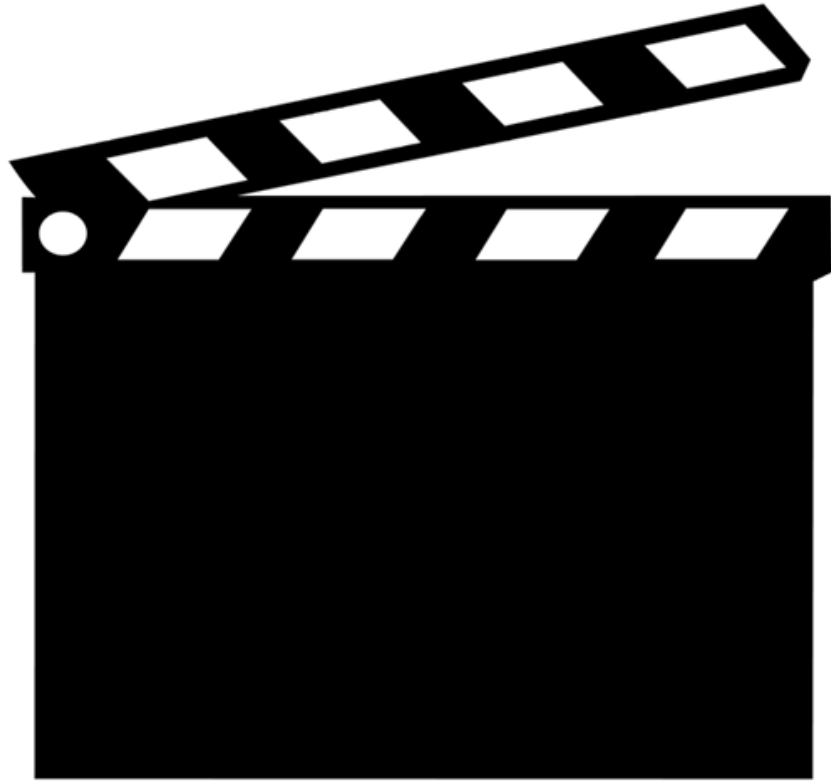


C018SA-W1-S5



SEMAINE 1 : Transactions et concurrence

1. Introduction : les transactions
2. Les problèmes
3. Sériabilité
4. Estampillage
- 5. Verrouillage à 2 phases (2PL)**
6. Degrés d'isolation dans les SGBD
7. Verrouillage hiérarchique

Le verrouillage : pourquoi partir lorsqu'on peut attendre

- Estampillage : soit tout se passe bien, soit on annule
- Fonctionne-t-on comme ça « dans la vraie vie » ?
Files d'attente, trains, etc ...
- Garantir le bon fonctionnement
 - On pose des **verrous** de plusieurs type pour empêcher une utilisation potentiellement conflictuelle des ressources (nuplets)
 - Selon le type de verrou, d'autres transactions pourront ou non interagir avec la ressource

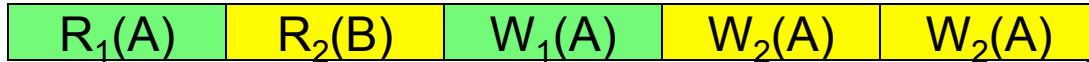
Principe de fonctionnement 2-Phase Locking

Phase 1 : Verrouillage

Phase 2 : libération lors du commit ou du rollback

- Lecture : Verrou « S » (Shared)
- Écriture : Verrou « X » (Exclusive)

| nuplet | Verrou |
|--------|--------------------------|
| A | S₁ |
| B | S ₂ |



- Tenter d'exécuter une lecture (SELECT) ou une écriture (UPDATE)
= essayer de poser un verrou

