УДК 004.65 DOI: 10.17587/prin.13.32-40

Е. А. Курако, канд. техн. наук, ст. науч. сотр., kea@ipu.ru,
В. Л. Орлов, канд. техн. наук, вед. науч. сотр., ovl@ipu.ru,
Институт проблем управления им. В. А. Трапезникова РАН, Москва

К вопросу миграции баз данных из среды Oracle в среду PostgreSQL

Рассмотрены методы переноса данных из среды Oracle в среду PostgreSQL с учетом поэтапной организации действий и определения последовательности миграции. Выделены основные направления для преобразования структуры баз, форматов данных и выполняемых объектов. Определены методы преобразования хранимых процедур и триггеров, написанных на процедурных языках, в качестве которых используются PL/SQL и PL/pgSQL. Рассмотрены вопросы изменения программного обеспечения на узлах, непосредственно связанных (сопряженных) с сервером базы данных и вопросы минимизации остановок в процессе миграции, что дает возможность проводить перенос для географически распределенных и круглосуточно функционирующих объектов. Приведены примеры изменения скорости выполнения запросов до и после миграции для систем различных типов.

Ключевые слова: *база данных, БД, СУБД, Oracle, PostgreSQL, миграция, структура, данные, функции, хранимые процедуры, PL/SQL, PL/pgSQL*

Введение

Под миграцией баз данных (БД) подразумевается перевод той или иной БД, работающей под управлением определенной системы управления базами данных (СУБД), на функционирование под управлением другой СУБД. Отметим, что СУБД взаимодействует с другими компьютерами информационной системы, образующими сложную сеть. Изменение узла с БД при этом потребует изменения узлов, связанных с ним.

Если раньше необходимость смены СУБД была практически не востребована, то сейчас миграция в ряде случаев становится просто необходимой. Эта необходимость в основном вызывается перечисленными далее обстоятельствами.

- Информация в базах данных накапливается в течение нескольких лет, а иногда и десятилетий. Казалось бы, при столь долгой эксплуатации программное обеспечение (ПО), обслуживающее хранимые и корректируемые данные, не должно изменяться. Однако ценность накапливаемых данных растет, а к процессу управления этими данными возникают дополнительные требования. Иногда эти требования приводят к необходимости замены СУБД.
- Часто возникают не чисто технические проблемы, а обстоятельства, связанные с обеспечением требований безопасности. Это происходит в тех случаях, когда применяемая СУБД — закрытая, доступ к ее исходным кодам практически отсутствует и возникает подозрение о возможности использования поставщиком закладок и других недокументированных средств, позволяющих обойти декларированные средства защиты. В этом случае также требуется замена СУБД при сохранении содержимого БД.
- Существует также вопрос эксплуатационной стоимости, которая обычно включает не только стоимость лицензии, но и затраты на техническое сопровождение. Здесь преимущество получают открытые системы, которые распространяются бесплатно, а стоимость технического сопровождения существенно ниже. В этом случае появляется перспектива существенной экономии средств.

Вместе с тем перенос системы под управление другой СУБД в сложных сетях является далеко не тривиальной задачей. Это определяется тем обстоятельством, что СУБД имеют описанные далее особенности.

- Каждая СУБД имеет свою структуру данных. Нужно иметь в виду, что, хоть эти структуры и близки, но их описание различается, а значит скрипты, по которым формируется база данных, будут различными.
- Перенос и преобразование данных проводятся в соответствии с учетом согласования структур.
- В развитых БД неотъемлемой частью является также использование хранимых процедур (функций), которые содержат написанный на определенном языке исходный код, выполняемый при соответствующих вызовах. Следует иметь в виду, что в этом случае возможна разница не только в синтаксисе, но и в семантике, так как здесь обрабатываются различные структуры различными методами.
- Кроме того, в различных СУБД имеются различные дополнительные параметры, обработка которых при перенесении данных должна проводиться с учетом особенностей этих параметров.

Основные способы миграции баз данных из среды СУБД Oracle в среду СУБД PostgreSQL

Рассмотрим способы миграции БД из среды СУБД Oracle в среду СУБД PostgreSQL. Это преобразование вызывает особый интерес, так как СУБД этих типов могут работать с большими массивами данных, что важно для современных систем. Кроме того, одна из СУБД — коммерческая (Oracle), другая — открытая (PostgreSQL), что предпочтительнее с точки зрения анализа безопасности, так как в последнем случае всегда доступны исходные тексты. Следует отметить, что и та и другая СУБД включают в свой состав языки для написания хранимых процедур PL/SQL и PL/pgSQL соответственно, что существенно для построения эффективных структур работы с БД.

Заметим, что процедуру миграции в автоматическом режиме полностью провести невозможно в силу сложности задачи и необходимости учета смысловых особенностей алгоритма при переносе хранимых процедур. Вместе с тем существует довольно большое число коммерческих и некоммерческих инструментальных средств [1], которые позволяют облегчить этот процесс (табл. 1).

Как известно, БД непосредственно включает следующие составляющие: описание структуры, данные, выполняемые объекты (рис. 1).

