Потоки данных в Oracle – это очень просто

Владимир Пржиялковский
Преподаватель технологий Oracle
prz@yandex.ru
http://www.ccas.ru/prz

Июнь 2006 г.
Правка: декабрь 2007 г.

Знает названье потока лишь тот, кто вблизи обитает.

Гесиод, Теогония

От Махачкалы до Баку
Луны плавают на боку,
И, качаясь, плывут валы
От Баку до Махачкалы.

Качка в Каспийском море. Стихи Б. Корнилова,
музыка Ю. Визбора

Введение

Потоки данных в Oracle – более поздняя, чем «обычная» репликация (односторонняя, двусторонняя и многосторонняя), модель организации непрерывного переноса данных как внутри БД, так и между базами. Это значительно упрощенная реализация идей, изложенных, например, в *http://www-db.stanford.edu/~widom/stream.ppt*: в частности реализация Oracle Streams не предлагает языкового оформления, а только уровень API. Технически потоки Oracle Streams опираются на созданный независимо и ранее аппарат организации очередей передачи сообщений, известный под названием Oracle Advanced Queuing.

Потоки данных появились в Oracle версии 9, а в следующих версиях получили свое развитие: например, в версии 10 по части возможностей (например, Down Stream) и организации (например, собственный источник рабочей памяти streams pool в SGA). Ряд усовершенствований внесла версия 11, например, общий аппарат советников (advisors) пополнился автоматическим советником по производительности потоков.

В отличие от «обычной» репликации Oracle Streams *не* требует заведения особых структур в БД (журналов таблиц, materialized views). Подбно механизму репликации, уже давно использовавшемуся в Sybase, репликация в Oracle Streams основана на обработке информации из журнала БД. В Oracle этот механизм сильнее всего напоминает ранее существовавшую технику многосторонней репликации (multimaster replication): хотя там и не используется информация из журнала БД, но уже существует понятие очереди сообщений.

Основные понятия

В потоковой передаче данных участвуют следующие основные элементы:

- **Процесс захвата изменений** (Capture Process). Фоновый процесс, постоянно просматривает средствами LogMiner рабочие и архивированные журналы; выбирает из них требуемые записи об изменениях в

исходной таблице/схеме/БД (INSERT, UPDATE, DELETE, MERGE, обновления полей LOB); формирует из этих записей *логическую запись об изменении*, Logical Change Record (*LCR*); помещает LCR в качестве *события* в *очередь*, сформированную средствами Streams Advanced Queuing (SAQ)[10-].

- **Процесс передачи изменений** (Propagation Process). Постоянно выбирает события из очереди в исходной БД и передает их в очереди на принимающих БД через Oracle Net.
- **Процесс внесения, применения изменений** (Apply Process). Постоянно выбирает события из очереди в принимающей СУБД. LCR либо применяются непосредственно к таблицам принимающей БД, либо передаются программе обработки, написанной пользователем на свое усмотрение.
- **Очередь** (queue). Она может складываться из упорядоченного множества (списка) объектов конкретного типа, но чаще используются очереди из объектов типа SYS.ANYDATA. В очередь попадают LCR (автоматически, в соответствии с заданными правилами, или же явным добавлением из программы) или более общие *сообщения* (вставляются и извлекаются вручную). Очередь моделируется с помощью специально создаваемых служебных таблиц, но для автоматически размещаемых в очереди событий LCR дополнительно имеется буфер в SGA.

[10-] до версии 10 использовалось название Advanced Queuing (AQ).

Конфигурация СУБД и БД для возможности организации потоков

Параметры СУБД

Для организации потоков данных нужно иметь должные значения целого ряда параметров СУБД, однако чаще всего достаточно удостовериться в следующем:

COMPATIBLE >= 9.2
Далее предполагается >= 10.1.0.

GLOBAL_NAMES = TRUE
для каждой БД, участвующей в переносе данных.

STREAMS_POOL_SIZE >= 200m
Параметр существует с версии 10.1 и задает область памяти для временного размещения захваченных событий. Если STREAMS_POOL_SIZE = 0, будет использована память из shared pool, вплоть до 10% от этой области.

При расчете нужно учитывать следующее:
+ 10m для каждого нового уровня параллелизма процесса захвата
+ 1m для каждой степени параллелизма процесса применения
+ 10m для каждой новой очереди захваченных событий.

