Сопровождение PostgreSQL

Темы

Процедура очистки и задачи, которые она решает

Мониторинг индексов и их перестроение

Задачи сопровождения

Обязательные периодические задачи

- очистка страниц от старых версий строк
- обновление статистики
- обновление карт видимости и свободного пространства
- избежание переполнения номера транзакции

} vacuum

- мониторинг индексов
- резервное копирование

Выполняются фоновыми процессами или вручную (скриптами)

Очистка

Фоновый процесс autovacuum

- выполняется параллельно с другими транзакциями,
но создает дополнительную нагрузку на подсистему ввода-вывода
- частота запуска динамически меняется в зависимости от частоты изменений
- работает постранично и не уменьшает размер файлов
- настраивается конфигурационными параметрами

Очистка

VACUUM

`# vacuum` *таблица*	очистка таблицы
`# vacuum`	очистка всей БД
`$ vacuumdb`	обертка для использования в ОС

позволяет выполнять очистку по желанию

можно отключить фоновый процесс параметром autovacuum и выполнять очистку в часы низкой нагрузки

при высокой частоте обновлений размер объектов может успеть существенно вырасти

Очистка

VACUUM FULL

`# vacuum full` *таблица*	очистка таблицы
`# vacuum full`	очистка всей БД
`$ vacuumdb --full`	обертка для использования в ОС

полностью перезаписывает содержимое таблиц и индексов, уменьшая размер файлов и минимизируя место

требует эксклюзивной блокировки

полезен при полном или существенном изменении таблиц

альтернативой может оказаться TRUNCATE вместо DELETE

Обновление статистики

Фоновый процесс autovacuum

автоматически обновляет статистику при значительных изменениях

ANALYZE

`# analyze [`*`таблица`*`]`	обновление статистики (анализ)
`$ vacuumdb --analyze-only`	обертка

обновляет статистику по случайной выборке данных

размер выборки настраивается

VACUUM

`# vacuum analyze [`*`таблица`*`]`	очистка и анализ
`$ vacuumdb --analyze`	обертка

помимо очистки также обновляет статистику

Обновление карт

Карта видимости (vm)

- битовая карта страниц с видимыми всем версиями строк (отмечает «давно не менявшиеся» страницы)
- признак видимости сбрасывается при любом изменении, а устанавливается автоматически в процессе очистки
- используется для оптимизации доступа

Карта свободного пространства (fsm)

- структура, отмечающая не полностью заполненные страницы
- обновляется в том числе при очистке
- используется для поиска страниц при вставке новых строк

Переполнение

Номер транзакции упорядочивает события

- для MVCC важно не точное время, а порядок транзакций
- транзакция началась позже = номер транзакции больше
- под номер отведено 32 бита
- из-за возможности переполнения номера закольцованы: для каждой транзакции половина номеров — «до» и половина — «после»

Фоновый процесс autovacuum

- время от времени помечает (замораживает) «достаточно старые» транзакции, чтобы их номер можно было безопасно переиспользовать
- заморозка выполняется, даже если autovacuum выключен
- если дошли по кругу до не замороженной транзакции, сервер принудительно останавливается

Мониторинг индексов

«Лишние» индексы

не используемые (мониторинг pg_stat_all_indexes.idx_scan)
дублирующиеся или пересекающиеся (запросы к pg_index)

проблемы: накладные расходы на изменения, место на диске

решение: удаление лишних индексов (аккуратное)

Разрастание индексов

мониторинг pg_relation_size()

проблемы: место на диске, уменьшение эффективности

решение: перестроение индексов

Перестроение индексов

REINDEX

`# reindex index` *индекс*	перестроить индекс
`# reindex table` *таблица*	перестроить все индексы таблицы
`# reindex database` *база*	перестроить все индексы БД
`# reindex system`	перестроить все индексы

