Master d'Informatique

BDR - 4I803 - Cours 7

Transactions réparties
2016

1

Transactions réparties

Gestion de transactions

Transactions dans un système réparti

Protocoles de verrouillage

Concepts des moniteurs transactionnels

Exemples

2

Concept de transaction

Une transaction est une collection d'actions qui transforment la BD (ou des fichiers) depuis un état cohérent en un autre état cohérent

– BD cohérente (garantie par le système)

– transaction cohérente (garantie par le programmeur)

3

Exemple de transaction

Réduire la cde n°10 de 5 unités et les reporter à la cde n°12

```
Transaction   Report-qté
begin
    exec sql      UPDATE   Cde
                  SET      qte = qté - 5
                  WHERE    ncde = 10;

    exec sql      UPDATE   Cde
                  SET      qté = qté + 5
                  WHERE    ncde = 12;

    exec sql      COMMIT WORK;
end.
```

4

Problèmes liés à la répartition

- Atomicité :
  - Les mises à jour sont effectuées sur TOUS les sites, ou sur AUCUN => nécessité de coordonner.
- Isolation :
  - Items locaux (physique) et items globaux (logique).
    Réplication = plusieurs items physique pour une item logique
  - Transaction globale et transaction locale : le système doit garantir la sérialisabilité globale.
    - Verrous globaux
    - Graphe de précédence réparti
    - Estampillage global
  - Verrouillage
    - centralisé vs décentralisé
    - verrous logiques et verrous physiques
    - interblocage intersites (difficile à détecter, prévention)

5

Rappel : sérialisabilité et graphe de précédence

Données : X, Y, Z, T Transactions T_1 à T_5

- L4(Z) L5(X) L3(Y) L4(Y) E5(X) E3(Y) L4(T) L3(Y) L3(Z) L4(Z) E4(T) **L5(T)** E1(T) **E1(X)** L2(Y) **L4(Z)** **L3(X)** **E2(T)**
- On suppose que le **commit** est juste après la dernière opération
- Représenter les précédences sous forme de graphe
  - Précédence : conflit entre deux transactions, la première doit forcément passer avant la deuxième dans l'ordre séquentiel équivalent
- L'exécution est sérialisable ssi son graphe de précédence est sans-circuit
  - Détection de circuit et ordre séquentiel équivalent : tri topologique

6

Sérialisabilité globale vs locale

- Deux sites S1 et S2
  - S1[x]: L1(x), E1(x), L2(x), E2(x)
  - S2 [y]: L2(y), E2(y), L1(y), E1(y)

- Cet exemple simple montre qu'un contrôle local n'est pas suffisant pour garantir la sérialisabilité globale
- Problème similaire pour garantir la réplication synchrone (remplacer y par x. ordres compatibles)
- Problème similaire aussi pour la détection d'interblocage

7

Contrôle de concurrence optimiste

- On laisse les transactions s'exécuter et on vérifie à la validation qu'il n'y a pas de défaut de sérialisation (efficace si peu de conflit)
- En réparti, il faut s'échanger les précédences pour tester, peut être couteux

8

Estampillage

Estampilles globales

même principe qu'en centralisé. Les transactions s'exécutent sur n'importe quel site, et laissent une estampille pour la copie en question. En cas d'écriture, la transaction écrit sur tous les sites où se trouvent des copies.

Rappel :

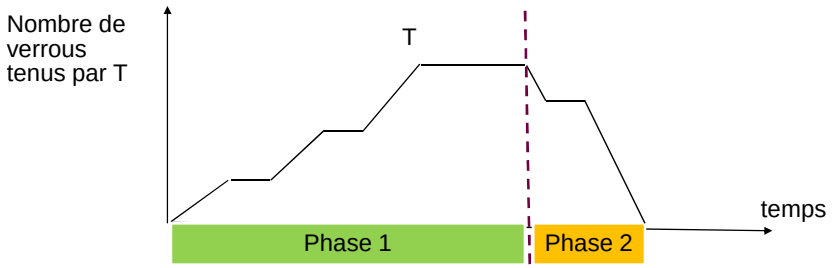
une transaction peut lire un objet ssi l'estampille d'écriture de l'objet est plus petite, sinon abandon.

une transaction peut écrire un objet ssi les deux estampilles de l'objet sont plus petite. Si *seule* l'estampille de lecture est plus petite, l'écriture est refusée mais pas besoin d'abandonner (règle de Thomas), sinon abandon. Ex : R1(A), W2(A), W1(A)

Pb. Étendre la notion d'estampille au réparti, harmoniser les horloges (voir cours de systèmes répartis).

