INF3710 –Fichiers et Bases de données

Hiver 2019

TP No. 4

Groupe 4

1847125 – Julien Legault

1846754 – Kevin Pastor

Soumis à : Manel Grichi

24 mars 2019

Partie 2

1.1

1. $\sigma_{annee=2010}(Spectacle)$
2. $\sigma_{age<20 \vee age>29}(Danseur)$
3. $\pi_{nom}(\sigma_{nationalite=canadien}(DirecteurArtistique))$
4. $\pi_{nom,titre}(Danseur \bowtie (Spectacle \bowtie Performance))$
5. $\pi_{nom,annee}(\sigma_{role=cygne}(Danseur \bowtie (Spectacle \bowtie Performance)))$
6. $Danseur \ltimes (Spectacle \bowtie Performance))$
7. $SpectaclesPhilippe \leftarrow \sigma_{nom=Philippe}(Danseur \bowtie (Spectacle \bowtie Performance)))$

   $SpectaclesKate \leftarrow \sigma_{nom=Kate}(Danseur \bowtie (Spectacle \bowtie Performance)))$

   $\pi_{titre}(SpectaclesPhilippe \cap SpectaclesKate)$

1.2

8. $\rho_R(AgeMoyen)(\tau_{AVG\ age}(Danseur))$

   $SELECT\ AVG(age)\ AS\ AgeMoyen\ FROM\ Danseur;$

9. $PerfDans \leftarrow Performance \bowtie Danseur$

   $_{nom}\pi_{nom}(PerfDans - \sigma_{nom=Lucie\ Tremblay}(PerfDans))$

   $SELECT\ nom\ FROM\ Danseur\ d, Performance\ p$
   $WHERE\ d.DanseurId = p.DanseurId$
   $GROUP\ BY\ nom$
   $EXCEPT$
   $SELECT\ nom\ FROM\ Danseur\ d, Performance\ p$
   $WHERE\ d.DanseurId = p.DanseurId$
   $AND\ d.nom = 'Lucie\ Tremblay'$
   $GROUP\ BY\ nom;$

10. $\rho_R(nbSpectacle)(\tau_{COUNT\ SpectacleId}(Performance \bowtie \sigma_{id=1}(Danseur)))$

    $SELECT\ COUNT(*)\ AS\ nbSpectacle\ FROM\ Danseur\ d, Performance\ p$
    $WHERE\ d.DanseurId = p.DanseurId$
    $AND\ d.danseurId = 1;$

11. $PerfDans \leftarrow Danseur ⟕ Performance$
    $\pi_{DanseurId,nom,nationalite,age,SpectacleId}(PerfDans)$

    $SELECT\ d.DanseurId, d.nom, d.nationalite, d.age, p.SpectacleId$
    $FROM\ Danseur\ d$
    $LEFT\ JOIN\ Performance\ p\ ON\ p.DanseurId = d.DanseurId;$

12. $\pi_{categorie,nbSpectacles}(\rho_R(count\ as\ nbSpectacles)({}_{categorie}\tau_{COUNT\ SpectacleId}(Spectacle)))$

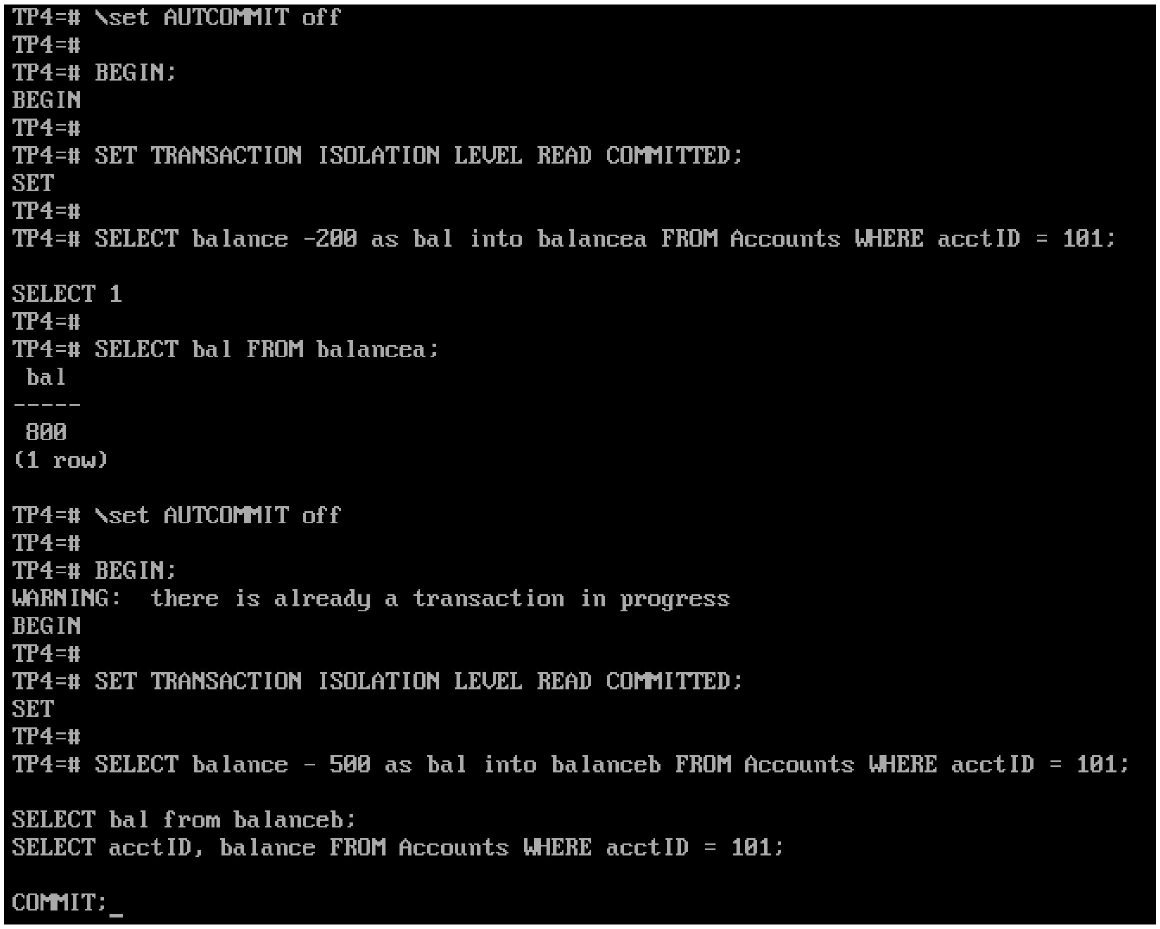
    $SELECT\ COUNT(*)\ AS\ nbSpectacles, categorie$
    $FROM\ Spectacle\ GROUP\ BY\ categorie;$

