Veri Tabanı Yönetimi ve Modellemesi

HAFTA 13

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. **Arasınav**
8. Geçici Tablolar, Kontrol Yapıları
9. Stored Procedure
10. Fonksiyonlar, Örnek Çözümler
11. **Tetikleyiciler ve Yedekleme**
12. **Kullanıcı Türleri ve Kullanıcı Yönetimi**
13. No-SQL Veri Tabanları

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 gereken 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

Örnek:İnsert ifadesinden sonra çalışan bir trigger yazılması

```
CREATE TRIGGER addPersonelTrigger

ON tbl_Personel

AFTER  INSERT

NOT FOR REPLICATION

AS
  SELECT 'trigger executed'
```

Örnek:

Daha önce baktığı herhangi bir hasta varsa ilgili doktorun silinmesi engelleyen bir trigger yazınız.

Örnek:Delete işleminden önce kontrol işlemi yapan bir trigger yazılması

```
CREATE Trigger deletePersonelTrigger
 ON  tbl_Personel instead of delete As
Begin
  IF Exists (select PersonelId from tbl_HastaMuayene where PersonelId in (Select id from deleted)
            AND RandevuTarihi<=GETDATE())
  BEGIN
            RAISERROR('Once baktığı hasta vardır, doktor çalışma durumu pasif olarak değiştirilmiştir.',16,1)
            ROLLBACK
  END
 Update tbl_personel SET aktif=0 Where id in (Select id from deleted)
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_Personel DISABLE trigger deletePersonelTrigger
- ENABLE
  - Alter Table tbl_name ENABLE TRIGGER trigger_name
  - ALTER TABLE dbo.tbl_Personel ENABLEtrigger deletePersonelTrigger
- 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

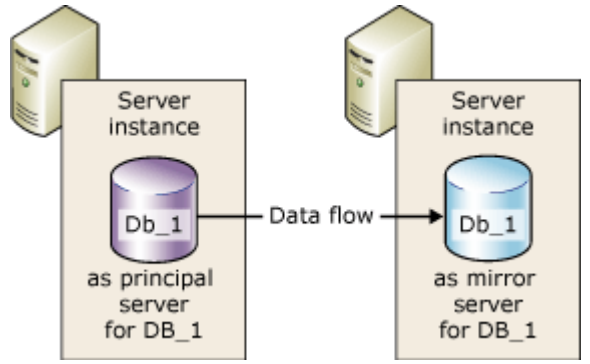
- Back-up
  - BACKUP DATABASE DBVirus TO DISK='D:\DBVirus.bak‘
  - RESTORE DATABASE DBVirus FROM DISK='D:\DBVirus.bak'
- Attach – Detach
  - Ldf-Mdf

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:\Ders_DBVirus'+CONVERT(varchar,getdate(),104)+'.bak'

BACKUP DATABASE DBHavayolu TO DISK=@path
```

- **Job Oluşturma Kodu sonraki slaytta**

```
USE [msdb]
GO
BEGIN TRANSACTION
DECLARE @ReturnCode INT
SELECT @ReturnCode = 0IF 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'backupDBVirus',
                    @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
/****** Object:  Step [1]    Script Date: 01/06/2021 10:30:28 ******/
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:\Ders_ DBVirus''+CONVERT(varchar,getdate(),104)+''.bak''
BACKUP DATABASE DBHavayolu TO DISK=@path',
                    @database_name=N' DBVirus',
                    @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'saat',
                    @enabled=1,
                    @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=20210106,
                    @active_end_date=99991231,
                    @active_start_time=102921,
                    @active_end_time=235959,
                    @schedule_uid=N'11e12fbe-c37b-47f3-89f6-
b621f6ea273c'
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
```

Kullanıcı Türleri

- Database Admin
- Database Designer
- Users

Database Admin

- Veritabanına erişim izinleri
- Veritabanı kullanıcılarının koordinasyonu
- Yazılım ve donanım kaynaklarının kullanımı
- Job, kullanıcı tanımı, otomatik yedekleme..vb gibi işlemler
- Güvenlik ihlalleri

Database Designer

- Saklanacak olan verilerin modellemesi
- Verilerin türlerinin belirlenmesi
- Kısıtların oluşturulması
- Şemaların tespiti

Users

- Raporları sorgulamak
- Verileri güncellemek ve oluşturmak
- Kullanıcı kategorisi :
- Sıradan son kullanıcılar zaman zaman veritabanına erişir, her seferinde farklı bilgilere ihtiyaç duyabilirler(üst düzey yöneticiler)
- Yerel veya parametrik kullanıcılar: Başlıca iş işlevlerini(ekleme, güncelleme, silme) gibi işlemler. Sürekli sorgulama (Stok takibi)
- Örnek:
  - Banka gişe görevlisi hesap bakiyelerini kontrol eder ve para çekme ve yatırma işlemlerini yapar.
  - Rezervasyon acenteleri veya havayolları, oteller ve araç kiralama şirketleri için müşteriler belirli bir istek için uygunluğunu kontrol eder ve rezervasyon yapar.

T-Sql

- Dışarıdan erişim için bir LOGIN oluşturuldu
- CREATE LOGIN userName WITH PASSWORD=‘password’
- Sql içerisinde işlem yapmak için bir USER oluşturuldu
- CREATE USER userName FOR LOGIN userName
- İzinlerin verilmesi

Kulanım İzinleri

- GRANT: Bir iznin verilmesini sağlar
  - GRANT CREATE TABLE TO Username
  - GRANT INSERT,UPDATE,DELETE TO Username
  - GRANT SELECT ON tblName TO Username
- WITH GRANT: Kullanıcı kendisinde bulunan yetkileri başka kullanıcılara verebilir.
  - GRANT SELECT,INSERT ON tblName TO Username WITH GRANT OPTION
- DENY: Kullanıcının yetkilerinin geri alınmasını sağlar.
  - DENY INSERT, SELECT ON tblName TO Username
- REVOKE: GRANT ile değiştirdiğimiz hakları eski haline döndürmek için kullanılır. Bir nesneyi oluşturan kullanıcının REVOKE ile nesne üzerindeki yetkilendirme ve kullanma hakkı yok edilemez.
  - REVOKE ALL ON REGION TO Username

Arayüz aracılığı ile Kullanıcı tanımlama işlemleri