ИТМО ВТ

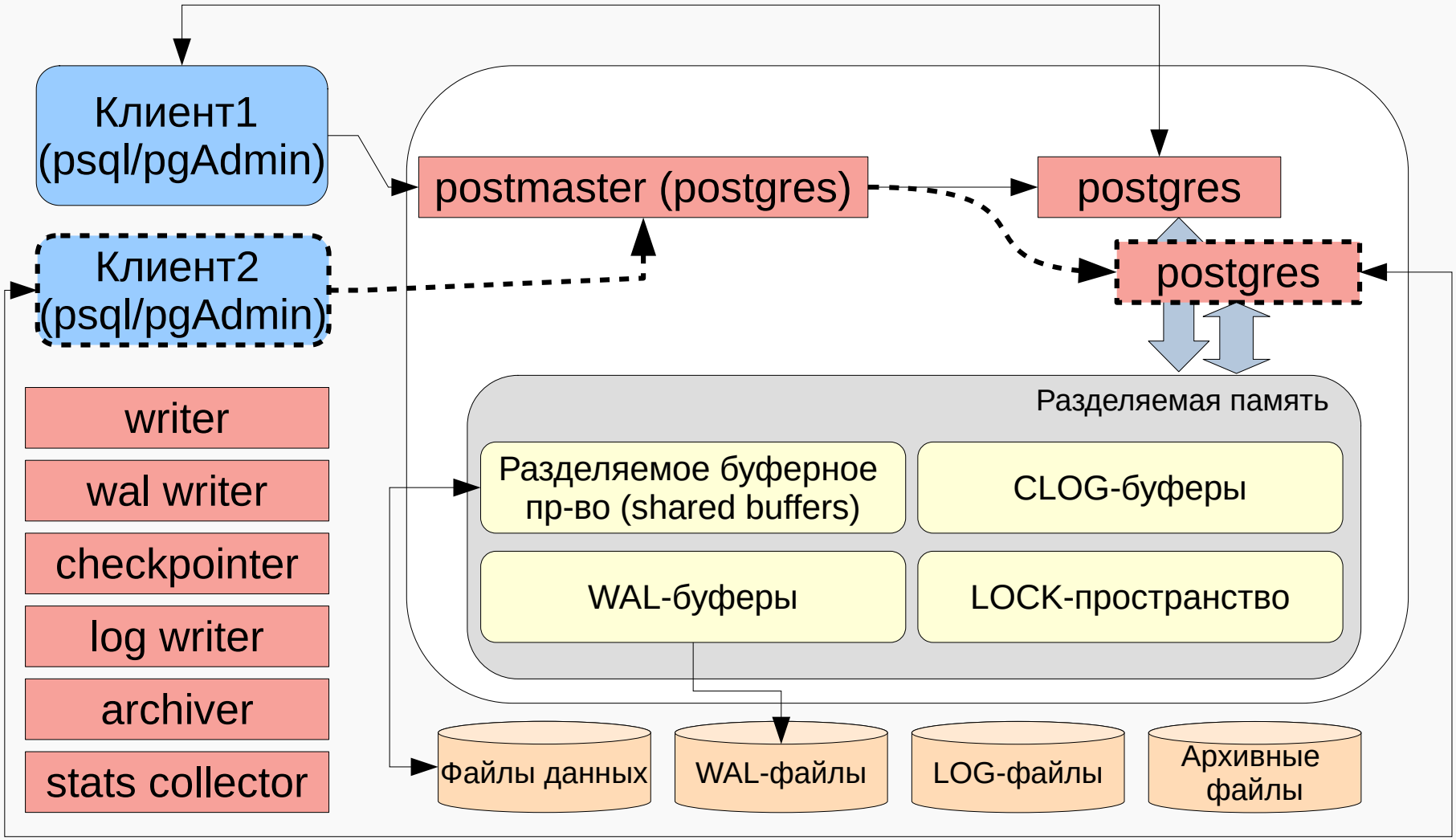
Лекция 8

Лектор:
Николаев В.В.

1. WAL в PostgreSQL

ИТМО ВТ

Архитектура PostgreSQL



ИТМО ВТ

WAL buffers

WAL — Write Ahead Log:

- Хранит информацию об изменениях данных в БД — WAL-записи (XLOG).
- Эта информация используется для воссоздания актуального состояния данных в случае восстановления базы данных (например, после сбоя).
- Настраивается через параметр wal_buffers — размер wal-буфера:
  - по умолчанию 1/32 от разделяемого буфера;

ИТМО ВТ

WAL-записи

- WAL-записи фиксируют различные изменения состояния данных в БД:
  - INSERT, UPDATE, DELETE, COMMIT, ...;
- WAL-записи записываются в:
  - Write Ahead Log;
  - При операции COMMIT — происходит синхронизация с WAL (логом транзакций);
- WAL — общий для всего кластера.

ИТМО ВТ

PostgreSQL: WAL-записи

- Поддерживается REDO-лог.
- Запись об изменении попадает в журнал (на диск) раньше, чем данные на диск и clog.
- PGDATA/pg_wal:
  - журнал разбит на сегменты;
  - wal-segsize — размер сегмента;
  - wal-segsize можно изменить только при инициализации кластера PostgreSQL;

Особенности работы журнала

- В журнал не попадают:
  - действия над временными таблицами;
  - действия над UNLOGGED-таблицами;
  - **до** версии 10: хэш-индексы.

- UNLOGGED TABLE: изменения для данных в такой таблице не хранятся в WAL:
  - CREATE **UNLOGGED** TABLE Students ...

UNLOGGED можно использовать, **только** когда данные в таблице **не важны** (**не страшны** потери), и при этом требуется минимизировать число обменов IO

ИТМО ВТ

PostgreSQL: WAL-записи

Основные составляющие записи в журнале:

- Заголовок:
  - размер всей записи;
  - номер транзакции;
  - размер данных;
  - ссылка на предыдущую запись;
  - контрольная сумма;
- Данные об изменении — могут быть по-разному представлены.

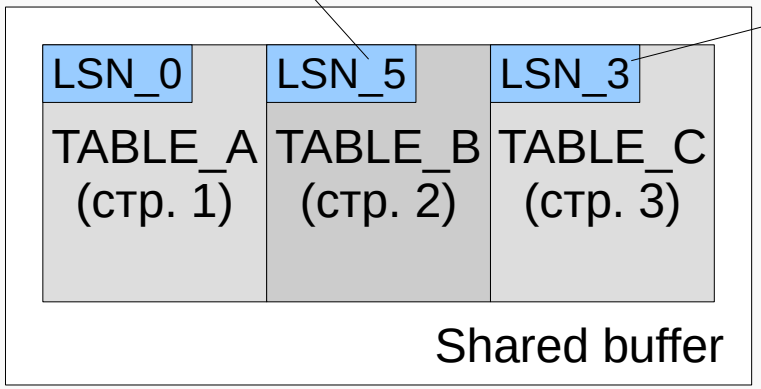
ИТМО ВТ

WAL-записи

- У каждой WAL-записи есть **LSN** (**Log Sequence Number**) — уникальный идентификатор WAL-записи.
- Используется для нахождения позиции записи в WAL:
  - LSN — смещение до записи от начала журнала;
- LSN последней WAL-записи, относящейся к некоторой странице хранится в заголовке страницы.
- Для определения того, актуальна или нет некоторая страница (например, на момент восстановления) её LSN сравнивается с LSN в журнале.

WAL-записи

У страниц есть заголовок.



В заголовке параметр pd_lsn: **последний** LSN с модификацией в этой странице

ИТМО ВТ

Восстановление

- WAL (**REDO-лог**) — журнал последовательно читается и перевыполняются операции:
  - для определения записей, которые нужно заново «выполнить» используется LSN.
- Чтобы сократить число записей для просмотра: используются **контрольные точки**:
  - процесс **checkpointer**;
  - начало последней контрольной точки — REDO point.

