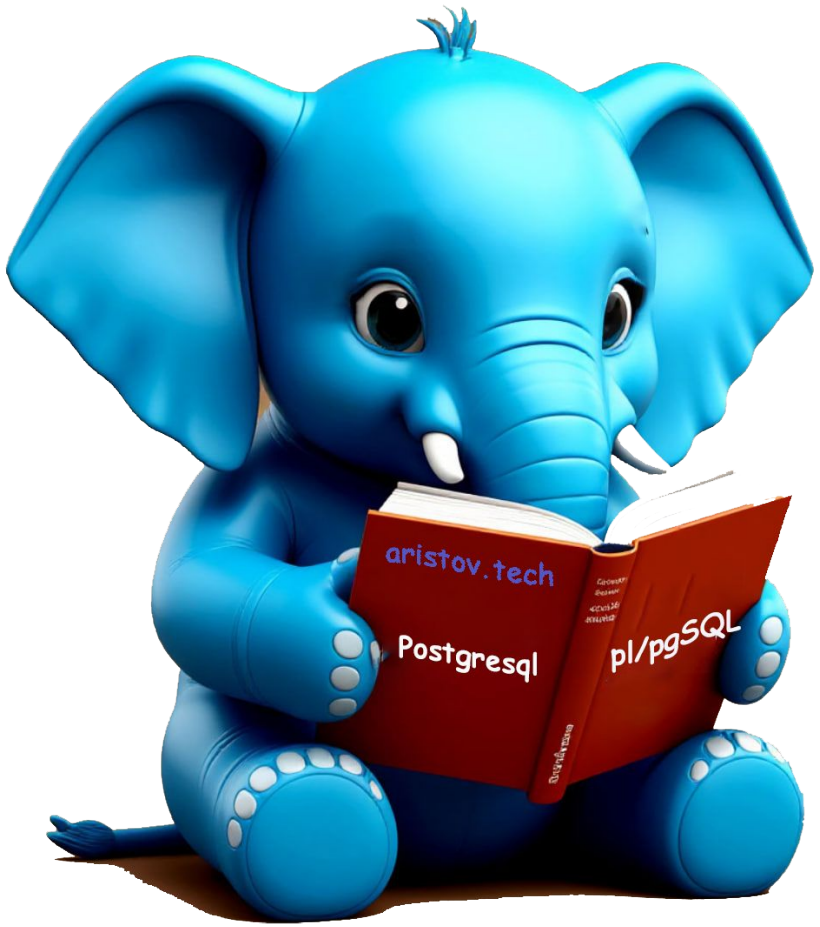
Аристов Евгений

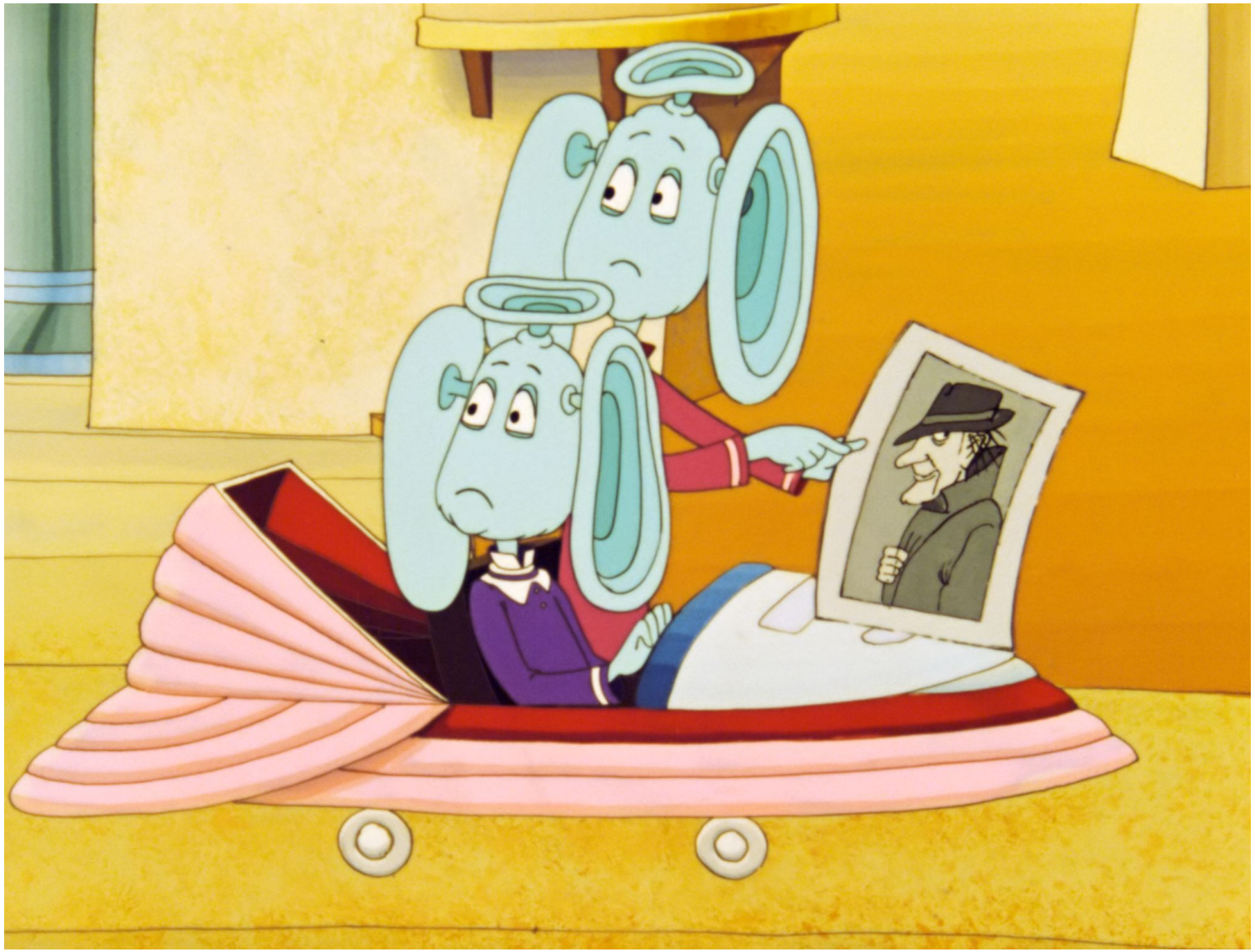
PL/pgSQL в PostgreSQL

за 31 занятие



Курсоры

https://aristov.tech



Аристов Евгений Николаевич

Founder & CEO aristov.tech
25 лет занимаюсь разработкой БД и ПО

Архитектор высоконагруженных баз данных и инфраструктуры

Спроектировал и разработал более ста проектов для финансового сектора, сетевых магазинов, фитнес-центров, отелей.

Сейчас решаю актуальные для бизнеса задачи: аудит и оптимизация БД и инфраструктуры, миграция на PostgreSQL, обучение сотрудников.

Автор более 10 практических курсов по PostgreSQL, MySQL, Mongo и др..

Автор книг по PostgreSQL. Новинка **PostgreSQL 16: лучшие практики оптимизации**

Правила вебинара

Задаем вопрос в чат

Вопросы вижу, отвечу в момент логической паузы

Если есть вопрос голосом - поставьте знак ? в чат

Если остались вопросы, можно их задать на следующем занятии или в комментариях к записи

https://aristov.tech

Маршрут вебинара

Понятие курсора

Использование курсора

Динамический курсор

Ссылочный курсор

Особенности реализации

https://aristov.tech

Курсоры

https://www.postgresql.org/docs/current/plpgsql-cursors.html

Вместо того чтобы сразу выполнять весь запрос, есть возможность настроить курсор, инкапсулирующий запрос, и затем получать результат запроса по нескольку строк за раз.

Одна из причин так делать заключается в том, чтобы избежать переполнения памяти, когда результат содержит большое количество строк (в PL/pgSQL не нужно об этом беспокоиться, так как циклы FOR автоматически используют курсоры, чтобы избежать проблем с памятью)

Также можно вернуть из функции ссылки на курсор, что позволяет вызывающему получать строки запроса. Это эффективный способ получать большие наборы строк из функций.

Курсоры

Доступ к курсорам в PL/pgSQL осуществляется через курсорные переменные, которые всегда имеют специальный тип данных **refcursor**. Один из способов создать курсорную переменную, просто объявить её как переменную типа refcursor. Другой способ заключается в использовании синтаксиса объявления курсора, который в общем виде выглядит так:

имя [[NO] SCROLL] CURSOR [(аргументы)] FOR запрос;

С указанием SCROLL курсор можно будет прокручивать назад. При NO SCROLL прокрутка назад не разрешается - быстрее. Если ничего не указано, то возможность прокрутки назад зависит от запроса.

