İSTANBUL TOPKAPI ÜNİVERSİTESİ

Ders Adı: Veritabanı Yönetim Sistemleri

Ödev: SQL Ödevi -2

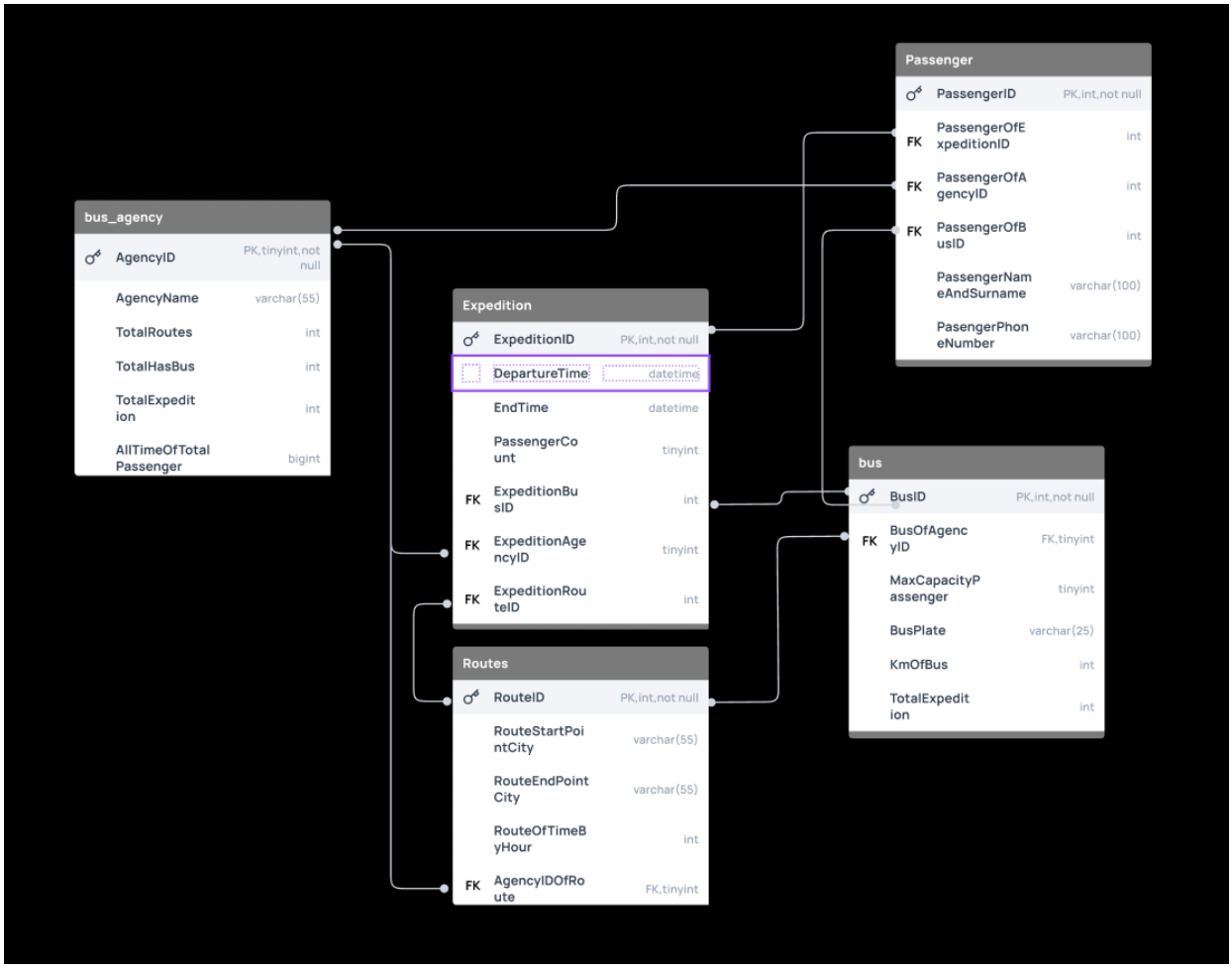
Öğrenci Adı ve Soyadı: Berkant Çalıkuşu

Öğrenci NO: 2004301064

Bölüm: Yazılım Mühendisliği

Dönem:2.Sınıf

DİAGRAM HATIRLATMA



VIEW-1 ORNEK

VIEW ile kardeşler turizime ait mevcut olan seferlerdeki yolcularının isimlerini oluştur.

```
CREATE VIEW kardeslerTurizimSeferindekiYolcular AS
SELECT
AgencyName AS Acenta_Ismi,
ExpeditionID AS Sefer_No,
PassengerNameAndSurname AS Yolcu,
b.BusPlate AS Plaka,
b.BusOfAgencyID AS AcentaID
FROM ((bus_agency BA
INNER JOIN expedition e ON e.ExpeditionAgencyID = BA.AgencyID AND
e.ExpeditionAgencyID = 1)
INNER JOIN bus b ON b.BusID = e.ExpeditionBusID
```

```
INNER JOIN passenger_table p ON p.PassengerOfExpeditionID = e.ExpeditionID

);

SELECT

AgencyName AS Acenta_Ismi,

ExpeditionID AS Sefer_No,

PassengerNameAndSurname AS Yolcu,

b.BusPlate AS Plaka,

b.BusOfAgencyID AS AcentaID

FROM ((bus_agency BA

INNER JOIN expedition e ON e.ExpeditionAgencyID = BA.AgencyID AND
e.ExpeditionAgencyID = 1)

INNER JOIN bus b ON b.BusID = e.ExpeditionBusID

INNER JOIN passenger_table p ON p.PassengerOfExpeditionID = e.ExpeditionID

);
```

VIEW-2 ORNEK

Rota uzunluğu 7 saatten düşük olan ve İstanbul turizime ait rotaları rota detaylarıyla birlikte view olarak oluşturun

