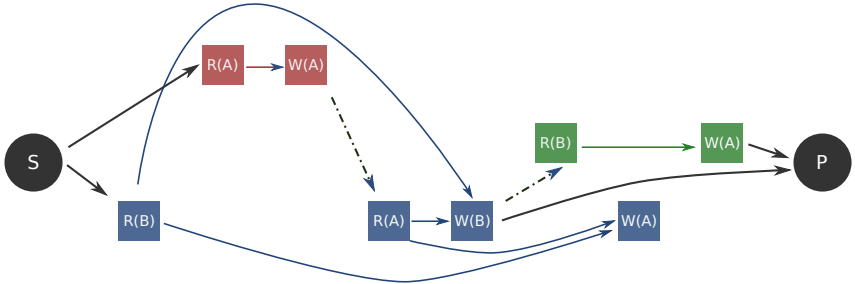


Vue sérialisabilité

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B)	R(A)	W(A)	R(A)	W(B)	R(B)	W(A)	C	W(A)	C	C

Sérialisable ?

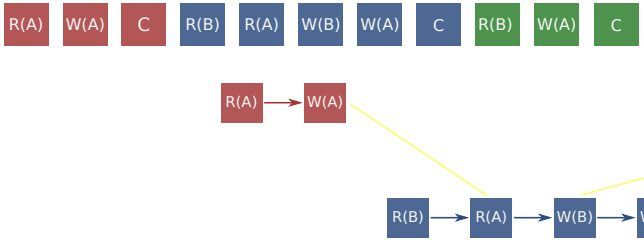


$\rightarrow$ **Trouver une exécution en série avec le même graphe !!!**

Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):

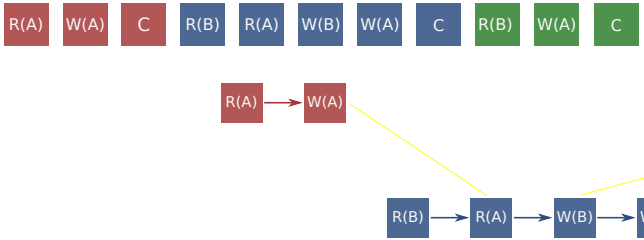
→ **Trouver une exécution en série avec le même graphe !!!**

Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):



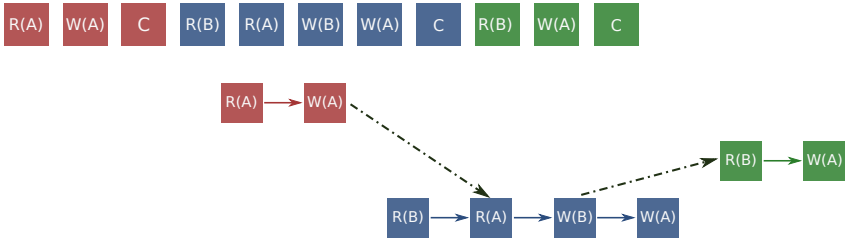
→ **Trouver une exécution en série avec le même graphe !!!**

Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):



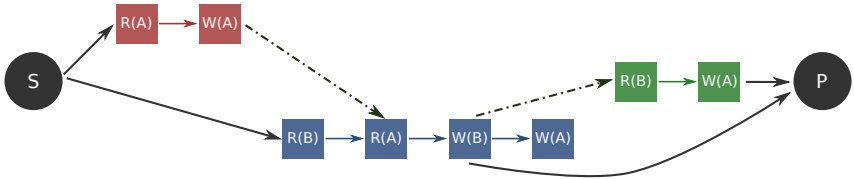
→ **Trouver une exécution en série avec le même graphe !!!**

Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):



→ **Trouver une exécution en série avec le même graphe !!!**

Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):



Même graphe → équivalent à $T_1 \rightarrow T_3 \rightarrow T_2$

Sérialisabilité globale

équivalence globale
Équivalence où l'on considère seulements les actions qui ont un impact sur le flux de sortie

équivalence globale
Équivalence où l'on considère seulements les actions qui ont un impact sur le flux de sortie

1. Construction du Graphe $G(S)$ pour l'exécution concurrente S : les noeuds sont des actions
   - Ajouter les arcs intra-transaction : $A_i^k \rightarrow A_i^p$, tel que les deux actions A_i^k est une lecture et A_i^p est une écriture et que A_i^k précéde A_i^p
   - Ajouter les arcs inter-transaction : $A_i^k \rightarrow A_j^p$ tel que A_i^k portent sur le même élément, A_i^k est une ecriture et A_j^p une lecture, A_i^k précéde A_j^p.

