**Proje 6: Veritabanı Yükseltme ve Sürüm Yönetimi**

**Kullanılan veri tabanı:** Powerlifting Database

**Link:** <https://www.kaggle.com/datasets/open-powerlifting/powerlifting-database?resource=download>

**Video Linki:**

Bu projede istenen adımlar 3 temel başlık altında değerlendirilmiş:

1. Veritabanı Yükseltme Planı
2. Sürüm Yönetimi
3. Test ve Geri Dönüş Planı,

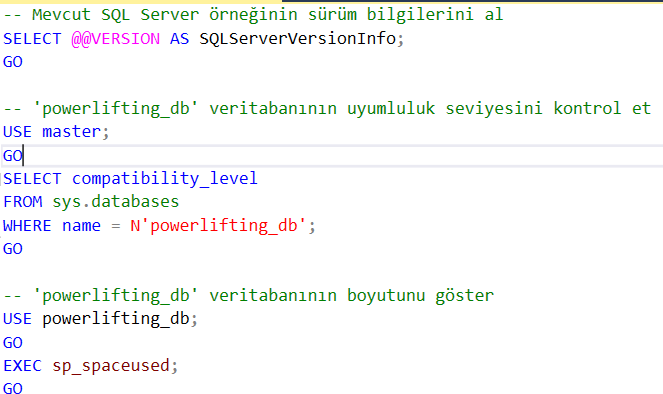
Biz de yaptığımız adımları bu başlıklar üzerinde değerlendirecek olursak:

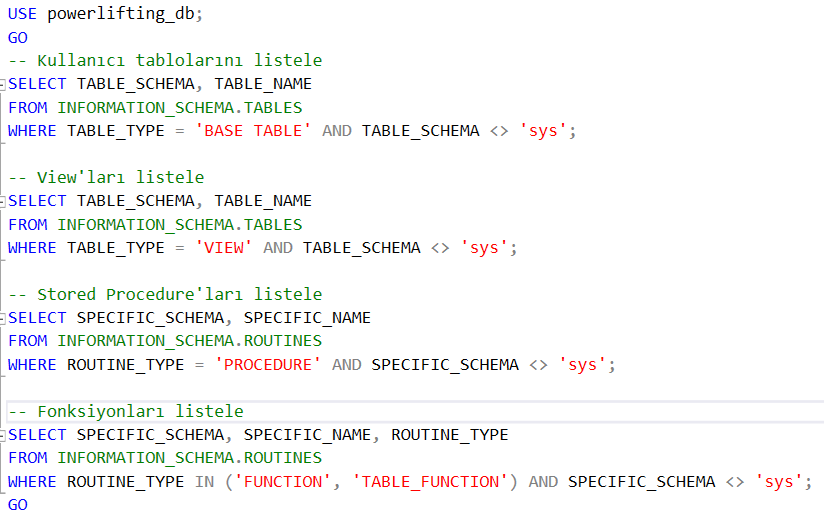
**Bölüm 1: Veritabanı Yükseltme Planı**

Bu bölüm, mevcut powerlifting\_db veritabanının daha yeni bir SQL Server sürümüne veya aynı sürümün farklı bir yapısına yükseltilmesi sürecini planlamaya odaklanmaktadır. Bu bölümde, yükseltme stratejisinin temel adımları ve dikkat edilmesi gereken noktalar ele alınacaktır. Bu bölümdeki adımlar genellikle doğrudan SQL çalıştırmaktan çok, planlama ve belgeleme faaliyetlerini içerir ancak mevcut durumu anlamak için bazı sorgular kullanılacaktır.

**Adım 1.1: Mevcut Veritabanı Durumunun Analizi ve Belgelenmesi**

Yükseltme planının ilk ve en kritik adımı, mevcut SQL Server ve powerlifting\_db ortamının detaylı bir şekilde analiz edilmesi ve belgelenmesidir. Bu, mevcut sürüm, yapılandırma, boyut, içerik ve bağımlılıklar hakkında net bir fikir edinmemizi sağlar. Aşağıdaki sorgular, bu temel bilgileri toplamak için kullanılabilir.





Yukarıdaki sorgular çalıştırıldığında sonuçları aşağıda görüldüğü gibidir:

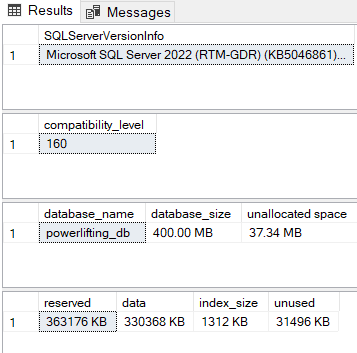
**SQL Server Sürümü:** Mevcut sunucu Microsoft SQL Server 2022 (RTM-GDR) sürümünde çalışmaktadır.

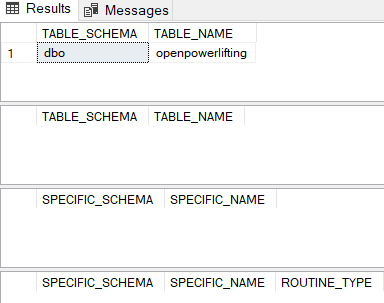
**Veritabanı Uyumluluk Seviyesi:** powerlifting\_db veritabanının uyumluluk seviyesi 160'tır (SQL Server 2022).

**Veritabanı Boyutu:** Veritabanının toplam boyutu yaklaşık 400 MB'tır ve bunun ~355 MB'ı veri ve indeksler için ayrılmıştır.

