ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM4522 - Ağ Tabanlı Paralel Dağıtım SistemleriGeliştirme

Yasin Çelik

21290412

Github Linki - https://github.com/YasinCelik01/SQL
Video Linki Youtube – https://youtu.be/Gu6jRHtIzXQ
Video Linki Youtube – https://youtu.be/d_6qjWygkeg
Video Linki Youtube – https://youtu.be/wP55oQe3UBE

1. VERİTABANI YEDEKLEME VE FELAKETTEN KURTARMA PLANI

1.1. Giriş

Bu bölümde, SQL Server ortamında çalışan veritabanlarının sürdürülebilirliğini sağlamak amacıyla yedekleme stratejileri ve felaketten kurtarma planları açıklanmaktadır. Amaç; veri kaybını önlemek, sistem sürekliliğini sağlamak ve olası felaket senaryolarında hızlıca kurtarma yapabilmektir.

1.2. Tam, Artık ve Fark Yedeklemeleri

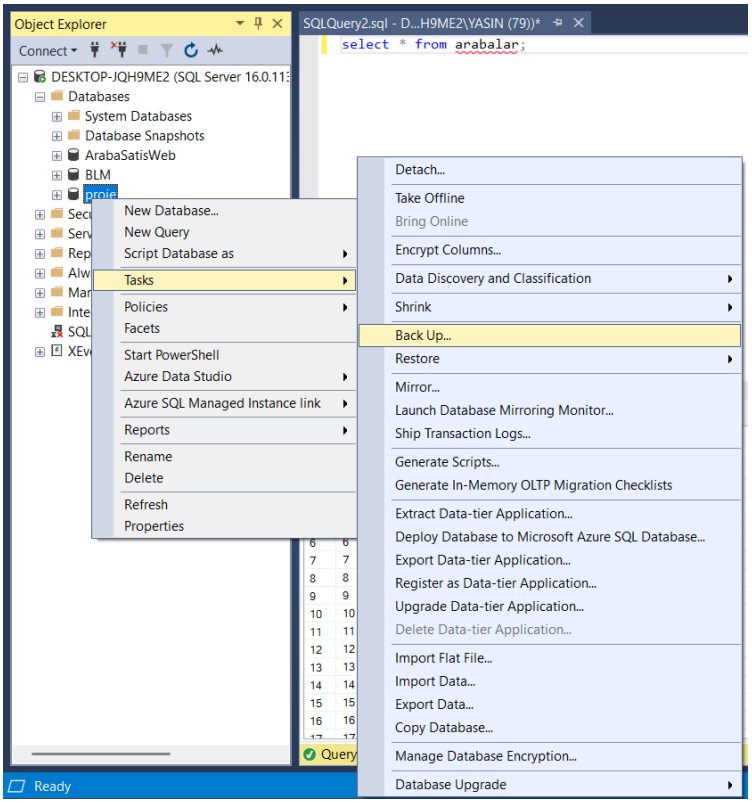
SQL Server'da veri bütünlüğünü sağlamak için aşağıdaki yedekleme türleri kullanılabilir. Tüm yedekleme işlemi hem arayüz ile hem de komut satırı aracılığı ile yapılabilir. Her bir yedekleme türü için yedekleme işlemleri adımı adım aşağıda gösterilmektedir:

