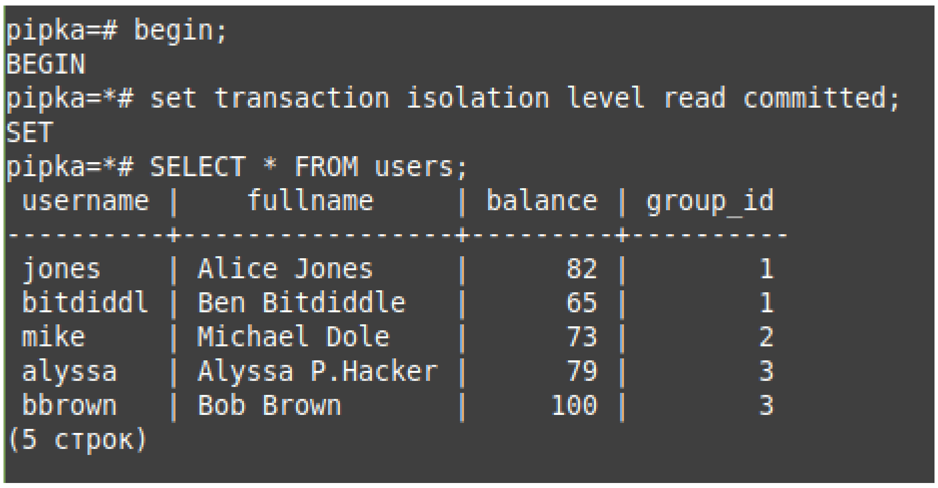
Exercise 2 results. Dzhemilya Gizutdinova

Test with Read committed

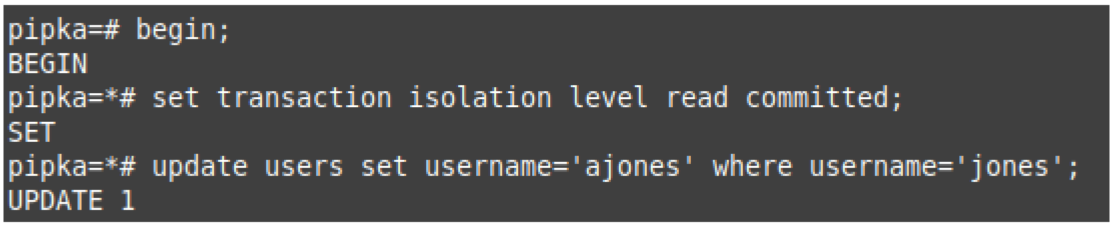
link onto my conspectum: https://docs.google.com/document/d/1hK0tPNXfMBKSe7fh0LwFmiGi2lJscVaEMjrfABvhxYg/edit#

STEP 1 TERMINAL 1: Start a transaction and display the accounts information



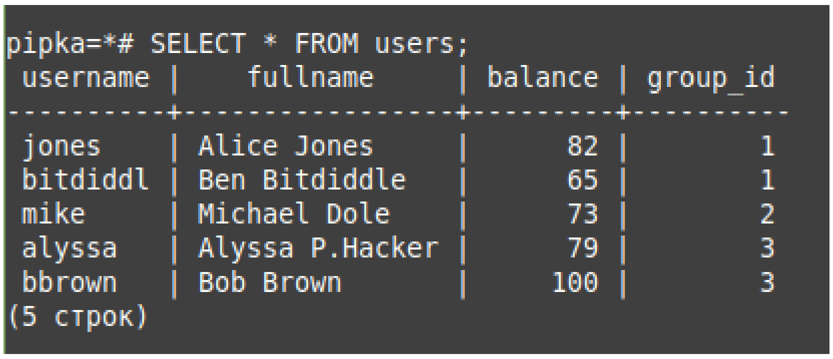
```
pipka=# begin;
BEGIN
pipka=*# set transaction isolation level read committed;
SET
pipka=*# SELECT * FROM users;
 username |    fullname      | balance | group_id
----------+-----------------+---------+----------
 jones    | Alice Jones      |      82 |        1
 bitdiddl | Ben Bitdiddle    |      65 |        1
 mike     | Michael Dole     |      73 |        2
 alyssa   | Alyssa P.Hacker |      79 |        3
 bbrown   | Bob Brown        |     100 |        3
(5 строк)
```

STEP 2 TERMINAL 2: Start a transaction and update the username for “Alice Jones” as “ajones”



