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

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

CREATE TABLE IF NOT EXISTS public.Publishing\_Companies

(

"Name" character varying NOT NULL, -- Название издательства (первичный ключ)

EstablishmentYear integer, -- Год основания

ContactInfo character varying, -- Контактная информация

City character varying, -- Город

CONSTRAINT "Publishing\_Companies\_pkey" PRIMARY KEY ("Name") -- Определение первичного ключа

)

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" character varying NOT NULL, -- ISBN книги (первичный ключ)

"Name" character varying, -- Название книги

PublicationYear integer, -- Год публикации

AgeLimit real, -- Возрастное ограничение

PublishingCompany character varying, -- Название издательства (внешний ключ)

PageCount integer, -- Количество страниц

"Language" character varying, -- Язык книги

"Cost" real, -- Цена книги

"Count" integer, -- Количество экземпляров

CONSTRAINT "Books\_pkey" PRIMARY KEY ("ISBN"), -- Определение первичного ключа

CONSTRAINT "FK\_Books\_Publishing\_Companies" FOREIGN KEY (PublishingCompany)

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

(

"ID" integer NOT NULL, -- Идентификатор автора (первичный ключ)

FIO character varying, -- ФИО автора

BirthDate date, -- Дата рождения

Country character varying, -- Страна

Nickname character varying, -- Псевдоним

CONSTRAINT "Authors\_pkey" PRIMARY KEY ("ID") -- Определение первичного ключа

)

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

(

BookISBN character varying NOT NULL, -- ISBN книги (внешний ключ)

AuthorID integer NOT NULL, -- ID автора (внешний ключ)

CONSTRAINT "Authorships\_pkey" PRIMARY KEY (BookISBN, AuthorID), -- Составной первичный ключ

CONSTRAINT "FK\_Authorships\_Authors" FOREIGN KEY (AuthorID)

REFERENCES public.Authors ("ID") ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT "FK\_Authorships\_Books" FOREIGN KEY (BookISBN)

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

(

BookISBN character varying NOT NULL, -- ISBN книги (внешний ключ)

"Style" character varying NOT NULL, -- Жанр книги

CONSTRAINT "Books\_Styles\_pkey" PRIMARY KEY (BookISBN, "Style"), -- Составной первичный ключ

CONSTRAINT "FK\_Books\_Styles\_Books" FOREIGN KEY (BookISBN)

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 AddPublishingCompany (

publishingCompanyName character varying, -- Название издательства (обязательно)

publishingCompanyEstablishmentYear integer, -- Год основания

publishingCompanyContactInfo character varying, -- Контактная информация

publishingCompanyCity character varying -- Город

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Проверка, что название издательства не NULL

IF publishingCompanyName IS NOT NULL

THEN

-- Вставка новой записи в таблицу "Publishing\_Companies"

INSERT INTO Publishing\_Companies (

"Name",

EstablishmentYear,

ContactInfo,

City

)

VALUES (

publishingCompanyName,

publishingCompanyEstablishmentYear,

publishingCompanyContactInfo,

publishingCompanyCity

);

ELSE

-- Сообщение пользователю, если название не указано

RAISE NOTICE 'Введите название издательства';

END IF;

END;

$$;

-- Добавить книгу

CREATE OR REPLACE PROCEDURE AddBook (

bookISBN character varying, -- ISBN книги (обязательно)

bookName character varying, -- Название книги

bookYear integer, -- Год публикации

bookAgeLimit real, -- Возрастное ограничение

bookPublishingCompany character varying,-- Название издательства

bookPageCount integer, -- Количество страниц

language0 character varying, -- Язык книги

cost0 real, -- Цена книги

bookCount integer -- Количество экземпляров

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Проверка, что ISBN указан

IF bookISBN IS NOT NULL

THEN

-- Вставка новой записи в таблицу "Books"

INSERT INTO Books (

"ISBN",

"Name",

PublicationYear,

AgeLimit,

PublishingCompany,

PageCount,

"Language",

"Cost",

"Count"

)