**Veritabanı Nesneleri:** Veritabanı yapısı oldukça basittir; tek bir kullanıcı tablosu (dbo.openpowerlifting) içermekte ve şu anda özel view, stored procedure veya fonksiyon barındırmamaktadır.

Bu bilgiler, yükseltme stratejisi oluşturmak için kritik bir temel sağlar. Mevcut sistemin güncel bir SQL Server sürümünde ve uyumluluk seviyesinde olması, veritabanı boyutunun yönetilebilir olması ve özel kod nesnelerinin bulunmaması, olası bir yükseltme veya güncelleme işlemini teknik açıdan basitleştirebilir. Ancak, yine de sunucu yapılandırmaları, güvenlik ayarları, bakım planları, yedekleme stratejileri ve özellikle dbo.openpowerlifting tablosuna bağımlı uygulamaların detaylı analizi ve belgelenmesi gereklidir.





**Adım 1.2: Yükseltme Hedefleri ve Kapsamının Belirlenmesi**

powerlifting\_db veritabanı ve barındığı SQL Server örneği için planlanan yükseltmenin temel hedefleri, kapsamı ve kısıtları aşağıdaki gibi belirlenmiştir:

**Gerekçe:** En güncel SQL Server güvenlik yamalarından, potansiyel performans iyileştirmelerinden faydalanmak ve platformu güncel tutmak.

**Hedef Sürüm:** Mevcut SQL Server 2022 örneğini, yayınlanmış **en son Kararlı Kümülatif Güncelleme (CU)** paketine yükseltmek.

**Kapsam:**

Yalnızca mevcut SQL Server veritabanı motoru örneği güncellenecektir.

İşletim sistemi veya donanımda değişiklik yapılmayacaktır.

powerlifting\_db veritabanı şemasında veya uyumluluk seviyesinde değişiklik yapılmayacaktır.

Veritabanına bağlanan uygulamaların uyumluluğu ayrı bir test sürecinde doğrulanacaktır.

**Temel Kısıtlar/Beklentiler:**

Yükseltme işlemi için planlanan maksimum kesinti süresi 2 saattir.

Yükseltme sonrası veritabanı performansının en az mevcut seviyede kalması veya iyileşmesi beklenmektedir.

**Adım 1.3: Yükseltme Yönteminin Seçilmesi ve Risk Analizi**

Önceki adımlarda yapılan analizler (mevcut SQL Server 2022 sürümü, veritabanının basit yapısı) ve belirlenen hedefler (en son Kümülatif Güncelleme (CU) paketini uygulama) doğrultusunda, bu plan için Yerinde Yükseltme yöntemi tercih edilmiştir. Bu yöntemde, mevcut SQL Server örneği üzerine doğrudan güncelleme yapılır.

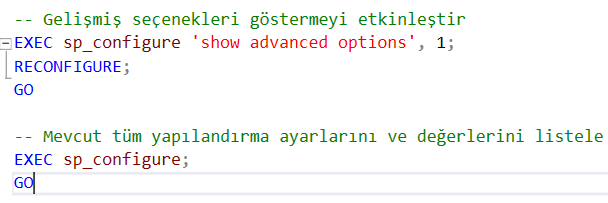
Bu yöntemle ilişkili temel riskler şunlardır:

* Geri Dönüş Zorluğu: Yükseltme sırasında beklenmedik bir sorun olursa, önceki duruma geri dönmek (rollback) daha karmaşıktır. Tam yedeklemeler esastır.
* Beklenmedik Sorunlar: Kurulum hataları veya yapılandırma sorunları yaşanabilir.
* Uygulama Uyumluluğu: Veritabanına bağlanan uygulamalarda uyumluluk sorunları çıkabilir.
* Performans Değişiklikleri: Yükseltme sonrası performansta düşüşler olabilir.
* Yetersiz Test: Yükseltme öncesi yetersiz test, canlı ortamda sorun riskini artırır.

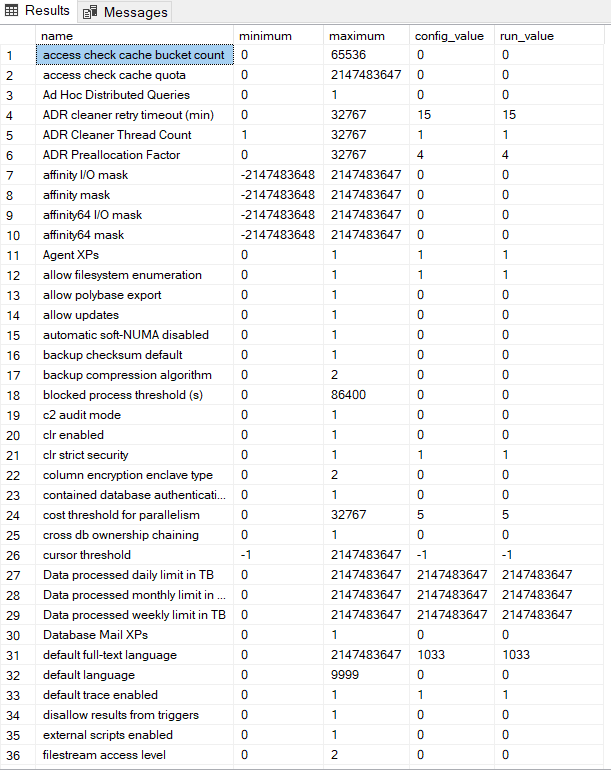
Bu tanımlanan riskler için alınacak önlemler ve test stratejileri Bölüm 3 (Test ve Geri Dönüş Planı) altında detaylandırılacaktır.

