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
Video Linki Youtube - https://youtu.be/3SpssOP-53w
Video Linki Youtube - https://youtu.be/G8C7Kixd6uE
Video Linki Youtube - https://youtu.be/vQCGlczwCuk

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

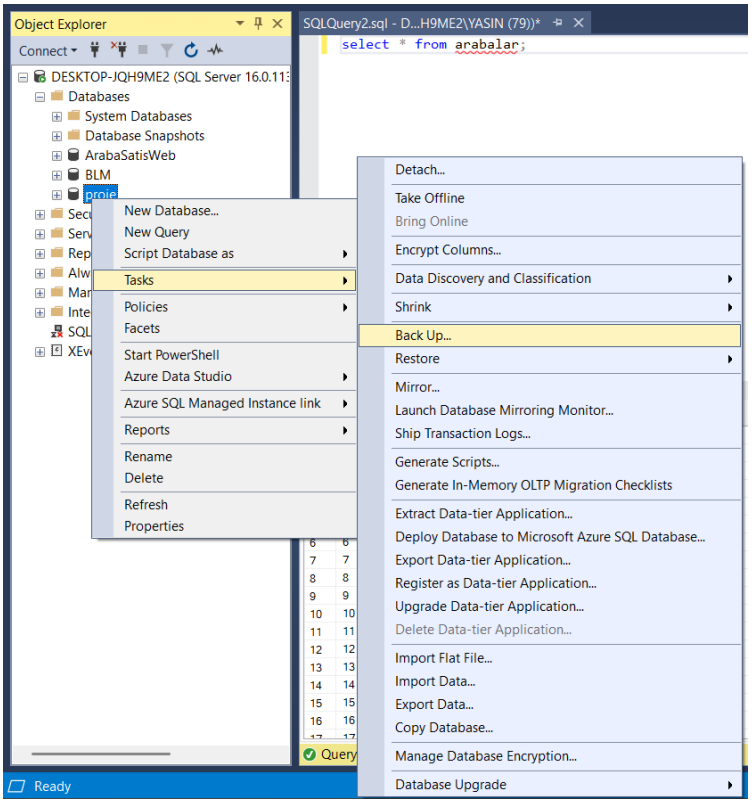
SQL Server'da veri bütünlüğünü sağlamak için aşağıdaki yedekleme türleri kullanılabilir. Tüm yedekleme işlemi hem arayüz ile hem de komut satırı aracılığı ile yapılabilir. Her bir yedekleme türü için yedekleme işlemleri adımı adım aşağıda gösterilmektedir:

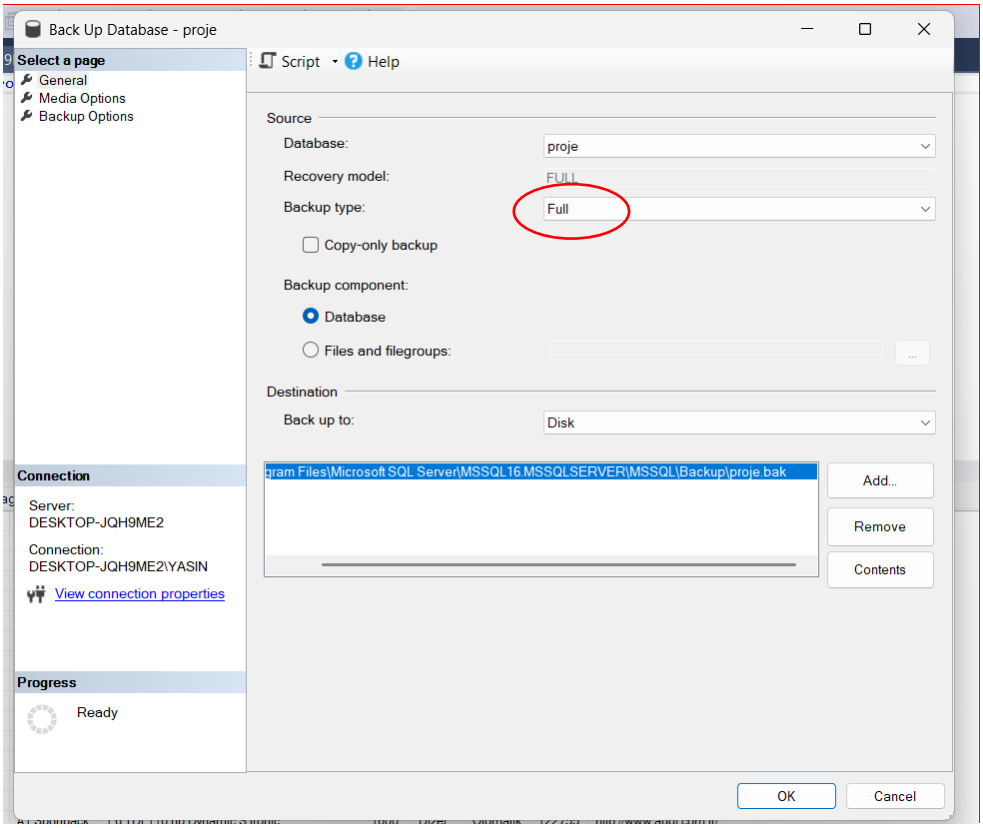
1.2.1 Tam (Full) Yedekleme: Yedekleme işlemi için yapılandırılan verilerin bütün bir kopyasının alınmasıdır.

- **Komut Satırı ile Tam Yedekleme**

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

- **Arayüz ile Tam Yedekleme**



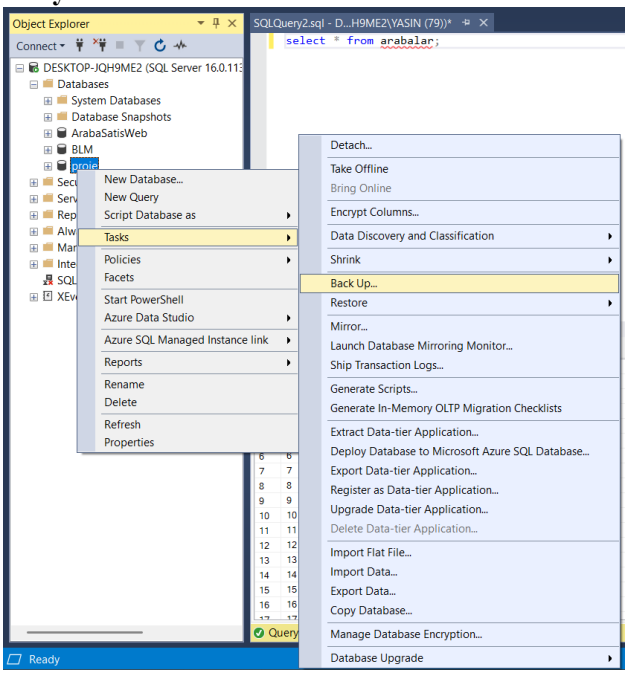


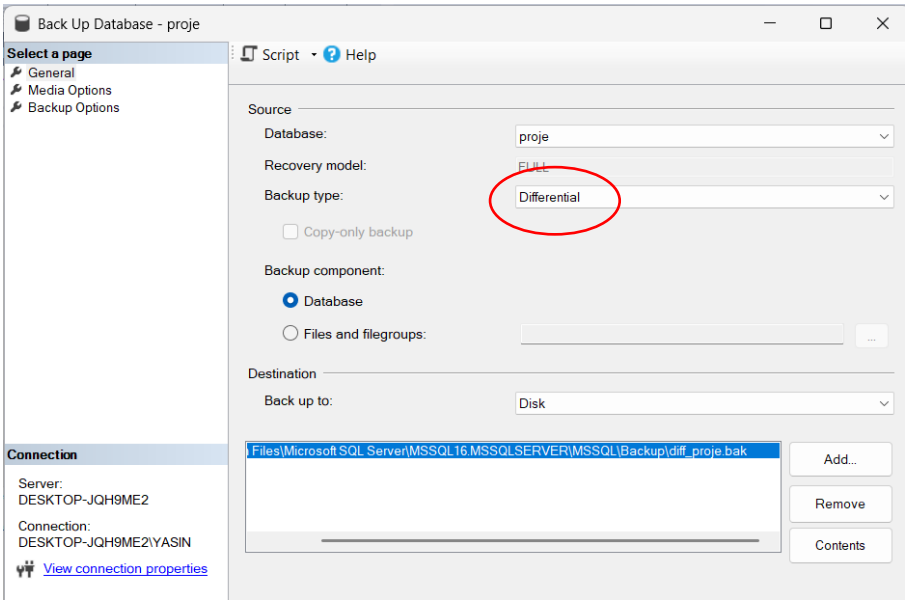
1.2.2 Fark (Differential) Yedekleme: En son yapılan tam yedeklemeden sonra değişen verileri yedekler.

