

Ora2PgSyncX

утилита репликации данных из Oracle в Postgres.

Руководство администратора.

версия 5.1

© ФОРС Телеком 2025.

Оглавление

1.	Описание технологии репликации	5
1.1.	Назначение и принципы работы утилиты	5
1.2.	Oracle XStream outbound server	7
1.3.	Обработка LCR и LCR position	8
1.4.	Обработка данных, получаемых из OLR	10
1.5.	Принципиальная схема процесса репликации с XStream	11
1.6.	Принципиальная схема процесса репликации с OLR	12
1.7.	Режимы работы утилиты и её отчёты	13
2.	Подготовка к репликации	14
2.1.	Настройки и предварительные операции в Oracle	14
2.1.1.	Режим архивирования журналов Oracle	14
2.1.2.	Создание табличного пространства для хранения очередей Row LCR	14
2.1.3.	Создание пользователя	14
2.2.	Настройки и предварительные операции в целевом PostgreSQL	15
2.2.1.	Уровень журналирования	15
2.2.2.	Права пользователя	15
2.3.	Настройки и предварительные операции в служебном PostgreSQL	15
2.4.	Установка утилиты и технические требования для её работы	15
2.5.	Хранение паролей в зашифрованном виде	16
2.6.	Запуск утилиты для подготовки репликации	16
2.6.1.	Конфигурация утилиты для подготовки репликации	17
2.6.2.	Результаты работы утилиты для подготовки репликации и структура таблиц с описанием реплицируемых данных	21
2.6.3.	Устранение проблем, выявленных утилитой	26
2.6.4.	Повторное выполнение подготовки репликации	26
2.6.5.	Риски разделения таблицы по нескольким слотам	27
2.6.5.1.	Обновление разных строк одинаковыми значениями уникального ключа.	27
2.6.5.2.	Использование одинаковых значений уникального ключа при вставке и удалении разных строк	29
2.7.	Выполнение сгенерированных скриптов	30
3.	Выполнение репликации	31

3.4.	Запуск и конфигурационные параметры утилиты репликации	31
3.5.	Установка начальной точки применения изменений	34
3.5.1.	При репликации через XStream	34
3.5.1.1.	При первом запуске	34
3.5.1.2.	При последующих запусках	35
3.5.2.	При репликации через OLR.....	35
3.5.2.1.	При первом запуске	35
3.5.2.2.	При последующих запусках	35
3.6.	Структура таблиц для мониторинга и управления репликацией	35
3.7.	Служебная таблица в исходной БД.....	40
3.8.	Служебная таблица в целевой БД.....	40
3.9.	Ограничения целостности и триггеры целевой БД	40
3.10.	Остановка утилиты репликации.....	41
3.10.1.	Остановка в результате ошибки репликации	41
3.10.2.	Внеплановая остановка по команде пользователя.....	41
3.10.3.	Плановая остановка по команде пользователя	41
3.10.4.	Плановая остановка по команде из внешней системы	42
3.11.	Возобновление работы утилиты репликации.....	42
3.11.1.	Внеплановый быстрый перезапуск.....	42
3.11.2.	Плановый быстрый перезапуск.....	42
3.11.3.	Повторный запуск репликации	43
3.12.	Режим отладки репликации	43
3.13.	Обработка ошибок	44
3.14.	Репликация данных таблицы по условию	48
3.14.1.	Применение правил Oracle XStream.....	48
3.14.2.	Применение правил перезаписи PostgreSQL.....	48
3.14.3.	Применение условий репликации в утилите	49
3.14.3.1.	Синтаксис условий репликации	49
3.14.3.2.	Трансформация оператора UPDATE.....	53
3.14.3.3.	Использование констант заданных пользователем.....	54
3.14.4.	Применение запросов фильтрации в утилите	55
3.15.	Трансформация данных.....	57
3.15.1.	Применение функций PostgreSQL.....	57
3.15.2.	Применение правил перезаписи и триггеров PostgreSQL	57
3.16.	XStream - репликация данных из одной исходной СУБД в несколько целевых СУБД	57

3.16.1.	Постановка задачи.....	57
3.16.2.	Решение задачи.....	58
3.16.2.1.	Создание схем в служебном Postgres.....	58
3.16.2.2.	Подготовка файлов.....	58
3.16.2.3.	Генерация и выполнение скриптов	60
3.16.2.4.	Выполнение репликации в две целевые БД.....	62
3.16.	Проблемы в процессе репликации.....	64
3.16.1.	После запуска утилиты репликации не происходит передача данных из Oracle, при этом состояние процесса репликации IDLE (ora2pg_proc.status).....	64
3.16.2.	Во время репликации наблюдается неприемлемый рост лага отставания целевой БД от исходной БД	65
3.16.3.	Превышение памяти выделенной для JVM	65
3.16.4.	После создания XStream outbound появляются лишние группы дополнительного журналирования.....	65
4.	Синхронизация текущих значений последовательностей	66
5.	Ограничения	66

1. Описание технологии репликации

Описание утилиты намеренно начинается с описания технологий, а не с инструкции по установке. Читатель может не захотеть стать пользователем утилиты после знакомства с применяемой в ней технологией. Здесь и далее намеренно повторяются важные сведения о технологии и алгоритмах утилиты. Это сделано для любителей чтения “по диагонали”.

В документе много таблиц. Читатель не должен думать, что чтение их содержимого ненужное ему занятие, если он надеется стать пользователем утилиты.

1.1. Назначение и принципы работы утилиты

Утилита предназначена для репликации данных из СУБД Oracle в СУБД Postgres. Утилита может использовать две технологии репликации:

- Oracle XStream производства Oracle Corp.
- Oracle Logical Replicator (OLR) производства ООО Форс Телеком

Утилита может применяться в целях интеграции систем и в целях тестирования производительности БД Postgres до принятия решения о переходе с Oracle на Postgres. Утилита может применяться для репликации данных при выполнении стратегии частичного и постепенного перехода прикладной системы из Oracle на Postgres. Она может применяться для завершения процесса миграции данных из Oracle в Postgres с целью минимизации времени простоя. Для этой цели более предпочтительным является применение утилиты Ora2PgSyncX с модулем OLR.

Работающий экземпляр утилиты использует один, и только один XStream outbound server (или один сервер OLR) и реплицирует данные в одну и только в одну целевую БД Postgres. Несколько работающих экземпляров утилиты могут быть сконфигурированы для эффективной репликации данных из одной БД Oracle в две и более целевых БД Postgres.

Изменения данных, выполняемые в таблицах БД Oracle (операторы DML) в закодированном виде поступают из XStream или OLR на вход модуля replicator утилиты. Перечень реплицируемых таблиц определяется заданной конфигурацией. Replicator, в соответствии с заданной конфигурацией, принимает решение о необходимости выполнения этих изменений в таблицах БД PostgreSQL. Для параллельного применения изменений утилита может использовать несколько соединений с Postgres. Для каждого соединения ведётся очередь, в которую replicator помещает операторы DML. Обработчики очередей являются отдельными параллельными процессами внутри утилиты. Совокупность очереди, процесса её обслуживания и сеанса в целевой БД в данном документе называется слотом. Термин “слот” заимствован из технологии логической репликации Postgres.

Важным является обеспечение равномерной нагрузки в слотах. Т.е. важно избежать ситуации, когда одни слоты простаивают, а другие перегружены и не успевают применять изменения в целевой БД.

Утилита может использовать два основных способа обеспечения равномерной нагрузки. Первый состоит в том, что таблицы исходной СУБД приблизительно равномерно распределяются по слотам в соответствии с неким рейтингом нагруженности. При этом, один слот обслуживает несколько таблиц. А, одна таблица реплицируется только через один слот. Второй способ состоит в том, что данные одной таблицы примерно равномерно распределяются по нескольким слотам. Ключом разделения, как правило, является значение числового первичного ключа, взятое по модулю от общего количества слотов для разделения таблиц. Т.о. второй способ обеспечивает деление таблиц на несколько не связанных друг с другом частей. При этом, один слот

обрабатывает части нескольких разных таблиц. Управление распределением обеспечивается настройками и заданной конфигурацией утилиты. В качестве ключа разделения строк таблицы по слотам может выступать любой not-null столбец или их группа. Но данные в таких столбцах не должны меняться в прикладной системе операторами Update. Кроме того, в разделяемой по слотам таблице не должны изменяться операторами update значения первичных и уникальных ключей. Это может вызвать ошибки репликации и нецелостное состояние данных таблицы в целевой БД. В качестве эффективного ключа разделения может выступать псевдостолбец ROWID. Однако, есть случаи, когда он не применим:

а) Index Organized Table

б) Partitioned table with row movement option

в) Если приложение меняет первичный ключ в таблице. Или удаляет запись и вставляет с тем же значением первичного ключа или уникального ключа.

Перечень имён столбцов таблицы для разделения таблицы на параллельные сеансы (слоты) репликации утилита выявляет автоматически. Но, пользователь может ввести собственные настройки. Если утилита не сможет выявить в таблице критерий разделения по слотам (например, нет первичного ключа и уникального ключа), то таблица будет привязана к одному из слотов репликации целиком в соответствии с её рейтингом нагрузки. Пользователь может сам указать, в каком слоте должна реплицироваться таблица.

При получении из Oracle оператора commit, утилита посылает его в очереди всех слотов. В целевой СУБД PostgreSQL создана служебная таблица, в которой сохраняются SCN и позиция в XStream последнего commit для каждого слота. Такое сохранение выполняется в той же транзакции, что и прикладные изменения. После выполнения каждого commit одним из слотов утилита оценивает необходимость отправки в XStream или OLR данных о последней позиции, применённой в целевой СУБД. Такая необходимость возникает в следующих случаях.

- Ещё не было отправки в XStream последней позиции
- Позиция ранее выполненной отправки меньше позиции данного оператора commit.

При возобновлении работы утилиты после плановой или внезапной остановки репликация будет продолжена с самой маленькой последней позиции из всех слотов. При этом, начальные транзакции при возобновлении могут быть пропущены некоторыми слотами, если их последние позиции больше поступающих из XStream транзакций.

Если для репликации применяется XStream, то когда в исходной БД нет изменений для репликации, т.е. прикладная система простаивает, утилита через API опрашивает XStream для выяснения того, до какой позиции выполнено чтение данных внутри процессов XStream. Если эта позиция выше позиции последнего commit направленного в очередь слота, то в очередь направляется дополнительное сообщение, называемое в утилите “commit холостого хода” – idle commit. При обработке этого сообщения процесс слота выполняет commit в целевой БД. Позиция этого “commit холостого хода” также запоминается в служебной таблице. Это позволяет выполнять более быстрый перезапуск репликации после длительных простоев.

Типы данных реплицируемые утилитой:

- VARCHAR2
- NVARCHAR2
- NUMBER
- FLOAT
- LONG
- DATE
- BINARY_FLOAT
- BINARY_DOUBLE

- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- RAW
- LONG RAW
- CHAR
- NCHAR
- CLOB
- NCLOB
- BLOB
- XMLType stored as CLOB
- XMLType stored as binary XML

Утилита предполагает разумно устроенное соответствие типов столбцов таблицы в целевой БД. Обеспечена поддержка LargeObject в Postgres для данных превышающих 1Гб.

Операторы SQL реплицируемые утилитой:

- INSERT
- UPDATE
- DELETE
- TRUNCATE TABLE (если таблица не разделяется по слотам репликации и репликация выполняется через XStream а не через OLR)

Кроме целевого Postgres, утилита использует обслуживающий Postgres. В этом вспомогательном Postgres ведутся таблицы конфигураций и мониторинга процессов репликации. Эти данные могут размещаться и в целевой БД, но желательно освободить её от дополнительной нагрузки.

1.2. Oracle XStream outbound server

XStream out входит в состав СУБД Oracle. Его работа опирается на сервер Oracle Streams Capture, Oracle LogMiner и Oracle AQ для выполнения захвата и обработки изменений в таблицах. Подробно об это написано в <https://docs.oracle.com/en/database/oracle/oracle-database/12.2/xstrm/xstream-out.html>

Вся инфраструктура процессов репликации данных из Oracle является затратной по ресурсам, как памяти, так и процессора. Особенно это относится к LogMiner и Oracle Streams capture. Это объясняет использование утилитой лишь одной сессии LogMiner, одного процесса захвата и одного сервера XStream out.

Oracle AQ используется в технологии XStream для организации очередей, в которых операторы DML и DDL хранятся в формате Logical Change Record (LCR). Если размер транзакции большой, и она перестаёт помещаться в оперативной памяти, то данные, подготавливаемые для передачи клиенту логической репликации, будут сохраняться в таблицах Oracle AQ.

Данные, передаваемые процессом захвата серверу XStream и сервером XStream клиенту (утилите), фильтруются т.н. правилами (Oracle Rules). Существуют два основных типа правил генерируемых утилитой для создания серверов Oracle Streams и Oracle XStream :

правила схем и правила таблиц. Правило схем позволяет задать список схем, таблицы которых подлежат репликации. Правило таблиц позволяет задать список реплицируемых таблиц имена, которых включают в себя имя схемы. Если используется правила схем, то для каждой схемы в Oracle будут создавать два правила: одно для Streams и другое для XStream. Аналогично и для правил таблиц – по два правила для каждой реплицируемой таблицы. Большое количество правил может стать причиной слишком долгого процесса старта репликации и задержки самой репликации. Очевидна не оптимальность такого решения в Oracle. Кроме того, эти правила не позволяют решить следующую простую задачу. Пусть в БД есть схема с 10-тью тысячами таблиц. И пусть, реплицироваться должны 9-ть тысяч из них. Oracle позволяет задать негативные и позитивные правила. Можно сделать позитивное правило для схемы и одну тысячу негативных правил для таблиц, не подлежащих репликации. Однако, в этом случае, ненужные таблицы всё равно будут реплицироваться. Для них сработает позитивное правило схемы. В связи с этим утилита генерирует только позитивные правила либо таблиц, либо схем. И не генерирует их сочетаний.

Данные, которые поступают из XStream, и которые не нужны в целевой БД будут отфильтрованы в самой утилите и направлены в служебный слот с кодом DUMMY. Этот слот используется лишь для индикации потока посторонних данных и хранения последней наименьшей позиции и SCN commit для более быстрого перезапуска утилиты.

Если применяется правило схем и нагрузка на реплицируемых таблицах сопоставима с нагрузкой на ненужных таблицах, то производительность репликации может деградировать. Особенно, если в ненужных таблицах изменяются LOB-данные. В этом случае рекомендуется использовать правило таблиц.

К одному процессу Oracle Streams Capture может быть подключено несколько серверов XStream out. И можно организовать многопоточность за счёт тиражирования XStream out. Однако, это вызовет увеличение количества проверяемых правил. Разделение данных одной таблицы на несколько серверов XStream out возможно, Об этом написано в https://docs.oracle.com/en/database/oracle/oracle-database/12.2/arpls/DBMS_XSTREAM_ADM.html#GUID-9DE9C191-D9E8-411D-A723-385B46570962 Но при тестировании, были выявлены ситуации некорректного срабатывания этих правил подмножеств (subset rules). Поэтому авторы утилиты отказались от идеи такой организации многопоточности.

Применение нескольких XStream out в сочетании с одним процессом Capture целесообразно при необходимости репликации данных в несколько целевых БД (несколько экземпляров Postgres). См. п.3.16.

Достоинством технологии XStream является возможность репликации данных в режиме реального времени с минимальным лагом времени между исходной БД и целевой БД. Это достигается тем, что в Oracle процесс захвата изменений (Oracle Streams Capture) делает захват данных не только из архивных журналов, но и из оперативных журналов, и даже из оперативной буферной памяти оперативных журналов.

1.3. Обработка LCR и LCR position

Утилита получает от XStream out данные в формате LCR.

LCR – Logical Change Record, используется в Oracle Streams, Oracle XStream и в Oracle Golden Gate для организации логической репликации данных. LCR делится на два основных типа:

- DDL LCR – запись об изменении структур таблиц и других объектов
- Row LCR – запись об операторах insert/update/delete и commit выполненных в исходной БД.

Подробно об этом написано в документации: <https://docs.oracle.com/en/database/oracle/oracle-database/12.2/arpls/Logical-Change-Record-TYPEs.html#GUID-1462FDB7-E00B-4F7B-8C59-4AEA5FFCF6BD>

Примечательно то, что если LCR относится к оператору DML, в котором фигурируют данные столбцов типов LONG, LONGRAW, CLOB, NCLOB, BLOB, XMLType, то такие данные поделены на части (chunks). Для чтения частей данных в XStream предусмотрены отдельные функции. Программа – клиент логической репликации не может перейти к следующему LCR, пока не прочтёт содержимое всех частей, всех столбцов таких типов участвующих в операторе DML. Попытка это сделать вызовет ошибку генерируемую со стороны Oracle.

Важным атрибутом LCR является т.н. position – позиция LCR в потоке репликации. Позиция это бинарная последовательность из 33-х байт. Каждая LCR имеет уникальную позицию. XStream передаёт клиенту логической репликации сообщения LCR в возрастающем порядке позиций. В Oracle есть SCN, которым помечены любые изменения в БД. Однако, SCN не является достаточным для организации потока репликации в котором:

- а) Транзакции передаются в порядке очередности их завершения в исходной БД
- б) Все операции как внутри одной транзакции, так и среди всех транзакций потока репликации имеют уникальные возрастающие идентификаторы

Эти требования позволяют клиенту логической репликации построить многопоточную программу выполнения операций в целевой БД с соблюдением целостности в рамках отдельно взятой таблицы. Т.е. никакая строка таблицы не будет удаляться или изменяться раньше её вставки. Эти требования обосновывают реализацию в Oracle Streams LCR с атрибутом position в дополнении к SCN.

Если транзакция 1 началась раньше транзакции 2, а закончилась позже транзакции 2, то XStream будет выдавать сначала транзакцию 2, а потом транзакцию 1. При этом SCN некоторых операторов в транзакции 2 будут больше SCN некоторых операторов транзакции 1. Позиции, в получаемых от XStream LCR, транзакции 2 будут возрастать. И продолжат возрастать, когда начнут поступать LCR транзакции 1.

В Oracle есть функции для получения SCN из позиции LCR. И наоборот – LCR position из SCN. Однако нельзя полагаться на их результат при принятии решения о порядке выполнения операций. Эти функции преобразований работают не всегда корректно.

Данные из DDL LCR, относящиеся к реплицируемым таблицам утилита направляет в слот DUMMY. Вполне вероятно, что сразу после этого возникнет ошибка репликации из-за асимметрии структуры таблиц.

Row LCR операций insert, update и delete утилита преобразовывает в оператор SQL для применения в целевой БД, снабжает его массивом bind-переменных и направляет в очередь одного из слотов. Операции DML для не реплицируемых таблиц помещаются в очередь слота DUMMY.

LCR position является достаточным для организации многопоточной репликации без дублирования и потерь. Тем не менее, утилита сохраняет и SCN последнего commit в специальной таблице целевой БД. Это сделано в целях совместимости работы с OLR, мониторинга и создания дополнительного критерия возможности удаления архивных лог файлов Oracle.

Если DDL LCR содержит оператор TRUNCATE TABLE, и если он относится к реплицируемой таблице, и если эта таблица реплицируется через один слот и не разделяется между слотами, то TRUNCATE TABLE будет применён к целевой таблице.

Все остальные DDL LCR утилита не реплицирует и направляет в слот DUMMY, который не применяет их в целевой БД.

Невозможность репликации TRUNCATE TABLE в случае разделения таблицы на параллельные потоки репликации через несколько слотов обусловлена необходимостью соблюдения целостности. Не корректно выполнять TRUNCATE для таблицы, если очереди в нескольких слотах содержат для неё операторы DML. Эти DML могут быть выполнены до или после TRUNCATE в зависимости от длины очередей и скоростей их обработки. Тем самым состояние данных в целевой таблице может в итоге не соответствовать состоянию данных в исходной таблице.

