--===============================СОЗДАНИЕ БД====================================

CREATE DATABASE "BookKatalog"

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC\_COLLATE = 'Russian\_Russia.1251'

LC\_CTYPE = 'Russian\_Russia.1251'

LOCALE\_PROVIDER = 'libc'

TABLESPACE = pg\_default

CONNECTION LIMIT = -1

IS\_TEMPLATE = False;

CREATE TABLE IF NOT EXISTS public.Publishing\_Companies

(

"Name" character varying COLLATE pg\_catalog."default" NOT NULL,

EstablishmentYear integer,

ContactInfo character varying COLLATE pg\_catalog."default",

City character varying COLLATE pg\_catalog."default",

CONSTRAINT "Publishing\_Companies\_pkey" PRIMARY KEY ("Name")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.Publishing\_Companies

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.Books

(

"ISBN" character varying COLLATE pg\_catalog."default" NOT NULL,

"Name" character varying COLLATE pg\_catalog."default",

PublicationYear integer,

AgeLimit real,

PublishingCompany character varying COLLATE pg\_catalog."default",

PageCount integer,

"Language" character varying COLLATE pg\_catalog."default",

"Cost" real,

"Count" integer,

CONSTRAINT "Books\_pkey" PRIMARY KEY ("ISBN"),

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

REFERENCES public.Publishing\_Companies ("Name") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.Books

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.Authors

(

"ID" integer NOT NULL,

FIO character varying COLLATE pg\_catalog."default",

BirthDate date,

Country character varying COLLATE pg\_catalog."default",

Nickname character varying COLLATE pg\_catalog."default",

CONSTRAINT "Authors\_pkey" PRIMARY KEY ("ID")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.Authors

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.Authorships

(

BookISBN character varying COLLATE pg\_catalog."default" NOT NULL,

AuthorID integer NOT NULL,

CONSTRAINT "Authorships\_pkey" PRIMARY KEY (BookISBN, AuthorID),

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

REFERENCES public.Authors ("ID") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

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

REFERENCES public.Books ("ISBN") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.Authorships

OWNER to postgres;

CREATE TABLE IF NOT EXISTS public.Books\_Styles

(

BookISBN character varying COLLATE pg\_catalog."default" NOT NULL,

"Style" character varying COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Books\_Styles\_pkey" PRIMARY KEY (BookISBN, "Style"),

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

REFERENCES public.Books ("ISBN") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS public.Books\_Styles

OWNER to postgres;

--===============================ХРАНИМЫЕПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ====================================

--ХП для работы с таблицами

--Добавить издательство

CREATE OR REPLACE PROCEDURE AddPublishingCompany (publishingCompanyName character varying, publishingCompanyEstablishmentYear integer, publishingCompanyContactInfo character varying, publishingCompanyCity character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

--Если введённые минимальные данные корректны

IF publishingCompanyName IS NOT NULL

THEN

INSERT INTO Publishing\_Companies VALUES(publishingCompanyName, publishingCompanyEstablishmentYear, publishingCompanyContactInfo, publishingCompanyCity);

ELSE

RAISE NOTICE 'Введите название издательства';

END IF;

END;

$$;

--Добавить книгу

CREATE OR REPLACE PROCEDURE AddBook(bookISBN character varying, bookName character varying, bookYear integer, bookAgeLimit real, bookPublishingCompany character varying, bookPageCount integer, language0 character varying, cost0 real, bookCount integer)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Если введенные минимальные данные корректны

IF bookISBN IS NOT NULL

THEN

INSERT INTO Books VALUES(bookISBN, bookName, bookYear, bookAgeLimit, bookPublishingCompany, bookPageCount, language0, cost0, bookCount);

ELSE

RAISE NOTICE 'Введите ISBN';

END IF;

END;

$$;

--Назначить жанр книге

CREATE OR REPLACE PROCEDURE AddStyleToBook (bookISBN character varying, bookStyle character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Если существует указанная книга

IF EXISTS(SELECT \* FROM Books WHERE Books.ISBN = bookISBN)

THEN

--Если введённые минимальные данные корректны

IF bookStyle IS NOT NULL

THEN

INSERT INTO Books\_Styles VALUES(bookISBN, bookStyle);

ELSE

RAISE NOTICE 'Введите Жанр';

END IF;

ELSE

RAISE NOTICE 'Введите ISBN';

END IF;

END

$$;

--Добавить автора

CREATE OR REPLACE PROCEDURE AddAuthor(authorID integer, authorFIO character varying, authorBirthday date, authorCountry character varying, authorNickname character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Если введенные минимальные данные корректны

