НИЯУ МИФИ. Лабораторная работа №2-2

Транзакции в PostgreSQL

Журбенко Василий, Б21-525

2024

Цель работы

Изучение механизма транзакций в PostgreSQL и понимание различных уровней изоляции транзакций.

На защиту

Какие уровни изоляции Вы бы использовали в базе данных, которая собирает данные с погодных станций и генерирует прогнозы погоды?

1. Для приёма измерений с большого числа станций обычно достаточно уровня **READ COMMITTED**, чтобы исключить «грязные» (не зафиксированные) данные, но не создавать излишних блокировок и не тратить ресурсы на более жёсткую изоляцию.
2. Для задачи генерации прогноза (особенно если расчёты идут непрерывно и занимают ощутимое время), требуется иметь стабильную картину данных для вычислений. Иначе один и тот же отчёт или прогноз может скакать в пределах одной транзакции. **REPEATABLE READ** гарантирует, что данные не изменятся во время транзакции, хотя могут появиться новые строки.

Если вдруг возникнут действительно жёсткие требования к строго сериализованному доступу, можно отдельно запускать транзакции в режиме SERIALIZABLE, но это замедлит работу из-за блокировок и откатов, а выгоды для погодной модели невелики.

Ход работы

1. Эксперимент с READ COMMITTED

Сессия 1:

```
jewelry_store=# BEGIN;
BEGIN
jewelry_store=# SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

```
SET
jewelry_store=*# INSERT INTO suppliers (name, contact_info) VALUES ('Test
Supplier', 'Phone: +123456789');
INSERT 0 1
jewelry_store=*# SELECT * FROM suppliers WHERE name = 'Test Supplier';
 id |     name      |   contact_info
----+---------------+------------------
  5 | Test Supplier | Phone: +123456789
(1 row)

jewelry_store=*# ROLLBACK;
ROLLBACK
jewelry_store=# SELECT * FROM suppliers WHERE name = 'Test Supplier';
 id | name | contact_info
----+------+--------------
(0 rows)
```

Сессия 2:

```
jewelry_store=# SELECT * FROM suppliers WHERE name = 'Test Supplier';
 id | name | contact_info
----+------+--------------
(0 rows)

jewelry_store=# SELECT * FROM suppliers WHERE name = 'Test Supplier';
 id | name | contact_info
----+------+--------------
(0 rows)
```

Результат: При выполнении ROLLBACK в первой сессии, изменения отменяются и не видны во второй сессии.

Повторный эксперимент с COMMIT:

Сессия 1:

```
jewelry_store=# BEGIN;
BEGIN
jewelry_store=*# INSERT INTO suppliers (name, contact_info) VALUES ('Test
Supplier', 'Phone: +123456789');
INSERT 0 1
jewelry_store=*# SELECT * FROM suppliers WHERE name = 'Test Supplier';
 id |     name      |   contact_info
----+---------------+------------------
  6 | Test Supplier | Phone: +123456789
```

```
(1 row)

jewelry_store=*# COMMIT;
COMMIT
jewelry_store=# SELECT * FROM suppliers WHERE name = 'Test Supplier';
 id |     name      |   contact_info
----+---------------+------------------
  6 | Test Supplier | Phone: +123456789
(1 row)
```

Сессия 2:

```
jewelry_store=# SELECT * FROM suppliers WHERE name = 'Test Supplier';
 id | name | contact_info
----+------+--------------
(0 rows)

jewelry_store=# SELECT * FROM suppliers WHERE name = 'Test Supplier';
 id |     name      |   contact_info
----+---------------+------------------
  6 | Test Supplier | Phone: +123456789
(1 row)
```

Результат: После выполнения COMMIT изменения стали видны во второй сессии.

2. Эксперимент с REPEATABLE READ и SERIALIZABLE

REPEATABLE READ:

```
jewelry_store=# BEGIN;
BEGIN
jewelry_store=*# SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
SET
jewelry_store=*# SELECT * FROM suppliers;
 id |          name           |                      contact_info
----+-------------------------+-----------------------------------------------
-------
  1 | ООО "Золотая россыпь"   | Телефон: +7 (495) 123-45-67, Email:
gold@ros.ru
  2 | ИП "Серебряный ветер"   | Телефон: +7 (495) 234-56-78, Email:
silver@wind.ru
  3 | АО "Бриллиант Плюс"     | Телефон: +7 (495) 345-67-89, Email:
brillant@plus.ru
  4 | ООО "Драгоценные камни" | Телефон: +7 (495) 456-78-90, Email:
gems@precious.ru
```