1.4. Обработка данных, получаемых из OLR

OLR передаёт утилите данные в своём собственном формате. Он описан в документации к OLR. В этом формате у каждого DML- оператора есть собственный SCN возрастающий в рамках транзакции. Кроме этого каждый DML – оператор, полученный от OLR снабжён COMMIT_SCN – SCN оператора COMMIT транзакции. Используя эти данные, утилита синтезирует LCR position, и преобразовывает оператор DML, полученный от OLR, к формату LCR.

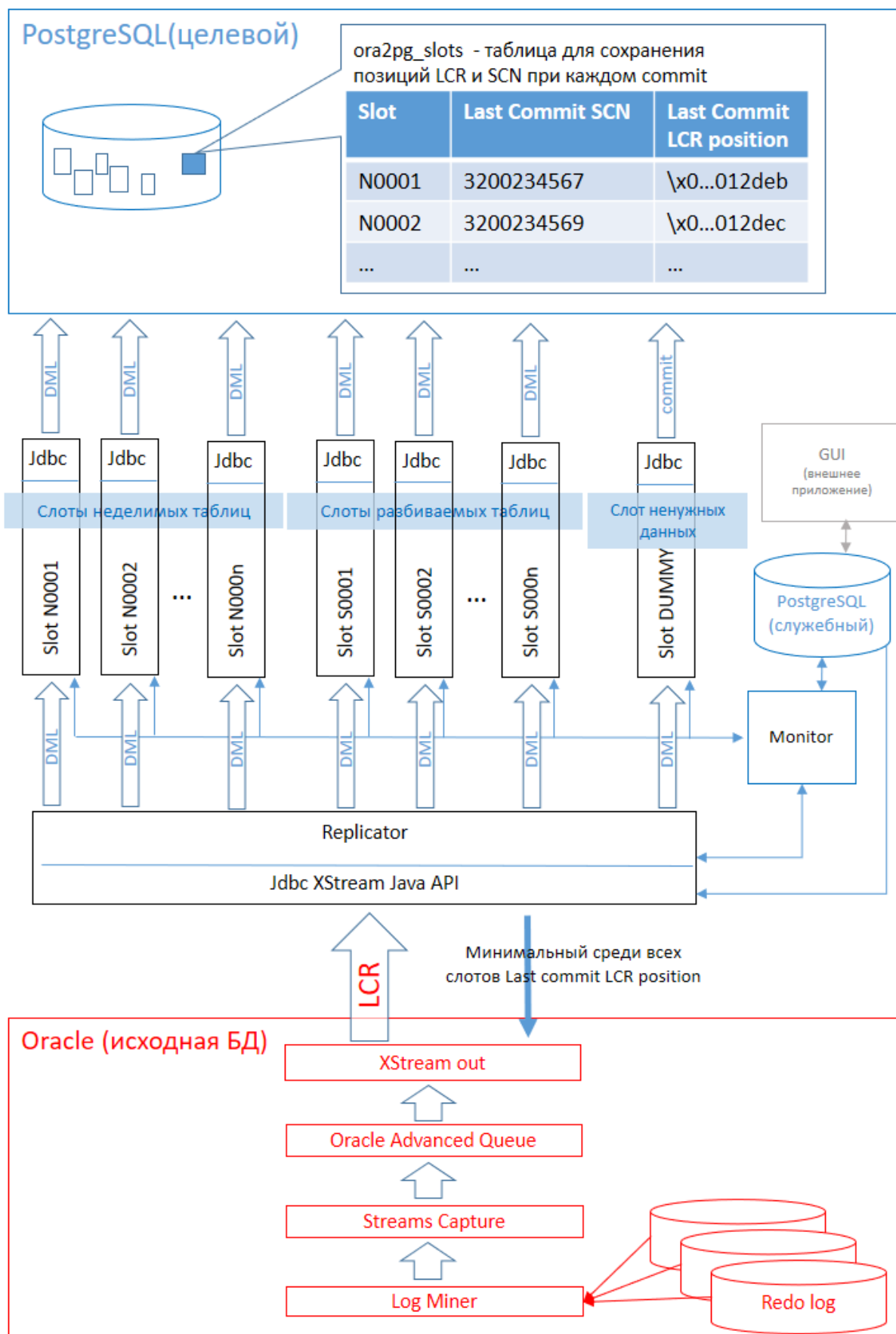
OLR не реплицирует операторы DDL в том числе TRUNCATE.

При подключении к OLR, утилита передаёт список интересующих её таблиц исходя из данных в ora2pg_tab.

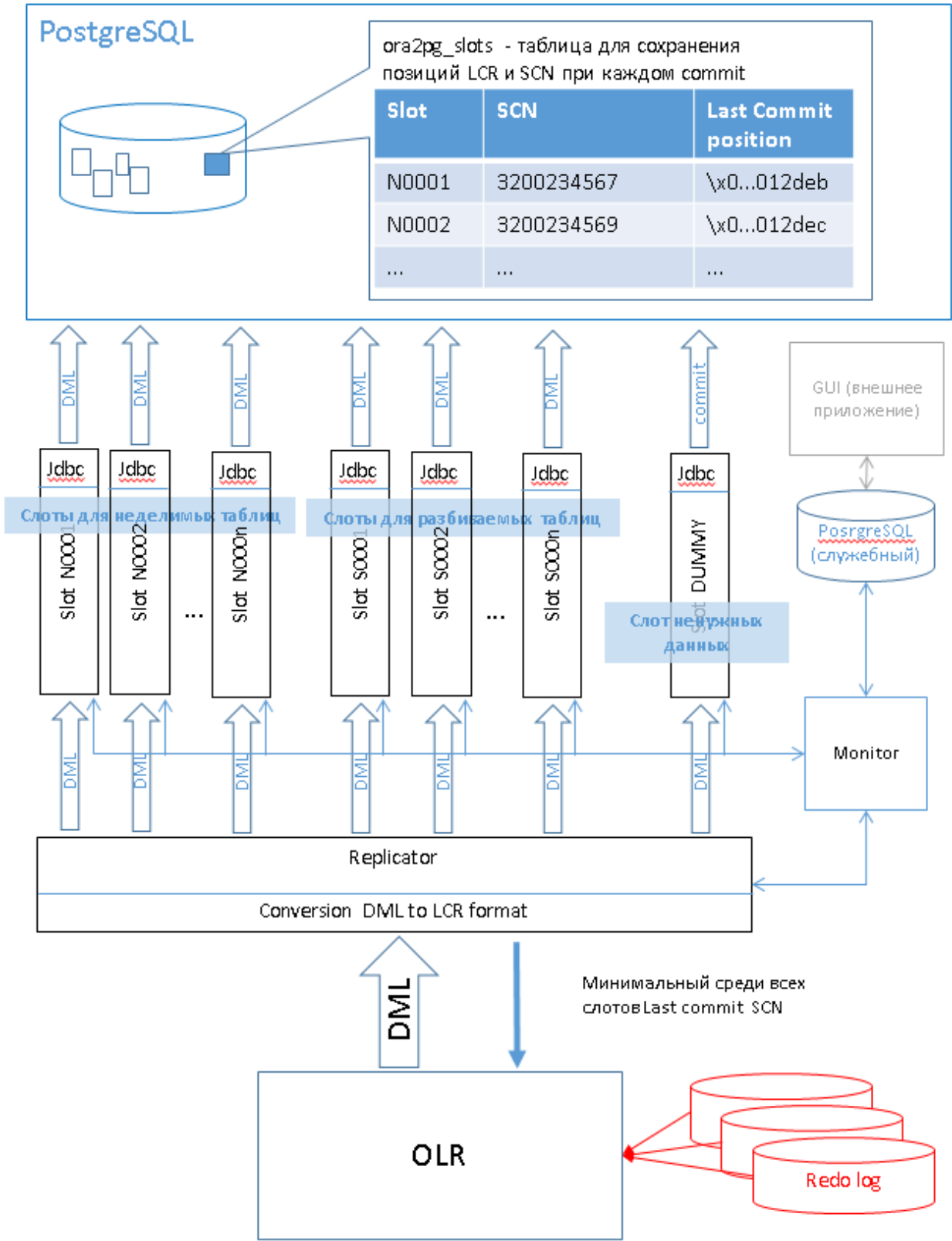
В ходе репликации от утилиты в OLR постоянно поступает минимальный SCN транзакции зафиксированный слотами. OLR запоминает его в специальном файле и использует для последующих перезапусков.

OLR не оказывает влияние на производительность сервера Oracle. OLR реплицирует данные из архивных redolog файлов. Скорость репликации через OLR как правило выше скорости репликации через Oracle XStream. Оперативность доставки новых изменений в целевой Postgres существенно ниже, чем при применении XStream. Это связано с тем, что новые изменения, выполненные в БД, далеко не сразу отразятся в архивных redolog файлах.

1.5. Принципиальная схема процесса репликации с XStream



1.6. Принципиальная схема процесса репликации с OLR



1.7. Режимы работы утилиты и её отчёты

Конфигурационный файл в формате json в параметре “action” задаёт режим работы. Для каждого из режимов предусмотрен собственный файл json и командный скрипт запуска.

Помимо выдачи сообщений на консоль утилита формирует файл с отчётом о работе.

В ходе работы утилиты этот файл будет иметь имя следующей структуры <режим>.rep.work.

По окончании работы файл переименовывается в <режим>.rep.

Ниже представлена таблица режимов и файлов

Режим “action”:”...”	Назначение	json	Командный скрипт	Файл отчёта
prepare	Начальная подготовка репликации с чтением данных о реплицируемых таблицах	prepare.json	prepare.bat prepare.sh	prepare.rep
prepare_scripts	Повторная подготовка без наполнения данных о реплицируемых таблицах, но с учётом пользовательских изменений в описании реплицируемых таблиц	prepare_scripts.json	prepare_scripts.bat prepare_scripts.sh	prepare_scripts.rep
replication	Выполнение репликации	replication.json	replication.bat replication.sh	replication.rep
seq_oracle2pg	Синхронизация последовательностей исходной и целевой БД	seq_oracle2pg.json	seq_oracle2pg.bat seq_oracle2pg.sh	seq_oracle2pg.rep

2. Подготовка к репликации

2.1. Настройки и предварительные операции в Oracle

2.1.1. Режим архивирования журналов Oracle

Этот режим нужен XStream для выполнения репликации.

Включён ли этот режим можно узнать, выполнив запрос в БД Oracle:

```
SQL> SELECT LOG_MODE FROM V$DATABASE;
```

Возможные значения:

- ARCHIVELOG – режим архивирования логов включён.
- NOARCHIVELOG – режим архивирования логов выключен.

Если режим архивирования логов не активен, то его следует включить.

а) Вход в sqlplus администратором БД на сервере Oracle.

```
sqlplus /nolog
SQL> connect / as sysdba
```

б) Указание места для хранения и формата именования лог-файлов, если ещё не указано.

Например:

```
ALTER SYSTEM SET log_archive_dest_1=
'location=/opt/oracle/product/12.2/dbs/arch' SCOPE=spfile;
ALTER SYSTEM SET log_archive_format='%t_%s_%r.arc' SCOPE=spfile;
```

в) Выполнение последовательности команд:

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

2.1.2. Создание табличного пространства для хранения очередей Row LCR

```
CREATE TABLESPACE xstrmadmin_tbs DATAFILE
'/opt/oracle/oradata/orcl/xstrmadmin.dbf' SIZE 25M REUSE AUTOEXTEND ON
MAXSIZE UNLIMITED;
```

2.1.3. Создание пользователя

```
CREATE USER xstrmadmin IDENTIFIED BY xstrmadmin
                        DEFAULT TABLESPACE xstrmadmin_tbs
                        QUOTA UNLIMITED ON xstrmadmin_tbs;
GRANT CREATE SESSION TO xstrmadmin;
GRANT DBA TO xstrmadmin;
GRANT SELECT ANY TABLE TO xstrmadmin;
GRANT ALTER ANY TABLE TO xstrmadmin;
--выполнение этого блока требуется только для репликации через Xstream
BEGIN
DBMS_XSTREAM_AUTH.GRANT_ADMIN_PRIVILEGE (
    grantee=>'xstrmadmin',
    privilege_type=>'CAPTURE',
    grant_select_privileges => TRUE
);
END;
/
```

2.2. Настройки и предварительные операции в целевом PostgreSQL

2.2.1. Уровень журналирования

Для повышения производительности репликации для параметра `wal_level` желательно установить значение `minimal`. К сожалению, это не всегда приемлемо по соображениям обеспечения надёжности.

2.2.2. Права пользователя

Пользователь должен иметь права на `insert/update/delete` для всех реплицируемых таблиц.

Пользователь должен иметь право создания таблицы в схеме и с именем задаваемым параметром `ora2pg_slots_t`.

Пользователь должен иметь право установки `session_replication_role = replica`. Это право в PostgreSQL имеет суперпользователь. Или пользователь, которому оно дано оператором `grant set on parameter session_replication_role to <имя пользователя>;`

2.3. Настройки и предварительные операции в служебном PostgreSQL

Пользователь должен иметь право создания таблиц в схеме задаваемой параметром `pgschema`. По умолчанию – “public”. Если пользователь служебной БД не является супер пользователем и/или размещение служебных таблиц в схеме `public` не желательно, то необходимо создать пользователя и схему. Пример:

```
create user ora2pgsyncx with password 'ora2pgsyncx';
create schema ora2pgsyncx;
alter schema ora2pgsyncx owner to ora2pgsyncx;
```

Если Ваша поставка утилиты включает в себя GUI-приложение разработанное для сервера приложений LUI, то после установки сервера приложений LUI необходимо выполнить следующее:

1. От имени пользователя `lui` необходимо дать права созданному пользователю на роли `lui_common` и `lui_admin`. Пример:

```
grant lui_common to ora2pgsyncx;
grant lui_admin to ora2pgsyncx;
```
2. Выполнить скрипт `Ora2PgSyncX_LUI.sql` пользователем `lui` в обслуживаемой БД.

Приложение `ORA2PGSYNCX` в сервере приложений LUI станет доступным для входа после первого запуска скрипта `prepare.sh`.

2.4. Установка утилиты и технические требования для её работы

Утилита устанавливается простым копированием всех файлов из дистрибутива в нужный пользователю каталог.

Для её работы требуется:

- JVM версии 11 или выше
- Oracle (исходная БД) версии 12.2 или выше
- PostgreSQL (целевая БД) версии 15 или выше

- PostgreSQL (обслуживающая БД) версии 12 или выше
- Дисковая память 40Мб (минимум)
- Оперативная память 8Гб (минимум)
- Oracle Instant Client 12.2 или выше (Это требование Oracle XStream Java API, если репликация выполняется через OLR – не требуется)

Для работы через (OCI-драйвер) на сервере, в котором будет работать Ora2PgSyncX, должен быть установлен Oracle Instant Client version 19

(<https://www.oracle.com/database/technologies/instant-client/downloads.html>). Если у Вас нет возможности установить Oracle Instant Client version 19, но у Вас уже есть Oracle Instant Client версии 12 и выше, то можно выполнить следующее. Скопируйте ojdbc8.jar из папки, в которой установлен Oracle Instant Client в папку lib утилиты Ora2PgSyncX. При этом дайте файлу имя ojdbc8-19.8.0.0.jar. Т.е. следует заменить файл ojdbc8-19.8.0.0.jar.

На клиенте Oracle для лучшей производительности желательно отключить генерацию файлов диагностики и журналов. Для этого в файле sqlnet.ora установите параметры

TRACE_LEVEL_CLIENT = OFF

LOG_LEVEL_CLIENT = OFF

TRACE_DIRECTORY_CLIENT= /dev/null

LOG_DIRECTORY_CLIENT = /dev/null

LOG_FILE_CLIENT = /dev/null

DIAG_ADR_ENABLED=OFF

DIAG_DDE_ENABLED=FALSE

DIAG_SIGHANDLER_ENABLED=FALSE

Если Oracle client установлен в Windows замените “/dev/null” на “nul”.

2.5. Хранение паролей в зашифрованном виде

Утилита позволяет хранить в конфигурационных файлах пароли доступа к базам данных в зашифрованном виде. Шифрование и дешифрование производится самой утилитой.

Скрипт encrypt.sh (bat) содержит пример вызова утилиты для получения из файла conn.json нового файла с уже зашифрованными паролями. Новый файл получает имя conn_c.json.

После получения нового файла его имя необходимо указать в параметре “connections” во всех json-файлах конфигураций.

Параметры конфигурации с зашифрованными паролями имеют такие же коды, как и параметры паролей без шифрования, но в конце кода добавлен символ “с”.

2.6. Запуск утилиты для подготовки репликации

Утилита для подготовки репликации запускается командным файлом prepare.sh или prepare.bat. Задачами утилиты для подготовки являются:

- Формирование перечня реплицируемых таблиц и их свойств
- Формирование перечня столбцов реплицируемых таблиц и их свойств
- Выявление проблем мешающих репликации
- Выявление таблиц, каждая из которых может быть разделена на параллельные потоки репликации
- Выявление таблиц, не подлежащих разделению на параллельные потоки и автоматическое распределение таких таблиц по слотам репликации

- Выявление последовательностей применяемых в реплицируемых таблицах с целью выравнивания текущих значений в исходной и целевой БД после завершения репликации
- Генерация скриптов подготавливающих исходную БД к репликации

2.6.1. Конфигурация утилиты для подготовки репликации

Утилита для подготовки репликации использует конфигурационный файл `prepare.json`.
Ниже описаны элементы этого конфигурационного файла.

