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

Отчёт о лабораторной работе №5

Дисциплина: Базы данных

Тема: Хранимые процедуры

Выполнил студент гр. 43501/1 О.В. Горемыкина

Руководитель А.В. Мяснов

“ ” 2016 г.

Санкт -Петербург

2016

1. Цели работы

Познакомить студентов 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
3. Выложить скрипт с созданными сущностями в svn
4. Продемонстрировать результаты преподавателю

3. Язык PSQL

Procedural SQL (PSQL) — процедурное расширение языка SQL. Это подмножество языка используется для написания хранимых процедур, триггеров и PSQL блоков. Это расширение содержит все основные конструкции классических языков программирования. Кроме того, в него входят немного модифицированные DML операторы (SELECT, INSERT, UPDATE, DELETE и др.).

Процедурное расширение может содержать объявления локальных переменных и курсоров, операторы присваивания, условные операторы, операторы циклов, выброса пользовательского исключений, средства для обработки ошибок, отправки сообщений (событий) клиентским программам. Кроме того, в триггерах доступны специфичные контекстные переменные, такие как NEW и OLD. В PSQL не допустимы операторы модификации метаданных (DDL операторы).

Хранимая процедура является программой, хранящейся в области метаданных базы данных и выполняющейся на стороне сервера. К хранимой процедуре могут обращаться хранимые процедуры (в том числе и сама к себе), триггеры и клиентские программы. Если хранимая процедура вызывает саму себя, то такая хранимая процедура называется рекурсивной.

Хранимые процедуры имеют следующие преимущества:

1. Модульность. Приложения, работающие с одной и той же базой данных, могут использовать одну и ту же хранимую процедуру, тем самым уменьшив размер кода приложения и устранив дублирование кода.

2. Упрощение поддержки приложений. При изменении хранимой процедуры, изменения отражаются сразу во всех приложениях, без необходимости их перекомпиляции.

3. Увеличение производительности. Поскольку хранимые процедуры выполняются на стороне сервера, а не клиента, то это уменьшает сетевой трафик, что повышает производительность

4. Ход работы

Были созданы 2 хранимые процедуры согласно индивидуальному заданию:

4.1. Процедура копирования комплектации со связями с двигателями, трансмиссиями

Параметры: идентификатор комплектации, название новой комплектации.

```
CREATE PROCEDURE copy_comp
    ( id INT, name VARCHAR(30))
AS
    declare variable carcaseType VARCHAR(10);
    declare variable id_engine_trans INT ;
BEGIN
    select carcaseType, id_engine_trans
        from Complectation
        where id_comp = :id
        into :carcaseType, :id_engine_trans;

    insert into Complectation (id_comp, compName, carcaseType, id_engine_trans)
        select max(Complectation.id_comp)+1, :name, :carcaseType,
:id_engine_trans
        from Complectation;
END
^
```

```
execute procedure copy_comp(0,'drive');
 select * from complectation;

 ID_COMP COMPNAME                         CARCASETYPE ID_ENGINE_TRANS
======== ================================ =========== ================
     100 drive                            wagon                    49
       0 gti                              wagon                    49
```

4.2. Процедура подбора рекомендуемой комплектации

Например, клиент хочет купить комплектацию А и набор дополнительных опций (указано в заказе и связях с ним), по этим данным произвести подбор комплектации аналогичного автомобиля (модель), которая бы максимально покрывала список дополнительных опций из заказа клиента и минимально бы отличалась от суммарной стоимости текущего заказа.

Процедура поиска комплектаций, содержащих одну более опцию из заказа:

```
-- search all complectations with same options
CREATE or alter PROCEDURE find_all_complect (id_sale INT)
RETURNS (id_comp int, cnt int)
AS
declare variable sale_comp int;
declare variable delta int;
BEGIN
    select id_comp
    from modelcomplectation
    join car_available using (id_mc)
    join sale using (id_car)
    where id_sale = :id_sale
    into :sale_comp;
    for
        select id_comp, count(id_comp) as cnt from
        (
            select cop2.id_comp
            FROM comp_option as cop1, comp_option as cop2
            where cop1.id_op = cop2.id_op
            and cop1.id_comp <> cop2.id_comp
            and cop1.id_comp = :sale_comp
            union all

            select cop.id_comp
            FROM optionlist as op, comp_option as cop
            where op.id_op = cop.id_op
            and op.id_sale = :id_sale
            and op.id_op not in
                (select id_op from comp_option where id_comp = :sale_comp)
        )
        group by id_comp
        order by cnt desc
        into :id_comp, :cnt
    do begin
        suspend;
    end
end^
```

