DEV : Gestion de la concurrence dans un SGBD

Inspiration : Bases de données, Hainault.

Rappel : Une transaction est une suite d'opérations constituant une unité logique de traitement et pour laquelle on exige que les opérations vérifient les garanties ACID dans leur globalité.

On suppose que les garanties ACID sont déjà connues.

Exemple : Transaction bancaire

```
Transfert (C, S) {
  début transaction
    (1) M := SELECT montant FROM compte WHERE ID=C;
    (2) M := M + S
    (3) UPDATE compte SET montant := M WHERE ID=C;
  fin transaction
}
```

Définition : On dit que plusieurs transactions sont concurrentes si elles accèdent aux mêmes données en même temps.

↳ Peut engendrer des problèmes en cas de modification des données.

Exemple : Problème courant : Mise à jour perdue.

On reprend l'exemple.

2 exécutions possibles

↳ exécution séquentielle : (A.1), (A.2), (A.3), (B.1), (B.2), (B.3)

↳ Pas de problème par cohérence et isolement

↳ exécution entrelacée : (A.1), (A.2) ←1000, (B.1), (B.2) ←500, (A.3), (B.3)

↳ Montant augmenté de 500 et non de 1500

↳ Perte de la première mise à jour.

Solution possible : Verrouillage (solution la plus utilisée)

↳ Une transaction peut demander de poser un verrou sur une donnée (valeur, ligne, ensemble de lignes, table voire base de données entière). Celui-ci peut être partagé ou exclusif.

Utilisation

↳ Avant de lire/modifier une donnée, la transaction doit poser un verrou partagé/exclusif.

↳ A la fin de l'utilisation, le verrou en question est supprimé.

↳ Verrou partagé présent => Pas de verrou exclusif

↳ Verrou exclusif présent => Pas de verrou (partagé ou exclusif)

Exemple : On reprend le précédent.

La transaction nécessite un verrou exclusif sur la ligne : la seconde transaction est suspendue par le SGBD jusqu'à la libération du verrou.

Définitions: Soit $T = \{T_1, \dots, T_n\}$ un ensemble de transactions.

- On dit qu'une exécution de T est en série si elle est de la forme: la transaction T_i est exécutée, puis T_j, ...
- On dit qu'une exécution de T est sérialisable si elle est équivalente à une exécution en série.

Objectif: Garantir la sérialisabilité de l'exécution.

↳ Solution: verrouillage en 2 phases

Verrouillage à 2 phases

↳ Protocole de gestion des verrous au sein d'une transaction

↳ Phase croissante: demande de verrous sans relâchement

↳ Phase décroissante: relâchement des verrous sans demande.

(↳) On ne peut plus demander de verrou après en avoir relâché un.

Problème → Interblocage (dead lock)

Exemple: T1 demande un verrou exclusif sur D1

T2 " " D2

T1 " " D2

T2 " " D1

Solutions

↳ Prévention → SGBD empêche cette situation d'avoir lieu

↳ Correction → SGBD détecte le blocage, annule T2 puis la réexécute après T1.