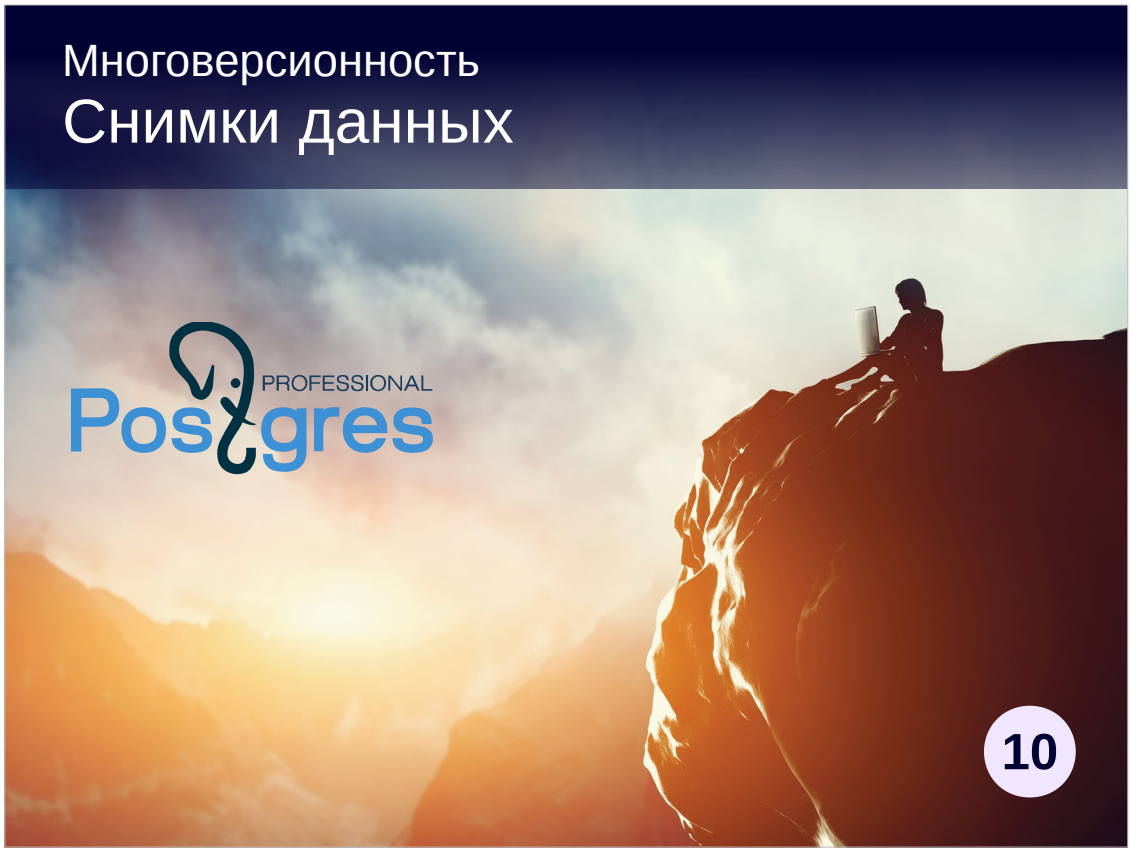


Авторы: Егор Рогов, Павел Лузанов

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

Темы

Видимость версий строк

Снимок данных

«Горизонт событий»

Экспорт снимка

Снимок

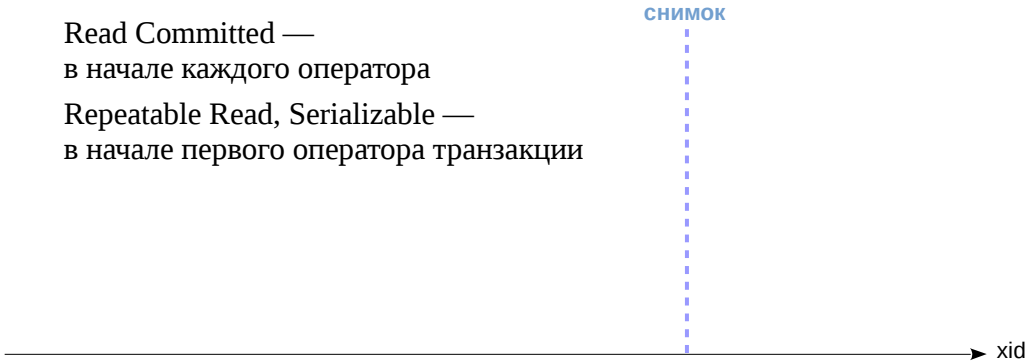
Дает согласованную картину данных на момент времени

неформально: видны только зафиксированные на этот момент данные

Создается:

Read Committed —
в начале каждого оператора

Repeatable Read, Serializable —
в начале первого оператора транзакции



В теме «Страницы и версии строк» мы видели. что в страницах данных физически могут находиться несколько версий одной и той же строки.