В версии 9.2 нагрузка на выделение памяти под нужды потоков ложится на shared pool.

SHARED_POOL_SIZE
Каждый процесс захвата требует 10M в памяти shared pool для буфера очереди; в то же время все нужды Oracle Streams в shared pool не могут занимать более 10% этой области. С версии 10, при включенной автоматической настройке областей памяти в SGA, устанавливать конкретно этот параметр нет нужды, а упомянутые запросы памяти следует учесть при установлении SGA_MAX_SIZE или MEMORY_MAX_TARGET (с версии 11).

SGA_MAX_SIZE
(Если речь идет о версиях 10+). Значение должно учитывать нужды частей SGA (см. выше), особенно для выполнения захвата изменений с помощью LogMiner. Пример, приводимый ниже, в силу его простоты работает даже при значении SGA_MAX_SIZE = 400m.

Конфигурация БД

БД, поддерживающая процесс захвата изменений, должна работать в режиме архивирования.

БД, поддерживающая процесс захвата изменений, должна обеспечить на уровне отдельных таблиц или всей БД режим расширенной журнализации (supplemental logging). В этом режиме журнальные записи об изменениях в таблицах заносятся в расширенном формате, включая данные старых и новых значений полей

(независимо от того, какие поля фактически изменялись) для того, чтобы процесс применения изменения в принимающей СУБД смог однозначно воспроизвести изменение.

Расширенную журнализацию можно включать не обязательно для всей БД, но достаточно для реплицируемых таблиц. Значение столбца в таблице исходной БД должно безусловно (ALWAYS, *unconditionally*) попадать в журнал, если соответствующий столбец в таблице принимающей БД:

- индексирован (хотя бы вследствие имеющегося ограничения целостности),
- участвует в правиле преобразования данных или обрабатывается программой обработки (handler).

Как БД-источник, так и БД-получатель используют рабочие таблицы для хранения данных очередей и прочих нужд. Для их размещения целесообразно выделить отдельные табличные пространства. В БД-источнике желательно назначить процессу LogMiner табличное пространство, иное, чем SYSTEM.

Системные пакеты

Технологически организация потоков осуществляется через употребление ряда встроенных пакетов из схемы SYS:

DBMS_APPLY_ADM
DBMS_CAPTURE_ADM
DBMS_PROPAGATION_ADM
DBMS_STREAMS_ADM
DBMS_STREAMS
DBMS_STREAMS_MESSAGING
DBMS_RULE_ADM
DBMS_RULE
DBMS_STREAMS_AUTH[10.2-]
DBMS_STREAMS_TABLESPACE_ADM[10.2-]
[10.2-] Начиная с версии 10.2.

Пример построения потока изменений

В этом примере БД-источник потока носит имя MAINDB.CLASS, БД-приемник потока носит имя SUBDB1.CLASS. Сетевые имена баз в Oracle Net соответственно SOURCE и DESTINATION. Предполагается, что в обеих БД имеется схема SCOTT.

Пример приводится для версии 10.2. Предполагается, что команды выдаются в SQL*Plus.

Подготовка

Переведем БД-источник в режим архивирования журнальных файлов:

```
CONNECT /@source AS SYSDBA
STARTUP MOUNT FORCE
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

Создадим рабочие табличные пространства в обоих БД, например:

```
CREATE TABLESPACE streams_ts
DATAFILE 'C:\oracle\oradata\maindb\streams_ts.dbf' SIZE 25m;

CONNECT /@destination AS SYSDBA
CREATE TABLESPACE streams_ts
DATAFILE 'C:\oracle\oradata\subdb1\streams_ts.dbf' SIZE 25m;
```

В версия 9.2 в БД-источнике желательно назначить процессу LogMiner табличное пространство, отличное от SYSTEM, например:

```
CONNECT /@source AS SYSDBA
EXECUTE DBMS_LOGMNR_D.SET_TABLESPACE ( 'TOOLS' )
```

С версии 10 хороший кандидат – табличное пространство SYSAUX.

В обеих базах создадим администратора потоков:

```
CONNECT /@source AS SYSDBA

CREATE USER streamadmin IDENTIFIED BY streamadmin
DEFAULT TABLESPACE streams_ts
TEMPORARY TABLESPACE temp
QUOTA UNLIMITED ON streams_ts
;

GRANT dba TO streamadmin;

EXECUTE DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE ( 'streamadmin' )
```

Повторить те же действия для SUBDB1.CLASS.

В БД-источнике заведем связь с БД-получателем. Так как БД-получатель именована глобально, имя связи обязано совпадать с этим глобальным именем:

```
CONNECT streamadmin/streamadmin@source

CREATE DATABASE LINK subdb1.class
CONNECT TO streamadmin
IDENTIFIED BY streamadmin
USING 'destination'
;
```

Формирование потоков

Создадим очередь для передачи событий в БД-источнике и очередь для применения событий в БД-получателе, например:

```
EXECUTE DBMS_STREAMS_ADM.SET_UP_QUEUE ( )

CONNECT streamadmin/streamadmin@destination

EXECUTE DBMS_STREAMS_ADM.SET_UP_QUEUE ( )
```

Коли указано специально, очереди в обеих БД (и таблицы для данных этих очередей) получили умолчательные названия. Их можно наблюдать так:

```
SQL> CONNECT streamadmin/streamadmin@source
Connected.
SQL> SELECT name, queue_table FROM user_queues;

NAME                           QUEUE_TABLE
------------------------------ ------------------------------
STREAMS_QUEUE                  STREAMS_QUEUE_TABLE
AQ$_STREAMS_QUEUE_TABLE_E      STREAMS_QUEUE_TABLE
```

Очередь AQ$_*_E создается автоматически для сообщений об ошибках обработки событий.

Для возможности передавать потоком изменения в исходной таблице SCOTT.EMP требуется заявить расширенную журнализацию хотя бы для этой таблицы:

```
CONNECT scott/tiger@source

ALTER TABLE emp ADD SUPPLEMENTAL LOG DATA ( PRIMARY KEY ) COLUMNS;
```

Проверка:

```
SQL> SELECT always, table_name, log_group_type FROM user_log_groups;

ALWAYS      TABLE_NAME                     LOG_GROUP_TYPE
----------- ------------------------------ ------------------
ALWAYS      EMP                            PRIMARY KEY LOGGING
```

Теперь правка *любого* поля в таблице EMP будет сопровождаться (*безусловно*) занесением в журнал не только старого и нового значений этого поля, но также и значения ключевого поля (то есть EMPNO).

В БД-источнике создадим процесс захвата изменений, одновременно указав правила отбора изменений в очередь:

```
CONNECT streamadmin/streamadmin@source

BEGIN
DBMS_STREAMS_ADM.ADD_TABLE_RULES (
  table_name     => 'scott.emp'
, streams_type   => 'capture'
, streams_name   => 'capture_stream'
, include_ddl    => TRUE
);
END;
/
```

Проверка:

```
SQL> SELECT capture_name, queue_name, queue_owner, status
  2  FROM all_capture;

CAPTURE_NAME       QUEUE_NAME         QUEUE_OWNER        STATUS
------------------ ------------------ ------------------ --------
CAPTURE_STREAM     STREAMS_QUEUE      STREAMADMIN        DISABLED
```

Среди прочих умолчаний при создании процесса захвата изменений выше использовано подразумеваемое молчаливо имя очереди STREAMS_QUEUE. В нашем случае это можно было бы обозначить явно, указав параметр QUEUE_NAME => 'streamadmin.streams_queue'. Этим же параметром можно воспользоваться, когда процесс захвата потребуется связать с очередью под иным именем.

Правила отбора изменений в очередь STREAMS_QUEUE также были построены автоматически, но могли бы быть дополнены, или даже выписаны явно с помощью других параметров процедуры ADD_TABLE_RULES.

Создадим процесс переноса изменений:

```
BEGIN
DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES (
  table_name            => 'scott.emp'
, streams_name          => 'maindb_to_subdb1'
, source_queue_name     => 'streamadmin.streams_queue'
, destination_queue_name
                        => 'streamadmin.streams_queue@subdb1.class'
, source_database       => 'maindb.class'
, include_ddl           => TRUE
);
END;
/
```

Проверка:

```
SQL> SELECT propagation_name, source_queue_name,
  2         destination_queue_name, status
  3  FROM dba_propagation;
```

