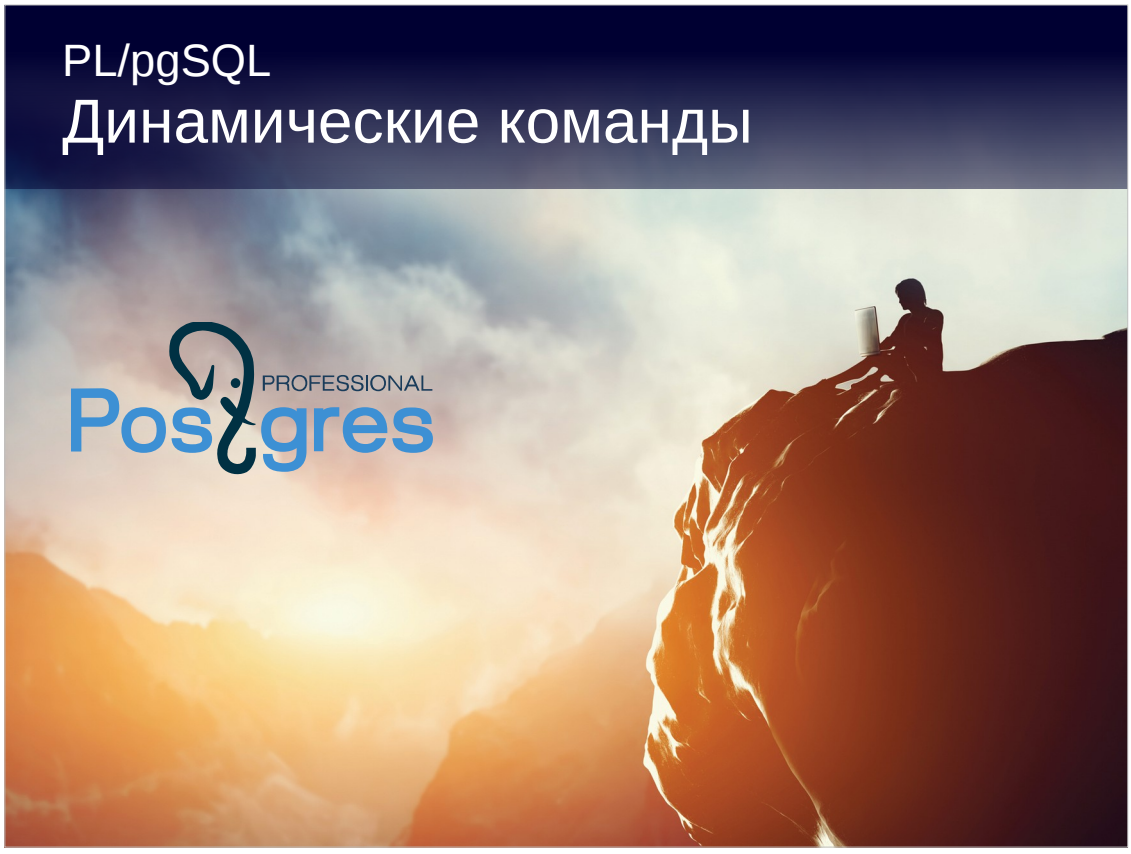


Авторы: Егор Рогов, Павел Лузанов

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Причины использования

Выполнение динамического запроса

Формирование строки с командой

Конструкции динамических команд

Динамический SQL

Динамические команды

- команды SQL в функциях PL/pgSQL
- текст команды формируется в момент выполнения

Причины использования

- дополнительная гибкость в приложении
- оптимизация отдельных запросов

Цена

- не используются подготовленные операторы
- возрастает риск внедрения SQL-кода
- возрастает сложность сопровождения

Под динамическими командами понимаются команды SQL, текст которых формируется и затем выполняется внутри PL/pgSQL-блока. Например в хранимых функциях или в анонимных блоках на этом процедурном языке.