Транзакции работают со *снимком данных,* который определяет, какие версии должны быть видны, а какие — нет, чтобы обеспечить согласованную картину данных на определенный момент времени (на момент создания снимка — показан на рисунке синим цветом).

Согласованность здесь понимается в обычном ACID-смысле: данные должны быть корректны, что, по сути, означает, что видеть нужно только те данные, которые были зафиксированы на момент создания снимка.

На уровне изоляции Read Committed снимок создается в начале каждого оператора транзакции. Снимок активен, пока выполняется оператор.

На уровнях Repeatable Read и Serializable снимок создается один раз в начале первого оператора транзакции. Такой снимок остается активным до самого конца транзакции.

Видимость версий строк

Видимость версии строки ограничена xmin и xmax

Версия попадает в снимок, когда

- изменения транзакции xmin видны для снимка,
- изменения транзакции xmax не видны для снимка

Изменения транзакции видны, когда

- либо это та же самая транзакция, что создала снимок,
- либо она завершилась фиксацией до момента создания снимка

Отдельные правила для видимости собственных изменений

- учитывается порядковый номер операции в транзакции (cmin/cmax)

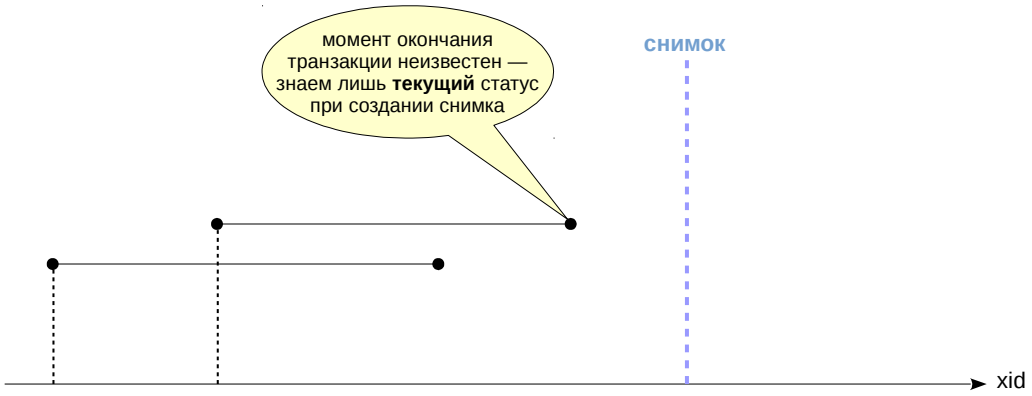
Будет или нет данная версия строки видна в снимке, определяется двумя полями ее заголовка — xmin и xmax, — то есть номерами создавшей и удалившей транзакций. Такие интервалы не пересекаются, поэтому одна строка представлена в любом снимке максимум одной своей версией.

Точные правила видимости довольно сложны и учитывают различные «крайние» случаи. Чуть упрощая, можно сказать, что версия строки видна, когда в снимке *видны* изменения, сделанные транзакцией xmin, и *не видны* изменения, сделанные транзакцией xmax.

В свою очередь, изменения транзакции видны в снимке, если либо это та же самая транзакция, что создала снимок (она сама видит свои же изменения), либо транзакция была зафиксирована до создания снимка.

Несколько усложняет картину случай определения видимости собственных изменений транзакции. Здесь может потребоваться видеть только часть таких изменений. Например, курсор, открытый в определенный момент, ни при каком уровне изоляции не должен увидеть изменений, сделанных после этого момента. Для этого в заголовке версии строки есть специальное поле (псевдостолбцы cmin и cmax), показывающее номер операции внутри транзакции, и этот номер тоже принимается во внимание.

Видимость версий строк



транзакции завершились (фиксацией) до создания снимка — изменения видны

Заметим, что нам известен момент начала транзакции (xid = xmin или xmax), но момент завершения транзакции нигде не записывается. Мы лишь можем узнать текущий статус транзакций при создании снимка.