**Adım 1.4: Önemli Yapılandırma Ayarlarının Belgelenmesi**

Yükseltme planlamasının bir parçası olarak, mevcut SQL Server örneğinin önemli yapılandırma ayarlarının belgelenmesi, yükseltme sonrası tutarlılığı sağlamak ve olası performans değişikliklerini analiz etmek açısından önemlidir. sp\_configure sistem saklı yordamı bu ayarları görüntülemek için kullanılır. Tüm ayarları görebilmek için öncelikle 'show advanced options' seçeneğinin etkinleştirilmesi gerekebilir.



Yukarıdaki sorgular çalıştırıldığında sonuçları aşağıda görüldüğü gibidir. İlk sorgu, gelişmiş yapılandırma seçeneklerinin görünürlüğünü sağlar. İkinci sorgu (sp\_configure) ise, SQL Server örneğinin çok sayıda yapılandırma ayarını ve bu ayarların mevcut çalışan değerlerini (run\_value) listeler. Aşağıdaki görselde görülen listeden, örneğin cost threshold for parallelism ayarının çalışan değerinin 5 olduğu gibi bilgiler elde edilir. Yükseltme planı için, bu çıktıdan özellikle max server memory, max degree of parallelism gibi performansla ilgili kritik ayarların mevcut çalışan değerleri dikkatlice not edilmelidir. Bu belgeleme, yükseltme sonrasında bu önemli ayarların değişip değişmediğini kontrol etmek ve gerekirse eski haline getirmek veya bilinçli olarak yeni değerlerde bırakmak için temel oluşturur.



**Bölüm 2: Sürüm Yönetimi**

Bu bölüm, powerlifting\_db veritabanının yapısında zaman içinde meydana gelen değişiklikleri yönetmeye ve izlemeye odaklanmaktadır. Etkili sürüm yönetimi, veritabanı şemasının farklı versiyonları arasındaki farkları anlamayı, planlı değişiklikleri kontrollü bir şekilde uygulamayı ve beklenmedik veya yetkisiz değişiklikleri tespit etmeyi içerir. Bu bölümün ilk adımlarında, veritabanının belirli sürümleri arasındaki farkları analiz etme ve planlama yöntemlerine değinilecek, ardından ise DDL tetikleyicileri kullanarak anlık şema değişikliklerini izleme mekanizması kurulacaktır.

**Adım 2.1: Mevcut Şemanın Dışa Aktarılması**

Veritabanı sürüm yönetiminin temel adımlarından biri, mevcut (baseline) veritabanı şemasının tam bir anlık görüntüsünü (snapshot) oluşturmaktır. Bu anlık görüntü, gelecekteki değişiklikleri karşılaştırmak veya gerektiğinde eski bir sürüme dönmek için referans noktası görevi görür. Bu projede tercih edilen yöntem DACPAC (Data-Tier Application Component Package) Çıkarma'dır.

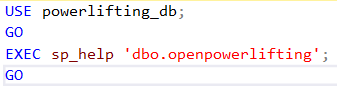
Veritabanının tüm şema nesnelerini (tablolar, view'lar, prosedürler, fonksiyonlar, indeksler, kullanıcılar vb.) içeren bir .dacpac dosyası oluşturulur.

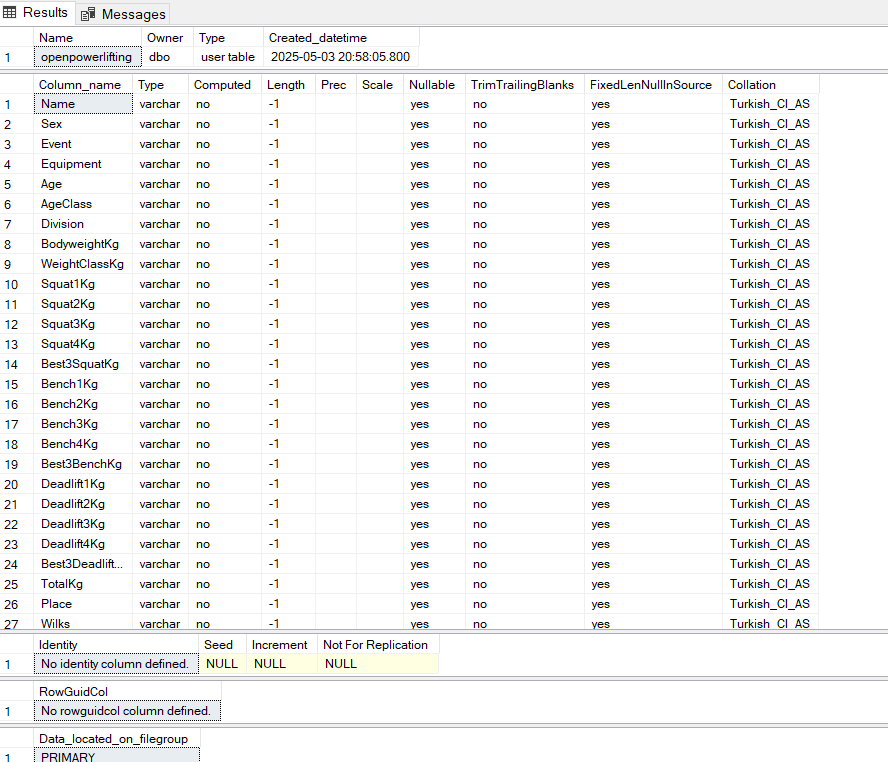
Bu DACPAC dosyası, şemanın taşınabilir, model tabanlı bir temsilidir ve "v1" olarak kabul edilen temel şema versiyonunu kayıt altına alır. Bu dosya, sürüm kontrol sistemlerinde saklanabilir ve sonraki adımlarda şema karşılaştırma işlemleri için kaynak olarak kullanılır.