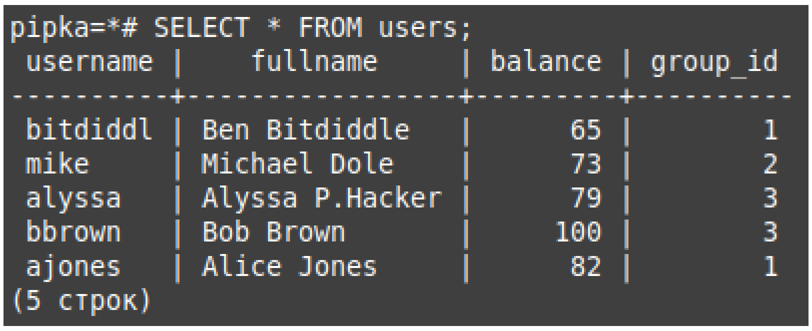
```
pipka=# begin;
BEGIN
pipka=*# set transaction isolation level read committed;
SET
pipka=*# update users set username='ajones' where username='jones';
UPDATE 1
```

STEP 3 TERMINAL 1: Display again the accounts table



```
pipka=*# SELECT * FROM users;
 username |    fullname      | balance | group_id
----------+-----------------+---------+----------
 jones    | Alice Jones      |      82 |        1
 bitdiddl | Ben Bitdiddle    |      65 |        1
 mike     | Michael Dole     |      73 |        2
 alyssa   | Alyssa P.Hacker |      79 |        3
 bbrown   | Bob Brown        |     100 |        3
(5 строк)
```

STEP 4 TERMINAL 2: Display again the accounts table



```
pipka=*# SELECT * FROM users;
 username |     fullname     | balance | group_id
----------+-----------------+---------+----------
 bitdiddl | Ben Bitdiddle    |      65 |        1
 mike     | Michael Dole     |      73 |        2
 alyssa   | Alyssa P.Hacker |      79 |        3
 bbrown   | Bob Brown        |     100 |        3
 ajones   | Alice Jones      |      82 |        1
(5 строк)
```

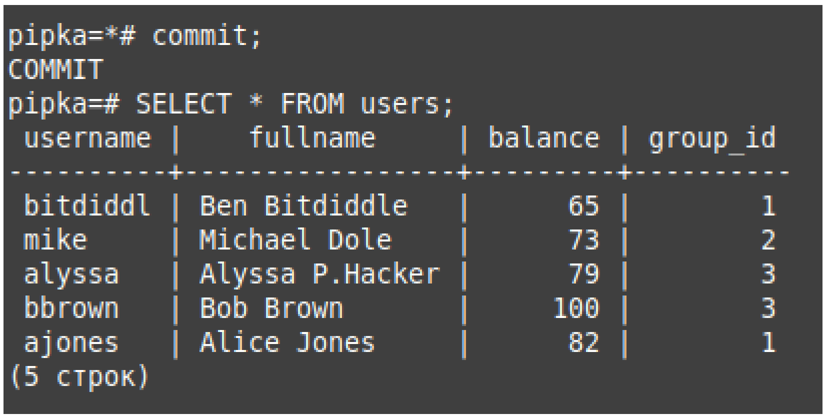
Do both terminals show the same information? Explain the reason

ANSWER:

No changes in terminal 1, since we didn't commit anything in the terminal 2. Because READ COMMITTED guarantees that any data is read is committed at the moment it is read.

STEP 5 TERMINAL 2: Commit the changes and compare again both sessions.

terminal 2:



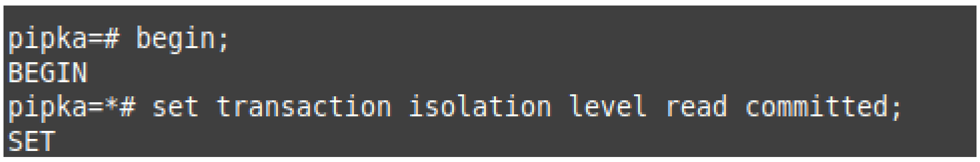
```
pipka=*# commit;
COMMIT
pipka=# SELECT * FROM users;
 username |     fullname     | balance | group_id
----------+-----------------+---------+----------
 bitdiddl | Ben Bitdiddle    |      65 |        1
 mike     | Michael Dole     |      73 |        2
 alyssa   | Alyssa P.Hacker |      79 |        3
 bbrown   | Bob Brown        |     100 |        3
 ajones   | Alice Jones      |      82 |        1
(5 строк)
```

terminal 1:

Answer:

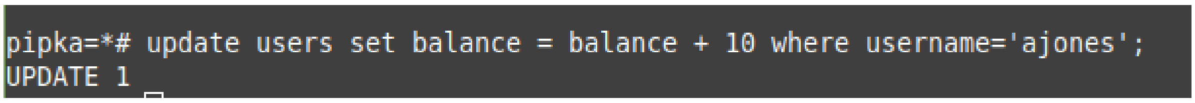
After commit, T1 and T2 shows the same data. As was said above, READ COMMITTED guarantees that any data is read is committed at the moment it is read. Thus, first terminal display the data which was committed in second terminal.

