

View Stored Procedure Trigger



View

- Bir veya birden fazla tablodan istenilen verilerin bir arada sunulmasını sağlayan tanımlanmış sorgulardır.
- View'ler «select» ifadesi ile tanımlanmış **sanal tablolar**dır. Temel amacı tabloların içerisinde veri kümesi getirip ortaya çıkan sonucu sanal tabloymuş gibi yeniden sorgulayabilmemizi sağlamaktır.
- View'ler sadece tablolar içerisindeki verileri görüntülemeye yarar. View içerisinde verileri etkileyecek herhangi bir işlem yapılamaz.

Neden Kullanılır

- karmaşık sorguları basitleştirmek;
- sorgu süresini kısaltmak ve ağ üzerindeki trafiği düşürmek;
- erişim izinlerini düzenlemek;
- farklı sunuculardaki benzer verileri karşılaştırmak.

View

Örnek olarak her ay sonunda mali raporu oluşturmak üzere kullanılan bir sorgu olduğunu düşünelim.

Karmaşık olan bu sorguyu her ay script olarak çalıştırmak hem kod karmaşası hem de güvenliği açısından risk oluşturacaktır. Böyle bir durumda ilgili rapor için kullanılan sorgu scriptinin view olarak veritabanında nesne olarak saklanması daha uygun olacaktır. Ayrıca veritabanı nesnesi olarak saklandığı için yetki bazlı çalışma sistemi ile güvenlik önlemleri de alınabilir.

View kullanılarak son kullanıcıların tablolara direk erişimi kısıtlanarak sadece view erişim izni verilebilir. Sadece son kullanıcıların görmesi istenilen kolonlar eklenerek sadece o kolonlara erişim sağlanabilir. Örneğin çalışanların TC,ad,soyad,adres,anne kızlık soyadı,maaş vb. birçok bilgisinin bulunduğu bir çalışan tablosu olduğu düşünelim. Şirket çalışanlarının ad soyad bilgilerine ihtiyaç duyulan bir yazılım geliştirilmesi gereken bir durumda tüm çalışan tablosunun erişime açılması TC, kızlık soyadı ve maaş gibi erişim gerekmeyen özel bilgilere de erişilmesini sağlayacaktır. Bu nedenle direkt olarak tabloya yetki vermek yerine sadece ad ve soyad kolonlarını getiren bir select sorgusu yazıp bunu view olarak oluşturmak ve yazılımcıya oluşturulan view üzerinden veriye erişebilme yetkisi vermek daha uygun olacaktır.

View

- Viewler son kullanıcılara verileri filtreli ve istenilen kolonlar şeklinde sunmanın etkili bir yöntemidir.
- Çok sayıda join ve kolon seçimi işlemini gizlemek için kullanılabilir.
- Kullanıcılara sadece viewlere erişim izni verilerek güvenlik düzeyi ayarlanır.
- View oluşturulduğunda tanımlaması veritabanı içerisinde saklanır, fakat döndürdüğü veri gerçek veri tablolarından ayrı bir yerde tutulmaz.
- Parametrik olarak kullanılamazlar. Stored procedure ler ve kullanıcı tanımlı fonksiyonlar çalışma anında parametre alarak farklı sonuç setleri döndürebilir. Ancak view'ler parametre almaz bu nedenle farklı veri setleri için ayrı ayrı view oluşturulması gerekir.
 - Örneğin 81 ilde şubesi bulunan bir şirketi düşünürsek; her ilde yapılan faaliyetlere yönelik bir rapor istendiğinde her il için ayrı ayrı view oluşturmak mantıklı olmayacaktır. Bunun yerine parametrik çalışabilen kullanıcı tanımlı fonksiyon (User defined function) kullanımı daha yerinde bir çözüm olacaktır.
- View'ler bilinçli bir şekilde oluşturulmazsa performansı olumsuz etkileyebilir:
 - Kullanılan sorguların çok iyi optimize edilmesi gerekir. View'ler veri tutmaz, sadece sorgu sırasında kullanılan tablolardan veri okur. Bu nedenle sorgu olabildiğince optimize edilmelidir.
 - Birbirini çağıran viewler oluşturmamaya özen gösterilmelidir. İç içe viewler çok sık kullanılırsa sorgu optimizasyonu, performans ve troubleshooting aşamalarında zorluk çıkarır.
 - View yerine kullanma imkanı varsa stored procedure kullanımı tercih edilebilir. Stored procedure avantajları performansı arttıracaktır.

