Transactions dans les systèmes multibases

Master ALMA
Patricia Serrano Alvarado
Laboratoire d' Informatique de Nantes Atlantique (LINA)
Patricia.Serrano-Alvarado@univ-nantes.fr

Autonomie, Hétérogénéité, Distribution

- On peut classifier les systèmes de gestion de bases de données (SGBD) selon trois dimensions génériques : leur degré d'autonomie, d'hétérogénéité et de distribution

- Autonomie

  Indique le degré d'indépendance des composants du système. Ceci concerne, entre autres, la liberté de choisir le type de données utilisées, la manière de les manipuler, la liberté de décider le type d'information à partager avec le système global, ainsi que l'indépendance d'exécution.

  Fait référence à la distribution du contrôle

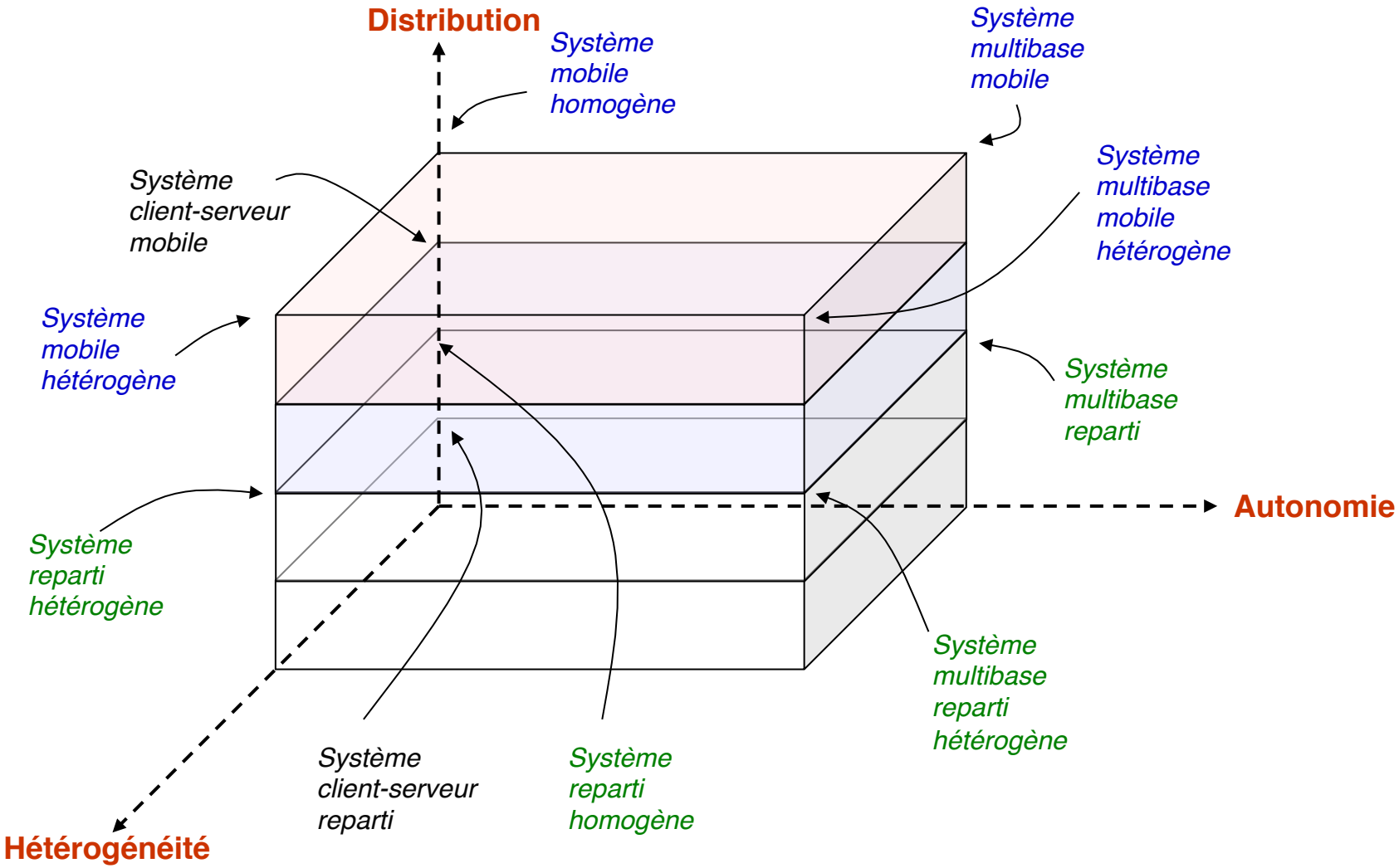
- Hétérogénéité

  Fait référence à l'hétérogénéité du matériel informatique mais aussi des gestionnaires de données : langages de requêtes, gestionnaires de requêtes, gestionnaires de transactions, etc.

- Distribution

  Fait référence à la distribution physique des données mais aussi aux fonctions de gestion de données. En général, trois types de distribution sont considérés : centralisé, client-serveur et pair à pair.

Classification de SGBD



Plan

- Ordonnancement de transactions
  - Sérialisabilité des transactions centralisées
  - Sérialisabilité répartie globale (ordonnancements sérialisables, fortement sérialisables, rigoureux, sérialisabilité locale, à deux niveaux, Epsilon sérialisabilité)
- Protocoles de validation de transactions réparties
  - Validation à deux phases (2PC) et ses variations (2PC-PC, 2PC-PA)
  - Refaire et compenser les transactions annulées
- Transactions étendues
  - Transactions emboîtées fermées
  - Transactions emboîtées ouvertes
  - DOM
  - Sagas
  - Transactions flexibles

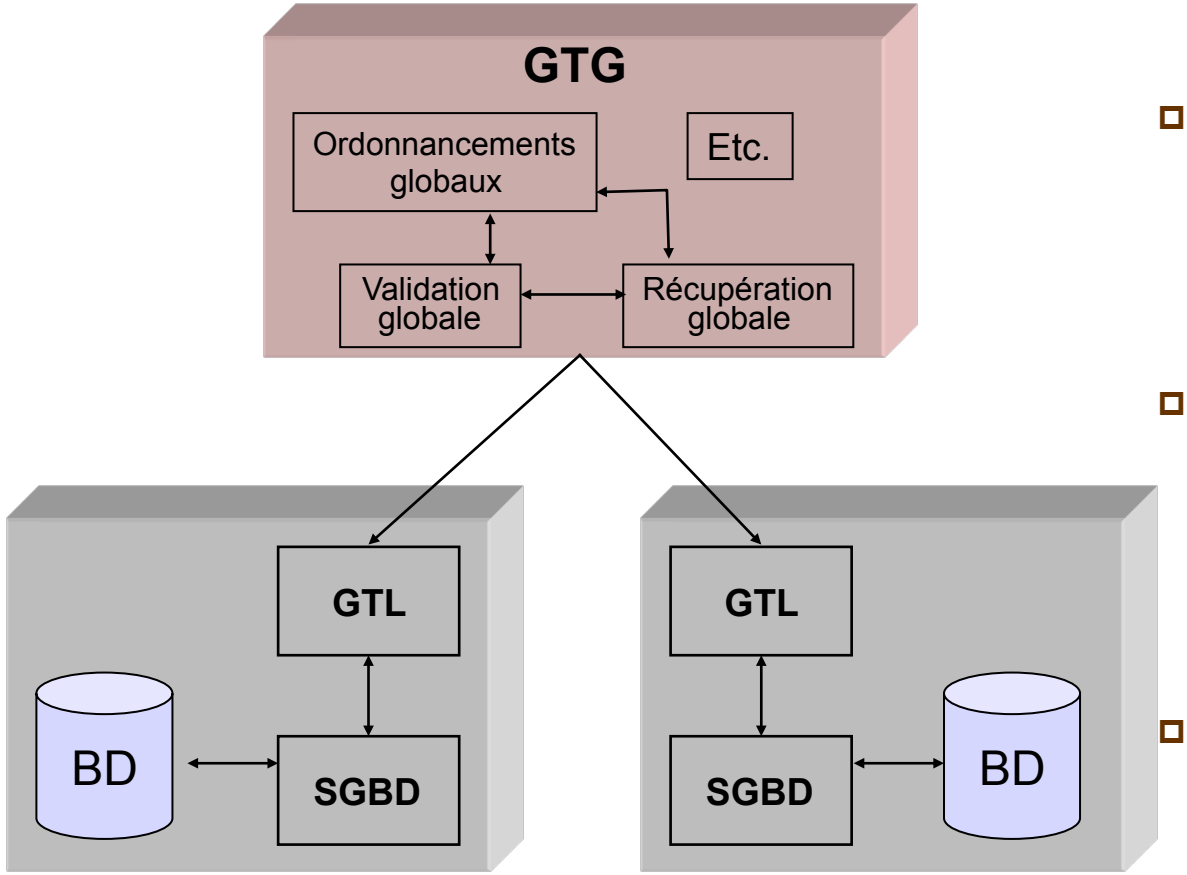
Plan

- Ordonnancement de transactions
  - Sérialisabilité des transactions centralisées
  - Sérialisabilité répartie globale (ordonnancements sérialisables, fortement sérialisables, sérialisabilité locale, à deux niveaux, Epsilon sérialisabilité)
- Protocoles de validation de transactions réparties
  - Validation à deux phases (2PC) et ses variations (2PC-PC, 2PC-PA)
  - Refaire et compenser les transactions annulées
- Transactions étendues
  - Transactions emboîtées fermées
  - Transactions emboîtées ouvertes
  - DOM
  - Sagas
  - Transactions flexibles

