

## РЕФЕРАТ

Отчёт содержит 21 стр., 3 рис., 1 табл., 3 источн.

В данной работе рассматривается процесс настройки и управления репликацией в СУБД Tarantool, с акцентом на создание и конфигурацию репликасетов, подробно описывается процесс репликации и подключения узлов к репликасету. Работа также включает разработку дизайн-документа по фильтрации репликационного потока для улучшения работы Change Data Capture (CDC) в соответствии с требованиями заказчика.

## СОДЕРЖАНИЕ

РЕФЕРАТ . . . . .	3
ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ . . . . .	5
ВВЕДЕНИЕ . . . . .	7
ОСНОВНАЯ ЧАСТЬ . . . . .	8
1 Характеристика организации . . . . .	8
2 Репликация Tarantool . . . . .	9
2.1 Общая информация . . . . .	9
2.2 Настройка репликasetа . . . . .	10
2.3 Подключение узла к репликasetу . . . . .	12
3 Файловый JOIN . . . . .	14
4 Персистентный GC . . . . .	16
5 Фильтрация репликационного потока . . . . .	19
ЗАКЛЮЧЕНИЕ . . . . .	20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ . . . . .	21

## ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Анонимная реплика — это тип реплики, которая подключается к ведущему узлу для получения данных, но не участвует в кворуме репликации и не может стать мастером репликасета.

База данных (БД) — это организованная коллекция данных, которая структурирована таким образом, чтобы данные можно было легко хранить, управлять, изменять и извлекать.

Захват изменения данных (Change Data Capture, CDC) — это программное обеспечение для отслеживания и записи изменений, происходящих в данных базы данных. Оно позволяет фиксировать все вставки, обновления и удаления записей в режиме реального времени или с минимальной задержкой, что позволяет синхронизировать данные между различными системами, вести аудит изменений, и создавать системы резервного копирования или аналитики.

Кластер — это совокупность нескольких репликасетов, каждый из которых чаще всего хранит разный набор данных.

Локальный Спейс (local space) — это спейс, данные из которого не реплицируются и остаются локальными для конкретного узла.

Репликасет (replicaset) — это группа узлов (инстансов), работающих в режиме репликации и объединенных для обеспечения отказоустойчивости и доступности данных.

Система управления базами данных (СУБД) — это программное обеспечение, предназначенное для создания, управления и обеспечения доступа к базам данных. Оно позволяет пользователям определять, создавать, изменять и управлять базой данных, а также обеспечивает взаимодействие между пользователями и базой данных через запросы и команды. Основные функции включают хранение, поиск, обновление и удаление данных, а также обеспечение целостности, безопасности и управления доступом к данным.

Спейс (space) — это основная логическая единица хранения данных Tarantool, аналогичная таблице в традиционных реляционных базах данных. Спейс содержит набор записей, каждая из которых называется кортежем (tuple). Структура спейса определяется схемой, которая включает количество и типы полей в кортежах, а также индексы для быстрого доступа к данным.

LSN — это монотонно возрастающий идентификатор записи.

Vclock — это массив LSN, идентификаторами в котором являются ID узлов. Vclock представляет собой набор логических счетчиков для каждого узла в кластере, позволяя определить, какие изменения были применены на конкретном узле и какие еще предстоит синхронизировать.

## ВВЕДЕНИЕ

Место прохождения практики – общество с ограниченной ответственностью «ВК Цифровые Технологии», отдел Research & Development, команда Tarantool Platform Core. Период прохождения практики - с 1 июля 2024 года по 28 июля 2024 года.

В задачи прохождения практики входило:

- а) Разобраться в работе асинхронной репликации СУБД Tarantool;
- б) Изучить протокол подключения новых реплик к существующему репликасету Tarantool;
- в) Разработать дизайн-документ по фильтрации репликационного потока.

