Yıldız Teknik Üniversitesi

Elektrik-Elektronik Fakültesi

Bilgisayar Mühendisliği Bölümü

BLM3041 – Veritabanı Yönetimi

Dr. Öğretim Üyesi MUSTAFA UTKU KALAY

İşBul - İş Başvuru Sistemi G:25

İsim: Doğukan Baş
No:21011003
E-posta: dogukan.bas@std.yildiz.edu.tr

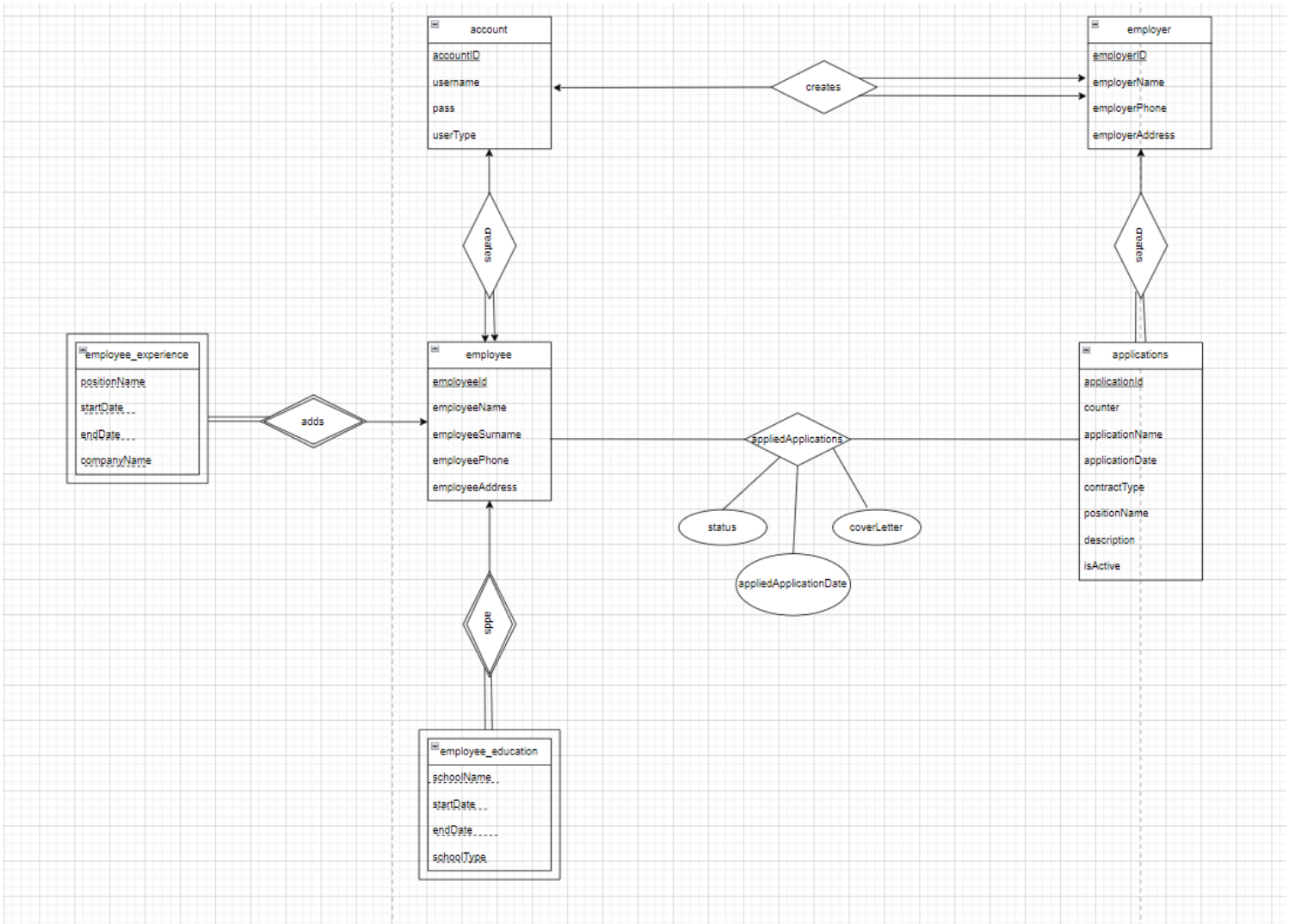
İsim: Berkan Eti
No: 21011001
E-posta:berkan.eti@std.yildiz.edu.tr

İsim: Bilal Müftüoğlu
No:20011007
E-posta: bilal.muftuoglu1@std.yildiz.edu.tr

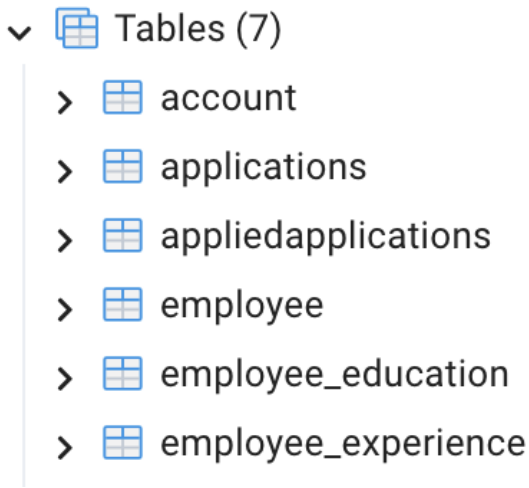
İsim: Hüseyin Said Arıcı
No: 20011015
E-posta: said.arici@std.yildiz.edu.tr

İçindekiler

ER DİYAGRAMI



KULLANILAN TABLOLAR



account: Her hesabın ID'si primary key olarak tanımlanır. Tabloda ayrıca kullanıcı adı, şifre, kullanıcı türü gibi özellikler tutulur. Parola uzunluğu kontrol edilir.

```
create table account(
    accountId int primary key,
    username varchar unique not null,
    pass varchar not null,
    userType bool not null,
    CONSTRAINT check_pass_length CHECK (LENGTH(pass) > 5 and LENGTH(pass) < 13)
);
```

	accountid [PK] integer	username character varying	pass character varying	usertype boolean
1	1	bilal	123456	true
2	2	hüseyin	123123	true
3	3	berkan	asd123	true
4	4	doğukan	654321	true
5	5	said	123asd	true
6	6	ahmet	12345678	true
7	7	mehmet	asdfasdf	true
8	8	mustafa	987654321	true
9	9	ömer	1234asdf	true
10	10	talha	asdf1234	true
11	11	bosch	boschpass	false
12	12	soft	softpass	false
13	13	yapıkredi	ykpass	false
14	14	aselsan	aselsanpass	false
15	15	tusaş	tusaşpass	false
16	16	baykar	baykarpass	false
17	17	akbank	akbankpass	false
18	18	arçelik	arçelikpass	false
19	19	vodafone	vodafonepass	false
20	20	roketsan	roketsanpass	false

employee: Her çalışanın ID'si primary key olarak tanımlanır. Aynı zamanda bu primary key, account tablosundaki accountId için bir foreign keydir. Tabloda ek olarak çalışanların ismi, soy ismi, telefonu ve adresi tutulur. Telefon numarasının uygunluğu kontrol edilir.

```
create table employee(
    employeeId int primary key references account(accountId) on delete cascade,
    employeeName varchar,
    employeeSurname varchar,
    employeePhone varchar,
    employeeAddress varchar,
    CONSTRAINT check_numeric_characters CHECK (employeePhone  ~ '^[0-9]+$')
);
```

	employeeid [PK] integer	employeename character varying	employeesurname character varying	employeephone character varying	employeeaddress character varying
1	1	Bilal	Müftüoğlu	05435434343	Bağcılar/İstanbul
2	2	Hüseyin	Arıcı	05555555555	Kayaşehir/İstanbul
3	3	Berkan	Eti	05134568745	Gaziosmanpaşa/İstanbul
4	4	Doğukan	Baş	05324567890	Beylikdüzü/İstanbul
5	5	Said	Görmez	05433456785	Esenler/İstanbul
6	6	Ahmet	Dursun	05431235432	Esenyurt/İstanbul
7	7	Mehmet	Efe	05324560987	Levent/İstanbul
8	8	Mustafa	Güzel	05063219647	Beşiktaş/İstanbul
9	9	Ömer	Diner	05057469364	Üsküdar/İstanbul
10	10	Talha	Çelik	05339743578	Bahçelievler/İstanbul

