Микросервис обработки архивных файлов СУБД Oracle для программ репликации данных в различные СУБД

Oracle Logical Replicator
(OLR)
версия 1.1

Описание технологии разработки

Всего страниц 24

Оглавление

1. Термины и сокращения

DML	Data Manipulation Language - группа операторов SQL, которые применяются для работы с данными, которые хранятся в базе данных (INSERT, UPDATE, DELETE),
DDL	Data Definition Language - группа операторов SQL, которые применяются для создания, изменения и удаления объектов базы данных (таблиц, индексов, схем и т. д.),
БД	База данных,
SCN	System change number - уникальный идентификатор, который присваивается каждой транзакции, изменяющей базу данных. Каждый раз, когда транзакция фиксируется в базе данных Oracle, SCN увеличивается,
JSON	JavaScript Object Notation - текстовый формат обмена данными, представленными в виде ключ-значение,
URL	Uniform Resource Locator — это уникальный адрес ресурса в интернете,
Архивный журнал	Совокупность файлов журнала повторов (Archived Redo Log Files),
Архивный лог	Файл журнала повторов (Archived Redo Log File),
TCP	Протокол передачи данных интернета в стеке протоколов TCP/IP, выполняющий функции транспортного уровня модели OSI.

2. Введение

Для решения задачи получения изменений, выполняемых приложением в БД Oracle есть несколько средств. Наиболее популярными являются следующие.

1. Триггеры для каждой интересующей таблицы
2. Журналы материализованных представлений
3. Log Miner и его представление V$LOGMNR_CONTENTS
4. Oracle Streams
5. Oracle XStream
6. Oracle GoldenGate

Первые два средства не связаны с извлечением данных из Oracle Redo Log. Их использование может привести к существенному снижению производительности прикладной системы. Кроме того, эти два средства связаны с необходимостью разработки генератора триггеров и материализованных представлений.

Применение Log Miner имеет следующие недостатки. Если Log Miner работает в режиме COMMITED_DATA_ONLY, то V$LOGMNR_CONTENTS будет выдавать данные о DML операторах в порядке возрастания COMMIT SCN транзакций, и DML SCN внутри транзакции. И это позволяет организовать многопоточную репликацию в иную БД без потери и дублирования данных. Однако, в режиме COMMITED_DATA_ONLY весьма сложно узнать в каких log-файлах расположена транзакция. Это нужно для определения множества журнальных файлов при каждом запуске Log Miner. Чтобы это определить приходится выполнять чтение из системного представления flashback_transaction_query. Это представление хранит данные о транзакциях, выполненных за последние N минут. N определяется параметром СУБД DB_FLASHBACK_RETENTION_TARGET. По умолчанию он равен 1440 минут. В БД с большой нагрузкой запрос к этому представлению может выполняться очень долго и режим COMMITED_DATA_ONLY оказывается не жизнеспособным. Если COMMITED_DATA_ONLY не используется, то необходимо в однопоточном режиме фильтровать незавершённые транзакции, накапливать завершённые и применять их в целевой системе по возрастанию SCN COMMIT. Кроме того, использование пакета DBMS_LOGMNR с множественными регулярными вызовами DBMS_LOGMNR.ADD_LOGFILE, DBMS_LOGMNR.START_LOGMNR, BMS_LOGMNR.END_LOGMNR приводит к потерям скорости репликации.

Применение Oracle Streams для отправки изменений за пределы среды Oracle вызывает необходимость построения программы чтения очередей Advanced Queue и декодирования её элементов (Logical Change Record) на PL/SQL, что тоже может не дать требуемой производительности. Кроме того, начиная с Oracle 12 Streams считается устаревшим, а в Oracle 19 не гарантирована корректная работа Streams.

Oracle XStream используется в некоторых сторонних продуктах. Например, в Debezium. Применение Oracle XStream во многих случаях (проверено на практике) отягощено медленным повторным запуском процесса XStream Capture, повторным чтением словаря

БД и высокой нагрузкой, как по процессору, так и по памяти на сервер Oracle. Кроме того, для мониторинга состояния процесса репликации, поиска причин отказов и потерь производительности необходимо использовать более 10-ти системных представлений.

Oracle Golden Gate (OGG) – высокоуровневое, но сложное средство, требующее обучения и практики. Есть и более существенные ограничения:

- При организации репликации в PostgreSQL не поддерживаются large objects.
- Не гарантируется поддержка клонов PostgreSQL
- Не используется возможность не отключать триггеры и не удалять внешние ключи при репликации в PostgreSQL с параметром сеанса session_role=replica
- Поток изменений, извлекаемый одним процессом extract нельзя разделить на потоки replica потаблично. OGG построен по принципу соблюдения целостности транзакции, и сохраняет её в целевой системе такой, какой она была выполнена в исходной системе

К недостаткам OGG можно отнести и тот факт, что репликация выполняется через промежуточные файлы.

