МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Практическое занятие № 3

Тема занятия: Настройка и поддержка транзакций в СУБД PostgreSQL

Выполнил студент 3 курса группы
ЭФБО-10-23 Ефремов А.И.

Проверил доцент Бочаров М.И.

Москва
2025 г.

Ефремов ЭФБО-10-23 Практическая 3

Часть 1

1. Создадим произвольную таблицу.

```
    CREATE TABLE yefremov (id integer);
    ```
![[Pasted image 20251007111448.png]]
```

2. Вставим первую строку в таблицу.

```
    INSERT INTO yefremov VALUES (100);
    ```
![[Pasted image 20251007111508.png]]
```

3. Посмотрим, какой номер транзакции xmin: `SELECT xmin, xmax, * FROM a;`

```
    SELECT xmin, xmax, * FROM yefremov;
    ```
![[Pasted image 20251007111539.png]]
```

4. Начнем явную транзакцию: BEGIN;

```
    BEGIN;
    ```
![[Pasted image 20251007111607.png]]
```

5. Обновим первую строчку.

```
UPDATE yefremov SET id = 200 WHERE id = 100;
```

```
mirea=*#    UPDATE yefremov SET id = 200 WHERE id = 100;
UPDATE 1
```

6. Обратимся и посмотрим, что получилось: SELECT xmin, xmax, * FROM a;

```
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=*#    SELECT xmin, xmax, * FROM yefremov;
 xmin | xmax | id
------+------+-----
  881 |    0 | 200
(1 row)
```

7. Во втором терминале обращаемся к таблице. Посмотрим, какой номер транзакции xmin: SELECT xmin, xmax, * FROM a;

```
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=#    SELECT xmin, xmax, * FROM yefremov;
 xmin | xmax | id
------+------+-----
  880 |  881 | 100
(1 row)
```

8. В первом терминале фиксируем транзакцию

```
COMMIT;
```

```
mirea=*#    COMMIT;
COMMIT
```

9. Во втором терминале теперь видим вторую строку: SELECT xmin, xmax, * FROM a

```
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=#    SELECT xmin, xmax, * FROM yefremov;
 xmin | xmax | id
------+------+-----
  881 |    0 | 200
(1 row)
```

10. Теперь посмотрим, как выглядит удаление. Откроем транзакцию в первом терминале.

```
BEGIN;
```

```
mirea=#    BEGIN;
BEGIN
```

11. Удаляем строчку.

```
DELETE FROM yefremov WHERE id = 200;
```

```
mirea=*#    DELETE FROM yefremov WHERE id = 200;
DELETE 1
```

12. Посмотрим результат: SELECT xmin, xmax, * FROM a; Первая транзакция не видит строчку, она удалена, но изменение пока не зафиксировано.

```
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=*#    SELECT xmin, xmax, * FROM yefremov;
 xmin | xmax | id
------+------+-----
(0 rows)
```

13. Посмотрим результат во втором терминале: SELECT xmin, xmax, * FROM a;

```
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=#    SELECT xmin, xmax, * FROM yefremov;
 xmin | xmax | id
------+------+-----
  881 |  882 | 200
(1 row)
```

Строка еще видна, но xmax опять изменился.

14. В первом терминале фиксируем транзакцию

```
COMMIT;
```

```
mirea=*#    COMMIT;
COMMIT
```

15. Во втором терминале теперь видим изменение

```
sql SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=#     SELECT xmin, xmax, * FROM yefremov;
 xmin | xmax | id
------+------+----
(0 rows)
```

Часть 2

16. Начнем явную транзакцию в первом терминале: BEGIN

```
BEGIN;
```

```
mirea=#     BEGIN;
BEGIN
```

17. Посмотрим уровень изоляции: SHOW transaction_isolation;

```
SHOW transaction_isolation;
```

```
mirea=*#     SHOW transaction_isolation;
 transaction_isolation
-----------------------
 read committed
(1 row)
```

18. Добавим новую строку.

```
INSERT INTO yefremov VALUES (300);
```

```
mirea=*#     INSERT INTO yefremov VALUES (300);
INSERT 0 1
```

19. Начнем вторую транзакцию во втором терминале и обратимся к таблице.

```
BEGIN;
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=#     BEGIN;
        SELECT xmin, xmax, * FROM yefremov;
BEGIN
 xmin | xmax | id
------+------+----
(0 rows)

mirea=*#
```

20. Посмотрим уровень изоляции

```
SHOW transaction_isolation;
```

```
  mirea=*#     SHOW transaction_isolation;
   transaction_isolation
  -----------------------
   read committed
  (1 row)
```

21. Пока новая строка не видна. Зафиксируем первую транзакцию

```
COMMIT;
```

```
mirea=*#     COMMIT;
COMMIT
```