View lerde Ne Yapılamaz?

- **İsimsiz bir kolon kullanılamaz.**
 - Örneğin Sum fonksiyonu kullanılacak ise As ile kesin bir isim verilmelidir.
- **En fazla 1024 kolon seçilebilir.**
- **view içerisinde parametre gönderilemez.**
- **View içerisinde order by (sıralama) fonksiyonu kullanılamaz. (TOP vb. ile birlikte kullanım)**
- **DML işlemleri (insert, update, delete) view içinde yazılamaz.**


View Oluşturma

```
CREATE VIEW view_adı
```

```
AS
```

```
SELECT sütun_adları
```

```
FROM temel_tablo
```



```
CREATE VIEW view_adı  
WITH ENCRYPTION  
AS  
SELECT sütun_adları  
FROM temel_tablo
```

- SQL server’da nesneleri güvenlik sebebiyle kilitlemek isteyebiliriz. Böyle bir kullanımda with encryption ifadesi kullanılabilir. With encryption ile oluşturulan nesneye **design** seçeneği ile erişilemez.
- SQL server kullanan yazılım firmaları view ve stored procedurelerini değiştirilmeye, çalınmaya ve rakip firmalara göstermemek adına şifrelerler.
- Şifreli view kullanılacaksa orijinal tanımlamasını başka bir yere kopyalamanız gerekir.

View
Güncelleştirme
(Alter View)

Alter VIEW view_adı

AS

SELECT sütun_adları

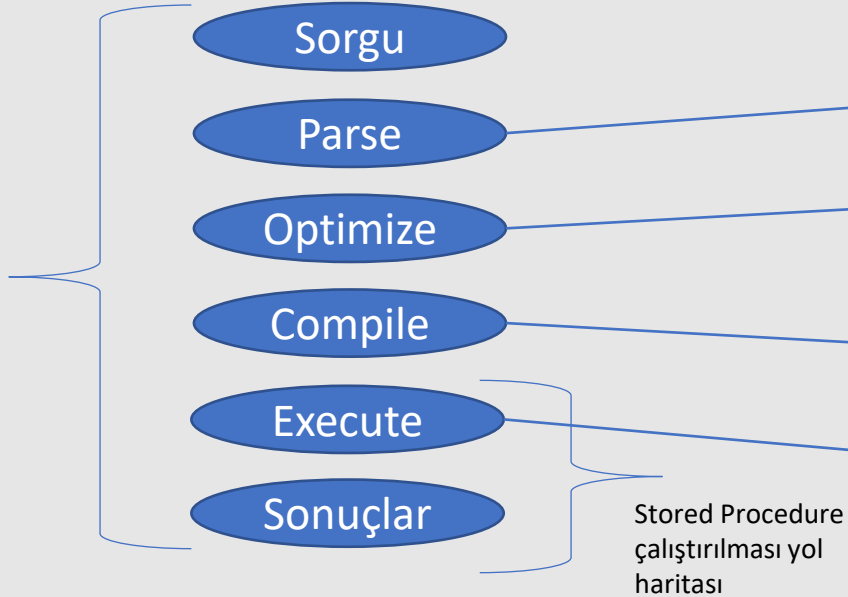
FROM temel_tablo

Stored Procedure

- Bir veya daha fazla TSQL ifadesinin kod bloğu şeklinde bir defa yazılıp saklanması ve ihtiyaç anında kullanılması için oluşturulan SQL Server objeleridir.
- Normal sorgulardan farklı olarak ilk oluşturulduklarında Cache belleğe kaydedilip sonraki çalışmalarda bu meta verilerin tekrar üretilmesi yerine Cache bellekten okuyarak normal sorgulara göre daha performanslı çalışır.

Stored Procedure

Genel bir TSQL
sorgusu
çalıştırılması



- SP için Execute işlemi öncesi adımlar ilk çalıştırılmada yapıp belleğe kaydedilir.
- **Parse** : SQL Server'a gelen sorguların TSQL yazım kurallarına uygunluğu kontrol edilir.
- **Optimize**: Sorgunun çalıştırılabilmesi için izlenebilecek yollar belirlenir. Olası yollar ve maliyetler hesaplanır.
- **Compile**: En uygun yol seçilip sorgunun çalıştırılması için sonraki adıma geçilir.
- **Execute**: Sorgu çalıştırılır ve veriler hazırlanır.

