

## 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 $$  
DECLARE  
    v_company CHAR(20);  
    v_fio CHAR(20);  
    v_city CHAR(10);  
BEGIN  
    CALL GET_CUST(cust_id, v_company, v_fio, v_city);  
    RAISE NOTICE 'Company: %', v_company;  
    RAISE NOTICE 'FIO: %', v_fio;  
    RAISE NOTICE 'City: %', v_city;  
END;  
$$ LANGUAGE plpgsql;  
  
CALL GET_CUST1('1');
```




Data Output	Messages	Notifications
-------------	----------	---------------

NOTICE:	Company: Фишер	
NOTICE:	FIO: Петров	
NOTICE:	City: Лондон	
CALL		

Query returned successfully in 43 msec.

2. Написать хранимую процедуру CHK\_TOT, 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);
```

Data Output		Mess:
<div><div><div>≡+</div><div></div><div>▼</div><div></div><div>▼</div></div></div>		
	sum_salaes money	
1	\$2,000.00	

3. Написать хранимую процедуру для добавления данных о новом клиенте в таблицу OFFICES.

- Добавить новую строку в таблицу OFFICES. Разрешается добавление 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 CHK\_TOT, 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 $$  
DECLARE  
    sum_sales_cust MONEY;  
BEGIN  
    IF EXISTS (SELECT 1 FROM customers WHERE cust_num = cust_num_cust) THEN  
        INSERT INTO offices  
        VALUES (idoff_cust, target_cust, city_cust, cust_num_cust);  
    END IF;  
    CALL CHK_TOT(cust_num_cust, sum_sales_cust);  
    UPDATE salespers  
    SET QUOTA = CASE  
        WHEN sum_sales_cust < 20000::MONEY  
        THEN QUOTA + target_cust  
        WHEN sum_sales_cust = 20000::MONEY  
        THEN QUOTA + 20000  
    END  
    WHERE empl_num IN  
        (SELECT cust_rep FROM customers WHERE cust_num = cust_num_cust);  
END;  
$$ LANGUAGE plpgsql;  
  
CALL add_cust(5, 1, 'Минск', '211');
```

	fio character	empl_num [PK] character	quota double precision
1	Петров	121	105001
2	Иванов	122	14
3	Сидоров	123	0
4	Васечкин	124	150000

	idcust [PK] numeric (9)	cust_num character	company character	cust_rep character
1	1	211	Фишер	121
2	2	212	Графт	124
3	3	212	Графт	124
4	4	213	Шредер	121

**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();
```

**5. 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();
```