Bu adımda doğrudan çalıştırılacak bir T-SQL sorgusu yoktur; işlem, belirtilen yönetim araçları üzerinden gerçekleştirilir ve sonuç olarak bir .dacpac dosyası elde edilir.

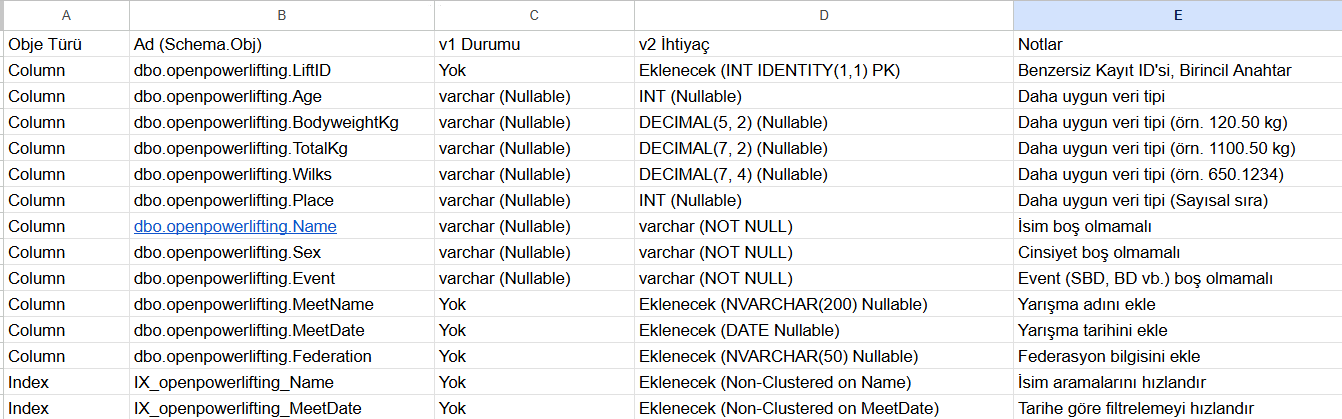
**Adım 2.2: Hedef Sürüm (v2) Değişikliklerinin Tanımlanması**

Mevcut veritabanı şemasının anlık görüntüsü (v1.dacpac) alındıktan ve dbo.openpowerlifting tablosunun yapısı incelendikten sonra, "v2" sürümü için planlanan yapısal değişiklikler ve iyileştirmeler bir fark tablosunda değerlendirilmek için veri tabanının yapısını öğrenmemiz gerekir. Bunun için aşağıdaki sorgu çalıştırılarak sonucu incelenmesi uygundur.



****

Yukarıdaki sorgu sonucunun analiz edilmesine dayanarak, powerlifting\_db veritabanının "v2" sürümü için aşağıdaki gibi iyileştirmeler ve değişiklikler planlayabiliriz.



Bu değişiklik listesi, v2 sürümünün hedeflerini yansıtır ve ilgili paydaşlar tarafından onaylandıktan sonra geliştirme ve uygulama için temel oluşturur.

**Adım 2.3: Şema Karşılaştırma**

SSMS'te doğrudan bir şema karşılaştırma aracı bulunmadığından, iki veritabanı sürümü (örn. v1 ve v2) arasındaki farkları bulmak için manuel bir yöntem izlenebilir. Bu yöntem, her iki sürümün şema script'lerini oluşturup bir metin karşılaştırma aracıyla incelemeyi içerir.

**Adım 2.3.1: Kaynak Şema Script'ini Oluşturma (v1)**

SSMS Object Explorer'da powerlifting\_db veritabanı üzerinden generate script aracılığyla karşılaştırmanın temeli olacak mevcut (v1) veritabanı şemasının tam bir SQL script'ini oluşturulur.

**Adım 2.3.2: Hedef Şema Script'ini Oluşturma (v2)**

Adım 2.2'de Fark Tablosu'nda tanımlanan tüm v2 değişikliklerinin uygulandığı bir veritabanı (örn. geliştirme veya test ortamındaki powerlifting\_db\_v2) üzerinde Adım 2.3.1'deki aynı adımlar ve aynı gelişmiş script seçenekleri tekrarlanır.

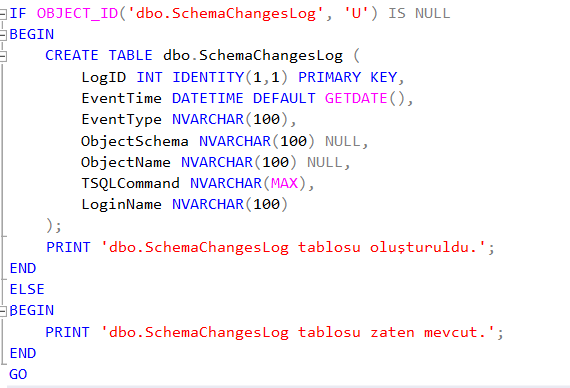
Eğer v2 değişikliklerinin uygulandığı ayrı bir veritabanı yoksa, bu adım atlanır ve karşılaştırma doğrudan Fark Tablosu'na göre yapılır veya sadece v1 script'i referans olarak kullanılır.