Ordonnancements dans les systèmes multibases

- Ordonnancements sérialisables
- Fortement sérialisables
- Sérialisabilité locale
- À deux niveaux
- Epsilon sérialisabilité

SMBD
(Système multibase de données)



- Un gestionnaire de transactions globales (GTG) coordonne un ensemble de gestionnaires de transactions locales (GTL)
- Une transaction globale (TG_i) est composée d'un ensemble de transactions locales appelées sous-transactions (tg_{ij}). Chaque tg_{ij} s'exécute sur un GTL_j différent
- L'ordonnancement des transactions exécutées sur un GTL_j (appelé ordonnancement locale) est composé des transactions locales (tl_j) et des sous-transactions qui y sont exécutées.
- En général, les GTL_j sont autonomes et peuvent générer leurs ordonnancements en utilisant le protocole de contrôle de concurrence de leur choix

Sérialisabilité globale (1)

- La sérialisabilité globale veut que :
  - Chaque GTL_j assure un ordonnancement local sérialisable et que
  - Au niveau global les sous-transactions soient perçues dans le même ordre relatif

$TG_i = \{(tg_{ii} \rightarrow GTL_i\ A), (tg_{ij} \rightarrow GTL_j\ B\}$

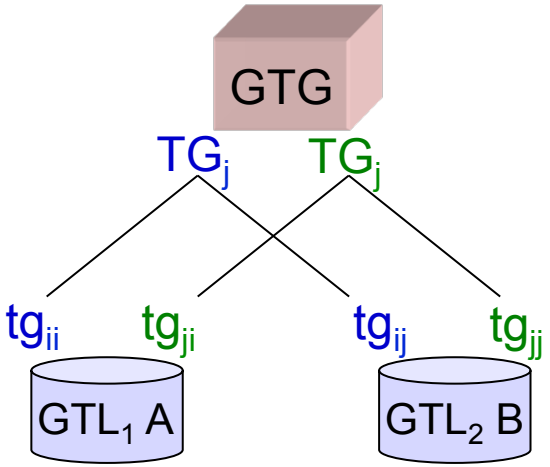
$TG_j = \{(tg_{ji} \rightarrow GTL_i\ A), (tg_{jj} \rightarrow GTL_j\ B\}$

Deux ordonnancements corrects possibles :

$S_1 = \{tg_{ii} < tg_{ji}\}\ \{tg_{ij} < tg_{jj}\}$

$S_2 = \{tg_{ji} < tg_{ii}\}\ \{tg_{jj} < tg_{ij}\}$

Où < est une relation d'ordre



Sérialisabilité globale (2)

- Si les SGBD sont autonomes, en général, ils ne partagent pas l'information sur leurs ordonnancements
- Ceci peut rendre difficile la tache d'assurer la sérialisabilité globale
- Si les ordonnancements locaux sont hors de la porté du GTG seule la demande d'exécution de sous-transactions est sous control global
- Les conflits indirects sont l'une des conséquences de ses caractéristiques
- Un conflit indirect peut être généré lorsque
  - Dans un ordonnancement local, une sous-transaction globale tg_{ii} a une relation d'ordre avec une transaction locale tl_j, laquelle a une relation d'ordre avec une sous-transaction tg_{ij}
  - Puisque tg_{ii} et tg_{ij} ne sont pas en conflit direct, l'ordonnanceur global ne peut pas percevoir la relation d'ordre entraînée par tl_j

Sérialisabilité globale (3)

Exemple de conflit indirect

Soient TG_i et TG_j deux transactions globales :
TG_i={(lecture_tg_{ii} → GTL_i A), (écriture_tg_{ij} → GTL_j B)}
TG_j={(écriture_tg_{ji} → GTL_i A), (lecture_tg_{jj} → GTL_j C)}

Soit tl_k une transaction locale exécuté sur GTL_j
tl_k={(lecture_tl_k → GTL_j B), (écriture_tl_k → GTL_j C)}

Soit O_i l'ordonnancement généré par une exécution séquentielle :
O_i = {(lecture_tgii → A) < validation_tg_{ii} < (écriture_tg_{ji} → A) < validation_tg_{ji}}

Ceci donne : TG_i < TG_j

Soit O_j l'ordonnancement généré en utilisant une approche par certification :
O_j = {lecture_tl_k → B) < (écriture_tg_{ij} → B) < validation_tg_{ij} < (lecture_tg_{jj} → C) < validation_tg_{jj} < (écriture_tl_k → C) < validation_tl_k}

Ceci donne : TG_j < tl_k < TG_i
Ce qui donne un cycle dans le graphe de sérialisation global !

Sérialisabilité globale (3)

Exemple de conflit indirect

Soient TG_i et TG_j deux transactions globales :
TG_i={(lecture_tg_{ii} → GTL_i A), (lecture_tg_{ij} → GTL_j B)}
TG_j={(écriture_tg_{ji} → GTL_i A), (écriture_tg_{jj} → GTL_j C)}

Soit tl_k une transaction locale exécuté sur GTL_j
tl_k={(lecture_tl_k → GTL_j B), (écriture_tl_k → GTL_j C)}

Soit O_i l'ordonnancement généré par une exécution séquentielle :
O_i = {(lecture_tgii → A) < validation_tg_{ii} < (écriture_tg_{ji} → A) < validation_tg_{ji}}

Ceci donne : TG_i < TG_j

Soit O_j l'ordonnancement généré en utilisant une approche par certification en avant:
O_j = {écriture_tl_k → B) < (lecture_tg_{ij} → B) < validation_tg_{ij} < (écriture_tg_{jj} → C) < validation_tg_{jj}
< (écriture_tl_k → C) < validation_tl_k}

Ceci donne : TG_j < tl_k < TG_i
Ce qui donne un cycle dans le graphe de sérialisation global !

Optimistic Ticket Method (OTM)

- Technique qui assure une sérialisabilité globale
- Approche :
  - Considère de SGBD locaux où différents protocoles de contrôle de concurrence peuvent être utilisés
  - Introduit de conflits directs entre les transactions globales à l'aide d'une donnée particulier (un ticket)
  - L'accès au ticket dans chaque site entraîne l'ordre relatif entre les sous-transactions
  - Les SGBD locaux doivent générer des ordonnancements sérialisables et recouvrables
  - Les SGBD locaux doivent atteindre la validation globale pour valider localement leurs transactions (état de préparation du protocole 2PC)

Ordonnancement fortement sérialisable

- Un ordonnancement fortement sérialisable assure que pour toute paire de transactions t et t', si la dernière opération de t est exécutée avant la première opération de t', alors t précède t' dans l'ordonnancement

- Entre autres, 2PL génère des ordonnancements fortement sérialisables

- Si tous les sites génèrent des ordonnancements fortement sérialisables alors l'exécution séquentielle de TG garanti la sérialisabilité globale

Ordonnancements à point de sérialisation

- Un point de sérialisation ps_i est une opération particulière d'une transaction ti.
- Un ordonnancement est à point de sérialisation s'il assure que pour toute paire de transactions t_i, t_j, si ps_i est exécutée avant ps_j, alors t_i précède t_j dans l'ordre de sérialisabilité
- Les ordonnancement à point de sérialisation sont un sous ensemble des ordonnancements fortement sérialisables
- Les ordonnancements à point de sérialisation permettent plus de concurrence que ceux fortement sérialisables. Seul les point à sérialisation sont exécutés dans le même ordre dans tous les sites
- Un cas particulier d'un point de sérialisation est la validation
- Les protocoles utilisés par les GTL peuvent être hétérogènes. Néanmoins, le GTG doit connaître les points de sérialisation

Ordonnancements globaux localement sérialisables (LSR)

- Technique relâchant la sérialisabilité globale
- Appliquée s'il n'y a pas de control global et si les GTL assurent la sérialisabilité
- Afin de garantir un ordonnancement LSR les contraintes d'intégrité globales doivent être interdites uniquement les contraintes d'intégrité locales sont permises
  - Une contrainte d'intégrité globale lie des données provenant de plusieurs sites. La vérification de ces contraintes est de la responsabilité du GTG
- Une approche similaire, nommée quasi-sérialisabilité, interdit en plus les dépendances de valeur
  - t_i a une dépendance de valeur sur t_j si t_i dépend d'une valeur produite par t_j

Sérialisabilité à deux niveaux (2LSR)

- Un ordonnancement est 2LSR si
  - Il est LSR et
  - Sa projection sur les transactions globales est sérialisable
- Deux types de données sont considérées
  - Locales : existaient avant la définition du système multibases
  - Globales : données insérées lors de la création du système multibases
- La répartition des données en global et local dans chaque site permet de définir une hiérarchie de modèles de systèmes multibases qui assurent la 2LSR, voici les restriction du modèle le plus général
  - Restrictions du modèle appelé trivial
    - Les transactions locales lissent et écrivent seulement sur des données locales
    - Les transactions globales lissent et écrivent seulement sur des données globales
  - Dans ce modèle, le niveau local est complètement découplé du global. D' autres modèles ont été proposés afin de faire interagir les deux niveaux.

