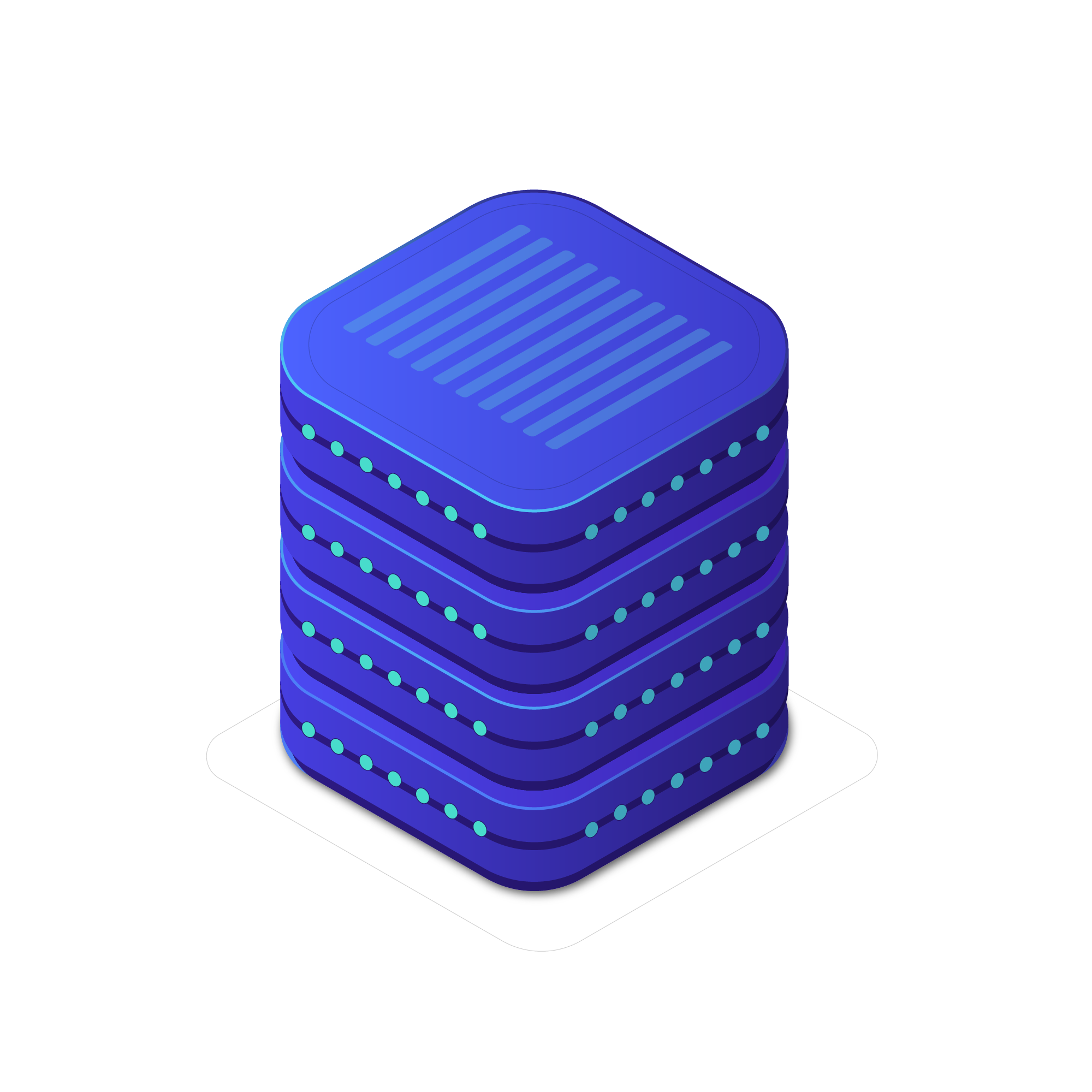


Bilgisayar ve Bilişim Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü

**VERİTABANI YÖNETİM SİSTEMLERİ**



Ad ve Soyad :Bilal RASLEN

Öğrenci No : G201210597

Group : 2B

# Uygulama Tanımı

Projem kitap ödünç almak ve satmak için bir kütüphane hakkında , Program başlatıldığında, bir öğrenci veya bir danışman seçeceğimiz bir pencere açılacaktır.

Seçtikten sonra yine bir pencere açılacak , pencere içinde 5 düğme vardır : Insert(Ekleme) , Delete(Silme) , Update(Güncelleme) , Search(Arama) ve Listing(Listeleme) .

O düğmelerle işlemleri Yapabiliriz , işlemler yapacağız ve aynı zamanda bu işlemler bizim oluşturduğumuz veri tabanında da gerçekleşecektir.

# İş Kuralları

* Her kitap için en fazla çok ödünç alan ve en az sıfır ödünç alan vardır.
* Her kitabın en fazla bir satışı vardır ve en az satışı yoktur.
* Her kitabın en az bir türü ve en fazla birden fazla türü vardır.
* Kitabın en az bir baskısı ve en fazla birden fazla baskısı vardır.
* Her kitabın en az bir dili ve en fazla birden fazla dili vardır.
* Her kitap en fazla bir rafta ve en az.
* Her rafta en fazla birden fazla kitap ve en az sıfır kitap vardır.
* Her yayınevinin en az sıfır kitabı ve en fazla birden fazla kitabı vardır.
* Yayınevi en az ve en fazla bir ülkeye ait olmalıdır.
* Ülkenin en azından bir yayınevi olmamalı ve ülkede çok sayıda yayınevi var.
* Yazarın en az bir kitabı yoktur ve en fazla çok kitabı vardır.
* Bir kişi mümkün olduğu kadar çok borçlanma yapabilir , en çokta.
* Bir kişi en fazla birden fazla alışveriş yapabilir , en çokta.
* Alıcı en fazla bir çok öneride bulunabilir.
* Alıcı en azından öneride bulunamaz .