**Adım 2.3.3: Script Dosyalarını Karşılaştırma**

Oluşturulan iki script dosyası arasındaki farkları bularak v1'den v2'ye geçiş için gereken DDL değişikliklerini tespit edilir.

**Adım 2.4: Şema Değişiklikleri Log Tablosunun Oluşturulması**

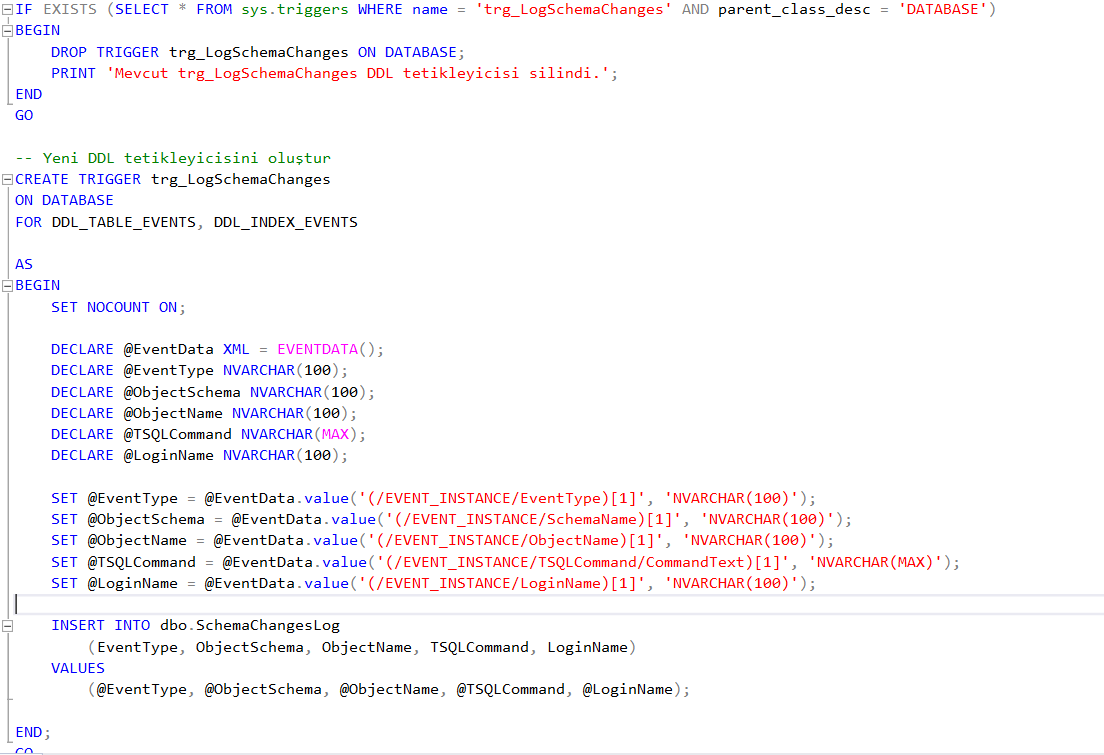
Veritabanı şemasında yapılacak değişiklikleri DDL tetikleyicisi ile otomatik olarak izleyebilmek için, bu değişikliklere ait bilgilerin kaydedileceği bir log tablosuna ihtiyaç vardır. Bu adımda, dbo.SchemaChangesLog adında, DDL olayının tipi, zamanı, ilgili nesne, işlemi yapan kullanıcı ve çalıştırılan T-SQL komutu gibi bilgileri içerecek olan log tablosu oluşturulacaktır.



Yukarıdaki sorgu çalıştırıldığında, şema değişikliklerini loglamak için kullanılacak hedef tablo (dbo.SchemaChangesLog) veritabanında oluşturulur veya zaten var olduğu belirtilir. Bu tablo, bir sonraki adımda oluşturulacak DDL tetikleyicisi için bir önkoşuldur.

**Adım 2.5: Şema Değişikliklerini İzlemek İçin DDL Tetikleyicisi Oluşturma**

Veritabanı şemasında kimin, ne zaman, ne değişiklik yaptığını takip etmek için otomatik bir mekanizma kurmak önemlidir. Bu adımda, powerlifting\_db veritabanı üzerinde gerçekleşen belirli DDL (Veri Tanımlama Dili - örn. CREATE, ALTER, DROP) olaylarını yakalayan bir DDL Tetikleyicisi (DDL Trigger) oluşturulacaktır. Bu tetikleyici, yakaladığı olayın detaylarını (olay tipi, nesne adı, T-SQL komutu, yapan kullanıcı vb.) EVENTDATA() fonksiyonu aracılığıyla alacak ve Adım 2.4'te oluşturduğumuz dbo.SchemaChangesLog tablosuna kaydedecektir. Bu örnekte, tetikleyici şimdilik sadece Tablo ve İndeks olaylarını (DDL\_TABLE\_EVENTS, DDL\_INDEX\_EVENTS) izleyecektir.



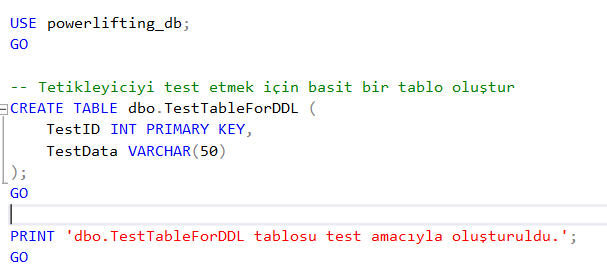
Yukarıdaki sorgu çalıştırıldığında, trg\_LogSchemaChanges adında veritabanı seviyesinde bir DDL tetikleyicisi oluşturulur (veya varsa güncellenir). Bu tetikleyici artık aktiftir ve powerlifting\_db veritabanında bir tablo veya indeks üzerinde CREATE, ALTER veya DROP işlemi yapıldığı anda otomatik olarak çalışacak ve işlem detaylarını dbo.SchemaChangesLog tablosuna kaydedecektir.