ИТМО ВТ

wal_level

Параметр позволяет задать уровень детализации операций в WAL:

- **minimal** — только для восстановления после сбоя или immediate shutdown;
- **replica** — minimal + поддержка архивирования WAL и физической репликации;
- **logical** — replica + поддержка логической репликации (логического декодирования WAL);

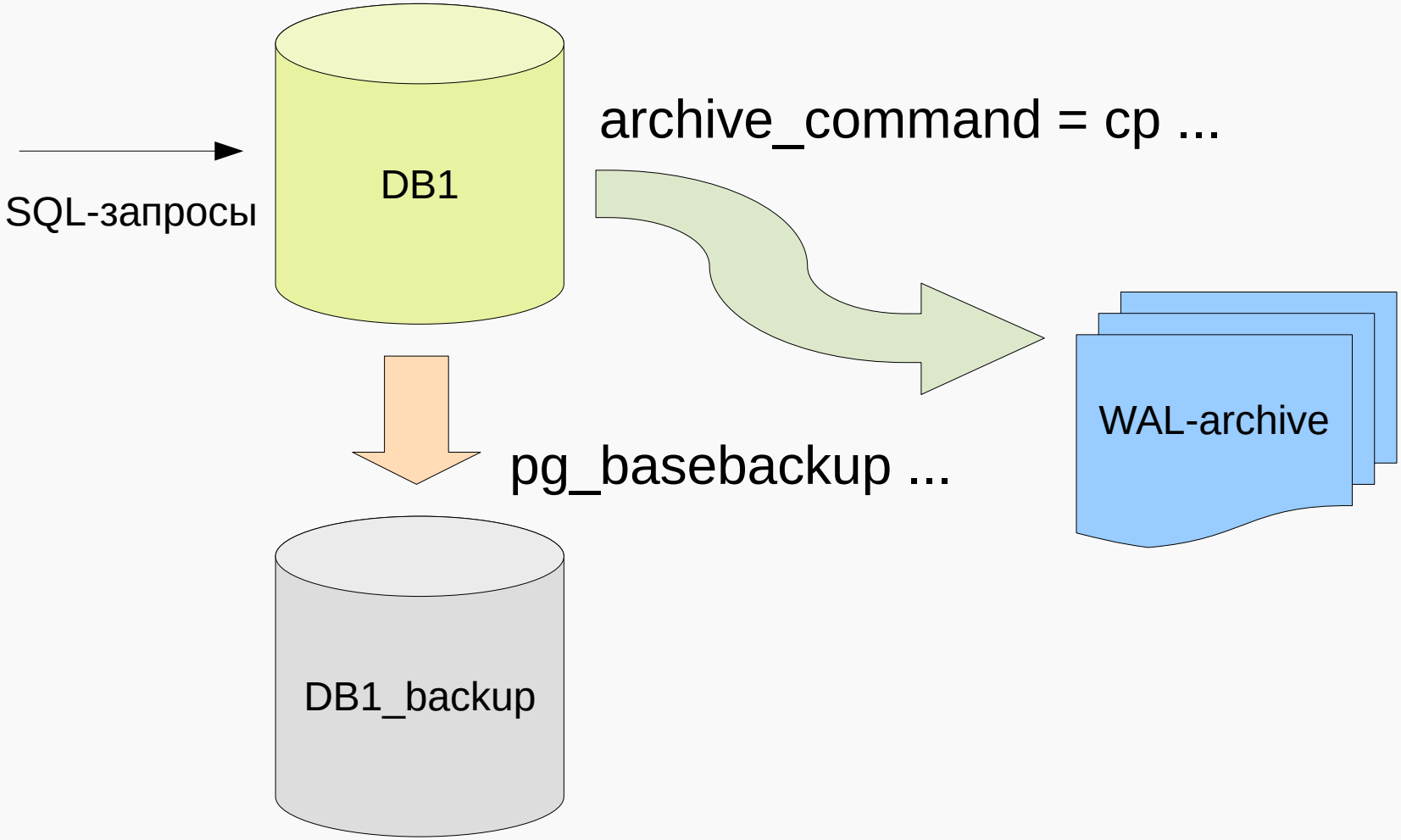
ИТМО ВТ

Непрерывное архивирование

- Базируется на существовании WAL (PGDATA/pg_wal):
  - состояние БД может быть восстановлено путем повтора зафиксированных в логе операций;
  - конфигурационные файлы не восстанавливаются через WAL (postgresql.conf, ...);

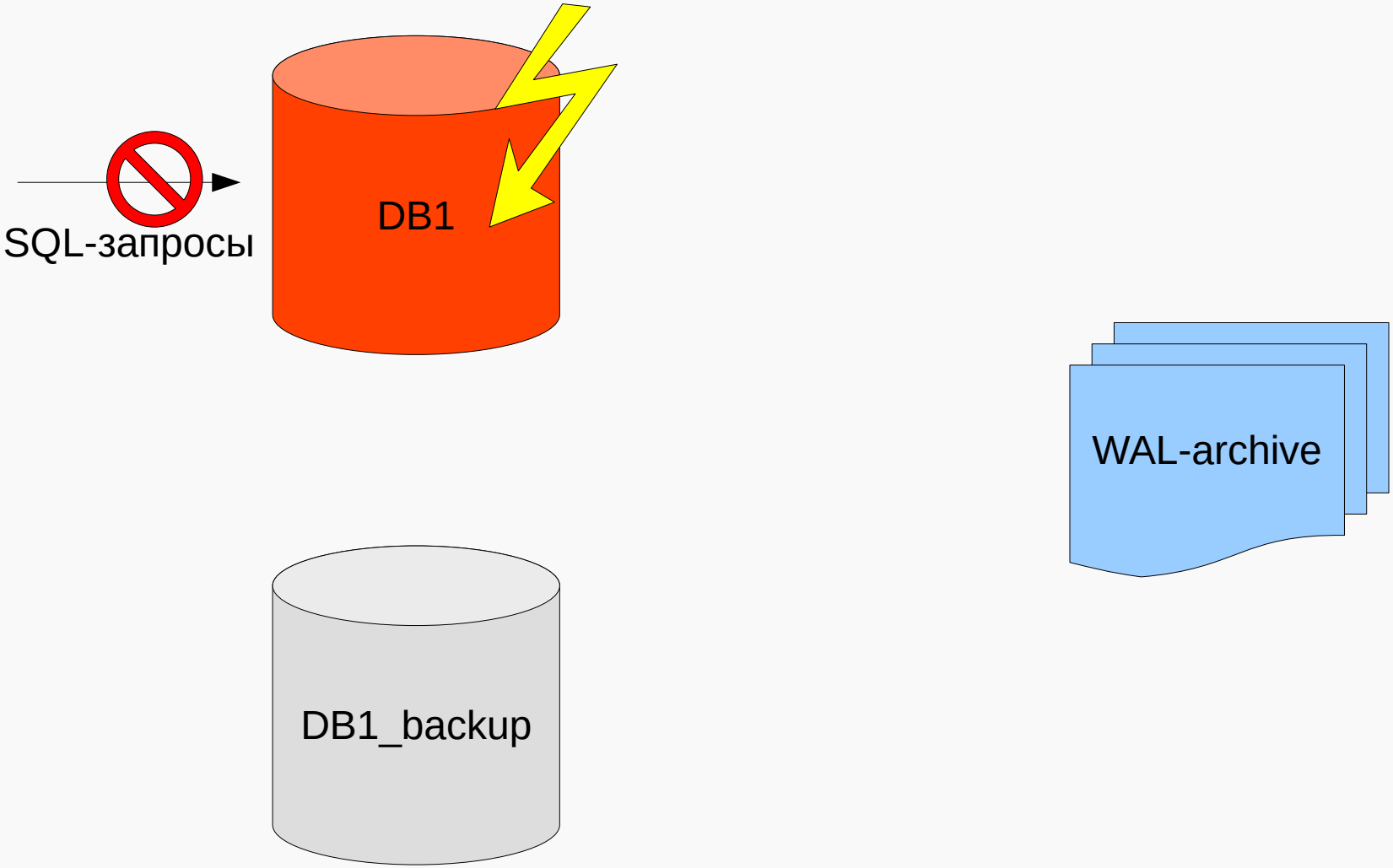
Резервная копия = полная копия + WAL-файлы

