T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİ TABANI DERSİ

PROJE

G171210016

HİLAL İLAYDA TEZGİDER

2C GRUBU

hilal.tezgider@ogr.sakarya.edu.tr

Kaynak Kodlar: https://github.com/ilaydaTezgider/LibraryApp

Tanıtım Videosu: https://youtu.be/KH9iLHjUhw4

UYGULAMANIN TANITIMI

Bir kütüphane içerisinde kütüphane kullanıcılarının ve kütüphane çalışanlarının kullanabileceği ve belirli fonksiyonlara sahip olan bir uygulama tasarımıdır. Bu fonksiyonlar; kullanıcıların kitap listesi ve kiralama geçmişini görüntüleyebilmesi, sisteme kitap ekleyebilmesi, sistemden kitap kiralayabilmesi, kitap bilgilerini sisteme tanımlayabilmesi gibi işlemlerdir. Bunlara ek olarak admin yetkisine sahip olanların ise kitap silebileceği sistemdir. Bu uygulama sayesinde kütüphanelerde manuel olarak gerçekleştirilen işlemler bir otomasyon ile gerçekleştirilerek gündelik hayatı kolaylaştıracaktır.

İŞ KURALLARI

Kişiler id, ad, soyad, kullanıcı ve yazar bilgisini barındırır.

Bir kullanıcı hiçbir role sahip olmayabilir veya birden fazla role sahip olabilir.

Roller id ve rol adı bilgilerini barındırır.

Bir rol birden fazla kullanıcıya ait olabilir veya hiçbir kullanıcıya ait olmayabilir.

Sisteme kayıtlı yazar veya kullanıcı şeklinde 2 tür kişi olabilir. Yazarlar id ve kitap numarası bilgilerini, kullanıcılar ise id, ad, şifre ve kitap numarası bilgilerini barındırır.

Kiralananlar id, dönüş tarihi, fiyat, iade durumu, kitap ve kullanıcı id bilgilerini barındırır.

Bir kullanıcı birden fazla kiralama işlemi yapabilir veya hiç kiralama işlemi yapmayabilir. Bir kiralama işlemi bir kullanıcıya ait olmak zorundadır.

Bir kitap için birden fazla kiralama işlemi yapılabilir veya hiç kiralama işlemi yapılmayabilir. Bir kiralama işleminin yalnızca bir kitaba ait olması zorunludur.

Kitaplar id, ad, yayın tarihi, yazar, yayınevi, kitap, dil ve kiralama ücreti bilgilerine sahiptir.

Kitaplar yalnızca bir tip bilgisi içermektedir. Bir tip birden fazla kitaplara ait olabilir.

Tipler id ve ad bilgilerini barındırır.

Kitapların yalnızca bir dili vardır. Bir dil birden fazla kitaba ait olabilir veya hiçbir kitaba ait olmayabilir.

Diller id ve dil adı bilgisi barındırır.

Kategoriler id ve name bilgilerini barındır.

Kitap kategorileri id, kategori ve kitap id bilgileri barındır.

Bir kitabın en az bir kategorisi bulunmak zorundadır. Bir kategori ise birden fazla kitap için tanımlanabilirken hiçbir kitap için tanımlanmamış olabilir.

Yayınevi id, isim ve adres id bilgilerine sahiptir.

Bir yayınevi hiçbir kitaba ait olmayabilir veya birden fazla kitaba ait olabilir. Bir kitap bir yayınevine aittir.

Adres id, ilçe ve açık adres bilgileri barındırır.

Bir adres bir veya daha çok yayınevine sahip olabilir. Bir yayınevi bir veya sıfır adrese sahip olabilir.

İlçeler id, ad ve şehir id bilgilerini barındırır.

Bir ilçe hiçbir adres içermeyebilir veya birden fazla adres içerebilir. Bir adres mutlaka bir ilçe içermek zorundadır.

İller id ve isim bilgileri barındırır.

İller hiçbir ilçeye sahip olmayabilir veya birden çok ilçeye sahip olabilir. İlçeler bir ile ait olmak zorundadır.

İLİŞKİSEL ŞEMA

Hire (**hireID:integer**, personTime:date, price:integer, isBack:smallint, bookId:integer, personId:integer)

BookType (**typeId:integer**, typeName:character varying(50))

Book (**bookId:integer**, bookName:character varying(50), printeryDate:date, writerId:integer, printeryId:integer, bookTypeId:integer, langId:integer, hirePrice:integer)

BookLang (**langId:integer**, language:character varying(50))

Adress (**adressId:integer**, townId:integer, adres:character varying(150))

City (**cityId:integer**, cityName:character varying(50))

Town (**townId:integer**, townName:character varying(50), cityId:integer)

Printery (**printeryId:integer**, printeryName:character varying(50), adressId:integer)

Person (**personId:integer**, Name:character varying(50), Surname:character varying(50), isMember:smallint, isWriter:smallint)

Writer (**personId:integer**, bookNumber:integer)

UserRole (**roleId:integer**, **personId:integer**)

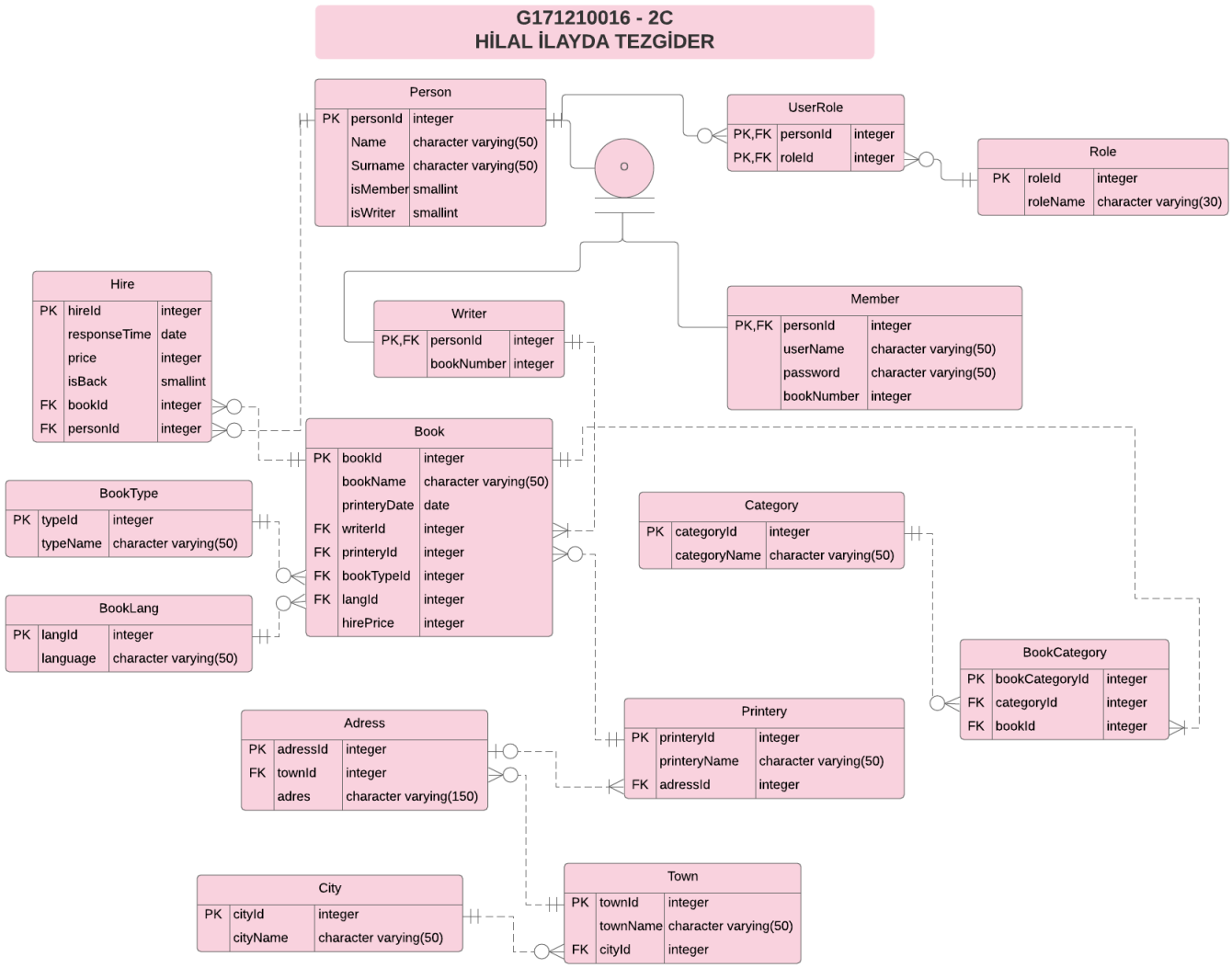
Member (**personId:integer**, username:character varying(50), password:character varying(50), bookNumber:integer)