```
(4 rows)

jewelry_store=*# SELECT * FROM suppliers;
 id |          name           |                      contact_info
----+-------------------------+-----------------------------------------------
-------
  1 | ООО "Золотая россыпь"   | Телефон: +7 (495) 123-45-67, Email:
gold@ros.ru
  2 | ИП "Серебряный ветер"   | Телефон: +7 (495) 234-56-78, Email:
silver@wind.ru
  3 | АО "Бриллиант Плюс"     | Телефон: +7 (495) 345-67-89, Email:
brillant@plus.ru
  4 | ООО "Драгоценные камни" | Телефон: +7 (495) 456-78-90, Email:
gems@precious.ru
(4 rows)

jewelry_store=*# COMMIT;
COMMIT
jewelry_store=# SELECT * FROM suppliers;
 id |          name           |                      contact_info
----+-------------------------+-----------------------------------------------
-------
  1 | ООО "Золотая россыпь"   | Телефон: +7 (495) 123-45-67, Email:
gold@ros.ru
  2 | ИП "Серебряный ветер"   | Телефон: +7 (495) 234-56-78, Email:
silver@wind.ru
  3 | АО "Бриллиант Плюс"     | Телефон: +7 (495) 345-67-89, Email:
brillant@plus.ru
  4 | ООО "Драгоценные камни" | Телефон: +7 (495) 456-78-90, Email:
gems@precious.ru
  9 | Test Supplier           | Phone: +99999999
```

Параллельная сессия:

```
jewelry_store=# INSERT INTO suppliers (name, contact_info) VALUES ('Test
Supplier', 'Phone: +99999999');
INSERT 0 1
jewelry_store=# SELECT * FROM suppliers;
 id |          name           |                      contact_info
----+-------------------------+-----------------------------------------------
-------
  1 | ООО "Золотая россыпь"   | Телефон: +7 (495) 123-45-67, Email:
gold@ros.ru
  2 | ИП "Серебряный ветер"   | Телефон: +7 (495) 234-56-78, Email:
silver@wind.ru
```

```
  3 | АО "Бриллиант Плюс"      | Телефон: +7 (495) 345-67-89, Email:
brillant@plus.ru
  4 | ООО "Драгоценные камни" | Телефон: +7 (495) 456-78-90, Email:
gems@precious.ru
  9 | Test Supplier            | Phone: +99999999
(5 rows)
```

Комментарий: В режиме REPEATABLE READ транзакция видит снимок данных на момент начала транзакции и не видит изменений, сделанных другими транзакциями после ее начала, до момента завершения текущей транзакции.

SERIALIZABLE:

```
-- Сессия 1
jewelry_store=# BEGIN;
BEGIN
jewelry_store=*# SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET
jewelry_store=*# UPDATE products SET sale_price = 60000 WHERE id = 1;
UPDATE 1
jewelry_store=*# COMMIT;
COMMIT

-- Сессия 2
jewelry_store=# BEGIN;
BEGIN
jewelry_store=*# SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET
jewelry_store=*# UPDATE products SET sale_price = 65000 WHERE id = 1;
ERROR:  could not serialize access due to concurrent update
```

Комментарий: В режиме SERIALIZABLE попытка параллельного изменения данных приводит к ошибке сериализации для обеспечения целостности данных.

3. Модификация записей без завершения транзакции

Сессия A:

```
jewelry_store=# BEGIN;
BEGIN
jewelry_store=*# UPDATE products
jewelry_store-*# SET sale_price = 80000
jewelry_store-*# WHERE id = 1;
UPDATE 1
```

```
jewelry_store=*# COMMIT;
COMMIT
```

Сессия B:

```
jewelry_store=# BEGIN;
BEGIN
jewelry_store=*# UPDATE products
jewelry_store-*# SET sale_price = 85000
jewelry_store-*# WHERE id = 1;
UPDATE 1
```

4. Эксперимент с SERIALIZABLE

Сессия A:

```
jewelry_store=# BEGIN;
BEGIN
jewelry_store=*# UPDATE products
SET sale_price = 90000
WHERE id = 1;
UPDATE 1
jewelry_store=*# COMMIT;
COMMIT
```

Сессия B (SERIALIZABLE):

```
jewelry_store=# BEGIN;
BEGIN
jewelry_store=*# SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET
jewelry_store=*# SELECT id, name, sale_price
jewelry_store-*# FROM products
jewelry_store-*# WHERE id = 1;
 id |      name       | sale_price
----+-----------------+------------
  1 | Золотое кольцо |      85000
(1 row)

jewelry_store=*# UPDATE products
jewelry_store-*# SET sale_price = 95000
jewelry_store-*# WHERE id = 1;
ERROR:  could not serialize access due to concurrent update
```

Заключение

В ходе лабораторной работы были изучены различные уровни изоляции транзакций в PostgreSQL:

1. READ COMMITTED - базовый уровень изоляции, обеспечивающий видимость только зафиксированных изменений
2. REPEATABLE READ - обеспечивает согласованное чтение данных в рамках транзакции
3. SERIALIZABLE - наиболее строгий уровень изоляции, предотвращающий аномалии параллельного доступа