Bases de Données

Cyril Labbé

LIG, Université de Grenoble, France
first.last@imag.fr

Table of Contents

Verrouillages

Verrous

Le contrôle de concurrence est assuré par la mise en place de verrous qui sont susceptibles de bloquer l'accès aux données pour les autres transactions. Il y a 2 types de verrous :

- Verrous partagés : verrous en lecture. Une transaction qui dispose d'un verrou partagé sur l'object x peut lire mais pas écrire x. Les autres transactions peuvent aussi lire mais pas écrire
- Verrous exclusifs : verrou en écriture. Une transaction qui dispose d'un verrou exclusif sur l'object x peut écrire x. Aucune autre transaction ne peut accéder à x.

Demande de verrou

- Toute transaction demande un verrou ($VE(x)$ $VL(x)$) avant d'accéder à x.
- Si la données est déjà verrouillée par un verrou non compatible, la transaction doit attendre que le verrou se libère.
- La transaction conserve le verrou jusqu'à libération explicite $Lib(x)$ ou lorsqu'elle se termine (commit, abort)

Garantie de cohérence

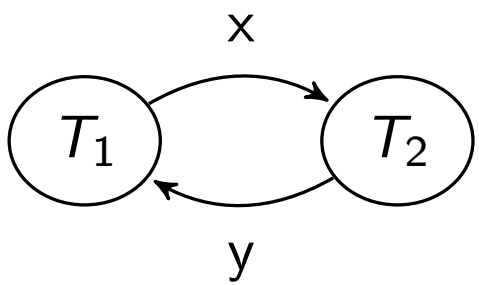
Le mécanisme de verrouillage n'est pas suffisant pour garantir la cohérence.

Verrouillage et ordonnancement non cohérent

Ordonnancement avec verrouillage

T_1	T_2
$VE(x)$	.
$E(x)$	.
$Lib(x)$	.
.	$VE(x)$
.	$E(x)$
.	$Lib(x)$
.	$VE(y)$
.	$E(y)$
.	$Lib(y)$
.	$Validation$
$VE(y)$	.
$E(y)$	.
$Lib(y)$	.
$Validation$	.

Graphe de dépendance



Pas d'équivalent séquentiel

L'ordonnancement n'est équivalent ni à T_1puis T_2, ni à T_2puis T_1.

Protocole 2*PL*

Pour garantir un ordonnancement sérialisable, il faut utiliser le protocole de verrouillage à 2 phases (Two phase locking 2*PL*).

2*PL*

Une transaction suit le protocole de verrouillage à deux phases si :

- toutes les opérations de verrouillage précédent le premier déverrouillage.

2*PL*

Une transaction qui libère un verrou ne peut plus en acquérir d'autres.

Théorème

Si dans un ordonnancement S toutes les transactions suivent le 2*PL* alors S est équivalent à un ordonnancement séquentiel.

Exemples 2*PL*

Ordonnancement sans 2*PL*

T_1	T_2
$VE(x)$	.
$E(x)$	.
$Lib(x)$	.
.	$VE(x)$
.	$E(x)$
.	$Lib(x)$
.	$VE(y)$
.	$E(y)$
.	$Lib(y)$
.	$Validation$
$VE(y)$	.
$E(y)$	.
$Lib(y)$	.
$Validation$	.

Ordonnancement avec 2*PL*

T_1	T_2
$VE(x)$	.
$E(x)$	.
~~$Lib(x)$~~	.
.	$VE(x)$
$VE(y)$	.
$E(y)$	.
$Lib(x)$	.
.	$E(x)$
.	$VE(y)$
Lib(y)	.
.	E(y)
$Validation$	.
.	$Lib(y)$
.	$Validation$

Protocole 2*PL* et inter-blocages

2*PL*

Le protocole 2*PL* assure la sérialisabilité mais peut engendrer des situations d'inter-blocage (deadlock).

2*PL* / Inter-blocage / deadlock

T_1	T_2
$VE(x)$	.
.	$VE(y)$
$E(x)$	.
.	$E(y)$
$VE(y)$	.
.	$VE(x)$
$Attente - y$	.
.	$Attente - x$
.	.
.	.

