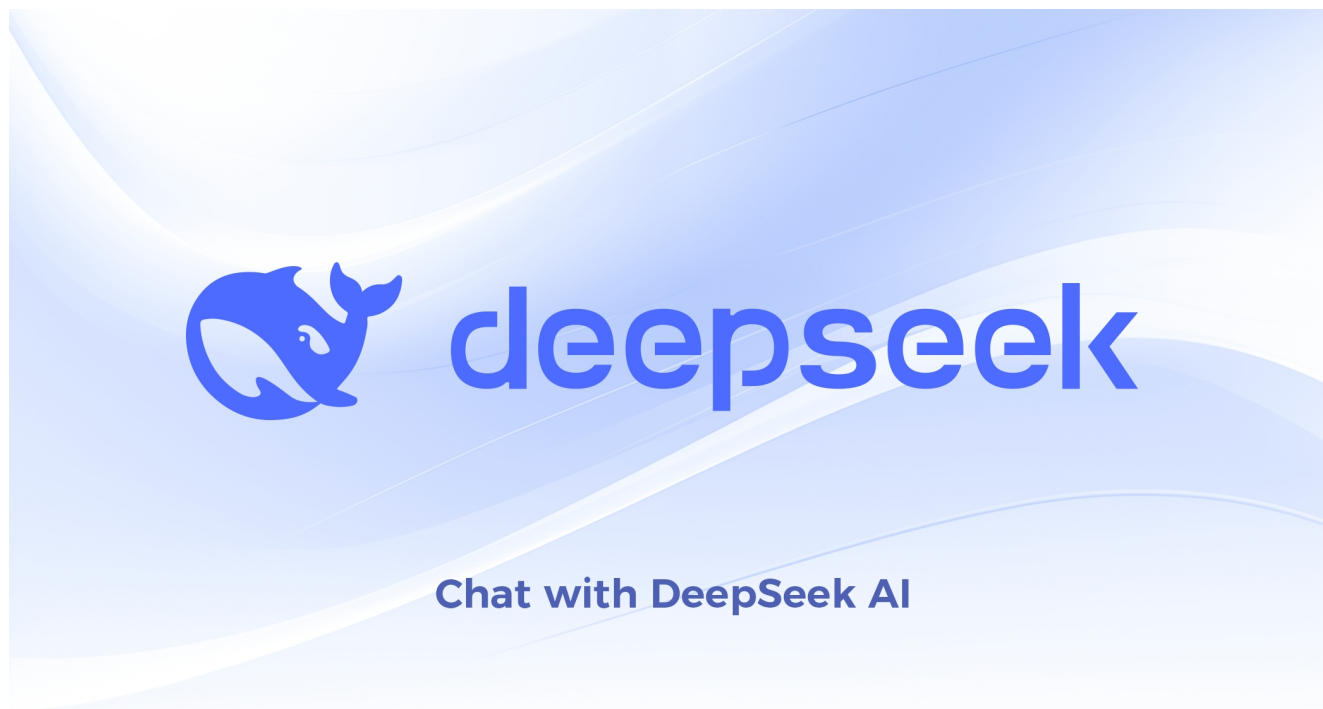


# DeepSeek - Into the Unknown

 [chat.deepseek.com/a/chat/s/51759610-b727-4ead-a50f-0cd0908e51f3](https://chat.deepseek.com/a/chat/s/51759610-b727-4ead-a50f-0cd0908e51f3)



Копирование базы данных с FDW

...

Мониторинг скорости чтения/записи nvme0n1

...

Yesterday

Расчет эффективности индексов в PostgreSQL

...

Настройка параметров CREATE TABLE в PostgreSQL

...

7 Days

Структура и использование pg\_largeobject

...

Почему отображаются только индексы и TOAST

...

Исправление ошибки в SQL-запросе

...

PostgreSQL Table Size Statistics Script Fix

...

Объединение сводных таблиц в SQL

...

SQL Query for Database Table Statistics

...

для списка баз данных "casino\_bo

...

Переписанные заголовки для таблиц БД

...

SQL Query Error: Column Not Found

...

Забыть предыдущую переписку и начать заново

...

SQL Query Syntax Error Correction Guide

...

Изменение фона комментариев в DataGrip

...

Removing Quotes from CSV Output

...

Сравнение размеров таблиц в PostgreSQL

...

Сформирован список команд подключения к БД

...