В большинстве случаев можно обойтись без динамических команд, но иногда они могут предоставить дополнительную гибкость. Например, можно встроить в определенные места приложения возможность выполнять команды, считанные из настроек системы. Управлять такими настройками могут специалисты поддержки во время эксплуатации приложения, а не программисты в момент разработки.

Иногда, при формировании отчета с большим количеством необязательных параметров, бывает проще формировать текст запроса прямо во время выполнения только для указанных параметров, чем заранее, при разработке, писать сложный запрос, учитывающий все возможные комбинации параметров.

Ценой за использование динамических команд будет отказ от подготовленных операторов, которые по умолчанию используются в PL/pgSQL. Также нужно следить за безопасностью кода динамических команд. Это удобное место для внедрения SQL-кода.

Следует отметить и существенное возрастание сложности сопровождения. Ведь исходный код приложения не содержит всех выполняемых команд.

Выполнение запроса

Основной синтаксис

```
EXECUTE строка-команды
        [ INTO [STRICT] цель ]
        [ USING выражение [, ...] ];
```

Особенности

цель может быть переменной составного типа или списком скалярных переменных

STRICT — гарантия одной строки

USING — подстановка значений параметров

проверка результата — GET DIAGNISTICS, FOUND

Для выполнения динамических операторов SQL в PL/pgSQL используется команда EXECUTE.

Строка-команды представляет собой выражение, формирующее строку с текстом оператора SQL, который будет выполнен.

Результат работы команды можно поместить в *цель*, указав фразу INTO. Здесь *цель* — переменная составного типа или набор скалярных переменных. В *цель* будет записана только первая строка результата.

Динамический запрос может содержать параметры. В тексте команды параметры обозначаются как $1, $2 и т. д., а значения параметров указываются во фразе USING.

Результат работы динамической команды проверяется так же, как и для статической команды. Например, при помощи GET DIAGNOSTICS или FOUND.

https://postgrespro.ru/docs/postgresql/9.6/plpgsql-statements.html#plpgsql-statements-executing-dyn

Формирование команды

Подстановка значений параметров

- конструкция USING

Подстановка имен объектов

- полезные функции: `format('%I'), quote_ident`

Подстановка литералов

- полезные функции: `format('%L'), quote_literal, quote_nullable`
- экранирование строк: `$$ ... $$`

При формировании текста команды могут быть полезны следующие инструменты.

Если у команды есть параметры, то нужно использовать фразу USING.

Для подстановки имен объектов (имена таблиц, представлений, столбцов и пр.) удобно использовать функцию format со спецификатором %I или функцию quote_ident. Эти функции формируют правильные имена идентификаторов, при необходимости заключая их в двойные кавычки и экранируя специальные символы.

Для подстановки литералов внутри текста команды можно использовать функции quote_literal, quote_nullable или функцию format со спецификатором %L.

Если использования кавычек не избежать, то можно воспользоваться экранированием при помощи конструкции $$ … $$.

Использование с циклом

Синтаксис

```
FOR цель IN EXECUTE строка-команды [USING выражение [, ...]]
LOOP
    тело-цикла
END LOOP;
```

Особенности

цель может быть переменной составного типа
или списком скалярных переменных

необходимые переменные должны быть объявлены

работает аналогично циклу по запросу

Цикл FOR по запросу можно использовать и с динамическими запросами. Для этого используется ключевое слово EXECUTE. В такой запрос можно передать параметры (USING).

В остальном такой цикл по запросу не отличается от рассмотренного в теме «Курсоры».

Использование с курсором

Синтаксис

```
OPEN курсорная-переменная
    FOR EXECUTE строка-команды [USING выражение [, ...]]
...
CLOSE курсорная-переменная;
```

Особенности

курсорная-переменная должна иметь тип REFCURSOR
(не должна быть связана с запросом)

после открытия, используется как обычная курсорная переменная

Курсорную переменную также можно использовать с динамическим запросом.

Сама курсорная переменная должна быть несвязанной с запросом. Текст динамического запроса указывается при открытии курсора командой OPEN после ключевого слова EXECUTE.