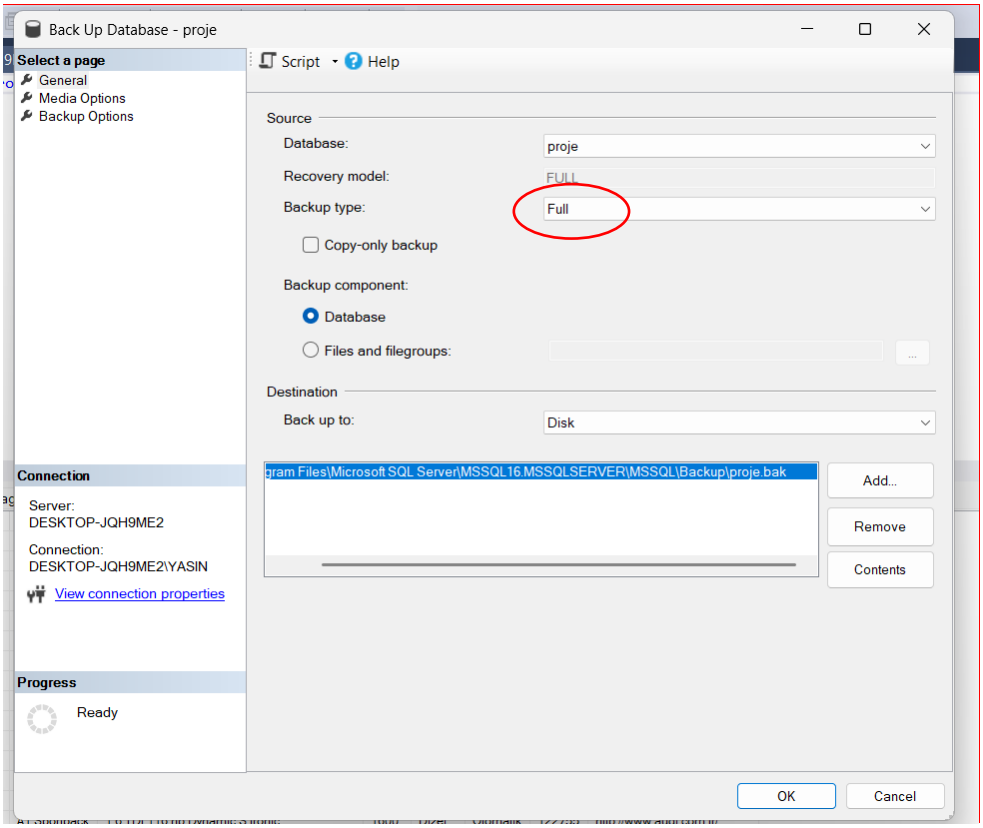
1.2.1 Tam (Full) Yedekleme: Yedekleme işlemi için yapılandırılan verilerin bütün bir kopyasının alınmasıdır.

- **Komut Satırı ile Tam Yedekleme**

```
BACKUP DATABASE YourDatabase
TO DISK = 'C:\Backups\YourDatabase_Full.bak'
WITH INIT;
```