Role (**roleId:integer**, roleName:character varying(50))

Category (**categoryId:integer**, categoryName:character varying(50))

BookCategory (**bookCategoryId:integer**, categoryId:integer, bookId:integer)

VARLIK BAĞINTI MODELİ



SQL İFADELERİ

```
CREATE SCHEMA libapp;

ALTER SCHEMA libapp OWNER TO postgres;

CREATE FUNCTION libapp.f_prc_get_book_categories() RETURNS TABLE(id integer,
categoryname character varying)

   LANGUAGE plpgsql

   AS $$

begin

return query

        select * from libapp.category;

end

$$;

ALTER FUNCTION libapp.f_prc_get_book_categories() OWNER TO postgres;

CREATE FUNCTION libapp.f_prc_get_book_types() RETURNS TABLE(typename character
varying)

   LANGUAGE plpgsql

   AS $$

begin

return query

        select booktype.typename from libapp.booktype;

end

$$;

ALTER FUNCTION libapp.f_prc_get_book_types() OWNER TO postgres;

CREATE FUNCTION libapp.f_prc_get_books() RETURNS TABLE(id integer, bookname character
varying, writername text, printeryname character varying, printerydate date, hireprice integer)

   LANGUAGE plpgsql

   AS $$

begin
```

```
return query
	select book.bookid,book.bookname,(writerinf.name || ' ' || writerinf.surname) as writername,printery.printeryname,book.printerydate,book.hireprice
	from libapp.book as book
	inner join (
		select person.personid,person.name,person.surname from libapp.writer
		inner join libapp.person on person.personid = writer.personid
	) as writerinf on book.writerid = writerinf.personid
	inner join libapp.printery on book.printeryid = printery.printeryid;
end
$$;
ALTER FUNCTION libapp.f_prc_get_books() OWNER TO postgres;
CREATE FUNCTION libapp.f_prc_get_books_by_param(p_book_name character varying) RETURNS TABLE(id integer, bookname character varying, writername text, printeryname character varying, printerydate date, hireprice integer)
   LANGUAGE plpgsql
   AS $$
begin
return query
	select book.bookid,book.bookname,(writerinf.name || ' ' || writerinf.surname) as writername,printery.printeryname,book.printerydate,book.hireprice
	from libapp.book as book
	inner join (
		select person.personid,person.name,person.surname from libapp.writer
		inner join libapp.person on person.personid = writer.personid
	) as writerinf on book.writerid = writerinf.personid
	inner join libapp.printery on book.printeryid = printery.printeryid
	where lower(book.bookname) like lower('%' || p_book_name ||'%');
```

```
end
$$;
ALTER FUNCTION libapp.f_prc_get_books_by_param(p_book_name character varying) OWNER TO postgres;
CREATE FUNCTION libapp.f_prc_get_city() RETURNS TABLE(cityname character varying)
   LANGUAGE plpgsql
   AS $$
begin
return query
        select city.cityname from libapp.city;
end
$$;
ALTER FUNCTION libapp.f_prc_get_city() OWNER TO postgres;
CREATE FUNCTION libapp.f_prc_get_hired_book(p_username character varying) RETURNS TABLE(hi integer, bn character varying, rt date, hp integer, d text)
   LANGUAGE plpgsql
   AS $$
begin
return query
        select hire.hireid, book.bookname, hire.responsetime, book.hireprice,
        (case isback when 0 then 'İade edilmemiş' else 'İade edilmiş' end) durumu
         from libapp.hire
        inner join libapp.member on( member.personid =hire.personid and member.username= p_username)
        inner join libapp.book on book.bookid=hire.bookid;
end
$$;
```

```
ALTER FUNCTION libapp.f_prc_get_hired_book(p_username character varying) OWNER TO postgres;

CREATE FUNCTION libapp.f_prc_get_lang() RETURNS TABLE(langid integer, lang character varying)

    LANGUAGE plpgsql

    AS $$

begin

return query

        select * from libapp.booklang;

end

$$;

ALTER FUNCTION libapp.f_prc_get_lang() OWNER TO postgres;

CREATE FUNCTION libapp.f_prc_get_printery() RETURNS TABLE(printeryname character varying)

    LANGUAGE plpgsql

    AS $$

begin

return query

        select printery.printeryname from libapp.printery;

end

$$;

ALTER FUNCTION libapp.f_prc_get_printery() OWNER TO postgres;

CREATE FUNCTION libapp.f_prc_get_town(p_cityname character varying) RETURNS TABLE(townname character varying)

    LANGUAGE plpgsql

    AS $$

declare v_cursor refcursor;

begin
```

```
return query
	select town.townname from "libapp"."town"
inner join "libapp"."city" on ("city".cityname=p_cityname and "city".cityid="town".cityid);
end
$$;
ALTER FUNCTION libapp.f_prc_get_town(p_cityname character varying) OWNER TO postgres;
CREATE FUNCTION libapp.f_prc_login(p_username character varying, p_password character varying) RETURNS TABLE(personid integer, name character varying, surname character varying, ismember smallint, iswriter smallint, personid2 integer, username character varying, password character varying, booknumber integer, roleid integer)
   LANGUAGE plpgsql
   AS $$
declare v_cursor refcursor;
begin
return query
	select person.*, "member".*, (select userrole.roleid from libapp.userrole where userrole.personid=person.personid)
from libapp."person" as person
		inner join (
			select * from libapp."member"
			where "member"."username" = p_username
			and "member"."password" = p_password) as member
		on member.personid = person.personid
	where person.ismember = 1;
end
$$;
ALTER FUNCTION libapp.f_prc_login(p_username character varying, p_password character varying) OWNER TO postgres;
CREATE FUNCTION libapp.getbookid() RETURNS integer
```

```
   LANGUAGE plpgsql
   AS $$
declare bookid int;
begin
        select max(book.bookid) into bookid from libapp.book;
        return bookid;
end
$$;
ALTER FUNCTION libapp.getbookid() OWNER TO postgres;
CREATE FUNCTION libapp.getbooktypeid(p_booktype character varying) RETURNS integer
   LANGUAGE plpgsql
   AS $$
declare booktypeid int;
begin
        select booktype.typeid into booktypeid
        from libapp.booktype
        where booktype.typename = p_booktype;
        return booktypeid;
end;
$$;
ALTER FUNCTION libapp.getbooktypeid(p_booktype character varying) OWNER TO postgres;
CREATE FUNCTION libapp.getcategoryid(p_category character varying) RETURNS integer
   LANGUAGE plpgsql
   AS $$
declare p_categoryid int;
begin
        select category.categoryid into p_categoryid
```

```
        from libapp.category
        where category.categoryname = p_category;
        return p_categoryid;
end;
$$;
ALTER FUNCTION libapp.getcategoryid(p_category character varying) OWNER TO postgres;
CREATE FUNCTION libapp.getprinteryid(p_printeryname character varying) RETURNS integer
  LANGUAGE plpgsql
  AS $$
declare printeryid int;
begin
        select printery.printeryid into printeryid
        from libapp.printery
        where printery.printeryname = p_printeryname;
        return printeryid;
end;
$$;
ALTER FUNCTION libapp.getprinteryid(p_printeryname character varying) OWNER TO postgres;
CREATE FUNCTION libapp.getwriterid(p_writername character varying, p_writersurname
character varying) RETURNS integer
  LANGUAGE plpgsql
  AS $$
declare writerid int;
begin
        select writer.personid into writerid
        from libapp.writer
        inner join libapp.person on writer.personid = person.personid
```

```
	where person.name = p_writername and person.surname = p_writersurname and
person.iswriter = 1;

	return writerid;

end;

$$;

ALTER FUNCTION libapp.getwriterid(p_writername character varying, p_writersurname
character varying) OWNER TO postgres;

CREATE FUNCTION libapp.isexistswriter(p_writername character varying, p_writersurname
character varying) RETURNS integer

   LANGUAGE plpgsql

   AS $$

declare isexists int;

begin

	select  count(*) into isexists

	from libapp.writer

	inner join libapp.person on writer.personid = person.personid

	where person.name = p_writername and person.surname = p_writersurname and
person.iswriter = 1;

	return isexists;

end;

$$;

ALTER FUNCTION libapp.isexistswriter(p_writername character varying, p_writersurname
character varying) OWNER TO postgres;