Все приведенные средства обеспечивают, по крайней мере, преобразование и перенос одной составляющей — структуры БД. Большинство проводит анализ структуры, принадлежащей БД Oracle, готовит на основе этого анализа семейство скриптов, позволяющее создать новую структуру, но уже формата PostgreSQL. Таким образом, результатом является новая структура, созданная с учетом особенностей PostgreSQL, включающая таблицы, представления, ключи, связи и другие объекты. Вместе с тем созданная к этому времени база является пустой.

Создание новой структуры возможно, например, с использованием таких инструментальных срдеств, как Enterprise DB Migration Toolkit, Full Convert, Ora2Pg. Кроме того, для компактного преобразования структуры в Институте проблем управления Российской академии наук (ИПУ РАН) разработана отечественная программа "Транслятор структур Oracle—PostgreSQL (OracleToPostgres_Scheme)" [2]. Особенность этого ПО заключается в том, что в качестве исходного материала берется скрипт, описывающий создание базы данных в Oracle. Этот скрипт или формируется изначально при проектировании БД и существует к моменту преобразования, например, созданный таким средством, как ErWin. Или такой скрипт может быть восстановлен из реальной БД с использованием сторонних инструментов, таких как, например, dbForge for Oracle. Результатом работы программы OracleToPostgres_Scheme является также скрипт для создания структуры БД, но уже представленный в формате PostgreSQL.

Отметим, что помимо множества необходимых преобразований структуры (описаний таблиц, связей, представлений), основным является преобразование типов данных, некоторые из которых приведены в табл. 2.

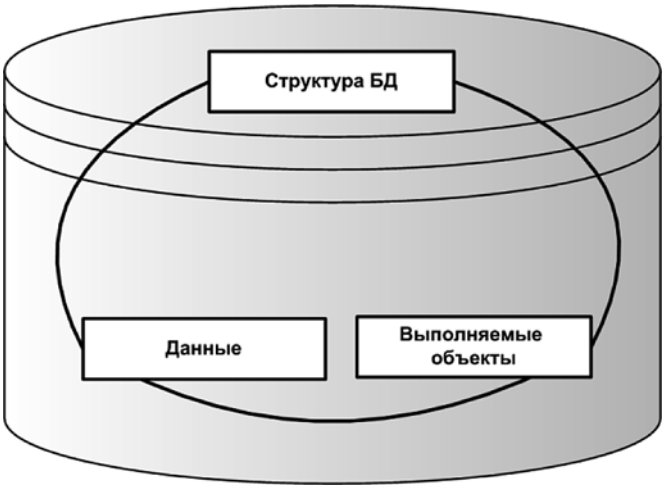

Рис. 1. Основные составляющие БД

Таблица 1

Основные инструментальные средства для миграции БД

Инструментальное средство	Коммерческое/ некоммерческое	Основные функции	Недостатки
Enterprise DB Migration Toolkit	Коммерческое	Переносит структуру и данные	Не переносит хранимые процедуры
Oracle Golden Gate	Коммерческое	Обеспечивает репликацию БД, осуществляя перемещение в базы другого вида, в том числе и PostgreSQL	Высокая стоимость. Работает только с данными
SymmetricDS	Некоммерческое	Обеспечивает репликацию БД, перенося данные, в частном случае, в PostgreSQL	Работает только с данными. Есть ограничения при использовании полей типа BLOB
Full Convert	Коммерческое	Переносит структуру и данные. Работает с различными СУБД.	Минимальная настройка
Ora2Pg	Некоммерческое	Переносит структуру, данные. Проект активно развивается, надежен	Усложненная настройка

Таблица 2

Основные типы данных для перевода

Основные типы данных Oracle	Соответствующие типы данных PostgreSQL	Примечание
BLOB	BYTEA	⩽ 4 ГБ
CHAR	CHAR	$1 \leqslant n \leqslant 2000$
CLOB	TEXT	⩽ 4 ГБ
DATE	TIMESTAMP	Дата и время
TIMESTAMP	TIMESTAMP	—
DECIMAL	DECIMAL	—
FLOAT	DOUBLE PRECISION	—
LONG	TEXT	⩽ 2 ГБ
NUMBER	INTEGER	—
VARCHAR	VARCHAR	$1 \leqslant n \leqslant 4000$
VARCHAR2	VARCHAR	$1 \leqslant n \leqslant 4000$
XMLTYPE	XML	XML data

Вторая стадия миграции обеспечивает перенос реальных данных из базы данных Oracle в БД PostgreSQL. Для этого может использоваться Ora2Pg или специально для этого разработанная отечественная программа "Трансфер данных Oracle—PostgreSQL (OracleToPostgres_Data)" [3].

Для ускорения переноса данных программа OracleToPostgres_Data временно устраняет все связи в структуре БД, созданной на первой стадии миграции. После этого проводится перенос потаблично всех данных. При этом данные читаются из таблиц Oracle и записываются в таблицы PostgreSQL. Когда перемещение данных завершено, то все связи восстанавливаются.

На этом этапе также важно провести анализ последовательностей, использующихся в Oracle, и в процессе переноса завести соответствующие последовательности в PostgreSQL, установив их текущие значения на момент миграции из первичной базы.

