PG BootCamp Russia 2025 Ekaterinburg

PGBootCamp.ru

Археология данных в PostgreSQL. Как «выкопать» данные из нечитаемых ~~гробниц~~ таблиц?

Михаил Торговкин

Виды поврежденных таблиц в БД

Логические повреждения — это повреждения, при которых нарушается логическая связь между связанными записями в системе (рассматривать не будем в этом докладе).

Физические повреждения — это повреждения структуры файлов таблиц или их содержимого в каталоге PGDATA, включая пользовательские данные и\или системные файлы базы данных.

Виды ошибки, указывающие на физические повреждения таблиц:

1. `MultiXactId 2181984 has not been created yet -- apparent wraparound`
2. `found xmax 25973 from before relfrozenxid 106464494`
3. `failed to find parent tuple for heap-only tuple at (0,1) in table`
4. `Invalid memory alloc requestsize`

Важные моменты и обязательные действия

Перед диагностикой и восстановлением данных сделайте копию каталога БД.

Ваши средства бэкапирования могут уже не работать.

Важно понимать, что после того, как данные удалось восстановить, база продолжает оставаться неконсистентной.

При физических повреждениях возникает угроза некорректной работы СУБД и потери информации.

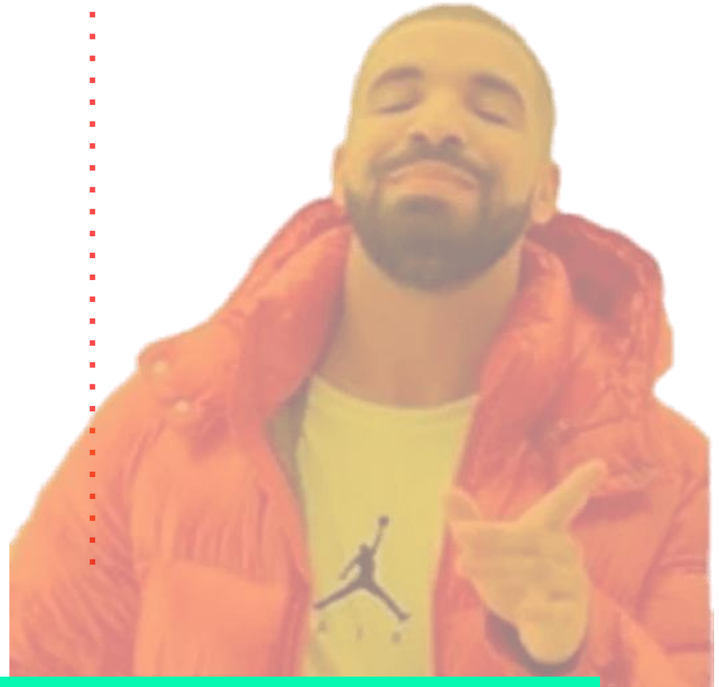
После восстановления данных использовать БД нельзя.

Восстановить кластер PG из последнего бэкапа неповрежденной БД, загрузить последние изменения из БД где были восстановленные таблицы.

Что делать?

«Что делать, если система не читает таблицу?»

Ответ прост:
Действуйте как опытный диагност, не спешите с восстановлением, а проведите полную диагностику.



И что НЕ НАДО делать.

НЕ НАДО:

Лечить только симптомы болезни, не выяснив её причину: нельзя ограничиваться исправлением лишь видимых неполадок:

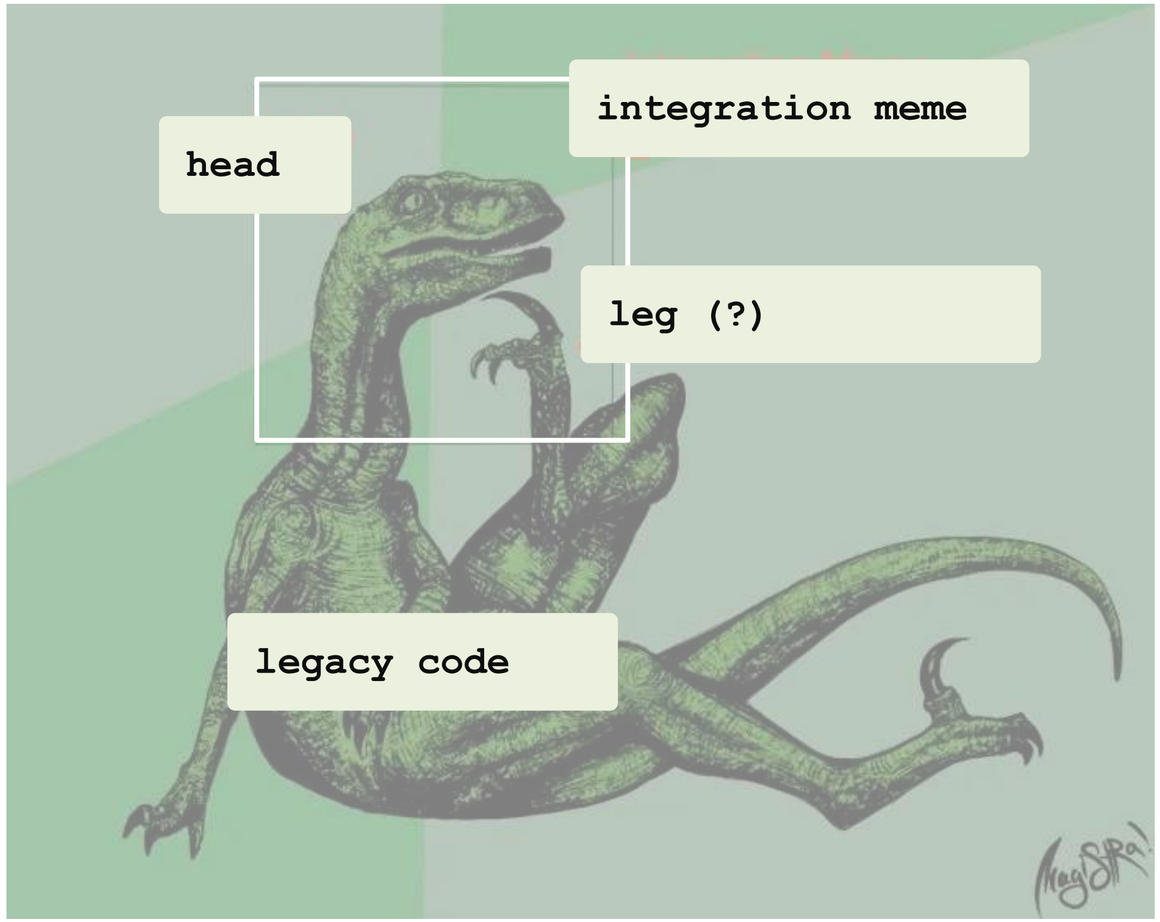
- **это может скрыть более глубокие проблемы в будущем;**
- **восстановив данные в нескольких таблицах, продолжать использовать базу.**

Шаг 1. Реконструкция событий сбоя, поиск ошибок, диагностика первопричин

```
lnav
/var/lib/postgresql/16/main/log/*.log

поиск error, fatal
```

Получим информацию с чего всё началось, получим имена поврежденных таблиц, но важно понимать, что в ошибках лога могут быть не все поврежденные таблицы.



Шаг 2. Получение списка поврежденных таблиц

На этом шаге получаем **список поврежденных таблиц** и делаем **выгрузку данных** из «живых» таблиц на диск.

Инструменты:

```
- pg_dump
```

Запустить для всей базы, или сгенерировать скрипт выгрузив каждую таблицу в отдельный файл.

— команда COPY. сгенерировать скрипт со списком таблиц и выгрузить каждую в отдельный файл.

