Санкт-Петербургский государственный политехнический
университет Петра Великого
**Кафедра компьютерных систем и программных технологий**

**Отчет по лабораторной работе**
**Дисциплина:** Базы данных
**Тема:** SQL-программирование: хранимые процедуры

Выполнил
студент гр. 43501/3 _____ Д. А. Зобков
(подпись)

Преподаватель _____ А. В. Мяснов
(подпись)

“___”_____ 2016 г.

Санкт-Петербург
2016 г.

## 1 Цель работы

Познакомиться 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 системе контроля версий;
4. Продемонстрировать результаты преподавателю.

## 3 Ход работы

3.1 Реализовать процедуру выдачи награды за самый популярный по продажам альбом: за заданный период вычислять самый продаваемый альбом и выдавать ему награду заданного типа

```
CREATE OR ALTER PROCEDURE sp1(
    START_DATE DATE,
    FINISH_DATE DATE,
    AWARD_ID INTEGER
)
AS
DECLARE VARIABLE TOP_ALBUM INTEGER;
DECLARE VARIABLE ID INTEGER;
BEGIN
    /* Находим самый продаваемый альбом */
    SELECT FIRST 1 ALB_IN_CD.ALB_ID AS ALB FROM SELLING
        JOIN ALB_IN_CD ON SELLING.CD_ID = ALB_IN_CD.CD_ID
        WHERE SOLD_DATE BETWEEN :START_DATE AND :FINISH_DATE
        GROUP BY ALB
        ORDER BY SUM(SOLD) DESC
        INTO :TOP_ALBUM;
    /* Вычисляем количество записей в таблице */
    SELECT COUNT(*) FROM (SELECT * FROM ALB_AWARDS) INTO :ID;
    /* Инкрементируем первичный ключ */
    ID = ID + 1;
    /* Добавляем запись */
    INSERT INTO ALB_AWARDS VALUES (:ID, :TOP_ALBUM, :AWARD_ID);
END
-- Завершение транзакции
COMMIT;
```

3.2 Реализовать процедуру вычисления агрегатов — сумму продаж по дням, неделям, месяцам, годам, городам, странам. Для хранения вычисленных значений необходимо создать отдельную таблицу.

Для хранения вычисленных значений создадим по таблице для каждого агрегата:

```
CREATE TABLE SOLD_DAY_SUMMARY (
    SUMMARY_DATE DATE    NOT NULL,
    SOLD_SUM     INTEGER NOT NULL
);

CREATE TABLE SOLD_WEEK_SUMMARY (
    START_DATE  DATE    NOT NULL,
    FINISH_DATE DATE    NOT NULL,
    SOLD_SUM    INTEGER NOT NULL
);

CREATE TABLE SOLD_MONTH_SUMMARY (
    START_DATE  DATE    NOT NULL,
    FINISH_DATE DATE    NOT NULL,
    SOLD_SUM    INTEGER NOT NULL
);

CREATE TABLE SOLD_YEAR_SUMMARY (
    START_DATE  DATE    NOT NULL,
    FINISH_DATE DATE    NOT NULL,
    SOLD_SUM    INTEGER NOT NULL
);

CREATE TABLE SOLD_CITY_SUMMARY (
    CITY_ID  INTEGER NOT NULL,
    SOLD_SUM INTEGER NOT NULL
);

CREATE TABLE SOLD_COUNTRY_SUMMARY (
    COUNTRY_ID INTEGER NOT NULL,
    SOLD_SUM   INTEGER NOT NULL
);

ALTER TABLE SOLD_CITY_SUMMARY
    ADD FOREIGN KEY (CITY_ID) REFERENCES CITY_LIST (CITY_ID);
ALTER TABLE SOLD_COUNTRY_SUMMARY
    ADD FOREIGN KEY (COUNTRY_ID) REFERENCES COUNTRY_LIST (COUNTRY_ID);
-- Завершение транзакции
COMMIT;
```

