ora2pgpro

ora2pgpro

Глава 1. ora2pgpro

Описание

ora2pgpro — это утилита, используемая при переносе базы данных из Oracle™ в схему, совместимую с Postgres Pro™. Утилита подключается к БД Oracle, автоматически сканирует её и извлекает её структуру или данные, а затем генерирует SQL-скрипты, которые можно загрузить в базу данных Postgres Pro. ora2pgpro можно использовать для разных целей от обратного проектирования БД Oracle до миграции баз данных крупных предприятий или для простого переноса данных из БД Oracle в БД Postgres Pro. Для использования утилиты не требуются знания БД Oracle — достаточно предоставить параметры подключения к БД Oracle.

Функциональные возможности

Утилита ora2pgpro состоит из Perl-скрипта `ora2pgpro` и Perl-модуля `Ora2PgPro.pm`. Для использования утилиты достаточно указать в файле конфигурации `ora2pgpro.conf` DSN для подключения к БД Oracle и, при необходимости, имя схемы. После этого необходимо указать тип экспорта: `TABLE` с ограничениями, `VIEW`, `MVIEW`, `TABLESPACE`, `SEQUENCE`, `INDEXES`, `TRIGGER`, `GRANT`, `FUNCTION`, `PROCEDURE`, `PACKAGE`, `PARTITION`, `TYPE`, `INSERT` или `COPY`, `FDW`, `QUERY`, `SYNONYM`.

По умолчанию ora2pgpro экспортирует данные в файл, который можно загрузить в БД Postgres Pro, используя клиент psql. Кроме того, можно настроить импорт напрямую в БД Postgres Pro, указав DSN в файле конфигурации. Множество параметров конфигурации `ora2pgpro.conf` позволяют гибко управлять процессом экспорта. Доступны следующие функциональные возможности:

- Полный экспорт схемы БД (таблицы, представления, последовательности, индексы) с ограничениями уникальности, первичным ключом, внешними ключами и ограничениями-проверками.
- Экспорт прав пользователей и групп.
- Экспорт секций по спискам и по диапазонам.
- Экспорт нескольких таблиц (с указанием имён).
- Экспорт схемы Oracle как схемы Postgres Pro.
- Экспорт предопределённых функций, триггеров, процедур, пакетов и тел пакетов.
- Экспорт всех данных или выбранных предложением `WHERE`.
- Полная поддержка экспорта объектов BLOB Oracle как bytea Postgres Pro.
- Экспорт представлений Oracle как таблиц Postgres Pro.
- Экспорт определяемых пользователем типов Oracle.
- Базовое автоматическое преобразование кода PL/SQL в код PL/pgSQL.
- Работа на любой платформе.
- Экспорт таблиц Oracle как таблиц, реализованных через обёртку сторонних данных.
- Экспорт материализованных представлений.
- Вывод отчёта о содержимом БД Oracle.
- Оценка стоимости миграции БД Oracle.
- Оценка сложности миграции БД Oracle.

- Оценка стоимости миграции кода PL/SQL из файла.
- Оценка стоимости миграции SQL-запросов Oracle, хранящихся в файле.
- Экспорт Oracle Locator и пространственных данных в PostGIS.
- Экспорт `DBLINK` в Oracle FDW.
- Экспорт `SYNONYM` как представлений.
- Экспорт `DIRECTORY` как внешних таблиц или каталога для расширения `external_file`.
- Передача SQL-команд с использованием нескольких соединений с Postgres Pro.
- Сравнение БД Oracle и Postgres Pro для целей тестирования.

Утилита ora2pgpro делает всё возможное, чтобы преобразовать БД Oracle в Postgres Pro автоматически, но некоторые действия необходимо выполнить вручную, например, проверить код PL/SQL, сгенерированный для функций, процедур, пакетов и триггеров, на соответствие синтаксису Postgres Pro. Некоторые рекомендации по портированию кода из PL/SQL в PL/pgSQL даны в разделах Портирование из Oracle PL/SQL и Инструменты миграции в Postgres Pro.

Установка и подготовка

Установка Oracle Instant Client

Утилите ora2pgpro требуется Oracle Instant Client, который не входит в состав ora2pgpro. Необходимо скачать RPM-пакет `oracle-instantclient12.1-basic` (например, файл `oracle-instantclient12.1-basic-12.1.0.2.0-1.x86_64.rpm`) отсюда или из другого источника и установить его. Для ora2pgpro используется старая версия 12.1, поскольку новые версии 19 и 21 могут обновиться в любой момент, и пакетный менеджер может посчитать их несовместимыми с пакетами Postgres Pro.

При использовании системы на базе Debian (Debian, Ubuntu, Astra Linux) необходимо конвертировать RPM-пакет в Debian-пакет, воспользовавшись alien или dpkg.

Подключение дистрибутива

- Загрузите скрипт подключения `pgpro-repo-add.sh`. Это также можно сделать, используя wget:

```
  $ wget --user your-username --ask-password https://
repo.postgrespro.ru/ora2pgpro/keys/pgpro-repo-add.sh
```

- Запустите его с правами `root`. Если дистрибутив поддерживается, то репозиторий будет подключён. Обратите внимание, что для системы на базе Debian следует подключить не только раздел `main`, но и раздел `contrib`, поскольку необходимый пакет `libdbd-oracle-perl` находится именно в нём.

Установка пакета

Установите пакет ora2pgpro с помощью пакетного менеджера. В зависимости от дистрибутива это может быть:

- `apt-get`: Debian, Ubuntu, Astra, Альт Линукс

- `dnf`: Системы Red Hat Enterprise Linux (RHEL) и производные от них: РЕД ОС, РОСА
- `yum`: старые версии РОСА
- `zypper`: SLES

В любом случае укажите параметры `install ora2pgpro`.

Поддерживаемые дистрибутивы

- Альт Линукс 10
- Astra Linux «Орёл» 2.12
- Debian 10 (buster) и 11 (bullseye)
- РЕД ОС 7.3
- Red Hat Enterprise Linux 8/9
- Rosa Fresh 2021.1
- SUSE Linux Enterprise Server 15
- Ubuntu 20.04 (focal) и 22.04 (jammy)

Настройка

Настройка ora2pgpro заключается в выборе экспортируемой БД Oracle и типа экспорта.

Используя один файл конфигурации `ora2pgpro.conf`, можно полностью управлять процессом миграции БД Oracle. В файле записывается имя директивы заглавными буквами и её значение после символа табуляции. Строки, начинающиеся с `#`, представляют собой комментарии. Директивы можно задавать в любом порядке, а включаются они при считывании значения в файле конфигурации.

Директивы, принимающие одно значение, можно задавать в файле конфигурации несколько раз, при этом будет использоваться последнее вхождение. Директивы, принимающие список значений, также можно задавать несколько раз, и значения из последующих вхождений будут добавляться в список. Если для загрузки пользовательского файла конфигурации используется директива `IMPORT`, директивы из этого файла будут сохраняться в файле конфигурации после директивы `IMPORT`, поэтому рекомендуется задавать её в конце файла конфигурации.

Значения, переданные в командой строке, переопределяют значения из файла конфигурации.

Использование

Сначала необходимо убедиться, что пути ко всем библиотекам и двоичным файлам включают каталог установки Oracle Instant Client:

```
export LD_LIBRARY_PATH=/usr/lib/oracle/11.2/client64/lib
export PATH="/usr/lib/oracle/11.2/client64/bin:$PATH"
```

По умолчанию ora2pgpro выполняет поиск файла конфигурации `/etc/ora2pgpro/ora2pgpro.conf`, и если этот файл существует, достаточно выполнить команду:

```
/usr/local/bin/ora2pgpro
```

Если необходимо вызвать другой файл конфигурации, задайте путь в командной строке:

```
/usr/local/bin/ora2pgpro -c /etc/ora2pgpro/new_ora2pgpro.conf
```

Ниже приведены все доступные параметры командной строки для ora2pgpro:

```
ora2pgpro [-dhpqv --estimate_cost --dump_as_html] [--параметр
 значение]
```

`-a, --allow`

Задать список экспортируемых объектов, разделённых запятыми. Параметр может также использоваться с типом `SHOW_COLUMN`.

`-b, --basedir`

Задать выходной каталог для файлов, полученных в результате экспорта.

`-c, --conf`

Указать файл конфигурации, отличный от используемого по умолчанию `/etc/ora2pgpro/ora2pgpro_scanner.conf`.

`-C, --cdc_file`

Указать файл для хранения/чтения SCN таблиц во время экспорта. По умолчанию: `TABLES_SCN.log` в текущем каталоге. Для создания этого файла используйте параметр `--cdc_ready`.

`-d, --debug`

Выводить подробные сообщения.

`-D, --data_type`

Разрешить использование пользовательских типов в командной строке.

`-e, --exclude`

Исключить из экспорта список указанных объектов, разделённых запятыми. Может использоваться с типом `SHOW_COLUMN`.

`-h, --help`

Выводить краткую справку.

`-g, --grant_object`

Извлекать права для данного типа объектов. Возможные значения даны в описании конфигурации `GRANT_OBJECT`.

`-i, --input`

Указать файл, содержащий код PL/SQL для портирования, когда подключение к БД Oracle отсутствует.

`-j, --jobs`

Задать число параллельных процессов при передаче данных в Postgres Pro.

`-J, --copies`

Задать число параллельных соединений для извлечения данных из Oracle.

`-l, --log`

Указать файл журнала. По умолчанию: `stdout`.

`-L, --limit`

Задать количество кортежей, извлекаемых из Oracle и сохраняемых в памяти до записи, по умолчанию 10000.

`-n, --namespace`

Указать схему Oracle для извлечения данных.

`-N, --pg_schema`

Задать `search_path` для Postgres Pro.

`-o, --out`

Указать путь к выходному файлу, в который записываются команды SQL. По умолчанию: `output.sql` в текущем каталоге.

`-p, --plsql`

Включить преобразование кода PL/SQL в код PL/pgSQL.

`-P, --parallel`

Задать число таблиц для параллельного извлечения данных.

`-q, --quiet`

Отключить вывод индикатора выполнения.

`-r, --relative`

Использовать параметр `\ir` вместо `\i` в создаваемых скриптах psql.

`-s, --source`

Задать источник данных DBI для Oracle.

`-S, --scn`

Задать системный номер изменения Oracle (System Change Number, SCN) для экспорта данных. Он используется в предложениях `WHERE` для получения данных с типами экспорта `COPY` или `INSERT`.

`-t, --type`

Задать тип экспорта. Параметр переопределяет значение, заданное в файле конфигурации (`TYPE`).

`-T, --temp_dir`

Указать отдельный временный каталог для одновременного запуска двух или более экземпляров ora2pgpro.

`-u, --user`

Указать пользователя для подключения к БД Oracle. Вместо этого параметра можно использовать переменную окружения `ORA2PG_USER`.

`-v, --version`

Показать версию ora2pgpro и прервать выполнение.

`-w, --password`

Указать пароль для пользователя БД Oracle. Вместо этого параметра можно использовать переменную окружения `ORA2PG_PASSWD`.

`-W, --where`

Задать предложение `WHERE`, добавляемое к запросу Oracle для получения данных. Параметр можно указывать несколько раз.

`--forceowner`

Принудительно назначать владельца таблиц и последовательностей, как в Oracle. Если в значении указать имя пользователя, он будет назначен владельцем. По умолчанию владельцем будет пользователь, используемый для подключения к БД Postgres Pro.

`--nls_lang`

Задать кодировку клиента Oracle в `NLS_LANG`.

`--client_encoding`

Задать кодировку клиента Postgres Pro.

`--view_as_table`

Указать список представлений, разделённых запятыми, для экспорта в виде таблиц.

`--estimate_cost`

Активировать оценку стоимости миграции для `SHOW_REPORT`.

`--cost_unit_value`

Задать число минут для оценки стоимости. По умолчанию 5 минут, что соответствует скорости миграции, выполняемой специалистом Postgres Pro. Для первой миграции установите значение 10.

`--dump_as_html`

Выгружать отчёт в формате HTML. Параметр используется только с `SHOW_REPORT`. По умолчанию отчёт выгружается в формате простого текста.

`--dump_as_csv`

Аналогично предыдущему, но отчёт выгружается в формате CSV.

`--dump_as_sheet`

Выводить отчёт по оценке миграции с отдельной строкой в формате CSV для каждой БД.

`--init_project`

Инициализировать стандартное дерево проекта ora2pgpro. Каталог верхнего уровня создаётся в базовом каталоге проекта.

`--project_base`

Указать базовый каталог для деревьев проекта ora2pgpro. По умолчанию это текущий каталог.

`--print_header`

Выводить заголовок CSV, что особенно полезно при первом запуске ora2pgpro. Параметр используется вместе с параметром `--dump_as_sheet`.

`--human_days_limit`

Задать предел человеко-дней, по достижении которого уровень оценки миграции меняется с B на C. По умолчанию 5 человеко-дней.

`--audit_user`

Указать список имён пользователей для фильтрации запросов в таблице `AUDIT_USER`. Используется только с типами экспорта `SHOW_REPORT` и `QUERY`.

`--pg_dsn`

Указать источник данных для прямого импорта в Postgres Pro.

`--pg_user`

Указать пользователя Postgres Pro.

`--pg_pwd`

Указать пароль пользователя Postgres Pro.

`--count_rows`

Принудительно подсчитывать действительное количество строк при выполнении `TEST`, `TEST_COUNT` и `SHOW_TABLE`.

`--no_header`

Не выводить заголовок ora2pgpro в выходной файл.

`--oracle_speed`

Узнать скорость, с которой могут передаваться данные из Oracle. Данные не обрабатываются и не записываются.

`--ora2pg_speed`

Узнать скорость, с которой ora2pgpro может отправлять преобразованные данные. Данные не записываются.

`--blob_to_lo`

Экспортировать BLOB в виде больших объектов. Параметр может использоваться только при выполнении `SHOW_COLUMN`, `TABLE` и `INSERT`.

`--cdc_ready`

Использовать текущие SCN для экспорта данных из таблиц и записать их в файл под названием `TABLES_SCN.log` по умолчанию. Другое имя файла можно указать, используя параметр `-C|--cdc_file`.

`--lo_import`

Использовать команду psql `\lo_import` для импорта BLOB в виде больших объектов. Параметр можно использовать для импорта с типом `COPY` и последующим импортом большого объекта вручную. Необходим для импорта BLOB размером более 1 ГБ.

`--mview_as_table`

Указать список материализованных представлений, разделённых запятыми, для экспорта в виде обычных таблиц.

`--drop_if_exists`

Удалить объект перед созданием, если он уже существует.

В случае успеха ora2pgpro возвращает 0, а в случае ошибки 1. Если дочерний процесс прерывается, возвращается 2 и выводится следующее предупреждение: «WARNING: an error occurs during data export. Please check what's happen.» (ПРЕДУПРЕЖДЕНИЕ: Во время экспорта данных произошла ошибка. Обратите внимание.) В большинстве случаев ошибка вызвана нехваткой памяти — попробуйте уменьшить значение `DATA_LIMIT`.

Обратите внимание, что производительность можно увеличить, обновив статистику в Oracle:

```
DBMS_STATS.GATHER_SCHEMA_STATS
DBMS_STATS.GATHER_DATABASE_STATS
DBMS_STATS.GATHER_DICTIONARY_STATS
```

Если заданы параметры `--project_base` и `--init_project`, ora2pgpro создаёт шаблон проекта, содержащий дерево проекта, файл конфигурации и скрипт для экспорта всех объектов из БД Oracle. Пример использования команды:

```
ora2pgpro --project_base /app/migration/ --init_project test_project
 Creating project test_project.
```

```
/app/migration/test_project/
 schema/
  dblinks/
  directories/
  functions/
  grants/
  mviews/
  packages/
  partitions/
  procedures/
  sequences/
  synonyms/
  tables/
  tablespaces/
  triggers/
  types/
  views/
 sources/
  functions/
  mviews/
  packages/
  partitions/
  procedures/
  triggers/
  types/
  views/
 data/
 config/
 reports/

Generating generic configuration file
Creating script export_schema.sh to automate all exports.
Creating script import_all.sh to automate all imports.
```

При этом создаётся стандартный файл конфигурации, в котором можно задать параметры подключения к БД Oracle, и скрипт оболочки `export_schema.sh`. Каталог `sources/` будет содержать код Oracle, каталог `schema/` — код для портирования в Postgres Pro. Каталог `reports/` будет содержать HTML-отчёты с оценкой стоимости миграции.

Если вы хотите использовать собственный файл конфигурации, укажите путь к нему в параметре `-c`. Допишите к имени файла суффикс `.dist`, чтобы добавить в него стандартные параметры конфигурации ora2pgpro, в противном случае файл будет копирован в исходном состоянии.

После установки подключения к БД Oracle можно выполнить скрипт `export_schema.sh`, чтобы экспортировать все типы объектов из БД Oracle и вывести файлы DDL в подкаталоги схемы. По завершении процесса, когда импортируемая схема будет проверена, будет выдана команда для последующего экспорта данных.

Можно загрузить созданные файлы DDL вручную или воспользоваться скриптом `import_all.sh`, чтобы импортировать файлы интерактивно. Если миграция не выполняется сразу, рекомендуется использовать эти скрипты.

Подключение к базе данных Oracle

Следующие директивы управляют доступом к БД Oracle.

ORACLE_HOME

Задаёт переменную окружения ORACLE_HOME для библиотек Oracle, используемых модулем Perl DBD::Oracle.

ORACLE_DSN

Задаёт имя источника данных в формате DBI DSN. Например:

```
dbi:Oracle:host=oradb_host.myhost.com;sid=DB_SID;port=1521
```

или

```
dbi:Oracle:DB_SID
```

Пример для 18c:

```
dbi:Oracle:host=192.168.1.29;service_name=pdb1;port=1521
```

Для использования второй нотации необходимо объявить SID в файле $ORACLE_HOME/network/admin/tnsnames.ora или в пути, указанном в переменной окружения TNS_ADMIN.

