T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİTABANI YÖNETİM SİSTEMLERİ PROJESİ

G191210385 - İbrahim Yiğit Tın

ibrahim.tin@ogr.sakarya.edu.tr

i.yigittin@hotmail.com

SAKARYA

Ağustos, 2021

Veritabanı Yönetim Sistemleri

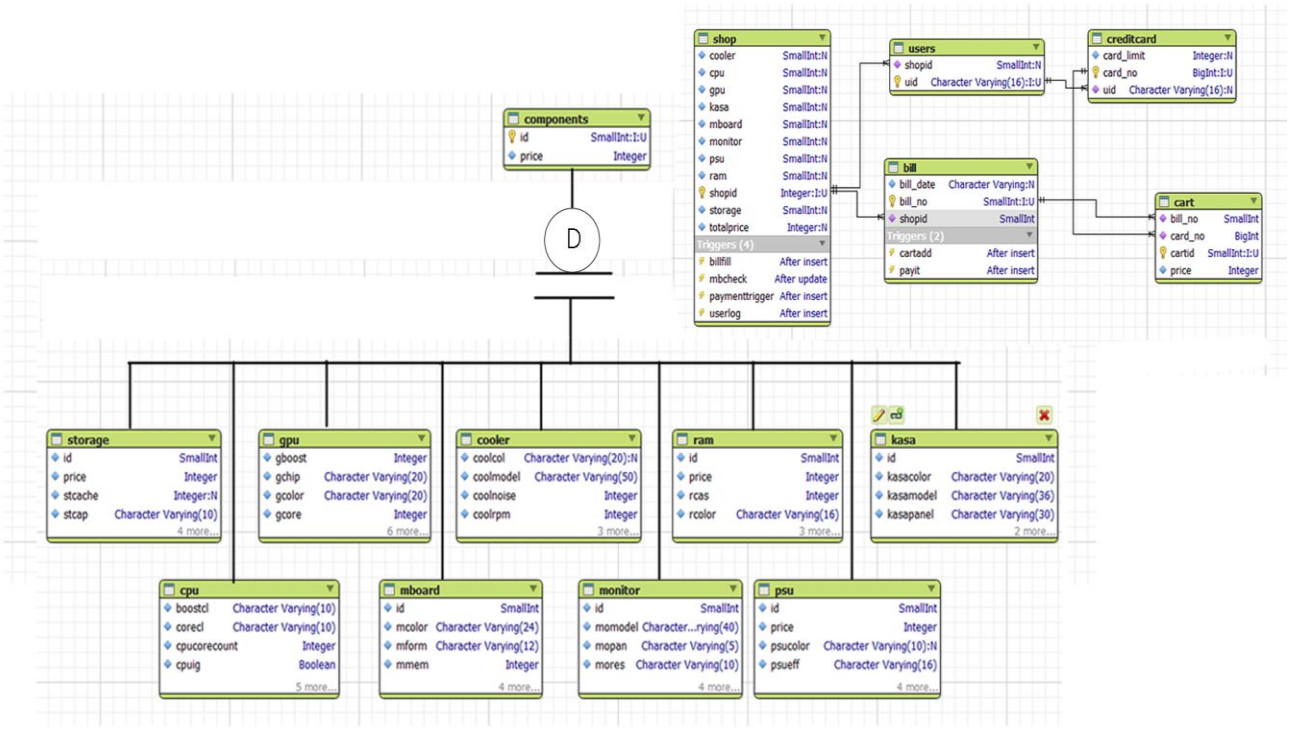
1. Uygulamanın Özeti

Bu web uygulamasında bilgisayar bileşenlerini seçerek toplanacak bilgisayarın özelliklerini ve toplam fiyatını görebildiğimiz bir PERN (PostgreSQL – Express – React – NodeJs) web uygulaması.

2. İş Kuralları

- Kullanıcı her parçadan sadece bir tane seçebilir.
- Her parça farklı sepetlerde bulunabilir.
- Bir parça değiştirildiğinde uyumsuzluk kontrol edilir.
- Kullanıcı tüm parçaları seçmeden alışveriş yapabilir.
- Kullanıcının seçtiği ürünler uyumlu olmak zorundadır.
- Kullanıcı istediği parçayı silme ve değiştirme işlemi yapabilmelidir.
- Kullanıcı satın alma işlemi yapabilir.
- Satın alma işlemi sonrası fatura verilmelidir.
- Satın alma işlemi için kart limiti kontrol edilmelidir.