➢ **Komut Satırı ile Fark Yedekleme**

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

➢ **Arayüz ile Fark Yedekleme**



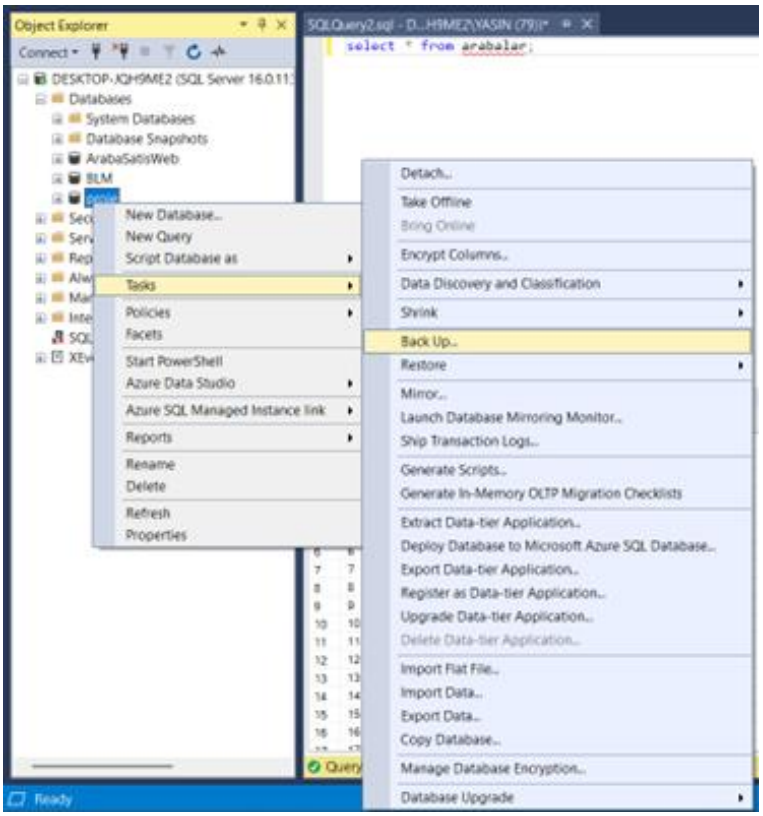


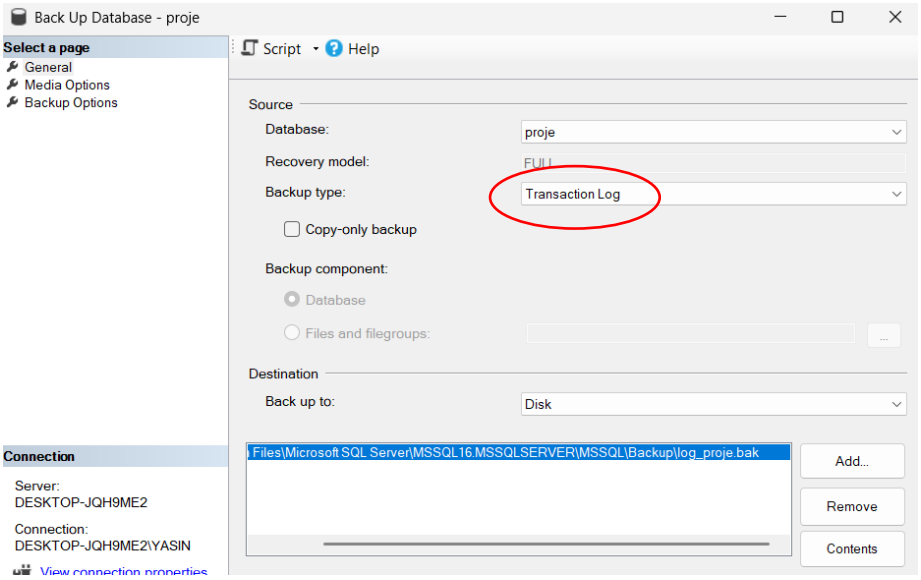
1.2.3 Artık (Transaction Log) Yedekleme: Daha önce yapılan yedeklemeden sonraki değişmiş olan verileri yedekler. Transaction log dosyasını yedekler. **Sadece Full Recovery Mode** altında çalışır.

- **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.

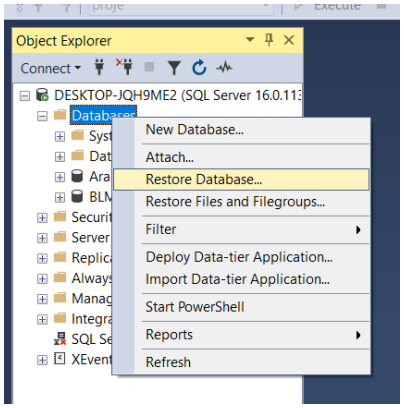
Yedekleme Türü	Sıklık	Zaman
Tam Yedek	Haftalık	Her Pazar 02:00
Fark Yedek	Günlük	Her gün 01:00
Transaction Log	15 Dakikada	00:00 – 23:45 arası

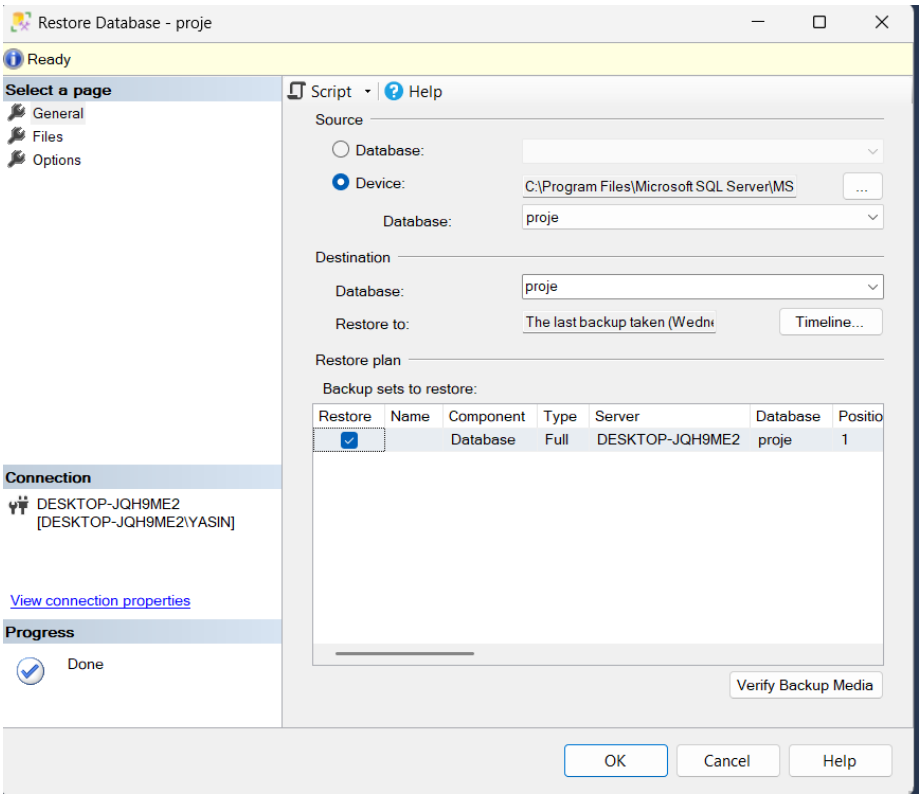
SQL Serverda arayüz kullanarak veya PowerShell betikleri ile bu işler otomatize edilebilir. (Nasıl yapıldığını ilerki projede anlattım)

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

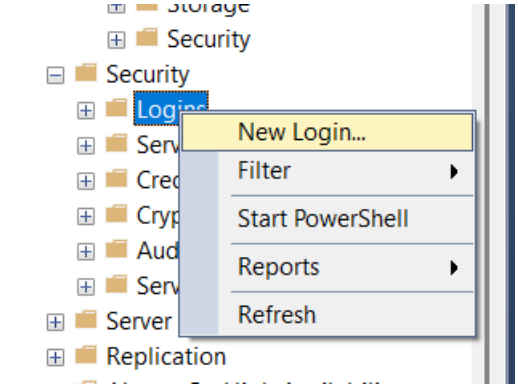
SQL Server'da iki temel kimlik doğrulama yöntemi kullanılmaktadır:

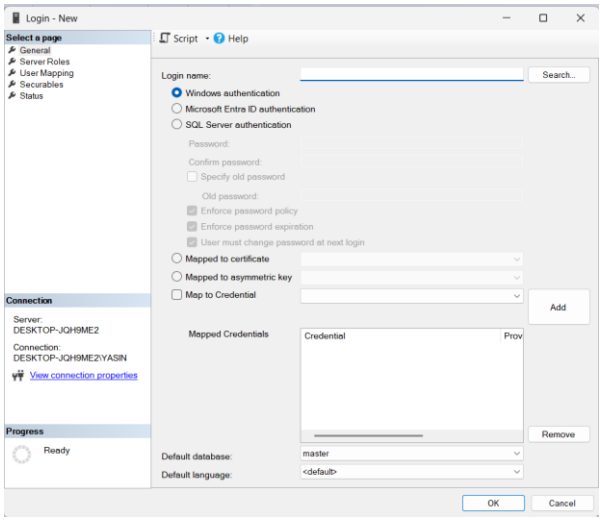
- **Windows Authentication**

  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.