```
CREATE VIEW istanbulSeyehatRotaları AS

SELECT
roa.RouteID,AgencyName,RouteStartPointCity,roa.RouteEndPointCity,roa.RouteOfTime
ByHour

FROM routes_of_agency roa,bus_agency ba

WHERE ba.AgencyID = 2 AND roa.RouteOfTimeByHour<7

SELECT * FROM kardeslerTurizimSeferindekiYolcular
```

INDEX-1 ORNEK

```
CREATE INDEX name_surname

ON passenger_table (PassengerNameAndSurname);
```

INDEX-2 ORNEK

```
CREATE INDEX passengerID
ON passenger_table (PassengerID);
```

Tablolarımda sorgularımız için bir ufak değişikliğe gidiyoruz.

```
ALTER TABLE routes_of_agency
ADD prices int;

UPDATE routes_of_agency
SET prices = CRYPT_GEN_RANDOM(2) % 10000
```

FONKSIYON-1 ORNEK

Tüm acentaların saatlik ücretinin 50 tl olduğu rotaların fiyatlarının güncellenmesi... Fonksiyon kullanarak.

```
UPDATE routes_of_agency
SET prices =
dbo.definePricesRouteOfAgency(routes_of_agency.RouteOfTimeByHour,50)
```

Bir acentanın mevcut rotasındaki fiyatı belirleme fonksiyonu örneğin 8 saatlik bir kardeşler turizimin saatlik olarak 50 TL ücretinin 8 saatlik bir yolculukta 50*8 den 400TL lik bir ücretinin olduğunu belirtiriz.

```
CREATE FUNCTION definePricesRouteOfAgency(@perHourly int,@timeOfRoute int)
returns int
as
BEGIN
return @timeOfRoute * @perHourly
END
```

