МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра программных систем

**ОТЧЕТ**  
по лабораторному практикуму по дисциплине

«Распределённые БД»

Лабораторная работа № 4

Обучающийся группы 6131-020402D В.Д. Гижевская

Обучающийся группы 6131-020402D Д.А. Кремнёв

Обучающийся группы 6131-020402D М.Г. Алкеев

Проверил М.В. Додонов

Самара 2023

ЗАдание

Тема работы: создание и использование триггеров.

Цель работы: знакомство с возможностями СУБД по созданию триггеров.

Подготовка к работе:

1. Изучите теоретические основы, команды создания последовательностей и триггеров,

рассмотренные в лекционном курсе, методических указаниях или других источниках.

2. Ответьте на вопросы к лабораторной работе.

3. Подготовьте отчет к оформлению хода выполнения работы.

Порядок выполнения работы:

1. Создайте необходимые последовательности для суррогатных первичных ключей (Oracle).

2. Создайте триггеры вставки и изменения записей таблиц, определяющие значение суррогатного первичного ключа по умолчанию.

3. Создайте триггер удаления записей из таблиц, удаляющие связанные записи из дочерних таблиц.

4. Внесите такие изменения в триггеры вставки и изменения записей таблиц, которые не позволят добавить или изменить записи с дублирующими названиями.

5. Создайте триггеры для необновляемых представлений, позволяющие изменять данные.

6. Оформите отчет о выполнении лабораторной работы.

вариант

Предметная область: Книжный сайт.

Основные предметно-значимые сущности: Серии книг, Авторы, Книги, Главы.

Основные предметно-значимые атрибуты сущностей:

* серии книг – название серии книг;
* авторы – ФИО автора;
* книги – название книги, описание книги, ключевые слова;
* главы – название главы, время чтения, количество страниц.

Основные требования к функциям системы:

* выбрать все книги по определённому автору;
* выбрать все главы определённой книги;
* вывести рейтинг книг по времени чтения;
* вывести все книги определённой серии;
* вывести все книги по определённому ключевому слову;
* подсчитать количество страниц нужной книги;
* подсчитать количество страниц нужной серии;
* подсчитать время чтения нужной книги;
* подсчитать время чтения выбранной серии.

1. Создайте необходимые последовательности для суррогатных первичных ключей

**Создание последовательности для таблицы authors:**

CREATE SEQUENCE "authors\_id\_seq"

INCREMENT 1

START 1

MINVALUE 1

MAXVALUE 2147483647

CACHE 1

OWNED BY "authors".id\_author;

**Создание последовательности для таблицы** **book\_series:**

CREATE SEQUENCE "book\_series\_id\_seq"

INCREMENT 1

START 1

MINVALUE 1

MAXVALUE 2147483647

CACHE 1

OWNED BY "book\_series".id\_book\_series;

**Создание последовательности для таблицы** **books:**

CREATE SEQUENCE "books\_id\_seq"

INCREMENT 1

START 1

MINVALUE 1

MAXVALUE 2147483647

CACHE 1

OWNED BY "books".id\_book;

**Создание последовательности для таблицы** **books:**

CREATE SEQUENCE "chapters\_id\_seq"

INCREMENT 1

START 1

MINVALUE 1

MAXVALUE 2147483647

CACHE 1

OWNED BY "chapters".id\_chapter;

1. Создайте триггеры вставки и изменения записей таблиц, определяющие значение суррогатного первичного ключа по умолчанию

**Создание триггерной функции и триггера для таблицы authors:**

CREATE FUNCTION public.authors\_before\_add()

RETURNS trigger

LANGUAGE 'plpgsql'

NOT LEAKPROOF

AS $BODY$

BEGIN

NEW.id\_author := nextval('"authors\_id\_seq"');

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.authors\_before\_add()

OWNER TO postgres;

CREATE TRIGGER authors\_before\_add\_trigger

BEFORE INSERT OR UPDATE

ON authors

FOR EACH ROW

EXECUTE FUNCTION public.authors\_before\_add();

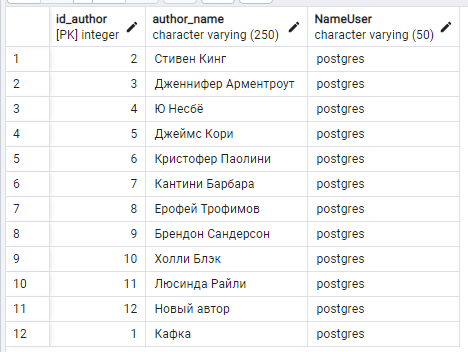
**Проверка работы триггера:**

INSERT INTO authors(

"author\_name", "NameUser")