перестраивает индекс, минимизируя занимаемое место

устанавливает эксклюзивную блокировку

VACUUM FULL

вместе с таблицей перестраивает и индексы

также устанавливает эксклюзивную блокировку

Перестроение индексов

Пересоздание без эксклюзивной блокировки

```
# create index new on ... concurrently;
# drop index old;
```

допускается создание нескольких индексов по одним и тем же полям

не все типы индексов поддерживают неблокирующее создание

неблокирующее создание не транзакционно

неблокирующее создание может завершиться неудачно

Индекс с ограничением целостности

```
# create unique index new on ... concurrently;
# alter table ...
  drop constraint old,
  add constraint new [unique|primary key] using index new;
```

Итоги

Рассмотрели процесс очистки и задачи, которые он решает:

- удаление старых версий строк
- сбор статистики
- обновление карт видимости и свободного пространства
- борьба с переполнением номера транзакции

Разобрались с мониторингом и пересозданием индексов

Практика

1. Отключить процесс autovacuum и убедиться, что он не работает.
2. В базе данных DB15 создать таблицу с одним числовым столбцом и индекс по этой таблице.
3. Вставить в таблицу 100 000 случайных чисел.
4. Несколько раз изменить половину строк таблицы, контролируя на каждом шаге размер таблицы и индекса
5. Выполнить vacuum full.
6. Повторить пункт 4, вызывая после каждого изменения vacuum.
7. Включить autovacuum.

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

Темы

Процедура очистки и задачи, которые она решает

Мониторинг индексов и их перестроение

Задачи сопровождения

Обязательные периодические задачи

- очистка страниц от старых версий строк
- обновление статистики
- обновление карт видимости и свободного пространства
- избежание переполнения номера транзакции

} vacuum

- мониторинг индексов
- резервное копирование

Выполняются фоновыми процессами или вручную (скриптами)

Сопровождение СУБД требует обязательного периодического выполнения ряда задач. Часть задач решается фоновыми процессами PostgreSQL, часть может выполняться вручную или скриптами, поставленными на расписание.

Задачи очистки страниц от старый версий строк, обновления карт видимости и свободного пространства, избежания циклического переполнения номера транзакции решаются процессом очистки (vacuum).

http://www.postgresql.org/docs/current/static/routine-vacuuming.html

Обновление статистики может выполняться как процессом очистки, так и отдельно.

Индексы требует определенного присмотра и иногда может потребоваться их перестроение.

http://www.postgresql.org/docs/current/static/routine-reindex.html

Резервное копирование — достаточно большая самостоятельная тема и разбирается отдельно.

Очистка

Фоновый процесс autovacuum

- выполняется параллельно с другими транзакциями, но создает дополнительную нагрузку на подсистему ввода-вывода
- частота запуска динамически меняется в зависимости от частоты изменений
- работает постранично и не уменьшает размер файлов
- настраивается конфигурационными параметрами

Необходимость очистки вызвана тем, что при изменении и удалении строк в страницах сохраняются их старые версии, как того требует механизм многоверсионности (см. тему «Архитектура»).

Существует несколько способов очистки страниц от версий строк, которые больше не требуются ни для одного снимка.

Основной способ — фоновый процесс autovacuum. Он выполняется периодически с частотой, которая зависит от частоты изменений в системе, чтобы не допустить существенного разрастания файлов. Процесс работает параллельно с другими транзакциями, обрабатывая файлы постранично. Из-за этого он не минимизирует место, занимаемое объектами, и не уменьшает размер файлов (за исключением случая, когда несколько страниц в конце файла освобождаются полностью).

