Программа переноса данных из СУБД Oracle в СУБД PostgreSQL (Ora2PgCopy)

Руководство пользователя

Всего страниц 8

Оглавление

1. Назначение 2
2. Требования 2
3. Установка программы 2
4. Старт и эксплуатация программы 5
5. Особенности программы 6

1. Назначение

Основной задачей программы Ora2PgCopy является перенос данных из СУБД Oracle в данные СУБД PostgreSQL с максимально возможной скоростью, разработана на Java для совместимости с различными операционными системами и использует PostgreSQL-команду COPY для обеспечения высокой скорости загрузки данных из внешних файлов.

Ora2PgCopy применяется в ходе процесса миграции в тот момент, когда в PostgreSQL уже созданы таблицы и перенесён программный код. Для обеспечения более быстрой миграции целесообразно индексы, ограничения целостности и активацию триггеров выполнять после переноса данных.

Ora2PgCopy может функционировать как модуль в составе системы автоматизации миграции (CAM) LUI4ORA2PG, так и независимо от неё.

2. Требования

Программа Ora2PgCopy работает на платформах, поддерживающих работу с Java, СУБД PostgreSQL, СУБД Oracle.

Минимальный объём дисковой памяти для установки программы Ora2PgCopy составляет 10 МБ.

Используются языки программирования: Java, SQL, PL/pgSQL, PL/SQL

Для сборки и работы программы необходимы библиотеки jar:

- commons-io-2.4.jar
- gson 2.7.jar
- joda-time-2.11.0.jar
- ojdbc8-19.8.0.0.jar
- postgresql-42.2.14.jar
- ru.fors.utils.lightsmtp.jar

3. Установка программы

Установка программы выполняется распаковкой файла дистрибутива ora2pgcopy.zip, который можно получить на сайте http://ora2pgcopy.forstelecom.ru или по запросу на e-mail support@forstelecom.ru .

Структура каталогов после распаковки:

./ora2pgcopy/lib/

commons-io-2.4.jar
gson 2.7.jar
joda-time-2.11.0.jar
ojdbc8-19.8.0.0.jar
postgresql-42.2.14.jar
ru.fors.utils.lightsmtp.jar

