Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа программной инженерии

Отчет по лабораторной работе

1.4 «Контроль целостности данных»

по дисциплине

«Системы управления базами данных»

Выполнила

Студентка гр. 5130904/10101 Никифорова Е. А.

Руководитель Прокофьев О. В.

Санкт-Петербург

2024

Постановка задачи

Необходимо подготовить SQL-скрипты для проверки наличия аномалий (потерянных изменений, грязных чтений, неповторяющихся чтений, фантомов) при параллельном исполнении транзакций на различных уровнях изолированности SQL/92 (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE). Подготовленные скрипты должны работать с таблицей, содержащей 3-4 атрибута и до 30 строк. Для проверки наличия аномалий потребуются два параллельных сеанса, операторы в которых выполняются пошагово:

* Установить в обоих сеансах уровень изоляции READ COMMITED/READ UNCOMMITTED (для postgresql это синонимы). Выполнить сценарии проверки наличия аномалий потерянных изменений и грязных чтений.
* Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки наличия аномалий грязных чтений и неповторяющихся чтений.
* Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки наличия аномалий неповторяющихся чтений и фантомов, сериализации.
* Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценарий проверки фиксации аномалии сериализации.
* Необходимо составить скрипт для создания 1-2х триггеров, а также подготовить несколько запросов для проверки и демонстрации его полезных свойств:
  + Изменение данных для сохранения целостности.
  + Проверка транзакций и их откат в случае нарушения целостности.



Решение:

CREATE TABLE lights (id serial, lamp text, state text);

INSERT INTO lights (lamp, state) VALUES (‘red’, ‘on’), (‘green’, ‘off’);