3. Varlık Bağıntı Modeli



4. SQL İfadeleri

```
4
5  CREATE TABLE public.shop
6  (
7      shopid integer NOT NULL DEFAULT nextval('shop_shopid_seq'::regclass),
8      cpu smallint,
9      gpu smallint,
10     mboard smallint,
11     cooler smallint,
12     psu smallint,
13     ram smallint,
14     storage smallint,
15     monitor smallint,
16     kasa smallint,
17     totalprice integer,
18     CONSTRAINT shop_pkey PRIMARY KEY (shopid)
19 )
20
21 TABLESPACE pg_default;
22
23 ALTER TABLE public.shop
24     OWNER to postgres;
25
26 -- Trigger: billfill
27
28 -- DROP TRIGGER billfill ON public.shop;
29
30 CREATE TRIGGER billfill
31     AFTER INSERT
32     ON public.shop
33     FOR EACH STATEMENT
34     EXECUTE PROCEDURE public.billing();
35
```

```
-- Trigger: mbcheck

-- DROP TRIGGER mbcheck ON public.shop;

CREATE TRIGGER mbcheck
    AFTER UPDATE
    ON public.shop
    FOR EACH ROW
    EXECUTE PROCEDURE public.mboardcheck();

-- Trigger: paymenttrigger

-- DROP TRIGGER paymenttrigger ON public.shop;

CREATE TRIGGER paymenttrigger
    AFTER INSERT
    ON public.shop
    FOR EACH ROW
    EXECUTE PROCEDURE public.payment();

-- Trigger: userlog

-- DROP TRIGGER userlog ON public.shop;

CREATE TRIGGER userlog
    AFTER INSERT
    ON public.shop
    FOR EACH ROW
    EXECUTE PROCEDURE public.log_shop();
```

6

```
CREATE TABLE public.bill
(
    bill_no smallint NOT NULL,
    bill_date character varying COLLATE pg_catalog."default",
    shopid smallint NOT NULL,
    CONSTRAINT bill_pkey PRIMARY KEY (bill_no),
    CONSTRAINT fk_shop FOREIGN KEY (shopid)
        REFERENCES public.shop (shopid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE public.bill
    OWNER to postgres;

-- Trigger: cartadd

-- DROP TRIGGER cartadd ON public.bill;

CREATE TRIGGER cartadd
    AFTER INSERT
    ON public.bill
    FOR EACH ROW
    EXECUTE PROCEDURE public.cartit();

-- Trigger: payit

-- DROP TRIGGER payit ON public.bill;

CREATE TRIGGER payit
    AFTER INSERT
    ON public.bill
    FOR EACH STATEMENT
    EXECUTE PROCEDURE public.payment();
```

```
4
5  CREATE TABLE public.cart
6  (
7      cartid smallint NOT NULL,
8      card_no bigint NOT NULL,
9      bill_no smallint NOT NULL,
10     price integer NOT NULL,
11     CONSTRAINT cart_pkey PRIMARY KEY (cartid),
12     CONSTRAINT fk_bill FOREIGN KEY (bill_no)
13         REFERENCES public.bill (bill_no) MATCH SIMPLE
14         ON UPDATE NO ACTION
15         ON DELETE NO ACTION,
16     CONSTRAINT fk_card FOREIGN KEY (card_no)
17         REFERENCES public.creditcard (card_no) MATCH SIMPLE
18         ON UPDATE NO ACTION
19         ON DELETE NO ACTION
20 )
21
22 TABLESPACE pg_default;
23
24 ALTER TABLE public.cart
25     OWNER to postgres;
```

```
4
5  CREATE TABLE public.components
6  (
7      id smallint NOT NULL DEFAULT nextval('"Components_id_seq"'::regclass),
8      price integer NOT NULL,
9      CONSTRAINT "Components_pkey" PRIMARY KEY (id)
10 )
11
12 TABLESPACE pg_default;
13
14 ALTER TABLE public.components
15     OWNER to postgres;
```

```
4
5  CREATE TABLE public.cooler
6  (
7      -- Inherited from table public.components: id smallint NOT NULL DEFAULT nextval('"Components_id_seq"'::regclass),
8      -- Inherited from table public.components: price integer NOT NULL,
9      coolmodel character varying(50) COLLATE pg_catalog."default" NOT NULL,
10     coolrpm integer NOT NULL,
11     coolnoise integer NOT NULL,
12     coolcol character varying(20) COLLATE pg_catalog."default",
13     coolsize integer
14 )
15     INHERITS (public.components)
16 TABLESPACE pg_default;
17
18 ALTER TABLE public.cooler
19     OWNER to postgres;
```

```
CREATE TABLE public.cpu
(
    -- Inherited from table public.components: id smallint NOT NULL DEFAULT nextval('"Components_id_seq"'::regclass),
    -- Inherited from table public.components: price integer NOT NULL,
    cpucorecount integer NOT NULL,
    corecl character varying(10) COLLATE pg_catalog."default" NOT NULL,
    boostcl character varying(10) COLLATE pg_catalog."default" NOT NULL,
    cputdp integer NOT NULL,
    cpusocket character varying(12) COLLATE pg_catalog."default" NOT NULL,
    cpuig boolean NOT NULL,
    cpumodel character varying(36) COLLATE pg_catalog."default" NOT NULL
)
    INHERITS (public.components)
TABLESPACE pg_default;

ALTER TABLE public.cpu
    OWNER to postgres;
```

```
CREATE TABLE public.bill
(
    bill_no smallint NOT NULL,
    bill_date character varying COLLATE pg_catalog."default",
    shopid smallint NOT NULL,
    CONSTRAINT bill_pkey PRIMARY KEY (bill_no),
    CONSTRAINT fk_shop FOREIGN KEY (shopid)
        REFERENCES public.shop (shopid) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE public.bill
    OWNER to postgres;

-- Trigger: cartadd

-- DROP TRIGGER cartadd ON public.bill;

CREATE TRIGGER cartadd
    AFTER INSERT
    ON public.bill
    FOR EACH ROW
    EXECUTE PROCEDURE public.cartit();

-- Trigger: payit

-- DROP TRIGGER payit ON public.bill;

CREATE TRIGGER payit
    AFTER INSERT
    ON public.bill
    FOR EACH STATEMENT
    EXECUTE PROCEDURE public.payment();
```