Всё это явилось основанием для разработки независимого от Oracle средства декодирования двоичных Oracle Redo Log файлов.

- OLR не создаёт никакой дополнительной нагрузки на исходную БД в процессе репликации.
- OLR передаёт получателю содержимое транзакций в порядке SCN commit.
- OLR не требует времени на выполнение перезапуска процесса репликации.

Кроме того, OLR обеспечивает клиента возможностью повторения транзакции с целью оперативного преодоления ошибок, возникающих при репликации данных в целевую систему.

3. Описание программы и краткий принцип работы

Программа “Oracle Logical Replicator” (OLR) представляет из себя сервер, предоставляющий возможность чтения и разбора файлов архивного журнала повторов (Archive Redo Log Files) БД Oracle с последующей отправкой транзакций с DML над указанными таблицами в унифицированном формате в сетевой сокет. OLR по сути представляет из себя основу для программ, осуществляющих репликацию из экземпляра БД Oracle в другие базы данных. Программа написана на языке Rust 2021 издания и протестирована на версии Oracle 19.

В процессе работы программы создаётся 3 активных потока. Один поток отвечает за чтение архивных логов из файловой системы. Два остальных делят ответственность за работу репликатора:

1. Первый поток отвечает за:
    - Чтения словаря (описаний таблиц из системных представлений),
    - Опрашивание БД на наличие ещё не прочитанных архивных журналов,
    - Декодирование логов, извлечение информации о выполненных DML,
    - Приготовление транзакций (накапливание векторов изменений, восстановление DML, выполненных в рамках транзакции), их упорядочивание по возрастанию SCN коммитов,
    - Сборку транзакций в формат для передачи клиенту;
2. Второй поток отвечает за:
    - Приём и отдачу команд клиенту,
    - Подготовку транзакций к выводу,
    - За перезапуск программы,
    - Сохранение состояния репликации (фиксирование нижней границы по SCN среди неподтверждённых транзакций).

При включении сбора метрик, создаётся дополнительный поток, отвечающий за сбор статистики и экспорт её в целевую систему.

4. Установка OLR

4.1. Установка OLR

Программа написана на языке Rust, являющимся компилируемым языком программирования на основе проекта LLVM. Целевой ОС программы является Linux, разработка и тестирование проводилась на архитектуре X86-64.

Пакет установки представлен архивным файлом: OLR_FILES.zip или OLR_FILES.tar.gz

Содержимое пакета нужно распаковать в удобную директорию (к примеру, домашнюю). В примере ниже показаны шаги для установки в домашнюю директорию пользователя *olr* и последующий запуск программы.

```
igor@i-glushatov2:~/replicator$ sudo su - olr
$ ls
OLR_FILES.tar.gz
$ gzip -d OLR_FILES.tar.gz && tar -xf OLR_FILES.tar
$ cd OLR_FILES
$ ls
configs  x86_64
$ ls configs
ReplicatorExample.json
$ ./x86_64/replicator --file configs/ReplicatorExample.json --log-level 3
2025-09-11 14:10:25  [INFO] - OLR Copyright (c) 2024-2025 «FORS Telecom» LLC.
2025-09-11 14:10:25  [INFO] - Version: 1.2.0
2025-09-11 14:10:25  [INFO] - OS: linux; Arch: x86_64; Family: unix
2025-09-11 14:10:25  [INFO] - Main PID: 65564
```

Однако перед тем, как запускать программу необходимо произвести некоторые дополнительные приготовления, о которых написано дальше.

4.2. Структура и состав каталогов

- *replicator* – исполняемый файл программы
- *configs/* – стандартная директория с конфигом репликатора
    - *Replicator.json* – файл конфигурации репликатора (имя может быть изменено)
- *olr_data/* - директория с сохранённым состоянием программы (создаётся при первом запуске программы), временными файлами и данными необходимыми для работы
    - *checkpoint.bin* – файл с сохранёнными данными для возобновления репликации в случае остановки

Структура директорий не фиксирована, есть возможность переопределения местоположения настройками файла конфигурации и cli аргументами.

4.3. Описание конфигурационного файла

Содержимое файла представляется в формате JSON. Он указывает необходимые и опциональные параметры для репликации.

