Разработка серверной части приложений PostgreSQL

Базовый курс

Преподаватель:

Валентин Степанов

День 3

- PL/pgSQL (продолжение)
  - Выполнение запросов
  - Курсоры
  - Динамические команды
  - Массивы
  - Обработка ошибок

PL/pgSQL

Выполнение запросов

Команды без результата

- Команды SQL встраиваются в код PL/pgSQL
  - как и в выражениях:
  - запрос подготавливается,
  - переменные PL/pgSQL подставляются как параметры
- SELECT → PERFORM
  - удобно для вызова функций с побочными эффектами
  - запросы, начинающиеся на WITH, надо «оборачивать» в SELECT
- INSERT, UPDATE, DELETE и другие команды SQL
  - кроме служебных команд
  - управление транзакциями – только в процедурах и анонимных блоках

Одна строка результата

- SELECT ... INTO
  - получение первой по порядку строки результата
  - одна переменная составного типа или подходящее количество скалярных переменных
- INSERT, UPDATE, DELETE RETURNING ... INTO
  - получение вставленной (измененной, удаленной) строки
  - одна переменная составного типа или подходящее количество скалярных переменных

Проверка результата

- INTO STRICT
  - гарантия получения ровно одной строки
- Диагностика ROW_COUNT
  - число строк, возвращенных (обработанных) последней командой SQL
- Переменная FOUND
  - после команды SQL: истина, если команда вернула (обработала) строку
  - после цикла: признак того, что выполнилась хотя бы одна итерация

Табличные функции

- Строки запроса
  - RETURN QUERY запрос;
- Одна строка
  - RETURN NEXT выражение; если нет выходных параметров
  - RETURN NEXT; если есть выходные параметры
- Особенности
  - строки добавляются к результату, но выполнение функции не прекращается
  - команды можно выполнять несколько раз
  - результат не возвращается, пока функция не завершится

Итоги

- PL/pgSQL тесно интегрирован с SQL
  - в процедурном коде можно выполнять запросы (оформленные как выражения или отдельные команды)
  - в запросах можно использовать переменные
  - можно получать результаты запросов и их статус
- Нужно следить за неоднозначностями разрешения имен

Практика №16

1. Напишите функцию add_author для добавления новых авторов. Функция должна принимать три параметра (фамилия, имя, отчество) и возвращать идентификатор нового автора.

Проверьте, что приложение позволяет добавлять авторов.

2. Напишите функцию buy_book для покупки книги. Функция принимает идентификатор книги и уменьшает количество таких книг на складе на единицу. Возвращаемое значение отсутствует.

Проверьте, что в «Магазине» появилась возможность покупки книг.

Практика №17

Напишите игру, в которой сервер пытается угадать загаданное пользователем животное, задавая последовательные уточняющие вопросы, на которые можно отвечать «да» или «нет».

Если сервер предложил неправильный вариант, он запрашивает у пользователя имя животного и отличающий вопрос. Эта новая информация запоминается и используется в следующих играх.

1. Создайте таблицу для представления информации.
2. Придумайте интерфейс и реализуйте необходимые функции.
3. Проверьте реализацию.

PL/pgSQL

Курсоры

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)