./ora2pgcopy/

ora2pgcopy-start.sh
ora2pgcopy-start.bat
ora2pgcopy.json
ora2pgcopy.jar

./ora2pgcopy/doc/

Ora2PgCopy.Referat.docx
Ora2PgCopy.Users guide.docx

Описание примера файла конфигурации проекта ora2pgcopy.json :

```
{ /* пример JSON - структуры для утилиты ora2pgcopy.*/
  "sessions": 7, /*количество потоков - максимальное количество одновременно
                  выгружаемых/загружаемых таблиц. необязательный параметр. default=7*/

   /* "outfile": "myoutfile",*/ /*имя sh-файла, в котором будут генерироваться вызовы
для последовательных загрузок утилитой psql. необязательный параметр. Если не указан файл не
будет создаваться.*/

  "truncate": true, /*необходимость выполнить в PG trucate table перед загрузкой.
                      необязательный параметр. default=false*/

  "zip":false, /* следует ли сжимать выходные sql-файлы утилитой zip. Необязательный
                  параметр. default=false*/

  "oradb": "usr:1521:dbs", /*jdbc-url для подключения к oracle. <хост>:<порт>:<sid>
                             или //<хост>/<сервис> .*/

  "orausername": "magnit",/*имя пользователя oracle*/
  "orapassword": "magnit",/*пароль пользователя oracle*/
  "pghost": "localhost",  /*имя или ip-адрес PG-сервера*/
  "pgdb": "postgres",     /*имя базы данных PG*/
  "pgport":"5432",        /*порт листенера PG. необязательный параметр. default=5432*/
  "pgusername": "pguser", /*имя пользователя PG*/
  "pgpassword": "pgpswd", /*пароль пользователя PG*/

  "set2unlogged":false,  /*следует ли перед загрузкой в PG выполнять для таблицы
                           alter table set unlogged. необязательный параметр.
                           default=false*/

  "set2logged":false,     /*следует ли после загрузки в PG выполнять для таблицы
                            alter table set logged. необязательный параметр.
                            default=false*/

  "unload":"if target empty", /*следует ли выполнять выгрузку из oracle. Возм.
                                значения: "yes","no","if target empty". Необязательный
                                параметр. default="yes".
                               "if target empty" - выгрузка будет выполнться если в
                                таблице PG нет строк. */

  "load":"directbinary", /*следует ли выполнять загрузку в PG. Возм. значения:
                          "yes","no","if target empty","direct". Необязательный
                           параметр. default="no".
                           "if target empty" - загрузка будет выполняться если в
                           таблице PG нет строк.
                            "direct" - выгружаемые данные загружаются напрямую без
                           промежуточного sql-файла. "direct" допустим, если
                           unload="yes" или "if target empty"
```

```
                                "directbinary" - выгружаемые данные загружаются напрямую с
                               использованием двоичного формата команды copy "directbinary"
                               допустим, если unload="yes" или "if target empty"*/


        "index":"no", /*следует ли выполнять операторы индексирования, если они указаны.
                        Возм. значения: "yes","no","if not exist". необязательный параметр.
                        default="yes".
                        "if not exists" подавляет ошибки типа "такое уже есть"*/

        "sessionwriters":4,  /*количество подпроцессов паралельной выгрузки (в случае
                              "direct" и загрузки в PG) таблицы из oracle. Необязательный
                               параметр. default=8
                               замечание: если выгрузка таблицы выполняется в несколько
                        подпроцессов (т.е. sessionwriters >1), то в строки в PG могут
                        оказаться загруженными не в том порядке, в каком они выгружались*/

        "prefetchrows":1024,  /*количество строк опережающего fetch из oracle для снижения
                               частоты обращений к нему. необязательный параметр.
                               default=256 */

        "lobprefetchsize":262144, /*количество байт опережающего fetch из oracle LOB для
                                    снижения частоты обращений к нему. Необязательный
                                    параметр. default=32768 */

        "lobstreambuffer":262144, /*размер буфера для выкачивания данных из потока
                                    oracle LOB. необязательный параметр. default=16384 */

        "work": [ /*массив описаний таблиц для выгрузки/загрузки. В каждом элементе массива
                    могут быть переопределены параметры, описанные выше. (кроме sessions и
                    sendmail, конечно)
                    замечание: если переопределить параметры подключения к oracle и pg, то
                            можно выкачивать данные из разных баз и закачивать в разные базы
                    замечание: подключение к PG указанное на верхнем уровне используется для
                               измерения скоростей загрузок и индексирования в PG,
                               и PG-подключения, заданные на уровне элемента массива
                              выпадают из области измерений скоростей*/
          {
          "query": "select * from TBL_SYS_FILECONTENT01",
          "target": "lui.TBL_SYS_FILECONTENT01",
          "locolumns": "CONTENT1;CONTENT2"; /* перечень столбцов с large object – применять
если в oracle LOB>1Гб, разделитель - точка с запятой, имена д.б. в верхнем регистре */
          "freeze": true
        }
        ]

        /* следить за ходом выгрузки/загрузки/индексирования можно с помощью запроса:
      select
      case row_number() OVER()::text
                     when '1' then 'TotalTables'
                     when '2' then 'TotalUnloaded'
                     when '3' then 'TotalLoaded'
                     when '4' then 'TotalIndexed'
                     when '5' then 'NowUnloading'
                     when '6' then 'NowLoading'
                     when '7' then 'NowIndexing'
                     when '8' then 'UnloadingSpeed'
                     when '9' then 'LoadingSpeed'
                     when '10' then 'IndexSpeed'
                     when '11' then 'Errors'
                     end code,
                     r::numeric value
      from(
        select   regexp_split_to_table(replace(application_name,'ora2pgcopy ',''),';') r
```

```
        from pg_stat_activity
        where application_name like 'ora2pgcopy %')q
         where q.r>'';



         следить за ходом создания больших объектов можно с помощью запроса:
       select
       case row_number() OVER()::text
                       when '1' then 'Object'
                       when '2' then 'Bytes'
                       end code,
       r::numeric value
       from (select regexp_split_to_table(REPLACE(application_name,'ora2pgcopy: loading large
object ',''),' ') r
       from pg_stat_activity where application_name like 'ora2pgcopy: loading large objects%'
       )q;
       */
       }
```

4. Старт и эксплуатация программы

Программа выполняется в пакетном режиме специально подготовленным скриптом, например ora2pgcopy-start.sh, где в приведённом примере выделяется 8 ГБ ОЗУ для JVM (это при необходимости работы с большим числом параллельных потоков, обычно не надо указывать этот параметр):

```
java -Xmx8192m -Dfile.encoding=UTF-8 -jar ora2pgcopy.jar ora2pgcopy.json
```

После старта утилита выводит строки диагностики выполнения в стандартный поток данных, также образуются файлы данных и возникающих ошибок.

Настройка работы программы выполняются в конфигурационном файле, см. пример файла ora2pgcopy.json в разделе п.3, где приведено описание параметров настроек.

Программа не создаёт таблицы, целевая таблица должна уже существовать.

Имена целевых столбцов должны совпадать с именами столбцов запроса, порядок столбцов не имеет значения.

Типы данных каждого столбца должны быть совместимы.

Как это работает. Утилита подключается к базе данных Oracle, выбирает данные из представления/таблицы и записывает данные в файл дампа базы данных PostgreSQL. Этот файл, в свою очередь, может быть прочитан утилитой psql для импорта данных в Postgres. Файл дампа можно сжать в формате gzip.

Ещё один сгенерированный файл ./myoutfile.sh (в зависимости от настроек) — это сценарий оболочки для импорта в PostgreSQL, который можно запустить в любой системе Posix, в которой есть gunzip и psql.

5. Особенности программы

Java. Программа Ora2PgCopy разработана на Java для обеспечения независимости от типа операционной системы (ОС), при этом поддерживается версии JVM от 8 до 19. Поддержка устаревших версий Java объясняется их присутствием в дистрибутивах во многих отечественных ОС.

