Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

# Отчет по лабораторной работе № 4

“Хранимые процедуры”

по дисциплине "Базы данных"

Группа: 43501/3
Студент: Кирпиченков П.С.
Преподаватель: Мяснов А.В.

Санкт-Петербург
2016

1. Программа работы

- Изучить возможности языка PSQL
- Создать хранимую процедуру в соответствии с индивидуальным заданием, полученным у преподавателя
- Выложить скрипт с созданными сущностями в систему контроля версий
- Продемонстрировать результаты преподавателю

2. Теоретические положения

Хранимые процедуры – это находящиеся в базе данных функции, которые хранятся и выполняются на сервере, после чего результат передается клиенту.
Для создания хранимых процедур используется PSQL – процедурный SQL. Он обладает расширенным синтаксисом по сравнению с обычным SQL.
Далее описаны использованные в индивидуальном задании конструкции языка PSQL.

- SET TERM позволяет на время изменить терминирующий символ. Т.к. хранимые процедуры не должны исполняться во время создания, на это время нужно заменить используемую в качестве терминирующего символа точку с запятой на другой. В этом случае процедура считается целиком до нового терминирующего символа, после чего старый вернется обратно.

Вызов: SET TERM <новый> <старый>;

- CREATE PROCEDURE – создание процедуры. Имеет следующий синтаксис:

```
CREATE PROCEDURE имя_процедуры
  [(<входной параметр> [, <входной параметр> ...])]
  [RETURNS (<выходной параметр> [, <выходной параметр> ...])]
AS
  [<объявления переменных>]
BEGIN
  [<PSQL-выражения>]
END
```

- DECLARE VARIABLE – объявление переменной. В функции могут использоваться вспомогательные переменные, тип и имя которых должны быть объявлены в соответствующем разделе.

Синтаксис:

```
DECLARE [VARIABLE] имя тип [{= | DEFAULT} значение_по_умолчанию];
```

- Операторы SELECT в хранимых процедурах должны заканчиваться INTO, т.к. внутри самой процедуры вывод куда-либо не предполагается. В блоке INTO перечисляются имена переменных, в которые производится выборка. Их число и область допустимых значений должны соответствовать указанным в SELECT атрибутам.
- IF THEN ELSE – оператор условного выполнения. Синтаксис:

```
IF <условие> THEN <выражение>
  [ ELSE <выражение> ]
```

- FOR SELECT используется для проведения действий над каждой записью выборки. Синтаксис:

FOR <select-выражение>
  INTO <переменная> [, <переменная> ...]
  [AS CURSOR имя]
DO
  <выражение>

- Оператор присваивания – служит для присвоения нового значения переменной. <переменная> = <новое_значение>.

3. Ход выполнения

Получено следующее индивидуальное задание:

Для заданного оружия скопировать набор возможных боеприпасов и креплений из предыдущего калибра с учетом изменения калибра.

Задание было интерпретировано следующим образом: для выбранного наименования оружия найти калибр, диаметр которого не превышает текущий, который был бы максимален среди удовлетворяющих первому условию. В случае совпадения диаметров сравнивается длина патрона и выбирается калибр с максимальной длиной. Для выбранного нового калибра копируются все боеприпасы в таблице SHELL с установкой нового значения калибра. Т.к. ни крепления, ни аксессуары не связаны с калибром оружия, изменения в соответствующие им таблицы не вносятся.

Алгоритм следующий: сначала производится поиск нового калибра, его ключ помещается в переменную. Затем, если калибр найден, производится выборка всех подходящих патронов и добавление в таблицу новых записей с измененным калибром. Возвращается число добавленных записей. Если не нашлось оружия с указанным названием или патронов под его калибр, возвращается отрицательный код.

Код процедуры:

```
 set term ^;
create procedure downgrade
        ( WeaponName varchar(100) )
returns
        ( CopiedShells integer)
as
        declare variable scount integer;
        declare variable id_caliber integer;
        declare variable id_caliber_prev integer;
        declare variable name varchar(100);
        declare variable description varchar(1000);
        declare variable id_manufacture integer;
        declare variable id_shell_type integer;
        declare variable price integer;
begin
        scount = 0;
        id_caliber = NULL;
        select caliber.id
        from weapon
        join caliber on weapon.id_caliber = caliber.id
        where weapon.name = :WeaponName
        into :id_caliber;
        if ( id_caliber is not NULL) then
        begin
                id_caliber_prev = NULL;
                select first 1 id
                from caliber
                where caliber.diameter <
                        (select diameter from caliber where id = :id_caliber)
                order by diameter desc, length desc
                into :id_caliber_prev;
                if ( id_caliber_prev is not NULL) then
                begin
                        for select name, description, id_manufacture, id_shell_type, price
                                from shell
                                where id_caliber = :id_caliber
                                into :name, :description, :id_manufacture, :id_shell_type, :price
                        do begin
                                insert into shell (id, name, id_caliber, description, id_manufacture, id_shell_type, price)
                                select max(shell.id)+1, :name, :id_caliber_prev, :description, :id_manufacture,
                                                        :id_shell_type, :price
                                from shell;
                                scount = scount + 1;
                        end
                        CopiedShells = scount;
                end
                else CopiedShells = -2;
        end
        else
                CopiedShells = -1;
end^
set term ;^
```

Для проверки работы были произведены выборки из таблицы SHELL.
SQL> select name from shell;

NAME
==========
5.45x39 b
7.62x39 ap
7.62x51 ap
9x19 b
9x18 b
9x18 tr
5.56x45 tr
9x18 bdowngraded
9x18 trdowngraded

После вызова процедуры:
SQL> execute procedure downgrade 'SCAR-L';
SQL> select shell.name as shell_name, caliber.diameter as diameter, caliber.length as length, manufacture.name as manufacture
CON> from shell join caliber on shell.id_caliber = caliber.id
CON> join manufacture on manufacture.id = shell.id_manufacture;

| SHELL_NAME | DIAMETER | LENGTH | MANUFACTURE |
|---|---|---|---|
| 5.45x39 b | 5.45 | 39.00 | Kalashnikov Concern (Izhmash) |
| 7.62x39 ap | 7.62 | 39.00 | Kalashnikov Concern (Izhmash) |
| 9x18 b | 9.00 | 18.00 | Kalashnikov Concern (Izhmash) |
| 9x18 tr | 9.00 | 18.00 | Kalashnikov Concern (Izhmash) |
| 9x18 bdowngraded | 7.62 | 51.00 | Kalashnikov Concern (Izhmash) |
| 9x18 trdowngraded | 7.62 | 51.00 | Kalashnikov Concern (Izhmash) |
| 9x19 b | 9.00 | 19.00 | Heckler & Koch |
| 7.62x51 ap | 7.62 | 51.00 | Fabrique Nationale d'Herstal |
| 5.56x45 tr | 5.56 | 45.00 | Fabrique Nationale d'Herstal |
| 5.56x45 tr | 5.45 | 39.00 | Fabrique Nationale d'Herstal |

В последней строке видим копию единственного подходящего наименования патрона под выбранное оружие.

4. Выводы

Хранимые процедуры могут быть использованы для организации вычислений в самой базе данных. Это позволяет уменьшить сетевой трафик, увеличить производительность трудозатратных операций. Кроме того, если часто используемые функции реализованы в базе, клиентские приложения могут использовать их вместо написания собственных, использование базы становится более однообразным.