

# Многоверсионность АВТОЧИСТКА



## **Авторские права**

© Postgres Professional, 2019 год.

Авторы: Егор Рогов, Павел Лузанов

## **Использование материалов курса**

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

## **Обратная связь**

Отзывы, замечания и предложения направляйте по адресу:

[edu@postgrespro.ru](mailto:edu@postgrespro.ru)

## **Отказ от ответственности**

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Автоматическая очистка (autovacuum)

Автоанализ

Настройка процесса автоочистки

Работает аналогично обычной очистке

Выполняется периодически

для таблиц с определенным количеством изменений  
в том числе для toast-таблиц

Процесс `autovacuum launcher`

постоянно запущен  
планирует запуск рабочих процессов

Процессы `autovacuum worker`

запускаются процессом `postmaster` по просьбе `autovacuum launcher`  
подключаются к заданной БД, перебирают и очищают таблицы

Автоматическая очистка — механизм, позволяющий запускать обычную очистку в определенные моменты времени, в зависимости от количества изменений в таблицах. Это удобнее и правильнее, чем запуск по расписанию (`cron`), поскольку учитывает динамику системы.

При включенной автоочистке в системе всегда присутствует процесс `autovacuum launcher`, который занимается планированием работы. Реальную очистку выполняют процессы `autovacuum worker`, несколько экземпляров которых могут работать параллельно.


Процессы `autovacuum worker` запускаются процессом `postmaster` по просьбе `autovacuum launcher` (поскольку именно `postmaster` отвечает за порождение всех новых процессов).

<https://postgrespro.ru/docs/postgresql/10/routine-vacuuming#AUTOVACUUM>

## Алгоритм

для каждой базы данных (в которой есть активность)  
запускать рабочий процесс раз в *autovacuum\_naptime*  
количество рабочих процессов  $\leq$  *autovacuum\_max\_workers*

## Настройки

*autovacuum* = on  
*track\_counts* = on } должны быть включены оба  
*autovacuum\_naptime* = 60 s  
*autovacuum\_max\_workers* = 3 

Процесс *autovacuum launcher* составляет список баз данных, в которых есть какая-либо активность (точнее, для которых собирается статистика использования).

Новый рабочий процесс запускается раз в *autovacuum\_naptime* для каждой БД (то есть при наличии N баз процессы будут порождаться в N раз чаще). Общее количество рабочих процессов ограничено параметром *autovacuum\_max\_workers*.

Для того, чтобы автоматическая очистка работала в принципе, должны быть установлены параметры *autovacuum* и *track\_counts*. Последний включает сбор статистики использования.

## Алгоритм

выполнять по очереди очистку всех таблиц (включая TOAST),  
в которых число ненужных версий строк превышает  
 $autovacuum\_vacuum\_threshold +$   
 $+ autovacuum\_vacuum\_scale\_factor * \text{число строк в таблице}$

а также выполнять анализ всех таблиц,  
в которых число изменившихся версий строк превышает  
 $autovacuum\_analyze\_threshold +$   
 $+ autovacuum\_analyze\_scale\_factor * \text{число строк в таблице}$

в одной БД может параллельно работать несколько процессов

Рабочий процесс подключается к указанной базе данных и строит список всех таблиц, материализованных представлений и toast-таблиц, требующих очистки — у которых число ненужных («мертвых») версий строк превышает пороговое значение, заданное двумя параметрами:

- *autovacuum\_vacuum\_threshold* определяет абсолютный минимум,
- *autovacuum\_vacuum\_scale\_factor* определяет относительную долю изменившихся строк.

А также строит список объектов, требующих анализа — у которых число измененных версий строк превышает пороговое значение, заданное двумя аналогичными параметрами:

- *autovacuum\_analyze\_threshold*,
- *autovacuum\_analyze\_scale\_factor*.

При этом число строк определяется приблизительно, по статистике:

- общее число — `pg_class.reltuples`,
- число ненужных — `pg_stat_all_tables.n_dead_tup`,
- число измененных — `pg_stat_all_tables.n_mod_since_analyze`.