Продолжить для списка баз данных

...

PL/pgSQL и переключение баз данных

...

postgresql

...

30 Days

Настройка DataGrip: отключение подсветки комментариев

...

PostgreSQL 17 psql Commands Reference Guide

...

Запрос о дата-центре для ответа

...

Анализ параллельного выполнения в PostgreSQL

...

Преобразование скрипта в CTE без изменений

...

New chat

...

Исправление SQL-запроса для таблиц

...

Find Largest Tables in PostgreSQL Databases

...

Получение привилегий в PostgreSQL

...

Создание функции для базы данных PostgreSQL

...

PostgreSQL Script Syntax Error Fix

...

Исправление скрипта для переноса баз данных

...

Миграция данных между PostgreSQL через PL/pgSQL

...

Исправление синтаксической ошибки и упрощение кода

...

Исправление ошибки копирования данных SQL

...

Скрипт PostgreSQL для предоставления прав

...

Инициализация PostgreSQL без локалей

...

PL/pgSQL: ошибка dblink\_exes и исправление

...

Вызов удаленной функции через dblink в PL/pgSQL

...

Скрипт PL/pgSQL для синхронизации таблиц

...

Замена \*\*\* на JSONB в функции

...

Редактирование PL/pgSQL для вызова удаленной функции

...

Исправление ошибок в PL/pgSQL коде

...

Функция PL/pgSQL для интервалов времени

...

Экранирование одинарных кавычек в PL/pgSQL

...

PostgreSQL dblink pgbadger Execution Error Fix

...

PostgreSQL запрос с ошибками и исправлениями

...

Конструкция CASE в PL/pgSQL подробно

...

2025-03

Ограничения размера HTML-файла в браузерах

...

Проверка создания логов PostgreSQL

...

PostgreSQL log\_statement значение none объяснение

...

Проверка работы таблиц логирования PostgreSQL

...

Определение значения по умолчанию PostgreSQL

...

Запуск pgBadger для анализа PostgreSQL логов

...

Создание таблицы с динамическим пользователем

...

Git push rejected: resolve conflicts and pull

...

SQL-запрос с JOIN для объединения таблиц

...

Fixing SQL Unterminated String Error

...

Отключение двойных кавычек в Notepad++

...

PostgreSQL Ambiguous Column Error Resolution

...

Запуск cron для PostgreSQL процедуры

...

Исправление ошибок в коде PostgreSQL

...

Удаление записи в журналы и исключений

...

Упрощение скрипта PostgreSQL для роли

...

Создание процедуры для реиндексации индексов

...

Ошибка деления на ноль в SQL-запросе

...

PostgreSQL DBLINK Update Error Fix

...

SQL Syntax Error in DBLINK Query Fix

...

...

Определение доступа роли к базам данных

...



Get App **NEW**

My Profile



Копирование базы данных с FDW

Как с помощью FDW скопировать базу данных из примапленного сервера?



## Копирование базы данных с помощью FDW (Foreign Data Wrapper) из примапленного сервера

---

Для копирования базы данных с удаленного сервера с использованием FDW в PostgreSQL можно выполнить следующие шаги:

### 1. Установка расширения FDW

---

sql



Copy



Download

```
CREATE EXTENSION postgres_fdw;
```

### 2. Создание сервера-источника

---

sql



Copy



Download

```
CREATE SERVER remote_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'имя_хоста_или_ip', port '5432', dbname 'имя_базы_данных');
```

### 3. Создание mapping пользователя

---

sql



Copy



Download

```
CREATE USER MAPPING FOR текущий_пользователь
SERVER remote_server
OPTIONS (user 'удаленный_пользователь', password 'пароль');
```

### 4. Создание внешних таблиц

---

#### Вариант А: Импорт всей схемы

---

sql



Copy



Download

```
IMPORT FOREIGN SCHEMA public
FROM SERVER remote_server
INTO public;
```

#### Вариант В: Импорт конкретных таблиц

---

sql



Copy



Download

```
CREATE FOREIGN TABLE remote_table_name (
    id integer,
    column1 text,
    column2 timestamp
)
SERVER remote_server
OPTIONS (schema_name 'public', table_name 'actual_table_name');
```

