|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 - ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА**

**Отчет**

**по лабораторной работе № 3**

**Вариант №1**

**Название:** Пользовательские функции и процедуры

**Дисциплина:** Базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-31Б |  |  | Г.А. Артемьев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.П. Кудрявцев |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

**Цель -** данная лабораторная работа призвана сформировать у студента понимание назначения пользовательских функции и процедур, их написание и использование.

**Задачи:**

* Получить теоретические знания о назначении функций и процедур БД.
* Изучить синтаксис функций и процедур.
* Научиться добавлять функции и процедуры в БД.
* Научиться удалять и изменять функции и процедуры.
* Научится использовать функции и процедуры.

**Задание:**

Написать хранимые функции и процедуры, выполняющие следующие операции для своей предметной области.

* Добавление одной записи, только через вызов процедуры, для таблиц *Книга, Выдача, Читатель*. Проверять там, где надо, на дублирование записей с выдачей сообщения об ошибке. Проверять на корректность ввода данных, например, даты выдачи позже даты возврата.
* Изменение одной записи, только через вызов процедуры, для таблиц *Книга, Выдача, Читатель*. Проверять там, где надо, на дублирование записей с выдачей сообщения об ошибке.
* Удаление одной записи только, через вызов процедуры, для таблиц *Книга, Выдача, Читатель*. Проверять на возможность удаления. Например, удаляем книгу, которая находится на руках у читателя (не допускать формирование исключения о нарушении целостности БД).
* Запрос читателей по первичному ключу, по адресу проживания, по выданной книге.
* Запрос списка книг, выданных заданному читателю.

**Добавление записей**

**Процедура для добавления книги**

CREATE OR REPLACE PROCEDURE add\_book(

p\_title VARCHAR(255),

p\_author VARCHAR(255),

p\_publisher VARCHAR(255),

p\_theme VARCHAR(100)

)

LANGUAGE plpgsql

AS $$

BEGIN

IF EXISTS (

SELECT \* FROM Book

WHERE title = p\_title AND author = p\_author AND publisher = p\_publisher

) THEN

RAISE EXCEPTION 'Книга уже существует';

END IF;

INSERT INTO Book (title, author, publisher, theme)

VALUES (p\_title, p\_author, p\_publisher, p\_theme);

END;

$$;

**Процедура для добавления выдачи**

CREATE OR REPLACE PROCEDURE add\_distribution(

p\_book\_id INT,

p\_reader\_id INT,

p\_issue\_date DATE,

p\_due\_date DATE

)

LANGUAGE plpgsql

AS $$

BEGIN

IF NOT EXISTS (SELECT 1 FROM Book WHERE id = p\_book\_id) THEN

RAISE EXCEPTION 'Книга с таким ID не найдена';

END IF;

IF NOT EXISTS (SELECT 1 FROM Reader WHERE id = p\_reader\_id) THEN

RAISE EXCEPTION 'Читатель с таким ID не найден';

END IF;

IF p\_due\_date < p\_issue\_date THEN

RAISE EXCEPTION 'Дата возврата не может быть раньше даты выдачи';

END IF;

INSERT INTO Distribution (book\_id, reader\_id, issue\_date, due\_date)

VALUES (p\_book\_id, p\_reader\_id, p\_issue\_date, p\_due\_date);

END;

$$;

**Процедура для добавления читателя**

CREATE OR REPLACE PROCEDURE add\_reader(

p\_fio VARCHAR(255),

p\_address VARCHAR(255),

p\_phone VARCHAR(20),

p\_passport\_number VARCHAR(50)

)

LANGUAGE plpgsql

AS $$

BEGIN

IF EXISTS (SELECT \* FROM Reader WHERE passport\_number = p\_passport\_number) THEN

RAISE EXCEPTION 'Читатель с таким номером паспорта уже существует';

END IF;

INSERT INTO Reader (fio, address, phone, passport\_number)

VALUES (p\_fio, p\_address, p\_phone, p\_passport\_number)

Returning(id into );

END;

$$;

**Изменение записей:**

**Процедура для изменения книги**

CREATE OR REPLACE PROCEDURE update\_book(

p\_book\_id INT,

p\_new\_title VARCHAR(255),

p\_new\_author VARCHAR(255),

p\_new\_publisher VARCHAR(255),

p\_new\_theme VARCHAR(100)

)