```
PROPAGATION_NAME SOURCE_QUEUE_NAME DESTINATION_QUEUE_NAM STATUS
---------------- ----------------- -------------------- -------
MAINDB_TO_SUBDB1 STREAMS_QUEUE     STREAMS_QUEUE         ENABLED
```

Теперь для правильного воспроизведения изменений в принимающей БД требуется передать ей в качестве «точки отсчета» номер изменений в БД-источнике. Передаваться получателям будут только изменения в EMP с номерами более поздними:

```
BEGIN
DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN@subdb1.class (
  source_object_name   => 'scott.emp'
, source_database_name => 'maindb.class'
, instantiation_scn    => DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER
);
END;
/
```

Убедиться в учете процессом применения для таблиц точки отсчета можно запросом:

```
SQL> COLUMN source_database FORMAT A20
SQL> SELECT
  2    source_object_name, source_object_type, instantiation_scn
  3  FROM  dba_apply_instantiated_objects@subdb1.class;

SOURCE_OBJECT_NAME             SOURCE_OBJE INSTANTIATION_SCN
------------------------------ ----------- -----------------
EMP                            TABLE                 1200698
```

Принимающая БД готова к активации процесса применения изменений:

```
CONNECT streamadmin/streamadmin@destination

BEGIN
DBMS_STREAMS_ADM.ADD_TABLE_RULES (
  table_name      => 'scott.emp'
, streams_type    => 'apply'
, streams_name    => 'apply_stream'
, source_database => 'maindb.class'
, include_ddl     => TRUE
);
END;
/
```

Проверка:

```
SQL> SELECT apply_name, queue_name, status FROM all_apply;

APPLY_NAME                QUEUE_NAME                STATUS
------------------------- ------------------------- --------
APPLY_STREAM              STREAMS_QUEUE             DISABLED
```

Для удобства отключим реакцию на ошибки, иначе процесс применения изменений может самопроизвольно прекращаться:

```
BEGIN
DBMS_APPLY_ADM.SET_PARAMETER (
  apply_name  => 'apply_stream'
, parameter   => 'disable_on_error'
, value       => 'N'
);
END;
/
```

Осталось запустить процессы захвата и примения изменений:

```
CONNECT streamadmin/streamadmin@source

EXECUTE DBMS_CAPTURE_ADM.START_CAPTURE ( 'capture_stream' )

EXECUTE -
DBMS_APPLY_ADM.START_APPLY@subdb1.class ( 'apply_stream' )
```

Проверка

Проверка:

```
SQL> CONNECT streamadmin/streamadmin@source
Connected.
SQL> SELECT empno FROM scott.emp MINUS
  2  SELECT empno FROM scott.emp@subdb1.class
  3  .
SQL> SAVE delta REPLACE
Wrote file delta.sql
SQL> @delta

no rows selected

SQL> INSERT INTO scott.emp ( empno ) VALUES ( 3333 );

1 row created.

SQL> @delta

     EMPNO
----------
      3333

SQL> COMMIT;

Commit complete.

SQL> @delta

no rows selected
```

Заметьте, что поток переносит изменения только в одну сторону. Таблица-приемник при этом *не* закрыта для обычной правки. Однако же такую правку следует выполнять осмотрительно, поскольку она может привести к ошибкам при автоматическом изменении данных потоком (при необходимости эта проблема решается специально средствами разрешении конфликтов). Вдобавок учтите, что множественные операции INSERT, UPDATE, DELETE применяются в принимающей БД в рамках одной (автономной) транзакции (невзирая на то, что в журнале БД множественные изменения фиксируются набором однострочных изменений). Следовательно ошибка при попытке изменения хотя бы одной-единственной строки приведет к отказу от множественной операции со всеми ее изменениями.

Упражнение. Внести изменения в таблицу SCOTT.EMP на принимающей БД. Убедиться в сохраняющихся расхождениях в таблицах БД-источника и БД-получателя.

Упражнение. Проверить передачу изменений DDL. Добавить столбец в таблицу SCOTT.EMP@MAINDB.CLASS. Наблюдать результат в SCOTT.EMP@SUBDB1.CLASS. Изменить тип столбца, наблюдать результат в базе-получателе.