Epsilon sérialisabilité (ESR)

- Relâche la sérialisabilité en tirant profit des incohérences tolérées par les applications afin d'augmenter les performances
- Permet d'une façon explicite une quantité limitée d'incohérence nommée epsilon
- Les limites d'incohérence sont représentés par un nombre de conflits acceptés entre les transactions
  - Une transaction contenant uniquement de lectures a une limite dite d'importation
  - Une transaction contenant d'écritures a une limite dite d'exportation
- Si le nombre de conflits acceptés es égal à zéro, alors les ordonnancements ESR sont sérialisables
- Des méthodes pour contrôler la divergence créé ont été proposés
- Si la limite est dépassée, des méthodes de restauration doivent être employées. La restauration peut être basée sur des techniques de compensation ou de refaire et défaire des transactions

Protocoles de validation

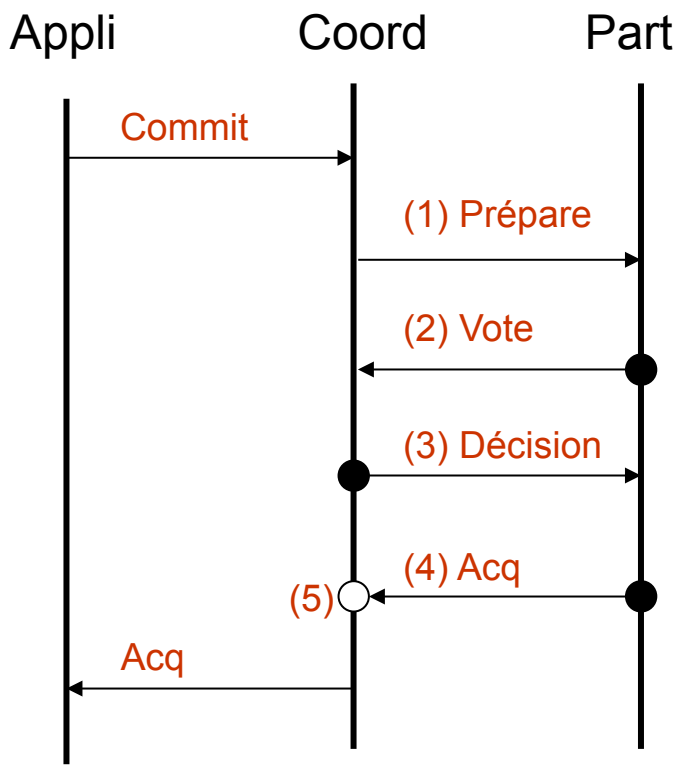
- Validation à deux phases (2PC)
- 2PC hiérarchique
- 2PC abandon/commit présumé
- Validation à trois phases (3PC)
- Validation à une phase (1PC)
- Reprise, transactions de compensation

L'atomicité

- L'atomicité est la notion du tout ou rien
  - Le gestionnaire des transactions doit garantir que toutes les opérations d'une transaction sont exécutées ou aucune d'entre elles
- Les protocoles de validation assurent la propriété d'atomicité
  - Protocoles à une, deux ou trois phases, etc.
  - Techniques pour refaire ou défaire
  - Techniques de compensation qui ne garantissent pas l'atomicité mais l'atomicité dite sémantique
- Protocoles utilisés dans un système réparti mais aussi dans un contexte centralisé multi-processus

Validation à deux phases - 2PC (1)

Schéma de communication



Journalisation

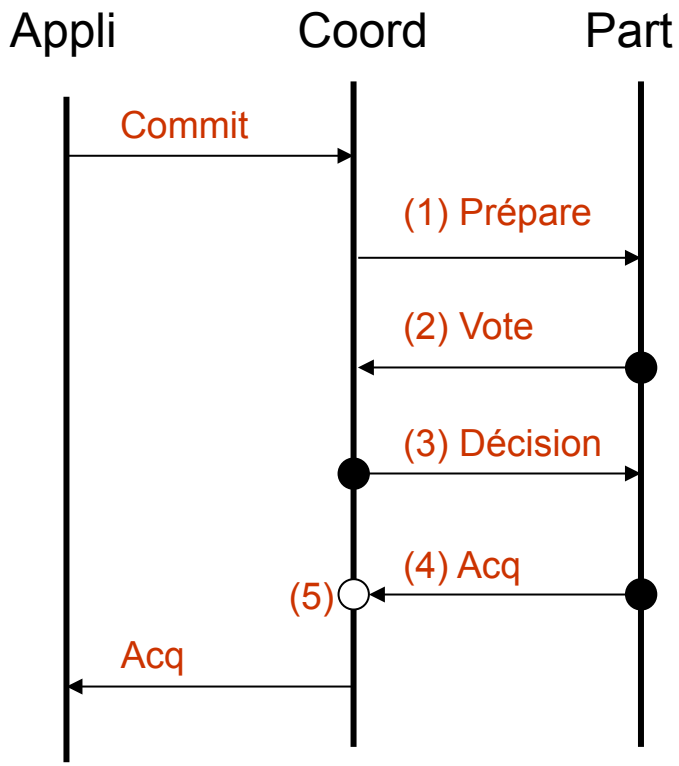
Qui	Quoi	Type
Part	(2) Vote, t_id, all_parts, coord	Forcée
Coord	(3) Décision	Forcée
Part	(4) Décision	Forcée
Coord	(5) Terminaison (après tous les acq)	Simple

- ● Journalisation forcée
  - Écriture immédiate sur disque
- ○ Journalisation simple
  - Écriture éventuelle sur le disque

- Il est conseillé d'utiliser 2PC en collaboration avec 2PL pour garantir la reprise

Validation à deux phases - 2PC (2)

Schéma de communication



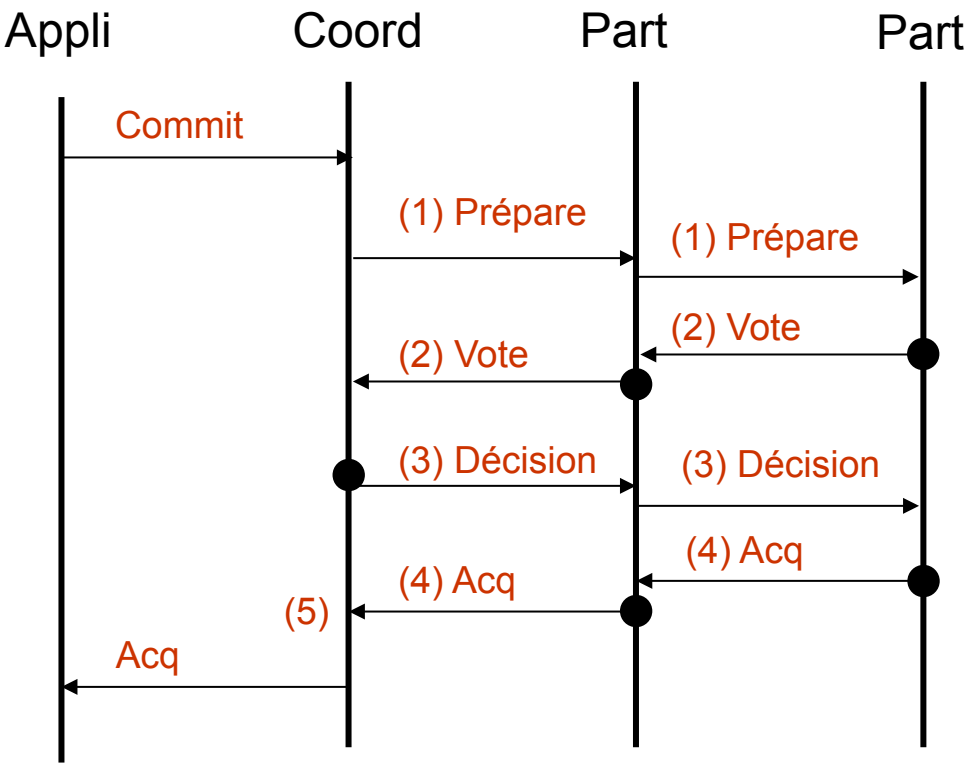
Reprise

Qui	Si panne après
Coord	(1) Abandon de la transaction et envoi d'abandon à all_parts et attente des acq. (2) Pas de décision : abandon. (3) Renvoi de décision et attente des acq. (5) Rien.
Part	- Pas de vote : abandon de la transaction. (2) Vote non : envoi d'acq au coord. (2) et (4) Vote oui et décision : envoi d'acq au coord. Vote oui pas décision : demande décision aux part. Pas de décision entre all_parts : blocage