IF authorID IS NOT NULL

THEN

INSERT INTO Authors VALUES (authorID, authorFIO, authorBirthday, authorCountry, authorNickname);

ELSE

RAISE NOTICE 'Введите ID';

END IF;

END;

$$;

--Назначить автора книге

CREATE OR REPLACE PROCEDURE AddAuthorToBook (authorID integer, bookISBN character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Если существует указанный автор

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

THEN

-- Если существует указанная книга

IF EXISTS(SELECT \* FROM Books WHERE ISBN = bookISBN)

THEN

INSERT INTO Authorships VALUES(bookISBN, authorID);

ELSE

RAISE NOTICE 'Введите ISBN существующей книги';

END IF;

ELSE

RAISE NOTICE 'Введите ID существующего автора';

END IF;

END

$$;

--Удалить издательство

CREATE OR REPLACE PROCEDURE DeletePublishingCompany (publishingCompanyName character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Authorships

WHERE BookISBN IN (SELECT ISBN

FROM Books

WHERE PublicationCompany IN(SELECT "Name"

FROM Publishing\_Companies

WHERE "Name" = publishingCompanyName));

DELETE FROM Books\_Styles

WHERE BookISBN IN (SELECT ISBN

FROM Books

WHERE PublicationCompany IN(SELECT "Name"

FROM Publishing\_Companies

WHERE "Name" = publishingCompanyName));

DELETE FROM Books

WHERE PublicationCompany IN(SELECT "Name"

FROM Publishing\_Companies

WHERE "Name" = publishingCompanyName);

DELETE FROM Publishing\_Companies

WHERE "Name" = publishingCompanyName;

END

$$;

--Удалить автора книги

CREATE OR REPLACE PROCEDURE DeleteAuthorOfBook (authorID integer, bookISBN character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Authorships

WHERE Authorships.AuthorID = authorID

AND Authorships.BookISBN = bookISBN;

END;

$$;

--!!!!!!!!!!!!!!!!!!!!!!!!!!Удалить неиспользуемых авторов книги!!!!!!!!!!!!!!!!!!!!!!!!!!

CREATE OR REPLACE PROCEDURE ClearNonUsedAuthorsOfBook()

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Authors

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

END;

$$;

--Удалить книгу

CREATE OR REPLACE PROCEDURE DeleteBook (bookISBN character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Authorships

WHERE BookISBN = bookISBN;

DELETE FROM Books\_Styles

WHERE BookISBN = bookISBN;

DELETE FROM Books

WHERE Books.ISBN = bookISBN;

END;

$$;

--Удалить жанр книги

CREATE OR REPLACE PROCEDURE DeleteBookStyle (bookISBN character varying, bookStyle character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Books\_Styles

WHERE Books\_Styles.BookISBN = bookISBN

AND Books\_Styles.Style = bookStyle;

END

$$;

--Удалить автора книги

CREATE OR REPLACE PROCEDURE DeleteBookAuthor (bookISBN character varying, authorID int)

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Authorships

WHERE Authorships.BookISBN = bookISBN

AND Authorships.AuthorID = authorID;

END;

$$;

--Изменить издательство

CREATE OR REPLACE PROCEDURE UpdatePublishingCompany(publishingCompanyOldName character varying, publishingCompanyEstablishmentYear integer, publishingCompanyContactInfo character varying, publishingCompanyCity character varying, publishingCompanyNewName character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Если есть такое издательство

IF publishingCompanyOldName IS NOT NULL

THEN

IF publishingCompanyNewName IS NOT NULL

THEN

IF publishingCompanyOldName != publishingCompanyNewName

THEN

INSERT INTO Publishing\_Companies VALUES(publishingCompanyNewName, publishingCompanyEstablishmentYear, publishingCompanyContactInfo, publishingCompanyCity);

UPDATE Books

SET PublishingCompany = publishingCompanyNewName

WHERE PublishingCompany = publishingCompanyOldName;

DELETE FROM Publishing\_Companies WHERE Publishing\_Companies."Name" = publishingCompanyOldName;

ELSE

UPDATE Publishing\_Companies

SET EstablishmentYear = publishingCompanyEstablishmentYear,

ContactInfo = publishingCompanyContactInfo,

City = publishingCompanyCity

WHERE "Name" = publishingCompanyOldName;

END IF;

ELSE

RAISE NOTICE 'Введите новый ISBN';

END IF;

ELSE RAISE NOTICE 'Введите старый ISBN';

