# Технологии обработки транзакций клиент-серверных приложений зима-весна 2022

ФИО преподавателя: Матчин Василий Тимофеевич

e-mail: matchin@mirea.ru

# Условия обучения

- По итогам изучения дисциплины проводится экзамен
- В течение семестра необходимо выполнить все задания по календарному плану, которые опубликованы на Учебном портале
- Баллы за активность до 25 баллов

ТЕМА

# Изоляция транзакций

# Аномалия потерянное обновление

При обновлении поля двумя транзакциями одно из изменений теряется.

| Транзакция 1 | Транзакция 2 |
|---|---|
| SELECT x FROM tbl WHERE y=1; | SELECT x FROM tbl WHERE y=1; |
| UPDATE tbl SET x=5 WHERE y=1; | |
| | UPDATE tbl SET x=3 WHERE y=1; |

Потерянное обновление не допускается стандартом ни на одном уровне изоляции.

# Аномалия грязное чтение

Чтение данных, полученных в результате действия транзакции, которая после этого откатится.

| Транзакция 1 | Транзакция 2 |
|---|---|
| SELECT x FROM tbl WHERE y=1; | |
| UPDATE tbl SET x=x+1 WHERE y=1; | |
| | SELECT x FROM tbl WHERE y=1; |
| ROLLBACK; | |

Грязное чтение допускается на уровне изоляции Read Uncommitted.

# Аномалия неповторяющееся чтение

Возникает, когда в течение одной транзакции при повторном чтении данные оказываются перезаписанными.

| Транзакция 1 | Транзакция 2 |
|---|---|
| SELECT x FROM tbl WHERE y=1; | SELECT x FROM tbl WHERE y=1; |
| UPDATE tbl SET x=x+1 WHERE y=1; | |
| COMMIT; | |
| | SELECT x FROM tbl WHERE y=1; |

Неповторяющееся чтение допускается стандартом на уровнях Read Uncommitted и Read Committed.

# Аномалия фантомное чтение

Отличие от предыдущей аномалии в том, что при повторном чтении одна и та же выборка дает разные множества строк.

| Транзакция 1 | Транзакция 2 |
|---|---|
| | SELECT SUM(x) FROM tbl; |
| INSERT INTO tbl (x, y) VALUES (5, 3); | |
| | SELECT SUM(x) FROM tbl; |

Фантомное чтение допускается стандартом на уровнях Read Uncommitted, Read Committed, Repeatable Read.

# Отсутствие аномалий и Serializable

Уровень Serializable должен предотвращать вообще все аномалии.

# Уровни изоляции и аномалии по стандарту

| | Потерянные изменения | Грязное чтение | Неповторяющееся чтение | Фантомное чтение | Другие аномалии |
|---|---|---|---|---|---|
| Read Uncommitted | — | да | да | да | да |
| Read Committed | — | — | да | да | да |
| Repeatable Read | — | — | — | да | да |
| Serializable | — | — | — | — | — |

# Эволюция блокировок транзакций

2PL – двухфазная блокировка

Snapshot Isolation - протокол изоляции на основе снимков

MVCC (multiversion concurrency control) - управление параллельным доступом посредством многоверсионности. Является расширением над Snapshot Isolation.

# Уровни изоляции и аномалии в PostgreSQL

| | Потерянные изменения | Грязное чтение | Неповторяющееся чтение | Фантомное чтение | Другие аномалии |
|---|---|---|---|---|---|
| Read Uncommitted | — | — | да | да | да |
| Read Committed | — | — | да | да | да |
| Repeatable Read | — | — | — | — | да |
| Serializable | — | — | — | — | — |

# Уровни изоляции на практике

```
=> CREATE TABLE accounts(
  id integer PRIMARY KEY GENERATED BY DEFAULT AS IDENTITY,
  number text UNIQUE,
  client text,
  amount numeric
);
=> INSERT INTO accounts VALUES
  (1, '1001', ‘alex', 1000.00),
  (2, '2001', ‘petr', 100.00),
  (3, '2002', ‘petr', 900.00);
```

# Уровень изоляции по умолчанию

```
=> BEGIN;
=> SHOW transaction_isolation;
 transaction_isolation
-----------------------
 read committed
(1 row)
```

# Уровень изоляции по умолчанию

```
=> SHOW default_transaction_isolation;
 default_transaction_isolation
-------------------------------
 read committed
(1 row)
```

# Транзакция T1

```
=> UPDATE accounts SET amount = amount - 200 WHERE id = 1;
=> SELECT * FROM accounts WHERE client = ‘alex';
 id | number | client | amount
----+--------+--------+--------
  1 | 1001   | alex   | 800.00
(1 row)
```

# Транзакция T2

```
|  => BEGIN;
|  => SELECT * FROM accounts WHERE client = ‘alex';
|   id | number | client | amount
|  ----+--------+--------+---------
|    1 | 1001   | alex   | 1000.00
|  (1 row)
```

# Неповторяющееся чтение

=> COMMIT;

| => SELECT * FROM accounts WHERE client = ‘alex’;

| id | number | client | amount
| ----+--------+--------+--------
| 1 | 1001 | alice | 800.00
| (1 row)

| => COMMIT;

# Принятие решения

```
IF (SELECT amount FROM accounts WHERE id = 1) >= 1000 THEN
    UPDATE accounts SET amount = amount - 1000 WHERE id = 1;
END IF;
```

# Принятие решения

```
IF (SELECT amount FROM accounts WHERE id = 1) >= 1000 THEN
      -----
|     UPDATE accounts SET amount = amount - 200 WHERE id = 1;
|  COMMIT;
      -----
      UPDATE accounts SET amount = amount - 1000 WHERE id = 1;
END IF;
```

# Корректный код

Как написать код корректно

Не писать код, который приведет двоякому смыслу. (можно использовать ограничение целостности)

ALTER TABLE accounts ADD CHECK amount >= 0;

Использовать один SQL-оператор (общие табличные выражения (CTE) или INSERT ON CONFLICT)

Пользовательские блокировки (SELECT FOR UPDATE, LOCK TABLE)

# Вопросы

# Список литературы

1. Gerhard Weikum,Gottfried Vossen. Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery / Morgan Kaufmann Publishers Inc., 340 Pine Street, Sixth Floor, San Francisco, CA, United States, May 2001, 852 p. – ISBN 978-0-08-051956-2. — Текст : электронный // Доступна для чтения и скачивания в pdf формате на англ. языке с ресурса http://nozdr.ru/ по запросу в Яндекс: Transactional Information Systems

# Спасибо за внимание!