STEP 6 TERMINAL 2: Start a new transaction



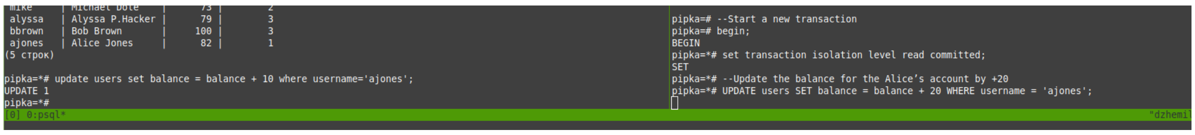
```
pipka=# begin;
BEGIN
pipka=*# set transaction isolation level read committed;
SET
```

STEP 7 TERMINAL 1: Update the balance for the Alice’s account by +10.



```
pipka=*# update users set balance = balance + 10 where username='ajones';
UPDATE 1
```

STEP 8 TERMINAL 2: Update the balance for the Alice’s account by +20



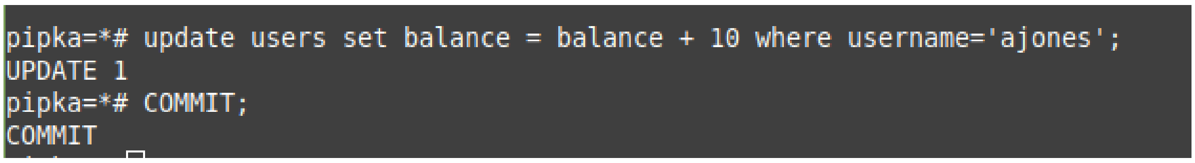
```
 mike     | Michael Dole     |      73 |        2
 alyssa   | Alyssa P.Hacker  |      79 |        3
 bbrown   | Bob Brown        |     100 |        3
 ajones   | Alice Jones      |      82 |        1
(5 строк)

pipka=*# update users set balance = balance + 10 where username='ajones';
UPDATE 1
pipka=*#
[0] 0:psql*
```

```
pipka=# --Start a new transaction
pipka=# begin;
BEGIN
pipka=*# set transaction isolation level read committed;
SET
pipka=*# --Update the balance for the Alice’s account by +20
pipka=*# UPDATE users SET balance = balance + 20 WHERE username = 'ajones';
"dzhemil
```

Explain the output form the second terminal

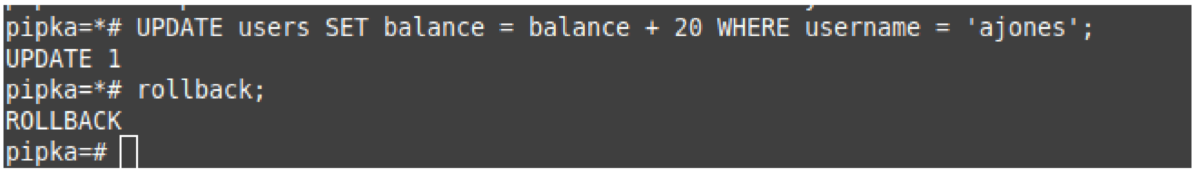
After update statement T2 starts blocking state and waits for commit of T2 transaction since they both holds the same record changed. Situation is similar to a deadlock.

STEP 9 TERMINAL 1: Commit the changes.



```
pipka=*# update users set balance = balance + 10 where username='ajones';
UPDATE 1
pipka=*# COMMIT;
COMMIT
```

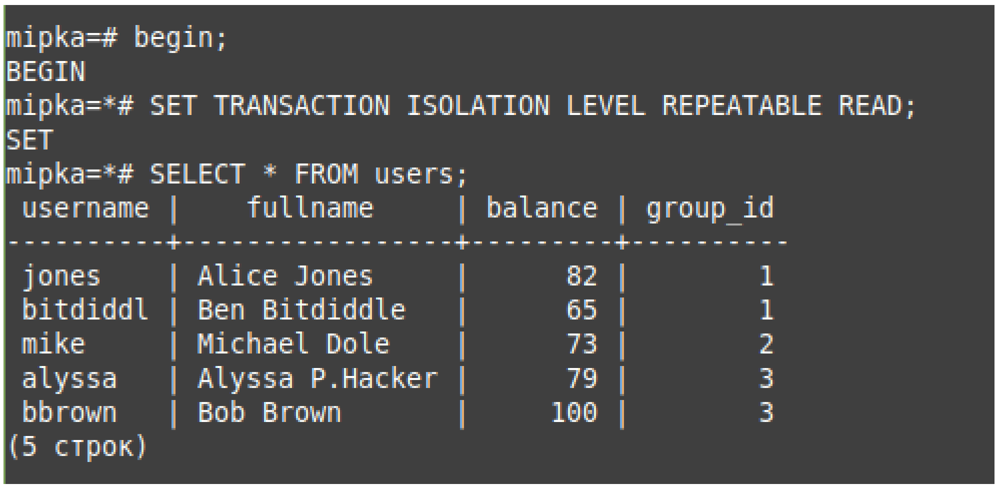
STEP 10 TERMINAL 2: Rollback



```
pipka=*# UPDATE users SET balance = balance + 20 WHERE username = 'ajones';
UPDATE 1
pipka=*# rollback;
ROLLBACK
pipka=#
```