- **Arayüz ile Tam Yedekleme**



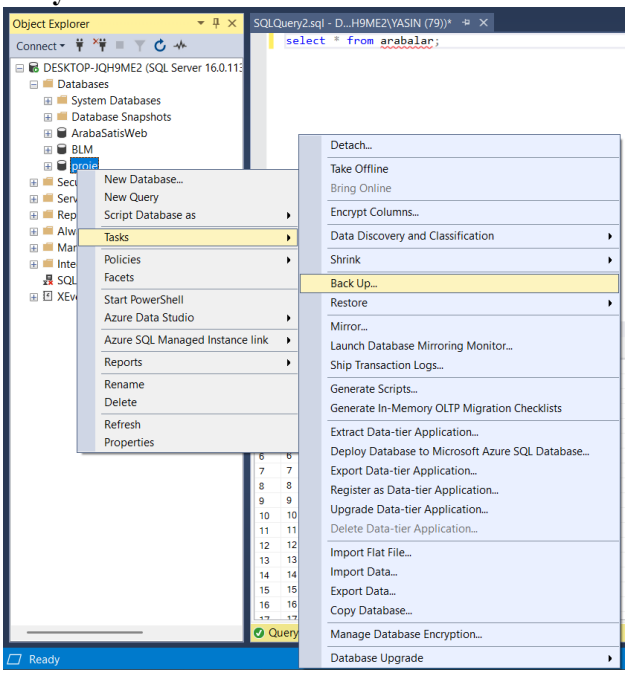


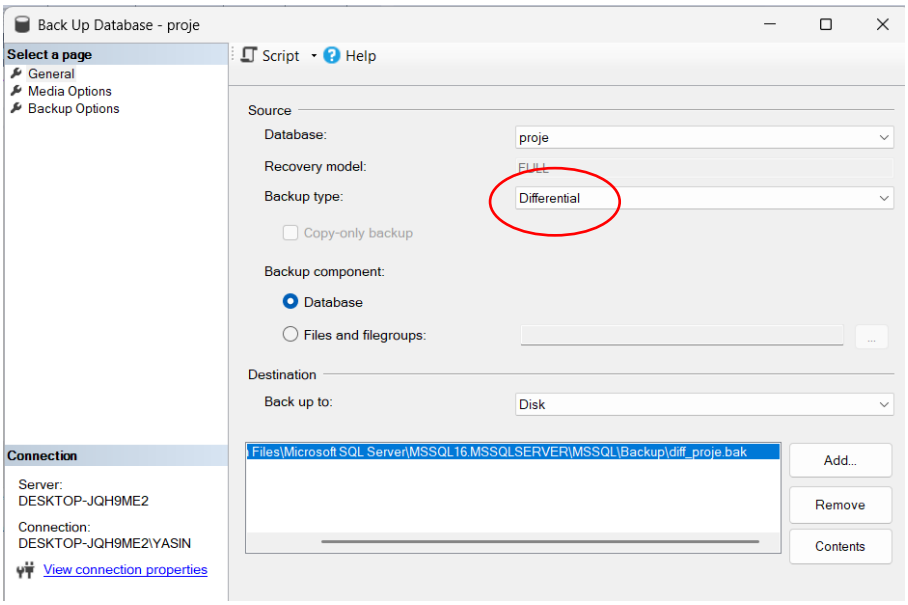
1.2.2 Fark (Differential) Yedekleme: En son yapılan tam yedeklemeden sonra değişen verileri yedekler.

➢ **Komut Satırı ile Fark Yedekleme**

```
BACKUP DATABASE YourDatabaseTO
DISK = 'C:\Backups\YourDatabase_Diff.bak'
WITH DIFFERENTIAL;
```

