Veri Tabanı Yönetimi ve Modellemesi

HAFTA 9

Haftalık Ders Akışı

1. Veritabanı Kavramlarına Giriş
2. Veri Tabanı Türleri, İlişkisel Veri Tabanı Tasarımı
3. ER Diyagramları ve Normalizasyon
4. SQL Server Arayüzü, Veri Tabanı Nesneleri
5. T-SQL ve SQL Sorguları
6. İndeks ve View
7. **Stored Procedure ve Fonksiyonlar**
8. Ara Sınav
9. Tetikleyiciler
10. Transaction Kavramları ve Yedekleme
11. Kullanıcı Türleri ve Kullanıcı Yönetimi
12. No-SQL Veri Tabanları
13. No-SQL Veri Tabanları
14. Proje Sunumu
15. Proje Sunumları

Stored Procedure(SP)

- SP belirli bir işlevi, görevi yerine getirmek için yazılan SQL ifadeleri
- Sık kullanılan yapıların SP olarak tek bir defa yazılması ve kullanılmak istendiği yerde sadece çağrılması (Tekrar tekrar kullanılması)
- İlk çalıştığında derlenir, sonrasında sadece işletilir derleme ihtiyacına gerek kalmaz
- Server tarafında saklananır ve parametre alabilir
- Verinin saklanma biçiminin ve/veya tabloların tasarımının saklanması

Stored Procedure Türleri

- **Extended Stored Procedure:** Genellikle *.dll şeklinde prosedürlerdir.
- **CLR Stored Procedure:** SQL Server 2005 sonrasında CLR ortamında herhangi bir dili kullanarak kodlanan SP'lerdir.
- **System Stored Procedure:** Genellikle sp_ ön ekiyle başlarlar ve hepsi master veri tabanında tutulan SP'lerdir.
- **Kullanıcı Tanımlı SP:** Kullanıcıların tanımladığı SP'lerdir.

Stored Procedure

- Genel Yapısı
  - CREATE PROCEDURE/PROC prosedur_adı

    Parametre seçenekleri

    AS

    BEGIN

    T-SQL ifadeleri

    END
  - Çalıştırma Seçeneği:
    - EXEC prosedur_adı

Stored Procedure

```
CREATE PROC  sp_getAnimal

AS

BEGIN

 Select tbl_animal.*,info from tbl_animal

 INNER JOIN tbl_animalType

 ON tbl_animal.animalType_Id=tbl_animalType.id

END
```

Çalışma:

```
EXEC sp_getAnimal
```

Parametre Alan SP

```
CREATE PROCEDURE sp_animal_food_Schedule_deficent
  @dayNumber   int
AS
  BEGIN
            DECLARE @idInfo AS INT
            DECLARE @name AS NVARCHAR(50)
            DECLARE @dayinfo AS INT
            DECLARE @tmp TABLE
            (
                        id INT,
                        name NVARCHAR(50),
                        deficentdayCount INT
            )
            DECLARE MealControl Scroll CURSOR FOR
            SELECT  A.id, A.Name,COUNT(DISTINCT AFS.day)
            FROM tbl_Animal A LEFT JOIN
            tbl_Animal_Food_Schedule AFS ON A.id=AFS.animal_Id
            GROUP BY A.id,A.Name
            OPEN MealControl
            FETCH MealControl INTO @idInfo,@name,@dayinfo
            WHILE (@@Fetch_Status<>-1)
BEGIN
            IF(@dayinfo=@dayNumber)
                        INSERT INTO @tmp values(@idInfo,@name,0)
            ELSE
                        INSERT INTO @tmp
values(@idInfo,@name,(@dayNumber-@dayinfo))
            FETCH MealControl INTO @idInfo,@name,@dayinfo

End

            CLOSE MealControl
            DEALLOCATE MealControl
            SELECT * FROM                @tmp
END
```

Çalışma:

```
EXEC sp_animal_food_Schedule_deficent 7

EXEC sp_animal_food_Schedule_deficent @daynumber=7
```

Parametre Döndüren SP

```
CREATE PROCEDURE sp_animalCount
@sayi AS INT OUT
AS
BEGIN
        SELECT @sayi=COUNT(*) FROM tbl_Animal
END
```

Çalışma:

```
DECLARE @sayi AS INT
EXEC sp_animalCount @sayi OUT
SELECT @sayi
```

SP-INSERT

```
ALTER PROCEDURE sp_Add_Animal

@name AS NVARCHAR(50) ,

@arrivalDate AS DATETIME,

@birthDate AS DATETIME,

@animalType_Id AS INT

AS

BEGIN

INSERT  INTO tbl_Animal values(@name,@arrivalDate,@birthDate,@animalType_Id)

END
```

