TD3 - Transactions

1 Préambule

Les objectifs de ce TD sont l'observation et la compréhension des comportements de PostgreSQL dans la gestion des transactions concurrentes, notamment la pose des verrous sur les tables et sur les n-uplets. Veillez à vous connecter en **mode "auto-commit" désactivé** pour effectuer les simulations de transaction.
Pour effectuer les manipulations, nous aurons besoin de la table suivante :

```
CREATE TABLE enfant(
   nopers     INTEGER,
   prenom     VARCHAR(20),
   points     INTEGER,
   classe     VARCHAR(10)
);
```

à remplir avec les n-uplets suivants :

nopers	prenom	points	classe
1	Jean	10	CE1_2
2	Pierre	17	CE1_2
3	Alfred	15	CE1_2
4	Aline	9	CE2_2

Dans la suite, les instructions sont fournies dans des séquences numérotées (t1, t2, *etc.*) et doivent être saisies dans cet ordre. Les transactions sont notées U1, U2, *etc.*
L'exécution, notamment à cause des verrous, ne se fera pas forcément dans l'ordre de la saisie. Vous devrez expliquer précisemment ce qui se passe à chaque exercice, et pour quelles raisons.
Sauf indication contraire, les transactions s'exécutent dans le mode d'isolation par défaut de PostgreSQL.

2 Exercices

Ex. 1 — A partir de deux transactions (U1 et U2), observer, noter et expliquer le comportement de la séquence suivante.

```
U1   t1   START TRANSACTION ;
U2   t2   START TRANSACTION ;
U1   t3   SELECT classe FROM enfant WHERE nopers = 1 ;
U2   t4   UPDATE enfant SET classe = 'CM1_1' WHERE nopers = 1 ;
U1   t5   SELECT classe FROM enfant WHERE nopers = 1 ;
U2   t6   COMMIT ;
U1   t7   SELECT classe FROM enfant WHERE nopers = 1 ;
U1   t8   COMMIT ;
U1   t9   SELECT classe FROM enfant WHERE nopers = 1 ;
```

1. Donnez pour chaque étape de la séquence, les verrous demandés.
2. Y-a-t'il des verrous incompatibles entre eux ? Est-ce que chaque verrou demandé est posé ici ?
3. Quelle est alors la séquence rééelle d'éxécution ?
4. Quel genre de défaut est mis en évidence ici ?

Ex. 2 — Remettre la table `enfant` dans son état initial.
A partir de deux transactions (U1 et U2), observer, noter et expliquer le comportement de la séquence suivante :

```
U1   t1   START TRANSACTION ;
U2   t2   START TRANSACTION ;
U1   t3   SET TRANSACTION ISOLATION LEVEL REPEATABLE READ ;
U1   t4   SELECT classe FROM enfant WHERE nopers = 1 ;
U2   t5   UPDATE enfant SET classe = 'CM1_1' WHERE nopers = 1 ;
```

```
U1   t6   SELECT classe FROM enfant WHERE nopers = 1 ;
U2   t7   COMMIT ;
U1   t8   SELECT classe FROM enfant WHERE nopers = 1 ;
U1   t9   COMMIT ;
U1   t10  SELECT classe FROM enfant WHERE nopers = 1 ;
```

1.Donnez pour chaque étape de la séquence, les verrous demandés.

2.Y-a-t'il des verrous incompatibles entre eux ? Est-ce que chaque verrou demandé est posé ici ?

3.Quelle est alors la séquence réélle d'éxécution ?

4.Est-ce que le défaut mis en évidence dans l'exercice précédent est encore présent ici ? Pourquoi ?

Ex. 3 — Remettre la table `enfant` dans son état initial.
A partir de deux transactions (U1 et U2), observer, noter et expliquer le comportement de la séquence suivante :

```
U1   t1   START TRANSACTION ;
U2   t2   START TRANSACTION ;
U1   t3   SELECT points FROM enfant WHERE nopers = 1 FOR UPDATE ;
U2   t4   SELECT points FROM enfant WHERE nopers = 1 FOR UPDATE ;
U2   t5   UPDATE enfant SET points = points + 5 WHERE nopers = 1 ;
U1   t6   UPDATE enfant SET points = points + 8 WHERE nopers = 1 ;
U2   t7   COMMIT ;
U1   t8   COMMIT ;
```

1.Donnez pour chaque étape de la séquence, les verrous demandés.

2.Y-a-t'il des verrous incompatibles entre eux ? Est-ce que chaque verrou demandé est posé ici ?

3.Quelle est alors la séquence réélle d'éxécution ?

Ex. 4 — Remettre la table `enfant` dans son état initial.
A partir de deux transactions (U1 et U2), observer, noter et expliquer le comportement de la séquence suivante :

```
U1   t1   START TRANSACTION ;
U2   t2   START TRANSACTION ;
U1   t3   UPDATE enfant SET points = points + 5 WHERE nopers = 1 ;
U2   t4   UPDATE enfant SET points = points + 7 WHERE nopers = 2 ;
U1   t5   UPDATE enfant SET points = points + 9 WHERE nopers = 2 ;
U2   t6   UPDATE enfant SET points = points + 11 WHERE nopers = 1 ;
U2   t7   COMMIT ;
U1   t8   COMMIT ;
```

1.Donnez pour chaque étape de la séquence, les verrous demandés.

2.Y-a-t'il des verrous incompatibles entre eux ? Est-ce que chaque verrou demandé est posé ici ?

3.Quelle est alors la séquence réélle d'éxécution ? Que se passe-t'il lors des COMMIT ?

4.Quel genre de défaut est mis en évidence ici ?

Ex. 5 — Remettre la table `enfant` dans son état initial.
A partir de deux transactions, observer et expliquer le comportement de la séquence (toujours en notant les types de verrous) :

```
U1   t1   START TRANSACTION ;
U2   t2   START TRANSACTION ;
U1   t3   ALTER TABLE enfant ADD COLUMN nom VARCHAR(20);
U2   t4   LOCK TABLE enfant IN EXCLUSIVE MODE ;
U2   t5   DROP TABLE enfant;
U1   t6   UPDATE enfant SET nom = 'Dupont' WHERE nopers = 1 ;
U2   t7   ROLLBACK ;
U1   t8   ROLLBACK ;
```

1.Quelle est alors la séquence réélle d'éxécution ?

2.Que se passe-t'il lorsqu'on change l'instruction en t4 en `LOCK TABLE IN EXCLUSIVE MODE NOWAIT;` ?

ANNEXE : Table des verrous conflictuels

Verrou demandé	Commande	Verrou déjà détenu							
		AS	RS	RX	SUX	S	SRX	X	AX
AS (Access Share)	SELECT								X
RS (Row Share)	SELECT FOR UPDATE/-FOR SHARE							X	X
RX (Row eXclusive)	UPDATE, DELETE, INSERT					X	X	X	X
SUX (Share Update eXclusive)	VACCUM, ANALYZE , CREATE INDEX CONCURRENTLY, ALTER TABLE (qques formes)				X	X	X	X	X
S (Share)	CREATE INDEX			X	X		X	X	X
SRX (Share Row eXclusive)	LOCK TABLE ...			X	X	X	X	X	X
X (eXlusive)	LOCK TABLE ...		X	X	X	X	X	X	X
AX (Access eXclusive)	ALTER TABLE, DROP TABLE, TRUNCATE, REINDEX, CLUSTER, VACUUM FULL	X	X	X	X	X	X	X	X