2PC hiérarchique (1)

Le cas de la validation

Schéma de communication



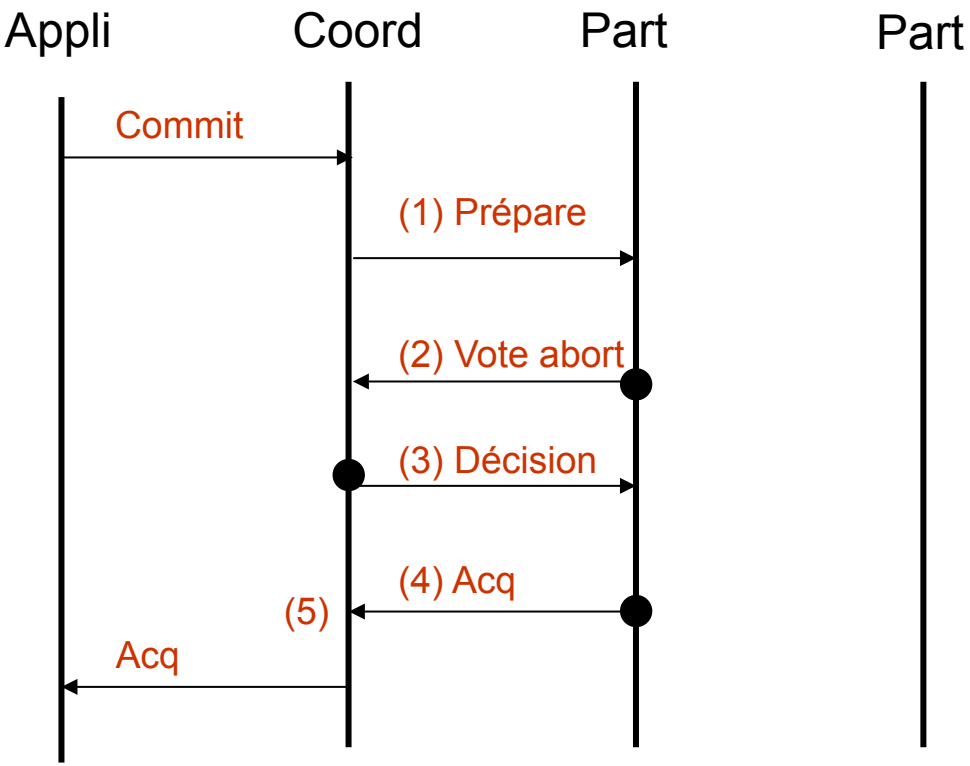
La communication entre subordonné et supérieur permet de limiter la communication entre sites distants.

Pas de différence au niveau des messages et de la journalisation avec 2PC

2PC hiérarchique (2)

Le cas de l'abandon

Schéma de communication



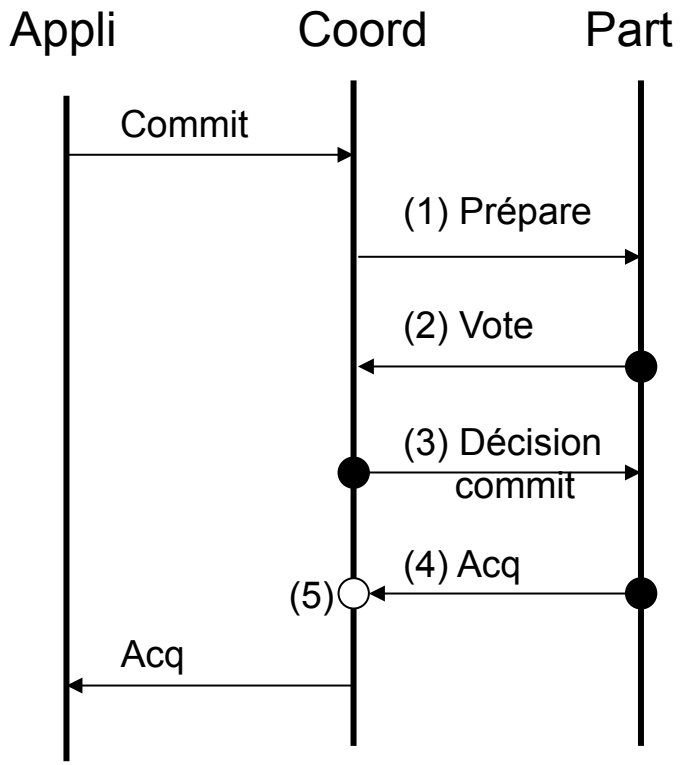
Si un site supérieur vote abandon, il n'envoi pas le message prépare à ses subordonnés.

Souvent utilisé avec des timeouts

Ce mécanisme a été adopté comme standard de mise en œuvre dans la validation atomique répartie (OSI TP).

2PC Abandon présumé - 2PC-PA (1)

Schéma de communication



Le cas de la validation

Journalisation

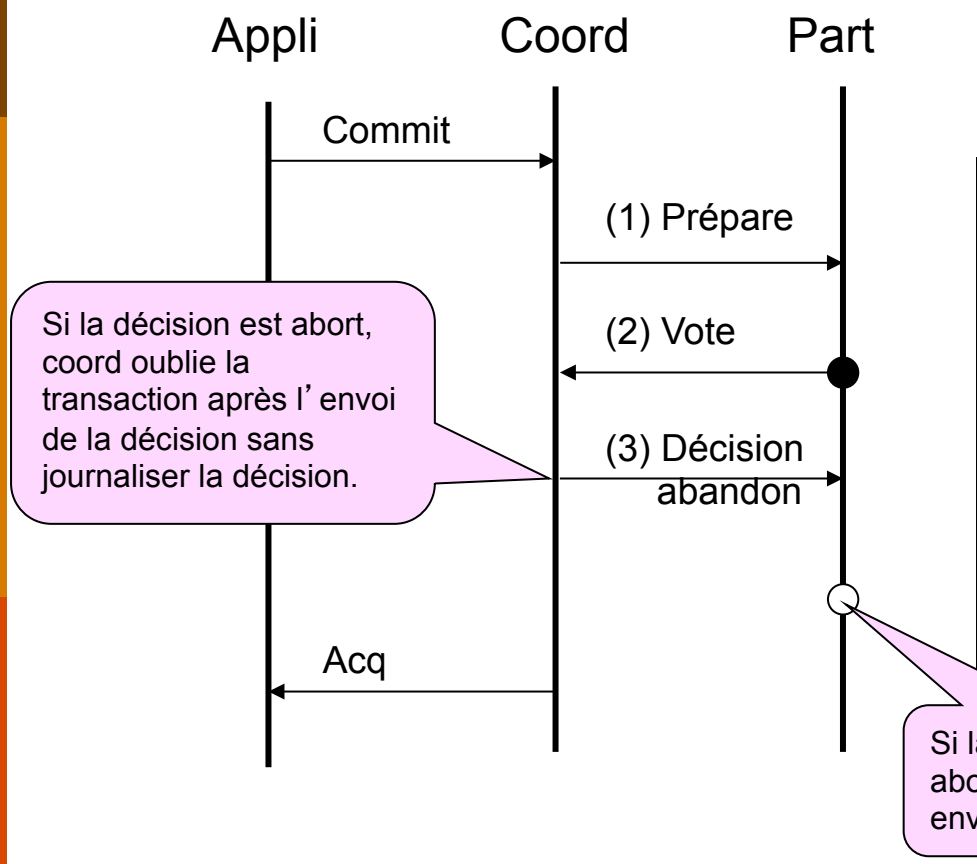
Qui	Quoi	Type
Part	(2) Vote, t_id, all_parts, coord	Forcée si vote oui Simple si vote non
Coord	(3) Décision	Forcée si commit
Part	(4) Décision	Forcée si commit Simple si abort
Coord	(5) Terminaison (après tous les ack en cas de commit)	Simple

- Le cas de la validation est le même que dans 2PC

2PC Abandon présumé - 2PC-PA (2)