COPY. Для переноса данных программа Ora2PgCopy применяет команду COPY СУБД PostgreSQL, которая быстрее всех иных средств загружает данные в таблицы Postgres из внешних источников.

Ora2PgCopy поддерживает бинарный формат входных данных команды COPY. Бинарный формат в ряде случаев работает существенно быстрее текстового. Например, когда таблица состоит из небольшого количества числовых столбцов и/или столбцов типов дата/время. А также в том случае, когда в Postgres по сети передаются данные типа bytea (двоичная строка переменной длины).

Многопоточность. Ora2PgCopy использует многопоточные технологии Java. Одновременно выполняется обработка нескольких таблиц. При этом, обработка каждой таблицы выполняется также в несколько потоков.

LOB/CLOB. Особое внимание в ходе разработки было уделено переносу данных типа LOB и CLOB:

- учтены рекомендации Oracle Corp. по разработке программ для быстрого чтения таких данных
- реализована миграция в Postgres LOB и CLOB полей, объём которых превышает 1Гб. Известно, что Postgres не может в одном поле таблицы хранить данные длиннее одного гигабайта. Для таких данных в Postgres применяется специальная технология “large objects”.

Сценарии. Новое средство миграции данных Ora2PgCopy поддерживает следующие сценарии миграции данных:

Сценарий 1 – Выгрузка.

Выгрузка всех требуемых таблиц из Oracle в файлы, которые в дальнейшем обрабатываются утилитой psql (аналог Oracle SQL*Plus в Postgres). Эти файлы содержат SQL-команду COPY и массивы данных для неё. Полученные файлы в последующем могут обрабатываться уже отдельно вне зоны внимания системы и модуля миграции данных. Кроме того, полученные файлы могут обрабатываться модулем Ora2PgCopy. В этом случае программа автоматически запустит psql в несколько потоков. Данный сценарий может быть применён для тех случаев, когда физический перенос носителя с файлами из сервера Oracle в сервер Postgres выполняется быстрее, чем передача данных по сети.

Сценарий 2 – Выгрузка/загрузка.

Выгрузка всех требуемых таблиц из Oracle в файлы. Причём, как только таблица целиком выгружена в файл, сразу же начинается его загрузка в Postgres. После успешной загрузки файл удаляется.

Сценарий 3 – Прямое копирование.

Чтение данных из Oracle с их одновременной загрузкой в Postgres без создания промежуточных файлов. В этом режиме посредничество утилиты psql не требуется загрузка данных выполняется посредством Java-API команды COPY, который разработан и поставляется Postgre-сообществом.

Этот сценарий самый быстрый и удобен тогда, когда места для хранения промежуточных файлов практически нет. Однако, в случае возникновения ошибки при загрузке в Postgres какой-либо таблицы, хотя в журналах сохранится сообщение об ошибке и номер строки исходных данных, но, отсутствие промежуточного файла с данными вызовет некоторые трудности с поиском причины ошибки. Поэтому этот режим следует применять после итеративной отладки процесса миграции с помощью сценариев 1 и/или 2.

Доп. возможности. Дополнительные возможности Ora2PgCopy:

- усечение таблиц перед загрузкой
- многопоточное индексирование после загрузки
- уведомление по e-mail о нормальном или аварийном завершении
- перевод требуемых таблиц в nologged - состояние перед загрузкой и logged после загрузки
- источником данных может быть не только таблица, но и любой sql-запрос в Oracle
- источниками данных могут быть несколько разных баз данных (экземпляров) Oracle
- целевые таблицы могут быть размещены в нескольких разных базах данных Postgres
- широкий набор настроек: количество одновременно обрабатываемых таблиц, количество потоков на таблицу, размеры буферной памяти и т.д.
- преобразование символьных и числовых данных к boolean
- за ходом процесса можно наблюдать не только на консоли, но и запросами в Postgres, извлекая следующие данные:
  - количество таблиц в процессе выгрузки
  - количество таблиц в процессе загрузки
  - количество таблиц в процессе индексирования
  - количество выгруженных таблиц

- количество загруженных таблиц
- количество проиндексированных таблиц
- текущая скорость выгрузки, загрузки и индексирования.

Заключение. Применение программы Ora2PgCopy позволяет минимизировать время простоя при переключении пользователей с прикладной системы, работающей на СУБД Oracle на систему, основанную на СУБД PostgreSQL. Минимизация времени простоя достигается за счёт уникально высокой скорости миграции данных.

По результатам тестов Ora2PgCopy работает заметно быстрее таких аналогов как: Ispirer (convertum), oracle_fdw, ora2pg, Pentaho kettle.

Таблица сравнение с аналогами по скорости загрузки данных в таблицы PostgreSQL из внешних файлов:

Инструмент	Скорость загрузки данных	
Ispirer (convertum)		-
		LOB
oracle_fdw		-
		LOB
ora2pg		-
		LOB
Pentaho kettle		-
		LOB
Ora2PgCopy		-
		LOB