CREATE PROCEDURE libapp.prc_dml_book(p_bookid integer, p_dmltype character varying,
p_bookname character varying, p_printerydate character varying, p_writername character
varying, p_writersurname character varying, p_printery character varying, p_category character
varying, p_typename character varying, p_hireprice integer, p_langid integer)

   LANGUAGE plpgsql

   AS $$

declare category varchar(50);
```

```
                    categoryid int;
                    bookid int;
                    rec record;
begin
        if(p_dmltype = 'i') then
                if((select libapp.isexistswriter(p_writername,p_writersurname)) = 0) then
                        insert into libapp.person
                                (name
                                ,surname
                                ,ismember
                                ,iswriter)
                         values
                                (p_writername,p_writersurname,0,1);
                insert into libapp.book
                                (bookname
                                ,printerydate
                                ,writerid
                                ,printeryid
                                ,booktypeid
                                ,hireprice
                                ,langid)
                        values
                                ( p_bookname
                                ,to_date(p_printerydate , 'YYYY-MM-DD')
                                ,(select libapp.getwriterid(p_writername,p_writersurname))
                                ,(select libapp.getprinteryid(p_printery))
                                ,(select libapp.getbooktypeid(p_typename))
```

```
                        ,p_hireprice
                        ,p_langid);
            end if;

                for rec in (select unnest(string_to_array(p_category, ';')))
            loop
                select libapp.getcategoryid(rec.unnest) into categoryid;
                select libapp.getbookid() into bookid;

                insert into libapp.bookcategory
                                (categoryid
                                ,bookid)
                            values(
                                categoryid,
                                bookid);
            end loop;
        elsif(p_dmltype = 'd')  then
            if((select count(*) from libapp.hire where hire.bookid=p_bookid and
hire.isback='0')=0) then
                delete from libapp.hire where hire.bookid=p_bookid;
                delete from libapp.bookcategory where bookcategory.bookid=p_bookid;
                delete from libapp.book where book.bookid=p_bookid;
            else raise 'Bu kitap bir kullanıcı tarafından kiralanmış!';
            end if;
        end if;


end
```

```
$$;

ALTER PROCEDURE libapp.prc_dml_book(p_bookid integer, p_dmltype character varying,
p_bookname character varying, p_printerydate character varying, p_writername character
varying, p_writersurname character varying, p_printery character varying, p_category character
varying, p_typename character varying, p_hireprice integer, p_langid integer) OWNER TO
postgres;

CREATE PROCEDURE libapp.prc_dml_hire(p_dmltype character varying, p_bookid integer,
p_username character varying, p_responsetime date, p_isback smallint, p_price integer,
p_hireid integer)

   LANGUAGE plpgsql

   AS $$

declare userid int;

begin

        select "member".personid into userid from libapp.member where
"member".username=p_username;

        if(p_dmltype='i') then

                if((select count(*) from libapp.hire where hire.personid = userid and hire.isback =
0) < 2) then

                        insert into libapp.hire

                                (responsetime

                                ,price

                                ,isback

                                ,bookid

                                ,personid)

                        values

                                (p_responsetime,

                                p_price,

                                p_isback,

                                p_bookid,
```

```
                                        (select "member".personid from libapp.member where
"member".username=p_username));
                else
                        raise 'zaten 2 tane kitap kiralamışsın';
                end if;
        elsif(p_dmltype = 'u') then
                if((select hire.isback from libapp.hire where hire.hireid = p_hireid) = 0 ) then
                                update libapp.hire set isback=1
                                where hire.hireid = p_hireid;
                else
                                raise 'bu kitap İade edilmis';
                end if;
        end if;


end;
$$;
ALTER PROCEDURE libapp.prc_dml_hire(p_dmltype character varying, p_bookid integer,
p_username character varying, p_responsetime date, p_isback smallint, p_price integer,
p_hireid integer) OWNER TO postgres;
CREATE PROCEDURE libapp.prc_dml_member(p_dmltype character varying, p_name character
varying, p_surname character varying, p_username character varying, p_password character
varying)
   LANGUAGE plpgsql
   AS $$
begin
        if(p_dmltype = 'i') then
                if((select count(*) from libapp.member where
"member".username=p_username) = 0) then
                        insert into libapp.person
```

```
                    ("name"
                    ,surname
                    ,ismember
                    ,iswriter)
            values
                    (p_name,p_surname,1,0);
                insert into libapp.member
                    (personid
                    ,username
                    ,"password"
                    ,booknumber)
            values
                    (
                    (select max(personid) from libapp.person),
                    p_username,
                    p_password,
                    0
                    );

            insert into libapp.userrole
                (personid,
                 roleid)
        values ((select personid from libapp.member where username=p_username),
2);

    else
            raise 'bu kullanici adi kayitli';
```

```
                end if;
        end if;
end
$$;
ALTER PROCEDURE libapp.prc_dml_member(p_dmltype character varying, p_name character varying, p_surname character varying, p_username character varying, p_password character varying) OWNER TO postgres;
CREATE PROCEDURE libapp.prc_dml_printery(p_printeryname character varying, p_townname character varying, p_cityname character varying, p_adress character varying)
  LANGUAGE plpgsql
  AS $$
begin
        insert into libapp.adress
                        (townid
                        ,adres)
                values(
                        (select town.townid from libapp.town
                        inner join libapp.city on city.cityid = town.cityid
                        where town.townname = p_townname and city.cityname = p_cityname),
                        p_adress
                        );
        insert into libapp.printery
       (printeryname
       ,adressid)
   values(
       p_printeryname,
                (select max(adressid) from libapp.adress)
```

```
                    );
end;
$$;
ALTER PROCEDURE libapp.prc_dml_printery(p_printeryname character varying, p_townname character varying, p_cityname character varying, p_adress character varying) OWNER TO postgres;
CREATE FUNCTION libapp.t_member_book_inc_func() RETURNS trigger
   LANGUAGE plpgsql
   AS $$
begin
                if(((new.isback) > (old.isback)) and (new.isback) = 1)
                then

                        update libapp.member
                        set booknumber = booknumber + 1
                        where "member".personid=(new.personid);
                end if;
return null;
