Проектування високонавантажених систем

Лабораторна робота №2

Реалізація каунтера з використанням PostgreSQL

Виконав: Мартиненко Денис ФБ-42мп

- Файл docker-compose.yml:

```yaml
services:
    postgres:
        container_name: lab2
        environment:
            - POSTGRES_DB=lab2
            - POSTGRES_USER=den4ik
            - POSTGRES_PASSWORD=12345
        ports:
            - '5432:5432'
        image: 'postgres:latest'
```

Реалізація:

```python
import psycopg2
import threading
import time


def initialize_database():
    connection = psycopg2.connect(
        host="localhost",
        dbname="lab2",
        user="den4ik",
        password="12345"
    )
    cursor = connection.cursor()

    # Створення таблиці з полем для версії
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS user_counter (
            user_id SERIAL PRIMARY KEY,
            counter INTEGER DEFAULT 0,
            version INTEGER DEFAULT 0
        );
    ''')

    # Ініціалізація значень
    for user_id in range(1, 5):
        cursor.execute('''
            INSERT INTO user_counter (user_id, counter, version)
            VALUES (%s, 0, 0)
            ON CONFLICT (user_id)
            DO NOTHING;
        ''', (user_id,))

    connection.commit()
    cursor.close()
    connection.close()

```

```python
def print_counter(user_id):
    connection = psycopg2.connect(
        host="localhost",
        dbname="lab2",
        user="den4ik",
        password="12345"
    )
    cursor = connection.cursor()
    cursor.execute('SELECT counter FROM user_counter WHERE user_id = %s;', (user_id,))
    counter_value = cursor.fetchone()[0]
    print(f"Counter for user_id {user_id}: {counter_value}")
    cursor.close()
    connection.close()
```

```python
if __name__ == '__main__':
    initialize_database()

    tasks = [
        ("Lost-update", lost_update_task, 1),
        ("In-place update", in_place_update_task, 2),
        ("Row-level locking", row_level_locking_task, 3),
        ("Optimistic concurrency control", optimistic_concurrency_control_task, 4)
    ]

    for task_name, task, user_id in tasks:
        print(f"Starting {task_name}...")
        threads = [threading.Thread(target=task) for _ in range(10)]
        start_time = time.time()

        # Запуск потоків
        for thread in threads:
            thread.start()

        # Очікування завершення всіх потоків
        for thread in threads:
            thread.join()

        print(f"Time for {task_name}: {time.time() - start_time} seconds")
        print_counter(user_id)
```

1. Lost-update

```python
def lost_update_task():
    connection = psycopg2.connect(
        host="localhost",
        dbname="lab2",
        user="den4ik",
        password="12345"
    )
    cursor = connection.cursor()

    for _ in range(10000):
        cursor.execute('SELECT counter FROM user_counter WHERE user_id = 1;')
        current_value = cursor.fetchone()[0]

        # Інкремент значення
        new_value = current_value + 1

        # Оновлення таблиці
        cursor.execute('UPDATE user_counter SET counter = %s WHERE user_id = 1;', (new_value,))
        connection.commit()

    cursor.close()
    connection.close()
```

```
Starting Lost-update...
Time for Lost-update: 341.80020570755005 seconds
Counter for user_id 1: 10894
```

2. In-place update

```python
def in_place_update_task():
    connection = psycopg2.connect(
        host="localhost",
        dbname="lab2",
        user="den4ik",
        password="12345"
    )
    cursor = connection.cursor()

    for _ in range(10000):
        # Оновлення значення каунтера без попереднього читання
        cursor.execute('UPDATE user_counter SET counter = counter + 1 WHERE user_id = 2;')
        connection.commit()

    cursor.close()
    connection.close()
```

```
Starting In-place update...
Time for In-place update: 350.8868672847748 seconds
Counter for user_id 2: 100000
```

3. Row-level locking

```python
def row_level_locking_task():
    connection = psycopg2.connect(
        host="localhost",
        dbname="lab2",
        user="den4ik",
        password="12345"
    )
    cursor = connection.cursor()

    for _ in range(10000):
        # Використання блокування на рівні рядка для уникнення втрат оновлень
        cursor.execute('SELECT counter FROM user_counter WHERE user_id = 3 FOR UPDATE;')
        current_value = cursor.fetchone()[0]
        new_value = current_value + 1

        cursor.execute('UPDATE user_counter SET counter = %s WHERE user_id = 3;', (new_value,))
        connection.commit()

    cursor.close()
    connection.close()
```

```
Starting Row-level locking...
Time for Row-level locking: 430.539089679718 seconds
Counter for user_id 3: 100000
```

4. Optimistic concurrency control

```python
def optimistic_concurrency_control_task():
    connection = psycopg2.connect(
        host="localhost",
        dbname="lab2",
        user="den4ik",
        password="12345"
    )
    cursor = connection.cursor()

    successful_updates = 0

    while successful_updates < 10000:  # Кожен потік повинен виконати рівно 10000 успішних оновлень
        # Отримання поточного значення каунтера і версії
        cursor.execute('SELECT counter, version FROM user_counter WHERE user_id = 4;')
        current_value, version = cursor.fetchone()

        # Інкремент значення
        new_value = current_value + 1

        # Оновлення з використанням контролю версії
        cursor.execute('''
            UPDATE user_counter
            SET counter = %s, version = %s
            WHERE user_id = 4 AND version = %s;
        ''', (new_value, version + 1, version))

        connection.commit()

        # Перевіряємо, чи було оновлено хоча б один рядок
        if cursor.rowcount > 0:
            successful_updates += 1

    cursor.close()
    connection.close()
```

```
Starting Optimistic concurrency control...
Time for Optimistic concurrency control: 2590.5291588306427 seconds
Counter for user_id 4: 100000
```