Пример файла ReplicatorExample.json:

```
{
    "version": "1.2.0",
    "context": {
        "memory": {
            "min-mb": 16,
            "max-mb": 1024,
            "max-tx-msgs": 100
        },
        "data": {
            "path": "olr_data"
        }
    },
    "metrics": {
        "postgres": {
            "connect-string": "postgresql://postgres:postgres@localhost:5432/postgres",
            "table-scheme": "postgres",
            "table-name": "olr_metrics"
        }
    },
    "source": {
        "user": "SCOTT",
        "password": "TIGER",
        "server": "//172.27.12.137/SE",
        "path-mapping": [
            "/path/to/oracle/archivelog",
            "/mnt_oracle/path/to/oracle/archivelog"
        ]
    },
    "target": {
        "address": "0.0.0.0:64000"
    }
}
```

Описание полей:

- *version* – версия OLR. Указывается для контроля совместимости конфигурационного файла с программой
- *context* – опциональные настройки репликатора
    - *memory* – настройки использования памяти
        - *min-mb* – минимальный объём памяти, который резервирует программа
        - *max-mb* – максимальный объём памяти, который программа может зарезервировать (потребление может быть превышено, если это необходимо)
        - *max-ts-msgs* – максимальное количество транзакций, которое может хранить очередь, пока они не будут прочитаны клиентом
    - *data* – местоположение и название каталога с артефактами программы
- *metrics* – настройки экспортёра метрик

    - *postgres* – настройки для экспортёра в БД Postgres
        - *connect-string* – строка подключения к БД Postgres
        - *table-scheme* – название схемы
        - *table-name* – название таблицы, в которую будут зкспортироваться метрики
- *source* – настройки источника архивных логов и метаинформации БД
    - *user* – имя пользователя для подключения к БД
    - *password* – пароль для подключения к БД
    - *server* – URL для подключения к БД в формате Oracel Easy Connect вида: “[//]<host>[:<port>]/<service_name>”
    - *path-mapping* – Список пар путей в файловой системе [before1, after1, before2, after2, ...]. Каждый путь (к архивному журналу повторов) согласуется с этим списком. Если префикс пути к архивному журналу совпадает с beforeX, он заменяется на afterX. К примеру, имея список ["/opt/oracle/fra/FST/archivelog", "/mnt_oracle/data/d3/oracle/fra/FST/archivelog"], путь к архивному журналу (взятый из системного представления) “/opt/oracle/fra/FST/archivelog/2024_08_14/o1_mf_1_20_mcs6xs2t_.arc” преобразуется в “/mnt_oracle/data/d3/oracle/fra/FST/archivelog/2024_08_14/o1_mf_1_20_mcs6xs2t_.arc”
- *target* – атрибуты для подключения клиента к OLR
    - *address* – tcp хост и порт, который будет прослушивать программа для получения команд и выдачи результатов

5. Запуск программы

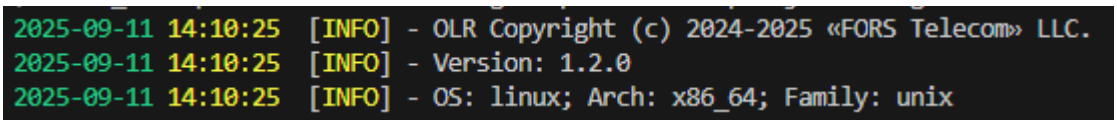
5.1. Аргументы командной строки

При запуске программы можно указать два опциональных параметра:

1. --log-level – уровень ведения логов в stderr и stdout
2. --file – путь к конфигурационному файлу

Пример: ./replicator --log-level 3 --file configs/Replicator.json

При запуске должна вывестись небольшая справка о версии программы, целевой архитектуре сборки и пути считанного конфига.

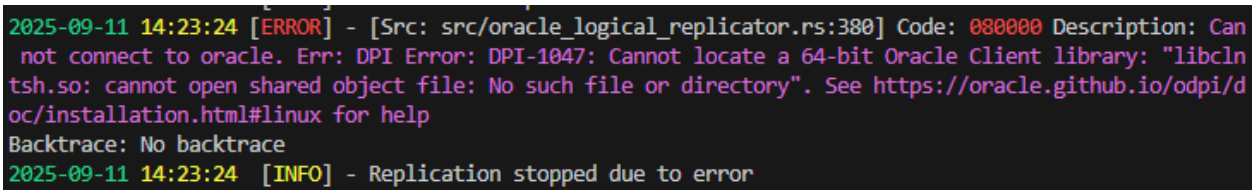


Параметры логирования:

- 0 – ничего
- 1 – только критические ошибки
- 2 – ошибки и предупреждения, вызванные неожиданным поведением, которые могут повлечь дальнейшие проблемы
- 3 – информационные сообщения об обработанных файлах, статистика и т.п.