### 5. Копирование данных

---

#### Способ 1: Создание новой таблицы и копирование

---

sql



Copy



Download

```
CREATE TABLE local_table AS  
SELECT * FROM remote_table_name;
```

## Способ 2: Использование pg\_dump/pg\_restore

---

Если у вас есть доступ к серверу, можно использовать:

bash



Copy



Download

```
pg_dump -h удаленный_сервер -U пользователь -d база_данных > dump.sql  
psql -h локальный_сервер -U пользователь -d целевая_база < dump.sql
```

## Способ 3: Использование dblink (альтернатива FDW)

---

sql



Copy



Download

```
-- Создание подключения  
SELECT dblink_connect('myconn', 'dbname=remote_db user=remote_user password=secret  
host=remote_host');  
  
-- Копирование данных  
CREATE TABLE local_table AS  
SELECT * FROM dblink('myconn', 'SELECT * FROM remote_table') AS t(id int, col1 text,  
col2 date);  
  
-- Закрытие подключения  
SELECT dblink_disconnect('myconn');
```

## Важные замечания:

---

1. Для больших баз данных копирование через FDW может быть медленным
2. Убедитесь, что у вас есть необходимые права на удаленном сервере



3. Для сложных структур (индексы, триггеры, ограничения) может потребоваться дополнительная настройка
4. Рассмотрите использование инструментов типа `pg_dump/pg_restore` для полного копирования базы

Если вам нужно скопировать всю базу данных с сохранением всех объектов, лучше использовать `pg_dump` напрямую с удаленного сервера.



Как? Мне нужно скопировать всю базу данных с сохранением всех объектов, `pg_dump` напрямую с удаленного сервера.



## Копирование всей базы данных с удаленного PostgreSQL сервера с помощью `pg_dump`

---

Для полного копирования базы данных со всеми объектами (таблицами, индексами, триггерами, функциями, схемами и т.д.) лучше всего использовать утилиту `pg_dump` напрямую.

### 1. Базовый синтаксис `pg_dump`

---

bash



Copy



Download

```
pg_dump -h <remote_host> -p <port> -U <username> -d <database_name> -F c -f <output_file> -v
```

Где:

- `-h` - хост удаленного сервера
- `-p` - порт (по умолчанию 5432)
- `-U` - имя пользователя

- **-d** - имя базы данных
- **-F c** - формат вывода (custom, рекомендуется для полного копирования)
- **-f** - имя выходного файла
- **-v** - verbose режим (опционально)

## 2. Пример полного копирования

---

bash



Copy



Download

```
pg_dump -h remote.example.com -p 5432 -U postgres -d my_database -F c -f my_database.dump -v
```

После выполнения команды вам будет предложено ввести пароль.

## 3. Восстановление базы данных на целевом сервере

---

bash



Copy



Download

```
pg_restore -h localhost -p 5432 -U postgres -d new_database -C -v my_database.dump
```

Опции **pg\_restore**:

- **-C** - создает базу данных перед восстановлением
- Остальные параметры аналогичны **pg\_dump**

## 4. Альтернативные варианты

---

### Копирование напрямую между серверами (без промежуточного файла)

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db | psql -h local_host -U local_user  
-d local_db
```

## Копирование с сжатием

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db | gzip > remote_db_dump.gz  
# Для восстановления:  
gunzip -c remote_db_dump.gz | psql -h local_host -U local_user -d local_db
```

## 5. Дополнительные полезные опции pg\_dump

---

- **--schema-only** - только схема, без данных
- **--data-only** - только данные, без схемы
- **--no-owner** - не сохранять владельцев объектов
- **--no-privileges** - не сохранять права доступа
- **--clean** - добавить команды DROP перед CREATE
- **--create** - добавить команду CREATE DATABASE

## 6. Для очень больших баз данных

---

Для больших баз можно использовать параллельный дамп:

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -j 4 -F d -f dump_dir
```

Где:

- **-j 4** - количество параллельных задач

- **-F d** - формат директории (создает папку с файлами)

Для восстановления параллельного дампа:

bash



Copy



Download