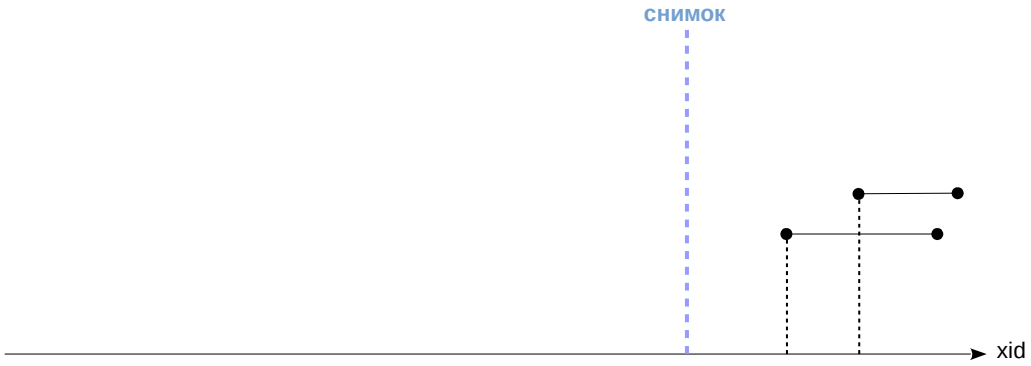
Рассмотрим несколько вариантов.

На рисунках синим цветом отмечен момент создания снимка, черные линии показывают транзакции. Точный момент завершения транзакции хоть и показан на рисунке, но фактически неизвестен; мы лишь знаем, что на момент создания снимка транзакция была завершена.

Если транзакция завершилась фиксацией (имеет статус committed в XACT), то ее изменения будут видны в снимке.

Если транзакция была прервана (имеет статус aborted в XACT), то ее изменения не будут видны в снимке.

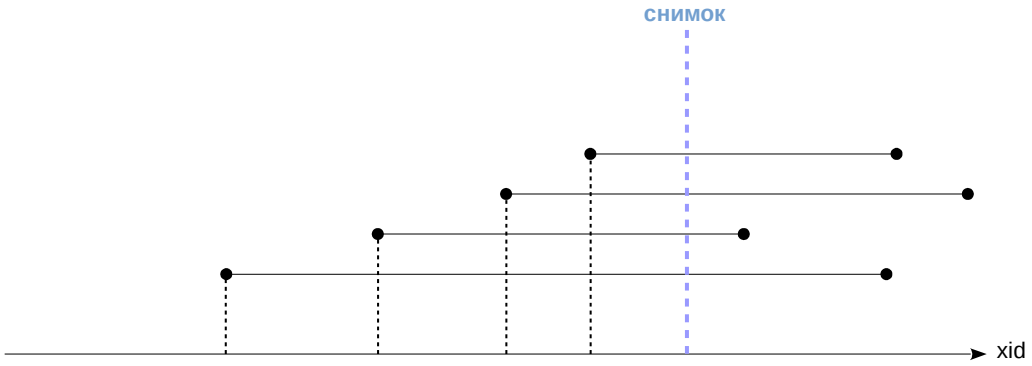
Видимость версий строк



транзакции начались после создания снимка — изменения не видны

Если транзакция началась после создания снимка, то ее изменения не будут видны в снимке.

Видимость версий строк



транзакции были активны в момент создания снимка — изменения не видны

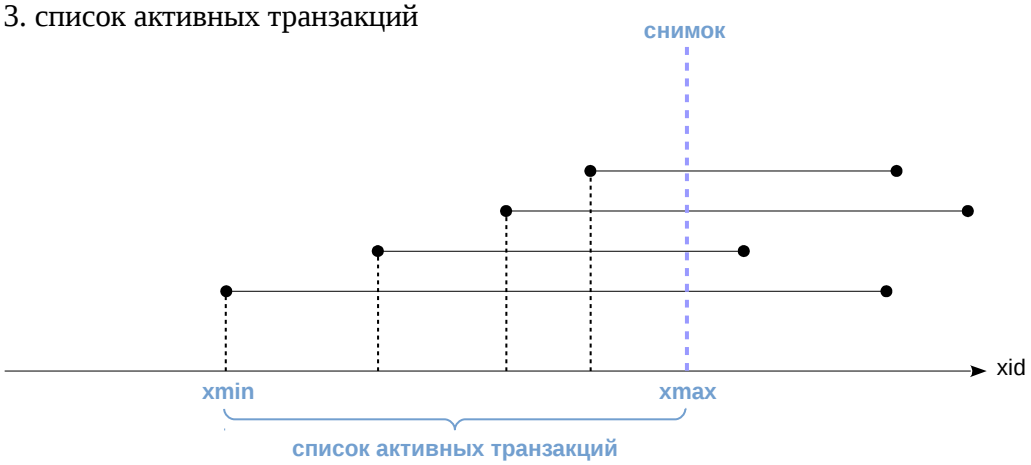
Наконец, остался случай транзакций, которые были начаты до создания снимка, но в момент создания снимка еще не завершились. Изменения таких транзакций также не должны быть видны.