employer: Her işverenin ID'si primary key olarak tanımlanır. Aynı zamanda bu primary key, account tablosundaki accountId için bir foreign keydir. Tabloda ek olarak işverenlerin ismi, telefonu ve adresi tutulur. Telefon numarasının uygunluğu kontrol edilir.

```
create table employer(
    employerId int primary key references account(accountId) on delete cascade,
    employerName varchar,
    employerPhone varchar,
    employerAddress varchar,
    CONSTRAINT check_numeric_characters CHECK (employerPhone  ~ '^[0-9]+$')
);
```

	employerid [PK] integer	employername character varying	employerphone character varying	employeraddress character varying
1	11	Bosch	02163456789	Maltepe/İstanbul
2	12	Softtech	02126579867	Maslak/İstanbul
3	13	Yapıkredi	02624569876	Gebze
4	14	Aselsan	03123459546	Ankara
5	15	Tusaş	02124568435	Teknopark/YTÜ
6	16	Baykar	02129456794	Hadımköy
7	17	Akbank	02168357648	Maltepe
8	18	Arçelik	0212845737	İstanbul
9	19	Vodafone	02126842578	Beşiktaş
10	20	Roketsan	03123977364	Ankara

employee_education: Çalışanların ID'si, employee tablosundaki employeeId için bir foreign key olarak tanımlanmıştır. Tabloda ek olarak okul adı, okula başlangıç tarihi, okulun bitiş tarihi, okul tipi tutulur. Okul tipinin uygunluğu enum kullanılarak kontrol edilir

```
create table employee_education(
    employeeId int references employee(employeeId),
    schoolName varchar,
    startDate date,
    endDate date,
    schoolType varchar CHECK (schoolType In ('High School','Bachelors','Masters'))
    );
```

	employeeid integer	schoolname character varying	startdate date	enddate date	schooltype character varying
1	1	Yıldız Teknik Üniversitesi	2020-08-10	2025-06-06	Bachelors
2	1	Gelenbevi Anadolu Lisesi	2016-09-10	2020-05-04	High School
3	2	İstanbul Üniversitesi	2020-08-10	2025-06-06	Bachelors
4	2	Yaşar Acar Fen Lisesi	2016-09-10	2020-05-04	High School
5	3	İstanbul Teknik Üniversitesi	2021-10-11	2025-06-07	Bachelors
6	3	Çapa Fen Lisesi	2017-09-09	2021-05-06	High School
7	4	Boğaziçi Üniversitesi	2020-08-10	2025-06-06	Bachelors
8	4	Boğaziçi Üniversitesi	2026-09-07	2027-07-06	Masters
9	6	Aydın Üniversitesi	2022-08-10	2026-05-04	Bachelors
10	8	Marmara Üniversitesi	2019-03-04	2025-02-03	Bachelors
11	1	Harvard University	2024-01-13	2024-01-27	Masters

employee_experience: Çalışanların ID'si, employee tablosundaki employeeId için bir foreign key olarak tanımlanmıştır. Tabloda ek olarak geçmiş deneyimler ve bu deneyimlerin başlangıç tarihi, bitiş tarihi, pozisyon ismi, ve şirket ismi tutulur.

```
create table employee_experience(
    employeeId int references employee(employeeId),
    startDate date,
    endDate date,
    positionName varchar,
    companyName varchar);
```

	employeeid integer	startdate date	enddate date	positionname character varying	companyname character varying
1	1	2019-01-01	2022-01-01	Software Developer	Google
2	2	2023-05-08	2023-08-09	Frontend Developer	Meta
3	2	2020-02-03	2021-03-02	Project Manager	Amazon
4	3	2023-01-01	2024-01-01	Control Engineer	THY
5	4	2017-01-01	2020-02-02	RPA Developer	Twitter
6	5	2011-09-10	2013-01-06	Calibration Engineer	Vestel
7	6	2022-06-03	2022-06-06	AI Engineer	Tüpraş
8	7	2024-01-01	2024-02-03	Data Scientist	Turkcell
9	8	2021-01-03	2023-03-03	Software Developer	Aselsan
10	1	2024-01-13	2024-01-20	Mobile Developer	Kıyı Emniyet

applications: Her ilanın ID'si primary key olarak tanımlanmıştır. İşverenlerin ID'si employer tablosundaki employerId için foreign keydir. Tabloda aynı zamanda sayaç, ilan adı,ilan tarihi, sözleşme tipi, pozisyon adı, açıklama ve ilanın aktifliği tutulur.

```
create table applications(
    employerId int references employer(employerId),
    applicationId int primary key,
    counter int,
    applicationName varchar,
    applicationDate date,
    contractType varchar CHECK (contractType In ('Part Time','Full Time','Intern')),
    positionName varchar,
    description varchar,
    isActive bool
    );
```

	employerid integer	applicationid [PK] integer	counter integer	applicationname character varying	applicationdate date	contracttype character varying	positionname character varying	description character varying
1	11	2	0	Bosch Hardware Specialist	2022-05-10	Part Time	Software Developer	Bosch Hardware Specialist
2	11	3	0	Bosch System Analyst	2023-04-02	Full Time	System Analyst	Bosch System Analyst
3	12	6	0	Softtech ERP Developer	2024-01-01	Part Time	ERP Developer	Softtech ERP Developer
4	14	10	0	Aselsan Software Developer	2023-09-08	Full Time	Software Developer	Aselsan Software Developer
5	11	1	3	Bosch Software Developer	2023-01-01	Part Time	Software Developer	Bosch Software Developer
6	11	4	1	Bosch Software Intern	2022-10-01	Intern	Software Intern	Bosch Software Intern
7	12	5	3	Softtech Software Developer	2023-12-09	Full Time	Software Developer	Softtech Software Developer
8	13	7	1	Yapıkredi Software Intern	2023-10-10	Intern	Software Intern	Yapıkredi Software Intern
9	13	8	2	Yapıkredi Software Developer	2023-11-01	Full Time	Software Developer	Yapıkredi Software Developer
10	14	9	1	Aselsan Embedded System Engineer	2023-10-10	Full Time	Embedded System Engineer	Aselsan Embedded System Eng
11	11	12	0	Mobile Developer Long Term Internship	2024-01-13	Intern	Mobile Developer	For summer semester

appliedApplications: Başvuru yapan işçilerin ID'leri employee tablosundaki employeeId için foreign keydir. Aynı zamanda başvurulan ilanların ID'leri applications tablosundaki applicationId için foreign keydir. Tabloda ek olarak başvurunun durum bilgisi, başvuru tarihi ve başvuru mesajı tutulur.

```
create table appliedApplications(
    employeeId int references employee(employeeId),
    applicationId int references applications(applicationId),
    status varchar CHECK (status In ('waiting','rejected','approved','canceled')),
    coverLetter varchar,
    applicationDate date
);
```

	employeeid integer	applicationid integer	status character varying	coverletter character varying	applicationdate date
1	2	1	waiting	I am a very good software developer	2023-10-11
2	3	1	waiting	I am a master software developer	2023-09-12
3	1	4	waiting	I want to attend internship program	2023-08-10
4	1	5	waiting	I am a software developer	2023-07-11
5	2	5	waiting	I am a very good software developer	2023-06-12
6	3	5	waiting	I am a master software developer	2023-05-13
7	4	7	waiting	I want to attend internship program	2023-04-14
8	4	8	waiting	I am a software developer	2023-03-15
9	5	8	waiting	I am a very good software developer	2023-02-16
10	5	9	waiting	I am a embedded software engineer	2023-01-17
11	1	1	approved	I am a software developer	2024-01-13

KISITLAR

Silme Kısıtı - Numerik karakter kısıtı:

```
create table employer(
    employerId int primary key references account(accountId) on delete cascade,
    employerName varchar,
    employerPhone varchar,
    employerAddress varchar,
    CONSTRAINT check_numeric_characters CHECK (employerPhone  ~ '^[0-9]+$')
);

create table employee(
    employeeId int primary key references account(accountId) on delete cascade,
    employeeName varchar,
    employeeSurname varchar,
    employeePhone varchar,
    employeeAddress varchar,
    CONSTRAINT check_numeric_characters CHECK (employeePhone  ~ '^[0-9]+$')
);
```