Поскольку процесс создает дополнительную нагрузку на систему, в частности, на ввод-вывод, он может быть настроен таким образом, чтобы оказывать меньшее влияние за счет более длительного суммарного времени выполнения.

http://www.postgresql.org/docs/current/static/runtime-config-resource.html

Очистка

VACUUM

`# vacuum` *таблица*	очистка таблицы
`# vacuum`	очистка всей БД
`$ vacuumdb`	обертка для использования в ОС

позволяет выполнять очистку по желанию

можно отключить фоновый процесс параметром autovacuum и выполнять очистку в часы низкой нагрузки

при высокой частоте обновлений размер объектов может успеть существенно вырасти

Ручной запуск vacuum при выключенном процессе autovacuum позволяет перенести очистку на те часы, когда нагрузка на систему минимальна (например, раз в сутки ночью). Однако при большом объеме изменений (в частности, внеплановом) файлы могут успеть существенно вырасти в размерах.

Ручной запуск может использоваться и вместе с autovacuum.

http://www.postgresql.org/docs/current/static/sql-vacuum.html

Очистка

VACUUM FULL

`# vacuum full` *таблица*	очистка таблицы
`# vacuum full`	очистка всей БД
`$ vacuumdb --full`	обертка для использования в ОС

полностью перезаписывает содержимое таблиц и индексов, уменьшая размер файлов и минимизируя место

требует эксклюзивной блокировки

полезен при полном или существенном изменении таблиц

альтернативой может оказаться TRUNCATE вместо DELETE

При хорошо настроенном процессе файлы данных вырастают на некоторую постоянную величину за счет обновлений между запусками очистки. Если файлы выросли больше, чем это представляется разумным (например, за счет массовых внеплановых изменений), то для освобождения места потребуется запуск vacuum full.

Эта программа полностью перезаписывает содержимое таблиц и индексов, минимизируя занимаемое место. Однако этот процесс требует эксклюзивной блокировки и поэтому не может выполняться параллельно с другими транзакциями.

Если речь идет об очистке таблиц после массовых удалений строк, то хорошей альтернативой является выполнение truncate вместо delete. Truncate сразу же освобождает все строки, не требуя дополнительной очистки. Но надо иметь в виду, что truncate хотя и является транзакционной командой (может быть отменен командой rollback), но ведет себя не так, как delete: он устанавливает эксклюзивную блокировку на таблицу. Таким образом, truncate начнет выполняться только после завершения всех транзакций, использующих таблицу (хотя бы и для чтения), и пока транзакция с truncate не завершится, любой доступ к таблице будет блокирован (хотя бы и на чтение).

Если продолжительная эксклюзивная блокировка нежелательна, можно рассмотреть стороннее расширение pg_repack, позволяющее выполнить перестроение таблицы без нее.

Обновление статистики

Фоновый процесс autovacuum

автоматически обновляет статистику при значительных изменениях

ANALYZE

`# analyze [`*`таблица`*`]`	обновление статистики (анализ)
`$ vacuumdb --analyze-only`	обертка

обновляет статистику по случайной выборке данных

размер выборки настраивается

VACUUM

`# vacuum analyze [`*`таблица`*`]`	очистка и анализ
`$ vacuumdb --analyze`	обертка

помимо очистки также обновляет статистику

Точная и актуальная статистика жизненно необходима для правильной работы оптимизатора.

Фоновый процесс autovacuum, помимо очистки, выполняет сбор и обновление статистики по изменяющимся таблицам.

Если этого недостаточно, статистику можно обновить в ручном режиме с помощью отдельной команды (analyze) или вместе с очисткой (vacuum analyze).

При сборе статистики читается случайная выборка данных. Это позволяет быстро собирать статистику даже по большим таблицам. Результат получается не точный, но в большинстве случаев в этом нет ничего страшного. При необходимости размер выборки можно увеличить.

http://www.postgresql.org/docs/current/static/sql-analyze.html

Обновление карт

Карта видимости (vm)

- битовая карта страниц с видимыми всем версиями строк (отмечает «давно не менявшиеся» страницы)
- признак видимости сбрасывается при любом изменении, а устанавливается автоматически в процессе очистки
- используется для оптимизации доступа

Карта свободного пространства (fsm)

- структура, отмечающая не полностью заполненные страницы
- обновляется в том числе при очистке
- используется для поиска страниц при вставке новых строк

Карта видимости отмечает страницы, которые содержат только такие версии строк, которые видны всем транзакциям. Иными словами, в странице не должно быть двух версий одной и той же строки. Фактически, это страницы, которые не изменялись достаточно давно.

