PG BootCamp Russia 2025 Ekaterinburg

PGBootCamp.ru

Изменение структуры таблиц в PostgreSQL в online с помощью DBMS_REDEFINITION

Игорь Мельников

melnikov_ii@mail.ru

Игорь Мельников

2006-2022: Oracle СНГ: главный консультант

2022-2024: Postgres Pro: ведущий консультант

2024 – настоящее время: независимый консультант

Сферы деятельности: внедрение Extended RAC,, апгрейд версии СУБД Oracle, уменьшение размера БД с помощью Advanced Compression, оптимизация кода PL/SQL, миграция с RISC-платформ на Linux x86 и т.д.

Также в последнее время занимаюсь проектами по миграции с Oracle на PostgreSQL, апгрейдом версии PostgreSQL и технической поддержкой СУБД Oracle и PostgreSQL.

План

Задача переопределения таблиц на “лету” (с минимальным простоем)

Технология DBMS_REDEFINITION в СУБД PostgreSQL

Пример использования “пакета” на примере реальной задачи

Планы по развитию

Переопределение структуры таблиц с минимальным простоем

Большой downtime приложения на изменение структуры таблиц

История реального заказчика

- СУБД PostgreSQL 9.5.25, размер БД ~1ТБ, максимальное время простоя – 15 минут
- Задача: Необходимо обновить до PostgreSQL 15

- Проблема: часть таблиц имеют поля типа OID, которые используются приложением; утилита pg_upgrade выдает ошибку

```
Your installation contains tables declared WITH OIDS, which is not
supported anymore.  Consider removing the oid column using
    ALTER TABLE ... SET WITHOUT OIDS;
A list of tables with the problem is in the file:
    tables_with_oids.txt
```

- Рекомендуемое решение “зависает” на несколько часов:

```
ALTER TABLE has_oids ADD newoid bigint DEFAULT 0;
UPDATE has_oids SET newoid = oid;
ALTER TABLE has_oids SET WITHOUT OIDS;
…   …   …
--производим апгрейд
```

```
ALTER TABLE has_oids RENAME newoid TO oid;
CREATE SEQUENCE has_oids_oid_seq OWNED BY
has_oids.oid;
ALTER TABLE has_oids ALTER oid SET DEFAULT
nextval('has_oids_oid_seq');
SELECT setval('has_oids_oid_seq', ???);
```

Если необходимо изменить структуру таблицы в PostgreSQL

Длительный downtime приложения для больших таблиц

DDL-операции устанавливают на таблицу эксклюзивную или разделяемую блокировку

- до PostgreSQL 11 операция добавления столбца с значением по умолчанию блокирует таблицу на изменения:

```
ALTER TABLE customers ADD COLUMN x text DEFAULT 'xx';
```

- Перемещение таблицы в другое табличное пространство полностью блокирует таблицу (операции чтения и записи невозможны):

```
ALTER TABLE customers SET TABLESPACE arch_data;
```

- Перевод несекционированной таблицы в секционированную, для согласованности должен выполняться с блокировкой на запись (DML запрещен)

```
LOCK TABLE customers IN SHARE MODE;
```

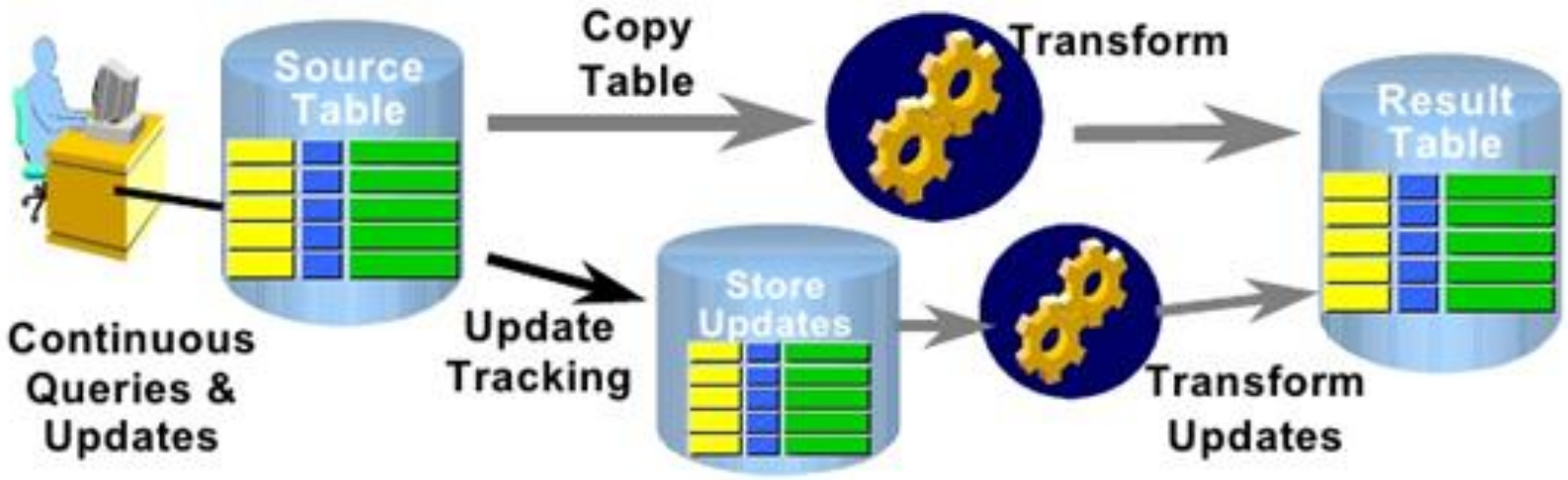
- DML-операции блокируют записи для изменения для обеспечения согласованности
  - Установка значений для новых столбцов блокирует таблицу на запись

```
UPDATE customers SET city=get_city(address_str), street=get_street(address_str), ...
```