**Adım 2.6: DDL Tetikleyicisinin Test Edilmesi**

Bu adım, tetikleyicinin beklendiği gibi çalışıp çalışmadığını görmek için kontrollü bir DDL işlemi yapmayı ve ardından log tablosunu incelemeyi içerir.

**Adım 2.6.1: Test Amaçlı Tablo Oluşturma**

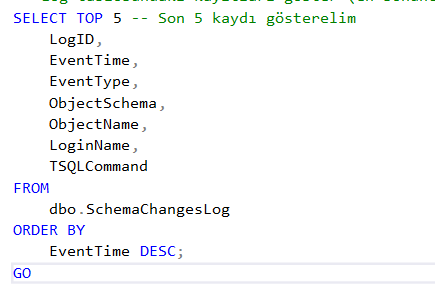
DDL tetikleyicisini (trg\_LogSchemaChanges) test etmek amacıyla, kapsamı dahilinde olan (DDL\_TABLE\_EVENTS) basit bir DDL komutu çalıştırılacaktır. Bunun için dbo.TestTableForDDL adında geçici bir test tablosu oluşturulacaktır. Bu işlemin tetikleyicimizi çalıştırması ve log tablosuna bir kayıt eklemesi beklenir.



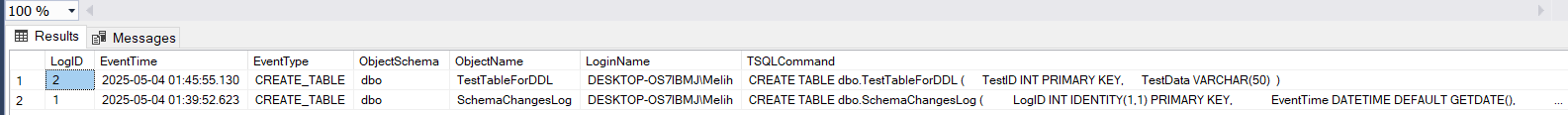
Yukarıdaki CREATE TABLE komutu çalıştırıldığında, test tablosu oluşturulur. Bu işlemin trg\_LogSchemaChanges DDL tetikleyicisini çalıştırmış olması gerekir.

**Adım 2.6.2: Log Tablosunun Kontrol Edilmesi**

CREATE TABLE işleminin loglanıp loglanmadığını görmek için dbo.SchemaChangesLog tablosu sorgulanacaktır. En son olayın en üstte görünmesi için EventTime'a göre tersten sıralama yapılması faydalıdır. CREATE\_TABLE tipinde yeni bir kayıt görmeyi bekliyoruz.

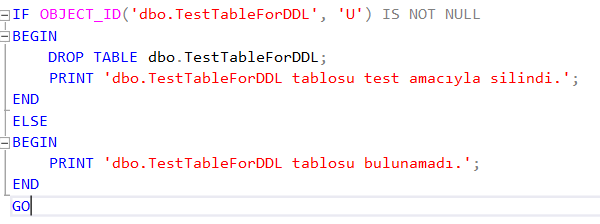


Yukarıdaki sorgu çalıştırıldığında sonuçları aşağıda görüldüğü gibidir. Sonuçlarda, en güncel kaydın (LogID 2) EventType değerinin CREATE\_TABLE ve ObjectName değerinin TestTableForDDL olduğu görülmektedir. Ayrıca çalıştırılan T-SQL komutu da loglanmıştır. Bu, DDL tetikleyicisinin test tablosunun oluşturulma olayını başarıyla yakaladığını doğrular.



**Adım 2.6.3: Test Amaçlı Tablonun Silinmesi (DDL İşlemi)**

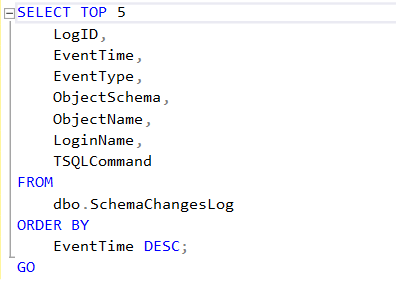
Test işlemini tamamlamak ve ortamı temizlemek için oluşturulan test tablosu şimdi silinecektir. Bu DROP TABLE işlemi de tetikleyici tarafından yakalanmalı ve loglanmalıdır.



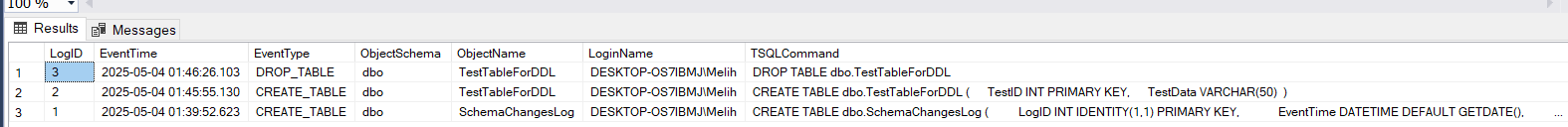
Yukarıdaki DROP TABLE komutu çalıştırıldığında, test tablosu silinir. Bu işlemin de trg\_LogSchemaChanges DDL tetikleyicisini çalıştırmış olması gerekir.