Détection des inter-blocages

Détection des inter-blocages

La détection des inter-blocages peut être réalisés par la construction d'un graphe d'attente.

Graphe d'attente

Un graphe d'attente comprend :

- un nœud par transaction (Nœud T_i pour la transaction T_i)
- un arc de T_i vers T_j si la transaction T_i attend pour verrouiller un élément déjà verrouillé par T_j.

Inter-blocage

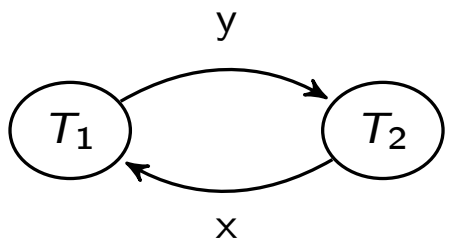
La présence d'un cycle dans le graphe signale une situation d'inter-blocage.

Graphe d'attente : exemple

2*PL* / Inter-blocage / deadlock

T_1	T_2
$VE(x)$	.
.	$VE(y)$
$E(x)$	.
.	$E(y)$
$VE(y)$	.
.	$VE(x)$
$Attente - y$	.
.	$Attente - x$
.	.
.	.

Graphe d'attente



Inter-blocage

- T_1 attend que T_2 libère le verrou sur y
- T_2 attend que T_1 libère le verrou sur x

Politique d'annulation

Si un cycle est détecté

Le SGBD doit annuler une ou plusieurs transactions.

Politique d'annulation

Plusieurs politiques permettent de choisir une "victimes" :

- la transaction la plus jeune (il faut mieux annuler une transaction qui commence)
- celle qui a fait le moins de modifications (annuler une transaction qui à fait peut de mise à jour). *i*.e. le coût de l'annulation n'est pas trop important.
- une transaction qui a encore beaucoup de travail. *i*.e. en fonction du nombre de données restant à mettre à jour. *P*roblème, le SGBD ne le sait pas forcement.

Famines

Pour éviter les "famines". Le SGBD peut compter le nombre de $\times$ qu'une transaction est annulée.

Estampillage

Pour faire du contrôle de concurrence à l'aide d'estampilles, on utilise deux sortes d'estampilles.

Estampilles de transaction

On associe à chaque transaction T un identificateur (l'estampille) $e(T)$ unique créé par le SGBD. Cet identificateur indique l'heure de démarrage d'une transaction. C'est un compteur logique qui est incrémenté à chaque début de transaction.

Estampilles sur les données

Chaque donnée x possède :

- une estampille de lecture $el(x)$ égale à l'estampille de la dernière transaction qui a lu la donnée.
- une estampille d'écriture $ee(x)$ égale à l'estampille de la dernière transaction qui a écrit la donnée.

Mise à jour des estampilles

Si T effectue une lecture sur x

if $(e(T) < ee(x))$ **then** //T plus *jeune* que la dernière Tr ayant écrit x
  $Annuler\ T$;
  $relancerT$;
else //T plus *vielle* que la dernière Tr ayant écrit x
  $L_T(x)$;
  $el(x) \leftarrow max(e(T), el(x))$;

Si T effectue une écriture sur x

if $(e(T) < el(x)) \vee (e(T) < ee(x))$ **then** //T plus *jeune* que la dernière Tr ayant écrit/lu x
  $Annuler\ T$
  $relancer\ T$;
else //T plus *vielle* que la dernière Tr ayant écrit/lu x
  $E_T(x)$; $ee(x) \leftarrow e(T)$;

=> Garanti un ordonnancement sérialisable.

Granularité des verrous

Granularité

Choisir le bon niveau de verrouillage. On peut verrouiller tuple par tuple ou un ensemble de tuples. Le verrouillage au niveau tuples peut entraîner le problème des "lectures fantômes".

exemple

Lectures d'objets (tuples) qui ne devraient pas exister :

		T_1	T_2
t_1		$S_1 \leftarrow$ Lire(Sells,price>2)	
t_2			
t_3			Validation
t_4		$S_2 \leftarrow$ Lire(Sells,price>2)	
t_5		$S \leftarrow S1 \cap S_2$	
t_6		Si (S=$\emptyset$) alors Validation sinon Annulation	

Eviter les lectures fantômes

Verrous sur prédicat et/ou des tables.