# Ilişkisel Şema

* author(**authorID : integer** , authorName : Character Varying , date : date , publisherID **:** integer);
* book(**bookID : integer**, bookName : Character Varying , authorID : integer , categoryID : integer , placeID : integer , priceID : integer , languageID : integer , publisherID : integer);
* borrower(**studentID : integer** , firstDate : date , finishDate : date);
* buyer(**studentID : integer** , moneyID : integer);
* category(**categoryID : integer**, categoryName : Character Varying);
* country(**countryID : integer** , countryAbb : Character Varying(3) , countryName : Character Varying(50) , languageID : integer , publisherID : integer ;
* edition(editionID : integer);
* language(**languageID : integer** , languageName : Character Varying(35) , languageAbb : Character Varying(3) , **countryID : integer**;
* money(**moneyID : integer** , moneyMiktari : money);
* notes(**noteID : integer** , noteText : Text , borrowerID : integer , buyerId : integer);
* place(**placeID : integer** , placeAdress : Character Varying(150);
* price(**priceID : integer** , priceMiktari : money);
* publisher(**publisherID : integer** , publisherName : Character Varying(160) , publisherAdress : Text , publisherMail : Character Varying(100) , countryID **:** integer );
* student(**studentID : integer** , studentnName : Character Varying(50) , studentDate : date , studentPhoneNo : Character Varying(12) , borrowerID : integer , buyerID: integer;

# Varlık Bağıntı Diyagramı

A picture containing diagram

Description automatically generated

Uygulamanın Düğmeler Kodları

**Listing (Listeleme) :**

Text

Description automatically generated

Graphical user interface

Description automatically generated

private void BTNListing\_Click(object sender, EventArgs e)

{

string sql = "SELECT \"public\".\"book\".\"bookID\",\r\n \"public\".\"book\".\"bookName\",\r\n \"public\".\"author\".\"authorName\",\r\n \"public\".\"category\".\"categoryName\",\r\n \"public\".\"edition\".\"editionID\",\r\n \"public\".\"language\".\"languageName\",\r\n \"public\".\"language\".\"languageabb\",\r\n \"public\".\"publisher\".\"publisherName\"\r\nFROM \"public\".\"author\" \r\nRIGHT OUTER JOIN \"public\".\"book\" ON \"public\".\"author\".\"authorID\" = \"public\".\"book\".\"authorID\" \r\nINNER JOIN \"public\".\"category\" ON \"public\".\"category\".\"categoryID\" = \"public\".\"book\".\"categoryID\" \r\nINNER JOIN \"public\".\"edition\" ON \"public\".\"edition\".\"editionID\" = \"public\".\"book\".\"editionID\" \r\nINNER JOIN \"public\".\"language\" ON \"public\".\"language\".\"languageID\" = \"public\".\"book\".\"languageID\" \r\nINNER JOIN \"public\".\"publisher\" ON \"public\".\"publisher\".\"publisherID\" = \"public\".\"book\".\"publisherID\" ";

NpgsqlDataAdapter da = new NpgsqlDataAdapter(sql, conn);

DataSet ds = new DataSet();

da.Fill(ds);

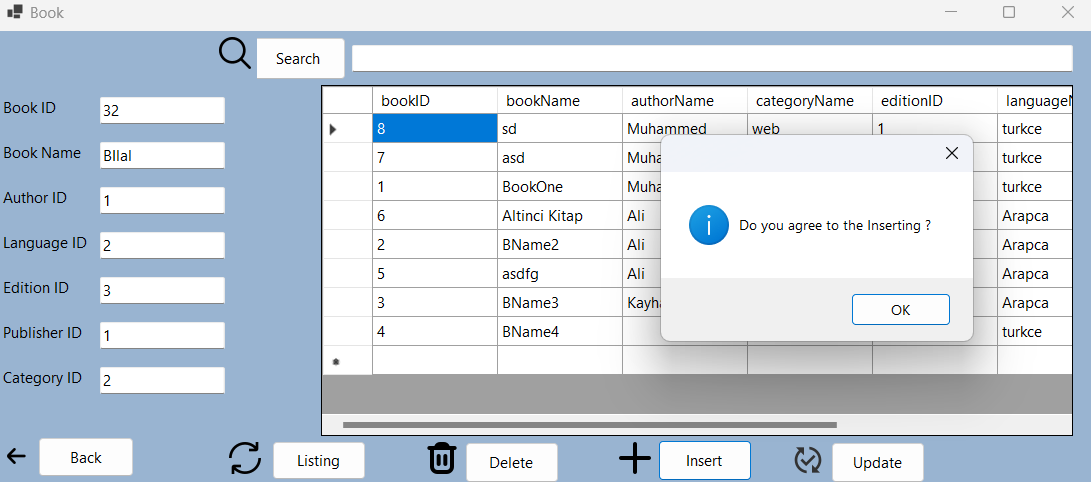
dataGridView1.DataSource = ds.Tables[0];

}

**Insert (Ekleme) :**

**Text

Description automatically generated**

 private void BTNInsert\_Click\_1(object sender, EventArgs e)

{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand("insert into \"book\" (\"bookID\",\"authorID\",\"bookName\",\"categoryID\",\"editionID\",\"languageID\",\"publisherID\") values (@p1,@p2,@p3,@p4,@p5,@p6,@p7)", conn);

cmd.Parameters.AddWithValue("@p1", int.Parse(TBBookID.Text));

cmd.Parameters.AddWithValue("@p2", int.Parse(TBAuthorName.Text));

cmd.Parameters.AddWithValue("@p3", TBBookName.Text);

cmd.Parameters.AddWithValue("@p4", int.Parse(TBCategory.Text));

cmd.Parameters.AddWithValue("@p5", int.Parse(TBEdition.Text));

cmd.Parameters.AddWithValue("@p6", int.Parse(TBLanguage.Text));

cmd.Parameters.AddWithValue("@p7", int.Parse(TBPublisherName.Text));

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Do you agree to the Inserting ?", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("PK Not NULL and Unique", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

conn.Close();

// MessageBox.Show("Do you agree to the Inserting ?", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

**Update (Güncelleme) :**

**Text

Description automatically generated**

# Graphical user interface, application Description automatically generated

private void BTNUpdate\_Click(object sender, EventArgs e)