SELECT \* FROM lights;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Уровень изоляции** | **Транзакция А** | **Транзакция В** | **Вывод** | **Аномалии транзакции** |
| READ UNCOMMITTED | BEGIN ISOLATION LEVEL READ UNCOMMITTED;  SELECT \* FROM lights ORDER BY id; |  | id | lamp | state  ----+-------+-------  1 | red | on  2 | green | off | Грязное чтение  (не происходит, В не видит незафиксированное изменение А) |
|  | BEGIN ISOLATION LEVEL READ UNCOMMITTED; |  |
| UPDATE lights SET state = 'off' WHERE lamp = 'red';  SELECT \* FROM lights ORDER BY id; |  | id | lamp | state  ----+-------+-------  1 | red | off  2 | green | off |
|  | SELECT \* FROM lights ORDER BY id; | id | lamp | state  ----+-------+-------  1 | red | on  2 | green | off |
| ROLLBACK; |  |  |
|  | COMMIT; |  |
|  | | | | |
|  | BEGIN ISOLATION LEVEL READ UNCOMMITTED;  SELECT \* FROM lights;  UPDATE lights SET state = ‘off’ WHERE lamp = ‘red’; |  | id | lamp | state  ----+-------+-------  2 | green | on  1 | red | on  UPDATE 1 | Потерянные изменения (не происходит, В не игнорирует изменения А) |
|  |  | BEGIN ISOLATION LEVEL READ UNCOMMITTED;  UPDATE lights SET state = CASE WHEN state = ‘on’ THEN ‘off’ ELSE ‘on’ END WHERE lamp = ‘red’; |  |  |
|  | COMMIT; |  | UPDATE 1 |  |
|  |  | SELECT \* FROM lights; | id | lamp | state  ----+-------+-------  2 | green | on  1 | red | on |  |
|  |  | COMMIT; |  |  |
|  | | | | |
| READ COMMITTED | BEGIN ISOLATION LEVEL READ COMMITTED;  SELECT \* FROM lights ORDER BY id; |  | id | lamp | state  ----+-------+-------  1 | red | on  2 | green | off | Грязное чтение  (не происходит, во время выполнения оператора А, В успела зафиксировать изменения, но их не видно) |
|  |  | BEGIN ISOLATION LEVEL READ COMMITTED; |  |
|  | UPDATE lights SET state = 'off' WHERE lamp = 'red';  SELECT \* FROM lights ORDER BY id; |  | id | lamp | state  ----+-------+-------  1 | red | off  2 | green | off |
|  |  | SELECT \* FROM lights ORDER BY id; | id | lamp | state  ----+-------+-------  1 | red | on  2 | green | off |
|  | ROLLBACK; |  |  |
|  |  | COMMIT; |  |
|  | | | | |
|  | BEGIN ISOLATION LEVEL READ COMMITTED;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  1 | red | on  2 | green | off | Неповторяющееся чтение  (B видит зафиксированное изменение А при повторном чтении) |
|  | UPDATE lights SET state = ‘off’ WHERE lamp = ‘red’;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | off |
|  |  | BEGIN ISOLATION LEVEL READ COMMITTED;  SELECT \* FROM lights; | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | on |
|  | COMMIT; |  |  |
|  |  | SELECT \* FROM lights;  COMMIT; | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | off |
| UPDATE lights SET state = ‘on’ WHERE lamp = ‘red’; | | | | |
|  |  |  |  |  |
| REPEATABLE READ | BEGIN ISOLATION LEVEL REPEATABLE READ;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | on | Неповторяющееся чтение (не происходит, B не видит зафиксированное изменение А при повторном чтении) |
|  | UPDATE lights SET state = ‘off’ WHERE lamp = ‘red’;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | off |
|  |  | BEGIN ISOLATION LEVEL REPEATABLE READ;  SELECT \* FROM lights; | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | on |
|  | COMMIT; |  |  |
|  |  | SELECT \* FROM lights;  COMMIT; | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | on |
|  | | | | |
|  | BEGIN ISOLATION LEVEL REPEATABLE READ;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | off | Фантомное чтение (не происходит, в процессе операции А, В изменяет данные, но А не видит изменения) |
|  |  | BEGIN ISOLATION LEVEL REPEATABLE READ; |  |
|  | SELECT pg\_sleep(3), \* FROM lights; |  |  |
|  |  | UPDATE lights SET state = ‘on’;  COMMIT; |  |
|  |  |  | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | off |
|  | COMMIT; |  |  |
|  | SELECT \* FROM lights; | | id | lamp | state  ----+-------+-------  2 | green | on  1 | red | on |
| UPDATE lights SET state = ‘off’ WHERE lamp = ‘red’; | | | | |
|  | | | | |
|  | BEGIN ISOLATION LEVEL REPEATABLE READ;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  2 | green | on  1 | red | off | Аномалия сериализации (отличается от результата последовательного выполнения операций) |
|  |  | BEGIN ISOLATION LEVEL REPEATABLE READ;  SELECT \* FROM lights; | id | lamp | state  ----+-------+-------  2 | green | on  1 | red | off |
|  | UPDATE lights SET state = ‘off’ WHERE state = ‘on’;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  1 | red | off  2 | green | off |
|  |  | UPDATE lights SET state = ‘on’ WHERE state = ‘off’;  SELECT \* FROM lights; | id | lamp | state  ----+-------+-------  2 | green | on  1 | red | on |
|  | COMMIT; |  |  |
|  |  | COMMIT; |  |
|  | SELECT \* FROM lights; | | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | on |
|  |  |  |  |  |
| SERIALIZABLE | BEGIN ISOLATION LEVEL SERIALIZABLE;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | on | Аномалия сериализации |
|  |  | BEGIN ISOLATION LEVEL SERIALIZABLE;  SELECT \* FROM lights; | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | on |
|  | UPDATE lights SET state = ‘off’ WHERE state = ‘on’;  SELECT \* FROM lights; |  | id | lamp | state  ----+-------+-------  2 | green | off  1 | red | off |
|  |  | UPDATE lights SET state = ‘on’ WHERE state = ‘off’;  SELECT \* FROM lights; | id | lamp | state  ----+-------+-------  1 | red | on  2 | green | on |
|  | COMMIT; |  |  |
|  |  | COMMIT; | ОШИБКА: не удалось сериализовать доступ из-за зависимостей чтения/записи между транзакциями  ПОДРОБНОСТИ: Reason code: Canceled on identification as a pivot, during commit attempt.  ПОДСКАЗКА: Транзакция может завершиться успешно при следующей попытке. |

**Триггер:**

CREATE OR REPLACE FUNCTION check\_state()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.state NOT IN ('on', 'off') THEN

RAISE EXCEPTION 'Invalid state';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check\_state\_trigger

BEFORE INSERT OR UPDATE ON lights

FOR EACH ROW

EXECUTE FUNCTION check\_state();

BEGIN;

INSERT INTO lights (lamp, state) VALUES ('green', 'invalid\_state');

COMMIT;

INSERT INTO lights (lamp, state) VALUES ('green', 'invalid\_state');

UPDATE lights SET state = 'invalid\_state' WHERE lamp = 'green';

Вывод:

В ходе лабораторной работы были изучены и протестированы аномалии такие как грязное чтение, потерянные изменения, неповторяющееся чтение, фантомное чтение, аномалия сериализации. Протестированы с различными уровнями изоляций. Также был создан триггер для проверки целостности данных, приведены примеры операций, на которых триггер срабатывает.