

# Лекция 6. Часть II

# 1. REDO-журнал

# Журнал повторений

- Другой способ ведения журнала — журнал повторений, REDO LOG (повторяются завершенные транзакции).

Возможные записи в журнале повторений:

- BEGIN Tx      -- начало транзакции;
- Tx: R, new\_val -- изменение: транзакция Tx  
                  -- изменяет значение  
                  -- на new\_val в R;
- COMMIT Tx    -- успешное завершение транзакции;
- ABORT Tx     -- преждевременное завершение транз-ии;
- START CHKn:  $T_n, \dots, T_m$  -- начало создания контр. точки
- END CHKn     -- завершение создания контр. точки

# Журнал повторений

**Правило записи** в REDO LOG (правило Write Ahead Logging):

- Запись изменения данных (Tx: R, new\_val) записывается в журнал (на диск) **ДО** сохранения обновленного значения на диске.
- COMMIT Tx должен быть добавлен в журнал (на диск) **ДО** обновления любого из файлов данных, связанных с изменениями данной транзакции (до синхронизации буферного пространства данных с файлами данных).
- То есть в WAL (на диске) вносятся записи изменения, которые произошли в Tx, и запись COMMIT Tx, и только после этого — идет синхронизация буферов с диском.

# Есть ли на слайде ошибка?

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}, 374$
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}, 375$
		COMMIT T1
WriteLog		
$D \leftarrow M(R_{374}), D \leftarrow M(R_{375})$	$R_{374}=1331, R_{375}=1330$	

# Пример

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>
		<b>COMMIT T1</b>
WriteLog		
$D \leftarrow M(R_{374}), D \leftarrow M(R_{375})$	$R_{374}=1331, R_{375}=1330$	

Лог уже на диске, а буферы не синхронизированы с файлами данных

# Использование журнала повторений для восстановления данных (1)

Действия при восстановлении данных (1):

- СУБД сканирует журнал от **старых записей до новых**. Отдельно фиксируются транзакции:
  - Группа 1: для которых есть запись COMMIT Tx;
  - Группа 2: транзакции для которых есть запись ABORT Tx, незавершенные транзакции.

# Использование журнала повторений для восстановления данных (2)

Действия при восстановлении данных (2):

- Когда в журнале встречается запись изменения (Tx: R, old\_val), анализируется Tx. Если Tx:
  - из группы 1: для R перезаписывается new\_val (происходит «повтор», перезапись изменений).
  - из группы 2:
    - СУБД записывает ABORT Tx для каждой незавершенной транзакции из группы 2 (у которой не было ABORT).
    - Запускается операция WriteLog, чтобы записи вида ABORT Tx оказались в журнале.



# Пример 1 (REDO LOG)


Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>



Произошел сбой!

# Действия для восстановления?

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>



Произошел сбой!

# Восстановление

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>

Нет COMMIT: транзакция  
не завершена. ABORT.

# Пример 2 (REDO LOG)

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>
		<b>COMMIT T1</b>
WriteLog		

Произошел сбой!

# Действия для восстановления?

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>
		<b>COMMIT T1</b>
WriteLog		

Произошел сбой!

# Восстановление


Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>
		<b>COMMIT T1</b>
WriteLog		



Транзакция COMMIT и  
зафиксирована → повторяем сначала

# Пример 1 (REDO LOG)

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>
		<b>COMMIT T1</b>



Произошел сбой!

# Действия для восстановления?

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>
		COMMIT T1

Произошел сбой!



# Восстановление

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>
		<b>COMMIT T1</b>

Зависит от того попал ли в WAL на диске COMMIT. WriteLog мог выполнить другой процесс

# Контрольные точки (REDO LOG)

Создание контрольной точки не одномоментно:

- 1) В журнал добавляется START CHKn:  $T_n, \dots, T_m$ , фиксируется на диске через WriteLog.
- 2) Сохранение всех изменений («грязных страниц») на диск, измененных завершенными транзакциями на момент START CHKn.
- 3) В журнал добавляется END CHKn, фиксируется на диске через WriteLog.

# REDO LOG и контрольные точки

Действия при восстановлении данных (3):

- СУБД сканирует журнал в поисках данных о последней контрольной точке и встречается запись END CHKn (а не START CHKn:  $T_n, \dots, T_m$ ):
  - Остальные действия как раньше, но только для транзакций ( $T_n, \dots, T_m$ ) или тех, которые были начаты после записи START CHKn:  $T_n, \dots, T_m$ .

# REDO LOG и контрольные точки

Действия при восстановлении данных (4):

- СУБД сканирует журнал в поисках данных о последней контрольной точке и встречается запись START CHKn:  $T_n$ , ...,  $T_m$  (а не END CHKn):
  - Значит установка контрольной точки  $n$  не завершена (проблема во время установки);
  - Ищем END CHKn-1, дальнейшие действия, как на пред. слайде, только для CHKn-1.

