Veri Tabanı Yönetimi ve Modellemesi

HAFTA 11

Dr. Fatmana Şentürk

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. Geçici Tablolar, Kontrol Yapıları
8. Ara Sınav
9. Stored Procedure
10. Fonksiyonlar
11. **Tetikleyiciler(Trigers)**
12. **Yedekleme**
13. Kullanıcı Türleri ve Kullanıcı Yönetimi
14. No-SQL Veri Tabanları

Örnek

- XYZ isimli persian türündeki hayvan, bugün Ayşe Demir isimli veterinere muayene olmuş ve 1 doz dış parazit, 1 doz iç parazit aşısı yapılmıştır. Tüm bu işlemleri veritabanına ekleyen bir Store Procedure (SP) yazınız. Bu SP, parametre olarak, hayvanın adını, veterinerin adını, veterinerin soyadını ve «asiId-doz;asiId-doz» şeklinde aşı ve doz bilgisini almalıdır. (Aşı ve doz bilgilerini parçalamak için udf_getlist isimli user defined function'ı kullanabilirsiniz.)

Örnek çözümü:

```
CREATE PROC SP_addAsi
@hayvanAdi VARCHAR(30),@turAdi VARCHAR(30),
@veterinerAdi VARCHAR(30),@veterinerSoyAdi
VARCHAR(30),
@asi VARCHAR(500) AS BEGIN
DECLARE @HayvanId INT
DECLARE @veterinerId INT
SELECT @HayvanId=id FROM tbl_Animal WHERE
name=@hayvanAdi and animalTypeId = (SELECT id from
tbl_AnimalType
WHERE name=@turAdi)
SELECT @veterinerId =TS.id from tbl_Staff TS INNER
JOIN tbl_StaffRole TSR ON TS.staffRoleId=TSR.id
WHERE TS.name=@veterinerAdi AND TSR.name='Veteriner'
AND TS.surname=@veterinerSoyAdi
IF @HayvanId IS NOT NULL BEGIN
IF @veterinerId IS NOT NULL BEGIN
BEGIN TRANSACTION
INSERT INTO tbl_Staff_Animal
VALUES(@veterinerId,@HayvanId,GETDATE())
DECLARE @current NVARCHAR(50)
DECLARE ItemKontrol SCROLL CURSOR FOR select Item
from UDF_GetList(@asi,';')
OPEN ItemKontrol
FETCH ItemKontrol INTO @current
WHILE (@@FETCH_STATUS<>-1) BEGIN
INSERT INTO tbl_VaccineAnimalDosage VALUES(
(SELECT SUBSTRING(@current,1,CHARINDEX('-',@current)-
1)),
@HayvanId,GETDATE(),@veterinerId)
FETCH ItemKontrol INTO @current
END
CLOSE ItemKontrol
DEALLOCATE ItemKontrol
IF @@ERROR>0 ROLLBACK TRANSACTION
ELSE COMMIT TRANSACTION
END
END
END
```

Örnek Çalıştırma:

```
EXEC SP_addAsi
@hayvanAdi='XYZ',@turAdi='Persian',
@veterinerAdi='Ayşe',@veterinerSoyAdi='Demir'
,@asi='4-1;5-1'
```

Triggers

- Bir olay meydana geldiğinde veritabanının gerçekleştirmesi gereken bir eylem/eylemler
- Trigger'lar
  - Bütünlük kısıtlamaları
  - Veri değişikliklerini denetlemek
  - Karmaşık kısıtlamaları yönetmek
- Trigger işlemi:
  - BEFORE | AFTER | INSTEAD OF
  - INSERT | DELETE | UPDATE

Genel Yapısı

CREATE TRIGGER TriggerName

BEFORE | AFTER | INSTEAD OF

INSERT | DELETE | UPDATE [OF TriggerColumnList**]**

ON TableName

[**REFERENCING {OLD | NEW} AS {OldName | NewName}**

[**FOR EACH {ROW | STATEMENT}]**

[**WHEN Condition]**

<trigger action>

Trigger Event-Condition-Action (ECA)