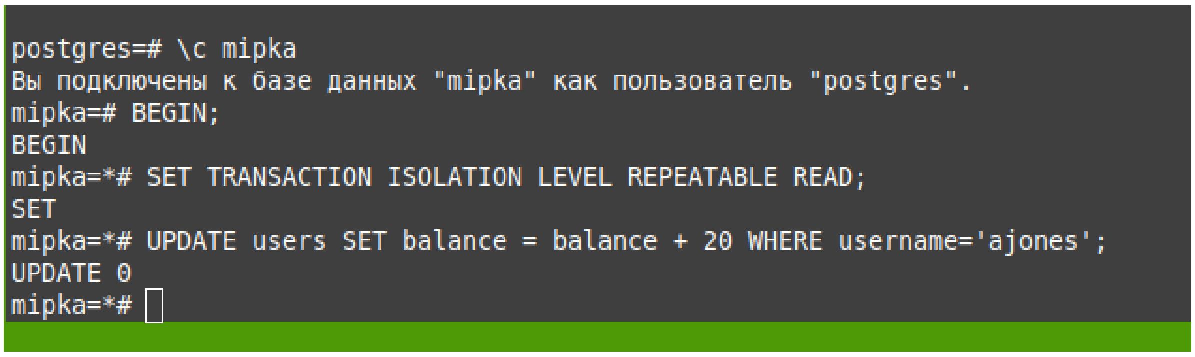
Test with Read committed

STEP 1 TERMINAL 1: Start a transaction and display the accounts information



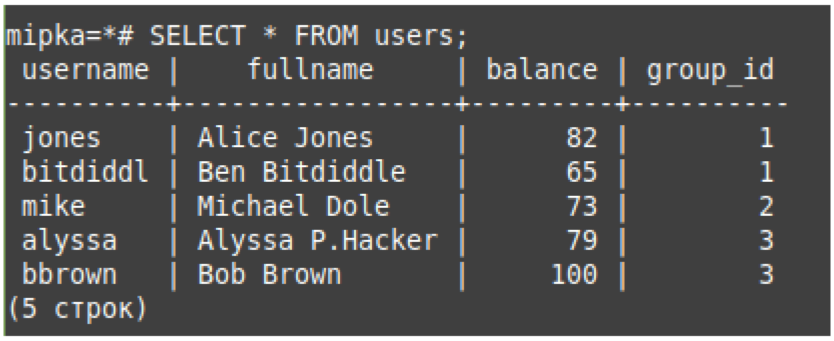
```
mipka=# begin;
BEGIN
mipka=*# SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
mipka=*# SELECT * FROM users;
 username |    fullname     | balance | group_id
----------+-----------------+---------+----------
 jones    | Alice Jones     |      82 |        1
 bitdiddl | Ben Bitdiddle   |      65 |        1
 mike     | Michael Dole    |      73 |        2
 alyssa   | Alyssa P.Hacker |      79 |        3
 bbrown   | Bob Brown       |     100 |        3
(5 строк)
```

STEP 2 TERMINAL 2: Start a transaction and update the username for "Alice Jones" as "ajones"



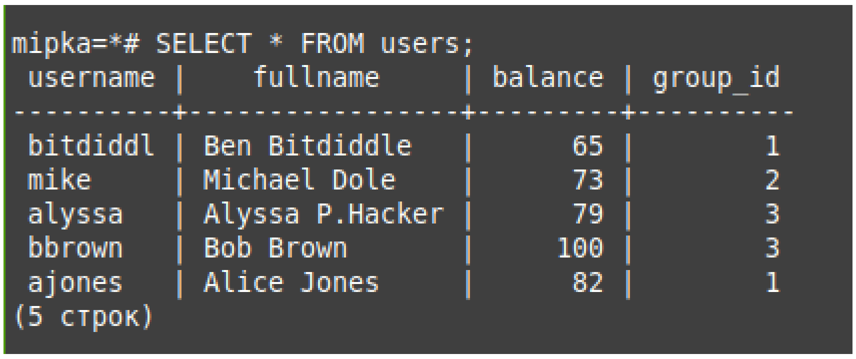
```
postgres=# \c mipka
Вы подключены к базе данных "mipka" как пользователь "postgres".
mipka=# BEGIN;
BEGIN
mipka=*# SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
mipka=*# UPDATE users SET balance = balance + 20 WHERE username='ajones';
UPDATE 0
mipka=*#
```