1. Нормальный процесс работы при непрерывном архивировании



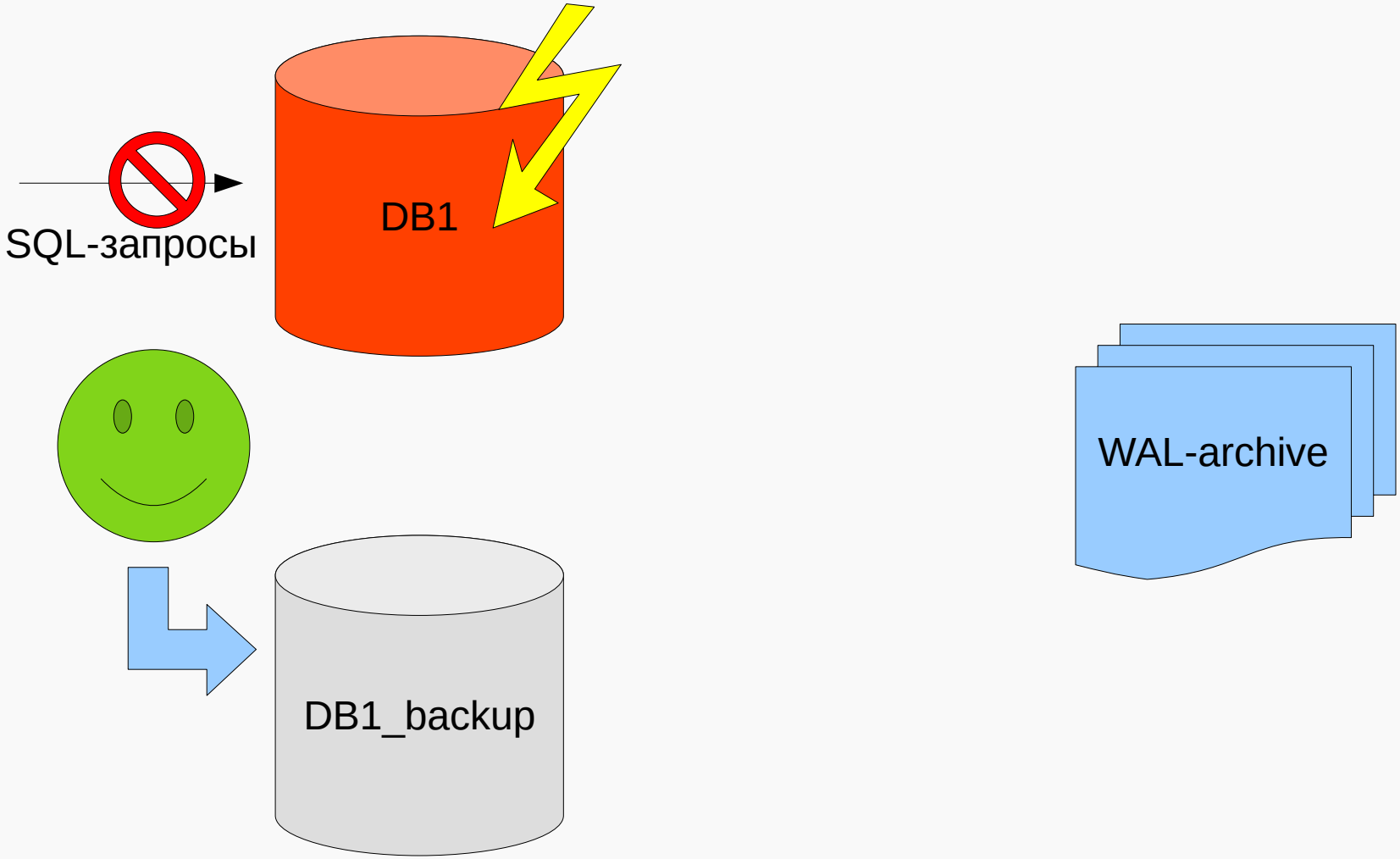
ИТМО ВТ

2. Сбой



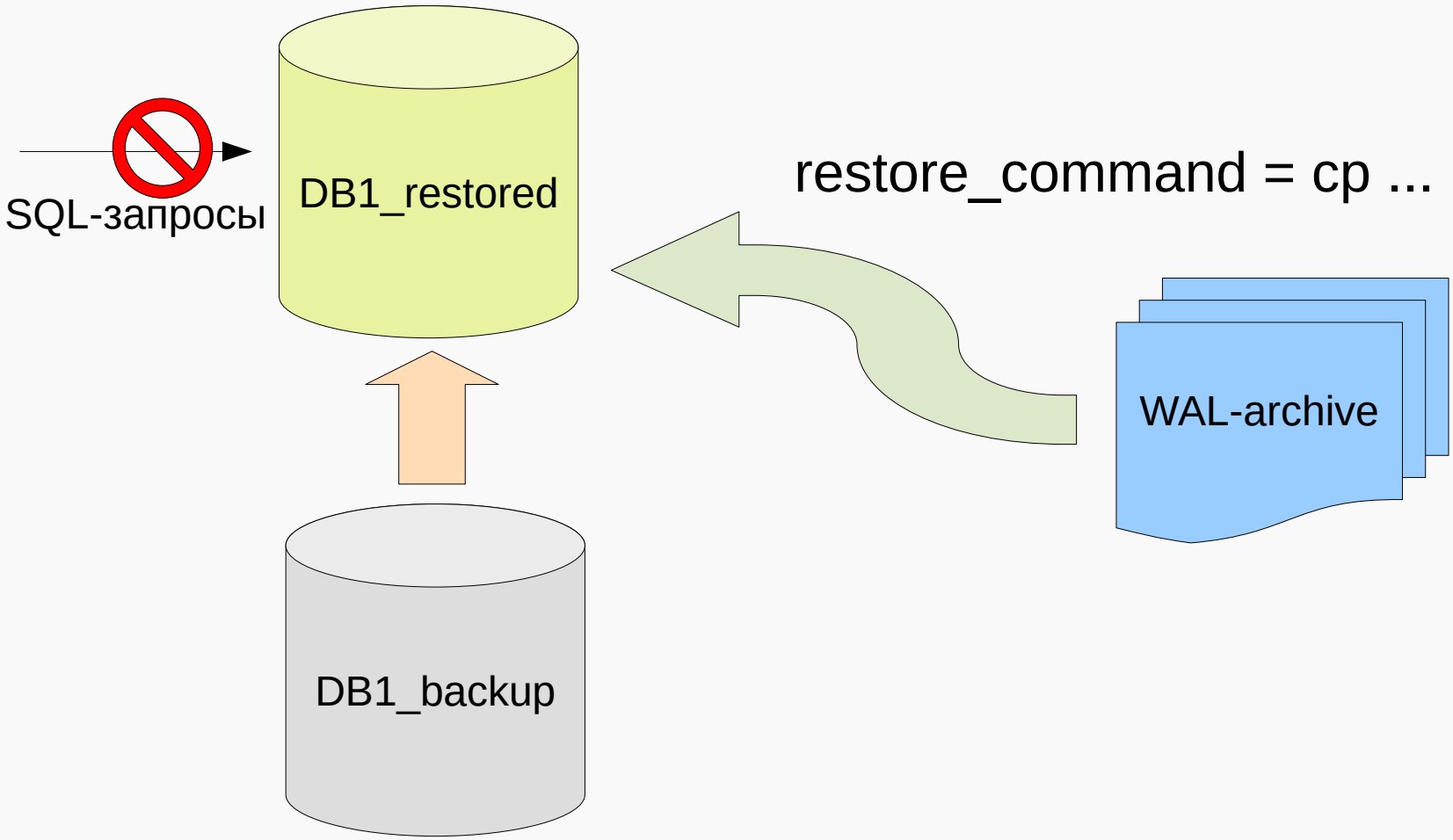
ИТМО ВТ

3. Обнаружение проблемы

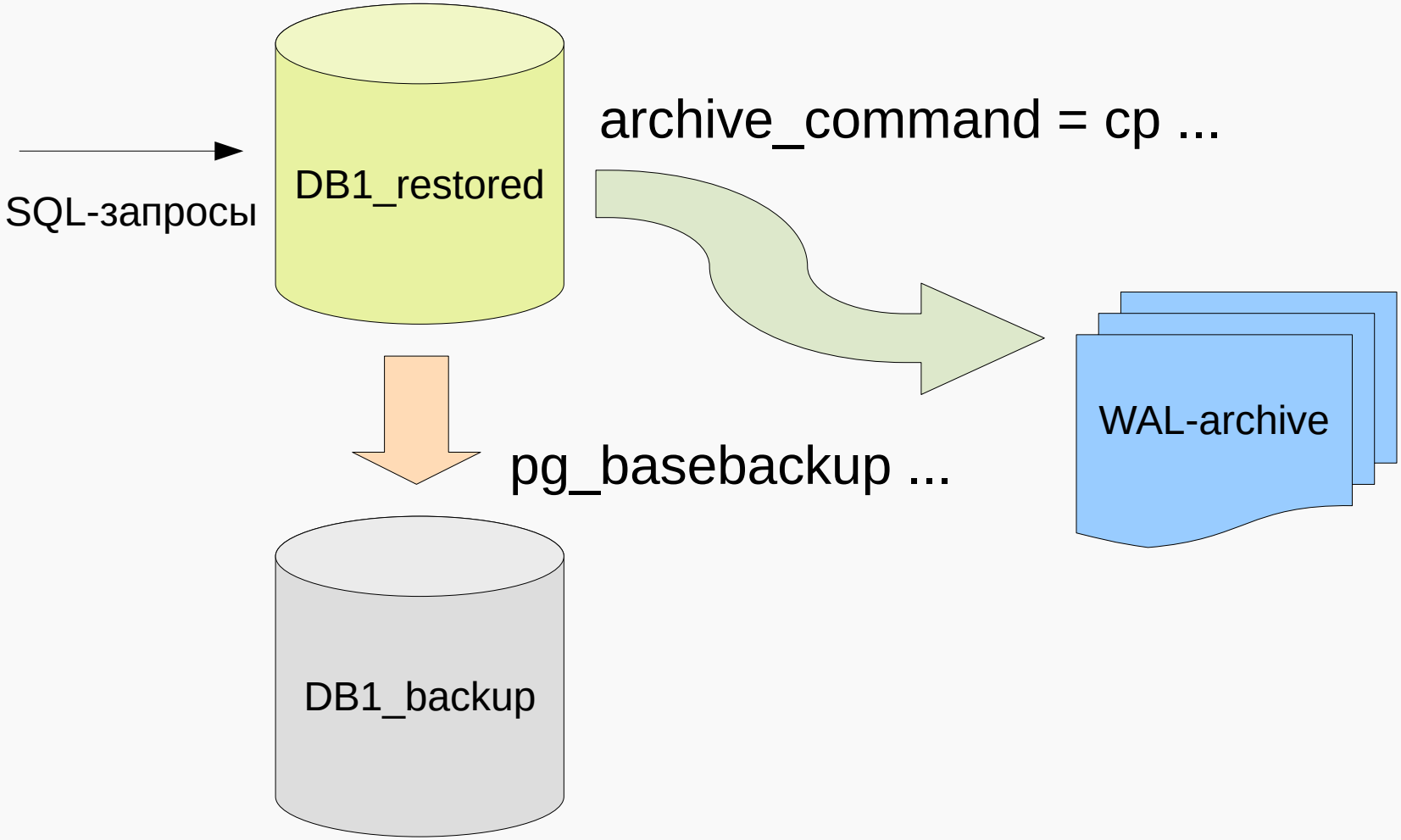


