Transactions

Pour chaque transaction, la séquence de requêtes n'est pas interchangeable. On cherche à tester pas à pas deux sessions simultanées ; celle de Quentin et celle de Grégoire. Vous devez donc vérifier vous-même la cohérence, faire un ROLLBACK si besoin est. Pour vérifier la condition (1), on vous demande de faire deux requêtes SQL distinctes, l'une pour obtenir le nombre de places réservées (partie gauche de l'égalité), l'autre pour extraire le résultat de la partie droite de l'égalité. Ainsi vous verrez si les deux nombres obtenus sont égaux ou pas. Le but est de réserver des places : Quentin réserve 2 places pour Harry Potter, et Grégoire en réservé 5. On a donc deux exécutions de la procédure de réservation, l'une à dérouler dans la session 1, l'autre dans la session 2. On se place en mode sérialisable.

1. Commencez par écrire les requêtes effectuées par chaque transaction : a) La réservation de Quentin : T1 b) La réservation de Grégoire : T2 Pensez à vérifier la cohérence (1) à réaliser dans l'une et l'autre session (dans T1 et T2). T1 :

```
begin transaction;
set session transaction isolation level serializable;
insert into reservation values ('Quentin', 'Happy Potter', 2);
update spectacle set nbplaces = nbplaces -2 where noms='Happy Potter';
update client set solde = solde - 2*20 where client='Quentin';
commit
```

T2 :

```
begin transaction;
set session transaction isolation level serializable;
insert into reservation values ('Gregoire', 'Happy Potter', 5);
update spectacle set nbplaces = nbplaces -5 where noms='Happy Potter';
update client set solde = solde - 2*20 where client='Gregoire';
commit;
```

2. Effectuez les deux transactions l'une après l'autre (en série). Quel est l'état de la base à la fin?

```
--session 1
tp2db=> begin transaction;
BEGIN
tp2db=*> set session transaction isolation level serializable;
SET
tp2db=*> insert into reservation values ('Quentin', 'Happy Potter', 2);
INSERT 0 1
tp2db=*> update spectacle set nbplaces = nbplaces -2 where noms='Happy
Potter';
UPDATE 1
tp2db=*> commit;
COMMIT
```

```
tp2db=> select * from spectacle ;
     noms     | nbplaces | nbplaceslibres | tarif
--------------+----------+----------------+-------
 Happy Potter |      248 |            250 | 20.00
(1 row)
--session2
tp2db=> begin transaction;
BEGIN
tp2db=*> set session transaction isolation level serializable;
SET
tp2db=*> insert into reservation values ('Gregoire', 'Happy Potter', 5);
INSERT 0 1
tp2db=*> update spectacle set nbplaces = nbplaces -5 where noms='Happy
Potter';
UPDATE 1
tp2db=*> commit;
COMMIT;
tp2db=> select * from reservation ;
   nomc   |     noms     | nbplacesreservees
----------+--------------+-------------------
 Quentin  | Happy Potter |                 2
 Gregoire | Happy Potter |                 5
(2 rows)

tp2db=> select * from spectacle;
     noms     | nbplaces | nbplaceslibres | tarif
--------------+----------+----------------+-------
 Happy Potter |      243 |            250 | 20.00
(1 row)
```

3. Réinitialiser vos tables.

```
tp2db=> truncate table reservation ;
TRUNCATE TABLE
tp2db=> select * from reservation;
 nomc | noms | nbplacesreservees
------+------+-------------------
(0 rows)
tp2db=> truncate table client cascade;
NOTICE:  truncate cascades to table "reservation"
TRUNCATE TABLE
tp2db=> select * from client;
 nomc | solde
------+-------
(0 rows)tp2db=> insert into client values ('Quentin', 50);INSERT 0 1
tp2db=> insert into client values ('Gregoire', 50);INSERT 0 1
tp2db=> select * from client;
   nomc   | solde
----------+-------
 Quentin  |    50
 Gregoire |    50
(2 rows)
```

```
p2db=> update spectacle set nbplaces = 250 where noms = 'Happy Potter';
UPDATE 1
tp2db=> select * from spectacle;
     noms     | nbplaces | nbplaceslibres | tarif
--------------+----------+----------------+-------
 Happy Potter |      250 |            250 | 20.00
(1 row)
```