Sayı Kısıtı:

```
create table account(
    accountId int primary key,
    username varchar unique not null,
    pass varchar not null,
    userType bool not null,
    CONSTRAINT check_pass_length CHECK (LENGTH(pass) > 5 and LENGTH(pass) < 13)
);
```

ARAYÜZDE INSERT İŞLEMİ

Kod Bloğu:

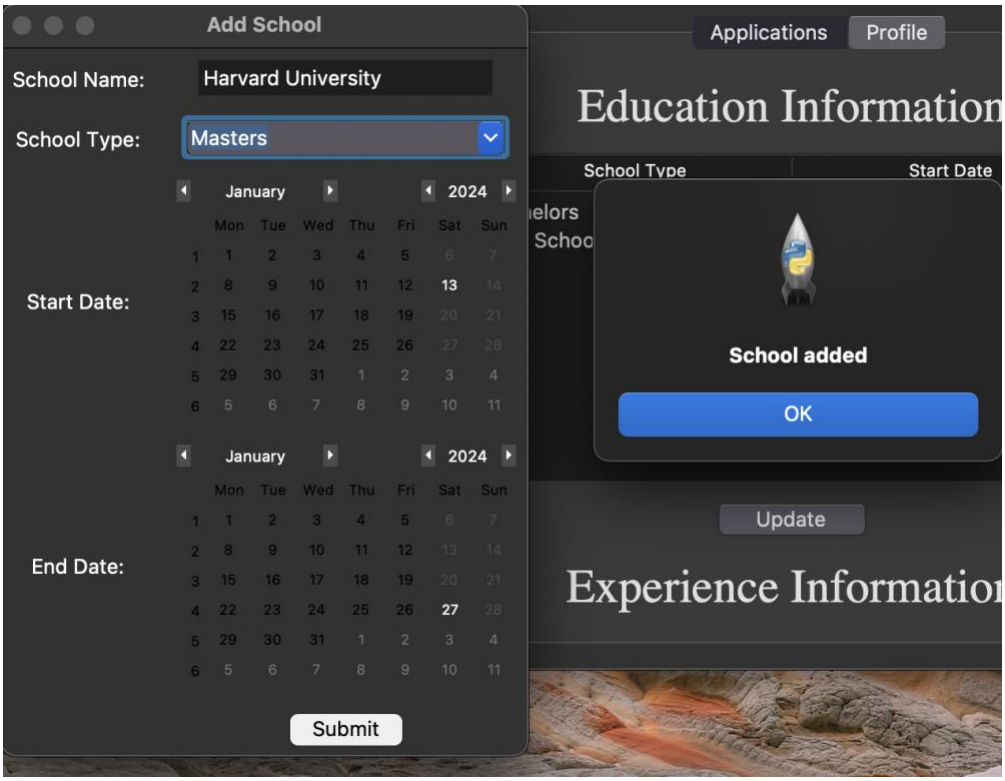
```
def addEducation(education):
    if(education.schoolName == None or education.startDate == None or education.schoolType ==
None):
        return "School name, start date and school type cannot be empty."

    if(education.endDate != None and stringToDate(education.endDate) <
stringToDate(education.startDate)):
        print(type(education.endDate))
        return "End date cannot be before start date."

    if(checkExistanceEducation(education)):
        print("School already exists")
        return "School already exists"

    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor

    try:
        insertQuery = "INSERT INTO employee_education (employeeid, schoolname, startdate,
enddate, schooltype) VALUES (%s, %s, %s, %s, %s)"
        values = (education.employeeId, education.schoolName, education.startDate,
education.endDate, education.schoolType)
        cur.execute(insertQuery,values)
        conn.commit()
        return True
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error
```

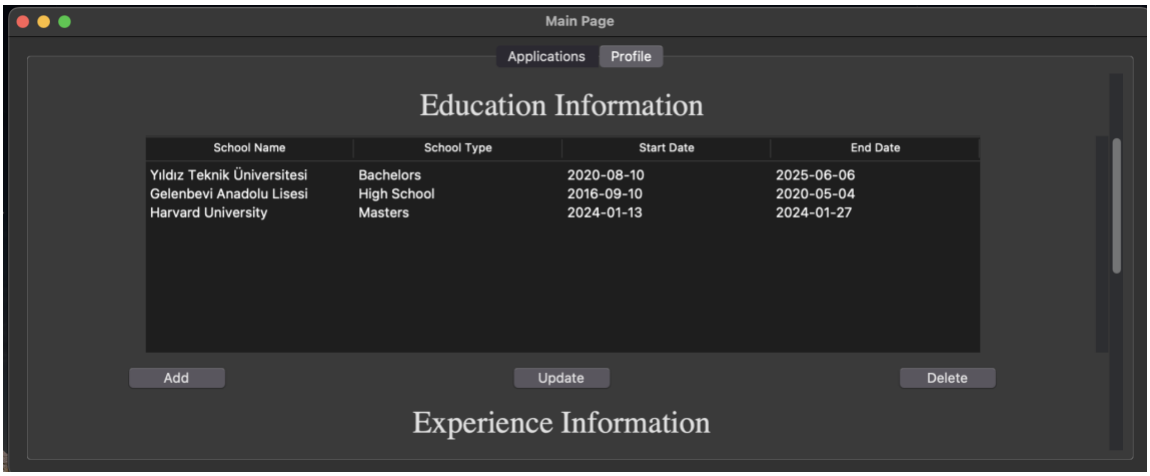
-employee_education tablosu üzerinde arayüz içinde insert işlemi ve tabloların değişimi:

employee_education tablosunun ilk hali:

	employeeid integer	schoolname character varying	startdate date	enddate date	schooltype character varying
1	1	Yıldız Teknik Üniversitesi	2020-08-10	2025-06-06	Bachelors
2	1	Gelenbevi Anadolu Lisesi	2016-09-10	2020-05-04	High School

Arayüzde insert işlemi:





Insert işlemi sonrası tablo:

	employeeid integer	schoolname character varying	startdate date	enddate date	schooltype character varying
1	1	Yıldız Teknik Üniversitesi	2020-08-10	2025-06-06	Bachelors
2	1	Gelenbevi Anadolu Lisesi	2016-09-10	2020-05-04	High School
3	1	Harvard University	2024-01-13	2024-01-27	Masters

ARAYÜZDE DELETE İŞLEMİ

Kod Bloğu:

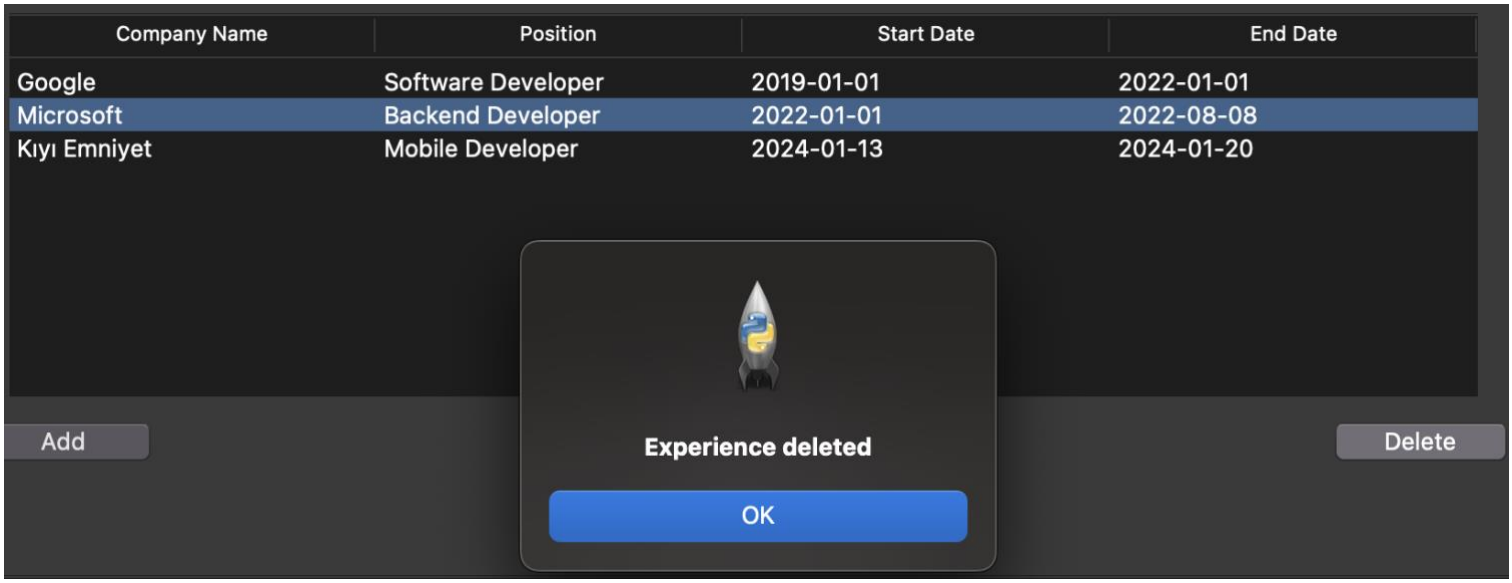
```
def deleteEducation(education):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor

    try:
        insertQuery = "DELETE FROM employee_education WHERE employeeid = %s and schoolname = %s
and startdate = %s and enddate = %s and schooltype = %s"
        values = (education.employeeId, education.schoolName, education.startDate,
education.endDate, education.schoolType)
        cur.execute(insertQuery,values)
        conn.commit()
        return True
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error
```

