

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Бази даних та інформаційні системи

ЛАБОРАТОРНА РОБОТА №10

Тема: «Транзакції в СКБД PostgreSQL»

Виконав:

Ст. Лук'янчук Денис

Група ПМІ-33

2025

Тема: «Транзакції в СКБД PostgreSQL».

Мета роботи: Ознайомлення з використанням транзакцій, їх розробкою та застосуванням, рівнями ізоляції та механізмом управління одночасним доступом у СКБД PostgreSQL.

Завдання лабораторної роботи:

- Опрацювати теоретичний матеріал щодо транзакцій та їх властивостей (атомарність, узгодженість, ізоляція, довговічність).
- Розробити кілька транзакцій для бази даних `spare_parts_shop`, демонструючи команди `BEGIN`, `COMMIT`, `SAVEPOINT` та `ROLLBACK TO`.
- На прикладі паралельного виконання двох транзакцій заданого рівня ізоляції пояснити їх роботу з погляду видимості змін під час виконання та після фіксації.

Теоретичний матеріал

Транзакції є основою роботи реляційних баз даних, об'єднуючи послідовність операцій за принципом «все або нічого». Вони характеризуються чотирма ключовими властивостями:

- **Атомарність:** Гарантує, що транзакція виконується повністю або не виконується зовсім.
- **Узгодженість:** Забезпечує відповідність змін у базі даних визначеним правилам.
- **Ізоляція:** Визначає, коли зміни однієї транзакції стають видимими для інших (залежить від рівня ізоляції).
- **Довговічність:** Гарантує збереження зафіксованих результатів навіть після збою системи.

У PostgreSQL транзакції керуються командами `BEGIN`, `COMMIT` і `ROLLBACK`. Для детального контролю використовуються точки збереження (`SAVEPOINT`) з можливістю відкатів (`ROLLBACK TO`). Рівні ізоляції (`Read Committed`, `Repeatable Read`, `Serializable`) регулюють видимість змін:

- **Read Committed:** Дозволяє бачити лише зафіксовані зміни інших транзакцій.
- **Repeatable Read:** Захищає від неповторних читань і фантомних записів.
- **Serializable:** Гарантує повну ізоляцію, еквівалентну послідовному виконанню, але може спричиняти конфлікти.

Хід роботи

База даних `spare_parts_shop` моделює магазин автозапчастин, де транзакції керують замовленнями (`Order`), клієнтами (`Customer`) та запчастинами (`Part`). Усі приклади виконуються в двох паралельних сеансах `pgAdmin` для демонстрації конкуренції.

Пара 1: Оновлення кількості замовлення (Рівень ізоляції `Read Committed`)

- Транзакція 1: Збільшує кількість замовлення для клієнта Alice (`Customer_ID=1`) з використанням `SAVEPOINT` і `ROLLBACK TO`.
- Транзакція 2: Перевіряє загальну суму замовлень для Alice до й після оновлення.
- Мета: Демонстрація видимості зафіксованих змін.

Пара 2: Оновлення статусу замовлення (Рівень ізоляції Serializable)

- Транзакція 1: Змінює статус замовлення на "Shipped" з SAVEPOINT і ROLLBACK TO.
- Транзакція 2: Спроба змінити кількість замовлення.
- Мета: Демонстрація найсуворішого рівня ізоляції з можливими конфліктами.