Contrôle de concurrence optimiste

Un protocole de concurrence optimiste comporte en générale 2 phases (T en lecture seule) ou 3 phases (T en écriture).

- phase de lecture (lecture et mise à jour sur copies locales)
- phase de validation (On teste les conflits par exemple à l'aide d'estampille)
- phase d'écriture (les valeurs de copies locales sont appliquées à la BD)

=> Techniques optimistes sont efficaces quand les conflits sont rares

Table of Contents

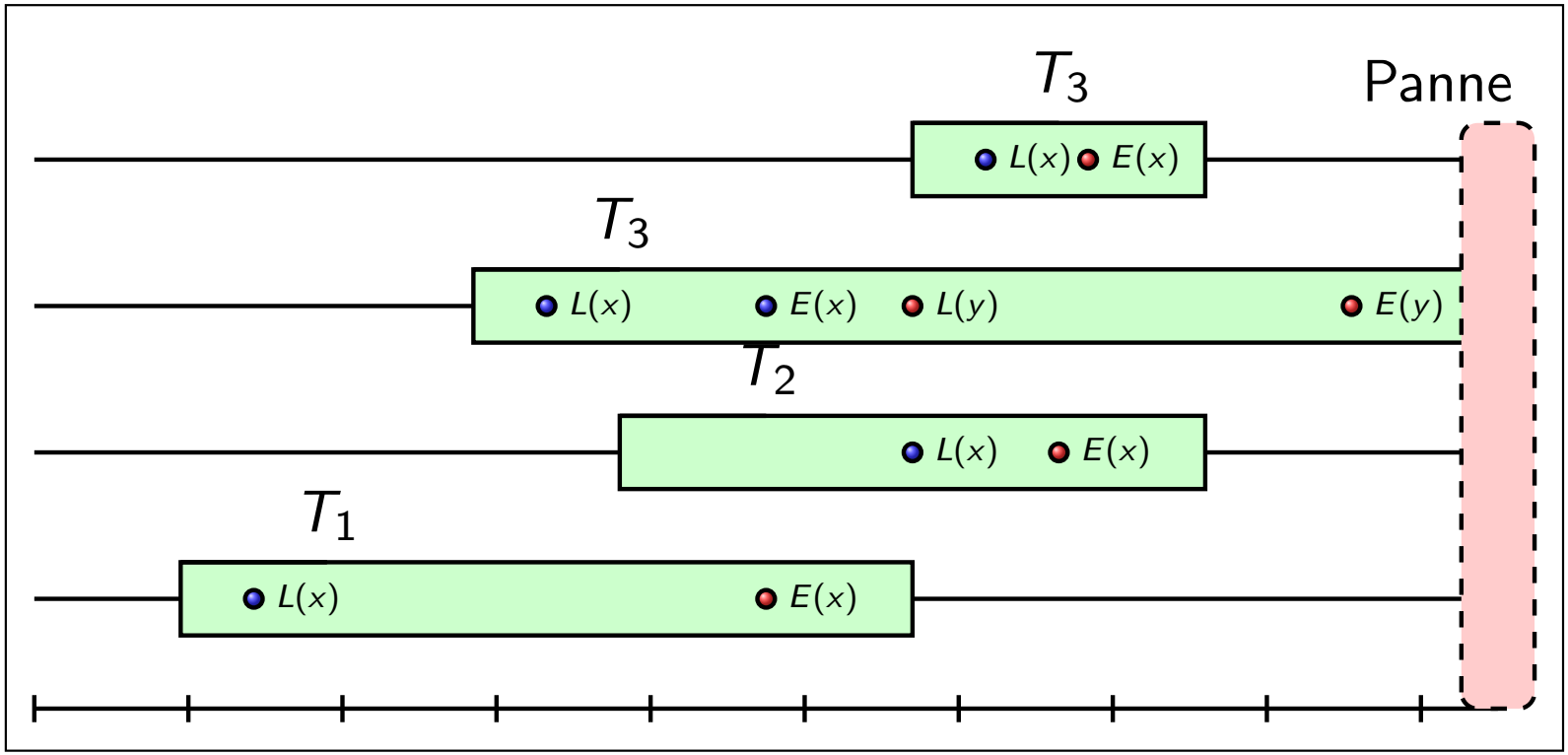
Transfert de données

En cas de panne toutes les écritures des transactions ayant validées doivent être effectives. Les différentes étapes d'une modification sont :

1. transfert des blocs disques en mémoire (lecture)
2. modifications des pages en mémoire
3. écriture des pages mémoires modifiées sur des disques

La manière dont les pages sont gérées en mémoires décide des instants d'écriture/lecture sur le disque.