{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand("update \"book\" set \"bookName\"=@p1 ,\"authorName\"=@p3,\"categoryID\"=@p4,\"editionID\"=@p5,\"languageID\"=@p6,\"publisherName\"=@p7 where \"bookID\"=@p2", conn);

cmd.Parameters.AddWithValue("@p1", TBBookName.Text);

cmd.Parameters.AddWithValue("@p2", int.Parse(TBBookID.Text));

cmd.Parameters.AddWithValue("@p3", TBAuthorName.Text);

cmd.Parameters.AddWithValue("@p4", int.Parse(TBCategory.Text));

cmd.Parameters.AddWithValue("@p5", int.Parse(TBEdition.Text));

cmd.Parameters.AddWithValue("@p6", int.Parse(TBLanguage.Text));

cmd.Parameters.AddWithValue("@p7", TBPublisherName.Text);

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Do you agree to the Updating ?", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("PK Not NULL and Unique", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

conn.Close();

// MessageBox.Show("Do you agree to the Updating ?", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

**Delete (Silme) :**

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

private void BTNDelete\_Click(object sender, EventArgs e)

{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand("Delete From \"author\" Where \"authorID\"=@p2 ", conn);

cmd.Parameters.AddWithValue("@p2", int.Parse(TBAuthorID.Text));

cmd.ExecuteNonQuery();

conn.Close();

MessageBox.Show("Do you agree to the Deletion ?", "", MessageBoxButtons.OK, MessageBoxIcon.Stop);

}

**Search (Arama) :**

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

private void BTNSearch\_Click(object sender, EventArgs e)

{

conn.Open();

NpgsqlCommand cmd = new NpgsqlCommand("select \* from \"author\" where \"authorID\" = " + TBSerarch.Text, conn);

NpgsqlDataAdapter da = new NpgsqlDataAdapter(cmd);

DataSet ds = new DataSet();

da.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

conn.Close();

}

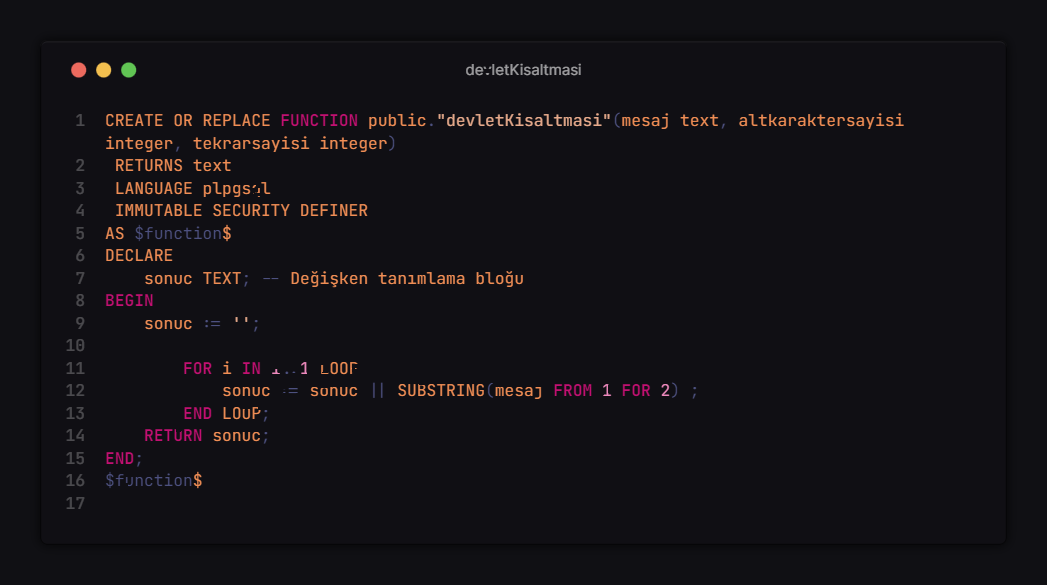
Fonkisyonlar

SearchWithName(name\_ text);

Text

Description automatically generated

devletKisaltmasi(mesaj text, altkaraktersayisi integer, tekrarsayisi integer);



fiatlarfiltre (id integer, bookname character varying, fiat money);

Text

Description automatically generated

findlanguage(inkey character varying);

Text

Description automatically generated

serchwithid(id integer);

Text

Description automatically generated

TRIGGERS

Trigger\_1 : fiat guncelledikten sonra yeni bir tabloda eski fiat ve yeni fiat ve degistirme zamani NewFiat tablosuna eklenir.

A screenshot of a computer

Description automatically generated with medium confidence

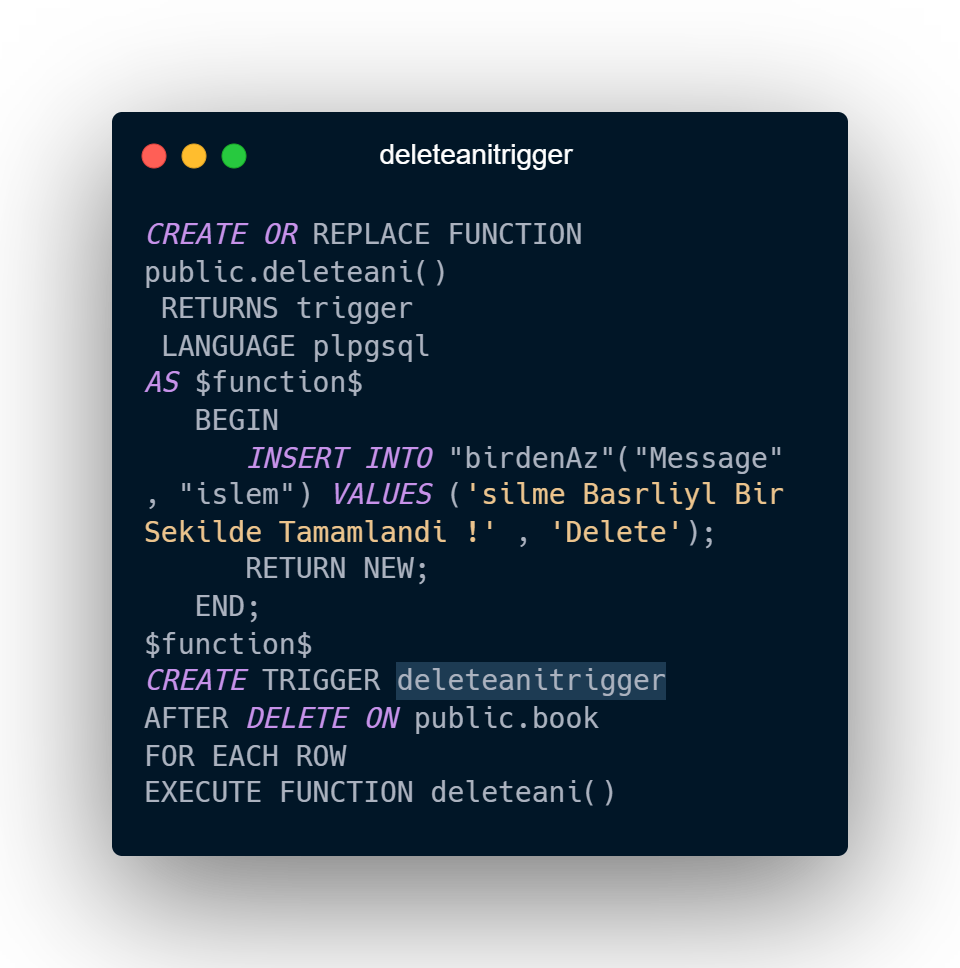
price\_trigger : kitab ekledikten Sonra yeni bir tabloda islem Tipi ve basarli mesaj cikilacak;