VALUES (

bookISBN,

bookName,

bookYear,

bookAgeLimit,

bookPublishingCompany,

bookPageCount,

language0,

cost0,

bookCount

);

ELSE

-- Сообщение пользователю, если ISBN не указан

RAISE NOTICE 'Введите ISBN';

END IF;

END;

$$;

-- Назначить жанр книге

CREATE OR REPLACE PROCEDURE AddStyleToBook (

bookISBN character varying, -- ISBN книги (обязательно)

bookStyle character varying -- Жанр книги (обязательно)

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Проверка, существует ли книга с указанным ISBN

IF EXISTS(SELECT 1 FROM Books WHERE "ISBN" = bookISBN)

THEN

-- Проверка, что жанр указан

IF bookStyle IS NOT NULL

THEN

-- Вставка новой записи в таблицу "Books\_Styles"

INSERT INTO Books\_Styles (

BookISBN,

"Style"

)

VALUES (

bookISBN,

bookStyle

);

ELSE

-- Сообщение пользователю, если жанр не указан

RAISE NOTICE 'Введите Жанр';

END IF;

ELSE

-- Сообщение пользователю, если ISBN не найден

RAISE NOTICE 'Книга с ISBN % не найдена', bookISBN;

END IF;

END

$$;

-- Добавить автора

CREATE OR REPLACE PROCEDURE AddAuthor (

authorID integer, -- Идентификатор автора (обязательно)

authorFIO character varying, -- ФИО автора

authorBirthday date, -- Дата рождения

authorCountry character varying, -- Страна

authorNickname character varying -- Псевдоним

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Проверка, что ID автора указан

IF authorID IS NOT NULL

THEN

-- Вставка новой записи в таблицу "Authors"

INSERT INTO Authors (

"ID",

FIO,

BirthDate,

Country,

Nickname

)

VALUES (

authorID,

authorFIO,

authorBirthday,

authorCountry,

authorNickname

);

ELSE

-- Сообщение пользователю, если ID не указан

RAISE NOTICE 'Введите ID';

END IF;

END;

$$;

-- Назначить автора книге

CREATE OR REPLACE PROCEDURE AddAuthorToBook (

authorID integer, -- Идентификатор автора (обязательно)

bookISBN character varying -- ISBN книги (обязательно)

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Проверка, существует ли автор с указанным ID

IF EXISTS(SELECT 1 FROM Authors WHERE "ID" = authorID)

THEN

-- Проверка, существует ли книга с указанным ISBN

IF EXISTS(SELECT 1 FROM Books WHERE "ISBN" = bookISBN)

THEN

-- Вставка новой связи в таблицу "Authorships"

INSERT INTO Authorships (

BookISBN,

AuthorID

)

VALUES (

bookISBN,

authorID

);

ELSE

-- Сообщение пользователю, если ISBN книги не найден

RAISE NOTICE 'Книга с ISBN % не найдена', bookISBN;

END IF;

ELSE

-- Сообщение пользователю, если ID автора не найден

RAISE NOTICE 'Автор с ID % не найден', authorID;

END IF;

END;

$$;

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

-- Удалить издательство и связанные с ним данные

CREATE OR REPLACE PROCEDURE DeletePublishingCompany (

publishingCompanyName character varying -- Название издательства для удаления

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление записей из "Authorships" связанных с книгами этого издательства

DELETE FROM Authorships

WHERE BookISBN IN (

SELECT "ISBN"

FROM Books

WHERE PublishingCompany = publishingCompanyName

);

-- Удаление записей из "Books\_Styles" связанных с книгами этого издательства

DELETE FROM Books\_Styles

WHERE BookISBN IN (

SELECT "ISBN"

FROM Books

WHERE PublishingCompany = publishingCompanyName

);

-- Удаление записей из "Books" связанных с этим издательством

DELETE FROM Books

WHERE PublishingCompany = publishingCompanyName;

-- Удаление самого издательства из "Publishing\_Companies"

DELETE FROM Publishing\_Companies

WHERE "Name" = publishingCompanyName;

END;

$$;