```
pg_restore -h local_host -U local_user -d local_db -j 4 -C dump_dir
```

## Важные замечания:

---

1. Убедитесь, что у пользователя есть права на чтение всей базы данных
2. Для больших баз может потребоваться значительное время и место на диске
3. Рекомендуется использовать формат "custom" (**-F c**) для полного сохранения всех объектов
4. При копировании между разными версиями PostgreSQL могут быть проблемы с совместимостью



`pg_dump -h remote_host -U remote_user -d remote_db -j 4 -F d -f dump_dir` для  
Копирование напрямую между серверами (без промежуточного файла)



Чтобы выполнить прямое копирование базы данных между серверами **без промежуточного файла**, используя параллельный режим (**-j N**), можно воспользоваться комбинацией **pg\_dump** и **pg\_restore** через пайп (pipe).

## 1. Прямое копирование с параллельным дампом и восстановлением

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -j 4 -F c | \  
pg_restore -h local_host -U local_user -d local_db -j 4 --clean --if-exists -C
```

### Опции:

---

- **-F c** – формат **custom** (обязателен для параллельного режима **-j** в **pg\_dump**).
  - **-j 4** – количество потоков (можно увеличить, если сервер мощный).
  - **--clean** – удаляет существующие объекты перед восстановлением.
  - **--if-exists** – добавляет **IF EXISTS** к командам DROP (избегает ошибок, если объекты не существуют).
  - **-C** – создает базу данных перед восстановлением (если её нет).
- 

## 2. Если нужно сохранить владельцев и привилегии

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 -C --no-owner --no-privileges
```

### Когда использовать **--no-owner** и **--no-privileges**?

---

- Если пользователи на **удаленном** и **локальном** серверах разные, лучше отключить сохранение владельцев (**--no-owner**), иначе восстановление может завершиться ошибкой.
  - **--no-privileges** отключает копирование прав доступа (GRANT/REVOKE).
- 

## 3. Если база уже существует и нужно просто обновить данные

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 --clean --if-exists
```

(без **-c**, если база уже создана).

---

#### 4. Альтернатива: **psql** + **pg\_dump** (если **pg\_restore** недоступен)

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F p -j 4 | \  
psql -h local_host -U local_user -d local_db
```

- **-F p** (plain-text SQL) – работает, но медленнее, чем бинарный формат (**-F c**).
- Не поддерживает **-j** в **psql**, поэтому распараллеливание только на этапе дампа.

---

#### Ошибки и решения

---

1. **"Role does not exist"** → Используйте **--no-owner**.
2. **"Database already exists"** → Уберите **-c** или сначала удалите старую БД.
3. Нет доступа к **pg\_dump** на удаленном сервере? → Используйте SSH-туннель:

bash



Copy



Download

```
ssh user@remote_host "pg_dump -U remote_user -d remote_db -F c -j 4" | \  
pg_restore -h localhost -U local_user -d local_db -j 4 -C
```

---

#### Вывод

---

Лучший способ для быстрого прямого копирования – это:

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 -C --clean --if-exists
```

Этот метод **не сохраняет временный файл**, работает быстро (благодаря параллельности) и подходит для больших БД.



```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 -C --clean --if-exists
```

Если база уже существует и нужно просто обновить данные



Если база данных уже существует на локальном сервере и нужно **обновить данные** (сохраняя структуру), команда будет немного отличаться. Вот оптимальный способ:

## 1. Обновление существующей базы (с очисткой старых данных)

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 --clean --if-exists
```

### Что делает:

---

- **--clean** → удаляет существующие объекты перед восстановлением (**DROP TABLE**, **DROP FUNCTION** и т.д.)
- **--if-exists** → добавляет **IF EXISTS** к командам удаления, чтобы не было ошибок, если объект отсутствует
- **Без -c** → не пытается создать базу заново (т.к. она уже есть)

## 2. Если нужно просто добавить данные без удаления (например, инкрементальное обновление)

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 --data-only
```

### Ключевые опции:

---

- **--data-only** → переносит только данные, без изменения структуры (не удаляет таблицы, не создаёт индексы заново)
  - Полезно, если схема идентична, а нужно только обновить записи.
- 

## 3. Если нужно сохранить некоторые таблицы, а остальные перезаписать

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 --exclude-table=temp_* | \  
pg_restore -h local_host -U local_user -d local_db -j 4 --clean --if-exists
```

### Опции:

---

- **--exclude-table=temp\_\*** → игнорирует таблицы, начинающиеся с **temp\_**
  - Можно также использовать **-t таблица** для выбора конкретных таблиц.
- 

## 4. Важно: возможные проблемы и решения

---

### ◆ Ошибка: *"role 'postgres' does not exist"*

---

→ Используйте **--no-owner**, если пользователи на серверах разные:

bash



Copy



Download

