


Вставка данных в PostgreSQL

 chatgpt.com/c/6812145a-6c7c-8006-b914-849494b2ebc9



Дамп plpgsql 14 Gb содержит COPY public.user_groups (user_id, group_id)
FROM stdin; Мне нужно из этого дампа вставить строки в готовую
структуру таблицы нужной мне базы данных. Дамп в Windows. Как?

Чтобы вставить данные из дампа PostgreSQL с `COPY ... FROM stdin` в существующую
таблицу в базе данных, можно воспользоваться утилитой `psql`. Ниже пошаговая
инструкция для Windows:

 **Что у вас есть:**

- **Файл дампа** (например, `dump.sql`) содержит строку вида:

```
sql
```

```
COPY public.user_groups (user_id, group_id) FROM stdin;  
...  
\.  

```

- **Существующая база данных**, в которой уже есть таблица `public.user_groups` с нужной структурой.
- **PostgreSQL установлен в Windows** (в т.ч. доступна утилита `psql`).

✓ **Способ 1: Использовать `psql` напрямую**

Шаг 1: Откройте командную строку (cmd)

Шаг 2: Выполните команду:

```
bash
```

```
psql -h <host> -U <user> -d <dbname> -f "C:\path\to\your\dump.sql"
```

Замените:

- `<host>` — хост PostgreSQL (например, `localhost`)
- `<user>` — имя пользователя
- `<dbname>` — имя базы, куда вставляете

- "C:\path\to\your\dump.sql" — полный путь к дампу

Вам будет предложено ввести пароль, если нужно.

Важно:

- Таблица `public.user_groups` должна существовать заранее и иметь точно такие же поля (`user_id`, `group_id`) и типы данных.
- Убедитесь, что пользователь PostgreSQL имеет права **INSERT** на таблицу.
- Если в дампе есть другие команды (**DROP, CREATE** и т.д.), предварительно удалите их из файла или создайте копию дампа с только нужной частью.

Способ 2: Извлечь только данные COPY из дампа

Если дамп большой и вы хотите обработать только нужную таблицу, сделайте следующее:

Шаг 1: Скопируйте нужный фрагмент COPY ... FROM stdin в отдельный файл, например `copy_user_groups.sql`:

sql

```
COPY public.user_groups (user_id, group_id) FROM stdin;
1      10
2      10
3      11
\.
```

Шаг 2: Выполните:

```
bash
```

```
psql -h <host> -U <user> -d <dbname> -f "C:\path\to\copy_user_groups.sql"
```

Проверка

После выполнения убедитесь, что строки добавились:

```
sql
```

```
SELECT COUNT(*) FROM public.user_groups;
```

Хотите, я помогу вырезать **COPY** для **user_groups** из большого дампа?

Способ 2: Извлечь только данные COPY из дампа Если дамп большой и вы хотите обработать только нужную таблицу, сделайте следующее: Шаг 1: Скопируйте нужный фрагмент COPY ... FROM stdin в отдельный файл, например copy_user_groups.sql: для Linux

Вот как выполнить **Шаг 1**: извлечь только нужную секцию `COPY ... FROM stdin` из большого SQL-дампа в Linux:

Способ: Использовать `sed` или `awk` в Linux

Предположим, ваш дамп называется `dump.sql`, а вы хотите извлечь только блок для таблицы `public.user_groups`.

✓ Вариант 1: С помощью `awk`

```
bash
```

```
awk '/^COPY public\.user_groups /, /\^\^\^\^\./' dump.sql > copy_user_groups.sql
```

Пояснение:

- `/^COPY public\.user_groups /` — начало блока
 - `/\^\^\^\^\./` — окончание блока (`\.` — конец ввода в `COPY`)
 - `awk` вырежет всё от начала до конца включительно
-

✓ Вариант 2: С помощью `sed`

```
bash
```

```
sed -n '/^COPY public\.user_groups /,/^\.\./p' dump.sql > copy_user_groups.sql
```

После этого:

Вы можете загрузить полученный файл в нужную базу:

```
bash
```

```
psql -h <host> -U <user> -d <dbname> -f copy_user_groups.sql
```

Нужна команда, чтобы найти и проверить, сколько строк будет вставлено перед импортом?