Guncellemeanitrigger : kitab ekledikten Sonra yeni bir tabloda islem Tipi ve basarli mesaj cikilacak;



Guncellemeanitrigger : kitab ekledikten Sonra yeni bir tabloda islem Tipi ve basarli mesaj cikilacak;



SQL Kodları

--

-- PostgreSQL database dump

--

-- Dumped from database version 12.12

-- Dumped by pg\_dump version 15rc2

SET statement\_timeout = 0;

SET lock\_timeout = 0;

SET idle\_in\_transaction\_session\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SELECT pg\_catalog.set\_config('search\_path', '', false);

SET check\_function\_bodies = false;

SET xmloption = content;

SET client\_min\_messages = warning;

SET row\_security = off;

--

-- Name: public; Type: SCHEMA; Schema: -; Owner: postgres

--

-- \*not\* creating schema, since initdb creates it

ALTER SCHEMA public OWNER TO postgres;

--

-- Name: devletKisaltmasi(text, integer, integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public."devletKisaltmasi"(mesaj text, altkaraktersayisi integer, tekrarsayisi integer) RETURNS text

    LANGUAGE plpgsql IMMUTABLE SECURITY DEFINER

    AS $$

DECLARE

    sonuc TEXT; -- Değişken tanımlama bloğu

BEGIN

    sonuc := '';

        FOR i IN 1..1 LOOP

            sonuc := sonuc || SUBSTRING(mesaj FROM 1 FOR 2) ;

        END LOOP;

    RETURN sonuc;

END;

$$;

ALTER FUNCTION public."devletKisaltmasi"(mesaj text, altkaraktersayisi integer, tekrarsayisi integer) OWNER TO postgres;

--

-- Name: fiatlarfiltre(); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.fiatlarfiltre() RETURNS TABLE(id integer, bookname character varying, fiat money)

    LANGUAGE plpgsql

    AS $$

BEGIN

    RETURN QUERY SELECT   "public"."book"."bookID",

         "public"."book"."bookName",

         "public"."price"."price"

FROM     "public"."price"

RIGHT OUTER JOIN "public"."book" ON "public"."price"."priceID" = "public"."book"."priceID"

                  ORDER BY  "public"."price"."price" ASC;

END;

$$;

ALTER FUNCTION public.fiatlarfiltre() OWNER TO postgres;

--

-- Name: findlanguage(character varying); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.findlanguage(inkey character varying) RETURNS TABLE(bookname character varying, languagename character varying, abb character varying)

    LANGUAGE plpgsql

    AS $$

BEGIN

    RETURN QUERY SELECT   "public"."book"."bookName",

         "public"."language"."languageName",

         "public"."language"."languageabb"

FROM     "public"."language"

RIGHT OUTER JOIN "public"."book"  ON "public"."language"."languageID" = "public"."book"."languageID"

                 WHERE "public"."language"."languageName" = inkey;

                 END;

$$;

ALTER FUNCTION public.findlanguage(inkey character varying) OWNER TO postgres;

--

-- Name: serchwithid(integer); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.serchwithid(iddd integer) RETURNS TABLE(numara integer, adi character varying, soyadi character varying)

    LANGUAGE plpgsql

    AS $$

BEGIN

    RETURN QUERY SELECT   "public"."book"."bookID",

         "public"."book"."bookName",

         "public"."book"."authorName"

FROM     "public"."book"

                 WHERE "public"."book"."bookID" = iddd;

END;

$$;

ALTER FUNCTION public.serchwithid(iddd integer) OWNER TO postgres;

--

-- Name: serchwithname(text); Type: FUNCTION; Schema: public; Owner: postgres

--

CREATE FUNCTION public.serchwithname(name\_ text) RETURNS TABLE(numara integer, adi character varying, soyadi character varying)

    LANGUAGE plpgsql

    AS $$

BEGIN

    RETURN QUERY SELECT   "public"."book"."bookID",

         "public"."book"."bookName",

         "public"."book"."authorName"

FROM     "public"."book"

                 WHERE "public"."book"."bookName" = name\_;

END;

$$;

ALTER FUNCTION public.serchwithname(name\_ text) OWNER TO postgres;

SET default\_tablespace = '';

SET default\_table\_access\_method = heap;

--

-- Name: author; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.author (

    "authorID" integer NOT NULL,

    "authorName" character varying,

    date text,

    "publisherID" integer,

    "publisherID2" integer

);

ALTER TABLE public.author OWNER TO postgres;