Пара 1: Оновлення кількості замовлення (Read Committed)

Транзакція 1 (Сеанс 1):

1 SELECT o.*, p.Name, p.Price

2 FROM "Order" o

3 JOIN Part p ON o.Part_ID = p.Part_ID

4 WHERE o.Customer_ID = 85;

Data Output

Messages

Notifications

SQL

Showing rows: 1 to 3

Page No: 1 of 1

	order_id integer	customer_id integer	part_id integer	order_date timestamp without time zone	status character varying (20)	quantity integer	name character varying (200)	price numeric (10,2)
1	1	85	35	2024-12-17 19:17:52.265148	Processing	9	Part_35	81.72
2	99	85	38	2024-12-21 08:35:52.862473	Cancelled	9	Part_38	126.25
3	64	85	77	2025-08-21 17:39:07.788162	New	7	Part_77	719.07

1 BEGIN;

2 UPDATE "Order" SET Quantity = Quantity + 1

3 WHERE Customer_ID = 85 AND Part_ID = (SELECT Part_ID FROM Part WHERE Serial_Number = 'SN001');

4 SAVEPOINT update_quantity;

5 SELECT Customer_ID, Quantity

6 FROM "Order"

7 WHERE Customer_ID = 85 AND Part_ID = (SELECT Part_ID FROM Part WHERE Serial_Number = 'SN001');

8 ROLLBACK TO update_quantity;

9 COMMIT;

Data Output

Messages

Notifications

COMMIT

1 SELECT o.*, p.Name, p.Price

2 FROM "Order" o

3 JOIN Part p ON o.Part_ID = p.Part_ID

4 WHERE o.Customer_ID = 85;

Data Output

Messages

Notifications

SQL

Showing rows: 1 to 3

Page No: 1 of 1

	order_id integer	customer_id integer	part_id integer	order_date timestamp without time zone	status character varying (20)	quantity integer	name character varying (200)	price numeric (10,2)
1	1	85	35	2024-12-17 19:17:52.265148	Processing	10	Part_35	81.72
2	99	85	38	2024-12-21 08:35:52.862473	Cancelled	9	Part_38	126.25
3	64	85	77	2025-08-21 17:39:07.788162	New	7	Part_77	719.07

Транзакція 2 (Сеанс 2):

```
1 SELECT o.*, p.Name, p.Price
2 FROM "Order" o
3 JOIN Part p ON o.Part_ID = p.Part_ID
4 WHERE o.Customer_ID = 85;
```

Data Output Messages Notifications

	order_id integer	customer_id integer	part_id integer	order_date timestamp without time zone	status character varying (20)	quantity integer	name character varying (200)	price numeric (10,2)
1	1	85	35	2024-12-17 19:17:52.265148	Processing	11	Part_35	81.72
2	99	85	38	2024-12-21 08:35:52.862473	Cancelled	9	Part_38	126.25
3	64	85	77	2025-08-21 17:39:07.788162	New	7	Part_77	719.07

```
1 BEGIN;
2 SELECT SUM(o.Quantity * p.Price) AS total_amount
3 FROM "Order" o
4 JOIN Part p ON o.Part_ID = p.Part_ID
5 WHERE o.Customer_ID = 85;
```

Data Output Messages Notifications

	total_amount numeric
1	7068.66

```
1 SELECT SUM(o.Quantity * p.Price) AS total_amount
2 FROM "Order" o
3 JOIN Part p ON o.Part_ID = p.Part_ID
4 WHERE o.Customer_ID = 85;
```

Data Output Messages Notifications

	total_amount numeric
1	7068.66

Пара 2: Оновлення статусу замовлення (Serializable)

Транзакція 1 (Сеанс 1):

```
1 SELECT o.*, p.Name, p.Price
2 FROM "Order" o
3 JOIN Part p ON o.Part_ID = p.Part_ID
4 WHERE o.Customer_ID = 85;
```

Data Output Messages Notifications

	order_id integer	customer_id integer	part_id integer	order_date timestamp without time zone	status character varying (20)	quantity integer	name character varying (200)	price numeric (10,2)
1	1	85	35	2024-12-17 19:17:52.265148	Processing	11	Part_35	81.72
2	99	85	38	2024-12-21 08:35:52.862473	Cancelled	9	Part_38	126.25
3	64	85	77	2025-08-21 17:39:07.788162	New	7	Part_77	719.07

```

1 BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
2 ✓ UPDATE "Order" SET Status = 'Completed'
3 WHERE Customer_ID = 85 AND Part_ID = (SELECT Part_ID FROM Part WHERE Serial_Number = 'SN000035');
4 SAVEPOINT update_status;
5 ✓ SELECT Status
6 FROM "Order"
7 WHERE Customer_ID = 85 AND Part_ID = (SELECT Part_ID FROM Part WHERE Serial_Number = 'SN000035');
8 ROLLBACK TO update_status;
9 COMMIT;