Необходима технология для переопределения структуры таблиц в online

для больших баз данных, где важна непрерывность работы

- Необходима технология для изменения структуры таблиц “на лету” – без простоя, или с минимальным простоем (околонулевым простоем)
- Без необходимости каждый раз разрабатывать и отлаживать SQL-скрипты



Технология DBMS_REDEFINITION для изменения структуры таблиц в Online

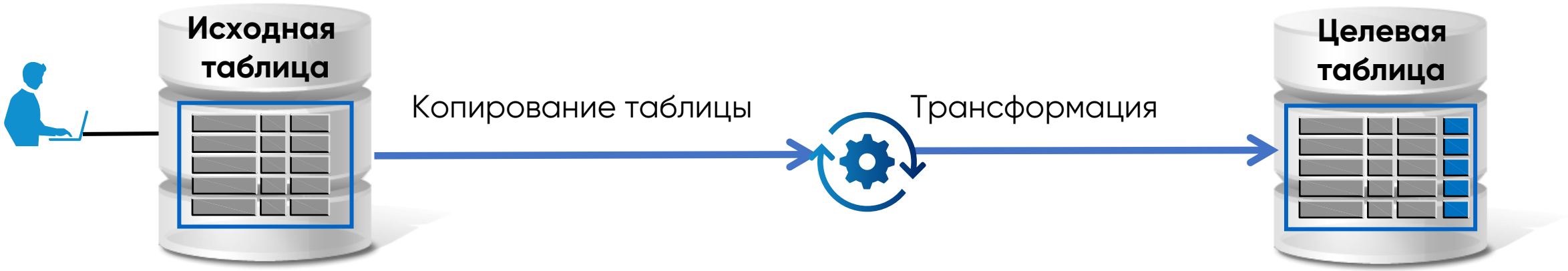
Что такое пакет DBMS_REDEFINITION в PostgreSQL

Набор функций и таблиц в схеме DBMS_REDEFINITION

- Состав и название функций и их сигнатура совпадает с пакетом DBMS_REDEFINITION в СУБД Oracle Database
- Разработан на чистом PL/pgSQL – объем кода ~2 тыс. строк
- Поддерживаются “ванильный” PostgreSQL версии 9.5 и выше

“Пакет” (схема) DBMS_REDEFINITION

1. Создание новой таблицы и начальное копирование



Непрерывный поток
запросов и DML-операций

Таблица доступа для чтения и записи в других сессиях

- С началом копирования, запускается захват изменений в таблице

1. Создаем новую таблицу без столбца OID

Сначала проверяем возможность переопределения “на лету”

- Проверяем возможность изменения структуры таблицы на лету – нужен Primary Key

```
DO
$$
BEGIN
   PERFORM dbms_redefinition.can_redef_table(uname => 'SCOTT',
                                             tname => 'HAS_OIDS');
END;
$$; Status: Ok (PK)
```

- Создаем новую таблицу без OID, с полем newoid

```
CREATE TABLE scott.without_oids
(
    …    …    …
   status   CHAR(3),
   newoid   BIGINT NOT NULL
) WITHOUT OIDS;
```