13. $Danseur - \pi_{DanseurId,nom,nationalite,age}({}_{DanseurId}(Danseur \bowtie Performance))$

    $SELECT * FROM\ Danseur$
    $EXCEPT$
    $SELECT\ d.DanseurId, d.nom, d.nationalite, d.age\ FROM\ Danseur\ d,$
    $NATURAL\ JOIN\ Performance\ p$
    $GROUP\ BY\ DanseurId;$

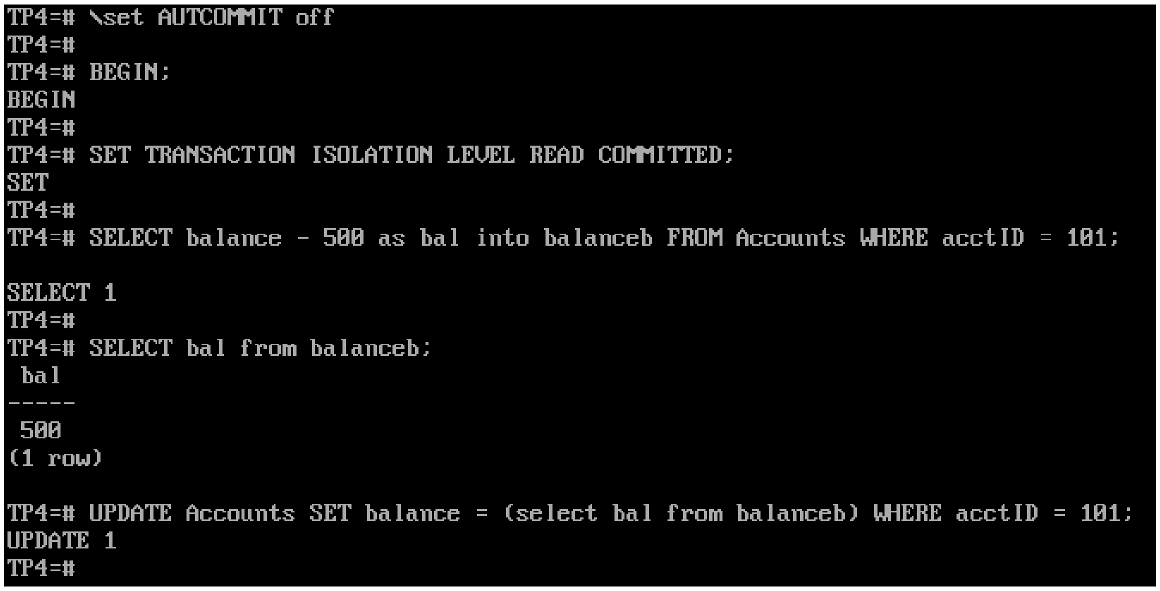
Partie 3

Q1-a



```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
TP4=#
TP4=# SELECT balance -200 as bal into balancea FROM Accounts WHERE acctID = 101;

SELECT 1
TP4=#
TP4=# SELECT bal FROM balancea;
 bal
-----
 800
(1 row)

TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
WARNING:  there is already a transaction in progress
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
TP4=#
TP4=# SELECT balance - 500 as bal into balanceb FROM Accounts WHERE acctID = 101;

SELECT bal from balanceb;
SELECT acctID, balance FROM Accounts WHERE acctID = 101;

COMMIT;_
```

Transaction A

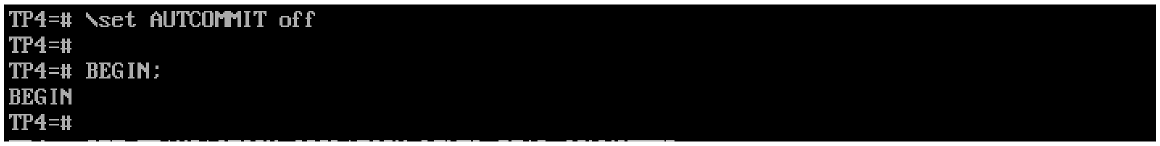


```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
TP4=#
TP4=# SELECT balance - 500 as bal into balanceb FROM Accounts WHERE acctID = 101;

SELECT 1
TP4=#
TP4=# SELECT bal from balanceb;
 bal
-----
 500
(1 row)

TP4=# UPDATE Accounts SET balance = (select bal from balanceb) WHERE acctID = 101;
UPDATE 1
TP4=#
```