Основная цель прохождения практики заключалась в разработке дизайн-документа по фильтрации репликационного потока в СУБД Tarantool для улучшения работы CDC. От заказчика (команда разработки CDC) к новой функциональности были предъявлены следующие требования:

- а) Фильтрация репликационного потока по именам спейсов. Это необходимо для того, чтобы только выбранные спейсы передавались репликам, что обеспечивает более гибкое управление репликацией и снижает нагрузку на систему;
- б) Возможность продолжения подключения реплики к репликасету с места предыдущей остановки. Это позволяет уменьшить время восстановления работы реплик CDC после сбоев;
- в) Сохранение данных для анонимных реплик. Это необходимо, чтобы исключить ошибки репликации, связанные с удалением ненужных с точки зрения репликасета данных механизмом очистки (GC).

Каждое из вышеуказанных требований направлено на улучшение работы кластерной системы Tarantool, обеспечивая высокую отказоустойчивость, масштабируемость и эффективность обработки данных в распределенной среде. Разработанный документ является основой для последующей реализации предложенных решений в коде и их интеграции в существующую архитектуру Tarantool.

## ОСНОВНАЯ ЧАСТЬ

### 1 Характеристика организации

VK Цифровые Технологии – подразделение VK, развивающее продукты и сервисы для цифрового бизнеса. В основе экосистемы решений VK Цифровые технологии лежит многолетний опыт развития интернет-сервисов и технологий на базе открытого кода. VK Цифровые Технологии предоставляет готовые сервисы для решения бизнес задач любой сложности, занимается заказной разработкой и управлением ИТ-инфраструктурой на аутсорсе [1].

В портфеле VK цифровые Технологии — облачные сервисы VK Cloud Solutions, платформа in-memory вычислений Tarantool, платформа взаимодействия бизнеса и государства VK Tax Monitoring, а также линейка программных продуктов для управления персоналом, автоматизации производства и бизнес-процессов.

Tarantool как продукт появился 4 апреля 2016 года, когда Mail.ru Group (на данный момент известная как VK) сообщила о создании нового направления бизнеса, в рамках которого компания начала предоставлять корпоративным клиентам услуги в области хранения данных.

Изначально Tarantool применялся только в собственных проектах Mail.ru, в том числе в почтовом сервисе и облачном хранилище «Облако Mail.Ru». Затем компания превратила эту СУБД в продукт с открытым исходным кодом, который к началу апреля 2016 года внедрен рядом российских и международных компаний. В частности, Tarantool начал использоваться сервисом бесплатных объявлений Avito, социальной сетью знакомств Badoo и разработчиком систем информационной безопасности Wallarm.

На сегодняшний день Tarantool активно используется в банковской сфере (среди клиентов Tarantool можно выделить ВТБ, Альфа Банк, Банк Открытие и Газпромбанк) и для ретейла и e-commerce (Магнит, Wildberries, Ситилинк, X5Group) [2].

## 2 Репликация Tarantool

В данном разделе будет подробно рассмотрен механизм репликации в СУБД Tarantool. Будет описано, как Tarantool обеспечивает передачу данных между узлами в репликasetе, чтобы поддерживать актуальность и согласованность данных на всех репликах. Также будет рассмотрено, как осуществляется настройка репликасетов, выбор ведущего узла и подключение анонимных реплик. Особое внимание будет уделено протоколу подключения новых реплик в уже существующий репликасет.

### 2.1 Общая информация

В Tarantool группа узлов, которые работают с копиями одной и той же базы данных, объединяется в репликасет. Внутри репликасета каждому узлу назначается определенная роль: мастер или реплика. Только мастер может обрабатывать DDL, DML и DCL запросы, реплики же обрабатывают только DQL.

Репликация представляет собой процесс копирования данных с одного узла на другие узлы в репликasetе. В Tarantool репликация происходит не только с мастеров на реплики, но и с реплик на реплики. Репликация в базе данных необходима для решения следующих проблем:

- а) Обеспечение отказоустойчивости. Репликация позволяет создать несколько копий данных на разных узлах или серверах. В случае сбоя одного из узлов (например, из-за аппаратных проблем или аварийного отключения), другие узлы, содержащие реплики данных, могут продолжить обслуживание запросов, минимизируя простой системы и предотвращая потерю данных;
- б) Масштабирование чтения. За счет репликации можно распределять нагрузку на чтение между несколькими узлами. Это особенно полезно в системах, где количество операций чтения значительно превышает количество операций записи. Реплики могут обрабатывать запросы на чтение, разгружая ведущий узел, что улучшает общую производительность сервиса.