5. Triggerlar

```
CREATE FUNCTION public.billing()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
    buyed_shop smallint;
    billinc smallint;
    BEGIN
    billinc:=(select bill_no from bill order by bill_no desc limit 1)+1;
    buyed_shop:=(select shopid from shop order by shopid desc limit 1);
    INSERT INTO bill(bill_no,bill_date,shopid) VALUES (billinc,CURRENT_DATE,buyed_shop);
    RETURN NEW;
    END
$BODY$;

ALTER FUNCTION public.billing()
    OWNER TO postgres;
```

10

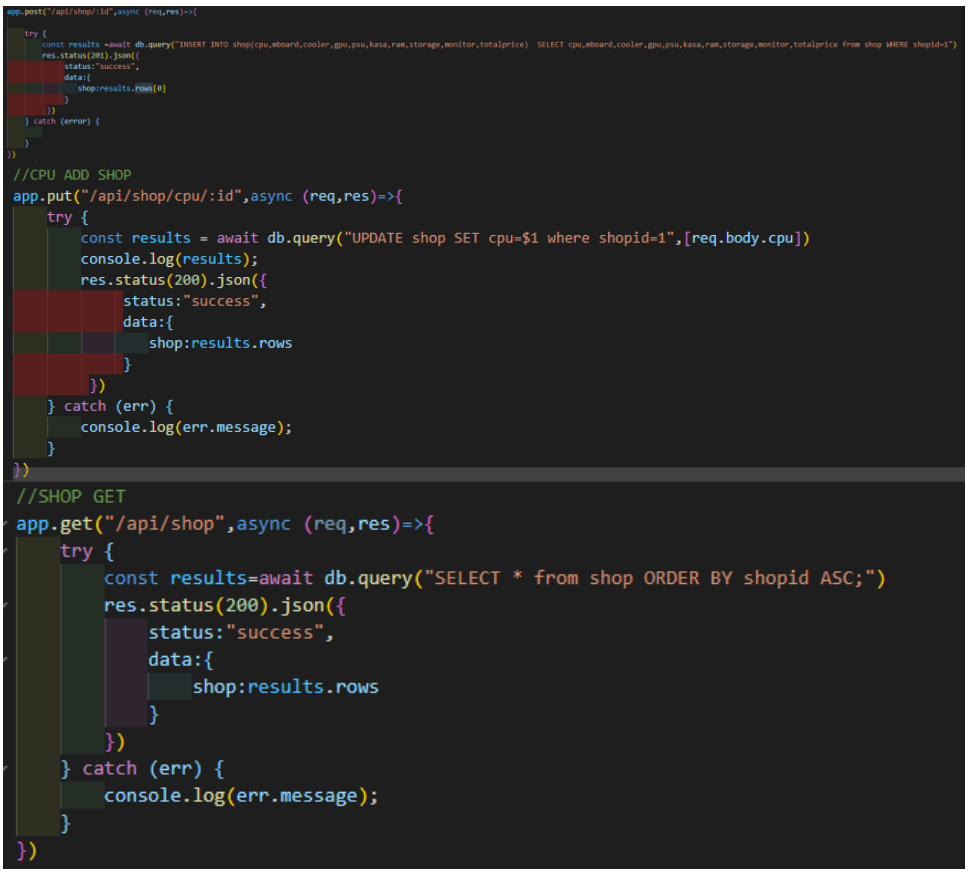
```
CREATE FUNCTION public.cartit()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
credit_ bigint;
billno smallint;
para integer;
cartinc smallint;
BEGIN
credit_:=(select card_no from creditcard);
billno:=(select bill_no from bill order by bill_no desc limit 1);
para:=(select totalprice from shop order by shopid desc limit 1);
cartinc:=(select cartid from cart order by cartid desc limit 1)+1;
INSERT INTO cart(cartid,card_no,bill_no,price) VALUES (cartinc,credit_,billno,para);
RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.cartit()
    OWNER TO postgres;

```

```
CREATE FUNCTION public.log_shop()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
declare
 shop_id integer;
 begin
 shop_id:=(select shopid from shop order by shopid desc limit 1) ;
 update users set shopid=shop_id;
return new;
end;
$BODY$;

ALTER FUNCTION public.log_shop()
    OWNER TO postgres;

```

```
CREATE FUNCTION public.mboardcheck()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
cpusoc character varying;
mboardsoc character varying;
shopinfo integer;
BEGIN
shopinfo:=(select cpu from shop where shopid=1);
cpusoc:=(select cpusocket from cpu where id=shopinfo);
shopinfo:=(select mboard from shop where shopid=1);
mboardsoc:=(select msocket from mboard where id=shopinfo);

IF mboardsoc <> cpusoc THEN