```
pg_restore ... --no-owner --no-privileges
```



### ◆ Ошибка: *"database is being accessed by other users"*

---

→ Остановите все подключения к БД или выполните:

bash



Copy



Download

```
psql -h local_host -U postgres -c "SELECT pg_terminate_backend(pid) FROM  
pg_stat_activity WHERE datname='local_db';"
```

### ◆ Ошибка: *"out of memory"* на больших БД

---

→ Уменьшите число потоков (-j 2 вместо -j 4) или используйте физический бекап (pg\_basebackup).

## Итоговая рекомендация

---

Если нужно **полностью перезаписать БД**, используйте:

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 --clean --if-exists
```

Если нужно **дополнить данные без удаления**:

bash



Copy



Download

```
pg_dump ... | pg_restore ... --data-only
```



нужно дополнить данные без удаления - по какому алгоритму будут дополняться данные (действия при конфликте)



При дополнении данных без удаления (**--data-only**) в PostgreSQL поведение зависит от структуры таблиц и наличия ограничений (PRIMARY KEY, UNIQUE). Вот как это работает:

## Алгоритм добавления данных при **--data-only**

---

### 1. Для таблиц без PRIMARY KEY/UNIQUE:

- Данные всегда добавляются как новые строки (INSERT)
- Возможны дубликаты

### 2. Для таблиц с PRIMARY KEY/UNIQUE:

- При попытке вставить строку с существующим ключом возникает ошибка
- Вся операция восстановления прерывается

## Решения для обработки конфликтов

---

### 1. Использование **--on-conflict-do-nothing** (PostgreSQL 15+)

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 --data-only --on-conflict-do-  
nothing
```

Пропускает конфликтующие строки, продолжая вставку остальных

### 2. Использование **--disable-triggers** + кастомного скрипта

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F c -j 4 | \  
pg_restore -h local_host -U local_user -d local_db -j 4 --data-only --disable-  
triggers
```

Отключает триггеры и ограничения на время загрузки

### 3. Ручное управление через промежуточный скрипт

---

bash



Copy



Download

```
pg_dump -h remote_host -U remote_user -d remote_db -F p --data-only > data.sql
```

Затем редактируете data.sql, добавляя для проблемных таблиц:

sql



Copy



Download

```
INSERT INTO table_name VALUES (...) ON CONFLICT (key_column) DO UPDATE SET  
    column1 = EXCLUDED.column1,  
    column2 = EXCLUDED.column2;
```

### 4. Для старых версий PostgreSQL (до 15)

---

Используйте временные таблицы:

sql



Copy



Download

```
-- На целевом сервере
CREATE TEMP TABLE temp_import AS SELECT * FROM target_table WITH NO DATA;

-- Загружаем данные во временную таблицу
pg_restore -h local_host -U local_user -d local_db --data-only --table=temp_import

-- Объединяем данные
INSERT INTO target_table
SELECT * FROM temp_import
ON CONFLICT (id) DO UPDATE SET
    name = EXCLUDED.name,
    value = EXCLUDED.value;

DROP TABLE temp_import;
```

## Рекомендации для разных сценариев

---

### 1. Полная синхронизация (обновление существующих + добавление новых):

bash



Copy



Download

```
pg_dump ... | pg_restore ... --data-only --on-conflict-do-update
```

### 2. Только добавление новых записей (без обновления существующих):

bash



Copy



Download

```
pg_dump ... | pg_restore ... --data-only --on-conflict-do-nothing
```

### 3. Для таблиц с последовательностями (serial/id):

Добавьте **--use-insert-commands** чтобы корректно обрабатывались  
автоинкрементные поля

Важно: при работе с большими таблицами предварительно протестируйте подход на копии базы.



PostgreSQL 16 Обновление существующей базы "local" (1go\_mbss\_stage) (с очисткой старых данных) из исходной базы данных "remote" (1go\_mbss\_master) нужно дополнить данные (таблицы ( users, user\_project, user\_groups, groups )) без удаления, пропустить конфликтующие строки, продолжая вставку остальных В существующей базе "local" нужно сохранить владельцев, привилегии из "remote" не переносить Прямое копирование с параллельным дампом и восстановлением Исползовать длинные команды типа (не -h а --host= ) пользователь для всех серверов gtimofeyev пароль d56f2c4eFDnofa1w "local" - prd-chat-pg-03.maxbit.private port 15434 "remote" - prd-chat-pg-02.maxbit.private port 5434