9

Isolation par **verrouillage** 2 phases (V2P)



- Les verrous en lecture sont partageables; ceux en écriture sont **exclusifs**
- Règle 1
  - avant d'accéder à un granule x, une transaction doit acquérir un verrou sur x. Si x est déjà verrouillé de façon exclusive, la transaction attend
- Règle 2
  - dès qu'une transaction relâche un verrou, elle **ne peut plus** acquérir de nouveau verrou

10

Performances

L'objectif est de réduire:

- les blocages : L1(x), E2(x),….
  - une transaction attend qu'une autre transaction relâche ses verrous
- les abandons anormaux : E1(x), L2(x), E2(x), abort1
  - une transaction à lu des données « sales »

  (correction : ne peut valider avant celles qui a écrit)
- les inter-blocages : L1(x), L2(y), E1(y), E2(x), …
  - un ensemble de transactions attend que l'une d'entre elles relâche ses verrous : « circuit d'attente »

11

Degrés d'isolation standardisés (SQL2)

- Degré 0 : Read Uncommitted
  - Une transaction de degré 0 est sans perte de mise à jour
  - Interdiction de faire des écritures
  - Possibilité de lecture sale, non reproductible, fantôme
- Degré 1 : Read Committed
  - Une transaction de degré 1 satisfait le degré 0
  - Elle ne fait pas de lecture sale
- Degré 2 : Repeatable Read
  - Une transaction de degré 2 satisfait le degré 1
  - Elle ne fait pas de lecture non reproductible
- Degré 3 : Serializable
  - Satisfait le degré 2
  - Pas de lecture fantôme
- Commande SQL : *set transaction isolation level*