На втором этапе миграция баз данных могла бы быть завершена. Действительно — создана новая база данных, созданы таблицы, индексы, последовательности, установлены все необходимые связи, перенесены все данные. Однако существуют по крайней мере несколько существенных причин, по которым работу нельзя признать законченной. Это определяется тем обстоятельством, что в современных БД хранятся не только статические данные, но и выполняемые объекты, по существу представляющие собой программы для обработки данных. Например, к ним относятся хранимые процедуры, триггеры, типы, представления и материализованные представления.

Перенос выполняемых объектов

Хранимые процедуры и триггеры обычно пишутся на процедурном языке, в качестве которого в Oracle используется PL/SQL, а в PostgreSQL — язык PL/pgSQL [4]. Эти языки весьма близки, в то же время они имеют существенные отличия, которые исключают простой формальный перевод программ. В большинстве случаев тело функции содержит выражения на языке SQL, которые также могут иметь разный синтаксис в различных СУБД.

Прежде всего, нужно иметь в виду, что процедуры Oracle компонуются в пакеты (*Packages*). Эта возможность отсутствует в PostgreSQL, что вызывает и будет еще вызывать определенные споры среди разработчиков, однако факт остается фактом. В процессе переноса все функции и процедуры Oracle, входящие в определенный пакет, принято объединять в схему PostgreSQL с названием, идентичным названию исходного пакета. Но вместе с пакетами исчезают и пакетные переменные, а это изменяет логику программы.

Отметим, что в PostgreSQL применяются, как правило, функции. Если требуется перевести какую-либо процедуру из Oracle, то она тоже оформляется как функция. При этом все функции вызываются из других функций или внешних программ. Триггеры выполняются как реакция на то или иное событие.

Начинать перенос целесообразно с поиска используемых ключевых слов. Списки слов есть на официальных сайтах, рассматриваемых СУБД в разделе "Документация". Например, в СУБД PostgreSQL в качестве ключевого слова используется `current _ date`, которое заменяет ключевое слово `sysdate`, используемое в Oracle. В Oracle есть функция с одноименным названием, но СУБД позволяет использовать `current _ date` в качестве столбца. И будет корректен такой SQL-запрос:

```
insert into voc _ table (first _ f, current _ date)
values (1, sysdate);
```

В PostgreSQL это выражение вызовет ошибку. В то же время, использование ключевого слова `sysdate` легко обнаруживается анализатором PostgreSQL. Как уже отмечалось выше, языки PL/SQL и PL/pgSQL очень похожи. Рассмотрим фрагмент заголовка типовой функции для Oracle:

```
function s _ main.get _ doc _ list _ count(p _ num
in integer,
            p _ type  in  varchar2)  return
            integer is
```

И заголовок аналогичной функции для PostgreSQL:

```
create or replace function s _ main.get _ doc _
list _ count(p _ num in integer,
          p _ type in varchar) returns integer
language plpgsql
as $function$
```

Как можно видеть, из первого заголовка достаточно легко получить второй. Сложности начинаются, когда требуется возвратить несколько параметров. Пример для Oracle:

```
function get_filename(p_num in integer,
                      p_type in varchar2,
                      p_path out varchar2,
                      p_filename out varchar2)
                      return integer is
```

Который будет преобразован в:

```
create or replace function get_filename(p_
num in integer,
                    p_type in varchar,
                    p_path out varchar,
                    p_file out varchar,
                    p_return_code  out
                    integer) returns record
language plpgsql
as $function$
```

Потребовалось добавить один возвращаемый параметр типа `integer` и результат функции преобразовать в составной тип `record`, так как возвращаются разные типы в параметрах. В предыдущем примере возвращаемый тип был один.

Основную сложность вызывает использование в заголовках в качестве параметров специальных или пользовательских типов. Программам перевода потребуется помощь программиста, например, в определении уровня доступа пользовательского типа. Так в Oracle есть различные уровни доступа к пользовательскому типу, в числе которых доступ на уровне пакета функций (package). Значит, два пакета могут иметь одно и то же название типа с различным описанием. При автоматическом переводе потребуется уточнить, что надо сделать с этими типами: объединить, использовать только первый, переименовать второй и т. д.

При изменении исходного кода даже в простых вещах возможны ошибки или семантические неопределенности при конвертации. Рассмотрим исходный код для Oracle:

```
if (p_value is null) then ...
```

В этом выражении идет проверка на пустое значение. Если переменная `p_value` равна `null` или `p_value` равна `''`, то в Oracle условие выполнится. В PostgreSQL данное условие выполнится, если только `p_value` равна `null`. Таким образом, эквивалентное выражение в PostgreSQL:

```
if ((p_value is null) or (p_value = '')) then ...
```

Однако следует учитывать логику программы, в которой может потребоваться вариант проверки переменой на строгое равенство значению `null`, т. е. необходимо вмешательство программиста.

Рассмотрим второй пример с объединением строк:

```
'value='||p_value
```

В Oracle, если переменная `p_value` равна `null`, то результат будет `'value='`. В PostgreSQL ситуация другая, результат будет `null`. Для получения одинакового результата подходит выражение

```
concat('value=', p_value)
```

Здесь следует обратить внимание программиста на данное выражение для уточнения логики выполнения.