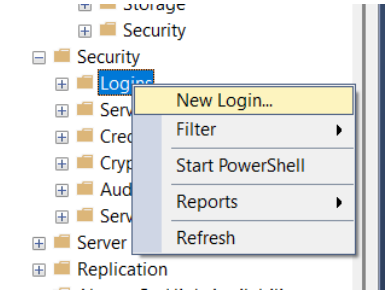


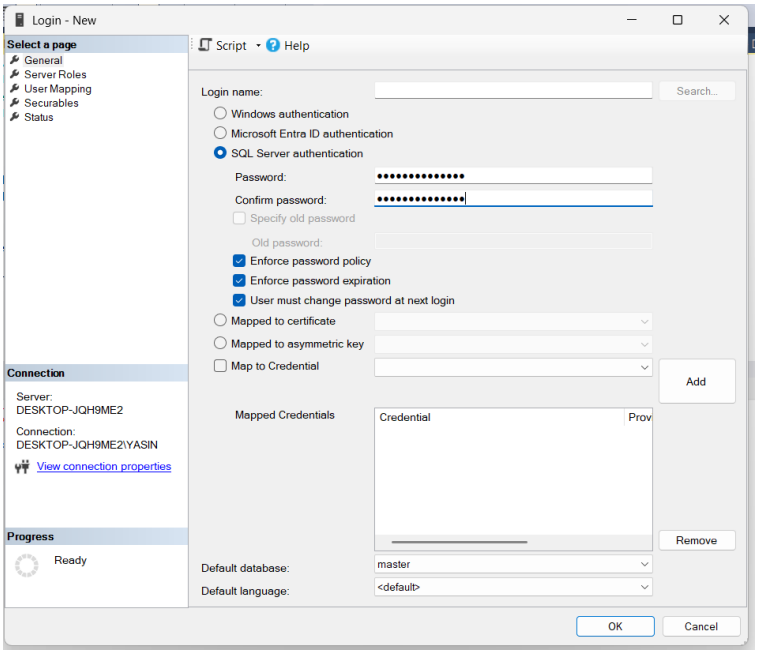
- **SQL Server Authentication**

  Aşağıdaki betik ile kullanıcı oluşturulabilir:

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

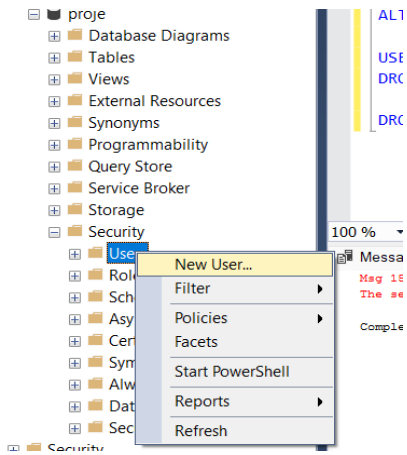
- Kullanıcı adı ve parola SQL Server üzerinde tanımlanır.
- Uzak bağlantılar ve harici uygulamalar için esneklik sağlar.
- Parola karma algoritması ile saklanır.
- Kullanıcılar rollerle sınırlandırılır (db_datareader, db_datawriter, db_owner vb.).