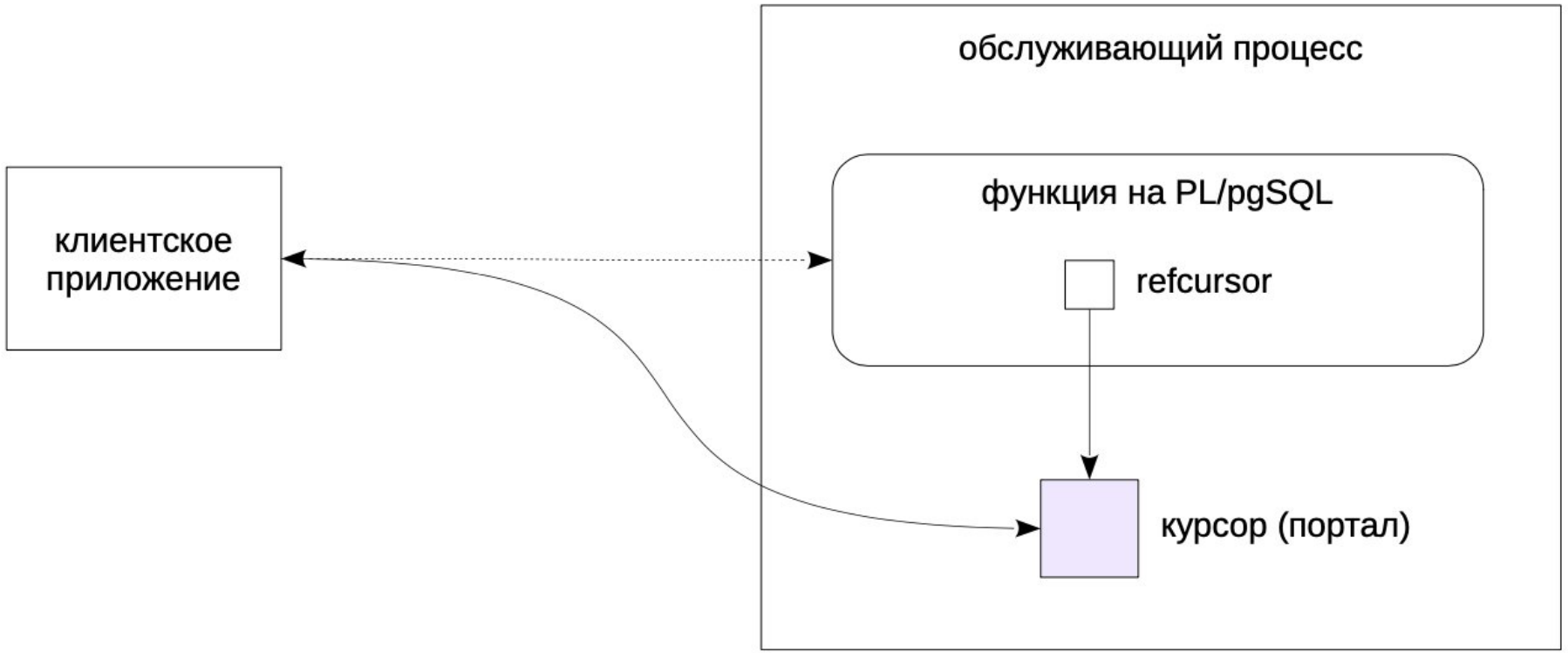
Объявление и открытие

- 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) – нет в SQL
  - 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

Операции с курсором

- Выборка
  - только по одной строке – в SQL размер выборки определяется
- Обращение к текущей строке курсора
  - возможно только для простых запросов (одна таблица, без группировок и сортировок)
- Обычно обработка выполняется в цикле
  - цикл FOR по курсору
  - цикл FOR по запросу без явного определения курсора
- Закрытие
  - явно или автоматически при завершении транзакции – в SQL DECLARE WITH HOLD

Передача курсора клиенту



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 №18

1. Измените функцию book_name: если у книги больше двух авторов, то в названии указываются только первые два и в конце добавляется «и др.».

Проверьте работу функции в SQL и в приложении.

2. Попробуйте написать функцию book_name на SQL. Какой вариант нравится больше – PL/pgSQL или SQL?

Практика №19

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.

PL/pgSQL

Динамические команды

Динамический SQL

- Текст SQL-команды формируется в момент выполнения
- Причины использования
  - дополнительная гибкость в приложении
  - формирование нескольких конкретных запросов вместо одного универсального для оптимизации
- Цена
  - операторы не подготавливаются
  - возрастает риск внедрения SQL-кода
  - возрастает сложность сопровождения

Выполнение запроса

- Оператор EXECUTE
  - выполняет строковое представление SQL-запроса
  - позволяет использовать параметры
  - переменные PL/pgSQL не становятся неявными параметрами
- Может использоваться вместо SQL-запроса
  - сам по себе
  - при открытии курсора
  - в цикле по запросу
  - в предложении RETURN QUERY

Формирование команды

- Подстановка значений параметров
  - предложение USING
  - гарантируется невозможность внедрения SQL-кода
- Экранирование значений
  - идентификаторы: format('%I'), quote_ident
  - литералы: format('%L'), quote_literal, quote_nullable
  - внедрение SQL-кода невозможно при правильном использовании
- Обычные строковые функции
  - конкатенация и др.
  - возможно внедрение SQL-кода!

Итоги

- Динамические команды дают дополнительную гибкость
- Формирование отдельных запросов для разных значений параметров с целью оптимизации
- Не подходят для коротких, частых запросов
- Увеличивается сложность поддержки

Практика №20

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».

Убедитесь, что реализация не допускает возможности внедрения SQL-кода.

Проверьте работу функции в приложении.

Практика №21

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

Столбцы должны содержать имена владельцев функций, строки — названия схем, а ячейки — количество функций данного владельца в данной схеме.

Как можно вызвать такую функцию?