Schéma de communication

Le cas de l'abandon



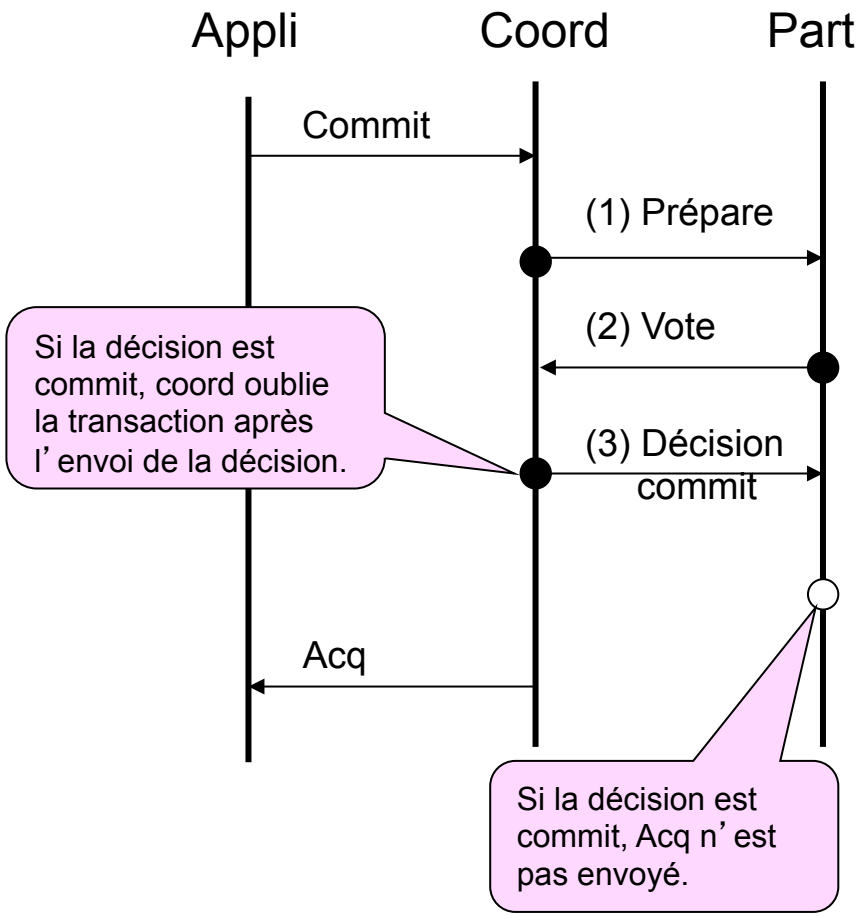
Journalisation

Qui	Quoi	Type
Part	(2) Vote, t_id, all_parts, coord	Forcée si vote oui Simple si vote non
Coord	(3) Décision	Forcée si commit
Part	(4) Décision	Forcée si commit Simple si abort
Coord	(5) Terminaison (après tous les ack en cas de commit)	Simple Rien si abort

- **Principe** : s'il n'y a pas d'information alors abandon de la transaction
- Elimine une journalisation forcée et l'acq d'une décision abort.
- Il a été adopté par ISO-OSI, DTP X/Open et OTS de l'OMG
- Utilisé également avec un schéma de communication hiérarchique et timeouts.

2PC Commit présumé -2PC-PC (1)

Schéma de communication



Le cas de la validation

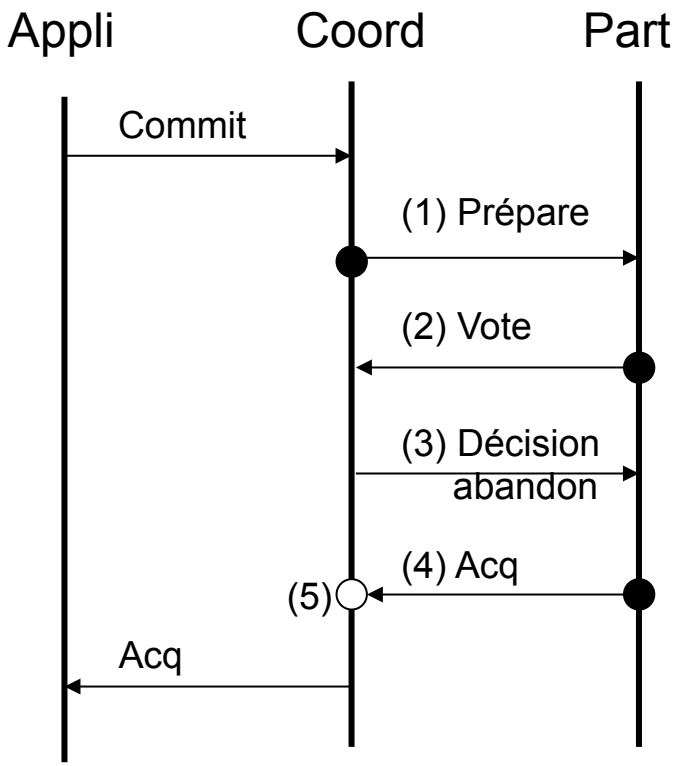
Journalisation

Qui	Quoi	Type
Coord	Prépare	Forcée
Part	(2) Vote, t_id, all_parts, coord	Forcée si vote oui Simple si vote non
Coord	(3) Décision	Forcée si commit Rien si abort
Part	(4) Décision	Simple si commit Forcée si abort
Coord	(5) Terminaison (après tous les ack)	Simple

- **Principe** : s'il n'y a pas d'info alors commit de la transaction
- Elimine l'acq d'une décision de commit

2PC Commit présumé -2PC-PC (2)

Schéma de communication



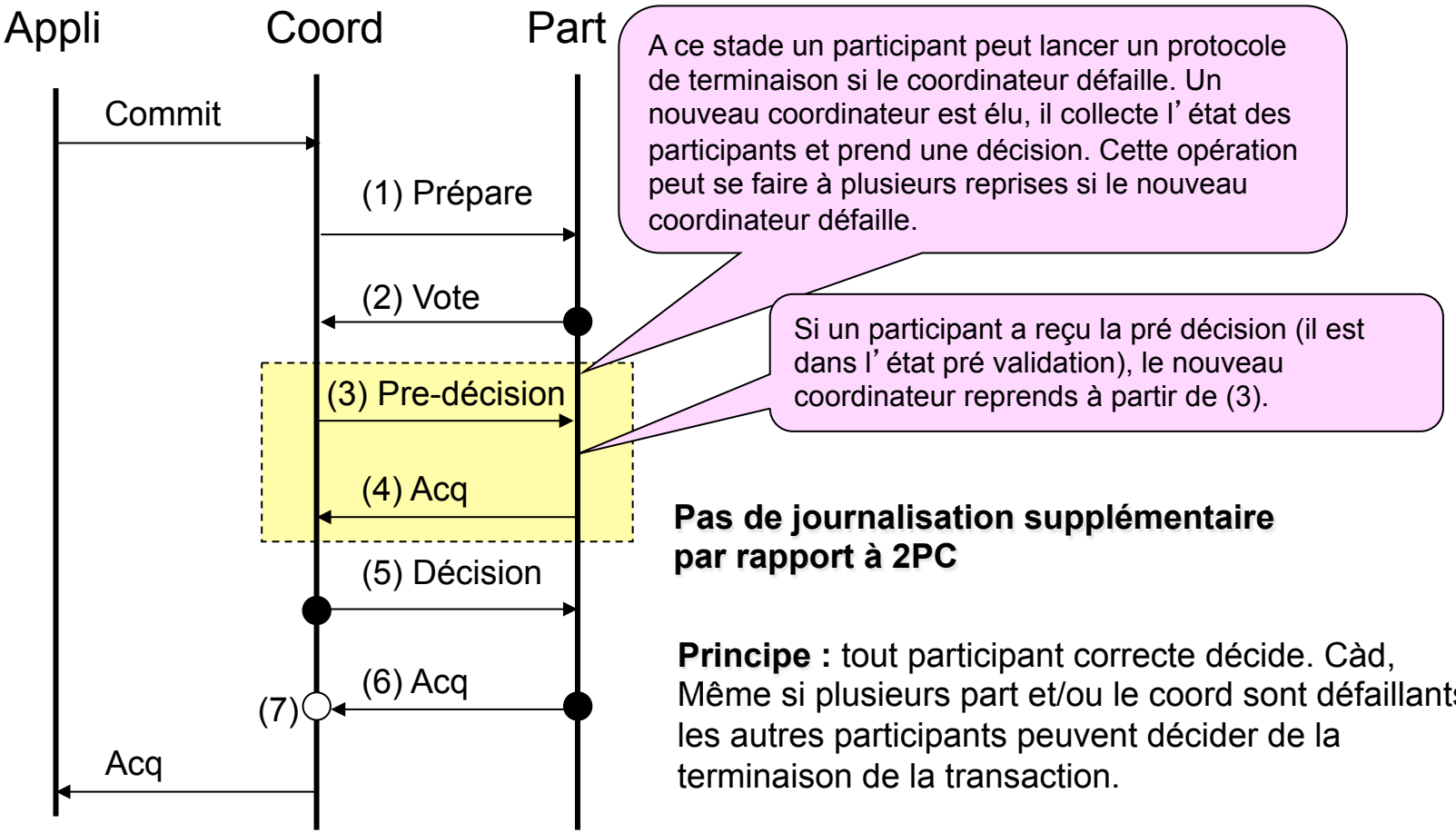
Le cas de l'abandon

Journalisation

Qui	Quoi	Type
Coord	Prépare	Forcée
Part	(2) Vote, t_id, all_parts, coord	Forcée si vote oui Simple si vote non
Coord	(3) Décision	Forcée si commit Rien si abort
Part	(4) Décision	Simple si commit Forcée si abort
Coord	(5) Terminaison (après tous les ack)	Simple