Дальше процесс по очереди очищает и/или анализирует отобранные объекты и по окончании очистки завершается.

В одной БД может одновременно работать несколько процессов, параллельно обрабатывая разные таблицы. На уровне одной таблицы параллелизма нет.

## Настройки автоочистки

*autovacuum\_vacuum\_threshold* = 50

*autovacuum\_vacuum\_scale\_factor* = 0.2



## Параметры хранения таблиц

*autovacuum\_enabled*

*toast.autovacuum\_enabled*

*autovacuum\_vacuum\_threshold*

*toast.autovacuum\_vacuum\_threshold*

*autovacuum\_vacuum\_scale\_factor*

*toast.autovacuum\_vacuum\_scale\_factor*

Настройки по умолчанию предполагают вызов автоочистки при изменении таблицы на 20%. Это достаточно большое значение: автоочистка будет вызываться редко, но работать будет долго, особенно для больших таблиц.

В случае необходимости, можно настроить эти параметры на уровне отдельных таблиц с помощью параметров хранения (`create table ... with (параметр=значение)`). Причем параметры можно отдельно настраивать для toast-таблиц.

Кроме того, на уровне отдельных таблиц автоочистку можно отключить.

<https://postgrespro.ru/docs/postgresql/10/runtime-config-autovacuum>

<https://postgrespro.ru/docs/postgresql/10/sql-createtable>

## Настройки автоанализа

*autovacuum\_analyze\_threshold* = 50

*autovacuum\_analyze\_scale\_factor* = 0.1



## Параметры хранения таблиц

*autovacuum\_analyze\_threshold*

*autovacuum\_analyze\_scale\_factor*

для TOAST-таблиц анализ не выполняется

Настройки по умолчанию предполагают вызов автоанализа при изменении таблицы на 10%.

В случае необходимости, можно настроить эти параметры на уровне отдельных таблиц с помощью параметров хранения.

Toast-таблицы не анализируются, поэтому соответствующих параметров для них нет.

## Настройки для регулирования нагрузки

*autovacuum\_vacuum\_cost\_limit* = -1  
(при -1 используется *vacuum\_cost\_limit* = 200)

общий предел  
для всех рабочих  
процессов

*autovacuum\_vacuum\_cost\_delay* = 20 ms  
(при -1 используется *vacuum\_cost\_delay* = 0)

*vacuum\_cost\_page\_hit*, *vacuum\_cost\_page\_miss*, *vacuum\_cost\_page\_dirty*

## Параметры хранения таблиц

*autovacuum\_vacuum\_cost\_limit*  
*toast.autovacuum\_vacuum\_cost\_limit*

*autovacuum\_vacuum\_cost\_delay*  
*toast.autovacuum\_vacuum\_cost\_delay*

Регулирование нагрузки при автоматической очистке работает так же, как и при обычной. Однако существуют дополнительные параметры *autovacuum\_vacuum\_cost\_limit* и *autovacuum\_vacuum\_cost\_delay*, которые, если не равны -1, перекрывают параметры *vacuum\_cost\_limit* и *vacuum\_cost\_delay*.

Со значениями по умолчанию *autovacuum\_vacuum\_cost\_limit* получает значение 200, что довольно мало. Обычно имеет смысл установить значение в районе 1000-2000.

Кроме того следует учитывать, что предел, устанавливаемый этим параметром, общий для всех рабочих процессов. Поэтому при увеличении *autovacuum\_max\_workers* следует увеличивать и *autovacuum\_vacuum\_cost\_limit*.

Также эти параметры могут указываться и на уровне отдельных таблиц.

