TD5 - Isolation des transactions

Objectif : comprendre les **mécanismes de gestion de la concurrence** par PostgreSQL et les différents **niveaux d'isolation**.

La norme SQL définit 4 niveaux d'isolation, qui sont du moins restrictif au plus restrictif :

- Mode `READ UNCOMMITTED` : une transaction voit toutes les modifications des autres transactions (lecture sale).
- Mode `READ COMMITTED` : une transaction ne voit que les modifications validées par les autres transactions (écriture sale).
- Mode `REPEATABLE READ` : une transaction peut lire des lignes fantômes (tuples insérés par une transaction et ne pouvant être totalement pris en compte par une autre transaction).
- Mode `SERIALISABLE` : une transaction s'exécute de la même façon que si elle était la seule transaction en cours de la BD (deadlock).

Dans PostgreSQL, seuls trois niveaux distincts d'isolation des transactions sont implémentés (le mode `Read Uncommitted` de PostgreSQL se comporte comme le mode `Read Committed`).

Niveau d'isolation	Lecture sale	Lecture non reproductible	Lecture fantôme	Anomalie de sérialisation
Read Uncommitted	Autorisé, mais pas dans PostgreSQL	possible	possible	possible
Read Committed	impossible	possible	possible	possible
Repeatable Read	impossible	impossible	Autorisé, mais pas dans PostgreSQL	possible possible
Serializable	impossible	impossible	impossible	impossible

TABLE 1 – Niveaux d'isolation des transactions

Source : `https://docs.postgresql.fr/13/transaction-iso.html`

Exercice 1 :

Soit la table `Gardien` du script `zoo.sql`.

1. Commit / Rollback / Savepoint

(a) Commencer une transaction (par `BEGIN ;`), et dans cette transaction, essayer de faire les manipulations ci-dessous :

```
BEGIN;
INSERT INTO GARDIEN VALUES (80, 'Durand', 'aa', '1990-01-16');
SELECT * FROM GARDIEN ;
ROLLBACK ;
SELECT * FROM GARDIEN ;
```

Commenter chaque action ! Que fait le serveur Postgres ?

(b) Recommencer une autre transaction, et dans cette transaction, essayer de faire les manipulations ci-dessous :

```
BEGIN;
INSERT INTO GARDIEN VALUES (800, 'Dupont', 'aa', '1990-01-16');
SELECT * FROM GARDIEN ;
SAVEPOINT pointControl ;
INSERT INTO GARDIEN VALUES (900, 'Durand', 'aa', '1990-01-16');
SELECT * FROM GARDIEN;
ROLLBACK TO pointControl ;
SELECT * FROM GARDIEN ;
ROLLBACK ;
SELECT * FROM GARDIEN ;
```

Commenter chaque action ! Que fait le serveur Postgres ?

(c) Recommencer une autre transaction, et dans cette transaction, essayer de faire les manipulations ci-dessous :

```
BEGIN;
INSERT INTO GARDIEN VALUES (80, 'Durand', 'aa', '1990-01-16');
SELECT * FROM GARDIEN ;
COMMIT ;
SELECT * FROM GARDIEN ;
```

Commenter chaque action ! Que fait le serveur Postgres ?

(d) Commencer une transaction contenant un `BEGIN;`, puis 3 insertions dans `Gardien`, suivies d'un commit, suivi d'une insertion, une mise à jour et une suppression, suivies d'un `BEGIN;`, suivi d'une sélection complète sur la table, suivi de 2 mises à jour, suivies d'une sélection complète sur la table, suivie d'un rollback, suivi d'une sélection complète sur la table.
Commenter chaque action ! Que fait le serveur Postgres ?

2. Intégrité des données

(a) Recommencer une autre transaction, et dans cette transaction, essayer de faire les manipulations ci-dessous :

```
BEGIN;
INSERT INTO GARDIEN VALUES (null, 'Dupont', 'Montreuil', '1970-10-25');
SELECT * FROM GARDIEN;
COMMIT ;
SELECT * FROM GARDIEN;
```

Commenter chaque action ! Que fait le serveur Postgres ?

(b) Recommencer une autre transaction, et dans cette transaction, essayer de faire les manipulations ci-dessous :

```
BEGIN;
INSERT INTO GARDIEN VALUES (85, 'Dupont', 'Montreuil', '1970-10-25');
SELECT * FROM GARDIEN where nom = 'Dupont';
COMMIT ;
SELECT * FROM GARDIEN where nom = 'Dupont';
```

Commenter chaque action ! Que fait le serveur Postgres ?