STEP 3 TERMINAL 1: Display again the accounts table



```
mipka=*# SELECT * FROM users;
 username |     fullname      | balance | group_id
----------+-------------------+---------+----------
 jones    | Alice Jones       |      82 |        1
 bitdiddl | Ben Bitdiddle     |      65 |        1
 mike     | Michael Dole      |      73 |        2
 alyssa   | Alyssa P.Hacker   |      79 |        3
 bbrown   | Bob Brown         |     100 |        3
(5 строк)
```

STEP 4 TERMINAL 2: Display again the accounts table



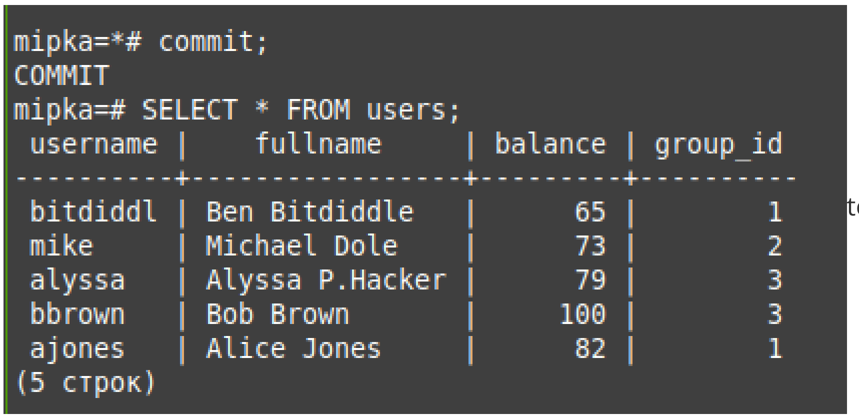
```
mipka=*# SELECT * FROM users;
 username |     fullname      | balance | group_id
----------+-------------------+---------+----------
 bitdiddl | Ben Bitdiddle     |      65 |        1
 mike     | Michael Dole      |      73 |        2
 alyssa   | Alyssa P.Hacker   |      79 |        3
 bbrown   | Bob Brown         |     100 |        3
 ajones   | Alice Jones       |      82 |        1
(5 строк)
```

Do both terminals show the same information? Explain the reason

ANSWER:

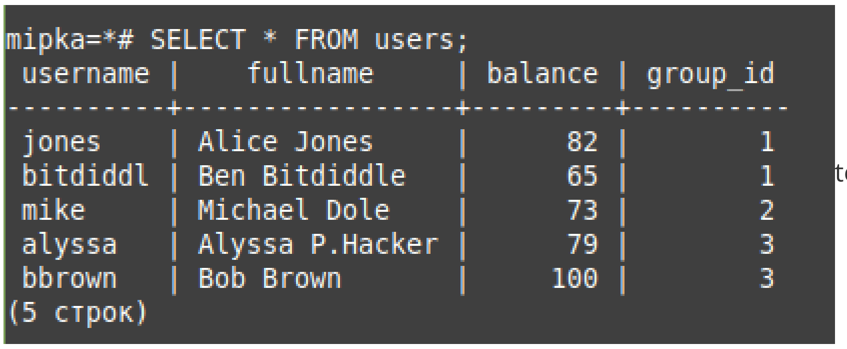
No, T1 terminal shows still shows `'jones'`, while T2 shows updated value `'ajones'`. It's because repeatable read as read committed guarantees that any data is read is committed at the moment it is read, but transaction in T2 is not committed yet. After commit, T1 still shows old data because it keeps all records that were read in current transaction. T2 shows new data.

STEP 5 TERMINAL 2: Commit the changes and compare again both sessions.



```
mipka=*# commit;
COMMIT
mipka=# SELECT * FROM users;
 username |     fullname      | balance | group_id
----------+-----------------+---------+----------
 bitdiddl | Ben Bitdiddle    |      65 |        1
 mike     | Michael Dole     |      73 |        2
 alyssa   | Alyssa P.Hacker  |      79 |        3
 bbrown   | Bob Brown        |     100 |        3
 ajones   | Alice Jones      |      82 |        1
(5 строк)
```

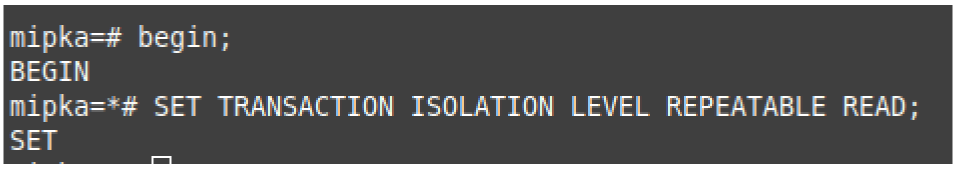
terminal 2



```
mipka=*# SELECT * FROM users;
 username |     fullname      | balance | group_id
----------+-----------------+---------+----------
 jones    | Alice Jones      |      82 |        1
 bitdiddl | Ben Bitdiddle    |      65 |        1
 mike     | Michael Dole     |      73 |        2
 alyssa   | Alyssa P.Hacker  |      79 |        3
 bbrown   | Bob Brown        |     100 |        3
(5 строк)
```