LANGUAGE plpgsql

AS $$

BEGIN

-- Проверка существования книги

IF NOT EXISTS (SELECT 1 FROM Book WHERE id = p\_book\_id) THEN

RAISE EXCEPTION 'Книга с таким ID не найдена';

END IF;

-- Обновление записи

UPDATE Book

SET title = p\_new\_title,

author = p\_new\_author,

publisher = p\_new\_publisher,

theme = p\_new\_theme

WHERE id = p\_book\_id;

END;

$$;

**Процедура для изменения выдачи**

CREATE OR REPLACE PROCEDURE update\_distribution(

p\_distribution\_id INT,

p\_new\_book\_id INT,

p\_new\_reader\_id INT,

p\_new\_issue\_date DATE,

p\_new\_due\_date DATE

)

LANGUAGE plpgsql

AS $$

BEGIN

-- Проверка существования выдачи

IF NOT EXISTS (SELECT 1 FROM Distribution WHERE id = p\_distribution\_id) THEN

RAISE EXCEPTION 'Запись о выдаче с таким ID не найдена';

END IF;

-- Проверка на корректность дат

IF p\_new\_due\_date < p\_new\_issue\_date THEN

RAISE EXCEPTION 'Дата возврата не может быть раньше даты выдачи';

END IF;

-- Обновление записи

UPDATE Distribution

SET book\_id = p\_new\_book\_id,

reader\_id = p\_new\_reader\_id,

issue\_date = p\_new\_issue\_date,

due\_date = p\_new\_due\_date

WHERE id = p\_distribution\_id;

END;

$$;;

**Удаление записей:**

**Процедура для удаления книги**

CREATE OR REPLACE PROCEDURE delete\_book(p\_book\_id INT)

LANGUAGE plpgsql

AS $$

BEGIN

-- Проверка на наличие связанных записей

IF EXISTS (SELECT 1 FROM Distribution WHERE book\_id = p\_book\_id) THEN

RAISE EXCEPTION 'Нельзя удалить книгу, так как она выдана читателю';

END IF;

DELETE FROM Book WHERE id = p\_book\_id;

END;

$$;

**Процедура для удаления выдачи**

CREATE OR REPLACE PROCEDURE delete\_distribution(p\_distribution\_id INT)

LANGUAGE plpgsql

AS $$

BEGIN

IF NOT EXISTS (SELECT 1 FROM Distributio WHERE id = p\_distribution\_id) THEN

RAISE EXCEPTION 'Запись о выдаче с таким ID не найдена';

END IF;

DELETE FROM Distribution WHERE id = p\_distribution\_id;

END;

$$;

**Запросы**

**Получение информации о читателях по адресу**

CREATE OR REPLACE FUNCTION get\_readers\_by\_address(p\_address VARCHAR(255))

RETURNS TABLE (

id INT,

fio VARCHAR(255),

phone VARCHAR(20)

)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT id, fio, phone

FROM Reader

WHERE address = p\_address;

END;

$$;

**Получение списка книг, выданных заданному читателю**

CREATE OR REPLACE FUNCTION get\_books\_by\_reader(p\_reader\_id INT)

RETURNS TABLE (

book\_id INT,

title VARCHAR(255),

issue\_date DATE,

due\_date DATE

)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT b.id, b.title, d.issue\_date, d.due\_date

FROM Distribution d

JOIN Book b ON d.book\_id = b.id

WHERE d.reader\_id = p\_reader\_id;

END;

$$;

**Примеры вызова**

**Добавление:**

CALL add\_book('Книга 1', 'Автор 1', 'Издательство 1', 'Тема 1');

CALL add\_distribution(1, 1, '2024-01-01', '2024-01-10');

CALL add\_reader('Иван Иванов', 'ул. Ленина, 1', '1234567890', '9876543210');

**Изменение:**

CALL update\_book(1, 'Новое название', 'Новый автор', 'Новое издательство', 'Новая тема');

CALL update\_distribution(1, 1, 1, '2024-01-05', '2024-01-15');

**Удаление:**

CALL delete\_book(1);

CALL delete\_distribution(1);

**Запросы:**

SELECT \* FROM get\_readers\_by\_address('ул. Ленина, 1');

SELECT \* FROM get\_books\_by\_reader(1);

**Вывод:** в ходе лабораторной работы были написаны процедуры и функции по условиям заданий по варианту предметной области “Кооперативы”