équivalence globale
Équivalence où l'on considère seulements les actions qui ont un impact sur le flux de sortie

1. Construction du Graphe $G(S)$ pour l'exécution concurrente S : les noeuds sont des actions
   - Ajouter les arcs intra-transaction : $A_i^k \rightarrow A_i^p$, tel que les deux actions A_i^k est une lecture et A_i^p est une écriture et que A_i^k précéde A_i^p
   - Ajouter les arcs inter-transaction : $A_i^k \rightarrow A_j^p$ tel que A_i^k portent sur le même élément, A_i^k est une ecriture et A_j^p une lecture, A_i^k précéde A_j^p.
2. Connecter un noeud S (source) à toutes les premières lectures de la valeur initiale et un noeud P (puit) à toutes les dernières écritures
3. Suppression de la totalité des noeuds n'ayant pas de chemin jusqu'à la sortie

Sérialisabilité globale

équivalence globale
Équivalence où l'on considère seulements les actions qui ont un impact sur le flux de sortie

1. Construction du Graphe $G(S)$ pour l'exécution concurrente S : les noeuds sont des actions
   - Ajouter les arcs intra-transaction : $A_i^k \rightarrow A_i^p$, tel que les deux actions A_i^k est une lecture et A_i^p est une écriture et que A_i^k précéde A_i^p
   - Ajouter les arcs inter-transaction : $A_i^k \rightarrow A_j^p$ tel que A_i^k portent sur le même élément, A_i^k est une ecriture et A_j^p une lecture, A_i^k précéde A_j^p.
2. Connecter un noeud S (source) à toutes les premières lectures de la valeur initiale et un noeud P (puit) à toutes les dernières écritures
3. Suppression de la totalité des noeuds n'ayant pas de chemin jusqu'à la sortie

S est globalement sérialisable s'il existe un graphe d'une exécution en série équivalent

sérialisabilité globale

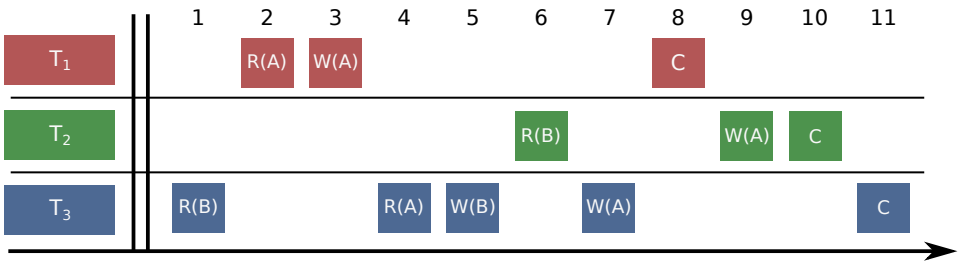
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

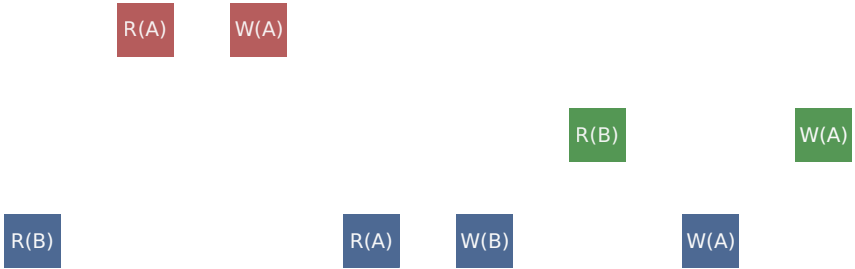
Sérialisable ?



sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

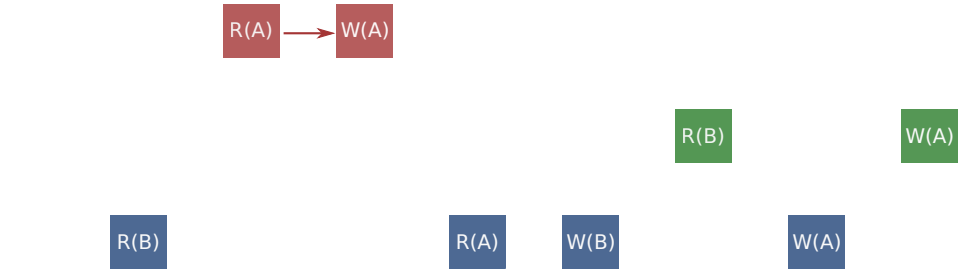


sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

Sérialisable ?

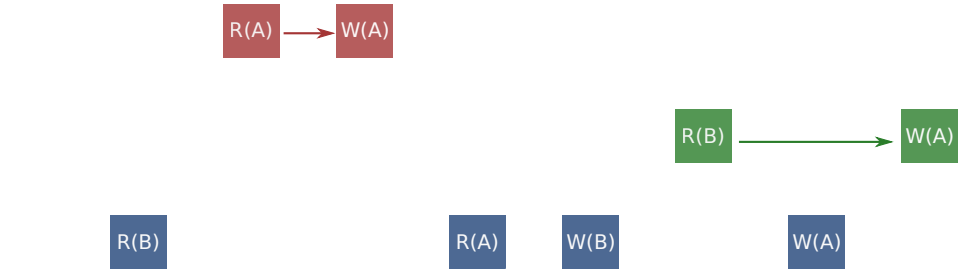


sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

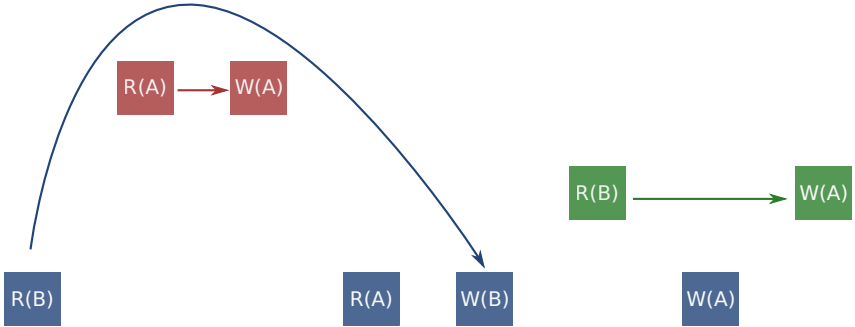
Sérialisable ?



sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

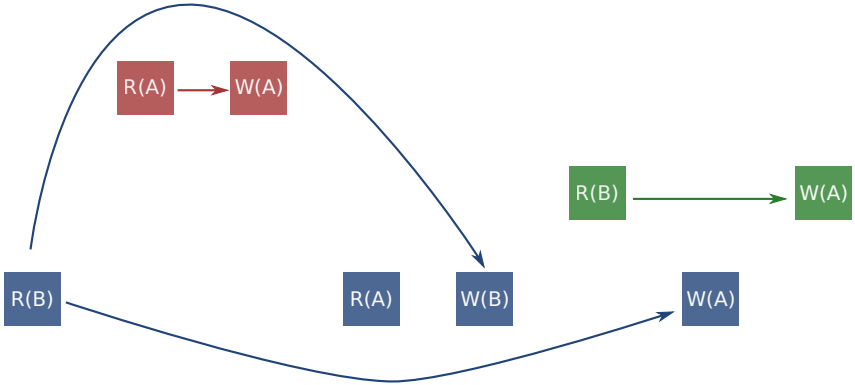
Sérialisable ?



sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

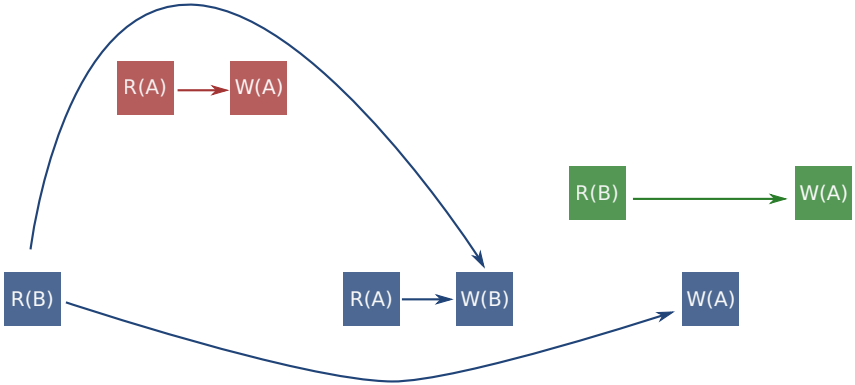
Sérialisable ?



sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

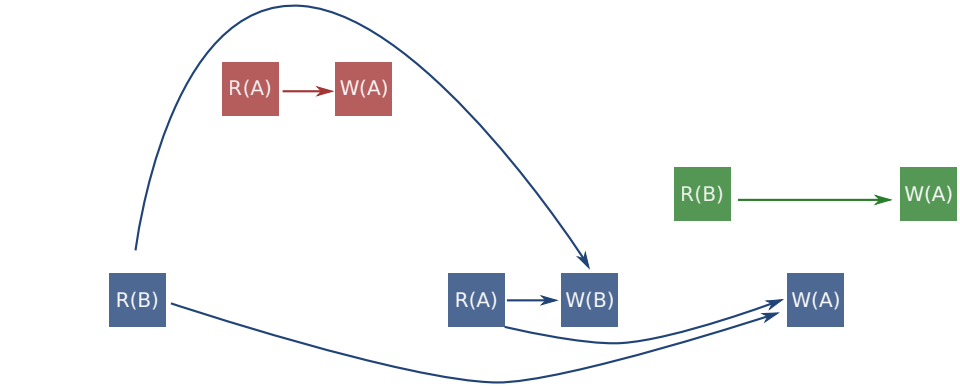


sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

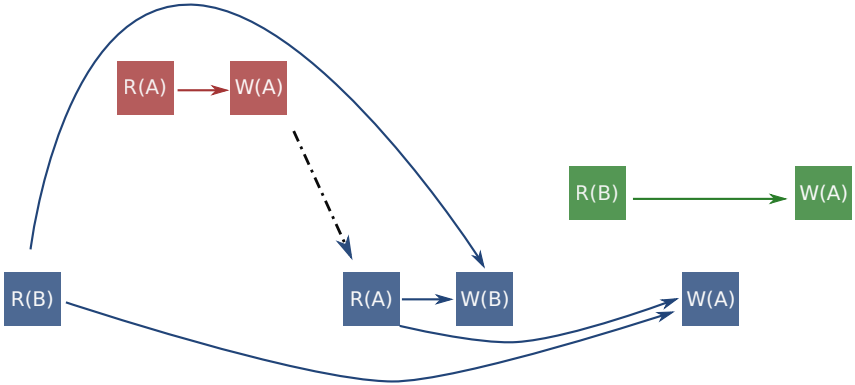
Sérialisable ?



sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?

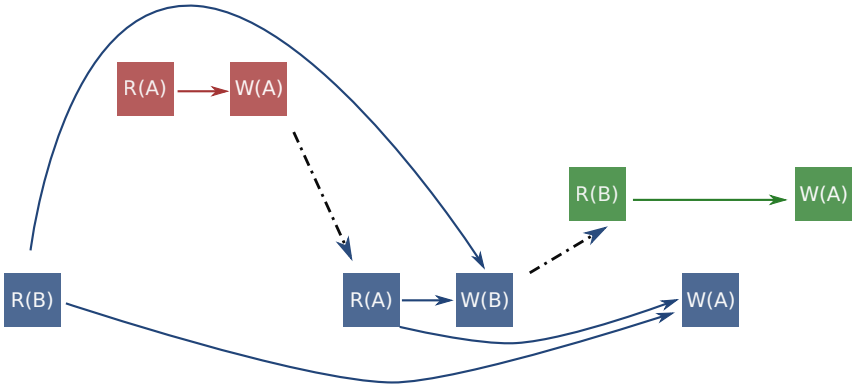


sérialisabilité globale

On considère l'exécution de trois transactions (T_1, T_2, T_3)

