Программирование на стороне сервера

Переменные

Работа с локальными переменными

- Использование переменных позволяет:
  - обеспечить временное хранение данных
  - манипулировать хранимыми значениями
  - повторно использовать хранимые значения
  - упростить сопровождение программного кода

Объявление и инициализация переменных

- Переменные объявляются и инициализируются в секции объявлений - **DECLARE**
  - по умолчанию переменная инициализируется **NULL**
  - переменные с ограничением **NOT NULL** должны быть инициализированы при объявлении
  - при использовании многострочного текста каждой переменной отводится своя строка

```
Имя [ CONSTANT ] тип_данных [ COLLATE имя_правила_сортировки ]
[ NOT NULL ] [ { := | = | DEFAULT } выражение_по_умолчанию ];
```

- **Переменная, используемая в качестве счётчика цикла FOR, не указывается в секции объявлений!**

Пример

```
DO LANGUAGE plpgsql
$$
DECLARE
   v_salary NUMERIC(9,2) NOT NULL DEFAULT 1000;
   v_first_name VARCHAR(50);
   v_birth_date DATE := CURRENT_DATE;
   v_comm CONSTANT numeric := 25.5;
BEGIN
   RAISE NOTICE '%, %', v_birth_date, v_comm;
END
$$;
```

Правила именования переменных

- Имя может включать буквы, цифры и некоторые специальные символы ($, _)
- Должно начинаться с буквы или подчёркивания
- Не должно быть зарезервированным словом
- Максимальная длина имени – **63 байта**
  - идентификаторы длиннее автоматически обрезаются до максимальной длины
  - длина в символах зависит от используемой кодировки
- Не должны конфликтовать с используемыми в блоке именами объектов БД
- Для переменных и параметров программ рекомендуется использовать префиксы, например:
  - **v_** - для обычных переменных
  - **p_** - для параметров
  - **c_** - для курсоров

salary$	salary!
first_day, _day	1_day
l_name	name
Bdate	date
v_fromcity	from

Типы данных PL/pgSQL

- Используются при объявлении переменных, параметров подпрограмм и возвращаемых ими значений
  - ***Скалярные SQL-типы***: числовые, денежные, символьные, бинарные, дата и время, логический, геометрические и другие
  - ***Составные типы данных*** (записи)
  - ***Множества записей***: SETOF или TABLE
  - ***Массивы***: ARRAY
  - ***Псевдотипы*** (шаблоны): RECORD, TRIGGER, VOID, VARIADIC
  - ***Полиморфные типы данных***: ANYELEMENT, ANYARRAY, ANYNONARRAY, ANYENUM и ANYRANGE

Наследование типов

- Применяется только в секции **DECLARE**
- Атрибут **%TYPE** позволяет задать характеристики переменной на основе столбца таблицы или на основе уже объявленной переменной
  - Перед %TYPE необходимо указать **имя_таблицы.имя_столбца** или имя уже объявленной скалярной переменной
- Атрибут **%ROWTYPE** позволяет создать переменную со структурой некоторой таблицы
  - редко используется из-за наличия одноимённых таблицам составных типов, но полезен в плане портируемости
- Если тип данных оригинала изменится, изменится и унаследованный тип переменной

Пример

```
do language plpgsql
$block_name$
DECLARE
   v_unitprice "Production"."Products".UNITPRICE%TYPE := 128;
   v_name      "Production"."Products".productname%TYPE;
   v_products  "Production"."Products"%ROWTYPE;
begin
--пустое тело блока
end
$block_name$;
```

Присвоение и вывод значений переменных

- Присвоение значений переменным производится в исполняемой секции или секции исключений с помощью:
  - операторов `:=` или `=`
  - команды `SELECT INTO STRICT`
  - курсоров
- Для вывода значения переменной на экран используют команду **RAISE NOTICE**

```
RAISE NOTICE 'Сообщение %[, %]…', выражение[, выражение]…
```

  - **%** - подстановочный символ
  - количество **выражений** должно соответствовать количеству подстановочных символов в строке
  - значения подставляются в соответствии с порядком упоминания
  - любые выражения в PL/pgSQL вычисляются путём выполнения соответствующих SQL-запросов (с параметрами)

Пример

```
DO LANGUAGE plpgsql
$block_name$
DECLARE
    v_unitprice "Production"."Products".unitprice%type = 128;
    v_name "Production"."Products".productname%TYPE;
BEGIN
    RAISE NOTICE 'Значение переменной v_unitprice = %', v_unitprice;
    v_unitprice = 256;
    RAISE NOTICE 'Значение переменной v_unitprice = %', v_unitprice;

    SELECT          unitprice,   productname
       INTO STRICT v_unitprice, v_name
    FROM "Production"."Products"
    WHERE productid =1;

    RAISE NOTICE 'Продукт %, цена %', v_name, v_unitprice;
END
$block_name$;
```

Вывод

```
Значение переменной v_unitprice = $128.00
Значение переменной v_unitprice = $256.00
Продукт Product HHYDP, цена $18.00
```