# Недостатки REDO LOG

- Нельзя «сбросить» данные из буферов на диск до завершения транзакции → требуется много памяти под буферы.

## 2. UNDO/REDO-журнал

# Журнал отмены/повторений

- Еще один способ — журнал отмены/повторений, UNDO/REDO LOG.

Возможные записи в журнале повторений:

- BEGIN Tx -- начало транзакции;
- Tx: R, **old\_val**, **new\_val** -- изменение: транзакция Tx  
-- изменяет значение  
-- на new\_val в R;
- COMMIT Tx -- успешное завершение транзакции;
- ABORT Tx -- преждевременное завершение транз-ии;
- START CHKn:  $T_n, \dots, T_m$  -- начало создания контр. точки
- END CHKn -- завершение создания контр. точки

# Журнал отмены/повторений

## Правило записи в UNDO/REDO LOG:

- Запись изменения данных (Tx: R, old\_val, new\_val) записывается в журнал (на диск) **ДО** сохранения обновленного значения на диске.
- COMMIT Tx может быть добавлен в журнал (на диск) как до, так и после обновления любого из файлов данных, связанных с изменениями данной транзакции.



# Пример

Операция	Данные на диске	Журнал (WAL)
		BEGIN T1;
$T(v_R, Tx) \leftarrow M(R_{374})$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1331;$	$R_{374}=374, R_{375}=375$	
$M(R_{374}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{374}$ , <b>1331</b>
$T(v_R, Tx) \leftarrow M(R_{375});$	$R_{374}=374, R_{375}=375$	
$V_R(\text{Room\_Num}) = 1330;$	$R_{374}=374, R_{375}=375$	
$M(R_{375}) \leftarrow T(v_R, Tx);$	$R_{374}=374, R_{375}=375$	T1: $R_{375}$ , <b>1330</b>
WriteLog		
$D \leftarrow M(R_{374}), D \leftarrow M(R_{375})$	$R_{374}=1331, R_{375}=1330$	
		COMMIT T1
WriteLog		

# Использование UNDO/REDO для восстановления данных

Действия при восстановлении данных:

- повторить операции для завершенных транзакций (от ранних к поздним);
- отменить операции для незавершенных транзакций (от поздних к ранним);
- запись COMMIT должна быть сразу синхронизирована с журналом;

# Контрольные точки (UNDO/REDO LOG)

Создание контрольной точки не одномоментно:

- 1) В журнал добавляется START CHKn:  $T_n, \dots, T_m$ , фиксируется на диске через WriteLog.
- 2) Сохранение всех изменений («грязных страниц») на диск.
  - Если транзакция может прервать работу (ROLLBACK), ее изменения не должны фиксироваться в буферах данных.
- 3) В журнал добавляется END CHKn, фиксируется на диске через WriteLog.

# UNDO/REDO LOG и контрольные точки

Действия при восстановлении данных (2):

- Состоит из шагов для UNDO и REDO.
- Благодаря тому, что при установке контрольной точки происходит сохранение всех изменений («грязных страниц») на диск, REDO нужно возвращаться только к последней START CHKn, для которой есть END CHKn:
  - если на момент START CHKn были незавершенные транзакции, их изменения зафиксированы при создании контрольной точки:
    - × Если транзакция в итоге завершиться успешно — REDO (с момента начала этой контр. точки);
    - × Если неуспешно — UNDO — как обычно.

# steal/no-steal, force/no-force

- **no-steal** — подход, при котором страница из буфера данных не может быть записана на диск до завершения (COMMIT) транзакции, в ходе которой она была изменена.
- **steal** — если позволено записывать в файлы данных данные еще незавершенных транзакций (для оптимизации запросов к диску или если буферная память заканчивается).
- **force** — если требуется, чтобы все страницы, измененные транзакцией, записывались сразу перед COMMIT.
- **no-force** — изменения, осуществленные завершенными транзакциями могут быть не зафиксированы на диске.

ARIES (Algorithms for Recovery and Isolation Exploiting Semantics) — алгоритм для восстановления данных.

Базируется на:

- WAL;
- steal;
- no-force;

ARIES происходит в 3 этапа (3 фазы):

- **Анализ** — поиск «грязных» страниц и незавершенных транзакций;
- **REDO** — повторение всех действий;
- **UNDO** — откат действий незавершенных транзакций;

Для работы требуется LSN, таблица транзакций, таблица «грязных страниц».

## Документация PostgreSQL:

<https://www.postgresql.org/docs/14/index.html>

## Лицензия PostgreSQL:

PostgreSQL is released under the PostgreSQL License, a liberal Open Source license, similar to the BSD or MIT licenses.

PostgreSQL Database Management System  
(formerly known as Postgres, then as Postgres95)

Portions Copyright © 1996-2022, The PostgreSQL Global Development Group

Portions Copyright © 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

<https://www.interdb.jp/pg/index.html>