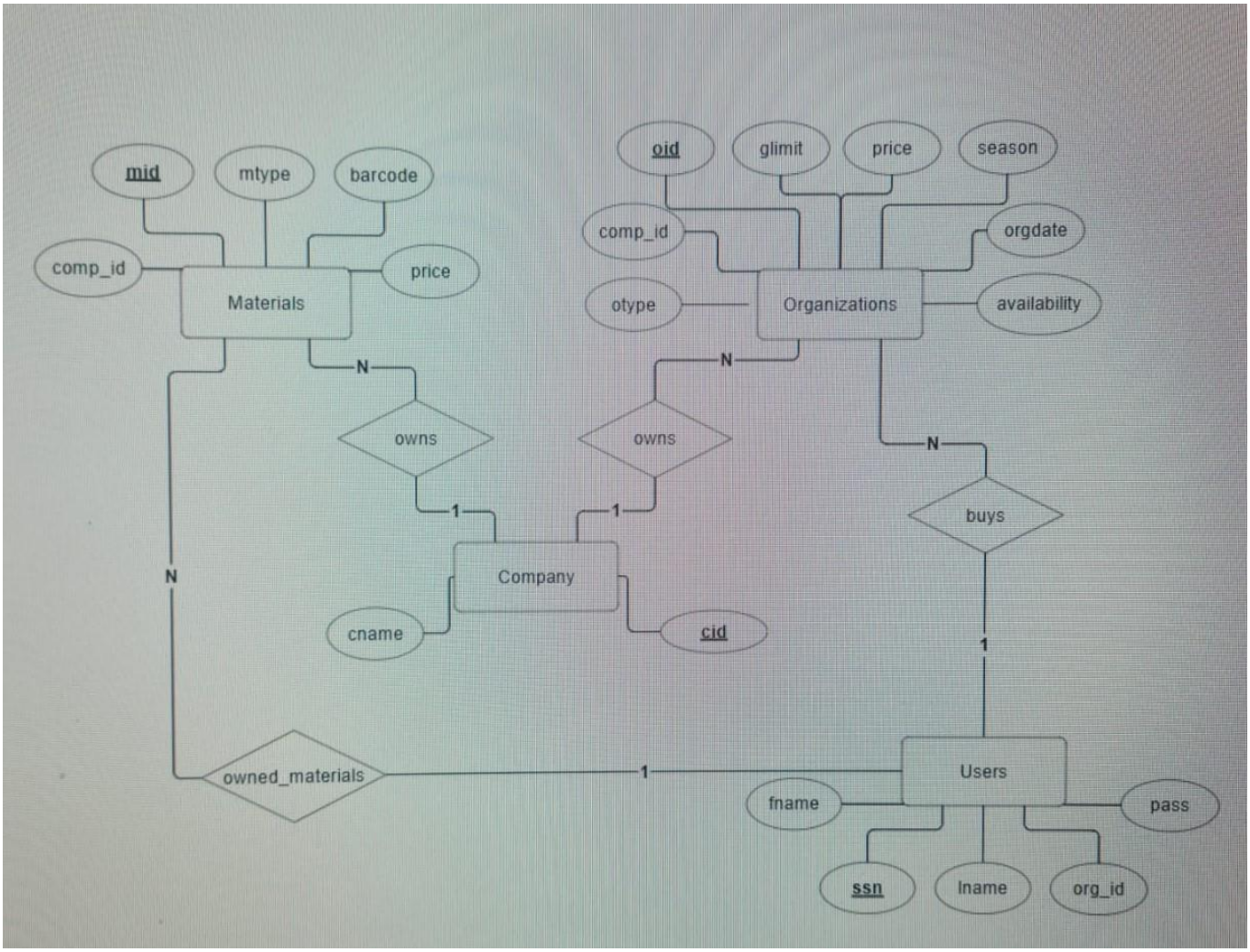
YILDIZ TEKNİK ÜNİVERSİTESİ

Veritabanı Yönetimi Projesi

Emre Furkan Bayraklı 20011064
Yusuf Güney 20011010
Ergün İzmirlioğlu 21011604
Selin Çırak 21011603



	oid [PK] integer	comp_id character varying (9)	otype character varying (20)	glimit integer	season character varying (15)	price double precision	orgdate integer	availability boolean
1	1	hasanmu	wedding	1000	summer	1000	2024	true
2	2	sesegel	birthday	500	winter	1000	2026	false
3	4	eew	graduation	200	fall	800	2024	false
4	5	ups	birthday	200	spring	1000	2026	true
5	6	trendyol	birthday	1000	winter	5000	2024	false
6	7	getir	graduation	1000	summer	1000	2025	false
7	8	erenkoca	birthday	250	spring	2500	2025	true
8	9	uygurlar	graduation	500	spring	500	2024	false
9	10	furkani	birthday	200	spring	900	2024	true
10	11	upg	wedding	600	summer	1500	2024	true