-employee_experience tablosu üzerinde arayüz içinde delete işlemi ve tabloların değişimi:

employee_experience tablosunun ilk hali:

	employeeid integer	startdate date	enddate date	positionname character varying	companyname character varying
1	1	2019-01-01	2022-01-01	Software Developer	Google
2	1	2022-01-01	2022-08-08	Backend Developer	Microsoft
3	1	2024-01-13	2024-01-20	Mobile Developer	Kıyı Emniyet

Arayüzde delete işlemi:



Delete işlemi sonrası tablo :

	employeeid integer	startdate date	enddate date	positionname character varying	companyname character varying
1	1	2019-01-01	2022-01-01	Software Developer	Google
2	1	2024-01-13	2024-01-20	Mobile Developer	Kıyı Emniyet

ARAYÜZDE UPDATE İŞLEMİ

Kod Bloğu:

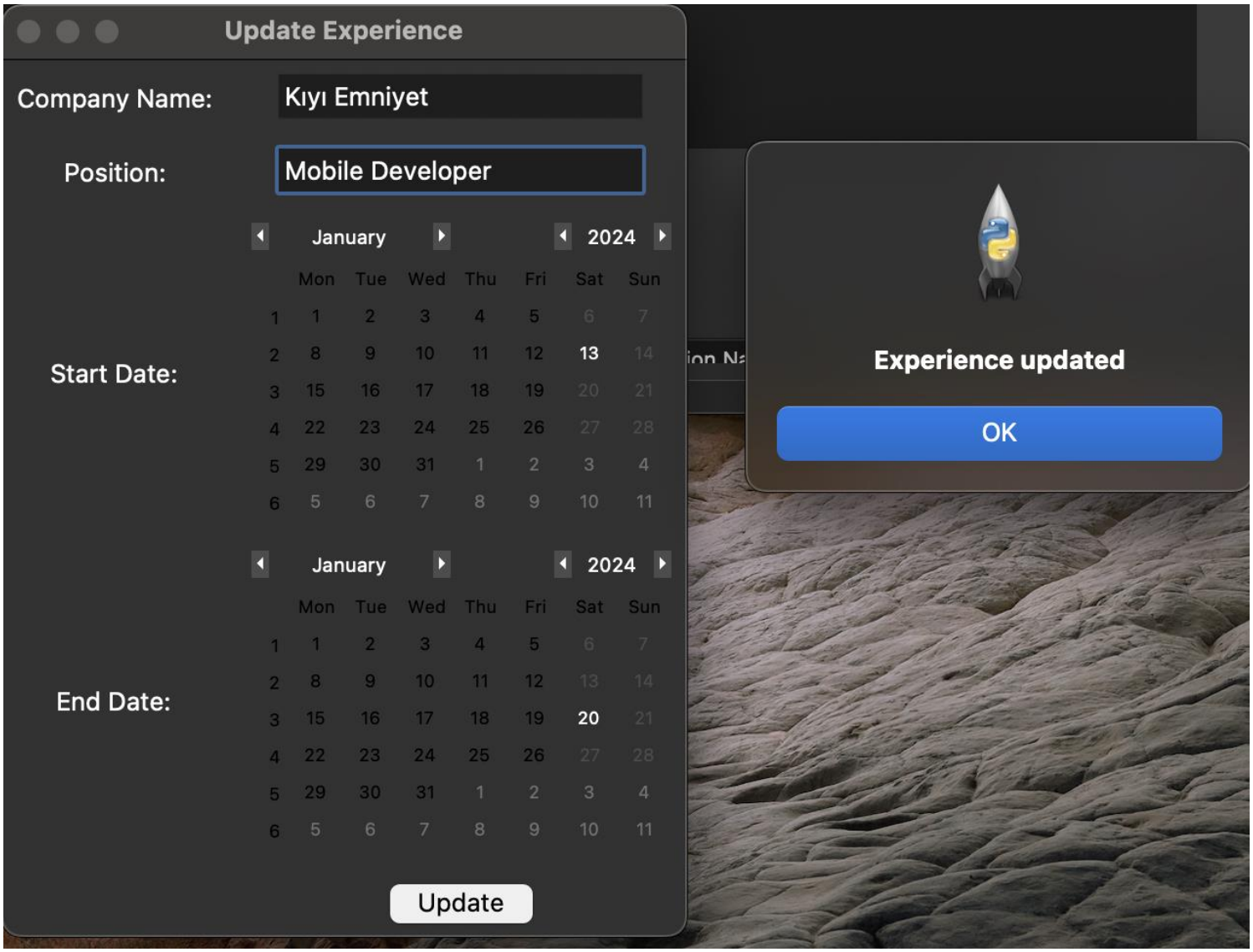
```
def updateEducation(oldEducation,newEducation):
    if(oldEducation.schoolName == newEducation.schoolName and
stringToDate(oldEducation.startDate).date() == stringToDate(newEducation.startDate) and
stringToDate(oldEducation.endDate).date() == stringToDate(newEducation.endDate) and
oldEducation.schoolType == newEducation.schoolType):
        return True

    if(newEducation.schoolName == None or newEducation.startDate == None or
newEducation.schoolType == None):
        return "School name, start date and school type cannot be empty."

    if(newEducation.endDate != None and stringToDate(newEducation.endDate) <
stringToDate(newEducation.startDate)):
        return "End date cannot be before start date."

    if(checkExistanceEducation(newEducation)):
        print("School already exists")
        return "School already exists"

    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor

    try:
        insertQuery = "UPDATE employee_education SET schoolname = %s, startdate = %s, enddate =
%s, schooltype = %s WHERE employeeid = %s and schoolname = %s and startdate = %s and enddate = %s
and schooltype = %s"
        values = (newEducation.schoolName, newEducation.startDate, newEducation.endDate,
newEducation.schoolType,oldEducation.employeeId, oldEducation.schoolName, oldEducation.startDate,
oldEducation.endDate, oldEducation.schoolType)
        cur.execute(insertQuery,values)
        conn.commit()
        return True
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error
```

-employee_experience tablosu üzerinde arayüz içinde update işlemi ve tabloların değişimi:

employee_experience tablosunun ilk hali:

	employeeid integer	startdate date	enddate date	positionname character varying	companyname character varying
1	1	2019-01-01	2022-01-01	Software Developer	Google
2	1	2022-01-01	2022-08-08	Backend Developer	Microsoft
3	1	2024-01-13	2024-01-27	Mobile Dev	Kıyı Emniyet

Arayüzde update işlemi:



Update işlemi sonrası tablo :