Если указаны аргументы, то они должны представлять собой пары имя тип_данных, разделённые через запятую. Эти пары определяют имена, которые будут заменены значениями параметров в данном запросе. Фактические значения для замены этих имён появятся позже, при открытии курсора.

https://aristov.tech

Курсоры. Пример объявления

```
DECLARE
curs1 refcursor;
curs2 CURSOR FOR SELECT * FROM tenk1;
curs3 CURSOR (key integer) FOR SELECT * FROM tenk1 WHERE unique1 = key;
```

Все три переменные имеют тип данных refcursor. Первая может быть использована с любым запросом, вторая связана (bound) с полностью сформированным запросом, а последняя связана с параметризованным запросом. (key будет заменён целочисленным значением параметра при открытии курсора.) Про переменную curs1 говорят, что она является несвязанной (unbound), так как к ней не привязан никакой запрос.

Курсоры

Открытие курсора

Прежде чем получать строки из курсора, его нужно открыть. (Это эквивалентно действию SQL-команды DECLARE CURSOR.) В PL/pgSQL есть три формы оператора OPEN, две из которых используются для несвязанных курсорных переменных, а третья для связанных.

Связанные курсорные переменные можно использовать с циклом FOR без явного открытия курсора.

Курсоры

OPEN FOR запрос

OPEN несвязанная_переменная_курсора [[NO] SCROLL] FOR запрос;

Курсорная переменная открывается и получает конкретный запрос для выполнения. Курсор не может уже быть открытым, а курсорная переменная обязана быть несвязанной (то есть просто переменной типа refcursor). Запрос должен быть командой SELECT или любой другой, которая возвращает строки (к примеру EXPLAIN). Запрос обрабатывается так же, как и другие команды SQL в PL/pgSQL: имена переменных PL/pgSQL заменяются на значения, **план запроса кеширует для повторного использования**.

Подстановка значений переменных PL/pgSQL проводится при открытии курсора командой OPEN, последующие изменения значений переменных не влияют на работу курсора. SCROLL и NO SCROLL имеют тот же смысл, что и для связанного курсора.

Пример:

*OPEN curs1 FOR SELECT * FROM foo WHERE key = mykey;*

https://aristov.tech

Динамический курсор

OPEN FOR EXECUTE

OPEN несвязанная_переменная_курсора [[NO] SCROLL] FOR EXECUTE строка_запроса

[USING выражение [, ...]];

Переменная курсора открывается и получает конкретный запрос для выполнения. Курсор не может быть уже открыт и он должен быть объявлен как несвязанная переменная курсора (то есть, как просто переменная refcursor). Запрос задаётся строковым выражением, так же, как в команде EXECUTE. Это также означает, что замена переменных происходит не в самой строке команды. Как и с EXECUTE, значения параметров вставляются в динамическую команду, используя format() и USING. Параметры SCROLL и NO SCROLL здесь действуют так же, как и со связанным курсором.

Пример:

*OPEN curs1 FOR EXECUTE format('SELECT * FROM %I WHERE col1 = $1',tabname) USING keyvalue;*

Курсоры

Открытие связанного курсора

OPEN связанная_переменная_курсора [([имя_аргумента :=] значение_аргумента [, ...])];

Эта форма OPEN используется для открытия курсорной переменной, которая была связана с запросом при объявлении. Курсор не может уже быть открытым. Список фактических значений аргументов должен присутствовать только в том случае, если курсор объявлялся с параметрами. Эти значения будут подставлены в запрос.

План запроса для связанного курсора всегда считается кешируемым. В этом случае нет эквивалента EXECUTE. Обратите внимание, что SCROLL и NO SCROLL не могут быть указаны в этой форме OPEN, возможность прокрутки назад была определена при объявлении курсора.

Примеры (курсоры объявлены ранее):

OPEN curs2;

OPEN curs3(42);

OPEN curs3(key := 42);

Курсоры

Так как для связанного курсора выполняется подстановка значений переменных, то, на самом деле, существует два способа передать значения в курсор. Либо использовать явные аргументы в OPEN, либо неявно, ссылаясь на переменные PL/pgSQL в запросе. В связанном курсоре можно ссылаться только на те переменные, которые были объявлены до самого курсора. В любом случае значение переменной для подстановки в запрос будет определяться на момент выполнения OPEN.