Работа с исключениями отличается мало и хорошо автоматически переводится. Пример для Oracle:

```
begin
  INSERT INTO voc_table VALUES (1, 'red');
exception
when others then
  raise_application_error(-20001,'Не удалось записать в БД '|| sqlerrm)
end;
```

Для PostgreSQL:

```
begin
  INSERT INTO voc_table VALUES (1, 'red');
exception
  when others then
    raise exception 'Не удалось записать в БД: %',sqlerrm;
  end;
```

Как видно, в данных примерах отличается лишь вызов исключения. Однако в случае использования пользовательских исключений опять придется использовать труд программиста, так как в PostgreSQL нет пользовательских исключений.

Наиболее важное отличие заключается в том, что нет управления транзакциями внутри хранимых функций в PostgreSQL. Транзакции возможны только во внешнем коде, на компьютерах, которые непосредственно работают с узлом БД, которые к тому же называются сопряженными. В случае использования внутри функции Oracle блока транзакции, такую функцию необходимо будет переписать вручную, разделив на несколько функций, каждая из которых отвечает за свой блок транзакции. Соответственно, вызывать ее придется во внешнем коде.

При преобразовании SQL-выражений в случае простых запросов могут потребоваться лишь небольшие правки, или можно будет обойтись без правок. Например, в PostgreSQL не поддерживаются сокращенные наименования (алиасы) в выражениях `INSERT` и `UPDATE` (в блоке изменяемых столбцов), который легко правится автоматически.

```
UPDATE voc_table vt SET vt.first_f = 1 WHERE
vt.second_f = 1;
```

Указанное выражение будет работать в Oracle, но выдаст ошибку в PostgreSQL. Потребуется убрать алиас `"vt."` из секции `SET`:

```
UPDATE voc _ table vt SET first _ f = 1 WHERE
vt.second _ f = 1;
```

Еще пример простого запроса, который можно легко изменить, связан с ключевым словом rownum:

```
SELECT rownum as rownum, first _ f as id
FROM voc _ table;
```

В БД PostgreSQL это будет выглядеть так:

```
SELECT row _ number() over () as rownum,
first _ f as id FROM voc _ table;
```

В случае сложных запросов, например, рекурсивных, потребуется полностью вручную переписать сам запрос, так как PostgreSQL не поддерживает "connect by". То есть запрос вида

```
SELECT iie.e _ type, iie.name FROM item _
in _ folder iie,
    (SELECT level lv,il.* FROM item _ location il
    START WITH (il.i _ num = 111 AND il.i _
type = 'D')
    CONNECT BY PRIOR il.folder _ num = il.i _ num
AND PRIOR il.folder _ type = il.i _ type) rrr
WHERE iie.i _ num = rrr.i _ num AND iie.i _
type = rrr.i _ type
ORDER BY rrr.lv;
```

должен быть преобразован в

```
WITH RECURSIVE rrr AS (
  SELECT 1 as level, il.* FROM item _
location il
  WHERE il.i _ num = 111 AND il.i _ type = 'D' AND
  UNION
SELECT rrr.level + 1, lev.* as level FROM
item _ location lev
JOIN rrr ON rrr.folder _ num = lev.i _ num
AND rrr.folder _ type = lev.i _ type)
SELECT iie.element _ type, iie.name FROM
rrr, item _ in _ folder iie
WHERE iie.i _ num = rrr.i _ num AND iie.i _
type = rrr.i _ type
ORDER BY rrr.level)
```

Хотелось бы также отметить, что использование встроенных функций в Oracle очень полезно, однако автоматический перевод их затруднителен. Понятно, что аналог каждой функции можно написать для PostgreSQL, но их очень много. Как следствие, обычно при переводе исходников создают только те функции, которые реально используются.

Также в силу того, что PostgreSQL не транслирует исходный код при сохранении в БД, возникают ситуации, когда функция при очередном выполнении выдает ошибку. Значит, написан исходный код функции, где есть ветвления. Вследствие особенностей использования функции, есть одна ветка ветвления, которая используется очень редко. В этой ветке есть, например, ссылка на несуществующий столбец. Тогда обычно программа будет работать правильно, но когда-нибудь внезапно будет появляться неприятная ошибка.

Что же касается представлений и материализованных представлений, то это, как известно, результат выполнения SQL-запроса. Преобразование такого SQL-выражения выполняется таким же образом, как и в хранимых процедурах.

Таким образом, перевод исходных кодов — сложный и кропотливый процесс, который требует как большого количества рутинной работы, так и достаточного опыта разработки для СУБД Oracle и PostgreSQL. Отечественная программа (ИПУ РАН) "Предконвертер хранимых процедур Oracle—PostgreSQL (OracleToPostgres_Procedure)" [5] часть исходного кода, которая относится к типовой, переводит самостоятельно, а остальную часть пытается распознать и дать подсказки.

Полученный в результате миграции код программа OracleToPostgres_Procedure проверяет на известные ей ошибки.