--

-- Name: book; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.book (

    "bookID" integer NOT NULL,

    "bookName" character varying(100),

    "authorName" character varying,

    "categoryID" integer,

    "editionID" integer,

    "languageID" integer,

    "placeID" integer,

    "priceID" integer,

    "publisherName" character varying,

    "authorID" integer,

    "publisherID" integer,

    "buyerID" integer,

    "borrowerID" integer

);

ALTER TABLE public.book OWNER TO postgres;

--

-- Name: bookANDcategory; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public."bookANDcategory" (

    "bookID" integer NOT NULL,

    "categoryID" integer NOT NULL

);

ALTER TABLE public."bookANDcategory" OWNER TO postgres;

--

-- Name: bookANDlanguage; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public."bookANDlanguage" (

    "bookID" integer NOT NULL,

    "languageID" integer NOT NULL

);

ALTER TABLE public."bookANDlanguage" OWNER TO postgres;

--

-- Name: bookToEdition; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public."bookToEdition" (

    "bookID" integer NOT NULL,

    "editionID" integer NOT NULL

);

ALTER TABLE public."bookToEdition" OWNER TO postgres;

--

-- Name: book\_bookID\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public."book\_bookID\_seq"

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public."book\_bookID\_seq" OWNER TO postgres;

--

-- Name: book\_bookID\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public."book\_bookID\_seq" OWNED BY public.book."bookID";

--

-- Name: borrower; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.borrower (

    "studentID" integer NOT NULL,

    "startDate" date,

    "finishDate" date

);

ALTER TABLE public.borrower OWNER TO postgres;

--

-- Name: borrowerProcess; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public."borrowerProcess" (

    "processID" integer NOT NULL,

    "startDate" date NOT NULL,

    "finishDate" date NOT NULL,

    "noteID" integer NOT NULL,

    "borrowerID" integer NOT NULL

);

ALTER TABLE public."borrowerProcess" OWNER TO postgres;

--

-- Name: buyProcess; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public."buyProcess" (

    "processID" integer NOT NULL,

    notes text,

    "buyerID" integer NOT NULL,

    "remainingMoney" money NOT NULL

);

ALTER TABLE public."buyProcess" OWNER TO postgres;

--

-- Name: buyProcess\_processID\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public."buyProcess\_processID\_seq"

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public."buyProcess\_processID\_seq" OWNER TO postgres;

--

-- Name: buyProcess\_processID\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public."buyProcess\_processID\_seq" OWNED BY public."buyProcess"."processID";

--

-- Name: buyer; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.buyer (

    "studentID" integer NOT NULL,

    "moneyID" integer NOT NULL

);

ALTER TABLE public.buyer OWNER TO postgres;

--

-- Name: category; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.category (

    "categoryID" integer NOT NULL,

    "categoryName" text

);

ALTER TABLE public.category OWNER TO postgres;

--

-- Name: country; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.country (

    "countryID" integer NOT NULL,

    "countryName" character varying NOT NULL,

    countryabb character varying NOT NULL,

    "languageID" integer NOT NULL,

    "publisherID" integer NOT NULL

);

ALTER TABLE public.country OWNER TO postgres;

--

-- Name: edition; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.edition (

    "editionID" integer NOT NULL

);

ALTER TABLE public.edition OWNER TO postgres;

--

-- Name: language; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.language (

    "languageID" integer NOT NULL,

    "languageName" character varying,

    languageabb character varying,

    "countryID" integer

);

ALTER TABLE public.language OWNER TO postgres;

--

-- Name: money; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.money (

    "moneyID" integer NOT NULL,

    money character varying

);

ALTER TABLE public.money OWNER TO postgres;

--

-- Name: notes; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.notes (

    "noteID" integer NOT NULL,

    "noteText" text,

    "borrowerID" integer NOT NULL,

    "ProcessID" integer NOT NULL,

    "buyerID" integer NOT NULL

);

ALTER TABLE public.notes OWNER TO postgres;

--

-- Name: notes\_noteID\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public."notes\_noteID\_seq"

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public."notes\_noteID\_seq" OWNER TO postgres;

--

-- Name: notes\_noteID\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public."notes\_noteID\_seq" OWNED BY public.notes."noteID";

--

-- Name: place; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.place (

    "placeID" integer NOT NULL,

    "placeAdress" text

);

ALTER TABLE public.place OWNER TO postgres;

--

-- Name: price; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.price (

    "priceID" integer NOT NULL,

    price money NOT NULL

);

ALTER TABLE public.price OWNER TO postgres;

--

-- Name: price\_priceID\_seq; Type: SEQUENCE; Schema: public; Owner: postgres

--

CREATE SEQUENCE public."price\_priceID\_seq"

    AS integer

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;

ALTER TABLE public."price\_priceID\_seq" OWNER TO postgres;

--

-- Name: price\_priceID\_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner: postgres

--

ALTER SEQUENCE public."price\_priceID\_seq" OWNED BY public.price."priceID";

--

-- Name: publisher; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.publisher (

    "publisherID" integer NOT NULL,

    "publisherName" character varying,

    "publisherAdress" text,

    "publisherMail" character varying,

    "countryID" integer

);

ALTER TABLE public.publisher OWNER TO postgres;

--

-- Name: student; Type: TABLE; Schema: public; Owner: postgres

--

CREATE TABLE public.student (

    "studentID" integer NOT NULL,

    "buyerID" integer,

    "browwerID" integer,

    "studentName" character varying,

    "studentDate" character varying,

    "StudentPhoneNo" character varying

);

ALTER TABLE public.student OWNER TO postgres;

--

-- Name: book bookID; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book ALTER COLUMN "bookID" SET DEFAULT nextval('public."book\_bookID\_seq"'::regclass);

--

-- Name: buyProcess processID; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."buyProcess" ALTER COLUMN "processID" SET DEFAULT nextval('public."buyProcess\_processID\_seq"'::regclass);

--

-- Name: notes noteID; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.notes ALTER COLUMN "noteID" SET DEFAULT nextval('public."notes\_noteID\_seq"'::regclass);

--

-- Name: price priceID; Type: DEFAULT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.price ALTER COLUMN "priceID" SET DEFAULT nextval('public."price\_priceID\_seq"'::regclass);

