Санкт-Петербургский государственный политехнический университет

Кафедра компьютерных систем и программных технологий

**Отчёт лабораторной работе №4**

**Дисциплина**: Базы данных

**Тема**: Хранимые процедуры

Выполнил студент гр. 43501/1 А.М. Зинченко

(подпись)

Руководитель А.В. Мяснов

(подпись)

“ ” 2014 г.

Санкт - Петербург

2014

1. **Цель работы**

Изучить возможности языка PSQL, реализовать процедуры

1. **Проведение работы**

- Определить длину маршрута, суммарное время в пути и время остановок для заданного маршрута

Алгоритм процедуры состоит в следующем: необходимо выбрать такую запись из таблицы Routes, что id записи совпадает со значением поля стартовой точки. Значение следующего пункта назначения сохраняется в отдельную переменную для того, чтобы в цикле можно запросить запись, соответствующую следующей остановке. Поля, соответствующие протяженности и времени извлекаются и суммируются. Цикл продолжается пока значение следующего пункта не станет NULL.

Код процедуры приведен ниже:

SET TERM ^ ;

create or alter procedure SHOW\_STAT\_OF\_ROUTE (

NAMERTS\_ID integer)

returns (

DAY\_RUN\_ROUTE integer,

TIME\_RUN\_ROUTE time,

TIME\_PAUSE\_ROUTE time,

LENGTH\_ROUTE decimal(15,2))

as

declare variable TEMP\_CHAR varchar(20);

declare variable TEMP\_PAUSE\_SEC integer;

declare variable TEMP\_RUN\_SEC integer;

declare variable TEMP\_DIST decimal(15,2);

declare variable TEMP\_RUN time;

declare variable TEMP integer;

declare variable TEMP\_PAUSE time;

declare variable TEMP\_NXT integer;

begin

/\* Procedure Text \*/

day\_run\_route=0;

length\_route=0;

temp\_pause\_sec=0;

temp\_run\_sec=0;

select routes.distance,routes.time\_run,routes.time\_pause,routes.nxt from name\_routes,routes

where name\_routes.name\_routes\_id=:namerts\_id and name\_routes.start\_point=routes.routes\_id

into :temp\_dist,:temp\_run,:temp\_pause,:temp\_nxt;

while(:temp\_nxt is not NULL)

do

begin

length\_route=length\_route+temp\_dist;

if(temp\_pause is not NULL) then

temp\_pause\_sec=temp\_pause\_sec+extract(SECOND from temp\_pause)+60\*extract(MINUTE from temp\_pause);

temp\_run\_sec=temp\_run\_sec+extract(SECOND from temp\_run)+60\*extract(MINUTE from temp\_run)+360\*extract(HOUR from temp\_run);

select routes.distance,routes.time\_run,routes.time\_pause,routes.nxt from routes

where :temp\_nxt=routes.routes\_id

into :temp\_dist,:temp\_run,:temp\_pause,:temp\_nxt;

end

day\_run\_route = temp\_run\_sec/86400;

temp\_run\_sec=mod(temp\_run\_sec,86400);

temp=temp\_run\_sec/3600;

temp\_run\_sec = mod(temp\_run\_sec,3600);

temp\_char=temp||':';

temp=temp\_run\_sec/60;

temp\_run\_sec = mod(temp\_run\_sec,60);

temp\_char=temp\_char||temp||':'||temp\_run\_sec;

time\_run\_route=cast(temp\_char as time);

--Расчет времени паузы

temp=temp\_pause\_sec/3600;

temp\_pause\_sec = mod(temp\_pause\_sec,3600);

temp\_char=temp||':';

temp=temp\_pause\_sec/60;

temp\_pause\_sec = mod(temp\_pause\_sec,60);

temp\_char=temp\_char||temp||':'||temp\_pause\_sec;

time\_pause\_route=cast(temp\_char as time);

suspend;

end^

SET TERM ; ^

GRANT SELECT ON NAME\_ROUTES TO PROCEDURE SHOW\_STAT\_OF\_ROUTE;

GRANT SELECT ON ROUTES TO PROCEDURE SHOW\_STAT\_OF\_ROUTE;

GRANT EXECUTE ON PROCEDURE SHOW\_STAT\_OF\_ROUTE TO SYSDBA;

- Определить все маршруты, проходящие между двумя заданными станциями с учетом порядка следования.

Алгоритм процедуры состоит в следующем: детально (знаем список всех остановок) просматриваются все маршруты. Если находим остановку первой заданной станции, переключаем флаг. Если находим остановку второй заданной станции и флаг включен, значит, маршрут соответствует нашим критериям – выводим.

Код процедуры приведен ниже:

SET TERM ^ ;

create or alter procedure FIND\_NEED\_ROUTES (

FROM\_STATION integer,

TO\_STATION integer)

returns (

ID integer)

as

declare variable BOOL\_FF integer;

declare variable TEMP\_ST integer;

declare variable TEMP\_NXT integer;

begin

/\* Procedure Text \*/

for select name\_routes.name\_routes\_id, routes.st,routes.nxt from name\_routes,routes

where name\_routes.start\_point=routes.routes\_id

into :id,:temp\_st,:temp\_nxt

do

begin

bool\_ff=0;

while(:temp\_nxt is not NULL)

do

begin

if(temp\_st=from\_station) then bool\_ff=1;

if(temp\_st=to\_station and bool\_ff=1) then

begin

temp\_nxt=null;

suspend;

end

select routes.st,routes.nxt from routes

where routes.routes\_id=:temp\_nxt

into :temp\_st,:temp\_nxt;

end

end

end^

SET TERM ; ^

GRANT SELECT ON NAME\_ROUTES TO PROCEDURE FIND\_NEED\_ROUTES;

GRANT SELECT ON ROUTES TO PROCEDURE FIND\_NEED\_ROUTES;

GRANT EXECUTE ON PROCEDURE FIND\_NEED\_ROUTES TO SYSDBA;

- Определить десять наиболее загруженных поездов

Можно обойтись обычным SELECT – процедура здесь не требуется

select first 10 trip.train, count(tickets.tickets\_id) tick\_ct from tickets, trip

where tickets.trip=trip.trip\_id

group by trip.train

order by tick\_ct desc

1. **Вывод**

В ходе выполнения данной работы были изучены основы языка PSQL и созданы в соответствии с заданием необходимые процедуры. Процедуры необходимы для организации вычислительного процесса на стороне сервера.

Cложностей c освоением не возникло, поскольку синтаксис языка схож с другими языками, такими как, например, C или Pascal. Но были сложности с суммированием дат – в PSQL в отличие от SQL запрещено суммирование переменных типа DATE, TIME.

Хранимые процедуры позволяют:

- Установить те ограничения целостности, которые невозможно установить через описательные ограничения

- Выполнение процедур на стороне сервера БД – гарант целостности и согласованности данных, и также позволяет разгрузить канал между сервером приложений и сервером БД.