-- Удалить автора книги

CREATE OR REPLACE PROCEDURE DeleteAuthorOfBook (

authorID integer, -- Идентификатор автора

bookISBN character varying -- ISBN книги

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление связи между автором и книгой из "Authorships"

DELETE FROM Authorships

WHERE AuthorID = authorID

AND BookISBN = bookISBN;

END;

$$;

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

CREATE OR REPLACE PROCEDURE ClearNonUsedAuthorsOfBook()

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление авторов, чьи ID отсутствуют в таблице "Authorships"

DELETE FROM Authors

WHERE "ID" NOT IN (SELECT AuthorID FROM Authorships);

END;

$$;

-- Удалить книгу и связанные с ней данные

CREATE OR REPLACE PROCEDURE DeleteBook (

bookISBN character varying -- ISBN книги для удаления

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление связей из "Authorships"

DELETE FROM Authorships

WHERE BookISBN = bookISBN;

-- Удаление связей из "Books\_Styles"

DELETE FROM Books\_Styles

WHERE BookISBN = bookISBN;

-- Удаление самой книги из "Books"

DELETE FROM Books

WHERE "ISBN" = bookISBN;

END;

$$;

-- Удалить жанр книги

CREATE OR REPLACE PROCEDURE DeleteBookStyle (

bookISBN character varying, -- ISBN книги

bookStyle character varying -- Жанр для удаления

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление записи из "Books\_Styles" для указанной книги и жанра

DELETE FROM Books\_Styles

WHERE BookISBN = bookISBN

AND "Style" = bookStyle;

END;

$$;

-- Удалить автора книги

CREATE OR REPLACE PROCEDURE DeleteBookAuthor (

bookISBN character varying, -- ISBN книги

authorID integer -- Идентификатор автора

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление записи из "Authorships" для указанной книги и автора

DELETE FROM Authorships

WHERE BookISBN = bookISBN

AND AuthorID = authorID;

END;

$$;

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

-- Изменить информацию об издательстве

CREATE OR REPLACE PROCEDURE UpdatePublishingCompany(

publishingCompanyOldName character varying, -- Старое название издательства

publishingCompanyEstablishmentYear integer, -- Новый год основания

publishingCompanyContactInfo character varying, -- Новая контактная информация

publishingCompanyCity character varying, -- Новый город

publishingCompanyNewName character varying -- Новое название издательства

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Проверка, что старое название издательства указано

IF publishingCompanyOldName IS NOT NULL

THEN

-- Проверка, что новое название издательства указано

IF publishingCompanyNewName IS NOT NULL

THEN

-- Если старое и новое названия отличаются

IF publishingCompanyOldName != publishingCompanyNewName

THEN

-- Вставка новой записи с новым названием

INSERT INTO Publishing\_Companies (

"Name",

EstablishmentYear,

ContactInfo,

City

)

VALUES (

publishingCompanyNewName,

publishingCompanyEstablishmentYear,

publishingCompanyContactInfo,

publishingCompanyCity

);

-- Обновление записей в "Books" с новым названием издательства

UPDATE Books

SET PublishingCompany = publishingCompanyNewName

WHERE PublishingCompany = publishingCompanyOldName;

-- Удаление старой записи издательства

DELETE FROM Publishing\_Companies

WHERE "Name" = publishingCompanyOldName;

ELSE

-- Если название не изменилось, просто обновляем остальные поля

UPDATE Publishing\_Companies

SET

EstablishmentYear = publishingCompanyEstablishmentYear,

ContactInfo = publishingCompanyContactInfo,

City = publishingCompanyCity

WHERE "Name" = publishingCompanyOldName;

END IF;

ELSE

-- Сообщение пользователю, если новое название не указано

RAISE NOTICE 'Введите новое название издательства';

END IF;

ELSE

-- Сообщение пользователю, если старое название не указано

RAISE NOTICE 'Введите старое название издательства';

END IF;

END;

$$;