- E*vent (or events):*
  - *INSERT, UPDATE yada DELETE işlemlerinin yapılabildiği tablolar üzerinde kuralların tetiklenmesidir.*
  - Before event yada after event şeklinde özelleşebilir.
- *Condition:*
  - *Yürütülmesi gerek action kısmının tanımlanmasıdır.*
  - *Condition opsiyonel olabilir, ancak koşullar doğru ise action çalıştırılabilir*
- *Action:*
  - *Trigger ifadesi verildiğinde ve doğru olarak çalıştırıldığında verilen T-SQL ifadesi çalıştırılabilir.*

Trigger

- Trigger iki seviyede olabilir
- Satır bazında (row level): Her bir satır etkilendiğinde işletilir
- İfade bazında (statement level): Birden fazla satır etkilendiğinde işletilir
- Ayrıca doğrudan (INSERT, UPDATE ve DELETE) ile değiştirilemeyen görünümleri değiştirmeyi sağlayan INSTEAD OF ifadesini triggerlar destekler
- INSTEAD OF triggerlar:
- Orjinal SQL ifadesi yerine başka bir trigger'ı tetikleyebilir

Örnek:İnsert ifadesinden sonra çalışan bir trigger yazılması

```
CREATE TRIGGER addStafTrigger
ON tbl_Staff
AFTER  INSERT
NOT FOR REPLICATION
AS
SELECT 'trigger executed'
```

Örnek:Delete işleminden önce kontrol işlemi yapan bir trigger yazılması

```
CREATE Trigger deleteStaffRole
 ON  tbl_StaffRole instead of delete As
Begin
   IF Exists (select staffRoleId from tbl_Staff where staffRoleId in
                    (Select id from deleted))
   BEGIN
RAISERROR('Once diger tablolardan veriler silinmeli',16,1)
  ROLLBACK
   END
END
```

Örnek:Tablo üzerinde değişiklik yapılmasını engelleyen bir trigger yazılması

```
CREATE TRIGGER tblSecurity

ON DATABASE FOR DROP_TABLE, ALTER_TABLE

AS BEGIN

PRINT 'Bu tablonun degistirilmesi admin tarafindan
engellenmistir'

ROLLBACK

END
```

Trigger Aktif-Pasif Durumu

- DISABLE
  - Alter Table tbl_name DISABLE TRIGGER trigger_name
  - DISABLE Trigger ALL ON ALL SERVER;
  - ALTER TABLE dbo.tbl_Yolcu DISABLE trigger deleteYolcuTrigger
- ENABLE
  - Alter Table tbl_name ENABLE TRIGGER trigger_name
  - ALTER TABLE dbo.tbl_Yolcu ENABLEtrigger deleteYolcuTrigger
- DROP (Silme)
  - Drop trigger trigger_name

Avantaj ve Dezavantaj

- Kod tekrarının kaldırılması
  - Log tablolarına verilerin otomatik olarak eklenmesi
- Değişikliklerin tek seferde uygulanması
  - İki ayrı veritabanı arasındaki bağlantı
- Güvenlik
- Bütünlük: Veri bütünlüğünü sağlamak için kontrollerin yapılması triggerlar aracılığı ile sağlanabilir.
- İşlem Gücü
- Client-Server Mimarisine uygun

Avantaj ve Dezavantaj

- Server’ın iş yükünün artması
- Triggerların bir birini tetiklemesi ve bunun öngörülemez olması
- Zamanlama
- Taşınabilirliği az: Birçok server triggerlar için kendi standardını kullanır
  - Örneğin: MsSql – before işlemi yerine instead of kullanılıyor

Yedekleme

- Back-up

```
BACKUP DATABASE DBAnimalClinic TO DISK='D:\DBAnimalClinic.bak'


RESTORE DATABASE DBAnimalClinic FROM DISK='D:\DBAnimalClinic.bak'
```

