

## BLM4522-AĞ TABANLI PARALEL DAĞITIM SİSTEMLERİ

Konu: Veritabanı Performans Optimizasyonu ve İzleme Veritabanı Yedekleme ve Felaketten Kurtarma Planı Veritabanı Yedekleme ve Otomasyon Çalışması

Hazırlayanlar: Kaan BAYRAM 21290154
Ömer Asaf DEMİR 20290240

Github hesapları: kaanceng

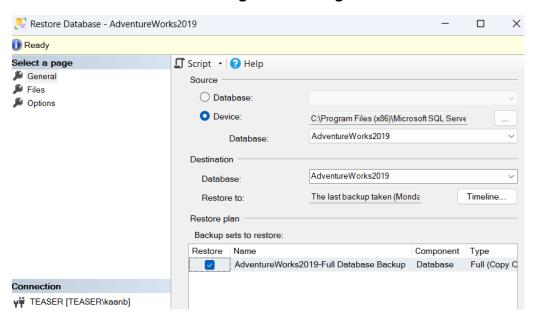
**AsafDemir** 

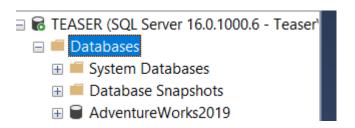
# PROJE 1: Veritabanı Performans Optimizasyonu ve İzleme Giriş:

Bu projede, SQL Server Management Studio (SSMS) ortamında AdventureWorks2019 örnek veritabanı kullanılarak bir veritabanının performansını izleme ve iyileştirme süreçleri uygulanmıştır. Amaç; sistemdeki yavaş sorguları ve performans darboğazlarını tespit edip, uygun indeksleme, sorgu optimizasyonu ve erişim yönetimi teknikleriyle verimliliği artırmaktır. Ayrıca, SQL Profiler ve Dynamic Management Views (DMV) araçları kullanılarak sistem içi kaynak tüketimi analiz edilmiştir. Elde edilen bulgular, uygulanan sorgular ve yapılan geliştirmeler ekran görüntüleriyle desteklenerek raporlanmıştır.

#### 1. Aşama: Örnek Veritabanını Restore Etme

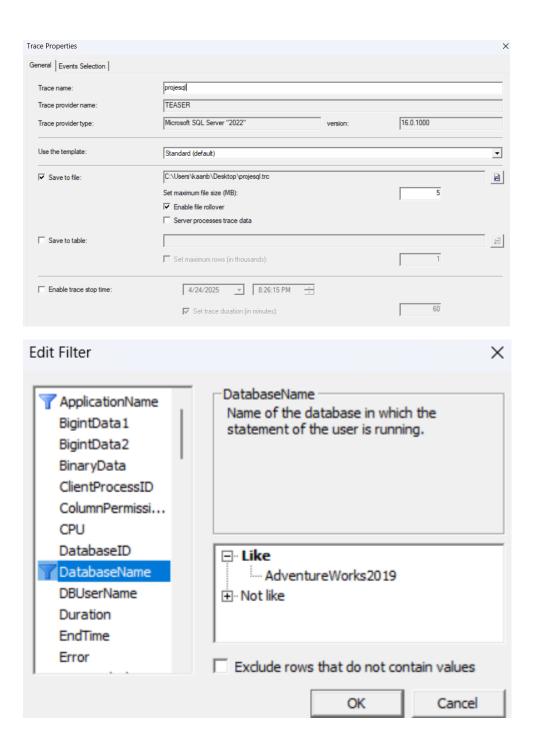
AdventureWorks2019.bak dosyası, SQL Server Management Studio üzerinden restore edilerek veritabanı sisteme yüklendi. Bu adım, projede kullanılacak örnek veritabanının aktif hale gelmesini sağlar.

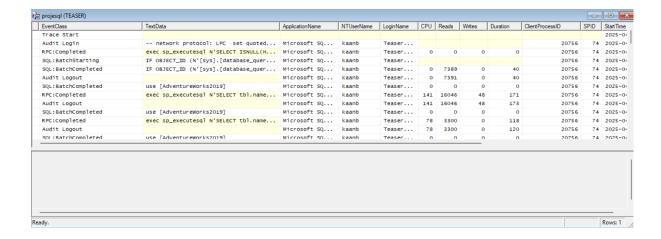




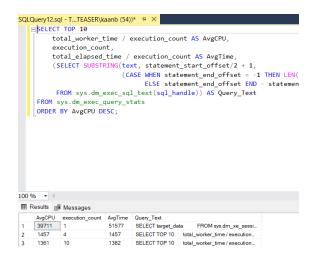
#### 2. AŞAMA: SQL Profiler ile İzleme

SQL Server Profiler ile çalıştırılan sorgular anlık olarak izlenmiş ve hangi sorguların sisteme daha fazla yük bindirdiği tespit edilmiştir. Events Selection ve Column Filters yardımıyla sadece AdventureWorks2019 veritabanına ait işlemler filtrelenerek izlenmiştir.





3. AŞAMA: Dynamic Management Views (DMV) ile Performans Kontrolü sys.dm\_exec\_query\_stats DMV'si kullanılarak CPU ve süre açısından en çok kaynak tüketen sorgular analiz edilmiştir. Bu analiz, hangi sorguların iyileştirilmesi gerektiğini belirlemek için yapılır.



