Авторские права
© Postgres Professional, 2017 год.
Авторы: Егор Рогов, Павел Лузанов

Использование материалов курса
Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

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

Отказ от ответственности
Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или непрямым, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Темы

Причины использования

Объявление и открытие курсора

Операции с курсором

Циклы по курсору и по результатам запроса

Передача курсора клиенту

Причины использования

Курсор подразумевает итеративную обработку

- полная выборка занимает слишком много памяти
- нужна не вся выборка, но размер заранее неизвестен
- способ отдать управление выборкой клиенту
- действительно требуется построчная обработка (обычно нет)

С концепцией курсоров мы уже знакомы по теме «Взаимодействие клиента с СУБД». Там речь шла о курсорах, как о возможности, предоставляемой сервером, и мы смотрели, как к этой возможности обращаться средствами SQL. Теперь поговорим о том, как использовать те же самые курсоры в языке PL/pgSQL.

Зачем вообще может возникнуть необходимость в курсорах? Декларативный SQL в первую очередь предназначен для работы с множествами строк — в этом его сила и преимущество. PL/pgSQL, как процедурный язык, вынужден работать со строками по одной за раз, используя явные циклы. Этого как раз можно добиться, используя курсоры.

Например, полная выборка может занимать слишком много места, так что приходится обрабатывать результаты по частям. Или требуется выборка неизвестного заранее размера — то есть в процессе выборки нужно вовремя остановиться. Или есть необходимость предоставить управление выборкой клиенту.

(Однако еще раз отметим, что, хотя необходимость такой построчной обработки может возникать, в большом количестве случаев ее можно заменить чистым SQL, и код в итоге окажется проще и будет быстрее работать.)

Объявление и открытие

Несвязанные с запросом курсорные переменные

- объявляется переменная типа `refcursor`
- конкретный запрос указывается при открытии

Связанные с запросом курсорные переменные

- при объявлении указывается запрос (возможно, с параметрами)
- при открытии указываются фактические значения параметров

Особенности

- значение курсорной переменной — имя курсора
(можно задать явно или сгенерируется автоматически)
- переменные PL/pgSQL в запросе становятся неявными параметрами
(значения подставляются при открытии курсора)
- запрос предварительно подготавливается

Как мы видели, в SQL курсор объявлялся и открывался одновременно командой DECLARE. В PL/pgSQL это два отдельных шага. Кроме того, для доступа к курсорам используются так называемые курсорные переменные, имеющие тип refcursor и, фактически, содержащие имя курсора (причем если не указывать это имя явно, PL/pgSQL сам позаботится о его уникальности).

Курсорную переменную можно объявить, не связывая ее с конкретным запросом. Тогда при открытии курсора нужно будет указать запрос.

Другой вариант — уже при объявлении переменной указать запрос, возможно с параметрами. При открытии курсора указываются уже только фактические параметры.