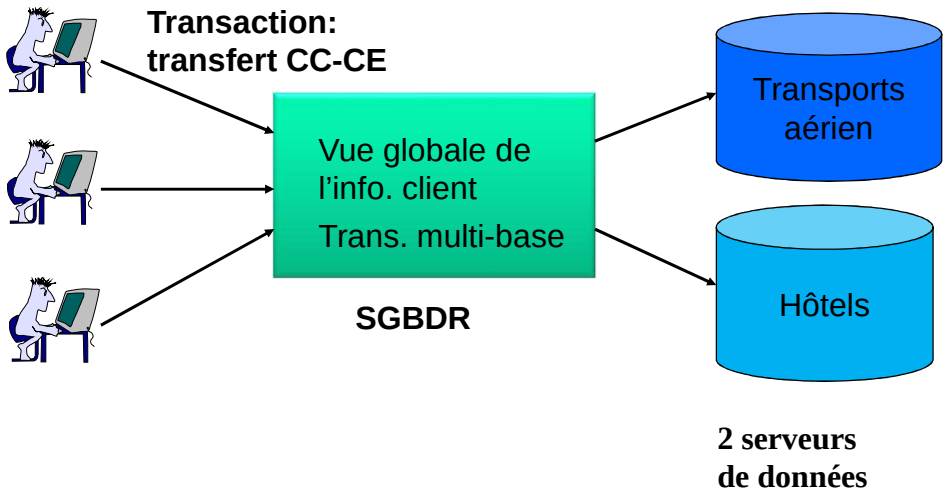
12

Degrés d'isolation et verrouillage

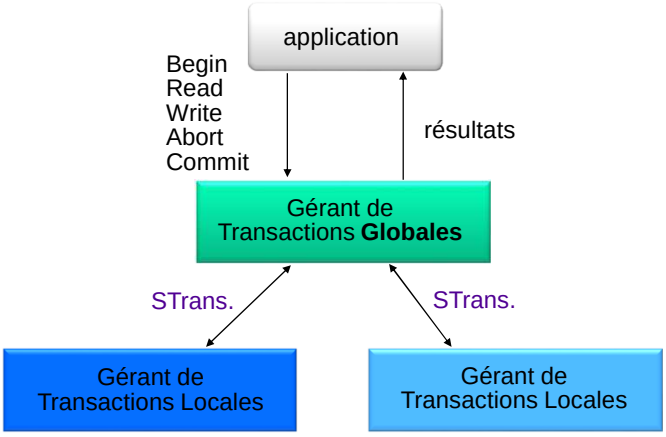
- Chaos
  - Exclusion mutuelle pour l'écriture
  - Pas de verrous en lecture
  - Inexistant en SQL2
- Read Uncommited
  - Deux phases pour l'écriture
  - Pas de verrous en lecture
  - Appelé aussi *Browse Access*
- Read Committed
  - Deux phases (verrou long) pour l'écriture
  - Exclusion mutuelle (verrou court) pour la lecture
  - Appelé aussi *Cursor Stability*
- Repeatable Read
  - Deux phases pour l'écriture
  - Deux phases pour la lecture

13

Exemple de transaction répartie



Gestion de transactions réparties



15

Protocole de validation en deux étapes
2 Phases Commit (2PC)

- Objectif : Exécuter COMMIT pour une transaction répartie
- Etape1
  - Préparer à écrire les résultats des mises à jour dans la BD
- Etape 2
  - Ecrire ces résultats dans la BD
- Coordinateur
  - composant d'un site qui applique le protocole
    - lance la transaction, la découpe en sous-transactions à exécuter sur les différents sites, coordonne la terminaison de la transaction
- Participant
  - composant d'un autre site qui participe dans l'exécution de la transaction

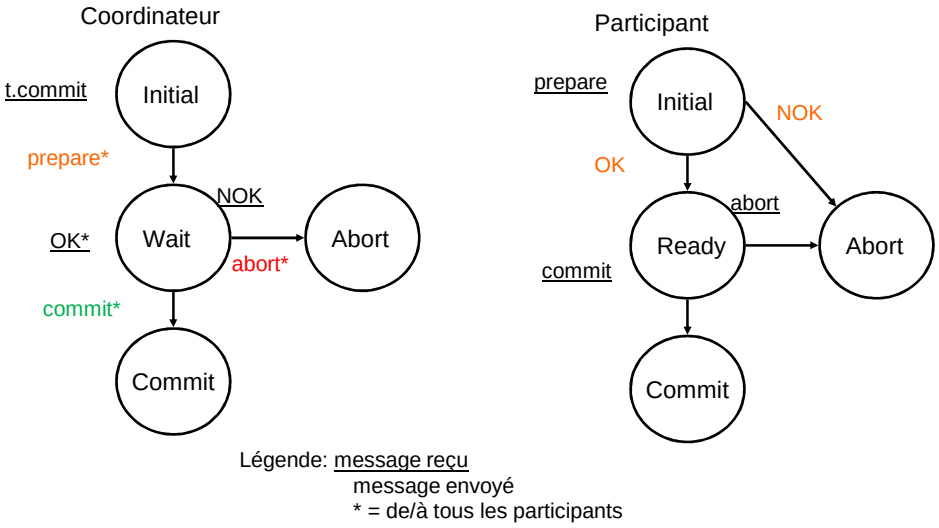
16

Protocole de validation en deux étapes
2 Phases Commit (2PC)