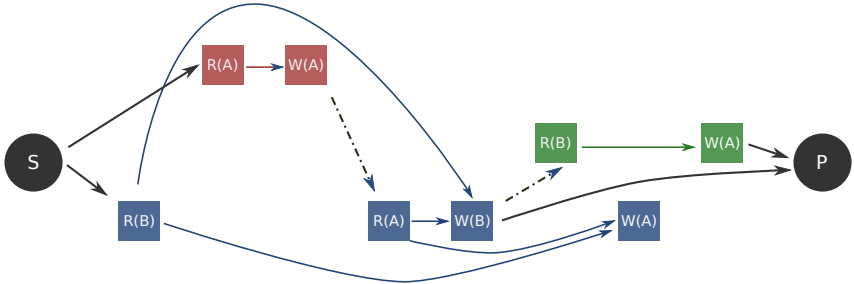
R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

Sérialisable ?



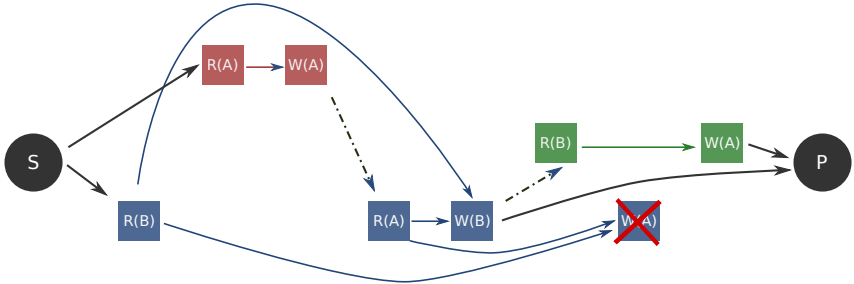
On considère l'exécution de trois transactions (T_1, T_2, T_3)

Sérialisable ?



On considère l'exécution de trois transactions (T_1, T_2, T_3)

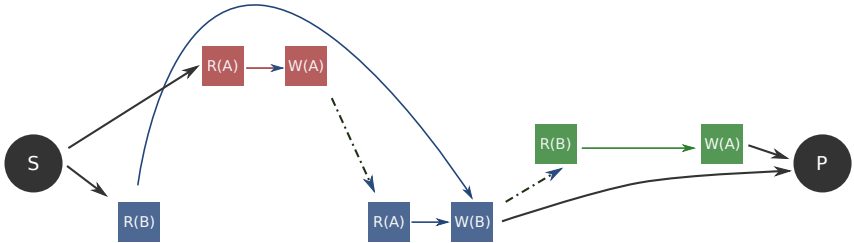
Sérialisable ?



On considère l'exécution de trois transactions (T_1, T_2, T_3)

R(B) R(A) W(A) R(A) W(B) R(B) W(A) C W(A) C C

Sérialisable ?



Sérialisabilité globale

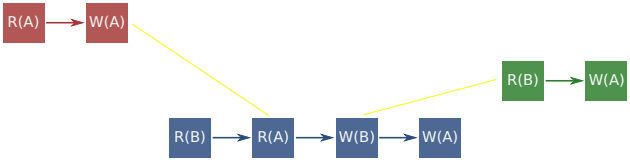
$\rightarrow$ **Trouver une exécution en série avec le même graphe !!!**

Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):

Sérialisabilité globale

→ **Trouver une exécution en série avec le même graphe !!!**

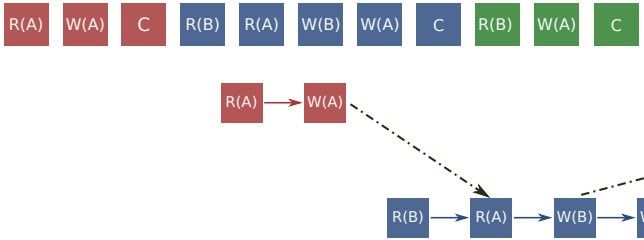
Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):