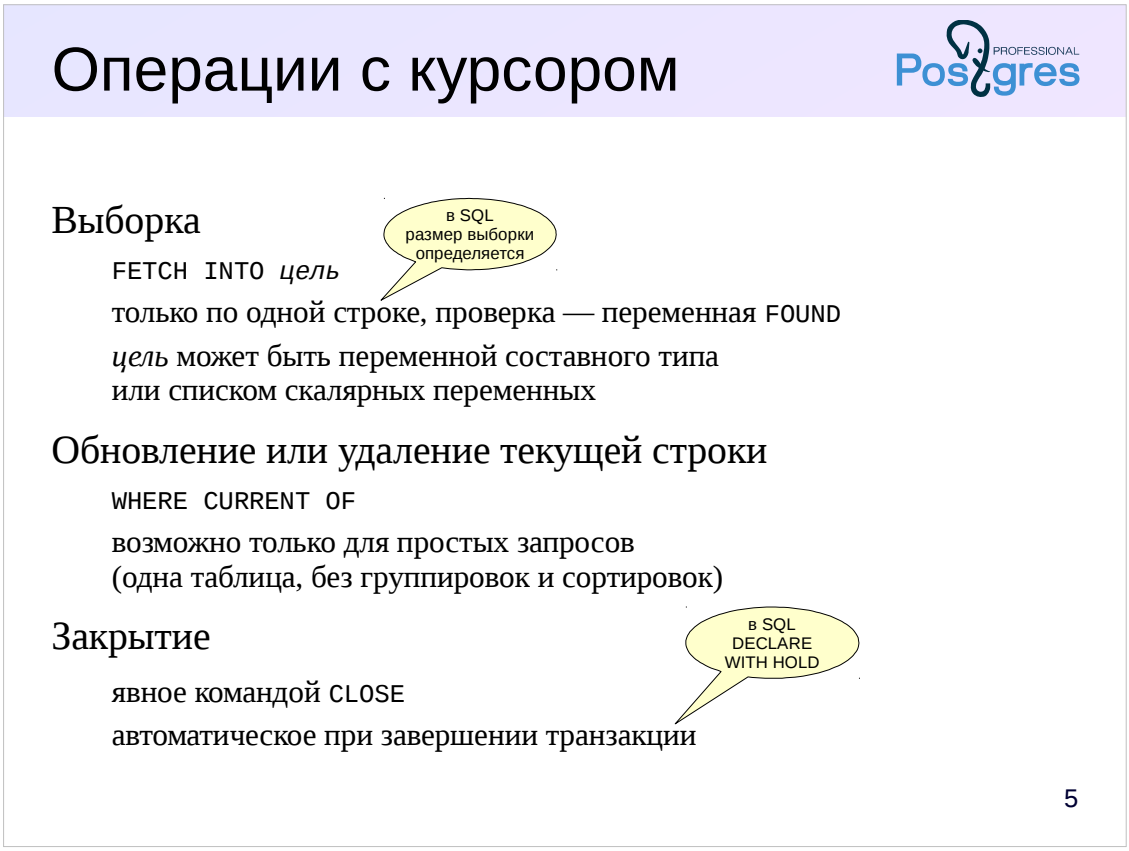
Оба способа равноценны; какой использовать — дело вкуса. И в том, и в другом варианте запрос может иметь неявные параметры — переменные PL/pgSQL.

Напомним, что запрос, открытый с помощью курсора, подготавливается.

https://postgrespro.ru/docs/postgresql/9.6/plpgsql-cursors.html#plpgsql-cursor-declarations

https://postgrespro.ru/docs/postgresql/9.6/plpgsql-cursors.html#plpgsql-cursor-opening

Операции с курсором



Выборка строк из курсора возможна в PL/pgSQL только построчно. Для этого служит команда FETCH INTO, которая читает строку либо в переменную составного типа, либо в набор скалярных переменных (аналогично SELECT INTO). Также есть команда MOVE, сдвигающая «окно» курсора, но не выбирающая данные.

Если запрос достаточно простой (одна таблица, без группировок и сортировок), то в процессе работы с курсором можно обращаться к текущей строке с помощью конструкции CURRENT OF. Она использует прямую ссылку на табличную строку и эффективно работает без индексной поддержки.

Курсор можно закрыть явно командой CLOSE, но он в любом случае будет закрыт при окончании транзакции (в SQL курсор может оставаться открытым и после завершения транзакции, если указать фразу WITH HOLD).

https://postgrespro.ru/docs/postgresql/9.6/plpgsql-cursors.html#plpgsql-cursor-using

Цикл по курсору

Возможный синтаксис

```
OPEN курсорная_переменная FOR запрос;
LOOP
    FETCH курсорная_переменная INTO цель;
    EXIT WHEN NOT FOUND;
    тело-цикла
END LOOP;
CLOSE курсорная_переменная;
```


Процедурная обработка данных подразумевает работу с циклами. Можно организовать перебор и обработку всех строк, возвращаемых курсором, в помощью тех управляющих команд, что мы уже знаем.

При этом для выхода из цикла можно воспользоваться переменной FOUND, которая показывает, была ли выбрана очередная строка командой FETCH.

Цикл по курсору

Синтаксис

```
FOR запись IN курсорная_переменная LOOP
    тело-цикла
END LOOP;
```

Особенности

переменная цикла определяется автоматически
и существует только на время работы цикла

курсор открывается и закрывается автоматически

курсорная переменная должна быть связана с запросом

Но, поскольку часто нужен именно такой цикл, в PL/pgSQL есть специальная команда FOR. Целочисленный вариант FOR мы уже видели в теме «Функции и конструкции языка», а этот вариант работает с курсором.