Exemple



Reprise

Si les modifications de T_3 persistes il faut les annuler (*undo*). Si les modifications de T_1, T_2, T_4 n'ont pas persistées, il faut les refaire (*redo*).

Annulation en cascade

Est-il possible que l'annulation de T_3 entraîne l'annulation de $T2$?

Reprise

Pour permettre la récupération, le SGBD doit proposer :

- un mécanisme de sauvegarde de copies de la BD (sur un autre périphérique copie de sauvegarde)
- un mécanisme qui permet de forcer la permanence des modifications à un instant (check point).
- un gestionnaire de récupération pour restaurer la BD suite à une défaillance (journal / log).

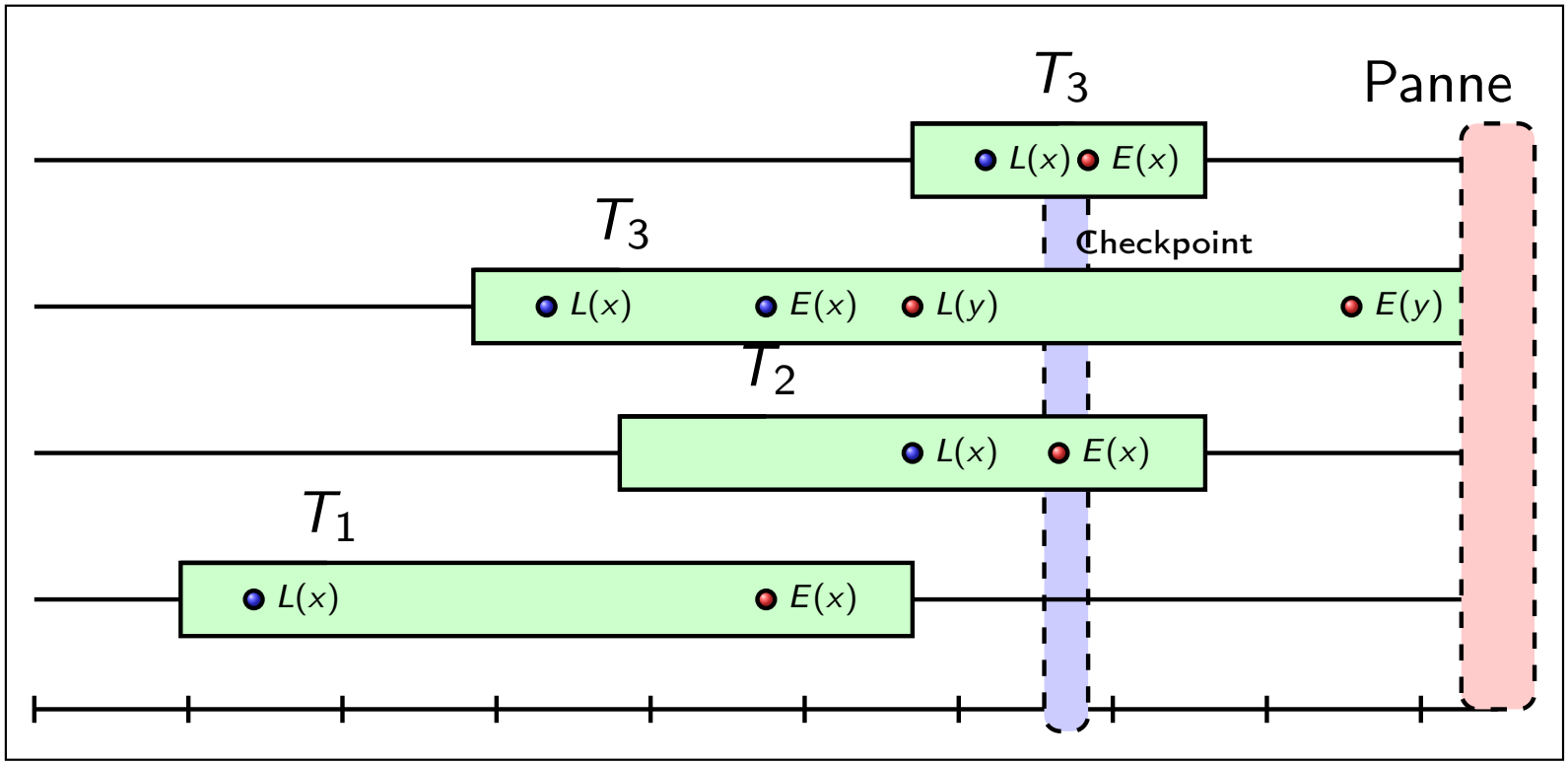
Point de contrôle :