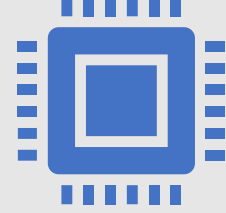
Stored Procedure

- Geliştirdiğimiz uygulamalar da adhoc sorgu yerine SP kullanılması uygulama yönetimini kolaylaştıracaktır.
- Çok fazla kod içeren Spler normal sorgulara göre daha az network trafiği oluşturur.
 - Uygulama ile SQL Server arasında çok satır kod yerine ilgili SP adı ve parametrelerin gönderimi ağ trafiğini azaltır. Çok satır kod içeren ve sık kullanılan SP lerde bu avantaj performansı ciddi oranda arttırır.
- Kod aynı şekilde veya farklı parametrelerle tekrar tekrar kullanılabilir.
- SP'lerin içeriği kullanıcılardan gizlenerek (information hiding) kullanıcının sadece SP'nin yaptığı işle ilgilenmesi sağlanır.
- SP kullanılmadan geliştirilen uygulamalarda SQL Injection sistem açıkları da beraberinde gelebilir.
- SP'ler her zaman normal sorgulara göre daha performanslı çalışmayabilir (optimize adımındaki değişikliklerden dolayı) ama daha stabil çalışacaktır.

Stored Procedure



SQL' de vazgeçilmez veri tabanı
objelerindendir.



Database server ında tutulan ve ilk
derlemeden sonra bir daha **derlenmeye**
ihtiyaç duyulmayan SQL ifadeleridir.



Kısaca Derlenmiş SQL cümlecığı diyebiliriz.



Programlama dillerinde olduğu gibi
parametreler içerebilirler. Bu parametrelere
göre çalışıp sonuçları listeler.

Stored Procedure Özellikleri

Bir SP içinde başka SP'ler çağırılabilir.

Veri tabanlarında saklandığından dolayı **daha hızlı** çalışırlar.

Normal kod ile çağırarak yerine kodla sadece SP'ye parametre gönderip çağırarak performans açısından daha verimlidir.

Güvenli bağlantı oluşturmak için parametrik SP kullanılabilir.

SP'ler ilk çalıştıklarında derlenirler. Tekrar çalıştıklarında derlenmezler. Bu işlemde performans açısından güzel bir olaydır. Halbuki Uygulama katmanında yazılmış olan SQL kodu her çağırma da tekrar tekrar derlenmek zorundadır.

Stored Procedure Özellikleri

SQL komutu çağırıldığında ayrıştırma, derleme ve çalıştırma aşamalarından geçmektedir. SP'ler ağ trafiğini de yormayan yapısıyla programlamada çok kullanılan bir kod parçacıdır.

Stored procedure ler yazılırken programlama ifadelerine ek olarak (if, while, for vb.) T-SQL komutlarını da kullanabilirsiniz.

Stored Procedure Tipleri



EXTENDED STORED PROCEDURE:

GENELLİKLE *.DLL ŞEKLİNDE
PROSEDÜRLERDİR.



CLR STORED PROCEDURE:

SQL SERVER 2005 SONRASINDA CLR
ORTAMINDA HERHANGİ BİR DİLİ KULLANARAK
KODLANAN SP'LERDİR.



SYSTEM STORED PROCEDURE:

GENELLİKLE SP_ ÖN EKİYLE BAŞLARLAR VE
HEPSİ MASTER VERİ TABANINDA TUTULAN
SP'LERDİR.



KULLANICI TANIMLI SP:

KULLANICILARIN TANIMLADIĞI SP'LERDİR.
BU SP LERİ ÇOK RAHAT OLUŞTURUP SİLEBİLİR
VE DEĞİŞTİREBİLİRİZ.

Stored Procedure Genel Yapısı

```
CREATE PROCEDURE Procedureismi  
(  
    @Parametre1 veritipi,  
    @Parametre2 veritipi,  
    ...  
)  
As  
SQL komutları (yani yapılacak işlemler)
```

Stored Procedure Çalıştırma

- Exec ve Execute deyimi ile çalıştırılır.
- Örneğin :
- Exec Procedureismi (varsa parametre)



Trigger

- **Trigger** Türkçe anlamı olarak **Tetikleyici** demektir.
- Trigger, Stored Procedureler gibi Sql Server içinde bileşen olarak bulunmaktadır. Trigger'lar (tetikleyiciler), veri ya da sistemle ilgili değişimlerde otomatik olarak tetiklenen TSQL kod bloklarıdır.
- Triggerlar, tablo üzerinde bir işlem gerçekleştiğinde (**DDL**: Create, Alter, Drop **DML**: insert, update, delete) başka bir işlem daha yapılmak istendiği zaman kullanılır.
- Trigger'ların Stored Procedure'lerden farkı; dışarıdan parametre almaması, dışarıya parametre göndermemesi ve bir kullanıcı tarafından değil, bir olay tarafından tetiklenmesidir. (Procedure exec komutu ile çalışır)

Trigger Özellikleri

- **For - After Kavramları**
- Kullanacağımız ilk komut ya **for** ya da **after** dır.