5.2. Необходимые приготовления

Для того, чтобы сервер OLR мог подключиться к серверу Oracle и обслуживать запросы клиента, нужно сделать некоторые приготовления. При запуске может возникнуть ошибка:



Она означает, что программа, запущенная, под текущим пользователем, не смогла найти динамическую библиотеку для клиента Oracle. Для работы программы обязательно требуется клиент, поэтому в первую очередь следует его установить, если его нет. Далее есть два варианта решения этой проблемы:

1. В переменную окружения пользователя LD_LIBRARY_PATH добавить каталог с динамическими библиотеками Oracle клиента. К примеру, так:

   export LD_LIBRARY_PATH="/opt/oracle/instantclient_19_24/:$LD_LIBRARY_PATH"

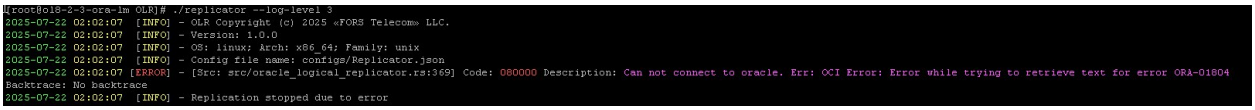
   Или в случае запуска на той же машине, где установлен Oracle:

   export ORACLE_HOME=/opt/oracle/product/19.3

   export LD_LIBRARY_PATH="$ORACLE_HOME/lib/:$LD_LIBRARY_PATH"

2. Создать в директории */etc/ld.so.conf.d/* файл (к примеру: *oracle-instantclient.conf*) с следующим содержанием:
   # Oracle Client 19.0 libraries
   /opt/oracle/instantclient_19_28

На сервере с базой данных необходимо определить переменную ORACLE_HOME для предотвращения ошибки:



Если OLR запускается не на том же сервере, где стоит база данных, то для доступа к архивным логам нужно либо организовать автоматическое копирование логов, либо смонтировать директорию с ними. В случае использования ASM монтировать не получится, поэтому можно воспользоваться первым способом, либо отправлять изменения на Standby, откуда уже будут собираться архивные файлы журналов повтора. Во всех случаях важно, чтобы в параметре конфигурационного файла *path-mapping* были прописаны префиксы для замены путей.

Также необходимо включить дополнительное журналирование в БД Oracle (Supplemental Logging). Достаточно включить дополнительное журналирование первичных ключей:

ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;

Архивные журналы не содержат всех данных, необходимых для репликации таблиц. Поэтому получение этих данных происходит через обращение к словарю БД. Для этого нужно выдать права пользователю, которым репликатор подключается к Oracle, на чтение следующих таблиц:

- SYS.OBJ$
- SYS.TAB$
- SYS.USER$
- SYS.TS$
- SYS.IND$
- SYS.INDPART$
- SYS.COL$
- SYS.CCOL$
- SYS.ECOL$
- SYS.CDEF$
- SYS.TABPART$
- SYS.TABCOMPPART$
- SYS.TABSUBPART$
- SYS.LOB$
- SYS.LOBCOMPPART$

- SYS.LOBFRAG$
- V$ARCHIVED_LOG
- V$DATABASE
- V$TRANSPORTABLE_PLATFORM
- V$DATABASE,
- V$TRANSPORTABLE_PLATFORM,
- PRODUCT_COMPONENT_VERSION
- Таблицы схемы XDB
    - XDB.XDB$TTSET
    - XDB.X$QN<token_suf>
    - XDB.X$NM<token_suf>
    - XDB.X$PT<token_suf>

Репликатор разработан исходя из того, что в отношении реплицируемых таблиц не выполняются никакие DDL – операторы во время процесса репликации.

6. Работа программы

6.1. Общий алгоритм

При запуске программа ожидает запрос на tcp-подключение. Когда оно устанавливается, репликатор ожидает получения двух сообщений от клиента:

1. Список таблиц для репликации,
2. Начальный SCN, указывающий на SCN начала первой реплицируемой транзакции.

Если это не первый запуск репликатора, то он уже может хранить минимальный SCN начала необработанных с прошлого запуска транзакций в файле *olr_data/checkpoint.bin* или *<data_path>/checkpoint.bin*, если в конфигурации переопределён путь для сохранения состояния. Его можно запросить специальной командой и передать как точку старта программе. После этого запускается два потока: “процесс чтения” и “процесс передачи”.