Un point de contrôle est un point de synchronisation entre la BD et le journal :

- l'écriture de tous les tampons (mémoire) est forcée sur le stockage secondaire.

L'objectif est de savoir exactement quelles sont les transactions à refaire/défaire sans avoir à analyser toute l'histoire.

Exemple



Reprise

Parcours du journal :

- Rien à faire pour T_1.
- Refaire T_3 et T_2.
- Défaire T_3.

Récupération

Il faut stocker le journal sur un disque $\neq$. C'est le journal qui fait fois ! On écrit toujours dans le journal en premier.

Si la BD est endommagée, on part de la dernière sauvegarde. A l'aide du journal on refait toutes les transactions validées. Sinon, il faut refaire/défaire des choses.

Les deux principales méthodes :

- mises à jour différées
- mises à jour immédiates

Mises à jour différées

Mises à jour différées

Les modifications d'une T ne sont pas écrites sur le support de stockage tant qu'elle n'a pas validée.

- Il n'y a jamais d'opération à défaire.
- Il faut refaire les T validées dont les màj n'ont pas été répercutées sur la BD.

Contenu du Journal

Le journal comprend donc les informations suivantes :

- Début de T (heure, identifiant de T)
- Pour chaque opération d'écriture : la nature de l'opération (suppr, modif, ajout) avec la nouvelle valeur
- validation de T / annulation

Validation / Annulation

La transaction est validée uniquement une fois que les enregistrements du journal ont été répercutés sur le support de stockage. Si une Transaction annule d'elle même, il n'y a rien à faire.

Reprise

En cas de défaillance le journal est examiné en commençant par la dernière entrée et en remontant au point de contrôle le + récent.

Toute T démarrée et validée doit être refaite

On réécrit les nouvelles valeurs dans l'ordre chronologique (on refait tout, même ce qui a déjà été écrit).

Toute T démarrée non validée ou annulée est ignorée

On ne doit pas les défaire, les écritures n'ont pas étaient propagées sur le support persistant.

Immédiates

Mises à jour Immédiates

Les mises-à-jour sont appliquées au fur à mesure à la BD sans attendre les validations. Les mises à jour de la BD sont gérée par le gestionnaire de mémoire. En cas de panne, il est nécéssaire de défaire certaine T et d'en refaire d'autres.

Contenu du Journal

Le journal comprend donc les informations suivantes :

- Début de T (heure, identifiant de T)
- Pour chaque opération d'écriture : la nature de l'opération (suppr, modif, ajout) avec l'ancienne et la nouvelle valeur
- validation de T / annulation

Annulation

Si une transaction annule d'elle même les informations du journal sont utilisées pour la défaire.

Reprise

En cas de défaillance le journal est examiné en commençant par la dernière entrée et en remontant au point de contrôle le + récent.

Toute T démarrée et validée doit être refaite.

On réécrit les nouvelles valeurs dans l'ordre chronologique (on refait tout, même ce qui a déjà été écrit).

Toute T démarrée non validée ou annulée doit être défaite.

On réécrit les anciennes valeurs dans l'ordre anté-chronologique (on défait tout, même ce qui n'a pas été écrit).

Bibliographie.