<https://postgrespro.ru/docs/postgresql/10/runtime-config-resource#RUNTIME-CONFIG-RESOURCE-VACUUM-COST>



## Настройки памяти

`autovacuum_work_mem = -1`  
(при `-1` используется `maintenance_work_mem = 64MB`)

настройки действуют для каждого рабочего процесса, память выделяется сразу в полном объеме

большой объем памяти уменьшает избыточную обработку индексных страниц

## Мониторинг

`log_autovacuum_min_duration`  
`pg_stat_progress_vacuum`

Кроме настроек, влияющих на то, когда и как работать процессу автоочистки, есть возможность отрегулировать выделяемую память для рабочих процессов (autovacuum worker).

По умолчанию размер памяти ограничен параметром `maintenance_work_mem`, который действует не только на автоочистку, но и на все остальные служебные фоновые процессы. Обычно этот параметр можно установить в достаточно большое значение, поскольку фоновых процессов не так много. Однако число рабочих процессов автоочистки (регулируемое параметром `autovacuum_max_workers`) может быть большим, а память выделяется сразу полностью (а не по необходимости). Поэтому для процессов автоочистки можно настроить отдельное ограничение с помощью параметра `autovacuum_work_mem`.

Как говорилось в теме «Очистка», процесс очистки может работать и с минимальным объемом памяти. Но если на таблице созданы индексы, то небольшое значение может привести к повторным сканированиям одних и тех же индексных страниц — автоочистка будет работать медленнее. В идеале следует подобрать такое минимальное значение, при котором нет повторных сканирований.

Для мониторинга есть параметр `log_autovacuum_min_duration`, который выводит информацию об очистке в журнал сообщений сервера.

Напомним, что зачастую следует не увеличивать размер памяти, а уменьшать порог срабатывания очистки, чтобы за один раз обрабатывалось меньше данных.

## Баланс между разрастанием и накладными расходами

итеративный процесс

### Основные параметры

*autovacuum\_vacuum\_scale\_factor* — частота обработки

*autovacuum\_max\_workers* — параллелизм

*autovacuum\_vacuum\_cost\_limit* — скорость работы

индивидуальная настройка важных таблиц параметрами хранения

### Мониторинг

разрастание таблиц

очередь таблиц, ожидающих очистки

нагрузка на диски при работе очистки

Автоочистка управляется довольно большим числом взаимосвязанных параметров. В хорошо настроенной системе автоочистка не дает таблицам неадекватно разрастаться, и при этом не создает лишних накладных расходов.

Поиск баланса — итеративный процесс. Нужно выставить разумные начальные значения (исходя из размера и характера использования таблиц), проводить постоянный мониторинг и вносить коррективы.

Увеличение *threshold/scale\_factor* приводит к большему разрастанию таблиц, но очистка выполняется реже и большими порциями (возможны пиковые нагрузки), суммарные накладные расходы ниже. Уменьшение параметров приводит к более частой обработке небольшими порциями, накладные расходы в этом случае выше, а таблицы разрастаются меньше.

Пиковые нагрузки можно попытаться сгладить параметрами *cost\_limit/cost\_delay*, уменьшающими скорость работы очистки.

Значение, заданное параметрами *threshold/scale\_factor*, определяет лишь желаемый момент срабатывания очистки. При большом числе сильно измененных таблиц процесс очистки может долго выполнять итерацию и приступит к обработке очередной таблицы далеко не сразу. В этом случае надо либо увеличивать скорость работы очистки (если она была уменьшена параметрами *cost\_limit/cost\_delay*), либо увеличивать число параллельно работающих процессов параметром *max\_workers* (что приведет к увеличению накладных расходов).



Автоматическая очистка запускает очистку и анализ таблиц, динамически реагируя на изменения данных

Автоочистка допускает тонкую настройку на уровне как системы, так и отдельных таблиц

Настройка автоочистки — итеративный поиск баланса

1. Настройте автоочистку на запуск при изменении 10 % строк, время «сна» — одна секунда.
2. Создайте таблицу с большим количеством строк.
3. Двадцать раз с интервалом в несколько секунд изменяйте по 5–6 % случайных строк. Каждое изменение выполняйте в отдельной транзакции.
4. Сколько раз отработала автоочистка? На сколько разрослась таблица? Совпадают ли результаты с ожидаемыми и как их объяснить?