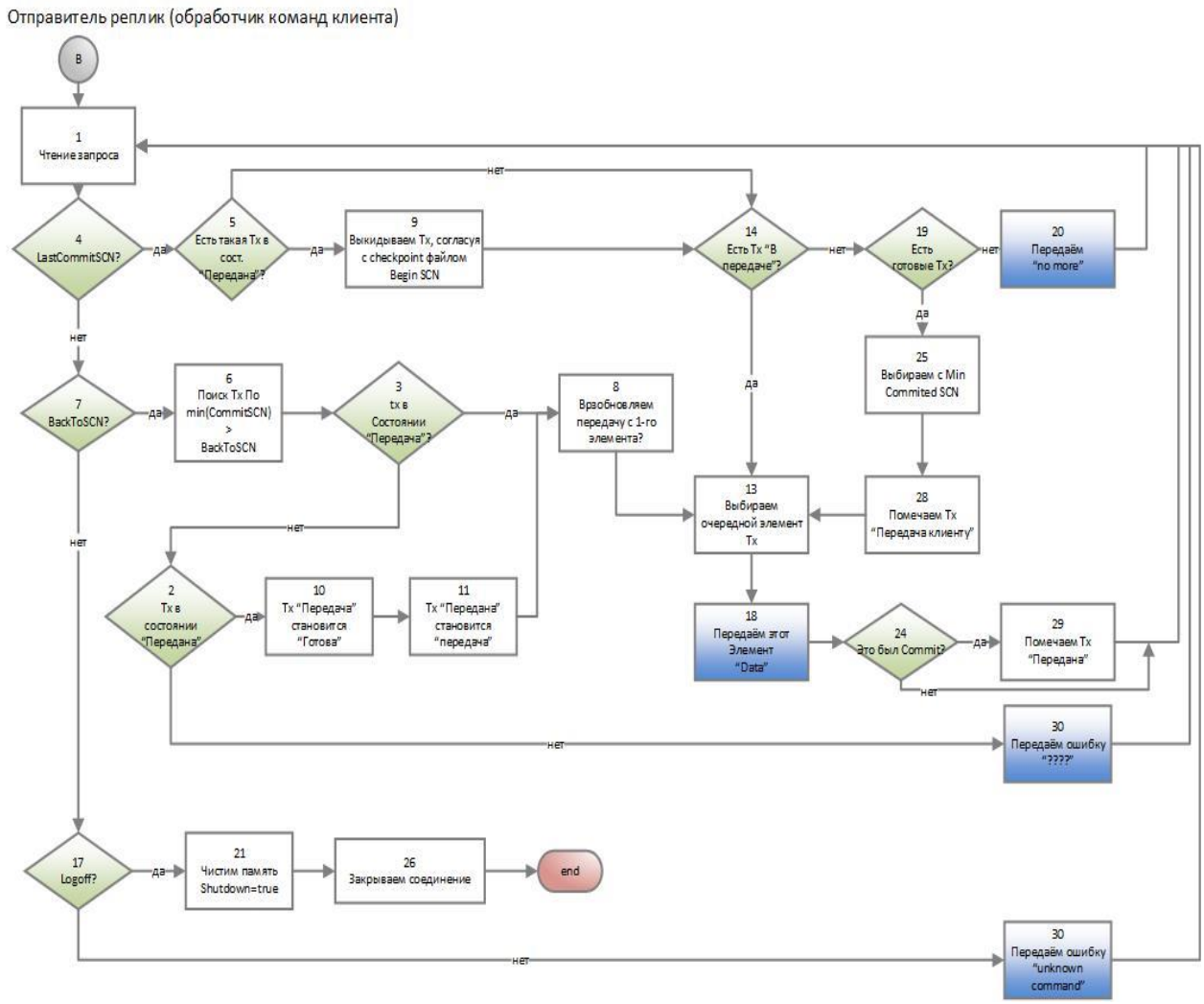
Первый опрашивает БД на получение списка новых не обработанных архивных логов. Найденные логи обрабатываются друг за другом, и вся нужная информация сохраняется в пул транзакций. Транзакция является атомарным объектом для передачи между потоками и для механизма сохранения состояния репликации. Если какая-то транзакция завершается, то она преобразуется в выходной формат, удаляется из пула и ставится на ожидание обработки вторым потоком. Размер данной очереди ограничивается настройкой конфигурации. В случае нехватки места, поток блокируется на операции добавления в очередь до тех пор, пока предыдущие транзакции не будут запрошены пользователем.

Второй поток ожидает получение команд от клиента. На запрос о готовности клиента принять DML в памяти OLR разыскивается готовая (т.е. завершённая оператором COMMIT) транзакция и переводится в состояние передачи клиенту. Каждый новый запрос на получение DML будет передавать клиенту одну из следующих операций: BEGIN, COMMIT, INSERT, DELETE, UPDATE, CHUNK. При завершении передачи транзакции, она переводится в состояние “передана”. Со стороны клиента ожидаются сообщения о SCN оператора COMMIT транзакции обработанной клиентом. Это указывает серверу OLR на то, что можно удалить из памяти все транзакции находящиеся в состоянии “передана”, SCN оператора COMMIT которых меньше или равен указанному клиентом. Если же по какой-то причине надо вернуться к раннее переданной, но не подтверждённой клиентом транзакции, то клиент передаёт команду для возврата на определённый SCN. Транзакции, которые имеют SCN оператора COMMIT строго больше переданного, будут возвращены в очередь на обработку вместе с текущей транзакцией находящейся в состоянии передачи клиенту. После этого OLR ожидает запроса на передачу.

