ChatGPT - SQL Expert (QueryGPT)

chatgpt.com/g/g-m5lMeGifF-sql-expert-querygpt/c/68302b1b-0c90-8006-8202-88666aa0192b

Postgresql: Могут ли TOAST-данные встроены в основной файл?

В PostgreSQL TOAST (The Oversized-Attribute Storage Technique) используется для хранения больших значений таких типов, как `TEXT`, `BYTEA`, `JSONB`, и др., которые не помещаются в обычную строку таблицы.

Короткий ответ:
Да, TOAST-данные *могут быть встроены* в основной файл таблицы, если они достаточно малы. Это называется **in-line storage**.

Подробнее:

PostgreSQL использует следующий порядок при хранении больших атрибутов:

1. **In-line (обычное хранение):**
   Если значение достаточно маленькое (до ~2KB), оно хранится прямо в основной строке таблицы, в основном файле (heap).

2. **Compressed in-line:**
   Если значение чуть больше, PostgreSQL может попытаться сжать его и оставить в строке (всё ещё in-line, но в сжатом виде).

3. **Out-of-line (TOAST):**
   Если значение всё равно слишком большое, оно помещается во внешнюю TOAST-таблицу, и в основной строке остаётся только ссылка.

Управление поведением:

Параметр хранения можно задать при создании таблицы или столбца:

```
sql
```

```
CREATETABLE example (
    data TEXT STORAGE PLAIN   -- принудительно in-line, без TOAST
);
```

Типы STORAGE:

- `PLAIN` – всегда in-line, без сжатия.
- `EXTENDED` – сжатие и TOAST (по умолчанию).
- `EXTERNAL` – без сжатия, но можно вынести в TOAST.
- `MAIN` – попытаться оставить in-line, но можно сжать.

Проверка, как хранятся данные:

```
sql
```

```
SELECT pg_column_size(data), pg_total_relation_size('example')
FROM example;
```

Для анализа TOAST можно использовать `pg_toast.*` или функции вроде `pg_size_pretty`.

Follow GPT creator profile on LinkedIn https://linkedin.com/in/khanukov/

> SELECT pg_column_size(attributes), pg_total_relation_size('messages') FROM messages;
> показал, что pg_column_size;pg_total_relation_size 5;35644260352 5;35644260352 5;35644260352 5;35644260352 5;35644260352 5;35644260352

Результаты `pg_column_size(attributes) = 5` говорят о следующем:

- Значение в столбце `attributes` занимает **всего 5 байт** в каждой строке.
- Это означает, что фактические данные **не хранятся в самой строке**, а скорее всего находятся во **внешней TOAST-таблице**, а в строке — только **указатель (TOAST reference)**.

Значение `pg_column_size` включает:

- 1 байт — тег типа (если variable-width тип),
- 4 байта — указатель на TOAST (если используется TOAST).

Подтверждение:

Если хотите проверить, используется ли TOAST-таблица, выполните:

```
sql



SELECT relname
FROM pg_class
WHERE relname LIKE'pg_toast%'AND relnamespace = (
    SELECT relnamespace FROM pg_class WHERE relname ='messages'
);
```

Затем посмотрите объём TOAST-таблицы:

```
sql



SELECT pg_size_pretty(pg_total_relation_size('pg_toast_<oid>'));
```

Замените `pg_toast_<oid>` на имя, полученное выше.

Если вы хотите вернуть данные обратно in-line:

1. Убедитесь, что размер значений не превышает лимиты (примерно 2KB).
2. Измените `STORAGE`:

```
sql
```

```
ALTERTABLE messages ALTERCOLUMN attributes SET STORAGE MAIN;
```

3. Принудите PostgreSQL пересохранить строки (например, `UPDATE` с самим собой):

```
sql
```

```
UPDATE messages SET attributes = attributes;
```

Это может вернуть данные обратно в основную таблицу — если они достаточно малы.

Follow GPT creator profile on LinkedIn https://linkedin.com/in/khanukov/

> SELECT relname FROM pg_class WHERE relname LIKE 'pg_toast%' AND relnamespace = (SELECT relnamespace FROM pg_class WHERE relname = 'messages'); пусто

Если запрос не вернул строк, это означает следующее:

Возможные причины:

1. **TOAST-таблица действительно не используется:**
   - Значения в `attributes` могут быть *короткими* (или пустыми), и TOAST не был задействован.
   - `pg_column_size = 5` может указывать на `NULL` или короткие значения типа `''` (пустая строка), которые PostgreSQL оптимизирует очень сильно.
2. **TOAST-таблица может существовать, но в другом `namespace`:**

   Ваш подзапрос ищет TOAST-таблицу **в том же namespace**, что и `messages`. Это корректно, но если `messages` — представление (`view`) или `foreign table`, это может не сработать.

3. `attributes` **— тип, не использующий TOAST (например,** `INT` **или** `BOOLEAN`**)**

   TOAST работает только для типов с переменной длиной (`TEXT`, `BYTEA`, `JSONB`, и т.п.). Если `attributes` — это, например, `INTEGER[]`, PostgreSQL может использовать другие механизмы хранения.

