--===============================СОЗДАНИЕ ТАБЛИЦ====================================

-- Создание таблицы "Publishing\_Companies" для хранения информации об издательствах

CREATE TABLE IF NOT EXISTS public.publishing\_companies

(

name VARCHAR NOT NULL,

establishment\_year INTEGER,

contact\_info VARCHAR,

city VARCHAR,

CONSTRAINT pk\_publishing\_companies PRIMARY KEY (name)

)

TABLESPACE pg\_default;

-- Установка владельца таблицы "Publishing\_Companies" на пользователя "postgres"

ALTER TABLE IF EXISTS public.publishing\_companies

OWNER TO postgres;

-- Создание таблицы "Books" для хранения информации о книгах

CREATE TABLE IF NOT EXISTS public.books

(

isbn VARCHAR NOT NULL,

name VARCHAR,

publication\_year INTEGER,

age\_limit REAL,

publishing\_company VARCHAR,

page\_count INTEGER,

language VARCHAR,

cost REAL,

count\_of\_books INTEGER,

CONSTRAINT pk\_books PRIMARY KEY (isbn),

CONSTRAINT fk\_books\_publishing\_companies FOREIGN KEY (publishing\_company)

REFERENCES public.publishing\_companies (name)

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

-- Установка владельца таблицы "Books" на пользователя "postgres"

ALTER TABLE IF EXISTS public.books

OWNER TO postgres;

-- Создание таблицы "Authors" для хранения информации об авторах

CREATE TABLE IF NOT EXISTS public.authors

(

author\_id INTEGER NOT NULL,

fio VARCHAR,

birth\_date DATE,

country VARCHAR,

nickname VARCHAR,

CONSTRAINT pk\_authors PRIMARY KEY (author\_id)

)

TABLESPACE pg\_default;

-- Установка владельца таблицы "Authors" на пользователя "postgres"

ALTER TABLE IF EXISTS public.authors

OWNER TO postgres;

-- Создание таблицы "Authorships" для связывания авторов и книг (многие ко многим)

CREATE TABLE IF NOT EXISTS public.authorships

(

book\_isbn VARCHAR NOT NULL,

author\_id INTEGER NOT NULL,

CONSTRAINT pk\_authorships PRIMARY KEY (book\_isbn, author\_id),

CONSTRAINT fk\_authorships\_authors FOREIGN KEY (author\_id)

REFERENCES public.authors (author\_id)

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fk\_authorships\_books FOREIGN KEY (book\_isbn)

REFERENCES public.books (isbn)

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

-- Установка владельца таблицы "Authorships" на пользователя "postgres"

ALTER TABLE IF EXISTS public.authorships

OWNER TO postgres;

-- Создание таблицы "Books\_Styles" для связывания книг и жанров (многие ко многим)

CREATE TABLE IF NOT EXISTS public.books\_styles

(

book\_isbn VARCHAR NOT NULL,

style\_name VARCHAR NOT NULL,

CONSTRAINT pk\_books\_styles PRIMARY KEY (book\_isbn, style\_name),

CONSTRAINT fk\_books\_styles\_books FOREIGN KEY (book\_isbn)

REFERENCES public.books (isbn)

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

-- Установка владельца таблицы "Books\_Styles" на пользователя "postgres"

ALTER TABLE IF EXISTS public.books\_styles

OWNER TO postgres;

--===============================ХРАНИМЫЕ ПРОЦЕДУРЫ ДЛЯ РАБОТЫ С ТАБЛИЦАМИ====================================

-- Добавить издательство

CREATE OR REPLACE PROCEDURE add\_publishing\_company (

in\_publishing\_company\_name VARCHAR,

in\_establishment\_year INTEGER,

in\_contact\_info VARCHAR,

in\_city VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

IF in\_publishing\_company\_name IS NOT NULL THEN

INSERT INTO public.publishing\_companies

(

name,

establishment\_year,

contact\_info,

city

)

VALUES

(

in\_publishing\_company\_name,

in\_establishment\_year,

in\_contact\_info,

in\_city

);

ELSE

RAISE NOTICE 'Введите название издательства';

END IF;

END;

$$;

-- Добавить книгу

CREATE OR REPLACE PROCEDURE add\_book (

in\_book\_isbn VARCHAR,

in\_book\_name VARCHAR,

in\_book\_year INTEGER,

in\_book\_age\_limit REAL,

in\_publishing\_company VARCHAR,

in\_book\_page\_count INTEGER,

in\_language VARCHAR,

in\_cost REAL,

in\_count\_of\_books INTEGER

)

LANGUAGE plpgsql

AS $$

BEGIN

IF in\_book\_isbn IS NOT NULL THEN

INSERT INTO public.books

(

isbn,

name,

publication\_year,

age\_limit,

publishing\_company,

page\_count,

language,

cost,

count\_of\_books

)