**Adım 2.6.4: Log Tablosunun Kontrol Edilmesi (İkinci Kontrol)**

DROP TABLE işleminin loglanıp loglanmadığını görmek için dbo.SchemaChangesLog tablosu tekrar sorgulanacaktır. Şimdi en üstte DROP\_TABLE tipinde yeni bir kayıt görmeyi bekliyoruz.



Yukarıdaki sorgu çalıştırıldığında sonuçları aşağıda görüldüğü gibidir. Sonuçlarda, en güncel kaydın (LogID 3) EventType değerinin DROP\_TABLE ve ObjectName değerinin TestTableForDDL olduğu görülmektedir. Bu, DDL tetikleyicisinin test tablosunun silinme olayını da başarıyla yakalayıp logladığını teyit eder. Tetikleyici mekanizması beklendiği gibi çalışmaktadır.



**Bölüm 3: Test ve Geri Dönüş Planı**

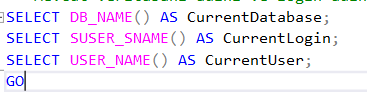
Bu bölüm, Bölüm 1'de planlanan veritabanı yükseltme veya Bölüm 2 kapsamında uygulanan önemli şema değişiklikleri sonrasında gerçekleştirilecek kritik adımları ele alır: Test ve Geri Dönüş (Rollback). Test süreci, yapılan değişikliklerin beklendiği gibi çalıştığını, sistemin kararlılığını ve performansını doğrulamayı amaçlar. Geri Dönüş Planı ise, testler sırasında veya değişiklik sonrası operasyonda ciddi sorunlar yaşanması durumunda, sistemi bilinen son kararlı duruma güvenli bir şekilde döndürmek için izlenecek adımları tanımlar. Bu bölümdeki adımlar da büyük ölçüde planlama ve prosedür tanımlama odaklıdır.

**Adım 3.1: Yükseltme Sonrası Testlerin Gerçekleştirilmesi**

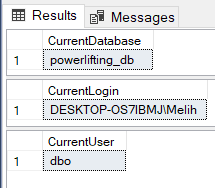
Yükseltme işlemi (veya önemli değişiklikler) tamamlandıktan sonra, sistemin sağlığını ve beklendiği gibi çalıştığını doğrulamak için bir dizi test gerçekleştirilir. Bu adımda, temel bazı testler SQL sorguları aracılığıyla uygulanacak ve sonuçları belgelenecektir.

**Adım 3.1.1: Temel Bağlantı ve Kullanıcı Doğrulama**

İlk kontrol, veritabanına başarılı bir şekilde bağlanıldığını ve doğru kullanıcı bağlamında olunduğunu teyit etmektir.



Yukarıdaki sorgu çalıştırıldığında sonuçları aşağıda görüldüğü gibidir. Sonuçlar, bağlantının powerlifting\_db veritabanına yapıldığını ve işlemi gerçekleştiren login/kullanıcı bilgilerini doğrulamaktadır. Temel erişimde bir sorun olmadığı teyit edilmiştir.

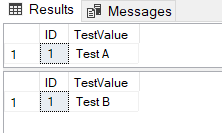
****

**Adım 3.1.2: Fonksiyonel Test**

Veritabanının temel veri işleme (CRUD - Create, Read, Update, Delete) fonksiyonlarının çalıştığını test etmek için geçici bir tablo üzerinde basit işlemler yapılacaktır.

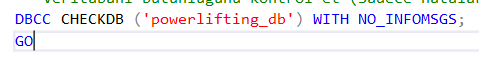
**

Yukarıdaki SQL script bloğu hatasız bir şekilde çalıştırılmıştır. PRINT mesajları işlemlerin sırayla tamamlandığını göstermiş, SELECT sorgularının sonuçları ise verinin beklendiği gibi eklendiğini ('Test A') ve güncellendiğini ('Test B') doğrulamıştır. Bu, temel CRUD işlemlerinin veritabanı üzerinde sorunsuz çalıştığını teyit eder.



**Adım 3.1.3: Veri Bütünlüğü Kontrolü**

Veritabanının fiziksel ve mantıksal bütünlüğünde bir sorun olup olmadığını kontrol etmek için DBCC CHECKDB komutu çalıştırılacaktır. Bu komut, yükseltme sonrası rutin kontrollerin önemli bir parçasıdır. Sadece hataları göstermesi için WITH NO\_INFOMSGS parametresi kullanılacaktır.



Yukarıdaki sorgu çalıştırıldığında herhangi bir hata mesajı döndürmemiştir. WITH NO\_INFOMSGS kullanıldığı için sonuç döndürmemesi, powerlifting\_db veritabanında herhangi bir tutarlılık veya bozulma hatası bulunmadığını gösterir. Veritabanı bütünlüğü testi başarıyla tamamlanmıştır.

**Adım 3.2: Geri Dönüş (Rollback) Stratejisinin Belirlenmesi**

Herhangi bir veritabanı yükseltme veya önemli değişiklik işleminde, işlerin beklenmedik bir şekilde ters gitmesi ihtimaline karşı net bir Geri Dönüş Stratejisi bulunmalıdır. Bu strateji, sistemi değişikliğin yapılmasından önceki, bilinen son kararlı duruma döndürmeyi amaçlar.

