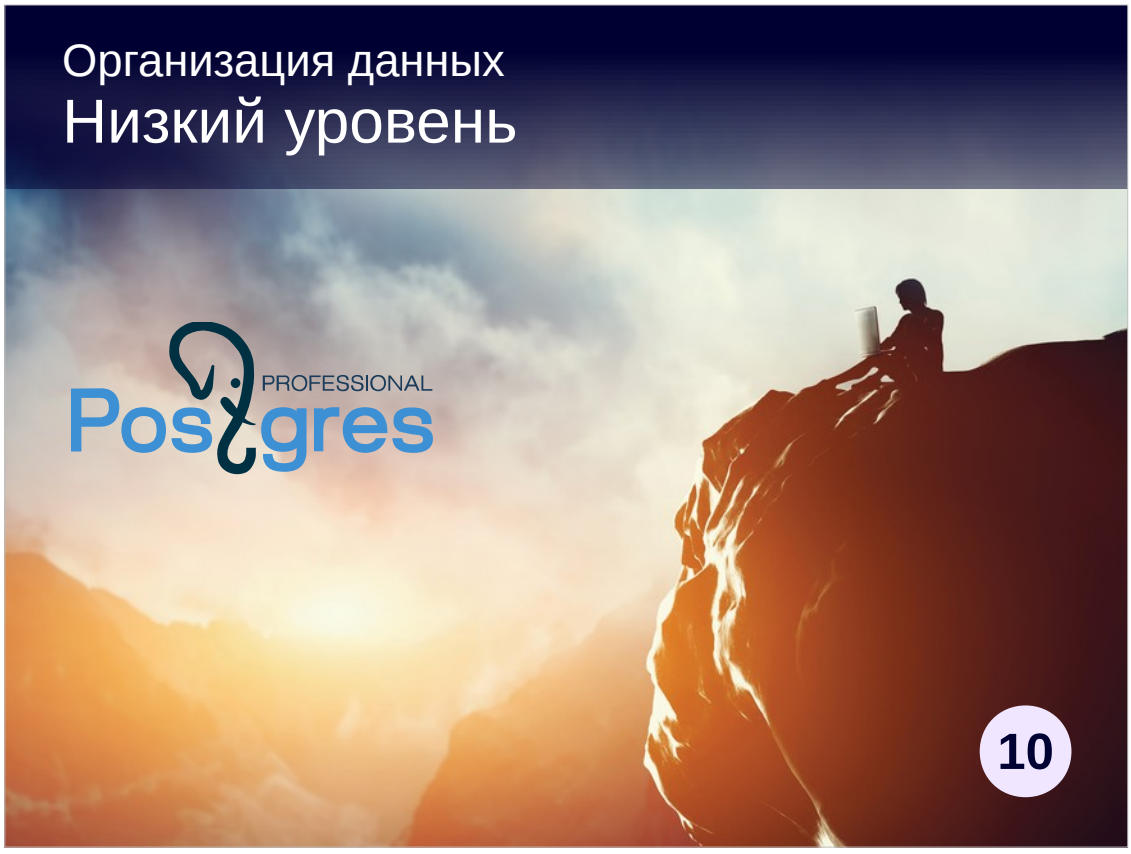


Авторы: Егор Рогов, Павел Лузанов

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

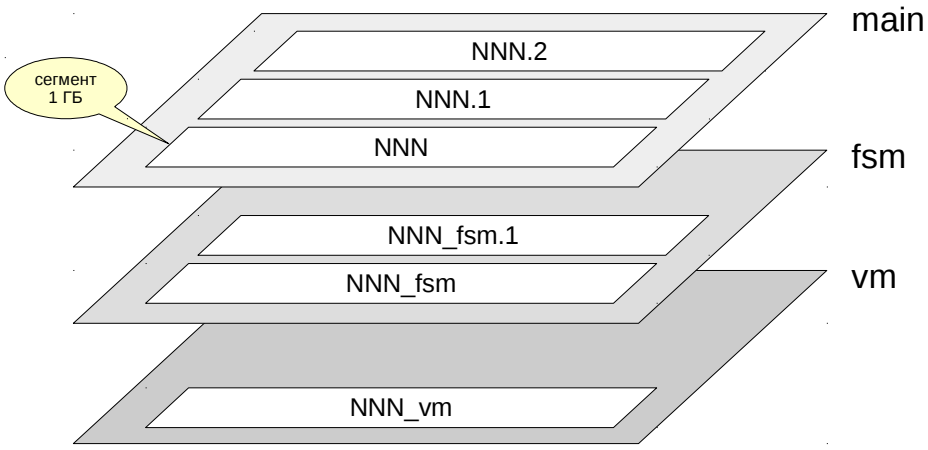
Темы

Файлы и страницы

Слои: данные, карты видимости и свободного пространства

Длинные строки и TOAST

Слои объекта



Обычно каждому объекту БД, хранящему данные (таблице, индексу, последовательности, материализованному представлению), соответствует несколько *слоев* (forks). Каждый слой содержит определенный вид данных.