VALUES

(

in\_book\_isbn,

in\_book\_name,

in\_book\_year,

in\_book\_age\_limit,

in\_publishing\_company,

in\_book\_page\_count,

in\_language,

in\_cost,

in\_count\_of\_books

);

ELSE

RAISE NOTICE 'Введите ISBN книги';

END IF;

END;

$$;

-- Назначить жанр книге

CREATE OR REPLACE PROCEDURE add\_style\_to\_book (

in\_book\_isbn VARCHAR,

in\_style\_name VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

IF EXISTS (SELECT 1 FROM public.books WHERE isbn = in\_book\_isbn) THEN

IF in\_style\_name IS NOT NULL THEN

INSERT INTO public.books\_styles (book\_isbn, style\_name)

VALUES (in\_book\_isbn, in\_style\_name);

ELSE

RAISE NOTICE 'Введите жанр книги';

END IF;

ELSE

RAISE NOTICE 'Книга с ISBN % не найдена', in\_book\_isbn;

END IF;

END;

$$;

-- Добавить автора

CREATE OR REPLACE PROCEDURE add\_author (

in\_author\_id INTEGER,

in\_author\_fio VARCHAR,

in\_birth\_date DATE,

in\_country VARCHAR,

in\_nickname VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

IF in\_author\_id IS NOT NULL THEN

INSERT INTO public.authors

(

author\_id,

fio,

birth\_date,

country,

nickname

)

VALUES

(

in\_author\_id,

in\_author\_fio,

in\_birth\_date,

in\_country,

in\_nickname

);

ELSE

RAISE NOTICE 'Введите ID автора';

END IF;

END;

$$;

-- Назначить автора книге

CREATE OR REPLACE PROCEDURE add\_author\_to\_book (

in\_author\_id INTEGER,

in\_book\_isbn VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

IF EXISTS (SELECT 1 FROM public.authors WHERE author\_id = in\_author\_id) THEN

IF EXISTS (SELECT 1 FROM public.books WHERE isbn = in\_book\_isbn) THEN

INSERT INTO public.authorships (book\_isbn, author\_id)

VALUES (in\_book\_isbn, in\_author\_id);

ELSE

RAISE NOTICE 'Книга с ISBN % не найдена', in\_book\_isbn;

END IF;

ELSE

RAISE NOTICE 'Автор с ID % не найден', in\_author\_id;

END IF;

END;

$$;

--===============================ХРАНИМЫЕ ПРОЦЕДУРЫ ДЛЯ УДАЛЕНИЯ ДАННЫХ====================================

-- Удалить издательство и связанные с ним данные

CREATE OR REPLACE PROCEDURE delete\_publishing\_company (

in\_publishing\_company\_name VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

-- Удаляем связи "автор–книга" для книг данного издательства

DELETE FROM public.authorships

WHERE book\_isbn IN (

SELECT isbn FROM public.books

WHERE publishing\_company = in\_publishing\_company\_name

);

-- Удаляем жанры книг данного издательства

DELETE FROM public.books\_styles

WHERE book\_isbn IN (

SELECT isbn FROM public.books

WHERE publishing\_company = in\_publishing\_company\_name

);

-- Удаляем сами книги издательства

DELETE FROM public.books

WHERE publishing\_company = in\_publishing\_company\_name;

-- Удаляем издательство

DELETE FROM public.publishing\_companies

WHERE name = in\_publishing\_company\_name;

END;

$$;

-- Удалить автора книги

CREATE OR REPLACE PROCEDURE delete\_author\_of\_book (

in\_author\_id INTEGER,

in\_book\_isbn VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.authorships

WHERE author\_id = in\_author\_id

AND book\_isbn = in\_book\_isbn;

END;

$$;

-- Удалить неиспользуемых авторов (авторов, не связанных ни с одной книгой)

CREATE OR REPLACE PROCEDURE clear\_non\_used\_authors\_of\_book()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.authors

WHERE author\_id NOT IN (SELECT author\_id FROM public.authorships);

END;

$$;

-- Удалить книгу и связанные с ней данные

CREATE OR REPLACE PROCEDURE delete\_book (

in\_book\_isbn VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.authorships

WHERE book\_isbn = in\_book\_isbn;

DELETE FROM public.books\_styles

WHERE book\_isbn = in\_book\_isbn;

DELETE FROM public.books

WHERE isbn = in\_book\_isbn;

END;

$$;

-- Удалить жанр книги

CREATE OR REPLACE PROCEDURE delete\_book\_style (

in\_book\_isbn VARCHAR,

in\_style\_name VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.books\_styles

WHERE book\_isbn = in\_book\_isbn

AND style\_name = in\_style\_name;

