

**Лабораторные работы по курсу
Базы данных**

**Лабораторная работа 6
«Процедуры и функции»**

Москва, 2023

Оглавление

1. Теоретическая часть	3
1.1. Хранимые подпрограммы	3
1.1.1. Процедуры	3
1.1.2. Функции	4
2. Практическая часть	6
2.1. Задание 1	6
2.2. Задание 2	6
Список литературы	6

1. Теоретическая часть

1.1.Хранимые подпрограммы

При выполнении запросов к базе данных происходит пересылка данных по сети от приложения клиента к серверу баз данных. В некоторых случаях, данная операция является затратной по времени выполнения. Для того, чтобы выполнять запросы более эффективно, возможно создать отдельную подпрограмму, которая будет выполняться в рамках процессов сервера баз данных. В СУБД PostgreSQL существуют два типа подобных подпрограмм – процедуры и функции. Их код может быть написан как на языке SQL, так и на других языках программирования – C, PL/pgSQL, Python, Tcl, Perl, R, Java, JavaScript и др. В данном лабораторном практикуме будут рассмотрены подпрограммы, написанные на языке SQL.

Функции и процедуры имеют несколько особенностей, отличающих их друг от друга. Приведем их сравнение в виде таблицы.

Функция	Процедура
Имеет возвращаемый тип и возвращает значение	Не имеет возвращаемого типа
Использование запросов на добавление, изменение и удаление строк невозможно . Разрешены только SELECT-запросы	Использование запросов на добавление, изменение и удаление строк возможно
Не имеет выходных аргументов	Имеет входные и выходные аргументы
Использовать транзакции запрещено	Возможно использовать операции управлением транзакциями

Рассмотрим их более подробно.

1.1.1. Процедуры

Процедуры – подпрограммы, которые не могут возвращать значения. Чаще всего процедуры используются для модификации данных в таблице – добавление, изменение или удаление.

Для создания процедуры существует команда CREATE PROCEDURE. Её сокращенный синтаксис приведен ниже.

```
CREATE [ OR REPLACE ] PROCEDURE
    имя ( [ [ режим_аргумента ] [ имя_аргумента ] тип_аргумента [ { DEFAULT |
= } выражение_по_умолчанию ] [, ...] ] )
LANGUAGE имя_языка
AS $$
...
$$;
```

Рассмотрим несколько примеров. Создадим процедуру для добавления студента в базу данных.

```
CREATE PROCEDURE add_Student(Student_ID bigint, datebegin date, dateend date,
F VARCHAR(30), I VARCHAR(30), O VARCHAR(30), groupe VARCHAR(7) ,birthday
date, email VARCHAR(30))
LANGUAGE SQL
AS $$
INSERT INTO "Студент" VALUES (Student_ID, F, I, O, groupe, birthday, email);
INSERT INTO "Студенческий билет" VALUES (Student_ID, datebegin, dateend);
$$;
```

После ключевых слов CREATE PROCEDURE следует название процедуры. Далее в скобках указываются входные параметры, в виде «параметр тип». В теле процедуры указывается язык, на котором она написана и непосредственно сам её код.

Приведем еще один пример. Аналогично создадим процедуру, позволяющую удалять записи о студенте из базы данных.

```
CREATE PROCEDURE del_Student(Student_ID bigint)
LANGUAGE SQL
AS $$
DELETE FROM "Студенческий билет" WHERE "Номер" = Student_ID;
DELETE FROM "Студент" WHERE "Номер студенческого билета" = Student_ID;
DELETE FROM "Результат освоения дисциплины" WHERE "Студент" = Student_ID;
$$;
```

1.1.2. Функции

Функции – подпрограммы, которые могут возвращать значения. Сокращенный синтаксис функции представлен ниже.

```
CREATE [ OR REPLACE ] FUNCTION
    имя ( [ [ режим_аргумента ] [ имя_аргумента ] тип_аргумента [ { DEFAULT |
= } выражение_по_умолчанию ] [, ...] ] )
    [ RETURNS тип_результата
    | RETURNS TABLE ( имя_столбца тип_столбца [, ...] ) ]
    { LANGUAGE имя_языка
    }
```

В зависимости от количества возвращаемых значений функции могут быть скалярными и составными.


