TD6 - Isolation des transactions (suite)

Exercice 1 :

Connectez-vous à votre base de données sur un terminal que l'on appellera `term1`, et placez la fenêtre du terminal à gauche de votre écran. Exécutez le script `lire.sql`. Connectez-vous à votre base de données sur un second terminal que l'on appellera `term2`, et placez la fenêtre du terminal à droite de votre écran. Vérifiez sur `term2` que le schéma `lire` a bien été créé (`\dn`) ainsi que les tables (`\dt lire.*`), les vues (`\dv lire.*`) et les fonctions (`\df lire.*`). Pointez le chemin de parcours sur le schéma `lire` dans le terminal `term2`.

(a)

- `term1` : Insérer la valeur (default, 'Drew') dans la table `lecteur`.
- `term2` : Est-ce que la valeur est visible ? Pourquoi ?

(b)

- `term1` : Commencer une transaction.
- `term1` : Insérer la valeur (default, 'Eric') dans la table `lecteur`.
- `term2` : Est-ce que la valeur est visible ? Pourquoi ?
- `term1` : Validez la transaction.
- `term2` : Est-ce que la valeur est visible ? Pourquoi ?

(c) Dans chacun des deux terminaux, exécuter la commande `\set AUTOCOMMIT off` pour désactiver le mode `COMMIT` implicite (chaque requête SQL exécutée en dehors d'un bloc de transaction est considérée comme une transaction dans Postgresql). Attention, il faut respecter la casse. Vérifier que la commande a bien été prise en compte en exécutant dans chacun des deux terminaux : `\echo :AUTOCOMMIT`

- `term1` : Commencer une transaction.
- `term1` : Faire emprunter le livre 6 par le lecteur 4.
- `term1` : Interroger les relations avec la vue `infos`.
- `term2` : Interroger les relations avec la vue `infos`.
- `term2` : Est-ce que la mise à jour a été prise en compte ? Pourquoi ?
- `term2` : Faire emprunter le livre 7 par le lecteur 5.
- `term2` : Quelle(s) mise(s) à jour a (ont) été prise(s) en compte ? Pourquoi ?
- `term1` : Quelle(s) mise(s) à jour a (ont) été prise(s) en compte ? Pourquoi ?
- `term1` : Validez la transaction.
- `term2` : Quelle(s) mise(s) à jour a (ont) été prise(s) en compte ? Pourquoi ?
- `term1` : Quelle(s) mise(s) à jour a (ont) été prise(s) en compte ? Pourquoi ?

(d) Dans chacun des deux terminaux, exécuter la commande `\set AUTOCOMMIT on` pour résactiver le mode `COMMIT` implicite. Vérifier que la commande a bien été prise en compte en exécutant dans chacun des deux terminaux : `\echo :AUTOCOMMIT`

- `term1` : Créer une table avec un seul attribut de type entier et utiliser la fonction `generate_series` pour y insérer les nombres impairs compris entre 1 et 10.
- `term2` : Vérifier que les valeurs sont visibles.
- `term1` : Commencer une transaction.
- `term1` : Multiplier toutes les valeurs par 2.
- `term2` : Multiplier toutes les valeurs par 3. Que s'est-il passé ?

- `term1` : Valider la transaction.
- `term2` : Que s'est-il passé ?
- `term1` : Vérifier le contenu de la table. Que s'est-il passé ?

(e)

- `term2` : Créer une table avec un seul attribut de type entier et utiliser la fonction `generate_series` pour y insérer tous les nombres compris entre 1 et 10.
- `term2` : Commencer une transaction.
- `term1` : Commencer une transaction.
- `term1` : Incrémenter toutes les valeurs d'une unité.
- `term2` : Incrémenter toutes les valeurs d'une unité.
- `term1` : Incrémenter toutes les valeurs d'une unité.
- `term1` : Valider la transaction.
- `term2` : Sans exécuter une requête SQL, de combien d'unités ont été incrémentées les valeurs de la table ? Pourquoi ?
- `term2` : Vérifier la réponse avec une requête.
- `term1` : Sans exécuter une requête SQL, de combien d'unités ont été incrémentées les valeurs de la table ? Pourquoi ?
- `term1` : Vérifier la réponse avec une requête.