Изменение программного обеспечения на сопряженных узлах

Миграция БД не исчерпывается преобразованием структуры, модернизацией и переносом данных, преобразованием хранимых процедур (функций) и триггеров.

Также требуется изменение ПО, которое функционирует на узлах, непосредственно связанных (сопряженных) с узлом БД (рис. 2). Это связано с двумя обстоятельствами. Во-первых, для вызова из внешнего ПО базы данных в зависимости от типа БД используется своя процедура вызова. Таким образом, требуется заменить все точки вызова из ПО. Во-вторых, необходимо учитывать, что основная часть работы с БД может быть сосредоточена либо в хранимых процедурах (в теле БД), либо во внешнем ПО, где формируются внешние SQL-запросы и направляются в БД для выполнения. В последнем случае большинство процедур для работы с базой данных будут размещены вне БД. Более того, такие процедуры могут формироваться динамически. Здесь нужно иметь в виду две особенности. С одной стороны, для подготовки фрагментов для манипуляции с данными в БД используется SQL-язык и в процессе преобразования Oracle—PostgreSQL нужно менять только его диалекты. С другой стороны, обработка полученных из БД данных проводится на одном из языков программирования (а может и нескольких), причем фрагменты подготовки запросов к БД и другие информационные фрагменты, как правило, не отделены друг от друга. То есть при переносе нужно учитывать достаточно широкий круг обработки, что требует тщательных проверок. Конечно, если запросы изначально проектировались так, что подразумевалась возможность последующей миграции, то это меняет дело. Но, как правило, при написании ПО в первую очередь код оптимизируется для быстродействия в конкретной СУБД. Необходимо отметить, что есть два способа работы с БД:

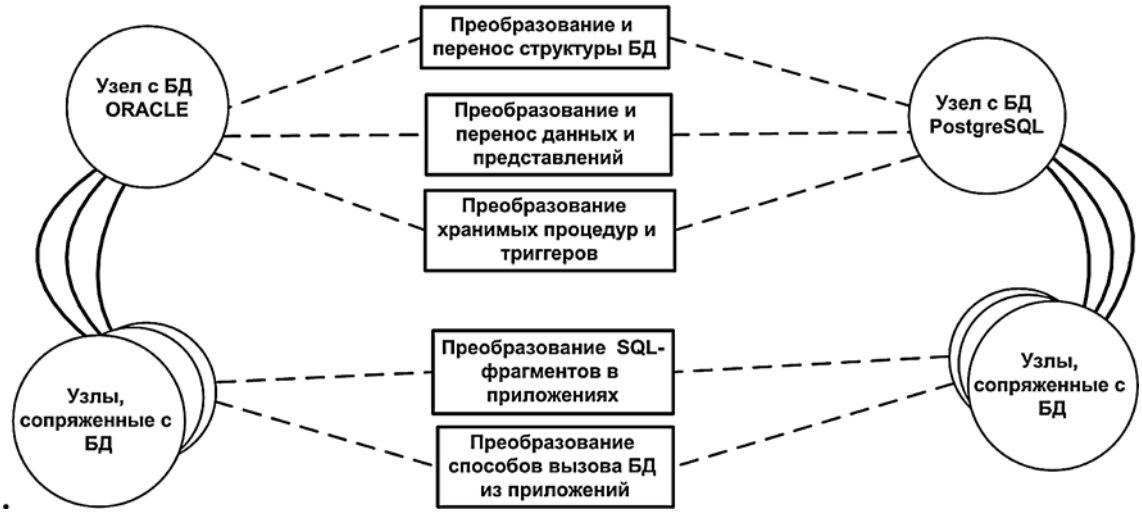

Рис. 2. Изменение серверов БД и сопряженных узлов

1) когда практически вся логика обработки данных сосредоточена в хранимых процедурах, которые являются неотъемлемой частью БД;

2) когда хранимые процедуры отсутствуют, а запросы к БД подготавливаются и запускаются из приложений, например, используются Object-relational mapping (ORM — объектно-реляционное отображение, или преобразование [6]).

Конечно, можно предположить и смешенный вариант, но обычно традиции проектирования в конкретной компании перевешивают к выбору одной из двух указанных выше возможностей. При использовании хранимых процедур пропадает необходимости каждый раз доставлять запрос на сервер базы данных. Кроме того, хранимые процедуры в теле БД обычно оттранслированы и даже могут быть постоянно размещены в оперативной памяти, что повышает быстродействие выполнения запросов.

Но соблюдение принципа, согласно которому в базе практически хранятся только данные, а все выполняемые программы размещены вне БД, также имеет право на существование, и даже, более того, широко используется. Отметим также, что программы, написанные таким образом, могут легче переводиться под управление другой СУБД. Так как здесь речь идет только о приведении в соответствие диалектов SQL, что в принципе легче, чем перевод процедур, написанных на одном языке (например, PL/SQL), на другой язык (например, PL/pgSQL) с учетом многих особенностей. Хотелось бы особо отметить применение ORM, который предназначен для сокрытия языка SQL и облегчения миграции на другую БД. Однако следует заметить, что в этом случае жертвуют производительностью.

Не вдаваясь в дискуссию о том, какая же возможность имеет преимущество, отметим лишь, что в практической жизни приходится менять и программы подготовки запросов в приложениях, и хранимые процедуры.

