

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №2

Транзакції в СКБД PostgreSQL

Виконав:

Ст. Заречанський

Олексій

Група ПМІ-33

Оцінка

Прийняв:

ас. Жировецький В.В.

Тема

Вивчення понять транзакції та управління конкурентним доступом в СКБД PostgreSQL .

Мета роботи

Ознайомлення з використанням транзакцій, їх рівнями ізоляцій та механізмом управління конкурентним доступом в СКБД PostgreSQL, процесом їх розробки та застосування.

Завдання

Розробити базу даних для web представлення системи управління документами. Система підтримує інформацію про документи, які проходять через неї. Крім власне документа важливими є інформація про його створення, історія змін, користувачі, які вносили зміни, та інформація про оффлайнове представлення документа (паперовий, CD, flash). Користувачі можуть створювати і змінювати документи і асоціювати їх з оффлайновим представленням, а також шукати документи за користувачами, які змінювали їх, станом чи представленням.

Звіт

Ця процедура використовує savepoint та ролбек, вона змінює прізвище користувача на 'Shreker', зберігає створює точку збереження, тоді змінює прізвище користувача на 'Sh', але після ролбеку до точки, там має бути 'Shreker'.

```
1  begin;
2
3      update customer
4      set surname = 'Shreker'
5      where email = 'last@mail.ua';
6
7      Savepoint savedSurname;
8
9      update customer
10     set surname = 'Sh'
11     where email = 'last@mail.ua';
12
13     rollback to savedSurname;
14
15 commit;
```

До:

4	last@mail.ua	password	Shrek
---	--------------	----------	-------

Після:

4	last@mail.ua	password	Shreker
---	--------------	----------	---------

Рівні ізоляції

Read uncommitted та Read committed:


Read uncommitted: Читає зміни які не були закомічені.

Read committed: Читає зміни які були закомічені.

Ця транзакція отримує дані вмісту документу.

```
begin isolation level read uncommitted;  
select content  
from document  
where id = 1;
```


Дані до:

	content	
	text	
1	Some content	


Виконаємо транзакцію зміни без коміту:

```
begin;  
update document  
set content = 'Test content'  
where id = 1;
```

Дані після:

	content	
	text	
1	Some content	

Після виконання коміту дані змінились.

	content	
	text	
1	Test content	

Хоч ми використовували read uncommitted, проте він не підтримується постгресом, тому наші зміни зчитались лише після коміту, тобто спрацював Read committed.


Repeatable read:

Читає зміни лише на початку транзакції, створюючи копію початкових даних.

Ця транзакція отримує дані вмісту документу.

```
begin isolation level read uncommitted;  
  select content  
  from document  
  where id = 1;
```

Дані до:

	content	
	text	
1	Some content	


Виконаємо апдейт з комітом:

```
begin transaction isolation level repeatable read;  
  update document  
  set content = 'Test content'  
  where id = 1;  
commit;
```


Виконаємо ще один селект в незакоміченій транзакції:

```
begin transaction isolation level repeatable read;  
  select content  
  from document  
  where id = 1;  
commit;
```

Результат:

	content	
	text	
1	Some conte...	

Після коміту данної транзакції з селектом, та запуску її заново, селект покаже вже нові дані, бо після перезапуску транзакції ми отримали нові дані таблиці.

	content	
	text	
1	Test content	

Serializable:

Так само як repeatable read створює копію таблиць на початку, але також блокує транзакції які при паралельному виконанні дають не той самий результат що при послідовному.

Перша транзакція допише до вмісту документу 1, а інша 2.

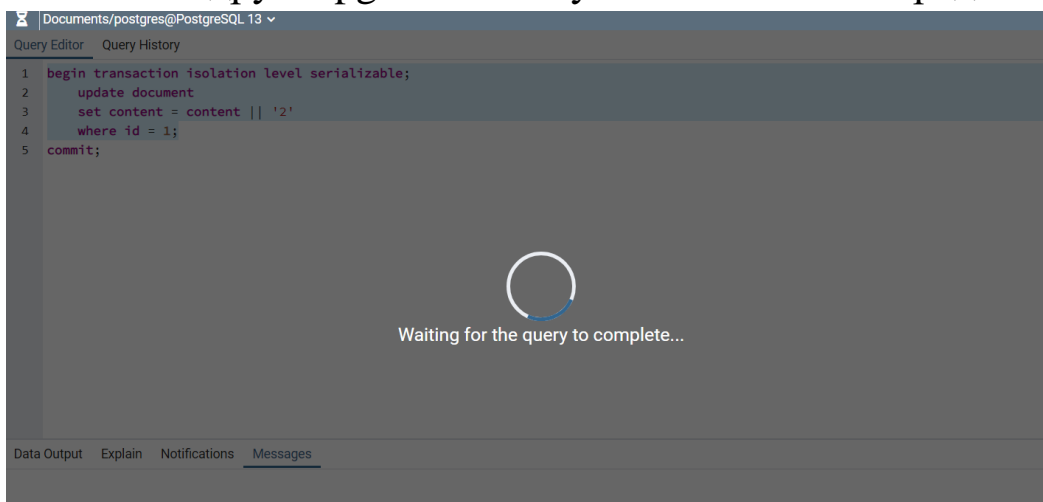
```
begin transaction isolation level serializable;
  update document
  set content = content || '1'
  where id = 1;
commit;

begin transaction isolation level serializable;
  update document
  set content = content || '2'
  where id = 1;
commit;
```

Початковий вміст:

content
text
Some content

Починаємо першу транзакцію без коміту, при початку виконання другої pgadmin очікує закінчення попередньої.



Після коміту першої отримуємо помилку:

Data Output Explain Notifications Messages

ERROR: could not serialize access due to concurrent update
SQL state: 40001

Бачимо що виконалась тільки перша транзакція, а друга була відмінена, бо отримуємо в ній помилку при спробі там далі працювати.


content text	
Some content1	

```
1 begin transaction isolation level serializable;
2   update document
3     set content = content || '2'
4     where id = 1;
5 commit;
```

[Data Output](#) [Explain](#) [Notifications](#) [Messages](#)

ERROR: current transaction is aborted, commands ignored until end of transaction block
SQL state: 25P02

При послідовному запуску обох транзакцій з комітами, отримуємо успішний результат.

content text	
Some content12	