Параметр	Значение по умолчанию	Пример значения	Назначение
Подключения. Атрибуты подключений к База Данных можно указать в отдельном файле его имя должно быть задано в параметре "connections". Пример : "connections": "conn.json". Наличие отдельного файла с описанием подключений выгодно по соображениям безопасности и по соображениям недублируемости информации при конфигурировании утилиты для запуска репликации.			
oradb		jdbc:oracle:oci:@localhost:1521:orcl	URL подключения к исходной БД
orausername		xstrmadmin	Имя пользователя в исходной БД
orapassword			Пароль пользователя в исходной БД
pgtdb		targethost:5432/postgres	URL подключения к целевой БД
pgtusername		postgres	Имя пользователя в целевой БД
pgtpassword		postgres	Пароль пользователя в исходной БД
pgdb		maintain:5432/postgres	URL подключения к обслуживающей БД
pgusername		postgres	Имя пользователя в обслуживающей БД
pgpassword		postgres	Пароль пользователя в обслуживающей БД
pgschema	public	ora2pgsyncx	Схема служебных таблиц
Режим			
action		prepare	<p>Код действия выполняемого программой</p> <ul style="list-style-type: none"> • prepare – первоначальное наполнение (полное обновление) служебных таблиц, выявление проблем, распределение по слотам, генерация скриптов • prepare scripts – обновление не

			выполняется, выполняется выявление проблем, распределение по слотам, генерация скриптов
Имена процессов Xstream в Oracle			
xout	ora2pg		Имя XStream out сервера и, одновременно, идентификатор экземпляра утилиты одновременно
xcapture	ora2pg_capture		Имя XStream capture процесса
xqueue	ora2pg_queue		Имя XStream queue – очередь AQ для XStream
Определение источника данных			
dmlsource	xstream	olr	xstream – репликация через XStream olr – репликация через OLR
dmlsourcehost		172.25.5.13	Хост сервера OLR
dmlsourceport		64000	Порт сервера OLR
Прочие параметры			
XStreamRuleType	schemas		Тип правил XStream для захвата и репликации требуемых данных (используется для создания и обновления сервера XStream): <ul style="list-style-type: none"> • schemas – формируется перечень реплицируемых схем • tables – формируется перечень реплицируемых таблиц
slots4notsplit	5	10	Количество слотов для таблиц, не подлежащих делению на сеансы параллельной репликации.
query4tables		ora:select t.OWNER ora_schema, --имя схемы в Oracle t.TABLE_NAME ora_table, --имя таблицы in Oracle lower(t.OWNER) pg_schema, --имя схемы в Pg lower(t.TABLE_NAME) pg_table, --имя таблицы в Pg (nvl(t.AVG_ROW_LEN,0)+1)*(nvl(t.NUM_ROWS,0)+1) -- рейтинг нагруженности для распределения неделимых таблиц по потокам	Запрос, возвращающий список реплицируемых таблиц. Запрос должен возвращать следующие столбцы в указанном порядке : <ul style="list-style-type: none"> • Имя схемы в исходной БД

		<pre> from sys.dba_tables t where t.OWNER in ('SCHEMA1','SCHEMA2') -- перечень реплицируемых схем and t.TABLE_NAME not in ('...','...') -- исключаемые таблицы </pre>	<ul style="list-style-type: none"> Имя таблицы в исходной БД Имя схемы в целевой БД Имя таблицы в целевой БД Рейтинг нагрузки (число) <p>Оператору SQL должен предшествовать префикс, указывающий в какой БД следует выполнять запрос. ORA: - в исходной БД PGT: - в целевой БД PG: - в обслуживающей БД</p>
query4splitTablesByModulo		<pre> pg:select c.ora_schema,c.ora_table, split_part(string_agg(c.ora_column,',')',',',MAX(c.ora_pk)) split from public.ora2pg_col c where c.ora_pk is not null group by c.ora_schema,c.ora_table having string_agg(c.ora_type,',') like '%NUMBER' </pre> <p>Этот запрос возвращает список разделяемых по слотам таблиц. В него входят таблицы, у которых есть первичный ключ и последний (или единственный) член первичного имеет тип NUMBER.</p> <p>Тип NUMBER в столбцах, по которым делается разделение таблицы на группы реплицируемых строк не обязателен. Если столбец имеет иной тип, то значения будут конвертированы в число хэш-функцией. Остаток от деления на количество слотов заданное в параметре slots4split в replication.json будет вычисляться от суммы значений столбцов или их хэш-значений. Таким образом, из запроса можно удалить фразу having. Это будет целесообразно, например, если в приложении есть сильно нагруженные таблицы с нечисловым первичным ключом.</p> <p>Внимание! Таблицы, в которых операторами update могут меняться значения первичных или уникальных ключей должны быть исключены из результатов этого запроса. Это можно сделать, добавив во фразу where предикат "and ora_table not in ('...','...','...')"</p>	<p>Запрос, возвращающий список таблиц подлежащих разделению на сеансы параллельной репликации. Запрос должен возвращать следующие столбцы в указанном порядке :</p> <ul style="list-style-type: none"> Имя схемы в исходной БД Имя таблицы в исходной БД Ключ разделения (перечень столбцов через запятую) <p>Оператору SQL должен предшествовать префикс, указывающий в какой БД следует выполнять запрос. ORA: - в исходной БД PGT: - в целевой БД PG: - в обслуживающей БД</p>
query4sequences		<pre> ora:select s.SEQUENCE_OWNER,s.SEQUENCE_NAME, lower(s.SEQUENCE_OWNER), lower(s.SEQUENCE_NAME) from dba_sequences s where s.SEQUENCE_OWNER in ('...') </pre>	<p>Запрос, возвращающий список последовательностей. Запрос должен возвращать следующие столбцы в указанном порядке :</p> <ul style="list-style-type: none"> Имя схемы в исходной БД Имя последовательности в исходной БД Имя схемы в

			<p>целевой БД</p> <ul style="list-style-type: none"> Имя последовательности в целевой БД <p>Оператору SQL должен предшествовать префикс, указывающий в какой БД следует выполнять запрос. ORA: - в исходной БД PGT: - в целевой БД PG: - в обслуживающей БД</p>
query4column		<pre>select 'MY_SCHEMA' ora_schema, 'MY_TABLE' ora_table, 'MY_COLUMN_ORA' ora_column, "my_column_pg" pg_column from dual</pre>	<p>Запрос, возвращающий соответствие (mapping) столбцов таблиц Oracle столбцам таблиц Postgres.</p> <p>По умолчанию утилита подбирает для столбца таблицы oracle столбец таблицы Postgres по соответствию имён без учёта регистра. Однако, можно задать собственное соответствие имён столбцов. Запрос должен возвращать столбцы в указанном порядке:</p> <ul style="list-style-type: none"> Имя схемы в исходной БД Имя таблицы в исходной БД Имя столбца в исходной БД Имя столбца в целевой БД <p>Оператору SQL должен предшествовать префикс, указывающий в какой БД следует выполнять запрос. ORA: - в исходной БД PGT: - в целевой БД PG: - в обслуживающей БД</p>

После установки утилиты пользователь должен внести изменения в файл prepare.json. Следует как минимум в параметре query4tables и в параметре query4sequences ввести имена реплицируемых схем.

2.6.2. Результаты работы утилиты для подготовки репликации и структура таблиц с описанием реплицируемых данных

Результаты работы утилиты в режиме подготовки репликации отражаются в служебных таблицах

ora2pg_tab, ora2pg_col, ora2pg_seq

Ниже описана структура этих таблиц.

ora2pg_tab – описание реплицируемых таблиц

column_name	description
ora_schema	Схема в Oracle
ora_table	Таблица в Oracle
pg_schema	Схема в Postgres
pg_table	Таблица в Postgres
pg_name	Имя таблицы в Postgres по правилам Postgres
ora_xsup	Возможна ли репликация таблицы через XStream out или OLR FULL - да Все остальные значения - нет
ora_iot_type	Тип Index Organized Table
ora_part	Признак секционированности таблицы в Oracle
ora_part_row_mov	Row movement опция для секционированной таблицы в Oracle
Split	Перечень имён столбцов таблицы (через запятую) для разделения таблицы на параллельные сеансы (слоты) репликации (т.е. ключ разделения). Генерируется программой.
split_u	Перечень имён столбцов таблицы для разделения таблицы на параллельные сеансы (слоты) репликации, введённый пользователем
Slot	Имя слота привязки для неразделяемой по слотам таблицы
slot_u	Имя слота привязки для неразделяемой по сеансам таблицы. Вводится пользователем.
Problem	Описание проблемы, которая делает невозможной репликацию таблицы
Warning	Описание проблемы, которая может мешать репликации
Ratio	Числовой рейтинг нагрузки. Чем больше, тем выше.
Exclude	Признак исключения таблицы из репликации. Любой текст, введённый пользователем в это поле, исключит таблицу из репликации.
ignore_updcnt	Признак необходимости игнорировать ошибки обработки нулевого количества строк Если пусто или N – не игнорировать Если Y-игнорировать Если C-игнорировать с учётом значения переменной сеанса ORA2PG.DISCARD
Notes	Примечание пользователя
refreshed	Статус обновления при выполнении повторного запуска I-новая таблица, U- данные о таблице были обновлены, N

	– нет изменений
condition	Условие репликации, заданное пользователем
condition_engine	EvalEx или JavaScript – вычислитель условия
condition_error	Ошибка, выявленная при проверке условия репликации
condition_binds	Переменные условия репликации
discard	Запрос к целевой БД для фильтрации DML перед применением.
discard_error	Ошибка, выявленная при проверке запроса фильтрации
discard_binds	Переменные запроса фильтрации, выявленные при проверке запроса
discard_mode	N- если запрос не вернул строк DML не выполняется Y- если запрос вернул хотя-бы одну строку DML не выполняется
discard_cmpld	Запрос фильтрации “скомпилированный” для jdbc генерируется при проверке запроса
discard_db	pg –запрос фильтрации будет выполняться в Postgres ora – запрос фильтрации будет выполняться в Oracle
discrd_upd_src	Источник данных указанный пользователем для запроса фильтрации при обработке оператора UPDATE: N – новая запись O – старая запись

Столбцы split_u и slot_u созданы для ввода данных пользователем, чтобы сохранить данные генерируемые утилитой.

Правило обработки следующее:

Если split_u!=null и slot_u!=null, то таблица не будет делиться на несколько потоков. Все её данные будут обрабатываться в указанном пользователем слоте.

Если split_u!=null и slot_u=null, то таблица будет делиться на несколько потоков.

Если split_u=null и slot_u=null и split!=null и slot!=null, то таблица не будет делиться на несколько потоков. Все её данные будут обрабатываться в слоте, выбранном утилитой.

Если split_u=null и slot_u=null и split!=null и slot==null, то таблица будет делиться на несколько потоков.

ora2pg_col – описание столбцов реплицируемых таблиц

column_name	description
Id	ID описания
ora_schema	Схема в Oracle
ora_table	Таблица в Oracle
ora_column	Столбец в Oracle
ora_type	Тип данных в Oracle
ora_len	Макс.длина в Oracle
ora_prec	Точность чисел и времени в Oracle
ora_scale	Размер дробной части чисел в Oracle
ora_nullable	Возможность null в Oracle (Y/N)
ora_pk	Порядок столбца в первичном ключе в Oracle
ora_uk	Порядок столбца в уникальном ключе в Oracle. (Первый встречный uk без nullabe столбцов)
ora_tc	Категория типов данных в Oracle S - строки символов N - числа

	D - дата/время T - интервал даты/времени или года/месяца U - двоичные данные
pg_schema	Схема в Postgres
pg_table	Таблица в Postgres
pg_column	Столбец в Postgres
pg_column_name	Имя столбца в Postgres по правилам в Postgres
pg_type	Тип данных в Postgres
pg_len	Макс.длина в Postgres
pg_prec	Точность чисел и времени в Postgres
pg_scale	Точность чисел и времени в Postgres
pg_nullable	Возможность null в Postgres (Y/N)
pg_func	Функция - обёртка в целевой БД над данными столбца Пример для конвертирования Oracle CHAR(1) в Postgres boolean: case when ?='Y' then true else false end
pg_pk	Порядок столбца в первичном ключе в Postgres
pg_uk	Порядок столбца в уникальном ключе в Postgres. (Первый встречный uk без nullabe столбцов)
pg_ukh	Признак вхождения столбца в какой-нибудь уникальный ключ в Postgres (Y/N)
pg_tc	Категория типов данных в Postgres S - строки символов N - числа D - дата/время T - интервал даты/времени или года/месяца U - двоичные данные
problem	Описание проблемы, которая не даёт возможности реплицировать данные столбца или таблицы в целом.
warning	Описание проблемы рискованной репликации
ident	Признак вхождения столбца в идентификатор, используемый для идентификации строки в целевой БД Как правило, этот столбец часть первичного ключа в целевой БД. Генерируется утилитой. (Y/N)
ident_u	Признак вхождения столбца в идентификатор, используемый для идентификации строки в целевой базе. Указывается пользователем. (Y/N)
exclude	Признак исключения столбца из репликации. Любой текст, введённый в это поле исключает столбец из репликации
expression	Выражение для синтеза значения столбца (пока не используется)
expression_binds	Переменные выражения для синтеза значения столбца (пока не используется)
notes	Примечания пользователя
refreshed	Статус обновления при выполнении повторного запуска I-новый столбец, U- данные о столбце были обновлены, N – нет изменений
oracolumn_id	Порядковый номер столбца при создании таблицы в

Oracle

Значения в этих столбцах `ident` и `ident_u` используются утилитой для:

- Генерации скрипта с операторами `alter table ... add supplemental logging`, чтобы Oracle всегда и для любого `update` и `delete` записывал данные этих столбцов в `redolog`.
- Синтеза фразы `where` реплицируемых `update` и `delete`

Столбец `ident_u` создан для ввода данных пользователем, чтобы сохранить данные генерируемые утилитой. Данные, введенные в `ident_u` доминируют над `ident`. Если пользователем будут внесены изменения в `ident_u` хотябы для одного столбца таблицы, то при репликации и при генерации команд `"ALTER TABLE ... add SUPPLEMENTAL LOG GROUP ..."` в файле `prepareXadmSL.sql` будут использованы только данные указанные в столбце `ident_u`, данные сгенерированные утилитой в столбце `ident` не будут использоваться.

ora2pg_seq

column_name	description
<code>pg_schema</code>	Схема в Postgres
<code>pg_sequence</code>	Sequence в Postgres
<code>pg_last_value</code>	Последнее значение в Postgres
<code>pg_max_value</code>	Максимальное значение в Postgres
<code>pg_min_value</code>	Минимальное значение в Postgres
<code>pg_incr</code>	Инкремент в Postgres
<code>pg_cch</code>	Размер кэш в Postgres
<code>ora_schema</code>	Схема в Oracle
<code>ora_sequence</code>	Sequence в Oracle
<code>ora_last_value</code>	Последнее значение в Oracle
<code>ora_max_value</code>	Максимальное значение в Oracle
<code>ora_min_value</code>	Минимальное значение в Oracle
<code>ora_incr</code>	Инкремент в Oracle
<code>ora_cch</code>	Размер кэш в Oracle
<code>sql4set</code>	Оператор SQL для установки значения в Postgres
<code>err</code>	Ошибка установки значения в Postgres
<code>problem</code>	Описание проблемы выявленной на этапе подготовки
<code>warning</code>	Предупреждение о возможных проблемах
<code>exclude</code>	Признак исключения последовательности из процесса установки значений в Postgres
<code>Datestamp</code>	Время и дата изучения последовательности
<code>dateora2pg</code>	Время и дата установки значения в postgres

Если для таблицы выявлена проблема, её описание помещается в поле `ora2pg_tab.problem` Таблицы, для которых это поле не пусто не реплицируются.

При анализе таблиц утилита может выявить следующие проблемы:

- `Postgres table not found` – для таблицы в Oracle не была найдена соответствующая таблица в PostgreSQL
- `XStream and OLR not full support` – репликация таблицы не поддерживается. Это возможно в случаях:
 - Используются не скалярные типы данных (`nested table`)
 - Временная таблица

- Внешняя таблица
- Таблица принадлежит одной из схем:
 - CTXSYS
 - DBSNMP
 - DMSYS
 - DVSYS
 - EXFSYS
 - LBACSYS
 - MDDATA
 - MDSYS
 - OLAPSYS
 - ORDDATA
 - ORDPLUGINS
 - ORDSYS
 - OUTLN
 - SI_INFORMTN_SCHEMA
 - SYS
 - SYSMAN
 - SYSTEM
 - WMSYS
 - XDB

При анализе таблиц утилита может сделать в столбце warning следующие предупреждения:

- PK/UK absent – в PostgreSQL отсутствует первичный ключ или уникальный ключ с not-null столбцами. Это предупреждение, а не проблема потому, что при репликации строк такой таблицы они будут идентифицированы в Postgres по значениям всех столбцов, кроме столбцов типов: BLOB, CLOB, NCLOB, LONG, LONGRAW, XMLType.

Если для столбца выявлена проблема, её описание помещается в поле ora2pg_col.problem.

Столбцы, для которых это поле не пусто игнорируются при репликации.

При анализе столбцов таблицы утилита может выявить следующие проблемы:

- PK hole – в PostgreSQL столбец является частью первичного ключа, а в Oracle одно из:
 - Нет соответствующего столбца
 - Столбец nullable
 - Поле exclude не пусто
- UK hole - в PostgreSQL столбец является частью уникального ключа, а в Oracle одно из:
 - Нет соответствующего столбца
 - Столбец nullable и в PostgreSQL not null
 - Поле exclude не пусто
- Column hard hole - в PostgreSQL столбец not null, а в Oracle одно из:
 - Нет соответствующего столбца
 - Столбец nullable
 - Поле exclude не пусто

При анализе столбцов таблицы утилита может в столбце warning сделать следующие предупреждения:

- Column soft hole – в PostgreSQL столбец nullable, а в Oracle одно из:
 - Нет соответствующего столбца
 - Поле exclude не пусто

- Datatype conflict – несоответствие категорий типов данных

После запуска prepare.bat (или prepare.sh) пользователь может менять содержимое служебных таблиц для устранения проблем. Пользователь может разработать собственное приложение с GUI-интерфейсом, опираясь на знание структуры таблиц. Или обратиться к поставщику утилиты для получения такого приложения.

После этого можно запустить prepare_scripts.bat (или prepare_scripts.sh) для повторного анализа и генерации скриптов.

Утилита генерирует следующие скрипты:

- prepare_XsdmSL.sql – для настройки передачи в redolog Oracle дополнительной информации о выполняемых DML-операциях. В этом файле операторы “alter table add SUPPLEMENTAL LOG GROUP ...” будут генерироваться только для таблиц, для которых в ora2pg_tab.exclude пусто (is null). В скрипте перед добавлением нужных “log group” выполняется удаление системных групп и групп, ранее созданных утилитой. Имена групп содержат префикс равный имени сервера XStream и генерируемый порядковый номер. В группе перечислены столбцы, совокупность значений в которых идентифицирует строку в таблице целевой БД.
- prepare_XadmCX.sql – для создания в Oracle сервера XStream out.
- prepare_XadmRS.sql – для перезапуска в Oracle сервера XStream out с нужного SCN
- prepare_XadmDX.sql – для удаления в Oracle сервера XStream out.

Замечание: prepare_scripts.bat (или prepare_scripts.sh) генерирует скрипты, начинающиеся с “prepare_scripts”.

2.6.3. Устранение проблем, выявленных утилитой

Одной из главных задач при подготовке к репликации является устранение проблем, выявленных утилитой. Способы устранения проблем:

- 1 Удаление описаний проблемных таблиц из таблицы ora2pg_tab и ora2pg_col. После этого потребуется запуск утилиты в режиме prepare_scripts.
- 2 Изменение условия запроса указанного в параметре query4tables в prepare.json для исключения проблемных таблиц. После этого потребуется повторный запуск утилиты в режиме prepare.
- 3 Исправление проблем с помощью операторов DDL в исходной и/или целевой БД. После этого потребуется повторный запуск утилиты в режиме prepare.
- 4 Удалить все данные в столбце problem таблиц ora2pg_col и ora2pg_tab

2.6.4. Повторное выполнение подготовки репликации

При повторном запуске prepare.sh пользователь должен ответить на вопрос, который возникает на экране консоли:

Table list already consists of nnnn tables.

Do you want to delete old table definitions? (Y/N)

Если пользователь ответит “Y”, то все данные из таблиц ora2pg_tab и ora2pg_col будут удалены, включая изменения введенные пользователем.

Если пользователь ответит “N”, то выполняется обновление данных в этих таблицах.

При этом вновь выполняются запросы, указанные в параметрах query4tables и query4columns.

Если будут выявлены новые таблицы и(или) новые столбцы, то они будут добавлены (refreshed=I).

Если будут выявлены новые изменения в описаниях таблиц и(или) столбцов, то они будут изменены (refreshed=U). При обновлении не меняются данные в столбцах с постфиксом “_u”.

Если будут выявлены старые таблицы и(или) столбцы, которых нет среди таблиц/столбцов возвращаемых запросами, то такие таблицы/столбцы будут удалены.

Повторная подготовка репликации может быть использована при плановом перезапуске репликации для реконфигурации при изменении структур данных.

В файле `prepare_XadmSL` операторы `alter table` разделены на группы:

- new tables
- changed tables
- non-changed tables

Для успешного перезапуска рекомендуется выполнить первые две группы.

При повторном выполнении `prepare.sh` если `XStreamRuleType = tables`, в случае выявления новых таблиц генерируется скрипт `prepare_XadmAX.sql`. Его следует использовать для добавления новых таблиц в набор правил репликации для ранее созданного сервера XStream out.