Миграция баз данных и минимизация остановок

С учетом стадий миграции, представленных на рис. 2, необходимо проводить перевод БД в описанной далее последовательности.

1. Подготовить скрипт для создания пустой БД нового вида, на основе существующей, например, с использованием программы "Транслятор структур Oracle—PostgreSQL (OracleToPostgres_Scheme)".

2. С помощью этого скрипта создать пустую БД нового вида.

3. После этого перенести данные из действующей БД во вновь созданную пустую БД, заполнив ее с использованием, например, программы "Трансфер данных Oracle—PostgreSQL (OracleToPostgres_Data)". Этот перенос носит предварительный характер и необходим для тестирования работ, выполняемых на следующих этапах.

4. На этом этапе необходимо подготовить все хранимые процедуры и триггеры для работы с созданной БД. Этот этап наиболее длительный, так как он подразумевает создание нового множества функций и триггеров на основе существующих. На начальном этапе нужно создать шаблоны функций с использованием программы "Предконвертер хранимых процедур Oracle—PostgreSQL (OracleToPostgres_Procedure)". Нужно отдавать себе отчет, что в автоматизированном режиме перевод всех хранимых процедур является практически невозможной задачей. Это определяется тем, что для перевода необходимо знать не только синтаксис двух языков, но и семантическое содержание каждой функции. Вместе с тем использование предконвертера упрощает проблему, по крайней мере с формальной точки зрения. Однако объем "ручных" работ, включая тестирование, на этом этапе весьма существенен.

5. Параллельно можно провести работу по преобразованию SQL-фрагментов в приложениях и преобразованию способов вызова запросов к БД из приложений.

6. Когда все готово, если содержание базы изменилось, можно приступить к окончательному переносу. Для этого следует очистить базу, созданную на этапе 3, затем поместить в нее все хранимые процедуры и триггеры, затем скопировать все измененные модули приложений туда, откуда они должны запускаться.

7. На этом этапе необходимо корректно остановить действующую БД и скопировать из нее все данные в новую БД, используя, например, ранее применяемую программу OracleToPostgres_Data.

8. Запустить новую базу и продолжить работу.

Следует отметить, что остался один проблемный вопрос. Если БД сравнительно небольшая, то последняя стадия ее миграции (остановка и копирование данных) проходит от нескольких минут до нескольких часов. На это время активная работа останавливается, и обработка текущей информации не ведется. Для многих баз такая остановка допустима, особенно в ночное время. Однако БД, обслуживающие множество пользователей, которые обращаются к ней по сети Интернет и имеют широкий географический разброс клиентов, должны работать в непрерывном режиме. Здесь остановки исключаются или могут, в крайнем случае, укладываться в несколько минут. Для таких баз алгоритм должен несколько меняться. При этом возможно использование нескольких способов.

Первый подразумевает использование параллельной записи в две базы данных Oracle и PostgreSQL из приложений. В этом случае возможен вариант с поочередным подключением таблиц. Это значит, что может отлаживаться дублирование первой таблицы, затем второй или нескольких и т. д. Разумеется, связанные таблицы должны подключаться одновременно. Результатом является параллельное использование двух БД с проведением контроля, сравнения и с последующим отключением Oracle. Этот вариант непростой, однако он является для заказывающей организации удобным, хотя и затратным, так как обеспечивает параллельное функционирование двух баз с возможным возвратом на первоначальную базу в течение определенного периода в случае выявления недостатков. В то же время — это решение задачи "в лоб" и на него разработчики, как правило, не идут.

Второй способ — использование механизмов репликации и сторонних программ, которые умеют работать с несколькими типами БД одновременно. Выше упоминалась возможность применения такого инструмента, как SymmetricDS, который позволяет одновременно подключаться к БД Oracle и PostgreSQL. Использование SymmetricDS для работ, проводимых в рамках перевода некоторых БД на Яндексе из Oracle в среду PostgreSQL [1], позволило добиться результата практически без остановки системы.

Функционирование после миграции

Очень важным вопросом является сравнительный анализ функционирования БД после миграции. Все разработчики отдают себе отчет, что Oracle — хорошо проработанная СУБД и сравнительные проверки скорости выполнения запросов, полученные на вновь созданной базе, обычно могут уступать [1] первоначальным исходным значениям на 10...15 % после переноса и первичной доработки.

Поэтому авторам было интересно проверить, как реально изменяется скорость после процедуры миграции без какой-либо первичной доработки и оптимизации. Для этого были выбраны две БД:

1) промышленная БД с накоплением информации более 10 лет объемом несколько ТБ;

2) финансовая база данных небольшого предприятия с периодом хранения информации 2—3 года объемом до 100 МБ.

Приведем примеры измерений времени поиска до и после переноса данных. Результаты отдельных измерений для промышленной базы данных представлены на рис. 3.

Здесь важным является то, что даже без каких-либо дополнительных доработок скорость поиска в БД PostgreSQL оказалось не меньше, чем в Oracle, а даже несколько больше (в пределах погрешности измерений).