Что можно сделать дальше:

1. Узнать точный тип столбца:

```
sql



SELECT column_name, data_type, udt_name
FROM information_schema.columns
WHERE table_name ='messages'AND column_name ='attributes';
```

2. Проверить, есть ли TOAST-таблица напрямую через `pg_class` и `pg_toast_relid`:

```
sql



SELECT reltoastrelid
FROM pg_class
WHERE relname ='messages';
```

Если `reltoastrelid = 0`, значит **TOAST-таблица отсутствует**.

Если не 0, получить её имя:

```
sql



SELECT relname
FROM pg_class
WHERE oid = (SELECT reltoastrelid FROM pg_class WHERE relname ='messages');
```

Если `attributes` — это `TEXT`, `BYTEA`, `JSONB` и значения реально большие, но TOAST не создаётся — возможно, вы используете `STORAGE PLAIN`:

```
sql
```

```
SELECT attname, attstorage
FROM pg_attribute
WHERE attrelid ='messages'::regclass AND attname ='attributes';
```

Значения `attstorage`:

- `p` = PLAIN
- `e` = EXTENDED (по умолчанию для `TEXT`)
- `m` = MAIN
- `x` = EXTERNAL

Если стоит `p`, TOAST будет отключён.

Follow GPT creator profile on LinkedIn https://linkedin.com/in/khanukov/

> SELECT relname FROM pg_class WHERE oid = (SELECT reltoastrelid FROM pg_class WHERE relname = 'messages'); вернул pg_toast_17017342 как найти путь физического хранения файла

Чтобы найти **физический путь хранения файла TOAST-таблицы** (например, `pg_toast_17017342`), нужно понимать, как PostgreSQL организует файлы на диске:

📂 Где хранятся данные PostgreSQL:

В файловой системе PostgreSQL таблицы и индексы хранятся в каталоге `base/` внутри `PGDATA`.

Путь будет примерно таким:

```
bash
```

```
$PGDATA/base/<database_oid>/<relfilenode>
```

🔍 Как найти `relfilenode` и `database_oid`:

1. Получить `relfilenode` TOAST-таблицы:

```
sql
```

```
SELECT relname, relfilenode
FROM pg_class
WHERE relname ='pg_toast_17017342';
```

2. Получить OID вашей базы данных:

```
sql
```

```
SELECT oid, datname
FROM pg_database
WHERE datname = current_database();
```

📄 Итоговый путь:

Если, например:

- `relfilenode = 1234567`
- `database_oid = 98765`
- `$PGDATA = /var/lib/postgresql/15/main`

Тогда файл будет:

```
swift
```

```
/var/lib/postgresql/15/main/base/98765/1234567
```

🔁 Учитывайте:

- Если таблица большая, она может разбиваться на сегменты:

```
lua
```

```
.../1234567-- первый сегмент
.../1234567.1-- второй
.../1234567.2-- и т.д.
```

- Индексы TOAST тоже будут отдельными файлами (`pg_toast_17017342_index`, и т.п.).

🛑 ВАЖНО:

- **Никогда не редактируйте эти файлы вручную.**
- Используйте `pgstattuple`, `pg_toast`, или `pg_dump` для безопасного анализа/экспорта содержимого.

Follow GPT creator profile on LinkedIn https://linkedin.com/in/khanukov/

relname;relfilenode pg_toast_17017342;23835274 /var/lib/pgsql/14/messenger/base/16709033
какой будет итоговый путь

У вас есть все необходимые данные:

- `relfilenode = 23835274`
- `database_oid = 16709033`

- `PGDATA = /var/lib/pgsql/14/messenger`

📄 Итоговый путь к файлу TOAST-таблицы:

```
swift

/var/lib/pgsql/14/messenger/base/16709033/23835274
```

📝 Примечания:

- Это основной сегмент TOAST-таблицы. Если таблица большая, также могут быть:

  ```
  swift

  /var/lib/pgsql/14/messenger/base/16709033/23835274.1/var/lib/pgsql/14/messenger/base/1670
  9033/23835274.2...
  ```

- Если вы хотите убедиться в использовании пространства или анализировать размер:

  ```
  bash

  du -sh /var/lib/pgsql/14/messenger/base/16709033/23835274*
  ```

- **Никогда не редактируйте** эти файлы напрямую. Для безопасной работы используйте SQL и утилиты PostgreSQL.

Follow GPT creator profile on LinkedIn https://linkedin.com/in/khanukov/

Отлично! Теперь пойдем от обратного: 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.1 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.10 1.1G

/var/lib/pgsql/14/messenger/base/16709033/23835271.11 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.12 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.13 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.14 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.15 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.16 344M
/var/lib/pgsql/14/messenger/base/16709033/23835271.17 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.2 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.3 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.4 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.5 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.6 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.7 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.8 1.1G
/var/lib/pgsql/14/messenger/base/16709033/23835271.9 Как найти имя с OID "23835271" и вообще, все имена объектов в базе данных "16709033" ?