Скалярные функции – функции, возвращающие одно значение базового типа, например integer.

Приведем пример скалярной функции, возвращающей значение даты окончания действия студенческого билета по его номеру.

```
CREATE OR REPLACE FUNCTION find_date(Student_ID bigint) RETURNS date
LANGUAGE SQL
AS $$
SELECT "Действителен по"
FROM "Студенческий билет"
WHERE "Номер" = Student_ID
$$;
```

Вызов скалярной функции происходит с помощью запроса SELECT:

```
SELECT find_date(841576)
```

	find_date 
1	2024-08-31

Составные функции

Функция с PostgreSQL может возвращать набор некоторых значений. Далее к этой функции возможно обращаться, как к таблице и выбирать из нее данные. Для того, чтобы она возвращало множество значений, необходимо определить **составной тип**. Составной тип очень похож на структуру в языке C. Он состоит из списка имён полей и соответствующих им типов данных. В общем виде, синтаксис можно представить следующим образом:

```
CREATE TYPE название AS (
    переменная тип,
    переменная тип,
    ...
);
```

Приведем пример следующей задачи. Необходимо написать функцию, которая выводит информацию о студентах, чей студенческий билет закончил действовать до указанной даты. Для этого создадим составной тип `Student_ID_date`.

```
CREATE TYPE Student_ID_date AS (  
    Фамилия varchar(30),  
    Имя varchar(30),  
    Дата date  
);
```

Далее создадим составную функцию `find_id_date`. Её основное отличие в синтаксисе от скалярной заключается в том, что функция возвращает тип `Student_ID_date` с использованием ключевого слова `SETOF`.

```
CREATE OR REPLACE FUNCTION find_id_date(id_date date) RETURNS SETOF  
Student_ID_date  
LANGUAGE SQL  
AS $$  
SELECT "Фамилия", "Имя", "Действителен по" AS "Учебный курс"  
FROM "Студенческий билет"  
INNER JOIN "Студент" ON "Студент"."Номер студенческого билета" =  
"Студенческий билет"."Номер"  
WHERE "Действителен по" < id_date  
$$;
```

Вызов функции будет совпадать с обращением к таблице:

```
SELECT * FROM find_id_date('10/09/2024')
```

	Фамилия character varying (30)	Имя character varying (30)	Дата date
1	Дроздов	Марк	2024-08-31
2	Самсонов	Максим	2024-08-31
3	Смирнов	Ярослав	2024-08-31
4	Соловьев	Сергей	2024-08-31
5	Селиванов	Дмитрий	2024-08-31
6	Егоров	Артём	2024-08-31

2. Практическая часть

2.1. Задание 1.

Разработайте 5 **осмысленных** процедур к учебной базе данных.

2.2. Задание 2.

Разработайте 5 **осмысленных** функций к учебной базе данных. 3 из них должны быть скалярными, 2 – составными.

Список литературы

- [1] «Исходный код СУБД postgres,» [В Интернете]. Available: <https://github.com/postgres/postgres>. [Дата обращения: 30 01 2023].
- [2] Документация к PostgreSQL 15.1, 2022.
- [3] Е. Рогов, PostgreSQL изнутри, 1-е ред., Москва: ДМК Пресс, 2023, р. 662.
- [4] Б. А. Новиков, Е. А. Горшкова и Н. Г. Графеева, Основы технологии баз данных, 2-е ред., Москва: ДМК пресс, 2020, р. 582.
- [5] Е. П. Моргунов, PostgreSQL. Основы языка SQL, 1-е ред., Санкт-Петербург: БХВ-Петербург, 2018, р. 336.