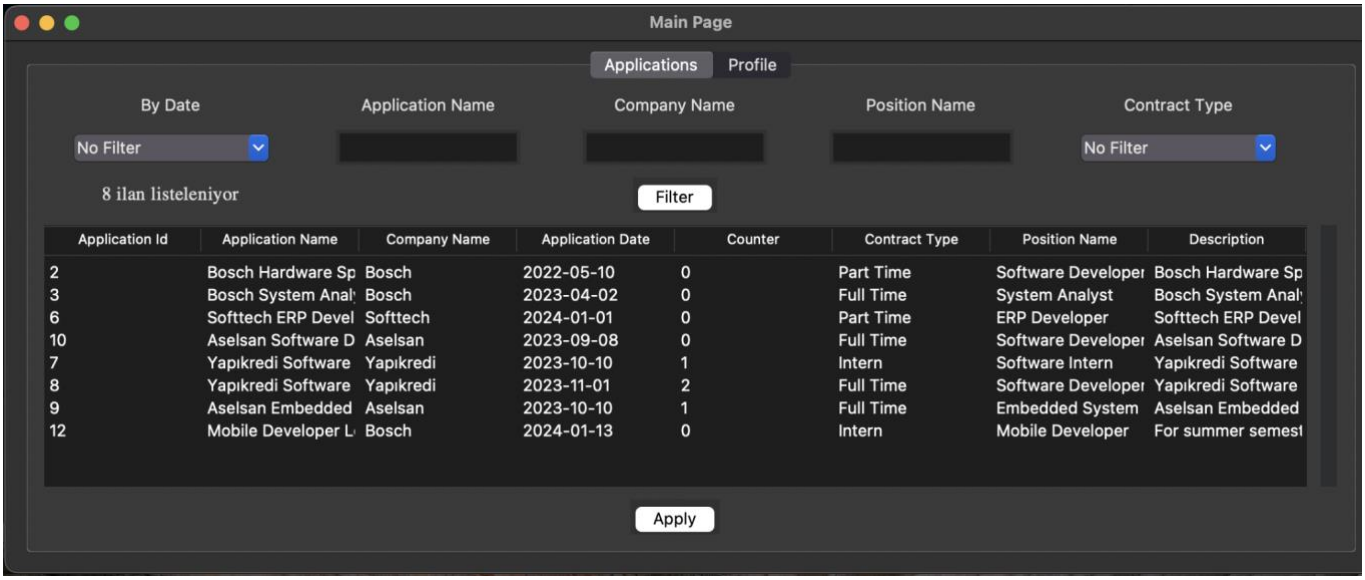
	employeeid integer	startdate date	enddate date	positionname character varying	companyname character varying
1	1	2019-01-01	2022-01-01	Software Developer	Google
2	1	2022-01-01	2022-08-08	Backend Developer	Microsoft
3	1	2024-01-13	2024-01-20	Mobile Developer	Kıyı Emniyet

ARAYÜZDEN GİRİLEN DEĞERE GÖRE SONUÇ LİSTELEME

Kod Bloğu:

```
def showAllApplications(employeeId):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    try:
        query="Select *  from applications where isActive = true and applicationId not in (Select
applicationId from appliedapplications where employeeId = " + str(employeeId) + ")"
        cur.execute(query)
        applications = cur.fetchall()
        applicationList = []
        # query for count of applicants
        query="SELECT COUNT(*) FROM applications WHERE isActive = true and applicationId not in
(Select applicationId from appliedapplications where employeeId = " + str(employeeId) + ")"
        cur.execute(query)
        count = cur.fetchone()[0]
        for app in applications:
            newApplication =
Entities.Application(app[0],app[1],app[2],app[3],app[4],app[5],app[6],app[7],app[8])
            query="SELECT employername FROM employer where employerId = %s"
            values=(app[0],)

            cur.execute(query,values)
            employerName=cur.fetchone()[0]
            print("employename: ",employerName)
            applicationList.append((employerName,newApplication))
        return applicationList,count
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error,0
```

Arayüzde gösterimi:



KULLANILAN VIEWLAR

Veritabanımızda kullanılan Viewlar aşağıdaki gibidir:

applicationView'ın tanımlanması :

```
CREATE VIEW applicationView AS
SELECT
    aa.employeeId,
    app.applicationId,
    employer.employerName,
    app.counter,
    app.applicationName,
    app.applicationDate,
    app.contractType,
    app.positionName,
    app.description,
    aa.status,
    aa.coverLetter,
    aa.applicationDate AS appliedApplicationDate
FROM
    applications app
JOIN
    appliedApplications aa ON app.applicationId = aa.applicationId
join employer on employer.employerId = app.employerId;
```

	employeeid integer	applicationid integer	employernam character var	counter integer	applicationname character varying	applicationdate date	contracttype character varying	positionname character varying	description character varying	status character var	coverletter character varying	appliedapplicationd date
1	2	1	Bosch	3	Bosch Software Devel...	2023-01-01	Part Time	Software Developer	Bosch Software...	waiting	I am a very g...	2023-10-11
2	3	1	Bosch	3	Bosch Software Devel...	2023-01-01	Part Time	Software Developer	Bosch Software...	waiting	I am a maste...	2023-09-12
3	1	4	Bosch	1	Bosch Software Intern	2022-10-01	Intern	Software Intern	Bosch Software...	waiting	I want to atte...	2023-08-10
4	1	5	Softtech	3	Softtech Software De...	2023-12-09	Full Time	Software Developer	Softtech Softw...	waiting	I am a softwa...	2023-07-11
5	2	5	Softtech	3	Softtech Software De...	2023-12-09	Full Time	Software Developer	Softtech Softw...	waiting	I am a very g...	2023-06-12
6	3	5	Softtech	3	Softtech Software De...	2023-12-09	Full Time	Software Developer	Softtech Softw...	waiting	I am a maste...	2023-05-13
7	4	7	Yapıkredi	1	Yapıkredi Software Int...	2023-10-10	Intern	Software Intern	Yapıkredi Softw...	waiting	I want to atte...	2023-04-14
8	4	8	Yapıkredi	2	Yapıkredi Software De...	2023-11-01	Full Time	Software Developer	Yapıkredi Softw...	waiting	I am a softwa...	2023-03-15
9	5	8	Yapıkredi	2	Yapıkredi Software De...	2023-11-01	Full Time	Software Developer	Yapıkredi Softw...	waiting	I am a very g...	2023-02-16
10	5	9	Aselsan	1	Aselsan Embedded Sy...	2023-10-10	Full Time	Embedded System Engineer	Aselsan Embed...	waiting	I am a embed...	2023-01-17
11	1	1	Bosch	3	Bosch Software Devel...	2023-01-01	Part Time	Software Developer	Bosch Software...	approved	I am a softwa...	2024-01-13

Arayüzden çağrılan View sorgusu:

```
def getApplicationView(employeeId) :
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    try:
        query = "SELECT * FROM applicationView where employeeId = %s"
        values = (employeeId,)
        cur.execute(query,values)
        applications = cur.fetchall()
        applicationList = []
        for app in applications:
            newApplication = Entities.ApplicationView(app[0],app[1],app[2],app[3],app[4],app[5],app[6],app[7],app[8],app[9],app[10],app[11])
            applicationList.append(newApplication)
        return True,applicationList
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return False,error
```

applicantsView'ın tanımlanması:

```
CREATE VIEW applicantsView AS
SELECT
    aa.employeeId,
    aa.applicationId,
    aa.status,
    aa.coverLetter,
    aa.applicationDate,
    emp.employeeName,
    emp.employeeSurname,
    emp.employeePhone,
    emp.employeeAddress
FROM
    employee emp
JOIN
    appliedApplications aa ON emp.employeeId = aa.employeeId;
```

	applicationid integer	status character varying	coverletter character varying	applicationdate date	employeename character varying	employeesurname character varying	employeephone character varying	employeeaddress character varying
1	1	waiting	I am a very good software developer	2023-10-11	Hüseyin	Arıcı	05555555555	Kayaşehir/İstanbul
2	1	waiting	I am a master software developer	2023-09-12	Berkan	Eti	05134568745	Gaziosmanpaşa/İstanbul
3	4	waiting	I want to attend internship program	2023-08-10	Bilal	Müftüoğlu	05435434343	Bağcılar/İstanbul
4	5	waiting	I am a software developer	2023-07-11	Bilal	Müftüoğlu	05435434343	Bağcılar/İstanbul
5	5	waiting	I am a very good software developer	2023-06-12	Hüseyin	Arıcı	05555555555	Kayaşehir/İstanbul
6	5	waiting	I am a master software developer	2023-05-13	Berkan	Eti	05134568745	Gaziosmanpaşa/İstanbul
7	7	waiting	I want to attend internship program	2023-04-14	Doğukan	Baş	05324567890	Beylikdüzü/İstanbul
8	8	waiting	I am a software developer	2023-03-15	Doğukan	Baş	05324567890	Beylikdüzü/İstanbul
9	8	waiting	I am a very good software developer	2023-02-16	Said	Görmez	05433456785	Esenler/İstanbul
10	9	waiting	I am a embedded software engineer	2023-01-17	Said	Görmez	05433456785	Esenler/İstanbul
11	1	approved	I am a software developer	2024-01-13	Bilal	Müftüoğlu	05435434343	Bağcılar/İstanbul