➢ **Arayüz ile Fark Yedekleme**



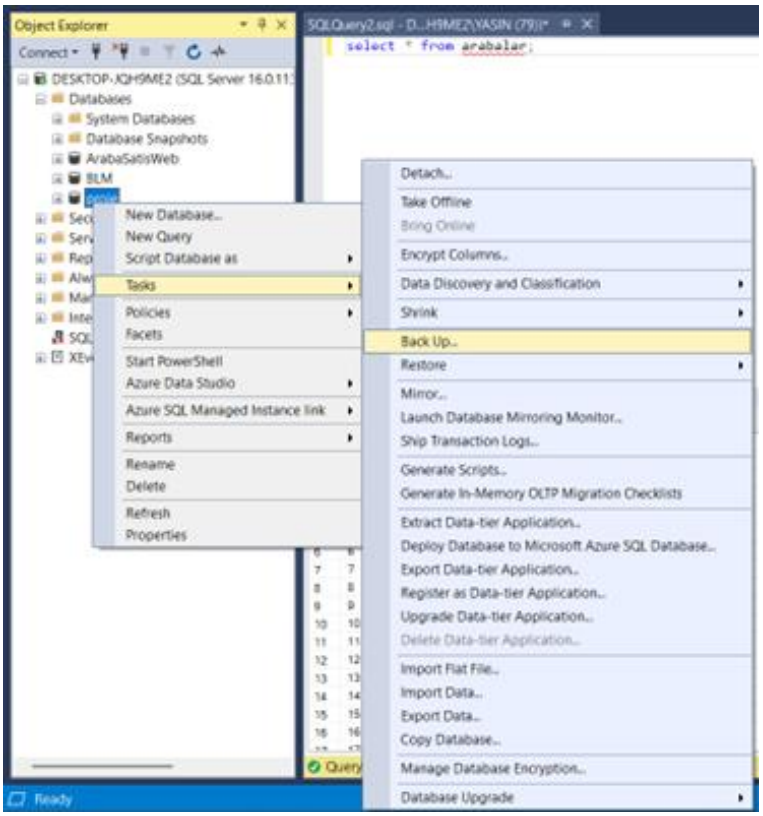


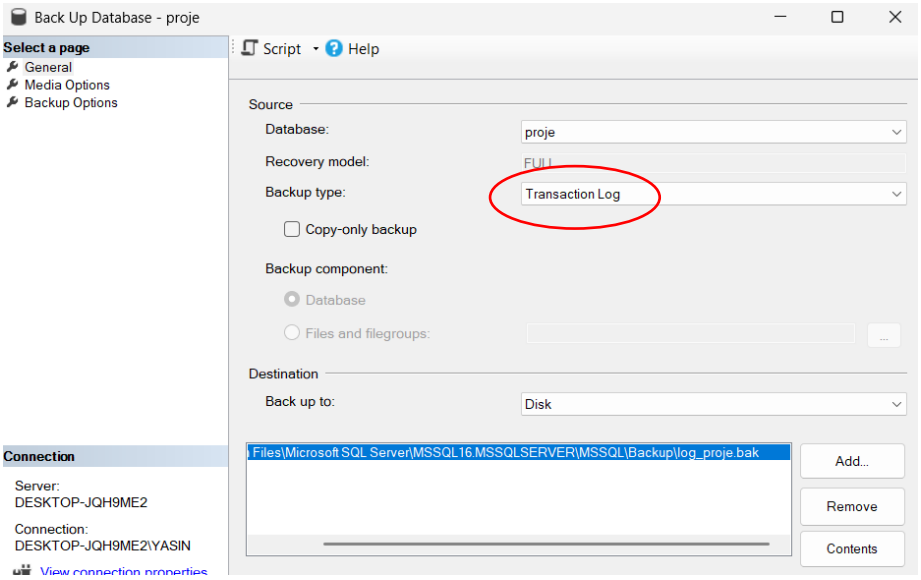
1.2.3 Artık (Transaction Log) Yedekleme: Daha önce yapılan yedeklemeden sonraki değişmiş olan verileri yedekler. Transaction log dosyasını yedekler. **Sadece Full Recovery Mode** altında çalışır.

- **Komut Satırı ile Artık Yedekleme**

```
BACKUP LOG YourDatabase
TO DISK = 'C:\Backups\YourDatabase_Log.bak;
```

- **Arayüz ile Artık Yedekleme**





1.3.Zamanlayıcılarla Otomatik Yedekleme

SQL Server Agent kullanılarak örnek bir yedekleme işlemi aşağıdaki gibi planlanabilir.

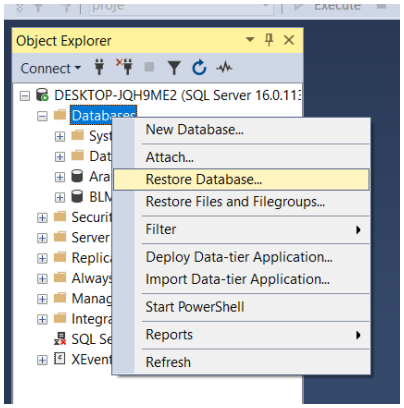
Yedekleme Türü	Sıklık	Zaman
Tam Yedek	Haftalık	Her Pazar 02:00
Fark Yedek	Günlük	Her gün 01:00
Transaction Log	15 Dakikada	00:00 – 23:45 arası

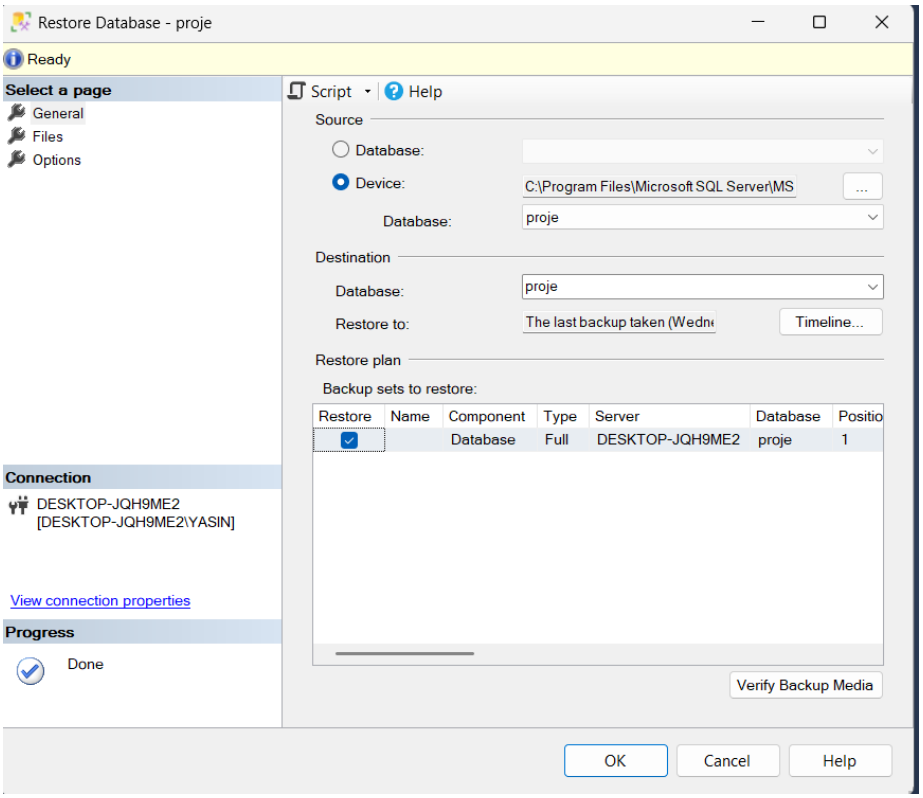
SQL Serverda arayüz kullanarak veya PowerShell betikleri ile bu işler otomatize edilebilir. (Nasıl yapıldığını ilerki projede anlattım)