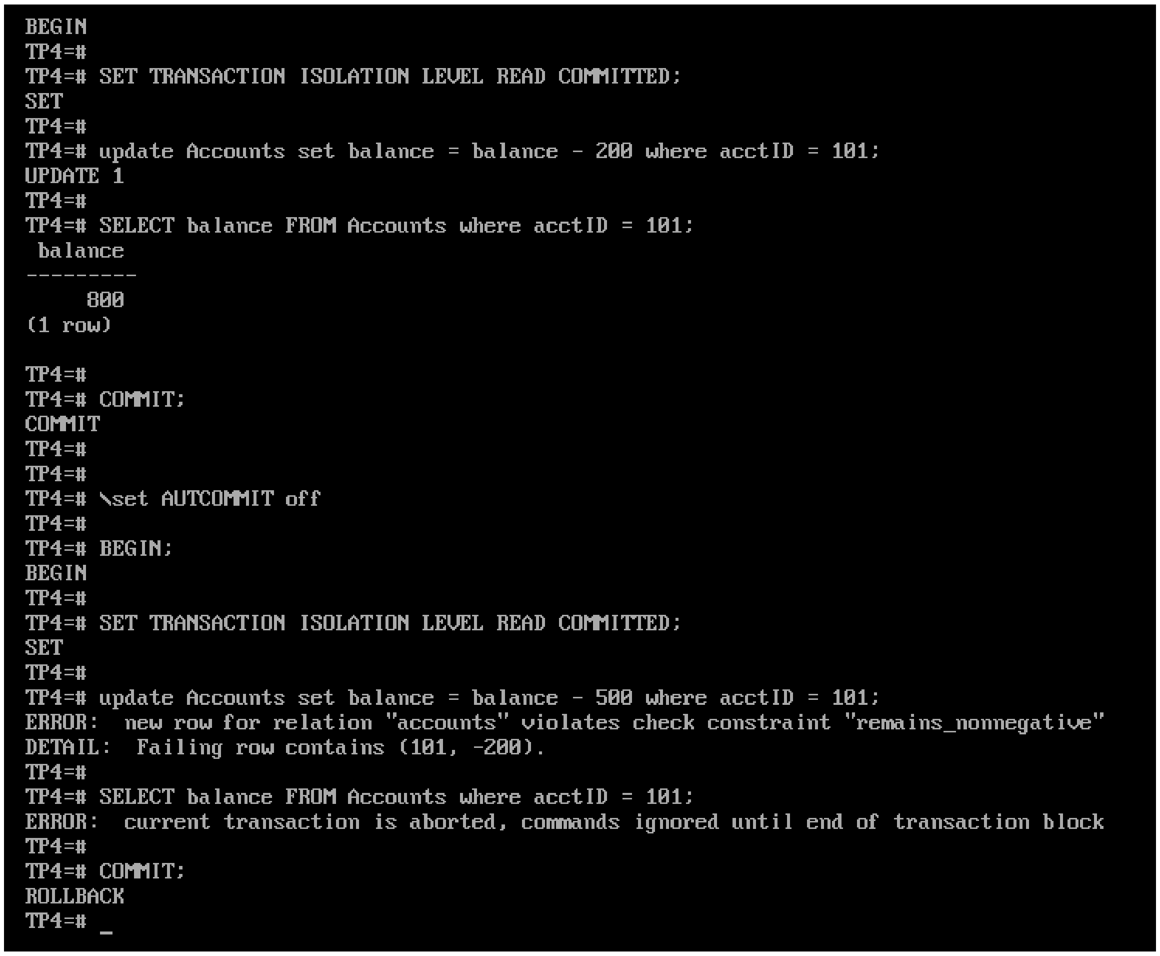
Transaction B

La deuxième transaction A est mise en attente puisqu'aucun COMMIT n'est effectué entre les deux begins.

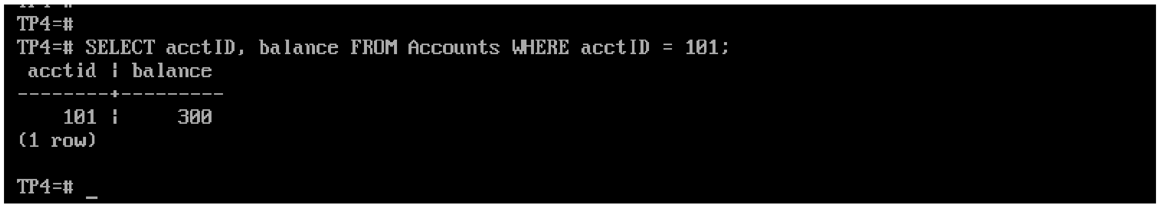
Q1-b



```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
```