ИТМО ВТ

4. Восстановление

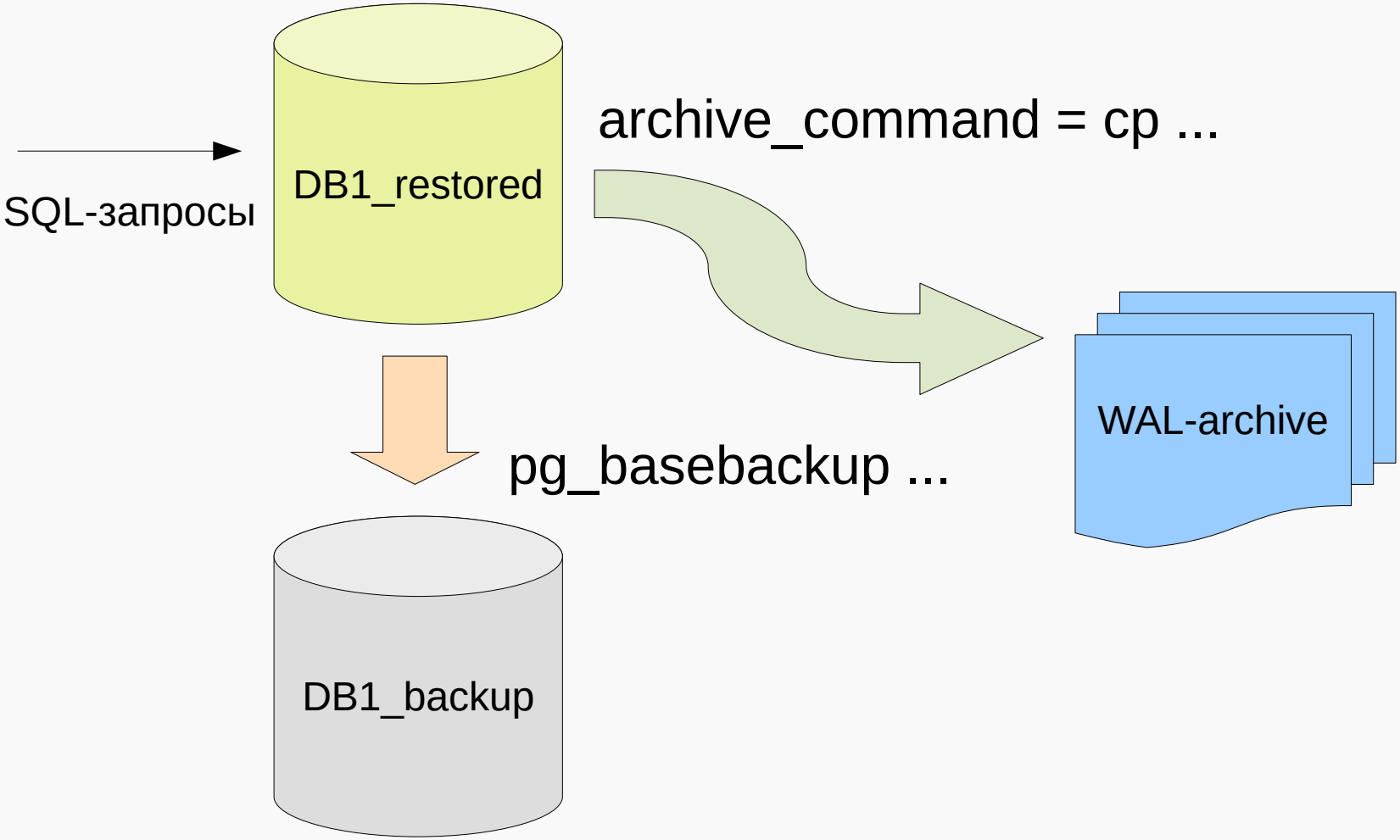


5. Состояние восстановлено



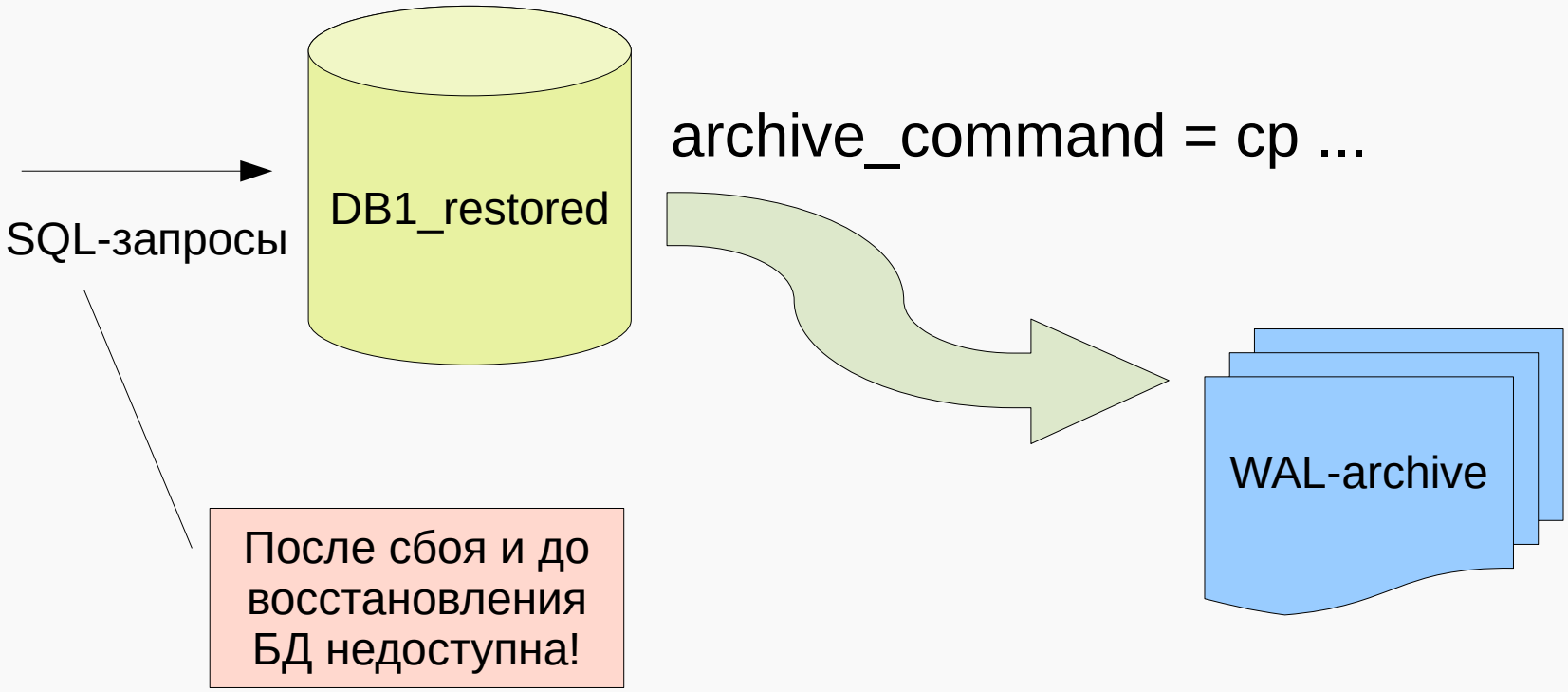
ИТМО ВТ

В чем проблема?



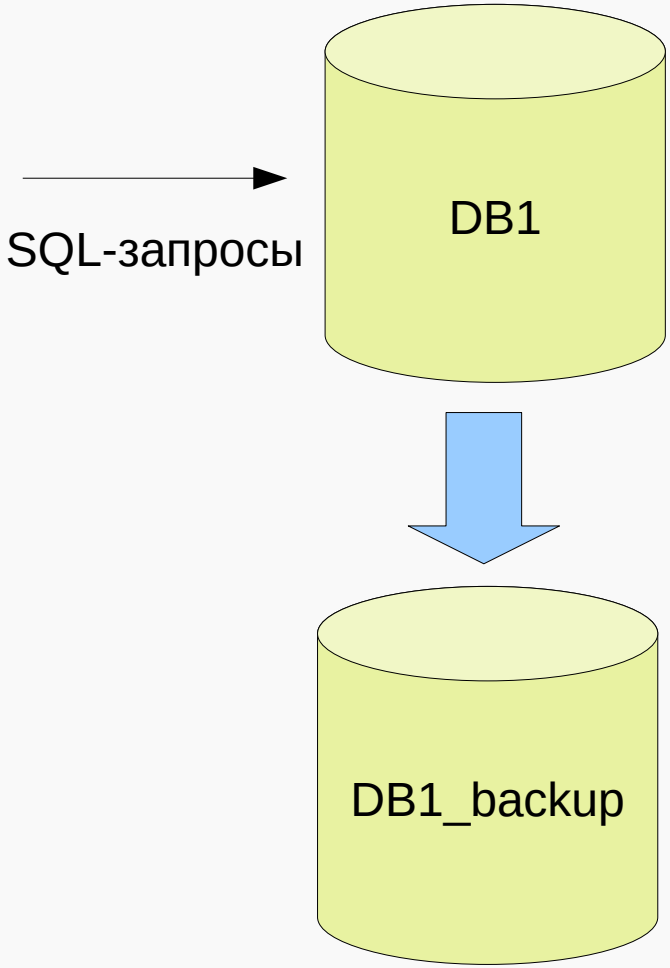
ИТМО ВТ

В чем проблема?