	mid [PK] character varying (9)	comp_id character varying (9)	mtype character varying (20)	barcode character varying (12)	price double precision
1	m1	hasanmu	mesale	12379	50.5
2	m2	hasanmu	mesale	12378	120.7
3	sapka1	ups	sapka	12370	60.5
4	sapka2	erenkoca	sapka	12370	70.5
5	sapka3	hasanmu	sapka	12376	20.5
6	sus1	upg	sus	12345	50
7	sus2	erenkoca	sus	12346	55
8	sus3	getir	sus	12348	70
9	sus4	erenkoca	sus	12349	20.5
10	sus5	ups	sus	12340	60.5

	cid [PK] character varying (9)	cname character varying (20)
1	eew	Enterprise E Win
2	erenkoca	Eren Kocabaş Anonim
3	furkani	Furkan Bayraklı
4	getir	Getir Yemek
5	hasanmu	Hasan Mustan
6	sesegel	Sese Doğru
7	trendyol	Popüler Yol
8	upg	upgp gpg
9	ups	Uf Program System
10	uygurlar	uygurlar

	user_ssn character varying (9)	material_id character varying (9)
1	ergunizm	sapka1
2	ergunizm	sapka2
3	ergunizm	sus1
4	abyilmaz	sus2
5	abyilmaz	sapka2
6	abyilmaz	m1
7	furkan	sus1
8	furkan	sus5
9	vtoroytom	sus4
10	yusufis	m2

3- En az bir tabloda silme kısıtı ve sayı kısıtı olmalıdır.

```
ALTER TABLE organizations
ADD CONSTRAINT chk_year
CHECK (orgdate > 2023);
```

4- Arayüzden en az birer tane insert, update ve delete işlemi gerçekleştirilebilmelidir.

```
public static String addOrgToUser(String user_ssn, int orgId) {
   String result;
   try {
      CallableStatement cstmt = connection.prepareCall("{? = call add_org_to_user(?, ?)}");
      cstmt.setInt(2, orgId);
      cstmt.setString(3, user_ssn.trim());
      cstmt.registerOutParameter(1, 12);
      cstmt.execute();

      result = cstmt.getString(1);

   } catch (SQLException e) {
      throw new RuntimeException(e);
   }

   return result;
}
```

```
public static String deleteOrganization(String orgId) {
   String query = "delete from organizations where oid = ?;";

   try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
```

```
        preparedStatement.setString(1, orgId);

        preparedStatement.executeUpdate();

        return orgId+" is deleted!";

    } catch (SQLException ex) {
        throw new RuntimeException(ex);
    }
}


public static String updateMaterial(String oldId, Material material) {
    String query = "update materials set mid = ?, comp_id = ?, mtype= ?, barcode = ?, price = ? WHERE
mid = ?";

    try (PreparedStatement preparedStatement = connection.prepareStatement(query)) {
        preparedStatement.setString(1, material.getMaterialId());
        preparedStatement.setString(2, material.getCompanyId());
        preparedStatement.setString(3, material.getMaterialType());
        preparedStatement.setString(4, material.getBarcode());
        preparedStatement.setDouble(5, material.getPrice());
        preparedStatement.setString(6, oldId);

        int result = preparedStatement.executeUpdate();

        if (result > 0) {
            return "Trigger executed : Material updated successfully!";
        }
    } catch (SQLException ex) {
        throw new RuntimeException(ex);
    }
    return null;
}
```

5- Arayüzden girilecek bir değere göre ekrana sonuçların listelendiği bir sorgu yazmalısınız.

```
public static <Organization> List<Organization> selectOrgOfCompany(String cid) {
    String query = "SELECT * FROM organizations where comp_id=?;";

    ResultSet rs = null;
    try {
        PreparedStatement ps = connection.prepareStatement(query);
        ps.setString(1,cid);
        rs = ps.executeQuery();
    } catch (SQLException e) {
```

```
        e.printStackTrace();
    }

    List<Organization> orgs = null;

    try {
        orgs = ResultSetToListUtil.convert(rs, "organizations");
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }

    return orgs;
}
```