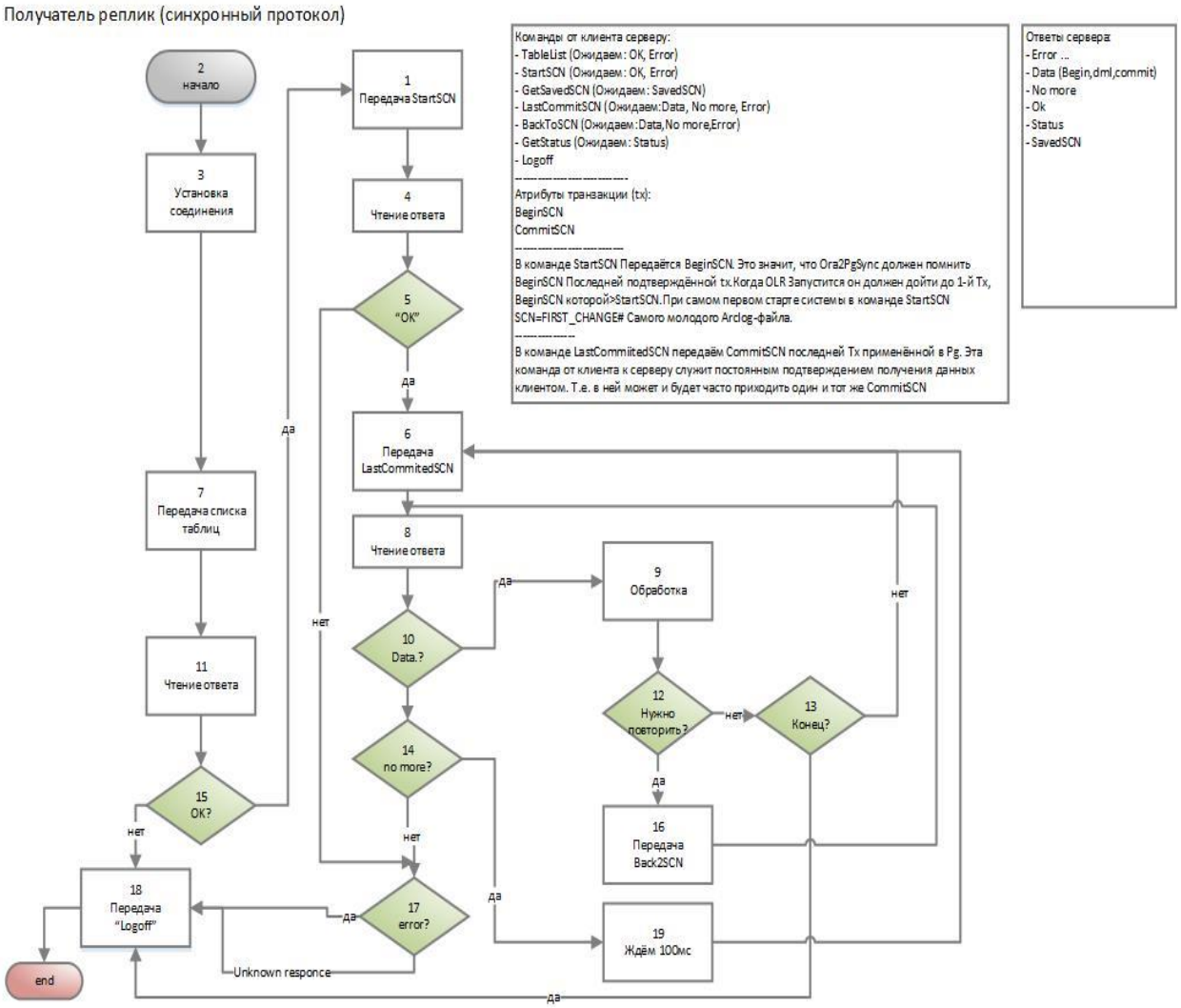
6.2. Блок-схема работы OLR

OLR работает по принципу синхронного взаимодействия и является серверной стороной, ожидающей команд от внешней стороны. Для уменьшения задержек OLR готовит данные для вывода заранее, но только в ограниченном объёме. Пока клиент не начнёт запрашивать DML, OLR не будет удалять найденные транзакции или сохранять их на диск.