1.4.Kaza ile Silinen Verilerin Geri Getirilmesi

Veritabanındaki bir tablo yanlışlıkla silinirse daha önce aldığımız yedekleme dosyaları ile aşağıdaki adımları izleyerek verileri geri kurtarabiliriz.

- En son tam yedek + ilgili transaction log dosyaları alınır.
- Point-in-time restore işlemi ile tablo silinmeden önceki duruma veritabanı geri döndürülür.
- Kurtarılan veri, mevcut veriyle karşılaştırılıp merge edilir.





1.5. Test Yedekleme Senaryoları

Yedekleme sisteminin güvenilirliği aşağıdaki testlerle doğrulanabilir:

- **Aylık Restore Testleri**: Her ayın sonunda, alınan yedekler test ortamına restore edilerek geçerliliği kontrol edilir.
- **Checksum ve Verify Option**: Yedekleme sırasında SQL Server tarafından yedek bütünlüğü otomatik olarak kontrol edilir.
- **Simülasyonlar**: Rastgele oluşturulan senaryolarda veri kaybı ve kurtarma süreleri ölçülerek sistemin etkinliği izlenir.

2. VERİTABANI GÜVENLİĞİ VE ERİŞİM KONTROLÜ

2.1. Giriş

Bu bölümde, SQL Server veritabanlarının yetkisiz erişimlere ve siber tehditlere karşı korunması amacıyla alınan güvenlik önlemleri detaylandırılmaktadır. Ana hedef; veri bütünlüğünü, gizliliğini ve erişilebilirliğini sağlamaktır.

2.2.SQL Server Authentication ve Windows Authentication

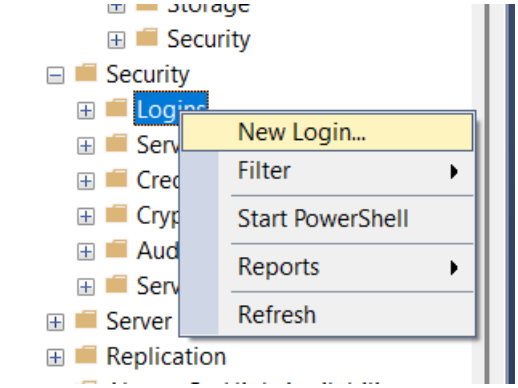
SQL Server'da iki temel kimlik doğrulama yöntemi kullanılmaktadır:

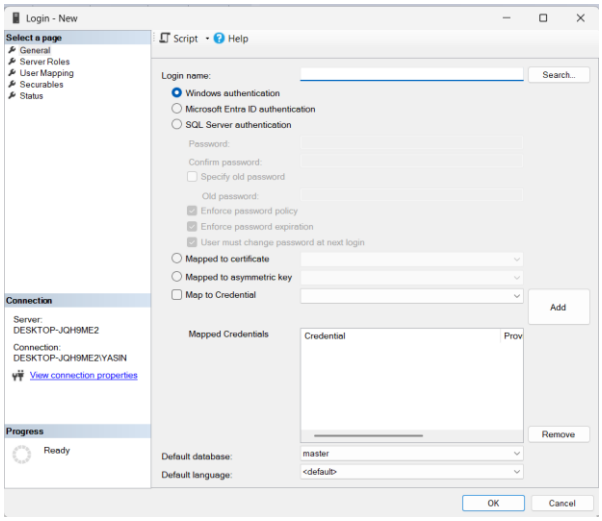
- **Windows Authentication**

  Aşağıdaki betik ile kullanıcı oluşturulabilir:

```
-- Windows Authentication için
CREATE LOGIN [DOMAIN\Username] FROM WINDOWS;
```

  - Active Directory ile entegre çalışır.
  - Kullanıcı kimliği merkezi olarak yönetilir.
  - Güvenli ve tercih edilen yöntemdir.
  - Parola politikaları doğrudan Windows güvenlik ilkelerine bağlıdır.