Validation à trois phases -3PC

Schéma de communication

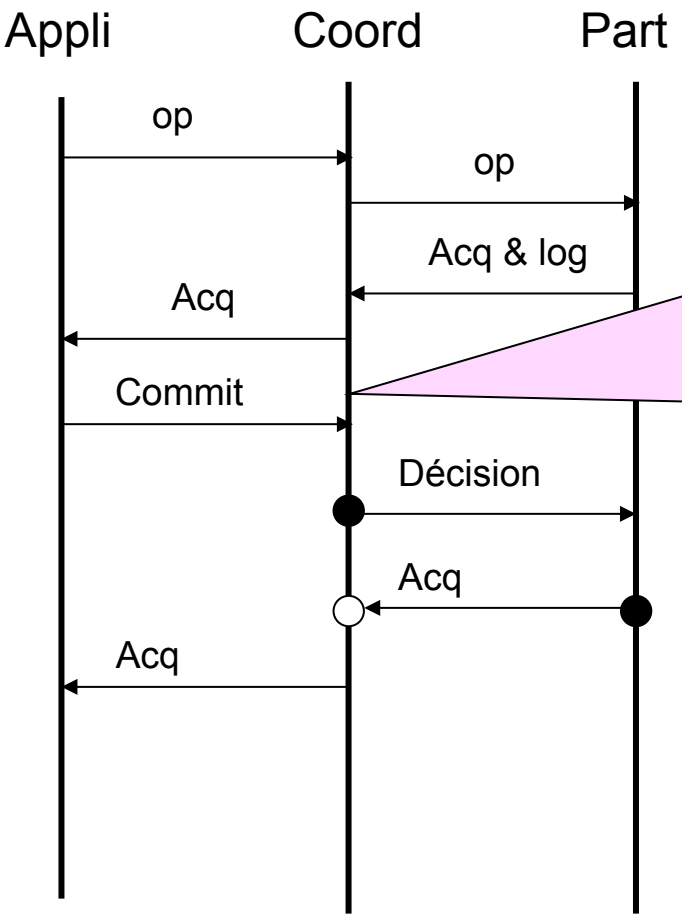


Pas de journalisation supplémentaire par rapport à 2PC

Principe : tout participant correcte décide. Càd, Même si plusieurs part et/ou le coord sont défaillants, les autres participants peuvent décider de la terminaison de la transaction.

Protocoles de validation à 1 phase

Schéma de communication



Hypothèses importantes avant le début de la validation.
- Opérations exécutées avec succès (acq)
- Contraintes d'intégrité vérifiées
- Protocoles CC pessimistes (non annulation en cascade)
- Durabilité garantie (les effets des transactions stockés dans des supports fiables)
- Envoi de journaux physiques ou logiques au coord pour assurer la récupération

Le partage des journaux limite l'autonomie des participants.
Protocole à utiliser avec de systèmes fédérés.

D'autres protocoles

- Unsolicited-vote
  - Chaque participant sait quelle est sa dernière opération
  - Après la dernière opération le participant envoi son vote au coordinateur
- Early prepare
  - Chaque participant utilise 2PL
  - Chaque opération est acquittée avec une journalisation forcée dans le journal du participant
  - Chaque acquittement est comme un vote prêt à valider
  - Pas de demande de vote

Différences entre les protocoles

- Nombre de messages
- Messages forcés ou simples
- Taille des messages
- Hypothèses faites avant de commencer le protocole de validation
  - Contrôle de concurrence
  - Tolérance aux pannes
    - Traitement des opérations par les participants (refaire/défaire, journaux logiques/physiques)
    - Journalisation dans le journal du coordinateur
    - Rôle du coordinateur
- Processus de récupération
- Moment de libération des journalisations du journal
- Propriétés
  - Validité : si tous les participants votent commit le commit est décidé
  - Unanimité : deux participant ne décident pas différemment
  - Non-trivialité : si tous les part votent commit et il n'y a pas de défaillance, tous les participants décident commit
  - Terminaison : si toutes les défaillances sont réparées et il n'y a pas de nouvelle défaillance alors tous les participants arrivent à une décision
  - Non-blocage : ni les participants ni le coordinateur reste bloqué

Tableau comparatif

Coût des protocoles

Protocole	Messages		Journalisations forcées	
	Commit	Abandon	Commit	Abandon
2PC				
2PC-PA				
2PC-PC				
3PC				
1PC				

2PC hiérarchique n'est pas inclus car son coût dépend de l'organisation de la hiérarchisation

D'autres techniques pour assurer l'atomicité

- Dans les systèmes multibases, lorsque les SGBD ne supportent pas l'état préparation, une simulation peut être faite
- Un représentant du gestionnaire global, appelé agent, est installé sur chaque site du système multibase
- Les agents jouent le rôle de participant et le gestionnaire global de coordinateur du protocole 2PC
- Problème à gérer : les SGBD peuvent abandonner une transaction même après que l'agent correspondant ait voté pour une validation globale
- Deux solutions peuvent être utilisées
  - Reprise des mises à jour
  - Reprise totale

Reprise de mises à jour (*redo*)

- Une transaction de reprise des mises à jour est constituée des opérations de mise à jour journalisées par l'agent lors de l'exécution de la sous-transaction abandonnée
- Localement, la transaction de reprise est une nouvelle transaction qui doit être sérialisée
- Globalement, la transaction de reprise fait partie de la sous-transaction qu'elle reprend
- L'exécution des transaction de reprise relâche la sérialisabilité globale avec la m-sérialisabilité
  - La m-sérialisabilité considère que les lectures faites par la transaction abandonné et les écritures faites par sa transaction de reprise sont une seule transaction
- Les ordonnancements globaux présentés peuvent être obtenus en présence de la m-sérialisabilité

Reprise totale (*retry*)

- La transaction de reprise totale est composée des opération de lecture et d'écriture de la sous-transaction à reprendre
- Cette stratégie est possible si les sous-transactions abandonnées sont ré jouables
- Ainsi, la validation de la transaction de reprise doit être certaine après un nombre borné d'essaies quelque soit l'état des données au moment de la reprise
- Les dépendances de valeur entre les sous-transactions sont interdites
- Les ordonnancements présentés plutôt peuvent être obtenus en présence de transactions de reprise totale

Compensation de transactions

- Une transaction, appelée transaction de compensation (TC) défait sémantiquement Les opérations d'écriture faites par une sous-transaction dont la transaction globale a été abandonnée
- Exemple : si une sous-transaction effectue la réservation d'une place d'avion, la TC doit annuler la réservation
- Comme les sous-transactions compensées peuvent être visibles par d'autres transactions, l'état du système n'est pas forcement le même que si la sous-transaction n'a eu lieu
- Cette technique se base sur la richesse sémantique des applications
- Propriétés des TC
  - Elles sont exécutées comme n'importe quelle autre transaction
  - Elles doivent forcement valider, ceci est appelé persistances de compensation