terminal 1

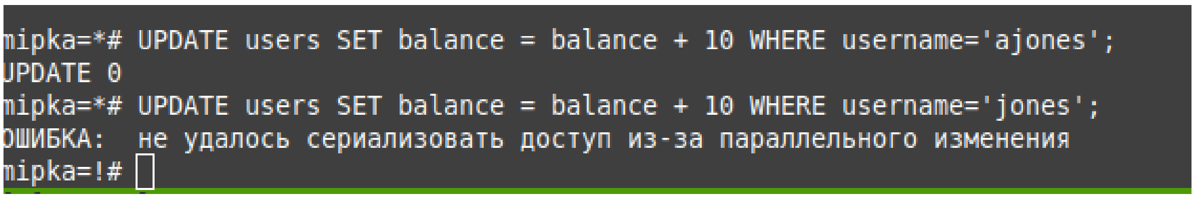
STEP 6 TERMINAL 2: Start a new transaction



```
mipka=# begin;
BEGIN
mipka=*# SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
```

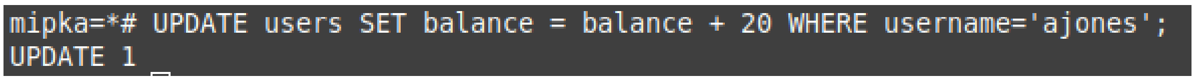
STEP 7 TERMINAL 1: Update the balance for the Alice’s account by +10

T1 update will not work with such error: ERROR: could not serialize access due to concurrent update because transaction of T2 has updated the same record (Alice) and due to the guarantees of the repeatable read isolation level, the statement in this session was cancelled. So after commit; we will see ROLLBACK in T1 transaction.



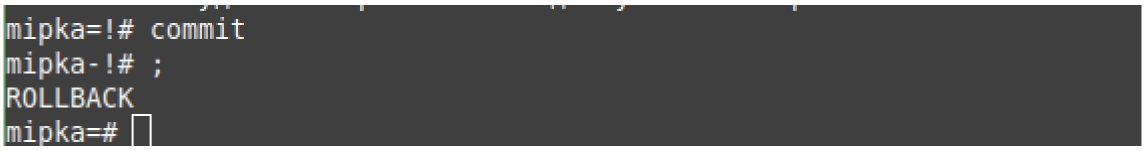
```
nipka=*# UPDATE users SET balance = balance + 10 WHERE username='ajones';
UPDATE 0
nipka=*# UPDATE users SET balance = balance + 10 WHERE username='jones';
ОШИБКА:  не удалось сериализовать доступ из-за параллельного изменения
nipka=!# 
```

STEP 8 TERMINAL 2: Update the balance for the Alice’s account by +20



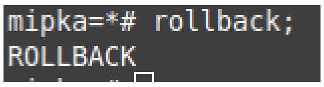
```
mipka=*# UPDATE users SET balance = balance + 20 WHERE username='ajones';
UPDATE 1
```

STEP 9 TERMINAL 1: Commit the changes.



```
mipka=!# commit
mipka-!# ;
ROLLBACK
mipka=# 
```

STEP 10 TERMINAL 2: Rollback



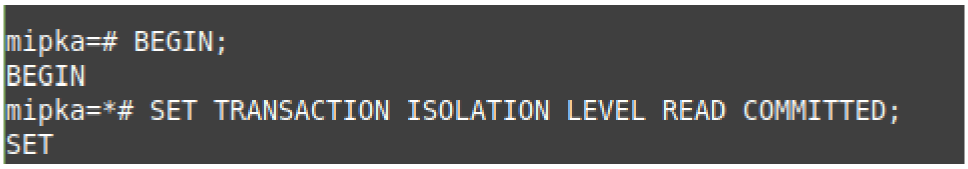
```
mipka=*# rollback;
ROLLBACK
```

EXERCISE 2 TASK 2.

FOR READ COMMITTED

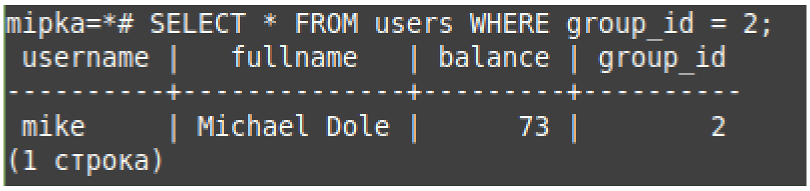
The main idea is that is this isolation level only the committed data could be read. E.g. you can start transaction and change data during the another transaction and if you committed it another transaction will see it.