For: Trigger'ın o an çalışması için kullanılır.

After: Tablo üzerinde her ne yapıyorsanız, işleminizi yaptıktan sonra trigger çalışmasını sağlar.

For veya after seçildikten sonra DML işlemleri seçilir. Yani tablo üzerinde ne yapılacaksa o işlem seçilir. Trigger ona göre çalışacaktır.





Trigger Çeşitleri




- **Ardı Sıra Tetikleyiciler (After Triggers)**

Bir tabloya **UPDATE**, **INSERT** veya **DELETE** işlemi yapıldıktan sonra bir takım işlemlerin yapılması için kullanılan tetikleyicilere **Ardı Sıra Tetikleyici** denir. Bu tür tetikleyiciler pek çok değişik iş yapabilirler. Bir başka tabloya veri girişi yapmak veya tabloyu güncellemek, tablolar arasında uyumu sağlamak için bu tür tetikleyiciler çok uygundur.

- **Yerine Tetikleyiciler(INSTEAD OF TRIGGER)**

Bir INSERT, UPDATE veya DELETE işlemi bir tabloya uygulandığında bu tablo üzerinde , sırasıyla bir Instead Of INSERT, Instead Of UPDATE veya Instead Of DELETE tetikleyici varsa bu işlem tablo üzerinde **gerçekleşmez**. Onun yerine tetikleyici içinde yazılı kodlar yapılır.



Trigger Genel Yapısı

```
CREATE TRIGGER trigger_adi  
ON tablo_adi  
{FOR|AFTER|INSTEAD OF} {INSERT|UPDATE|DELETE}  
AS  
Begin  
        --Sql ifadeler  
End
```

Örnek

- Bir tabloyu silmeye çalıştığımızda o tablonun silinmemesini, uyarı mesajı verilmesini sağlayalım. Tabloyu "sildiğimiz anda" uyarı mesajı verilir, buradaki tetikleyici olay tabloyu silmek olur. Tetikleyici hangi tabloda ise o tablo üzerinde çalışır.
- create trigger Sildirme
on Urunler
for Delete
as
Print ('Bu tabloyu silemezsiniz!')
rollback transaction -- yaptığı işlemi geri aldırır
- Delete From Urunler

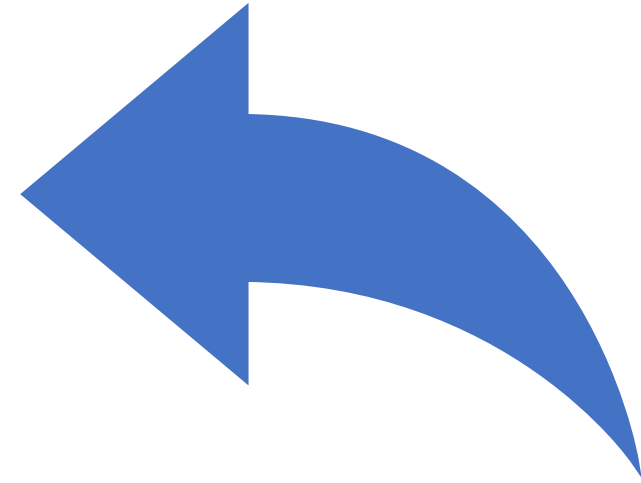
Trigger Durumları

- **Enable (Aktifleştirme):**

enable **trigger** trigger_adı **On** tablo_adı

- **Disable (Pasifleştirme):**

disable **trigger** trigger_adı **On** tablo_adı



Trigger Durumları

- Tüm Triggerları Pasifleştirme / Aktifleştirme

enable **trigger all On** tablo_adı

disable **trigger all On** tablo_adı

- Trigger Kaldırma

drop trigger trigger_adı



Referanslar

«SQL Server 2017» İsmail Adar