DATABASE

Öğr. Gör. Evgin GÖÇERİ

3 Kasım 2014

VIEW

View Nedir?

Gerçekte var olmayan, SELECT ifadeleri ile tanımlanmış sanal tablolardır

Ne için View kullanılır?

- View'ler, sorguları basitleştirmek,
- Erişim izinlerini düzenlemek,
- Farklı sunuculardaki eş değer verileri karşılaştırmak veya bazı durumlarda sorgu süresini kısaltmak için kullanılır

View ile Çalışmak - 1

- View'lere çeşitli kısıtlar dahilinde yeni kayıt eklenebilir, kayıt silinebilir veya güncelleme yapılabilir
- Tabloları olduklarından farklı gösterecek filtrelere ihtiyaç duyulduğunda view'lar kullanılır
- View'lar kaydedilmiş sorgulardan ibarettir
- Tablo gibi kullanılsalarda, gerçekte öyle bir tablo yoktur
- View'lar bir tabloymuş gibi sorgulanabilir

View ile Çalışmak - 2

- Bir view ile verilerin çekildiği gerçek tabloya *base table* (temel tablo) denir
- Bir view, temel tablonun bütün satır veya sütunlarını, veya bazı satırlarını kapsayabileceği gibi, birden fazla temel tablo üzerinde de tanımlanabilir
- SQL Server'da, view'ların mutlaka tablolara dayalı olması bir zorunluluk değildir (Bir view, başka bir view üstünde de tanımlanabilir)
- SQL Server 2000'den itibaren, view'lar üstünde Clustered Indeks tanımlanabilir. Bu sayede, view'daki veriler ayrı bir nesne olarak derlenir ve performans artar

View'ler şu görevler için kullanılır:

1. Kullanıcıların bazı kritik tabloların sadece belli sütunlarını görmesi gerektiğinde

2. Kullanıcıların, çeşitli birim dönüşümlerinden geçmiş değerler görmeleri gerektiğinde

3. Halihazırdaki tablolarda var olan verilerin başka bir tablo formatında sunulması gerektiğinde

4. Çok karmaşık sorguları basitleştirmek için

View İçin Alternatifler

- View'ları genel olarak, tasarlanmış tablolarımızı farklı şekilde gösterebilmek için kullanırız. Fakat, bazı durumlarda (view yetersiz kaldığında) view yerine şu nesneler de kullanılabilir:
  - Bazen view'ların parametrik olanına gereksinim duyulabilir. Bu durumda view yerine kullanıcı tanımlı fonksiyonlar kullanılabilir
  - Bazen view'leri üstünde SELECT yapmaktan ziyade, bir ifadeyi kısaltmak için kullanmak gerekebilir. Bu ifadenin dışarıdan parametre alması veya içinde if-else yapısının kullanılmasına ihtiyaç duyulabilir. Bu durumda view yerine stored procedure(saklı yordam) kullanılır (Stored procedure'ler çalıştırılır; View'lar ise sorgulanırlar. Hangisinin kullanılacağına karar verilmesinde bu noktaya dikkat edilmesi gerekir)
  - View'lar sadece tabloları adlarına bir isim vermek üzere kullanılabilirler. Başka türden nesne adlarına isim verilecekse, synonymus nesnesi tanımlamak düşünülebilir (Bir synonymus nesnesi, herhangi bir veritabanı nesnesi için, o nesneye isim vermek(lakap takmak) için kullanılabilir)

View Oluşturmak

```
CREATE VIEW view_adi
 AS
 SELECT sutun_adlari
 FROM base_tablo_adi
```

Not: View'ler SELECT ifadesinde ORDER BY almazlar

Örnek:

- Dukkan veritabanı içindeki tblUrun tablosu üstünde, dolar ile satılan ürünlerin urunKod, urunAd, listeFiyat sütunlarını içeren bir tablonun görevini yerine getirecek bir vwDolarUrunler adında view ;

```
USE dukkan
 GO
 CREATE VIEW vwDolarUrunler    AS
  SELECT urunKod, urunAd, listeFiyat
  FROM tblUrun
  WHERE dovizKod='USD'
  GO
```

```
SELECT  * FROM vwDolarUrunler
```

Not: View'lerde sütun adları, özellikle belirtilmediği sürece base tablolardaki sütun adları ile aynı olur. Farklı bir sütun adına ihtiyaç duyulduğunda bunu kendimiz belirtebiliriz. View'larda sütunlar için veri tipi belirtilmez, çünkü veri tipi, SELECT ifadesini takip eden sütun tipi ile aynıdır

