**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Институт информационных технологий и управления**

**Кафедра компьютерных систем и программных технологий**

**ОТЧЕТ**

**по лабораторной работе №5**

**«SQL-программирование: хранимые процедуры»**

**Информационное обеспечение систем управления**

Студент гр. 43501/1 Васильев А.М.

Преподаватель Мяснов А.В.

**Санкт-Петербург**

**2015**

# 1. Цель работы

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

# 2. Программа работы

1. Изучить возможности языка PSQL

2. Создать две хранимые процедуры в соответствии с индивидуальным заданием, полученным у преподавателя

Индивидуальное задание:

1. Для всех тарифных планов вывести количество сообщений за выбранный период, суммарную длительность разговоров, суммарное поступление средств от клиентов.

2. Начислить бонус наиболее активным абонентам за заданный период в заданном размере.

В работе №4 были созданы хранимые процедуры для выполнения операций sms и звонков, пополнения счета, а также автоматического заполнения БД тестовыми данными. Они приведены в соответствующем отчете.

# 3. Индивидуальное задание

## 3.1 Для всех тарифных планов вывести количество сообщений за выбранный период, суммарную длительность разговоров, суммарное поступление средств от клиентов.

Существует особый тип процедур, возвращающих отношение как результат. Такие процедуры назваются selectable procedures и имеют особый синтаксис, основанный на инструкции suspend, выводящей результат для текущей строки таблицы в виде кортежа из возвращаемых переменных.

set term ^ ;

create procedure tariff\_stat (starttime timestamp,endtime timestamp) returns

(tariff\_name varchar(20),sum\_payments decimal(10,2),sum\_sms int,sum\_minutes int) as

declare variable tariff\_id int;

declare variable sub\_id int;

begin

for select id from tariffs into :tariff\_id do

begin

sum\_payments=0;

sum\_sms=coalesce((select sum(amount) from transactions t where tariff=:tariff\_id and

exists(select \* from sms\_prices where id=t.action) and actime between :starttime and :endtime),0);

sum\_minutes=coalesce((select sum(amount) from transactions t where tariff=:tariff\_id and

exists(select \* from call\_prices where id=t.action) and actime between :starttime and :endtime),0);

for select id from subs where tariff=:tariff\_id into :sub\_id do

begin

sum\_payments = :sum\_payments+coalesce(

(select sum(amount) from payments where sub=:sub\_id and ptime between :starttime and :endtime),0);

end

tariff\_name = (select tariff\_name from tariffs where id = :tariff\_id);

suspend;

end

end ^

set term ; ^

Производится выборка по номеру тарифа и для каждого в цикле производится суммирование осуществленных платежей по всем абонентам, имеющим данный тариф. Также в одном запросе выполняется выборка суммы по количествам соответствующих услуг (звонок в минутах или sms) из таблицы транзакций. Используется встроенная функция coalesce для замены null на 0. В конце выполнения запросов для заданного номера тарифа вызывается suspend.

Синтаксис описания хранимых процедур в скрипте предполагает временную замену терминального символа set term с ; на произвольный символ.

Вызов ХП:

select \* from tariff\_stat(dateadd(-30 day to сurrent\_timestamp),current\_timestamp);

Результат для БД с 100000 записей в определенных таблицах:

SQL> input C:/db/scripts/procedures.txt;

TARIFF\_NAME SUM\_PAYMENTS SUM\_SMS SUM\_MINUTES

==================== ===================== ============ ============

Простецкий 526854.78 4823 74175

SMS-зависимый 1059023.68 9381 147037

Звонить маме 1050233.62 9345 148710

Все по 3 1037908.60 9629 146906

Модем 1047334.73 9691 147845

Стабильный 522987.64 4725 73448

## 3.2 Начислить бонус наиболее активным абонентам за заданный период в заданном размере.

Алгоритм начисления бонусов следующий: бонусы начисляются по результатам текущего месяца, могут иметь ограничение по длительности или ограничение по количеству льготных звонков или sms, или и то, и другое.

Начисление производится по любой комбинации условий: минимальное количество минут разговора, минимальное количество sms, минимальный суммарный объем пополнений счета за текущий месяц.

Типы бонусов хранятся в таблице bonuses. Начисленные абонентам бонусы – в таблице subs\_bonuses.