Схематическое описание работы потока отправителя, коммуницирующего с клиентским приложением:



Пример схемы работы клиентского приложения:



6.3. Форматы сообщений

6.3.1. Описание команд от клиента к серверу

Поле	Тип/Размер	Описание
TableList		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для TableList - 1
SQLString	[u8; MessageSize - 2]	SQL запрос в виде UTF-8 строки на получение таблиц, который будет исполняться в базе данных. Вывод запроса должен состоять из двух столбцов: 1. Владелец таблицы (OWNER) 2. Имя таблицы (NAME)
StartSCN		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для StartSCN - 2
StartSCN	u64	Стартовый SCN
LastCommitedSCN		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для LastCommitedSCN - 3
LastCommitedSCN	u64	SCN последней подтверждённой транзакции пользователем
BackToSCN		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для BackToSCN - 4
BackToSCN	u64	Все транзакции, CommitSCN которых больше BackToSCN, переходят в режим “Готово к передаче”. Отправка транзакций возобновляется с начала самой ранней по CommitSCN транзакции

LogOff		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для LogOff - 5
GetStatus		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для GetStatus - 6
GetSavedSCN		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для GetSavedSCN - 7

TableList ожидается репликатором после запуска для извлечения данных из словаря БД Oracle. В ответ сервер передаст **Ok** или **Error**, если по каким-то причинам программа не смогла получить данные (прервалось соединение, неправильный формат SQL запроса, наличие неподдерживаемых для репликации таблиц и т.п.).

StartSCN ожидается сервером после передачи Ok в ответ на TableList. Все транзакции, начало которых больше или равны переданному значению, начнут обрабатываться. В ответ серверможет отправить **Ok** или **Error**.

GetSavedSCN возвращает сохранённый минимальный SCN начала неподтверждённых клиентом транзакций. В ответ отправляется сообщение **SavedSCN**, которое может содержать, а может и не содержать значение, если это первый запуск репликатора или после последнего запуска файл checkpoint.json был удалён.

GetStatus возвращает статус работы репликатора. В ответ отправляется сообщение **Status**, которое содержит код статуса. Подробно состояния репликатора описанны ниже.

LogOff завершает работу репликатора. Сервер ничего не отправляет в ответ.

LastCommitedSCN запрашивает очередной DML. Если таковой существует, то отправляется сообщение **Data**, в противном случае отправляется ответ **NoMore**.

BackToSCN как и **LastCommitedSCN** так же запрашивает DML. Однако состояние передачи откатывается на переданный SCN. Откат идёт относительно транзакций, поэтому клиент должен ожидать, что данная сервер ответит сообщением **Data** с **Begin** содержимом, либо **NoMore**.

6.3.2. Описание ответов от сервера клиенту

Поле	Тип/Размер	Описание
Ok		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для Ok - 1
NoMore		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для NoMore - 2
Error		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для Error - 3
ErrorCode	u32	Код ошибки
ErrorString	[u8; MessageSize - 2]	Описание ошибки в виде строки в UTF-8
Data		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для Data - 4
Data	[u8; MessageSize - 2]	Данные
Status		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для Status - 5
Status	u16	Статус работы программы: 1. WaitTableList – 1 2. WaitStartSCN – 2 3. Replicating – 3
SavedSCN		
MessageSize	u32	Размер сообщения
OperationCode	u16	Код операции. Для SavedSCN - 6

Flag	u16	Есть ли информация о сохранённом SCN. 1 – есть, 0 – нет.
MinStartSCN	u64	Минимальный Start SCN среди тех транзакций, которые ещё не были подтверждены

6.3.1. Описание данных внутри ответа Data

Поле	Тип/Размер	Описание
Begin		
OperationCode	u8	Код операции. Для Begin - 1
SCN	u64	SCN начала транзакции.
CommitSCN	u64	SCN оператора COMMITтранзакции.
XID	u64	XID транзакции (0xAABBCCCC). 16 бит – undo segment number 16 бит – undo slot number 32 бит – sequence number
Timestamp	u32	Timestamp операции.
SerialNumber	u16	Серийное число сессии
SessionNumber	u32	Сессионное число
∑	**35 байт**	**Суммарная длина сообщения**
Commit		
OperationCode	u8	Код операции.– 2.
SCN	u64	SCN DML - операции.
CommitSCN	u64	SCN оператора COMMIT транзакции.
XID	u64	XID транзакции.
Timestamp	u32	Timestamp операции.
∑	**29 байта**	**Суммарная длина сообщения**