2.6.5. Риски разделения таблицы по нескольким слотам

Разделение таблицы на несколько потоков репликации по ROWID на первый взгляд привлекательно. И кажется идеальным решением для любых случаев. Однако, есть случаи, когда он не применим:

а) Index Organized Table

б) Partitioned table with row movement option

в) Если приложение меняет первичный ключ в таблице. Или удаляет запись и вставляет с тем же значением первичного ключа или уникального ключа.

Как уже было сказано выше, разделение таблицы на несколько потоков репликации следует выполнять по столбцу или группе столбцов, которые имеют ограничение `not null` и значения которых не изменяются операторами `update` в приложении. Как правило, такое разделение делается по числовому первичному ключу, значения которого берутся из последовательности. Однако, если таблица помимо первичного ключа имеет ещё и уникальный ключ (один или более), существуют риски получения ошибок и/или искажение данных в целевой БД. В результате множества экспериментов были выявлены две наиболее типичные ситуации проявления этих рисков. Ниже эти ситуации описаны двумя примерами.

2.6.5.1. Обновление разных строк одинаковыми значениями уникального ключа.

Создание таблицы в Postgres

```
create table test(id int not null, uk char(1) not null,
primary key(id), unique (uk));
```

Создание таблицы в Oracle

```
create table test(id number not null, uk char(1) not null,
primary key(id), unique (uk));
```

Заполнение таблицы в Oracle

```
insert into test(id, uk) values (1,'1');
insert into test(id, uk) values (2,'2');
insert into test(id, uk) values (3,'3');
insert into test(id, uk) values (4,'4');
commit;
```

Репликация таблицы через два сеанса. Нечётные ID в 1-м сеансе, чётные – во втором сеансе. Эти четыре оператора insert выполнены в Postgres двумя транзакциями в двух сеансах.

Затем в прикладной системе в рамках одного сеанса на стороне Oracle выполнена транзакция:

```
update test set uk='5' where id=1;
update test set uk='6' where id=2;
update test set uk='1' where id=1;
update test set uk='5' where id=4;
update test set uk='2' where id=2;
update test set uk='6' where id=3;
commit;
```

Состояние таблицы в результате:

Id	uk
1	1
2	2
3	6
4	5

Репликация этой транзакции выполняется в двух сеансах в следующей последовательности.

№№	1-й сеанс	2-й сеанс	Результат										
1.	update test set uk='5' where id=1;		1 row updated										
2.	update test set uk='1' where id=1;		1 row updated										
3.		update test set uk='6' where id=2;											
4.		update test set uk='5' where id=4;	Внутренняя блокировка в Postgres. 2-й сеанс висит в блокировке										
5.	update test set uk='6' where id=3;		1-й сеанс получит ошибку SQL Error [40P01]: ERROR: deadlock detected и выполнит rollback 2-й сеанс успешно выполнит действие №4										
6.		update test set uk='2' where id=2;	1 row updated										
7.		commit;	Commted Состояние таблицы: <table><tr><th>Id</th><th>Uk</th></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr><tr><td>3</td><td>3</td></tr><tr><td>4</td><td>5</td></tr></table>	Id	Uk	1	1	2	2	3	3	4	5
Id	Uk												
1	1												
2	2												
3	3												
4	5												
1-й сеанс делает попытку повторить свою транзакцию													
8.	update test set uk='5' where id=1;		1-й сеанс получит ошибку SQL Error [23505]: ERROR: duplicate key value violates unique constraint Key (uk)=(5) already exists										

2.6.5.2. Использование одинаковых значений уникального ключа при вставке и удалении разных строк

Создание таблицы в Postgres

```
create table test(id int not null, uk char(1) not null,
primary key(id), unique (uk));
```

Создание таблицы в Oracle

```
create table test(id number not null, uk char(1) not null,
primary key(id), unique (uk));
```

До начала репликации таблица пуста. Репликация таблицы через два сеанса. Нечётные ID в 1-м сеансе, чётные – во втором сеансе.

В прикладной системе в рамках одного сеанса на стороне Oracle выполнена транзакция:

```
insert into test(id, uk) values (1,'1');
delete from test where id=1;
insert into test(id, uk) values (2,'2');
delete from test where id=2;
insert into test(id, uk) values (3,'2');
delete from test where id=3;
insert into test(id, uk) values (4,'1');
commit;
```

В результате в таблице окажется всего одна строка:

Id	uk
4	1

Репликация этой транзакции выполняется в двух сеансах в следующей последовательности.

№№	1-й сеанс	2-й сеанс	Результат
1	insert into test(id, uk) values (1,'1');		1 row inserted
2	delete from test where id=1;		1 row deleted
3		insert into test(id, uk) values (2,'2');	1 row inserted
4		delete from test where id=2;	1 row deleted
5	insert into test(id, uk) values (3,'2');		Внутренняя блокировка в Oracle. 1-й сеанс висит в блокировке
6		insert into test(id, uk) values (4,'1');	2-й сеанс получит ошибку SQL Error [40P01]: ERROR: deadlock detected и выполнит rollback 1-й сеанс успешно выполнит действие №5
7	delete from test where id=3;		

8	<code>commit;</code>		Commted Состояние таблицы: Нет строк – пуста				
2-й сеанс делает попытку повторить свою транзакцию							
9		<code>insert into test(id, uk) values (2, '2');</code>	1 row inserted				
10		<code>delete from test where id=2;</code>	1 row deleted				
11		<code>insert into test(id, uk) values (4, '1');</code>	1 row inserted				
12		<code>commit;</code>	Commted Состояние таблицы: <table><tr><td>id</td><td>Uk</td></tr><tr><td>4</td><td>1</td></tr></table>	id	Uk	4	1
id	Uk						
4	1						

Таким образом, мы видим, что наибольшую опасность представляет собой update значений уникального ключа. Ошибки, возникающие при использовании одинаковых значений уникального ключа при вставке и удалении разных строк, могут быть преодолены при повторении транзакции. Пользователь может настроить такую обработку ошибок.

2.7. Выполнение сгенерированных скриптов

Необходимо от имени xstrmadmin выполнить скрипт prepare_XsdmSL.sql и скрипт prepare_XadmCX.sql если репликация выполняется через Oracle XStream.

При выполнении скрипта prepare_XadmCX.sql в исходной БД не должно быть активных транзакций. Их наличие можно проверить запросом

```
select s.sid, s.serial#, s.username, s.PROGRAM, s.MODULE, s.ACTION, s.CLIENT_INFO
from v$transaction t
inner join v$session s on t.addr = s.taddr;
```

Наличие активных транзакций может вызвать блокировки и ошибки выполнения скриптов.

Если есть необходимость начать репликацию с некоторого SCN, то можно использовать скрипт и prepare_XadmRSX.sql, внося в него изменения.

Пример скрипта:

```
--restart outbound:
DECLARE new_start_scn number:=null;
BEGIN
-- new_start_scn must be greater then first scn of the capture process
select first_scn into new_start_scn from dba_capture where capture_name='ORA2PG_CAPTURE';
DBMS_XSTREAM_ADM.STOP_OUTBOUND('ora2pg',false);
DBMS_CAPTURE_ADM.STOP_CAPTURE('ora2pg_capture');
DBMS_CAPTURE_ADM.ALTER_CAPTURE(
capture_name=>'ora2pg_capture',
start_scn=>new_start_scn);
DBMS_CAPTURE_ADM.START_CAPTURE('ora2pg_capture');
DBMS_XSTREAM_ADM.START_OUTBOUND('ora2pg');
END;
/
```

Действия для модификации и выполнения скрипта.

- 1) После выполнения prepare_XadmCX.sql узнайте требуемый Вам SCN. На пример выполните `select cuurent_scn from v$database;`
- 2) Убедитесь в том что требуемый SCN больше, чем first_scn процесса CAPTURE: `select first_scn from dba_capture where capture_name='ORA2PG_CAPTURE';`

3) Внесите изменения в скрипт:

```
--restart outbound:
DECLARE new_start_scn number:= Ваш SCN;
BEGIN
-- new_start_scn must be greater then first scn of the capture process
--select first_scn into new_start_scn from dba_capture where capture_name='ORA2PG_CAPTURE';
DBMS_XSTREAM_ADM.STOP_OUTBOUND('ora2pg',false);
DBMS_CAPTURE_ADM.STOP_CAPTURE('ora2pg_capture');
DBMS_CAPTURE_ADM.ALTER_CAPTURE(
capture_name=>'ora2pg_capture',
start_scn=>new_start_scn);
DBMS_CAPTURE_ADM.START_CAPTURE('ora2pg_capture');
DBMS_XSTREAM_ADM.START_OUTBOUND('ora2pg');
END;
/
```

4) Выполните исправленный скрипт.

Внимание ! Замечено, что в некоторых случаях установка start_scn для процесса XStream Capture с помощью DBMS_CAPTURE_ADM.ALTER_CAPTURE не приводит к ожидаемым результатам. И на вход утилиты поступают LCR с SCN меньшим, чем заказанный. Тогда следует применить альтернативный способ начать репликацию с нужного SCN – параметр startSCN в replication.json.

3. Выполнение репликации

3.4. Запуск и конфигурационные параметры утилиты репликации

Репликация запускается командным скриптом replication.bat или (replication.sh).

Конфигурационные параметры настраиваются в replication.json

Ниже описаны элементы этого конфигурационного файла.

Параметр	Значение по умолчанию	Пример значения	Назначение
Подключения. Атрибуты подключений к Базам Данных можно указать в отдельном файле его имя должно быть задано в параметре "connections". Пример : "connections": "conn.json"			
Oradb		jdbc:oracle:oci:@localhost:1521:orcl	URL подключения к исходной БД
Orausername		xstrmadmin	Имя пользователя в исходной БД
orapassword			Пароль пользователя в исходной БД
pgtdb		targethost:5432/postgres	URL подключения к целевой БД
pgtusername		postgres	Имя пользователя в целевой БД
pgtpassword		postgres	Пароль пользователя в исходной БД
pgdb		maintain:5432/postgres	URL подключения к обслуживающей БД
pgusername		postgres	Имя пользователя в обслуживающей БД
pgpassword		postgres	Пароль пользователя в обслуживающей БД
pgschema	public		Схема для служебных таблиц
Режим			
action		replication	Код действия выполняемого программой
Имена процессов XStream в Oracle			
xout	ora2pg		Имя XStream out сервера и, одновременно, идентификатор

			экземпляра утилиты репликации
xcapture	ora2pg_capture		Имя процесса XStream capture
xqueue	ora2pg_capture		Имя XStream AQ очереди
Определение источника данных			
dmlsource	xstream	olr	xstream – репликация через XStream olr – репликация через OLR
dmlsourcehost		172.25.5.13	Хост сервера OLR
dmlsourceport		64000	Порт сервера OLR
slots4split	5	10	Количество слотов для таблиц подлежащих делению на сеансы параллельной репликации
Таблица, создаваемая в целевой БД			
ora2pg_slots_t	public.ora2pg_slots		Таблица для хранения Last Committed SCN и LCR position слотов репликации
Параметры, изменяемые в ходе репликации			
Внимание! При повторном запуске репликации эти параметры будут считываться из таблицы ora2pg_proscrap			
ArclogDelete	False		Следует ли выполнять автоматическое удаление архивных лог файлов не требующихся более процессу репликации. Для удаления архивных журналов используется пакет Oracle dbms_backup_restore
ArclogMaxSize	1024	2	Максимальный суммарный размер в Гб архивных лог файлов. При его достижении может начаться удаление не нужных файлов.
BatchMaxBytes	100	1024	Максимальный размер batch в Кб
BatchMaxDMLs	200	100000	Максимальное количество операторов DML в batch
CompressLOB	1024		Минимальный размер LOB начиная с которого будет применяться сжатие данных при постановке оператора в очередь
ShowDiscardDMLhistory	true		Следует ли отображать отвернутые DML в истории операций слота (не DUMMY)
PgDMLStat	false		Следует ли собирать статистику о выполнении DML в целевой БД. Статистика собирается в таблице ora2pg_pgdmstats
Debug	false		Режим отладки
DebugBySteps	true		Пошаговая отладка. Если true и Debug=true, то после каждого отладочного сообщения необходимо нажатие Enter
MutValDelete	false		Если true, то одинаковые операторы delete будут собираться не в традиционный jdbc-batch, а в единый SQL-оператор с фразой usage (values (...),(...),...)
MultValInsert	false		Если true, то одинаковые операторы insert будут собираться не в традиционный jdbc-batch, а в единый SQL-оператор с фразой (values (...),(...),...)
MultValUpdate	false		Если true, то одинаковые операторы update будут собираться не в традиционный jdbc-batch, а в единый

			SQL-оператор с фразой from (values (...),(...),...)
OptimizeSelfUpdate	false		Если true, то утилита будет сокращать фразу set оператора update, если значения в полях “до изменения” и “после изменения” равны. Возможна ситуация когда фраза set окажется пустой, тогда такой оператор не будет выполняться в целевой БД.
ThrottleMemPct	90		Процент занятости памяти JVM, при достижении которого начнётся снижение скорости чтения данных из Oracle XStream и даже установка состояния PAUSE процесса репликации
DiscardQueryesResultCache	0	1000000000000	Размер кэша результатов запросов фильтрации. Измеряется в записях.
TxMinBytes	0	1024	Группа параметров для управления размером транзакций за счёт т.н. “прореживания” операторов commit. Если хотя бы один из этих параметров равен нулю, то прореживание не выполняется. Commit, поступивший из Oracle XStream будет направлен во все очереди слотов, если размер транзакции достигнет указанного в TxMinBytes количества Кб, или указанного в TxMinDMLs количества операторов SQL, или время от предыдущего commit достигнет указанного в TxMinMsec количества миллисекунд.
TxMinDMLs	0	100000	
TxMinMsec	0	60000	
Прочие параметры			
CharSet4XMLtype	UTF-8 Или Windows-1251		Этот параметр задаёт кодировку в терминах java.nio (https://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html). Эта кодировка будет применяться при чтении из oracle данных типа XMLTYPE. Данные этого типа Oracle XStream передаёт утилите в бинарном формате и их надо перекодировать в UTF-8. В БД oracle эти данные хранятся в кодировке БД. Имена кодировок БД не соответствуют именам кодировок java.nio. Если этот параметр не указан, а кодировка БД (SELECT value FROM nls_database_parameters WHERE parameter = 'NLS_CHARACTERSET') содержит строку ‘1251’. То параметр получит значение “Windows-1251”. В противном случае – “UTF-8”.
query4arclogs	select RECID, STAMP, NAME, THREAD#, SEQUENCE#, RESETLOGS_CHANGE#, FIRST_CHANGE#,		Запрос, возвращающий список архивных журналов. Этот запрос используется для подсчёта текущего суммарного размера архивных журналов при мониторинге работы репликации. Кроме того, если ArclogDelete=true, то удаляться будут только файлы возвращенные этим

	BLOCK_SIZE, BLOCKS, NEXT_CHANGE#, DICTIONARY_BEGIN , DICTIONARY_END from v\$archived_log l where l.ARCHIVED='YES' and l.DELETED='NO' and l.STATUS='A' and l.STATUS!='D'		запросом и ни какие другие журнальные файлы.
ReplStat2Console	true		Следует ли выводить в консоль статистику репликации
ApplyToPg	true		Следует ли выполнять DML в Postgres Этот параметр бывает полезен для определения “узкого места” в ходе тестирования репликации
ErrorProcessing	false		Следует ли использовать модуль обработки ошибок с возможностью их игнорирования и повторения транзакций.
path4tmpfiles	SLOTS_TMP_FILES		Имя и путь к каталогу хранения временных файлов используемых слотами для больших транзакций в режиме ErrorProcessing=true
historyDepth	40		Глубина истории операторов DML выполняемых слотами для отображения в столбце ora2pg_slots.history
startSCN	0		SCN, начиная с которого данные будут реплицироваться в целевую БД. Все RowLCR, у которых SCN меньше указанного в этом параметре будут игнорированы.

Если утилита репликации запускается впервые, то она начнёт получать изменения, которые выполнены в исходной БД начиная с момента создания процесса захвата (Oracle Streams Capture) или с заданного Вами SCN. См. П.2.7.

3.5. Установка начальной точки применения изменений

3.5.1. При репликации через XStream

3.5.1.1. При первом запуске

При первом запуске захват изменений процессом CAPTURE начнётся с SCN, который указан при создании CAPTURE – процесса. См.П.2.7 Если начальный SCN не указан, то захват изменений начнётся с current SCN БД. Применение изменений в целевой БД начнётся со startSCN, указанного в конфигурационном файле. Если startSCN не указан в конфигурационном файле, то будут применяться все изменения без каких-либо пропусков (за исключением репликации по условиям и запросов фильтрации, описанных ниже).

3.5.1.2. При последующих запусках

При последующих запусках утилита указывает серверу XStream позицию LCR, начиная с которой она ожидает получение данных. Эта позиция вычисляется запросом

```
SELECT min (lastcommitPos) from ora2pg_slots_t where xout=<Xout>
```

Т.е. извлекается минимальная позиция последней зафиксированной транзакции в слотах.

Сервер XStream начнёт передавать данные с первой позиции, которая превосходит вычисленную этим запросом. Слоты, у которых позиция выше вычисленной будут игнорировать лишние начальные транзакции.

3.5.2. При репликации через OLR

3.5.2.1. При первом запуске

При первом запуске утилита вычисляет начальный SCN и передаёт его в OLR. OLR начнёт захват изменений начиная с наименьшего SCN, который превосходит или равен переданному.

Вычисление начального SCN выполняется следующим образом. Если указан параметр конфигурации startSCN, то используется его значение. Если startSCN не указан, то выполняется запрос `select next_change# from v$log order by sequence# desc` и берётся значение в первой полученной строке.

3.5.2.2. При последующих запусках

При последующих запусках OLR-сервер уже будет знать минимальный SCN транзакций подтверждённых утилитой и применённых в целевой БД. И начнёт захват со следующего SCN. Слоты, у которых SCN выше будут игнорировать лишние начальные транзакции.

OLR запоминает SCN в файле checkpoint.bin. Если он был удалён, то будет использован SCN полученный от утилиты при её подключении к OLR. Утилита вычисляет SCN следующим образом `SELECT min(lastcommitSCN) from ora2pg_slots_t where xout=<Xout>`

Т.е. извлекается минимальный SCN последней зафиксированной транзакции в слотах.

3.6. Структура таблиц для мониторинга и управления репликацией

При запуске репликации, утилита создаёт в служебном Postgres несколько таблиц, если их нет. Эти таблицы позволяют пользователю выполнять мониторинг и управление репликацией. Пользователь может разработать собственное приложение с GUI-интерфейсом, опираясь на знание их структуры. Или обратиться к поставщику утилиты для получения такого приложения. Можно просто с частотой в 1 сек выполнять запросы к `ora2pg_proc`, `ora2pg_slots`, `ora2pg_stats` и отображать их результаты на экране.