VALUES ('Кафка', DEFAULT);

SELECT \* FROM authors



**Создание триггерной функции и триггера для таблицы authors:**

CREATE FUNCTION public.book\_series\_before\_add()

RETURNS trigger

LANGUAGE 'plpgsql'

NOT LEAKPROOF

AS $BODY$

BEGIN

NEW.id\_book\_series := nextval('"book\_series\_id\_seq"');

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.book\_series\_before\_add()

OWNER TO postgres;

CREATE TRIGGER book\_series\_before\_add\_trigger

BEFORE INSERT OR UPDATE

ON public.book\_series

FOR EACH ROW

EXECUTE FUNCTION public.book\_series\_before\_add();

**Проверка работы триггера:**

INSERT INTO public.book\_series(

"id\_author", "book\_series\_name", "NameUser")

VALUES ('3', 'Сталкер', DEFAULT);

SELECT \* FROM book\_series;



**Создание триггерной функции и триггера для таблицы books:**

CREATE FUNCTION public.books\_before\_add()

RETURNS trigger

LANGUAGE 'plpgsql'

NOT LEAKPROOF

AS $BODY$

BEGIN

NEW.id\_book := nextval('"books\_id\_seq"');

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.books\_before\_add()

OWNER TO postgres;

CREATE TRIGGER books\_before\_add\_trigger

BEFORE INSERT OR UPDATE

ON public.books

FOR EACH ROW

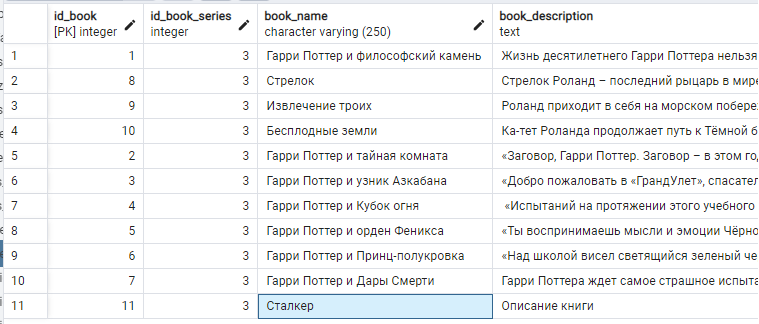
EXECUTE FUNCTION public.books\_before\_add();

**Проверка работы триггера:**

INSERT INTO public.books("id\_book\_series", "book\_name", "book\_description", "key\_words", "NameUser")

VALUES ('3', 'Сталкер', 'Описание книги', 'Ключевые слова', DEFAULT);

SELECT \* FROM books;



**Создание триггерной функции и триггера для таблицы chapters:**

CREATE FUNCTION public.chapters\_before\_add()

RETURNS trigger

LANGUAGE 'plpgsql'

NOT LEAKPROOF

AS $BODY$

BEGIN

NEW.id\_chapter := nextval('"chapters\_id\_seq"');

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.chapters\_before\_add()

OWNER TO postgres;

CREATE TRIGGER chapters\_before\_add\_trigger

BEFORE INSERT OR UPDATE

ON public.chapters

FOR EACH ROW

EXECUTE FUNCTION public.chapters\_before\_add();

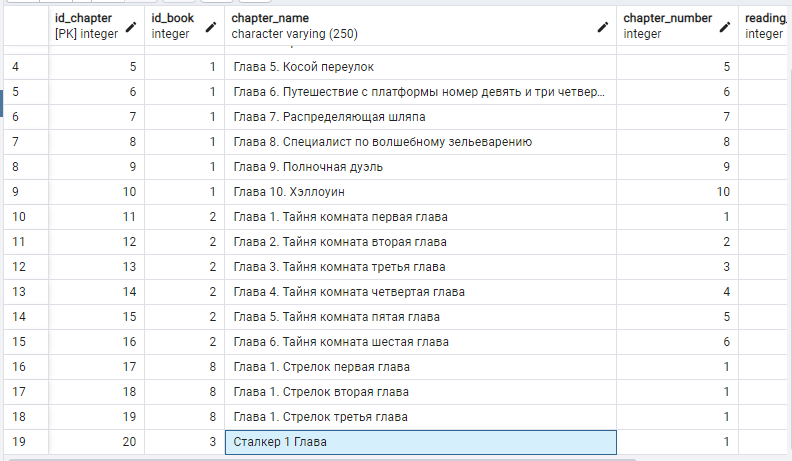
**Проверка работы триггера:**

INSERT INTO public.chapters(

"id\_book", "chapter\_name", "chapter\_number", "reading\_time", "pages\_number", "NameUser")