END;

$$;

-- Удалить автора книги

CREATE OR REPLACE PROCEDURE delete\_book\_author (

in\_book\_isbn VARCHAR,

in\_author\_id INTEGER

)

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.authorships

WHERE book\_isbn = in\_book\_isbn

AND author\_id = in\_author\_id;

END;

$$;

--===============================ХРАНИМЫЕ ПРОЦЕДУРЫ ДЛЯ ОБНОВЛЕНИЯ ДАННЫХ====================================

-- Изменить информацию об издательстве

CREATE OR REPLACE PROCEDURE update\_publishing\_company(

in\_old\_name VARCHAR,

in\_establishment\_year INTEGER,

in\_contact\_info VARCHAR,

in\_city VARCHAR,

in\_new\_name VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

IF in\_old\_name IS NOT NULL THEN

IF in\_new\_name IS NOT NULL THEN

IF in\_old\_name != in\_new\_name THEN

-- Добавляем новую запись под другим именем

INSERT INTO public.publishing\_companies

(

name,

establishment\_year,

contact\_info,

city

)

VALUES

(

in\_new\_name,

in\_establishment\_year,

in\_contact\_info,

in\_city

);

-- Перевязываем книги на новое издательство

UPDATE public.books

SET publishing\_company = in\_new\_name

WHERE publishing\_company = in\_old\_name;

-- Удаляем старую запись

DELETE FROM public.publishing\_companies

WHERE name = in\_old\_name;

ELSE

-- Если название не изменилось, обновляем только поля

UPDATE public.publishing\_companies

SET

establishment\_year = in\_establishment\_year,

contact\_info = in\_contact\_info,

city = in\_city

WHERE name = in\_old\_name;

END IF;

ELSE

RAISE NOTICE 'Введите новое название издательства';

END IF;

ELSE

RAISE NOTICE 'Введите старое название издательства';

END IF;

END;

$$;

-- Изменить информацию о книге

CREATE OR REPLACE PROCEDURE update\_book(

in\_old\_isbn VARCHAR,

in\_name VARCHAR,

in\_publication\_year INTEGER,

in\_age\_limit REAL,

in\_page\_count INTEGER,

in\_language VARCHAR,

in\_cost REAL,

in\_count\_of\_books INTEGER,

in\_new\_isbn VARCHAR,

in\_publishing\_company VARCHAR

)

LANGUAGE plpgsql

AS $$

BEGIN

IF in\_old\_isbn IS NOT NULL THEN

IF in\_new\_isbn IS NOT NULL THEN

IF in\_old\_isbn != in\_new\_isbn THEN

-- Создаём новую запись книги

INSERT INTO public.books

(

isbn,

name,

publication\_year,

age\_limit,

publishing\_company,

page\_count,

language,

cost,

count\_of\_books

)

VALUES

(

in\_new\_isbn,

in\_name,

in\_publication\_year,

in\_age\_limit,

in\_publishing\_company,

in\_page\_count,

in\_language,

in\_cost,

in\_count\_of\_books

);

-- Перевязываем стили (жанры)

UPDATE public.books\_styles

SET book\_isbn = in\_new\_isbn

WHERE book\_isbn = in\_old\_isbn;

-- Перевязываем авторство

UPDATE public.authorships

SET book\_isbn = in\_new\_isbn

WHERE book\_isbn = in\_old\_isbn;

-- Удаляем старую книгу

DELETE FROM public.books

WHERE isbn = in\_old\_isbn;

ELSE

-- Если ISBN не изменился, обновляем только поля

UPDATE public.books

SET

name = in\_name,

publication\_year = in\_publication\_year,

age\_limit = in\_age\_limit,

publishing\_company = in\_publishing\_company,

page\_count = in\_page\_count,

language = in\_language,

cost = in\_cost,

count\_of\_books = in\_count\_of\_books

WHERE isbn = in\_old\_isbn;

END IF;

ELSE

RAISE NOTICE 'Введите новый ISBN книги';

END IF;

ELSE

RAISE NOTICE 'Введите старый ISBN книги';

END IF;

END;

$$;

-- Изменить информацию об авторе

CREATE OR REPLACE PROCEDURE update\_author(

in\_old\_author\_id INTEGER,

in\_fio VARCHAR,

in\_birth\_date DATE,

in\_country VARCHAR,

in\_nickname VARCHAR,

in\_new\_author\_id INTEGER

)

LANGUAGE plpgsql

AS $$

BEGIN

IF in\_old\_author\_id IS NOT NULL THEN

IF in\_new\_author\_id IS NOT NULL THEN

IF in\_old\_author\_id != in\_new\_author\_id THEN

-- Создаем новую запись автора

INSERT INTO public.authors

(

author\_id,

fio,

birth\_date,

country,

nickname

)