end
$$;
ALTER FUNCTION libapp.t_member_book_inc_func() OWNER TO postgres;
CREATE FUNCTION libapp.t_person_delete_func() RETURNS trigger
   LANGUAGE plpgsql
   AS $$
begin
if(old.ismember = 1) then

        delete from libapp.member
```

```
        where personid = (old.personid);
        delete from libapp.person
        where personid = (old.personid);
end if;
return null;
end
$$;
ALTER FUNCTION libapp.t_person_delete_func() OWNER TO postgres;
CREATE FUNCTION libapp.t_person_type_func() RETURNS trigger
   LANGUAGE plpgsql
   AS $$
begin
if(new.iswriter = 1) then

        insert into libapp.writer(personid,booknumber)
        values(
        (new.personid),
        0);
end if;
return null;
end
$$;
ALTER FUNCTION libapp.t_person_type_func() OWNER TO postgres;
CREATE FUNCTION libapp.t_writer_book_inc_func() RETURNS trigger
   LANGUAGE plpgsql
   AS $$
begin
```

```
                update libapp.writer
                set booknumber = booknumber + 1
                where writer.personid=new.writerid;
return null;
end
$$;
ALTER FUNCTION libapp.t_writer_book_inc_func() OWNER TO postgres;
SET default_tablespace = '';
SET default_table_access_method = heap;
CREATE TABLE libapp.adress (
    adressid integer NOT NULL,
    townid integer NOT NULL,
    adres character varying(150)
);
ALTER TABLE libapp.adress OWNER TO postgres;
CREATE SEQUENCE libapp.adress_adressid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.adress_adressid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.adress_adressid_seq OWNED BY libapp.adress.adressid;
CREATE TABLE libapp.book (
    bookid integer NOT NULL,
    bookname character varying(50) NOT NULL,
```

```
    printerydate date NOT NULL,
    writerid integer NOT NULL,
    printeryid integer NOT NULL,
    booktypeid integer NOT NULL,
    hireprice integer NOT NULL,
    langid integer
);
ALTER TABLE libapp.book OWNER TO postgres;
CREATE SEQUENCE libapp.book_bookid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.book_bookid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.book_bookid_seq OWNED BY libapp.book.bookid;
CREATE TABLE libapp.bookcategory (
    bookcategoryid integer NOT NULL,
    categoryid integer NOT NULL,
    bookid integer NOT NULL
);
ALTER TABLE libapp.bookcategory OWNER TO postgres;
CREATE SEQUENCE libapp.bookcategory_bookcategoryid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
```

```
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.bookcategory_bookcategoryid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.bookcategory_bookcategoryid_seq OWNED BY
libapp.bookcategory.bookcategoryid;
CREATE TABLE libapp.booklang (
    langid integer NOT NULL,
    language character varying(50) NOT NULL
);
ALTER TABLE libapp.booklang OWNER TO postgres;
CREATE TABLE libapp.booktype (
    typeid integer NOT NULL,
    typename character varying(50) NOT NULL
);
ALTER TABLE libapp.booktype OWNER TO postgres;
CREATE SEQUENCE libapp.booktype_typeid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.booktype_typeid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.booktype_typeid_seq OWNED BY libapp.booktype.typeid;
CREATE TABLE libapp.category (
    categoryid integer NOT NULL,
```

```
    categoryname character varying(50) NOT NULL
);
ALTER TABLE libapp.category OWNER TO postgres;
CREATE SEQUENCE libapp.category_categoryid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.category_categoryid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.category_categoryid_seq OWNED BY libapp.category.categoryid;
CREATE TABLE libapp.city (
    cityid integer NOT NULL,
    cityname character varying(50) NOT NULL
);
ALTER TABLE libapp.city OWNER TO postgres;
CREATE SEQUENCE libapp.city_cityid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.city_cityid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.city_cityid_seq OWNED BY libapp.city.cityid;
CREATE TABLE libapp.hire (
```

```
    hireid integer NOT NULL,
    responsetime date NOT NULL,
    price integer NOT NULL,
    isback smallint NOT NULL,
    bookid integer NOT NULL,
    personid integer NOT NULL
);
ALTER TABLE libapp.hire OWNER TO postgres;
CREATE SEQUENCE libapp.hire_hireid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.hire_hireid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.hire_hireid_seq OWNED BY libapp.hire.hireid;
CREATE TABLE libapp.member (
    personid integer NOT NULL,
    username character varying(50) NOT NULL,
    password character varying(50) NOT NULL,
    booknumber integer NOT NULL
);
ALTER TABLE libapp.member OWNER TO postgres;
CREATE TABLE libapp.person (
    personid integer NOT NULL,
    name character varying(50) NOT NULL,
```

```
    surname character varying(50) NOT NULL,
    ismember smallint NOT NULL,
    iswriter smallint NOT NULL
);
ALTER TABLE libapp.person OWNER TO postgres;
CREATE SEQUENCE libapp.person_personid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.person_personid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.person_personid_seq OWNED BY libapp.person.personid;
CREATE TABLE libapp.printery (
    printeryid integer NOT NULL,
    printeryname character varying(50) NOT NULL,
    adressid integer
);
ALTER TABLE libapp.printery OWNER TO postgres;
CREATE SEQUENCE libapp.printery_printeryid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
```

```
ALTER TABLE libapp.printery_printeryid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.printery_printeryid_seq OWNED BY libapp.printery.printeryid;
CREATE TABLE libapp.role (
    roleid integer NOT NULL,
    rolename character varying(30) NOT NULL
);
ALTER TABLE libapp.role OWNER TO postgres;
CREATE TABLE libapp.town (
    townid integer NOT NULL,
    townname character varying(50) NOT NULL,
    cityid integer NOT NULL
);
ALTER TABLE libapp.town OWNER TO postgres;
CREATE SEQUENCE libapp.town_townid_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE libapp.town_townid_seq OWNER TO postgres;
ALTER SEQUENCE libapp.town_townid_seq OWNED BY libapp.town.townid;
CREATE TABLE libapp.userrole (
    personid integer NOT NULL,
    roleid integer NOT NULL
);
ALTER TABLE libapp.userrole OWNER TO postgres;
```

```
CREATE TABLE libapp.writer (
    personid integer NOT NULL,
    booknumber integer NOT NULL
);
ALTER TABLE libapp.writer OWNER TO postgres;
ALTER TABLE ONLY libapp.adress ALTER COLUMN adressid SET DEFAULT
nextval('libapp.adress_adressid_seq'::regclass);
ALTER TABLE ONLY libapp.book ALTER COLUMN bookid SET DEFAULT
nextval('libapp.book_bookid_seq'::regclass);
ALTER TABLE ONLY libapp.bookcategory ALTER COLUMN bookcategoryid SET DEFAULT
nextval('libapp.bookcategory_bookcategoryid_seq'::regclass);
ALTER TABLE ONLY libapp.booktype ALTER COLUMN typeid SET DEFAULT
nextval('libapp.booktype_typeid_seq'::regclass);
ALTER TABLE ONLY libapp.category ALTER COLUMN categoryid SET DEFAULT
nextval('libapp.category_categoryid_seq'::regclass);
ALTER TABLE ONLY libapp.city ALTER COLUMN cityid SET DEFAULT
nextval('libapp.city_cityid_seq'::regclass);
ALTER TABLE ONLY libapp.hire ALTER COLUMN hireid SET DEFAULT
nextval('libapp.hire_hireid_seq'::regclass);
ALTER TABLE ONLY libapp.person ALTER COLUMN personid SET DEFAULT
nextval('libapp.person_personid_seq'::regclass);
ALTER TABLE ONLY libapp.printery ALTER COLUMN printeryid SET DEFAULT
nextval('libapp.printery_printeryid_seq'::regclass);
ALTER TABLE ONLY libapp.town ALTER COLUMN townid SET DEFAULT
nextval('libapp.town_townid_seq'::regclass);
INSERT INTO libapp.adress (adressid, townid, adres) VALUES (1, 1, 'fevzi çakmak mh., 756. sk
no:4, 34250 ');
INSERT INTO libapp.adress (adressid, townid, adres) VALUES (2, 2, 'kemalpasa mah. 191.sk no:9
d:4');
INSERT INTO libapp.adress (adressid, townid, adres) VALUES (3, 5, 'kemalpasa mah. no:252');
INSERT INTO libapp.adress (adressid, townid, adres) VALUES (4, 6, 'selam sok. no:3');
```