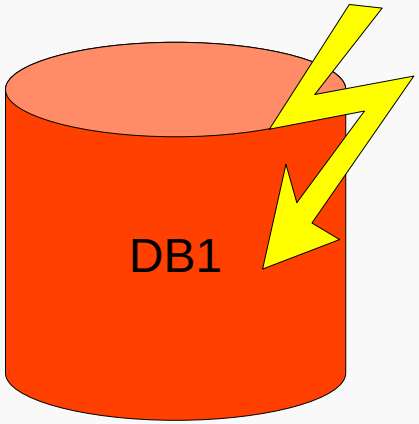
ИТМО ВТ

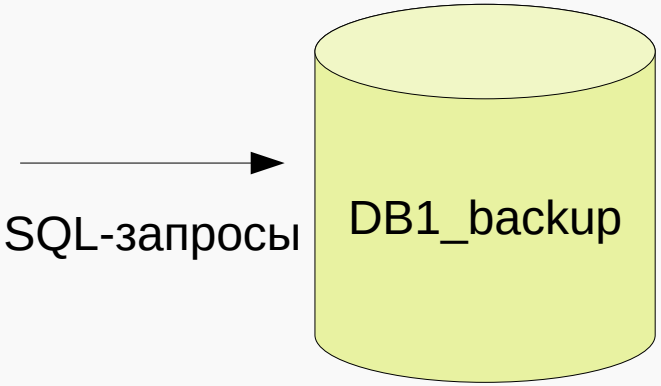
Возможное решение



ИТМО ВТ

Возможное решение:





2. Репликация данных в PostgreSQL

ИТМО ВТ

Что мы хотим?

Высокий уровень доступности — (high availability) способность системы работать без сбоев (непрерывно) в течение определенного периода времени.

Как обеспечить высокий уровень доступности:

- обеспечить избыточность важных частей системы;
  - репликация данных.
- Failover, switchover.

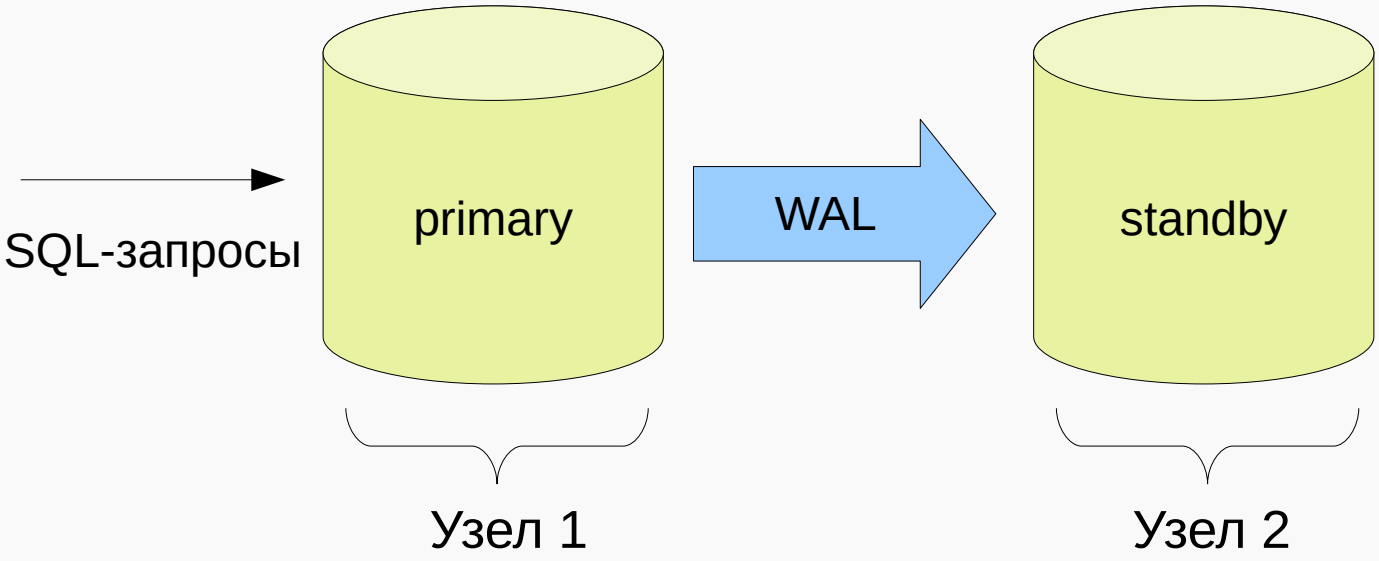
ИТМО ВТ

Репликация

- Репликация данных — средство обеспечения избыточности. Заключается в сохранении и поддержании нескольких копий данных.
- Репликация в СУБД:
  - **Узлы** — отдельные серверы БД, обеспечивающие работу. Совокупность узлов — **кластер**.
  - У каждого узла — своя БД (оригинал или копия, реплика). Копии данных синхронизируются в реальном времени.
  - Один из серверов БД — главный (**master, primary**). Режим — чтение/запись. Если несколько master-узлов — multimaster-репликация.
  - Остальные узлы — зависимые (**slave**, **standby**). Если режим — только чтение — **hot standby**.

ИТМО ВТ

Репликация



ИТМО ВТ

Репликация в PostgreSQL

- В PostgreSQL репликация может быть:
  - **Физическая** — пересылаются файлы (WAL). standby — содержит копию primary:
    - **асинхронная** — данные для синхронизации состояний серверов БД пересылаются без подтверждения получения;
    - **синхронная** — пересылки данных идут с подтверждением.
  - **Логическая** — пересылаются декодированные из WAL операции.