Çalışma:

```
EXEC sp_Add_Animal   @name='asd', @arrivalDate=NULL, @birthDate =NULL, @animalType_Id =11
```

SP-Transaction

```
ALTER PROCEDURE sp_Add_Animal
@name AS NVARCHAR(50) ,
@arrivalDate AS DATETIME,
@birthDate AS DATETIME,
@animalType_Id AS INT
AS
BEGIN

        BEGIN TRANSACTION

            INSERT  INTO tbl_Animal
values(@name,@arrivalDate,@birthDate,@animalType
_Id)

        IF (@@error <> 0)
        BEGIN
            SELECT 'ROLLBACK'
            ROLLBACK TRANSACTION
        END
    COMMIT TRANSACTION
END
```

Çalışma:

```
EXEC sp_Add_Animal   @name='asd',
@arrivalDate=NULL, @birthDate =NULL,
@animalType_Id =18
```

SP-Transaction

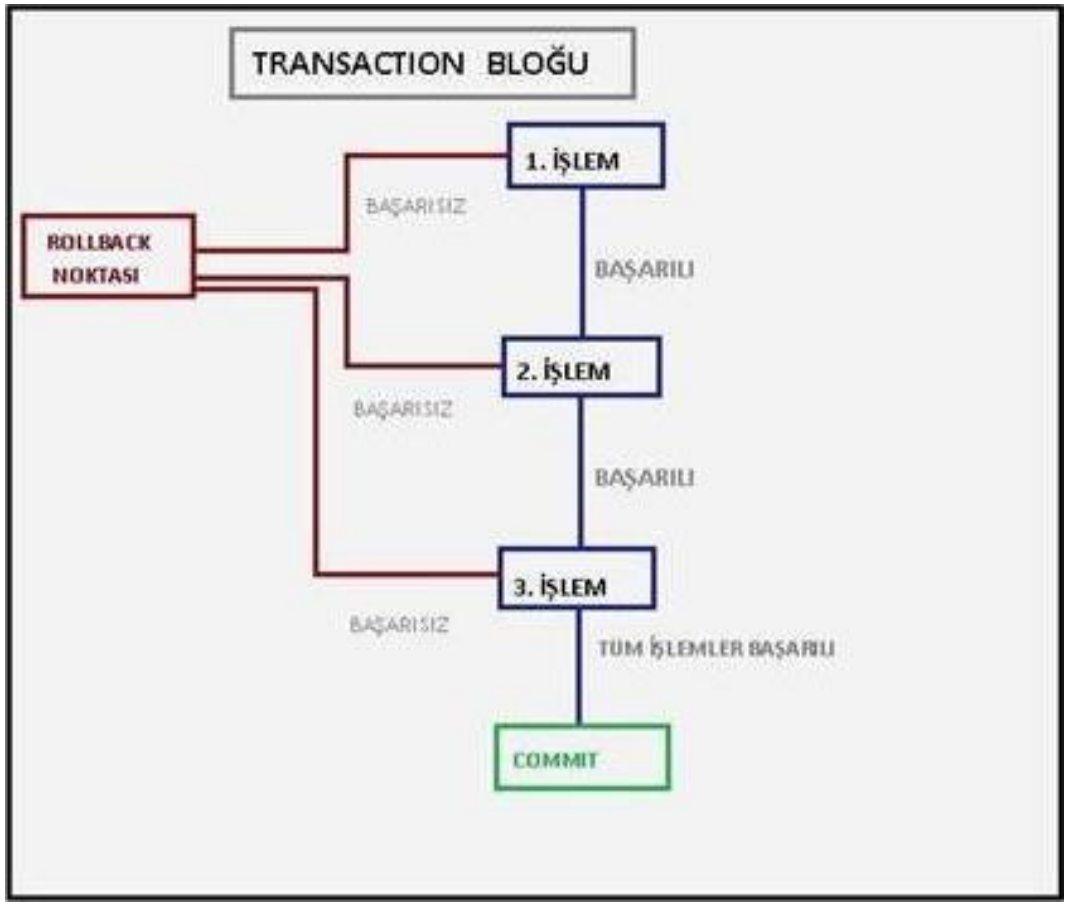
```
CREATE PROCEDURE sp_Add_Animal
@name AS NVARCHAR(50) ,
@arrivalDate AS DATETIME,
@birthDate AS DATETIME,
@animalType_Id AS INT,
@sonuc AS INT OUT
AS
BEGIN
        BEGIN TRANSACTION
        INSERT  INTO tbl_Animal
values(@name,@arrivalDate,@birthDate,@animalType_Id)
            IF (@@error <> 0)
            BEGIN
                SELECT 'ROLLBACK'
                SELECT @sonuc=0
                ROLLBACK TRANSACTION
            END
            SELECT @sonuc=1
        COMMIT TRANSACTION
END
```