`ora2pg_proc` – состояние процесса репликации

column_name	Description
xout	Xstream server
start_time	Время старта
duration	Продолжительность работы (interval)
arclog_size	Суммарный размер архивных журналов Oracle в байтах

jvm_mem_total	Память в распоряжении JVM в байтах
jvm_mem_used	Используемая память JVM в байтах
jvm_mem_used_pct	% используемой памяти JVM
status	Состояние процесса: ACTIVE - выполняется обработка LCR ERROR - процесс остановлен из-за ошибки ABORT - процесс срочно остановлен по команде пользователя STOP - процесс планово остановлен по команде пользователя IDLE - простой STARTING - процесс в состоянии запуска STOPPING - процесс в состоянии плановой остановки RESTARTING - процесс в состоянии перезапуска PAUSE – приостановка чтения из XStream SEEK – репликатор игнорирует LCR с SCN ниже startSCN
command	Команда пользователя ABORT- внеплановое прекращение работы утилиты STOP - плановое прекращение работы утилиты RESTART – внеплановый быстрый перезапуск репликации STOP_AND_RESTART – плановый быстрый перезапуск репликации CLEAR_HISTORY – очистка истории во всех слотах RESET_COUNTERS – установка в ноль статистики во всех слотах RESET_PGDMSTATS – удаление накопленной статистики DML из памяти утилиты и из таблицы ora2pg_pgdmstats SET_PARAMS – считать и применить параметры из ora2pg_proc_par PAUSE – приостановка процесса чтения сообщений из Oracle XStream CONTINUE – возобновление чтения сообщений из Oracle XStream
response	Ответ на команду пользователя
Error	Ошибка процесса
logmnr_state	Состояние сеанса LogMiner по V\$LOGMNR_SESSION
capture_state	Состояние процесса захвата по V\$XSTREAM_CAPTURE
xout_state	Состояние процесса XStream out по V\$XSTREAM_OUTBOUND_SERVER
Version	Информация о версии Ora2PgSyncX
config_name	Имя конфигурационного файла, с которым стартовала репликация
Pid	PID процесса репликации в ОС, в которой он запущен Т.е. это идентификатор процесса java.

При запуске утилита считывает конфигурацию из replication.json, дополняет её значениями по умолчанию, если какой-либо параметр не указан в файле. После чего параметры управления репликацией записываются в таблицу ora2pg_proc_par. Параметр не записывается в эту таблицу, если он там уже есть в результате предыдущих запусков. После этого выполняется чтение и применение параметров из таблицы. Таким образом, параметры, настроенные пользователем в таблице, доминируют над умолчаниями и параметрами конфигурационного файла.

Ora2pg_proc_par

column_name	description
Xout	XStream server
param_code	Код параметра

param_value	Значение параметра
param_type	Тип данных параметра
Minv	Минимальное возможное значение
Maxv	Максимальное возможное значение

Ora2pg_slots – состояние слотов репликации

column_name	Description
Xout	XStream server
slot_name	Имя слота
Status	Состояние: ACTIVE - процесс читает DML из очереди и применяет их в Postgres IDLE - простой (очередь DML пуста) ERROR - процесс в состоянии ошибки (либо умер, либо ждёт реакции пользователя на ошибку) ABORT - процесс срочно остановлен STOP - процесс остановлен штатно STARTING - процесс запускается SEEK – после запуска утилиты слот игнорирует LCR с позицией ниже стартовой (lastcommitpos из ora2pg_slots целевой БД)
in_bytes	Количество байтов записанных в очередь процессом чтения из Oracle
in_commits	Количество операторов COMMIT, записанных в очередь процессом чтения из Oracle
in_idle_commits	Количество commit холостого хода, записанных в очередь процессом чтения из Oracle
in_dmls	Количество операторов insert/update/delete записанных в очередь процессом чтения из Oracle
q_bytes	Количество байтов в очереди
q_commits	Количество операторов COMMIT в очереди
q_idle_commits	Количество commit холостого хода в очереди
q_dmls	Количество insert/update/delete в очереди
out_bytes	Количество байтов переданных в целевую БД
out_commits	Количество операторов COMMIT выполненных в целевой БД
out_idle_commit	Количество commit холостого хода выполненных в целевой БД
out_dmls	Количество insert/update/delete выполненных в целевой БД
discard_dmls	Количество DML, из числа out_dmls, отвергнутых запросами фильтрации. Для DUMMY discard_dmls=out_dmls.
discard_bytes	Количество байтов в DML отвергнутых запросами фильтрации, из числа out_bytes. Для DUMMY discard_bytes=out_bytess.
lastcommitoriginaldate	Время завершения транзакции в исходной БД
Lastcommitpos	LCR позиция последнего commit, который выполнен в целевой БД в т.ч. commit холостого хода
Lastcommitscn	SCN последнего commit, который выполнен в целевой БД в т.ч. commit холостого хода
History	Информация о последних 50-ти операциях в целевой БД Формат: hh:mm:ss.ms:[DISCARD] DML_TYPE schema.table (n1)(n2)(n3)[(n4)] - hh:mm:ss.ms – время суток, когда это изменение

	произошло в исходной БД - DISCARD – если DML отвергнут условием фильтрации - DML_TYPE – INSERT/UPDATE/DELETE/TRUNCATE - schema – имя схемы в целевой БД - table - имя таблицы в целевой БД - n1 – количество DML в batch - n2 - количество байт в batch - n3 – продолжительность выполнения в ms - n4 – продолжительность выполнения запроса фильтрации
rate_out_bytes	Текущая скорость выполнения операций в целевой БД + скорость отбрасывания запросами фильтрации (байт/сек)
when_status_changed	Время последнего изменения состояния
repeat_mode	Находится ли процесс в состоянии повтора транзакции при обработке ошибки
pgsession_state	Текущее состояние сеанса в целевой БД
Lag	Временной лаг в миллисекундах между исходной и целевой БД по последнему применённому в целевой БД commit.
executeBatchDuration	Длительность в ms выполнения в целевой БД текущего batch. Пусто если сейчас ничего не выполняется.
seek_scn	SCN последнего DML, полученного когда слот находится в состоянии SEEK (пропускает ранее применённые DML).
rate_in_bytes	Текущая скорость записи данных из Oracle в очередь в байт/сек

Ora2pg_stats

column_name	description
xout	XStream server
metrica	Метрика
value	Значение

Пример выборки из ora2pg_stats:

```
select ora2pg_stats.xout "XOUT" ,ora2pg_stats.metrica "METRICA" ,ora2pg_stats.value
"VALUE" ,ora2pg_stats.value::numeric/(case when ora2pg_stats.metrica ilike '%bytes%' then 1024 else
1000 end) "VALUE_D_1K" ,ora2pg_stats.value::numeric/(case when ora2pg_stats.metrica ilike
'%bytes%' then 1048576 else 1000000 end) "VALUE_D_1M" from public.ora2pg_stats ora2pg_stats
```

xout	metrica	value	value/1K	value/1M	Описание
ora2pg	InBytes	32 392 094	31 632,904	30,892	Помещено в очередь
ora2pg	InCommits	1 760	1,760	0,002	
ora2pg	InIdleCommits	8	0,008	0,000	
ora2pg	InDMLs	1 706 354	1 706,354	1,706	
ora2pg	InRate(bytes/sec)		0,000	0,000	Скорость наполнения очереди
ora2pg	Q_Bytes	0	0,000	0,000	Находится в очереди
ora2pg	Q_Commits	0	0,000	0,000	
ora2pg	Q_IdleCommits	0	0,000	0,000	
ora2pg	Q_DMLs	0	0,000	0,000	

ora2pg	OutBytes	32 392 094	31 632,904	30,892	Выполнено в Postgres
ora2pg	OutCommits	1 760	1,760	0,002	
ora2pg	OutIdleCommits	8	0,008	0,000	
ora2pg	OutDMLs	1 706 354	1 706,354	1,706	
ora2pg	OutRate(bytes/sec)	0	0,000	0,000	Скорость исчерпания очереди
ora2pg	DiscardBytes	100	0,100	0,010	Отвергнуто байт
ora2pg	DiscardDMLs	10	0,010	0,001	Отвергнуто DML
ora2pg	Lag(ms)	7 565	7,565	0,008	Лag отставания целевой БД от исходной БД
ora2pg	DiscardQueriesResultCacheAttempts	324 234	324,234	0,324	Количество поисков в кэше результатов запросов фильтрации
ora2pg	DiscardQueriesResultCacheHits	272 342	272,342	0,272	Количество успешных поисков в кэше результатов запросов фильтрации
ora2pg	DiscardQueriesResultCacheHitRatio (%)	84	0,084	0,00	% успешных поисков в кэше результатов запросов фильтрации
ora2pg	ExBatchDur(ms)	70	0	0	Длительность выполнения текущего batch

ora2pg_pgdmstats – статистика выполнения DML в Postgres

Данные в этой таблице очищаются всякий раз при запуске репликации или возобновлении репликации.

column_name	description
*xout	XStream server
*slot_name	Имя слота
*pg_schema	Имя схемы в Postgres
*pg_table	Имя таблицы в Postgres
*dml_type	Тип DML (INSERT/UPDATE/DELETE) или DISCARD_QUERY – затраты

	на выполнение запроса фильтрации DML
cnt	Количество выполненных DML
total_time	Общее время выполнения всех DML
min_time	Минимальная длительность выполнения DML
max_time	Максимальная длительность выполнения DML
first_time	Длительность выполнения первого DML
last_time	Длительность выполнения последнего DML
first_change	Дата и время первого выполнения DML
last_change	Дата и время последнего выполнения DML
bytes	Суммарное количество байт изменённых всеми DML
split	Признак разделения таблицы по слотам репликации

*В этой таблице статистика выполнения SQL-операторов сгруппирована по значениям столбцов отмеченных *. Используя эти данные можно выявлять наиболее затратные по времени операции и дисбаланс нагрузки на слотах. И применить эти данные для изменений рейтинга нагруженности таблиц.

Пример:

```
with gstat as (select pg_schema,pg_table,sum(cnt) cnt
from ora2pgsyncx.ora2pg_pgdmstats group by pg_schema,pg_table)
UPDATE ora2pgsyncx.ora2pg_tab t set ratio=gstat.cnt
from gstat
where t.pg_table=gstat.pg_table and t.pg_schema=gstat.pg_schema
```

3.7. Службная таблица в исходной БД

В исходной БД Oracle в схеме xstrmadmin создаётся таблица ora2pg_stoppoint

В ней два столбца:

- Xout – имя XStream out
- Command – команда утилите для плановой остановки процесса репликации

3.8. Службная таблица в целевой БД

В целевой БД PostgreSQL создаётся службная таблица ora2pg_slots_t.

column_name	Description
xout	XStream out server
slot_name	Slot
lastcommitpos	Last commit LCR position
lastcommitscn	Last commit SCN

По умолчанию таблица создаётся пользователем, заданным в параметре pgusername в схеме public. Для изменения этого можно использовать параметр or2pg_slots_t.

3.9. Ограничения целостности и триггеры целевой БД

Утилита не выполняет отключение внешних ключей, правил перезаписи и триггеров в целевой БД. Вместо этого, для обеспечения многопоточности и параллельности процесса репликации, сеансы в целевой БД имеют параметр session_replication_role = replica. Это указывает целевой БД на необходимость отмены проверки внешних ключей и отмены выполнения правил перезаписи и триггеров. Ограничения – проверки (check constraint) будут проверяться в целевой БД и могут быть

причиной ошибок (Exception) в ходе репликации, если эти ограничения не соответствуют ограничениям исходной БД. Это относится и к первичным и уникальным ключам. Пользователь, которым выполняется репликация в целевую БД должен иметь право установки `session_replication_role = replica`. Это право в PostgreSQL имеет суперпользователь. Или пользователь, которому оно дано оператором

```
grant set on parameter session_replication_role to <имя пользователя>;
(это возможно начиная с 15 –й версии PostgreSQL)
```

Если ограничения внешних ключей целевой БД не соответствуют (превосходят) ограничениям целостности исходной БД, то результатом репликации может стать их скрытое нарушение.

В целевой БД могут потребоваться триггеры и правила перезаписи для преодоления асимметрий структур данных исходной и целевой БД и для реализации репликации по условиям. В этом случае такие триггеры и правила должны быть помечены специальной опцией “enable replica” это делается оператором `alter table`.

3.10. Остановка утилиты репликации

3.10.1. Остановка в результате ошибки репликации

Утилита останавливается по ошибке, если ошибка произошла в процессе чтения данных из Oracle XStream. Утилита останавливается по ошибке, если она произошла в одном из слотов репликации и не является преодолимой с помощью пользователя.

3.10.2. Внеплановая остановка по команде пользователя

Утилита останавливается при нажатии Ctrl+C в консоли.

Утилита останавливается при вводе команды ABORT в `ora2pg_proc.command`.

3.10.3. Плановая остановка по команде пользователя

Плановую остановку следует выполнять:

- в случае завершения процесса репликации как финальной стадии сценария миграции данных из Oracle в Postgres
- в случае необходимости выполнения переконфигурирования утилиты (изменение состава таблиц, их распределения по слотам, изменение структуры таблиц и т.п.)
- в случае необходимости остановки Oracle или пересоздания процессов XStream

Утилита останавливается при вводе команды STOP в `ora2pg_proc.command`.

При этом выполняется оператор `update` в таблице `xstrmadmin.ora2pg_stoppoint` и `commit`. Эта транзакция реплицируется из Oracle в утилиту после всех ранее завершённых транзакций.

Утилита, получив эту транзакцию, направляет её в очереди всех слотов и перестаёт читать новые данные из XStream. Каждый слот, получив из очереди эту транзакцию, понимает, что требуется плановая остановка, перестаёт читать новые операций из очереди и завершается.

Когда завершатся все процессы обслуживания очередей слотов, будет выполнено завершение процесса мониторинга репликации и завершение работы утилиты в целом.

3.10.4. Плановая остановка по команде из внешней системы

Для плановой остановки, внешняя система должна выполнить в служебной БД Postgres оператор

```
update ora2pg_proc set command='STOP' where xout='<имя XStream out>';
commit;
```

После чего дождаться момента, когда оператор

```
select response from ora2pg_proc where xout='<имя XStream out>';
```

вернёт строку "STOP done".

3.11. Возобновление работы утилиты репликации

3.11.1. Внеплановый быстрый перезапуск

Внеплановый быстрый перезапуск следует выполнять:

- в случае возникновения ошибки в процессе слота, после принятия мер по её устранению
- в случае остановки XStream out по внутренней ошибке в одном из процессов Oracle, после принятия мер по её устранению

Внеплановый быстрый перезапуск выполняется при вводе команды RESTART в ora2pg_proc.command.

При этом все слоты перестают обслуживать очереди команд. Процесс чтения из Xstream завершается. Монитор завершается. Утилита прекращает работу. Утилита стартует вновь так, как если бы пользователь запустил replication.bat (sh). Далее читайте следующий раздел.

3.11.2. Плановый быстрый перезапуск

Плановый быстрый перезапуск следует применять:

- в случае необходимости изменения количества слотов репликации для оптимизации производительности
- в случае необходимости изменения состава реплицируемых таблиц
- в случае необходимости переноса неразделяемой таблицы из одного слота в другой
- в случае необходимости объявления разделяемой по слотам таблицы неразделяемой и наоборот

Плановый быстрый перезапуск выполняется в следующем порядке:

- Не останавливая процесс репликации, следует внести изменения в конфигурационный файл replicaton.json, внести изменения в таблицах ora2pg_tab, ora2pg_col. При необходимости изменения конфигурации в связи с появлением новых реплицируемых таблиц можно предварительно изменить настройки в prepare.json и выполнить prepare.sh, и выполнить alter table из скрипта prepare_XadmSL.sql для новых таблиц.
- Ввести команду STOP_AND_RESTART оператором update в ora2pg_proc. После этого утилита выполнит следующие действия. Update в таблице xstrmadmin.ora2pg_stoppoint и commit. В случае репликации через OLR, дополнительно будет выполнен оператор alter system switch logfile. Эта транзакция реплицируется из Oracle в утилиту после всех ранее завершённых транзакций. Утилита, получив эту транзакцию, направляет её в очереди всех слотов и перестаёт читать новые данные из XStream. Каждый слот, получив из очереди эту транзакцию, понимает, что требуется плановая остановка,

перестает читать новые операций из очереди и завершается. Когда завершатся все процессы обслуживания очередей слотов, будет выполнено повторное чтение replication.json, чтение описаний таблиц и столбцов, вычисление минимальной LCR position для возобновления репликации, изменение количества слотов и перераспределение таблиц по слотам. Затем утилита возобновит репликацию начиная с вычисленной LCR position.

3.11.3. Повторный запуск репликации

Возобновление репликации выполняется запуском скрипта replication.bat или replication.sh или в результате внепланового или планового быстрого перезапуска.

Если репликация выполняется через XStream, то утилита выполняет запрос в целевой БД для выявления LCR position, с которой необходимо возобновление репликации.

```
select min(lastcommitpos) from ora2pg_slots where xout='имя XStream out';
```

Затем выполняется подключение к серверу XStream out с указанием позиции.

XStream начнет передавать данные с 1-й позиции превосходящей указанную.

Утилита расшифровывает передаваемые ей LCR и определяет слот, в очередь которого должен быть направлен оператор DML синтезируемый из LCR. Если текущая позиция слота превосходит позицию LCR, то оператор DML будет игнорирован.

Если репликация выполняется через OLR, то OLR будет помнить последний минимальный подтвержденный SCN ранее полученный от утилиты и возобновит репликацию с этого SCN. Если последний SCN слота превосходит SCN в LCR синтезированный по данным от OLR, то оператор DML будет игнорирован.

3.12. Режим отладки репликации

Этот режим позволяет трассировать все события происходящие в утилите:

- Получение LCR
- Классификация LCR
- Ход синтеза оператора DML SQL и набора подставляемых переменных
- Определение имени нужного слота
- Помещение оператора в очередь слота
- Чтение операторов слотами
- Добавление оператора в список batch
- Выполнение batch
- Обработка commit

Отладка может вызвать очень быстрый рост размера файла журнала и таблицы ora2pg_debuglog в обслуживаемой БД.

Структура таблицы журнала отладки:

column_name	description
xout	XStream server (идентификатор экземпляра утилиты)
Id	id записи (первичный ключ)
log_datetime	дата и время выдачи отладочного сообщения
log_message	сообщение пользователю
responce_datetime	Дата и время прочтения (подтверждения) сообщения пользователем

Существует режим пошаговой отладки (DebugBySteps=true). В этом режиме пользователь после каждого сообщения выданного на консоль должен нажать Enter, для выполнения утилитой следующего шага.

Помимо отладки на консоли, может быть разработано GUI – приложение для пошаговой отладки путём ввода даты/времени в строку таблицы ora2pg_debuglog. Если Ваша поставка утилиты включает в себя GUI, то в сервере приложений LUI приложение ORA2PGSYNCX имеет эту функцию.

3.13. Обработка ошибок

Авторы утилиты убеждены в том, что если репликация выполняется в рамках сценария миграции данных из Oracle в Postgres, то ошибки репликации не допустимы. И при их обнаружении необходимо повторить миграцию заново после ликвидации причин ошибок.

Возобновление и продолжение репликации после ошибок, по мнению авторов, допустимо в сценариях тестирования и интеграции систем.

Ошибки могут быть обнаружены утилитой при работе следующих её процессов:

- Replicator – всегда выполняется аварийная остановка утилиты с выдачей ошибки на консоль и в журнал
- Monitor – аналогично Replicator
- Slot – выполняется переход процесса слота в состояние ERROR- ошибка. При этом продолжается работа других слотов и наполнение очереди слота попавшего в состояние ERROR

Ошибка, возникшая при работе слота, фиксируется в файле журнала утилиты и в таблице ora2pg_errlog в служебном PostgreSQL.

В большинстве случаев слот фиксирует ошибки класса SQLException. Т.е. целевая БД или её jdbc - драйвер по тем или иным причинам отказались выполнять операцию.

Помимо этого, в утилите реализована проверка на равенство единице количества уделённых строк одним оператором delete и проверка на равенство единице количества обновлённых строк одним оператором update. Неравенство единице может быть следствием причин:

- содержимое исходной и целевой БД не синхронизировано (не качественная миграция, активность вне процесса репликации и т.п.)
- целевая таблица не имеет первичного или уникального ключа и имеются дубли записей

Для фиксации такого рода ошибок в утилите генерируется собственный SQLException, у которого SQLErrorCode=99990, SQLState="UPDCNT0", если изменено (удалено) ноль записей и SQLState="UPDCNT2", если изменено (удалено) две или более записи.