ORACLE_DSN ORACLE_PWD

Задают имя пользователя и пароль для подключения к БД Oracle. Обратите внимание, что лучше входить в систему с правами суперпользователя Oracle во избежание проблем доступа во время сканирования базы данных и пропуска данных.

Если не задать пароль в ORACLE_PWD и установить Perl-модуль Term::ReadKey, ora2pgpro запросит пароль интерактивно. Если имя пользователя не задано в ORACLE_USER, его тоже нужно будет задать интерактивно.

Чтобы подключиться к локальному экземпляру Oracle с правами SYSDBA, необходимо задать для ORACLE_USER значение / и пустой пароль.

USER_GRANTS

Задайте для этого параметра значение 1, если вы подключаетесь к БД Oracle как простой пользователь без права извлекать данные из таблиц DBA_. Будут использоваться таблицы ALL_.

Предупреждение: при использовании типа экспорта GRANT необходимо задать для этого параметра значение 0, в противном случае он работать не будет.

TRANSACTION

Эту директиву можно использовать, чтобы изменить уровень изоляции по умолчанию для экспортируемых транзакций. По умолчанию задаётся сери-

ализуемый уровень изоляции для сохранения целостности данных. Допустимые значения директивы
- readonly: 'SET TRANSACTION READ ONLY',
- readwrite: 'SET TRANSACTION READ WRITE',
- serializable: 'SET TRANSACTION ISOLATION LEVEL SERIALIZABLE'
- committed: 'SET TRANSACTION ISOLATION LEVEL READ COMMITTED',

`ORA_INITIAL_COMMAND`

Эту директиву можно использовать для отправки начальных команд в Oracle сразу после подключения, например, чтобы разблокировать политику для чтения объектов или установить параметры сеанса. Директиву можно задавать несколько раз.

Шифрование данных с использованием Oracle Server

Если файл конфигурации клиента Oracle уже содержит метод шифрования, `DBD:Oracle` использует эти параметры для шифрования соединения во время извлечения данных. Например, файл конфигурации клиента Oracle (`sqlnet.or` или `.sqlnet`) содержит следующую информацию:

```
# Настройка шифрования подключений к Oracle
SQLNET.ENCRYPTION_CLIENT = required
SQLNET.ENCRYPTION_TYPES_CLIENT = (AES256, RC4_256)
SQLNET.CRYPTO_SEED = '7-10 случайных символов'
```

Любая утилита, использующая клиент Oracle для взаимодействия с базой данных, будет защищена шифрованием, если настроить шифрование сеанса вышеописанным способом.

Например, Perl DBI использует `DBD-Oracle`, который в свою очередь использует клиент Oracle для взаимодействия с базой данных. Если при установке клиента Oracle, используемого Perl, было настроено требование зашифрованных соединений, то и соединение Perl с БД Oracle будет зашифровано.

Тестирование подключения

После установки DSN БД Oracle можно запустить утилиту ora2pgpro, чтобы проверить, работает ли она:

```
ora2pgpro -t SHOW_VERSION -c config/ora2pgpro.conf
```

Эта команда выведет версию сервера БД Oracle. Кроме того, можно проверить правильность установки, поскольку большинство проблем возникает именно на этом этапе настройки, остальные этапы являются более техническими.

Решение проблем

Если файл `output.sql` не содержит ничего, кроме заголовка и нижнего колонтитула транзакции Postgres Pro, возможны следующие причины:

- Perl-скрипт ora2pgpro выдаёт ошибку `ORA-XXX` — это означает, что указан некорректный DSN или данные для входа в систему. Проверьте сообщение об ошибке и указанную информацию и попробуйте снова.

- Perl-скрипт ничего не выдаёт, а выходной файл пустой — пользователь не имеет доступа к данным из базы. Попробуйте подключиться к Oracle с правами суперпользователя или используйте USER_GRANTS выше или директивы из следующего раздела, в частности SCHEMA.

LOGFILE

По умолчанию все сообщения отправляются на стандартный вывод. Если указать в этой директиве путь к файлу, все строки вывода будут добавлены в этот файл.

Экспортируемая схема Oracle

Можно ограничить экспорт БД Oracle отдельной схемой или пространством имён вручную. Если для подключения указан пользователь с ограниченными правами, только соответствующие объекты будут выгружены.

SCHEMA

Директива используется для указания имени схемы, используемой во время экспорта. Например, с указанием SCHEMA APPS будут выгружены объекты, связанные со схемой APPS.

Если включить EXPORT_SCHEMA и не задать имя схемы, ora2pgpro экспортирует все объекты из всех схем экземпляра Oracle, добавляя к именам объектов имя схемы в виде префикса.

EXPORT_SCHEMA

По умолчанию схема Oracle не экспортируется в БД Postgres Pro, а все объекты создаются в пространстве имён Postgres Pro по умолчанию. Если необходимо и экспортировать схему, и создать все объекты в этом пространстве имён, задайте для EXPORT_SCHEMA значение 1. При этом в пути к схеме (search_path) в начале SQL-файла для экспорта будет записана схема, указанная в директиве SCHEMA (по умолчанию pg_catalog). Если вы хотите изменить путь, используйте директиву PG_SCHEMA.

CREATE_SCHEMA

Директива включает/отключает команду CREATE SCHEMA в начале выходного файла. По умолчанию включена, используется с типом экспорта TABLE.

COMPILE_SCHEMA

По умолчанию ora2pgpro экспортирует только рабочий код PL/SQL. Можно перекомпилировать нерабочий код в Oracle, чтобы добиться корректности и в дальнейшем экспортировать его.

Чтобы схема компилировалась в Oracle перед экспортом кода, включите эту директиву. В этом случае, если

При этом в Oracle отправится запрос на перекомпиляцию кода PL/SQL, который мог стать нерабочим, например, после экспорта/импорта. В результате для функций, процедур, пакетов и пользовательских типов проставляется статус VALID или INVALID. Кроме того, это распространяется на отключённые триггеры.

EXPORT_INVALID

Если предыдущей директивы недостаточно для проверки корректности кода PL/SQL, включите эту директиву, чтобы разрешить экспорт всего кода PL/SQL, даже если он помечен как нерабочий. В результате для функций, процедур, пакетов и пользовательских типов проставляется статус VALID или INVALID.

PG_SCHEMA

Позволяет указать / принудительно использовать схему Postgres Pro. По умолчанию, если для EXPORT_SCHEMA задано значение 1, для параметра search_path в Postgres Pro будет задано имя схемы, указанной в SCHEMA.

Значением может быть список имён схем, разделённых запятыми, но не с типом экспорта TABLE, потому что в этом случае создаётся оператор CREATE SCHEMA, не поддерживающий несколько имён схем. Например, если для PG_SCHEMA указать user_schema, public, путь поиска задаётся следующим образом:

```
SET search_path = user_schema, public;
```

В этом случае используется схема (здесь user_schema), отличная от схемы Oracle, которая указана в директиве SCHEMA.

Кроме того, для пользователя Postgres Pro, под именем которого вы подключаетесь к целевой базе данных, можно задать значение search_path по умолчанию следующим образом:

```
ALTER ROLE username SET search_path TO user_schema, public;
```

Тогда задавать PG_SCHEMA не понадобится.

SYSUSERS

Если не задать схему явным образом, ora2pgpro экспортирует все объекты, не принадлежащие системной схеме или роли:

```
SYSTEM,CTXSYS,DBSNMP,EXFSYS,LBACSYS,MDSYS,MGMT_VIEW,
OLAPSYS,ORDDATA,OWBSYS,ORDPLUGINS,ORDSYS,OUTLN,
SI_INFORMTN_SCHEMA,SYS,SYSMAN,WK_TEST,WKSYS,WKPROXY,
WMSYS,XDB,APEX_PUBLIC_USER,DIP,FLOWS_020100,FLOWS_030000,
FLOWS_040100,FLOWS_010600,FLOWS_FILES,MDDATA,ORACLE_OCM,
SPATIAL_CSW_ADMIN_USR,SPATIAL_WFS_ADMIN_USR,XS$NULL,PERFSTAT,
SQLTXPLAIN,DMSYS,TSMSYS,WKSYS,APEX_040000,APEX_040200,
DVSYS,OJVMSYS,GSMADMIN_INTERNAL,APPQOSSYS,DVSYS,DVF,
AUDSYS,APEX_030200,MGMT_VIEW,ODM,ODM_MTR,TRACESRV,MTMSYS,
OWBSYS_AUDIT,WEBSYS,WK_PROXY,OSE$HTTP$ADMIN,
AURORA$JIS$UTILITY$,AURORA$ORB$UNAUTHENTICATED,
DBMS_PRIVILEGE_CAPTURE,CSMIG,MGDSYS,SDE,DBSFWUSER
```

В зависимости от инсталляции Oracle, могут быть определены несколько системных ролей. Чтобы добавить этих пользователей в список исключений схемы, укажите их через запятую в директиве SYSUSERS. Например:

```
SYSUSERS        INTERNAL,SYSDBA,BI,HR,IX,OE,PM,SH
```

В результате пользователи INTERNAL и SYSDBA будут добавлены в список исключений.

FORCE_OWNER

По умолчанию владельцем объектов БД будет пользователь, используемый для подключения к БД Postgres Pro в psql. Если используется другой пользователь (например, postgres), можно указать ora2pgpro назначать владельцем объектов пользователя, используемого в БД Oracle, задав для этой директивы значение 1. Если в значении указать имя пользователя, он и будет назначен владельцем.

FORCE_SECURITY_INVOKER

Для экспорта функций ora2pgpro использует права из Oracle (чаще всего SECURITY DEFINER). Включите эту директиву, если необходимо переопределить права для всех функций на SECURITY INVOKER.

USE_TABLESPACE

Когда директива включена, ora2pgpro экспортирует все таблицы, ограничения и индексы, используя имя пространства имён, указанное в БД Oracle. Не работает с пространствами имён TEMP, USERS и SYSTEM.

WITH_OID

Когда директива включена, при создании таблиц или представлений как таблиц ora2pgpro добавляет указание WITH OIDS. По умолчанию отключена, как и в Postgres Pro.

LOOK_FORWARD_FUNCTION

Список схем, из которых извлекаются метаданные о функциях/процедурах, используемые в экспорте схем. Если при замене вызова функции с параметрами OUT эта функция объявлена в другом пакете, вызов не может быть перезаписан, поскольку ora2pgpro знает только про функции, объявленные в текущей схеме. Если задать в этой директиве список схем, разделённых запятыми, ora2pgpro будет выполнять поиск всех функций, процедур и объявлений пакетов по всем пакетам перед тем, как начать экспорт текущей схемы.

NO_FUNCTION_METADATA

Отключает поиск объявления функции для ora2pgpro. Обратите внимание, что при этом ora2pgpro не будет перезаписывать вызов функции. Включайте эту директиву, только если поиск функции нарушает экспорт.

Тип экспорта

Директива TYPE определяет выполняемые во время экспорта действия, другие директивы позволяют только уточнить детали экспорта.

TYPE

Для директивы TYPE можно задать следующие значения (по умолчанию TABLE):

- TABLE: извлекать все таблицы с индексами, первичными ключами, ограничениями уникальности, внешними ключами и ограничениями-проверками.
- VIEW: извлекать только представления.
- GRANT: извлекать роли, преобразуемые в группы Postgres Pro, пользователей и права для все объектов.
- SEQUENCE: извлекать все последовательности и их текущие значения.
- TABLESPACE: извлекать пространства для таблиц и индексов.
- TRIGGER: извлекать событийные триггеры.
- FUNCTION: извлекать функции.
- PROCEDURE: извлекать процедуры.
- PACKAGE: извлекать пакеты и тела пакетов.
- INSERT: извлекать данные через оператор INSERT.
- COPY: извлекать данные через оператор COPY.
- PARTITION: извлекать секции по диапазонам и спискам с подсекциями.
- TYPE: извлекать пользовательские типы Oracle.
- FDW: извлекать таблицы Oracle как сторонние таблицы для oracle_fdw.
- MVIEW: эскпортировать материализованные представления.
- QUERY: автоматически преобразовывать SQL-запросы Oracle.
- DBLINK: создать сервер обёртки сторонних данных Oracle, чтобы использовать его как dblink.
- SYNONYM: экспортировать синонимы Oracle как представления на базе других объектов схемы.
- DIRECTORY: экспортировать каталоги Oracle как объекты расширения external_file.
- LOAD: отправлять набор запросов с использованием нескольких соединений с Postgres Pro.
- TEST: выполнить сравнение БД Oracle и Postgres Pro.
- TEST_COUNT: выполнить сравнение количества строк в таблицах Oracle и Postgres Pro.
- TEST_VIEW: подсчитать число строк, возвращаемых представлениями на обеих сторонах.
- TEST_DATA: выполнить проверку корректности строк на обеих сторонах.
- SHOW_VERSION: показать версию Oracle.
- SHOW_SCHEMA: вывести список доступных схем.
- SHOW_TABLE: вывести список доступных таблиц.
- SHOW_COLUMN: вывести список доступных столбцов таблиц и тип, используемый ora2pgpro при преобразовании из Oracle в Postgres Pro. Кроме того, выводится предупреждение, если в именах объектов Oracle есть зарезервированные слова Postgres Pro.
- SHOW_REPORT: вывести подробный отчёт о содержимом БД Oracle для оценки размера объектов и стоимости миграции.

Только один тип экспорта можно выполнить за один раз, поэтому директива TYPE должна быть уникальной. Если указать директиву несколько раз, будет использоваться последнее обнаруженное значение.

Результаты экспорта некоторых типов невозможно или не следует загружать в БД Postgres Pro напрямую — требуются дополнительные действия вручную. Это относится к типам экспорта GRANT, TABLESPACE, TRIGGER, FUNCTION, PROCEDURE, TYPE, QUERY и PACKAGE, особенно если есть код PL/SQL или SQL, специфичный для Oracle.

Для типа экспорта TABLESPACE в системе должен существовать путь к файлу, а для SYNONYM — владельцы объектов и схемы должны соответствовать новому проекту БД Postgres Pro.

Обратите внимание, что можно запустить серию операций экспорта, указав в директиве TYPE список типов, разделённых запятыми, но типы COPY и INSERT с другими типами указывать нельзя.

ora2pgpro преобразовывает секции Oracle, используя наследование, триггеры и функции. За дополнительной информацией обратитесь к разделу Секционирование таблиц.

Директива TYPE позволяет экспортировать пользовательские типы Oracle. Если не использовать параметр --plsql в командной строке, пользовательский тип Oracle выгружается как есть, в противном случае ora2pgpro попробует преобразовать его с использованием синтаксиса Postgres Pro.

Пример вывода с использованием SHOW_COLUMN:

```
[2] TABLE CURRENT_SCHEMA (1 rows) (Warning: 'CURRENT_SCHEMA' is a
 reserved word in PostgreSQL)
  CONSTRAINT : NUMBER(22) => bigint (Warning: 'CONSTRAINT' is a
 reserved word in PostgreSQL)
  FREEZE : VARCHAR2(25) => varchar(25) (Warning: 'FREEZE' is a
 reserved word in PostgreSQL)
 ...
 [6] TABLE LOCATIONS (23 rows)
  LOCATION_ID : NUMBER(4) => smallint
  STREET_ADDRESS : VARCHAR2(40) => varchar(40)
  POSTAL_CODE : VARCHAR2(12) => varchar(12)
  CITY : VARCHAR2(30) => varchar(30)
  STATE_PROVINCE : VARCHAR2(25) => varchar(25)
  COUNTRY_ID : CHAR(2) => char(2)
```

Нижеописанные ключевые слова только показывают запрошенную информацию и прерывают выполнение. Это позволяет узнать, с чем предстоит работать.

С типом SHOW_COLUMN в командной строке разрешается передавать параметры ora2pgpro --allow relname или -a relname, чтобы ограничить выводимую информацию одной таблицей.

С типом SHOW_ENCODING отображаются значения NLS_LANG и CLIENT_ENCODING, которые будет использовать ora2pgpro, и фактическая кодировка БД Oracle с соответствующей кодировкой клиента, которая будет использоваться в Postgres Pro.

ora2pgpro может экспортировать определение таблицы Oracle для использования через обёртку сторонних данных oracle_fdw. Если задать тип FDW, таблицы Oracle экспортируются следующим образом:

```
CREATE FOREIGN TABLE oratab (
  id        integer           NOT NULL,
  text      character varying(30),
  floating  double precision  NOT NULL
 ) SERVER oradb OPTIONS (table 'ORATAB');
```

Теперь таблицу можно использовать как обычную таблицу Postgres Pro.

Также можно вывести более подробный отчёт с информацией о стоимости миграции, подробнее см. «Оценка стоимости миграции».

ESTIMATE_COST

Активирует оценку стоимости миграции. Используется только с типами экспорта SHOW_REPORT, FUNCTION, PROCEDURE, PACKAGE и QUERY. По умолчанию отключена. Эта же функциональность включается с помощью параметра --estimate_cost командной строки. Обратите внимание, что включение этой директивы активирует PLSQL_PGSQL.

COST_UNIT_VALUE

Задаёт значение в минутах для блока оценки стоимости миграции. По умолчанию 5 минут на блок. Чтобы поменять значение в командной строке, используйте --cost_unit_value.

DUMP_AS_HTML

Если включить эту директиву, ora2pgpro выведет отчёт о миграции в формате HTML. По умолчанию при использовании SHOW_REPORT отчёт выводится в формате простого текста.

HUMAN_DAYS_LIMIT

Используйте эту директиву, чтобы переопределить предел человеко-дней, по достижении которого уровень оценки миграции меняется с B на C. По умолчанию 10 человеко-дней.