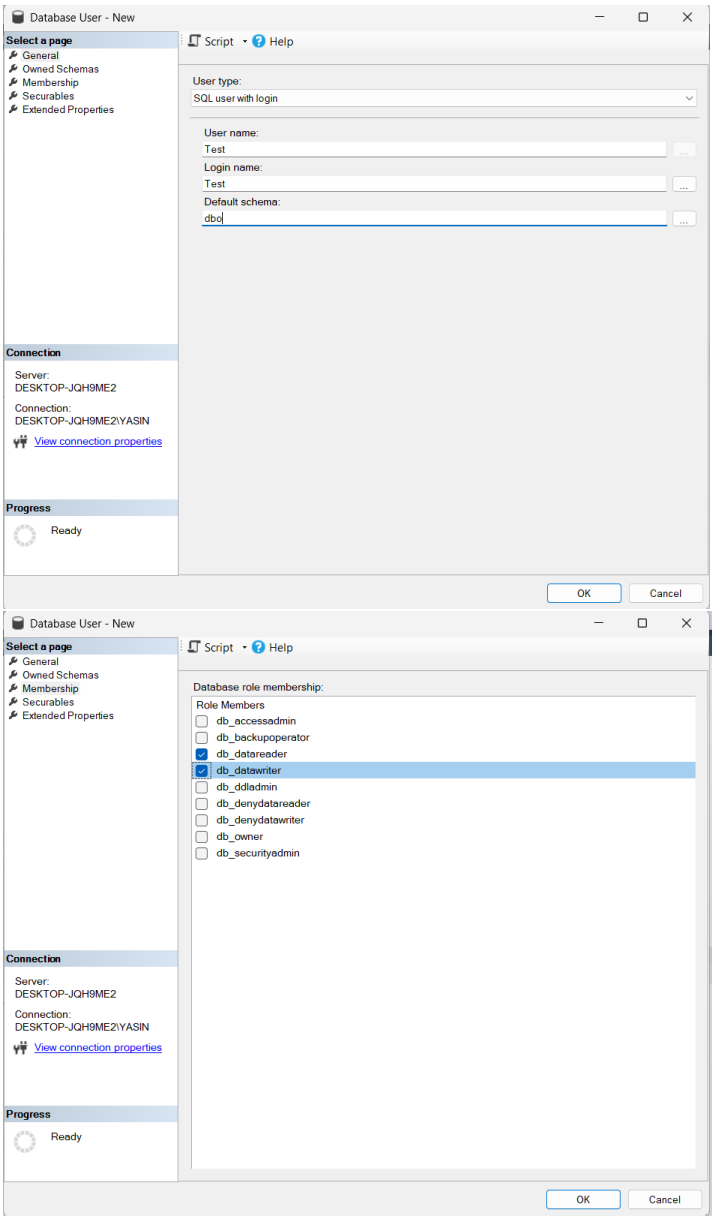




Aşağıdaki betik ile kullanıcılara yetkilendirme verilebilir:

```
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.

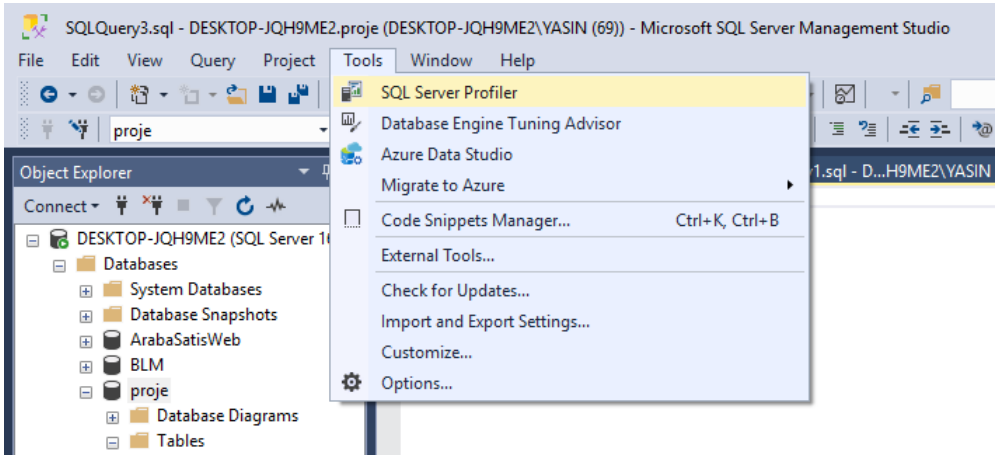
4. VERİTABANI PERFORMANS OPTİMİZASYONU VE İZLEME

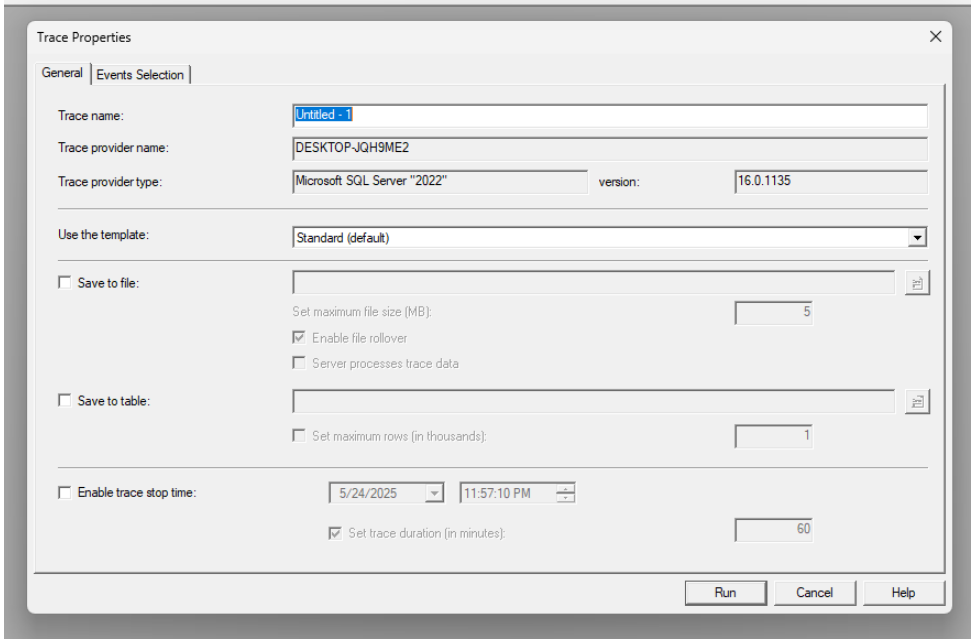
4.1. Giriş

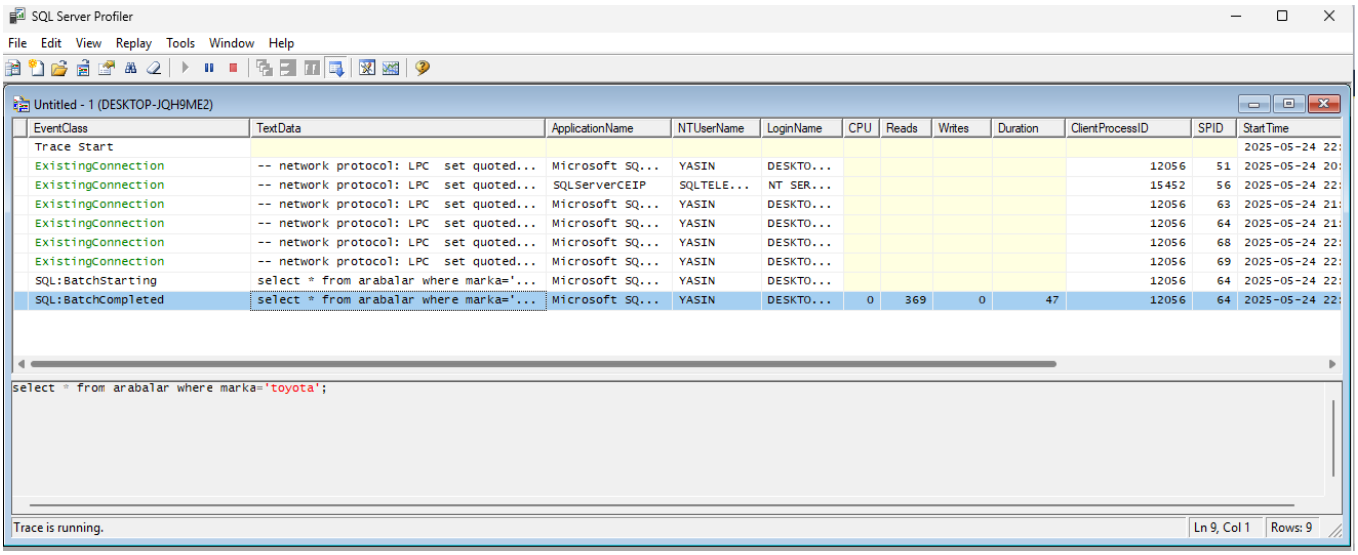
Bu bölümde, büyük bir veritabanı üzerinde performans izleme, analiz ve iyileştirme çalışmaları ele alınmıştır Ayrıca veri yöneticisi rolleriyle erisim kontrolü sağlanarak sistem güvenliği de gözetilmektedir.

4.2. Veritabanı İzleme

Performans takibi için **SQL Server Profiler** kullanılabilir. Bu araçla sorgu çalışma süreleri, kaynak kullanımı ve potansiyel darboğazlar tespit edilebilir.







4.3. İndeks Yönetimi

Sorgu hızını artırmak için uygun indekslerin kullanılması, gereksiz indekslerin kaldırılması önemlidir.

- İndekslerin Kullanım Durumu Kontrol Etme:

```
SELECT
    OBJECT_NAME(s.object_id) AS TableName,
    i.name AS IndexName,
    user_seeks, user_scans, user_lookups, user_updates
FROM sys.dm_db_index_usage_stats s
INNER JOIN sys.indexes i ON s.object_id = i.object_id AND
s.index_id = i.index_id
WHERE database_id = DB_ID()
ORDER BY user_seeks DESC;
```

- Yeni İndeks Oluşturma:

```
CREATE INDEX IX_Table_Column ON TableName(ColumnName);
```

- Gereksiz İndekslerin Kaldırılması:

```
DROP INDEX IndexName ON TableName;
```

4.4.Sorgu İyileştirme

Sorgu iyileştirme ile yavaş çalışan sorgular incelenir ve optimize edilir.

- Uzun süren sorgu:

```
SELECT * FROM Orders o JOIN Customers c ON o.CustomerID =
c.CustomerID WHERE c.Country = 'USA';
```

- Optimize edilmiş sorgu:

```
SELECT o.OrderID, o.OrderDate, c.CustomerName
FROM Orders o
INNER JOIN Customers c ON o.CustomerID = c.CustomerID
WHERE c.Country = 'USA';
```

4.5.Veri Yöneticisi Rolleri

Veritabanı güvenliğini artırmak için kullanıcılara farklı erişim düzeyleri tanımlanabilir.

- Kullanıcı Oluşturma:

```
CREATE USER readonly_user FOR LOGIN readonly_user;
```

- Kullanıcı yetkilendirme:

```
ALTER ROLE db_datareader ADD MEMBER readonly_user;
```

- Kullanıcı yetkisini silme:

```
ALTER ROLE db_datareader DROP MEMBER readonly_user;
```

- Kullanıcıyı silme:

```
DROP USER readonly_user;
```

5. VERİ TEMİZLEME VE ETL SÜREÇLERİ TASARIMI

5.1. Giriş

ETL süreçleri; verinin doğru, tutarlı ve analiz edilebilir olmasını sağlamak için kritik öneme sahiptir. Bu bölümde, ETL süreçlerinin temel bileşenleri olan veri temizleme, veri dönüştürme, veri yükleme ve veri kalitesi raporlarının nasıl gerçekleştirildiği açıklanmaktadır.

5.2. Veri Temizleme

Eksik, hatalı, tutarsız veya yanlış formatta olan verileri düzeltmek veya çıkarmak için yapılır.

- Eksik Verilerin Tespiti ve Yönetimi:

```
-- Eksik (NULL) değerlerin tespiti
SELECT * FROM tablo_adi WHERE kolon_adi IS NULL;


-- Eksik değerlerin doldurulması (örneğin sabit değerle)
UPDATE tablo_adi
SET kolon_adi = 'VarsayılanDeğer'
WHERE kolon_adi IS NULL;
```

- Tutarsızlıkların Düzeltilmesi:

```
UPDATE tablo_adi
SET cinsiyet = 'Erkek'
WHERE cinsiyet IN ('erkek', 'ERKEK', 'E');

UPDATE tablo_adi
SET cinsiyet = 'Kadın'
WHERE cinsiyet IN ('kadın', 'KADIN', 'K');
```

- Yanlış Formatların Düzeltilmesi:

```
UPDATE tablo_adi
SET tarih_kolonu = CONVERT(date, tarih_kolonu, 103)
-- 103 İngiliz tarih formatı (DD/MM/YYYY)
WHERE ISDATE(tarih_kolonu) = 1;
```

- Tekrarlayan Kayıtların Silinmesi:

```
DELETE FROM tablo_adi
WHERE id NOT IN (
    SELECT MIN(id)
    FROM tablo_adi
    GROUP BY tekrar_eden_kolonlar );
```

5.3. Veri Dönüştürme

Farklı kaynaklardan gelen verileri ortak bir standart formata getirmek için yapılır.

- Veri Tiplerinin Dönüştürülmesi:

```
SELECT
  CAST(kolon1 AS INT) AS kolon1_int,
  CONVERT(varchar, tarih_kolonu, 23) AS tarih_standard
 -- YYYY-MM-DD formatı
FROM kaynak_tablo;
```

- Veri Birleştirme (Join, Union):

```
SELECT a.kolon1, b.kolon2
FROM tablo1 a
INNER JOIN tablo2 b ON a.id = b.id;
```

- Kod Çevirme (Mapping):

```
CASE
  WHEN cinsiyet_kodu = 'M' THEN 'Erkek'
  WHEN cinsiyet_kodu = 'F' THEN 'Kadın'
  ELSE 'Bilinmiyor'
END AS cinsiyet
```

- Normalizasyon veya Hesaplamalar:

```
SELECT
  fiyat,
  fiyat * 1.18 AS fiyat_kdv_dahil
-- KDV hesaplama
FROM urunler;
```

5.4. Veri Yükleme

Dönüştürülmüş verileri hedef veritabanına doğru şekilde aktarmak için kullanılır.

- Insert ile Yeni Veri Ekleme:

```
INSERT INTO hedef_tablo (kolon1, kolon2, kolon3)
SELECT kolon1, kolon2, kolon3
FROM kaynak_tablo_temizlenmis;
```

- Güncelleme (Update) ile Veri Senkronizasyonu:

```
UPDATE hedef_tablo
SET kolon1 = kaynak.kolon1
FROM hedef_tablo h
INNER JOIN kaynak_tablo_temizlenmis kaynak ON h.id = kaynak.id;
```

5.5. Veri Kalitesi Raporları

Temizleme ve dönüştürme işlemlerinden sonra verinin kalitesini raporlamak için kullanılır.

- Eksik Veri Oranı Raporu:

```
SELECT
  COUNT(*) AS ToplamKayıt,
  SUM(CASE WHEN kolon1 IS NULL THEN 1 ELSE 0 END) AS EksikKolon1,
  CAST(SUM(CASE WHEN kolon1 IS NULL THEN 1 ELSE 0 END) AS FLOAT)
/ COUNT(*) * 100 AS EksikKolon1Yuzdesi
FROM hedef_tablo;
```

- Tutarsız Veri Sayısı:

```
SELECT COUNT(*) AS TutarsizKayıt
FROM hedef_tablo
WHERE cinsiyet NOT IN ('Erkek', 'Kadın');
```

6. VERİTABANI YÜK DENGELEME VE DAĞITIK VERİTABANI YAPILARI

6.1. Giriş

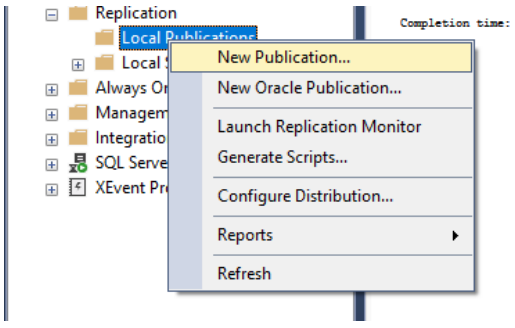
Modern uygulamalar yüksek erişilebilirlik, veri bütünlüğü ve performans gerektirir. Bu ihtiyaçları karşılamak için veritabanı mimarilerinde yük dengeleme, dağıtık yapılandırmalar ve replikasyon teknikleri kritik rol oynar. Bu bölümde, SQL Server ortamında bu mimarilerin nasıl uygulanabileceğini teknik olarak açıklanmaktadır.

6.2. Veritabanı Replikasyonu

Veri replikasyonu, bir veritabanındaki verilerin başka bir sunucuya kopyalanması ve senkronize edilmesidir. Bu yöntem ile veri güvenliği, yedekleme, okunabilir kopyalar ve raporlama işlemleri sağlanabilir.

Tür	Açıklama	Senaryo
Snapshot Replication	Tüm veritabanı anlık görüntü olarak aktarılır.	Küçük veri setlerinde veya seyrek değişen verilerde
Transactional Replication	Verideki her değişiklik anında hedef sunucuya iletilir.	Gerçek zamanlı raporlama sistemleri
Merge Replication	Hem kaynak hem hedef veritabanları değişebilir, veriler birleştirilir.	Offline çalışan mobil sistemlerde

- Replikasyon Kurulumu
  - SSMS > Replication > Local Publications > New Publication
  - İlgili replikasyon türünü seçilir
  - Hedef veritabanını subscriber olarak tanımlanır



6.3. Yük Dengeleme

Veritabanı sistemlerinin yüksek trafikli ortamlarda performans kaybı yaşamaması için yükün birden fazla sunucuya dağıtılması gerekir.

- Always On Availability Groups kullanarak yapılandırma:

  Gereksinimler:

  - En az 2 adet Windows Server işletim sistemi yüklü fiziksel ya da sanal makine.
  - Her sunucuda SQL Server Enterprise Edition kurulmuş olmalıdır.
  - Active Directory ortamı (aynı domain'e üye olmalı).
  - Windows Failover Clustering özelliği etkinleştirilmiş olmalı.
  - Tüm sunucular arasında düzgün çalışan bir ağ bağlantısı (aynı subnet önerilir).
  - DNS üzerinden çözülebilen host adları.
  - SQL Server Enterprise (AG özelliği için gereklidir).
  - SQL Server servis hesabı aynı veya karşılıklı güvenilir olmalı.
  - Veritabanı FULL Recovery Mode'da olmalıdır.

- Yapılandırma Adımları:

  Windows Failover Cluster Kurulumu:

  - Tüm sunucularda "Failover Clustering" özelliğini Server Manager > Add Roles and Features üzerinden yükleyin.
  - Failover Cluster Manager ile yeni bir cluster oluşturun.
    - Örnek cluster adı: SQLCluster01
    - DNS'e kaydedilecek sanal IP belirlenir.
  - Cluster kurulumundan sonra her iki sunucuya da cluster node olarak ekleyin.

- Always On Özelliğini Aktifleştirme:

  • SQL Server Configuration Manager > SQL Server Services
  • İlgili instance'a sağ tıklayın > Properties > **AlwaysOn High Availability** sekmesini açın.
  • Enable `AlwaysOn Availability Groups` kutusunu işaretleyin.
  • Sunucuyu yeniden başlatın.

- Availability Group Oluşturma:

  SSMS Üzerinden:

  1. Always On High Availability > Availability Groups > Sağ tıkla > "New Availability Group Wizard"
  2. **Adım 1:** Availability Group'a isim ver.
  3. **Adım 2:** Eklenecek veritabanını seç.
  4. **Adım 3:** Diğer instance'ları "Add Replica" ile ekle.
     - **Automatic Failover**
     - **Synchronous Commit**

- **Readable Secondary** (opsiyonel)

5. **Adım 4:** Listener oluştur (uygulamaların bağlanacağı sanal ad)
   - DNS adı, port numarası ve IP adresi girilir.
6. **Adım 5:** Yedeklerin aktarımı için yöntem seç:
   - Full: SSMS veritabanlarını otomatik kopyalar.
   - Join Only: Veritabanları elle restore edilmişse
   - Skip: Daha sonra el ile yapılacaksa
7. Yapılandırmayı doğrula ve kurulumu başlat.

6.4.Failover Senaryoları

Failover, bir sunucunun veya hizmetin başarısız olması durumunda sistemin, aynı hizmeti sağlayabilecek başka bir sunucuya otomatik ya da manuel olarak geçiş yapmasıdır.

Tür	Açıklama
Otomatik (Automatic)	**Sistem, bir hata tespit ettiğinde kendiliğinden geçiş yapar.**