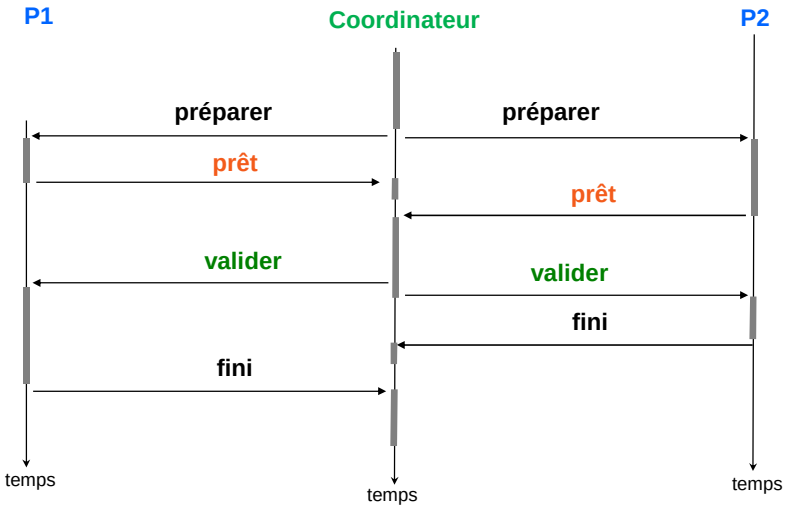
- Le coordinateur envoie « prepare » à tous les participants
- Participants :
  - Force un abandon et envoie « non », ou bien
  - Prepare le log pour écriture et envoie « oui »
- Coordinateur :
  - Reçoit au moins un « non » ou ne reçoit pas au moins un message (time out) : force un abandon et envoie « abandon »
  - Reçoit tous les « oui » : Écrit son log et envoie « commit » aux participants
- Participant :
  - Recoit « abort »: force un abandon
  - Reçoit « commit » : écrit son log et valide
  - Dans les deux cas, « ack » au coord /* pas besoin pour abort
- Coordinateur :
  - Reçoit tous les « ack » : écrit « fin » dans son log