INSERT INTO libapp.adress (adressid, townid, adres) VALUES (5, 5, 'deli sokak. no: 48');

INSERT INTO libapp.adress (adressid, townid, adres) VALUES (6, 3, '1');

INSERT INTO libapp.adress (adressid, townid, adres) VALUES (10, 6, 'Bilinmiyor.');

INSERT INTO libapp.adress (adressid, townid, adres) VALUES (11, 9, 'bilinmiyor');

INSERT INTO libapp.adress (adressid, townid, adres) VALUES (12, 8, 'Bilinmiyor');

INSERT INTO libapp.adress (adressid, townid, adres) VALUES (13, 9, 'Bilinmiyor');

INSERT INTO libapp.adress (adressid, townid, adres) VALUES (14, 6, 'Bilinmiyor');

INSERT INTO libapp.book (bookid, bookname, printerydate, writerid, printeryid, booktypeid, hireprice, langid) VALUES (29, 'Kürk Mantolu Madonna', '2020-12-04', 50, 9, 1, 3, 1);

INSERT INTO libapp.book (bookid, bookname, printerydate, writerid, printeryid, booktypeid, hireprice, langid) VALUES (30, 'Sis ve Gece', '2020-12-04', 51, 9, 1, 5, 1);

INSERT INTO libapp.book (bookid, bookname, printerydate, writerid, printeryid, booktypeid, hireprice, langid) VALUES (31, 'Kral Şakir Okulda İlk Gün', '2020-12-05', 53, 10, 3, 1, 1);

INSERT INTO libapp.book (bookid, bookname, printerydate, writerid, printeryid, booktypeid, hireprice, langid) VALUES (32, 'A Clockwork Orange', '2020-12-05', 55, 11, 1, 8, 2);

INSERT INTO libapp.bookcategory (bookcategoryid, categoryid, bookid) VALUES (32, 2, 29);

INSERT INTO libapp.bookcategory (bookcategoryid, categoryid, bookid) VALUES (33, 1, 30);

INSERT INTO libapp.bookcategory (bookcategoryid, categoryid, bookid) VALUES (34, 3, 31);

INSERT INTO libapp.bookcategory (bookcategoryid, categoryid, bookid) VALUES (35, 5, 31);

INSERT INTO libapp.bookcategory (bookcategoryid, categoryid, bookid) VALUES (36, 2, 32);

INSERT INTO libapp.booklang (langid, language) VALUES (1, 'Türkçe');

INSERT INTO libapp.booklang (langid, language) VALUES (2, 'ingilizce');

INSERT INTO libapp.booklang (langid, language) VALUES (3, 'Almanca');

INSERT INTO libapp.booklang (langid, language) VALUES (4, 'İtalyanca');

INSERT INTO libapp.booktype (typeid, typename) VALUES (1, 'roman');

INSERT INTO libapp.booktype (typeid, typename) VALUES (2, 'hikaye');

INSERT INTO libapp.booktype (typeid, typename) VALUES (3, 'masal');