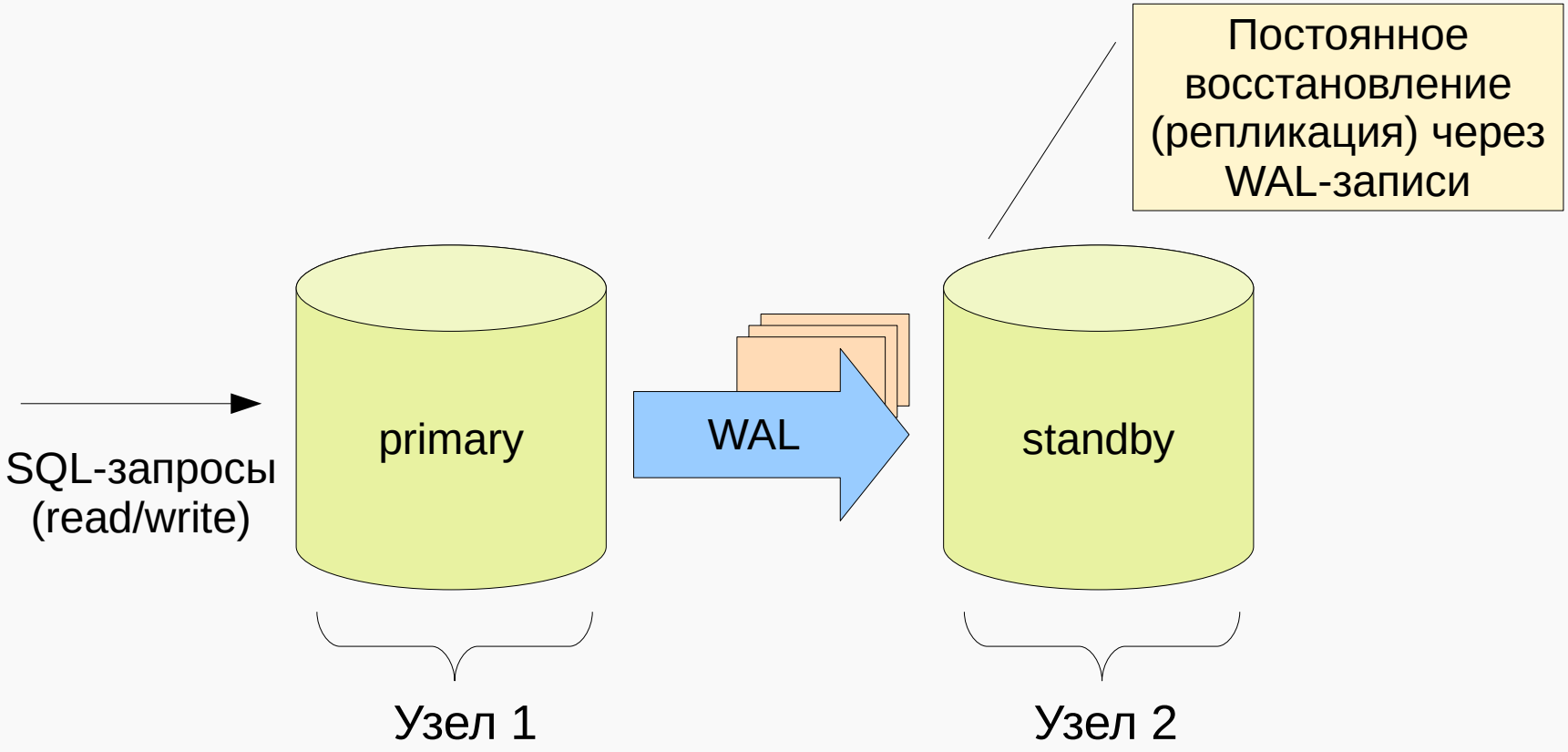
ИТМО ВТ

Физическая репликация

- **Физическая репликация** — пересылаются файлы (WAL). Standby — содержит копию primary.
- Требуется — wal_level = replica (по умолчанию).
- Потоковая репликация (**streaming replication**) — WAL-записи передаются по соединению между primary и standby:
  - Использует свой протокол для отправки WAL-файлов — WAL-sender, WAL-receiver.
- standby — осуществляет восстановление на основе полученных WAL для поддержки актуального состояния.

ИТМО ВТ

Потоковая репликация



ИТМО ВТ

Физическая репликация в PostgreSQL

- Пользователь для репликации:

  CREATE USER PHYSREPL **WITH REPLICATION** …
- pg_hba.conf: host replication physrepl all md5
- backup: pg_basebackup -d 'conn params' -D 'path'

primary

- standby.signal в PGDATA.
- postgresql.conf:

  primary_conninfo = 'host=… user=physrepl'

standby

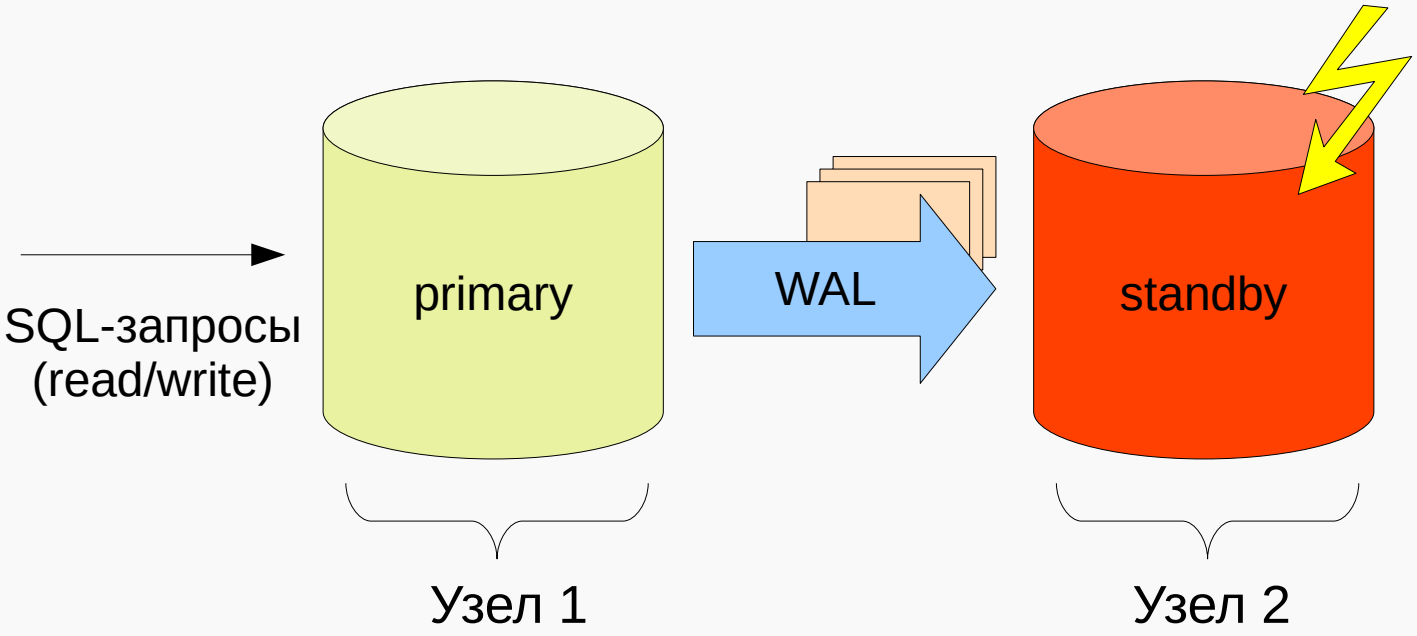
ИТМО ВТ

standby.signal

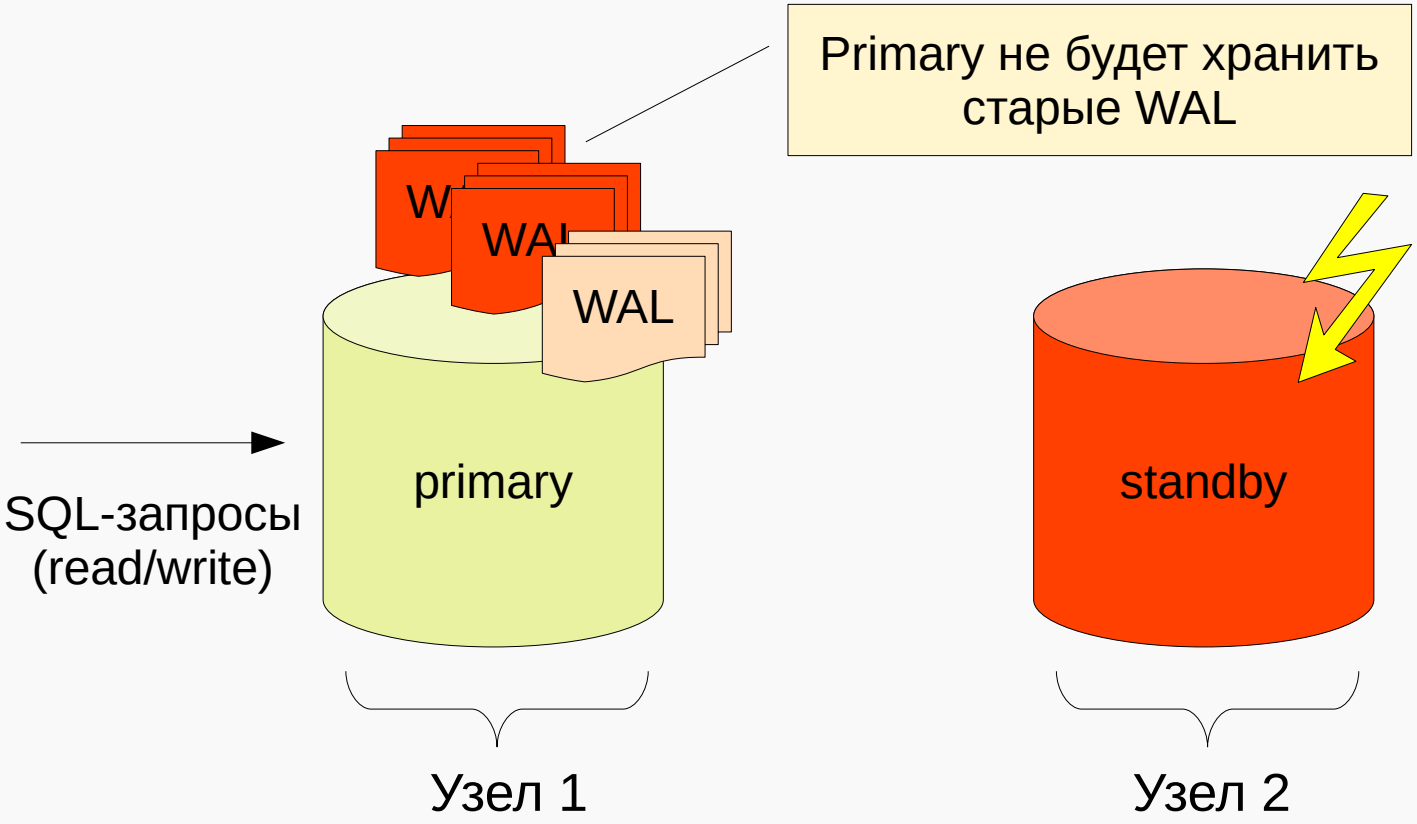
- Файл в PGDATA.
- Позволяет запустить сервер в режиме standby.
- Если установлен:
  - После обработки архивированных в данный момент WAL, сервер продолжает пытаться получить новые WAL-сегменты через:
    - restore_command
    - WAL-записи с мастера, используя primary_conninfo
- primary_conninfo — используется только, если standby.signal установлен.