6- Arayüzden çağrılan sorgulardan en az biri “view” olarak tanımlanmış olmalıdır.

```
create view materialNumberOfUsers as

select user_ssn, count(*)

from owned_materials

group by user_ssn

having count(*) > 0;
```

7- En az bir adet “sequence” oluşturmalı ve arayüzden yapılacak insert sırasında ilgili sütundaki değerlerin otomatik olarak atanmasını sağlamalısınız.

```
CREATE SEQUENCE public.organizations_oid

    START WITH 1

    INCREMENT BY 1

    NO MINVALUE

    NO MAXVALUE

    CACHE 1;
```

8- Arayüzden çağrılan sorgulardan en az birinde union veya intersect veya except kullanmış olmalısınız.

```
public static <Organization> List<Organization> selectSeasonTypeIntersect(String searchedSeason,
String type){
    ColoredOutput.print("Getting all " + type + " activities and Done on " + searchedSeason + ".",
```

```
        ColoredOutput.Color.MAGENTA_BOLD_BRIGHT);

   String query = "(SELECT DISTINCT * FROM organizations WHERE season=\'" +searchedSeason+ "\') "
+
        "INTERSECT (SELECT DISTINCT * FROM organizations WHERE otype=\'" + type + "\');";

   ResultSet rs = null;
   try {
      rs = connection.createStatement().executeQuery(query);
   } catch (SQLException e) {
      e.printStackTrace();
   }

   List<Organization> orgs = null;

   try {
      orgs = ResultSetToListUtil.convert(rs, "organizations");
   } catch (SQLException e) {
      e.printStackTrace();
   }

   return orgs;
}
```

9- Sorgularınızın en az biri aggregate fonksiyonlar içermeli, having ifadesi kullanılmalıdır.

```
select user_ssn, count(*)
from owned_materials
group by user_ssn
having count(*) > 0;
```

10- Arayüzden girilen değerleri parametre olarak alıp ekrana sonuç döndüren **3 farklı SQL fonksiyonu** tanımlamış olmalısınız. Bu fonksiyonların en az birinde “record” ve “cursor” tanımı-kullanımı olmalıdır.

```
create function material_priceof_user(ssn users.ssn%type)
returns double precision as $$
declare
        total_price double precision;
begin
        select sum(price) into total_price
        from owned_materials, materials
        where user_ssn=ssn and material_id=mid;
```

```
	return total_price;
end;
$$ language 'plpgsql';


create type mat as (mid varchar(9), comp_id varchar(9), typeof varchar(20), barcode varchar(12),
price double precision);
create function materialsof_user(ssn users.ssn%type)
returns mat[] as $$
declare
	matsofuser_cursor cursor for select mid, comp_id, mtype,barcode,price from
owned_materials,materials where user_ssn=ssn and material_id=mid;
	mats mat[];
	i integer;
begin
	i := 1;
	for tmpmat in matsofuser_cursor loop
			mats[i] = tmpmat;
			i := i + 1;
	end loop;
	return mats;
end;
$$ language 'plpgsql';


create or replace function add_org_to_user(org organizations.oid%type, user_ssn users.ssn%type)
returns TEXT as $$
declare
begin
	update organizations set availability = false where oid = org;
	update users set org_id = org where ssn=user_ssn;

	return org::TEXT || ' is added to ' || user_ssn;
end;

$$ language 'plpgsql';
```

11- 2 adet trigger tanımlamalı ve arayüzden girilecek değerlerle tetiklemelisiniz. Trigger'ın çalıştığına dair arayüze bilgilendirme mesajı döndürülmelidir.

```
CREATE OR REPLACE FUNCTION update_material_id()
RETURNS TRIGGER AS $$
BEGIN
	if new.mid <> old.mid then
    UPDATE owned_materials
    SET material_id = new.mid
    WHERE material_id = old.mid;
```

```
   end if;
        raise notice 'Material ID guncellendi';

        return new;
END;
$$ LANGUAGE 'plpgsql';

CREATE OR REPLACE TRIGGER trigger1
AFTER UPDATE ON materials
FOR EACH ROW
EXECUTE FUNCTION update_material_id();


CREATE OR REPLACE FUNCTION update_users_organization()
RETURNS TRIGGER AS $$
BEGIN
   IF TG_OP = 'UPDATE'THEN
      UPDATE users
      SET organization_id = NULL
      WHERE organization_id = OLD.organization_id;
   END IF;

        RAISE NOTICE 'Deleted from users who own this organization!';

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger2
AFTER UPDATE ON organizations
FOR EACH ROW
EXECUTE FUNCTION update_users_organization();
```