Introduction aux transactions

Adapté du cours de Philippe Rigaux (cnam)

De quoi s'agit-il ?

Deux hypothèses à reconsidérer.

- Hyp. 1 : **Un programme SQL s'exécute indépendamment des autres**
  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**.

La notion de **transaction** est centrale pour répondre à ce problème.

Objectifs du cours

Un concepteur d'application doit :

- Maîtriser la notion, très importante, de **transaction**.
- Réaliser l'impact des **exécutions** transactionnelles **concurrentes** sur les autres utilisateurs
- Choisir un **niveau d'isolation** approprié

Les techniques ? **Contrôle de concurrence**

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.

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 est le produit d'un échange entre un **processus client** et un **processus serveur** (SGBD).

On peut effectuer une ou plusieurs transactions successives dans un même processus : elles sont dites **sérielles**.

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

Notre exemple de base : les 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)`

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

Notre exemple de base : le programme (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;
```

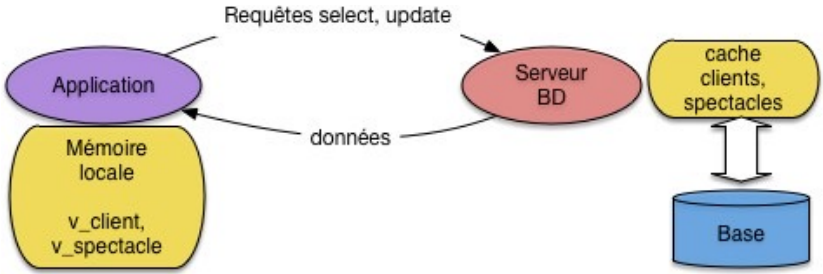
Le programme, suite (partie écritures)

```
        -- 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

Exécution = séquence de lectures et mises à jour = **transaction**.



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 $T_1, T_2, \cdots, T_n$;
- les "données" (tuples) échangées sont notées $x, y, z, \cdots$;
- 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 :

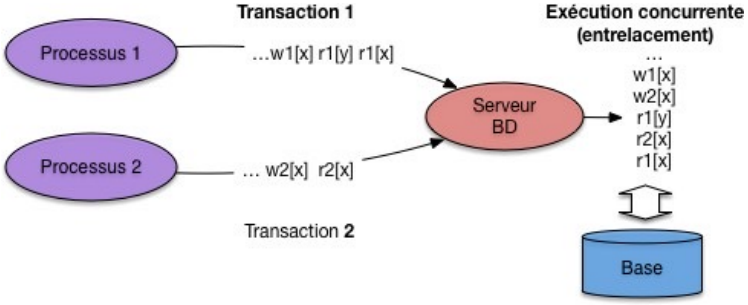
- $r[s_1]r[c_1]w[s_1]w[c_1]C$: on lit le spectacle $s1$, le client c_1, puis on les met à jour tous les deux ;
- $r[s_1]r[c_2]w[s_1]w[c_2]C$: un autre client (c_2) réserve pour le même spectacle (s_1) ;
- $r[s_1]$: on lit le spectacle s_1, et on s'arrête là (plus assez de places disponibles ?)

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

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

Exécutions concurrentes

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



On obtient potentiellement un **entrelacement des requêtes**.
L'entrelacement de requêtes issues de transactions concurrentes **peut** engendrer des anomalies.

Propriétés des transactions

Un système relationnel contrôle la concurrente 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**.

À retenir

Transaction = séquence de lecture et mises à jour soumises par une application client, se terminant par `commit` ou `rollback`

Le SGBD **garantit** que l'exécution des transactions respecte des propriétés dites ACID.

Cette garantie dépend du **niveau d'isolation** qui est choisi par l'application (son développeur)

Un niveau d'isolation insuffisant peut entraîner des anomalies très difficiles à comprendre.