Реализованная процедура:

```
/* Основная процедура */
CREATE OR ALTER PROCEDURE sp2(
    OPERATION VARCHAR(7),            -- Вид диапазона
    SUMMARY_DATE DATE                -- Дата для пересчета
)AS
```

```
DECLARE VARIABLE START_DATE DATE;     -- Дата начала диапазона
DECLARE VARIABLE FINISH_DATE DATE;    -- Дата конца диапазона
DECLARE VARIABLE ID INT;              -- ID городов/стран
DECLARE VARIABLE SOLD_COUNT INT;      -- Количество проданных дисков за
   диапазон
DECLARE VARIABLE QUERY VARCHAR(100); -- Строка с запросом
BEGIN
  IF (:OPERATION = 'DAY') THEN
  BEGIN
    /* Находим количество продаж */
    SELECT SUM(SOLD) FROM SELLING
      WHERE SOLD_DATE=:SUMMARY_DATE
      INTO :SOLD_COUNT;

    /* Добавляем новую запись либо обновляем старую */
    UPDATE OR INSERT INTO SOLD_DAY_SUMMARY
      VALUES (:SUMMARY_DATE, :SOLD_COUNT)
      MATCHING (SUMMARY_DATE);
  END

  ELSE IF (:OPERATION = 'WEEK' OR :OPERATION = 'MONTH' OR :OPERATION =
   'YEAR') THEN
  BEGIN
    /* Находим начало и конец диапазона */
    IF (:OPERATION = 'WEEK') THEN BEGIN
      START_DATE  = :SUMMARY_DATE+1-EXTRACT(WEEKDAY FROM :SUMMARY_DATE)
   ;
      FINISH_DATE = :SUMMARY_DATE+7-EXTRACT(WEEKDAY FROM :SUMMARY_DATE)
   ;
    END
    ELSE IF (:OPERATION = 'MONTH') THEN BEGIN
      START_DATE  = :SUMMARY_DATE-EXTRACT(DAY FROM :SUMMARY_DATE)+1;
      FINISH_DATE = DATEADD(MONTH, 1, :SUMMARY_DATE)-EXTRACT(DAY FROM :
   SUMMARY_DATE);
    END
    ELSE IF (:OPERATION = 'YEAR') THEN BEGIN
      START_DATE  = :SUMMARY_DATE-EXTRACT(YEARDAY FROM :SUMMARY_DATE);
      FINISH_DATE = DATEADD(YEAR, 1, :SUMMARY_DATE)-EXTRACT(YEARDAY
   FROM :SUMMARY_DATE);
    END

    /* Находим количество продаж */
    SELECT SUM(SOLD) FROM SELLING
      WHERE SOLD_DATE BETWEEN :START_DATE AND :FINISH_DATE
      INTO :SOLD_COUNT;

    /* Создаем запрос */
    QUERY = 'UPDATE OR INSERT INTO SOLD_' || :OPERATION || '_SUMMARY '
      || 'VALUES (?, ?, ?) '
      || 'MATCHING (START_DATE, FINISH_DATE)';

    /* Выполнение динамического запроса */
    EXECUTE STATEMENT (QUERY) (:START_DATE, :FINISH_DATE, :SOLD_COUNT);
  END
```

```
54
  ELSE IF (:OPERATION = 'CITY') THEN
56  BEGIN
    /* Находим кол-во продаж (диапазон не учитывается) */
58    FOR SELECT CITY_ID, SUM(SOLD)
      FROM SELLING
60    GROUP BY CITY_ID
      INTO :ID, :SOLD_COUNT
62    DO BEGIN
      UPDATE OR INSERT INTO SOLD_CITY_SUMMARY
64      VALUES (:ID, :SOLD_COUNT)
        MATCHING (CITY_ID);
66    END
  END
68
  ELSE IF (:OPERATION = 'COUNTRY') THEN
70  BEGIN
    /* Находим кол-во продаж (диапазон не учитывается) */
72    FOR SELECT CITIES_IN_COUNTRY.COUNTRY_ID AS COUNTRY,
      SUM(SELLING.SOLD) FROM SELLING
74    JOIN CITIES_IN_COUNTRY
      ON CITIES_IN_COUNTRY.CITY_ID = SELLING.CITY_ID
76    GROUP BY COUNTRY
      INTO :ID, :SOLD_COUNT
78    DO BEGIN
      UPDATE OR INSERT INTO SOLD_COUNTRY_SUMMARY
80      VALUES (:ID, :SOLD_COUNT)
        MATCHING (COUNTRY_ID);
82    END
  END
84 END
-- Завершение транзакции
86 COMMIT;
```

## 4 Выводы

В ходе выполнения данной работы было проведено ознакомление с хранимыми процедурами, хранящимися в базе данных и напоминающие функции из языков программирования. Хранимые процедуры позволяют упростить выполнение однотипных операций сложной выборки данных из базы, например, в случае отличия только в константах.

Использование хранимых процедур повышает безопасность, позволяя давать пользователю доступ только к ним и изолировать от самой базы данных, уменьшает количество запросов к серверу, вследствие чего повышается скорость работы.

К недостаткам хранимых процедур можно отнести отсутствие контроля целостности кода при изменении структуры БД, вследствие чего поддержка такой БД более затратна, „размазывание“ логики между клиентским приложением и БД, непереносимость между различными диалектами СУБД.