JOBS

Эта директива включает поддержку многопоточности для типов экспорта COPY, FUNCTION и PROCEDURE и задаёт число используемых процессов. По умолчанию многопоточность отключена.

Эта директива используется, чтобы указать число ядер для параллельного импорта данных в Postgres Pro. С типом экспорта FUNCTION или PROCEDURE каждая функция будет переведена в PL/pgSQL в новом отдельном процессе, что может дать значительное ускорение при большом количестве функций.

Параллельная обработка ограничивается только числом ядер и производительностью ввода-вывода Postgres Pro.

Не работает с ОС Windows, директива отключена.

ORACLE_COPIES

Эта директива включает поддержку многопоточности для извлечения данных из Oracle. Значением является число процессов для параллельного выполнения запроса. По умолчанию параллельность запросов отключена.

Параллельность построена на разделение запроса по числу ядер, переданному в качестве значения ORACLE_COPIES:

```
SELECT * FROM MYTABLE WHERE ABS(MOD(COLUMN, ORACLE_COPIES)) =
 CUR_PROC
```

Здесь COLUMN — это технический ключ, такой как первичный ключ или уникальный ключ, на основе которого осуществляется разделение и используется текущее ядро (CUR_PROC). В директиве DEFINED_PK можно задать имя используемого столбца.

Не работает с ОС Windows, директива отключена.

DEFINED_PK

Эта директива используется, чтобы определить технический ключ, используемый для разделения запроса по числу ядер, которое задано в переменной ORACLE_COPIES. Например:

```
DEFINED_PK      EMPLOYEES:employee_id
```

Предположим, что в -J или ORACLE_COPIES задано значение 8, используется параллельный запрос:

```
SELECT * FROM EMPLOYEES WHERE ABS(MOD(employee_id, 8)) = N
```

Здесь N — это текущий дочерний процесс (начиная с 0).

PARALLEL_TABLES

Эта директива используется, чтобы задать число таблиц, обрабатываемых параллельно для извлечения данных. Пределом является число ядер на машине. ora2pgpro устанавливает по одному подключению к БД для каждого параллельного процесса. Если значение этой директивы больше 1, ORACLE_COPIES отключается, но JOBS используется, так что фактическое число процессов = PARALLEL_TABLES * JOBS.

Обратите внимание, что если значение этой директивы больше 1, при экспорте в файлы автоматически активируется FILE_PER_TABLE, чтобы экспортировать таблицы и представления в разные файлы.

Используйте PARALLEL_TABLES, чтобы включить параллельность для действий COPY, INSERT и TEST_DATA. Также полезно для TEST, TEST_COUNT и SHOW_TABLE, если --count_rows используется для подсчёта действительного количества строк.

DEFAULT_PARALLELISM_DEGREE

Если задать для этой директивы значение больше 1, ora2pgpro будет принудительно использовать указание /*+ PARALLEL(tbname, degree) */ в каждом запросе экспорта данных из Oracle. Значение 0 или 1 отключает использование указаний для параллельности. По умолчанию отключена.

FDW_SERVER

Эта директива используется, чтобы указывать имя сервера сторонних данных для команды CREATE SERVER name FOREIGN DATA WRAPPER oracle_fdw. Затем это имя используется в командах CREATE FOREIGN TABLE и в импорте данных с помощью oracle_fdw. По умолчанию сторонний сервер не определён. Директива используется с типами экспорта FDW, COPY и INSERT. Для типа экспорта FDW значение по умолчанию — orcl.

FDW_IMPORT_SCHEMA

Схема, в которой будут созданы сторонние таблицы для миграции данных. Если для миграции данных используются несколько экземпляров ora2pgpro с использованием обёртки сторонних данных, может потребоваться изменять имя схемы для отдельных экземпляров. По умолчанию ora2pg_fdw_import.

DROP_FOREIGN_SCHEMA

По умолчанию перед каждым новым импортом ora2pgpro удаляет временную схему ora2pg_fdw_import, используемую для импорта сторонней схемы Oracle. Отключите директиву, если необходимо сохранить существующую схему из-за внесённых изменений или использования стороннего сервера.

EXTERNAL_TO_FDW

Эта директива (по умолчанию включена) позволяет экспортировать сторонние таблицы Oracle как сторонние таблицы file_fdw. Чтобы не экспортировать такие таблицы, задайте значение 0.

INTERNAL_DATE_MAX

Внутренние отметки времени, получаемые из пользовательских типов, извлекаются в следующем формате: 01-JAN-77 12.00.00.000000 AM. Поскольку точное столетие неизвестно, по умолчанию каждый год меньше 49 относится к 2000, а остальные — к 1900. Эта директива используется, чтобы изменить значение по умолчанию 49. Директива имеет смысл, только если есть пользовательский тип со столбцом timestamp.

AUDIT_USER

Задаёт список разделённых запятыми имён пользователей для фильтрации запросов к таблице DBA_AUDIT_TRAIL. По умолчанию эта таблица не сканируется, и поиск запросов не выполняется. Директива используется только с типами экспорта SHOW_REPORT и QUERY, когда не указан файл с запросами. Обратите внимание, что перед выводом выполняется нормализация текста запроса, в отличие от случаев, когда передаётся входной файл в параметре -i.

FUNCTION_CHECK

Отключите эту директиву, чтобы отключить check_function_bodies.

```
SET check_function_bodies = false;
```

При этом отключается проверка корректности строки с телом функции, передаваемой команде CREATE FUNCTION. По умолчанию используется параметр из postgresql.conf, включающий эту функциональность.

ENABLE_BLOB_EXPORT

Экспорт BLOB занимает время, поэтому иногда может потребоваться экспортировать все данные, кроме BLOB. В этом случае отключите эту директиву, и столбцы типа BLOB будут исключены из экспорта. Обратите внима-

ние, что у целевого столбца типа bytea не должно быть ограничения NOT NULL.

DATA_EXPORT_ORDER

По умолчанию экспорт данных выполняется в порядке имён таблиц. Если в базе есть огромные таблицы с именами в конце алфавита, при использовании многопоточности может быть полезно настроить сортировку по размеру, чтобы множество небольших таблиц были обработаны до того, как завершится обработка самых больших таблиц. В этом случае задайте для этой директивы значение size. Возможные значения: name и size. Обратите внимание, что с типами экспорта SHOW_TABLE и SHOW_COLUMN тоже можно использовать сортировку, не только с COPY или INSERT.

Ограничение экспорта

Иногда может потребоваться экспортировать только часть БД Oracle. Нижеописанные директивы позволяют ограничить экспорт отдельными объектами БД.

ALLOW

Эта директива позволяет задать список объектов, которые необходимо экспортировать, исключив все остальные объекты. Значением является список имён объектов, разделённых запятыми. В список можно добавлять регулярные выражения. Например:

```
ALLOW           EMPLOYEES SALE_.* COUNTRIES .*_GEOM_SEQ
```

С таким значением будут экспортированы объекты с именами EMPLOYEES, COUNTRIES, а также с именами, начинающимися на SALE_, и именами, заканчивающимися на _GEOM_SEQ. Типы объектов зависят от типа экспорта. Обратите внимание, что регулярные выражения не сработают в БД 8i, вместо них надо использовать местозаполнители %, чтобы утилита ora2pgpro использовала оператор LIKE.

Так задаются глобальные фильтры для текущего типа экспорта. Можно также задавать расширенные фильтры, применяемые к определённым объектам или для соответствующего типа экспорта. Например:

```
ora2pgpro -p -c ora2pgpro.conf -t TRIGGER -a 'TABLE[employees]'
```

При этом будут экспортироваться только триггеры для таблицы employees. Если нужны все триггеры, кроме некоторых INSTEAD OF:

```
ora2pgpro -c ora2pgpro.conf -t TRIGGER -e 'VIEW[trg_view_.*]'
```

Или в более сложном виде:

```
ora2pgpro -p -c ora2pgpro.conf -t TABLE -a 'TABLE[EMPLOYEES]' \
        -e 'INDEX[emp_.*];CKEY[emp_salary_min]'
```

Эта команда экспортирует определение таблицы employees, но без индексов, начинающихся с emp_, и ограничения CHECK с именем emp_salary_min.

При эскпорте секций можно исключить секционированные таблицы с помощью следующей команды:

```
ora2pgpro -p -c ora2pgpro.conf -t PARTITION -e
 'PARTITION[PART_199.* PART_198.*]'
```

При этом из экспорта будут исключены секции для лет с 1980 по 1999, но не основная таблица. Кроме того, будут исключены соответствующие триггеры.

С типом экспорта `GRANT` можно использовать такой расширенный формат, чтобы исключить пользователей из экспорта или ограничить отдельными пользователями:

```
ora2pgpro -p -c ora2pgpro.conf -t GRANT -a 'USER1 USER2'
or
ora2pgpro -p -c ora2pgpro.conf -t GRANT -a 'GRANT[USER1 USER2]'
```

Последняя команда ограничит экспорт пользователями `USER1` и `USER2`. Если же нет необходимости экспортировать права этих пользователей на определённые функции, можно использовать, например:

```
ora2pgpro -p -c ora2pgpro.conf -t GRANT -a 'USER1 USER2' -e
 'FUNCTION[adm_.*];PROCEDURE[adm_.*]'
```

Oracle не допускает использование выражений с предварительным просмотром, поэтому может потребоваться исключить из экспорта некоторые объекты, соответствующие заданному в `ALLOW` регулярному выражению. Например, если необходимо экспортировать все таблицы, начинающиеся на `E`, но исключить начинающиеся на `EXP`, отфильтровать таблицы одним выражением невозможно. Поэтому можно начать регулярное выражением с символа `!`, чтобы исключить объекты, соответствующие следующему регулярному выражению. Предыдущий пример можно записать следующим образом:

```
ALLOW   E.* !EXP.*
```

В выражении поиска объекта это будет выглядеть следующим образом:

```
REGEXP_LIKE(..., '^E.*$') AND NOT REGEXP_LIKE(..., '^EXP.*$')
```

`EXCLUDE`

Эта директива противоположна предыдущей — она позволяет задавать список разделённых пробелами или запятыми имён объектов, которые исключаются из экспорта. В списке можно указать регулярное выражение. Например:

```
EXCLUDE         EMPLOYEES TMP_.* COUNTRIES
```

При этом из экспорта будут исключены объекты с именами `EMPLOYEES`, `COUNTRIES`и все таблицы, начинающиеся на `tmp_`.

Например, с помощью этой директивы можно исключить ненужные функции:

```
EXCLUDE         write_to_.* send_mail_.*
```

В вышеприведённом примере из экспорта исключаются все функции, процедуры или функции из пакетов с именами, начинающимися как задано ре-

гулярным выражением. Обратите внимание, что регулярные выражения не сработают в БД 8i, вместо них надо использовать местозаполнители `%`, чтобы утилита ora2pgpro использовала оператор `NOT LIKE`. Расширенный синтаксис описан выше (см. директиву `ALLOW`).

`NO_EXCLUDED_TABLE`

По умолчанию ora2pgpro исключает из экспорта «мусорные» таблицы Oracle, которые никогда не должны включаться. При этом создаётся множество выражений `REGEXP_LIKE`, что замедляет анализ таблиц при экспорте. Чтобы отключить это поведение, включите данную директиву, но ненужные таблицы придётся впоследствии исключать или удалять вручную. Регулярные выражения для исключения таблиц задаются в массиве `@EXCLUDED_TABLES` в `lib/Ora2Pgpro.pm`. Обратите внимание, что это поведение не зависит от директивы `EXCLUDE`.

`VIEW_AS_TABLE`

Задаёт представления для экспорта как таблицы. По умолчанию никакие. В значении указывается регулярное выражение или список имён представлений, разделённых пробелами или запятыми. Если имя объекта — это представление и задан тип экспорта `TABLE`, представление экспортируется как оператор `CREATE TABLE`. С типом экспорта `COPY` или `INSERT` соответствующие данные экспортируются. За подробным описанием обратитесь к разделу Экспорт представлений как таблиц Postgres Pro.

`MVIEW_AS_TABLE`

Задаёт материализованные представления для экспорта как таблицы. По умолчанию никакие. В значении указывается регулярное выражение или список имён материализованных представлений, разделённых пробелами или запятыми. Если имя объекта — это материализованное представление и задан тип экспорта `TABLE`, представление экспортируется как оператор `CREATE TABLE`. С типом экспорта `COPY` или `INSERT` соответствующие данные экспортируются.

`NO_VIEW_ORDERING`

По умолчанию ora2pgpro пытается упорядочить представления во избежание ошибок со вложенными представлениями во время импорта. Если представлений очень много, это может занять много времени, поэтому можно отключить упорядочивание, задав эту директиву.

`GRANT_OBJECT`

Можно задать список разделённых запятыми объектов, права для которых экспортируются. По умолчанию права экспортируются для всех объектов. Возможные значения: `TABLE`, `VIEW`, `MATERIALIZED VIEW`, `SEQUENCE`, `PROCEDURE`, `FUNCTION`, `PACKAGE BODY`, `TYPE`, `SYNONYM`, `DIRECTORY`. Разрешается указывать только один тип объектов. Например, для экспорта прав только для таблиц задайте значение `TABLE`. Значение можно переопределить параметром `-g`.

`WHERE`

Эта директива позволяет задавать фильтр по предложению `WHERE` при выгрузке содержимого таблиц. Значение задаётся в формате

TABLE_NAME[WHERE_CLAUSE] или только предложение WHERE при наличии только одного такого предложения на каждую таблицу. Можно задавать оба варианта одновременно. Примеры:

```
# Глобальное предложение WHERE применяется ко всем таблицам в
 экспорте
WHERE  1=1

# Предложение WHERE только для таблицы TABLE_NAME
WHERE  TABLE_NAME[ID1='001']

# Два разных предложения для таблиц TABLE_NAME и OTHER_TABLE
# и общее предложение WHERE по DATE_CREATE для всех остальных
 таблиц
WHERE  TABLE_NAME[ID1='001' OR ID1='002] DATE_CREATE > '2001-01-01'
 OTHER_TABLE[NAME='test']
```

Если предложение WHERE указывается не в скобках с именем таблицы, оно применяется для всех экспортируемых таблиц, включая таблицы, указанные в предложении WHERE. Предложения WHERE полезны, если необходимо обратиться к архивным данным или, наоборот, недавно созданным.

Чтобы быстро протестировать импорт данных, удобно ограничить экспорт данных первой тысячей кортежей из каждой таблицы. Для Oracle укажите следующее предложение:

```
WHERE   ROWNUM < 1000
```

Также директиву можно использовать, чтобы ограничить экспорт несколькими таблицами.

Параметр -W или --where, передаваемый в командной строке, переопределит значение этой директивы глобально и для отдельных таблиц, если имена таблиц совпадают.

TOP_MAX

Эта директива используется, чтобы ограничить количество элементов, отображаемых в начале списков объектов с наибольшими характеристиками, например в списке таблиц с наибольшим количеством и списке самых больших таблиц по мегабайтам. По умолчанию 10 элементов.

LOG_ON_ERROR

Включите эту директиву, если хотите продолжать импорт данных в случае ошибки. Когда ora2pgpro получает ошибку в операторах COPY или INSERT в Postgres Pro, этот оператор записывается в файл журнала TABLENAME_error.log в выходном каталоге, и работа продолжается. Можно попробовать корректировать оператор и вручную перезагрузить файл журнала ошибок. По умолчанию импорт прерывается при ошибке.

REPLACE_QUERY

Иногда может потребоваться извлечь данные из таблицы Oracle, но для этого нужен отдельный запрос — не простой запрос SELECT * FROM table

от ora2pgpro, а более сложный. Эта директива позволяет переопределить запрос, используемый ora2pgpro для извлечения данных. Формат запроса: `TABLENAME[SQL_QUERY]`. Если таблиц много, с заменой запроса ora2pgpro можно указать несколько строк `REPLACE_QUERY`.

```
REPLACE_QUERY   EMPLOYEES[SELECT e.id,e.fisrtname,lastname FROM
 EMPLOYEES e JOIN EMP_UPDT u ON (e.id=u.id AND u.cdate>'2014-08-01
 00:00:00')]
```

Управление экспортом полнотекстового поиска

Для управления экспортом индексов для полнотекстового поиска в ora2pgpro предусмотрено несколько директив. По умолчанию индексы `CONTEXT` экспортируются в Postgres Pro как индексы `FTS`, а индексы `CTXCAT` экспортируются с использованием расширения pg_trgm.

`CONTEXT_AS_TRGM`

Принудительно переводит полнотекстовые индексы Oracle в индексы Postgres Pro с использованием расширения pg_trgm. По умолчанию индексы `CONTEXT` переводятся в индексы `FTS`, а индексы `CTXCAT` — с использованием расширения pg_trgm. Эта директива позволяет использовать pg_trgm, чего достаточно в большинстве случаев. Перед импортом объектов необходимо создать расширение pg_trgm в целевой базе данных.