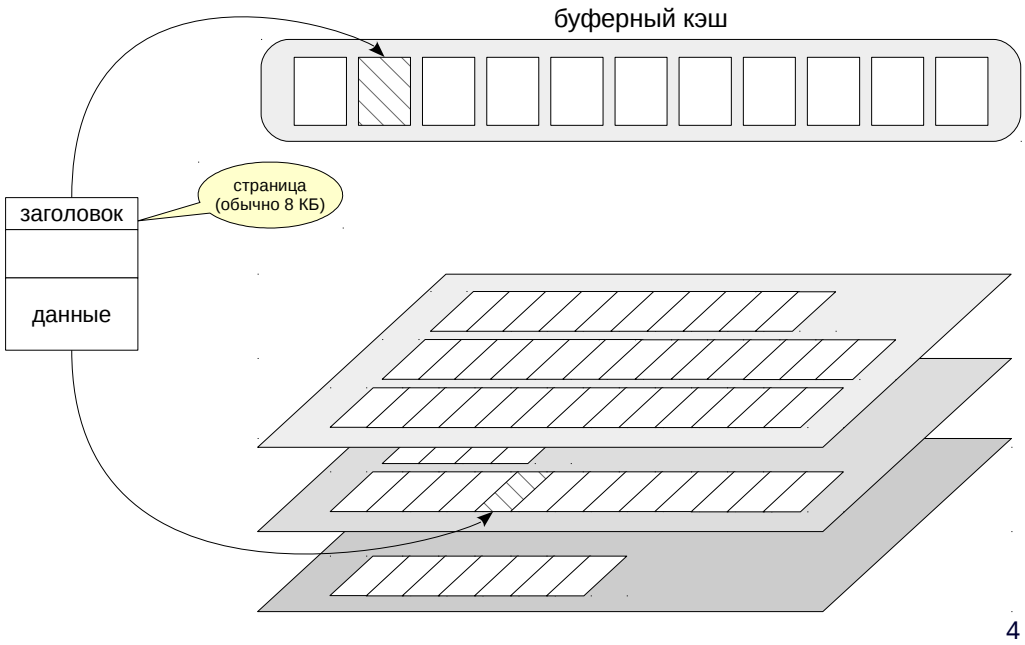
Вначале слой содержит один-единственный файл. Имя файла состоит из числового идентификатора, к которому может быть добавлено окончание, соответствующее имени слоя.

Файл постепенно растет и, когда его размер доходит до 1 ГБ, создается следующий файл этого же слоя. Такие файлы иногда называют *сегментами.* Порядковый номер сегмента добавляется в конец имени файла.

Ограничение размера файла в 1 ГБ возникло исторически для поддержки различных файловых систем, некоторые из которых не умеют работать с файлами большого размера.

Таким образом, одному объекту БД на диске может соответствовать несколько файлов. Для небольшой таблицы их будет 3, для индекса — два. Все файлы объектов, принадлежащих одному табличному пространству и одной БД, будут помещены в один каталог. Это необходимо учитывать, потому что файловые системы могут не очень хорошо работать с большим количеством файлов в каталоге.

Страницы файлов



Файлы, в свою очередь, разделены на *страницы* (иногда используется термин *блок*). Страница обычно имеет размер 8 КБ. Его в некоторых пределах можно поменять (16 КБ или 32 КБ), но только при сборке. Собранный и запущенный кластер может работать со страницами только одного размера.

Независимо от того, к какому слою принадлежат файлы, они используются буферным менеджером примерно одинаково. Страницы сначала читаются в буферный кэш, там их могут читать и изменять процессы PostgreSQL, затем при необходимости страницы вытесняются обратно на диск.

Каждая страница имеет внутреннюю разметку. Она содержит заголовок и полезные данные; между ними может находиться свободное пространство, если страница занята не полностью.

Слои

Основной

- собственно данные (версии строк)
- существует для всех объектов

Инициализации (init)

- заготовка пустого основного слоя
- используется при сбое; только для нежурналируемых таблиц

Посмотрим теперь на типы слоев.

Основной слой — это собственно данные: версии строк таблиц или строки индексов. Имена файлов основного слоя состоят только из идентификатора без дополнительного окончания.

Основной слой существует для любых объектов.

Слой инициализации имеет окончание «_init». Этот слой существует только для нежурналируемых таблиц (созданных с указанием UNLOGGED) и их индексов. Такие объекты ничем не отличаются от обычных, кроме того, что действия с ними не записываются в журнал упреждающей записи. За счет этого работа с ними происходит быстрее, но в случае сбоя их содержимое невозможно восстановить. Поэтому при восстановлении PostgreSQL просто удаляет все слои таких объектов и записывает слой инициализации на место основного слоя. В результате получается пустая таблица.

Слои

Карта свободного пространства (fsm)

- отмечает свободное пространство в страницах после очистки
- используется при вставке новых версий строк
- существует для всех объектов

Слой fsm (free space map) — карта свободного пространства.