Linux файл /var/lib/pgsql/dump/mbss_stage_15434.sql.bz2 распаковать в /home/reports Как

Чтобы распаковать файл `/var/lib/pgsql/dump/mbss_stage_15434.sql.bz2` в каталог `/home/reports` на Linux, выполните следующую команду:

```
bash
```

```
bzip2 -dkc /var/lib/pgsql/dump/mbss_stage_15434.sql.bz2 >
/home/reports/mbss_stage_15434.sql
```

Пояснение:

- `bzip2` — утилита для распаковки `.bz2` файлов.
- `-d` — распаковать (decompress).
- `-k` — сохранить оригинальный `.bz2` файл.
- `-c` — вывод распакованных данных в `stdout`.
- `>` — перенаправляет вывод в новый файл в `/home/reports`.

После выполнения:

Файл `/home/reports/mbss_stage_15434.sql` будет содержать распакованный SQL-дамп.

Нужно сразу вырезать из этого файла `COPY`-блок для `user_groups`?

```
INSERT INTO users (id, email, name, avatar_link, role, status, password,
created_at, updated_at, ready_after_login, max_active_tickets) SELECT id,
email, name, avatar_link, role, status, password, created_at, updated_at,
```

```
ready_after_login, max_active_tickets FROM users_old ON CONFLICT (email)
DO UPDATE SET id = EXCLUDED.id, name = EXCLUDED.name, avatar_link
= EXCLUDED.avatar_link, role = EXCLUDED.role, status =
EXCLUDED.status, password = EXCLUDED.password, created_at =
EXCLUDED.created_at, updated_at = EXCLUDED.updated_at,
ready_after_login = EXCLUDED.ready_after_login, max_active_tickets =
EXCLUDED.max_active_tickets; ERROR: there is no unique or exclusion
constraint matching the ON CONFLICT specification SQL state: 42P10
```

Ошибка:

```
pgsql
```

```
ERROR: there is no unique or exclusion constraint matching the ON CONFLICT
specification
SQL state: 42P10
```

означает, что в целевой таблице **users** нет уникального или исключающего ограничения по колонке **email**, а вы используете **ON CONFLICT (email)**.

✓ Решения

🔧 Вариант 1: Добавить уникальный индекс по **email**

Если **email** должен быть уникален (например, как логин пользователя), то добавьте уникальное ограничение:

sql

```
CREATE UNIQUE INDEX users_email_idx ON users(email);
```

Затем повторно выполните ваш `INSERT ... ON CONFLICT`.

Вариант 2: Использовать другой столбец, по которому уже есть уникальность

Если, например, `id` — это первичный ключ, вы можете изменить `ON CONFLICT`:

sql

```
ON CONFLICT (id) DO UPDATE ...
```

Важно: тогда `email` больше не будет ключом разрешения конфликта — обновление будет по `id`.

Вариант 3: Удалить `ON CONFLICT`, если вам не нужно поведение обновления

Если вы просто хотите вставлять данные, и дубликаты не критичны, можно удалить `ON CONFLICT`, либо использовать `ON CONFLICT DO NOTHING`.

Проверьте индекс заранее

Чтобы убедиться, есть ли нужный уникальный индекс, выполните:

sql

\d users

Или:

sql

```
SELECT indexname, indexdef
FROM pg_indexes
WHERE tablename = 'users';
```

Нужна помощь выбрать правильное поведение — по **email** или **id**?

дубликаты критичны по полю email

Если **дубликаты критичны по email**, и вы хотите, чтобы **INSERT ... ON CONFLICT (email)** работал корректно, вам **необходимо создать уникальный индекс или ограничение** на это поле.