Карта видимости хранится в отдельном (небольшом, по сравнению с размером данных) файле vm.

Карта используется для оптимизации доступа. В частности, процессу очистки нет необходимости просматривать отмеченные страницы, так как они давно не менялись и в них нечего очищать.

При всяком изменении данных в странице признак видимости этой страницы сбрасывается. Установка признака происходит автоматически при любом из вариантов очистки.

http://www.postgresql.org/docs/current/static/storage-vm.html

Карта свободного пространства отмечает страницы, не полностью заполненные данными.

Эта карта хранится в отдельном файле fsm.

Карта используется для поиска страницы при вставке (и обновлении, так как обновление трактуется как удаление и вставка) новых строк.

Карта свободного пространства обновляется, в том числе, процессом очистки.

http://www.postgresql.org/docs/current/static/storage-fsm.html

Переполнение

Номер транзакции упорядочивает события

- для MVCC важно не точное время, а порядок транзакций
- транзакция началась позже = номер транзакции больше
- под номер отведено 32 бита
- из-за возможности переполнения номера закольцованы: для каждой транзакции половина номеров — «до» и половина — «после»

Фоновый процесс autovacuum

- время от времени помечает (замораживает) «достаточно старые» транзакции, чтобы их номер можно было безопасно переиспользовать
- заморозка выполняется, даже если autovacuum выключен
- если дошли по кругу до не замороженной транзакции, сервер принудительно останавливается

PostgreSQL упорядочивает события с помощью номеров транзакций: из двух транзакций позже началась та, у которой номер больше. Такого порядка достаточно, так как для MVCC важно не точное время, а именно порядок транзакций.

Но под номер транзакции отведено 32 бита. В сильно нагруженной системе может произойти переполнение: при очередном увеличении номера он сбросится в ноль и упорядоченность нарушится. (Для этого система должна работать без перерыва с постоянной нагрузкой 1000 транзакций в секунду на протяжении полутора месяцев.)

Чтобы не допустить такой ситуации, пространство номеров рассматриваются в виде кольца: для любой транзакции половина номеров находится в прошлом, а другая половина — в будущем.

Однако дойдя по кольцу до очень старой транзакции, мы увидим ее в будущем, что приведет к нарушению целостности базы данных. Поэтому фоновый процесс autovacuum время от времени помечает транзакции, которые видны уже в каждом снимке, как «замороженные», причем это происходит, даже если autovacuum отключен. Замороженные транзакции всегда считаются принадлежащими прошлому, какой бы номер у них ни был.

Если же серверу встречается не замороженная транзакция, он принудительно останавливается. После этого администратор должен стартовать сервер и выполнить очистку, прежде чем можно будет продолжить использовать систему.

Мониторинг индексов

«Лишние» индексы

- не используемые (мониторинг pg_stat_all_indexes.idx_scan)
  дублирующиеся или пересекающиеся (запросы к pg_index)
- проблемы: накладные расходы на изменения, место на диске
- решение: удаление лишних индексов (аккуратное)

Разрастание индексов

- мониторинг pg_relation_size()
- проблемы: место на диске, уменьшение эффективности
- решение: перестроение индексов

Индексы — достаточно сложные структуры по сравнению с таблицами. При активном удалении и вставке строк, индексы могут необоснованно увеличиваться в объеме. Например, если речь идет о индексах b-tree, индексная страница при переполнении делится на две, которые уже никогда не объединяются, даже если все строки, на которые ссылались записи из этих страниц, удалены. Освободившееся место, безусловно, доступно для вставки новых записей, но при неудачном раскладе индекс тем не менее может «распухнуть».