В ней отмечено наличие пустого места внутри страниц. Это место постоянно меняется: при добавлении новых версий строк уменьшается, при очистке — увеличивается.

Карта свободного пространства используется при вставке новых версий строк, чтобы быстро найти подходящую страницу, на которую поместятся добавляемые данные.

https://postgrespro.ru/docs/postgresql/10/storage-fsm

Слои

Карта видимости (vm)

- отмечает страницы, на которых все версии строк видны во всех снимках
- используется для оптимизации работы процесса очистки и ускорения индексного доступа
- существует только для таблиц

Слой vm (visibility map) — битовая карта видимости. В ней отмечены страницы, которые содержат только актуальные версии строк, видимые во всех снимках данных. Иными словами, это страницы, которые давно не изменялись и успели полностью очиститься от неактуальных версий.

Карта видимости применяется для оптимизации очистки (отмеченные страницы не нуждаются в очистке) и для ускорения индексного доступа. Дело в том, что информация о версионности хранится только для таблиц, но не для индексов (поэтому у индексов не бывает карты видимости). Получив из индекса ссылку на версию строки, нужно прочитать табличную страницу, чтобы проверить ее видимость. Но если в самом индексе уже есть все нужные столбцы, и при этом страница отмечена в карте видимости, то к таблице можно не обращаться.

Начиная с версии 9.6 в этом же слое хранится и так называемая *карта заморозки.* Про нее речь идет в модуле «Задачи администрирования», тема «Сопровождение».

https://postgrespro.ru/docs/postgresql/10/storage-vm

TOAST

Версия строки должна помещаться на одну страницу

- можно сжать часть атрибутов,
- или вынести в отдельную TOAST-таблицу,
- или сжать и вынести одновременно

TOAST-таблица

- схемы pg_toast, pg_toast_temp_*N*
- таблица поддержана собственным индексом
- «длинные» атрибуты разделены на части размером меньше страницы
- читается только при обращении к «длинному» атрибуту
- собственная версионность
- работает прозрачно для приложения

Любая версия строки в PostgreSQL должна целиком помещаться на одну страницу. Для «длинных» версий строк применяется технология TOAST — The Oversized Attributes Storage Technique. Она подразумевает несколько стратегий. Подходящий «длинный» атрибут может быть сжат так, чтобы версия строки поместилась на страницу. Если это не получается, версия строки может быть отправлена в отдельную служебную таблицу. Могут применяться и оба подхода.

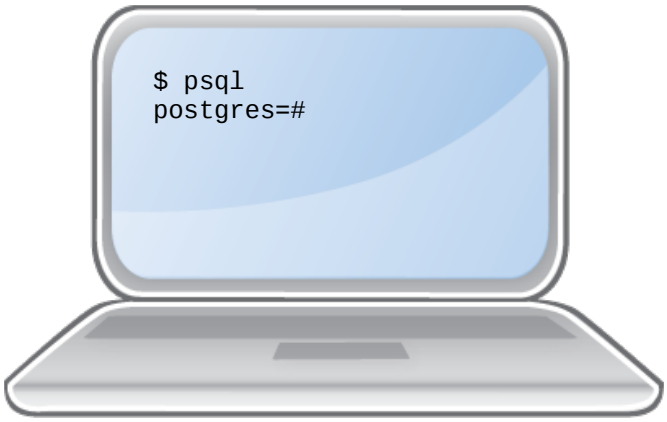
Для каждой основной таблицы при необходимости создается отдельная TOAST-таблица (и к ней специальный индекс). Такие таблицы и индексы располагаются в отдельной схеме pg_toast и поэтому обычно не видны (для временных таблиц используется схема pg_toast_temp_*N* аналогично обычной pg_temp_*N*).

Версии строк в TOAST-таблице тоже должны помещаться на одну страницу, поэтому «длинные» значения хранятся порезанными на части. Из этих частей PostgreSQL прозрачно для приложения «склеивает» необходимое значение.

TOAST-таблица используется только при обращении к «длинному» значению. Кроме того, для toast-таблицы поддерживается своя версионность: если обновление данных не затрагивает «длинное» значение, новая версия строки будет ссылаться на то же самое значение в TOAST-таблице — это экономит место.

https://postgrespro.ru/docs/postgresql/10/storage-toast

Демонстрация



Итоги

Объект представлен несколькими слоями

Слой состоит из одного или нескольких файлов-сегментов

Каждый файл разбит на страницы

Для «длинных» версий строк используется TOAST

Практика

1. Создайте нежурналируемую таблицу в пользовательском табличном пространстве и убедитесь, что для таблицы существует слой init.
2. Удалите созданное табличное пространство.
3. Создайте таблицу со столбцом типа text.
   Какая стратегия хранения применяется для этого столбца?
4. Измените стратегию на external и вставьте в таблицу короткую и длинную строки.
5. Проверьте, попали ли строки в toast-таблицу, выполнив прямой запрос к ней. Объясните, почему.