Sérialisabilité globale

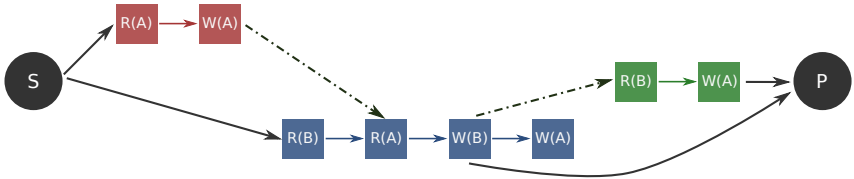
→ **Trouver une exécution en série avec le même graphe !!!**

Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):



→ **Trouver une exécution en série avec le même graphe !!!**

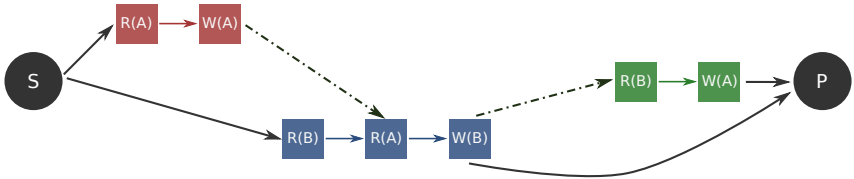
Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):



Même graphe → équivalent à $T_1 \rightarrow T_3 \rightarrow T_2$

→ **Trouver une exécution en série avec le même graphe !!!**

Essayons la suivante ($T_1 \rightarrow T_3 \rightarrow T_2$):



Même graphe → équivalent à $T_1 \rightarrow T_3 \rightarrow T_2$

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”
- L’éxécution de la sequence d’actions ne produit pas d’incohérence

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”
- L’éxécution de la sequence d’actions ne produit pas d’incohérence

Donc exécution de la transaction !!

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”
- L’éxécution de la sequence d’actions ne produit pas d’incohérence

Donc exécution de la transaction !!

Si elle est non sérialisable :

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”
- L’éxécution de la sequence d’actions ne produit pas d’incohérence

Donc exécution de la transaction !!

Si elle est non sérialisable :

⚠ **Risque de produire une incohérence**

→ On annule la transaction

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”
- L’éxécution de la sequence d’actions ne produit pas d’incohérence

Donc exécution de la transaction !!

Si elle est non sérialisable :

⚠ **Risque de produire une incohérence**

→ On annule la transaction

Puis ?

- Signifier à l’utilisateur que sa requête n’a pas pu être exécutée...

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”
- L’éxécution de la sequence d’actions ne produit pas d’incohérence

Donc exécution de la transaction !!

Si elle est non sérialisable :

⚠ **Risque de produire une incohérence**

→ On annule la transaction

Puis ?

- Signifier à l’utilisateur que sa requête n’a pas pu être exécutée...
- Trouver un nouvel ordre d’action...

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”
- L’éxécution de la sequence d’actions ne produit pas d’incohérence

Donc exécution de la transaction !!

Si elle est non sérialisable :

⚠ **Risque de produire une incohérence**

→ On annule la transaction

Puis ?

- Signifier à l’utilisateur que sa requête n’a pas pu être exécutée...
- Trouver un nouvel ordre d’action...
- Éviter la création d’incohérence en utilisant des verrous

Sérialisabilité : Problèmes ?

Si une exécution est sérialisable :

- Il existe une exécution en série “équivalente”
- L’éxécution de la sequence d’actions ne produit pas d’incohérence

Donc exécution de la transaction !!

Si elle est non sérialisable :

⚠ **Risque de produire une incohérence**

→ On annule la transaction

Puis ?

- Signifier à l’utilisateur que sa requête n’a pas pu être exécutée...
- Trouver un nouvel ordre d’action...
- Éviter la création d’incohérence en utilisant des verrous (**Prochain cours**)

Des questions ?