`FTS_INDEX_ONLY`

По умолчанию ora2pgpro создаёт индекс на основе функции, чтобы перенести полнотекстовые индексы Oracle.

```
CREATE INDEX ON t_document
  USING gin(to_tsvector('pg_catalog.french', title));
```

Необходимо переписать предложение `CONTAIN()` с использованием `to_tsvector()`, например:

```
SELECT id,title FROM t_document
  WHERE to_tsvector(title) @@ to_tsquery('search_word');
```

Отключите эту директиву, чтобы утилита ora2pgpro создавала дополнительный столбец типа tsvector с выделенными триггерами для индексов `FTS`. В этом случае ora2pgpro добавляет столбец следующим образом:

```
ALTER TABLE t_document ADD COLUMN tsv_title tsvector;
```

Затем столбец обновляется для вычисления векторов `FTS`, если данные были загружены до команды `UPDATE t_document SET tsv_title = to_tsvector('pg_catalog.french', coalesce(title,''));`. Чтобы автоматически обновлять столбец при изменении заголовка, ora2pgpro добавляет следующий триггер:

```
CREATE FUNCTION tsv_t_document_title() RETURNS trigger AS $$
BEGIN
       IF TG_OP = 'INSERT' OR new.title != old.title THEN
              new.tsv_title :=
```

```
                to_tsvector('pg_catalog.french',
 coalesce(new.title,''));
       END IF;
       return new;
END
$$ LANGUAGE plpgsql;
CREATE TRIGGER trig_tsv_t_document_title BEFORE INSERT OR UPDATE
 ON t_document
 FOR EACH ROW EXECUTE PROCEDURE tsv_t_document_title();
```

Если полнотекстовый индекс Oracle создан по нескольким столбцам, ora2pgpro будет использовать функцию `setweight()`, чтобы задать вес в порядке объявления столбцов.

`FTS_CONFIG`

Используйте эту директиву, чтобы принудительно задать конфигурацию текстового поиска. Если директива не задана, ora2pgpro автоматически определяет стеммер, используемый в Oracle для каждого индекса, и `pg_catalog.english`, если информация не найдена.

`USE_UNACCENT`

Включите эту директиву, если текстовый поиск должен выполняться без учёта символов ударения или акцента. ora2pgpro создаст вспомогательную функцию на `unaccent()` и индексы pg_trgm, использующие эту функцию. Для индексов `FTS` ora2pgpro переопределит конфигурацию текстового поиска, например:

```
CREATE TEXT SEARCH CONFIGURATION fr (COPY = french);
ALTER TEXT SEARCH CONFIGURATION fr
        ALTER MAPPING FOR hword, hword_part, word WITH unaccent,
 french_stem;
```

Затем задайте для директивы `FTS_CONFIG` в `ora2pgpro.conf` значение `fr` вместо `pg_catalog.english`.

Если директива включена, ora2pgpro создаёт функцию-обёртку:

```
CREATE OR REPLACE FUNCTION unaccent_immutable(text)
RETURNS text AS
$$
    SELECT public.unaccent('public.unaccent', $1);
$$ LANGUAGE sql IMMUTABLE
   COST 1;
```

Индексы экспортируются так:

```
CREATE INDEX t_document_title_unaccent_trgm_idx ON t_document
    USING gin (unaccent_immutable(title) gin_trgm_ops);
```

Чтобы использовать индекс на основе функции, в запросах необходимо использовать эту же функцию, например:

```
SELECT * FROM t_document
        WHERE unaccent_immutable(title) LIKE '%donnees%';
```

USE_LOWER_UNACCENT

То же, что и выше, но вызывается функция `lower()` в функции `unaccent_immutable()`:

```
CREATE OR REPLACE FUNCTION unaccent_immutable(text)
RETURNS text AS
$$
    SELECT lower(public.unaccent('public.unaccent', $1));
$$ LANGUAGE sql IMMUTABLE;
```

Изменение структуры объектов

Гибкость ora2pgpro позволяет перенести БД Oracle в БД Postgres Pro с другой структурой или схемой. Нижеописанные директивы позволяют настроить необходимое для этого сопоставление.

REORDERING_COLUMNS

Включите эту директиву, чтобы переупорядочить столбцы и уменьшить занимаемое место на диске. В результате на страницу поместится больше строк, что играет важную роль для увеличения скорости. По умолчанию отключена (`disabled`), то есть используется такой же порядок определения таблиц, как в Oracle, — для большинства случаев этого достаточно. Директива используется только с типом экспорта `TABLE`.

MODIFY_STRUCT

Эта директива позволяет ограничить извлекаемые столбцы для указанной таблицы. В значении задаётся список имён таблиц, разделённых пробелами, с набором столбцов в скобках: `MODIFY_STRUCT NOM_TABLE(nomcol1,nomcol2,...) ...`. Например:

```
MODIFY_STRUCT   T_TEST1(id,dossier) T_TEST2(id,fichier)
```

В примере выше извлекаются столбцы `id` и `dossier` из таблицы `T_TEST1` и столбцы `id` и `fichier` из таблицы `T_TEST2`. Директива может используется только с типами экспорта `TABLE`, `COPY` или `INSERT`. С типом `TABLE` оператор `CREATE TABLE` создаёт столбцы по новому списку, и никакие индексы или внешние ключи, связанные с исключёнными столбцами, не экспортируются.

EXCLUDE_COLUMNS

Вместо того, чтобы переопределять структуру таблицы с помощью `MODIFY_STRUCT`, можно исключить некоторые столбцы из экспорта. В значении задаётся список имён таблиц, разделённых пробелами, с набором столбцов в скобках: `EXCLUDE_COLUMNS NOM_TABLE(nomcol1,nomcol2,...) ...`. Например:

```
EXCLUDE_COLUMNS T_TEST1(id,dossier) T_TEST2(id,fichier)
```

В примере выше из экспорта исключаются столбцы `id` и `dossier` таблицы `T_TEST1` и столбцы `id` и `fichier` таблицы `T_TEST2`. Директива может используется только с типами экспорта `TABLE`, `COPY` или `INSERT`. С типом `TABLE` оператор `CREATE TABLE` создаёт столбцы по новому списку, и никакие индексы

или внешние ключи, связанные с исключёнными столбцами, не экспортируются.

REPLACE_TABLES

Эта директива позволяет соотнести список имён таблиц Oracle с именами таблиц Postgres Pro. В значении задаётся список имён таблиц, разделённых пробелами:

```
REPLACE_TABLES  ORIG_TBNAME1:DEST_TBNAME1 ORIG_TBNAME2:DEST_TBNAME2
```

Таблицы Oracle ORIG_TBNAME1 и ORIG_TBNAME2 будут переименованы в DEST_TBNAME1 и DEST_TBNAME2 соответственно.

REPLACE_COLS

Как и имена таблиц, имена экспортируемых столбцов можно переопределить следующим образом: REPLACE_COLS ORIG_TBNAME(ORIG_COLNAME1:NEW_COLNAME1,ORIG_COLNAME2:NEW_COLNAME2). Например:

```
REPLACE_COLS    T_TEST(dico:dictionary,dossier:folder)
```

При этом столбцы Oracle dico и dossier из таблицы T_TEST будут переименованы в dictionary и folder.

REPLACE_AS_BOOLEAN

Если во время экспорта необходимо поменять тип столбцов Oracle на boolean в Postgres Pro, можно указать в этой директиве список таблиц и столбцов, разделённых пробелами:

```
REPLACE_AS_BOOLEAN     TB_NAME1:COL_NAME1 TB_NAME1:COL_NAME2
 TB_NAME2:COL_NAME2
```

Значения в столбцах типа boolean будут заменены на t и f, при этом используются значения по умолчанию и заданные в директиве BOOLEAN_VALUES.

Обратите внимание, что если имя таблицы и/или имя столбца было изменено директивой REPLACE_TABLES, необходимо использовать имя оригинальной таблицы и/или столбца.

```
REPLACE_COLS            TB_NAME1(OLD_COL_NAME1:NEW_COL_NAME1)
REPLACE_AS_BOOLEAN      TB_NAME1:OLD_COL_NAME1
```

Можно также задать тип и точность для преобразования всех полей этого типа в boolean автоматически, например:

```
REPLACE_AS_BOOLEAN      NUMBER:1 CHAR:1 TB_NAME1:COL_NAME1
 TB_NAME1:COL_NAME2
```

В примере выше все поля типа number(1) или char(1) заменяются на boolean во всех экспортируемых таблицах.

BOOLEAN_VALUES

Используйте эту директиву, чтобы дать дополнительное определение для значений типа boolean в полях Oracle. Необходимо задать список значений

вида TRUE:FALSE, разделённых пробелами. По умолчанию ora2pgpro распознает следующие типы значений:

```
BOOLEAN_VALUES          yes:no y:n 1:0 true:false enabled:disabled
```

Любые значения, заданные в этой директиве, добавляются в список значений по умолчанию.

REPLACE_ZERO_DATE

Если утилита ora2pgpro находит «нулевую» дату (0000-00-00 00:00:00), то заменяет её на NULL. Это может привести к проблемам, если в столбце есть ограничение NOT NULL. Если удалить ограничение невозможно, используйте эту директиву, чтобы указать произвольную дату для использования вместо NULL. Если фиктивные даты использовать неуместно, можно задать значение -INFINITY.

INDEXES_SUFFIX

Добавляет заданное значение к именам индексов в виде суффикса. Полезно, если имена индексов совпадают с именами таблиц. Например:

```
INDEXES_SUFFIX          _idx
```

Ко всем именам индексов в конце добавится _idx. Не самый распространённый случай, но может быть полезно.

INDEXES_RENAMING

Включите эту директиву, чтобы переименовать все индексы в таком формате: tablename_columns_names. Может быть очень полезно в базах данных, где имена индексов совпадают между собой или с именем таблицы, что не допускается в Postgres Pro. По умолчанию отключена.

USE_INDEX_OPCLASS

Классы операторов text_pattern_ops, varchar_pattern_ops и bpchar_pattern_ops поддерживают индексы B-деревья соответствующих типов. Отличие от классов операторов по умолчанию заключается в том, что выполняется строгое посимвольное сравнение, а не по правилам сортировки, определяемым локалью, благодаря чему эти классы операторов удобно использовать в запросах с выражениями поиска по шаблону (регулярные выражения LIKE или POSIX), когда локаль базы данных отличается от используемой по умолчанию C. Если задать значение 1, ora2pgpro будет экспортировать все индексы, построенные по столбцам типа varchar2() и char(), с использованием этих операторов. Если задать значение больше 1, будут изменяться только индексы по столбцам с ограничением по символам больше или равным указанному значению. Например, при значении 128 будут создаваться индексы по столбцам типа varchar2(N), где N >= 128.

RENAME_PARTITION

Включите эту директиву, если необходимо переименовывать секционированные таблица. По умолчанию отключена. При большом количестве секционированных таблиц может случиться, что у секций разных родительских таблиц совпадают имена, что недопустимо в Postgres Pro, где имена таблиц

должны быть уникальными. По умолчанию применяются следующие правила переименования:
- Секция: "tablename"_part"pos", где "pos" — это номер секции.
- Подсекция: "tablename"_part"pos"_subpart"pos".
- Секция/подсекция: "tablename"_part_default "tablename"_part"pos"_subpart_default.

DISABLE_PARTITION

Включите эту директиву, если сохранять секционирование как в Oracle не требуется и достаточно экспортировать все данные секционированных таблиц в одну таблицу в Postgres Pro. По умолчанию секционирование сохраняется — ora2pgpro экспортирует данные из каждой секции и импортирует их в соответствующую секционированную таблицу в Postgres Pro.

DISABLE_UNLOGGED

По умолчанию ora2pgpro экспортирует таблицы Oracle со свойством NOLOGGING как UNLOGGED. Можно полностью отключить эту функциональность, поскольку данные из таблиц UNLOGGED могут быть потеряны в случае сбоя Postgres Pro. Если задать для директивы значение 1, все таблицы будут экспортировать как таблицы обычного типа.

Экспорт из Oracle Spatial в PostGIS

ora2pgpro поддерживает полный экспорт пространственных объектов из БД Oracle. Для управления этим типом экспорта предусмотрены следующие директивы.

AUTODETECT_SPATIAL_TYPE

По умолчанию ora2pgpro анализирует индексы, чтобы определить тип пространственного ограничения и размерность, определённую в Oracle. Эти ограничения передаются при создании индекса, например, так:

```
CREATE INDEX ... INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS('sdo_indx_dims=2, layer_gtype=point');
```

Если параметры ограничений в Oracle не указаны, по умолчанию эти столбцы экспортируются с универсальным типом GEOMETRY, чтобы получать любой тип пространственных данных.

Директива AUTODETECT_SPATIAL_TYPE заставляет ora2pgpro автоматически определять фактический тип пространственных данных и размерность, используемые в столбце с пространственными данными, в противном случае используется тип геометрии без ограничений. Когда эта функциональность включена (по умолчанию), ora2pgpro сканирует выборку из 50000 строк, чтобы проанализировать используемый тип GTYPE. Размер выборки можно увеличить или уменьшить, указав в AUTODETECT_SPATIAL_TYPE необходимое количество строк для сканирования.

Например, если директива AUTODETECT_SPATIAL_TYPE отключена, столбец shape с типом Oracle SDO_GEOMETRY будет преобразован следующим образом:

```
shape geometry(GEOMETRY) or shape geometry(GEOMETRYZ, 4326)
```

Если директива включена и столбец содержит отдельный тип геометрии, использующий одно измерение с двумерными или трёхмерными многоугольниками:

```
shape geometry(POLYGON, 4326) or shape geometry(POLYGONZ, 4326)
```

CONVERT_SRID

Эта директива позволяет управлять автоматическим преобразованием SRID Oracle в значения, соответствующие стандарту EPSG. Когда директива включена, ora2pgpro использует функцию Oracle `sdo_cs.map_oracle_srid_to_epsg()` для преобразования всех SRID. По умолчанию включена.

Если Oracle возвращает NULL для `SDO_SRID`, NULL заменяется значением по умолчанию 8307, преобразованным в значение по EPSG: 4326 (см. `DEFAULT_SRID`).

Если задать значение больше 1, все SRID получат это значение, то есть `DEFAULT_SRID` не используется, если Oracle возвращает NULL.

Обратите внимание, что значение `EPSG` можно задать на стороне Oracle, если `sdo_cs.map_oracle_srid_to_epsg()` возращает NULL:

```
  system@db> UPDATE sdo_coord_ref_sys SET legacy_code=41014 WHERE
 srid = 27572;
```

DEFAULT_SRID

Используйте эту директиву, чтобы переопределить значение EPSG для SRID по умолчанию (4326). `CONVERT_SRID` переопределяет это значение, см. выше.

GEOMETRY_EXTRACT_TYPE

Эта директива может принимать следующие значения: `WKT` (по умолчанию), `WKB` или `INTERNAL`. Если задано значение `WKT`, ora2pgpro использует `SDO_UTIL.TO_WKTGEOMETRY()`, чтобы извлечь геометрические данные. Когда задано значение `WKB`, ora2pgpro выводит их в двоичном формате с использованием `SDO_UTIL.TO_WKBGEOMETRY()`. Если извлекать данные вышеописанными способами в виде вызовов на стороне Oracle, при большом количестве строк может произойти нехватка памяти из-за значительного времени выполнения. Кроме того, с `WKB` 3D геометрии и некоторые другие геометрии, такие как `CURVEPOLYGON`, не извлекаются. В таких случаях можно использовать тип извлечения `INTERNAL`. При этом для преобразования данных `SDO_GEOMETRY` в `WKT` используется библиотека Pure Perl, а само преобразование осуществляется на стороне ora2pgpro. Эта функциональность требует доработки, поэтому перед использованием необходимо проверять корректность экспортированных геометрий. Для пространственных объектов по умолчанию используется тип извлечения `INTERNAL`.

POSTGIS_SCHEMA

Используйте эту директиву, чтобы добавить определённую схему в путь поиска для функций PostGIS.

ST_SRID_FUNCTION

Задаёт функцию Oracle для извлечения SRID из метаданных ST_Geometry. По умолчанию: ST_SRID. Для ArcSDE, например, следует указать sde.st_srid.

ST_DIMENSION_FUNCTION

Задаёт функцию Oracle для извлечения размерности из метаданных ST_Geometry. По умолчанию: ST_DIMENSION. Для ArcSDE, например, следует указать sde.st_dimension.

ST_GEOMETRYTYPE_FUNCTION

Задаёт функцию Oracle для извлечения типа геометрии из столбца ST_Geometry. По умолчанию: ST_GEOMETRYTYPE. Для ArcSDE, например, следует указать sde.st_geometrytype.

ST_ASBINARY_FUNCTION

Задаёт функцию Oracle для преобразования значения ST_Geometry в формат WKB. По умолчанию: ST_ASBINARY. Для ArcSDE, например, следует указать sde.st_asbinary .

ST_ASTEXT_FUNCTION

Задаёт функцию Oracle для преобразования значения ST_Geometry в формат WKT. По умолчанию: ST_ASTEXT. Для ArcSDE, например, следует указать sde.st_astext .

Импорт в Postgres Pro

По умолчанию результаты преобразования в формат Postgres Pro выводятся в файл output.sql.

```
psql mydb < output.sql
```

Эта команда импортирует содержимое файла output.sql в БД Postgres Pro mydb.

DATA_LIMIT

При экспорте типа INSERT/COPY ora2pgpro ведёт обработку блоками по DATA_LIMIT кортежей для увеличения скорости. Перед записью на диск кортежи хранятся в памяти, так что если необходимо увеличить скорость, то при наличии достаточных системных ресурсов можно увеличить этот предел, например, до 100000 или 1000000. Значение 0 означает, что используется размер блока по умолчанию: 10000.

BLOB_LIMIT

Когда ora2pgpro обнаруживает таблицу с BLOB, значение DATA_LIMIT автоматически уменьшается путём деления на 10 до тех пор, пока оно не станет меньше 1000. Можно отдельно указать директиву BLOB_LIMIT для экспорта

BLOB, но обратите внимание, что экспорт BLOB задействует много ресурсов, поэтому большое значение может вызвать нехватку памяти.

OUTPUT

Эта директива позволяет изменить выходной файл для ora2pgpro. Значение по умолчанию — `output.sql`. Если задать имя файла с расширением `.gz` или `.bz2`, вывод будет сжат автоматически. При этом для файлов `.gz` требуется установленный модуль Perl `Compress::Zlib`, а для `.bz2` — утилита командной строки bzip2.

OUTPUT_DIR

В этой директиве можно задать каталог для записи файла. Указанный каталог должен существовать.