ИТМО ВТ

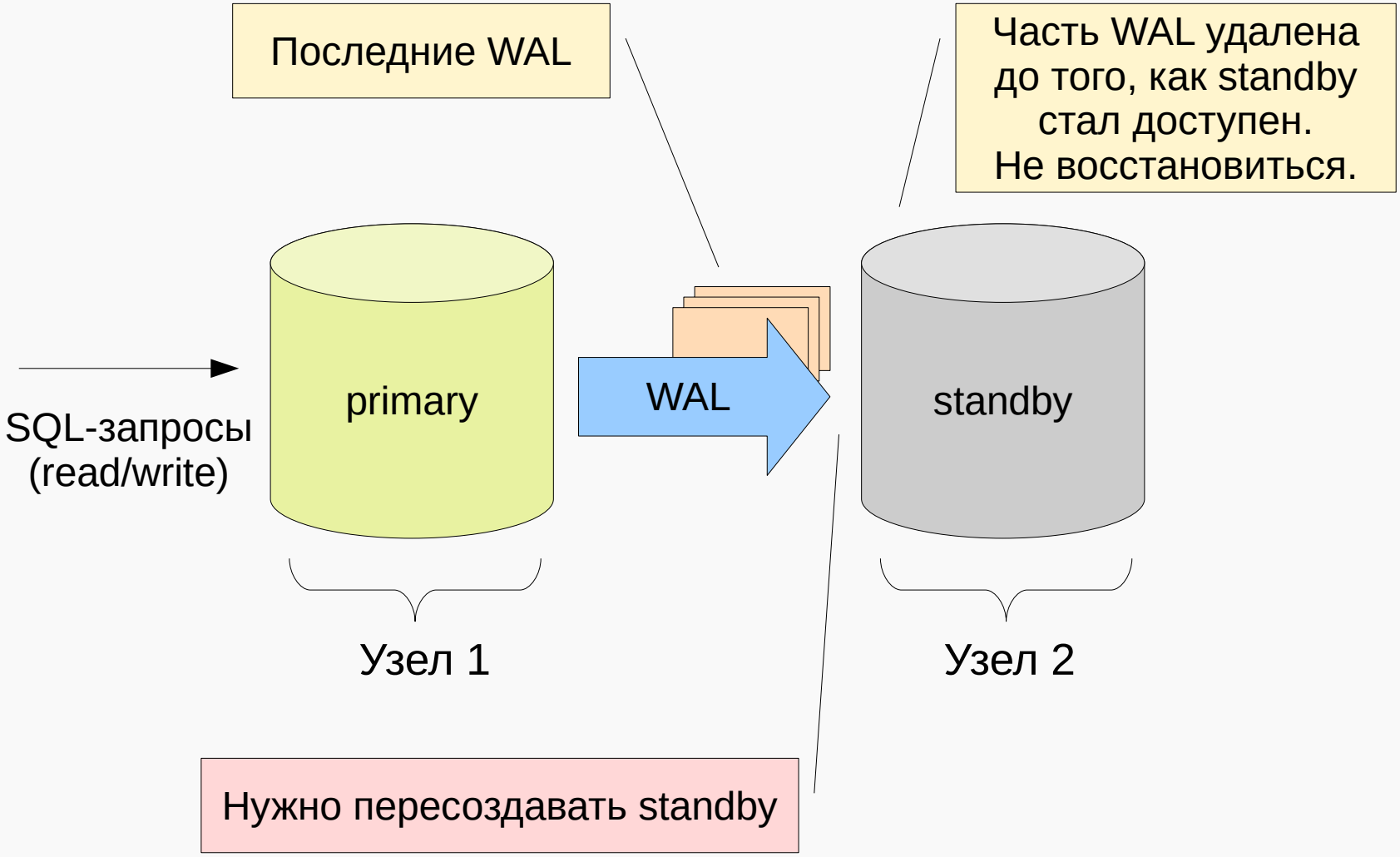
Что если произошла временная проблема со standby?



Что если произошла временная проблема со standby?



Что если произошла временная проблема со standby?



ИТМО ВТ

wal_keep_size

- Задается в postgresql.conf.
- Определяет минимальное количество ненужных (для мастера) WAL-сегментов, которые должны храниться.
- По умолчанию — 0.
- Эти WAL-сегменты будут доступны standby:
  - **НЕТ** гарантии, что для standby будет достаточно этих сегментов.

ИТМО ВТ

Слоты

- Позволяет хранить сегменты, пока они не отправлены standby.
- Таких сегментов может много накопиться (если standby не доступен).
- **max_slot_wal_keep_size** — позволяет ограничить число хранимых сегментов.
- Для использования:
  - нужно создать слот: SELECT * FROM **pg_create_physical_replication_slot**('node2_slot');
  - на standby: **primary_slot_name** = 'node2_slot'.

ИТМО ВТ

Полезные параметры/команды

- --max_wal_senders — число standby + 1. Если используется —wal_method=stream — еще +1 — для backup.
  - Определяет число процессов для отправки WAL (WAL sender).
- --wal_method=stream — при использовании pg_basebackup будет открыто второе соединение для передачи WAL параллельно с созданием копии.
  - Используется по умолчанию.
- Просмотр состояния реплики:
  - представление pg_stat_replication;

ИТМО ВТ

Синхронная репликация

- Определяет, нужно ли ждать обработки WAL на репликах, когда происходит COMMIT транзакции.
- параметр synchronous_commit (default = on) + synchronous_standby_names:
  - по умолчанию включена асинхронная репликация.
- Дает возможность подтвердить, что транзакция реплицирована хотя бы на одном standby.
- Указать список standby серверов с синхронной репликацией:

  synchronous_standby_names = 'node2, node3, node4'

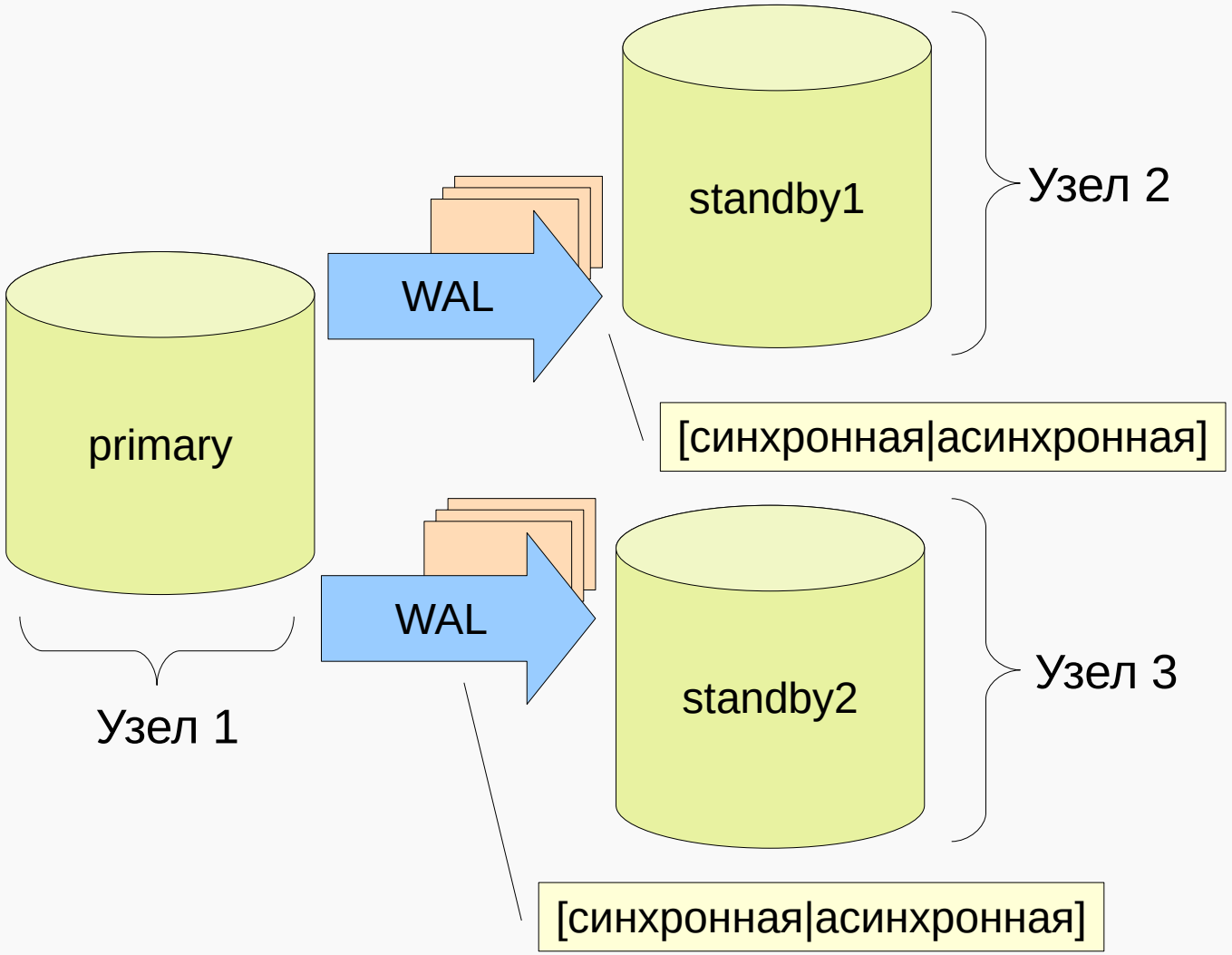
ИТМО ВТ

Synchronous commit

	Долгов-ть локального коммита	Долгов-ть сбой БД	Долгов-ть сбой ОС	standby - согл-ть запросов
remote_apply	+	+	+	+
on	+	+	+	
remote_write	+	+		
local	+			
off				