Формат таблиц:

create table bonuses (

id int primary key,

name varchar (20) not null unique,

minutes\_amount int,

sms\_amount int,

payments\_amount decimal(10,2),

init\_amount\_calls int,

init\_amount\_sms int,

days int);

create table subs\_bonuses(

id int primary key,

sub int references subs(id) not null,

bonus int references bonuses(id) not null,

amount\_left\_sms int,

amount\_left\_calls int,

start\_date date,

expire\_date date);

В хранимой процедуре перебираются все абоненты, для каждого вычисляется суммарная активность за месяц, для каждого типа бонуса проверяются присутствующие (not null) условия начисления. Если все они выполненены, данные о бонусе записываются в subs\_bonuses.

create procedure give\_bonuses as

declare variable sub\_id int;

declare variable sum\_minutes int;

declare variable sum\_sms int;

declare variable sum\_payments decimal(10,2);

declare variable cur\_month smallint;

declare variable bonus\_id int;

declare variable req\_minutes int;

declare variable req\_sms int;

declare variable req\_payments decimal(10,2);

declare variable give\_minutes int;

declare variable give\_sms int;

declare variable give\_start\_date date;

declare variable give\_days int;

declare variable flag int;

begin

cur\_month = extract(month from current\_date);

for select id from subs where balance > 0 into :sub\_id do

begin

sum\_minutes = coalesce((select sum(amount) from transactions tr where exists(select \* from call\_prices where id=tr.action)

and extract(month from actime)=:cur\_month and sub=:sub\_id),0);

sum\_sms = coalesce((select sum(amount) from transactions tr where exists(select \* from sms\_prices where id=tr.action)

and extract(month from actime)=:cur\_month and sub=:sub\_id),0);

sum\_payments = coalesce((select sum(amount) from payments where sub = :sub\_id and extract(month from ptime)=:cur\_month),0);

for select id from bonuses into :bonus\_id do

begin

req\_minutes = (select minutes\_amount from bonuses where id = :bonus\_id);

req\_payments = (select payments\_amount from bonuses where id = :bonus\_id);

req\_sms = (select sms\_amount from bonuses where id = :bonus\_id);

flag=1;

if(:req\_minutes is not null and :sum\_minutes < :req\_minutes) then

flag=0;

if(:req\_sms is not null and :sum\_sms < :req\_sms) then

flag=0;

if(:req\_payments is not null and :sum\_payments < req\_payments) then

flag=0;

if(flag=1) then

begin

give\_minutes=(select init\_amount\_calls from bonuses where id=:bonus\_id);

give\_sms=(select init\_amount\_sms from bonuses where id=:bonus\_id);

give\_start\_date=null;

give\_days = (select days from bonuses where id=:bonus\_id);

if(give\_days is not null) then

begin

give\_start\_date = current\_date;

end

insert into subs\_bonuses (id,sub,bonus,amount\_left\_sms,amount\_left\_calls,start\_date,expire\_date) values

(gen\_id(gen\_sbid,1),:sub\_id,:bonus\_id,:give\_sms,:give\_minutes,:give\_start\_date,dateadd(:give\_days day to :give\_start\_date));

end

end

end

end ^

set term ; ^

commit;

Вызов:

execute procedure give\_bonuses;

Результат выполнения процедуры:

CONSTANT

===============

Bonuses granted

SQL> select count(\*) from subs\_bonuses;

COUNT

============

1

SQL> select \* from subs\_bonuses;

ID SUB BONUS AMOUNT\_LEFT\_SMS AMOUNT\_LEFT\_CALLS START\_DATE EXPIRE\_DATE

============ ============ ============ =============== ================= =========== ===========

1 35128 3 100 100 <null> <null>

SQL> select \* from bonuses;

ID NAME MINUTES\_AMOUNT SMS\_AMOUNT PAYMENTS\_AMOUNT INIT\_AMOUNT\_CALLS INIT\_AMOUNT\_SMS DAYS

============ ==================== ============== ============ ===================== ================= =============== ============

1 Болтун 100 <null> <null> <null> <null> 30

2 Телеграфист <null> 300 <null> <null> 10 30

3 Меценат <null> <null> 400.00 100 100 <null>

Был начислен 1 бонус по результатам пополнения счета (требуется 400 руб). Проверим это, выведя топ 5 абонентов по пополнению счета за месяц:

SQL> select first 5 sub,sum(amount) from payments where extract(month from ptime)=1 group by sub order by -sum(amount);

SUB SUM

============ =====================

35128 403.74

3316 369.73

39575 369.06

28987 368.55

48177 360.12

Такой абонент действительно один.

# 4. Вывод

Работа была посвящена изучению хранимых процедур. В результате были созданы две хранимые процедуры, одна из них относится к так называемым selectable procedures, особому классу хранимых процедур с особым синтаксисом.

Хранимые процедуры используют расширение SQL под названием PSQL, в котором допускается создание переменных, и в то же время ограничено создание таблиц и представлений. За счет этого хранимые процедуры имеют большие возможности, чем один запрос DML.

Хранимые процедуры используются для переноса части операций в саму СУБД для уменьшения интенсивности сетевого взаимодействия.