BZIP2

В этой директиве можно указать полный путь к утилите bzip2, если она не обнаружена в переменной окружения `PATH`.

FILE_PER_CONSTRAINT

Позволяет сохранять ограничения для объектов в отдельном файле во время экспорта схемы. Файл будет назван `CONSTRAINTS_OUTPUT`, где `OUTPUT` — это значение соответствующей директивы. Для сжатия можно использовать расширение `.gz` или `.bz2`. По умолчанию все данные сохраняются в файле, указанном в `OUTPUT`. Эта директива используется только с типом экспорта `TABLE`.

Ограничения можно быстро импортировать в Postgres Pro с типом экспорта `LOAD`, чтобы создавать их параллельно, используя несколько соединений (`-j` или `JOBS`).

FILE_PER_INDEX

Позволяет сохранять индексы в отдельном файле во время экспорта схемы. Файл будет назван `INDEXES_OUTPUT`, где `OUTPUT` — это значение соответствующей директивы. Для сжатия можно использовать расширение `.gz` или `.bz2`. По умолчанию все данные сохраняются в файле, указанном в `OUTPUT`. Эта директива используется только с типами экспорта `TABLE` и `TABLESPACE`. С типом экспорта `TABLESPACE` команды `ALTER INDEX ... TABLESPACE ...` записываются в отдельный файл `TBSP_INDEXES_OUTPUT`, который можно загрузить в конце миграции после создания индексов для их перемещения.

Индексы можно быстро импортировать в Postgres Pro с типом экспорта `LOAD`, чтобы создавать их параллельно, используя несколько соединений (`-j` или `JOBS`).

FILE_PER_FKEYS

Позволяет сохранять объявления внешних ключей в отдельном файле во время экспорта схемы. По умолчанию внешние ключи экспортируются в основной выходной файл или `CONSTRAINT_output.sql`. Если директива задана, внешние ключи экспортируются в файл `FKEYS_output.sql`.

FILE_PER_TABLE

Позволяет сохранять результаты экспорта в отдельных файлах для каждой таблицы или представления. Файлы будут названы tablename_OUTPUT, где OUTPUT — это значение соответствующей директивы. Для сжатия можно использовать расширение .gz или .bz2 в директиве OUTPUT. С заданным значением 0 (по умолчанию) все данные сохраняются в одном файле, значение 1 включает эту функциональность. Используется только с типами экспорта INSERT или COPY.

FILE_PER_FUNCTION

Позволяет сохранять функции, процедуры и триггеры в отдельных файлах для каждого объекта. Файлы будут названы objectname_OUTPUT, где OUTPUT — это значение соответствующей директивы. Для сжатия можно использовать расширение .gz или .bz2 в директиве OUTPUT. С заданным значением 0 (по умолчанию) все данные сохраняются в одном файле, значение 1 включает эту функциональность. Используется только с соответствующим типом экспорта, а для пакетов предусмотрено другое поведение.

Если директива включена с типом экспорта PACKAGE, ora2pgpro создаёт отдельный каталог для каждого пакета с именем, называя его по имени пакета в нижнем регистре, и создаёт отдельные файлы для каждой функции/процедуры в этом каталоге. Если директива отключена, создаётся отдельный файл для каждого пакета с именем packagename_OUTPUT, где OUTPUT — это значение соответствующей директивы.

TRUNCATE_TABLE

Если задано значение 1, перед загрузкой данных добавляется команда TRUNCATE TABLE. Используется только с типами экспорта INSERT или COPY.

Когда директива включена, команда добавляется, только если нет глобального предложения DELETE или для текущей таблицы (см. ниже).

DELETE

Включает поддержку фильтрации по предложению DELETE FROM ... WHERE перед импортом данных с удалением строк вместо опустошения таблиц. Значение задаётся в таком формате: TABLE_NAME[DELETE_WHERE_CLAUSE], или если есть только одно предложение WHERE для всех таблиц, задаётся одно предложение DELETE в качестве значения. Можно задавать оба варианта одновременно. Примеры:

```
DELETE  1=1    # Применяется ко всем таблицам и удаляет все кортежи
DELETE  TABLE_TEST[ID1='001']   # Применяется только к таблице
 TABLE_TEST
DELETE  TABLE_TEST[ID1='001' OR ID1='002] DATE_CREATE >
 '2001-01-01' TABLE_INFO[NAME='test']
```

Последний вариант применяет два разных предложения DELETE ... WHERE к таблицам TABLE_TEST и TABLE_INFO и общее предложение DELETE по DATE_CREATE ко всем остальным таблицам. Если директива TRUNCATE_TABLE включена, она применяется ко всем таблицам, к которым не применяется DELETE. Такие предложения DELETE могут быть полезны при обычных изменениях.

`STOP_ON_ERROR`

Задайте для директивы значение 0, чтобы исключить вызов `\set ON_ERROR_STOP ON` из всех SQL-скриптов, создаваемых ora2pgpro. По умолчанию эта команда всегда присутствует, чтобы скрипт немедленно завершал работу при ошибках.

`COPY_FREEZE`

Включите эту директиву, чтобы использовать `COPY FREEZE` вместо обычно `COPY` для экспорта данных с уже замороженными строками. Это позволяет увеличить производительность при начальном добавлении данных. Строки будут замораживаться, только если загружаемая таблица была создана или опустошена в текущей подтранзакции. Работает только для экспорта в файл, когда не заданы ни параметр `-J`, ни директива `ORACLE_COPIES` или они равны 1. Может использоваться для импорта в Postgres Pro напрямую при тех же условиях, но `-j` и `JOBS` должны быть не заданы или равны 1.

`CREATE_OR_REPLACE`

По умолчанию ora2pgpro использует `CREATE OR REPLACE` в функциях и представлениях. Если нет необходимости переопределять существующие функции или представления, отключите эту директиву, и команды не будут содержать `OR REPLACE`.

`DROP_IF_EXISTS`

Чтобы добавлять команду `DROP` *`ОБЪЕКТ`* `IF EXISTS` перед созданием объекта, включите эту директиву. Может быть полезно в итерационной работе. По умолчанию директива отключена.

`EXPORT_GTT`

Postgres Pro не поддерживает глобальные временные таблицы, но для эмуляции этой функциональности можно использовать расширение pgtt. Включите эту директиву, чтобы экспортировать глобальные временные таблицы.

`NO_HEADER`

Если включить эту директиву, ora2pgpro не будет добавлять заголовок в выходные файлы, будет записываться только преобразованный код.

`PSQL_RELATIVE_PATH`

По умолчанию ora2pgpro использует команду psql `\i`, чтобы выполнять создаваемые SQL-файлы. Если задать эту директиву, будет использоваться команда `\ir`, чтобы интерпретировать имена файлов относительно каталога, в котором расположен скрипт. За подробной информацией обратитесь к справке по psql.

`DATA_VALIDATION_ROWS`

Число строк, которые необходимо получить с обеих сторон для проверки корректности данных. По умолчанию сравниваются первые 10000 строк. При значении 0 сравниваются все строки.

DATA_VALIDATION_ORDERING

После изменения данных порядок строк на обеих сторонах отличается. Директива включает упорядочивание данных по первичному ключу или уникальному индексу, так что данные таблицы без этих объектов сравнить невозможно. Если проверка корректности выполняется сразу после миграции, все таблицы могут быть проверены без упорядочивания.

DATA_VALIDATION_ERROR

Останавливает проверку данных таблицы после определённого количества несовпадающих строк. По умолчанию проверка останавливается после 10 строк с ошибками несовпадения.

TRANSFORM_VALUE

Используйте эту директиву, чтобы указать тип трансформации, применяемый к столбцам при экспорте данных, в виде списка значений, разделённых точкой с запятой:

```
TABLE[COLUMN_NAME, код в целевом списке SELECT]
```

Например, чтобы заменить строку «Oracle» на «PostgreSQL» в столбце типа varchar2, используйте следующее значение:

```
TRANSFORM_VALUE
 ERROR_LOG_SAMPLE[DBMS_TYPE:regexp_replace("DBMS_TYPE",'Oracle','PostgreSQL')]
```

Чтобы заменить все значения типа char(0) в строке пробельными символами:

```
TRANSFORM_VALUE   CLOB_TABLE[CHARDATA:translate("CHARDATA", chr(0),
 ' ')]
```

Выражение будет применяться в операторе SQL, используемом для извлечения данных из исходной базы данных.

Если для выгрузки данных в файл задан тип экспорта INSERT или COPY и включена директива FILE_PER_TABLE, будет выдано предупреждение, что ora2pgpro не будет экспортировать данные, если файл уже существует, чтобы не допустить повторную загрузку огромных таблиц. Чтобы принудительно загружать данные из таких таблиц, необходимо сначала удалить существующий выходной файл.

Если необходимо импортировать данные в БД Postgres Pro "на лету", можно настроить подключение, используя нижеописанные директивы, но только для типа экспорта COPY или INSERT, поскольку для схемы БД это не нужно.

PG_DSN

Используйте эту директиву, чтобы задать пространство имён в качестве источника данных Postgres Pro, используя модуль Perl DBD::Pg следующим образом:

```
dbi:Pg:dbname=pgdb;host=localhost;port=5432
```

Он подключается к базе данных pgdb на localhost по TCP-порту 5432.

Обратите внимание, что эта директива используется только для экспорта данных, результаты других типов экспорта необходимо импортировать вручную с использованием psql или любого другого клиента Postgres Pro.

Чтобы использовать зашифрованное соединение SSL, необходимо добавить в строку подключения `sslmode=require` следующим образом:

```
dbi:Pg:dbname=pgdb;host=localhost;port=5432;sslmode=require
```

`PG_USER`, `PG_PWD`

Эти директивы используются, чтобы задать имя пользователя и пароль. Если не задать пароль в `PG_PWD` и установить Perl-модуль `Term::ReadKey`, ora2pgpro запросит пароль интерактивно. Если имя пользователя не задано в `PG_USER`, его тоже нужно будет задать интерактивно.

`SYNCHRONOUS_COMMIT`

Указывает, что записи WAL должны быть записаны на диск до того, как команда фиксации транзакции сообщит клиенту об успешном завершении. Функциональность равнозначна установке параметра `synchronous_commit` в файле `postgresql.conf`. Используется только для загрузки данных в Postgres Pro напрямую, по умолчанию синхронная фиксация отключена для увеличения скорости записи данных.

`PG_INITIAL_COMMAND`

Эту директиву можно использовать для отправки начальных команд в Postgres Pro сразу после подключения, например, чтобы установить параметры сеанса. Директиву можно задавать несколько раз.

Управление типами столбцов

`PG_NUMERIC_TYPE`

Если задано значение 1, переносимые числовые типы заменяются внутренними типами Postgres Pro. Тип данных Oracle NUMBER(p,s) преобразовывается в типы данных Postgres Pro real и float приблизительно. Если есть поля с денежными суммами, во избежание проблем с округлением десятичных чисел необходимо сохранить тот же тип Postgres Pro numeric(p,s). Задавайте эту директиву, только если точность обязательна, поскольку преобразование в numeric(p,s) выполняется медленнее, чем в real или double precision .

`PG_INTEGER_TYPE`

Если задано значение 1, переносимые числовые типы заменяются внутренними типами Postgres Pro. Тип данных Oracle NUMBER(p) преобразовывается в типы данных Postgres Pro smallint, integer или bigint в зависимости от точности. NUMBER без указания точности преобразуется в `DEFAULT_NUMERIC` (см. ниже).

`DEFAULT_NUMERIC`

NUMBER без указания точности преобразуется по умолчанию в bigint, только если `PG_INTEGER_TYPE` имеет значение true. Можно поменять значение переменной на любой тип Postgres Pro, такой как integer или float.

DATA_TYPE

Если при преобразовании типов вы столкнулись с проблемами, используйте эту директиву, чтобы переопределить соответствие типов Oracle и Postgres Pro для преобразования ora2pgpro. Через запятую указываются пары типов `тип Oracle:тип Postgres Pro`. По умолчанию используется такой список:

```
DATA_TYPE
 VARCHAR2:varchar,NVARCHAR2:varchar,NVARCHAR:varchar,NCHAR:char,DATE:timestamp(
 RAW:bytea,CLOB:text,NCLOB:text,BLOB:bytea,BFILE:bytea,RAW(16):uuid,RAW(32):uui
 precision,DEC:decimal,DECIMAL:decimal,DOUBLE PRECISION:double
 precision,INT:integer,INTEGER:integer,REAL:real,SMALLINT:smallint,BINARY_FLOAT
 precision,BINARY_DOUBLE:double
 precision,TIMESTAMP:timestamp,XMLTYPE:xml,BINARY_INTEGER:integer,PLS_INTEGER:i
 WITH TIME ZONE:timestamp with time zone,TIMESTAMP WITH LOCAL TIME
 ZONE:timestamp with time zone
```

Имя директивы и определяемый список должны записываться в одной строке.

Обратите внимание, что если обнаруживаются столбцы типа RAW(16) и RAW(32) или в столбце типа RAW стоит значение по умолчанию `SYS_GUID()`, ora2pgpro автоматически преобразует тип столбца в uuid, что правильно в большинстве случаев. Такие данные автоматически переносятся как данные типа Postgres Pro uuid, предоставляемого модулем uuid-ossp.

Для замены типа с указанием точности и масштаба необходимо экранировать запятую с помощью символа обратной косой черты. Например, для преобразования всех значений типа NUMBER(*,0) в bigint вместо numeric(38), запишите так:

```
DATA_TYPE       NUMBER(*\,0):bigint
```

Записывайте только тот тип, который необходимо переопределить, не следует копировать все значения преобразования по умолчанию.

Преобразование BFILE имеет ряд особенностей. Если указать целевой тип TEXT, значения будут содержать только полный путь к внешнему файлу. Если указать BYTEA (по умолчанию), ora2pgpro экспортирует содержимое BFILE как bytea. Если указать целевой тип EFILE, ora2pgpro экспортирует данные как записи EFILE: (DIRECTORY, FILENAME). Используйте тип экспорта `DIRECTORY`, чтобы извлечь каталоги и права для этих каталогов.

SQL-функции для получения пути к BFILE нет, ora2pgpro создаёт её с использованием пакета `DBMS_LOB`.

```
CREATE OR REPLACE FUNCTION ora2pg_get_bfilename( p_bfile IN BFILE )
RETURN VARCHAR2
AS
    l_dir   VARCHAR2(4000);
    l_fname VARCHAR2(4000);
    l_path  VARCHAR2(4000);
BEGIN
    dbms_lob.FILEGETNAME( p_bfile, l_dir, l_fname );
    SELECT directory_path INTO l_path FROM all_directories
```

```
        WHERE directory_name = l_dir;
    l_dir := rtrim(l_path,'/');
    RETURN l_dir || '/' || l_fname;
END;
```

Эта функция создаётся, только если ora2pgpro находит таблицу со столбцом типа BFILE и указан целевой тип TEXT. По окончании экспорта функция удаляется. Директива используется с типами экспорта `COPY` и `INSERT`.

SQL-функции для получения записи типа EFILE из BFILE нет, ora2pgpro создаёт её с использованием пакета `DBMS_LOB`.

```
CREATE OR REPLACE FUNCTION ora2pg_get_efile( p_bfile IN BFILE )
RETURN VARCHAR2
AS
    l_dir   VARCHAR2(4000);
    l_fname VARCHAR2(4000);
BEGIN
    dbms_lob.FILEGETNAME( p_bfile, l_dir, l_fname );
    RETURN '(' || l_dir || ',' || l_fnamei || ')';
END;
```

Эта функция создаётся, только если ora2pgpro находит таблицу со столбцом типа BFILE и указан целевой тип EFILE. По окончании экспорта функция удаляется. Директива используется с типами экспорта `COPY` и `INSERT`.

Чтобы указать целевой тип, используйте директиву `DATA_TYPE`.

```
DATA_TYPE       BFILE:EFILE
```

Тип EFILE — это пользовательский тип, который можно создать с помощью расширения external_file для портирования типа BFILE в Postgres Pro.

SQL-функции для получения содержимого BFILE нет, ora2pgpro создаёт её с использованием пакета `DBMS_LOB`.

```
CREATE OR REPLACE FUNCTION ora2pg_get_bfile( p_bfile IN BFILE )
 RETURN
BLOB
  AS
        filecontent BLOB := NULL;
        src_file BFILE := NULL;
        l_step PLS_INTEGER := 12000;
        l_dir   VARCHAR2(4000);
        l_fname VARCHAR2(4000);
        offset NUMBER := 1;
  BEGIN
    IF p_bfile IS NULL THEN
      RETURN NULL;
    END IF;

DBMS_LOB.FILEGETNAME( p_bfile, l_dir, l_fname );
src_file := BFILENAME( l_dir, l_fname );
IF src_file IS NULL THEN
    RETURN NULL;
```

```
END IF;

    DBMS_LOB.FILEOPEN(src_file, DBMS_LOB.FILE_READONLY);
    DBMS_LOB.CREATETEMPORARY(filecontent, true);
    DBMS_LOB.LOADBLOBFROMFILE (filecontent, src_file,
 DBMS_LOB.LOBMAXSIZE, offset, offset);
    DBMS_LOB.FILECLOSE(src_file);
    RETURN filecontent;
END;
```

Эта функция создаётся, только если ora2pgpro находит таблицу со столбцом типа BFILE и указан целевой тип bytea (по умолчанию). По окончании экспорта функция удаляется. Директива используется с типами экспорта `COPY` и `INSERT`.

Типы ROWID и UROWID преобразуются в OID по умолчанию, но при импорте данных будут выдаваться ошибки. Аналогичного типа данных нет, поэтому используйте директиву `DATA_TYPE`, чтобы указать соответствующий тип Postgres Pro. Можно рассмотреть возможность замены на bigserial (последовательность с автоувеличением), text или uuid.