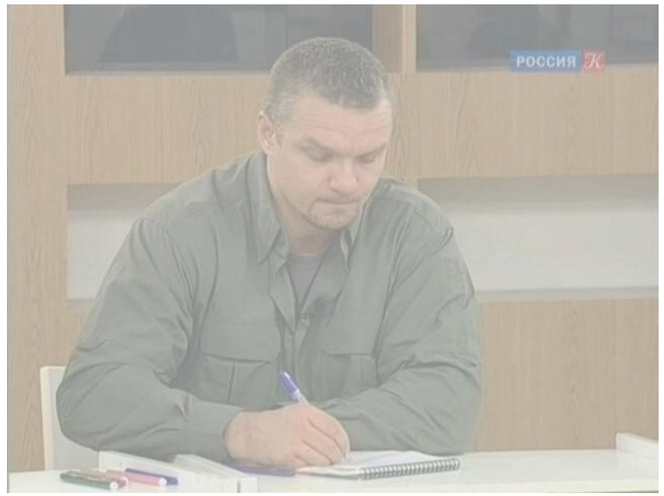
Восстановить в новом кластере может получиться быстрее.

Выбирайте способ и инструмент, которые вам представляются более предпочтительными, удобными, быстрыми.

Шаг 3. Составление плана работ по восстановлению таблиц

Нет универсальных советов по восстановлению, поскольку каждый случай уникален и требует учета конкретного состояния БД.

Но важно определиться, где сохранять восстанавливаемые данные.



Возможные варианты:

- сохранить данные в таблицах с другим именем в этой же базе.
- делать выгрузку в текстовые файлы.
- выгружать данные в удалённую таблицу – (FDW)
- проверка логистической целостности восстановленных данных.

Если есть возможность, хорошо будет, если ваш план кто-то посмотрит из коллег.

Целесообразно назначить ответственного сотрудника, который будет осуществлять мониторинг намеченных действий и их результатов, своевременно выявлять возможные отклонения от намеченного плана.

Шаг 4. Восстановление таблиц.

4.1 Определение количества строк в таблице.

В таблицах postgres есть системное поле **ctid** *(Current Tuple ID)*

Поле **ctid** состоит из двух частей Формат представления: *(block number, line number) (0,5).*

Блок (block number) — номер блока. Значение может быть 2^32 = 4294967296

Позиция (line number) — номер строки внутри блока значение от 0 до 255

Зная **ctid** можно прочитать конкретную строку, что мы и будем делать в двух вложенных циклах.

Первым делом нужно узнать примерное количество «живых» строк в таблице.

Важно смотреть, когда статистика по таблице собиралась последний раз.

```
SELECT n_live_tup, --строк в таблице,
  last_analyze --последний ручной analyze,
  last_autoanalyze --последний autoanalyze
FROM pg_stat_user_tables
WHERE relname = 'attachments';
```

Полученное значение **n_live_tup** следует записать, и увеличить его из расчёта вероятного количества строк сколько могло быть записано после последнего **analyze**. Это будет значение количества итераций первого цикла.

Если **analyze** давно не выполнялся, то это значение можно установить на своё усмотрение максимально возможное которое могло быть в таблице.

Шаг 4. Восстановление таблиц.

4.2 Создание таблиц для сохранения читаемых и нечитаемых строк таблицы

Для сохранения **ctid** создадим 2 таблицы

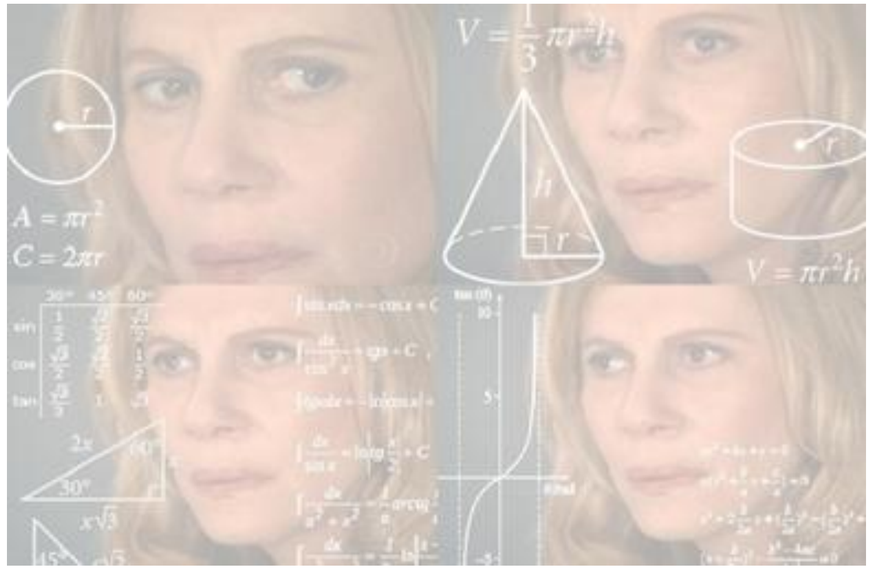
```
CREATE TABLE public.tmp_at_ok (
	mctid text NULL
);
CREATE TABLE
public.tmp_at_error (
	mctid text NULL
);
```