Необходимо отметить, что обычное исключение уровня SQL (SQLException) всегда приводит к откату (rollback) в PostgreSQL. Ошибки неравенства единице не вызывают автоматического отката транзакции. Если пользователь или утилита (в соответствии с правилами) примет решение об игнорировании ошибки неравенств единице, то необходимости в выполнении отката транзакции не возникает.

Структура таблицы ora2pg_errlog.

column_name	Description
xout	XStream server (идентификатор экземпляра утилиты)
slot_name	Slot
err_timestamp	Дата и время ошибки
sql_code	SQL error code
sql_state	SQL error state

ora_schema	Имя схемы Oracle
ora_table	Имя таблицы Oracle
pg_schema	Имя схемы Postgres
pg_table	Имя таблицы Postgres
pg_constraint	Нарушенное ограничение целостности в Postgres
dmltype	Тип DML: INSERT/UPDATE/DELETE/COMMIT
message	Error message
srcstmt	SQL оператор для подстановки переменных batch
synstmt	Синтезированный полный текст SQL оператора
noinbatch	Порядковый номер оператора в batch
noinbind	Прядковый номер bind-переменной (если ошибка вызвана в процессе bind)
position	LCR position
scn	SCN
skip_by_rule_no	ошибка игнорирована в соответствии с правилом указанного номера
repeat_tx_by_rule_no	ошибка вызвала повторение транзакции в соответствии с правилом указанного номера
phase	фаза возникновения ошибки: - prepare statement – подготовка оператора SQL в целевой БД - bind – подстановка переменной в подготовленный оператор - add batch – добавление подготовленного оператора в batch (пачку) - execute batch – выполнение batch - update SCN and position – сохранение SCN и LCR position в ora2pg_slots целевой БД перед выполнением commit - commit – выполнение commit в целевой БД - close batch – закрытие batch после успешного выполнения
batchdmls	Количество DML в batch
batchbytes	Количество байтов в batch
txdmls	Количество DML от начала транзакции
txbytes	Количество байт от начала транзакции
pgsession_state	Состояние сеанса в целевой БД
repeat_mode	Режим повтора транзакции (т.е. произошла ли ошибка, когда слот повторял транзакцию из-за ранее обнаруженной ошибки)
stack_trace	Стек вызовов в JVM
repeat_possible	Возможна ли повторная обработка транзакции
skip_possible	Возможность игнорировать ошибку
user_decision	Решение пользователя: A-abort program R-restart program r-repeat transaction s-skip error (ignore)
pg_maintainer_pid	backend pid в служебном Postgres
id	id записи (первичный ключ)

В зависимости от фазы обработки DML возможны различные варианты реакции на ошибку.

Фаза	Возможность повторения транзакции	Возможность игнорировать/повторить ошибочный оператор	Возможные действия пользователя А (Abort)- аварийное завершение утилиты
------	-----------------------------------	---	--

			R(Restart) – быстрый перезапуск утилиты r (repeat)– повторение транзакции без изъятия оператора вызвавшего ошибку s (skip)– повторение транзакции с изъятием оператора вызвавшего ошибку
prepare statement	X		A R r
Bind	X		A R r
Add batch	X		A R r
Execute batch	X	X	A R r s
Update SCN and position	X		A R r
Commit	X		A R r
Close batch	X		A R r
Set ORA2PG.DISCARD			A R
Get ORA2PG.DISCARD			A R
Execute discard query			A R

Для ввода варианта действий пользователь может использовать консоль. При возникновении ошибки на консоли после сообщения об ошибке появляется приглашение ко вводу решения пользователя.

Однако, такое управление не является удобным. Вы можете создать собственное GUI-приложение для отображение последней ошибки и ввода решения в поле ora2pg_errlog.user_decision. Реакция на такой ввод будет возможна, если в служебном Postgres жив сеанс, указанный в ora2pg.pg_maintainer_pid. Если Ваша поставка утилиты включает в себя GUI, то в сервере приложений LUI приложение ORA2PGSYNCX имеет эту функцию.

Пользователь может настроить правила автоматической обработки ошибок.

Эти правила настраиваются в таблице ora2pg_errhndl в служебном Postgres.

Структура таблицы ora2pg_errhndl

column_name	description
xout	XStream server (идентификатор экземпляра утилиты)
Id	id правила (первичный ключ)
rule_no	Порядковый номер правила
slot_name	Слот
ora_schema	Схема
ora_table	Таблица
dml_type	Тип DML INSERT/UPDATE/DELETE
sqlcode	SQL Error Code
sqlstate	SQL Error State
pg_constraint	Ограничение, нарушенное в целевой БД
what_to_do	“s”-автоматическое игнорирование ошибки, т.е. изъятие оператора вызвавшего ошибку из транзакции “r”-автоматическое повторение транзакции без изъятия оператора, вызвавшего ошибку. Если задано “r” и повтор транзакции не будет успешным

	нового, то нового повтора не произойдёт и слот перейдёт в состояние ERROR для обработки ошибки человеком. Автоматическое повторение транзакции может иметь смысл для ошибок возникших из-за слишком долгого выполнения DML (сработал statement_timeout).
notes	Примечание

При возникновении ошибки, для которой существует возможность обработки, процесс слота, до того как попасть в состояние ошибки, проверяет правила игнорирования, заданные пользователем. Проверка выполняется в порядке возрастания номеров правил. Первое встреченное правило вызовет игнорирование ошибки, но с фиксацией её в таблице ora2pg_errlog с заполнением столбца skip_by_rule_no. Большое число ошибок может вызвать бурный рост таблицы ora2pg_errlog. Внутри утилиты не предусмотрены механизмы очистки этой таблицы. При проверке правил сравнению с возникшей ошибкой подлежат только не пустые значения в полях slot_name, ora_schema, ora_table, dml_type, sqlcode, sqlstate, pg_constraint. Пустые значения будут восприниматься как удовлетворяющие условию сравнения с атрибутами возникшей ошибки.

Если ошибка привела к откату транзакции, то для её обработки необходимо повторение всех операторов DML от начала транзакции, за исключением оператора DML, который вызвал ошибку, если её надо игнорировать.

Для обеспечения возможности повторения транзакции и обработки ошибок в утилите предусмотрен специальный механизм. Этот механизм включается параметром ErrorProcessing=true. Если ErrorProcessing=false, то повторение транзакций и обработки ошибок не выполняется. И пользователю будут не доступны никакие действия по реакции на ошибку и утилита аварийно завершит свою работу записав ошибку в таблицу ora2pg_errlog. Суть специального механизма обработки транзакций состоит в следующем. Когда в слоте начинается новая транзакция, в памяти JVM создаётся специальный список. В нём сохраняются все операторы DML, считанные из очереди от момента начала транзакции. Если использованная в JVM память приблизится к 90%, то данные специального списка будут помещаться на диск по адресу, указанному в параметре конфигурации " path4tmpfiles ".

Этот список очищается при каждом успешном выполнении commit. Когда пользователь или утилита, на основании заданных правил принимает решение о необходимости повторения транзакции, слот переходит в режим повторения (repeat mode) . В этом режиме операторы DML извлекаются не из очереди, а из специального списка до его исчерпания. Один из операторов DML при этом может быть проигнорирован и не выполнен в целевой СУБД, в соответствии с правилом или решением пользователя. В процессе обработки одной транзакции режим повтора может возникать более одного раза. Тогда могут быть игнорированы два и более оператора DML. Этот механизм присутствует в утилите в связи с тем, что повторное подключение к Oracle XStream с указанием LCR position последней успешной транзакции выполняется в Oracle слишком долго. Механизм повторения транзакций, реализованный в утилите, требует повышенных затрат оперативной памяти и, в случае больших транзакций – дисковой.

3.14. Репликация данных таблицы по условию

В некоторых случаях может возникать необходимость реплицировать не все строки таблицы, а лишь те, которые удовлетворяют некоторым условиям. Существует три способа реализации таких условий, которые описаны ниже.

3.14.1. Применение правил Oracle XStream

Oracle XStream позволяет задавать т.н. Subset Rules для реплицируемых таблиц. Подробнее об этом написано в <https://docs.oracle.com/en/database/oracle/oracle-database/12.2/xstrm/managing-xstream-out.html#GUID-EDB7AAE5-8CF8-42AE-8294-8E4136431C18>

Однако, в ходе выполнения многочисленных тестов выявлена ненадёжность вычисления правил. И неоднозначность синтаксиса правил для задания правил репликации оператора update. Т.е. нет возможности указать для поля таблицы участвующего в условии к какой записи относится поле. К старой (до update) или к новой (после update).

Применение Subset Rules может вызвать повышение нагрузки в процессах XStream capture и apply.

3.14.2. Применение правил перезаписи PostgreSQL

О правилах перезаписи подробно написано в <https://postgrespro.ru/docs/postgrespro/17/rules-update>

Пример создания правила в целевом PostgreSQL для предотвращения операций insert если столбец ID меньше нуля:

```
CREATE OR replace rule
app_r_country_ins_neg
AS ON INSERT TO fwork.app_t_country
where id < 0 do instead nothing;
```

Установка для правила, что оно срабатывает в режиме репликации:

```
alter table fwork.app_t_country
enable replica rule app_r_country_ins_neg;
```

Помимо этого, в обслуживаемом Postgres в таблице ora2Pg_tab.ignore_updcnt0 должен быть равен "Y" для соответствующей таблицы. Это ослабляет контроль ошибок в утилите.

Для повышения уровня контроля ошибок правило перезаписи должно увеличивать на 1 значение переменной сеанса ORA2PG.DISCARD, а ora2Pg_tab.ignore_updcnt0 должен быть равен "C".

Ниже приведён пример такого правила перезаписи:

```
CREATE OR replace rule
ab_r_claim_upd_fl
AS ON UPDATE TO fwork.ab_t_claim
where (
select true from (
select true,
set_config('ORA2PG.DISCARD',(coalesce(nullif(current_setting('ORA2PG.DISCARD',true),''), '0')::int+1)::text,false)
from fwork.ct_t_contract ct,fwork.cl_t_client cl
where cl.id=ct.client_id and cl.legal_status!='ФЛ' and ct.id=old.contract_id
)q)
do instead nothing;
```

Перед выполнением оператора DML или их пачки (batch) утилита будет выполнять установку переменной сеанса ORA2PG.DISCARD равной нулю:

```
select set_config('ORA2PG.DISCARD', '0',false);
```

После выполнения DML или пачки DML утилита будет считывать значение переменной и добавлять его к общему количеству изменённых строк. Конечно, возможна сложная ситуация, когда в рамках одной пачки (batch) сходных операторов, например update, некоторые операторы будут отвергнуты правилом перезаписи, а другие вызовут изменение более 1 записи, а третьи будучи не отвергнутыми не изменят ни одной записи, и при этом общее количество изменений в сумме с переменной ORA2PG.DISCARD дадут равенство количеству операторов в batch. В этом случае ошибка будет утеряна. И если Вы опасаетесь такой ситуации, то используйте запросы фильтрации см. п. 3.14.4.

Применение правил перезаписи может вызвать повышение нагрузки на стороне целевой БД. Но, потери производительности будут меньше чем, при использовании запросов фильтрации.

3.14.3. Применение условий репликации в утилите

Условие репликации, которое будет проверяться утилитой, можно настроить для любой таблицы в поле condition таблицы ora2pg_tab. Для вычисления значений условий репликации утилита может использовать библиотеку EvalEx (<https://ezylang.github.io/EvalEx/>) или Rhino JavaScript (<https://github.com/mozilla/rhino>). Язык SQL исходной или целевой БД для вычисления условий репликации не эффективен и снижает общую производительность и не применяется. Выбор библиотеки вычисления условия указывается в столбце conditon_engine. Выбор следует делать исходя из личных предпочтений. Производительность обеих библиотек при вычислении условий не имеет существенных отличий. Условие репликации, заданное для таблицы, не может быть основано на данных из других таблиц.

После ввода условий следует повторно выполнить prepare.sh или prepare_scripts.sh. Условие может изменить состав данных, которые должны всегда присутствовать в redo log для любого изменения, добавления или удаления данных. Т.е. условие влияет на оператор “alter table ... add supplemental log...”, который генерируется в файле prepare_XadmSL.sql. Таким образом, условие следует задавать до запуска репликации. Если необходимость применения условий выявлена после запуска репликации, то следует применить плановую остановку репликации, настройку условий, выполнить prepare_scripts.sh, убедиться в отсутствии ошибок в условиях (select * from public.ora2pg_tab where condition_error is not null), выполнить prepare_scripts_XadmSL.sql, и возобновить репликацию. Следует помнить, что изменения в supplemental logging для таблицы будут влиять только на новые операции DML, но не на те, которые уже выполнены и находятся в redo log.

3.14.3.1. Синтаксис условий репликации

Условие это выражение типа boolean или char(1). Если выражение вернёт true, то оператор будет реплицирован. Если выражение вернёт false, то оператор будет направлен в слот DUMMY и не будет выполнен в целевой БД.

О применении выражений типа char(1) написано в следующем разделе.

В качестве переменных подставляемых в выражения могут использоваться данные столбцов (полей) реплицируемой строки таблицы из БД Oracle. Поддерживается участие в выражении столбцов следующих типов:

- CHAR

- VARCHAR2
- NVARCHAR2
- NUMBER
- FLOAT
- BINARY_FLOAT
- BINARY_DOUBLE
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIMESTAMPLTZ

Кроме того, могут использоваться константы INSERT, UPDATE и DELETE они имеют тип Boolean и равны true или false в зависимости от типа dml-оператора. В дополнение к ним может использоваться символьная константа I_U_D. Для оператора insert она равна "I", для update – "U", для delete – "D".

Для ссылок на поле новой записи применяется синтаксис ИМЯ_СТОЛБЦА[NEW] в случае insert и update. Для ссылок на поле старой записи применяется синтаксис ИМЯ_СТОЛБЦА[OLD] в случае update и delete.

В выражении могут применяться следующие константы, операторы и функции

Категория	Код	Описание
Константы		
Basic	NULL (NaN для JavaScript)	Null
Boolean	DELETE	True для delete, false для update и insert
Boolean	FALSE	Ложь
Boolean	INSERT	True для insert, false для update и delete
Boolean	TRUE	Истина
Boolean	UPDATE	Истина для update, ложь - для delete и insert
Char	I_U_D	I или U или D
Операторы (EvalEx и JavaScript)		
Arithmetic	%	Остаток от деления
Arithmetic	*	Умножение
Arithmetic	+	Сложение чисел и конкатенация строк
Arithmetic	-	Вычитание
Arithmetic	/	Деление
Boolean	!	Отрицание для Boolean
Boolean	!=	Не равно
Boolean	&&	Логическое И
Boolean	<	Меньше чем
Boolean	<=	Меньше или равно
Boolean	==	Равно
Boolean	>	Больше
Boolean	>=	Больше или равно
Boolean		Логическое ИЛИ
Операторы JavaScript		
Basic	If (..) ... ;else ...;	Управляющий оператор Если... то ..., иначе ...
Basic	Swch (...) { case ...: ... break; case...: break; default:...}	Управляющий оператор ветвления
Функции EvalEx		

Basic	COALESCE(,)	COALESCE(value1,value2 ...) Возвращает первый непустой параметр, или NULL если все параметры NULL
Basic	IF(, ,)	IF(условие, результат если ИСТИНА, результат если ЛОЖЬ)
Basic	SWITCH(, , , , ,)	SWITCH(expression, value1, result1, [value2-N, result2-N ...], [default]) Возвращает первый подходящий результат соответствующий значению выражения или умолчание если соответствий нет.
Date	DT_DATE_NEW()	DT_DATE_NEW(millis) Returns a new DATE_TIME from the epoch of 1970-01-01T00:00:00Z in milliseconds.
Date	DT_DATE_NEW(, ,)	DT_DATE_NEW(year, month, day [,hour, minute, second, millis, nanos] [,zoneId]) Returns a new DATE_TIME value with the given parameters. An optional time zone (string) can be specified, e.g. "Europe/Moscow", or "GMT+03:00". If no zone id is specified, the system zone id is used.
Date	DT_DATE_TO_EPOCH()	DT_DATE_TO_EPOCH(date_time) Converts the given DATE_TME to epoch timestamp in millisecond.
Date	DT_DURATION_FROM_MILLIS()	DT_DURATION_FROM_MILLIS(millis) Returns a new DURATION value with the given milliseconds.
Date	DT_DURATION_NEW()	DT_DURATION_NEW(days [,hours, minutes, seconds, nanos]) Returns a new DURATION value with the given parameters.
Date	DT_DURATION_TO_MILLIS(duration)	DT_DURATION_TO_MILLIS(duration) Converts the given duration to a milliseconds value.
Date	DT_NOW()	Produces a new DATE_TIME that represents the current moment in time.
Date	DT_TODAY()	DT_TODAY([zoneId]) Produces a new DATE_TIME that represents the current date, at midnight (00:00). An optional time zone (string) can be specified, e.g. "Europe/Moscow", or "GMT+03:00". If no zone id is specified, the system zone id is used.
Number	ABS()	Модуль (абсолютное значение для чисел!=0 или 0)
Number	CEILING()	CEILING(2.3)=3 Округление до целого в большую сторону
Number	FLOOR()	FLOOR(3.7)=3 Округление до целого в меньшую сторону
Number	MAX(,)	MAX(value1,value2 ...) Максимум среди указанных значений
Number	MIN(,)	MIN(value1,value2 ...) Минимум среди указанных значений
Number	ROUND(,)	ROUND(2.5, 0)=3 Округление по правилам математики до указанной точности.
Number	SUM(,)	SUM(value1,value2 ...) Сумма значений
String	STR_CONTAINS(,)	STR_CONTAINS(string, substring) ИСТИНА если строка содержит подстроку
String	STR_ENDS_WITH(,)	STR_ENDS_WITH(string, substring) ИСТИНА если

		строка оканчивается подстрокой
String	STR_LEFT(,)	STR_LEFT(string, n) Первые n символов строки
String	STR_LENGTH()	STR_LENGTH(string) Длина строки
String	STR_LOWER()	STR_LOWER(string) Строка в нижнем регистре
String	STR_MATCHES(,)	STR_MATCHES(string, pattern) Истина если строка соответствует регулярному выражению
String	STR_RIGHT(,)	STR_RIGHT(string, n) Последние n символов
String	STR_SPLIT(,)	STR_SPLIT(string, separator) Разделяет строку на массив строк по указанному разделителю
String	STR_STARTS_WITH(,)	STR_STARTS_WITH(string, substring) Истина если строка начинается с подстроки
String	STR_SUBSTRING(, ,)	STR_SUBSTRING(string, start[, end]) Выделяет подстроку с указанной начальной позиции [по указанную конечную].
String	STR_TRIM()	STR_TRIM(string) Удаляет левые и правые пробелы строки
String	STR_UPPER()	STR_UPPER(string) Строка в верхнем регистре
Функции JavaScript		
Date	new Date()	new Date(millis) Returns a new DATE_TIME from the epoch of 1970-01-01T00:00:00Z in milliseconds.
Date	new Date(, ,)	new Date(year, month(0-11), day [,hour, minute, second, millis) Returns a new DATE_TIME value with the given parameters. An optional time zone (string) can be specified, e.g. "Europe/Moscow", or "GMT+03:00". If no zone id is specified, the system zone id is used.
Number	Math.abs()	Модуль (абсолютное значение для чисел!=0 или 0)
Number	Math.ceil()	Math.ceil(2.3)=3 Округление до целого в большую сторону
Number	Math.floor()	Math.floor(3.7)=3 Округление до целого в меньшую сторону
Number	Math.max(,)	Math.max(value1,value2 ...) Максимум среди указанных значений
Number	Math.min(,)	Math.min(value1,value2 ...) Минимум среди указанных значений
Number	Math.round(,)	Math.round(2.5, 0)=3 Округление по правилам математики до указанной точности
Number	Math.sum(,)	Math.sum(value1,value2 ...) Сумма значений
String	.endsWith()	string.endsWith(substring) ИСТИНА если строка оканчивается подстрокой
String	.includes()	string.includes(substring) ИСТИНА если строка содержит подстроку
String	.length	string.length Длина строки
String	.match()	string.match(pattern) Истина если строка соответствует регулярному выражению
String	.split(" ")	string.split(separator) Разделяет строку на массив строк по указанному разделителю
String	.startsWith(" ")	string.startsWith(substring) Истина если строка начинается с подстроки
String	.substring(,)	string.substring(start[, end]) Выделяет подстроку

		с указанной начальной позиции [по указанную конечную].
String	.toLowerCase()	string.toLowerCase() Строка в нижнем регистре
String	.toUpperCase()	string.toUpperCase() Строка в верхнем регистре
String	.trim()	string.trim() Удаляет левые и правые пробелы строки

Пример условия репликации EvalEx или JavaScript:

```
(INSERT && DEPT_NO[NEW]=="40")
```

```
||
```

```
((DELETE | UPDATE) && DEPT_NO[OLD]=="40")
```

3.14.3.2. Трансформация оператора UPDATE

В некоторых случаях может возникнуть необходимость преобразований оператора UPDATE в оператор DELETE или в оператор INSERT. Пример: в целевой БД должны оказаться все данные таблицы EMPLOEE по определённому признаку. На пример SALARY>100000. Если для какой-то существующей записи это условие было ложным, а в результате выполнения UPDATE становится истинным, то новая запись должна появиться в целевой БД. И, наоборот, если это условие было истинным, а становится ложным, то запись должна исчезнуть из целевой БД.