Вот полная команда для выполнения задачи с учетом всех требований:

bash



Copy



Download

```
pg_dump \
--host=prd-chat-pg-02.maxbit.private \
--port=5434 \
--username=gtimofeyev \
--dbname=1go_mbss_master \
--format=custom \
--jobs=4 \
--table=users \
--table=user_project \
--table=user_groups \
--table=groups \
--no-privileges \
--no-owner \
--verbose \
| pg_restore \
--host=prd-chat-pg-03.maxbit.private \
--port=15434 \
--username=gtimofeyev \
--dbname=1go_mbss_stage \
--jobs=4 \
--data-only \
--no-privileges \
--no-owner \
--on-conflict-do-nothing \
--verbose
```

**Пояснения:**

---

### 1. Для pg\_dump:

- `--table` - выбираем только нужные таблицы
- `--no-privileges` и `--no-owner` - не экспортируем привилегии и владельцев
- `--format=custom` и `--jobs=4` - параллельный дамп в бинарном формате

### 2. Для pg\_restore:

- `--data-only` - только данные, без изменения структуры
- `--on-conflict-do-nothing` - пропускать конфликты (PostgreSQL 16+)
- Повторно указаны `--no-privileges` и `--no-owner` для надежности

### 3. Пароль:

Система запросит пароль `d56f2c4eFDnofa1w` дважды (для дампа и восстановления).

### Альтернатива с переменной окружения для пароля:

---

bash



Copy



Download

```
export PGPASSWORD="d56f2c4eFDnofa1w"
```

```
pg_dump \
--host=prd-chat-pg-02.maxbit.private \
--port=5434 \
--username=gtimofeyev \
--dbname=1go_mbss_master \
--format=custom \
--jobs=4 \
--table=users \
--table=user_project \
--table=user_groups \
--table=groups \
--no-privileges \
--no-owner \
--verbose \
| pg_restore \
--host=prd-chat-pg-03.maxbit.private \
--port=15434 \
--username=gtimofeyev \
--dbname=1go_mbss_stage \
--jobs=4 \
--data-only \
--no-privileges \
--no-owner \
--on-conflict-do-nothing \
--verbose
```

```
unset PGPASSWORD
```

## Важные замечания:

---

1. Команда предполагает, что структура таблиц идентична в обеих базах
2. Для PostgreSQL < 15 вместо `--on-conflict-do-nothing` потребуется другой подход
3. Рекомендуется предварительно сделать бэкап целевой базы
4. Для очень больших таблиц может потребоваться увеличить `--jobs` или добавить `--single-transaction`



PostgreSQL 16 1. Обновление существующей базы "local" (1go\_mbss\_stage) (с очисткой старых данных) из исходной базы данных "remote" (1go\_mbss\_master) 2. Нужно дополнить данные для таблиц users, user\_project, user\_groups, groups - пропустить конфликтующие строки, продолжая вставку остальных 3. В существующей базе "local"

нужно сохранить владельцев, привилегии из "remote" не переносить 4. Прямое копирование с параллельным дампом и восстановлением 5. Исползовать длинные команды типа (не -h а --host= ) 6. пользователь для всех серверов gtimofeyev пароль d56f2c4eFDnofa1w 7. "local" - prd-chat-pg-03.maxbit.private port 15434 8. "remote" - prd-chat-pg-02.maxbit.private port 5434



Вот полная команда для выполнения задачи с учетом всех требований:

bash



Copy



Download



```

export PGPASSWORD="d56f2c4eFDnofa1w"

# Шаг 1: Очистка данных в целевых таблицах локальной базы
psql \
  --host=prd-chat-pg-03.maxbit.private \
  --port=15434 \
  --username=gtimofeyev \
  --dbname=1go_mbss_stage \
  --command="TRUNCATE TABLE users, user_project, user_groups, groups CASCADE;"

# Шаг 2: Параллельный перенос данных с пропуском конфликтов
pg_dump \
  --host=prd-chat-pg-02.maxbit.private \
  --port=5434 \
  --username=gtimofeyev \
  --dbname=1go_mbss_master \
  --format=custom \
  --jobs=4 \
  --table=users \
  --table=user_project \
  --table=user_groups \
  --table=groups \
  --no-privileges \
  --no-owner \
  --verbose \
  | pg_restore \
  --host=prd-chat-pg-03.maxbit.private \
  --port=15434 \
  --username=gtimofeyev \
  --dbname=1go_mbss_stage \
  --jobs=4 \
  --data-only \
  --no-privileges \
  --no-owner \
  --on-conflict-do-nothing \
  --verbose

unset PGPASSWORD