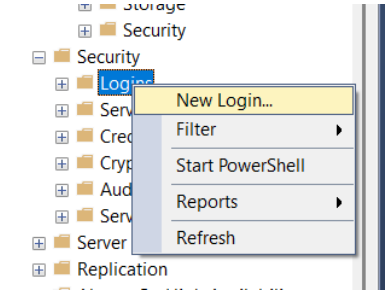


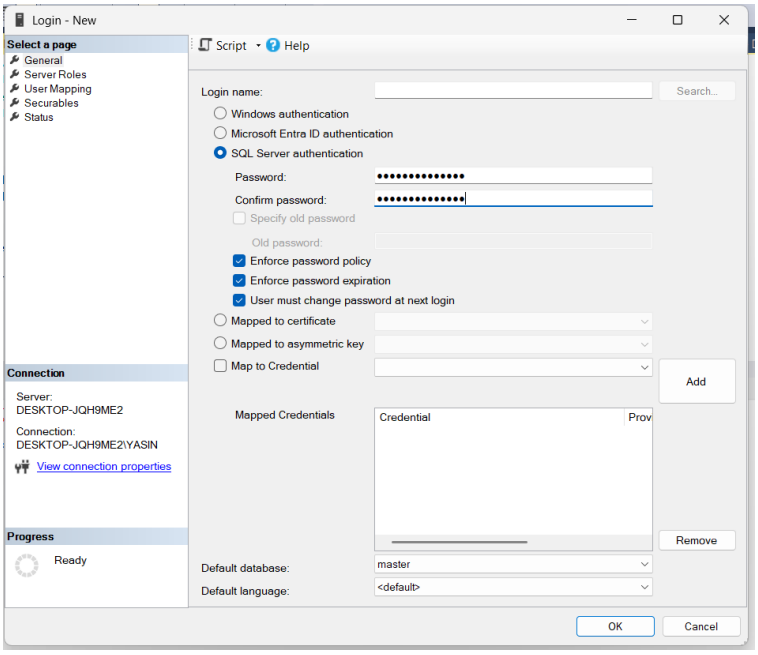
- **SQL Server Authentication**

  Aşağıdaki betik ile kullanıcı oluşturulabilir:

```
-- SQL Server Authentication ile kullanıcı
CREATE LOGIN userTest WITH PASSWORD = 'StrongPass123!';
```

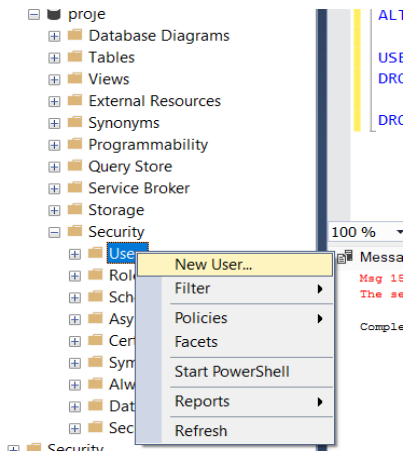
- Kullanıcı adı ve parola SQL Server üzerinde tanımlanır.
- Uzak bağlantılar ve harici uygulamalar için esneklik sağlar.
- Parola karma algoritması ile saklanır.
- Kullanıcılar rollerle sınırlandırılır (db_datareader, db_datawriter, db_owner vb.).

