ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №2

Транзакції в СКБД PostgreSQL

Виконала:

Ст. *Кравець Ольга*

ПМО-31

2022

Тема: Вивчення понять транзакції та управління конкурентним доступом в СКБД PostgreSQL.

Мета роботи: Ознайомлення з використанням транзакцій, їх розробкою та застосуванням, рівнями ізоляцій та механізмом управління конкурентним доступом в СКБД PostgreSQL

Хід роботи

Опрацювала теоретичні відомості про транзакції, атомарні транзакції, команди, скасування змін, точки збереження, повернення до точок збереження, рівні ізоляції, явні блокування, перевірки цілісності даних, обмеження, блокування та індекси.

Написала елементарну транзакцію з трьома операціями, одна з яких скасована з допомогою SAVEPOINT'а та операції повернення до конкретного моменту ROLLBACK TO (Рівень ізоляції - Read Committed). Операції - збільшення ціни товарів для певних виробників:

Query Editor

```
1  SELECT * FROM public.goods
2  ORDER BY id ASC
```

Data Output | Explain | Messages | Query History | Notifications

	id [PK] integer	price integer	weight integer	go_name character (64)	color character (32)	material character (64)	provider_id integer	quantity integer
1	1	2200	12	commode	black	chipboard	1	2
2	2	1500	3	chair	white	wood	2	5
3	3	15000	37	wardrobe	white	MDF	2	7
4	4	26000	48	closet	black	wood	3	1
5	5	1000	4	nightstand	brown	veneer	5	2

Query Editor

```
1  begin;
2  update goods set price = price*2 where provider_id = 3;
3  savepoint sp_operation;
4  update goods set price = price*10 where provider_id = 1;
5  rollback to sp_operation;
6  update goods set price = price*3 where provider_id = 2;
7  commit;
```

Data Output | Explain | Messages | Query History | Notifications

```
COMMIT

Query returned successfully in 129 msec.
```

Query Editor

```
SELECT * FROM public.goods
ORDER BY id ASC
```

Data Output | Explain | Messages | Query History | Notifications

	id [PK] integer	price integer	weight integer	go_name character (64)	color character (32)	material character (64)	provider_id integer	quantity integer
1	1	2200	12	commode	black	chipboard	1	2
2	2	4500	3	chair	white	wood	2	5
3	3	45000	37	wardrobe	white	MDF	2	7
4	4	52000	48	closet	black	wood	3	1
5	5	1000	4	nightstand	brown	veneer	5	2

Написала транзакції, для демонстрації рівнів ізоляції *Repeatable Read*.

Dirty Read

Транзакція для демонстрації вибору товару з ID = 2:

Query Editor

```
begin transaction isolation level repeatable read;
select * from goods where goods.id = 2;
```

Data Output | Explain | Messages | Query History | Notifications

	id [PK] integer	price integer	weight integer	go_name character (64)	color character (32)	material character (64)	provider_id integer	quantity integer
1	2	40500	3	bed	white	wood	2	5

Далі змінюю дані таблиці:

Query Editor

```
begin transaction isolation level repeatable read;
update goods set "go_name"='chair' where goods.id = 2;
commit;
```

Data Output | Explain | Messages | Query History | Notifications

```
COMMIT

Query returned successfully in 49 msec.
```

Виконую ще раз select у першій транзакції, щоб поглянути чи щось змінилось:

Query Editor

```
begin transaction isolation level repeatable read;
select * from goods where goods.id = 2;
```

Data Output | Explain | Messages | Query History | Notifications

	id [PK] integer	price integer	weight integer	go_name character (64)	color character (32)	material character (64)	provider_id integer	quantity integer
1	2	40500	3	bed	white	wood	2	5

Дані не змінилися, оскільки транзакція бачить лише зміни, що виконані при першому читанні.

Nonrepeatable Read

Транзакція для демонстрації вибору постачальника з ID = 2:

Query Editor

```
1  SELECT * FROM public.provider
2  ORDER BY provider_id ASC
```

Data Output | Explain | Messages | Query History | Notifications

	provider_id [PK] integer	id_number bigint	pr_name character (64)	pr_address text
1	1	2064002378	Komandor	Львів
2	2	8151364225	DLM	Дніпро

Виконую 1 транзакцію:

Query Editor

```
1  -- Транзакція 1
2  set transaction isolation level read committed;
3  begin transaction;
4  --1
5  select * from provider where provider.provider_id = 2;
6
```

Data Output | Explain | Messages | Query History | Notifications

	provider_id [PK] integer	id_number bigint	pr_name character (64)	pr_address text	pr_telephone character (15)
1	2	8151364225	DLM	Дніпро	+(380)680571515

Далі змінюю дані таблиці:

Query Editor

```
1  -- Транзакція 2
2  set transaction isolation level read committed;
3  update provider set "pr_address"='Київ' where provider.provider_id = 2;
4
```

Data Output | Explain | Messages | Query History | Notifications

```
UPDATE 1

Query returned successfully in 48 msec.
```

Виконую ще раз select у 1 транзакції:

Query Editor

```
1  -- Транзакція 1
2  set transaction isolation level read committed;
3  begin transaction;
4  --1
5  --select * from provider where provider.provider_id = 2;
6  --2
7  select * from provider where provider.provider_id = 2;
8
```

Data Output | Explain | Messages | Query History | Notifications

	provider_id [PK] integer	id_number bigint	pr_name character (64)	pr_address text	pr_telephone character (15)
1	2	8151364225	DLM	Київ	+(380)680571515

Дані змінилися, оскільки 1 транзакція читала дані, паралельно інша транзакція оновила дані і зафіксувала нові і тоді 1 транзакція перечитала ті самі дані і отримала інше значення:

Query Editor

```
1  SELECT * FROM public.provider
2  ORDER BY provider_id ASC
```

Data Output | Explain | Messages | Query History | Notifications

	provider_id [PK] integer	id_number bigint	pr_name character (64)	pr_address text	pr_telephone character (15)
1	1	2064002378	Komandor	Львів	+(380)670705089
2	2	8151364225	DLM	Київ	+(380)680571515

Phantom Read

Транзакція для демонстрації вибору замовлення з сумою = 0:

Query Editor

```
1  SELECT * FROM public.orders
2  ORDER BY id ASC
```

Data Output | Explain | Messages | Query History | Notifications

	id [PK] integer	delivery_method character (25)	client_id integer	sum integer
1	1	Post	2	17500
2	2	Self-pickup	1	0
3	3	Post	4	0

Виконую 1 транзакцію:

Query Editor

```
1  -- Транзакція 1
2  set transaction isolation level repeatable read;
3  begin transaction;
4  --1
5  select * from orders where orders.sum = 0;
```

Data Output | Explain | Messages | Query History | Notifications

	id [PK] integer	delivery_method character (25)	client_id integer	sum integer
1	2	Self-pickup	1	0
2	3	Post	4	0

Далі додаю ще один рядок до таблиці зі значення суми = 0:

Query Editor

```
1  insert into orders (id, delivery_method, client_id, sum)
2  values (6, 'Post', 4, 0)
```

Data Output | Explain | Messages | Query History | Notifications

```
INSERT 0 1

Query returned successfully in 57 msec.
```

Виконую ще раз select у 1 транзакції:

Query Editor

```
1  -- Транзакція 2
2  set transaction isolation level repeatable read;
3  begin transaction;
4  --2
5  select * from orders where orders.sum = 0;
```

Data Output | Explain | Messages | Query History | Notifications

	id [PK] integer	delivery_method character (25)	client_id integer	sum integer
1	2	Self-pickup	1	0
2	3	Post	4	0
3	6	Post	4	0

Тепер стало 3 рядки з даними, де сума=0, оскільки 1 транзакція читала рядки, в яких була сума=0, а тоді додався рядок, який відповідав критерію, що сума=0 і 1 транзакція повторно виконалася і вже отримала інший набір рядків.

Написала процедуру, яка змінює назву товару з заданим ID. Якщо товар з вказаним ID знайдено, то викликається ROLLBACK (скасовуються зміни) та повертається EXCEPTION.

Query Editor

```
1  create procedure change_goods_name(g_id int, new_name text)
2  language plpgsql
3  as $$
4  declare
5  begin
6  update goods
7  set "go_name"=new_name
8  where goods.id = g_id;
9  if not found then
10         rollback;
11         raise exception 'Goods with this ID is not found.';
12     else
13         commit;
14 end if;
15 end;
16 $$
```

Data Output | Explain | Messages | Query History | Notifications

```
CREATE PROCEDURE

Query returned successfully in 94 msec.
```

До виклику:

Query Editor

```
1  SELECT * FROM public.goods
2  ORDER BY id ASC
```

Data Output | Explain | Messages | Query History | Notifi

	id [PK] integer	price integer	weight integer	go_name character (64)
1	1	2200	12	commode
2	2	40500	3	bed
3	3	405000	37	wardrobe
4	4	208000	48	closet
5	5	1000	4	nightstand

Виклик:

Query Editor

```
1  call change_goods_name(1, 'chair');
```

Data Output | Explain | Messages | Query History | Notifications

```
CALL

Query returned successfully in 73 msec.
```

Після виклику:

Query Editor

```
1  SELECT * FROM public.goods
2  ORDER BY id ASC
```

Data Output | Explain | Messages | Query Histo

	id [PK] integer	price integer	weight integer	go_name character (64)
1	1	2200	12	chair
2	2	40500	3	bed
3	3	405000	37	wardrobe
4	4	208000	48	closet
5	5	1000	4	nightstand

Виклик процедури з неіснуючим товаром:

Query Editor

```
1  call change_goods_name(7, 'chair');
2
```

Data Output | Explain | Messages | Query History | Notifications

```
ERROR: ПОМИЛКА:  Goods with this ID is not found.
CONTEXT:  Функція PL/pgSQL change_goods_name(integer,text) рядок 9 в RAISE


SQL state: P0001
```

Повертається помилка, що не існує товару з вказаним ID.

Висновок: на цій лабораторній роботі я ознайомилася з поняттям транзакцій, команд, скасування змін, рівнів ізоляції, перевірки цілісності даних.