Arayüzden çağrılan View sorgusu:

```
def getApplicantsView(applicationId):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    try:
        query = "SELECT * FROM applicantsView where applicationId = %s and status = 'waiting'"
        values = (applicationId,)
        cur.execute(query,values)
        applicants = cur.fetchall()
        applicantList = []
        for app in applicants:
            newApplicant = Entities.ApplicantsView(app[0],app[1],app[2],app[3],app[4],app[5],app[6],app[7],app[8])
            applicantList.append(newApplicant)
        return True,applicantList
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return False,error
```

KULLANILAN SEQUENCE VE INSERT İŞLEMİNDE KULLANIMI

Sequenceların üretilmesi:

```
create sequence accountIdGenerator start with 1 increment by 1 no cycle;
create sequence advertisementIdGenerator start with 1 increment by 1 no cycle;
```

Insert işleminde kullanılması(advertisementIdGenerator): nextval kullanımı ile sıradaki sequence değerinin erişimi sağlanır.

```
if(application.applicationName == None or application.contractType == None or application.positionName == None):
    return False,"Application name, contract type and position cannot be empty."
conn = Helper.DataBaseConnector.singleton.connection
cur = Helper.DataBaseConnector.singleton.cursor
cur.execute("SELECT nextval('advertisementIdGenerator')")
applicationId= cur.fetchone()[0]
```

```
def addApplication(application):
    if(application.applicationName == None or application.contractType == None or
application.positionName == None):
        return False,"Application name, contract type and position cannot be empty."
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    cur.execute("SELECT nextval('advertisementIdGenerator')")
    applicationId= cur.fetchone()[0]

    try:
        insertQuery = "INSERT INTO applications (applicationId, applicationName, applicationDate,
contractType, positionName, description, employerId, counter,isactive) VALUES (%s, %s, %s, %s,
%s, %s, %s, %s,%s)"

        applicationDate= datetime.now().strftime("%Y-%m-%d")
        values = (applicationId,application.applicationName, applicationDate,
application.contractType, application.positionName, application.description,
application.employerId, 0,True)

        cur.execute(insertQuery,values)
        conn.commit()
        return True,
Entities.Application(application.employerId,applicationId,0,application.applicationName,applicati
onDate,application.contractType,application.positionName,application.description,application.isAc
tive)

    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error
```

Insert işleminde kullanılması(accountIdGenerator): nextval kullanımı ile sıradaki sequence değerinin erişimi sağlanır.

```python
def registerEmployer(employer,account):
    if(employer.employerName == None) :
        return "Name cannot be empty."
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    try:
        insertQuery="INSERT INTO account(accountId,username,pass,userType) VALUES(%s,%s,%s,%s)"
        cur.execute("SELECT nextval('accountIdGenerator')")
        accountID= cur.fetchone()[0]
        values=(accountID,account.userName,account.password,account.userType)
        cur.execute(insertQuery,values)
        insertQuery = "UPDATE employer SET employerName = %s, employerPhone = %s, employerAddress = %s WHERE employerId = %s"
        values = (employer.employerName, employer.employerPhone,employer.employerAdress, accountID)
        cur.execute(insertQuery,values)
        conn.commit()

    except( psycopg2.errors.UniqueViolation) as error:
        print("UniqueViolation" + employer.employerName + " " + employer.employerPhone + " " + employer.employerAdress)
        conn.rollback()
        return "The username already taken. Please try another one."

    except( psycopg2.errors.NotNullViolation) as error2:
        print("NotNullViolation")
        conn.rollback()
        return "Username and password cannot be empty."

    except( psycopg2.DatabaseError) as error:
        print(error)
        isPassError = str(error).find("check_pass_length")
        print("Transaction")
        conn.rollback()
        if(isPassError != -1):
            return ("Password must be between 6 and 12 characters.")
        return ("Unvalid Phone Number. Please try again.")
```

```python
def registerEmployee(employee,account):
    if (employee.employeeName == None or employee.employeeSurname == None):
        return "Name and Surname cannot be empty."

    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    try:
        insertQuery="INSERT INTO account(accountId,username,pass,userType) VALUES(%s,%s,%s,%s)"
        cur.execute("SELECT nextval('accountIdGenerator')")
        accountID= cur.fetchone()[0]
        values=(accountID,account.userName,account.password,account.userType)
        cur.execute(insertQuery,values)
        insertQuery = "UPDATE employee SET employeeName = %s, employeeSurname = %s, employeePhone = %s, employeeAddress = %s where employeeId
        values = (employee.employeeName, employee.employeeSurname,employee.employeePhone,employee.employeeAddress, accountID)
        cur.execute(insertQuery,values)
        conn.commit()
    except(psycopg2.errors.UniqueViolation) as error:
        print("UniqueViolation")
        conn.rollback()
        return "The username already taken. Please try another one."

    except( psycopg2.errors.NotNullViolation) as error2:
        print("NotNullViolation")
        conn.rollback()
        return "Username and password cannot be empty."

    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        isPassError = str(error).find("check_pass_length")
        print("Transaction")
        conn.rollback()
        if(isPassError != -1):
            return ("Password must be between 6 and 12 characters.")
        return ("Unvalid Phone Number. Please try again.")

    return True
```

INTERSECT KULLANIMI

-Filtreleme işlemi sırasında girilen filtre değerlerine ait sorguyu elde etmek için intersect kullanılmıştır.

```
def filterApplications(employeeId,filter):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    queryCounter = "Select count(*) from ("
    try:
        query="Select * from applications where  isActive = true and applicationId not in (Select
applicationId from appliedapplications where employeeId = " + str(employeeId) + ")"
        if(filter.applicationDate != None):
            if(filter.applicationDate == "Ascending"):
                query += " order by applicationDate asc"
            elif(filter.applicationDate == "Descending"):
                query += " order by applicationDate desc"

        if(filter.applicationName != None):
            query += " intersect "
            query += "Select * from applications where UPPER(applicationName) like UPPER('%" +
filter.applicationName + "%')"

        if(filter.companyName != None):
            query += " intersect "
            query += "Select * from applications where employerId in (Select employerId from
employer where UPPER(employerName) like UPPER('%" + filter.companyName + "%'))"

        if(filter.positionName != None):
            query += " intersect "
            query += "Select * from applications where UPPER(positionName) like UPPER('%" +
filter.positionName + "%')"

        if(filter.contractType != None):
            query += " intersect "
            query += "Select * from applications where UPPER(contractType) like UPPER('%" +
filter.contractType + "%')"

        queryCounter += query + ")I"
        cur.execute(query)
        applications = cur.fetchall()
        applicationList = []
        cur.execute(queryCounter)
```

AGGREGATE FONKSİYONU VE HAVING KULLANIMI

showAllApplications() fonksiyonunda count kullanımı: Aktif ve başvurulmayan başvuru sayısını verir.

```
def showAllApplications(employeeId):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    try:
        query="Select *  from applications where isActive = true and applicationId not in (Select
applicationId from appliedapplications where employeeId = " + str(employeeId) + ")"
        cur.execute(query)
        applications = cur.fetchall()
        applicationList = []
        # query for count of applicants
        query="SELECT COUNT(*) FROM applications WHERE isActive = true and applicationId not in
(Select applicationId from appliedapplications where employeeId = " + str(employeeId) + ")"
        cur.execute(query)
        count = cur.fetchone()[0]
        for app in applications:
            newApplication =
Entities.Application(app[0],app[1],app[2],app[3],app[4],app[5],app[6],app[7],app[8])
            query="SELECT employername FROM employer where employerId = %s"
            values=(app[0],)

            cur.execute(query,values)
            employerName=cur.fetchone()[0]
            print("employename: ",employerName)
            applicationList.append((employerName,newApplication))
        return applicationList,count
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error,0
```