Пример:

```
DECLARE
    key integer;
    curs4 CURSOR FOR SELECT * FROM t WHERE unique1 = key;
BEGIN
    key := 42;
    OPEN curs4;
```

Курсоры. Fetch

FETCH [направление { FROM | IN }] курсор INTO цель;

FETCH извлекает следующую строку из курсора в цель. В качестве цели может быть строковая переменная, переменная типа record, или разделённый запятыми список простых переменных, как и в SELECT INTO. Если следующей строки нет, цели присваивается NULL. Как и в SELECT INTO, проверить, была ли получена запись, можно при помощи специальной переменной FOUND.

Здесь направление может быть любым допустимым в SQL-команде FETCH вариантом, кроме тех, что извлекают более одной строки. А именно: NEXT, PRIOR, FIRST, LAST, ABSOLUTE число, RELATIVE число, FORWARD или BACKWARD. Без указания направления подразумевается вариант NEXT. Значения направления, которые требуют перемещения назад, приведут к ошибке, если курсор не был объявлен или открыт с указанием SCROLL.

Примеры:

FETCH curs1 INTO rowvar;

FETCH curs2 INTO foo, bar, baz;

FETCH LAST FROM curs3 INTO x, y;

FETCH RELATIVE -2 FROM curs4 INTO x;

Курсоры. Move

MOVE [направление { FROM | IN }] курсор;

MOVE перемещает курсор без извлечения данных. MOVE работает точно так же как и FETCH, но при этом только перемещает курсор и не извлекает строку, к которой переместился. Как и в SELECT INTO, проверить успешность перемещения можно с помощью специальной переменной FOUND.

Примеры:

MOVE curs1;

MOVE LAST FROM curs3;

MOVE RELATIVE -2 FROM curs4;

MOVE FORWARD 2 FROM curs4;

https://aristov.tech

Курсоры. UPDATE/DELETE WHERE CURRENT OF

UPDATE таблица SET ... WHERE CURRENT OF курсор;

DELETE FROM таблица WHERE CURRENT OF курсор;

Когда курсор позиционирован на строку таблицы, эту строку можно изменить или удалить при помощи курсора. **Не должно быть группировок.**

Пример:

UPDATE foo SET dataval = myval WHERE CURRENT OF curs1;

https://aristov.tech

Курсоры. Close

CLOSE курсор;

CLOSE закрывает связанный с курсором портал. Используется для того, чтобы освободить ресурсы раньше, чем закончится транзакция, или чтобы освободить курсорную переменную для повторного открытия.

При выходе из блока закрывается автоматически, если не передаётся далее.

Правила хорошего тона предписывают указывать.

Пример:

CLOSE curs1;

https://aristov.tech

Курсоры

Возврат курсора из функции

Курсоры можно возвращать из функции на PL/pgSQL. Это полезно, когда нужно вернуть множество строк и столбцов, особенно если выборки очень большие. Для этого, в функции открывается курсор и *его имя возвращается вызывающему* (или просто открывается курсор, используя указанное имя портала, каким-либо образом известное вызывающему). Вызывающий затем может извлекать строки из курсора. Курсор может быть закрыт вызывающим или он будет автоматически закрыт при завершении транзакции.

перпеменная типа REFCURSOR %)

Курсоры

Обработка курсора в цикле

Один из вариантов цикла FOR позволяет перебирать строки, возвращённые курсором. Вот его синтаксис:

[<<метка>>]

FOR переменная-запись IN связанная_переменная_курсора [([имя_аргумента :=] значение_аргумента [, ...])] LOOP

    операторы

END LOOP [метка];

Курсорная переменная должна быть связана с запросом при объявлении. Курсор не может быть открытым. Команда FOR автоматически открывает курсор и автоматически закрывает при завершении цикла. Список фактических значений аргументов должен присутствовать только в том случае, если курсор объявлялся с параметрами. Эти значения будут подставлены в запрос, как и при выполнении OPEN

Курсоры

Представление для анализа:

SELECT * FROM pg_cursors;

– как ни странно, все запросы DBeaver оборачиваются в курсор

Лучшие практики

- Используйте FOR record IN EXECUTE query для простых циклов - быстрее, чем объявление курсора вручную
- Курсор обычно дороже, чем другие методы
- Применяйте курсоры для очень больших наборов данных
- Оптимизируйте производительность с помощью батчинга - но возможно будет быстрее паджинация
- Тестируйте на реальных объемах данных
- Можно передавать курсор между функциями как ссылку на открытый набор данных
- Закрывайте курсор для более раннего освобождения ресурсов

https://aristov.tech

Практика

Итоги

Итоги

Остались ли вопросы?

Увидимся на следующем занятии

Спасибо за внимание!

Когда дальше и куда?

Аристов Евгений