### 4. AŞAMA: İndeks Yönetimi

sys.indexes ve sys.dm\_db\_missing\_index\_details tabloları üzerinden var olan ve eksik indeksler analiz edilmiştir. Eksik indeksler optimize edilerek sorgu performansı artırılmıştır.

#### 1- Mevcut indeksleri listeleme:

```
SQLQuery14.sql - T...TEASER\kaanb (54))* 

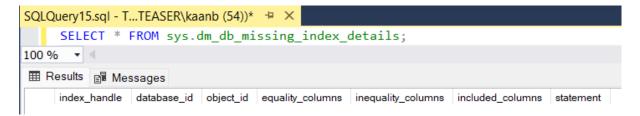
SELECT name, type_desc, is_unique, is_primary_key
FROM sys.indexes
WHERE object_id = OBJECT_ID('Person.Person');

100 % 

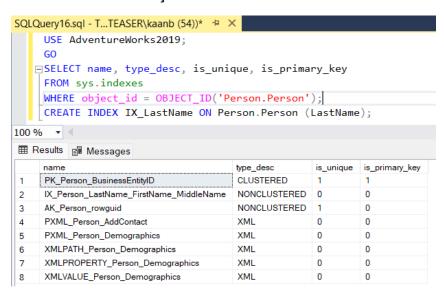
Results Messages

| Name | type_desc | is_unique | is_primary_key
```

#### 2- Eksik indeksleri analiz etme:

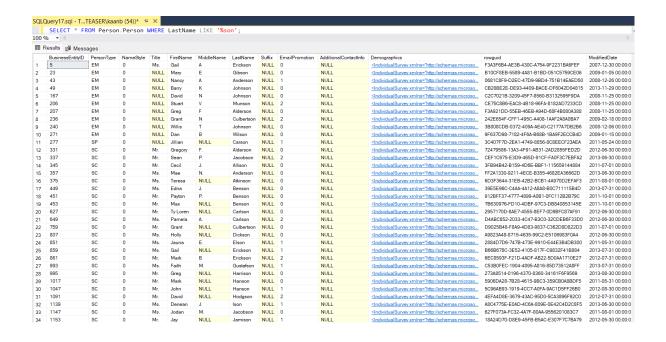


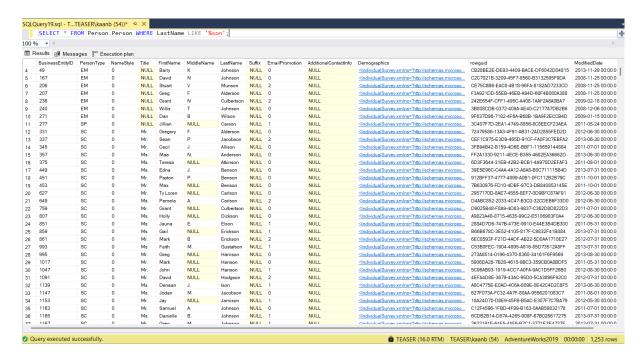
#### 3- Yeni bir indeks oluşturma:



## 5. AŞAMA: Sorgu İyileştirme

İlk olarak indeks olmayan bir sorgunun çalışma planı analiz edilmiş, ardından indeks eklenerek sorgunun çalışma süresi düşürülmüş ve yeni Execution Plan ile karşılaştırılmıştır.





## 6. AŞAMA: Veri Yöneticisi Rolleri (Erişim Yönetimi)

SQL Server'da yeni bir kullanıcı oluşturulmuş ve AdventureWorks2019 veritabanına sadece okuma izni verilmiştir. Böylece veri güvenliği sağlanmış ve yetki sınırlandırması yapılmıştır.

#### 1-Login ve user oluşturma:

```
SQLQuery20.sql - T...TEASER\kaanb (54))* 
CREATE LOGIN NewUser WITH PASSWORD = 'Test123!';
USE AdventureWorks2019;
CREATE USER NewUser FOR LOGIN NewUser;

100 % 
Messages
Commands completed successfully.
```

#### 2-Role ekleme:

```
SQLQuery21.sql - T...TEASER\kaanb (54))* 

EXEC sp_addrolemember 'db_datareader', 'NewUser';

100 % 

■ Messages

Commands completed successfully.
```

#### Sonuç:

Proje kapsamında yapılan çalışmalar sonucunda, SQL Server üzerinde çalışan sorguların kaynak tüketimleri analiz edilmiş, eksik indekslerin tespitiyle performans artışı sağlanmıştır. Özellikle uygun indeksleme ve sorgu iyileştirme teknikleri sayesinde sistemin işlem süreleri düşürülmüş, gereksiz yükler azaltılmıştır. Ayrıca, kullanıcı yetkilendirme işlemleri gerçekleştirilerek erişim güvenliği sağlanmıştır. Tüm işlemler GitHub platformuna yüklenmiş ve proje çıktıları dökümantasyon halinde sunulmuştur. Bu süreçler, büyük veritabanlarında yönetimsel verimliliği artırmak adına önemli bir temel oluşturmuştur.