• Новое поле на замену поля OID

2. Начинаем переопределение

Запускаем начальное копирование в целевую таблицу

- Создание mlog-таблицы, DML-триггеров на исходной таблице, запуск первой синхронизации (Initial Load) – занимает длительное время:

```
SELECT dbms_redefinition.start_redef_table(
      uname        => 'scott',
      orig_table   => 'has_oids',
      int_table    => 'without_oids',
      col_mapping  => '..., status, oid newoid'),
      degree       => 4);
```

• Соответствие полей: “старое новое”

- Проверяем статус первой синхронизации:

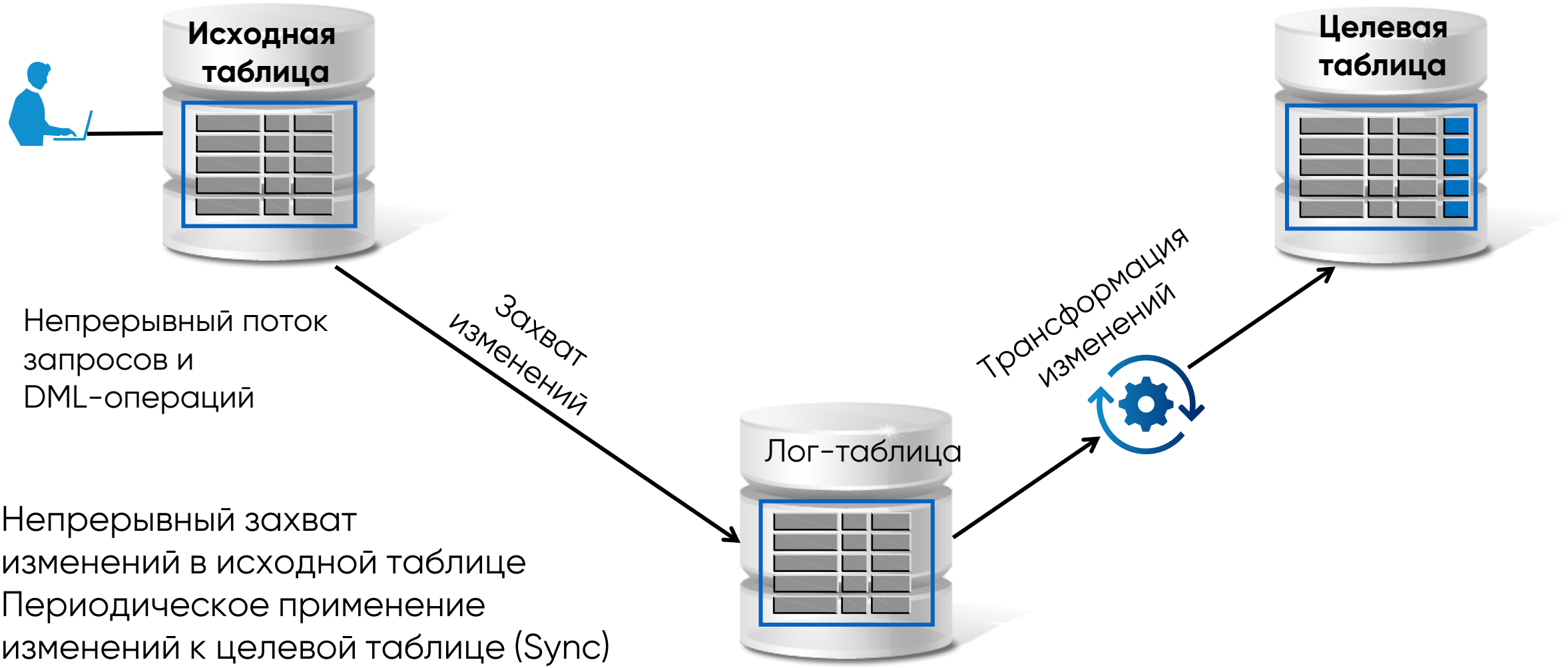
```
SELECT * FROM dbms_redefinition.redef_status;
```

• Степень параллелизма для pg_cron/pg_task

- **Запросы и DML над исходной таблице доступен – downtime НЕТ!**

Синхронизация дельты изменений

2. Захват дельты изменений и применение их к целевой таблице



- Непрерывный захват изменений в исходной таблице
- Периодическое применение изменений к целевой таблице (Sync)

3. Несколько раз запускаем инкрементальную синхронизацию

Дельта изменений между таблицами становится минимальной

- Запускаем инкрементальную синхронизацию (Incremental Load):

```
SELECT dbms_redefinition.sync_interim_table(
          uname      => 'scott',
          orig_table => 'has_oids',
          int_table  => 'without_oids');
```

