

Федеральное агентство по образованию

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт информационных технологий и управления

Кафедра компьютерных систем и программных технологий

ОТЧЕТ

По лабораторной работе №4

Хранимые процедуры

Студентка гр.43501/4 Перетяцько Е.В.

Преподаватель Мяснов А. В.

Санкт-Петербург

2015

Цель работы

Ознакомиться с возможностями реализации более сложной обработки данных на стороне сервера с помощью хранимых процедур.

Программа работы:

1. Изучить возможности языка PSQL
2. Создать две хранимые процедуры в соответствии с **индивидуальным заданием**, полученным у преподавателя
3. Выложить скрипт с созданными сущностями в svn
4. Продемонстрировать результаты преподавателю

Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). Кроме того, в хранимых процедурах возможны циклы и ветвления, то есть в них могут использоваться инструкции управления процессом исполнения.

Хранимые процедуры могут возвращать множества результатов, то есть результаты запроса SELECT. Такие множества результатов могут обрабатываться, используя курсоры, другими сохраненными процедурами, возвращая указатель результирующего множества, либо же приложениями. Хранимые процедуры могут также содержать объявленные переменные для обработки данных и курсоров, которые позволяют организовать цикл по нескольким строкам в таблице. Стандарт SQL предоставляет для работы выражения IF, LOOP, REPEAT, CASE и многие другие. Хранимые процедуры могут принимать переменные, возвращать результаты или изменять переменные и возвращать их, в зависимости от того, где переменная объявлена.

Индивидуальное задание:

1. Рассчитать динамику продаж по курортам поквартально в заданном году по сравнению с аналогичным кварталом годом ранее.

Листинг 1. Динамика продаж.

```
create procedure Kurort_Dynamic (d_year integer)
returns(
    kurort_id integer, absolut_prirost_I integer, temp_rosta_I float,
    absolut_prirost_II integer, temp_rosta_II float,
    absolut_prirost_III integer, temp_rosta_III float,
    absolut_prirost_IV integer, temp_rosta_IV float
) as
declare variable tourID integer;
declare variable kol_buy_this_year_I integer;
declare variable kol_buy_last_year_I integer;
```

```

declare variable kol_buy_this_year_II integer;
declare variable kol_buy_last_year_II integer;
declare variable kol_buy_this_year_III integer;
declare variable kol_buy_last_year_III integer;
declare variable kol_buy_this_year_IV integer;
declare variable kol_buy_last_year_IV integer;
declare variable buy_date date;
begin
  for select kurort_id from kurort order by kurort_id into :kurort_id do
    begin
      kol_buy_this_year_I=0;
      kol_buy_last_year_I=0;
      kol_buy_this_year_II=0;
      kol_buy_last_year_II=0;
      kol_buy_this_year_III=0;
      kol_buy_last_year_III=0;
      kol_buy_this_year_IV=0;
      kol_buy_last_year_IV=0;
      for select tour_id from tour where(id_kurort=:kurort_id) order by tour_id into :tourID
    do
      begin
        for select buy_time from contracts
          where (id_tour=:tourID)and((extract(year from buy_time)=:d_year)or(extract(year
from buy_time)=:d_year-1))
          into :buy_date do
          if (extract(year from buy_date)=:d_year) then
            begin
              if ((extract(month from buy_date)>=1)and(extract(month from buy_date)<=3))
            then
              begin
                kol_buy_this_year_I=kol_buy_this_year_I+1;
              end
              if ((extract(month from buy_date)>=4)and(extract(month from buy_date)<=6))
            then
              begin
                kol_buy_this_year_II=kol_buy_this_year_II+1;
              end
              if ((extract(month from buy_date)>=7)and(extract(month from buy_date)<=9))
            then
              begin
                kol_buy_this_year_III=kol_buy_this_year_III+1;
              end
              if ((extract(month from buy_date)>=10)and(extract(month from
buy_date)<=12)) then
                begin
                  kol_buy_this_year_IV=kol_buy_this_year_IV+1;
                end
              end
            else
              begin
                if ((extract(month from buy_date)>=1)and(extract(month from buy_date)<=3))
            then