- в) Географическое распределение данных. В распределенных системах, где пользователи могут находиться в разных географических регионах, репликация позволяет разместить копии данных ближе к конечным пользователям. Это снижает задержки при доступе к данным и улучшает пользовательский опыт.

Реплика непрерывно получает обновления данных от мастера или от другой реплики, применяя журнал записей операций WAL. Каждая запись в журнале представляет собой отдельный запрос (DDL, DML, DCL) и имеет монотонно растущий номер (LSN). WAL журналы в Tarantool представлены xlog файлами. Tarantool хранит snapshot файл, который содержит полное состояние БД в конкретный момент времени, и xlog файлы с момента записи последнего снапшота. Репликация ведется только из xlog файлов.

Tarantool поддерживает два типа репликации:

- а) Синхронная репликация. Обеспечивает высокую степень консистентности данных, так как изменения записываются на мастере и подтверждаются всеми репликами, участвующими в кворуме, прежде чем клиент получает подтверждение выполнения операции. Этот тип репликации минимизирует риск потери данных, но может увеличивать задержки при записи из-за необходимости ожидания подтверждения от всех реплик.
- б) Асинхронная репликация. Изменения данных сначала записываются на мастере, а затем передаются на реплики без ожидания их подтверждения. Это позволяет снизить задержки при записи, но может привести к временной неконсистентности данных на разных узлах и потенциальной потере данных в случае сбоя мастера.

Так как CDC не заинтересован в синхронной репликации, мы не будем рассматривать ее подробно.

## **2.2 Настройка репликасета**

### **Шаг 1. Подготовка узлов**

Для настройки репликасета необходимо подготовить несколько узлов, на которых будет запущен Tarantool. Каждый узел должен быть настроен и доступен для связи с другими узлами в кластере. Необходимо убедиться, что на каждом узле установлена одна и та же версия Tarantool. В примере настройка репликасета производится локально, на одном компьютере.

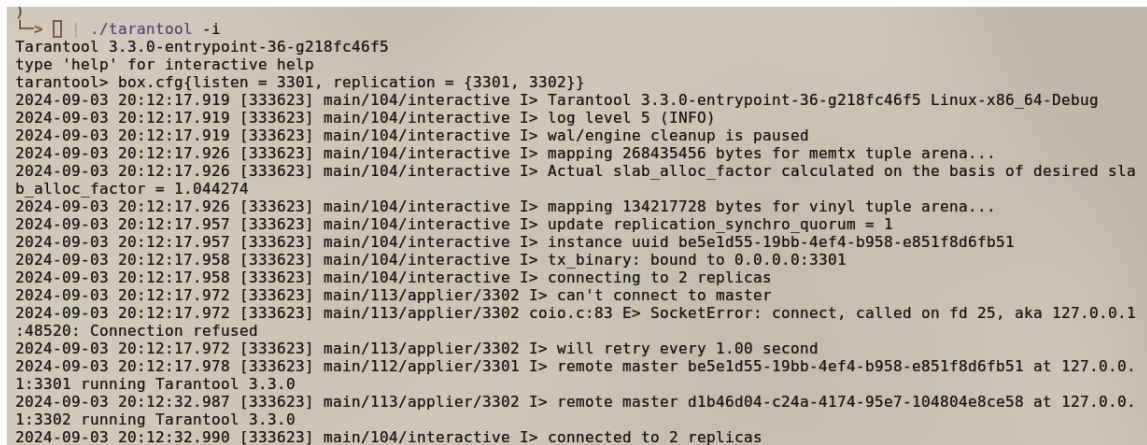


## Шаг 2. Конфигурация мастера

Первым шагом является настройка мастера, который будет основным источником данных для репликации. Его необходимо сконфигурировать со следующими параметрами [3]:

- *listen* - URI, на котором узел принимает входящие подключения.
- *replication* - список URI, с которых узел реплицирует данные.