DML операции		
OperationCode	u8	Код операции. Insert – 4 Delete – 5 Update – 6
SCN	u64	SCN операции.
CommitSCN	u64	SCN оператора COMMITтранзакции.
XID	u64	XID транзакции.
Timestamp	u32	Timestamp операции.
ObjectId	u32	ID объекта, над которым производится DML операция
SchemaNameSize	u8	Размер имени схемы
TableNameSize	u8	Размер имени таблицы
RowIdSize	u8	Размер RowId
SchemaName	[u8; SchemaNameSize]	Имя схемы
TableName	[u8; TableNameSize]	Имя таблицы
RowId	[u8; RowIdSize]	RowId
?BeforeData	Columns	Значения и информация столбцов. Записывается для Delete, Update
?AfterData	Columns	Значения и информация столбцов. Записывается для Insert, Update
$\sum$	**36 байт + SchemaNameSize + TableNameSize + RowIdSize + length(BeforeData) + length(AfterData)**	**Суммарная длина сообщения**

Columns		
ColumnsCount	u16	Количество столбцов.
ColumnsValue	[ColumnValue;ColumnsCount]	Значения столбцов.
ColumnValue		
ColumnNameSize	u8	Длина имени столбца
ColumnValueSize	u64	Размер значения столбцаNULL = 0
ColumnType	u16	Тип столбца Пример: 1 – VARCHAR2 / NVARCHAR2 2 – NUMBER / FLOAT 112 - CLOB
ColumnPrecision	i64	Precision для числовых типов NULL = i64::MIN
ColumnScale	i64	Scale для числовых типов NULL = i64::MIN
ColumnCSId	u64	CharsetId для строковых типов NULL = u64::MAX
ColumnCSForm	u8	CharsetForm для строковых типов NULL = u8::MAX
IsChunked	u8	Передаются ли данные этого столбца кусками 0 – Нет Иначе – Да
ColumnName	[u8; ColumnNameSize]	Имя столбца
ColumnValue	[u8; ColumnValueSize]	Значение столбца

Chunk		
OperationCode	u8	Код операции. Для Chunk – 7
SCN	u64	SCN операции.
CommitSCN	u64	SCN COMMT транзакции.
XID	u64	XID транзакции.
Timestamp	u32	Timestamp операции.
ColumnNameSize	u8	Длина имени столбца
ColumnName	[u8; ColumnNameSize]	Имя столбца
ColumnType	u16	Тип столбца 112 – CLOB / NCLOB 113 – BLOB
ChunkFlags	u16	Флаги сообщения: & 1 – последний кусок значения столбца & 2 – последний кусок для всей строки Т.е. диапазон 0 – 3.
ChunkDataSize	u64	Размер данных куска NULL = 0
ChunkData	[u8; ColumnDataSize]	Данные куска
∑	**42 байт + ColumnNameSize + ColumnDataSize**	**Суммарная длина сообщения**

6.3.1.1. Типы данных, таблиц и ограничения

OLR поддерживает репликацию следующих типов данных :

Тип	Код типа
Varchar2/NVarchar2	1
Number	2
Long	8
Varchar/NChar Varying	9
Date	12
Raw	23
LongRaw	24
XMLType (BinaryXML)*	58
Char/NChar	96
BinaryFloat	100
BinaryDouble	101
CLOB*	112
BLOB*	113
Time	178
TimeTZ	179
Timestamp	180
TimestampTZ	181
IntervalYM	182
IntervalDS	183
TimestampLTZ	231

- LOB с функциями DEDUPLICATION, COMPRESSION и ENCRYPTION не поддерживаются и интерпретируются всегда как NULL.
- BLOB и CLOB протестированы не на всех возможных операциях, доступных в БД Oracle, поэтому не все LOB локаторы могут быть корректно распознаны. В частности, могут возникнуть сбои при использовании функций ERASE; откате транзакции, в которой выполнялась операция WRITE или APPEND; не тестировались функции подмножества FRAGMENT (INSERT, DELETE, MOVE,

REPLACE). Не поддерживается репликация LOB’ов, созданные с параметрами DEDUPLICATE, COMPRESS или ENCRYPT.

- XMLType может храниться как CLOB и как BLOB, соответственно может наследовать проблемы, связанные с LOB’ами. При этом в формате BLOB (Binary XML) хранится не raw представление данных, а сжатое представление XML. Поэтому его разбор на клиентской стороне может быть затруднителен. Есть несколько вариантов использовать значение BinXML локатора: использовать JDBC или OCI для работы с XML (однако в этих интерфейсах нет возможности напрямую создавать объекты XML и подставлять локатор, нужно искать обходные пути), декодировать локатор самостоятельно (клиенту нужно реализовать декодер и уметь запрашивать коды атрибутов из реплицируемой БД). Если данные варианты по каким-то причинам недоступны, то имея информацию по остальным столбцам и ROWID, можно пробовать получить значение XML из БД по ним.