```

```

begin
    kol_buy_last_year_I=kol_buy_last_year_I+1;
end
if ((extract(month from buy_date)>=4)and(extract(month from buy_date)<=6))
then
begin
    kol_buy_last_year_II=kol_buy_last_year_II+1;
end
if ((extract(month from buy_date)>=7)and(extract(month from buy_date)<=9))
then
begin
    kol_buy_last_year_III=kol_buy_last_year_III+1;
end
if ((extract(month from buy_date)>=10)and(extract(month from
buy_date)<=12)) then
begin
    kol_buy_last_year_IV=kol_buy_last_year_IV+1;
end
end
end
absolut_prirost_I=kol_buy_this_year_I-kol_buy_last_year_I;
absolut_prirost_II=kol_buy_this_year_II-kol_buy_last_year_II;
absolut_prirost_III=kol_buy_this_year_III-kol_buy_last_year_III;
absolut_prirost_IV=kol_buy_this_year_IV-kol_buy_last_year_IV;
if(kol_buy_last_year_I<>0) then
begin
    temp_rosta_I=kol_buy_this_year_I/kol_buy_last_year_I;
end
else
begin
    temp_rosta_I=NULL;
end
if(kol_buy_last_year_II<>0) then
begin
    temp_rosta_II=kol_buy_this_year_II/kol_buy_last_year_II;
end
else
begin
    temp_rosta_II=NULL;
end
if(kol_buy_last_year_III<>0) then
begin
    temp_rosta_III=kol_buy_this_year_III/kol_buy_last_year_III;
end
else
begin
    temp_rosta_III=NULL;
end
if(kol_buy_last_year_IV<>0) then
begin
    temp_rosta_IV=kol_buy_this_year_IV/kol_buy_last_year_IV;
end
end

```

```

else
begin
    temp_rosta_IV=NULL;
end
suspend;
end

end

```

Результат работы скрипта:

KURORT_ID	ABSOLUT_PRIROST_I	TEMP_ROSTA_I	ABSOLUT_PRIROST_II	TEMP_ROSTA_II	ABSOLUT_PRIROST_III	TEMP_ROSTA_III	ABSOLUT_PRIROST_IV	TEMP_ROSTA_IV
79	-2	0,333	-2	0,333	-1	0,667	1	<null>
80	0	1,000	0	1,000	0	1,000	-1	0,000
81	0	<null>	0	<null>	0	<null>	0	<null>
82	0	<null>	0	<null>	0	<null>	0	<null>
83	0	<null>	0	<null>	0	<null>	0	<null>
84	0	<null>	0	<null>	0	<null>	0	<null>
85	0	<null>	0	<null>	0	<null>	0	<null>
86	0	<null>	0	<null>	0	<null>	0	<null>
87	0	<null>	0	<null>	0	<null>	0	<null>
88	0	<null>	0	<null>	0	<null>	0	<null>
89	-2	0,000	-1	0,500	3	4,000	-2	0,333

Время выполнения скрипта составило Execute time = 12s 384ms

2. Рассчитать по турам отношение суммарной стоимостей экскурсий к стоимостям туров за заданный период.

Листинг 2. Отношение стоимостей

```

create procedure Sum_excurs_div_price_tour (start_d date, end_d date)
returns(
    tourID integer, result float
) as
declare variable i integer;
declare variable summa float;
declare variable price_t integer;
declare variable contract integer;
begin
    for select tour_id, price from tour order by tour_id into :tourID, :price_t do
    begin
        summa=0;
        for select contract_id from contracts
        where (id_tour=:tourID)and(buy_time>=:start_d)and(buy_time<=:end_d) into :contract
        do
        begin
            select sum(price) from contracts_excursions, excursions
            where
            (contract_id=:contract)and(contracts_excursions.excursions_id=excursions.excursions_id) into
            :i;
            if(i>0) then
                summa=summa+i;
            end
        end
        result=summa/price_t;
        suspend;
    end
end

```

```
end  
end
```

При запуске данной процедуры, был задан период 28.02.2014 по 28.01.2015:

Input Parameters [SUM_EXCURS_DIV_PRICE_TOUR]

Parameters ▾ | Create Parameters History Table

Parameters | SQL

Name	Type	Null	Value
START_D	DATE	<input type="checkbox"/>	28.02.2014
END_D	DATE	<input type="checkbox"/>	28.01.2015

Результат работы скрипта:

TOURID	RESULT
1	0,000
2	0,000
3	0,056
4	0,107
5	0,000
6	0,015
7	0,000
8	0,000
9	0,200
10	0,000

Результат работы скрипта для таблицы с 100000 записями

TOURID	RESULT
6	0,045
7	0,809
8	0,041
9	0,058
10	0,030
11	0,082
12	0,359
13	0,080
14	0,772
15	0,092
16	0,059
17	0,073
18	0,034
19	0,040
20	0,041
21	0,015
22	0,026
23	0,042
24	0,015
25	0,027
26	71,191

Время выполнения скрипта Execute time = 32s 464ms

Выводы:

В результате выполнения работы были изучены хранимые процедуры. Хранимые процедуры позволяют хранить какие-либо сложные запросы в БД и выполнять их на стороне сервера. Выполнение функций на сервере снижает нагрузку на канал связи, поскольку передается только окончательный результат, при его наличии. На сервере хранимые процедуры хранятся уже в скомпилированном виде, поэтому их выполнение тратится меньше времени.

Процедуры позволяют организовать интерфейс доступа к данным и в случае изменений на серверной стороне, позволяют избежать необходимости переписывания клиентского приложения.

В хранимых процедурах могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). Это расширяет возможности работы с базами данных и позволяет легче реализовывать многие операции.