```

## Разъяснения:

---

### 1. Очистка данных:

- Используем **TRUNCATE TABLE ... CASCADE** для удаления данных из указанных таблиц и связанных объектов
- Выполняется отдельно перед загрузкой новых данных

## 2. Перенос данных:

- `--table` - выбираем только указанные таблицы
- `--no-privileges` и `--no-owner` - сохраняем текущие права и владельцев
- `--on-conflict-do-nothing` - пропускаем конфликтующие строки (для PostgreSQL 16+)

## 3. Параметры подключения:

- Явно указаны хосты и порты для обоих серверов
- Использованы длинные опции (`--host` вместо `-h`)
- Пароль задан через переменную окружения `PGPASSWORD`

## 4. Параллельное выполнение:

`--jobs=4` для параллельного выполнения и в `pg_dump`, и в `pg_restore`

## Альтернативный вариант (если нужно сохранить часть данных):

---

Если требуется не очищать таблицы полностью, а только добавить новые данные:

bash



Copy



Download

```
export PGPASSWORD="d56f2c4eFDnofa1w"
```

```
pg_dump \  
  --host=prd-chat-pg-02.maxbit.private \  
  --port=5434 \  
  --username=gtimofeyev \  
  --dbname=1go_mbss_master \  
  --format=custom \  
  --jobs=4 \  
  --table=users \  
  --table=user_project \  
  --table=user_groups \  
  --table=groups \  
  --no-privileges \  
  --no-owner \  
  --verbose \  
| pg_restore \  
  --host=prd-chat-pg-03.maxbit.private \  
  --port=15434 \  
  --username=gtimofeyev \  
  --dbname=1go_mbss_stage \  
  --jobs=4 \  
  --data-only \  
  --no-privileges \  
  --no-owner \  
  --on-conflict-do-nothing \  
  --verbose
```

```
unset PGPASSWORD
```

В этом случае конфликтующие строки будут пропущены, а новые - добавлены.



2. Нужно дополнить данные для таблиц users, user\_project, user\_groups, groups - пропустить конфликтующие строки, продолжая вставку остальных



Для дополнения данных в указанных таблицах с пропуском конфликтующих строк в PostgreSQL 16, используйте следующую команду:

```
bash
```



Copy



Download

```
export PGPASSWORD="d56f2c4eFDnofa1w"
```

```
pg_dump \  
  --host=prd-chat-pg-02.maxbit.private \  
  --port=5434 \  
  --username=gtimofeyev \  
  --dbname=1go_mbss_master \  
  --format=custom \  
  --jobs=4 \  
  --table=users \  
  --table=user_project \  
  --table=user_groups \  
  --table=groups \  
  --no-privileges \  
  --no-owner \  
  --verbose \  
| pg_restore \  
  --host=prd-chat-pg-03.maxbit.private \  
  --port=15434 \  
  --username=gtimofeyev \  
  --dbname=1go_mbss_stage \  
  --jobs=4 \  
  --data-only \  
  --no-privileges \  
  --no-owner \  
  --on-conflict-do-nothing \  
  --verbose
```

```
unset PGPASSWORD
```

## Ключевые моменты:

---

### 1. Только дополнение данных (без очистки):

- Не используем TRUNCATE, сохраняем существующие данные
- Добавляем только новые записи

### 2. Обработка конфликтов:

- `--on-conflict-do-nothing` пропускает строки с конфликтующими первичными ключами
- Вставятся только записи, которых еще нет в целевой базе

### 3. Сохранение владельцев:

- `--no-owner` сохраняет текущих владельцев объектов
- `--no-privileges` игнорирует привилегии из исходной базы

## 4. Параллельное выполнение:

--jobs=4 ускоряет процесс за счет параллелизма

## 5. Безопасность:

- Пароль задается через переменную окружения
- Используются полные имена параметров (--host вместо -h)

Для таблиц с составными первичными ключами или сложными ограничениями может потребоваться дополнительная настройка.



Где мне выполнить команду pg\_dump с параметрами?