Пример конфигурирования мастера приведен на рисунке 1.



```
./tarantool -i
Tarantool 3.3.0-entripoint-36-g218fc46f5
type 'help' for interactive help
tarantool> box.cfg{listen = 3301, replication = {3301, 3302}}
2024-09-03 20:12:17.919 [333623] main/104/interactive I> Tarantool 3.3.0-entripoint-36-g218fc46f5 Linux-x86_64-Debug
2024-09-03 20:12:17.919 [333623] main/104/interactive I> log level 5 (INFO)
2024-09-03 20:12:17.919 [333623] main/104/interactive I> wal/engine cleanup is paused
2024-09-03 20:12:17.926 [333623] main/104/interactive I> mapping 268435456 bytes for memtx tuple arena...
2024-09-03 20:12:17.926 [333623] main/104/interactive I> Actual slab_alloc_factor calculated on the basis of desired sla
b_alloc_factor = 1.044274
2024-09-03 20:12:17.926 [333623] main/104/interactive I> mapping 134217728 bytes for vinyl tuple arena...
2024-09-03 20:12:17.957 [333623] main/104/interactive I> update replication synchro quorum = 1
2024-09-03 20:12:17.957 [333623] main/104/interactive I> instance uuid be5e1d55-19bb-4ef4-b958-e851f8d6fb51
2024-09-03 20:12:17.958 [333623] main/104/interactive I> tx_binary: bound to 0.0.0.0:3301
2024-09-03 20:12:17.958 [333623] main/104/interactive I> connecting to 2 replicas
2024-09-03 20:12:17.972 [333623] main/113/applier/3302 I> can't connect to master
2024-09-03 20:12:17.972 [333623] main/113/applier/3302 coio.c:83 E> SocketError: connect, called on fd 25, aka 127.0.0.1
:48520: Connection refused
2024-09-03 20:12:17.972 [333623] main/113/applier/3302 I> will retry every 1.00 second
2024-09-03 20:12:17.978 [333623] main/112/applier/3301 I> remote master be5e1d55-19bb-4ef4-b958-e851f8d6fb51 at 127.0.0.
1:3301 running Tarantool 3.3.0
2024-09-03 20:12:32.987 [333623] main/113/applier/3302 I> remote master d1b46d04-c24a-4174-95e7-104804e8ce58 at 127.0.0.
1:3302 running Tarantool 3.3.0
2024-09-03 20:12:32.990 [333623] main/104/interactive I> connected to 2 replicas
```

Рисунок 1 – Конфигурация мастера

## Шаг 3. Конфигурация реплики

Реплики настраиваются аналогичным образом, но с указанием опции *read\_only*, устанавливающей режим только для чтения, чтобы запретить выполнение операций записи на реплике. Реплику можно сделать анонимной, указав параметр *replication\_anon*.

## Шаг 4. Проверка состояния репликасета

После настройки и запуска всех узлов необходимо убедиться, что репликасет работает корректно. Сделать это можно с помощью *box.info.replication*, как показано на рисунке 2. Эта команда включает информацию о подключении к узлу, статусе синхронизации (*vclock*) и задержке (*lag*) [3].

```

tarantool> box.info.replication
---
- 1:
  id: 1
  uuid: be5e1d55-19bb-4ef4-b958-e851f8d6fb51
  lsn: 5
  name: null
2:
  id: 2
  uuid: d1b46d04-c24a-4174-95e7-104804e8ce58
  lsn: 0
  upstream:
    status: follow
    idle: 0.74264272599976
    peer: '3302'
    lag: 8.9883804321289e-05
  name: null
  downstream:
    status: follow
    idle: 0.099791482996807
    vclock: {1: 5}
    lag: 0.00042939186096191
...

```

Рисунок 2 – Состояние подключения узла

## 2.3 Подключение узла к репликasetу

В данной части описывается протокол подключения реплики, так как это необходимо для понимания решений, предлагаемых к поставленным во введении задачам.