Поскольку *после* создания снимка мы уже не сможем понять, была ли транзакций активна *в момент создания,* необходимо заранее запомнить список текущих активных транзакций. Как уже говорилось, для этого в общей памяти сервера поддерживается структура ProcArray, которая содержит список всех активных сеансов и их транзакций.

Из сказанного следует, что снимок можно создать только в текущий момент, поскольку историческая информация об активности транзакций нигде не хранится. Нельзя, например, создать снимок, показывающий согласованные данные по состоянию на пять минут назад, даже если все необходимые версии строк существуют в табличных страницах.

Снимок данных

1. номер самой ранней активной транзакции
2. номер следующей транзакции (еще не существующей)
3. список активных транзакций



8

Итак, снимок данных определяется несколькими параметрами:

- моментом создания снимка, а именно, номером следующей, еще не существующей в системе, транзакции (xmax снимка);

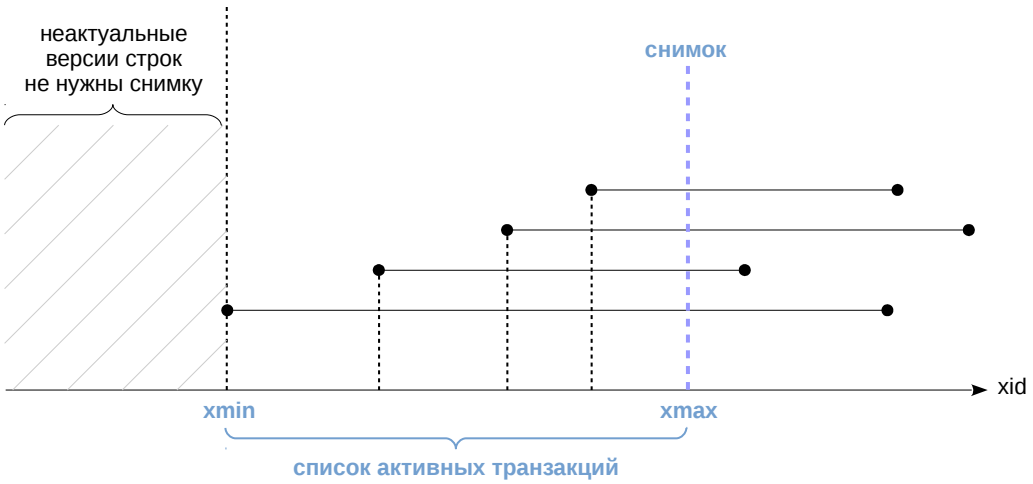
- списком активных транзакций на момент создания снимка.

Для удобства и оптимизации отдельно сохраняется и номер самой ранней из активных транзакций (xmin снимка).

Информацию о снимке можно получить с помощью функции txid_current_snapshot() и нескольких других, см. https://postgrespro.ru/docs/postgresql/10/functions-info

Подробнее это рассматривается в демонстрации.