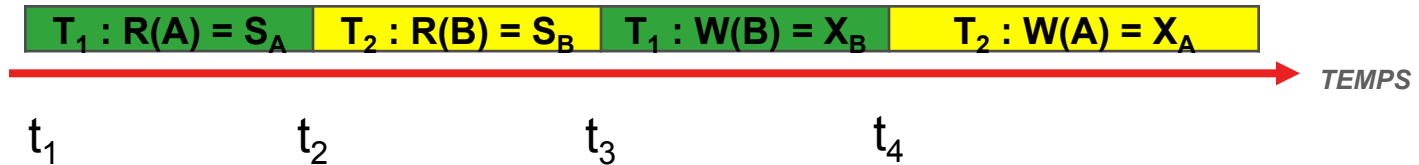
| Verrou demandé \ Verrou détenu | S | X |
|--------------------------------|-----------------|-----------------|
| | S | Accordé |
| X | Mise en attente | Mise en attente |

Théorème : un ordonnancement construit avec 2PL est sérialisable

Problèmes du verrouillage à 2 phases

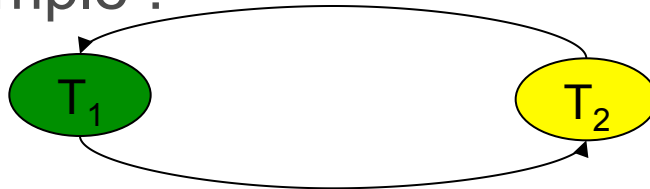
- **Verrou mortel** (cycle) ou « Deadlock »
 - Transactions qui s'attendent mutuellement
- **Performance**
 - Petit granule (nuplet) : → verrouillage coûteux
 - Gros granule (page, table) → temps d'attente plus important

Problèmes : verrou mortel

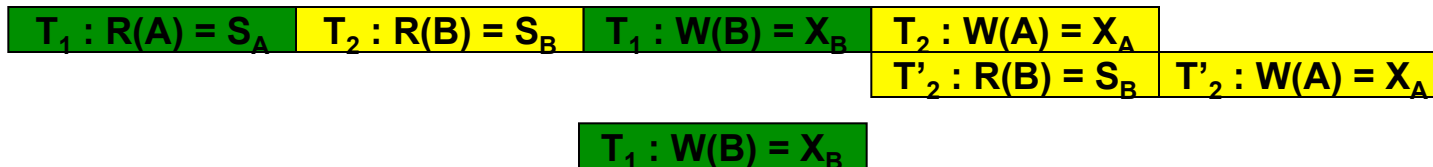


- Détection : graphe des attentes (comme le graphe de précédence)
 - T_i attend T_j si T_i demande un verrou détenu par T_j
 - Si un cycle apparaît, on a un verrou mortel !
 - Sur l'exemple :

DEADLOCK !

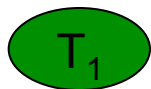


Résolution du problème de Deadlock : Wait-Die



- **Wait-Die** : Technique non-préemptive (n'interrompt pas)

- Si T_i demande un verrouillage accordé à T_j et T_i est **plus vieille** alors T_i est mise en attente
- Si T_i demande un verrouillage accordé à T_j et T_i est **plus jeune** alors T_i est annulée



T_1 : verrouillage accordé et continue



T_2 : annulée

Résolution du problème de Deadlock : Wound-Wait

$T_1 : R(A) = S_A$ | $T_2 : R(B) = S_B$ | $T_2 : W(A) = X_A$ | $T_1' : R(A) = S_A$ | $T_1' : W(B) = X_B$

- **Wound-Wait** : Technique préemptive (interrompt)

- Si T_i demande un verrouillage accordé à T_j et T_i est **plus vieille** alors T_j est annulée
- Si T_i demande un verrouillage accordé à T_j et T_i est **plus jeune** alors T_i est mise en attente



T_1 : termine avant
que T_1'

T_2 : termine

Résolution du problème de performances

- Demander la sérialisabilité *coûte cher* en terme de performances du SGBD
 - beaucoup d'opérations seront mises en attente.
 - Poser des verrous coûte cher
- Si le SGBD attend *sans rien faire* une transaction pour poursuivre c'est un gros problème !
 - on pourrait tolérer certaines anomalies : **Degrés d'isolation**
 - on pourrait verrouiller autrement : **Verrouillage hiérarchique**