22. Во втором окне повторно обратимся к таблице. Что увидим?

```
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=*#     SELECT xmin, xmax, * FROM yefremov;
 xmin | xmax | id
------+------+-----
  883 |    0 | 300
(1 row)
```

23. Зафиксируем вторую транзакцию Изменения стали видны. Это и есть аномалия неповторяющегося чтения. Теперь в первом окне начнем транзакцию на уровне repeatable read.

```
COMMIT;
```

```
mirea=*#     COMMIT;
COMMIT
```

24. Вставим еще одну строку: BEGIN ISOLATION LEVEL REPEATABLE READ

```
BEGIN ISOLATION LEVEL REPEATABLE READ;
INSERT INTO yefremov VALUES (400);
```

```
mirea=#     BEGIN ISOLATION LEVEL REPEATABLE READ;
    INSERT INTO yefremov VALUES (400);
BEGIN
INSERT 0 1
```

25. Во второй транзакции обратимся к таблице в новой транзакции на том же уровне.

```
BEGIN ISOLATION LEVEL REPEATABLE READ;
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=#     BEGIN ISOLATION LEVEL REPEATABLE READ;
        SELECT xmin, xmax, * FROM yefremov;
BEGIN
 xmin | xmax | id
------+------+-----
  883 |    0 | 300
(1 row)
```

26. Теперь фиксируем первую транзакцию

```
COMMIT;
```

27. Обратимся ко второй транзакции еще раз.
Изменения не видны. на этом уровне операторы транзакции работают только с одним снимком данных.

```
SELECT xmin, xmax, * FROM yefremov;
```

```
mirea=*#     SELECT xmin, xmax, * FROM yefremov;
 xmin | xmax | id
------+------+-----
  883 |    0 | 300
(1 row)
```

28. Зафиксируем вторую транзакцию.

```
COMMIT;
```

```
mirea=*#     COMMIT;
COMMIT
```

Часть 3

29. Откроем первую транзакцию, добавим строку и посмотрим после вставки состояние.

```
BEGIN;
INSERT INTO yefremov VALUES (500);
SELECT txid_current() AS current_txid;
```

```
mirea=#     BEGIN;
        INSERT INTO yefremov VALUES (500);
        SELECT txid_current() AS current_txid;
BEGIN
INSERT 0 1
 current_txid
--------------
          886
(1 row)
```

30. Видим вставку третьей строки. Посмотрим статус транзакции: SELECT pg_xact_status('1598');

```
COMMIT;
SELECT pg_xact_status('886');
```

```
mirea=# SELECT pg_xact_status('886');
 pg_xact_status
----------------
 committed
(1 row)
```

31. Теперь посмотрим, как поведет себя CLOG при откате транзакции: откроем транзакцию, добавим новую строку, просмотрим статус транзакции, выполним отказ транзакции, еще раз посмотрим статус

```
BEGIN;
INSERT INTO yefremov VALUES (600);
SELECT txid_current() AS current_txid;
```

```
mirea=#     BEGIN;
        INSERT INTO yefremov VALUES (600);
        SELECT txid_current() AS current_txid;
BEGIN
INSERT 0 1
 current_txid
--------------
          888
(1 row)
```

```
SELECT pg_xact_status('888');
ROLLBACK;
SELECT pg_xact_status('888');
```

```
mirea=*#       SELECT pg_xact_status('888');
       ROLLBACK;
       SELECT pg_xact_status('888');
 pg_xact_status
----------------
 in progress
(1 row)

ROLLBACK
 pg_xact_status
----------------
 aborted
(1 row)
```

Часть 4

33. В первой транзакции вставим новую строку и посмотрим блокировки с помощью pg_locks, для этого нам нужен pid обслуживаю pg_backend_pid();

```
SELECT pg_backend_pid();
BEGIN;
INSERT INTO yefremov VALUES (700);
```

```
mirea=#     SELECT pg_backend_pid();
       BEGIN;
       INSERT INTO yefremov VALUES (700);
 pg_backend_pid
----------------
           5494
(1 row)

BEGIN
INSERT 0 1
mirea=*#
```

34. Откроем транзакцию и обратимся к таблице.

```
BEGIN;
SELECT * FROM yefremov;
```

```
mirea=#     BEGIN;
       SELECT * FROM yefremov;
BEGIN
 id
-----
 300
 400
 500
(3 rows)

mirea=*#
```

35. Выполним обновление данных: UPDATE a SET id = id + 1;