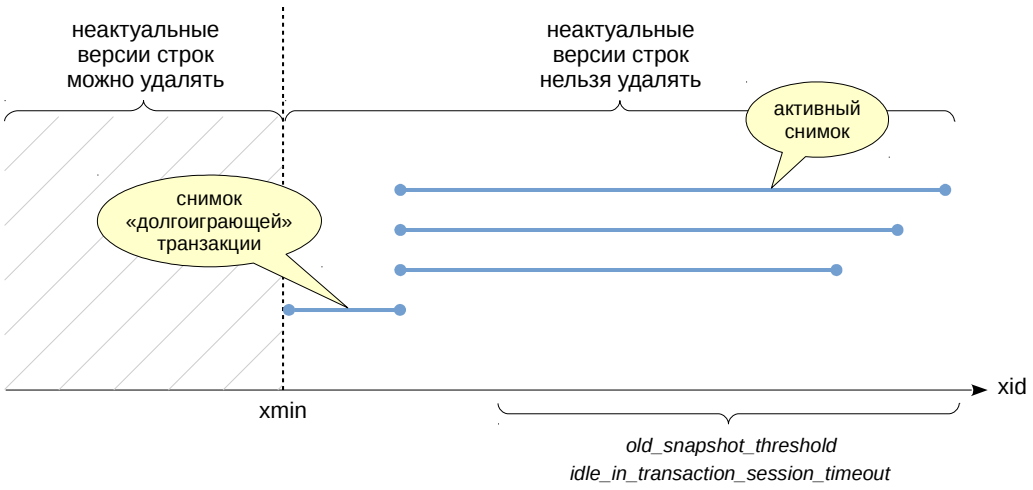
«Горизонт событий»



Номер самой ранней из активных транзакций (xmin снимка) имеет важный смысл — он определяет «горизонт событий» транзакции. А именно, за своим горизонтом транзакция всегда видит только актуальные версии строк.

Действительно, видеть неактуальную версию требуется только в том случае, когда актуальная создана еще не завершившейся транзакцией, и поэтому не видна. Но за «горизонтом» все транзакции уже гарантированно завершились.

«Горизонт событий» БД



10

Также можно определить и «горизонт событий» на уровне базы данных. Для этого надо взять все активные снимки и среди них найти минимальный xmin. Он и будет определять горизонт, за которым неактуальные версии строк в этой БД уже никогда не будут видны ни одной транзакции. Такие версии строк могут быть очищены.

Если какая-либо транзакция удерживает снимок в течении долгого времени, она тем самым удерживает и горизонт событий базы данных. Более того, незавершенная транзакция удерживает горизонт самим фактом своего существования, даже если в ней не удерживается снимок (на уровне Read Committed). Рисунок иллюстрирует такую ситуацию.

А это означает, что неактуальные версии строк в этой БД не смогут быть очищены. При этом «долгоиграющая» транзакция может никак не пересекаться по данным с другими транзакциями — это совершенно не важно, горизонт базы данных один на всех.

Если описанная ситуация действительно создает проблемы и нет способа избежать ее на уровне приложения, то, начиная с версии 9.6, доступны два параметра:

- *old_snapshot_threshold* определяет максимальное время жизни снимка. После этого времени сервер получает право удалять неактуальные версии строк, а если они понадобятся «долгоиграющей» транзакции, но она получит ошибку snapshot too old.

- *idle_in_transaction_session_timeout* определяет максимальное время жизни бездействующей транзакции. После этого времени транзакция прерывается.

Виртуальные транзакции

Только читающая транзакция никак не влияет на видимость

- обслуживающий процесс выделяет виртуальный номер
- виртуальный номер не учитывается в снимках
- виртуальный номер никогда не попадает в страницы
- «настоящий» номер выделяется при первом изменении данных

На практике PostgreSQL использует оптимизацию, позволяющую «экономить» номера транзакций.

Если транзакция только читает данные, то она никак не влияет на видимость версий строк. Поэтому вначале обслуживающий процесс выдает транзакции *виртуальный номер* (virtual xid). Номер состоит из идентификатора процесса и последовательного числа.

Выдача этого номера не требует синхронизации между всеми процессами и поэтому выполняется очень быстро. С другой причиной использования виртуальных номеров мы познакомимся в теме «Заморозка» этого модуля.

Виртуальные номера никак не учитываются в снимках данных.

В разные моменты времени в системе вполне могут оказаться виртуальные транзакции с номерами, которые уже использовались, и это нормально. Но такой номер нельзя записывать в страницы данных, потому что при следующем обращении к странице он может потерять всякий смысл.