Modèles de transactions étendues

Transactions emboîtées fermées

Transactions emboîtées ouvertes

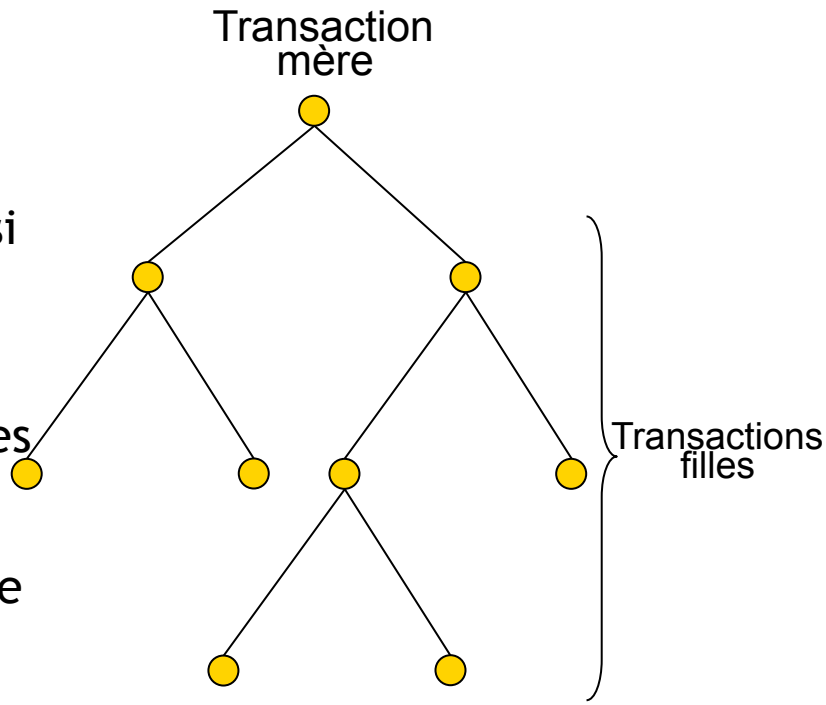
DOM

Sagas

Transactions flexibles

Transactions emboîtées fermées – TEF (1)

- Composée d'un ensemble de sous-transactions, lesquelles peuvent récursivement avoir d'autres sous-transactions
- Une transaction fille peut être démarrée uniquement après sa transaction mère
- La transaction mère ne peut valider que si toutes se filles ont terminé
- La validation de la transaction mère implique la validation de toutes les filles
- Si une transaction mère abandonne, toutes les filles doivent abandonner à leur tour
- Si une transaction fille abandonne, la transaction mère peut choisir une manière de récupération qui peut être la réexécution ou l'exécution d'une autre sous-transaction
- Les effets de la transaction mère ne sont pas aperçus par les autres transactions du système. Par contre, une (sous)transaction mère peut percevoir les effets de ses transactions filles



Transactions emboîtées fermées – TEF (2)

- Est-ce qu'elles assurent les propriétés ACID ?
  - Au niveau de la transaction mère on assure ACID :
    - Atomicité : oui car si elle abandon toutes les sous-transactions abandonnent et si elle valide toutes les sous-transactions valident
    - Cohérence : oui car elle est imposée
    - Isolation : oui car ses effets ne sont pas aperçus avant la validation
    - Durabilité : oui car une fois la transaction terminée elle persiste
  - Au niveau des sous-transactions on assure AI :
    - Atomicité : oui car chacune des sous-transaction doit valider
    - Cohérence : temporairement la cohérence peut être relâchée
    - Isolation : oui car ses effets ne peuvent pas être perçus avant la validation de la transaction mère
    - Durabilité : non car la durabilité d'une transaction fille dépend de celle de la transaction mère

Transactions emboîtées ouvertes (TEO)

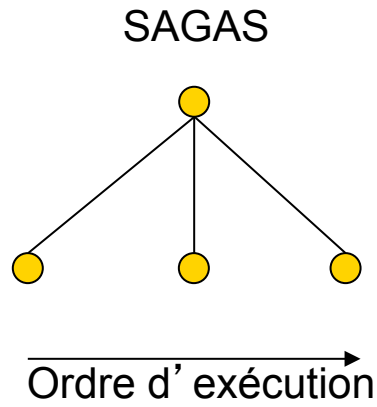
- Suivent les mêmes principes de structure des TEF
- Afin de permettre plus de concurrence, l'isolation est relâchée au niveau global. Les sous-transactions peuvent valider et rendre visibles ses effets avant la validation de la transaction racine
- Elles assurent les mêmes propriétés les TEF à l'exception de l'isolation au niveau global

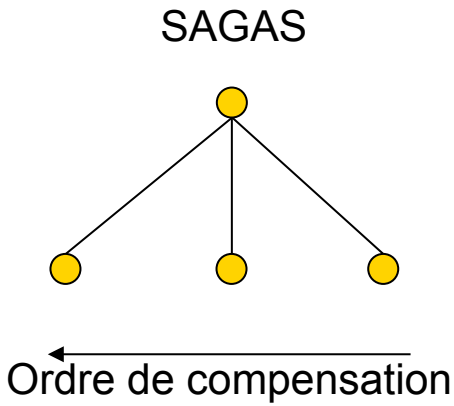
SAGAS

- Modèle conçu pour de transactions à longue durée
- Une SAGAS est une TEO à deux niveaux
- Les sous-transactions sont relativement indépendantes avec un ordre d'exécution prédéfini (séquentiel ou parallèle)
- Au fur et à mesure que les transactions valident, elles partagent leur résultats avant la validation de la SAGA
- L'exécution des SAGAS n'est pas contrainte à des ordonnancements sérialisables car uniquement les sous-transactions commutables sont considérées
- A chaque sous-transaction doit être associée une transaction de compensation qui est exécutée en cas d'abandon de la SAGA
- En cas d'abandon, les transactions de compensation sont exécutées dans l'ordre inverse de validation des sous-transactions correspondantes

SAGAS

- Est-ce qu'elles assurent les propriétés ACID ?
  - Au niveau de la SAGA on assure D :
    - L'atomicité dite sémantique est assurée à cause des transactions de compensation
    - Cohérence : elle n'est pas imposée
    - Isolation : non car les sous-transactions rendent visibles leurs résultats après leur validation (dite locale)
    - Durabilité : oui
  - Au niveau des sous-transactions AID :
    - Atomicité : oui, car chacune des sous-transaction doit valider
    - Cohérence : elle n'est pas imposée lors de l'exécution partielle d'une SAGAS
    - Isolation : oui car leurs résultats partiels ne sont pas visibles avant la validation locale
    - Durabilité : oui car on considère qu'une fois validées elles persistent. Les transactions de compensation sont considérées comme de sous-transactions différentes



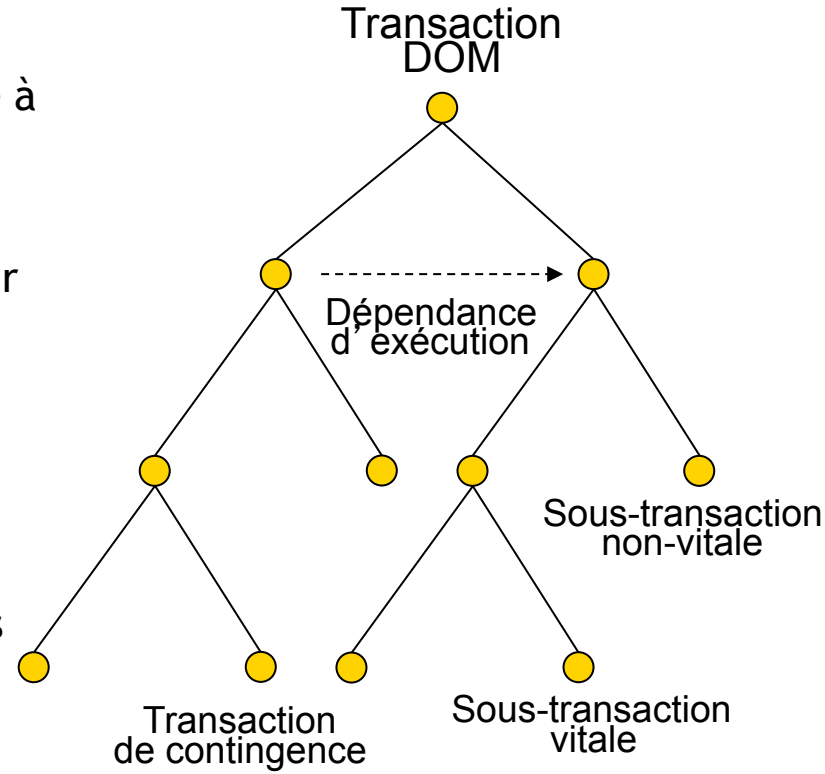


Transactions DOM (*Distributed Object Management*)

- Modèle conçu pour de transactions à longue durée dans les multibases de données
- Une DOM est composée d'un ensemble de sous-transactions qui peuvent être de TEO
- Les sous-transactions peuvent être vitales ou non-vitales
  - L'abandon d'une transaction vitale entraîne l'abandon de la transaction DOM, ce qui n'est pas le cas lors de l'abandon des non-vitales
- De dépendances de précédence peuvent être définies entre les sous-transactions
- Les transactions peuvent être exécutées et validées de manière indépendante sauf si des dépendances de précédence sont spécifiées
- A chaque sous-transaction doit être associée une transaction de compensation comme dans les SAGAS
- Il est possible d'associer de transactions de contingence aux sous-transactions
  - Les transactions de contingence sont exécutées lors qu'une condition de défaillance de la transaction correspondante survient
  - La condition de défaillance peut être une annulation ou un échec sémantique de la sous-transaction