STEP 1: Start a transaction (T1 & T2)



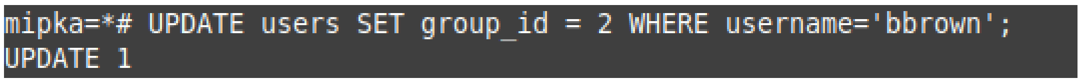
```
mipka=# BEGIN;
BEGIN
mipka=*# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET
```

STEP 2: Read accounts with group_id=2 (T1).



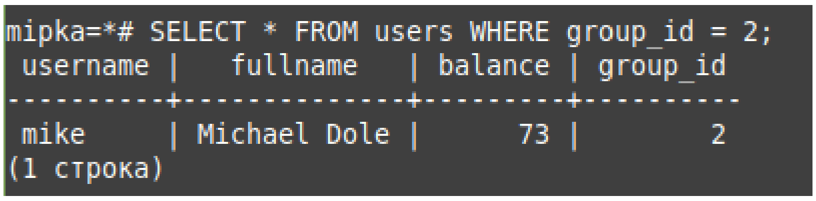
```
mipka=*# SELECT * FROM users WHERE group_id = 2;
 username |   fullname    | balance | group_id
----------+--------------+---------+----------
 mike     | Michael Dole |      73 |        2
(1 строка)
```

STEP 3: Move Bob to group 2(T2).



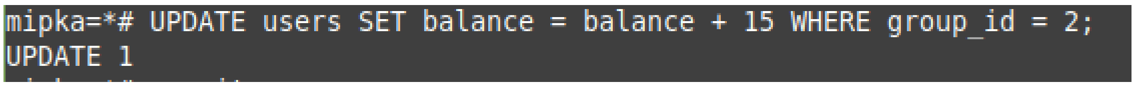
```
mipka=*# UPDATE users SET group_id = 2 WHERE username='bbrown';
UPDATE 1
```

STEP 4: Read accounts with group_id=2 (T1).



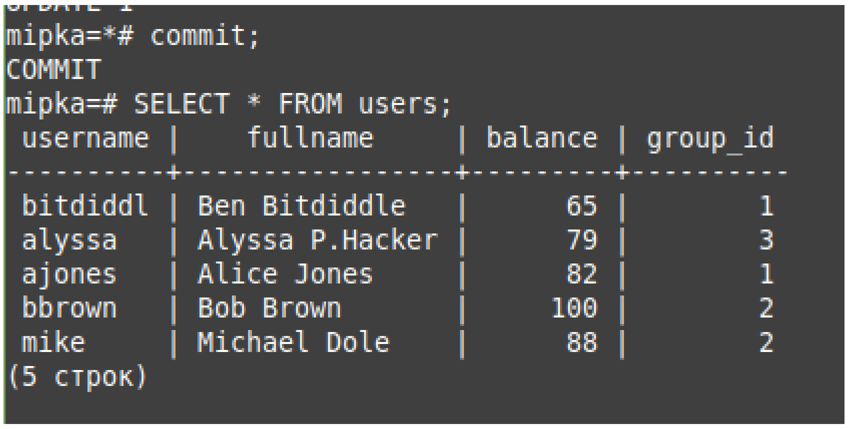
```
mipka=*# SELECT * FROM users WHERE group_id = 2;
 username |   fullname    | balance | group_id
----------+--------------+---------+----------
 mike     | Michael Dole |      73 |        2
(1 строка)
```

STEP 5: Update selected accounts balances by +15 (T1).



```
mipka=*# UPDATE users SET balance = balance + 15 WHERE group_id = 2;
UPDATE 1
```

STEP 6: Commit transaction (T1 & T2)



```
mipka=*# commit;
COMMIT
mipka=# SELECT * FROM users;
 username |    fullname     | balance | group_id
----------+-----------------+---------+----------
 bitdiddl | Ben Bitdiddle   |      65 |        1
 alyssa   | Alyssa P.Hacker |      79 |        3
 ajones   | Alice Jones     |      82 |        1
 bbrown   | Bob Brown       |     100 |        2
 mike     | Michael Dole    |      88 |        2
(5 строк)
```

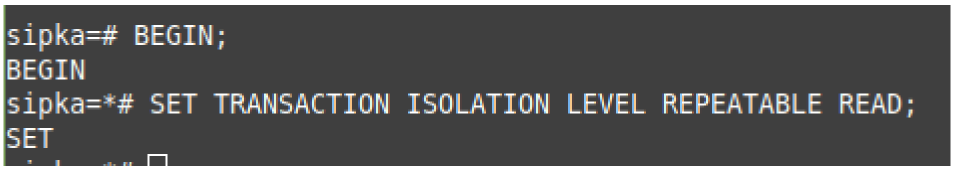
Answer:

We can see that at first `mike` has got +15 to balance but `bbrow`has not. It's because T1 transaction was processing while T2 transaction has not been committed yet, so `bbrow` has not been in group 2 yet.