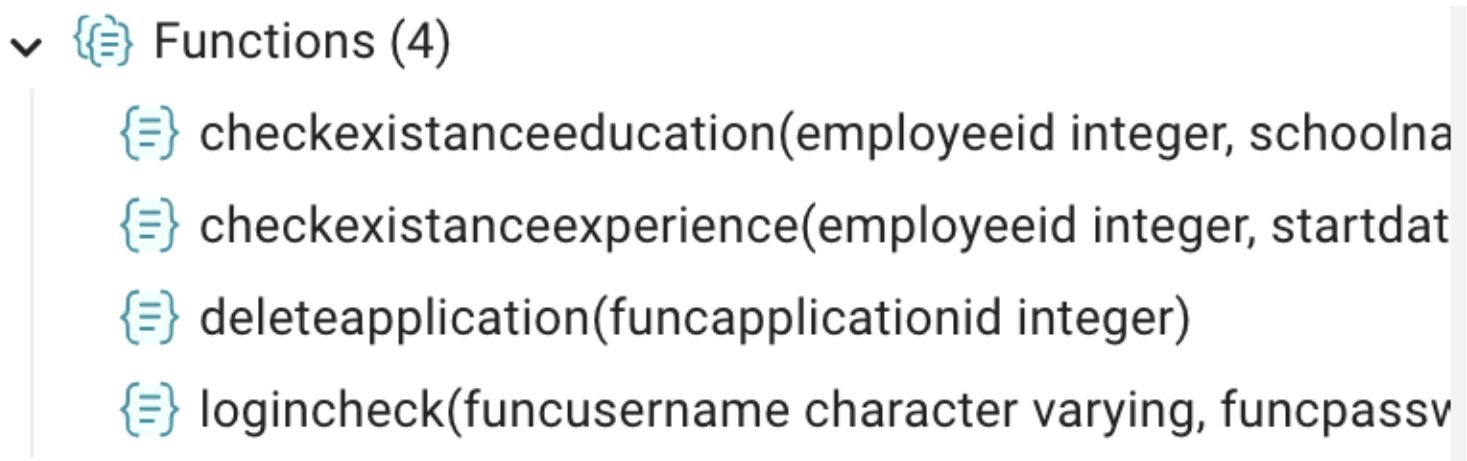
filterApplications() fonksiyonunda count kullanımı: Girilen filtreye uyan başvuru sayısını verir

```
def filterApplications(employeeId,filter):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    queryCounter = "Select count(*) from ("
    try:
        query="Select * from applications where  isActive = true and applicationId not in (Select
applicationId from appliedapplications where employeeId = " + str(employeeId) + ")"
        if(filter.applicationDate != None):
            if(filter.applicationDate == "Ascending"):
                query += " order by applicationDate asc"
            elif(filter.applicationDate == "Descending"):
                query += " order by applicationDate desc"

        if(filter.applicationName != None):
            query += " intersect "
            query += "Select * from applications where UPPER(applicationName) like UPPER('%" +
filter.applicationName + "%')"

        if(filter.companyName != None):
            query += " intersect "
            query += "Select * from applications where employerId in (Select employerId from
employer where UPPER(employerName) like UPPER('%" + filter.companyName + "%'))"

        if(filter.positionName != None):
            query += " intersect "
            query += "Select * from applications where UPPER(positionName) like UPPER('%" +
filter.positionName + "%')"

        if(filter.contractType != None):
            query += " intersect "
            query += "Select * from applications where UPPER(contractType) like UPPER('%" +
filter.contractType + "%')"

        queryCounter += query + ")I"
        cur.execute(query)
        applications = cur.fetchall()
        applicationList = []
        cur.execute(queryCounter)
```

applyApplication() fonksiyonu içinde count ve having kullanımı: Count, başvurulan ilanlardan 3 tanesi waiting durumundaysa yeni başvuru yapmayı engelleyen controlQueryde görev alır.

```
def applyApplication(appliedApplication):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    try:
        controlQuery = "select count(*) from appliedapplications where status = 'waiting' and
employeeid=%s having count(*) = 3"
        values = (appliedApplication.employeeId,)
        cur.execute(controlQuery,values)
        count = cur.fetchone()
        if(count !=  None):
            return "You have reached the maximum number of waiting applications."

        insertQuery = "INSERT INTO appliedapplications (employeeId, applicationId, status,
applicationDate,coverLetter) VALUES (%s, %s, %s, %s, %s)"
        applicationDate= datetime.now().strftime("%Y-%m-%d")
        values = (appliedApplication.employeeId, appliedApplication.applicationId,
appliedApplication.status, applicationDate, appliedApplication.coverLetter)
        cur.execute(insertQuery,values)
        conn.commit()
        return True
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error
```

KULLANILAN SQL FONKSİYONLARI



Record tanımı örneği:

```
create  type loginCheckType as (userType varchar, accountidd int);
```

loginCheck fonksiyonu ve cursor kullanımı: Login esnasında kullanıcının hangi tipte giriş yaptığını ve ID'sini record olarak döndürür.

```
CREATE OR REPLACE FUNCTION loginCheck(funcUsername varchar, funcPassword varchar)
RETURNS loginCheckType as $$

DECLARE
loginInfo loginCheckType;
counter int :=0;
accountCheck cursor for select username,pass,userType,accountid from account ;
begin
    loginInfo.userType:='No such user';
    loginInfo.accountidd := -1;
    FOR account_row in accountCheck LOOP
    if(account_row.username=funcUsername and account_row.pass=funcPassword) THEN
        loginInfo.accountidd=account_row.accountid ;
        IF(account_row.userType = true) then
            loginInfo.userType='Employee';
        ELSE
            loginInfo.userType='Employer';
        END IF ;
    END IF;
    END LOOP;
    return loginInfo;
END;
$$ LANGUAGE 'plpgsql';
```

Fonksiyonun çağırılması:

```
def loginCheck(account):
    if(account.userName is None or account.password is None):
        return "Username and password cannot be empty.",None
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor

    try:
        query = "SELECT * from loginCheck(%s,%s)"
        values = (account.userName, account.password)
        cur.execute(query,values)
        accountInfo = cur.fetchone()

        userType,accountId = accountInfo
        if(userType == "No such user"):
            return userType,None
        return userType,accountId

    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        return error
```

deleteApplication fonksiyonu: Application'ın aktiflik durumunu false yapar ve ilgili ilanları canceled yapar.

```
CREATE OR REPLACE FUNCTION deleteApplication(funcApplicationId int)
returns void as $$
BEGIN
    UPDATE applications
    SET isActive = false
    WHERE applicationId = funcApplicationId;

    UPDATE appliedApplications
    SET status = 'canceled'
    WHERE applicationId = funcApplicationId;

END;
$$ LANGUAGE 'plpgsql';
```

Fonksiyonun çağırılması:

```
def deleteApplication(application):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor
    applicationId= application.applicationId
    try:
        insertQuery = "select deleteapplication(%s)"
        values = (applicationId,)
        cur.execute(insertQuery,values)
        conn.commit()
        return True
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error
```

checkExistanceEducation fonksiyonu ve cursor kullanımı: Parametreleri verilen educationın hali hazırda tableda olup olmadığını kontrol eder.

```
CREATE OR REPLACE FUNCTION checkExistanceEducation(employeeId int,schoolname varchar,startdate date,enddate date,schooltype varchar)
returns bool as $$
DECLARE
isExist bool;
exist_cursor CURSOR FOR SELECT * from employee_education;
BEGIN
    isExist := false;
    FOR row_cursor in exist_cursor LOOP
        IF(employeeId = row_cursor.employeeId and schoolname = row_cursor.schoolname and startdate = row_cursor.startdate and enddate = row_cursor.end
            isExist := true;
        END IF;
    END LOOP;
    RETURN isExist;
END;
$$ LANGUAGE 'plpgsql';
```

Fonksiyonun çağırılması:

```
def checkExistanceEducation(education):
    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor

    try:
        query = "SELECT checkExistanceEducation(%s,%s,%s,%s,%s)"
        values = (education.employeeId, education.schoolName, education.startDate, education.endDate, education.schoolType)
        cur.execute(query,values)
        print(education)
        education = cur.fetchone()
        return education[0]

    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error
```

checkExistanceExperience fonksiyonu ve cursor kullanımı: Parametreleri verilen experience'ın halihazırda table'da olup olmadığını kontrol eder.