FONKSIYON-2 ORNEK

Seferde olan acentaların toplam o anki cirosunu hesaplayan bir fonksiyon yazınız.

```
CREATE FUNCTION howMuchGainIsThereExpeditionOfAgency(@agencyID
int,@pricePer int)

RETURNS INT

AS

BEGIN

DECLARE @passengerCount INT

DECLARE @totalHourly INT

SELECT @passengerCount = PassengerCount FROM expedition WHERE
ExpeditionAgencyID = @agencyID

SELECT @totalHourly = SUM(RouteOfTimeByHour) FROM expedition
ep,routes_of_agency roa WHERE ep.ExpeditionRouteID = roa.RouteID AND
ep.ExpeditionAgencyID = @agencyID

RETURN @passengerCount * @totalHourly * @pricePer

END

<-- Ornek Kardeşler turizim için onun id si olan biri yazıyoruz ve saatlik ücretini
giriyoruz...

SELECT dbo.howMuchGainIsThereExpeditionOfAgency(1,50)
```

PROCEDURE-1 ORNEK

Sefer ID sinin girildiği zaman o seferde olan yolcuları getiren sorguyu yazın...

```
CREATE PROCEDURE SelectAllPassengersOfExpedition @InputExpeditionID INT

AS

SELECT * FROM passenger_table pt WHERE @InputExpeditionID =
pt.PassengerOfExpeditionID

EXEC SelectAllPassengersOfExpedition @InputExpeditionID = 1;
```

PROCEDURE-2 ORNEK

Verilen acenta id ile seferde olan acentaların sefer bilgilerini getir...

```
CREATE PROCEDURE SelectAllPassengersOfExpedition @InputExpeditionID INT

AS
```

```
SELECT * FROM passenger_table pt WHERE @InputExpeditionID =
pt.PassengerOfExpeditionID

EXEC SelectAllPassengersOfExpedition @InputExpeditionID = 1;
```

CURSOR-1 ORNEK

Otobüsü 48 koltuktan az olan acentalarıyla birlikte plakalarını yazdıran CURSOR yazın

```
DECLARE @BusCapacity int, @BusAgency NVARCHAR(255),@Plate NVARCHAR(255)

DECLARE CursorBusCapacity CURSOR

FOR

SELECT b.MaxCapacityPassenger , ba.AgencyName , b.BusPlate FROM bus b,bus_agency
ba WHERE b.MaxCapacityPassenger < 48 AND b.BusOfAgencyID = ba.AgencyID

OPEN CursorBusCapacity

FETCH NEXT FROM CursorBusCapacity INTO @BusCapacity,@BusAgency,@Plate;

WHILE @@FETCH_STATUS = 0

BEGIN

PRINT 'Kapasite: '+CAST(@BusCapacity AS VARCHAR)+' Acentası: '+CAST(@BusAgency
AS VARCHAR) + ' Plakası: '+CAST(@Plate AS VARCHAR)

FETCH NEXT FROM CursorBusCapacity INTO @BusCapacity,@BusAgency,@Plate;

END

CLOSE CursorBusCapacity

DEALLOCATE CursorBusCapacity
```

CURSOR-2 ORNEK

Istanbuldan Anakaraya gidecek olan seferler düzenleyen uygun bir otobüs bul

Kosul-1 İstanbuldan ankaraya gidecek olan otobüsün 450.000 km geçmemesi gerekiyor.

Kosul-2 Kapasiteside 45 ten yukarı olması gerekiyor.

