МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Проектування високонавантажених систем

Лабораторна робота №2

Реалізація лічильника з використанням PostgreSQL

Перевірив:
Родіонов А. М.

Виконав:
студент І курсу
групи ФБ-41мп
Сахній Н. Р.

Київ 2024

Мета роботи: Необхідно декількома способами резалізувати оновлення значення каунтера в СКБД PostgreSQL та оцінити час кожного із варіантів.

Завдання до виконання:

0. INITIALIZATION of DBMS

1) `pip install psycopg2`

2) `cat ./docker-compose.yml`

```yaml
services:
    postgresql_dbms:
        container_name: 'massive_insert-postgresql'
        environment:
            - POSTGRES_DB=massive_insert-postgresql
            - POSTGRES_USER=sazan24
            - POSTGRES_PASSWORD=UIOP1234
        ports:
            - '5432:5432'
        image: 'postgres:latest'
```

3) `doccker-compose up`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Магістрат/Проектування ВС/Task_2-MassiveInsertPostgreSQL
$ docker-compose up
Creating massive_insert-postgresql ...
Creating massive_insert-postgresql ... done
Attaching to massive_insert-postgresql
massive_insert-postgresql | The files belonging to this database system will be owned by user "postgres".
massive_insert-postgresql | This user must also own the server process.
massive_insert-postgresql |
massive_insert-postgresql | The database cluster will be initialized with locale "en_US.utf8".
massive_insert-postgresql | The default database encoding has accordingly been set to "UTF8".
massive_insert-postgresql | The default text search configuration will be set to "english".
massive_insert-postgresql |
massive_insert-postgresql | Data page checksums are disabled.
massive_insert-postgresql |
massive_insert-postgresql | fixing permissions on existing directory /var/lib/postgresql/data ... ok
massive_insert-postgresql | creating subdirectories ... ok
massive_insert-postgresql | selecting dynamic shared memory implementation ... posix
massive_insert-postgresql | selecting default "max_connections" ... 100
massive_insert-postgresql | selecting default "shared_buffers" ... 128MB
massive_insert-postgresql | selecting default time zone ... Etc/UTC
massive_insert-postgresql | creating configuration files ... ok
massive_insert-postgresql | running bootstrap script ... ok
massive_insert-postgresql | performing post-bootstrap initialization ... ok
massive_insert-postgresql | syncing data to disk ... initdb: warning: enabling "trust" authentication for local connections
massive_insert-postgresql | initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local
massive_insert-postgresql | ok
massive_insert-postgresql |
massive_insert-postgresql |
massive_insert-postgresql | Success. You can now start the database server using:
```

4) `connect_db() & create_table()`

```python
# Функція для встановлення з'єднання з базою даних
def connect_db():
    return psycopg2.connect("host=localhost dbname=massive_insert-postgresql user=sazan24 password=UIOP1234")
```

```python
# Функція для створення таблиці та ініціалізації даних
def create_table():
    connect = connect_db()
    cursor = connect.cursor()

    cursor.execute("CREATE TABLE user_counter (user_id serial PRIMARY KEY, counter integer, version integer);")

    cursor.execute("INSERT INTO user_counter (user_id, counter, version) VALUES (1, 0, 0);")
    cursor.execute("INSERT INTO user_counter (user_id, counter, version) VALUES (2, 0, 0);")
    cursor.execute("INSERT INTO user_counter (user_id, counter, version) VALUES (3, 0, 0);")
    cursor.execute("INSERT INTO user_counter (user_id, counter, version) VALUES (4, 0, 0);")

    connect.commit()
    cursor.close()
    connect.close()
```

5) `select * from user_counter`

```
user_id, counter, version
(1, 0, 0)
(2, 0, 0)
(3, 0, 0)
(4, 0, 0)
```

1. LOST UPDATE

1) Snippet of Code – **lost_update**

```python
# Функція, яка демонструє сутність проблеми втраченого оновлення
def lost_update():
    connect = connect_db()
    cursor = connect.cursor()

    for _ in range(10000):
        counter = cursor.execute("SELECT counter FROM user_counter WHERE user_id = 1;")
        counter = cursor.fetchone()[0] + 1
        cursor.execute(f"UPDATE user_counter SET counter = {counter} WHERE user_id = 1;")
        connect.commit()

    cursor.close()
    connect.close()
```

2) Executing Result – **lost_update**

```
-----------------------------------------------------
Executing 1. Lost-Update...
-----------------------------------------------------
Task '1. Lost-Update' executed in: 211.37 seconds
Final counter value: 10424
-----------------------------------------------------
```

2. IN-PLACE UPDATE

1) Snippet of Code – **inplace_update**