Для таких случаев выражение условия репликации должно возвращать символьное значение char(1). Допускаются следующие символы.

Оператор	Символ (результат вычисления выражения)	Действие в целевой БД
INSERT	N	-
INSERT	I	INSERT
DELETE	N	-
DELETE	D	DELETE
UPDATE	N	-
UPDATE	U	UPDATE
UPDATE	I	INSERT
UPDATE	D	DELETE

Пример выражения EvalEx:

```
SWITCH( I_U_D,
  "I", IF(SALARY[NEW]>100000,"I","N") ,
  "U", IF(SALARY[NEW]>100000 && SALARY[OLD]>100000,"U",IF(SALARY[NEW]<=100000 &&
SALARY[OLD]<=100000,"N",IF(SALARY[OLD]>100000 && SALARY[NEW]<=100000,"D","I"))),
  "D", IF(SALARY[OLD]>100000,"D","N")
)
```

Пример выражения JavaScript:

```
switch (I_U_D){
  case "I":
    if (SALARY[NEW]>100000) "I";else "N";
    break;
  case "D":
    if (SALARY[NEW]>100000) "D";else "N";
    break;
  case "U":
    if (SALARY[NEW]>100000 && SALARY[OLD]>100000)"U";
    else if SALARY[NEW]<=100000 && SALARY[OLD]<=100000)"N";
    else if (SALARY[OLD]<=100000 && SALARY[NEW]>100000)"I";
    else "D";
```

```
break;
default: "N";
}
```

3.14.3.3. Использование констант заданных пользователем

Используя оператор SELECT выполняемый в исходной БД, пользователь может создать собственные константы и ссылаться на них в выражениях условий репликации таблиц. Константы получают свои значения один раз перед началом репликации при каждом её запуске и не могут быть изменены без остановки процесса репликации. Пользовательские константы задаются в таблице ora2pg_evalex_tems_u. Структура таблицы:

Столбец	Тип данных	Описание
Item	Varchar(32)	Первичный ключ. Уникальный код константы в верхнем регистре. Может содержать латинские буквы, цифры и знак подчёркивания. Должен начинаться с буквы. Если код совпадает с именем столбца реплицируемой таблицы, для которой задано условие репликации это вызовет ошибку.
Description	Text	Описание – произвольный текст.
Query	Text	Оператор Select, который извлекает одну строку и один столбец. Извлекаемое значение должно быть одного из поддерживаемых типов, допускаемых в выражении. Запрос выполняется в исходной БД от имени пользователя задаваемым параметром orausername.

Использование пользовательских констант особенно полезно когда выражение условия репликации необходимо построить на столбцах типов DATE, TIMESTAMP, TIMESTAMPTZ, Timestampltz, когда данные реплицируемой строки надо сравнить с какой-то конкретной датой.

Пример:

Необходимо реплицировать данные таблицы, введённые после 6-го июня 2025г.

Пользовательская константа:

ITEM	DESCRPTON	QUERY
D2025_06_05	2025-06-05	Select to_date('2025.06.05','yyyy.mm.dd') from dual

Выражение:

```
(INSERT && ENTER_DATE[new]>D2025_06_25)
```

```
||
```

```
(UPDATE && ENTER_DATE[old]>D2025_06_25)
```

||

(DELETE && ENTER_DATE[old]>D2025_06_25)

3.14.4. Применение запросов фильтрации в утилите

Условия репликации, задаваемые для таблицы в утилите, описанные выше не позволяют реализовать условия зависящие, от данных в других таблицах. Для преодоления этого ограничения предусмотрены запросы фильтрации DML. В столбце discard таблицы ora2pg_tab можно ввести запрос, который будет опираться на данные реплицируемых строк и извлекающий данные из других таблиц. В зависимости от того вернёт или не вернёт запрос хотя бы одну строку, утилита будет принимать решение о необходимости применения DML в целевой БД. Запросы фильтрации выполняются процессами обработки очередей, т.е. в слотах, а не в репликаторе.

Запросы могут выполняться в целевой БД (Postgres) или в БД-источнике (Oracle). Если запрос выполняется в Oracle, то слот будет создавать дополнительный отдельный сеанс в Oracle. Если запрос выполняется в Postgres, то слот будет использовать имеющийся сеанс в БД Postgres. Вполне нормальной является ситуация, когда для некоторых таблиц запросы фильтрации выполняются в Postgres, в то время, как для других – в Oracle.

Запрос должен возвращать не менее 1 столбца произвольного типа. Возвращаемое значение не анализируется. Запрос может содержать ссылки на столбцы таблицы целевой БД.

Пример. Пусть реплицируемая таблица “ab_t_claim” имеет столбец “contract_id”, реализующий внешний ключ на таблицу “ct_t_contract”. А она имеет столбец “client_id” – внешний ключ на таблицу “cl_t_client”. Тогда, запрос отбрасывающий DML по значению столбца legal_status таблицы cl_t_client может быть написан так:

```
select 1
from fwork.ct_t_contract ct,fwork.cl_t_client cl
where cl.id=ct.client_id and cl.legal_status!='ФЛ' and ct.id=:contract_id
```

В этом запросе “:contract_id” – ссылка на столбец таблицы “ab_t_claim”.

Ссылки допустимы на столбцы целевой таблицы при условии, что им соответствуют столбцы следующих типов Oracle:

- CHAR
- VARCHAR2
- NVARCHAR2
- NUMBER
- FLOAT
- BINARY_FLOAT
- BINARY_DOUBLE
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIMESTAMPLTZ

Отброшенные DML учитываются в статистике слота (таблица ora2pg_slots) в столбцах out_dmls,out_bytes и дополнительно в discard_dmls и discard_bytes. Аналогично всем DML проходящим через слот DUMMY.

В случае UPDATE запрос фильтрации может опираться либо на данные старой записи или на данные новой записи. Это настраивает пользователь в столбце discard_upd_src. ‘O’ – данные старой записи. ‘N’-данные новой записи. Запрос фильтрации не позволяет трансформировать оператор UPDATE в операторы INSERT или DELETE.

Несмотря на то, что запросы фильтрации выполняются многопоточно в параллельных процессах обслуживания очередей, они существенно снижают производительность. Даже больше чем правила перезаписи. Это происходит не только по причине возможно не эффективного запроса, но и по причине увеличения количества обращений к БД со стороны утилиты. Результаты запросов фильтрации могут быть кэшированы утилитой в целях сокращения количества обращений к БД и повышения производительности. Как правило, затраты памяти JVM на кэширование одного результата запроса фильтрации не превышают 20 байт. Каждый результат хранимый в кеше состоит из ключа и значения. Значение – это объект типа Boolean. Оно показывает возвращает запрос хотя бы одну строку или нет. Ключ состоит из набора полей:

- Уникальный номер запроса фильтрации
- Значения переменных подстановки в запрос (`ora2pg_tab.discard_binds`)

Все запросы фильтрации получают уникальный номер. Как правило запросов, меньше чем таблиц, в которых они указаны. Для выявления набора уникальных запросов утилита сравнивает их, предварительно приведя текст запроса к jdbc-требованиям (т.е. со знаками вопросов вместо имён подставляемых переменных). Размер кэша регулируется параметром `DiscardQueryesResultCache`, который задаёт максимальное количество результатов хранимых в кэше. По умолчанию этот параметр равен нулю. Перед тем как процесс обслуживания очереди слота выполнит запрос фильтрации, он проверит наличие готового результата в кэше. Если он будет найден, то запрос фильтрации не выполняется. И будет использовано значение из кэша. Если результата в кэше нет, то запрос выполняется и его результат сохраняется в кэше. При этом, если размер кэша близок к максимальному, то происходит удаление результата редко используемого в последнее время.

Для всех слотов и для всех запросов фильтрации используется один общий кэш результатов. Производительность общего потокобезопасного кэша выше нескольких отдельных кэшей с делением по слотам или запросам. И затраты памяти ниже. Запись в кэше порождённая одним слотом может быть использована другими слотами. Вполне нормальной является ситуация, когда максимальный размер кэша установлен равным нескольким миллиардам. Естественно, это требуется учитывать в значении параметра `-Xmx` запуска JVM в `replication.sh (.bat)`. Несмотря на кэширование, затраты на выполнение запросов фильтрации могут оставаться высокими. Особенно сразу после запуска, пока кэш не заполнен.

Тем не менее, запросы фильтрации представляют важную функциональность, которая позволяет, в том числе, обходить аномалии контролируемой рассинхронизации исходной и целевой баз данных. Причём, без понижения уровня контроля за ошибками репликации.

После ввода запросов фильтрации следует повторно выполнить `prepare.sh` или `prepare_scripts.sh`. Ссылки на столбцы могут изменить состав данных, которые должны всегда присутствовать в redo log для любого изменения, добавления или удаления данных. Т.е. запрос фильтрации влияет на оператор `“alter table ... add supplemental log...”`, который генерируется в файле `prepare_XadmSL.sql`. Таким образом, запросы следует вводить до запуска репликации. Если необходимость применения запросов фильтрации выявлена после запуска репликации, то следует применить плановую остановку репликации, настройку запросов, выполнить `prepare_scripts.sh`, убедиться в отсутствии ошибок в условиях (`select * from public.ora2pg_tab where discard_error is not null`), выполнить `prepare_scripts_XadmSL.sql`, и возобновить репликацию. Следует помнить, что изменения в supplemental logging для таблицы будут влиять только на новые операции DML, но не на те, которые уже выполнены и находятся в redo log.

3.15. Трансформация данных

3.15.1. Применение функций PostgreSQL

В обслуживаемом PostgreSQL в таблице описаний столбцов реплицируемых таблиц, в столбце pg_func можно настроить преобразование данных исходной БД в данные целевой БД.

Пример для конвертирования Oracle CHAR(1) в Postgres boolean:

```
case when ?='Y' then true else false end
```

3.15.2. Применение правил перезаписи и триггеров PostgreSQL

Для того, чтобы триггеры и правила перезаписи работали в целевой БД в период репликации данных следует для нужных таблиц выполнить:

```
alter table <имя схемы>.<имя таблицы>
enable replica|always trigger <имя триггера>;
```

```
alter table <имя схемы>.<имя таблицы>
enable replica|always rule <имя правила>;
```

3.16. XStream - репликация данных из одной исходной СУБД в несколько целевых СУБД

Утилита позволяет выполнять репликацию из одной исходной СУБД Oracle в две или более целевых СУБД Postgres. Это выполняется за счёт запуска нескольких экземпляров утилиты.

При этом возможно пересечение множеств таблиц реплицируемых в разные целевые БД.

В данном разделе описан пример конфигурации серверов и настроек утилиты для репликации в две целевых БД. Этот пример демонстрирует следующие возможности:

- оптимизация нагрузки на целевую СУБД
- независимое управление процессами репликации
- снижение затрат на служебный Postgres.

3.16.1. Постановка задачи

В исходной БД Oracle в схеме "DOC" имеются следующие таблицы:

DOC TABLE

Имя столбца	Тип данных	Перв. ключ
DOC_NO	NUMBER(20.0)	+
DOC_KEY	VARCHAR2(10)	
DOC_TEXT	VARCHAR2(1000)	
NOTES	VARCHAR2(1000)	

CLIENT TABLE

Имя столбца	Тип данных	Перв. ключ
CLIENT_NO	NUMBER(20.0)	+
CLIENT_NAME	VARCHAR2(100)	
CLIENT_ADR	VARCHAR2(1000)	
NOTES	VARCHAR2(1000)	

В первой целевой БД Postgres в схеме "doc" имеются следующие таблицы:

doc table

имя столбца	тип данных	перв. ключ
-------------	------------	------------

doc_no	number(20.0)	+
doc_key	varchar2(10)	
doc_text	varchar2(1000)	
notes	varchar2(1000)	

client table

имя столбца	тип данных	перв. ключ
client_no	number(20.0)	+
client_name	varchar2(100)	
client_adr	varchar2(1000)	
notes	varchar2(1000)	

Во второй целевой БД Postgres в схеме "doc" имеется таблица:

doc table

имя столбца	тип данных	перв. ключ
doc_no	number(20.0)	+
doc_key	varchar2(10)	+
doc_text	varchar2(1000)	
notes	varchar2(1000)	

Необходимо настроить репликацию обеих таблиц в первую и одну таблицу во вторую целевые БД. Следует обратить внимание на то, что во второй целевой БД таблица doc_table имеет иной первичный ключ. Такая

3.16.2. Решение задачи

3.16.2.1. Создание схем в служебном Postgres

Для разделения данных о реплицируемых таблицах необходимо создать две схемы в служебном Postgres.

```
create schema target1;
```

```
create schema target2;
```

3.16.2.2. Подготовка файлов

Необходимо настроить программы подготовки и запуска репликации. Выполним копирование файлов, создав их дополнительные экземпляры для двух репликаций.

Файл общий	Файл для репликации в целевую БД1	Файл для репликации в целевую БД2
prepare.sh	prepare_target1.sh	prepare_target2.sh
prepare.json	prepare_target1.json	prepare_target2.json
replication.sh	replication_target1.sh	replication_target2.sh
replication.json	replication_target1.json	replication_target2.json

Содержимое файлов:

Файл	Содержимое
prepare_target1.sh	java -Xmx8192m -Dfile.encoding=UTF-8 -jar Ora2PgSyncX.jar prepare_target1.json
prepare_target1.json	{ "oradb": "jdbc:oracle:oci:@orahost:1521:orcl", "orausername": "xstrmadmin", "orapassword": "xstrmadmin", "pgdb": "pg_maintain:5432/postgres", /*служебная БД Postgres*/

	<pre> "pgusername": "postgres", "pgpassword": "postgres", "pgchema": "target1", /*имя схемы в служебной БД Postgres*/ "pgtdb": "pg_target1:5432/postgres", /*целевая БД Postgres*/ "pgtusername": "postgres", "pgtpassword": "postgres", "action": "prepare", "XStreamRuleType": "tables", "xcapture": "ora2pg_capture", /*XStream capture name*/ "xqueue": "ora2pg_queue", /*XStream capture name*/ "xout": "target1", /*XStream outbound server name*/ "slots4split": 2, "slots4notsplit": 2, "query4tables": "ora:select t.OWNER, t.TABLE_NAME, lower(t.OWNER), lower(t.TABLE_NAME), (nvl(t.AVG_ROW_LEN,0)+1)*(nvl(t.NUM_ROWS,0)+1) from sys.dba_tables t where t.OWNER in ('DOC') order by 1,2" } </pre>
prepare_target2.sh	<pre> java -Xmx8192m -Dfile.encoding=UTF-8 -jar Ora2PgSyncX.jar prepare_target2.json </pre>
prepare_target2.json	<pre> { "oradb": "jdbc:oracle:oci:@orahost:1521:orcl", "orausername": "xstrmadmin", "orapassword": "xstrmadmin", "pgdb": "pg_maintain:5432/postgres", /*служебная БД Postgres*/ "pgusername": "postgres", "pgpassword": "postgres", "pgschema": "target2", /*имя схемы в служебной БД Postgres*/ "pgtdb": "pg_target2:5432/postgres", /*целевая БД Postgres*/ "pgtusername": "postgres", "pgtpassword": "postgres", "action": "prepare", "XStreamRuleType": "tables", "xcapture": "ora2pg_capture", /*XStream capture name*/ "xqueue": "ora2pg_queue", /*XStream capture name*/ "xout": "target2", /*XStream outbound server name*/ "slots4split": 1, "slots4notsplit": 1, "query4tables": "ora:select t.OWNER, t.TABLE_NAME, lower(t.OWNER), lower(t.TABLE_NAME), (nvl(t.AVG_ROW_LEN,0)+1)*(nvl(t.NUM_ROWS,0)+1) from sys.dba_tables t where t.OWNER in ('DOC') and t.TABLE_NAME in ('DOC_TABLE') order by 1,2" } </pre>
replication_target1.sh	<pre> java -Xmx8192m -Dfile.encoding=UTF-8 -jar Ora2PgSyncX.jar replication_target1.json </pre>
replication_target1.json	<pre> { "oradb": "jdbc:oracle:oci:@orahost:1521:orcl", "orausername": "xstrmadmin", "orapassword": "xstrmadmin", "pgdb": "pg_maintain:5432/postgres", /*служебная БД Postgres*/ "pgusername": "postgres", "pgpassword": "postgres", "pgschema": "target1", /*имя схемы в служебной БД Postgres*/ </pre>

	<pre>"pgtdb": "pg_target1:5432/postgres", /*целевая БД Postgres*/ "pgtusername": "postgres", "pgtpassword": "postgres", "action": "replication", "xout": "target1", /*XStream outbound server name*/ "slots4split": 2, "slots4notsplit": 2 }</pre>
replication_target2.sh	<pre>java -Xmx8192m -Dfile.encoding=UTF-8 -jar Ora2PgSyncX.jar replication_target2.json</pre>
replication_target2.json	<pre>{ "oradb": "jdbc:oracle:oci:@orahost:1521:orcl", "orausername": "xstrmadmin", "orapassword": "xstrmadmin", "pgdb": "pg_maintain:5432/postgres", /*служебная БД Postgres*/ "pgusername": "postgres", "pgpassword": "postgres", "pgschema": "target2", /*имя схемы в служебной БД Postgres*/ "pgtdb": "pg_target2:5432/postgres", /*целевая БД Postgres*/ "pgtusername": "postgres", "pgtpassword": "postgres", "action": "replication", "xout": "target2", /*XStream outbound server name*/ "slots4split": 1, "slots4notsplit": 1 }</pre>

3.16.2.3. Генерация и выполнение скриптов

После запуска prepare_target1.sh будут сгенерированы скрипты:

1. prepare_target1_SYS.sql – для создания пользователя xstrmadmin, его следует выполнить пользователем с правами DBA. Например пользователем SYS.
2. prepare_target1_XadmCX.sql – для создания процесса XStream capture с именем ora2pg_capture и процесса XStream out с именем target1, который будет реплицировать данные двух таблиц в первую целевую БД. Этот скрипт следует выполнять пользователем xstrmadmin. Ниже содержимое скрипта с комментариями.

```
-- установка имени таблицы AQ и имени очереди AQ для процесса XStream capture
BEGIN DBMS_XSTREAM_ADM.SET_UP_QUEUE(
queue_table => 'xstrmadmin.ora2pg_queue_tab',
queue_name => 'xstrmadmin.ora2pg_queue');END;
/

--создание процесса XStream capture с именем "ora2pg_capture"
BEGIN DBMS_CAPTURE_ADM.CREATE_CAPTURE(source_database=>'ORCL',
queue_name=>'xstrmadmin.ora2pg_queue',
capture_name=>'ora2pg_capture',
capture_class=>'xstream');END;
/

--установка параметров для XStream capture
BEGIN
DBMS_CAPTURE_ADM.include_extra_attribute(capture_name=>'ora2pg_capture',attribute_name=>'row_id',include=>true);
DBMS_XSTREAM_ADM.ENABLE_GG_XSTREAM_FOR_STREAMS(enable=>true);
DBMS_CAPTURE_ADM.SET_PARAMETER(capture_name=>'ora2pg_capture',parameter=>'capture_idkey_objects',value=>'Y');
DBMS_CAPTURE_ADM.SET_PARAMETER(capture_name=>'ora2pg_capture',parameter=>'inline_lob_optimization',value=>'Y');
--DBMS_CAPTURE_ADM.SET_PARAMETER(capture_name=>'ora2pg_capture',parameter=>'parallelism',value=>'2');
END;
```

```

/
-- создание процесса XStream out с именем target1 для репликации данных двух таблиц в первую целевую БД
DECLARE
tables DBMS_UTILITY.UNCL_ARRAY;
schemas DBMS_UTILITY.UNCL_ARRAY;
BEGIN
tables(1):='xstrmadmin.ora2pg_stoppoint';
tables(2):='DOC.CLIENT_TABLE';
tables(3):='DOC.DOC_TABLE';
schemas(1):=null;
DBMS_XSTREAM_ADM.ADD_OUTBOUND(
server_name => 'target1',
queue_name => 'xstrmadmin.ora2pg_queue',
source_database => 'ORCL',
include_ddl => true,
table_names => tables,
schema_names => schemas);
END;
/

```

3. prepare_target1_XadmSL.sql – для создания в таблицах исходной БД групп дополнительного журналирования данных в redolog – файлах. Группы включают в себя столбцы необходимые для идентификации строк таблиц первой целевой БД. Этот скрипт надо выполнить пользователем xstrmadmin. Ниже приведено содержимое скрипта с комментариями.

```

ALTER DATABASE ADD SUPPLEMENTAL LOG DATA --добавляет на уровне БД возможность дополнительного журналирования
/
ALTER DATABASE DROP SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE, FOREIGN KEY) COLUMNS --удаляет ненужные утилите
опции журналирования на уровне БД
/
alter table DOC.CLIENT_TABLE add supplemental log group target1_00001(CLIENT_NO) always -- группа для client_table
/
alter table DOC.DOC_TABLE add supplemental log group target1_00002(DOC_NO) always -- группа для doc_table
/

```

После запуска prepare_target2.sh будут сгенерированы скрипты:

1. prepare_target2_SYS.sql – для создания пользователя xstrmadmin, его содержимое будет совпадать с prepare_target1_SYS.sql. Его уже не требуется выполнять.
2. prepare_target2_XadmCX.sql – для создания процесса XStream capture с именем ora2pg_capture и процесса XStream out с именем target2, который будет реплицировать данные таблицы doc_table во вторую целевую БД. В этом скрипте следует выполнять пользователем xstrmadmin только последний анонимный PL/SQL-блок. Ниже содержимое скрипта с комметнариями.

```

-- установка имени таблицы AQ и имени очереди AQ для процесса XStream capture – НЕ ВЫПОЛНЯТЬ
BEGIN DBMS_XSTREAM_ADM.SET_UP_QUEUE(
queue_table => 'xstrmadmin.ora2pg_queue_tab',
queue_name => 'xstrmadmin.ora2pg_queue');END;
/
--создание процесса XStream capture с именем "ora2pg_capture" – НЕ ВЫПОЛНЯТЬ
BEGIN DBMS_CAPTURE_ADM.CREATE_CAPTURE(source_database=>'ORCL',
queue_name=>'xstrmadmin.ora2pg_queue',
capture_name=>'ora2pg_capture',
capture_class=>'xstream');END;
/
--установка параметров для XStream capture – НЕ ВЫПОЛНЯТЬ
BEGIN
DBMS_CAPTURE_ADM.include_extra_attribute(capture_name=>'ora2pg_capture',attribute_name=>'row_id',include=>true);
DBMS_XSTREAM_ADM.ENABLE_GG_XSTREAM_FOR_STREAMS(enable=>true);
DBMS_CAPTURE_ADM.SET_PARAMETER(capture_name=>'ora2pg_capture',parameter=>'capture_idkey_objects',value=>'Y');
DBMS_CAPTURE_ADM.SET_PARAMETER(capture_name=>'ora2pg_capture',parameter=>'inline_lob_optimization',value=>'Y');
--DBMS_CAPTURE_ADM.SET_PARAMETER(capture_name=>'ora2pg_capture',parameter=>'parallelism',value=>'2');
END;

```

```

/
-- создание процесса XStream out с именем target1 для репликации таблицы doc_table во вторую целевую БД
DECLARE
tables DBMS_UTILITY.UNCL_ARRAY;
schemas DBMS_UTILITY.UNCL_ARRAY;
BEGIN
tables(1):='xstrmadmin.ora2pg_stoppoint';
tables(2):='DOC.DOC_TABLE';
schemas(1):=null;
DBMS_XSTREAM_ADM.ADD_OUTBOUND(
server_name => 'target2',
queue_name => 'xstrmadmin.ora2pg_queue',
source_database => 'ORCL',
include_ddl => true,
table_names => tables,
schema_names => schemas);
END;
/

```

3. prepare_target2_XadmSL.sql – для создания в таблицах исходной БД групп дополнительного журналирования данных в redolog – файлах. Группа включает в себя столбцы необходимые для идентификации строк таблицы doc_table второй целевой БД. Этот скрипт надо выполнить пользователем xstrmadmin. Операторы ALTER DATABASE в этом скрипте можно не выполнять, хотя их повторное выполнение не приведёт к ошибке.

Ниже приведено содержимое скрипта с комментариями.

```

ALTER DATABASE ADD SUPPLEMENTAL LOG DATA -- МОЖНО НЕ ВЫПОЛНЯТЬ
/
ALTER DATABASE DROP SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE, FOREIGN KEY) COLUMNS -- МОЖНО НЕ ВЫПОЛНЯТЬ
/
alter table DOC.DOC_TABLE add supplemental log group target2_00001(DOC_NO, DOC_KEY) always -- группа для doc_table
/

```

Таким образом, для таблицы DOC_TABLE будут созданы две группы дополнительного журналирования. Это не вызовет ошибок в Oracle. Это не вызовет ошибок в утилите.

При репликации операторов UPDATE и DELETE в первую целевую БД фраза WHERE будет содержать только один предикат: WHERE DOC_NO=?. Т.к. в первой целевой БД только один столб

При репликации операторов UPDATE и DELETE во вторую целевую БД фраза WHERE будет содержать два предиката: WHERE DOC_NO=? and DOC_KEY=?.

В данном примере для нормальной репликации в оба источника для таблицы DOC_TABLE было бы достаточно группы target2_00001. Однако, если бы во второй целевой БД первичный ключ состоял бы только из столбца DOC_KEY, то потребовались бы обе группы обязательно. Когда Oracle выполняет запись в redolog изменений в таблице, использует “суперпозицию” столбцов всех групп созданных для таблицы.

3.16.2.4. Выполнение репликации в две целевые БД

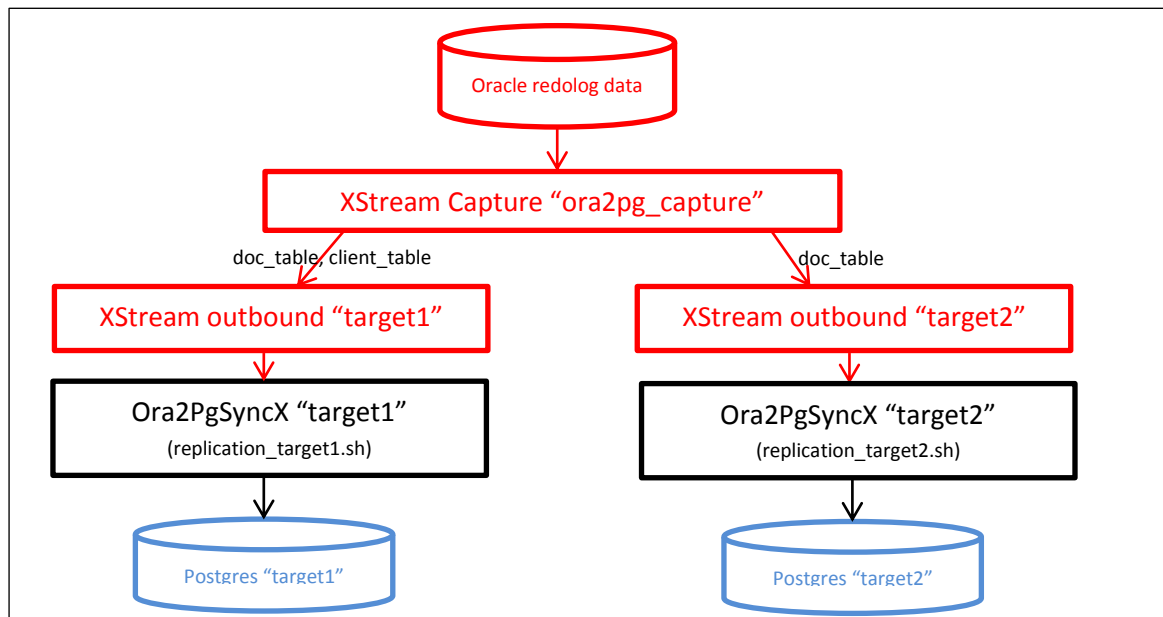
После выполнения скриптов следует запустить replication_target1.sh и replication_target2.sh. В результате репликация будет выполняться по схеме приведённой ниже. В этой схеме захват изменений из redolog выполняет один процесс capture, а передачу данных клиентам выполняют два процесса XStream out. Это сохраняет баланс между желанием снизить нагрузку на Oracle и желанием управлять процессами репликации в две целевые БД независимо друг от друга (запуск утилиты Ora2PgSyncX, её остановка, настройка её параметров). Кроме того, на стороне Oracle можно для target1 безболезненно для target2 останавливать, запускать вновь XStream out сервер, а также удалять и создавать вновь, используя вызовы:

```

DBMS_XSTREAM_ADM.STOP_OUTBOUND, DBMS_XSTREAM_ADM.START_OUTBOUND
BEGIN DBMS_XSTREAM_ADM.DROP_OUTBOUND, DBMS_XSTREAM_ADM.ADD_OUTBOUND.

```

Однако, это не относится к общему ora2pg_capture. Если требуется увеличить степень независимости друг от друга процессов репликации, то следует создать отдельные capture-процессы. Но это сильно увеличит нагрузку на Oracle Server – будут использоваться отдельные процессы LogMiner и двойное чтение redolog.



3.16. Проблемы в процессе репликации

Описание основных проблем, способы их выявления и устранения описаны в

<https://docs.oracle.com/en/database/oracle/oracle-database/12.2/xstrm/troubleshooting-xstream-out.html>

и

<https://docs.oracle.com/en/database/oracle/oracle-database/12.2/strms/identifying-problems-in-an-oracle-streams-environment.html>

Тем не менее, их следует дополнить.

3.16.1. После запуска утилиты репликации не происходит передача данных из Oracle, при этом состояние процесса репликации IDLE (ora2pg_proc.status)

В этом случае следует выполнить запрос в Oracle:

```
select sid, serial# ,state from v$xstream_capture where capture_name='ORA2PG_CAPTURE'
```

- если state='DICTIONARY INITIALIZING', то это означает наличие большого количества транзакций (активностей) при запуске или нехватку памяти для capture – процесса.
- Если state='PAUSED FOR FLOW CONTROL', то надо увеличить SGA для capture-процесса <https://docs.oracle.com/en/database/oracle/oracle-database/12.2/xstrm/configuring-xstream-out.html#GUID-C3391C7F-D2C5-4A65-AA29-4B0EBC2B205E>

п.4.1.2.6 Configure the Streams pool

- если state='INITIALIZING', то по значению полей sid и serial# надо найти сеанс в v\$session и посмотреть какой запрос выполняет этот сеанс. Если этот запрос похож на

```
SELECT count(*)
FROM SYSTEM.LOGMNR_OBJ$ O
,SYSTEM.LOGMNR_TAB$ T
,SYSTEM.LOGMNR_USER$ U
,SYSTEM.LOGMNR_UID$ LUID
,SYSTEM.LOGMNR_DID$ LDID
WHERE
LDID.SESSION# = :1 AND
LUID.LOGMNR_DID = LDID.LOGMNR_DID
AND U.LOGMNR_UID = LUID.LOGMNR_UID
AND O.LOGMNR_UID = LUID.LOGMNR_UID
AND O.LOGMNR_UID = T.LOGMNR_UID
AND O.OBJ# = T.OBJ#
AND O.TYPE# = 2
AND O.REMOTEOWNER IS NULL
AND O.LINKNAME IS NULL
AND O.OWNER# = U.USER#
AND BITAND(T.FLAGS, 536870912) = 0
AND BITAND(T.PROPERTY, 512) = 0
AND U.NAME NOT IN ('SYS', 'SYSTEM', 'CTXSYS'),
```

то следует выполнить **alter index SYSTEM.LOGMNR_DID\$_PK invisible;**

3.16.2. Во время репликации наблюдается неприемлемый рост лага отставания целевой БД от исходной БД

Лег можно посмотреть в `select value from ora2pg_stats where metrica='Lag(ms)'`.

Далее надо оценить состояние очередей `select value from ora2pg_stats where metrica='Q_DMLs'`.

Если очереди почти пусты (близки к нулю), то надо увеличить значение параметра `parallelism` для `capture`-процесса:

```
DBMS_CAPTURE_ADM.SET_PARAMETER(capture_name=>'ora2pg_capture',
                                parameter=>'parallelism',value=>'4');
```

Если очереди наполнены большим количеством DML, то можно:

- Изменяя значения параметров утилиты `OptimizeSelfUpdate`, `BatchMaxBytes`, `BatchMaxDMLs`, `CompressLOB`, `MultValDelete`, `MultValUpdate`, `MultValInsert`, `TxMinBytes`, `TxMinDMLs`, `TxMinMsec`, `DiscardQueryesResultCache` попытаться поднять производительность и уменьшить лаг.
- Оптимизировать параметры PostgreSQL (например `wal_level=minimal`)
- Оптимизировать запросы `update` и `delete` в Postgres
- Оптимизировать запросы фильтрации и/или правила перезаписи (если применяются)
- Увеличить количество слотов утилиты для разделяемых и для неразделяемых таблиц (для этого потребуется плановый быстрый перезапуск см. п. 3.11.2

Перед возобновлением для повторного распределения таблиц по слотам и определения критериев разделения таблиц можно применить `prepare_scripts.sh` внеся изменения в параметр `slot4splits` в `prepare_script.json`. Для увеличения количества слотов репликации)

Если в столбце `executeBatchDuration` таблицы `ora2pg_slots` служебного Postgres интервал более 1 сек, то это означает, что сейчас в сеансе Postgres выполняется пачка (batch) операторов SQL и длительность этого выполнения уже более 1 сек. И есть необходимость изучить статистику и планы выполнения операторов SQL на стороне целевого Postgres средствами мониторинга и сбора статистики Postgres.

3.16.3. Превышение памяти выделенной для JVM

Утилита аварийно завершается с сообщением “out of memory”. Причиной могут быть слишком большое количество больших транзакций выполненных в исходной БД. Рекомендуется

- увеличить память, выделенную для JVM
- уменьшить значение параметра `DiscardQueryesResultCache`, если используется кэш запросов фильтрации
- отказаться от режима обработки ошибок: “`ErrorProcessing`”:false
- отказаться от “прореживания” COMMIT:
 - “`TxMinBytes`”:0
 - “`TxMinDMLs`”:0
 - “`TxMinMsec`”:0

Другой причиной могут быть утечки памяти в утилите. Рекомендуется обратиться в службу поддержки.

3.16.4. После создания XStream outbound появляются лишние группы дополнительного журналирования

В некоторых версиях Oracle, если XStream outbound создаётся с указанием списка реплицируемых таблиц, автоматически создаются три системные группы. Пример:

```
select * from sys.dba_log_groups where table_name='DOC_TABLE'
```

OWNER	LOG_GROUP_NAME	TABLE_NAME	LOG_GROUP_TYPE	ALWAYS	GENERATED
DOC	ORA2PG_00037	DOC_TABLE	USER LOG GROUP	ALWAYS	USER NAME
DOC	SYS_C00206850	DOC_TABLE	LOGGING	ALWAYS	GENERATED NAME
DOC	SYS_C00206851	DOC_TABLE	UNIQUE KEY LOGGING	CONDITIONAL	GENERATED NAME
DOC	SYS_C00206852	DOC_TABLE	FOREIGN KEY LOGGING	CONDITIONAL	GENERATED NAME

Их наличие не приводит к ошибкам репликации, но может немного снизить производительность. Их следует удалить. Если ещё раз запустить утилиту в режиме `prepare_scripts`, то в `prepare_scripts_XadmSL.sql`, в появятся нужные операторы `ALTER TABLE ... DROP SUPPLEMENTAL LOG GROUP ...`.

4. Синхронизация текущих значений последовательностей

В ходе репликации в целевой БД выполняются операторы DML. Среди них есть операторы `insert`. Они выполняют вставку в таблицу значений всех её столбцов. В том числе значение синтетического первичного ключа. Который, как правило, является целым положительным числом. В исходной БД при выполнении `insert` это число, как правило, извлекается из соответствующей последовательности. При этом последовательность меняет своё текущее значение. Этих изменений текущих значений не происходит в ходе репликации.

Не смотря на то, что технология XStream позволяет реплицировать такие изменения, это не используется в утилите для снижения нагрузки на целевую БД. Вместо этого предусмотрен режим синхронизации значений. Запуск утилиты в этом режиме следует выполнять после планового завершения репликации.

Запуск выполняется скриптом `seq_ora2pg.bat` или `seq_ora2pg.sh`.

Оценку результатов и наличие ошибок можно выполнить запросом `select * from ora2pg_seq` в обслуживающем Postgres.

5. Ограничения

Самое важное ограничение – это репликация большого количества массовых DML-операций. Чем больше в прикладной системе таких массовых DML, тем сложнее процессам XStream и утилите с ними справиться. Тем сложнее с небольшим лагом выполнить синхронизацию данных в Postgres. Это займёт гораздо больше времени в Postgres (приблизительно в 2-3 раза больше при массовом изменении/удалении 100 000 записей), чем в Oracle.

Массовые операторы DML – это обычная проблема любой репликации, по крайней мере, это касается Oracle. И ни в одной системе репликации (в том числе [GoldenGate](#), [Debezium](#)) она не решена должным образом.

Можно реализовать свою систему отслеживания массовых DML в Oracle и последующего их выполнения в Postgres, что потребует модификации кода в исходной БД Oracle. И это совсем другой подход, который не рассматривается в данном документе.

Другое ограничение – репликация пользовательских типов данных. Скалярные типы данных при их репликации через XStream, Oracle будет стремиться отобразить в базовые типы

RDBMS. Если получится. Хуже дело обстоит с коллекциями и вложенными таблицами. В настоящей версии это не реализовано.