--

-- Data for Name: author; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.author VALUES

    (7, 'Bilal RASLEN', '01.01.2002', NULL, 55),

    (12, 'Bilal RASLEN', '01.01.2002', 5, 4),

    (8, 'avci 32', '01.2.1965', 6, 6),

    (6, 'asdf', '', 55, NULL),

    (1, 'Muhammed', '01.01.1987', 1, 1),

    (2, 'Ali', '01.01.1987', 2, 2),

    (3, 'Kayhan', '01.01.1987', 3, 3),

    (22, 'aswwww', '22.2.2022', 5, 5),

    (4, 'BLL', '01.01.2002', 1, NULL),

    (5, NULL, NULL, 6, 0);

--

-- Data for Name: book; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.book VALUES

    (5, 'asdfg', NULL, 33, 2, 2, 5, 3, NULL, 2, 2, NULL, NULL),

    (3, 'BName3', 'AName3', 2, 2, 2, 1, 1, '', 3, 1, NULL, NULL),

    (2, 'BName2', 'AName2', 2, 2, 2, 3, 3, '', 2, 3, NULL, NULL),

    (1, 'BookOne', '1', 1, 1, 1, NULL, 9, '', 1, 1, NULL, NULL),

    (4, 'BName4', 'AName4', 2, 1, 1, 5, 3, '', NULL, 2, NULL, NULL),

    (6, 'Altinci Kitap', NULL, 2, 2, 2, NULL, 4, NULL, 2, 2, NULL, NULL),

    (7, 'asd', '1', 1, 2, 1, NULL, 5, '1', 1, 1, NULL, NULL),

    (8, 'sd', NULL, 1, 1, 1, NULL, 6, NULL, 1, 1, NULL, NULL),

    (32, 'BIlal ', '1', 2, 4, 2, NULL, NULL, '1', 1, 1, NULL, NULL);

--

-- Data for Name: bookANDcategory; Type: TABLE DATA; Schema: public; Owner: postgres

--

--

-- Data for Name: bookANDlanguage; Type: TABLE DATA; Schema: public; Owner: postgres

--

--

-- Data for Name: bookToEdition; Type: TABLE DATA; Schema: public; Owner: postgres

--

--

-- Data for Name: borrower; Type: TABLE DATA; Schema: public; Owner: postgres

--

--

-- Data for Name: borrowerProcess; Type: TABLE DATA; Schema: public; Owner: postgres

--

--

-- Data for Name: buyProcess; Type: TABLE DATA; Schema: public; Owner: postgres

--

--

-- Data for Name: buyer; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.buyer VALUES

    (1, 1),

    (2, 2),

    (3, 3),

    (4, 4);

--

-- Data for Name: category; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.category VALUES

    (1, 'web'),

    (2, 'aps'),

    (9, 'web22'),

    (33, 'sssweb');

--

-- Data for Name: country; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.country VALUES

    (1, 'Turkey', 'TR', 1, 1),

    (2, 'Suriye', 'SYR', 2, 2);

--

-- Data for Name: edition; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.edition VALUES

    (1),

    (2),

    (3),

    (4);

--

-- Data for Name: language; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.language VALUES

    (1, 'turkce', 'tr', 1),

    (2, 'Arapca', 'ar', 2),

    (4, 'rusca', 'rc', 1),

    (3, 'Franch', 'FR', 2),

    (5, 'Farisca', 'FR', NULL),

    (9, 'ARABCA', 'ARB', NULL);

--

-- Data for Name: money; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.money VALUES

    (1, '22');

--

-- Data for Name: notes; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.notes VALUES

    (1, 'Makale, herhangi bir konuda, bir görüşü, bir düşünceyi savunmak ve kanıtlamak için yazılan yazılara denir.', 1, 1, 1),

    (2, 'Gazete ve dergilerde yayımlanır.', 2, 2, 2),

    (3, 'ilerde yayımlanır.', 3, 3, 3),

    (4, 'we cant googing to cahannam', 4, 4, 4);

--

-- Data for Name: place; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.place VALUES

    (3, '33:2022:B22:B.Blok'),

    (1, '33:2022:A22:A.Blok'),

    (5, '33:2022:B22:B.Blok');

--

-- Data for Name: price; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.price VALUES

    (1, '$55.00'),

    (3, '$37.00'),

    (2, '$123.00'),

    (4, '$34.00'),

    (5, '$43.00'),

    (6, '$344.00'),

    (7, '$54.00'),

    (8, '$66.00'),

    (9, '$56.00'),

    (10, '$88.00');

--

-- Data for Name: publisher; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.publisher VALUES

    (6, 'dar alshifa', '507 - 20th Ave. E.\nApt. 2A', 'islamicdarilnesir@gmail.com', 2),

    (66, 'we', 'we', 'cxwewe#ogr.tr', NULL),

    (9, 'ALi', 'SYR 2012', 'ajsj@gmail.com', NULL),

    (11, 'dd', 'd', 'd', NULL),

    (8, 'sdsdsd', 'sdsddsdssd', 'sssdsddsds', NULL),

    (3, 'islamic darilnesir', '507 - 20th Ave. E.\nApt. 2A', 'islamicdarilnesir@gmail.com', 2),

    (2, 'muaaz nesriyet', '722 Moss Bay Blvd.', 'muaaznesriyet@gmail.com', 1),

    (1, 'dar Publisher', 'Edgeham Hollow\nWinchester Way', 'darPublisher@edu.com', 1);

--

-- Data for Name: student; Type: TABLE DATA; Schema: public; Owner: postgres

--

INSERT INTO public.student VALUES

    (1, 1, 1, 'BLL', '1.2.1344', '+905383471275'),

    (2, 3, 2, 'ALL', '23.4.1876', '+905383471275'),

    (3, 4, 3, 'ML', '01.3.1976', '+905383471275'),

    (4, 5, 4, 'halit', 'ayse', '+905383471275'),

    (6, 7, 6, 'ulker', '2.2.2002', '+905383471275'),

    (5, 6, 5, 'mustafa', '1.1.2001', '+905383471275');