```python
# Функція, яка адекватно оновлює значення лічильника "на місці"
def inplace_update():
    connect = connect_db()
    cursor = connect.cursor()

    for _ in range(10000):
        cursor.execute("UPDATE user_counter SET counter = counter + 1 WHERE user_id = 2;")
        connect.commit()

    cursor.close()
    connect.close()
```

2) Executing Result – **inplace_update**

```
--------------------------------------------------
Executing 2. In-Place Update...
--------------------------------------------------
Task '2. In-Place Update' executed in: 206.37 seconds
Final counter value: 100000
--------------------------------------------------
```

3. ROW-LEVEL LOCKING

1) Snippet of Code – **rowlevel_locking**

```python
# Функція, яка використовує принцип блокувань на рівні рядків
def rowlevel_locking():
    for _ in range(10000):
        connect = connect_db()
        cursor = connect.cursor()

        cursor.execute("SELECT counter FROM user_counter WHERE user_id = 3 FOR UPDATE;")
        counter = cursor.fetchone()[0] + 1

        cursor.execute(f"UPDATE user_counter SET counter = {counter} WHERE user_id = 3;")
        connect.commit()

        cursor.close()
        connect.close()
```

2) Executing Result – **rowlevel_locking**

```
--------------------------------------------------
Executing 3. Row-Level Locking...
--------------------------------------------------
Task '3. Row-Level Locking' executed in: 555.50 seconds
Final counter value: 100000
--------------------------------------------------
```

4. OPTIMISTIC CONCURRENCY

1) Snippet of Code – **concurrency_control**

```python
# Функція, що використовує механізм керування паралельним доступом
def concurrency_control():
    connect = connect_db()
    cursor = connect.cursor()

    for _ in range(10000):
        while True:
            cursor.execute("SELECT counter, version FROM user_counter WHERE user_id = 4;")
            counter, version = cursor.fetchone()

            counter += 1
            cursor.execute(f"UPDATE user_counter SET counter = {counter}, version = {version + 1} \
                           WHERE user_id = 4 and version = {version};")
            connect.commit()

            count = cursor.rowcount
            if count > 0: break

    cursor.close()
    connect.close()
```

2) Executing Result – **concurrency_control**

```
-----------------------------------------------------
Executing 4. Concurrency Control...
-----------------------------------------------------
Task '4. Concurrency Control' executed in: 1635.24 seconds
Final counter value: 100000
-----------------------------------------------------
```

5. CURRENT CONTENT of TABLE

```
user_id, counter, version
(1, 10424, 0)
(2, 100000, 0)
(3, 100000, 0)
(4, 100000, 100000)
```

* ADDITIONAL FUNCTIONS

o view_table()

```python
# Функція для перегляду всіх записів у таблиці даних
def view_table():
    connect = connect_db()
    cursor = connect.cursor()

    cursor.execute("SELECT * FROM user_counter ORDER BY user_id;")
    table = cursor.fetchall()
    print("user_id, counter, version")
    for row in table: print(row)

    connect.commit()
    cursor.close()
    connect.close()
```

o drop_table

```python
# Функція для видалення таблиці даних "user_counter"
def drop_table():
    connect = connect_db()
    cursor = connect.cursor()

    cursor.execute("DROP TABLE user_counter;")

    connect.commit()
    cursor.close()
    connect.close()
```

o print_counter(i)

```python
# Функція для виведення значення лічильника з ID
def print_counter(i):
    connect = connect_db()
    cursor = connect.cursor()

    cursor.execute(f"SELECT counter FROM user_counter WHERE user_id = {i};")
    print(f"Final counter value: {cursor.fetchone()[0]}")

    connect.commit()
    cursor.close()
    connect.close()
```

o update_counter(i)

```python
# Функція для скидання лічильника та версії з ID
def update_counter(i):
    connect = connect_db()
    cursor = connect.cursor()

    cursor.execute(f"UPDATE user_counter SET counter = 0 WHERE user_id = {i};")
    cursor.execute(f"UPDATE user_counter SET version = 0 WHERE user_id = {i};")

    connect.commit()
    cursor.close()
    connect.close()
```