Cours de « Bases de données »

Année académique 2020-2021

T.P. 8 : Les transactions

Exercice 1

Une base de données bancaire contient les relations suivantes :

- *client(ID, NOM, PRENOM, ADRESSE)*
  contenant la liste des clients ;
- *compte(IBAN, SOLDE, #ID)*
  reprenant la liste des comptes bancaires et leur titulaire ;
- *transaction(NUM,#IBANFROM, #IBANTO, MONTANT, DATE)*
  reprenant la liste des transactions effectuées.

1. En utilisant `LOCK TABLES`, écrire un script pour une transaction qui effectuerait un transfert d'un compte vers un autre.
2. Améliorer le script pour que plusieurs transactions puissent travailler simultanément sur la base de données.
3. Offrir un compte épargne gratuit à 500 clients, tirés au hasard. Travailler avec 5 transactions en parallèle pour répartir la charge (le code est identique aux 5 transactions). Attention à ne pas ouvrir plusieurs comptes gratuits pour le même client.
4. Écrire un script permettant d'encoder un nouveau client, sachant que l'ID est unique, va croissant pour les clients, et que si N est le numéro maximum pour ID, alors tous les numéros strictement inférieurs à N et plus grands que 0 sont utilisés dans la bases de données. La politique de la maison est d'annoncer à la personne son futur numéro client au début de l'entretien. On propose une tasse de café au nouveau client pour l'accueillir.

Exercice 2

Une base de données contient les valeurs suivantes :

COL
1
2
3

Imaginer les scénarios suivants :

1. — (Transaction 1) : `SELECT SUM(COL) FROM TABLE;` $\Longrightarrow$ variable INT
   — (Transaction 1) : Wait 20 seconds
   — (Transaction 2) : `UPDATE TABLE SET COL = COL+1;`
   — (Transaction 2) : Wait 20 seconds
   — (Transaction 1) : `SELECT INT-SUM(COL) FROM TABLE;`
   — (Transaction 2) : `ROLLBACK;`
2. — (Transaction 1) : `SELECT SUM(COL) FROM TABLE;` $\Longrightarrow$ variable INT
   — (Transaction 1) : Wait 20 seconds
   — (Transaction 2) : `UPDATE TABLE SET COL = COL+1;`
   — (Transaction 2) : Wait 20 seconds
   — (Transaction 1) : `SELECT INT-SUM(COL) FROM TABLE;`
   — (Transaction 2) : `COMMIT;`
3. — (Transaction 1) : `SELECT SUM(COL) FROM TABLE;` $\Longrightarrow$ variable INT
   — (Transaction 1) : Wait 20 seconds
   — (Transaction 2) : `INSERT INTO TABLE VALUES (4);`
   — (Transaction 2) : Wait 20 seconds
   — (Transaction 1) : `SELECT INT-SUM(COL) FROM TABLE;`
   — (Transaction 2) : `COMMIT;`

Donner, pour chaque scénario, le résultat affiché par le dernier `SELECT` de la transaction 1, selon que le niveau d'isolation des transactions est, respectivement :

— READ UNCOMMITTED
— READ COMMITTED
— REPEATABLE READ
— SERIALIZABLE