Для финансовой базы небольшого предприятия картина несколько изменилась. Результаты времени подготовки простого отчета (отчет 1, без дополнительной оптимизации) приведены на рис. 4.

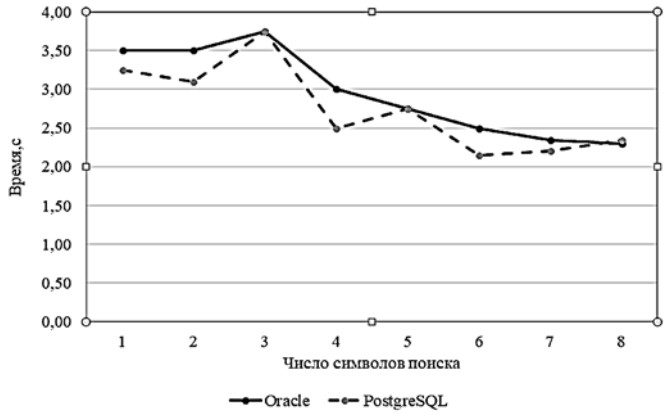

Рис. 3. Время поиска списков физических лиц в промышленной БД до переноса данных (Oracle) и после переноса (PostgreSQL)

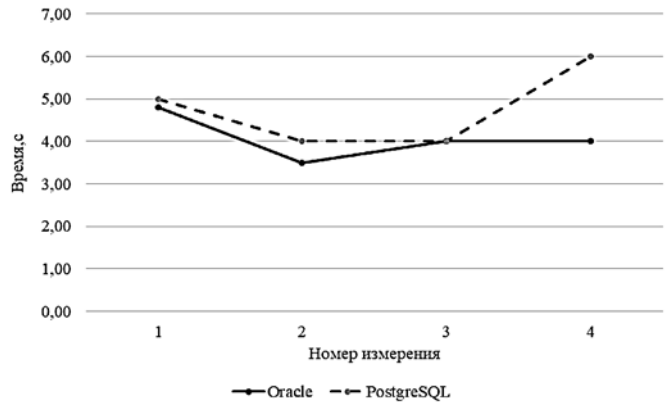

Рис. 4. Время подготовки отчета 1 в финансовой БД предприятия для Oracle и PostgreSQL

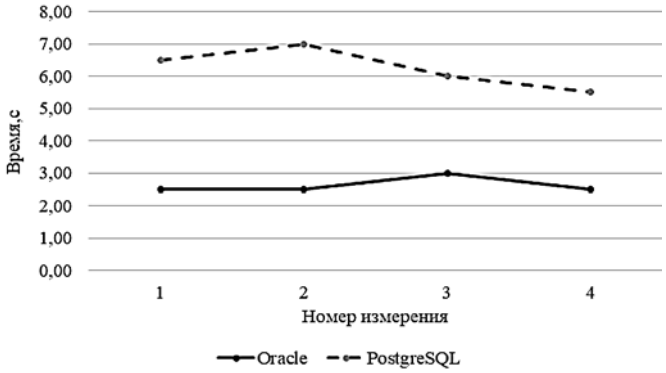

Рис. 5. Время подготовки отчета 2 в финансовой БД предприятия для Oracle и PostgreSQL

На рис. 4 видно, что время подготовки отчета для PostgreSQL несколько больше. Очевидно, объясняется это тем обстоятельством, что подготовка финансового отчета требует существенно более сложных SQL-скриптов, нежели процедура простого поиска. Поэтому для комплексных запросов в PostgreSQL нужно проводить дополнительную оптимизацию уже после переноса БД.

Это же подтверждают данные, приведенные на рис. 5, где отражены результаты времени подготовки отчета 2, несколько более сложного по структуре.

То есть для обеспечения необходимого времени выполнения запросов в БД PostgreSQL требуется проведение дополнительной оптимизации. Такая оптимизация возможна. Часто ее необходимость определяется более совершенным механизмом планирования запросов в Oracle. Хотя рассмотрение вопросов оптимизации не входит в задачу данной статьи, но из опыта авторов следует, что изменение структуры запросов часто дает положительный результат. И скорость выполнения существенно увеличивается, приближаясь к данным Oracle по крайней мере до уровня 10...15 %.

Заключение

Решение задач миграции и разработка способов перемещения с преобразованием БД из среды Oracle в среду PostgreSQL в настоящее время является весьма актуальным. Однако для сложных сетей, которые обладают развитым ПО и объемными базами данных с множеством клиентов, такая миграция характеризуется дополнительными особенностями. Они, как правило, обусловлены накоплением больших объемов данных, обеспечением требований безопасности и учетом ограничений, связанных с эксплуатационной стоимостью комплекса.

Миграция включает:

- преобразование и перенос структуры БД;
- преобразование и перенос данных и представлений;
- преобразование хранимых процедур и триггеров;
- преобразование SQL-фрагментов в приложениях;
- преобразование способов вызова БД из приложений.

В настоящей работе рассмотрено использование трех базовых программных комплексов для проведения преобразований, даны ответы на вопросы минимизации остановок в процессе миграции, а также необходимости повышения эффективности за счет оптимизации выполняемых запросов уже после выполнения основных изменений.