- а) В ходе конфигурации реплика генерирует UUID, являющийся уникальным идентификатором этого узла в репликasetе. Для каждого URI в *box.cfg.replication* создается сущность, называемая *applier*, задача которой состоит в применении данных, получаемых от мастера. *Applier* иницирует подключение к мастеру.
- б) *Applier* посылает сообщение *IPROTO\_JOIN*, при получении которого мастер создает *relay*, нужный для пересылки изменений БД на реплику. *IPROTO\_JOIN* представляет собой запрос на добавление узла к кластеру и необходим для получения начального состояния с мастера. Процесс отсылки начального состояния делится на две фазы.
- в) *Initial JOIN*. Мастер создает *read-view* (снимок) текущего состояния БД и посылает его реплике. По окончании *initial JOIN* мастер добавляет узел в репликaset путем вставки UUID реплики с соответствующим ID в спейс *\_cluster*.
- г) *Final JOIN*. С момента создания *read-view* до окончания его пересылки может пройти много времени и состояния БД наверняка изменится. Поэтому в фазе *final JOIN* мастер посылает все изменения, появившиеся со времени начала пересылки *read-view*.

- д) По окончании JOIN, реплика посылает запрос IPROTO\_SUBSCRIBE. Мастер отвечает своим текущим vclock-ом. С этого момента реплика переходит в стадию FOLLOW, она применяет все обновления WAL, исходящие от мастера.

В Tarantool также есть возможность создания анонимных реплик, которые не являются участником репликасета, не могут становиться мастером, не участвуют в кворуме синхронной репликации. Однако они получают и применяют поток репликационных данных. Вместо IPROTO\_JOIN они посылают IPROTO\_FETCH\_SNAPSHOT, который выполняет только первую фазу подключения: initial JOIN. Анонимные реплики не добавляются в спейс `_cluster`. В любой момент анонимная реплика может стать обычной, послав запрос IPROTO\_REGISTER.

### 3 Файловый JOIN

Одним из требований заказчика является возможность продолжения подключения реплики (JOIN) к репликasetу с места остановки в случае сбоя. Эта функциональность будет обеспечиваться так называемым файловым JOIN, подробности которого описаны в данной части. Он работает только для анонимных реплик, что достаточно для CDC.

Такое требование вызвано большим временем выкачивания read-view во время стадии initial JOIN, в течение которого могут происходить многочисленные ошибки сети. Однако продолжить скачивание после переподключения нельзя, так как read-view нигде не сохраняется.

Вместо read-view было принято решение использовать файлы snapshot для отсылки изначального состояния. Это делается с помощью модификации протокола запроса IPROTO\_FETCH\_SNAPSHOT, который отныне выглядит следующим образом:

- а) Реплика посылает запрос IPROTO\_FETCH\_SNAPSHOT, указывая IPROTO\_CURSOR (описание см. ниже).
- б) Мастер отвечает на IPROTO\_FETCH\_SNAPSHOT vclock-ом снапшота, который он собирается послать.
- в) После чего следует пересылка данных, каждая запись которых промаркирована с помощью LSN.
- г) В случае обрыва подключения, реплика посылает VCLOCK, полученный в пункте б, и LSN, до которого она успела получить данные.

Так как vclock и данные с LSN уже посылались и до этого, изменяется только IPROTO\_FETCH\_SNAPSHOT, приведенный на рисунке 3.

```
# <size>
msgpack({MP_UINT = size(<header>) + size(<body>)})

# <header>
msgpack({
  IPROTO_REQUEST_TYPE: IPROTO_FETCH_SNAPSHOT,
  IPROTO_SYNC: MP_UINT
})

# <body>
msgpack({
  # This one is added:
  (optional) IPROTO_CURSOR: MP_MAP
  VCLOCK: MP_MAP,
  LSN: MP_UINT,
})
```

Рисунок 3 – Изменения в запросе IPROTO\_FETCH\_SNAPSOT

Добавляется новое поле в тело запроса: `IPROTO_CURSOR`, представляющее собой таблицу с полями `VCLOCK` и `LSN`. Она указывает, откуда необходимо продолжить скачивание файла. Она может иметь следующие значения:

- $\{nil, nil\}$  - используется скачивания `read-view` для обратной совместимости
- $\{\{0\}, 0\}$  - курсор неизвестен, мастер берет последний сделанный `snapshot` и посылает его.
- $\{VCLOCK, LSN\}$  - курсор известен реплике и она хочет продолжить скачивание. Находим `snapshot`, идентифицируемый `vlock`-ом и начинаем пересылку с определенного `LSN`.