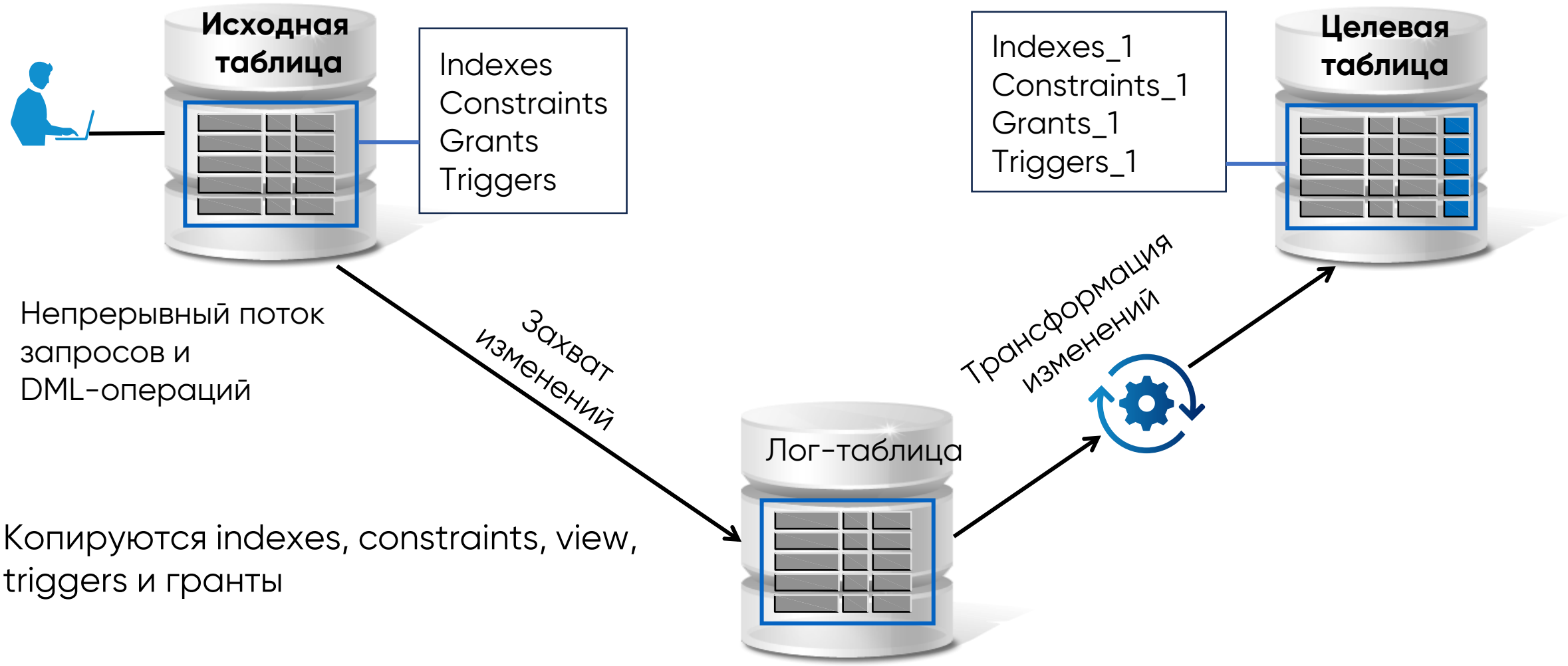
- Проверяем статус синхронизации:

```
SELECT * FROM dbms_redefinition.redef_status;
```

- **Запросы и DML над исходной таблице доступен – downtime НЕТ!**

Пакет DBMS_REDEFINITION в СУБД PostgreSQL

3. Копирование зависимых объектов



- Копируются indexes, constraints, view, triggers и гранты

4. Копируем зависимые объекты

Копируем индексы, триггеры, констрейнты, гранты

```
DO $$
DECLARE
      l_num_errors numeric;
BEGIN
  PERFORM dbms_redefinition.copy_table_dependents(
            uname              => 'scott',
            orig_table         => 'has_oids',
            int_table          => 'without_oids',
            copy_indexes       => True,
            copy_triggers      => True,
            copy_constraints => True,
            copy_privileges    => True,
            ignore_errors      => False,
            num_errors         => l_num_errors);

      RAISE NOTICE 'l_num_errors=%',l_num_errors;
END; $$;
```

- **Запросы и DML над исходной таблице доступен – downtime НЕТ!**

Пакет DBMS_REDEFINITION в СУБД PostgreSQL

4. Короткая блокировка исходной таблицы и переименование таблиц

Экслюзивная блокировка таблицы
(короткий DOWNTIME приложения)