INSERT INTO libapp.booktype (typeid, typename) VALUES (4, 'şiir');

```
INSERT INTO libapp.booktype (typeid, typename) VALUES (5, 'biyografi');

INSERT INTO libapp.category (categoryid, categoryname) VALUES (1, 'polisiye');

INSERT INTO libapp.category (categoryid, categoryname) VALUES (2, 'dram');

INSERT INTO libapp.category (categoryid, categoryname) VALUES (3, 'komedi');

INSERT INTO libapp.category (categoryid, categoryname) VALUES (4, 'korku');

INSERT INTO libapp.category (categoryid, categoryname) VALUES (5, 'eğlence');

INSERT INTO libapp.city (cityid, cityname) VALUES (2, 'sakarya');

INSERT INTO libapp.city (cityid, cityname) VALUES (3, 'ankara');

INSERT INTO libapp.city (cityid, cityname) VALUES (4, 'adıyaman');

INSERT INTO libapp.city (cityid, cityname) VALUES (5, 'adana');

INSERT INTO libapp.city (cityid, cityname) VALUES (6, 'erzurum');

INSERT INTO libapp.city (cityid, cityname) VALUES (7, 'çankırı');

INSERT INTO libapp.city (cityid, cityname) VALUES (8, 'sinop');

INSERT INTO libapp.city (cityid, cityname) VALUES (9, 'mersin');

INSERT INTO libapp.city (cityid, cityname) VALUES (10, 'zonguldak');

INSERT INTO libapp.city (cityid, cityname) VALUES (1, 'istanbul');

INSERT INTO libapp.hire (hireid, responsetime, price, isback, bookid, personid) VALUES (16,
'2020-12-11', 5, 0, 30, 52);

INSERT INTO libapp.hire (hireid, responsetime, price, isback, bookid, personid) VALUES (17,
'2020-12-12', 1, 0, 31, 54);

INSERT INTO libapp.hire (hireid, responsetime, price, isback, bookid, personid) VALUES (18,
'2020-12-12', 3, 1, 29, 54);

INSERT INTO libapp.hire (hireid, responsetime, price, isback, bookid, personid) VALUES (19,
'2020-12-12', 8, 0, 32, 54);

INSERT INTO libapp.hire (hireid, responsetime, price, isback, bookid, personid) VALUES (20,
'2020-12-24', 3, 0, 29, 1);

INSERT INTO libapp.hire (hireid, responsetime, price, isback, bookid, personid) VALUES (15,
'2020-12-11', 3, 1, 29, 1);
```

```
INSERT INTO libapp.member (personid, username, password, booknumber) VALUES (2, 'ilaydatezgider', '123', 0);

INSERT INTO libapp.member (personid, username, password, booknumber) VALUES (52, 'hilalt', 'hilalt', 0);

INSERT INTO libapp.member (personid, username, password, booknumber) VALUES (54, 'test', 'test', 1);

INSERT INTO libapp.member (personid, username, password, booknumber) VALUES (1, 'admin', 'admin', 2);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (2, 'ilayda', 'tezgider', 1, 0);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (1, 'admin', 'admin', 1, 0);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (50, 'Sabahattin', 'Ali', 0, 1);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (51, 'Ahmet', 'Ümit', 0, 1);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (52, 'hilal', 'tezgider', 1, 0);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (53, 'Varol', 'Yaşaroğlu', 0, 1);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (54, 'test', 'test', 1, 0);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (55, 'Anthony', 'Burgess', 0, 1);

INSERT INTO libapp.person (personid, name, surname, ismember, iswriter) VALUES (56, 'test', 'test', 0, 1);

INSERT INTO libapp.printery (printeryid, printeryname, adressid) VALUES (9, 'YKY', 12);

INSERT INTO libapp.printery (printeryid, printeryname, adressid) VALUES (10, 'Eksik Parça', 13);

INSERT INTO libapp.printery (printeryid, printeryname, adressid) VALUES (11, 'İş Bankası', 14);

INSERT INTO libapp.role (roleid, rolename) VALUES (1, 'admin');

INSERT INTO libapp.role (roleid, rolename) VALUES (2, 'member');
```

```
INSERT INTO libapp.town (townid, townname, cityid) VALUES (1, 'gaziosmanpaşa', 1);
INSERT INTO libapp.town (townid, townname, cityid) VALUES (2, 'serdivan', 2);
INSERT INTO libapp.town (townid, townname, cityid) VALUES (3, 'başakşehir', 1);
INSERT INTO libapp.town (townid, townname, cityid) VALUES (4, 'orta', 7);
INSERT INTO libapp.town (townid, townname, cityid) VALUES (5, 'erfelek', 8);
INSERT INTO libapp.town (townid, townname, cityid) VALUES (6, 'yenimahalle', 3);
INSERT INTO libapp.town (townid, townname, cityid) VALUES (10, 'salihli', 4);
INSERT INTO libapp.town (townid, townname, cityid) VALUES (9, 'bahcelievler', 1);
INSERT INTO libapp.town (townid, townname, cityid) VALUES (8, 'bahcelievler', 3);
INSERT INTO libapp.userrole (personid, roleid) VALUES (1, 1);
INSERT INTO libapp.userrole (personid, roleid) VALUES (2, 2);
INSERT INTO libapp.userrole (personid, roleid) VALUES (52, 2);
INSERT INTO libapp.userrole (personid, roleid) VALUES (54, 2);
INSERT INTO libapp.writer (personid, booknumber) VALUES (50, 1);
INSERT INTO libapp.writer (personid, booknumber) VALUES (51, 1);
INSERT INTO libapp.writer (personid, booknumber) VALUES (53, 1);
INSERT INTO libapp.writer (personid, booknumber) VALUES (55, 1);
INSERT INTO libapp.writer (personid, booknumber) VALUES (56, 1);
SELECT pg_catalog.setval('libapp.adress_adressid_seq', 14, true);
SELECT pg_catalog.setval('libapp.book_bookid_seq', 33, true);
SELECT pg_catalog.setval('libapp.bookcategory_bookcategoryid_seq', 39, true);
SELECT pg_catalog.setval('libapp.booktype_typeid_seq', 6, false);
SELECT pg_catalog.setval('libapp.category_categoryid_seq', 6, false);
SELECT pg_catalog.setval('libapp.city_cityid_seq', 11, false);
SELECT pg_catalog.setval('libapp.hire_hireid_seq', 20, true);
SELECT pg_catalog.setval('libapp.person_personid_seq', 56, true);
SELECT pg_catalog.setval('libapp.printery_printeryid_seq', 11, true);
```

```
SELECT pg_catalog.setval('libapp.town_townid_seq', 11, false);
ALTER TABLE ONLY libapp.booklang
    ADD CONSTRAINT booklang_pkey PRIMARY KEY (langid);
ALTER TABLE ONLY libapp.adress
    ADD CONSTRAINT pk_adress PRIMARY KEY (adressid);
ALTER TABLE ONLY libapp.book
    ADD CONSTRAINT pk_book PRIMARY KEY (bookid);
ALTER TABLE ONLY libapp.bookcategory
    ADD CONSTRAINT pk_bookcategory PRIMARY KEY (bookcategoryid);
ALTER TABLE ONLY libapp.booktype
    ADD CONSTRAINT pk_booktype PRIMARY KEY (typeid);
ALTER TABLE ONLY libapp.category
    ADD CONSTRAINT pk_category PRIMARY KEY (categoryid);
ALTER TABLE ONLY libapp.city
    ADD CONSTRAINT pk_city PRIMARY KEY (cityid);
ALTER TABLE ONLY libapp.hire
    ADD CONSTRAINT pk_hire PRIMARY KEY (hireid);
ALTER TABLE ONLY libapp.member
    ADD CONSTRAINT pk_member PRIMARY KEY (personid);
ALTER TABLE ONLY libapp.person
    ADD CONSTRAINT pk_person PRIMARY KEY (personid);
ALTER TABLE ONLY libapp.printery
    ADD CONSTRAINT pk_printery PRIMARY KEY (printeryid);
ALTER TABLE ONLY libapp.town
    ADD CONSTRAINT pk_town PRIMARY KEY (townid);
ALTER TABLE ONLY libapp.writer
    ADD CONSTRAINT pk_writer PRIMARY KEY (personid);
```

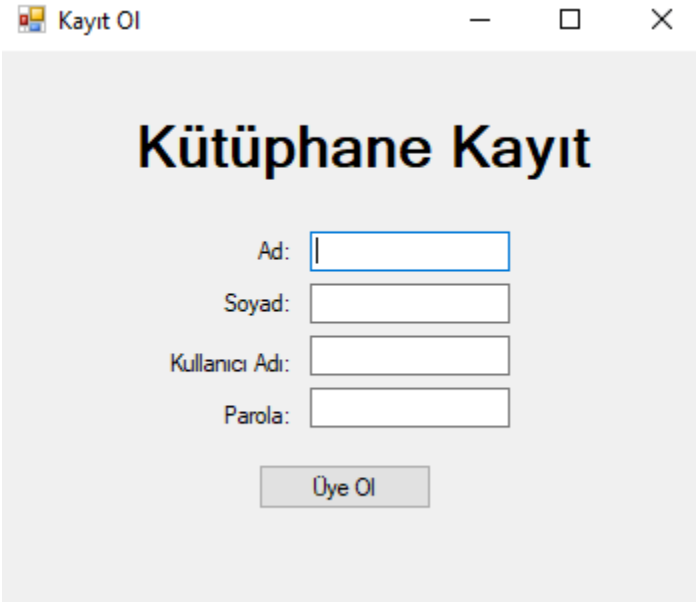
```
ALTER TABLE ONLY libapp.role
    ADD CONSTRAINT role_pkey PRIMARY KEY (roleid);
ALTER TABLE ONLY libapp.adress
    ADD CONSTRAINT uq_adress UNIQUE (adressid);
ALTER TABLE ONLY libapp.book
    ADD CONSTRAINT uq_book UNIQUE (bookid);
ALTER TABLE ONLY libapp.bookcategory
    ADD CONSTRAINT uq_bookcategory UNIQUE (bookcategoryid);
ALTER TABLE ONLY libapp.booklang
    ADD CONSTRAINT uq_booklang UNIQUE (langid);
ALTER TABLE ONLY libapp.booktype
    ADD CONSTRAINT uq_booktype UNIQUE (typeid);
ALTER TABLE ONLY libapp.category
    ADD CONSTRAINT uq_category UNIQUE (categoryid);
ALTER TABLE ONLY libapp.city
    ADD CONSTRAINT uq_city UNIQUE (cityid);
ALTER TABLE ONLY libapp.hire
    ADD CONSTRAINT uq_hire UNIQUE (hireid);
ALTER TABLE ONLY libapp.member
    ADD CONSTRAINT uq_member UNIQUE (personid);
ALTER TABLE ONLY libapp.printery
    ADD CONSTRAINT uq_printery UNIQUE (printeryid);
ALTER TABLE ONLY libapp.role
    ADD CONSTRAINT uq_role UNIQUE (roleid);
ALTER TABLE ONLY libapp.town
    ADD CONSTRAINT uq_town UNIQUE (townid);
ALTER TABLE ONLY libapp.userrole
```

```
    ADD CONSTRAINT uq_userrole UNIQUE (personid, roleid);

ALTER TABLE ONLY libapp.writer

    ADD CONSTRAINT uq_writer UNIQUE (personid);

ALTER TABLE ONLY libapp.userrole

    ADD CONSTRAINT userrole_pkey PRIMARY KEY (personid, roleid);

CREATE TRIGGER t_member_book_inc AFTER UPDATE ON libapp.hire FOR EACH ROW EXECUTE
FUNCTION libapp.t_member_book_inc_func();

CREATE TRIGGER t_person_delete AFTER DELETE ON libapp.person FOR EACH ROW EXECUTE
FUNCTION libapp.t_person_delete_func();

CREATE TRIGGER t_person_type AFTER INSERT ON libapp.person FOR EACH ROW EXECUTE
FUNCTION libapp.t_person_type_func();

CREATE TRIGGER t_writer_book_inc AFTER INSERT ON libapp.book FOR EACH ROW EXECUTE
FUNCTION libapp.t_writer_book_inc_func();

ALTER TABLE ONLY libapp.adress

    ADD CONSTRAINT fk_adres_town FOREIGN KEY (townid) REFERENCES libapp.town(townid)
NOT VALID;

ALTER TABLE ONLY libapp.book

    ADD CONSTRAINT fk_book_booktype FOREIGN KEY (booktypeid) REFERENCES
libapp.booktype(typeid) NOT VALID;

ALTER TABLE ONLY libapp.book

    ADD CONSTRAINT fk_book_lang FOREIGN KEY (langid) REFERENCES libapp.booklang(langid)
NOT VALID;

ALTER TABLE ONLY libapp.book

    ADD CONSTRAINT fk_book_printery FOREIGN KEY (printeryid) REFERENCES
libapp.printery(printeryid) NOT VALID;

ALTER TABLE ONLY libapp.book

    ADD CONSTRAINT fk_book_writer FOREIGN KEY (writerid) REFERENCES
libapp.writer(personid) NOT VALID;

ALTER TABLE ONLY libapp.bookcategory
```

