

# **Введение в реляционные базы данных**

Лекция 9: Транзакции

Артем Толканев

December 06, 2024

# Транзакции

Есть последовательности инструкций запроса и/или изменения данных.

Набор этих операций, которые можно считать в конкретном случае логически одной, то есть полностью произойдут (**COMMIT**) или полностью отменятся (**ROLLBACK**), называют **транзакцией**.



## **Atomicity (Атомарность)**

Либо фиксирует действия транзакции после завершения всех ее этапов, либо откатывает все ее действия в случае, если транзакция не смогла успешно выполнить все свои действия.

## **Consistency (Согласованность)**

После выполнения транзакции база данных остается “адекватной”.

## **Isolation (Изоляционность)**

Во время работы транзакции другие транзакции не должны оказывать влияние на результаты её работы.

## **Durability (Устойчивость)**

Внезависимости от того, что где-либо что-то происходит - если мы получили Ок после выполнения транзакции, то это действительно Ок - все изменения произведены даже если будет сбой.



Соответственно возможны аномалии:

**Read-Write (R-W)** (Unrepeatable Read)

Write-Read (W-R)

Write-Write (W-W)

<b>T1</b> <b>BEGIN</b> $R(A) = 2000$  $R(A) = 3000$ <b>COMMIT</b>	<b>T2</b> <b>BEGIN</b>  $R(A) = 2000$ $W(A) = 3000$ <b>COMMIT</b>
--	--



Соответственно возможны аномалии:

Read-Write (R-W)

**Write-Read (W-R)** Dirty Read

Write-Write (W-W)

<b>T1</b> <b>BEGIN</b> R (A) = 2000 W (A) = 3000          <b>ROLLBACK</b>	<b>T2</b> <b>BEGIN</b>   R (A) = 3000 W (A) = 3000 - 500  <b>COMMIT</b>
--	--



Соответственно возможны аномалии:

Read-Write (R-W)

Write-Read (W-R)

**Write-Write (W-W)** Lost Update

<b>T1</b> <b>BEGIN</b> $R(A) = 2000$  $W(A) = 2000 - 500$ <b>COMMIT</b>	<b>T2</b> <b>BEGIN</b> $R(A) = 2000$ $W(A) = 2000 - 500$ <b>COMMIT</b>
--	--



SERIAL

```
CREATE TABLE test(id serial, value int);
```

identity(1,1)



```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name ( [ column_name data_type [ STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN | DEFAULT } ] [ COMPRESSION compression ] [ opclass ] ] )
```

## Is a Postgresql UNLOGGED table completely lost on process crash?

Asked 1 year, 8 months ago Modified 1 year, 8 months ago Viewed 629 times



1



I use UNLOGGED tables for a few very large tables in a data warehouse style application.

Until recently, I understood UNLOGGED to mean “won’t write to the WAL” - which in turn means that *recent* changes maybe lost on a process crash / unclean termination, and that there will be no replication

Maybe I’m misunderstanding the language in the documentation (or maybe I’m not) but when I

The Over

✎ Eve  
deb

✎ Fou  
LLM





# CONSTRAINTS:

IMMEDIATE

DEFERRABLE

МГНОВЕННЫЕ



UNIQUE,  
PRIMARY KEY,  
REFERENCES (foreign key), EXCLUDE

отсроченные



# CONSTRAINTS:

IMMEDIATE

DEFERRABLE

МГНОВЕННЫЕ



UNIQUE,  
PRIMARY KEY,  
REFERENCES (foreign key), EXCLUDE

отсроченные

```
ALTER TABLE ... ADD CONSTRAINT ... UNIQUE (...) DEFERRABLE
```



# LOCK

<b>T1</b> <b>BEGIN</b> $W(A) = 2000 - 100$ $W(B) = 2000 + 100$ <b>COMMIT</b>	<b>T2</b> <b>BEGIN</b> $W(B) = 2000 + 50$ $W(A) = 2000 - 50$ <b>COMMIT</b>
--	--



## LOCK

Транзакции запрашивают блокировки на данные.

Диспетчер блокировок удовлетворяет или блокирует запросы.

Транзакции снимают блокировки.

Диспетчер блокировок обновляет свою внутреннюю таблицу блокировок.

<b>T1</b> <b>BEGIN</b> W(A) = 2000 - 100 W(B) = 2000 + 100 <b>COMMIT</b>	<b>T2</b> <b>BEGIN</b> W(B) = 2000 + 50 W(A) = 2000 - 50 <b>COMMIT</b>
--	--