Курсор открывается и закрывается автоматически; переменная цикла также определяется автоматически и всегда имеет тип record. Такой вариант не предусматривает указания запроса, так что курсорная переменная должна быть заранее связана с запросом.

https://postgrespro.ru/docs/postgresql/9.6/plpgsql-cursors.html#plpgsql-cursor-for-loop

Цикл по запросу

Синтаксис

```
FOR цель IN запрос LOOP
    тело-цикла
END LOOP;
```

Особенности

цель может быть переменной составного типа
или списком скалярных переменных

необходимые переменные должны быть объявлены

в качестве запроса могут выступать команды
SELECT, INSERT/UPDATE/DELETE RETURNING, …

внутри: курсор с выборкой по 10 строк

Более того, есть еще один вариант цикла FOR, в котором не требуется даже объявления курсора — в команде указывается сам запрос.

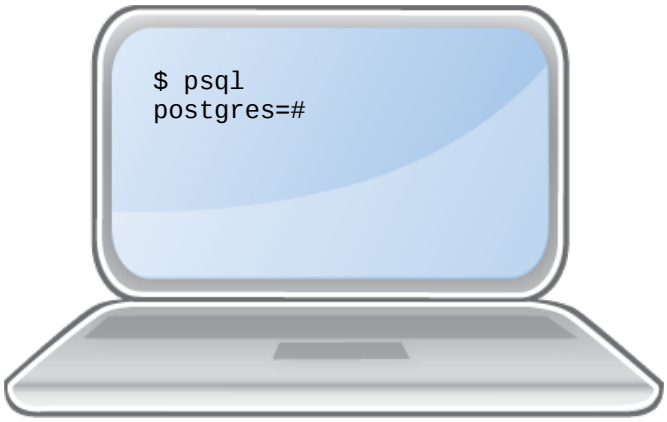
Здесь переменную цикла нужно объявлять; это может быть как переменная составного типа record, так и набор скалярных переменных.

В качестве запроса можно использовать как команду SELECT, так и любую другую команду, возвращающую результат (INSERT, UPDATE, DELETE + RETURNING, и даже EXPLAIN).

Помимо упрощения кода, такая конструкция еще и быстрее работает на больших объемах данных, поскольку размер выборки из курсора устанавливается равным 10, а не 1.

https://postgrespro.ru/docs/postgresql/9.6/plpgsql-control-structures.html

Демонстрация



Итоги

Курсор позволяет получать
и обрабатывать данные построчно

Цикл FOR упрощает работу с курсорами

Обработка в цикле естественна для процедурных языков,
но этим не стоит злоупотреблять

Практика

1. Измените функцию book_name: если у книги больше двух авторов, то в названии указываются только первые два и в конце добавляется «и др.».
2. Проверьте работу функции в SQL и в приложении.
3. Попробуйте написать функцию book_name на SQL. Какой вариант нравится больше — PL/pgSQL или SQL?

1. Например:

```
Хрестоматия. Пушкин А. С., Толстой Л. Н., Тургенев И. С. →
→ Хрестоматия. Пушкин А. С., Толстой Л. Н. и др.
```

Практика

1. Требуется распределить расходы на электроэнергию по отделам компании пропорционально количеству сотрудников (перечень отделов находится в таблице). Напишите функцию, которая примет общую сумму расходов и запишет распределенные расходы в строки таблицы. Числа округляются до копеек; сумма расходов всех отделов должна в точности совпадать с общей суммой.
2. Напишите табличную функцию, имитирующую сортировку слиянием. Функция принимает две курсорные переменные; оба курсора уже открыты и возвращают упорядоченные по неубыванию целые числа. Требуется выдать общую упорядоченную последовательность чисел из обоих источников.

1. В качестве таблицы можно взять:

```
CREATE TABLE depts(
    id serial PRIMARY KEY,
    employees integer,
    expenses numeric(10,2)
);
INSERT INTO depts(employees) VALUES (10),(10),(10);
```

Функция:

```
FUNCTION distribute_expenses(amount numeric) RETURNS void;
```

Ожидаемый результат после вызова функции с параметром 100.0::

```
 expenses
----------
 33.33
 33.34
 33.33
```

2. Функция:

```
FUNCTION merge(c1 refcursor, c2 refcursor) RETURNS SETOF integer;
```

Например, если первый курсор возвращает последовательность 1, 3, 5, а второй — 2, 3, 4, то ожидается результат:

```
 merge
-------
 1
 2
 3
 3
 4
 5
```