PL/pgSQL

Массивы

Тип массива

- Массив
  - набор пронумерованных элементов одного и того же типа
  - одномерные, многомерные
- Создание
  - использование без явного определения (имя-типа[])
  - неявно при создании базового типа или таблицы (_имя-типа)
- Использование
  - элементы как скалярные значения
  - срезы массива
  - операции с массивами: сравнение, вхождение, пересечение, конкатенация, использование с ANY и ALL вместо подзапроса, ...

Массивы и циклы

- Обычный цикл по индексам элементов
  - array_lower
  - array_upper
- Цикл FOREACH по элементам массива
  - проще, но индексы элементов недоступны

Массивы и подпрограммы

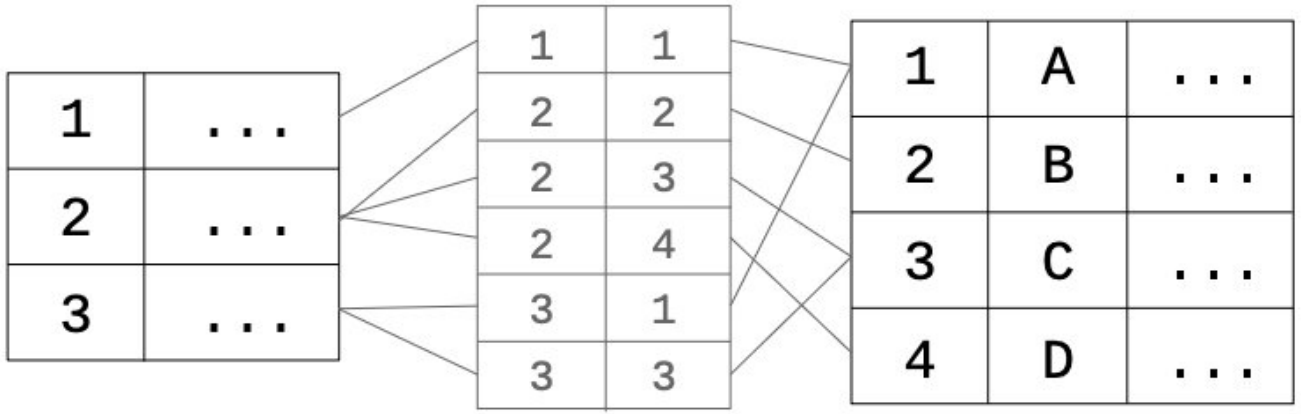
- Подпрограммы с переменным числом параметров
  - все необязательные параметры должны иметь одинаковый тип
  - необязательные параметры передаются в подпрограмму как массив
  - последний формальный параметр-массив объявляется как VARIADIC
- Полиморфные подпрограммы
  - работают со значениями разных типов; тип конкретизируется во время выполнения
  - используют полиморфные псевдотипы anyarray и anynonarray
  - могут иметь переменное число параметров

Массив или таблица?

- компактное представление
- не требуется соединение
- удобно в простых случаях

1	...	{A}
2	...	{B,C,D}
3	...	{A,C}

- отдельные таблицы: многие ко многим
- универсальное решение



Итоги

- Массив состоит из пронумерованных элементов одного и того же типа данных
- Столбец с массивом как альтернатива отдельной таблице: удобные операции и индексная поддержка
- Позволяет создавать функции с переменным числом параметров

Практика №22

1. Создайте функцию add_book для добавления новой книги.

Функция должна принимать два параметра – название книги и массив идентификаторов авторов – и возвращать идентификатор новой книги.

Проверьте, что в приложении появилась возможность добавлять книги.

Практика №23

1. Реализуйте функцию map, принимающую два параметра: массив вещественных чисел и название вспомогательной функции, принимающей один параметр вещественного типа.

Функция возвращает массив, полученный из исходного применением вспомогательной функции к каждому элементу.

2. Реализуйте функцию reduce, принимающую два параметра: массив вещественных чисел и название вспомогательной функции, принимающей два параметра вещественного типа.

Функция возвращает вещественное число, полученное последовательной сверткой массива слева направо.

3. Сделайте функции map и reduce полиморфными.