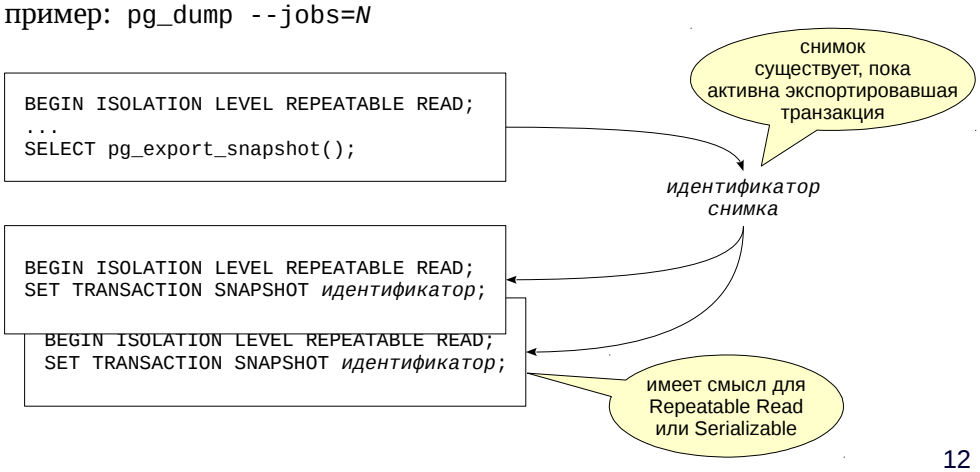
Если же транзакция начинает менять данные, ей выдается настоящий, уникальный номер транзакции.

Экспорт снимка

Задача

распределить работу между несколькими одновременно работающими транзакциями так, чтобы они видели одни и те же данные

пример: `pg_dump --jobs=N`



Бывают ситуации, когда несколько параллельных транзакций должны гарантированно видеть одну и ту же картину данных. Разумеется, нельзя полагаться на то, что картины совпадут просто из-за того, что транзакции запущены «одновременно». Для этого есть механизм экспорта и импорта снимка.

Функция pg_export_snapshot возвращает идентификатор снимка, который может быть передан (внешними по отношению к СУБД средствами) в другую транзакцию.

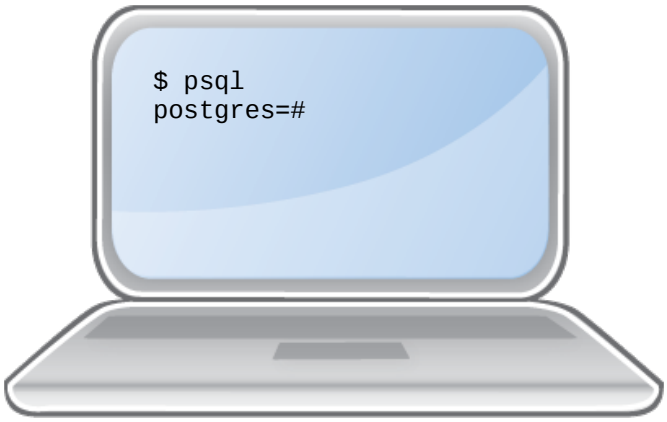
https://postgrespro.ru/docs/postgresql/10/functions-admin#FUNCTIONS-SNAPSHOT-SYNCHRONIZATION

Другая транзакция может импортировать снимок с помощью команды SET TRANSACTION SNAPSHOT до выполнения первого запроса в ней. Предварительно надо установить и уровень изоляции Repeatable Read или Serializable, потому что на уровне Read Committed операторы будут использовать собственные снимки.

Время жизни экспортированного снимка совпадает со временим жизни экспортирующей транзакции.

https://postgrespro.ru/docs/postgresql/10/sql-set-transaction

Демонстрация



Итоги

Снимок содержит информацию,
необходимую для определения видимости версий строк

Время создания снимка определяет уровень изоляции

Снимок определяет «горизонт событий», влияющий
на возможность удаления неактуальных версий

Практика

1. Воспроизведите ситуацию, при которой одна транзакция еще видит удаленную строку, а другая — уже нет.
   Посмотрите снимки данных этих транзакций и значения полей xmin и xmax удаленной строки.
   Объясните видимость на основании этих данных.
2. Если в запросе вызывается функция, содержащая другой запрос, какой снимок данных будет использоваться для «вложенного» запроса?
   Проверьте уровни изоляции Read Committed и Repeatable Read и категории изменчивости функций volatile и stable.

2. Можно воспользоваться следующим шаблоном функции, в предположении, что создана таблица t:

```
CREATE FUNCTION test() RETURNS integer AS $$
  SELECT count(*)::integer FROM t;
$$
VOLATILE -- или STABLE
LANGUAGE sql;
```

Также может пригодиться функция pg_sleep(n), вызывающая задержку на n секунд.