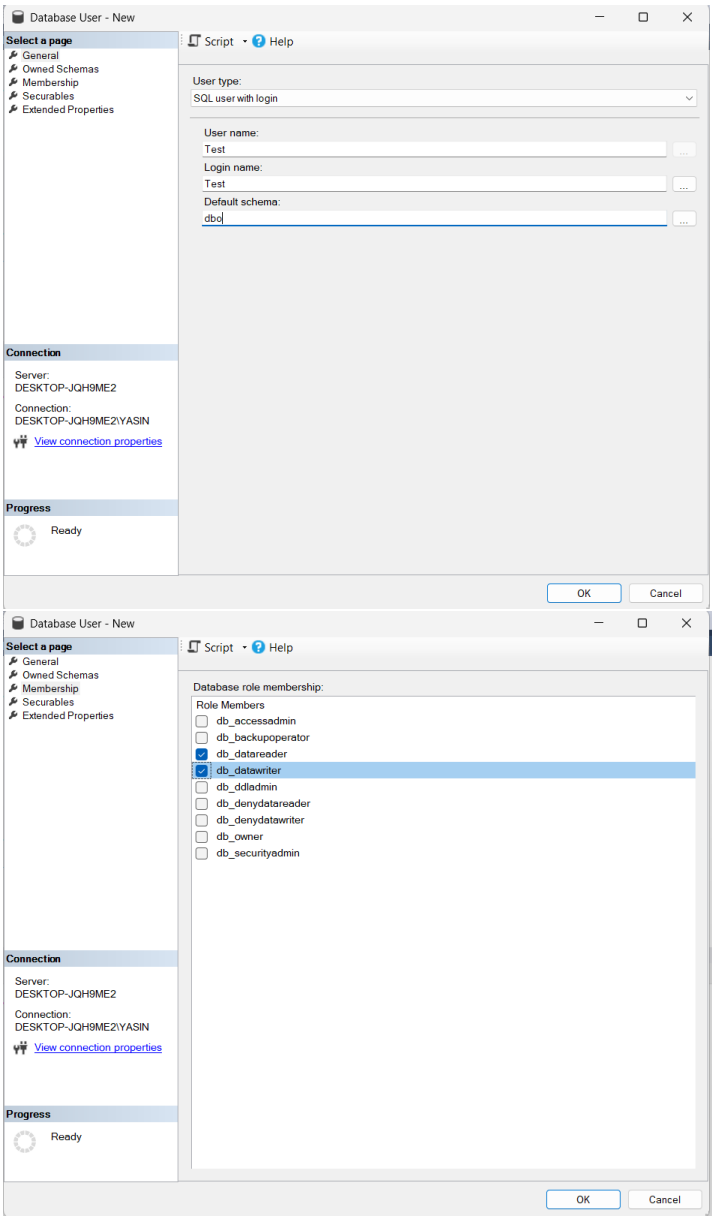




Aşağıdaki betik ile kullanıcılara yetkilendirme verilebilir:

```
USE YourDatabase;
CREATE USER userTest FOR LOGIN userTest;
ALTER ROLE db_datareader ADD MEMBER userTest;
ALTER ROLE db_datawriter ADD MEMBER userTest;
```





2.3. Veri Şifreleme

2.3.1 Transparent Data Encryption (TDE)

- **TDE**, veritabanı dosyalarını (MDF, LDF) şifreler.
- Disk çalınması veya yedeklerin ele geçirilmesi gibi durumlarda verilerin okunması engellenir.
- Uygulamalar tarafında herhangi bir değişiklik yapılmasına gerek kalmaz.

TDE Uygulama Adımları:

1. **Master Key** oluşturulur.

➢
```
USE master;
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'StrongKeyPass!';
```

2. **Certificate** üretilir ve güvenli bir şekilde saklanır.

➢
```
CREATE CERTIFICATE TDECert WITH SUBJECT = 'TDE Certificate';
```

3. **Database Encryption Key** tanımlanır.

```
USE YourDatabase;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE TDECert;
```

4. Şifreleme aktif hale getirilir:

```
ALTER DATABASE YourDatabase SET ENCRYPTION ON;
```

2.4.SQL Injection Testleri

- Parametreli sorgular (prepared statements) kullanılır.

```
SqlCommand cmd = new SqlCommand("SELECT * FROM Users WHERE
Username = @username", conn);
cmd.Parameters.AddWithValue("@username", input);
```

- Kullanıcı girişleri için **girdi doğrulama (input validation)** uygulanır.
- **Uygulama katmanında** whitelist/blacklist kontrolleri yapılır.
- Yalnızca gerekli haklara sahip kullanıcılar sorgu çalıştırabilir.

2.5.Audit Logları

- Kullanıcı aktiviteleri (login, sorgu çalıştırma, tablo erişimi) izlenebilir.
- Uyum gereklilikleri için kritik işlemler günlüklenir (GDPR, KVKK, ISO 27001).
- Audit log'ları dosya olarak veya Windows Application Log üzerinden saklanabilir.

Audit Oluşturma Örneği:

```
CREATE SERVER AUDIT [LoginAudit]
TO FILE (FILEPATH = 'C:\AuditLogs\');
ENABLE;

CREATE SERVER AUDIT SPECIFICATION [LoginAuditSpec]
FOR SERVER AUDIT [LoginAudit]
ADD (FAILED_LOGIN_GROUP),
ADD (SUCCESSFUL_LOGIN_GROUP);
ENABLE;
```

- Loglar düzenli olarak sistem yöneticileri tarafından kontrol edilir.
- Belirli eşiklerin aşılması durumunda e-posta veya SMS ile uyarı sistemi aktiftir.