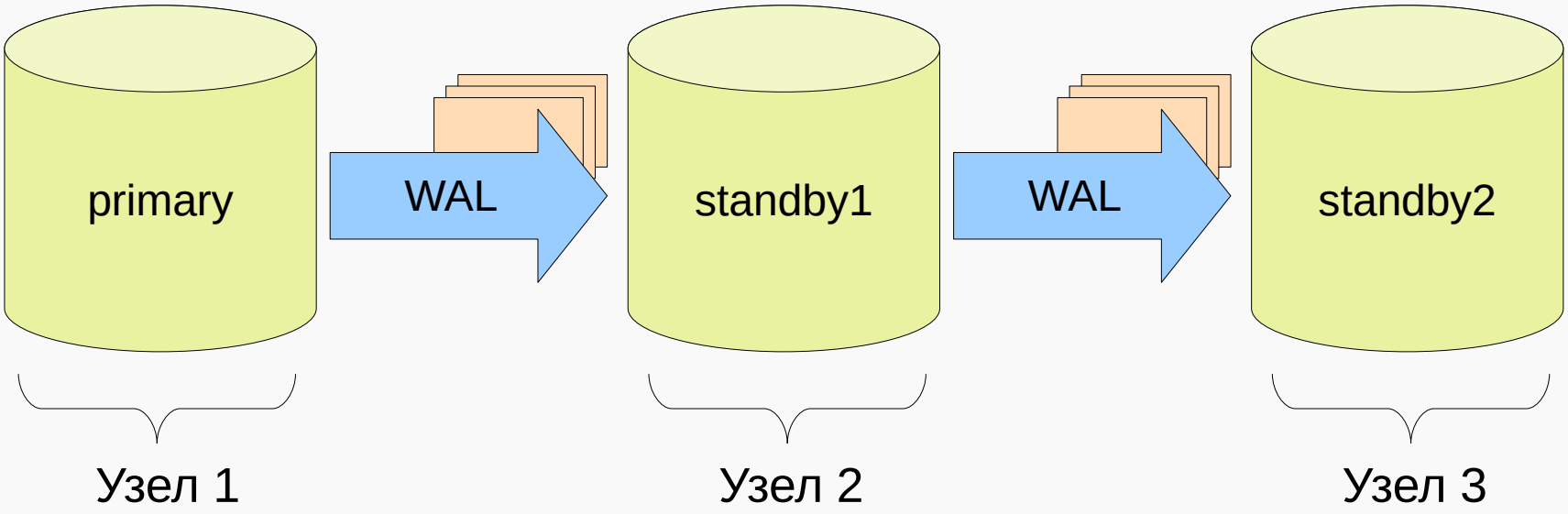
ИТМО ВТ

Репликация на несколько узлов



ИТМО ВТ

Ступенчатая репликация: cascading replication



ИТМО ВТ

Особенности standby

- **warm standby** — режим standby, при котором логи (WAL), архивированные primary-сервером, собираются standby и происходит непрерывное восстанавление состояния БД по полученным WAL. Warm standby не позволяет осуществлять запросы.
- **hot standby** — сервер работает в режиме чтения — кроме восстановления по WAL, позволяет осуществлять запросы к данным (read-only).

Параметр hot_standby = true (по умолчанию).

3. Логическая репликация данных в PostgreSQL

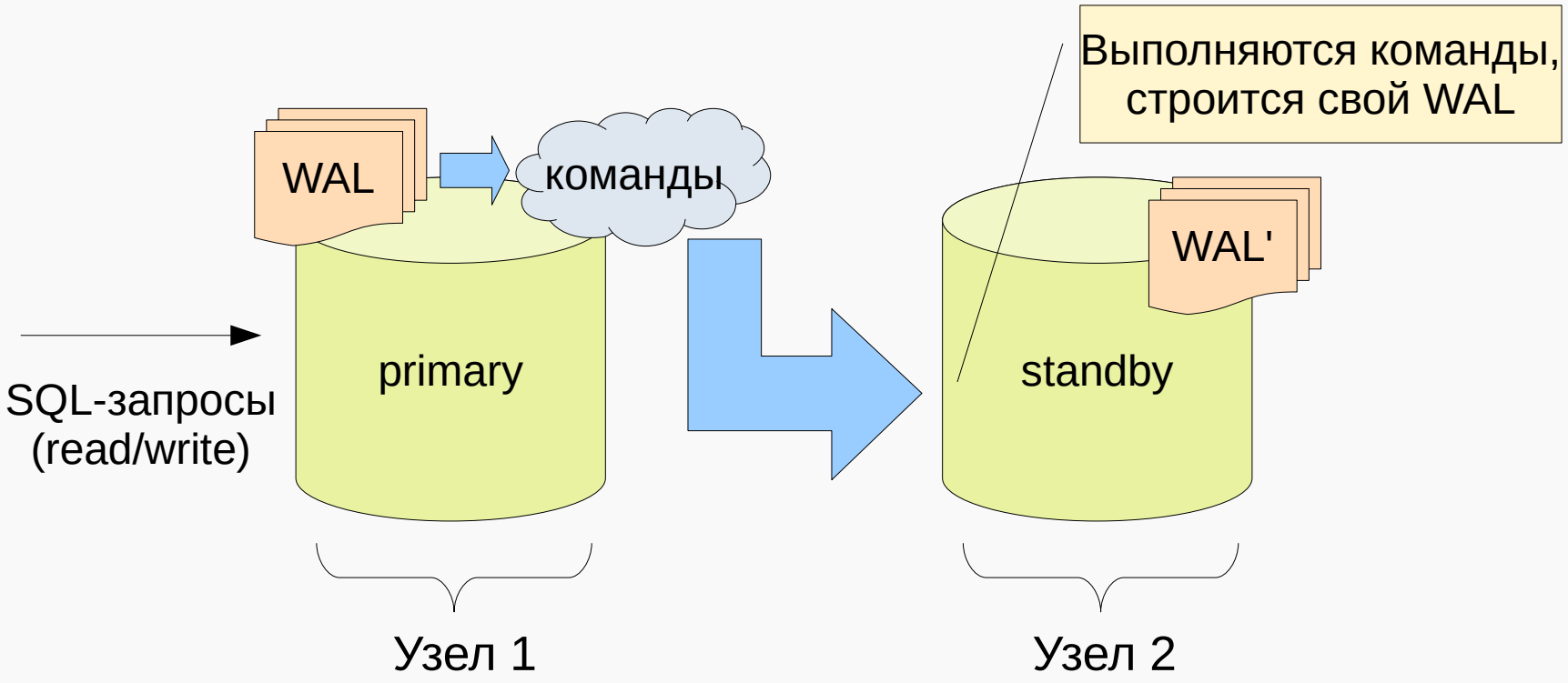
ИТМО ВТ

Логическая репликация

- **Логическая репликация** — пересылаются команды, реконструированные из WAL.
- Требуется — wal_level = logical.
- standby — осуществляет восстановление на основе полученных команд для поддержки актуального состояния.
- Используется модель «публикаций» «подписок»:
  - после логического декодирования команды публикуются на мастере;
  - опубликованные команды могут получить подписанные standby.

ИТМО ВТ

Логическая репликация



Логическая репликация в PostgreSQL (master)

- wal_level = logical
- Пользователь для репликации (master):

  CREATE USER LOGICREPL **WITH REPLICATION** …
- pg_hba.conf: host replication logicrepl all md5.
- Выдать права для LOGICREPL на нужные объекты.
- CREATE PUBLICATION all_t_publ FOR ALL TABLES;

primary

Логическая репликация в PostgreSQL

ИТМО ВТ

- wal_level = logical
- Воссоздать структуру таблиц.
- CREATE SUBSCRIPTION all_t_subscr

  CONNECTION 'user=logicrepl ... dbname=db_name'

  PUBLICATION all_t_publ;

standby

ИТМО ВТ

Особенности

- Можно реплицировать часть БД (отдельный набор таблиц).
- Не поддерживается репликация DDL.
- Возможна репликация даже для узлов с разными версиями PostgreSQL.
- Standby может работать не в read-only режиме.

ИТМО ВТ

Источники

Документация PostgreSQL:
https://www.postgresql.org/docs/14/index.html

Лицензия PostgreSQL:

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