- Применение последней дельты изменений
- Исходная таблица переименовывается в имя целевой таблицы
- Зависимые объекты в целевой таблице переименовываются в оригинальные имена
- Целевая таблица переименовывается в имя исходной таблицы

5. Делаем последнюю инкрементальную синхронизацию и переключаемся на новую таблицу

```
SELECT dbms_redefinition.sync_interim_table(
          uname      => 'scott',
          orig_table => 'has_oids',
          int_table  => 'without_oids');
```

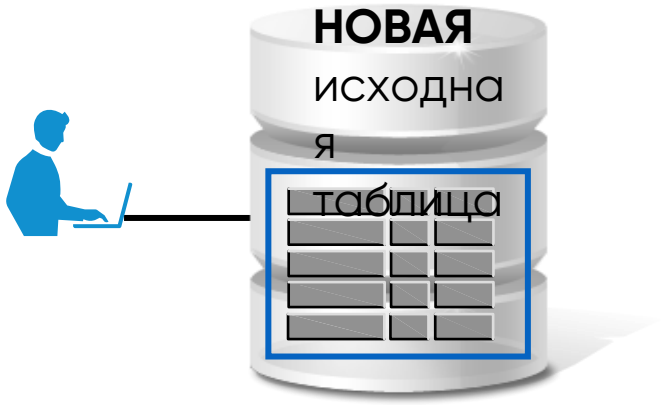
- Завершение процесса (переименование таблиц и их зависимых объектов) - **DOWNTIME**:

```
SELECT dbms_redefinition.finish_redef_table(
            uname      => 'scott',
            orig_table => 'has',
            int_table  => 'without_oids');
```

- Простой на *finish_redef_table* составил 10 секунд!

Пакет DBMS_REDEFINITION в СУБД PostgreSQL

5. Приложение работает с новой (реорганизованной) таблицей



Непрерывный поток запросов
и DML-операций

- Блокировка снимается
- Старая исходная таблица и лог-таблица удаляются

Функции в схеме DBMS_REDEFINITION

- Предоставляет API для переопределения таблиц с минимальным простоем

Функция	Назначение
can_redef_table	Проверка возможности переопределения таблицы – нужен PK
start_redef_table	Старт процесса переопределения таблицы, запуск полной синхронизации : (копирование) данных из старой в новую таблицу
register_dependent_object	Регистрация переносимых вручную объектов (опционально)
sync_interim_table	Инкрементальная синхронизация (копирование) данных из старой в новую таблицу
copy_table_dependents	Создание/копирование для новой таблицы объектов зависимых от старой (индексы, триггеры, констрейнты, гранты, mview)
finish_redef_table	Завершение процесса переопределения (переименование таблиц)
abort_redef_table	Отмена процесса переопределения (удаление созданных процедурой "copy_table_dependents" объектов)
uregister_dependent_object	Дерегистрация переносимых вручную объектов (опционально)

DBMS_REDEFINITION для PostgreSQL

Ограничения и проблемы

- Переопределяемая таблица должна иметь первичный ключ (Primary Key), не поддерживается изменение Primary Key в строке таблицы
- Если на исходную таблицу ссылалась другая таблица по FK, в момент переключения FK перевалидируется – это может занять значительное время (вынести в отдельный параметр?)
- Не поддерживается Table Rule
- В момент переключения на новую таблицу, исходная таблица НЕ доступна (Exclusive Lock)
- Материализованные представления после переключения нужно перестраивать вручную
- Необходимо свободное пространство на диске равное объему переопределяемой таблицы и ее индексов
- Начальное копирование занимает длительное время и нагружает СХД – рекомендуется запускать его в период минимальное нагрузки
- Копирование зависимостей может занимать длительное время: создание индексов и FK

DBMS_REDEFINITION для PostgreSQL

Заключение

- Позволяет реализовать переопределение таблиц с околонулевым простоем
- API полностью повторяет функционал одноименного пакета из СУБД Oracle Database
- Полностью автоматизирует процесс реорганизации таблицы на лету – без необходимости разработки сложных SQL-скриптов “перелива” данных
- В процессе реорганизации таблицы сохраняются все зависимые объекты: indexes, constraints, view и гранты
- Есть планы по развитию продукта

Планы по развитию

DBMS_REDEFINITION для PostgreSQL

Планы по развитию

- Выложить на github
- Разработать документацию
- Разработать примеры
- Разработать полный набор автотестов
- Сделать упрощение для удаления OID-поля
- Сделать упрощение и оптимизацию для перевода НЕсекционированной таблицы в секционированную в ONLINE

PG BootCamp Russia 2025 Ekaterinburg

PGBootCamp.ru

Спасибо за внимание