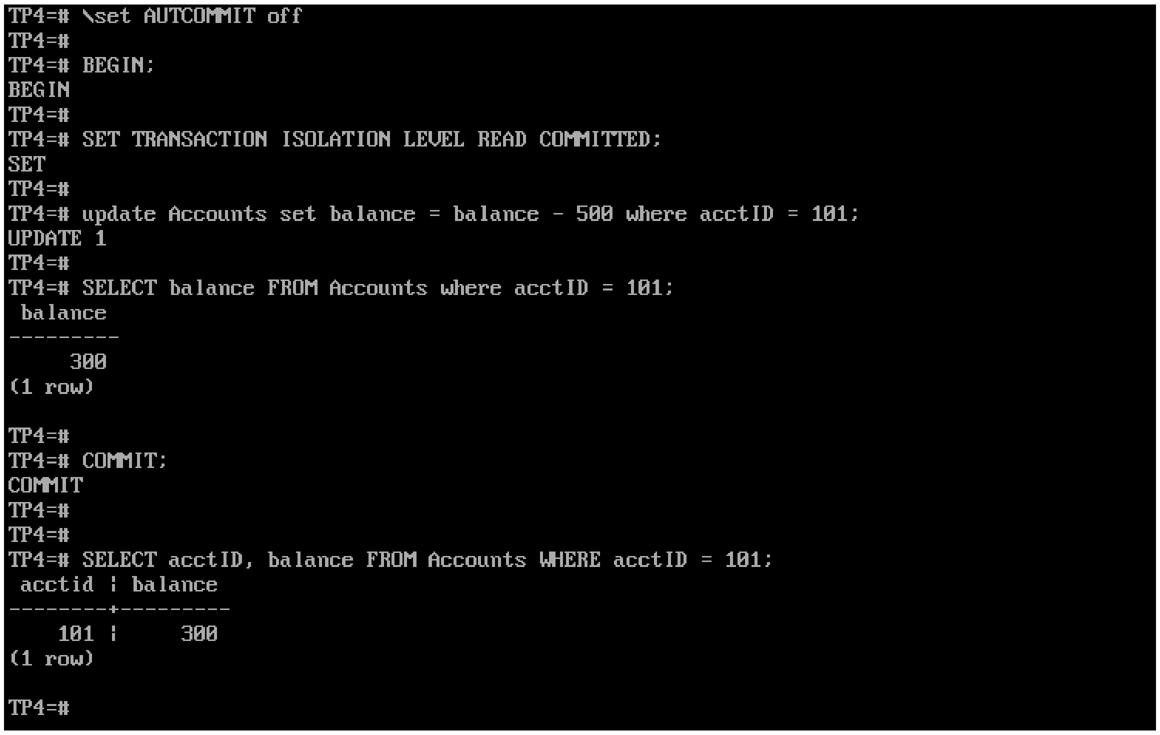


```
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
TP4=#
TP4=# update Accounts set balance = balance - 200 where acctID = 101;
UPDATE 1
TP4=#
TP4=# SELECT balance FROM Accounts where acctID = 101;
 balance
---------
     800
(1 row)

TP4=#
TP4=# COMMIT;
COMMIT
TP4=#
TP4=#
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
TP4=#
TP4=# update Accounts set balance = balance - 500 where acctID = 101;
ERROR:  new row for relation "accounts" violates check constraint "remains_nonnegative"
DETAIL:  Failing row contains (101, -200).
TP4=#
TP4=# SELECT balance FROM Accounts where acctID = 101;
ERROR:  current transaction is aborted, commands ignored until end of transaction block
TP4=#
TP4=# COMMIT;
ROLLBACK
TP4=# _
```



```
TP4=#
TP4=# SELECT acctID, balance FROM Accounts WHERE acctID = 101;
 acctid | balance
--------+---------
    101 |     300
(1 row)

TP4=# _
```

Transaction A

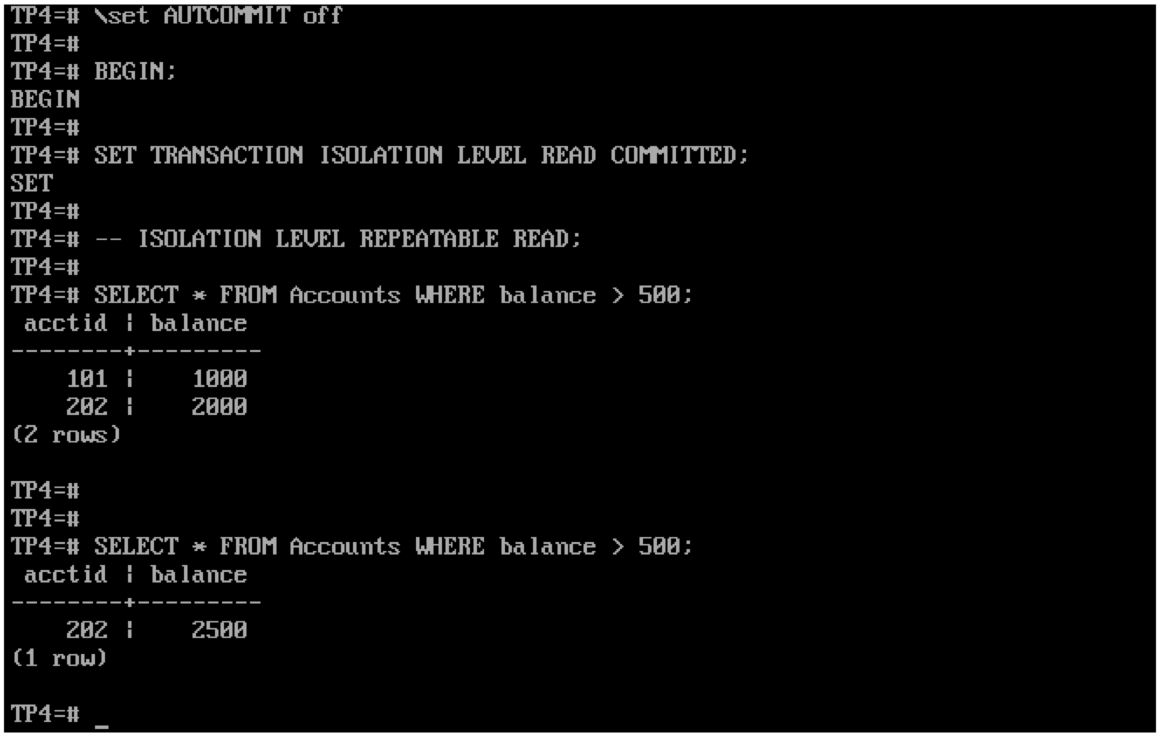


```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
TP4=#
TP4=# update Accounts set balance = balance - 500 where acctID = 101;
UPDATE 1
TP4=#
TP4=# SELECT balance FROM Accounts where acctID = 101;
 balance
---------
     300
(1 row)

TP4=#
TP4=# COMMIT;
COMMIT
TP4=#
TP4=#
TP4=# SELECT acctID, balance FROM Accounts WHERE acctID = 101;
 acctid | balance
--------+---------
    101 |     300
(1 row)

TP4=#
```