Transactions DOM (*Distributed Object Management*)

- Est-ce qu'elles assurent les propriétés ACID ?
  - Au niveau de la DOM on assure D :
    - L'atomicité dite sémantique est assurée à cause des transactions de compensation
    - Cohérence : elle n'est pas imposée
    - Isolation : non car les sous-transactions rendent visibles leurs résultats après leur validation (dite locale)
    - Durabilité : oui
  - Au niveau des sous-transactions AID :
    - Atomicité : oui car chacune des sous-transaction doit valider
    - Cohérence : elle n'est pas imposée
    - Isolation : oui car leurs résultats partiels ne sont pas visibles avant la validation locale
    - Durabilité : oui car on considère qu'une fois validées elles persistent. Les transactions de compensation et de contingence sont considérées comme de sous-transactions différentes



Transactions Flexibles

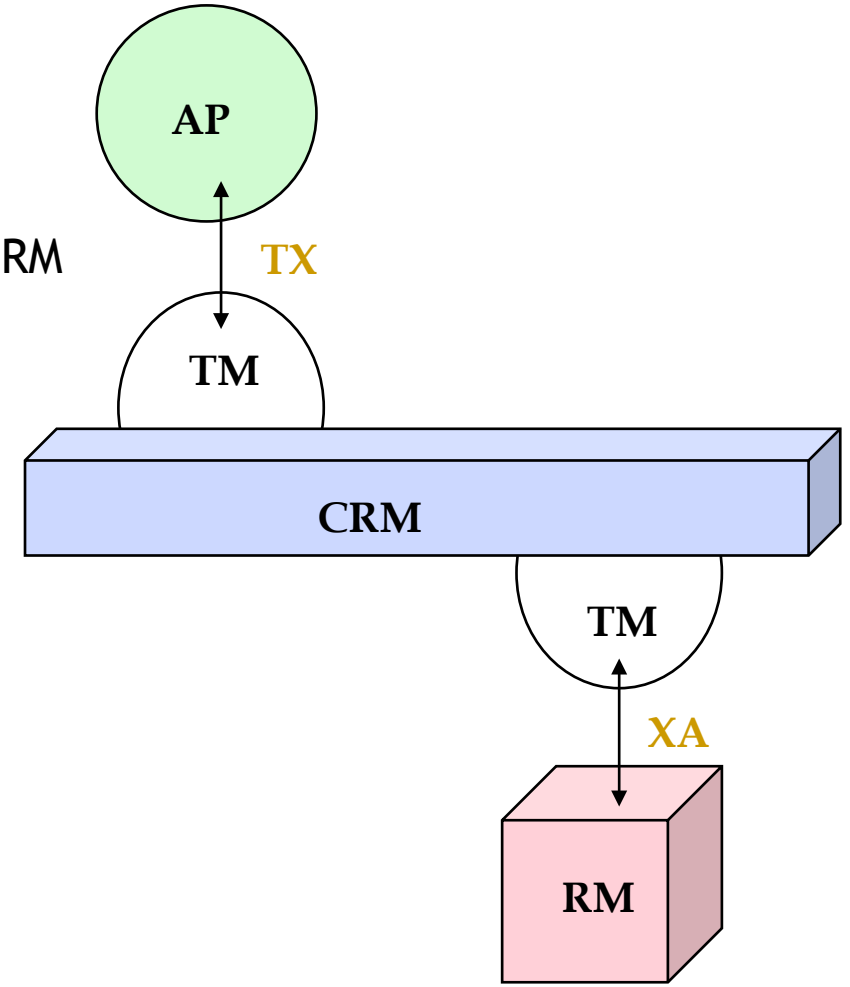
- Modèle conçu pour les transactions multibases de données
- Une transaction Flexible peut être composée d'un ensemble de sous-transactions fonctionnellement équivalentes où la terminaison de chacune représente l'accomplissement de la transaction Flexible
- De dépendances d'exécution entre les sous-transactions peuvent être spécifiées
  - Dépendances : de défaillance, de succès, d'initiation, de temps, etc.
- Les sous-transactions peuvent valider indépendamment de la transaction Flexible. Elles ont les mêmes propriétés que les sous-transactions DOM
- Les sous-transactions peuvent avoir de transactions de compensation
- Les transactions Flexibles
  - relâchent l'atomicité et proposent la semi-atomicité : uniquement un sous ensemble des sous-transactions est validé
  - relâchent l'isolation et proposent la F-serialisabilité : relâche la sérialisabilité pour prendre en compte la compensation et la ré exécution lors de la récupération après pannes

Moniteurs transactionnels

- Support de transactions ACID
- Accès continu aux données
- Reprise rapide du système en cas de panne
- Sécurité d'accès
- Performances optimisées
  - Partage de connexions
  - Réutilisation de transactions
- Partage de charge
  - Distribution de transactions
- Support de bases hétérogènes
- Respect des normes et standards

Moniteur transactionnel : Modèle

- Modèle DTP de X/OPEN
  - Programme d'application AP
  - Gestionnaire de transactions TM
  - Gestionnaire de communication CRM
  - Gestionnaire de ressources RM
- Interfaces standards
  - TX = interface du TM
  - XA = interface du RM
  - intégration de TP
- Types de RM
  - gestionnaire de fichiers
  - SGBD
  - périphérique



Interface applicative TX

- tx_open
  - ordonne au TM d'initialiser la communication avec tous les RM dont les librairies d'accès ont été liées à l'application.
- tx_begin
  - ordonne au TM de demander aux RM de débuter une transaction.
- tx_commit ou tx_rollback
  - ordonne au TM de coordonner soit la validation soit l'abandon de la transaction sur tous les RM impliqués.
- tx_set_transaction_timeout
  - positionne un " timeout " sur les transactions
- tx_info
  - permet d'obtenir des informations sur le statut de la transaction.

Interface ressource XA

- xa_open
  - ouvre un contexte pour l'application.
- xa_start
  - débute une transaction.
- xa_end
  - indique au RM qu'il n'y aura plus de requêtes pour le compte de la transaction courante.
- xa_prepare
  - lance l'étape de préparation du commit à deux phases.
- xa_commit
  - valide la transaction.
- xa_rollback
  - abandonne la transaction.

Principaux moniteurs (1)

- CICS de IBM (Customer Information Control System)
  - construit sur Encina (et DCE)
  - reprise de l'existant CICS (API et outils)
  - conformité DTP : Xa, CPI-C
- Encina de Transarc
  - issu de CMU (1992), racheté par IBM
  - construit sur DCE (OSF) pour la portabilité et la sécurité
  - transactions imbriquées
  - conformité DTP : Xa, CPI-C, TxRPC
  - Discontinué

Principaux moniteurs (2)

- Tuxedo (Transactions for Unix, Extended for Distributed Operations), produit Oracle depuis 2008.
  - Défini par AT&T, à la base de DTP
  - supporte l'asynchronisme, les priorités et le routage dépendant des données
  - conformité DTP: Xa, Tx, XaTMI, CPI-C, TxRPC
- Top End de NCR
  - produit stratégique d'AT&T
  - respecte le modèle des composants DTP (AP, RM, TM, CRM)
  - haute disponibilité
  - conformité DTP: Xa, Xa+, Xap-Tp, Tx
- Autres : UTM de Siemens, Unikix

MTS de Microsoft

- Microsoft Transaction Server
- Intégré à DCOM
- Partage de grappes de NT (cluster)
- Les disques sont supposés partagés
- Allocation des ressources en pool aux requêtes :
  - pool de connexion aux ressources (SQL Server)
  - pool de transactions (support)
  - pool de machines
- Ne suit pas les standards !

Object Transaction Service (OTS) de l'OMG

- Service transactionnel de CORBA offrant
  - transactions imbriquées
  - interopérabilité avec des transactions non-CORBA conformes au standard DTP
  - transactions multi-ORB

Principaux OTM

- OrbixOTM d'Iona
  - basé sur Encina (Transarc-IBM)
- M3 de BEA
  - basé sur Tuxedo
- Integrated Transaction Service d'Inprise
  - implém. de JTS (Java Transaction Service = spéc. Java de OTS)
- Open CICS et TXSeries d'IBM
  - basé sur Encina

Conclusion

- L'exécution correcte de transactions est un élément vital dans les SGBD répartis
- Les solutions transactionnelles ont été testées, prouvées et adoptées par les industriels (par ex 2PC, 2PL)
- Cependant, l'architecture des SGBD évolue (BD mobiles, BD senseurs, BD P2P, etc.)
- Donc on aura toujours besoin d'adapter ou de proposer de nouveaux protocoles et modèles transactionnels