Вопрос о трудозатратах при переносе конкретной базы данных нужно решать отдельно. И что интересно, трудозатраты зависят не столько от объема исходного текста, сколько от его структуры. Так, например, в случае значительного использования особенностей Oracle в текстах языка PL/SQL, объем работы при переводе на язык PL/pgSQL возрастает и требуется достаточно высокая квалификация разработчиков, при условии знания семантической структуры. При минимальном использовании особенностей или их полном отсутствии большая часть работы может выполняться в автоматизированном режиме, и трудозатраты снижаются.

В качестве примера отметим, что реальные трудозатраты при переносе базы [1] без хранимых процедур объемом около 50 таблиц с допустимыми остановками функционирования, не превышающими несколько минут, составило около 400 человеко-дней. Проект миграции базы объемом около 200 таблиц с обширным объемом хранимых процедур при допустимой остановке функционирования 1-2 дня составляет приблизительно 850 человеко-дней с использованием методов [2, 3, 5].

Список литературы

1. **Синицкий В.** Экстремальная миграция на PostgreSQL: без остановки, потерь и тестирования. URL: https://habr.com/ru/company/yoomoney/blog/326998/

2. **Курако Е. А.** Транслятор структур Oracle—PostgreSQL (OracleToPostgres_Scheme). Свидетельство о государственной регистрации программы для ЭВМ № 2019614473 РФ. М.: 05.04.2019.

3. **Курако Е. А.** Трансфер данных Oracle—PostgreSQL (OracleToPostgres_Data). Свидетельство о государственной регистрации программы для ЭВМ № 2019614472 РФ. М.: 05.04.2019.

4. **Документация** PostgreSQL и Postgres Pro. URL: https://postgrespro.ru/docs/

5. **Курако Е. А., Нога Н. Л.** Предконвертер хранимых процедур Oracle—PostgreSQL (OracleToPostgres_Procedure). Свидетельство о государственной регистрации программы для ЭВМ № 2019614616 РФ. М.: 09.04.2019.

6. **Object-relational** mapping. URL: https://en.wikipedia.org/wiki/Object%E2 %80 %93relational_mapping.

On the Issue of Migrating Databases from Oracle to PostgreSQL

E. A. Kurako, keaipu@yandex.ru, **V. L. Orlov**, ovl@ipu.ru, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation

Corresponding author:

Kurako Evgeny A., Researcher, V. A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, 117997, Russian Federation
E-mail: keaipu@yandex.ru

Received on October 19, 2021
Accepted on November 18, 2021

The methods of data transfer from Oracle to PostgreSQL environment are considered, taking into account the step-by-step organization of actions and deter-mining the sequence of migration. The main directions for the transformation of the structure of databases, data formats and executed objects are highlighted. New methods and programs for converting the structure and formats in automatic mode are proposed. The methods of transformation of stored procedures and triggers written in procedural languages, which are PL/SQL and PL/PGSQL, are defined. For cases of possible ambiguous translation, programs have been developed that form hints that facilitate manual operation.

The task of managing transactions inside stored functions in PostgreSQL is difficult. If such transactions are possible for Oracle, then in the PostgreSQL environment, it is necessary to organize them only in external code. They are usually published on computers that work directly with the database server. In this case, the division of the original function into several functions is used, each of which is responsible for its own transaction block. Each such function is called from external code.

The issues of changing software on nodes directly connected (interfaced) with the database server and the issues of minimizing stops during migration are considered, which makes it possible to carry out migration for geographically distributed and round-the-clock functioning objects. Examples of changes in the speed of execution of requests before and after migration for various types of systems are given.

Keywords: *database, DBMS, Oracle, PostgreSQL, migration, structure, data, function, stored procedures, PL/SQL, PL/pgSQL*

For citation:

Kurako E. A., Orlov V. L. On the Issue of Migrating Databases from Oracle to PostgreSQL, *Programmnaya Ingeneria,* 2022, vol. 13, no. 1, pp. 32—40.

DOI: 10.17587/prin.13.32-40

References

1. **Sinitsky V.** Extreme migration to PostgreSQL: without stopping, losses and testing, available at: https://habr.com/ru/company/yoomoney/blog/326998/ (in Russian).

2. **Kurako E. A.** Translator of Oracle—PostgreSQL structures (Oracle-ToPostgres_scheme). Certificate of state registration of the computer program No. 2019614473 of the Russian Federation, Moscow: 05.04.2019 (in Russian).

3. **Kurako E. A**. Oracle-PostgreSQL Data Transfer (Oracle-ToPostgres_data). Certificate of state registration of the computer program No. 2019614472 of the Russian Federation, Moscow: 05.04.2019. (in Russian).

4. **PostgreSQL** and Postgres Pro documentation, available at: https://postgrespro.ru/docs/

5. **Kurako E. A., Noga N. L.** Oracle—PostgreSQL stored procedure preconverter (oracletopostgres_procedure), Certificate of state registration of the computer program No. 2019614616 of the Russian Federation. Moscow: 09.04.2019 (in Russian).

6. **Object-relational** mapping, available at https://en.wikipedia.org/wiki/Object%E2 %80 %93relational_mapping