END IF;

END;

$$;

--Изменить информацию о книге

CREATE OR REPLACE PROCEDURE UpdateBook(bookOldISBN character varying, bookName character varying, bookYear integer, bookAgeLimit real, bookPageCount integer, language0 character varying, cost0 real, bookCount integer, bookNewISBN character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Если есть такая книга

IF bookOldISBN IS NOT NULL

THEN

IF bookNewISBN IS NOT NULL

THEN

IF bookOldISBN != bookNewISBN

THEN

INSERT INTO Books VALUES(bookNewISBN, bookName, bookYear, bookAgeLimit, bookPageCount, language0, cost0, bookCount);

UPDATE Books\_Styles

SET BookISBN = bookNewISBN

WHERE BookISBN = bookOldISBN;

UPDATE Authorships

SET BookISBN = bookNewISBN

WHERE BookISBN = bookOldISBN;

DELETE FROM Books WHERE Books.ISBN = bookOldISBN;

ELSE

UPDATE Books

SET "Name" = bookName,

PublicationYear = bookYear,

AgeLimit = bookAgeLimit,

PageCount = bookPageCount,

"Language" = language0,

"Cost" = cost0,

"Count" = bookCount

WHERE BookISBN = bookOldISBN;

END IF;

ELSE

RAISE NOTICE 'Введите новое название';

END IF;

ELSE RAISE NOTICE 'Введите старое название';

END IF;

END;

$$;

--Изменить автора

CREATE OR REPLACE PROCEDURE UpdateAuthor(authorOldID integer, authorFIO character varying, authorBirthday date, authorCountry character varying, authorNickname character varying, authorNewID integer)

LANGUAGE plpgsql

AS

$$

BEGIN

-- Если есть такой автор

IF authorOldID IS NOT NULL

THEN

IF authorNewID IS NOT NULL

THEN

IF authorOldID != authorOldID

THEN

INSERT INTO Authors VALUES(authorNewID, authorFIO, authorBirthday, authorCountry, authorNickname);

UPDATE Authorships

SET AuthorID = authorNewID

WHERE AuthorID = authorOldID;

DELETE FROM Authors WHERE Authors.ID = authorOldID;

ELSE

UPDATE Authors

SET FIO = authorFIO,

BirthDate = authorBirthday,

Country = authorCountry,

Nickname = authorNickname

WHERE AuthorID = authorOldID;

END IF;

ELSE

RAISE NOTICE 'Введите новый ID';

END IF;

ELSE RAISE NOTICE 'Введите старый ID';

END IF;

END;

$$;

--Хранимые процедуы для вывода информации

--Вывести книги

CREATE OR REPLACE PROCEDURE PrintAllBooks()

LANGUAGE plpgsql

AS

$$

BEGIN

SELECT \*

FROM Books;

END;

$$;

--Вывести книгу по ISBN

CREATE OR REPLACE PROCEDURE PrintBookByISBN(bookISBN character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

SELECT \*

FROM Books

WHERE Books.ISBN=bookISBN;

END;

$$;

--Вывести книги по жанру

CREATE OR REPLACE PROCEDURE PrintBookByStyle(style0 character varying)

LANGUAGE plpgsql

AS

$$

BEGIN

SELECT \*

FROM Books JOIN Books\_Styles ON Books.ISBN=Books\_Styles.bookISBN

WHERE Books\_Styles."Style" = style0;

END;

$$;

--Вывести книги по автору

CREATE OR REPLACE PROCEDURE PrintBookByAuthor(author integer)

LANGUAGE plpgsql

AS

$$

BEGIN

SELECT \*

FROM Books JOIN Authorships ON Books.ISBN=Authorships.BookISBN

WHERE Authorships.AuthorID=author;

END;

$$;

--!!!!!!!!!!!!!!!!!!!!Это процедуры для полного очищения отдельных таблиц!!!!!!!!!!!!!!!!!!!!

CREATE OR REPLACE PROCEDURE ClearStylesOfBook()

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Books\_Styles;

END;

$$;

CREATE OR REPLACE PROCEDURE ClearAuthorships()

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Authorships;

END;

$$;

CREATE OR REPLACE PROCEDURE ClearAuthors()

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Authors;

END;

$$;

CREATE OR REPLACE PROCEDURE ClearBooks()

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Books;

END;

$$;

CREATE OR REPLACE PROCEDURE ClearPublishingCompany()

LANGUAGE plpgsql

AS

$$

BEGIN

DELETE FROM Publishing\_Companies;

END;

$$;