-- Изменить информацию о книге

CREATE OR REPLACE PROCEDURE UpdateBook(

bookOldISBN character varying, -- Старый ISBN книги

bookName character varying, -- Новое название книги

bookYear integer, -- Новый год публикации

bookAgeLimit real, -- Новое возрастное ограничение

bookPageCount integer, -- Новое количество страниц

language0 character varying, -- Новый язык книги

cost0 real, -- Новая цена книги

bookCount integer, -- Новое количество экземпляров

bookNewISBN character varying, -- Новый ISBN книги

bookPublishingCompany character varying -- Новое название издательства

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Проверка, что старый ISBN указан

IF bookOldISBN IS NOT NULL

THEN

-- Проверка, что новый ISBN указан

IF bookNewISBN IS NOT NULL

THEN

-- Если старый и новый ISBN отличаются

IF bookOldISBN != bookNewISBN

THEN

-- Вставка новой записи с новым ISBN

INSERT INTO Books (

"ISBN",

"Name",

PublicationYear,

AgeLimit,

PublishingCompany,

PageCount,

"Language",

"Cost",

"Count"

)

VALUES (

bookNewISBN,

bookName,

bookYear,

bookAgeLimit,

bookPublishingCompany,

bookPageCount,

language0,

cost0,

bookCount

);

-- Обновление записей в "Books\_Styles" с новым ISBN

UPDATE Books\_Styles

SET BookISBN = bookNewISBN

WHERE BookISBN = bookOldISBN;

-- Обновление записей в "Authorships" с новым ISBN

UPDATE Authorships

SET BookISBN = bookNewISBN

WHERE BookISBN = bookOldISBN;

-- Удаление старой записи книги

DELETE FROM Books

WHERE "ISBN" = bookOldISBN;

ELSE

-- Если ISBN не изменился, просто обновляем остальные поля

UPDATE Books

SET

"Name" = bookName,

PublicationYear = bookYear,

AgeLimit = bookAgeLimit,

PublishingCompany = bookPublishingCompany,

PageCount = bookPageCount,

"Language" = language0,

"Cost" = cost0,

"Count" = bookCount

WHERE "ISBN" = bookOldISBN;

END IF;

ELSE

-- Сообщение пользователю, если новый ISBN не указан

RAISE NOTICE 'Введите новый ISBN';

END IF;

ELSE

-- Сообщение пользователю, если старый ISBN не указан

RAISE NOTICE 'Введите старый ISBN';

END IF;

END;

$$;

-- Изменить информацию об авторе

CREATE OR REPLACE PROCEDURE UpdateAuthor(

authorOldID integer, -- Старый ID автора

authorFIO character varying, -- Новое ФИО автора

authorBirthday date, -- Новая дата рождения

authorCountry character varying, -- Новая страна

authorNickname character varying, -- Новый псевдоним

authorNewID integer -- Новый ID автора

)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Проверка, что старый ID автора указан

IF authorOldID IS NOT NULL

THEN

-- Проверка, что новый ID автора указан

IF authorNewID IS NOT NULL

THEN

-- Проверка, что старый и новый ID отличаются

IF authorOldID != authorNewID

THEN

-- Вставка новой записи с новым ID

INSERT INTO Authors (

"ID",

FIO,

BirthDate,

Country,

Nickname

)

VALUES (

authorNewID,

authorFIO,

authorBirthday,

authorCountry,

authorNickname

);

-- Обновление записей в "Authorships" с новым ID

UPDATE Authorships

SET AuthorID = authorNewID

WHERE AuthorID = authorOldID;

-- Удаление старой записи автора

DELETE FROM Authors

WHERE "ID" = authorOldID;

ELSE

-- Если ID не изменился, просто обновляем остальные поля

UPDATE Authors

SET

FIO = authorFIO,

BirthDate = authorBirthday,

Country = authorCountry,

Nickname = authorNickname

WHERE "ID" = authorOldID;

END IF;

ELSE

-- Сообщение пользователю, если новый ID не указан

RAISE NOTICE 'Введите новый ID';

END IF;

ELSE

-- Сообщение пользователю, если старый ID не указан

RAISE NOTICE 'Введите старый ID';

END IF;

END;

$$;

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

-- Вывести все книги

CREATE OR REPLACE FUNCTION PrintAllBooks()