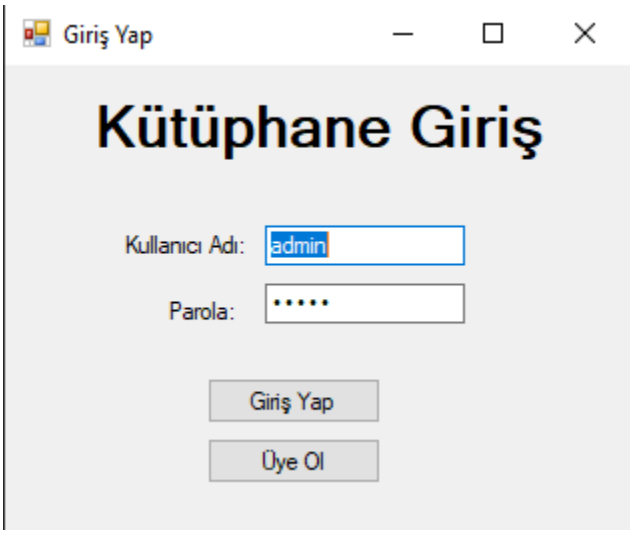
```
    ADD CONSTRAINT fk_bookcategory_book FOREIGN KEY (bookid) REFERENCES
libapp.book(bookid) NOT VALID;

ALTER TABLE ONLY libapp.bookcategory

    ADD CONSTRAINT fk_bookcategory_category FOREIGN KEY (categoryid) REFERENCES
libapp.category(categoryid) NOT VALID;

ALTER TABLE ONLY libapp.hire

    ADD CONSTRAINT fk_hire_book FOREIGN KEY (bookid) REFERENCES libapp.book(bookid) NOT
VALID;

ALTER TABLE ONLY libapp.hire

    ADD CONSTRAINT fk_hire_person FOREIGN KEY (personid) REFERENCES
libapp.person(personid) NOT VALID;

ALTER TABLE ONLY libapp.member

    ADD CONSTRAINT fk_member_person FOREIGN KEY (personid) REFERENCES
libapp.person(personid) NOT VALID;

ALTER TABLE ONLY libapp.printery

    ADD CONSTRAINT fk_printery_adress FOREIGN KEY (adressid) REFERENCES
libapp.adress(adressid) NOT VALID;

ALTER TABLE ONLY libapp.town

    ADD CONSTRAINT fk_town_city FOREIGN KEY (cityid) REFERENCES libapp.city(cityid) NOT
VALID;

ALTER TABLE ONLY libapp.userrole

    ADD CONSTRAINT fk_userrole_person FOREIGN KEY (personid) REFERENCES
libapp.person(personid) NOT VALID;

ALTER TABLE ONLY libapp.userrole

    ADD CONSTRAINT fk_userrole_role FOREIGN KEY (roleid) REFERENCES libapp.role(roleid) NOT
VALID;

ALTER TABLE ONLY libapp.writer

    ADD CONSTRAINT fk_writer_person FOREIGN KEY (personid) REFERENCES
libapp.person(personid) NOT VALID;

GRANT ALL ON SCHEMA libapp TO PUBLIC;
```

EKRAN GÖRÜNTÜLERİ

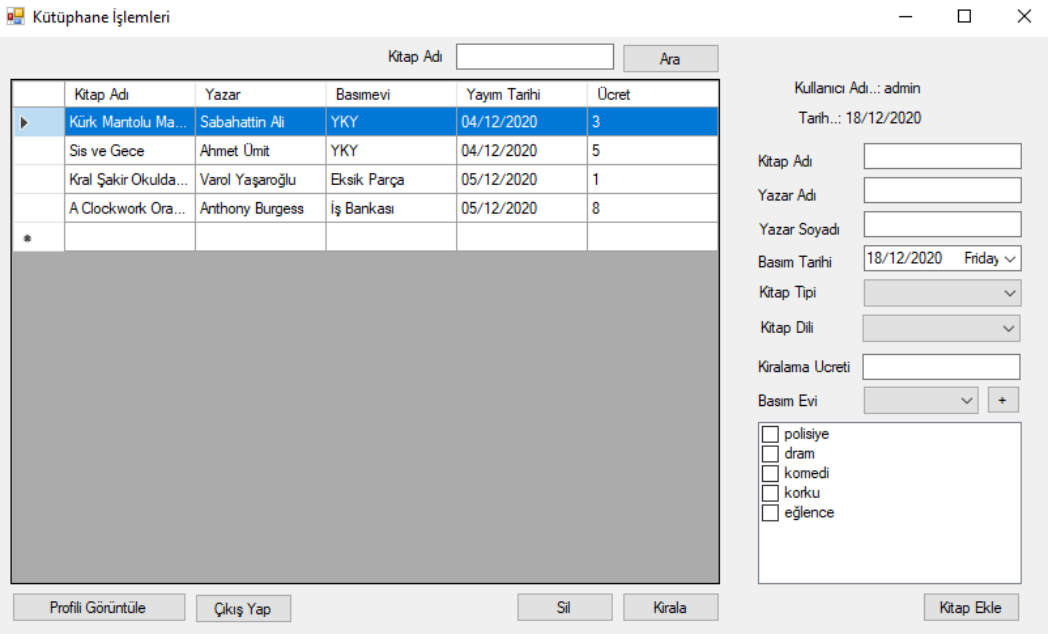
1. Kullanıcı Kayıt Olma Ekranı : Kullanıcı bilgileri kayıt edilir.