Transaction B

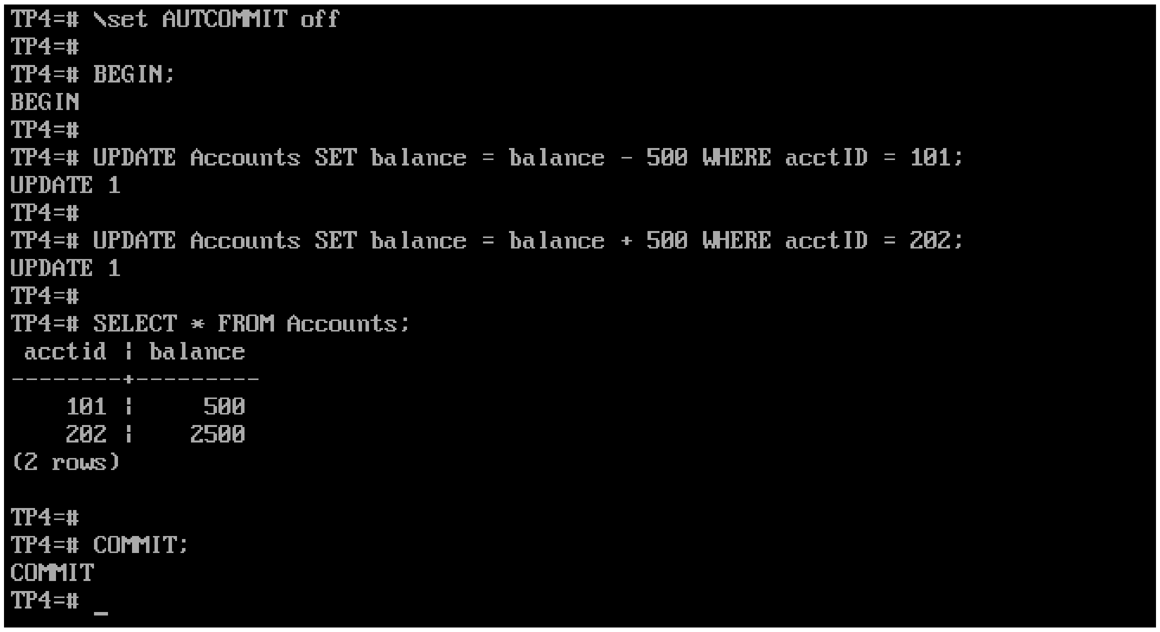
Nous avons retiré les variables intermédiaires balanceA et balanceB afin que les transactions s'effectuent sur une seule variable commune. Nous avons plutôt effectué les modifications sur l'attribut de la table accounts

Q2-a



```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
TP4=#
TP4=# -- ISOLATION LEVEL REPEATABLE READ;
TP4=#
TP4=# SELECT * FROM Accounts WHERE balance > 500;
 acctid | balance
--------+---------
    101 |    1000
    202 |    2000
(2 rows)

TP4=#
TP4=#
TP4=# SELECT * FROM Accounts WHERE balance > 500;
 acctid | balance
--------+---------
    202 |    2500
(1 row)

TP4=# _
```

