Université Sorbonne Paris Nord
École d'ingénieur Sup Galilée

TP n° 5 : Transactions et contrôle de concurrence

Rapport de TP

Réalisé par :

Célian WIRTZ (celian.wirtz@edu.univ-paris13.fr)

12100311

Date • 29/04/25

Année universitaire : 2024/2025

Responsable

M. Youcef

Exercice n° 1

On rappel que le Autocommit est désactivé.

Question 2 :

```
INSERT INTO transaction VALUES (’1’, 862);
INSERT INTO transaction VALUES (’2’, 1354);
UPDATE transaction SET valTransaction = 726 WHERE idTransaction = ’1’;
DELETE FROM transaction WHERE idTransaction = ’2’;
ROLLBACK;
```

Si on sélectionne avant le rollback on aura la transaction où id = ‘1’ et la valeur est de 726 cependant après le rollback il n’y a plus rien puisque rien n’avait été enregistré, même les insertions.

Question 3 :

Sans commit, les changements ne sont pas partagé à S1 et donc la table est vide dans la session S1.

Question 4 :

La fermeture brutale ne sauvegarde pas les données ainsi lorsque l’on se reconnecte avec la même session, les données sont perdues et la table contient les éléments initiaux mais pas ceux qu’on vient d’insérer.

Question 5 :

```
INSERT INTO transaction VALUES (’3’, 422);
ALTER TABLE transaction ADD val2Transaction NUMBER(10);
ROLLBACK;
```

Lorsqu’on fait un SELECT après le Rollback on se rend compte qu’il n’y a pas la transaction 3, cela peut paraître étonnant puisque nous n’avons même pas fermé la session. Mais cela vient du fait qu’une altération de la structure de la table provoque un auto-commit, celui-ci revient donc aux données sauvegardés (il fait un rollback qui supprime donc l’insertion) puis altère la table et enfin fait un commit. En conséquence quand on regarde la structure de la table après le rollback, val2Transaction est bien présente.

Question 6 :

Une session est un espace individuel à la base de données qui n'est pas partagé aux autres sauf lors d'un commit.
Une transaction est une ou plusieurs opérations qui viennent modifier l'état d'une session, celle-ci ne modifie pas la dase de donnée tant qu'un commit n'a pas eu lieu. Elle est perdue en cas de Rollback sans commit au préalable (sur un txt si on écrit, tant qu'on a le txt ouvert, les infos sont là mais si on le ferme puis qu'on l'ouvre mais qu'on avait pas sauvegardé c'est perdu, c'est le même mécanisme içi sauf que la sauvegarde est collective).
La validation c'est simplement un commit qui se déroule sans erreur.

Exercice n° 2

Nous travaillons en READ COMMITTED

On insère un vol et deux clients :

```
INSERT INTO vol VALUES ('1', 500, 378);
INSERT INTO client VALUES ('1', 'Albert', 3);
INSERT INTO client VALUES ('2', 'Marie', 4);
COMMIT;
```

Question 1 :

Puisqu'on effectue des transactions que dans T1 et qu'il n'y a pas de commit, on constate en effet que T1 à accès à ces transactions mais pas T2.

Question 2 :

Après un rollback comme prévu et de la même façon que dans l'exercice 1, la transaction est perdue pour T1 car on revient au commit précédent (donc avant la transaction) et T2 reste inchangé puisque de toute façon ces transactions ne l'ont jamais atteint.

Question 3 :

Sans commit, les changements ne sont pas partagé à S1 et donc la table est vide dans la session S1.

Question 4 :

Après un Commit, les transactions sont sauvegardés et si on les supprime puis qu'on fait un rollback on les retrouve donc. De son côté T2 voit apparaître la transaction de T1 suite au commit de T1, ce qui prouve que nous sommes en READ COMMITTED et non pas en REPEATABLE READ, si nous étions en REPEATABLE READ nous ne pourrions pas voir les changements tant que nous n'avons pas fait de Rollback avec T2 (ce qui irait alors prendre le dernier commit donc celui de T1).

isolation incomplète = incohérence possible :

En fait, lorsque à chaque client on modifie séparément le nombre de place prise pour la table client, il n'y a pas de conflit. En effet les transactions concernent deux clients différents donc ne modifient pas les mêmes valeurs donc au final les deux clients réservent le bon nombre de place.

Cependant pour la table vol, lorsque dans T1 on fait nb_places réservés =+ 2, on a alors dans la section active, nb_places réservés = 0 + 2 = 2.
Juste après dans T2 on rajoute 3 à cette variable sauf que on rajoute à la dernière valeur qui est enregistré (donc du commit donc 0) et on a 0 + 3 = 3.
C'est là où réside le problème, la valeur initial devrait être 2 mais puisque T1 n'a pas commit, cela reste 0. Alors lorsque T1 commit puis T2 commit la

table client voit les bonnes valeurs arrivés une après l'autre mais la table vol passe de nb_places réservés = 0 à nb_places réservés = 2 (commit T1) à nb_places réservés = 3 (commit T2) et non pas nb_places réservés = 2 + 3…