Как вариант, можно создать таблицу — копию структуры восстанавливаемой таблицы, и прочитанные строки сразу записывать в неё.

Если в таблице будет PK — сразу будет выявлены дублирующие индексы.

Шаг 4.
Восстановление таблиц.

4.3 Цикл сбора читаемых и нечитаемых строк



```
DO
$$ DECLARE
    block_num INT; row_num INT;
    max_block_num CONSTANT INT := 2147483647; -- количество обрабатываемых
блоков.
    max_row_num CONSTANT INT := 255;
        ctid_value TEXT; commit_counter INT := 0;
    err_message TEXT; err_detail TEXT; err_context TEXT; query_text TEXT;
BEGIN
FOR block_num IN 0..max_block_num LOOP
   FOR row_num IN 0..max_row_num LOOP
    ctid_value := FORMAT('(%s,%s)', block_num, row_num);
        query_text := FORMAT(
            'INSERT INTO tmp_at_ok (mctid)'
            'SELECT ctid FROM attachments WHERE ctid = ''%s'';',
            ctid_value
        );
        BEGIN
            EXECUTE query_text; commit_counter := commit_counter + 1;
        EXCEPTION
            WHEN OTHERS THEN GET STACKED DIAGNOSTICS
                    err_message = MESSAGE_TEXT,
                    err_detail = PG_EXCEPTION_DETAIL,
                    err_context = PG_EXCEPTION_CONTEXT;
                RAISE NOTICE
'Ошибка при обработке ctid: %, сообщение: %, подробности: %, контекст: %',
                ctid_value, err_message, err_detail, err_context;
                query_text := FORMAT(
                    'INSERT INTO tmp_at_error (mctid) VALUES (%L);',
                    ctid_value);
                EXECUTE query_text;
                CONTINUE;
        END;
        IF commit_counter >= 1000 THEN COMMIT; commit_counter := 0;
        END IF;
    END LOOP;
END LOOP;
COMMIT;
END
$$;
```

Шаг 4. Восстановление таблиц.

4.4 Восстановление данных из читаемых строк

Теперь на основе данных **tmp_at_ok** можно скопировать данные в новую таблицу, предварительно создав её на основе DDL старой таблицы.

```
insert into attachments_new
select
	attachments.*
from
	attachments
inner join tmp_at_ok on
	attachments.ctid =
tmp_at_ok.mctid::tid;
```

Это упрощенный скрипт, вставку данных лучше сделать в цикле, чтобы при вставке данных выявить ctid записи у которых есть дублированные ID.

Или сделать это отдельным скриптом выявить сразу все дублированные ID.

Новый кластер

После того как данные из таблиц были восстановлены, обязательно нужно поднять новый кластер БД.

Возможные варианты:

1. Поднять с нуля накатить DDL нужной базы залить в неё данные
2. Восстановить кластер PG из последнего бэкапа неповрежденной БД, загрузить последние изменения из БД где были восстановленные таблицы.

Экземпляр PG не запускается

При запуске СУБД проверяется необходимость восстановления БД из журналов WAL.

Если сервер не запускается из-за нехватки WAL или поврежденных WAL файлов.

Но не спешите паниковать!

Утилита **pg_resetwal** способна возродить базу данных.

Тестовый прогон:

```
pg_resetwal /path/to/data/directory
```

Если и это не помогло.

```
pg_resetwal -f /path/to/data/directory
```

https://postgrespro.ru/docs/postgresql/16/app-pgresetwal

Последний оплот спасения

`pg_filedump`

PG BootCamp Russia 2025 Ekaterinburg

PGBootCamp.ru

Ваши вопросы?