Transaction A

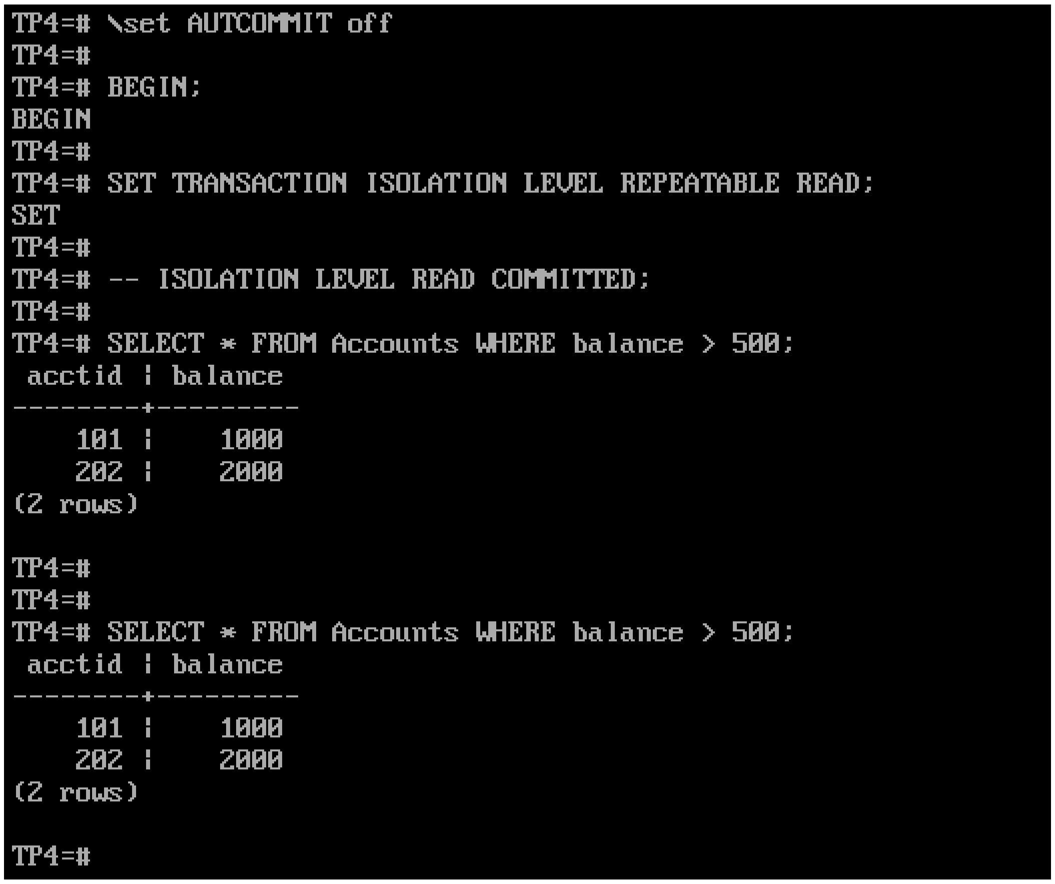


```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101;
UPDATE 1
TP4=#
TP4=# UPDATE Accounts SET balance = balance + 500 WHERE acctID = 202;
UPDATE 1
TP4=#
TP4=# SELECT * FROM Accounts;
 acctid | balance
--------+---------
    101 |     500
    202 |    2500
(2 rows)

TP4=#
TP4=# COMMIT;
COMMIT
TP4=# _
```

Transaction B

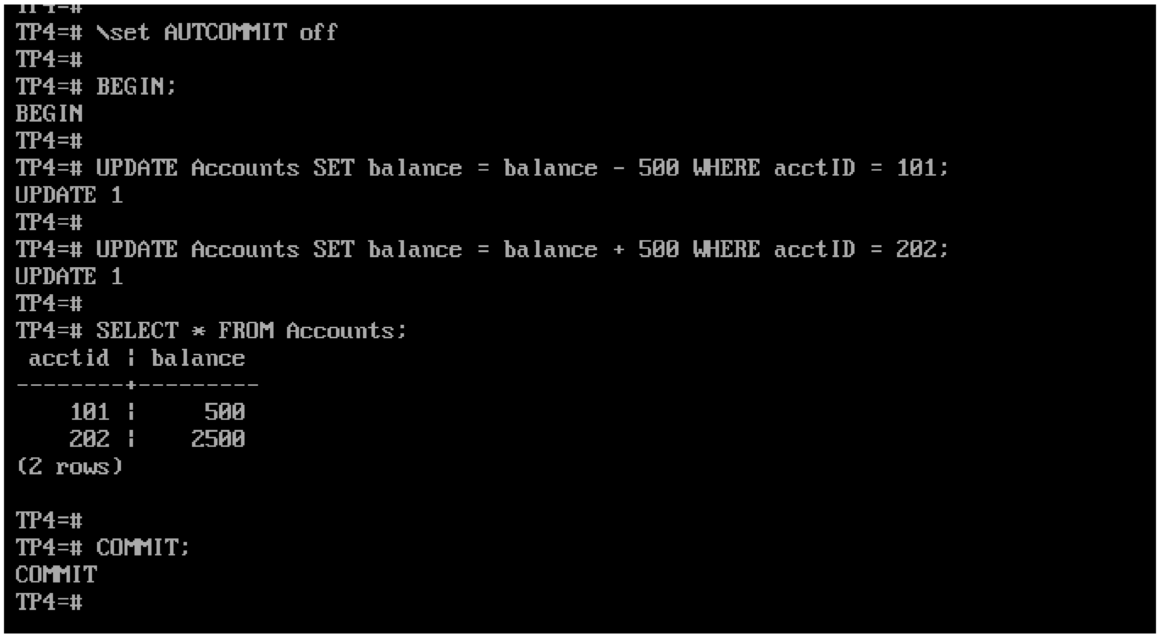
Il ne semble pas avoir de problème, la balance des deux comptes est correctement mis à jour. Puisqu'en niveau d'isolation read commited, la session A a acccès aux nouvelles données des commits de la session B, la session A aura les données à jour.

