

Цель

Перевести существующую таблицу на партиционирование в PostgreSQL, без простоя и без потери данных, при активной записи.

Рекомендуемый подход: Онлайн-миграция с использованием `pg_redef`

Инструмент `pg_redef` (ранее `pg_reorg`) — идеален для вашей задачи.

Он позволяет **переопределить структуру таблицы онлайн**, с минимальной блокировкой (только на несколько секунд в конце), и **автоматически синхронизирует изменения** через триггеры.


Шаги миграции

1. Установите `pg_redef`

```
# Установка через pgxn или из исходников
pgxn install pg_redef
```

Или вручную:

```
CREATE EXTENSION pg_redef;
```

 Убедитесь, что у вас есть права суперпользователя и доступ к установке расширений.

2. Создайте новую партиционированную таблицу (с той же структурой)

```
CREATE TABLE your_table_new (  
    -- все поля как в старой таблице
```

```


    id bigserial,
    created_at timestamp NOT NULL,
    amount numeric,
    account_id int,
    -- ...
) PARTITION BY RANGE (created_at);

-- Создайте несколько партиций (пример: по месяцам)
CREATE TABLE your_table_y2025m03 PARTITION OF your_table_new
    FOR VALUES FROM ('2025-03-01') TO ('2025-04-01');

CREATE TABLE your_table_y2025m04 PARTITION OF your_table_new
    FOR VALUES FROM ('2025-04-01') TO ('2025-05-01');

-- Добавьте партиции на будущее (или используйте pg_partman для автоматизации)

```

 Важно: структура `your_table_new` должна быть **идентична** старой таблице (включая NOT NULL, типы, порядок полей), иначе `pg_redef` не работает.

3. Запустите онлайн-переопределение

```

SELECT pg_redef.redef_table(
    'your_table_old',      -- исходная таблица
    'your_table_new',      -- новая партиционированная
    wait_for_lock := false, -- не блокировать сразу
    num_workers := 4        -- параллельная миграция (ускорит)
);

```

Что происходит:

- `pg_redef` создаёт **временные триггеры** на старой таблице.
- Копирует данные **пачками**.
- Все изменения (INSERT/UPDATE/DELETE) в старой таблице **автоматически реплицируются** в новую.
- Процесс можно прервать и возобновить.

4. Дождитесь завершения

Проверяйте статус:

```
SELECT * FROM pg_redef.status;
```

Когда `status = 'done'` — можно завершать.

5. Финальный своп (долг: < 1 секунды)

```
SELECT pg_redef.finish_redef_table('your_table_old');
```

Эта команда:

- На короткое время блокирует таблицу (долг: миллисекунды — секунды).
- Переключает старое имя на новую таблицу.
- Удаляет триггеры.
- Гарантирует целостность.



После этого:




- `your_table_old` — это **уже партиционированная таблица**.
 - `your_table_new` — временная таблица удаляется.
-

6. Очистка (опционально)

```
-- Убедитесь, что всё работает, затем:  
DROP TABLE your_table_new; -- если осталась
```

Преимущества этого подхода

Плюс	Описание
 Нет простоя	Только короткая блокировка на финальном шаге (~100–500 мс).
 Нет потери данных	Все INSERT/UPDATE/DELETE реплицируются через триггеры.

Плюс	Описание
 Можно прервать и возобновить	Если нужно остановить миграцию — просто остановите, потом продолжите.
 Производительность	<code>num_workers</code> позволяет ускорить копирование.
 Безопасно для банковского ПО	Нет ручных операций, нет риска потери данных.

Ограничения и советы

1. Первичный ключ и уникальные индексы

Должны включать **все поля из условия партиционирования** (например, `created_at`).

Иначе будет ошибка.

 Пример:

```
ALTER TABLE your_table_new ADD PRIMARY KEY (id, created_at);
```

2. Внешние ключи

Если есть FK, ссылки на эту таблицу будут временно "висеть".

Убедитесь, что `pg_redef` их поддерживает (он поддерживает, но лучше проверить).

3. Дисковое пространство

На время миграции нужно **дополнительное место** — старая + новая таблица + индексы (~2x объём).

4. Производительность во время миграции

- CPU и I/O нагрузка возрастёт.
- Лучше запускать в **период низкой нагрузки**.

5. Резервная копия перед началом

Обязательно сделайте `pg_dump` или логическую резервную копию.

Альтернатива: `pg_partman` + ручная миграция

Если `pg_redef` недоступен, можно использовать `pg_partman` с триггерами, но это сложнее и требует больше ручной работы.



Рекомендация

Используйте `pg_redef` — это **самый надёжный и проверенный способ** для онлайн-миграции таблицы в PostgreSQL без простоя.