Exercise 2

Create and fill table

```
CREATE TABLE balance_info(
2
        username
                       varchar(100),
3
        fullname
                        varchar(100).
4
        balance
                       integer,
5
        group_id
                       integer
6
   );
7
8
9 INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('jones', 'Alice Jones', 82, 1);
10 INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('bitdiddl', 'Ben Bitdiddle', 65, 1);
11 INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('mike', 'Michael Dole', 73, 2);
12 INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('alyssa', 'Alyssa P. Hacker', 79, 3);
13 INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('bbrown', 'Bob Brown', 100, 3);
```

First part with **read committed**:

```
tgres@localhost:lab11ex1> begin
      BEGIN
       Time: 0.000s
       postgres@localhost:lab11ex1> set transaction isolation level read committed
1)
       Time: 0.001s
       oostgres@localhost:lab11ex1> select * from balance_info
                                                  1
                    Alice Jones
                    Ben Bitdiddle
                                                  1
2
        bitdiddl
                                        65
        mike
                    Michael Dole
        alyssa
                    Alyssa P. Hacker
                                        79
                                                  3
        bbrown
                    Bob Brown
                                        100
       SELECT 5
       ime: 0.011s
       ostgres@localhost:lab11ex1> select * from balance_info
                    Alice Jones
                                                  1
2
        bitdiddl
                    Ben Bitdiddle
        mike
                    Michael Dole
                                                  30
                    Alyssa P. Hacker
        alyssa
                    Bob Brown
        bbrown
       SELECT 5
       Time: 0.005s
       ostgres@localhost:lab11ex1>
```

```
ostgres@localhost:lab11ex1> begin
2)
      BEGIN
       Time: 0.001s
       postgres@localhost:lab11ex1> set transaction isolation level read committed
       postgres@localhost:lab11ex1> update balance_info SET username = 'ajones' WHERE fullname = 'Alice Jones
       You're about to run a destructive command.
Do you want to proceed? (y/n): y
       our call!
        JPDATE 1
        ostgres@localhost:lab11ex1> select * from balance_info
         bitdiddl
                     Ben Bitdiddle
         mike
                     Michael Dole
         alyssa
                     Alyssa P. Hacker
                                          79
                                                     3
3
1
                     Bob Brown
                                          100
         bbrown
                     Alice Jones
         aiones
                                          82
        ime: 0.010s
        ostgres@localhost:lab11ex1>
```

After 4 steps, we can see that the <u>tables are different</u> because the committed read isolation level <u>is free from dirty reads</u>, which results in reading uncommitted changes from other transactions.

After 5 step the tables became the same because the committed read isolation level <u>isn't free from non-repeatable read</u>. A non-repeatable read means that one of the rows you requested at different stages of the transaction cannot be updated by other transactions.

After step 8, Alice's account balance <u>increases by 10</u> because the <u>committed read isolation level is not free of lost updates</u>. A lost update is when the first transaction reads data into its local memory and then the second transaction changes that data and commits its change. The first transaction then updates the same data based on what was read into memory prior to the second transaction. In this case, the update performed by the second transaction can be considered a lost update.

Into my computer in the second terminal process go to the infinite loop.

First part with repeatable read:

```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.001s
postgres@localhost:lab11ex1> set transaction isolation level repeatable read
SET
Time: 0.001s
postgres@localhost:lab11ex1> select * from balance_info
             Alice Jones
                                            1
                                 82
  jones
                                 65
                                            1
  bitdiddl
             Ben Bitdiddle
                                 73
                                            2
  mike
             Michael Dole
  alyssa
             Alyssa P. Hacker
                                 79
                                            3
  bbrown
             Bob Brown
                                 100
                                            3
SELECT 5
Time: 0.005s
postgres@localhost:lab11ex1> select * from balance info
                                 bal
  jones
             Alice Jones
                                 82
                                            1
                                            1
  bitdiddl
             Ben Bitdiddle
                                 65
  mike
             Michael Dole
                                 73
                                 79
                                            3
  alyssa
             Alyssa P. Hacker
             Bob Brown
                                 100
 bbrown
SELECT 5
Time: 0.015s
postgres@localhost:lab11ex1>
```

```
oostgres@localhost:lab11ex1> begin
BEGIN
Time: 0.000s
ostgres@localhost:lab11ex1> set transaction isolation level repeatable read
ostgres@localhost:lab11ex1> update balance_info SET username = 'a
                                                                              ' WHERE fullname =
Oo you want to proceed? (y/n): y
'our call!
ou're about to run a destructive command.
JPDATE 1
Time: 0.002s
 ostgres@localhost:lab11ex1> select * from balance_info
 bitdiddl
             Ben Bitdiddle
                                             1
2
3
1
 mike
             Michael Dole
             Alyssa P. Hacker
 alyssa
             Bob Brown
Alice Jones
 bbrown
                                  100
 ajones
ELECT 5
ime: 0.011s
 ostgres@localhost:lab11ex1> _
```

After 4 steps, we can see that the <u>tables are different</u> because the repeatable read isolation level <u>is free from dirty reads</u>,

which results in reading uncommitted changes from other transactions.