Q2-b



```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
TP4=#
TP4=# -- ISOLATION LEVEL READ COMMITTED;
TP4=#
TP4=# SELECT * FROM Accounts WHERE balance > 500;
 acctid | balance
--------+---------
    101 |    1000
    202 |    2000
(2 rows)

TP4=#
TP4=#
TP4=# SELECT * FROM Accounts WHERE balance > 500;
 acctid | balance
--------+---------
    101 |    1000
    202 |    2000
(2 rows)

TP4=#
```

Transaction A

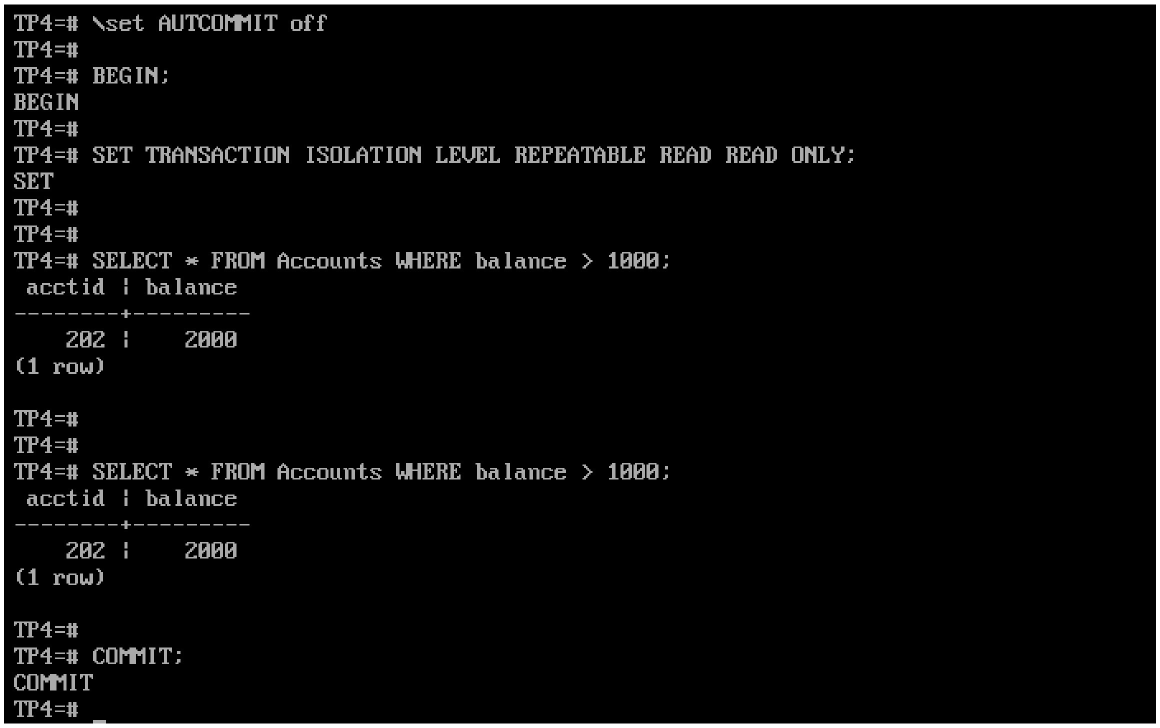


```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101;
UPDATE 1
TP4=#
TP4=# UPDATE Accounts SET balance = balance + 500 WHERE acctID = 202;
UPDATE 1
TP4=#
TP4=# SELECT * FROM Accounts;
 acctid | balance
--------+---------
    101 |     500
    202 |    2500
(2 rows)

TP4=#
TP4=# COMMIT;
COMMIT
TP4=#
```

Transaction B

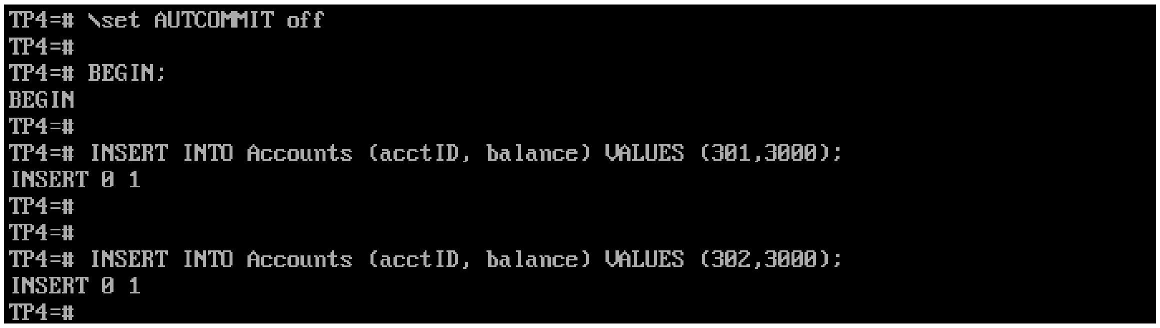
Puisqu'en niveau d'isolation repeatable read, la session A est isolé des commits de la session B, la session A aura toujours les mêmes données.