✓ Как это сделать:

Выполните один раз в вашей базе данных:

sql

```
ALTER TABLE users  
ADD CONSTRAINT users_email_unique UNIQUE (email);
```

Или, если вы предпочитаете индекс:

sql


```
CREATE UNIQUE INDEX users_email_idx ON users(email);
```

После этого:

Ваш SQL-запрос с **ON CONFLICT (email)** начнёт работать:

sql

```
INSERT INTO users (id, email, name, avatar_link, role, status, password,
created_at, updated_at, ready_after_login, max_active_tickets)
SELECT id, email, name, avatar_link, role, status, password, created_at,
updated_at, ready_after_login, max_active_tickets
FROM users_old
ON CONFLICT (email) DO UPDATE
SET
    id = EXCLUDED.id,
    name = EXCLUDED.name,
    avatar_link = EXCLUDED.avatar_link,
    role = EXCLUDED.role,
    status = EXCLUDED.status,
    password = EXCLUDED.password,
    created_at = EXCLUDED.created_at,
    updated_at = EXCLUDED.updated_at,
    ready_after_login = EXCLUDED.ready_after_login,
    max_active_tickets = EXCLUDED.max_active_tickets;
```

 **Важно:** Убедитесь, что в таблице уже **нет дубликатов по email**, иначе добавление ограничения завершится ошибкой.

Для проверки:

sql

```
SELECT email, COUNT(*)
FROM users
GROUP BY email
HAVING COUNT(*) > 1;
```

Нужна помощь с удалением или консолидацией дубликатов, если они есть?

```
INSERT INTO user_project (user_id, project_id) SELECT user_id, project_id
FROM user_project_old ON CONFLICT (user_id) DO NOTHING; ALTER
TABLE users ADD CONSTRAINT users_email_unique UNIQUE (email);
```

переписать для этого выражения ограничение

Вы хотите переписать команду `INSERT INTO user_project ... ON CONFLICT (user_id) DO NOTHING;`, но при этом правильно задать **уникальное ограничение**, подходящее для этой конструкции.

Проблема:

Команда `ON CONFLICT (user_id)` требует, чтобы в таблице `user_project` существовало **уникальное ограничение или индекс по полю `user_id`**, но чаще всего связь между `user_id` и `project_id` — **многие ко многим**, и дубликаты по `user_id` допустимы в **разных проектах**.

Правильное решение: уникальное ограничение по паре (`user_id, project_id`)

Если вы хотите:

- не вставлять дубликаты по сочетанию `user_id` и `project_id`,
- и корректно использовать `ON CONFLICT`,

то создайте **уникальное ограничение по двум полям**:

sql

```
ALTER TABLE user_project
ADD CONSTRAINT user_project_user_id_project_id_unique UNIQUE (user_id,
project_id);
```

✓ После этого команда работает:

sql

```
INSERT INTO user_project (user_id, project_id)
SELECT user_id, project_id FROM user_project_old
ON CONFLICT (user_id, project_id) DO NOTHING;
```

⚠ Почему нельзя использовать только **(user_id)**?

Потому что:

- один пользователь может участвовать в нескольких проектах,
- **user_id** не должен быть уникальным сам по себе в таблице связей.

Нужно помочь найти и удалить дубликаты перед созданием ограничения?

```
ALTER TABLE user_project ADD CONSTRAINT  
user_project_user_id_project_id_unique UNIQUE (user_id, project_id);  
INSERT INTO user_project (user_id, project_id) SELECT user_id, project_id  
FROM user_project_old ON CONFLICT (user_id) DO NOTHING; ERROR:  
there is no unique or exclusion constraint matching the ON CONFLICT  
specification SQL state: 42P10
```

Ошибка:

pgsql

```
ERROR: there is no unique or exclusion constraint matching the ON CONFLICT  
specification
```

возникает потому, что вы написали:

sql

```
ON CONFLICT (user_id) DO NOTHING;
```

а создали ограничение по (user_id, project_id):

sql

```
ALTER TABLE user_project  
ADD CONSTRAINT user_project_user_id_project_id_unique UNIQUE (user_id,  
project_id);
```

✓ Решение: Привести **ON CONFLICT** к существующему уникальному ограничению

Измените SQL-запрос так:

sql

```
INSERT INTO user_project (user_id, project_id)  
SELECT user_id, project_id FROM user_project_old  
ON CONFLICT (user_id, project_id) DO NOTHING;
```



Почему это работает?

- **ON CONFLICT (...)** требует **точного совпадения** с уникальным ограничением или уникальным индексом.
 - У вас уникальность по **(user_id, project_id)**, значит, и **ON CONFLICT** должен быть по **этим же полям**.
-

Нужно проверить, есть ли уже дубликаты по **(user_id, project_id)** в **user_project_old** перед вставкой?