After step 5

		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	Í 3 Í

```
postgres@localhost:lab11ex1> commit
Time: 0.002s
oostgres@localhost:lab11ex1>
Time: 0.000s
oostgres@localhost:lab11ex1> commit
ПРЕДУПРЕЖДЕНИЕ: нет незавершённой транзакции
Time: 0.003s
postgres@localhost:lab11ex1> select * from balance_info
 bitdiddl
            Ben Bitdiddle
            Michael Dole
 mike
 alyssa
            Alyssa P. Hacker
                                79
            Bob Brown
                                100
 bbrown
            Alice Jones
ELECT 5
Time: 0.005s
postgres@localhost:lab11ex1>
```

the tables remain distinct because the committed read isolation level is free from non-repeatable reads. This means that one of the rows requested by you at different stages of the transaction may be updated by other transactions.

After step 8,

```
postgres@localhost:lab11ex1> update balance_info set balance = balance + 10 where fullname You're about to run a destructive command.

Do you want to proceed? (y/n): y
        Your call!
1)
       oostgres@localhost:lab11ex1> update balance_info SET balance = balance + 20 WHERE fullname
You're about to run a destructive command.
Do you want to proceed? (y/n): y
      your call!
UPDATE 1
Time: 0.001s
postgres@localhost:lab11ex1>
Time: 0.000s
       oostgres@localhost:lab11ex1> select * from balance_info
                                                          65
73
79
         bitdiddl
                           Ben Bitdiddle
Michael Dole
                                                                          1
2
3
         mike
                           Alyssa P. Hacker
Bob Brown
         alyssa
                                                          100
102
         bbrown
                           Alice Jones
         ajones
```

In the second terminal Alice's account balance increases by 20 and the first terminal get the error because the <u>committed read</u> <u>isolation level is free of lost updates</u>.

Second part with **read committed**:

```
BEGIN
     postgres@localhost:lab11ex1> set transaction isolation level read committed SET
     Time: 0.001s
     oostgres@localhost:lab11ex1> select * from balance_info where group_id = 2
               | Michael Dole | 73
     Time: 0.011s
     ostgres@localhost:lab11ex1> select * from balance_info where group_id = 2
               | Michael Dole | 73
     SELECT 1
     postgres@localhost:lab11ex1> update balance_info SET balance = balance + 15 WHERE group_id = 2
/ou're about to run a destructive command.
po you want to proceed? (y/n): y
        call!
     ime: 0.001s
      ostgres@localhost:lab11ex1> commit
    Time: 0.001s
postgres@localhost:lab11ex1>
     postgres@localhost:lab11ex1> begin
     BEGIN
     Time: 0.001s
     postgres@localhost:lab11ex1> set transaction isolation level read committed
     SET
     Time: 0.000s
     postgres@localhost:lab11ex1> update balance info SET group id = 2 where fullname =
     You're about to run a destructive command.
     Do you want to proceed? (y/n): y
     Your call!
     UPDATE 1
     Time: 0.002s
     oostgres@localhost:lab11ex1> commit
     Time: 0.002s
     postgres@localhost:lab11ex1>
```

The list of accounts with the group_id = 2 in T1 was not updated after in T2 the Bob group_id was changed by 2 because the committed read isolation level is free from dirty reads, which results in reading uncommitted changes from other transactions. Since the balance was increases by 15 only for Michael Dole.

4	username character varying (100)	fullname character varying (100)	balance integer	group_id integer
1	jones	Alice Jones	82	1
2	bitdiddl	Ben Bitdiddle	65	1
3	alyssa	Alyssa P. Hacker	79	3
4	bbrown	Bob Brown	100	2
5	mike	Michael Dole	88	2

2)

Second part with repeatable read:

```
postgres@localhost:lab11ex1> begin
    BEGIN
    Time: 0.000s
    postgres@localhost:lab11ex1> set transaction isolation level repeatable read
    SET
    Time: 0.001s
    postgres@localhost:lab11ex1> select * from balance_info where group_id = 2
                Michael Dole | 73
                                         2
      mike
    SELECT 1
    Time: 0.006s
    postgres@localhost:lab11ex1> select * from balance_info where group_id = 2
                 Michael Dole | 73
      mike
    SELECT 1
    Time: 0.005s
    postgres@localhost:lab11ex1> update balance info SET balance = balance + 15 WHERE group id = 2
    You're about to run a destructive command.
    Do you want to proceed? (y/n): y
    Your call!
    UPDATE 1
    Time: 0.002s
    postgres@localhost:lab11ex1> commit
    COMMIT
    Time: 0.002s
1) postgres@localhost:lab11ex1>
    postgres@localhost:lab11ex1> begin
    BEGIN
    Time: 0.000s
    postgres@localhost:lab11ex1> set transaction isolation level repeatable read
    SET
    Time: 0.000s
    postgres@localhost:lab11ex1> update balance_info SET group_id = 2 where fullname =
    You're about to run a destructive command.
    Do you want to proceed? (y/n): y
    Your call!
    UPDATE 1
    Time: 0.001s
    postgres@localhost:lab11ex1> commit
    COMMIT
    Time: 0.001s
    postgres@localhost:lab11ex1>
2)
```

The results the same with the read committed isolation level since both isolation levels is free from dirty reads.

4	username character varying (100)	fullname character varying (100)	balance integer	group_id integer	4
1	jones	Alice Jones	82		1
2	bitdiddl	Ben Bitdiddle	65		1
3	alyssa	Alyssa P. Hacker	79		3
4	bbrown	Bob Brown	100		2
5	mike	Michael Dole	88		2