Bölüm 1, Adım 1.3'te bu proje için Yerinde Yükseltme (In-place Upgrade) yöntemi seçilmiştir. Bu yöntem için en güvenilir ve yaygın geri dönüş stratejisi şudur:

**Ana Strateji: Yedekten Geri Dönme**

**Ön Koşul:** Yükseltme işlemine başlamadan hemen önce, powerlifting\_db veritabanının ve kritik sistem veritabanlarının tamyedeklerinin alınmış ve bu yedeklerin doğrulanmış olması zorunludur. (Yedek doğrulaması Adım 3.3'te ele alınacaktır). Sunucu seviyesinde önemli değişiklikler yapılıyorsa (bu senaryoda olmasa da), işletim sistemi veya sanal makine seviyesinde anlık görüntü almak da değerlendirilebilir.

**Uygulama:** Eğer yükseltme sonrası yapılan testler (Adım 3.1) kritik hatalar ortaya çıkarırsa veya sistem kararsız hale gelirse ve sorun kısa sürede çözülemezse, Geri Dönüş Planı devreye alınır. Bu genellikle aşağıdaki adımları içerir:

1. Veritabanına gelen tüm bağlantıları durdurma.
2. SQL Server servislerini durdurma.
3. Yükseltme öncesi alınan tam yedekten powerlifting\_db veritabanını geri yükleme (RESTORE DATABASE ... WITH REPLACE).
4. Sistem veritabanları da etkilendiyse (genellikle büyük sürüm yükseltmelerinde daha olasıdır), onları da yedekten geri yükleme (bu daha karmaşık bir işlemdir).
5. SQL Server servislerini başlatma.
6. Temel bağlantı ve fonksiyon testleri yaparak sistemin eski kararlı duruma döndüğünü doğrulama.

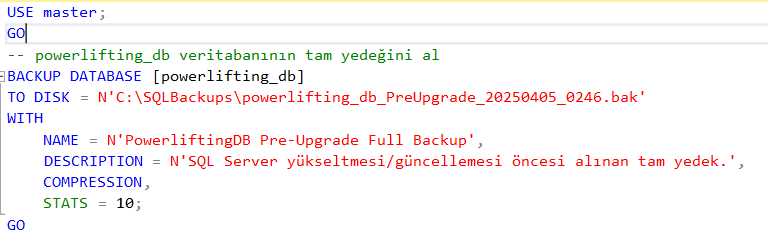
**Karar Verme:** Geri dönüş kararının hangi koşullarda ve kim tarafından verileceği önceden belirlenmelidir.

**Adım 3.3: Yedekleme ve Doğrulama Prosedürü**

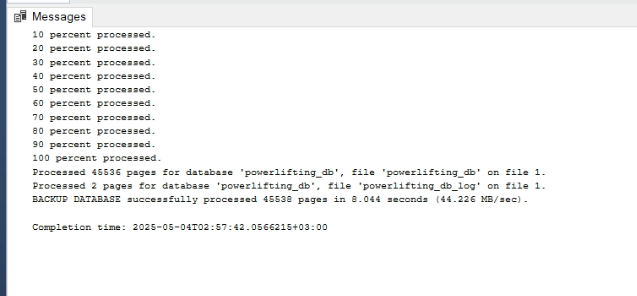
Adım 3.2'de belirlenen "Yedekten Geri Dönme" stratejisinin başarılı olabilmesi için, yükseltme veya değişiklik işlemine başlamadan hemen önce güvenilir bir tam veritabanı yedeğinin alınması ve bu yedeğin kullanılabilir olduğunun doğrulanması hayati öneme sahiptir. Bu adım, bu prosedürleri tanımlar.

**Adım 3.3.1: Yükseltme Öncesi Tam Yedek Alma**

Yükseltme işlemine başlamadan önceki son adımlardan biri, powerlifting\_db veritabanının tam yedeğini almaktır. Bu yedek, geri dönüş senaryosunda kullanılacak ana referanstır. Aşağıdaki komut, powerlifting\_db için tam yedek alır.

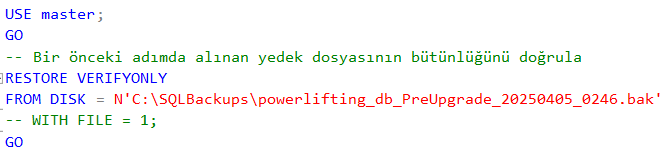


Yukarıdaki BACKUP DATABASE komutu çalıştırıldığında, powerlifting\_db veritabanının tam yedeği belirtilen dosya yoluna kaydedilir. Komutun başarıyla tamamlandığına dair mesaj alınmalıdır. Bu yedek dosyası güvenli bir yerde saklanmalıdır. Aşağıda mevcut sonucu gösteren çıktı görülmektedir.



**Adım 3.3.2: Yedek Dosyasını Doğrulama**

Yedek dosyasının fiziksel olarak oluşturulmuş olması, onun geçerli ve geri yüklenebilir olduğu anlamına gelmez. Yedeğin bütünlüğünü ve okunabilirliğini kontrol etmek için RESTORE VERIFYONLY komutu kullanılır. Bu komut, yedeği gerçekten geri yüklemez, sadece başlıklarını okur ve temel bütünlük kontrollerini yapar.



Yukarıdaki RESTORE VERIFYONLY komutu çalıştırıldığında, "The backup set on file 1 is valid." mesajı başarıyla alınmıştır. Bu, alınan yedeğin temel bütünlük kontrollerini geçtiğini ve geri dönüş (restore) senaryosunda kullanılabilir olduğunu doğrular. Geri dönüş planı için gerekli olan geçerli yedek artık mevcuttur ve doğrulanmıştır. Aşağıda ilgili sorgunun çıktısı çıktı görülmektedir.