--

-- Name: book\_bookID\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."book\_bookID\_seq"', 23, true);

--

-- Name: buyProcess\_processID\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."buyProcess\_processID\_seq"', 1, false);

--

-- Name: notes\_noteID\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."notes\_noteID\_seq"', 1, false);

--

-- Name: price\_priceID\_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

--

SELECT pg\_catalog.setval('public."price\_priceID\_seq"', 1, false);

--

-- Name: bookANDcategory bookANDcategory\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookANDcategory"

    ADD CONSTRAINT "bookANDcategory\_pkey" PRIMARY KEY ("bookID");

--

-- Name: bookANDlanguage bookANDlanguage\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookANDlanguage"

    ADD CONSTRAINT "bookANDlanguage\_pkey" PRIMARY KEY ("languageID", "bookID");

--

-- Name: bookToEdition bookToEdition\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookToEdition"

    ADD CONSTRAINT "bookToEdition\_pkey" PRIMARY KEY ("bookID", "editionID");

--

-- Name: borrowerProcess borrowerProcess\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."borrowerProcess"

    ADD CONSTRAINT "borrowerProcess\_pkey" PRIMARY KEY ("processID");

--

-- Name: edition edition\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.edition

    ADD CONSTRAINT edition\_pkey PRIMARY KEY ("editionID");

--

-- Name: publisher publisher\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.publisher

    ADD CONSTRAINT publisher\_pkey PRIMARY KEY ("publisherID");

--

-- Name: student student\_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.student

    ADD CONSTRAINT student\_pkey PRIMARY KEY ("studentID");

--

-- Name: author unique\_author\_authorID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.author

    ADD CONSTRAINT "unique\_author\_authorID" PRIMARY KEY ("authorID");

--

-- Name: author unique\_author\_pubID2; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.author

    ADD CONSTRAINT "unique\_author\_pubID2" UNIQUE ("publisherID2");

--

-- Name: bookANDcategory unique\_bookANDcategory\_bookID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookANDcategory"

    ADD CONSTRAINT "unique\_bookANDcategory\_bookID" UNIQUE ("bookID");

--

-- Name: bookANDlanguage unique\_bookANDlanguage\_bookID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookANDlanguage"

    ADD CONSTRAINT "unique\_bookANDlanguage\_bookID" UNIQUE ("bookID");

--

-- Name: bookANDlanguage unique\_bookANDlanguage\_languageID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookANDlanguage"

    ADD CONSTRAINT "unique\_bookANDlanguage\_languageID" UNIQUE ("languageID");

--

-- Name: bookToEdition unique\_bookToEdition\_bookID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookToEdition"

    ADD CONSTRAINT "unique\_bookToEdition\_bookID" UNIQUE ("bookID");

--

-- Name: bookToEdition unique\_bookToEdition\_editionID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookToEdition"

    ADD CONSTRAINT "unique\_bookToEdition\_editionID" UNIQUE ("editionID");

--

-- Name: book unique\_book\_bookID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT "unique\_book\_bookID" PRIMARY KEY ("bookID");

--

-- Name: borrowerProcess unique\_borrowerProcess\_borrowerID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."borrowerProcess"

    ADD CONSTRAINT "unique\_borrowerProcess\_borrowerID" UNIQUE ("borrowerID");

--

-- Name: borrowerProcess unique\_borrowerProcess\_noteID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."borrowerProcess"

    ADD CONSTRAINT "unique\_borrowerProcess\_noteID" UNIQUE ("noteID");

--

-- Name: borrower unique\_borrower\_borrowerID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.borrower

    ADD CONSTRAINT "unique\_borrower\_borrowerID" PRIMARY KEY ("studentID");

--

-- Name: buyProcess unique\_buyProcess\_buyerID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."buyProcess"

    ADD CONSTRAINT "unique\_buyProcess\_buyerID" UNIQUE ("buyerID");

--

-- Name: buyProcess unique\_buyProcess\_processID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."buyProcess"

    ADD CONSTRAINT "unique\_buyProcess\_processID" PRIMARY KEY ("processID");

--

-- Name: buyer unique\_buyer\_buyerID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.buyer

    ADD CONSTRAINT "unique\_buyer\_buyerID" PRIMARY KEY ("studentID");

--

-- Name: buyer unique\_buyer\_moneyID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.buyer

    ADD CONSTRAINT "unique\_buyer\_moneyID" UNIQUE ("moneyID");

--

-- Name: category unique\_category\_categoryID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.category

    ADD CONSTRAINT "unique\_category\_categoryID" PRIMARY KEY ("categoryID");

--

-- Name: country unique\_country\_countryID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.country

    ADD CONSTRAINT "unique\_country\_countryID" PRIMARY KEY ("countryID");

--

-- Name: country unique\_country\_languageID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.country

    ADD CONSTRAINT "unique\_country\_languageID" UNIQUE ("languageID");

--

-- Name: country unique\_country\_publisherID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.country

    ADD CONSTRAINT "unique\_country\_publisherID" UNIQUE ("publisherID");

--

-- Name: edition unique\_edition\_editionID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.edition

    ADD CONSTRAINT "unique\_edition\_editionID" UNIQUE ("editionID");

--

-- Name: language unique\_language\_languageID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.language

    ADD CONSTRAINT "unique\_language\_languageID" PRIMARY KEY ("languageID");

--

-- Name: money unique\_money\_moneyID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.money

    ADD CONSTRAINT "unique\_money\_moneyID" PRIMARY KEY ("moneyID");

--

-- Name: notes unique\_notes\_borrowerID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.notes

    ADD CONSTRAINT "unique\_notes\_borrowerID" UNIQUE ("borrowerID");

--

-- Name: notes unique\_notes\_buyerID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.notes

    ADD CONSTRAINT "unique\_notes\_buyerID" UNIQUE ("buyerID");

--

-- Name: notes unique\_notes\_buyerProcess; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.notes

    ADD CONSTRAINT "unique\_notes\_buyerProcess" UNIQUE ("ProcessID");

--

-- Name: notes unique\_notes\_noteNo; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.notes