```

Data Output Messages Notifications

COMMIT

Query returned successfully in 28 msec.

```

1 ✓ SELECT o.*, p.Name, p.Price
2 FROM "Order" o
3 JOIN Part p ON o.Part_ID = p.Part_ID
4 WHERE o.Customer_ID = 85;

```

Data Output Messages Notifications

	order_id integer	customer_id integer	part_id integer	order_date timestamp without time zone	status character varying (20)	quantity integer	name character varying (200)	price numeric (10,2)
1	1	85	35	2024-12-17 19:17:52.265148	Completed	11	Part_35	81.72
2	99	85	38	2024-12-21 08:35:52.862473	Cancelled	9	Part_38	126.25
3	64	85	77	2025-08-21 17:39:07.788162	New	7	Part_77	719.07

Транзакція 2 (Сеанс 2):

```

1 ✓ SELECT o.*, p.Name, p.Price
2 FROM "Order" o
3 JOIN Part p ON o.Part_ID = p.Part_ID
4 WHERE o.Customer_ID = 85;

```

Data Output Messages Notifications

	order_id integer	customer_id integer	part_id integer	order_date timestamp without time zone	status character varying (20)	quantity integer	name character varying (200)	price numeric (10,2)
1	1	85	35	2024-12-17 19:17:52.265148	Completed	11	Part_35	81.72
2	99	85	38	2024-12-21 08:35:52.862473	Cancelled	9	Part_38	126.25
3	64	85	77	2025-08-21 17:39:07.788162	New	7	Part_77	719.07

```

1 BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
2 ✓ UPDATE "Order" SET Quantity = Quantity + 1
3 WHERE Customer_ID = 85 AND Part_ID = (SELECT Part_ID FROM Part WHERE Serial_Number = 'SN000035');
4 ✓ SELECT Quantity
5 FROM "Order"
6 WHERE Customer_ID = 85 AND Part_ID = (SELECT Part_ID FROM Part WHERE Serial_Number = 'SN000035');
7 COMMIT;

```

Data Output Messages Notifications

COMMIT

Query returned successfully in 49 msec.

```

1 SELECT o.*, p.Name, p.Price
2 FROM "Order" o
3 JOIN Part p ON o.Part_ID = p.Part_ID
4 WHERE o.Customer_ID = 85;

```

Data Output Messages Notifications

	order_id integer	customer_id integer	part_id integer	order_date timestamp without time zone	status character varying (20)	quantity integer	name character varying (200)	price numeric (10,2)
1	1	85	35	2024-12-17 19:17:52.265148	Completed	12	Part_35	81.72
2	99	85	38	2024-12-21 08:35:52.862473	Cancelled	9	Part_38	126.25
3	64	85	77	2025-08-21 17:39:07.788162	New	7	Part_77	719.07

Висновок:

У ході виконання лабораторної роботи №10 «Транзакції в СКБД PostgreSQL» було продемонстровано принципи роботи транзакцій у реляційних базах даних на прикладі системи spare_parts_shop. Розроблено та протестовано дві пари паралельних транзакцій, які відображають різні рівні ізоляції та механізми управління одночасним доступом.

Перша пара, виконана на рівні ізоляції Read Committed, показала, що зміни стають видимими для інших транзакцій лише після їх фіксації командою COMMIT. Використання SAVEPOINT і ROLLBACK TO дозволило гнучко керувати оновленнями кількості замовлень, демонструючи атомарність і контроль над проміжними станами.