После открытия курсора, использование курсорной переменной не отличается от рассмотренного в теме «Курсоры».

Возврат набора строк

Синтаксис

```
RETURN QUERY EXECUTE строка-команды
            [USING выражение [, ...]];
```

Особенности

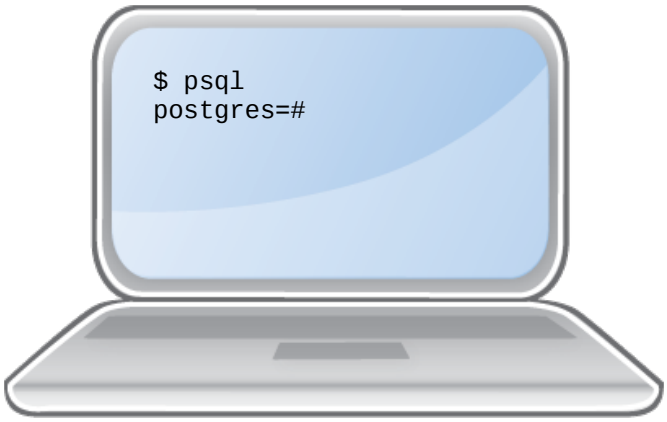
- используется в функциях, возвращающих множество строк (RETURNS SETOF)
- работает аналогично RETURN QUERY
- позволяет создавать функции, возвращающие произвольные множества строк

Еще одна разновидность динамических команд используется при возврате строк из функции.

Для функций, которые возвращают множество строк (RETURNS SETOF), в команде RETURN QUERY предусмотрено ключевое слово EXECUTE.

Такая возможность позволяет создавать функции, которые могут динамически возвращать разные наборы множеств.

Демонстрация



Итоги

Динамические команды PL/pgSQL

- дополнительная гибкость
- оптимизация отдельных запросов
- отказ от подготовленных операторов
- увеличения сложности поддержки

Практика

1. Измените функцию get_catalog так, чтобы запрос к представлению catalog_v формировался динамически и содержал условия только на те поля, которые заполнены на форме поиска в «Магазине».
2. Убедитесь, что реализация не допускает возможности внедрения SQL-кода.
3. Проверьте работу функции в приложении.

1. Скажем, если на форме поиска не заполнены поля «Название книги» и «Имя автора», но установлен флажок «Есть на складе», должен быть сформирован такой, например, запрос:

```
SELECT ... FROM catalog_v WHERE onhand_qty > 0;
```

Учтите, что поиск при такой реализации вовсе не обязательно будет работать эффективнее, но поддерживать его совершенно точно будет сложнее. Поэтому в реальной работе не стоит прибегать к такому приему, если для того нет веских оснований. Тема оптимизации запросов не рассматривается в этом курсе.

Практика

1. Создайте функцию, которая возвращает строки матричного отчета по функциям в базе данных.

   столбцы отчета — имена владельцев функций
   строки отчета — названия схем
   ячейки — количество функций

   Примерный вид результата:

```
        schema        total  postgres  student ...
information_schema      12        12        0
pg_catalog            2811      2811        0
public                   3         0        3
...
```

2. Как можно вызвать такую функцию?

1. Количество столбцов в запросе заранее не известно. Поэтому необходимо сконструировать запрос и затем динамически его выполнить. Текст запроса может быть таким:

```
SELECT pronamespace::regnamespace::text AS schema,
       COUNT(*) AS total
      ,SUM(CASE WHEN proowner = 10 THEN 1 ELSE 0 END) postgres
      ,SUM(CASE WHEN proowner = 16384 THEN 1 ELSE 0 END) student
FROM pg_proc GROUP BY pronamespace::regnamespace ORDER BY schema
```

Выделенные строки — динамическая часть — необходимо сформировать дополнительным запросом. Начало и конец запроса — статические.

Столбец proowner имеет тип OID, для получения имени владельца можно воспользоваться конструкцией: proowner::regrole::text