Kosul-3 Minimum Km Max Kapasiteye uygun bir otobüs bulunması gerekiyor

Bunu Cursor ile yapın...

```
DECLARE

@RoutaStart VARCHAR(100),
```

```
@RoutaEnd VARCHAR(100),
@RouteTime VARCHAR(100),
@AgencyName VARCHAR(100),
@BusCapacity INT,
@BusKm INT,
@Plate VARCHAR(30)
DECLARE CursorOptimumStandartForBus CURSOR
FOR
SELECT TOP 1
roa.RouteStartPointCity,
roa.RouteEndPointCity,
roa.RouteOfTimeByHour,
ba.AgencyName,
b.MaxCapacityPassenger,
b.KmOfBus,
b.BusPlate
FROM bus_agency ba, bus b,routes_of_agency roa
WHERE
roa.RouteStartPointCity = 'Istanbul' AND roa.RouteEndPointCity = 'Ankara' AND
b.MaxCapacityPassenger > 45 AND b.KmOfBus < 450000
ORDER BY b.KmOfBus ASC

OPEN CursorOptimumStandartForBus

FETCH NEXT FROM CursorOptimumStandartForBus INTO
```

```
@RoutaStart,
@RoutaEnd,
@RouteTime,
@AgencyName,
@BusCapacity,
@BusKm,
@Plate

WHILE @@FETCH_STATUS = 0
BEGIN
PRINT
'En optimum otobüs-> BAŞLANGIÇ ŞEHRİ: '+CAST(@RoutaStart AS VARCHAR)+
' Bitiş şehri: '+CAST(@RoutaEnd AS VARCHAR)+
' Rota Zamanı: '+CAST(@RouteTime AS VARCHAR)+
' Acenta Ismi: '+CAST(@AgencyName AS VARCHAR)+
' Otobüs Kapasitesi: '+CAST(@BusCapacity AS VARCHAR)+
' Otobüsün Km: '+CAST(@BusKm AS VARCHAR)+
' Otobüsün Plakası: '+CAST(@Plate AS VARCHAR)
FETCH NEXT FROM CursorOptimumStandartForBus INTO
@RoutaStart,
@RoutaEnd,
@RouteTime,
@AgencyName,
@BusCapacity,
@BusKm,
@Plate
```

```
END
CLOSE CursorOptimumStandartForBus
DEALLOCATE CursorOptimumStandartForBus
```

Tablolarımda sorgularımız için bir ufak değişikliğe gidiyoruz.

```
SEFERLER TABLOSUNDA BİR TANE COLUMN OLUŞTUR VE O COLUMN O
SEFERIN BİTİP BİTMEDİĞİ İLE İLGİLİ
DURUMU TUTSUN DAHA SONRA EĞER BİTTİYSE ONLARI TRUE BİTMEDİYSE FALSE
YAPICAKSIN SONRA BİTEN SEFERLERİ BAŞKA TABLOYA AKTARACAKSIN

ALTER TABLE expedition
ADD isGone BIT;
```

TRIGGER-1 ORNEK

Seferlerdeki seferin bitip bitmediğini güncellemeden önce end timeın kontrol ederek bugün ki değerleri geçip geçmediği kontrol ederek güncelle.

```
CREATE TRIGGER expeditionGoneTrigger ON expedition
AFTER UPDATE
AS
BEGIN
IF(exists(SELECT * FROM expedition e WHERE DATEDIFF(hour, GETDATE(),
e.EndTime) < 0))
BEGIN
PRINT 'TRIGGERED'
UPDATE expedition
```

```
SET isGone = 1
END
END
//show case
UPDATE expedition
SET isGone = 1
```

Tablolarımda sorgularımız için bir ufak değişikliğe gidiyoruz.

```
ALTER TABLE expedition
ADD isUpdatedOfBus BIT
```

TRIGGER-2 ORNEK

Otobüslerin total seferlerini güncellemeden önce o seferlerin bittiğini isGone parametresine göre kontrol edip güncelle

```
CREATE TRIGGER totalExpeditionOfBusUpdateTrıgger ON expedition
AFTER UPDATE
AS
BEGIN
IF(exists(SELECT * FROM expedition WHERE isGone = 1 AND isUpdatedOfBus = 0))
BEGIN
PRINT 'TRIGGERED'
UPDATE bus
SET TotalExpadition += 1
UPDATE expedition
SET isUpdatedOfBus = 1
```

```
END

END


UPDATE bus

SET TotalExpadition += 1
```