

Systèmes de gestion de bases de données

Introduction 'a la notion de transaction

P. Rigaux

Cnam, dépt. informatique

November 18, 2016

De quoi s'agit-il?

Deux hypothèses à reconsidérer.

- Hyp. 1: **Un programme SQL s'exécute indépendamment des autres ("en isolation")**
 - ▶ **faux**, car de grandes bases de données peuvent gérer des centaines, voire des milliers d'accès **concurrents** qui peuvent interagir les uns sur les autres.
- Hyp. 2: **Un programme s'exécute sans erreur et intégralement.**
 - ▶ **faux** pour des raisons innombrables: plantage de l'application, pb réseau, pb du serveur, panne électrique, etc. L'interruption peut laisser la base dans un état **incohérent**.

Transaction

La notion de **transaction** répond à ces deux problèmes.

Objectifs du cours

Sans même comprendre les techniques utilisées il faut

- Maîtriser la notion, centrale, de **transaction**.
⇒ La spécification d'une transaction nous revient.
- Réaliser l'impact des **exécutions** transactionnelles **concurrentes** sur les autres utilisateurs
⇒ Un défaut d'isolation entre transaction peut engendrer des **incohérences**.
- Choisir un **niveau d'isolation** approprié
⇒ Compromis entre **fluidité/anomalies** et **isolation/cohérence**.

Les techniques? **Contrôle de concurrence**, objet du prochain cours.

Qu'est-ce qu'une transaction

Définition

Une transaction est une séquence d'opérations de **lecture** ou d'**écriture**, se terminant par `commit` ou `rollback`.

Le `commit` est une instruction qui **valide** toutes les mises à jour.

Le `rollback` est une instruction qui **annule** toutes les mises à jour.

Essentiel

Les opérations d'une transaction sont solidaires: elles sont toutes validées, ou pas du tout (**atomicité**).

Pour bien comprendre

On parle de **transaction** plutôt que de **programme** : beaucoup plus précis.

Une transaction s'exécute dans le cadre d'un **processus client** connecté au **processus serveur** (SGBD).

On peut effectuer une ou plusieurs transactions dans un même processus: elles sont dites **sérielles** (si pas de parallélisme).

En revanche, deux processus **distincts** engendrent des **transactions concurrentes**.

Notre exemple de base: les données

Notre base de données:

- Des clients qui réservent des places pour des spectacles

Client (id_client, nb_places_réservées, solde)

- Des spectacles qui proposent des places à des clients.

Spectacle (id_spectacle, nb_places_offertes,
nb_places_prises, tarif)

À retenir

Cette base est **cohérente** si **le nombre de places prises** est égal à **la somme des places réservées**.

Notre exemple de base: le programme (partie lectures)

```
procedure Reservation (v_id_client INT, v_id_spectacle INT, nb_places INT)
  -- Variables
  v_client Client%ROWTYPE;
  v_spectacle Spectacle%ROWTYPE;
  v_places_libres INT;
  v_places_reservees INT;
BEGIN
  -- On recherche le spectacle
  SELECT * INTO v_spectacle
  FROM Spectacle WHERE id_spectacle=v_id_spectacle;

  -- S'il reste assez de places: on effectue la reservation
  IF (v_spectacle.nb_places_libres >= nb_places)
  THEN
    -- On recherche le client
    SELECT * INTO v_client FROM Client WHERE id_client=v_id_client;

    -- Calcul du transfert
    v_places_libres := v_spectacle.nb_places_libres - nb_places;
    v_places_reservees := v_client.nb_places_reservees + nb_places;
```

Le programme, suite (partie écritures)

```
-- On diminue le nombre de places libres
UPDATE Spectacle SET nb_places_libres = v_places_libres
  WHERE id_spectacle=v_id_spectacle;

-- On augmente le nombre de places reervees par le client
UPDATE Client SET nb_places_reservees=v_places_reservees
  WHERE id_client = v_id_client;

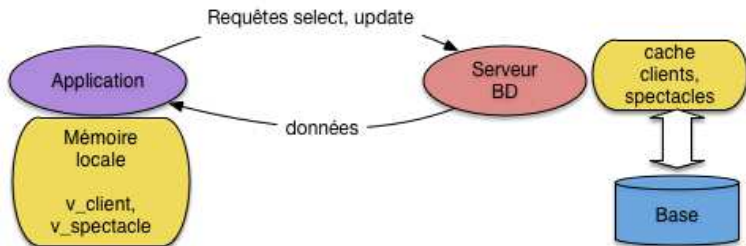
-- Validation
commit;
ELSE
  rollback;
END IF;
END;
```

NB: le langage (ici PL/SQL) **n'a aucun impact** sur la modélisation des transactions.