FOR REPEATABLE READ

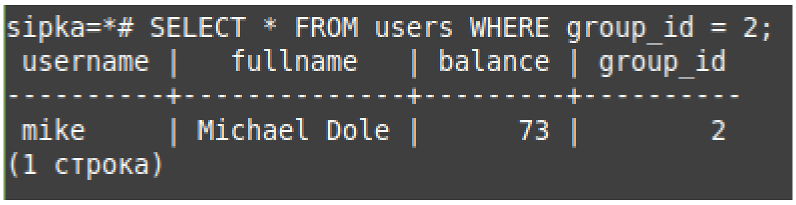
In case of repeatable read even when we commit changes in transaction, the another transaction won't see it before the end of transaction.

STEP 1: Start a transaction (T1 & T2)



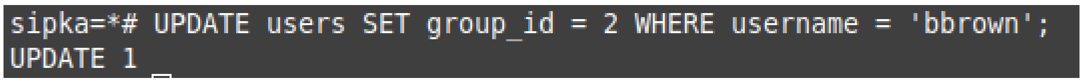
```
sipka=# BEGIN;
BEGIN
sipka=*# SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
```

STEP 2: Read accounts with group_id=2 (T1).



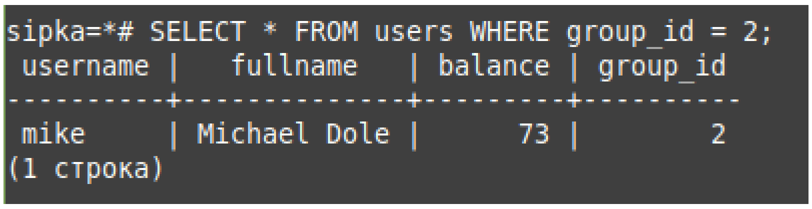
```
sipka=*# SELECT * FROM users WHERE group_id = 2;
 username |   fullname   | balance | group_id
----------+--------------+---------+----------
 mike     | Michael Dole |      73 |        2
(1 строка)
```

STEP 3: Move Bob to group 2(T2).



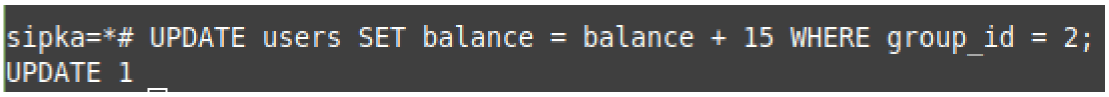
```
sipka=*# UPDATE users SET group_id = 2 WHERE username = 'bbrown';
UPDATE 1
```

STEP 4: Read accounts with group_id=2 (T1).



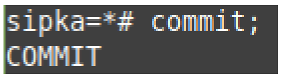
```
sipka=*# SELECT * FROM users WHERE group_id = 2;
 username |   fullname   | balance | group_id
----------+--------------+---------+----------
 mike     | Michael Dole |      73 |        2
(1 строка)
```

STEP 5: Update selected accounts balances by +15 (T1).

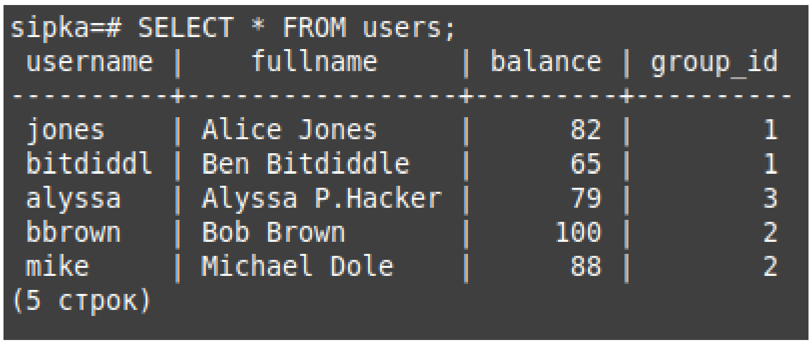


```
sipka=*# UPDATE users SET balance = balance + 15 WHERE group_id = 2;
UPDATE 1 _
```

STEP 6: Commit transaction (T1 & T2)



```
sipka=*# commit;
COMMIT
```



```
sipka=# SELECT * FROM users;
 username |    fullname     | balance | group_id
----------+-----------------+---------+----------
 jones    | Alice Jones     |      82 |        1
 bitdiddl | Ben Bitdiddle   |      65 |        1
 alyssa   | Alyssa P.Hacker |      79 |        3
 bbrown   | Bob Brown       |     100 |        2
 mike     | Michael Dole    |      88 |        2
(5 строк)
```

Answer:

We can see that the result is the same as in read committed. There are no errors because T2 updated record which had not been read by T1. Therefore, T1 & T2 updated different records and obtain the same result.