Администратор должен присматривать за индексами и при необходимости перестраивать их.

http://www.postgresql.org/docs/current/static/sql-reindex.html

Перестроение индексов

REINDEX

`# reindex index` *индекс*	перестроить индекс
`# reindex table` *таблица*	перестроить все индексы таблицы
`# reindex database` *база*	перестроить все индексы БД
`# reindex system`	перестроить все индексы

перестраивает индекс, минимизируя занимаемое место

устанавливает эксклюзивную блокировку

VACUUM FULL

вместе с таблицей перестраивает и индексы

также устанавливает эксклюзивную блокировку

Перестраивание может выполняться командой reindex. Она заново пересоздает индекс и заменяет им старый. Она, однако, устанавливает на таблицу эксклюзивную блокировку, поэтому любая параллельная работа с таблицей будет приостановлена.

Индексы также перестраиваются при выполнении vacuum full, но тоже с эксклюзивной блокировкой.

http://www.postgresql.org/docs/current/static/sql-reindex.html

Перестроение индексов

Postgres PROFESSIONAL

Пересоздание без эксклюзивной блокировки

```
# create index new on ... concurrently;
# drop index old;
```

допускается создание нескольких индексов по одним и тем же полям

не все типы индексов поддерживают неблокирующее создание

неблокирующее создание не транзакционно

неблокирующее создание может завершиться неудачно

Индекс с ограничением целостности

```
# create unique index new on ... concurrently;
# alter table ...
  drop constraint old,
  add constraint new [unique|primary key] using index new;
```


Альтернативный способ — перестроение индекса вручную в несколько шагов.

Сначала создается новый индекс в неблокирующем режиме (ключевое слово concurrently), который позволяет вести параллельную работу с таблицей. Однако неблокирующее создание может завершиться неудачно (из-за взаимоблокировок) — в таком случае индекс потребуется удалить и попробовать создать еще раз. Кроме того, такая операция не может выполняться внутри транзакции.

Затем старый индекс удаляется. Либо, если индекс участвует в ограничении целостности, старое ограничение удаляется (при этом удаляется и старый индекс) и создается новое.

http://www.postgresql.org/docs/current/static/sql-createindex.html

http://www.postgresql.org/docs/current/static/sql-altertable.html

Итоги

Рассмотрели процесс очистки и задачи, которые он решает:

- удаление старых версий строк
- сбор статистики
- обновление карт видимости и свободного пространства
- борьба с переполнением номера транзакции

Разобрались с мониторингом и пересозданием индексов

Практика

1. Отключить процесс autovacuum и убедиться, что он не работает.
2. В базе данных DB15 создать таблицу с одним числовым столбцом и индекс по этой таблице.
3. Вставить в таблицу 100 000 случайных чисел.
4. Несколько раз изменить половину строк таблицы, контролируя на каждом шаге размер таблицы и индекса
5. Выполнить vacuum full.
6. Повторить пункт 4, вызывая после каждого изменения vacuum.
7. Включить autovacuum.


```
# alter system set autovacuum = off;
$ pg_ctl reload
$ ps -o pid,command --ppid `head -n 1 $PGDATA/postmaster.pid`
# show autovacuum

# create database db15;
# \c db15
# create table t(n numeric);
# create index t_n on t(n);

# insert into t select random() from generate_series(1,100000);

# select pg_size_pretty(pg_table_size('t')),
  pg_size_pretty(pg_indexes_size('t'));
# update t set n=n where n < 0.5;
-- повторить (размер постоянно растет)


# vacuum full t;
-- размер таблицы вернулся к начальному, индекс стал компактнее

# select pg_size_pretty(pg_table_size('t')),
  pg_size_pretty(pg_indexes_size('t'));
# update t set n=n where n < 0.5;
# vacuum t;
-- повторить (размер увеличился один раз и стабилизировался)

# alter system set autovacuum = on;
$ pg_ctl reload
```