Manuel (Manual)	**Yönetici müdahalesi ile geçiş sağlanır.**
Planlı (Planned)	**Bakım, güncelleme gibi durumlarda elle yapılan geçiştir.**
Zorunlu (Forced)	**Tüm replikalarla bağlantı kesildiğinde, veri kaybı riski göze alınarak yapılan geçiştir.**

- Otomatik Failover İçin Gerekenler:
  - En az iki **synchronous** replica olmalı.
  - **Automatic Failover** seçeneği etkinleştirilmiş olmalı.
  - Her iki sunucuda da **sistem durumu izleme** hizmetleri aktif olmalı (WSFC - Windows Server Failover Clustering tarafından sağlanır).

- Failover Stratejileri
  - **Otomatik Failover Senaryosu**

Amaç: Sunuculardan biri (Primary) başarısız olduğunda, sistemin otomatik olarak Secondary sunucuya geçmesi.

Yapılandırma Adımları:

1. **SSMS > Availability Group** altında replikaları düzenle.
2. İki replica da **Synchronous Commit** modunda olmalı.
3. Her iki replica için de "**Automatic Failover**" seçeneği etkinleştirilmelidir.
4. Cluster Health Monitoring yapılandırılmalı (Heartbeat ayarları kritik).

- **Manuel Failover (Planlı Bakım Senaryosu)**

Amaç: Primary sunucuda bakım veya güncelleme yapılacaksa, veri kaybı olmadan kontrolü şekilde diğer replica'ya geçiş yapılması.

Adımlar:

1. **SSMS üzerinden Always On Dashboard** açılır.
2. Availability Group'a sağ tıklanır > **Failover…**
3. İlgili Secondary seçilir ve "Manual Failover" başlatılır.
4. İşlem sonrası kullanıcı bağlantıları kısa süreli kesilebilir.

- **Forced Failover (Veri Kaybı Olasılığı Olan Durumlar)**

Amaç: Primary sunucu tamamen kaybedildiğinde ve bağlantı yoksa, sistemin kullanıma devam edebilmesi için zorunlu failover yapılır.

Adımlar:

1. SSMS > AG > Sağ tıklayıp Failover…
2. "Connect to all secondary replicas" kısmı başarısız olur.
3. Forced failover seçeneği görünür.
4. WITH DATA LOSS uyarısı kabul edilerek geçiş sağlanır.

7. VERİTABANI YEDEKLEME VE OTOMASYON ÇALIŞMASI

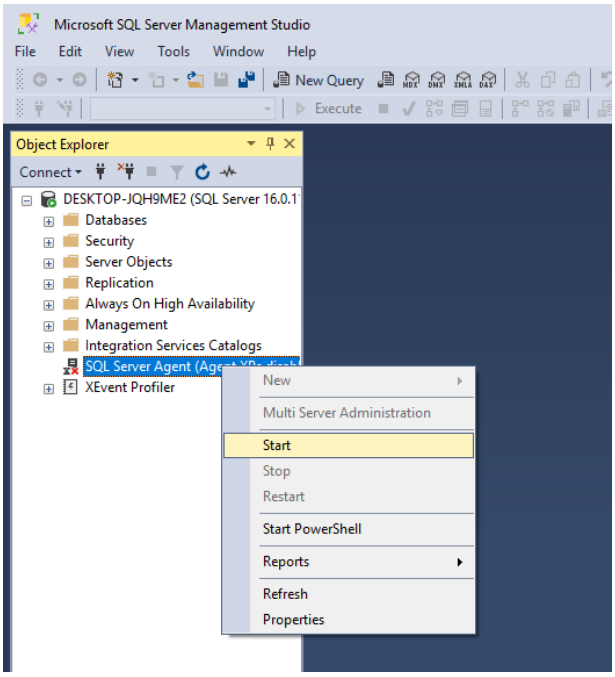
7.1. Giriş

Bu bölümde SQL Server ortamında veritabanı yedekleme işlemlerini otomatik hale getirme ve raporları mail yoluyla otomatik iletme işlemleri anlatılmaktadır.

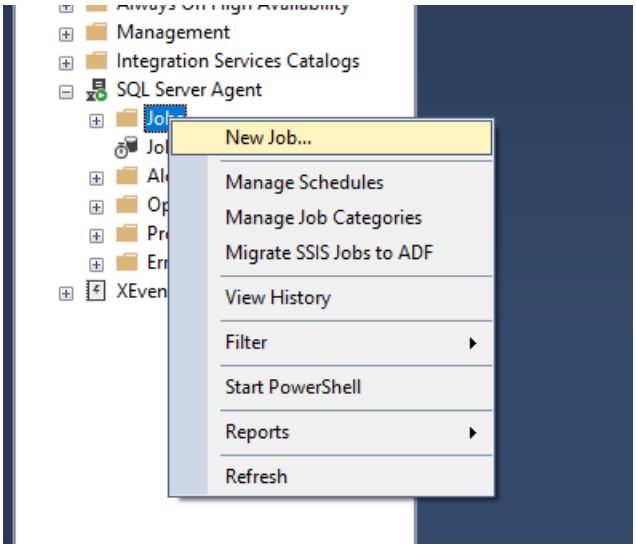
7.2. SQL Server Agent Kullanarak Yedekleme Süreçlerini Otomatikleştirme

➢ SQL Server Agent ile otomatik yedekleme işlemleri aşağıdaki şekilde yapılabilir.

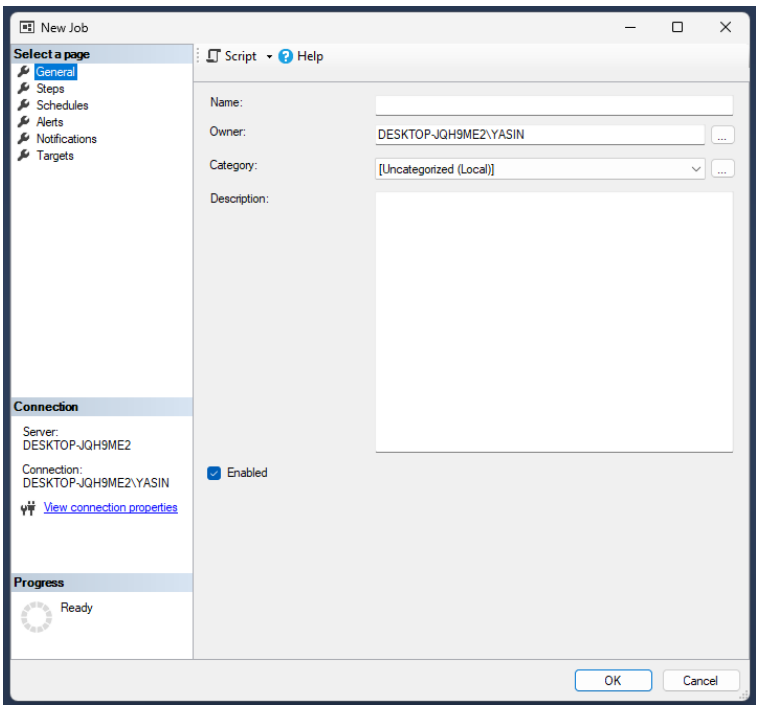
- İlk olarak Sql server agent özelliği başlatılır,



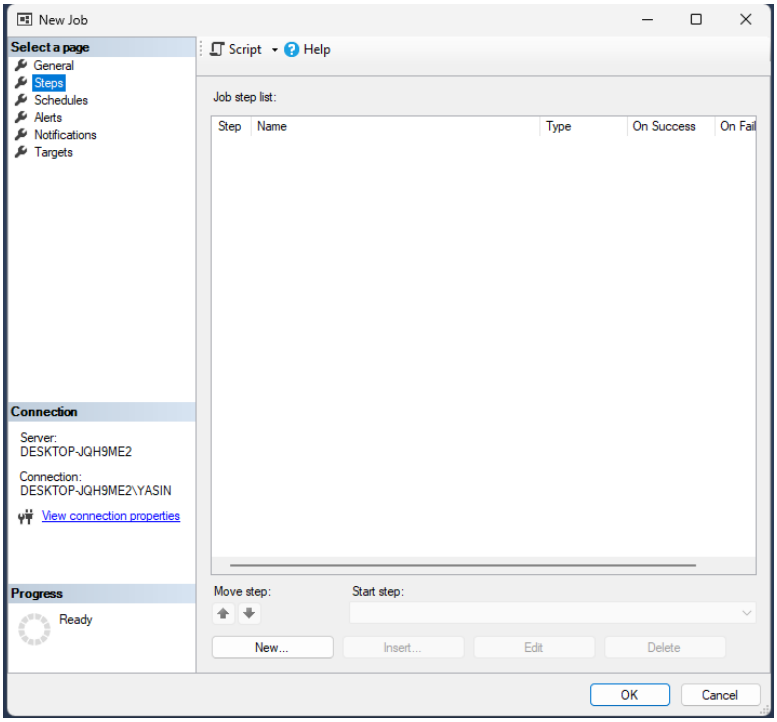
- Sql server agent özelliği altında bulunan jobs lar kısmından yeni job tanımlanır.

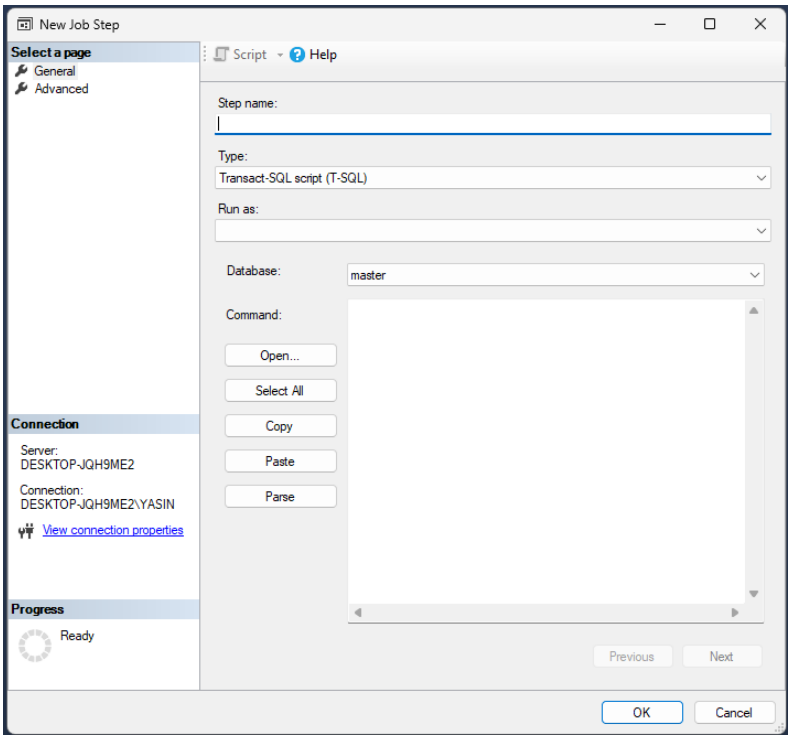


- Gelen ekranda General kısmında atanacak yeni job ın ismi ve açıklaması ayarlanabilir.

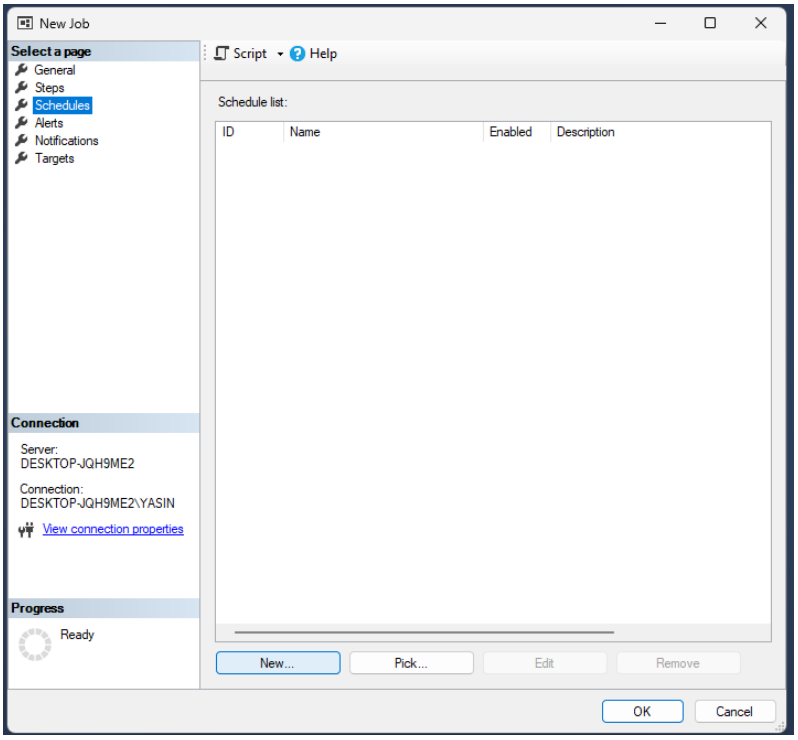


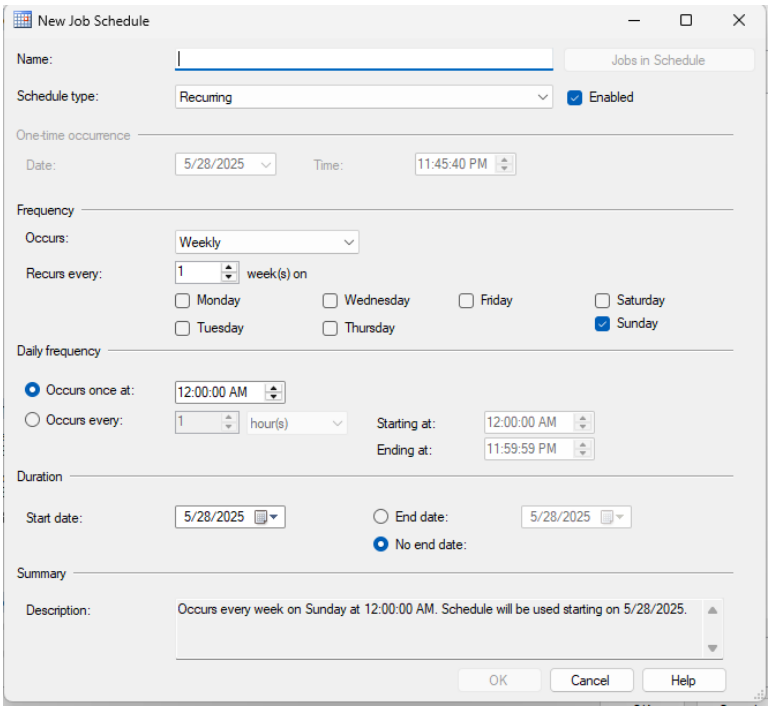
- Daha sonra steps seçeneğinde hangi işlemin otomatikleştirileceği belirlenir ve ilgili





- Schedules özelliğinden, yeni eklenen job ın ne sıklıkla hangi periyotlar aralağında gerçekleştirileceği belirlenir.





7.3. T-SQL Scripting ile yedekleme raporları oluşturma

Veritabanı sistemlerinde alınan yedeklerin takibi ve denetlenmesi, veri güvenliği açısından kritik bir öneme sahiptir. SQL Server, yedekleme geçmişini msdb sistem veritabanında saklar. Bu bilgileri kullanarak, T-SQL yardımıyla detaylı yedekleme raporları oluşturmak mümkündür.

- T-SQL ile Yedekleme Raporu Oluşturma:

  Aşağıdaki komutla yedekleme işlemlerinin raporları oluşturulabilir. İstenirse bu işlem bir job olarak atanıp düzenli olarak raporlar elde edilebilir.

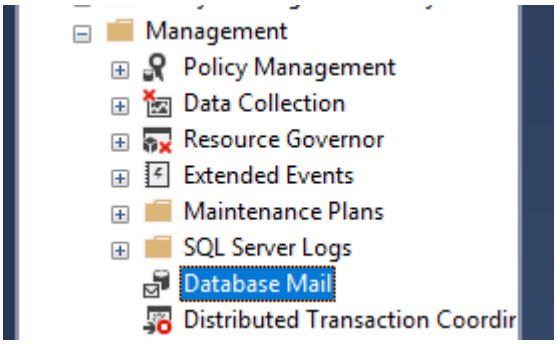
```
SELECT
  d.name AS DatabaseName,
  b.backup_start_date,
  b.backup_finish_date,
  b.type AS BackupTypeCode,
  CASE b.type
      WHEN 'D' THEN 'Full'
      WHEN 'I' THEN 'Differential'
      WHEN 'L' THEN 'Transaction Log'
      ELSE b.type
  END AS BackupType,
  b.backup_size / 1024 / 1024 AS BackupSizeMB,
  mf.physical_device_name AS BackupLocation,
  b.recovery_model
FROM
```