`MODIFY_TYPE`

Иногда может потребоваться принудительно использовать определённый целевой тип: например, столбец, экспортируемый ora2pgpro с типом timestamp, можно принудительно экспортировать как date. Значение следует указывать в формате `TABLE:COLUMN:TYPE` через запятую. Если внутри определения типа нужна запятая или пробел, следует экранировать их с помощью символа обратной косой черты.

```
MODIFY_TYPE     TABLE1:COL3:varchar,TABLE1:COL4:decimal(9\,6)
```

Тип `table1.col3` заменяется на varchar, а `table1.col4` — на decimal с указанием точности и масштаба.

Если задан пользовательский тип столбца, ora2pgpro автоматически определяет составной тип и экспортирует данные с помощью функции `ROW()`. Некоторые пользовательские типы Oracle представляют собой массивы из значений встроенных типов, в этом случае можно преобразовать значения такого типа в простой массив из значений встроенных типов Postgres Pro. Для этого укажите необходимый целевой тип, и ora2pgpro преобразует данные в массив. Например, если есть такое определение в Oracle:

```
CREATE OR REPLACE TYPE mem_type IS VARRAY(10) of VARCHAR2(15);
CREATE TABLE club (Name VARCHAR2(10),
        Address VARCHAR2(20),
        City VARCHAR2(20),
        Phone VARCHAR2(8),
        Members mem_type
);
```

Здесь пользовательский тип mem_type — это простой массив строк, который можно преобразовать следующим образом:

```
CREATE TABLE club (
```

```
        name varchar(10),
        address varchar(20),
        city varchar(20),
        phone varchar(8),
        members text[]
) ;
```

Директиву следует указывать следующим образом:

```
MODIFY_TYPE     CLUB:MEMBERS:text[]
```

ora2pgpro преобразует все данные из этого столбца в нужный формат. Поддерживаются массивы значений только символьных и числовых типов.

`TO_NUMBER_CONVERSION`

По умолчанию вызовы функции `TO_NUMBER` в Oracle преобразовываются в функции приведения к типу numeric. Например, `TO_NUMBER('10.1234')` преобразуется в вызов `to_number('10.1234')::numeric` в Postgres Pro. Измените значение этой директивы, если требуется выполнять приведение к типу integer или bigint. Формат приведения к типу тоже можно указать в значении директивы: например, со значением `TO_NUMBER_CONVERSION 99999999999999999999.9999999999` код выше будет преобразован в `TO_NUMBER('10.1234', '99999999999999999999.9999999999')`. Любое значение директивы, отличное от numeric, integer или bigint, принимается как маска формата. Если значение отсутствует, преобразование не выполняется.

`VARCHAR_TO_TEXT`

По умолчанию varchar2 без ограничения размера преобразуется в text. Если необходимо сохранить varchar, отключите эту директиву.

`FORCE_IDENTITY_BIGINT`

Как правило, столбец идентификации должен иметь тип bigint, чтобы соответствовать последовательности с автоувеличением, поэтому ora2pgpro всегда принудительно использует bigint. Если по каким-либо причинам необходимо сохранить тип данных, заданный для столбца идентификации, отключите эту директиву.

`TO_CHAR_NOTIMEZONE`

Включите эту директиву, если необходимо удалять информацию о часовом поясе для функций `TO_CHAR()`. По умолчанию отключена.

Настройка процесса экспорта

Нижеописанные директивы напрямую влияют на процесс экспорта и дают возможность оптимально настроить его.

`SKIP`

Если с типом экспорта `TABLE` не нужно экспортировать все ограничения схемы, укажите в директиве `SKIP` список ненужных ограничений, разделённых пробелами или запятыми. Возможные значения:

- fkeys: исключить ограничения внешнего ключа.
- pkeys: исключить первичные ключи.
- ukeys: исключить ограничения уникальности столбцов.
- indexes: исключить все остальные типы индексов.
- checks: исключить ограничения-проверки.

```
SKIP    indexes,checks
```

В примере выше из экспорта исключаются индексы и ограничения-проверки.

PKEY_IN_CREATE

Включите эту директиву, если необходимо добавить определение первичного ключа в оператор CREATE TABLE. Если директива отключена (по умолчанию), определение первичного ключа добавляется в оператор ALTER TABLE.

KEEP_PKEY_NAMES

По умолчанию имена первичных и уникальных ключей в исходной БД Oracle игнорируются, они генерируются автоматически в целевой БД Postgres Pro по соответствующим правилам именования. Если необходимо сохранить имена первичных и уникальных ключей из Oracle, задайте для этой директивы значение 1.

FKEY_ADD_UPDATE

Эта директива позволяет добавить параметр ON UPDATE CASCADE для внешнего ключа, когда есть ON DELETE CASCADE или всегда. Oracle не поддерживает эту функциональность, для добавления ON UPDATE CASCADE необходимо использовать триггер. Поскольку в Postgres Pro эта функциональность есть, можно выбрать метод добавления действия внешнего ключа. Для директивы доступны следующие значения:

- never: внешние ключи объявляются так, как в Oracle.
- delete: действие ON UPDATE CASCADE добавляется только, если для внешних ключей есть ON DELETE CASCADE.
- always: внешние ключи определяются во время изменения.

FKEY_DEFERRABLE

При экспорте таблиц ora2pgpro, как правило, экспортирует ограничения как есть: если они неоткладываемые, то экспортируются как неоткладываемые. Однако неоткладываемые ограничения могут вызвать проблемы при попытке импорта данных в Postgres Pro. Задайте для директивы FKEY_DEFERRABLE значение 1, чтобы все ограничения внешнего ключа экспортировались как откладываемые.

DEFER_FKEY

Когда для директивы DEFER_FKEY задано значение 1, добавляется команда для откладывания всех ограничений внешнего ключа во время экспорта, а импорт выполняется в одной транзакции. Это будет работать, только если внешние ключи экспортируются как откладываемые и данные не импортируются в Postgres Pro напрямую (не задана директива PG_DSN). Ограничения проверяются в конце транзакции.

Кроме того, директиву можно включить, чтобы принудительно создавать все внешние ключи как откладываемые и изначально отложенные во время экспорта схемы (тип экспорта TABLE).

DROP_FKEY

Если откладывание внешних ключей не представляется возможным из-за большого объёма данных в одной транзакции, внешние ключи не были экспортированы как откладываемые или импорт в Postgres Pro осуществляется напрямую, можно использовать директиву DROP_FKEY, чтобы удалить все внешние ключи до импорта данных и создать их заново в конце импорта.

DROP_INDEXES

Эта директива позволяет значительно увеличить скорость импорта данных путём удаления всех индексов, не являющихся автоматическими (индексы по первичным ключам), и создания их заново в конце импорта. Рекомендуется не импортировать индексы и ограничения до импорта всех данных.

DISABLE_TRIGGERS

Директива используется для отключения триггеров для всех таблиц при типе экспорта COPY или INSERT. Возможные значения: USER (отключение только пользовательских триггеров) и ALL (включая системные триггеры для ссылочной целостности). Значение по умолчанию 0 — для отключения триггера до импорта данных SQL-операторы не добавляются.

Если необходимо отключить триггеры во время миграции данных, при подключении под именем суперпользователя Postgres Pro задайте значение ALL, в противном случае —USER. Значение 1 равняется USER.

DISABLE_SEQUENCE

Со значением 1 директива отключает изменение последовательностей для всех таблиц, если указан тип экспорта COPY или INSERT. Полезно во избежание изменения последовательности во время миграции данных. Значение по умолчанию — 0, то есть последовательности меняются.

NOESCAPE

По умолчанию все данные типов, отличных от даты или времени, должны экранироваться. Если с этим возникают проблемы, задайте для данной директивы значение 1, чтобы отключить экранирование символов во время экспорта данных. Эта директива используется только с типом экспорта COPY. Включить/отключить экранирование символов для операторов INSERT можно с помощью директивы STANDARD_CONFORMING_STRINGS.

STANDARD_CONFORMING_STRINGS

Эта директива определяет, будет ли обратная косая черта в обычных строковых константах ('...') восприниматься буквально, как того требует стандарт SQL. По умолчанию директива включена, то есть при её ненулевом значении ora2pgpro использует синтаксис спецпоследовательностей (E'...'). Директива работает точно так же, как аналогичный параметр в Postgres Pro, и используется во время экспорта только для операторов INSERT. Вклю-

чить/отключить экранирование символов для операторов COPY можно с помощью директивы NOESCAPE.

TRIM_TYPE

Если для преобразования CHAR(n) из Oracle в varchar(n) или text в Postgres Pro задана директива DATA_TYPE, может потребоваться усечение данных. Если задать данную директиву, по умолчанию ora2pgpro автоматически находит и удаляет пробельные символы в начале и конце строки (значение BOTH). Если необходимо удалять пробелы только в начале строк, задайте значение LEADING. Если необходимо удалять пробелы только в конце строк, задайте значение TRAILING.

TRIM_CHAR

По умолчанию директива закомментирована, усекаются пробелы. Используйте директиву, чтобы поменять удаляемый символ: например, на –, чтобы усекать – в начале поля типа char(n).

PRESERVE_CASE

Если необходимо сохранить регистр имён объектов Oracle, задайте для этой директивы значение 1. По умолчанию ora2pgpro преобразует все имена объектов Oracle в нижний регистр. Не рекомендуется включать эту директиву, поскольку в таком случае придётся заключать в кавычки все имена объектов во всех SQL-скриптах.

ORA_RESERVED_WORDS

Разрешает экранировать имена столбцов с использованием зарезервированных слов Oracle. В значении указывается список зарезервированных слов, разделённых запятыми. По умолчанию: audit,comment,references.

USE_RESERVED_WORDS

Включите эту директиву, если есть имена таблиц или столбцов, являющиеся зарезервированными словами Postgres Pro. ora2pgpro заключит имена таких объектов в кавычки.

GEN_USER_PWD

Задайте для этой директивы значение 1, чтобы заменить пароль по умолчанию случайным паролем для всех пользователей во время экспорта GRANT.

PG_SUPPORTS_MVIEW

В Postgres Pro материализованные представления создаются с помощью SQL-синтаксиса CREATE MATERIALIZED VIEW. Используйте эту директиву (по умолчанию включена), чтобы в ora2pgpro использовалась встроенная поддержка Postgres Pro. Отключите директиву, чтобы использовать старый вариант синтаксиса с таблицей и набором функций.

PG_VERSION

Задаёт номер мажорной версии целевой БД Postgres Pro. Например: 11 или 16. Значение по умолчанию — последняя мажорная версия на момент релиза.

BITMAP_AS_GIN

Включает использование расширения btree_gin для создания индексов со сканированием битовой карты. Расширение должно быть создано. По умолчанию создаются индексы GIN, когда директива отключена — индексы B-деревья.

LONGREADLEN

Используйте эту директиву, чтобы задать предполагаемый размер самого большого объекта в Oracle по свойству LongReadLen. Значение по умолчанию — 1 МБ, чего может быть недостаточно для извлечения файлов BLOB или CLOB. Если размер LOB превышает LONGREADLEN, DBD::Oracle возвращает ошибку «ORA-24345: A Truncation» (Усечение). По умолчанию: 1023*1024 байт.

> **Важно**
>
> Если увеличить значение этой директивы, по всей вероятности нужно уменьшить значение DATA_LIMIT. Даже если размер файла BLOB составляет 1МБ, при попытке прочитать 10000 таких файлов (значение DATA_LIMIT по умолчанию) понадобится 10 ГБ памяти. Попробуйте извлекать данные из таких таблиц по очереди и задать для DATA_LIMIT значение 500 или ниже, в противном случае может произойти нехватка памяти.

LONGTRUNKOK

Если необходимо обойти ошибку «ORA-24345: A Truncation», задайте для этой директивы значение 1, чтобы данные усекались до значения LONGREADLEN. По умолчанию директива отключена, то есть будет выводиться предупреждение, если для LONGREADLEN задано недостаточно большое значение.

USE_LOB_LOCATOR

Отключите эту директиву, если необходимо загрузить содержимое файлов BLOB и CLOB полностью без использования указателей на LOB. Для этого необходимо выставить подходящее значение LONGREADLEN. Обратите внимание, что при этом скорость экспорта BLOB не увеличится, поскольку большая часть времени уходит на экранирование значений bytea и экспорт происходит построчно, а не блоками по DATA_LIMIT строк. По умолчанию директива включена, используются указатели на LOB.

LOB_CHUNK_SIZE

Oracle рекомендует чтение и запись LOB порциями величиной, кратной размеру блоков LOB. Размер блока по умолчанию составляет 8КБ (8192). Последние тесты показывали улучшение производительности при более высоких значениях, таких как 512КБ или 4МБ.

Результаты быстрого теста на 30120 строках с файлами BLOB разного размера (200x5МБ, 19800x212КБ, 10000x942КБ, 100x17МБ, 20x156МБ) при

DATA_LIMIT=100, LONGREADLEN=170МБ и общим размером таблицы 20 ГБ:

```
no lob locator  : 22m46,218s (1365 sec., avg: 22 recs/sec)
chunk size 8k   : 15m50,886s (951 sec., avg: 31 recs/sec)
chunk size 512k : 1m28,161s (88 sec., avg: 342 recs/sec)
chunk size 4Mb  : 1m23,717s (83 sec., avg: 362 recs/sec)
```

Таким образом, со значением LOB_CHUNK_SIZE в 4 МБ можно получить ускорение более чем в 10 раз. В этой директиве можно подбирать подходящее значение в зависимости от размера большинства объектов BLOB. Например, если размер большинства объектов LOB меньше 8КБ, достаточно задать для директивы значение 8192 для экономии памяти. Значение LOB_CHUNK_SIZE по умолчанию — 512000.

XML_PRETTY

Указывает использовать функцию getStringVal() вместо getClobVal() для экспорта данных XML. По умолчанию 1, функциональность включена для обратной совместимости. Задайте значение 0, чтобы извлекать данные как CLOB. Обратите внимание, что значение XML, извлекаемое с помощью with getStringVal() не должно превышать заданный в VARCHAR2 предел (4000), в противном случае выдаётся ошибка.

ENABLE_MICROSECOND

Задайте для директивы значение O, если необходимо отключить экспорт миллисекунд из столбцов типа timestamp Oracle. По умолчанию миллисекунды экспортируются с использованием следующего формата:

```
'YYYY-MM-DD HH24:MI:SS.FF'
```

При отключенной директиве будет использоваться такой формат:

```
to_char(..., 'YYYY-MM-DD HH24:MI:SS')
```

По умолчанию миллисекунды экспортируются.

DISABLE_COMMENT

Задайте для директивы значение 1, чтобы не экспортировать комментарии к таблицам и столбцам. По умолчанию директива включена.

Дополнительные параметры для настройки кодировок клиентов

NLS_LANG

По умолчанию ora2pgpro задаёт для NLS_LANG значение AMERICAN_AMERICA.AL32UTF8, а для NLS_NCHAR — AL32UTF8. Не рекомендуется изменять эти значения, но в некоторых случаях это может быть полезно. При изменении значений этой директивы кодировка клиента на стороне Oracle изменится с изменением значений переменных окружения $ENV{NLS_LANG} и $ENV{NLS_NCHAR}.

BINMODE

По умолчанию ora2pgpro заставляет Perl использовать кодировку ввода-вывода UTF8 путём вызова прагмы Perl:

```
use open ':utf8';
```

Кодировку можно переопределить, задав директиву BINMODE: например, задайте значение :locale, чтобы использовать локаль системы, или iso-8859-7.

```
use open ':locale';
use open ':encoding(iso-8859-7)';
```

Если в NLS_LANG задана локаль, отличная от кодировки UTF8, можно дополнительно указать эту директиву. Для большинства случаев оставьте директиву закомментированной.

CLIENT_ENCODING

Во избежание проблем кодировка клиента Postgres Pro автоматически указывается как UTF8. Если не используется значение NLS_LANG по умолчанию, может понадобиться изменить кодировку клиента Postgres Pro.

Список поддерживаемых Postgres Pro наборов символов описан в разделе Поддержка кодировок.

FORCE_PLSQL_ENCODING

Включите эту директиву, чтобы использовать кодировку UTF8 для экспортируемого кода PL/SQL. Директива может быть полезна в особых случаях.

Преобразование кода PL/SQL в код PL/pgSQL

Автоматическое преобразование кода из PL/SQL Oracle в PL/pgSQL Postgres Pro требует доработки в ora2pgpro и некоторых ручных действий. Код Perl code, используемый для преобразования, хранится в специально модуле Perl Ora2Pgpro/PLSQL.pm.

PLSQL_PGSQL

Включает/отключает преобразование PL/SQL в PL/pgSQL. По умолчанию включено.

NULL_EQUAL_EMPTY

ora2pgpro может заменять все условия с тестом на NULL вызовом функции coalesce(), чтобы имитировать поведение Oracle, где пустые строки считаются равными NULL.

```
(field1 IS NULL) is replaced by (coalesce(field1::text, '') = '')
(field2 IS NOT NULL) is replaced by (field2 IS NOT NULL AND
 field2::text <> '')
```

Такая замена может потребоваться, чтобы приложение работало аналогично, но рекомендуется преобразовывать пустые строки в NULL, поскольку Postgres Pro воспринимает их по-разному.

EMPTY_LOB_NULL

Включает экспорт функций empty_clob() и empty_blob() как NULL вместо пустой строки для первой и \x для второй. Если использование NULL в

столбце допускается, это поможет увеличить скорость экспорта при наличии большого количества пустых больших объектов. По умолчанию точно сохраняются данные из Oracle.