PL/pgSQL

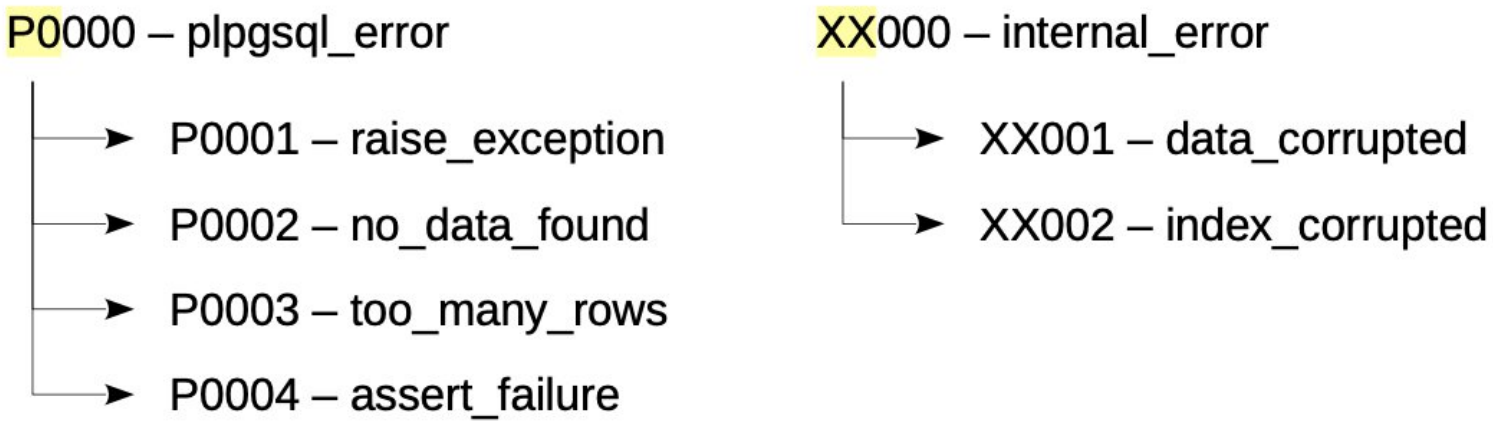
Обработка ошибок

Обработка ошибок в блоке

- Обработка выполняется, если есть секция EXCEPTION
- Выполняется откат к точке сохранения в начале блока
  - точка сохранения устанавливается неявно, если в блоке есть секция EXCEPTION
- Если имеется обработчик, соответствующий ошибке
  - выполняются команды обработчика блок завершается успешно
- Если нет подходящего обработчика
  - блок завершается ошибкой

Имена и коды ошибок

- Информация об ошибке
  - имя
  - пятисимвольный код
  - дополнительные сведения: сообщение, детальное сообщение, подсказка, имена объектов, связанных с ошибкой
- Двухуровневая иерархия



Поиск обработчика

- Необработанная ошибка «поднимается» на уровень выше
  - в объемлющий блок PL/pgSQL, если он есть
  - в вызывающую подпрограмму, если она есть
- Поиск обработчика определяется стеком вызовов
  - то есть не определен статически, а зависит от выполнения программы
- Никем не обработанная ошибка отправляется клиенту
  - транзакция переходит в состояние сбоя, клиент должен ее откатить
  - информация об ошибке записывается в журнал сообщений сервера

Накладные расходы

- Любой блок с секцией EXCEPTION выполняется медленнее
  - из-за установки неявной точки сохранения
- Дополнительные расходы при возникновении ошибки
  - из-за отката к точке сохранения
- Обработку ошибок можно и нужно использовать, когда в этом есть необходимость
  - PL/pgSQL и так интерпретируется и использует SQL для вычисления выражений
  - для многих задач эта скорость более чем удовлетворительна
  - проблемы с производительностью обычно связаны с запросами, а не с кодом PL/pgSQL

Итоги

- Поиск обработчика ошибки происходит «изнутри наружу» в порядке вложенности блоков и вызова функций
- В начале блока с EXCEPTION устанавливается неявная точка сохранения; при ошибке происходит откат к этой точке
- Не перехваченная ошибка приводит к обрыву транзакции; сообщения отправляются клиенту и в журнал сервера
- Обработка ошибок связана с накладными расходами

Практика №24

1. Если при добавлении новой книги указать одного и того же автора несколько раз, произойдет ошибка.

Измените функцию add_book: перехватите ошибку нарушения уникальности и вместо нее вызовите ошибку с понятным текстовым сообщением.

Проверьте изменения в приложении.

Практика №25

1. Ряд языков имеет конструкцию try ... catch ... finally ..., в которой try соответствует BEGIN, catch — EXCEPTION, а операторы из блока finally срабатывают всегда, независимо от того, возникло ли исключение и было ли оно обработано блоком catch. Предложите способ добиться подобного эффекта в PL/pgSQL.

2. Сравните стеки вызовов, получаемые конструкциями GET STACKED DIAGNOSTICS с элементом pg_exception_context и GET [CURRENT] DIAGNOSTICS с элементом pg_context.

3. Напишите функцию getstack, возвращающую текущий стек вызовов в виде массива строк. Сама функция getstack не должна фигурировать в стеке.