Örnek: Sütun adları belirttiğimiz bir view;

```
CREATE VIEW vwDolarUrunlerSutunAdli(URUNUN_KODU, URUNUN_ADI, URUNUN_FIYATI)
 AS
 SELECT urunKod, urunAd, listeFiyat
 FROM tblUrun
 WHERE dovizKod='USD'
 GO
```

İzinler ve Sahiplik

- View'in üstünde tanımlandığı veritabanına erişim hakkı olmayan kullanıcılar, view'e izin verilmediği sürece erişemezler
- Veritabanında izni olanlar miras yönetiminden dolayı view'e de erişebilir
- Bir view oluşturulduktan sonra kullanıcılar için izin tanımlanabilir. Bu durumda dikkat edilmesi gereken konu; Kullanıcılara izin verilirken temel tabloya asla erişim izni vermemek ve sadece view'a erişim tanımlamaktır
- Tablonun sahibi (genellikle dbo) ile view'in sahibinin aynı olması faydalıdır. Çünkü böyle olursa, SQL Server sadece kullanıcıların, sadece view'e erişim izni olup olmadığını sorgulamak durumundadır

İzinler ve Sahiplik - 2

- Sahibi olmadığınız bir tablonun, üstünde bulunduğu veritabanında size CREATE VIEW izni verilmiş ise bu tabloya bağlı bir view tanımlayabilirsiniz. Bu durumda temel tablo ile ona bağlı olan view sahipleri farklı olacaktır.

  Bu durumda, SQL Server, view'e bir kullanıcı erişmek istediğinde iki şeyi sorgulamak durumundadır:

  1. Kullanıcının view'e erişim izni var mı
  2. Kullanıcıların view'e erişim izni varsa, view'in sahibi kullanıcının temel tablolarına erişim izni var mı.

  Her bir erişim için teker teker izinlerin kontrol edilmesi SQL Server'e fazladan iş olarak yansıyacaktır

İzinler ve Sahiplik - 3

- Bir view'in hangi tablolara bağlı olduğunu ve bu tabloların sahiplerinin kim olduğunu sp_depends sistem stored procedure'ü ile öğrenmek mümkündür

- View ile temel tablolar (base tables) arasında sahiplik zincirinin kopması bazı durumlarda kullanıcıların verilere erişimini engelleyebilmektedir. Bu durumda, view silinip tekrar temel tablonun sahibi tarafından oluşturulmalı ve izinler yeniden tanımlanmalıdır.

- Genellikle SQL Server'daki nesnelerin sahibi dbo'dur. Bu nedenle, view oluştururken de sahibini dbo olarak belirlemek sonradan doğacak karmaşaları engeller

Birden Fazla Tablodan Veri Çeken View

- Genel olarak, parçalı bir view oluşturmak için şu kalıp kullanılır:

```
CREATE VIEW view_ismi (sutun_ismi,...)
 AS
SELECT sutun_ismi,…
FROM tablo1 JOIN tablo2
ON birleştirme şartları
```

Örnek:

- Dükkan veritabanında bir müşterinin kaç ürün aldığı ile ilgili bir view:
  - Müşterilerin bilgileri tblKullanici tablosunda tutulmaktadır. Fakat hangi siparişin kime ait olduğu tblSiparis tablosunda ve her bir sipariş için hangi üründen kaç adet alındığı da tblSiparisDetay tablosunda yer almaktadır.

Bu amaç için ihtiyaç duyulan view şöyle tanımlanır:

```
CREATE VIEW kisiBasiSatis
 AS
SELECT K.kullaniciKod, isim, soyad, SUM(adet) as ToplamUrun
FROM tblKullanici K LEFT JOIN tblSiparis S
ON S.kullaniciKod=K.KullaniciKod
 LEFT JOIN tblSiparisDetay SD
 ON SD.FaturaKod=S.FaturaKod
 GROUP BY K.KullaniciKod, K.isim, K.Soyad
```

Bir View Üstünde Değişiklik Yapmak

- Bir view üstünde değişiklik yapmak için ALTER deyimi şöyle kullanılır:

```
ALTER VIEW view_adi
 WITH seçenekler
 AS
 SELECT ifadesi
```

- Seçenekler kısmına ENCRYPTION veya SCHEMABINDING kelimelerinden biri veya bir kaçı aynı anda aralarında virgül olmak üzere kullanılabilir