Мастер при получении `IPROTO_CURSOR` проверяет, что `снапшот` с запрашиваемым `vclock`-ом еще существует и что в нем есть запрашиваемый `LSN`. В противном случае реплике возвращается ошибка, она должна снова передать  $\{\{0\}, 0\}$  и начать скачивание уже другого `снапшота` заново.

## 4 Персистентный GC

Следующим требованием заказчика было сохранение данных для анонимных реплик. Эта функциональность будет обеспечиваться персистентным GC, описание которого приводится в данной части.

Это требование вытекает из того факта, что репликация идет только из xlog файлов. Мастер постоянно отслеживает, кому и какие xlog файлы нужны в репликасете, и не удаляет их. Однако для анонимных реплик это не делается, так как они не входят в репликасет. Когда xlog файлы, которые еще не были среплицированы на анонимную реплику, удаляются, то реплика вынуждена делать процесс ребутстрапа, включающий в себя выкачивание состояния БД с нуля. Это приводит к существенному понижению доступности анонимной реплики.

Было принято решение полностью переделать систему GC, так как до этого состояние GC находилось в памяти и после перезапуска узел не знал, какие файлы нужно сохранять, а какие можно удалять. Это приводило к тому, что узел был вынужден ждать подключения всех реплик к текущему узлу.

### Локальный спейс `_gc_consumers`

Теперь состояние GC сохраняется в новом локальном спейсе `_gc_consumers`, который имеет следующий формат, приведенный в таблице 1.

Таблица 1 – Формат `_gc_consumers`

Поле	Имя	uuid
	Тип	string
Поле	Имя	vclock
	Тип	map

UUID используется вместо id реплики, так как отныне xlog файлы могут также сохраняться для анонимных реплик, у которых id равно 0. Объединение всех анонимных реплик в одну строку не является оптимальным решением.

Для каждой из реплик создается GC consumer, который отслеживает, до какого vclock реплика уже получила данные. Их реализация (в памяти) остается без изменений, однако теперь они создаются/удаляются с использованием on\_replace триггера спейса `_gc_consumers`. Во время запуска узла они будут созданы автоматически.

### **Подключение обычных реплик**

Для обычных реплик появляется новый on\_replace триггер на спейс `_cluster`, в который реплика вставляется после окончания фазы initial JOIN. Этот триггер вставляет кортеж в спейс `_gc_consumers`, что вызовет создание GC consumer для данной реплики. Удаление реплики из спейса `_cluster` вызывает удаление записи из `_gc_consumers`. Прямое удаление реплики из `_gc_consumers` запрещено.

Таким образом, процесс подключения неанонимной реплики выглядит следующим образом:

- а) Реплика хочет быть присоединенной к репликасету и отправляет `IPROTO_JOIN`.
- б) Мастер создает read-view, посылает его реплике. Реплика вставляется в спейс `_cluster`, on\_replace триггер создает запись в `_gc_consumers`. Отныне файлы сохраняются для этой реплики.
- в) Реплика входит в фазу FOLLOW.

### **Подключение анонимных реплик**

Так как анонимная реплика нигде не сохраняется, как это делается для обычных реплик (`_cluster`), нам необходимо добавить новый флаг в `IPROTO_FETCH_SNAPSHOT: IPROTO_IS_PERSISTENT_GC`. По умолчанию флаг равен false, GC consumer-ы для анонимных реплик не создаются. Также этот флаг необходимо добавить в `IPROTO_SUBSCRIBE` (инициация фазы FOLLOW).

Таким образом, процесс подключения анонимной реплики выглядит следующим образом:

- а) Реплика отправляет `IPROTO_FETCH_SNAPSHOT` с флагом `IPROTO_IS_PERSISTENT_GC`. Реплика получает в ответ read-view или файл. Отныне xlog файлы хранятся для нее.
- б) Реплика отправляет `IPROTO_SUBSCRIBE` с тем же флагом.