Отбор комплектаций, которые покрывающих более половины опций из заказа:

```
-- search complectations where count of the same options take more then 50%
CREATE or alter PROCEDURE find_optimal_complect (id_sale INT)
RETURNS (id_comp int, cnt int)
AS
declare variable maxcnt int;
BEGIN

    select max(cnt) as maxcnt
    from find_all_complect (:id_sale)
    into :maxcnt;

    maxcnt = :maxcnt/2;

    for
        select id_comp,cnt
        from find_all_complect (:id_sale)
        where cnt > :maxcnt
        into :id_comp,:cnt
    do begin
        suspend;
    end

END^
```

Отбор моделей с найденными комплектациями, соответствующих модели из заказа. Из-за недостаточного количества данных в базе отбор моделей при тестировании не производится. Подсчет разницы в цене.

```
-- search models with found complectations
-- search the difference in price
CREATE or alter PROCEDURE find_all_models (id_sale INT)
RETURNS (id_mc int, delta int)
AS
declare variable newprice int;
declare variable oldprice int;
declare variable model varchar(30);
BEGIN

-- find equal model ( not used due to insufficient data )
/*
    select model
    from modelcomplectation
    join car_available using (id_mc)
    join sale using (id_car)
    where id_sale = :id_sale
    into :model;
*/
    select totalprice from sale where id_sale = :id_sale
    into :oldprice;
    for
        select id_mc, price from modelcomplectation
        where id_comp in
        (
            select id_comp from find_optimal_complect (:id_sale)
        )
```

```
        --and model = :model
        order by :delta
        into :id_mc, :newprice
    do begin
        delta = abs(:oldprice - :newprice);
        suspend;
    end
end ^
```

Отображение подобранной комплектации:

```
-- output found configuration
CREATE or alter PROCEDURE find_equal (id_sale INT)
RETURNS (old_id int, old_model varchar (15), old_comp varchar(15), old_price
int,
        new_id int, find_model varchar (15), find_comp varchar(15), new_price
int )
AS
    declare variable mc int;
    declare variable id_comp int;
    declare variable min_delta INT ;
BEGIN
    select id_comp, compname, model, totalprice
    from complectation
    join modelcomplectation using (id_comp)
    join car_available using (id_mc)
    join sale using (id_car)
    where id_sale = :id_sale
    into :old_id, :old_comp, :old_model, :old_price;

    select first 1 id_mc, delta
    from find_all_models (:id_sale)
    order by delta
    into :mc, :min_delta;

    select id_comp, compname, model, price from complectation
    join modelcomplectation using (id_comp)
    where id_mc = :mc
    into :new_id,:find_comp, :find_model, :new_price;
    suspend;

END
^
```

Отображение дополнительных опций выбранной и найденной моделей:

```
CREATE or alter PROCEDURE old_options (id_sale INT)
RETURNS (old_options varchar (30))

AS
BEGIN
    for
        select optiontype
        from extraoption
        join optionlist using (id_op)
        where id_sale = :id_sale

        union
```

```
        select optiontype
        from extraoption
        join comp_option using (id_op)
        join complectation using (id_comp)
        join modelcomplectation using (id_comp)
        join car_available using (id_mc)
        join sale using (id_car)
        where id_sale = :id_sale
        into :old_options
    do begin
        suspend;
    end
END ^

CREATE or alter PROCEDURE new_options (id_comp INT)
RETURNS (new_options varchar (30))

AS
BEGIN
    for
        select optiontype
        from extraoption
        join comp_option using (id_op)
        where id_comp = :id_comp
        into :new_options
    do begin
        suspend;
    end
END ^
```

Подбор рекомендуемой комплектации для заказа 3:

```
SQL> select * from find_equal(3);

      OLD_ID OLD_MODEL       OLD_COMP            OLD_PRICE
============ ================ ================ ============
          44 Mazda CX-9       exclusive                1373

      NEW_ID FIND_MODEL      FIND_COMP           NEW_PRICE
============ ================ ================ ============
          84 Mazda3 sedan     supreme                  1265
```

```
SQL> select * from old_options(3);

OLD_OPTIONS
==============================
Audio system
Center armrest
Rear Parking Sensors
Winter set

SQL> select * from new_options (84);

NEW_OPTIONS
==============================
Center armrest
Rear Parking Sensors
```

5. Выводы

Хранимые процедуры позволяют сохранить часто используемые однотипные операции сложной выборки данных из базы. Особенно это полезно, если операции отличаются только константами, использующимися при наложении условий на данные. Для этих целей существуют параметры, передаваемые в хранимую процедуру.

Кроме того, большие и часто используемые операции имеют преимущество, выполняясь на сервере базы данных, так как это уменьшает количество передаваемых по сети данных.

Также необходимо отметить, что хранимые процедуры позволяют распределить доступ определённым группам пользователей БД к определённым хранимым процедурам.

Но у хранимых процедур есть и недостатки. Необходимо поддерживать актуальность используемых операций при изменении структуры базы данных. Также хранимые процедуры зависят от типа и версии используемой СУБД , то перенос проекта из одной СУБД в другую достаточно сложен.