17

Protocole 2PC : messages échangés

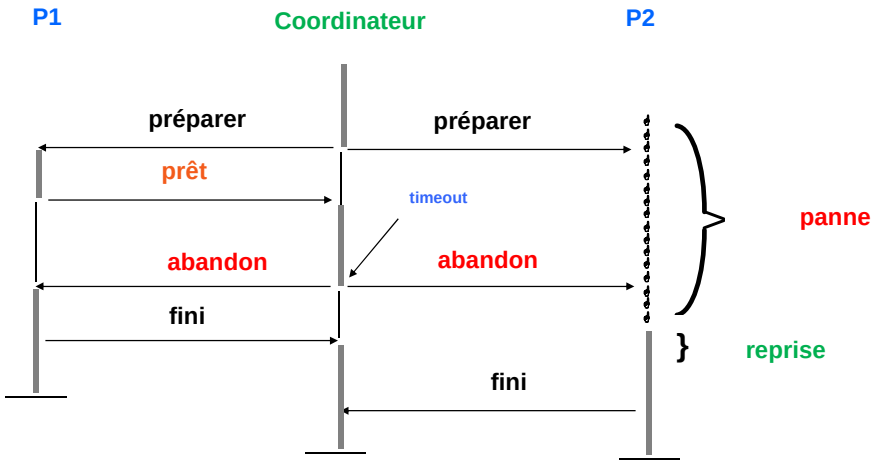


Validation normale : chronogramme



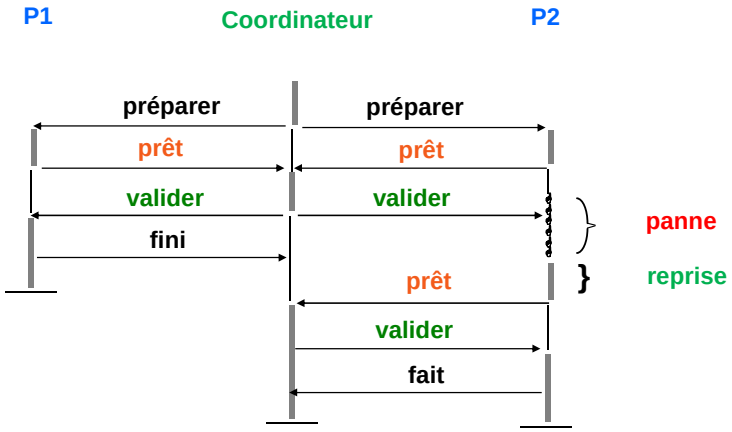
19

Panne d'un participant avant d'être prêt



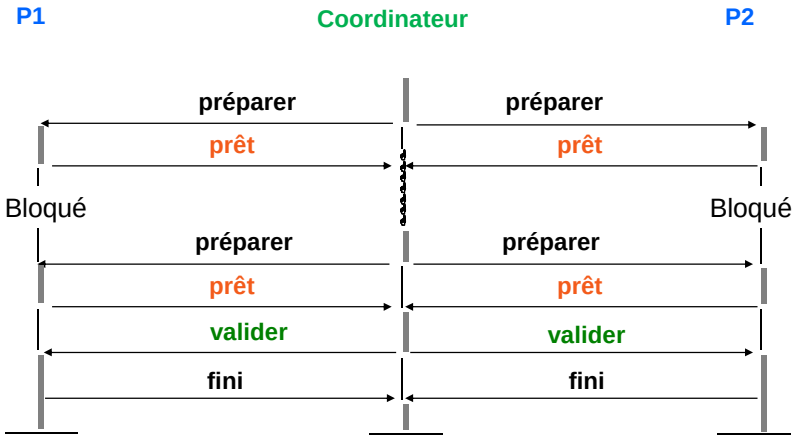
20

Panne d'un participant après s'être déclaré prêt



21

Panne du coordinateur

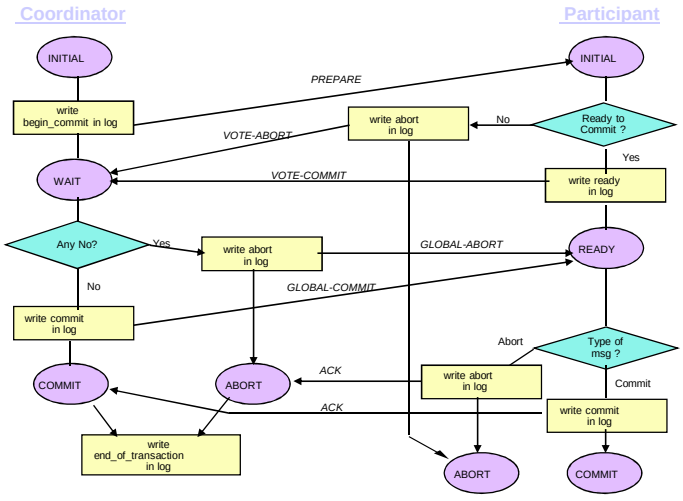


22

Points clé du protocole 2PC

- Lorsque le participant envoie OK au coordinateur
  - Le participant s'engage à valider
  - Le participant valide ssi le coordinateur le lui ordonne
  - Le coordinateur peut demander au participant d'abandonner
- Lorsque le participant envoie NOK au coordinateur
  - Le participant peut abandonner unilatéralement
- Le coordinateur décide de valider ssi tous les participants sont OK
- Chaque site doit maintenir un journal des messages échangés
  - reprise après panne

Actions du protocole



24

2PC bloquant

- Il existe des cas où le protocole ne termine pas
- Ces cas sont assez rares :
  - Panne simultanée d'un participant et du coordinateur
- 3PC non bloquant, mais recouvrement plus restreint et plus coûteux

25

Protocoles de verrouillages répartis

Traduire le verrouillage logique en verrouillages physiques.

1. Pas de réplication :
   - gestionnaire de verrous local sur chaque site
     + facile à implémenter
     + peu de messages (2 pour verrouiller, 1 pour libérer)
     - gestion des interblocages complexe
2. Avec réplication :
   - plusieurs méthodes, dont le coût (nb de messages) dépend du nombre de copies, du rapport lectures/écritures, de la concurrence (verrous refusés)

26

Méthode du nœud central

Un seul gestionnaire de verrous, sur un seul site.

Table de verrous logiques.

Si un verrou est accordé, la transaction peut lire la donnée sur n'importe quel site comportant une copie. Pour l'écriture, tous les sites sont impliqués.

+ peu de messages (3)
+ Implémentation simple
+ gestion des interblocages simple (algos classiques)
- goulot d'étranglement
- vulnérable : blocage si le site du gestionnaire de verrous est en panne.