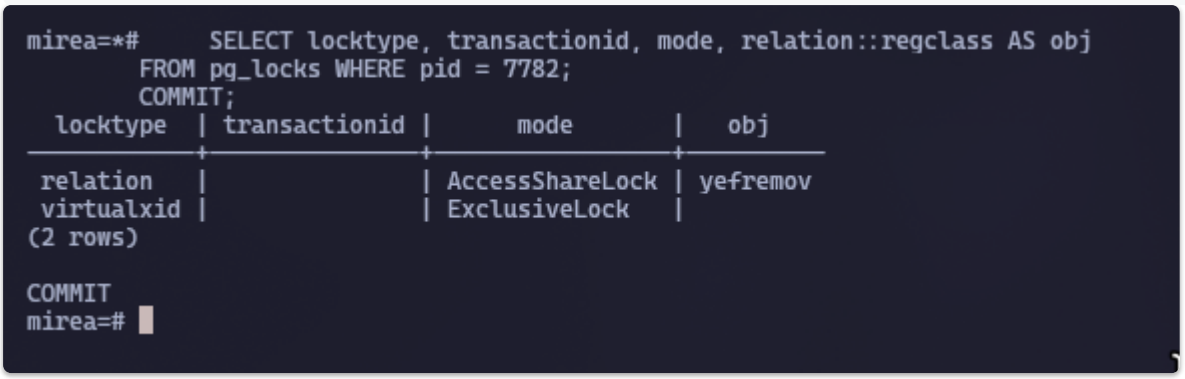
```
UPDATE yefremov SET id = id + 1;
```

```
mirea=*#     UPDATE yefremov SET id = id + 1;
UPDATE 3
mirea=*#
```

36. Выполним запрос: SELECT locktype, transactionid, mode, relation::regclass as obj FROM pg_locks where pid = 12948; Появилась блокировка на

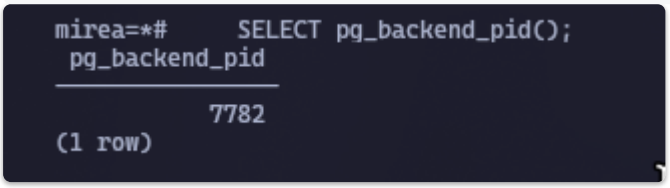
уровне таблицы RowExclusiveLock — накладывается в случае обновления строк.

```
SELECT locktype, transactionid, mode, relation::regclass AS obj
FROM pg_locks WHERE pid = 7782;
COMMIT;
```



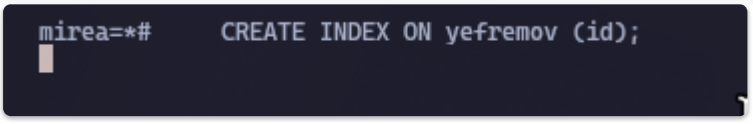
37. Во втором окне построим индекс по таблице, предварительно посмотрим pid процесса: SELECT pg_backend_pid()

```
SELECT pg_backend_pid();
```



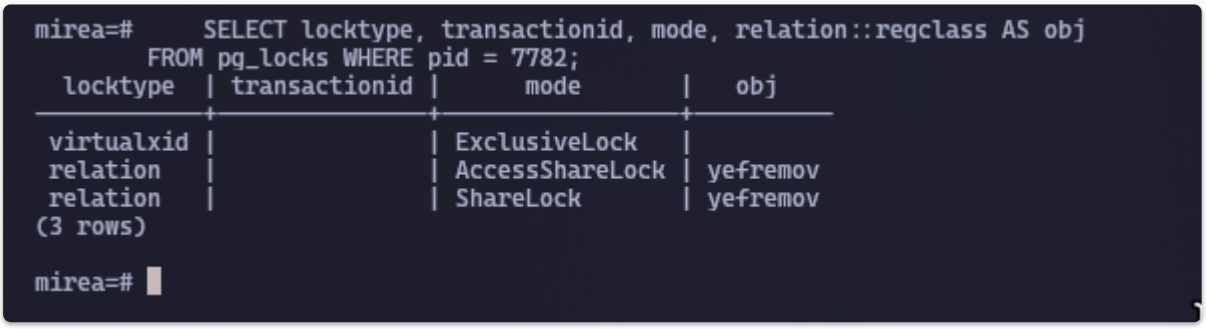
38. Создадим индекс: CREATE INDEX ON a (id);

```
CREATE INDEX ON yefremov (id);
```



39. Транзакция подвисла. В первом терминале посмотрим, что происходит во втором процессе: SELECT locktype, transactionid, mode, relation::regclass as obj FROM pg_locks where pid = 5308; Появилась блокировка ShareLock она не совместима RowExclusiveLock возникла блокировочная ситуация.

```
SELECT locktype, transactionid, mode, relation::regclass AS obj
FROM pg_locks WHERE pid = 7782;
```



40. Зафиксируем первую транзакцию. Тут же срабатывает команда во втором окне

```
COMMIT;
```

```
COMMIT
mirea=#
```