VALUES

(

in\_new\_author\_id,

in\_fio,

in\_birth\_date,

in\_country,

in\_nickname

);

-- Перевязываем все книги, связанные со старым автором

UPDATE public.authorships

SET author\_id = in\_new\_author\_id

WHERE author\_id = in\_old\_author\_id;

-- Удаляем старую запись автора

DELETE FROM public.authors

WHERE author\_id = in\_old\_author\_id;

ELSE

-- Если ID не менялся, обновляем только поля

UPDATE public.authors

SET

fio = in\_fio,

birth\_date = in\_birth\_date,

country = in\_country,

nickname = in\_nickname

WHERE author\_id = in\_old\_author\_id;

END IF;

ELSE

RAISE NOTICE 'Введите новый ID автора';

END IF;

ELSE

RAISE NOTICE 'Введите старый ID автора';

END IF;

END;

$$;

--===============================ФУНКЦИИ ДЛЯ ВЫВОДА ИНФОРМАЦИИ====================================

-- Вывести все книги

CREATE OR REPLACE FUNCTION print\_all\_books()

RETURNS TABLE (

isbn VARCHAR,

name VARCHAR,

publication\_year INTEGER,

age\_limit REAL,

publishing\_company VARCHAR,

page\_count INTEGER,

language VARCHAR,

cost REAL,

count\_of\_books INTEGER

)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT

b.isbn,

b.name,

b.publication\_year,

b.age\_limit,

b.publishing\_company,

b.page\_count,

b.language,

b.cost,

b.count\_of\_books

FROM public.books b;

END;

$$;

-- Вывести книгу по ISBN

CREATE OR REPLACE FUNCTION print\_book\_by\_isbn(

in\_book\_isbn VARCHAR

)

RETURNS TABLE (

isbn VARCHAR,

name VARCHAR,

publication\_year INTEGER,

age\_limit REAL,

publishing\_company VARCHAR,

page\_count INTEGER,

language VARCHAR,

cost REAL,

count\_of\_books INTEGER

)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT

b.isbn,

b.name,

b.publication\_year,

b.age\_limit,

b.publishing\_company,

b.page\_count,

b.language,

b.cost,

b.count\_of\_books

FROM public.books b

WHERE b.isbn = in\_book\_isbn;

END;

$$;

-- Вывести книги по жанру

CREATE OR REPLACE FUNCTION print\_book\_by\_style(

in\_style VARCHAR

)

RETURNS TABLE (

isbn VARCHAR,

name VARCHAR,

publication\_year INTEGER,

age\_limit REAL,

publishing\_company VARCHAR,

page\_count INTEGER,

language VARCHAR,

cost REAL,

count\_of\_books INTEGER

)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT

b.isbn,

b.name,

b.publication\_year,

b.age\_limit,

b.publishing\_company,

b.page\_count,

b.language,

b.cost,

b.count\_of\_books

FROM public.books b

JOIN public.books\_styles bs ON b.isbn = bs.book\_isbn

WHERE bs.style\_name = in\_style;

END;

$$;

-- Вывести книги по автору

CREATE OR REPLACE FUNCTION print\_book\_by\_author(

in\_author\_id INTEGER

)

RETURNS TABLE (

isbn VARCHAR,

name VARCHAR,

publication\_year INTEGER,

age\_limit REAL,

publishing\_company VARCHAR,

page\_count INTEGER,

language VARCHAR,

cost REAL,

count\_of\_books INTEGER

)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY

SELECT

b.isbn,

b.name,

b.publication\_year,

b.age\_limit,

b.publishing\_company,

b.page\_count,

b.language,

b.cost,

b.count\_of\_books

FROM public.books b

JOIN public.authorships a ON b.isbn = a.book\_isbn

WHERE a.author\_id = in\_author\_id;

END;

$$;

--===============================ХРАНИМЫЕ ПРОЦЕДУРЫ ДЛЯ ОЧИСТКИ ТАБЛИЦ====================================

-- Очистить все жанры книг

CREATE OR REPLACE PROCEDURE clear\_styles\_of\_book()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.books\_styles;

END;

$$;

-- Очистить все связи авторов и книг

CREATE OR REPLACE PROCEDURE clear\_authorships()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.authorships;

END;

$$;

-- Очистить всех авторов

CREATE OR REPLACE PROCEDURE clear\_authors()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.authors;

END;

$$;

-- Очистить все книги

CREATE OR REPLACE PROCEDURE clear\_books()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.books;

END;

$$;

-- Очистить все издательства

CREATE OR REPLACE PROCEDURE clear\_publishing\_company()

LANGUAGE plpgsql

AS $$

BEGIN

DELETE FROM public.publishing\_companies;

END;

$$;