Not: Bir view silip yeniden oluşturduğumuzda, üstündeki izin ve engellemeleri yeniden ayarlamak gerekir. Fakat bir view ALTER deyimi ile değiştirilirse, hakların yeniden verilmesine gerek yoktur

Tanımlanan View'ları Görmek

Bir veritabanında hangi view'ların tanımlandığını görmek için sistem view'lardan faydalanırız

- Bütün view'ların kaynak kodlarını görmek için, Örnek;

```
SELECT *
FROM INFORMATION_SCHEMA.VIEWS
```

- View'ların listesini almak için sys.views kataloğu sorgulanabilir ;

```
SELECT * FROM sys.views
```

View Tanımlarını Gizlemek

- Bazı durumlarda view oluştururken kaynak kodlarının başkaları tarafından görülmesini istemeyiz. Bu durumda, SQL Server kaynak kodları şifreleyebilir
- Dikkat edilmesi gereken nokta, şifrelenmiş bir view'in kaynak koduna biz de dahil hiç kimsenin ulaşamayacağıdır.
- Bir view şifrelenmeden önce kaynak kodunun bir kopyasını kaydetmek daha sonradan doğacak aksaklıkları önlemede yardımcı olur
- View şifrelemek için ENCYRPTION operatörü kullanılır.
- Bir view ilk oluşturulduğunda şifrelenebileceği gibi, daha sonradan da şifreli hale getirilebilir

View Tanımlarını Gizlemek - 2

- Şifreli bir view oluşturmak;

```
CREATE VIEW view_adi
 WITH ENCYRPTION
 AS
 SELECT ifadesi
```

- Oluşturulmuş bir view şifreli hale getirmek için;

```
ALTER VIEW view_adi
 WITH ENCYRPTION
 AS
 SELECT ifadesi
```

Örnek:

```
ALTER VIEW kisiBasiSatis
WITH ENCRYPTION
 AS
SELECT K.kullaniciKod, isim, soyad, SUM(adet) as ToplamUrun
FROM tblKullanici K LEFT JOIN tblSiparis S
ON S.kullaniciKod=K.KullaniciKod
 LEFT JOIN tblSiparisDetay SD
 ON SD.FaturaKod=S.FaturaKod
 GROUP BY K.KullaniciKod, K.isim, K.Soyad
```

- Daha sonra kaynak kodu görmek için

  sp_helptext kisiBasiSatis

  denildiğinde, şöyle bir sonuç ile karşılaşılır:

The text for object 'kisiBasiSatis' is encrypted

View Temel Tablolarının Şemasını Kilitlemek : SCHEMABINDING

- SCHEMABINDING operatörünün görevi, view'a ait olan temel tabloların şemalarında değişiklik yapılmasını engellemek için kilitleme sağlamak
- Bu operatör view'in tanımında olduğu sürece, view'e ait temel tablolar üstünde seçilen bir sütun silinemez, türü değiştirilemez gibi şekilsel değişikliklere müsaade edilmez
- Şemada bir değişiklik yapmak için bu operatörün tanımdan çıkarılması (view'in ALTER ile değiştirilmesi) gerekir

View ile INSERT, UPDATE, DELETE İfadeleri

- View'lar gerçek anlamdaki tablolar değildir. Sadece SELECT ifadesinden ibaret olan görüntülerdir.
- Bazı VTYS'lerde sorgu anlamına gelen "query" deyimi ile de anılabilir
- Bir view gerçekte olmasa da veri seçmenin yanı sıra veriyi güncelleme, ekleme, silme gibi amaçlarla da kullanılabilir
- View'ler bünyesinde veri tutmadıklarından ilgili değişiklikleri temel tablo üstünden yansıtırlar

View ile Veri Değişikliği Yapılması Durumundaki Kısıtlamalar

View'ler gerçek tablo olmadığından veri değişikliği yapılırken bazı kısıtlamalar vardır. Bunlar:

1. Bir view aynı anda sadece bir tek tablo üstünde veri değiştirebilir.

   Birden fazla tablodan kayıt çeken view'lar üstünde veri değişikliği yapılacaksa, aynı anda tek bir tablo etiketlenecek şekilde parçalara ayrılmalıdır. (Veya INSTEAD OF trigger'ları oluşturulup kullanılabilir)

ÖRNEK:

```
CREATE VIEW vw_Urunler
 AS
SELECT U.urunKod, U.urunAd, M.marka
FROM tblUrun U JOIN tblMarka M
ON M.markaKod = U.markaKod
```