PACKAGE_AS_SCHEMA

Если необходимо экспортировать пакеты как простые функции, а не схемы, можно заменить все вызовы `package_name.function_name`. Если отключить директиву `PACKAGE_AS_SCHEMA`, ora2pgpro заменит все вызовы `package_name.function_name()` на `package_name_function_name()`. По умолчанию пакеты экспортируются как схемы.

Замена будет производиться во всех операциях DDL или коде, разбираемом при преобразовании кода PL/SQL в PL/pgSQL. При этом должна быть включена директива `PLSQL_PGSQL`, или в командной строке передан параметр `-p`.

REWRITE_OUTER_JOIN

Включите эту директиву, если не работает модификация стандартного синтаксиса Oracle для `OUTER JOIN (+)`. В этом случае ora2pgpro не будет модифицировать такой код, по умолчанию сейчас производится модификация простой формы правого внешнего соединения.

UUID_FUNCTION

По умолчанию ora2pgpro преобразует вызов функции Oracle `SYS_GUID()` в вызов функции `uuid_generate_v4` расширения uuid-ossp. Чтобы использовать вместо неё функцию `gen_random_uuid` расширения pgcrypto, можно задать в качестве значения этой директивы имя функции. По умолчанию используется `uuid_generate_v4`.

Обратите внимание, что если обнаруживаются столбцы типа RAW(16) и RAW(32) или в столбце типа RAW стоит значение по умолчанию `SYS_GUID()`, ora2pgpro автоматически преобразует тип столбца в uuid, что правильно в большинстве случаев. В этом случае данные автоматически переносятся как данные типа Postgres Pro uuid, предоставляемого модулем uuid-ossp.

FUNCTION_STABLE

По умолчанию функции Oracle помечаются как `STABLE`, поскольку они не изменяют данные, кроме как в PL/SQL с назначением переменных или в качестве условного выражения. Отключите эту директиву, чтобы утилита ora2pgpro создавала такие функции как `VOLATILE`.

COMMENT_COMMIT_ROLLBACK

По умолчанию ora2pgpro оставляет вызовы `COMMIT`/`ROLLBACK` без изменений, чтобы пользователь проверил логику функции. Включите эту директиву, когда исходный код Oracle исправлен или необходимо закомментировать эти вызовы.

COMMENT_SAVEPOINT

В процедурах PL/SQL вызовам `SAVEPOINT` часто сопутствуют команды `ROLLBACK TO savepoint_name`. Когда директива `COMMENT_COMMIT_ROLLBACK`

включена и вызовы SAVEPOINT также нужно закомментировать, включите эту директиву.

STRING_CONSTANT_REGEXP

Во время преобразования кода PL/SQL в PL/pgSQL утилитаora2pgpro заменяет все строковые константы на текст в одинарных кавычках. Если в динамических вызовах используются местозаполнители в строках, в данной директиве можно задать список регулярных выражений, которые заменяются на время разбора. Например:

```
STRING_CONSTANT_REGEXP          <placeholder value=".*">
```

Регулярные выражения в списке разделяются точкой с запятой.

ALTERNATIVE_QUOTING_REGEXP

В данной директиве задайте в качестве значения регулярное выражение с поиском подходящих строк для извлечения текстовой части, чтобы включить поддержку механизма альтернативных кавычек ('Q' или 'q'). Например, для переменной, заданной как

```
c_sample VARCHAR2(100 CHAR) := q'{This doesn't work.}';
```

следует использовать регулярное выражение:

```
ALTERNATIVE_QUOTING_REGEXP     q'{(.*)}'
```

ora2pgpro использует разделитель $$, результат будет следующим:

```
c_sample varchar(100) := $$This doesn't work.$$;
```

Значение данной директивы представляет собой список регулярных выражений, разделённых точкой с запятой. Обязательно указать часть поиска подходящих строк (в скобках) для каждого выражения, если необходимо восстановить строковую константу.

USE_ORAFCE

Включите эту директиву, если необходимо использовать функции, определённые в библиотеке orafce, и не преобразовывать вызовы этих функций с помощью ora2pgpro.

По умолчанию ora2pgpro заменяет функции add_month(), add_year(), date_trunc() и to_char(), но можно использовать версии этих функций из orafce, которые не требуют преобразования кода.

INCLUDE_PACKAGES

Содержит список экспортируемых пакетов, разделённых запятыми. Используется только с типом экспорта PACKAGE и только последнее вхождение в файле конфигурации.

EXCLUDE_PACKAGES

Содержит список исключаемых из экспорта пакетов, разделённых запятыми. Используется только с типом экспорта PACKAGE и только последнее вхождение в файле конфигурации.

`POSTGRESPRO_ATX`

Если задано значение 1, автономные транзакции экспортируются напрямую как автономные транзакции Postgres Pro.

Материализованные представления

Материализованные представления экспортируются в виде снимка «Snapshot Materialized Views» (Снимок материализованных представлений), поскольку Postgres Pro поддерживает только полное обновление.

При экспорте материализованных представлений ora2pgpro сначала добавляет код SQL для создания таблицы `materialized_views` (материализованные представления):

```
CREATE TABLE materialized_views (
        mview_name text NOT NULL PRIMARY KEY,
        view_name text NOT NULL,
        iname text,
        last_refresh TIMESTAMP WITH TIME ZONE
);
```

Для каждого материализованного представления создаётся одна запись в этой таблице. Затем добавляется код PL/pgSQL для создания деревьев функций:

- Функция `create_materialized_view(text, text, text)` используется для создания материализованного представления.
- Функция `drop_materialized_view(text)` используется для удаления материализованного представления.
- Функция `refresh_full_materialized_view(text)` используется для обновления материализованного представления.

Далее добавляется SQL-код для создания представления и материализованного представления:

```
CREATE VIEW mviewname_mview AS
SELECT ... FROM ...;

SELECT
 create_materialized_view('mviewname','mviewname_mview', заменить на
 имя столбца, используемого для создания индекса);
```

Первый аргумент — это имя материализованного представления, второй — имя представления, на основе которого создаётся материализованное представление, а третий — имя столбца для построения индекса (а также первичного ключа в большинстве случаев). Этот столбец не рассчитывается автоматически, поэтому его имя необходимо заменить.

Как было сказано выше, ora2pgpro поддерживает только снимок материализованных представлений, поэтому таблица будет обновлена полностью при опустошении таблицы и повторной загрузке данных из представления:

```
refresh_full_materialized_view('mviewname');
```

Чтобы удалить материализованное представление, вызовите функцию `drop_materialized_view()` с именем материализованного представления в качестве параметра.

Прочие директивы

`DEBUG`

Со значением 1 выводятся подробные сообщения.

`IMPORT`

Общие директивы ora2pgpro можно записать в отдельном файле и импортировать этот файл в другие файлы конфигурации, используя директиву `IMPORT` следующим образом:

```
IMPORT  commonfile.conf
```

При этом все директивы из файла `commonfile.conf` будут импортированы в текущий файл конфигурации.

Экспорт представлений как таблиц Postgres Pro

Любое представление Oracle можно экспортировать как таблицу Postgres Pro, задав для директивы `TYPE` значение `TABLE`, чтобы создавались соответствующие операторы `CREATE TABLE`. Также можно использовать тип экспорта `COPY` или `INSERT`, чтобы экспортировать данные соответствующим образом. Чтобы это было возможно, укажите представления в директиве `VIEW_AS_TABLE`.

В этом случае, если ora2pgpro обнаруживает представление, то извлекает схемы (если `TYPE=TABLE`) в форме `CREATE TABLE`, а затем извлекает данные (если `TYPE=COPY` или `TYPE=INSERT`) по схеме.

Например, рассмотрим следующее представление:

```
CREATE OR REPLACE VIEW product_prices (category_id, product_count,
 low_price, high_price) AS
SELECT  category_id, COUNT(*) as product_count,
    MIN(list_price) as low_price,
    MAX(list_price) as high_price
 FROM   product_information
GROUP BY category_id;
```

Если для такого представления задать для директивы `VIEW_AS_TABLE` значение `product_prices` и использовать тип экспорта `TABLE`, ora2pgpro определит типы возвращаемых столбцов и создаст соответствующий оператор `CREATE TABLE`:

```
CREATE TABLE product_prices (
        category_id bigint,
        product_count integer,
        low_price numeric,
        high_price numeric
);
```

Данные будут загружены в зависимости от типа экспорта `COPY` или `INSERT` и объявления типа.

Кроме того, для фильтрации экспортируемых объектов можно использовать директивы `ALLOW` и `EXCLUDE`.

Оценка стоимости миграции

Нелегко оценить стоимость процесса миграции из Oracle в Postgres Pro. Чтобы достаточно хорошо произвести такую оценку, ora2pgpro анализирует все объекты в БД, все функции и хранимые процедуры, чтобы определить, есть ли объекты и код PL/SQL, которые невозможно преобразовать автоматически.

Для анализа БД Oracle в ora2pgpro есть специальный режим анализа, который позволяет сгенерировать отчёт о содержимом БД и возможности его экспорта.

Чтобы активировать режим анализа и отчёта, необходимо использовать тип экспорта `SHOW_REPORT` следующим образом:

```
ora2pgpro -t SHOW_REPORT
```

Пример отчёта, генерируемого командой:

```
--------------------------------------
Ora2Pg: Oracle Database Content Report
--------------------------------------
Version Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
Schema  HR
Size  880.00 MB

--------------------------------------
Object  Number  Invalid Comments
--------------------------------------
CLUSTER   2 0 Clusters are not supported and will not be exported.
FUNCTION  40  0 Total size of function code: 81992.
INDEX     435 0 232 index(es) are concerned by the export, others are
 automatically generated and will
                                do so on PostgreSQL. 1 bitmap
 index(es). 230 b-tree index(es). 1 reversed b-tree index(es)
                                Note that bitmap index(es) will be
 exported as b-tree index(es) if any. Cluster, domain,
                                bitmap join and IOT indexes will not
 be exported at all. Reverse indexes are not exported
                                too, you may use a trigram-based index
 (see pg_trgm) or a reverse() function based index
                                and search. You may also use
 'varchar_pattern_ops', 'text_pattern_ops' or 'bpchar_pattern_ops'
                                operators in your indexes to improve
 search with the LIKE operator respectively into
                                varchar, text or char columns.
MATERIALIZED VIEW 1 0 All materialized view will be exported as
 snapshot materialized views, they
                                are only updated when fully refreshed.
PACKAGE BODY  2 1 Total size of package code: 20700.
PROCEDURE 7 0 Total size of procedure code: 19198.
SEQUENCE  160 0 Sequences are fully supported, but all call to
 sequence_name.NEXTVAL or sequence_name.CURRVAL
                                will be transformed into
 NEXTVAL('sequence_name') or CURRVAL('sequence_name').
TABLE     265 0 1 external table(s) will be exported as standard
 table. See EXTERNAL_TO_FDW configuration
```

```
                                directive to export as file_fdw
 foreign tables or use COPY in your code if you just
                                want to load data from external files.
 2 binary columns. 4 unknown types.
TABLE PARTITION 8 0 Partitions are exported using table inheritance
 and check constraint. 1 HASH partitions.
                                2 LIST partitions. 6 RANGE partitions.
 Note that Hash partitions are not supported.
TRIGGER   30  0 Total size of trigger code: 21677.
TYPE      7 1 5 type(s) are concerned by the export, others are not
 supported. 2 Nested Tables.
                                2 Object type. 1 Subtype. 1 Type Boby.
 1 Type inherited. 1 Varrays. Note that Type
                                inherited and Subtype are converted as
 table, type inheritance is not supported.
TYPE BODY 0 3 Export of type with member method are not supported,
 they will not be exported.
VIEW      7 0 Views are fully supported, but if you have updatable
 views you will need to use
                                INSTEAD OF triggers.
DATABASE LINK 1 0 Database links will not be exported. You may try the
 dblink perl contrib module or use
                                the SQL/MED PostgreSQL features with
 the different Foreign Data Wrapper (FDW) extensions.

Note: Invalid code will not be exported unless the EXPORT_INVALID
 configuration directive is activated.
```

Поскольку утилита ora2pgpro может преобразовывать код SQL и PL/SQL из синтаксиса Oracle в Postgres Pro, она может оценивать трудности в коде и время, необходимое для выполнения полной миграции БД.

Для оценки стоимости миграции в человеко-днях, ora2pgpro предоставляет директиву `ESTIMATE_COST`, которую также можно включить в командной строке: `--estimate_cost`.

Эту функциональность можно использовать только с типами экспорта `SHOW_REPORT`, `FUNCTION`, `PROCEDURE`, `PACKAGE` и `QUERY`.

```
ora2pgpro -t SHOW_REPORT  --estimate_cost
```

Сгенерированный отчёт выглядит так же, как в предыдущем примере, но добавляется столбец `Estimated cost` (Предполагаемая стоимость) следующим образом:

```
---------------------------------------
Ora2Pg: Oracle Database Content Report
---------------------------------------
Version Oracle Database 10g Express Edition Release 10.2.0.1.0
Schema  HR
Size  890.00 MB

---------------------------------------
Object  Number  Invalid Estimated cost  Comments
```

```
-------------------------------------
DATABASE LINK  3 0 9 Database links will be exported as SQL/MED
 PostgreSQL's Foreign Data Wrapper (FDW) extensions
                                 using oracle_fdw.
FUNCTION  2 0 7 Total size of function code: 369 bytes. HIGH_SALARY:
 2, VALIDATE_SSN: 3.
INDEX 21  0 11  11 index(es) are concerned by the export, others are
 automatically generated and will do so
                                 on PostgreSQL. 11 b-tree index(es).
 Note that bitmap index(es) will be exported as b-tree
                                 index(es) if any. Cluster, domain,
 bitmap join and IOT indexes will not be exported at all.
                                 Reverse indexes are not exported too,
 you may use a trigram-based index (see pg_trgm) or a
                                 reverse() function based index and
 search. You may also use 'varchar_pattern_ops', 'text_pattern_ops'
                                 or 'bpchar_pattern_ops' operators in
 your indexes to improve search with the LIKE operator
                                 respectively into varchar, text or
 char columns.
JOB 0 0 0 Job are not exported. You may set external cron job with
 them.
MATERIALIZED VIEW 1 0 3 All materialized view will be exported as
 snapshot materialized views, they
                                       are only updated when fully
 refreshed.
PACKAGE BODY  0 2 54  Total size of package code: 2487 bytes. Number
 of procedures and functions found
                                       inside those packages: 7.
 two_proc.get_table: 10, emp_mgmt.create_dept: 4,
                                       emp_mgmt.hire: 13,
 emp_mgmt.increase_comm: 4, emp_mgmt.increase_sal: 4,
                                       emp_mgmt.remove_dept: 3,
 emp_mgmt.remove_emp: 2.
PROCEDURE 4 0 39  Total size of procedure code: 2436 bytes.
 TEST_COMMENTAIRE: 2, SECURE_DML: 3,
                                       PHD_GET_TABLE: 24,
 ADD_JOB_HISTORY: 6.
SEQUENCE  3 0 0 Sequences are fully supported, but all call to
 sequence_name.NEXTVAL or sequence_name.CURRVAL
                                       will be transformed into
 NEXTVAL('sequence_name') or CURRVAL('sequence_name').
SYNONYM   3 0 4 SYNONYMs will be exported as views. SYNONYMs do not
 exists with PostgreSQL but a common workaround
                                       is to use views or set the
 PostgreSQL search_path in your session to access
                                       object outside the current
 schema.
                                       user1.emp_details_view_v is an
 alias to hr.emp_details_view.
                                       user1.emp_table is an alias to
 hr.employees@other_server.
                                       user1.offices is an alias to
 hr.locations.
```

```
TABLE 17  0 8.5 1 external table(s) will be exported as standard
 table. See EXTERNAL_TO_FDW configuration
                                directive to export as file_fdw
 foreign tables or use COPY in your code if you just want to
                                load data from external files. 2
 binary columns. 4 unknown types.
TRIGGER 1 1 4 Total size of trigger code: 123 bytes.
 UPDATE_JOB_HISTORY: 2.
TYPE  7 1 5 5 type(s) are concerned by the export, others are not
 supported. 2 Nested Tables. 2 Object type.
                                1 Subtype. 1 Type Boby. 1 Type
 inherited. 1 Varrays. Note that Type inherited and Subtype are
                                converted as table, type inheritance
 is not supported.
TYPE BODY 0 3 30  Export of type with member method are not supported,
 they will not be exported.
VIEW  1 1 1 Views are fully supported, but if you have updatable views
 you will need to use INSTEAD OF triggers.
-------------------------------------
Total 65  8 162.5 162.5 cost migration units means approximatively 2
 man day(s).
```

Последняя строка показывает предполагаемое количество человеко-дней на миграцию кода с блоками миграции для каждого объекта. Блок миграции составляет 5 минут, что соответствует скорости миграции, выполняемой специалистом Postgres Pro. Для первой миграции можно увеличить это значение в директиве `COST_UNIT_VALUE` или передав параметр `--cost_unit_value` в командной строке:

```
ora2pgpro -t SHOW_REPORT  --estimate_cost --cost_unit_value 10
```

ora2pgpro может выдавать оценку уровня сложности миграции, например Migration level: B-5.

```
Уровни миграции:
    A - Миграцию можно запустить автоматически.
    B - Миграция требует замены кода, срок до 5 человеко-дней.
    C - Миграция требует замены кода, срок свыше 5 человеко-дней.
Технические уровни:
    1 = незначительный: без хранимых процедур и триггеров.
    2 = простой: без хранимых процедур, но с триггерами, без замены
 кода вручную.
    3 = средний: хранимые процедуры и/или триггеры, без замены кода
 вручную.
    4 = ручной: без хранимых процедур, но с триггерами или
 представлениями, с заменой кода.
    5 = сложный: хранимые процедуры и/или триггеры, с заменой кода.
```

Выдаваемый уровень оценки содержит букву (A и B), показывающую необходимость ручной замены кода, и цифру от 1 до 5, определяющую уровень технической сложности. Кроме того, предоставляется параметр `--human_days_limit`, в котором можно задать предел человеко-дней, по достижении которого уровень миграции меняется на C, чтобы показать необходимость проведения длительных работ и управления проектом миграции с поддержкой. Значение по

умолчанию — 10 человеко-дней. Чтобы изменить это значение на постоянной основе, используйте директиву `HUMAN_DAYS_LIMIT`.