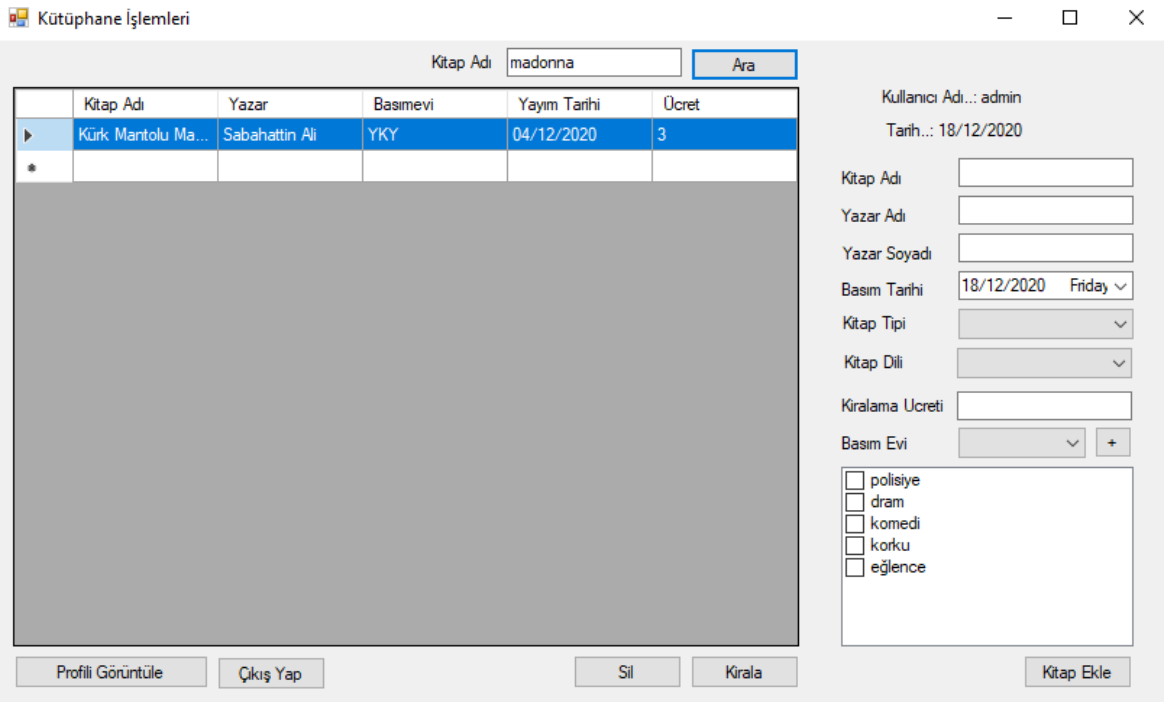
2. Kullanıcı Giriş Ekranı : Girilen kullanıcı adı ve parolaya göre ‘select’ işlemi yapılarak kullanıcı girişi sağlanır.



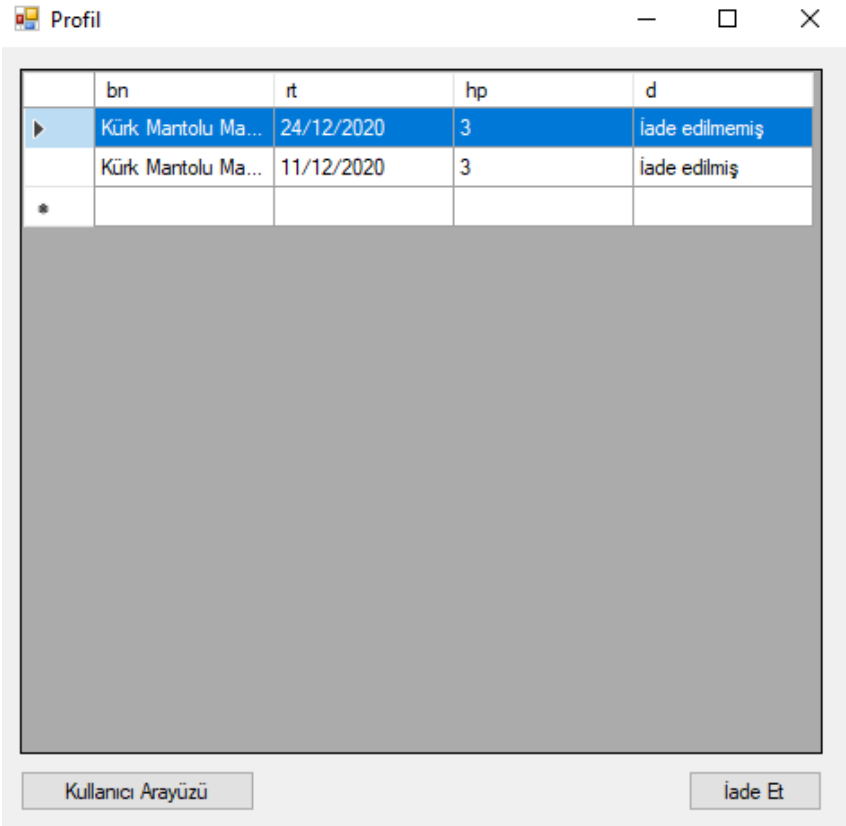
3. Kullanıcı Arayüz Ekranı : Kitap ekle butonu ile girilen bilgiler veritabanına kayıt edilir. Kitap Tipi, Kitap Dili, Basım Evi ve Kitap Türü bilgileri ‘select’ ifadesi ile çekilir. Sil butonu ile kitaplar silinir. Kirala butonu ile kullanıcıyla beraber kitap bilgisi ‘insert’ edilir.



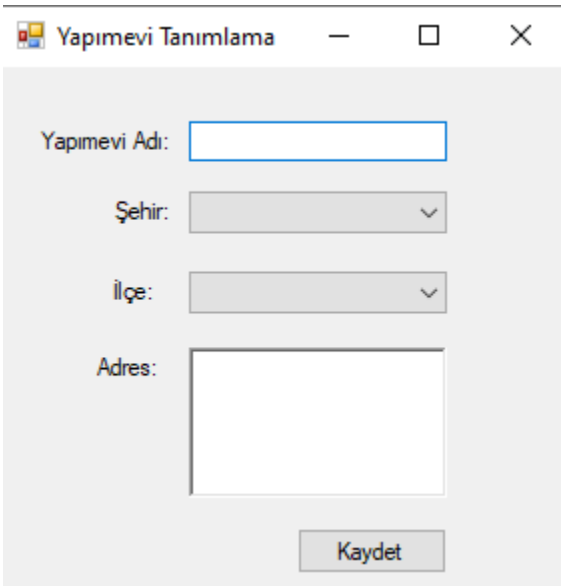
4. Arama İşleminin Yapılması : Girilen Kitap adına göre Ara butonuna basarak arama işlemi veritabanından ‘like’ ifadesi ile yapılır.



5. Kullanıcı Kitap Kiralama Geçmişi Ekranı : İade et butonu ile kullanıcı kitap bilgileri güncelleme yapılır.



6. Basım Evi Ekleme Ekranı : Girilen bilgilere göre sisteme basımevi eklenir.



UYGULAMANIN KAYNAK KODLARI

Kaynak kodları github'da yer almaktadır.

GitHub Adresi: https://github.com/ilaydaTezgider/LibraryApp

TANITIM VİDEOSU

Video youtube'da bulunmaktadır.

YouTube Adresi: https://youtu.be/KH9iLHjUhw4