Q2-c



```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SET TRANSACTION ISOLATION LEVEL REPEATABLE READ READ ONLY;
SET
TP4=#
TP4=#
TP4=# SELECT * FROM Accounts WHERE balance > 1000;
 acctid | balance
--------+---------
    202 |    2000
(1 row)

TP4=#
TP4=#
TP4=# SELECT * FROM Accounts WHERE balance > 1000;
 acctid | balance
--------+---------
    202 |    2000
(1 row)

TP4=#
TP4=# COMMIT;
COMMIT
TP4=# _
```

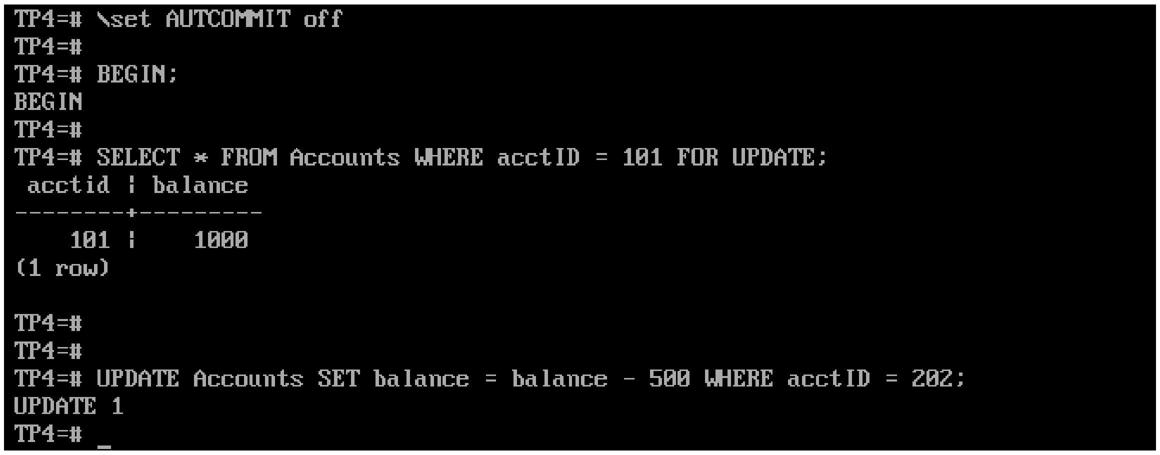
Transaction A


```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# INSERT INTO Accounts (acctID, balance) VALUES (301,3000);
INSERT 0 1
TP4=#
TP4=#
TP4=# INSERT INTO Accounts (acctID, balance) VALUES (302,3000);
INSERT 0 1
TP4=#
```

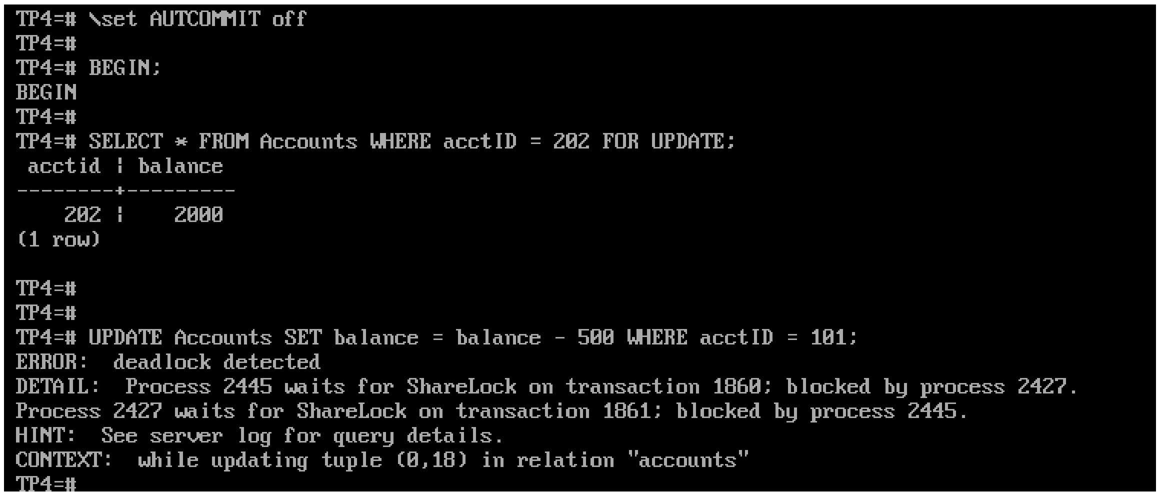
Transaction B

Puisque le niveau d'isolation est en repeatable read, la session A ne voit pas les nouvelles données de la session B. De plus, La session B ne fait aucun commits ce qui empêcherait la session A de voir les ajouts même en niveau d'isolation read commited.

Q3


```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SELECT * FROM Accounts WHERE acctID = 101 FOR UPDATE;
 acctid | balance
--------+---------
    101 |    1000
(1 row)

TP4=#
TP4=#
TP4=# UPDATE Accounts SET balance = balance - 500 WHERE acctID = 202;
UPDATE 1
TP4=# _
```

Transaction A


```
TP4=# \set AUTCOMMIT off
TP4=#
TP4=# BEGIN;
BEGIN
TP4=#
TP4=# SELECT * FROM Accounts WHERE acctID = 202 FOR UPDATE;
 acctid | balance
--------+---------
    202 |    2000
(1 row)

TP4=#
TP4=#
TP4=# UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101;
ERROR:  deadlock detected
DETAIL:  Process 2445 waits for ShareLock on transaction 1860; blocked by process 2427.
Process 2427 waits for ShareLock on transaction 1861; blocked by process 2445.
HINT:  See server log for query details.
CONTEXT:  while updating tuple (0,18) in relation "accounts"
TP4=#
```

Transaction B