Çalışma:

```
DECLARE @sonuc AS INT
EXEC sp_Add_Animal
@name='asd',@arrivalDate=NULL,@birthDate =NULL,@animalType_Id =11,
@sonuc=@sonuc out
SELECT @sonuc
```

Transaction

- Transaction en küçük işlem bloğu
- Transaction;
  - Ya bütün işlemleri gerçekleştirir ya da hiçbirini gerçekleştirmez.
- BEGIN ile başlar
- ROLLBACK : Tüm işlemleri geri alır
- COMMIT: Tüm işlemlerin başarılı şekilde yapıldığı düşünülür.
- ACID özelliği



ACID(**A**tomicity, **C**onsistency, **I**solation, **D**urability)

- **Atomicty (Bölünmezlik):** Bir transaction bloğu yarım kalamaz. Yarım kalan transaction bloğu veri tutarsızlığına neden olur. Ya tüm işlemler gerçekleştirilir, ya da transaction başlangıcına geri döner.
- **Consistency (Tutarlılık):** Transaction veri tutarlılığı sağlamalıdır. Yani bir transaction içerisinde güncelleme işlemi gerçekleştiyse ve ya kalan tüm işlemler de gerçekleşmeli ya da güncelle işlemi de geri alınmalıdır.
- **Isolation (İzolasyon):** Bir transaction tarafından gerçekleştirilen değişiklikler tamamlanmadan bir başka transaction tarafından görülememelsi gerekir. Yani her transaction ayrı ayrı işlenmelidir.
- **Durability (Dayanıklılık):** Transaction'lar veri üzerinde karmaşık işlemler gerçekleştirirken verinin bütününü güvence altına almalıdır. Bu sebeple transaction hatalara karşı dayanıklı olmalıdır.

USER DEFINED FUNCTIONS

- CLR yada T-SQL tabanlı olabilir
- Table-valued functions
- Scalar-valued functions
- Aggregate-valued functions
  - Min,Max,Count,Date...

Table-Valued UDF

```
CREATE FUNCTION udf_Animal()
RETURNS TABLE
AS RETURN
SELECT DISTINCT A.name,
        BeslenmeProgramiBilgisi=CASE
WHEN AFS.id IS NULL THEN 'YOK'
ELSE 'VAR'
END
FROM tbl_Animal A LEFT JOIN
tbl_Animal_Food_Schedule AFS ON A.id=AFS.animal_Id
```

Çalışma:

```
SELECT * FROM udf_Animal()
```

Table-Valued UDF

```
CREATE FUNCTION udf_Animal_Ver2()
RETURNS @tmp TABLE
(		name  NVARCHAR(50),
		ProgramBilgisi  NVARCHAR(50)
)
AS
BEGIN
   INSERT INTO @tmp
		SELECT DISTINCT A.name,
		BeslenmeProgramiBilgisi=CASE
		WHEN AFS.id IS NULL THEN 'YOK'
		ELSE 'VAR'
		END
		FROM tbl_Animal A LEFT JOIN
		tbl_Animal_Food_Schedule AFS ON A.id=AFS.animal_Id
		RETURN
END
```

Çalışma:

```
SELECT * FROM udf_Animal_Ver2()
```

Table-Valued UDF

```
CREATE FUNCTION udf_Animal_Ver3(
@animalType AS INT
)
RETURNS @tmp TABLE
(
name  NVARCHAR(50),
ProgramBilgisi  NVARCHAR(50)
)
AS
BEGIN
   INSERT INTO @tmp
            SELECT DISTINCT A.name,
                  BeslenmeProgramiBilgisi=CASE
WHEN AFS.id IS NULL THEN 'YOK'
ELSE 'VAR'
END
FROM tbl_Animal A LEFT JOIN
tbl_Animal_Food_Schedule AFS ON A.id=AFS.animal_Id
WHERE A.animalType_Id=@animalType
RETURN
END
```

Çalışma:

```
SELECT * FROM udf_Animal_ver3(3)
```

Scalar-valued functions

```
CREATE FUNCTION udf_Animal_Count(
)
RETURNS int
AS
BEGIN
        DECLARE @num AS INT
        SELECT @num=COUNT(*) FROM tbl_Animal
RETURN @num
END
```

Çalışma:

```
SELECT number=dbo.udf_Animal_Count()
```

Scalar-valued functions

```
CREATE FUNCTION udf_Animal_CountVer2(
@animalType AS INT
)
RETURNS INT AS
BEGIN
        DECLARE @num AS INT
        SELECT @num=COUNT(*) FROM tbl_Animal
        WHERE animalType_Id=@animalType
RETURN @num
END
```

Çalışma:

```
SELECT number=dbo.udf_Animal_CountVer2(3)
```