Санкт-Петербургский государственный политехнический университет

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №5**

**Дисциплина**: Базы данных

**SQL-программирование: Хранимые процедуры**

Выполнил студент гр. 43501/3 Муравьев Ф.Э.

Преподаватель: Мяснов А. В.

Санкт-Петербург

2015

**Цели работы**

Знакомство 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

**Индивидуальное задание:**

Реализовать следующие хранимые процедуры:

1. Отобразить статистику по клиентам: на какую сумму в среднем за год увеличивается суммарная стоимость ценностей данного клиента.
2. Реализовать процедуру проверки и внесения данных о доступе.

**Храниимая процедура** — объект [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), представляющий собой набор [SQL](https://ru.wikipedia.org/wiki/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](https://ru.wikipedia.org/wiki/DDL), так и [DML](https://ru.wikipedia.org/wiki/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.

**Выполнение работы:**

1. Отобразить статистику по клиентам: на какую сумму в среднем за год увеличивается суммарная стоимость ценностей данного клиента.

**Листинг 1. Статистика по клиентам**

|  |
| --- |
| create procedure ClientStat  returns(  ID integer, IDname varchar(50), sred\_cost float  ) as  declare variable cell integer;  declare variable start\_d date;  declare variable end\_d date;  declare variable acsess\_d date;  declare variable item integer;  declare variable acsess\_t integer;  declare variable cost\_i integer;  declare variable year\_start integer;  declare variable year\_finish integer;  declare variable prev\_year\_cost integer;  declare variable r integer;  declare variable year\_cost integer;  declare variable sred\_year\_cost\_up float;  declare variable kol integer;  declare variable i integer;  begin  for select client\_id, name from clients  order by client\_id into :ID, :IDname do  begin  i=0;  sred\_cost=0;  for select cell\_number, start\_date, end\_date from contracts where (client\_id=:ID)and(rule\_acsess=0)  order by cell\_number into :cell, :start\_d, :end\_d do  begin  year\_start=extract(year from start\_d);  year\_finish=extract(year from end\_d);  prev\_year\_cost=0;  sred\_year\_cost\_up=0;  kol=0;  year\_cost=0;  while (year\_start<year\_finish+1) do  begin  for select date\_acsess, item\_id, acsess\_type from history\_cell  where (cell\_number=:cell)and(extract(year from date\_acsess)=:year\_start)and(date\_acsess>:start\_d)and(date\_acsess<:end\_d)  order by date\_acsess into :acsess\_d, :item, :acsess\_t do  begin  select cost from items where (item\_id=:item) into :cost\_i;  if (acsess\_t=1) then  begin  year\_cost=year\_cost+cost\_i;  end  else if (acsess\_t=2) then  begin  year\_cost=year\_cost-cost\_i;  end  end  r=year\_cost-prev\_year\_cost;  prev\_year\_cost=year\_cost;  sred\_year\_cost\_up=sred\_year\_cost\_up+r;  kol=kol+1;  year\_start=year\_start+1;  end  if(kol<>0) then  begin  sred\_year\_cost\_up=sred\_year\_cost\_up/kol;  sred\_cost=sred\_cost+sred\_year\_cost\_up;  i=i+1;  end  end  if(i<>0) then  begin  sred\_cost=sred\_cost/i;  end  suspend;  end  end |

Результат запуска процедуры:



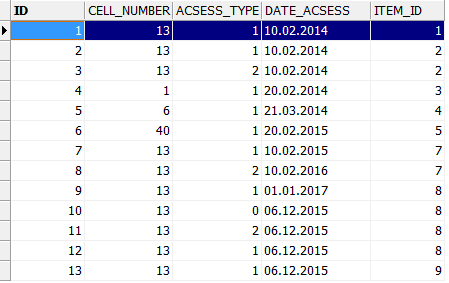
При работе с большими объемами данных(100000 записей в таблице) процедура выполняется за 2s 531ms.

1. Реализовать процедуру проверки и внесения данных о доступе.

**Листинг 2. Процедура проверки и внесения данных о доступе**

|  |
| --- |
| create procedure acsess\_check\_and\_write (client integer, cell integer, type\_a integer, item integer, name varchar(50), cost integer)  returns(  res\_message varchar(50)  ) as  declare variable c\_contract integer;  declare variable c\_group integer;  declare variable s\_date date;  declare variable e\_date date;  declare variable flag integer;  declare variable i integer;  begin  c\_contract=-1;  select contract\_ID, start\_date, end\_date from contracts  where (client\_ID=:client)and(cell\_number=:cell)and(rule\_acsess=0) into :c\_contract, :s\_date, :e\_date;  if(c\_contract=-1) then  begin  for select group\_ID from groups\_clients  where (client\_ID=:client) into :c\_group do  begin  select contract\_ID, start\_date, end\_date from contracts  where (group\_ID=:c\_group)and(cell\_number=:cell)and(rule\_acsess=1) into :c\_contract, :s\_date, :e\_date;  end  end  if(c\_contract=-1) then  begin  res\_message='Acsess denied. Cant find contracts for this data';  suspend;  exit;  end  else if(not ((s\_date<current\_date)and(e\_date>current\_date))) then  begin  res\_message='Acsess denied. Contract ended';  suspend;  exit;  end  if((type\_a=0) or (type\_a=2)) then  begin  flag=-1;  select item\_ID from items where (contract\_id=:c\_contract)and(item\_ID=:item) into :flag;  if (flag=-1) then  begin  res\_message='Acsess denied. Contract doesnt have this item';  suspend;  exit;  end  flag=-1;  select first 1 acsess\_type from history\_cell  where(acsess\_type=2)and(item\_id=:item)and(cell\_number=:cell)and(date\_acsess>:s\_date)and(date\_acsess<:e\_date)  order by date\_acsess desc into :flag;  if (flag=2) then  begin  res\_message='Acsess denied. The item is taken from the cell';  suspend;  exit;  end  select max(ID) from history\_cell into :i;  i=i+1;  insert into history\_cell(ID, cell\_number, acsess\_type, date\_acsess, item\_ID) values (:i, :cell, :type\_a, current\_date, :item);  res\_message='Acsess accept. History\_cell update';  suspend;  exit;  end else if(type\_a=1) then  begin  flag=-1;  select item\_ID from items where (contract\_id=:c\_contract)and(item\_ID=:item) into :flag;  if (flag=-1) then  begin  select max(item\_ID) from items into :item;  item=item+1;  insert into items(item\_ID, contract\_ID, name, cost) values (:item, :c\_contract, :name, :cost);  end  select max(ID) from history\_cell into :i;  i=i+1;  insert into history\_cell(ID, cell\_number, acsess\_type, date\_acsess, item\_ID) values (:i, :cell, :type\_a, current\_date, :item);  res\_message='Acsess accept. History\_cell update';  suspend;  exit;  end else  begin  res\_message='Error. Check enter data';  suspend;  exit;  end  end |

Результат запусков процедуры:



Были добавлены новые записи в таблицу History\_cell, в соответствии с вводимыми данными.

**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.