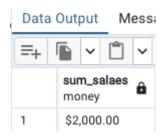
Лабораторная работа 12

1. Написать хранимую процедуру GET_CUST, которая получает идентификатор клиента и возвращает его имя (поле COMPANY таблицы CUSTOMERS), имя закрепленного за ним служащего (поле FIO таблицы SALESPERS) и название города, в котором расположен офис этого клиента (поле CITY таблицы OFFICES). Первый из передаваемых параметров - входной, остальные три – выходные (используются для передачи запрошенных данных вызывающей процедуре). Продемонстрировать вызов данной процедуры из другой процедуры Transact-SQL. Например, GET CUST1 (идентификатор клиента).

```
CREATE OR REPLACE PROCEDURE GET CUST (
     cust id NUMERIC,
    OUT company CHAR (20),
    OUT fio CHAR(20),
     OUT city CHAR(10)
 )
 AS $$
 BEGIN
     SELECT c.company, s.fio, o.city
     INTO company, fio, city
     FROM customers c
     JOIN salespers s ON c.cust rep = s.empl num
     JOIN offices o ON c.cust num = o.cust num
     WHERE c.idcust = cust id;
 END;
 $$ LANGUAGE plpgsql;
 CREATE OR REPLACE PROCEDURE GET CUST1 (cust id NUMERIC)
 AS $$
 DECLARE
     v company CHAR(20);
     v fio CHAR(20);
    v city CHAR(10);
     CALL GET CUST (cust id, v company, v fio, v city);
     RAISE NOTICE 'Company: %', v company;
     RAISE NOTICE 'FIO: %', v fio;
     RAISE NOTICE 'City: %', v city;
 END:
 $$ LANGUAGE plpgsql;
 CALL GET CUST1('1');
Data Output Messages
                      Notifications
NOTICE: Company: Фишер
NOTICE: FIO: Петров
NOTICE: City: Лондон
Query returned successfully in 43 msec.
```

2. Написать хранимую процедуру СНК_ТОТ, которая получает идентификатор клиента и вычисляет общую стоимость его заказов (поле AMOUNT таблицы CUSTOMERS) и в зависимости от того, превысит ли эта сумма 30 000\$, заносит в поле STATUS таблицы OFFICES одно из двух примечаний – "большой объем заказов", "малый объем заказов".

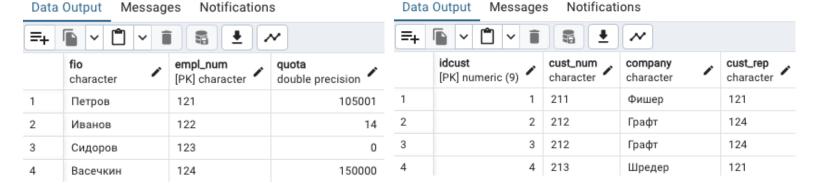
```
CREATE OR REPLACE PROCEDURE CHK TOT (
    id cust num CHAR(10),
    OUT sum salaes MONEY)
AS $$
BEGIN
 UPDATE offices
  SET status = CASE
  WHEN (SELECT SUM(amount)
        FROM customers
        WHERE cust num = id cust num) > CAST(30000 AS MONEY)
   THEN 'большой объем заказов'
  ELSE 'малый объем заказов'
  END
  WHERE cust num = id cust num;
   SELECT SUM (amount)
      INTO sum salaes
      FROM customers
      WHERE cust num = id cust num;
END;
$$ LANGUAGE plpgsql;
CALL CHK TOT ('213', NULL);
```



- Написать хранимую процедуру для добавления данных о новом клиенте в таблицу OFFICES.
- Добавить новую строку в таблицу OFFICES. Разрешается добовление только для клиентов, уже имеющих запись в таблице CUSTOMERS.
- Обновить запись в SALESPERS, увеличив поле QUOTA для соответствующего служащего (для каждого, с которым работает клиент) на величину объема продаж добавленной в предыдущем пункте записи. Плановый объем продаж служащего не может быть увеличен более чем на определенную величину. Если сумма заказов рассматриваемого клиента составляет менее 20 000\$ (вызов

процедуры СНК_ТОТ, которая должна возвращать эту сумму, например, в качестве выходного параметра), то величина объема продаж (поле TARGET таблицы OFFICES) будет добавлена к плану служащего. Если сумма заказов рассматриваемого клиента равна 20 000\$ к плану будут добавлены фиксированные 20 000\$. В противном случае — запретить добавление новой записи в OFFICES и обновление в SALESPERS. Предусмотреть промежуточный вывод суммы заказов рассматриваемого клиента.

```
CREATE OR REPLACE PROCEDURE add cust (
   idoff cust INTEGER,
   target cust DOUBLE PRECISION,
   city cust CHAR(10),
   cust num cust CHAR(10)
)
AS $$
DECLARE
   sum sales cust MONEY;
BEGIN
   IF EXISTS (SELECT 1 FROM customers WHERE cust num = cust num cust) THEN
       INSERT INTO offices
       VALUES (idoff cust, target cust, city cust, cust num cust);
   CALL CHK TOT(cust num cust, sum sales cust);
   UPDATE salespers
   SET QUOTA = CASE
   WHEN sum sales cust < 20000::MONEY
     THEN QUOTA + target cust
   WHEN sum sales cust = 20000::MONEY
     THEN QUOTA + 20000
   END
   WHERE empl num IN
        (SELECT cust rep FROM customers WHERE cust num = cust num cust);
$$ LANGUAGE plpgsql;
CALL add cust(5, 1, 'Минск', '211');
```



4. Создать триггер, распространяющий любое обновление столбца EMPL_NUM в таблице SALESPERS на столбец CUST REP таблицы CUSTOMERS.

```
CREATE OR REPLACE FUNCTION update_customers()
RETURNS TRIGGER
AS $$
BEGIN
     UPDATE customers
     SET cust_rep = NEW.empl_num
     WHERE cust_rep = OLD.empl_num;
     RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_cust_rep_trigger
AFTER UPDATE OF empl_num ON salespers
FOR EACH ROW
EXECUTE FUNCTION update_customers();
```

 Создать триггер, который запускается каждый раз, когда запись вставляется в таблицу CUSTOMERS или модифицируется. Если заказ сделан не в первые 15 дней месяца, то запись не принимается.

```
CREATE OR REPLACE FUNCTION check_order_date()
RETURNS TRIGGER AS $$
BEGIN
    IF EXTRACT(DAY FROM NEW.data_order) > 15 THEN
        RAISE EXCEPTION 'Forbidden date value!';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

CREATE TRIGGER order_date_trigger
BEFORE INSERT OR UPDATE ON customers
FOR EACH ROW
EXECUTE FUNCTION check_order_date();