Ce programme engendre des **transactions**

L'**exécution** de ce programme effectue des lectures et des mises à jour constituant des **transactions**.

Les deux processus communiquent par message (requêtes, données).



Le SGBD ne sait pas ce que fait l'application avec les données transmises. **Il ne voit que la séquence des lectures et des écritures.**

Représentation d'une transaction

On représente les transactions par ce qu'en connaît le système.

- les transactions, notées T_1, T_2, \dots, T_n ;
- les "données" (tuples) échangées sont notées x, y, z, \dots ;
- pour chaque transaction T_i , une lecture de x est notée $r_i[x]$ et une écriture de x est notée $w_i[x]$;
- C_i et R_i représentent un commit (resp. rollback) effectué par la transaction T_i .

Une transaction T_i est donc une séquence de lectures ou d'écritures se terminant par C_i ou R_i . Exemple:

$$r_i[x] w_i[y] r_i[y] r_i[z] w_i[z] C_i$$

Les transactions engendrées par Réservation

En s'exécutant, la procédure Réservation engendre des transactions.

Exemples:

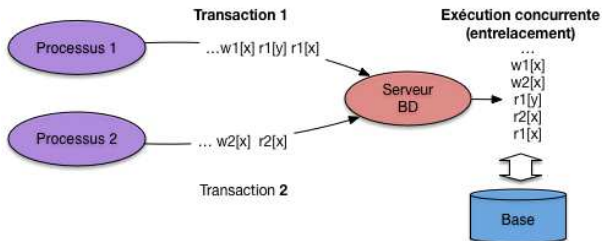
- $r[s_1]r[c_1]w[s_1]w[c_1]C$: on lit le spectacle s_1 , le client c_1 , puis on les met à jour tous les deux;
- $r[s_1]r[c_2]w[s_1]w[c_2]C$: un autre client (c_2) réserve pour le même spectacle (s_1);
- $r[s_1]$: on lit le spectacle s_1 , et on s'arrête là (plus assez de places disponibles?)

Un même processus peut effectuer des transactions **en série**:

$$r_1[s_1]r_1[c_1]w_1[s_1]w_1[c_1]C_1r_2[s_1]r_2[c_2]w_2[s_1]w_2[c_2]C_2 \dots$$

Exécutions concurrentes

Quand plusieurs programmes clients sont actifs simultanément, les transactions engendrées s'effectuent **en concurrence**.



On obtient potentiellement un **entrelacement des requêtes**.

Important

L'entrelacement de requêtes issues de transactions concurrentes engendre des anomalies.

Propriétés des transactions

Un système relationnel contrôle la concurrence et garantit un ensemble de propriétés rassemblées sous l'acronyme ACID.

- **A = Atomicité.** Une transaction est validée complètement ou pas du tout.
- **C = Cohérence.** Une transaction mène d'un état cohérent à un autre état cohérent.
- **I = Isolation.** Une transaction s'exécute comme si elle était seule.
- **D = Durabilité.** Quand le commit s'exécute, ses résultats sont définitifs.

Essentiel

L'isolation **par défaut** est seulement partielle: **meilleures performances** mais **risque d'anomalie**.

Vérifions

(Démonstration de sessions concurrentes avec MySQL)

Réfléchir:

- J'effectue un commit, puis je m'aperçois d'une erreur: est-il encore temps de faire un rollback?
- La transaction $r[s_1]r[c_1]w[s_2]w[c_1]C$ peut-elle être engendrée par la procédure Réservation?

- J'exécute la commande:

```
| DELETE * FROM Client; WHERE id=1;
```

Réponse: **100 000 lignes détruites**. Pourquoi et que faire?

- Exécution de Réservation: je lis un spectacle: il reste 10 places libres; je veux en réserver 5: on me répond qu'une autre transaction a tout pris. Est-ce possible dans un système transactionnel?