Эта функциональность помогает определить порядок миграции баз данных и команду для выполнения работ.

Глобальная оценка миграции

ora2pgpro содержит скрипт `ora2pgpro_scanner`, который можно использовать при большом количестве анализируемых экземпляров и схем.

```
ora2pgpro_scanner -l CSVFILE [-o OUTDIR]

-b | --binpath DIR: full path to directory where the ora2pgpro binary
 stays.
             Might be useful only on Windows OS.
-c | --config FILE: set custom configuration file to use otherwise
 ora2pgpro
             will use the default: /etc/ora2pgpro/ora2pgpro.conf.
-l | --list FILE : CSV file containing a list of databases to scan
 with
             all required information. The first line of the file
             can contain the following header that describes the
             format that must be used:

"type","schema/database","dsn","user","password"

-o | --outdir DIR : (optional) by default all reports will be dumped
 to a
             directory named 'output', it will be created
 automatically.
             If you want to change the name of this directory, set the
 name
             at second argument.

-t | --test : just try all connections by retrieving the required
 schema
              or database name. Useful to validate your CSV list file.
-u | --unit MIN : redefine globally the migration cost unit value in
 minutes.
              Default is taken from the ora2pgpro.conf (default 5
 minutes).

Here is a full example of a CSV databases list file:

"type","schema/database","dsn","user","password"
"MYSQL","sakila","dbi:mysql:host=192.168.1.10;database=sakila;port=3306","root","s
"ORACLE","HR","dbi:Oracle:host=192.168.1.10;sid=XE;port=1521","system","manager"
"MSSQL","HR","dbi:ODBC:driver=msodbcsql18;server=srv.database.windows.net;database

The CSV field separator must be a comma.

Note that if you want to scan all schemas from an Oracle instance you
 just
```

```
have to leave the schema field empty, Ora2PgPro will automatically
 detect all
available schemas and generate a report for each one. Of course you
 need to
use a connection user with enough privileges to be able to scan all
 schemas.
For example:

"ORACLE","","dbi:Oracle:host=192.168.1.10;sid=XE;port=1521","system","manager"
"MSSQL","","dbi:ODBC:driver=msodbcsql18;server=srv.database.windows.net;database=t

will generate a report for all schema in the XE instance. Note that in
 this
case the SCHEMA directive in ora2pgpro.conf must not be set.
```

В результате создаётся файл CSV с оценкой, в котором каждой строке соответствует одна схема или база данных, и подробный HTML-отчёт для каждой сканируемой БД.

Подсказка: используйте параметр `-t | --test`, чтобы в файле CSV протестировать все соединения.

Пользователям Windows следует использовать в командной строке параметр `-b`, чтобы указать каталог для хранения `ora2pgpro_scanner`, в противном случае вызовы команд ora2pgpro завершатся ошибкой.

ora2pgpro всегда включает 2 блока миграции с типом `TEST` и 1 блок с SIZE на 1000 строк кода в оценку миграции функций. Это означает, что по умолчанию на каждую функцию в оценку миграции добавляется 15 минут. Разумеется, при наличии модульных тестов или множества простых функций оценка будет завышенной.

Метод оценки миграции

Показатели миграции, назначаемые для каждого типа объекта БД Oracle, определяются в библиотеке Perl `lib/Ora2Pg/PLSQL.pm`, указанной в переменной `%OBJECT_SCORE`.

Количество строк PL/SQL для блока миграции также задаётся в этом файле в переменной `$SIZE_SCORE`.

Количество блоков миграции, связанных с трудностями в коде PL/SQL, также указываются в библиотеке Perl `lib/Ora2Pgpro/PLSQL.pm` в `%UNCOVERED_SCORE`.

Этот метод оценки требует доработки.

Увеличение скорости создания индексов и ограничений

С типом экспорта `LOAD` можно отправлять SQL-команды из файла по нескольким соединениям с Postgres Pro. Чтобы использовать эту функциональности, необходимо задать директивы `PG_DSN`, `PG_USER` и `PG_PWD`.

```
ora2pgpro -t LOAD -c config/ora2pgpro.conf -i schema/tables/
INDEXES_table.sql -j 4
```

При этом индексы будут создаваться одновременно по 4 соединениям с Postgres Pro. Это значительно ускорит часть процесса миграции с большим объёмом данных.

Экспорт LONG RAW

ora2pgpro не может экспортировать данные столбцов типа LONG RAW. Библиотека OCI не может экспортировать и всегда возвращает одну и ту же запись. Чтобы экспортировать такие данные, преобразуйте поле в BLOB путём создания временной таблицы до миграции данных.

```
SQL> DESC TEST_LONGRAW
 Name                  NULL ?   Type
 --------------------- -------- ----------------------------
 ID                             NUMBER
 C1                             LONG RAW
```

Например, таблицу Oracle выше нужно перенести в таблицу с BLOB следующим образом:

```
CREATE TABLE test_blob (id NUMBER, c1 BLOB);
```

А затем копировать данные с помощью такого запроса `INSERT`:

```
INSERT INTO test_blob SELECT id, to_lob(c1) FROM test_longraw;
```

Затем нужно исключить изначальную таблицу из экспорта (см. `EXCLUDE`) и переименовать новую временную таблицу "на лету" при помощи директивы `REPLACE_TABLES`.

Глобальные переменные

Oracle допускает использование глобальных переменных в пакетах. Утилита ora2pgpro экспортирует эти переменные как переменные пакетов Postgres Pro.

Тестирование миграции

Тип экспорта `TEST` позволяет проверить, что все объекты из БД Oracle были созданы в Postgres Pro. Для проверки стороны Postgres Pro необходимо задать `PG_DSN`.

Обратите внимание, что эта функциональность учитывает ограничения имён схем, если заданы директивы `EXPORT_SCHEMA` и `SCHEMA` или `PG_SCHEMA`. Если задана только директива `EXPORT_SCHEMA`, сканируются все схемы в Oracle и Postgres Pro. Можно отфильтровать до одной схемы с помощью `SCHEMA` и/или `PG_SCHEMA`, но нельзя указать список схем. Чтобы протестировать список схем, вызывайте ora2pgpro несколько раз, отдельно указывая имя необходимой схемы по очереди.

```
ora2pgpro -t TEST -c config/ora2pgpro.conf > migration_diff.txt
```

Например, эта команда создаст файл с отчётом по всем объектам с подсчётом строк на обеих сторонах, Oracle и Postgres Pro, с отдельной секцией ошибок с описанием различий для каждого объекта. Пример результата:

```
[TEST INDEXES COUNT]
ORACLEDB:DEPARTMENTS:2
POSTGRES:departments:1
ORACLEDB:EMPLOYEES:6
POSTGRES:employees:6
[ERRORS INDEXES COUNT]
Table departments don't have the same number of indexes in Oracle (2)
 and in PostgreSQL (1).

[TEST UNIQUE CONSTRAINTS COUNT]
ORACLEDB:DEPARTMENTS:1
POSTGRES:departments:1
ORACLEDB:EMPLOYEES:1
POSTGRES:employees:1
[ERRORS UNIQUE CONSTRAINTS COUNT]
OK, Oracle and PostgreSQL have the same number of unique constraints.

[TEST PRIMARY KEYS COUNT]
ORACLEDB:DEPARTMENTS:1
POSTGRES:departments:1
ORACLEDB:EMPLOYEES:1
POSTGRES:employees:1
[ERRORS PRIMARY KEYS COUNT]
OK, Oracle and PostgreSQL have the same number of primary keys.

[TEST CHECK CONSTRAINTS COUNT]
ORACLEDB:DEPARTMENTS:1
POSTGRES:departments:1
ORACLEDB:EMPLOYEES:1
POSTGRES:employees:1
[ERRORS CHECK CONSTRAINTS COUNT]
OK, Oracle and PostgreSQL have the same number of check constraints.

[TEST NOT NULL CONSTRAINTS COUNT]
ORACLEDB:DEPARTMENTS:1
POSTGRES:departments:1
ORACLEDB:EMPLOYEES:1
POSTGRES:employees:1
[ERRORS NOT NULL CONSTRAINTS COUNT]
OK, Oracle and PostgreSQL have the same number of not null
 constraints.

[TEST COLUMN DEFAULT VALUE COUNT]
ORACLEDB:DEPARTMENTS:1
POSTGRES:departments:1
ORACLEDB:EMPLOYEES:1
POSTGRES:employees:1
[ERRORS COLUMN DEFAULT VALUE COUNT]
OK, Oracle and PostgreSQL have the same number of column default
 value.

[TEST IDENTITY COLUMN COUNT]
ORACLEDB:DEPARTMENTS:1
POSTGRES:departments:1
```

```
ORACLEDB:EMPLOYEES:0
POSTGRES:employees:0
[ERRORS IDENTITY COLUMN COUNT]
OK, Oracle and PostgreSQL have the same number of identity column.

[TEST FOREIGN KEYS COUNT]
ORACLEDB:DEPARTMENTS:0
POSTGRES:departments:0
ORACLEDB:EMPLOYEES:1
POSTGRES:employees:1
[ERRORS FOREIGN KEYS COUNT]
OK, Oracle and PostgreSQL have the same number of foreign keys.

[TEST TABLE COUNT]
ORACLEDB:TABLE:2
POSTGRES:TABLE:2
[ERRORS TABLE COUNT]
OK, Oracle and PostgreSQL have the same number of TABLE.

[TEST TABLE TRIGGERS COUNT]
ORACLEDB:DEPARTMENTS:0
POSTGRES:departments:0
ORACLEDB:EMPLOYEES:1
POSTGRES:employees:1
[ERRORS TABLE TRIGGERS COUNT]
OK, Oracle and PostgreSQL have the same number of table triggers.

[TEST TRIGGER COUNT]
ORACLEDB:TRIGGER:2
POSTGRES:TRIGGER:2
[ERRORS TRIGGER COUNT]
OK, Oracle and PostgreSQL have the same number of TRIGGER.

[TEST VIEW COUNT]
ORACLEDB:VIEW:1
POSTGRES:VIEW:1
[ERRORS VIEW COUNT]
OK, Oracle and PostgreSQL have the same number of VIEW.

[TEST MVIEW COUNT]
ORACLEDB:MVIEW:0
POSTGRES:MVIEW:0
[ERRORS MVIEW COUNT]
OK, Oracle and PostgreSQL have the same number of MVIEW.

[TEST SEQUENCE COUNT]
ORACLEDB:SEQUENCE:1
POSTGRES:SEQUENCE:0
[ERRORS SEQUENCE COUNT]
SEQUENCE does not have the same count in Oracle (1) and in PostgreSQL
 (0).

[TEST TYPE COUNT]
ORACLEDB:TYPE:1
```

```
POSTGRES:TYPE:0
[ERRORS TYPE COUNT]
TYPE does not have the same count in Oracle (1) and in PostgreSQL (0).

[TEST FDW COUNT]
ORACLEDB:FDW:0
POSTGRES:FDW:0
[ERRORS FDW COUNT]
OK, Oracle and PostgreSQL have the same number of FDW.

[TEST FUNCTION COUNT]
ORACLEDB:FUNCTION:3
POSTGRES:FUNCTION:3
[ERRORS FUNCTION COUNT]
OK, Oracle and PostgreSQL have the same number of functions.

[TEST SEQUENCE VALUES]
ORACLEDB:EMPLOYEES_NUM_SEQ:1285
POSTGRES:employees_num_seq:1285
[ERRORS SEQUENCE VALUES COUNT]
OK, Oracle and PostgreSQL have the same values for sequences

[TEST ROWS COUNT]
ORACLEDB:DEPARTMENTS:27
POSTGRES:departments:27
ORACLEDB:EMPLOYEES:854
POSTGRES:employees:854
[ERRORS ROWS COUNT]
OK, Oracle and PostgreSQL have the same number of rows.
```

Проверка корректности данных

Проверка корректности данных заключается в сопоставлении данных, полученных из сторонней таблицы, с указанием на исходную таблицу Oracle и локальную таблицу Postgres Pro, получаемую в результате экспорта.

Чтобы проверить корректность данных, можно подключаться напрямую, как делает ora2pgpro, но также можно использовать расширение oracle_fdw, причём нужно задать директивы `FDW_SERVER` и `PG_DSN`.

По умолчанию ora2pgpro извлекает первые 10000 строк с обеих сторон, это количество можно изменить, задав директиву `DATA_VALIDATION_ROWS`. Когда задано нулевое значение, сравниваться будут все строки таблиц.

Для проверки корректности данных требуется первичный ключ или уникальный индекс не по столбцу LOB. Сортировка строк выполняется по этому уникальному ключу. По причине различий в поведении Oracle и Postgres Pro, если в Postgres Pro не используется правило сортировки `C`, порядок сортировки может отличаться от Oracle. В этом случае проверка завершится ошибкой.

Проверку данных следует выполнять до изменения данных.

ora2pgpro прервёт сопоставление двух таблиц после достижения `DATA_VALIDATION_ROWS` или после 10 ошибок; результат выводится в файл `data_validation.log`, записываемый в текущий каталог по умолчанию. Число

ошибок, после которых останавливается проверка, можно задать в директиве DATA_VALIDATION_ERROR. Для удобства анализа все строки с ошибками выводятся в выходной файл.

Можно выполнять проверку корректности данных в несколько потоков, задав параметр -P или соответствующую директиву PARALLEL_TABLES в ora2pgpro.conf.

Использование системного номера изменения (System Change Number, SCN)

ora2pgpro может экспортировать данные по указанному SCN. Задать его можно в командной строке в параметре -S или --scn. Можно указать нужный SCN или текущий SCN при первом подключении (значение current). В этом случае у пользователя, подключающегося к БД, должна быть роль SELECT ANY DICTIONARY или SELECT_CATALOG_ROLE — поиск текущего SCN выполняется в представлении v$database.

```
ora2pgpro -c ora2pgpro.conf -t COPY --scn 16605281
```

При этом в запрос извлечения данных добавляется такое предложение:

```
AS OF SCN 16605281
```

Кроме того, в параметре --scn вместо SCN можно указать выражение с меткой времени, чтобы вернуться к определённой точке в прошлом.

```
ora2pgpro -c ora2pgpro.conf -t COPY --scn "TO_TIMESTAMP('2021-12-01
 00:00:00', 'YYYY-MM-DD HH:MI:SS')"
```

При этом в запрос извлечения данных добавляется такое предложение:

```
AS OF TIMESTAMP TO_TIMESTAMP('2021-12-01 00:00:00', 'YYYY-MM-DD
 HH:MI:SS')
```

Пример извлечения только данных предыдущих суток:

```
ora2pgpro -c ora2pgpro.conf -t COPY --scn "SYSDATE - 1"
```

Захват изменения данных (CDC, Change Data Capture)

ora2pgpro не допускает импорт данных с применением изменений после первого импорта, но указав параметр --cdc_ready, можно экспортировать данные со времени экспорта таблицы. Все SCN для таблиц по умолчанию записываются в файл TABLES_SCN.log, каталог хранения можно задать в параметре -C|--cdc_file.

Эти SCN для каждой таблицы, записанные во время экспорта COPY или INSERT, можно экспортировать с помощью утилиты CDC. Формат записей в файле: tablename:SCN.

Импорт BLOB как больших объектов

По умолчанию ora2pgpro импортирует объекты BLOB как bytea, и целевой столбец создаётся с типом bytea. Если вместо bytea нужно использовать большие объекты, добавьте в команду ora2pgpro параметр --blob_to_lo. При этом со-

здаётся целевой столбец типа oid и BLOB сохраняются как большие объекты с использованием функции `lo_from_bytea()`. OID, возвращаемый вызовом `lo_from_bytea()`, записывается в целевой столбец вместо типа bytea. Поскольку используется функция, параметр разрешается только с типами экспорта `SHOW_COLUMN`, `TABLE` и `INSERT`. С типом экспорта `COPY` использовать этот параметр не допускается.

Если используется тип экспорта `COPY` или есть огромные BLOB (> 1ГБ), которые невозможно импортировать функцией `lo_from_bytea()`, в команду ora2pgpro можно добавить параметр `--lo_import`. Тогда данные можно будет импортировать в два этапа.

1. Сначала экспортируйте данные с типом экспорта `COPY` или `INSERT`, в процессе чего в целевом столбце OID для BLOB проставляются 0, а значение BLOB сохраняется в отдельный файл. Также создаётся скрипт оболочки для импорта файлов BLOB в базу данных с помощью команды psql `\lo_import` и обновления столбца OID возвращаемыми значениями OID. Скрипт называется `lo_import-TABLENAME.sh`.

2. Затем задайте переменную окружения `PGDATABASE` и, возможно, `PGHOST`, `PGPORT`, `PGUSER` и т. д., если их значения отличаются от значений по умолчанию в libpq, и выполните все скрипты `lo_import-TABLENAME.sh`.

Кроме того, можно вручную выполнить `VACUUM FULL` для таблицы, чтобы устранить раздувание, вызванное изменением таблицы.

Примечание

Ограничение: в таблице должен быть первичный ключ, он используется в предложениях `WHERE` для обновления столбца OID после импорта большого объекта. Импорт BLOB вторым способом (`--lo_import`) очень медленный, поэтому его лучше оставить для строк с BLOB > 1ГБ, а для других использовать `--blob_to_lo`. Чтобы отфильтровать строки, можно задать директиву `WHERE` в `ora2pgpro.conf`.