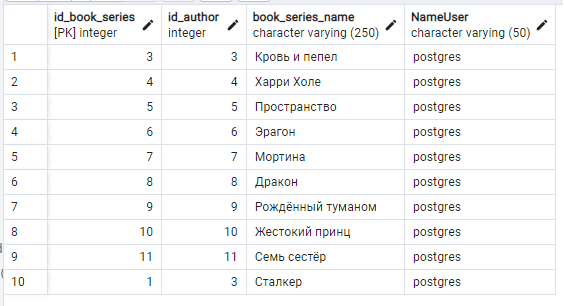
VALUES ('3', 'Сталкер 1 Глава', 1, 30, 60, DEFAULT);

SELECT \* FROM chapters;

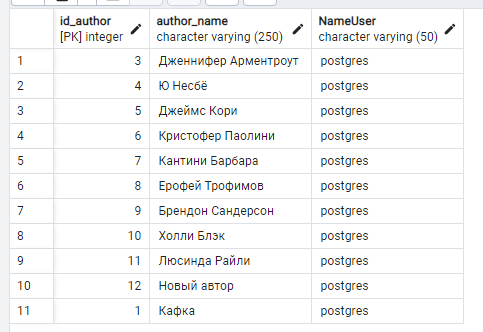


1. Создайте триггер удаления записей из таблиц, удаляющие связанные записи из дочерних таблиц

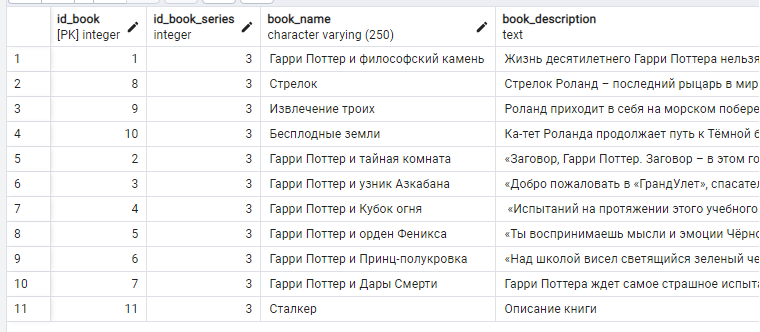
**Таблица book\_series до удаления:**



**Таблица authors до удаления:**



**Таблица books до удаления:**



**Триггеры:**

CREATE OR REPLACE FUNCTION delete\_books\_on\_book\_series\_delete()

RETURNS TRIGGER AS $$

BEGIN

DELETE FROM books WHERE id\_book\_series = OLD.id\_book\_series;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER books\_before\_delete

BEFORE DELETE ON book\_series

FOR EACH ROW

EXECUTE FUNCTION delete\_books\_on\_book\_series\_delete();

CREATE OR REPLACE FUNCTION delete\_book\_series\_on\_author\_delete()

RETURNS TRIGGER AS $$

BEGIN

DELETE FROM book\_series WHERE id\_author = OLD.id\_author;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER book\_series\_before\_delete

BEFORE DELETE ON authors

FOR EACH ROW

EXECUTE FUNCTION delete\_book\_series\_on\_author\_delete();

CREATE OR REPLACE FUNCTION delete\_chapters\_on\_book\_delete()

RETURNS TRIGGER AS $$

BEGIN

DELETE FROM chapters WHERE id\_book = OLD.id\_book;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER chapters\_before\_delete

BEFORE DELETE ON books

FOR EACH ROW

EXECUTE FUNCTION delete\_chapters\_on\_book\_delete();

**Выполним удаление:**

DELETE FROM authors WHERE id\_author = 3;

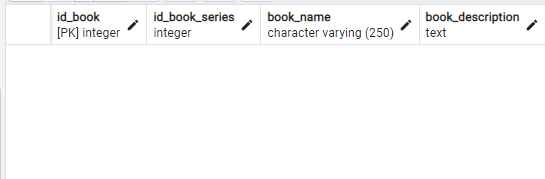
**Таблица book\_series после удаления:**



**Таблица authors после удаления:**



**Таблица books после удаления:**



1. Внесите такие изменения в триггеры вставки и изменения записей таблиц, которые не позволят добавить или изменить записи с дублирующими названиями

**Триггер для таблицы authors:**

CREATE OR REPLACE FUNCTION authors\_unique\_name() RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT 1 FROM Authors WHERE author\_name = NEW.author\_name AND id\_author <> NEW.id\_author) THEN

RAISE EXCEPTION 'Author with name "%" already exists', NEW.author\_name;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER authors\_unique\_name