RETURNS TABLE (

"ISBN" character varying,

"Name" character varying,

PublicationYear integer,

AgeLimit real,

PublishingCompany character varying,

PageCount integer,

"Language" character varying,

"Cost" real,

"Count" integer

)

LANGUAGE plpgsql

AS

$$

BEGIN

RETURN QUERY

SELECT

b."ISBN",

b."Name",

b.PublicationYear,

b.AgeLimit,

b.PublishingCompany,

b.PageCount,

b."Language",

b."Cost",

b."Count"

FROM Books b;

END;

$$;

-- Вывести книгу по ISBN

CREATE OR REPLACE FUNCTION PrintBookByISBN(

bookISBN character varying -- ISBN книги для поиска

)

RETURNS TABLE (

"ISBN" character varying,

"Name" character varying,

PublicationYear integer,

AgeLimit real,

PublishingCompany character varying,

PageCount integer,

"Language" character varying,

"Cost" real,

"Count" integer

)

LANGUAGE plpgsql

AS

$$

BEGIN

RETURN QUERY

SELECT

b."ISBN",

b."Name",

b.PublicationYear,

b.AgeLimit,

b.PublishingCompany,

b.PageCount,

b."Language",

b."Cost",

b."Count"

FROM Books b

WHERE b."ISBN" = bookISBN;

END;

$$;

-- Вывести книги по жанру

CREATE OR REPLACE FUNCTION PrintBookByStyle(

style0 character varying -- Жанр для поиска

)

RETURNS TABLE (

"ISBN" character varying,

"Name" character varying,

PublicationYear integer,

AgeLimit real,

PublishingCompany character varying,

PageCount integer,

"Language" character varying,

"Cost" real,

"Count" integer

)

LANGUAGE plpgsql

AS

$$

BEGIN

RETURN QUERY

SELECT

b."ISBN",

b."Name",

b.PublicationYear,

b.AgeLimit,

b.PublishingCompany,

b.PageCount,

b."Language",

b."Cost",

b."Count"

FROM Books b

JOIN Books\_Styles bs ON b."ISBN" = bs.BookISBN

WHERE bs."Style" = style0;

END;

$$;

-- Вывести книги по автору

CREATE OR REPLACE FUNCTION PrintBookByAuthor(

author integer -- ID автора для поиска

)

RETURNS TABLE (

"ISBN" character varying,

"Name" character varying,

PublicationYear integer,

AgeLimit real,

PublishingCompany character varying,

PageCount integer,

"Language" character varying,

"Cost" real,

"Count" integer

)

LANGUAGE plpgsql

AS

$$

BEGIN

RETURN QUERY

SELECT

b."ISBN",

b."Name",

b.PublicationYear,

b.AgeLimit,

b.PublishingCompany,

b.PageCount,

b."Language",

b."Cost",

b."Count"

FROM Books b

JOIN Authorships a ON b."ISBN" = a.BookISBN

WHERE a.AuthorID = author;

END;

$$;

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

-- Очистить все жанры книг

CREATE OR REPLACE PROCEDURE ClearStylesOfBook()

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление всех записей из таблицы "Books\_Styles"

DELETE FROM Books\_Styles;

END;

$$;

-- Очистить все связи авторов и книг

CREATE OR REPLACE PROCEDURE ClearAuthorships()

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление всех записей из таблицы "Authorships"

DELETE FROM Authorships;

END;

$$;

-- Очистить всех авторов

CREATE OR REPLACE PROCEDURE ClearAuthors()

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление всех записей из таблицы "Authors"

DELETE FROM Authors;

END;

$$;

-- Очистить все книги

CREATE OR REPLACE PROCEDURE ClearBooks()

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление всех записей из таблицы "Books"

DELETE FROM Books;

END;

$$;

-- Очистить все издательства

CREATE OR REPLACE PROCEDURE ClearPublishingCompany()

LANGUAGE plpgsql

AS

$$

BEGIN

-- Удаление всех записей из таблицы "Publishing\_Companies"

DELETE FROM Publishing\_Companies;

END;

$$;