Команду **pg\_dump** с параметрами нужно выполнять **на клиентской машине**, где установлены PostgreSQL-клиентские утилиты (включая **pg\_dump** и **pg\_restore**). Вот как это правильно сделать:

## 1. Где выполнять команду

---

- **На своем локальном компьютере** (если есть доступ к обоим серверам):

```
bash
```



Copy



Download

```
# Убедитесь, что установлен postgresql-client
# Для Ubuntu/Debian:
sudo apt-get install postgresql-client-16
```

- **На jump-сервере** (промежуточный сервер с доступом к обоим БД)
- **На одном из PostgreSQL-серверов** (если есть доступ между ними)

## 2. Проверка подключения

---

Сначала проверьте доступность серверов:

bash



Copy



Download

```
# Проверка подключения к remote
psql --host=prd-chat-pg-02.maxbit.private --port=5434 --username=gtimofeyev --
dbname=1go_mbss_master --command="SELECT 1"

# Проверка подключения к local
psql --host=prd-chat-pg-03.maxbit.private --port=15434 --username=gtimofeyev --
dbname=1go_mbss_stage --command="SELECT 1"
```

### 3. Полная команда для выполнения

---

bash



Copy



Download

```
# В Linux/macOS (в терминале)
export PGPASSWORD="d56f2c4eFDnofa1w"

pg_dump \
  --host=prd-chat-pg-02.maxbit.private \
  --port=5434 \
  --username=gtimofeyev \
  --dbname=1go_mbss_master \
  --format=custom \
  --jobs=4 \
  --table=users \
  --table=user_project \
  --table=user_groups \
  --table=groups \
  --no-privileges \
  --no-owner \
  --verbose \
| pg_restore \
  --host=prd-chat-pg-03.maxbit.private \
  --port=15434 \
  --username=gtimofeyev \
  --dbname=1go_mbss_stage \
  --jobs=4 \
  --data-only \
  --no-privileges \
  --no-owner \
  --on-conflict-do-nothing \
  --verbose

unset PGPASSWORD
```

## 4. Где именно запускать

Место выполнения	Плюсы	Минусы
<b>Локальный компьютер</b>	Безопасно, контроль над процессом	Нужно открыть доступ к серверам извне
<b>Сервер prd-chat-pg-02 (remote)</b>	Быстро (данные локально)	Нагружает production-сервер
<b>Сервер prd-chat-pg-03 (local)</b>	Проще доступ к целевой БД	Обратное подключение к remote
<b>Jump-сервер</b>	Оптимально для корпоративных сетей	Требует настройки

## 5. Если нет прямого доступа

Используйте SSH-туннель:

bash



Copy



Download

```
# Сначала создаем туннели
ssh -L 65434:prd-chat-pg-02.maxbit.private:5434 user@jump-server
ssh -L 15435:prd-chat-pg-03.maxbit.private:15434 user@jump-server

# Затем выполняем dump через туннель
pg_dump --host=localhost --port=65434 ... | pg_restore --host=localhost --port=15435
...
```

## 6. Для Windows

---

1. Установите PostgreSQL для Windows
2. Откройте **PowerShell** или **Command Prompt**
3. Используйте аналогичную команду, заменив `\` на ``` для многострочности:

powershell



Copy



Download



```

$env:PGPASSWORD="d56f2c4eFDnofa1w"
pg_dump `
  --host=prd-chat-pg-02.maxbit.private `
  --port=5434 `
  --username=gtimofeyev `
  --dbname=1go_mbss_master `
  --format=custom `
  --jobs=4 `
  --table=users `
  --table=user_project `
  --table=user_groups `
  --table=groups `
  --no-privileges `
  --no-owner `
  --verbose `
| pg_restore `
  --host=prd-chat-pg-03.maxbit.private `
  --port=15434 `
  --username=gtimofeyev `
  --dbname=1go_mbss_stage `
  --jobs=4 `
  --data-only `
  --no-privileges `
  --no-owner `
  --on-conflict-do-nothing `
  --verbose

```

## Важно!

---

1. Убедитесь, что ваша IP-адрес имеет доступ к обоим серверам
2. Для больших БД используйте **screen/tmux** (Linux) или запускайте в фоне (**nohup**)

Команда должна выполняться от пользователя с правами чтения на remote-сервере и записи на local-сервере.



🗨️ New chat



DeepThink (R1)



Search



AI-generated, for reference only