    ADD CONSTRAINT "unique\_notes\_noteNo" PRIMARY KEY ("noteID");

--

-- Name: place unique\_place\_placeID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.place

    ADD CONSTRAINT "unique\_place\_placeID" PRIMARY KEY ("placeID");

--

-- Name: price unique\_price\_priceID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.price

    ADD CONSTRAINT "unique\_price\_priceID" PRIMARY KEY ("priceID");

--

-- Name: student unique\_student\_studentID; Type: CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.student

    ADD CONSTRAINT "unique\_student\_studentID" UNIQUE ("studentID");

--

-- Name: index\_bookID; Type: INDEX; Schema: public; Owner: postgres

--

CREATE INDEX "index\_bookID" ON public."bookToEdition" USING btree ("bookID");

--

-- Name: index\_categoryID; Type: INDEX; Schema: public; Owner: postgres

--

CREATE INDEX "index\_categoryID" ON public."bookANDcategory" USING btree ("categoryID");

--

-- Name: index\_countryID; Type: INDEX; Schema: public; Owner: postgres

--

CREATE INDEX "index\_countryID" ON public.language USING btree ("countryID");

--

-- Name: index\_countryID1; Type: INDEX; Schema: public; Owner: postgres

--

CREATE INDEX "index\_countryID1" ON public.publisher USING btree ("countryID");

--

-- Name: index\_languageID; Type: INDEX; Schema: public; Owner: postgres

--

CREATE INDEX "index\_languageID" ON public."bookANDlanguage" USING btree ("languageID");

--

-- Name: book trigger1; Type: TRIGGER; Schema: public; Owner: postgres

--

CREATE TRIGGER trigger1 BEFORE DELETE ON public.book FOR EACH ROW EXECUTE FUNCTION unique\_key\_recheck();

--

-- Name: book author\_with\_book; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT author\_with\_book FOREIGN KEY ("authorID") REFERENCES public.author("authorID") MATCH FULL;

--

-- Name: bookANDcategory book\_with\_bookANDcategory; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookANDcategory"

    ADD CONSTRAINT "book\_with\_bookANDcategory" FOREIGN KEY ("bookID") REFERENCES public.book("bookID") MATCH FULL;

--

-- Name: book book\_with\_publisher; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT book\_with\_publisher FOREIGN KEY ("publisherID") REFERENCES public.publisher("publisherID") MATCH FULL;

--

-- Name: book borrower\_with\_book; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT borrower\_with\_book FOREIGN KEY ("borrowerID") REFERENCES public.borrower("studentID") MATCH FULL;

--

-- Name: book buyer\_with\_book; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT buyer\_with\_book FOREIGN KEY ("buyerID") REFERENCES public.buyer("studentID") MATCH FULL;

--

-- Name: book category\_with\_book; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT category\_with\_book FOREIGN KEY ("categoryID") REFERENCES public.category("categoryID") MATCH FULL;

--

-- Name: bookANDcategory category\_with\_bookANDcategory; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public."bookANDcategory"

    ADD CONSTRAINT "category\_with\_bookANDcategory" FOREIGN KEY ("categoryID") REFERENCES public.category("categoryID") MATCH FULL;

--

-- Name: publisher country\_with\_poblisher; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.publisher

    ADD CONSTRAINT country\_with\_poblisher FOREIGN KEY ("countryID") REFERENCES public.country("countryID") MATCH FULL;

--

-- Name: book editon\_with\_book; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT editon\_with\_book FOREIGN KEY ("editionID") REFERENCES public.edition("editionID") MATCH FULL;

--

-- Name: book language\_with\_book; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT language\_with\_book FOREIGN KEY ("languageID") REFERENCES public.language("languageID") MATCH FULL;

--

-- Name: country language\_with\_country; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.country

    ADD CONSTRAINT language\_with\_country FOREIGN KEY ("languageID") REFERENCES public.language("languageID") MATCH FULL;

--

-- Name: borrower note\_with\_borrower; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.borrower

    ADD CONSTRAINT note\_with\_borrower FOREIGN KEY ("studentID") REFERENCES public.notes("borrowerID") MATCH FULL;

--

-- Name: buyer note\_with\_buyer; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.buyer

    ADD CONSTRAINT note\_with\_buyer FOREIGN KEY ("studentID") REFERENCES public.notes("buyerID") MATCH FULL;

--

-- Name: book place\_with\_book; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT place\_with\_book FOREIGN KEY ("placeID") REFERENCES public.place("placeID") MATCH FULL;

--

-- Name: book price\_with\_book; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.book

    ADD CONSTRAINT price\_with\_book FOREIGN KEY ("priceID") REFERENCES public.price("priceID") MATCH FULL;

--

-- Name: borrower student\_with\_browwer; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.borrower

    ADD CONSTRAINT student\_with\_browwer FOREIGN KEY ("studentID") REFERENCES public.student("studentID") MATCH FULL ON UPDATE CASCADE ON DELETE CASCADE;

--

-- Name: buyer student\_with\_buyer; Type: FK CONSTRAINT; Schema: public; Owner: postgres

--

ALTER TABLE ONLY public.buyer

    ADD CONSTRAINT student\_with\_buyer FOREIGN KEY ("studentID") REFERENCES public.student("studentID") MATCH FULL ON UPDATE CASCADE ON DELETE CASCADE;

--

-- Name: SCHEMA public; Type: ACL; Schema: -; Owner: postgres

--

REVOKE USAGE ON SCHEMA public FROM PUBLIC;

GRANT ALL ON SCHEMA public TO PUBLIC;

--

-- PostgreSQL database dump complete

--

# Linkler

Píoject linfi on MediaFiíe : <https://www.mediafire.com/folder/8400rs4tfswwu/VTYSProject>

# Kaynaklar

### Fotogíaﬂaí : [Tıkla](https://iconarchive.com/)

### Celal Ceken GitHup [: https://github.com/celalceken/DatabaseManagementSystems](:%20https:/github.com/celalceken/DatabaseManagementSystems)

### Diagíam : [Tıkla](https://app.diagrams.net/)

### Code İmages : [Tıkla](https://carbon.now.sh/)