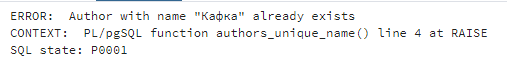
BEFORE INSERT OR UPDATE ON Authors

FOR EACH ROW

EXECUTE FUNCTION authors\_unique\_name();

**Запрос:**  
INSERT INTO public.authors("author\_name") VALUES ('Кафка');

**Результат:**



1. Создайте триггеры для необновляемых представлений, позволяющие изменять данные

**Создадим необновляемое представление:**

CREATE VIEW public."authors\_readonly\_view"

AS

SELECT subquery.id\_author,

subquery."author\_name"

FROM ( SELECT "authors".id\_author,

"authors"."author\_name"

FROM "authors") subquery;

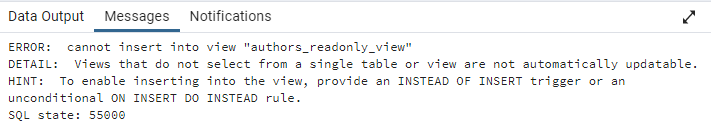
ALTER TABLE public."authors\_readonly\_view"

OWNER TO postgres;

**Добавление новой записи невозможно:**

INSERT INTO public."authors\_readonly\_view"("author\_name")

VALUES ('Автор');



**Создадим триггре-функцию:**

CREATE FUNCTION public.authors\_readonly\_view\_f()

RETURNS trigger

LANGUAGE 'plpgsql'

NOT LEAKPROOF

AS $BODY$

BEGIN

IF (TG\_OP = 'DELETE') THEN

DELETE FROM "authors" WHERE "author\_name" = OLD."author\_name";

IF NOT FOUND THEN RETURN NULL; END IF;

RETURN OLD;

ELSIF (TG\_OP = 'UPDATE') THEN

UPDATE "authors" SET "author\_name" = NEW."author\_name"

WHERE "author\_name" = OLD."author\_name";

IF NOT FOUND THEN RETURN NULL; END IF;

RETURN NEW;

ELSIF (TG\_OP = 'INSERT') THEN

INSERT INTO "authors"("author\_name") VALUES(NEW."author\_name");

RETURN NEW;

END IF;

END;

$BODY$;

ALTER FUNCTION public.authors\_readonly\_view\_f()

OWNER TO postgres;

**Создадим триггер:**

CREATE TRIGGER authors\_readonly\_view\_trigger\_f

INSTEAD OF INSERT OR DELETE OR UPDATE

ON public."authors\_readonly\_view"

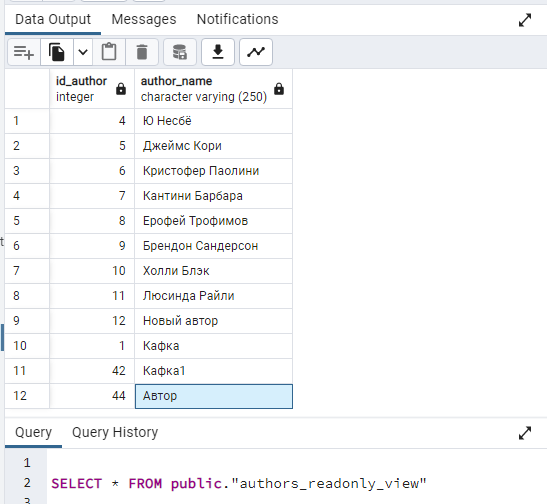
FOR EACH ROW

EXECUTE FUNCTION public.authors\_readonly\_view\_f();

**Проверка INSERT:**

INSERT INTO public."authors\_readonly\_view"("author\_name")

VALUES ('Автор');

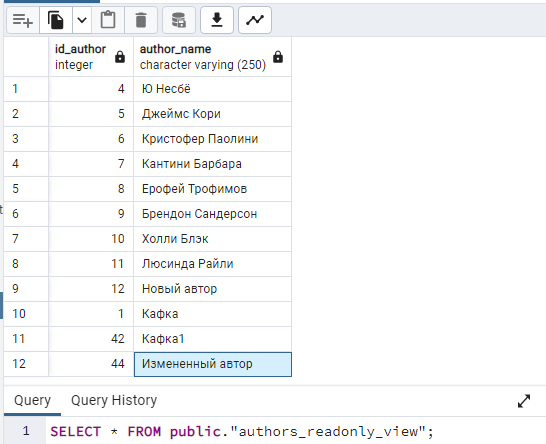


**Проверка UPDATE:**

UPDATE public."authors\_readonly\_view"

SET "author\_name"= 'Измененный автор'

WHERE "author\_name"= 'Автор';



**Проверка DELETE:**

DELETE FROM public."authors\_readonly\_view"

WHERE "author\_name"= 'Измененный автор';