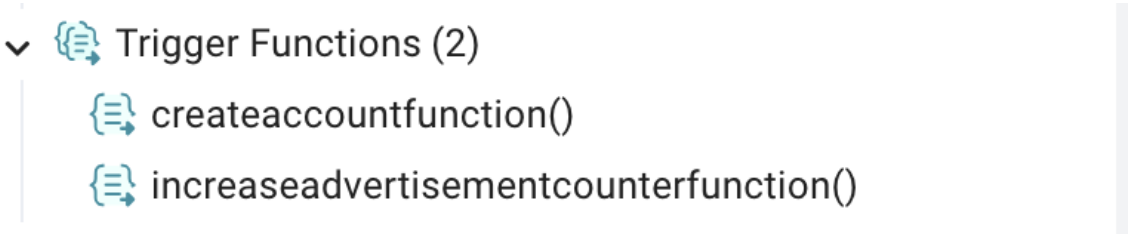
```
CREATE OR REPLACE FUNCTION checkExistanceExperience(employeeId int, startdate date, enddate date, positionname varchar, companyname varchar)
returns bool as $$
DECLARE
isExist bool;
exist_cursor CURSOR FOR SELECT * from employee_experience;
BEGIN
    isExist := false;
    FOR row_cursor in exist_cursor LOOP
        IF(employeeId = row_cursor.employeeId  and startdate = row_cursor.startdate and enddate = row_cursor.enddate and positionname = row_cursor.pos
            isExist := true;
        END IF;
    END LOOP;
    RETURN isExist;
END;
$$ LANGUAGE 'plpgsql';
```

Fonksiyonun çağırılması:

```python
def checkExistanceExperience(experience):

    conn = Helper.DataBaseConnector.singleton.connection
    cur = Helper.DataBaseConnector.singleton.cursor

    try:
        query = "SELECT checkExistanceExperience(%s,%s,%s,%s,%s)"
        values = (experience.employeeId, experience.startDate, experience.endDate, experience.positionName,experience.companyName)
        cur.execute(query,values)
        experience = cur.fetchone()
        print(experience)
        return experience[0]

    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
        conn.rollback()
        return error
```

KULLANILAN TRIGGERLAR

Veritabanımızda kullanılan triggerlar aşağıdaki gibidir:



createaccountfunction(): Her account oluşturulduğunda tipine göre ilgili tabloya aynı ID'de veri ekler

```sql
CREATE OR REPLACE FUNCTION createAccountFunction()
RETURNS TRIGGER AS $$
BEGIN
    case new.userType
       when False then insert into employer values(new.accountId,null,null,null);
       when True then insert into employee values(new.accountId,null,null,null,null);

    end case;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER createAccountTrigger
AFTER INSERT ON account
FOR EACH ROW
WHEN (NEW.accountId IS NOT NULL)
EXECUTE FUNCTION createAccountFunction();
```

Triggerın gerçekleştirilmesi:

	accountid [PK] integer	username character varying	pass character varying	usertype boolean
1	1	bilal	123456	true
2	2	hüseyin	123123	true
3	3	berkan	asd123	true
4	4	doğukan	654321	true
5	5	said	123asd	true
6	6	ahmet	12345678	true
7	7	mehmet	asdfasdf	true
8	8	mustafa	987654321	true
9	9	ömer	1234asdf	true
10	10	talha	asdf1234	true
11	11	bosch	boschpass	false
12	12	soft	softpass	false
13	13	yapıkredi	ykpass	false
14	14	aselsan	aselsanpass	false
15	15	tusaş	tusaşpass	false
16	16	baykar	baykarpass	false
17	17	akbank	akbankpass	false
18	18	arçelik	arçelikpass	false
19	19	vodafone	vodafonepass	false
20	20	roketsan	roketsanpass	false

	employeeid [PK] integer	employeename character varying	employeesurname character varying	employeephone character varying	employeeaddress character varying
1	1	[null]	[null]	[null]	[null]
2	2	[null]	[null]	[null]	[null]
3	3	[null]	[null]	[null]	[null]
4	4	[null]	[null]	[null]	[null]
5	5	[null]	[null]	[null]	[null]
6	6	[null]	[null]	[null]	[null]
7	7	[null]	[null]	[null]	[null]
8	8	[null]	[null]	[null]	[null]
9	9	[null]	[null]	[null]	[null]
10	10	[null]	[null]	[null]	[null]

	employerid [PK] integer	employername character varying	employerphone character varying	employeraddress character varying
1	11	[null]	[null]	[null]
2	12	[null]	[null]	[null]
3	13	[null]	[null]	[null]
4	14	[null]	[null]	[null]
5	15	[null]	[null]	[null]
6	16	[null]	[null]	[null]
7	17	[null]	[null]	[null]
8	18	[null]	[null]	[null]
9	19	[null]	[null]	[null]
10	20	[null]	[null]	[null]

increaseadvertisementcounterfunction(): Başvuru yapılan ilanın counterını 1 arttırır.

```
CREATE OR REPLACE FUNCTION increaseAdvertisementCounterFunction()
RETURNS TRIGGER AS $$
BEGIN
    update applications as adv set counter = counter + 1 where new.applicationId = adv.applicationId;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER increaseAdvertisementCounterTrigger
AFTER INSERT ON appliedApplications
FOR EACH ROW
WHEN (NEW.applicationId IS NOT NULL)
EXECUTE FUNCTION increaseAdvertisementCounterFunction();
```

Triggerın gerçekleştirilmesi:

	employeeid integer	applicationid integer	status character varying	coverletter character varying	applicationdate date
1	2	1	waiting	I am a very good software developer	2023-10-11
2	3	1	waiting	I am a master software developer	2023-09-12
3	1	4	waiting	I want to attend internship program	2023-08-10
4	1	5	waiting	I am a software developer	2023-07-11
5	2	5	waiting	I am a very good software developer	2023-06-12
6	3	5	waiting	I am a master software developer	2023-05-13
7	4	7	waiting	I want to attend internship program	2023-04-14
8	4	8	waiting	I am a software developer	2023-03-15
9	5	8	waiting	I am a very good software developer	2023-02-16
10	5	9	waiting	I am a embedded software engineer	2023-01-17
11	1	1	approved	I am a software developer	2024-01-13

	employerid integer	applicationid [PK] integer	counter integer	applicationname character varying	applicationdate date	contracttype character varying	positionname character varying	description character varying
1	11	2	0	Bosch Hardware Specialist	2022-05-10	Part Time	Software Developer	Bosch Hardware Specialist
2	11	3	0	Bosch System Analyst	2023-04-02	Full Time	System Analyst	Bosch System Analyst
3	12	6	0	Softtech ERP Developer	2024-01-01	Part Time	ERP Developer	Softtech ERP Developer
4	14	10	0	Aselsan Software Developer	2023-09-08	Full Time	Software Developer	Aselsan Software Developer
5	11	1	3	Bosch Software Developer	2023-01-01	Part Time	Software Developer	Bosch Software Developer
6	11	4	1	Bosch Software Intern	2022-10-01	Intern	Software Intern	Bosch Software Intern
7	12	5	3	Softtech Software Developer	2023-12-09	Full Time	Software Developer	Softtech Software Developer
8	13	7	1	Yapıkredi Software Intern	2023-10-10	Intern	Software Intern	Yapıkredi Software Intern
9	13	8	2	Yapıkredi Software Developer	2023-11-01	Full Time	Software Developer	Yapıkredi Software Developer
10	14	9	1	Aselsan Embedded System Engineer	2023-10-10	Full Time	Embedded System Engineer	Aselsan Embedded System Enç
11	11	12	0	Mobile Developer Long Term Internship	2024-01-13	Intern	Mobile Developer	For summer semester