27

Copie primaire

Amélioration de la méthode du nœud central :

Le gestionnaire est réparti sur plusieurs sites.

Le verrouillage d'un item logique est sous la responsabilité d'un seul site, le site primaire.

+ suppression du goulot d'étranglement
+ peu de messages (3 en général)
+ simple à implémenter
- complique la gestion des interblocages
- dictionnaire des copies primaires

28

Majorité

Un gestionnaire de verrous sur chaque site, qui gère les verrouillages (physique) des objets se trouvant sur le site.

Pour verrouiller (logique) un item X dupliqué sur n sites, la transaction envoie une demande de verrouillage aux sites. Chaque gestionnaire local détermine s'il accorde ou refuse le verrou.

Soit Ne (resp. Nl) le nombre de verrous en écriture (resp. lecture) physique à obtenir pour un verrou logique, on a les contraintes suivantes :

- Ne > (n+1)/2 (conflit écriture/écriture)
- Nl+Ne > n (conflit lecture/écriture)

+ pas de contrôle central

- beaucoup de messages (3 (n+1)/2)

- gestion de interblocages complexe, les interblocages peuvent même arriver sur une seule donnée!

Détection des interblocages

1. Timeout

   Pb : bien déterminer le délai

   + pas de messages

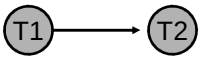
   - risque d'annulation de plusieurs transactions au lieu d'une seule

2. Graphe d'attente

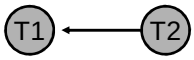
   Les nœuds sont les transactions.

   Arc de Ti →Tj si Ti attend un verrou tenu par Tj.

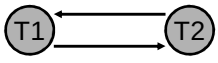
On construit des graphes locaux, mais les cycles doivent être détectés sur le graphe global (**union** des graphes locaux). Plusieurs algos pour minimiser nb et taille des msgs



Site 1



Site 2



Graphe global

Résolution d'un interblocage par graphe d'attente local / global

- Sur chaque site Si : graphe d'attente local Gi
  - Gi: Les nœuds sont les transactions. Arc de Ti →Tj si Ti attend un verrou tenu par Tj.
  - Distinguer: les transactions locales (sur un seul site), les transactions réparties (sur au moins 2 sites)
  - Gi' : sous graphe local composé seulement des transactions réparties et des *chemins* qui les relient.
- Graphe global : L'union des sous graphes locaux suffit pour détecter un circuit.
- Résolution de l'interblocage en supprimant un arc du circuit. Parfois possible d'abandonner une transaction locale (non répartie).
- Ex: Sur S1: T1→T2→T3 Sur S2: T3→T4→T1
  - Seules T1 et T3 sont réparties, T2 et T4 sont locales
  - Possibilité d'abandonner T2 pour résoudre l'interblocage

31

Détection des interblocages : Algorithme de Chandy-Misra-Haas

- Lorsque T1 demande un verrou détenu par T2
  - message *probe* envoyé à T2
- Si T2 est en attente
  - propager le message à toutes les transactions que T2 attend
- Si le message revient à T1, alors interblocage détecté
- Référence
  - A survey of distributed deadlock detection algorithms
    - A. K. Elmagarmid. Sigmod Records 15(3), 1986

Evitement des interblocages

- Soit Ti et Tj, avec les estampilles i < j
  - signifie que Ti est plus ancienne que Tj
- Principe: rendre l'interblocage impossible en imposant un ordre total sur les attentes: graphe d'attente sans circuit.
- Assurer qu'une transaction "plus ancienne" parvient à valider. Ne jamais forcer l'abandon d'une transaction "plus ancienne" à cause d'une "plus jeune".

