Fondements des bases de données

Programmation en PL/SQL Oracle – les transactions

Équipe pédagogique BD

`http://liris.cnrs.fr/~mplantev/doku/doku.php?id=lif10_2016a`

Version du 13 octobre 2016

Transactions

Problèmes de cohérence et transaction

Un SGBD doit pouvoir supporter :

- plusieurs utilisateurs l'utilisant en parallèle
- effectuant des opérations d'écriture et de lecture
- tout en garantissant la cohérence des données

Une transaction est un ensemble d'ordres (SQL) indivisibles, faisant passer la base de données d'un état cohérent à un autre en une seule étape.

Propriétés ACID

Atomicité Une transaction réussi ssi toutes ses opérations réussisent.

Cohérence Une transaction terminée laisse la base dans un état cohérent où les données sont intègres.

Isolation Les transactions doivent être rendues indépendantes les unes des autres.

Durabilité Les effets d'uns transaction terminée sont persistant.

Opérations COMMIT et ROLLBACK

ROLLBACK	*annule* entièrement une transaction : toutes les modifications depuis le début de la transaction sont alors défaites.
COMMIT	*valide* entièrement une transaction : les modifications deviennent définitives et visibles à tous les utilisateurs.
SAVEPOINT	point de contrôle, état de la base où l'on pourra revenir plus tard.

A noter

- En cours de transaction, *seul l'utilisateur ayant effectué les modifications les voit*.
- En cas de fin anormale d'une tâche utilisateur il y a automatiquement `ROLLBACK` des transactions non terminées.
- Une transaction commence (implicitement) à la première opération SQL rencontrée et dès qu'une transaction est terminée.
- Les commandes de définition de données sont automatiquement commitées (*auto-commit*) et valident donc les ordres précédent.
- Un mécanisme de vérouillage permet de gérer les conflits d'accès parallèle.

Exemple

```
INSERT INTO transactions VALUES (1,'Pas de savepoint');
-- 1 ligne creee.
INSERT INTO transactions VALUES (2,'Savepoint A');
-- 1 ligne creee.
SAVEPOINT A;
-- Savepoint cree.

INSERT INTO transactions VALUES (3,'Savepoint B');
-- 1 ligne creee.
SAVEPOINT B;
-- Savepoint cree.

INSERT INTO transactions VALUES (4,'Pas de savepoint');
-- 1 ligne creee.
```

Exemple

```
ROLLBACK TO SAVEPOINT B;
-- Annulation (rollback) effectuee.
SELECT * FROM transactions;
/*          ID DESCRIPTION
------------ ----------------------------
           1 Pas de savepoint
           2 Savepoint A
           3 Savepoint B                 */

ROLLBACK TO SAVEPOINT A;
--Annulation (rollback) effectuee.
SELECT * FROM transactions;
/*          ID DESCRIPTION
------------ ----------------------------
           1 Pas de savepoint
           2 Savepoint A                 */
```

Exemple

```
COMMIT;
-- Validation effectuee.
SELECT * FROM transactions;
        ID DESCRIPTION
---------- ------------------------------
/*       1 Pas de savepoint
         2 Savepoint A                  */
```

Triggers et transactions

Le trigger est une extension de l'opération de modification de donnée sur laquelle il se déclenche, or ces opérations sont atomiques et ne peuvent pas être divisées :

`COMMIT` et `ROLLBACK` sont donc interdits dans le corps des triggers.

Workaround : `PRAGMA AUTONOMOUS_TRANSACTION`

Permet de déclarer une transaction *autonome*. Attention aux effets : ici, si l'insert est annulé, le tuple de la relation `log` sera bien ajouté !

```
CREATE OR REPLACE TRIGGER tab1_trig
  AFTER insert ON tab1
  DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
  BEGIN
  INSERT INTO log VALUES (SYSDATE, 'Insert on TAB1');
  COMMIT; -- only allowed in autonomous triggers
  END;
/
```

Niveaux d'isolation

Du plus souple (garantie minimum) au plus restrictif :

`READ UNCOMMITED` Permet à une autre transaction de lire des données qui ont été changées, mais pas encore validées.

`READ COMMITED` C'est le paramètre par défaut pour Oracle. Il assure que chaque requête dans une transaction lit seulement les données validées.

`REPEATABLE READ` Ce niveau permet à une transaction de lire les mêmes données plusieurs fois avec la garantie qu'elle recevra les mêmes résultats à chaque fois.

`SERIALIZABLE` Une transaction ne prend en compte que les données validées avant le démarrage de la transaction.

Niveaux Oracle

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET TRANSACTION READ ONLY;
-- semblable a REPEATABLE READ mais
-- sans autoriser les modifications.
```

Fin du cours.