3. VERİTABANI YÜKSELTME VE SÜRÜM YÖNETİMİ

3.1. Giriş

Kurumsal bilgi sistemlerinin sürdürülebilirliği açısından veritabanlarının güncel tutulması kritik bir gerekliliktir. Bu bölüm, mevcut SQL Server veritabanlarının daha yeni bir sürüme yükseltilmesi, sürüm yönetimi süreçlerinin planlanması ve yükseltme sonrası test/geri dönüş senaryolarını kapsamaktadır.

3.2. Veritabanı Yükseltme Planı

- **Hazırlık Aşaması**
  - **Mevcut sürüm ve hedef sürüm belirleme:**
    Örnek: SQL Server 2016 → SQL Server 2022
  - **Uyumluluk analizi:**
    SQL Server Data Migration Assistant (DMA) aracı kullanılarak mevcut veritabanı yapısının, fonksiyonların ve stored procedure'lerin yeni sürüme uyumu analiz edilir.
  - **Yedekleme işlemi:**
    Tüm veritabanları için tam ve log yedekleri alınır.
  - **Performans izleme:**
    Yükseltme öncesi performans metrikleri kaydedilerek, yükseltme sonrası karşılaştırmalar yapılır.

- **Yükseltme Stratejileri**
  - **Yerinde Yükseltme (In-Place Upgrade):**
    Mevcut SQL Server örneği doğrudan yeni sürüme yükseltilir. Kolaydır fakat geri dönüş zordur.
  - **Yan Yana Yükseltme (Side-by-Side Upgrade):**
    Yeni bir SQL Server örneği kurulur. Veritabanı bu ortama aktarılır. Daha güvenlidir, test ve geri dönüş kolaydır.

- **Veri Taşıma Yöntemleri**
  - **Yedekten Geri Yükleme (Restore from Backup)**
  - **Detach / Attach yöntemi**
  - **Import/Export Wizard**
  - **Database Migration Tool (DMA veya SSMS ile)**

3.3. Sürüm Yönetimi

Veritabanı üzerinde yapılan yapısal değişikliklerin (DDL işlemleri) takip edilmesi, denetlenmesi ve gerekirse geri alınabilmesi için sürüm yönetimi süreci uygulanır.

- **DDL Trigger Kullanımı**

  DDL trigger'lar sayesinde aşağıdaki işlemler izlenebilir:

  - **CREATE, ALTER, DROP işlemleri**
  - **Yeni tablo/sütun ekleme, şema değişikliği**

Örnek DDL Trigger:

```
CREATE TRIGGER trg_DDL_Log
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
BEGIN
    INSERT INTO DDL_AuditLog (EventType, EventData, LoginName,
EventTime)
    SELECT
        EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]',
'nvarchar(100)'),

EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand/CommandText)[1
]', 'nvarchar(max)'),
        SYSTEM_USER,
        GETDATE();
END;
```

- **Sürüm Numaralandırması**

  Her yapısal değişiklik için versiyon numarası belirlenir.

  Örnek:

Versiyon	Açıklama	Tarih
v1.0	Başlangıç şeması	2025-01-10
v1.1	Yeni `Kullanicilar` tablosu eklendi	2025-02-05
v1.2	`Siparisler` tablosuna `durum` sütunu eklendi	2025-03-20

3.4. Test ve Geri Dönüş Planı

- **Yükseltme Sonrası Test Süreci**

  - **Birim testleri:** Stored procedure, fonksiyonlar, tetikleyiciler test edilir.
  - **Fonksiyonel testler:** Uygulama ile veritabanı arasındaki tüm işlemler (CRUD operasyonları) gözden geçirilir.
  - **Performans testleri:** Karşılaştırmalı sorgu süreleri ve işlem yükü analiz edilir.
  - **Log denetimi:** Hatalar ve güvenlik ile ilgili loglar kontrol edilir.

- **Geri Dönüş Planı**
    - **Yedekten geri yükleme:**
      Yükseltme başarısız olursa eski sürüme dönüş için tam yedekler kullanılır.
    - **Paralel çalışma:**
      Side-by-side yükseltmelerde eski sistem geçici süreyle aktif kalır.
    - **Rollback script'leri:**
      Şema değişikliklerinin geri alınması için özel scriptler hazırlanır.