Друга пара, реалізована на рівні ізоляції Serializable, підкреслила найсуворіший підхід до ізоляції, де паралельні оновлення статусу та кількості замовлення можуть призводити до конфліктів із помилкою "could not serialize". Це гарантує повну послідовність, але вимагає повторного запуску транзакцій у разі зриву.

Робота підтвердила, що правильний вибір рівня ізоляції та використання точок збереження є ключовими для забезпечення цілісності даних у багатопотоковому середовищі. Виконані транзакції відповідають властивостям ACID, забезпечуючи атомарність, узгодженість, ізоляцію та довговічність, що є основою надійної роботи СКБД PostgreSQL.

Відповіді на контрольні питання

1. Поясніть у чому суть поняття транзакції.

Транзакція – це послідовність операцій над базою даних, яка виконується як єдине ціле. Вона або повністю виконується, або повністю відміняється, забезпечуючи цілісність даних.

2. Назвіть властивості транзакцій.

Властивості описуються акронімом ACID:

- Атомарність (Atomicity)

- **Узгодженість (Consistency)**
- **Ізоляція (Isolation)**
- **Довговічність (Durability)**

3. Що означає атомарність транзакції?

Атомарність означає, що всі операції в межах транзакції виконуються як одне ціле: або всі виконані, або жодна (відкат у разі помилки).

4. Що означає узгодженість транзакції?

Узгодженість гарантує, що після виконання транзакції база даних переходить з одного правильного (узгодженого) стану в інший, не порушуючи обмеження цілісності.

5. Що означає ізоляція транзакції?

Ізоляція означає, що паралельні транзакції не впливають одна на одну так, щоб порушити правильність результатів. Кожна транзакція виконується так, ніби вона єдина в системі.

6. Що означає довговічність транзакції?

Довговічність означає, що після підтвердження (COMMIT) результат транзакції зберігається в базі даних і не втрачається навіть у разі збоїв.

7. Що таке точка збереження транзакції?

Точка збереження (SAVEPOINT) – це проміжна позначка всередині транзакції, до якої можна зробити частковий відкат, не скасовуючи всієї транзакції.

8. Що таке відкат виконання транзакції?

Відкат (ROLLBACK) – це скасування змін, зроблених у межах транзакції, з поверненням бази даних у початковий стан.

9. Які є рівні ізоляції транзакцій?

Стандарт SQL визначає 4 рівні ізоляції:

- **Read Uncommitted** (читання непідтверджених даних)
- **Read Committed** (читання тільки підтверджених даних)
- **Repeatable Read** (повторювані читання гарантують незмінність раніше прочитаних рядків)
- **Serializable** (повна ізоляція, транзакції виконуються ніби послідовно)

10. На основі якого механізму реалізовано рівні ізоляції транзакцій?

Рівні ізоляції реалізовані на основі механізму блокувань (locks) і/або механізму багатовершинного контролю версій (MVCC – Multiversion Concurrency Control).

11. Що означає поняття «брудне читання»?

Брудне читання (Dirty Read) – це коли транзакція читає дані, змінені іншою транзакцією, яка ще не підтверджена (і може бути відкотана).

12. Що означає поняття «неповторюване читання»?

Неповторюване читання (Non-Repeatable Read) – це коли транзакція двічі читає один і той самий рядок, але між читаннями інша транзакція його змінила, тому значення відрізняються.

13. Що означає поняття «фантомне читання»?

Фантомне читання (Phantom Read) – це коли при повторному виконанні одного й того ж запиту умови вибірки повертають різну кількість рядків через вставку або видалення іншою транзакцією.

14. Що означає поняття «аномалія серіалізації»?

Аномалія серіалізації – це ситуація, коли результат паралельного виконання транзакцій не еквівалентний жодному можливому послідовному порядку їх виконання, що може призвести до некоректних результатів.