Отличный вопрос! Чтобы найти имя объекта по его `relfilenode` (например, `23835271`) и получить список **всех объектов (таблиц, индексов, TOAST и т.д.) в базе с OID = 16709033**, нужно сделать следующее:

✔️ 1. Получить имя объекта по `relfilenode`

```
sql



SELECT relname, relkind, reltoastrelid, relnamespace::regnamespace, reltablespace
FROM pg_class
WHERE relfilenode =23835271;
```

Это вернёт:

- `relname` — имя объекта (таблицы, индекса и т.д.)
- `relkind` — тип объекта:
    - `r` = обычная таблица
    - `t` = TOAST
    - `i` = индекс
- `relnamespace` — схему
- `reltoastrelid` — если объект связан с TOAST-таблицей

✔️ 2. Все объекты в базе (OID: `16709033`)

```
sql
```

```
SELECT c.oid AS class_oid, c.relname, c.relkind, c.relfilenode, n.nspname
FROM pg_class c
JOIN pg_namespace n ON n.oid = c.relnamespace
ORDERBY c.relfilenode NULLS LAST;
```

Это покажет:

- имя объекта
- тип (таблица, индекс, TOAST)
- `relfilenode` — чтобы сопоставить с физическими файлами
- `nspname` — имя схемы

📝 Дополнительно: если у вас есть shell-доступ

Или из консоли `psql` в нужной базе (`16709033`) можно сделать:

```
bash
```

```
psql -d <dbname> -c "SELECT relname, relfilenode FROM pg_class WHERE relfilenode = 23835271;"
```

⚠️ Уточнение:

PostgreSQL использует `relfilenode` как имя файла на диске **только если таблица не перемещена вручную или не использует кастомный tablespace**.

Follow GPT creator profile on LinkedIn https://linkedin.com/in/khanukov/

SELECT relname, relkind, reltoastrelid, relnamespace::regnamespace, reltablespace FROM pg_class WHERE relfilenode = 23835271;
relname;relkind;reltoastrelid;relnamespace;reltablespace messages;r;17017354;public;0

Отлично! Вот расшифровка вашего результата:

🎯 Найденный объект:

- **Имя таблицы:** `messages`
- **Тип (`relkind`):** `r` — обычная таблица
- **TOAST-таблица OID:** `17017354`
- **Схема:** `public`
- **Tablespace:** `0` (по умолчанию, в `$PGDATA/base/`)

🔍 Что это значит:

- Файл с `relfilenode = 23835271` и его сегменты (например, `.1`, `.2`, ...) — это **основной heap-файл** таблицы `public.messages`.
- Он содержит **основные строки таблицы**, исключая вынесенные в TOAST значения (например, большие `TEXT`/`BYTEA`/`JSONB` поля).

➕ Дополнительно:

Если хотите посмотреть связанный TOAST-объект (с `relfilenode` для файла TOAST):

```
sql



SELECT relname, relfilenode
FROM pg_class
WHERE oid =17017354;
```

Так вы получите физический файл TOAST-таблицы для `messages`.

Follow GPT creator profile on LinkedIn https://linkedin.com/in/khanukov/

> Смотри, но у этой БД много toast?

Да, вы всё правильно заметили: в вашей базе данных действительно **много TOAST-таблиц** — на скриншоте видно как минимум 62 таблицы с префиксом `pg_toast_`.

📌 Почему так много TOAST-таблиц?

Каждая **обычная таблица**, содержащая хотя бы один потенциально "большой" столбец (например, `TEXT`, `BYTEA`, `JSONB`, `XML`, `ARRAY`, и т.п.), может иметь **свою собственную TOAST-таблицу**.

- TOAST-таблица создаётся **автоматически**, если хотя бы один столбец таблицы **использует** `STORAGE EXTENDED` **по умолчанию** и когда значения действительно выходят за пределы in-line хранения.
- Она не отображается в обычной схеме (`public`, `information_schema`), а именно в системной — `pg_toast`.

🔍 Как связать TOAST с основной таблицей?

Чтобы узнать, **какая TOAST-таблица принадлежит какой основной**, используйте:

```
sql
```

```
SELECT
  c.oid AS table_oid,
  c.relname AS table_name,
  c.reltoastrelid AS toast_oid,
  t.relname AS toast_table_name
FROM pg_class c
LEFTJOIN pg_class t ON t.oid = c.reltoastrelid
WHERE c.reltoastrelid !=0ORDERBY c.relname;
```

🧠 Полезно знать:

- TOAST-таблицы **не удаляются**, даже если вы очистили данные в основной таблице. Размер на диске остаётся до **VACUUM FULL** или **CLUSTER**.
- Вы можете оценить объёмы TOAST через `pg_total_relation_size()` или `pg_size_pretty()`.

Хорошая практика — регулярно мониторить размеры TOAST-таблиц в больших БД, особенно если вы храните JSON, логи или BLOB-данные.

Follow GPT creator profile on LinkedIn https://linkedin.com/in/khanukov/