• Solution 1 : attente dans l'ordre croissant des estampilles
  - Ti demande un verrou possédé par Tj : Ti attend
  - Tj - - - - - - Ti : Tj abandonne

•Solution 2 : attente dans l'ordre décroissant des estampilles
  - Ti demande un verrou possédé par Tj : **Tj** abandonne (préemption)
  - Tj - - - - - - Ti : Tj attend

Attribution répartie des estampilles i et j: harmoniser les horloges
• Recommencer une transaction abandonnée
  – converser l'estampille initiale

33

Moniteur transactionnel

Extension du concept de transaction à d'autres contextes que les BD

- Support des transactions ACID (à un ens. d'accès à des objets)
- Accès continu aux données
- Reprise rapide du système en cas de panne
- Sécurité d'accès
- Performances optimisées
  - partage des connexions
  - réutilisation de transactions
- Partage de charge
  - distribution de transactions sur les serveurs
- Support de fichiers et bases de données hétérogènes
- API standardisées

34

Fonctions transactionnelles

- Etapes du traitement transactionnel
  - traduire la requête utilisateur en format interne au moniteur
  - déterminer le type de transaction et le programme nécessaire à son exécution
  - débuter la transaction et lancer son exécution
  - valider, ou abandonner, la transaction (validation 2Phases)
  - retourner le résultat à l'émetteur de la requête

35

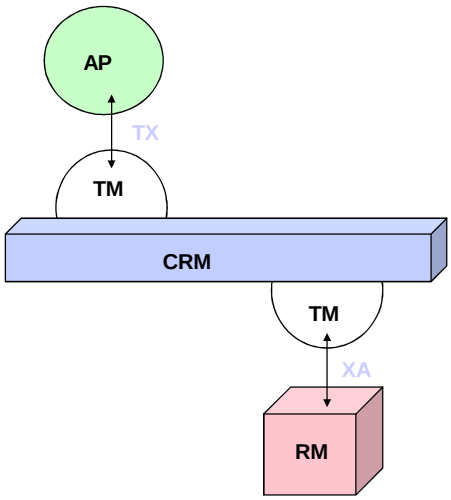
Fonctions systèmes

- Un moniteur assure aussi :
  - routage de la requête
  - équilibrage de charge entre requêtes et serveurs
  - résistance aux pannes (client, serveur)
  - gestion de configuration (processus, sessions, ...)
  - contrôle de performances et réglage
  - sécurité par certification des requêtes
  - gestion des communications et des services

36

Le modèle DTP de l'OpenGroup

- Composants
  - Programme d'application AP
  - Gérant de transactions TM
  - Gérant de communications CRM
  - Gérant de ressources RM
- Interfaces standards
  - TX = interface AP-TM
  - XA = interface TM- RM
  - intégration de TP
- Types de RM
  - gestionnaire de fichiers
  - SGBD
  - périphérique



37

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 une transaction

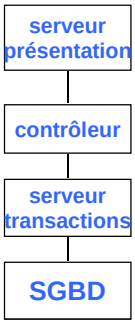
38

Interface ressource XA

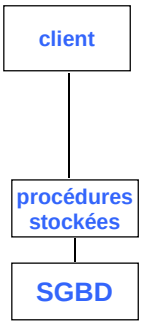
- **xa_open**
  - ouvre un contexte pour l'application
- **xa_start**
  - débute une transaction
- **xa_end**
  - indique au RM qu'il n'y aura plus de requêtes pour le compte de la transaction courante
- **xa_prepare**
  - lance l'étape de préparation du commit à deux phases
- **xa_commit**
  - valide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

39

SGBD vs Moniteur : configurations



Moniteur transactionnel



Serveur bases de données

40

SGBD vs Moniteur : propriétés

- Avantages moniteur :
  - routage dans des applications de très grande taille
  - langages et environnement plus riches
  - serveurs bases de données répartis hétérogènes
  - communications client-serveur
  - environnement de gestion système
- Avantages SGBD :
  - simplicité et prix
  - performances pour petites et moyennes applications

41

Conclusions

- Applications transactionnelles
  - OLTP: Online Transaction Processing
  - Solutions très performantes
    - benchmark TPC-C, 10^5 transactions par minute
  - Middleware transactionnel
    - préserve l'autonomie des BD et des applications
- Déploiement à large échelle
  - Dictionnaire réparti en mémoire (ex. Oracle Coherence)
  - Contrôle de concurrence réparti
- Perspectives
  - Transactions réparties pour des données semi-structurées
  - Contrôle flexible du niveau d'isolation :
    - Transactions dans les systèmes NewSQL, compromis cohérence/performance

42