Однако мы не можем позволить таким GC consumer-ам существовать вечно, как это сделано для обычных реплик. Иначе место на диске может быть быстро израсходовано. А потому добавляется новая опция конфигурирования: `wal_anon_gc_timeout`. Она показывает, сколько секунд живет GC consumer, после момента, когда мы в последний раз общались с репликой.



## 5 Фильтрация репликационного потока

Последним требованием заказчика была фильтрация репликационного потока. Это требование вытекает из необходимости снижения нагрузки на CDC реплики и заключается в частичной пересылки данных мастером.

Основные моменты:

- Фильтрация репликации производится по именам спейсов. Фильтруется как процесс подключения реплик, так и процесс применения изменений. Клиент может подписаться на изменения спейсов, которые еще не были созданы.
- Запрошенные спейсы реплицируются вместе с их метаданным: `_space`, `_index`, `_truncate` частично реплицируются, только данные, относящиеся к запрошенным спейсам, будут отправлены.
- При изменении имени или удаление спейса клиент получает информацию об этом последней записью. Последующие изменения (в случае переименования) не посылаются. Если будет создан спейс с именем, который запрошен клиентом, то клиент начинает получать данные из него.

Для обеспечения фильтрации в `IPROTO_FETCH_SNAPSHOT` и `IPROTO_SUBSCRIBE` добавляется новая опция: `IPROTO_SPACE_NAME_FILTER`, представляющая собой массив названий необходимых спейсов. Если опция не указана, полная репликация используется для обратной совместимости.

Управление подпиской на фильтрованный репликационный поток:

- Добавление нового еще не существующего спейса к фильтру. Клиент может инициировать `IPROTO_SUBSCRIBE` в любой момент, чтобы добавить новый спейс.
- Удаление спейса из фильтра. Делается таким же образом.
- Добавление уже существующего спейса в фильтр. Недопустимое изменение фильтра. На момент изменения фильтра клиент уже имеет какой-то `vclock` и мы продолжим посылать данные с этого `vclock`-а. Однако метаданные спейса будут скорее всего пропущены, так как спейс уже был создан до изменения фильтра.

## ЗАКЛЮЧЕНИЕ

В ходе прохождения практики был разработан новый протокол подключения анонимных реплик, удовлетворяющий требованиям команды CDC. Он выглядит следующим образом:

- а) Анонимная реплика отправляет `IPROTO_FETCH_SNAPSHOT`, сообщая мастеру, что она хочет реплицироваться с него. В этот запрос может быть включены следующие опции:
  - 1) `IPROTO_CURSOR` - реплика хочет использовать файловый JOIN для возможности продолжения подключения в случае разрыва подключения.
  - 2) `IPROTO_IS_PERSISTENT_GC` - необходимо создать анонимного GC consumer и сохранять xlog файлы для данной реплики.
  - 3) `IPROTO_SPACE_NAME_FILTER` - поток репликации должен быть фильтрованным. Требуется только частичные данные.
- б) В ответ мастер посылает vclock read-view или файла снапшота (в зависимости от `IPROTO_CURSOR`). Мастер создает анонимный GC consumer, если опция `IPROTO_IS_PERSISTENT_GC` равен true.
- в) Мастер начинает пересылку данных. Если `IPROTO_SPACE_NAME_FILTER` указан, то данные фильтруются на стороне сервера и посылаются частично.
- г) Реплика посылает `IPROTO_SUBSCRIBE`, указывая необходимость GC consumer и фильтрации потока.

В любой момент времени процесс подключения может быть разорван и продолжен с того же момента. Ошибки удаления файлов xlog больше не возникают, так как теперь есть возможность их сохранения для анонимных реплик. Нагрузка на CDC реплики была снижена, так как теперь им больше не приходится самостоятельно обрабатывать фильтрацию репликации.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. VKTech. Официальный сайт VK Tech. — 2024. — URL: <https://tech.mail.ru/> (дата обращения 28.07.2024).
2. Tarantool. Официальный сайт Tarantool. — 2024. — URL: <https://www.tarantool.io/ru/> (дата обращения 28.07.2024).
3. Tarantool. Documentation. — 2024. — URL: <https://www.tarantool.io/en/doc/latest/> (дата обращения 28.07.2024).