4. Testez par la suite, une histoire (une exécution, un entrelacement) ou plus d'une, sous forme de suite d'opérations grâce aux transactions que vous venez de produire. Analysez les situations.

```
tp2db=> begin transaction ;
BEGIN
tp2db=*> set session transaction isolation level serializable;
SET
tp2db=*> insert into reservation values ('Quentin', 'Happy Potter', 2);
INSERT 0 1
tp2db=*> update spectacle set nbplaces = nbplaces -2 where noms='Happy
Potter';
UPDATE 1
tp2db=*> commit;
COMMIT

tp2db=> begin transaction ;
BEGIN
tp2db=*> set session transaction isolation level serializable;
SET
tp2db=*> insert into reservation values ('Gregoire', 'Happy Potter', 5);
INSERT 0 1
tp2db=*> update spectacle set nbplaces=nbplaces-5 where noms='Happy
Potter';
ERROR:  could not serialize access due to concurrent update
tp2db=!>
```

5. Choisissez ensuite une des commandes ci-dessous. set session transaction isolation level read uncommitted; set session transaction isolation level read committed; set session transaction isolation level repeatable read;

set session transaction isolation level read uncommitted;

```
--session1
tp2db=*> set session transaction isolation level read uncommitted;
SET
tp2db=*> insert into reservation values ('Quentin', 'Happy Potter', 2);
INSERT 0 1
tp2db=*> update spectacle set nbplaces= 250-2 where noms= 'Happy Potter';
UPDATE 1
tp2db=*> commit;
```

```
COMMIT
tp2db=>
```

```
--session2
tp2db=> begin transaction;
BEGIN
tp2db=*>
tp2db=*> set session transaction isolation level read uncommitted;
SET
tp2db=*> insert into reservation values ('Gregoire', 'Happy Potter', 5);
INSERT 0 1
tp2db=*> select * from spectacle ;
     noms     | nbplaces | nbplaceslibres | tarif
--------------+----------+----------------+-------
 Happy Potter |      250 |            250 | 20.00
(1 row)

tp2db=*> select * from spectacle ;
     noms     | nbplaces | nbplaceslibres | tarif
--------------+----------+----------------+-------
 Happy Potter |      248 |            250 | 20.00
(1 row)
```

Postgres a une politique anti read uncommitted;

set session transaction isolation level read committed;

```
session1> begin transaction;
session1> set session transaction isolation level read committed;
session2> begin transaction;
session2> set session transaction isolation level read committed;
session1> select * from reservation;
         nomc | noms | nbplacesreservees
         ------+------+-------------------
         (0 rows)
session2> insert into reservation values ('Quentin', 'Happy Potter', 2);
session2> commit;
session1> select * from reservation;
        select * from reservation ;
        nomc   |     noms     | nbplacesreservees
        ----------+--------------+-------------------
        Gregoire | Happy Potter |                 5
        (1 row)
```

lecture sale ce n'est pas le cas avec le mode serializable

set session transaction isolation level repeatable read ;

```
session1=> begin transaction ;
BEGIN
session1=*> set session transaction isolation level repeatable read;
SET

session2=> begin transaction ;
BEGIN
session2=*> set session transaction isolation level repeatable read;
SET

session1=*> select * from reservation ;
 nomc | noms | nbplacesreservees
------+------+-------------------
(0 rows)

session2=*> select * from reservation ;
 nomc | noms | nbplacesreservees
------+------+-------------------
(0 rows)

session2=*> insert into reservation values ('Quentin', 'Harry Potter', 5);
INSERT 0 1

session2=*> select * from reservation ;
   nomc   |     noms     | nbplacesreservees
----------+--------------+-------------------
 Gregoire | Harry Potter |                 5
(1 row)

session2=*> commit;
COMMIT

session1=*> select * from reservation ;
  nomc   |     noms     | nbplacesreservees
---------+--------------+-------------------
(0 row)

session1=*> insert into reservation values ('Quentin', 'Harry Potter', 2);
INSERT 0 1

session1=*> select * from reservation ;
   nomc   |     noms     | nbplacesreservees
----------+--------------+-------------------
 Quentin  | Harry Potter |                 2
(1 row)

session2=*> commit;
COMMIT

session1/session2=> select * from reservation ;
   nomc   |     noms     | nbplacesreservees
----------+--------------+-------------------
 Gregoire | Harry Potter |                 5
```

```
 Quentin  | Harry Potter |                 2
(2 rows)
```

Partie 3