```
    msdb.dbo.backupset b
JOIN
    msdb.dbo.backupmediafamily mf ON b.media_set_id =
mf.media_set_id
JOIN
    sys.databases d ON b.database_name = d.name
WHERE
    b.backup_start_date >= DATEADD(DAY, -7, GETDATE())
ORDER BY
    b.backup_finish_date DESC;
```

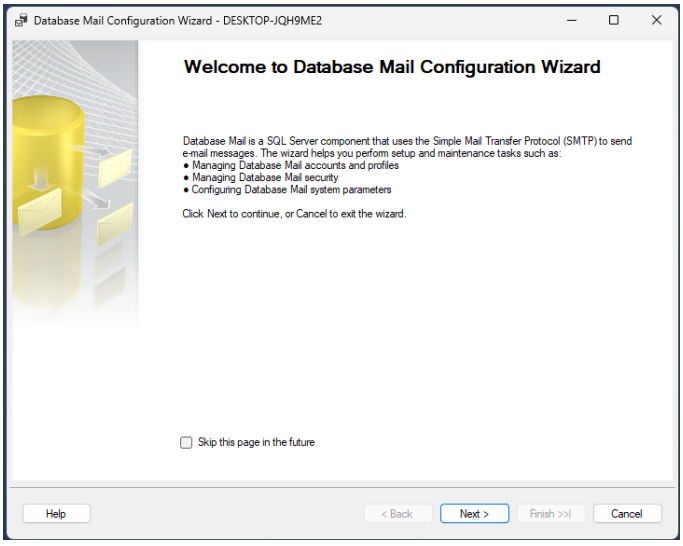
7.4.Otomatik Yedekleme Uyarıları

Bu bölümde, SQL Server ortamında gerçekleştirilen veritabanı yedekleme işlemlerinde oluşabilecek hataların tespiti halinde, ilgili sistem yöneticilerine otomatik e-posta ile uyarı gönderilmesini sağlayacak yapının kurulması ve yapılandırılması açıklanmaktır.

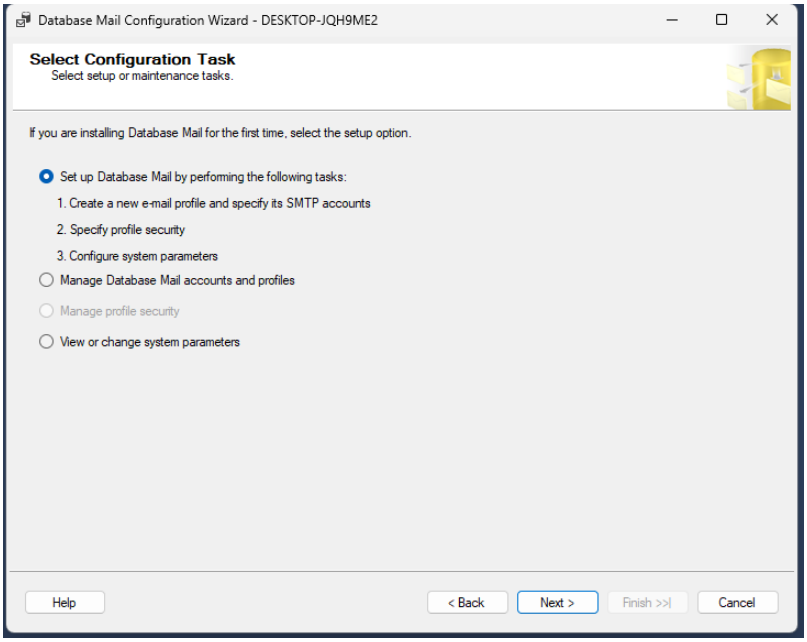
- Management altında bulunan Database Mail özelliği açılır.



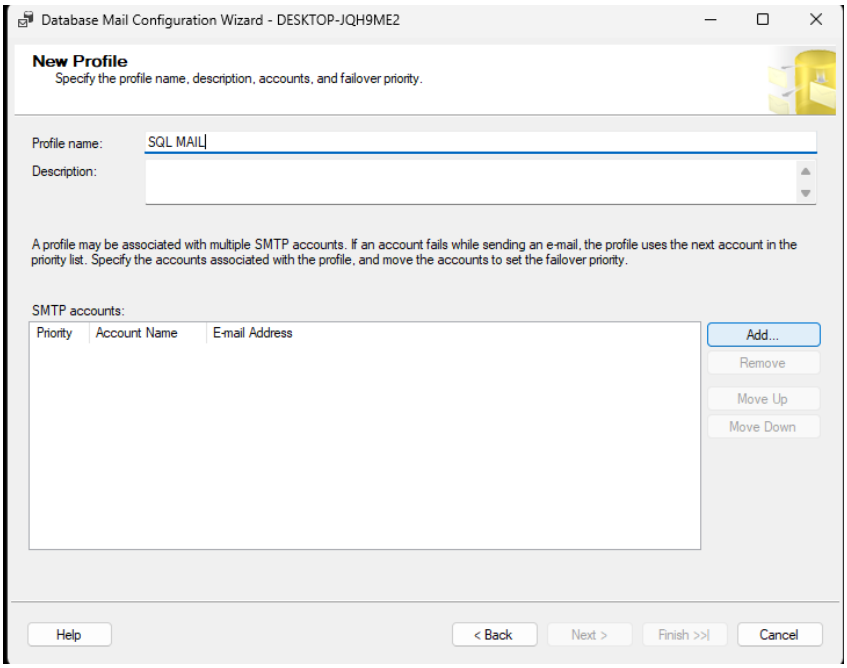
- Çıkan ekranda next seçeneği seçilir.



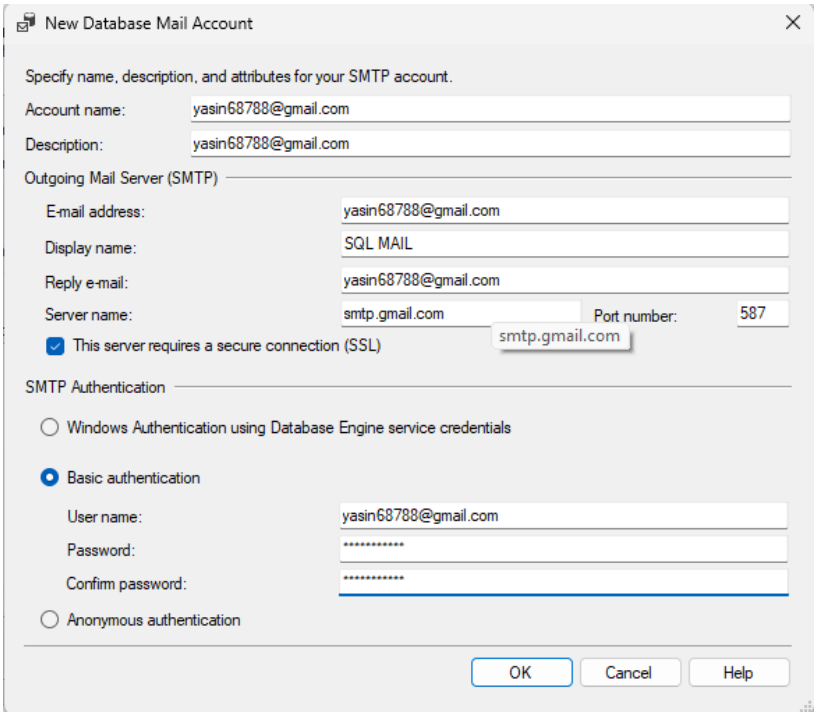
- Sonraki ekranda yeni mail ekleneceği için ilk seçenek seçilir.



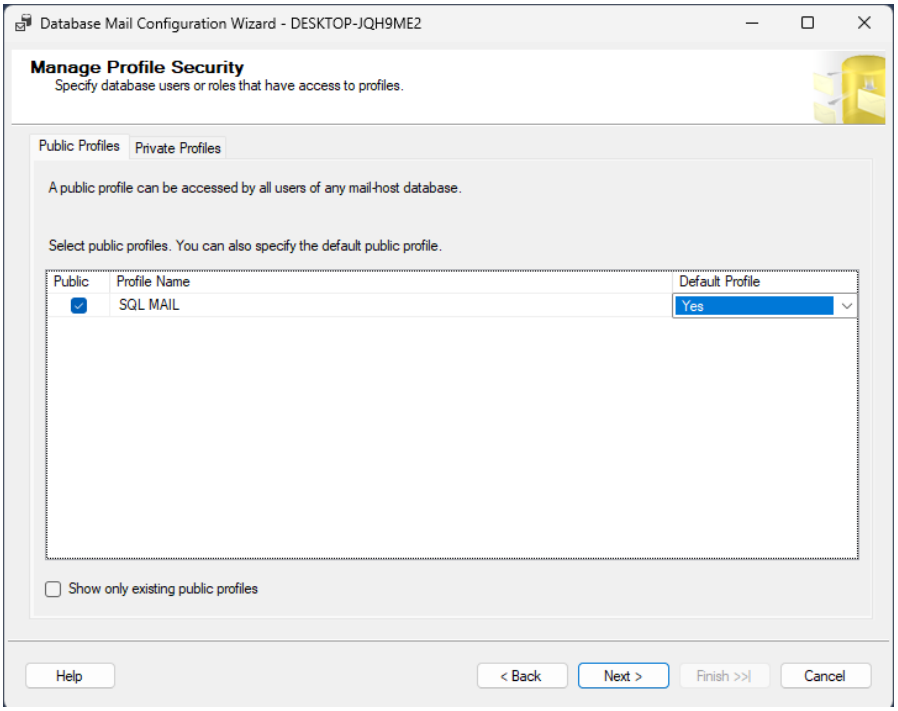
- Daha sonra profile ismi belirlenir ve add seçeneği ile mail akleme sayfasına gidilir.



- Gerekli ayarlamalar yapılır. Burda önemli olan Password kısmı gmailden uygulamalar için alınan şifre ile doldurulmalıdır.



- Sonraki sayfada seçenekler bu şekilde ayarlanmalıdır.



- Sonraki sayfada finish seçeneği seçilerek ayarlanma tamamlanır. Artık mail adresimiz eklenmiştir ve mail gönderebiliriz.