UPDATE shop set mboard=NULL where shopid=1;
END IF;
RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.mboardcheck()
    OWNER TO postgres;
```

```
CREATE FUNCTION public.payment()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
payment integer;
BEGIN
payment:=(select totalprice from shop order by shopid desc limit 1);
IF payment>0 THEN
UPDATE creditcard set card_limit=(card_limit-(payment/2));
END IF;
RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.payment()
    OWNER TO postgres;
```

6. Uygulama Ekran Görüntüleri



```
app.post("/api/shop/:id",async (req,res)=>{

    try {
        const results =await db.query("INSERT INTO shop(cpu,mboard,cooler,gpu,psu,kasa,ram,storage,monitor,totalprice)  SELECT cpu,mboard,cooler,gpu,psu,kasa,ram,storage,monitor,totalprice from shop WHERE shopid=1")
        res.status(201).json({
            status:"success",
            data:{
                shop:results.rows[0]
            }
        })
    } catch (error) {

    }
})
//CPU ADD SHOP
app.put("/api/shop/cpu/:id",async (req,res)=>{
    try {
        const results = await db.query("UPDATE shop SET cpu=$1 where shopid=1",[req.body.cpu])
        console.log(results);
        res.status(200).json({
            status:"success",
            data:{
                shop:results.rows
            }
        })
    } catch (err) {
        console.log(err.message);
    }
})
//SHOP GET
app.get("/api/shop",async (req,res)=>{
    try {
        const results=await db.query("SELECT * from shop ORDER BY shopid ASC;")
        res.status(200).json({
            status:"success",
            data:{
                shop:results.rows
            }
        })
    } catch (err) {
        console.log(err.message);
    }
})
```

7. Linkler

- GitHub: https://github.com/yigittin/VTYS-Proje
- Youtube: https://youtu.be/mz5ZLtIfjdM