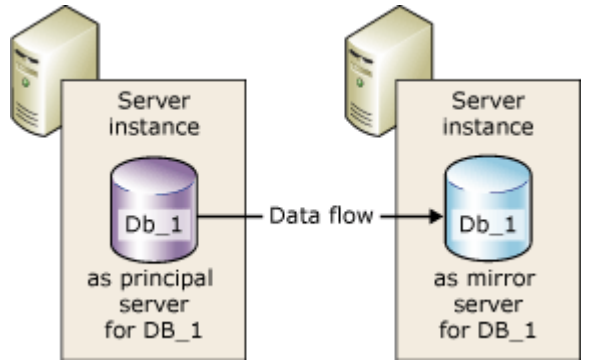
- Attach – Detach
  - Ldf-Mdf

Replication

- Farklı yerlerde üretilen verilerin bir biri ile bağlantılı şekilde çalışabilmesi
- MSSQL tarafından desteklenen türler:
  - Snapshot Replication: Belirli peryotlarla yedek server'a veri akışının sağlanması
  - Transactional Replication: Başlangıçta database'in bir snapshot'ı alınır. Sonrasında her transaction işleminden sonra yedek DB güncellenir
  - Merge Replication: Aynı anda iki DB üzerinde de değişiklik yapılmasını sağlar
  - Peer to Peer: Aynı anda birden fazla server'a eş zamanlı olarak değişiklik izni verir
  - Bidirectional: Her iki yönlü veri akışına izin verir.
  - Updatable Subscriptions: Bir server'ın yayınladığı veriye diğer serverlar abone olur. Yayın yapan server değişiklik yaptığı zaman, diğer server bu değişimi kendi sistemlerine uygular.

Mirroring

- DB server'ın her hangi bir problem karşısında çökmesi durumunda kullanılabilecek bir yöntem
- Bir veritabanının kullanılabilirliğini arttırır.
- Veri korumasını arttırır.
- Sistem güncellemeleri sırasında veritabanının kullanılabilirliğini iyileştirir



Job

- Belirli bir işin yapılması için gerekli tanımlamaların yapılabildiği SQL nesnesi
- Otomatik backup
- Herhangi bir hata durumunda e-mail gönderimi..vs

Backup Dosyası Oluşturma Kodu

```
DECLARE @path AS NVARCHAR(100)
SET @path='D:\DBAnimalClinic'+CONVERT(varchar,getdate(),104)+'.bak'
BACKUP DATABASE DBAnimalClinic TO DISK=@path
```

- **Job Oluşturma Kodu sonraki slaytta**

```
USE [msdb]
GO
BEGIN TRANSACTION
DECLARE @ReturnCode INT
SELECT @ReturnCode = 0
IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories
WHERE name=N'[Uncategorized (Local)]' AND
category_class=1)
BEGIN
EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB',
@type=N'LOCAL', @name=N'[Uncategorized (Local)]'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO
QuitWithRollback
END
DECLARE @jobId BINARY(16)
EXEC @ReturnCode =  msdb.dbo.sp_add_job
@job_name=N'Yedekleme',
@enabled=1, @notify_level_eventlog=0,
@notify_level_email=0, @notify_level_netsend=0,
@notify_level_page=0, @delete_level=0,
@description=N'No description available.',
@category_name=N'[Uncategorized (Local)]',
@owner_login_name=N'sa', @job_id = @jobId OUTPUT
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO
QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId,
@step_name=N'1',
@step_id=1, @cmdexec_success_code=0, @on_success_action=1,
@on_success_step_id=0, @on_fail_action=2,
@on_fail_step_id=0, @retry_attempts=0,
@retry_interval=0,
@os_run_priority=0, @subsystem=N'TSQL',
@command=N'DECLARE @path AS NVARCHAR(100)
SET
@path=''D:\DBAnimalClinic''+CONVERT(varchar,getdate(),104)
+''.bak''
BACKUP DATABASE DBAnimalClinic TO DISK=@path
',
@database_name=N'master',
@flags=0
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO
QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id =
@jobId, @start_step_id = 1
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO
QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobschedule
@job_id=@jobId, @name=N'Deneme',
@enabled=0, @freq_type=1, @freq_interval=0,
@freq_subday_type=0,  @freq_subday_interval=0,
@freq_relative_interval=0, @freq_recurrence_factor=0,
@active_start_date=20241223, @active_end_date=99991231,
@active_start_time=203700,  @active_end_time=235959,
@schedule_uid=N'0f24817b-2cdd-4719-85b8-a8592b319d10'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO
QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id =
@jobId, @server_name = N'(local)'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO
QuitWithRollback
COMMIT TRANSACTION
GOTO EndSave
QuitWithRollback:
    IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
GO
```