Bu view'e bir veri eklemek istendiğinde, Örnek;

```
INSERT INTO vw_Urunler(urunKod, urunAd, marka)
VALUES (1000, 'Elbise', 'Ali Aksu ModaEvi')
```

Bize bu ifadenin birden fazla tabloyu etkilediğini belirten şu hata verilecektir:

"vwUrunler" is not updatable because the modification affects multiple tables

View ile Veri Değişikliği Yapılması Durumundaki Kısıtlamalar - 2

2. Bir view'in temel tablo üstünde değişiklik yapılması durumunda;

Temel tabloda NULL olmayan sütunlar varsa ve bu sütunlar view'da yer almıyorsa, view'e veri eklemek istendiğinde bu sütunlar NULL geçilemeyeceği için, veri ekleme işleminde hata meydana gelecektir

ÖRNEK: INSERT INTO vwUrunler (urunKod, urunAd)
VALUES (1000, 'Elbise')

Yukarıdaki ifade hata mesajı verir. Çünkü, urunKod, tblurun tablosunda bir IDENTITY alandır. Buraya veri ekleme ve değiştirme işleminde müdahele edilmez. Bu nedenle, bu ifadenin şöyle değiştirilmesi gerekir:

```
INSERT INTO vwUrunler (urunAd)
VALUES ('Elbise')
```

Görüldüğü gibi; Bir view üstünden veri değişimi yapılabilmesi için 2 kural geçerlidir:

1. Bir seferde tek bir tabloyu muhatab alacak SQL cümleleri kurulmalıdır
2. SQL cümleleri IDENTITY, Unique indeks gibi sütunlar üstünde tanımlanan veri bütünlüğü öğelerine takılmamalıdır

View ile Veri Değişikliği Yapılması Durumundaki Kısıtlamalar - 3

3. Yatay view'lerde WHERE cümleciği ile filtrelenmiş bir kaydın, UPDATE cümleciği ile filtre dışına çıkarılması olasıdır. Aynı şekilde, yatay view'in seçemeyeceği bir kaydı eklemesi de olasıdır. Bu türden bir durumun önüne geçmek için, view tanımında CHECK OPTION operatörü kullanılır (Verilerin güvenliği açısından bu operatör koruma sağlar)

ÖRNEK: 100 birim fiyattan daha düşük fiyatı olan ürünleri göstermek üzere, vw_100birimAlti adında bir view şu şekilde oluşturalım;

```
CREATE VIEW vw_100birimAlti
AS
SELECT urunKod, UrunAd, urunDurum, listeFiyat
FROM tblUrun
WHERE listeFiyat<100
```

Stajyer ürün yöneticilerin sadece bu ürünlere erişmelerini sağlayacağımı varsayalım. Fakat bir stajyer şu şekilde ürün eklerde ne olur ?

```
INSERT INTO vw_100birimAlti (urunAd, urunDurum, listeFiyat)
    VALUES ('Staj Defteri', 1, 2500)
```

Bu view ile bir ekleme yapılır. Fakat stajyer eklediği kaydı göremez. Stajyer'in göremeyeceği kayıtlar ekleyememesi, görebildiği kayıtları da UPDATE ile göremeyeceği kayıtlar haline çevirememesi için, view'i şu şekilde değiştirmeliyiz :

View ile Veri Değişikliği Yapılması Durumundaki Kısıtlamalar - 3

```
ALTER VIEW vw_100birimAlti
AS
SELECT urunKod, UrunAd, urunDurum, listeFiyat
FROM tblUrun
WHERE listeFiyat<100
WITH CHECK OPTION
```

Burada view yapısına “WITH CHECK OPTION” eklenmiştir. Bu ifade view sonuna yazılır. Kısıtlayıcı bir tanımdır

STORED PROCEDURE
(SAKLI YORDAM)

Giriş

- Nesneye dayalı programlama bu kadar popüler değil iken, programlar sadece yordam yada fonksiyon denilen parçacıklardan oluşuyordu
- Her bir yordam, işlevi yerine getirmek için özellikle yapılandırılmış program parçacığıdır
- Bir yordam, başka bir yordam içerisinden çağrılabilir. Bu da sık kullanılan işlemler için yazılmış kodların, bir defa yazılıp çok defa kullanılmasını, böylelikle programlamayı kolaylaştırmayı amaçlar
- Fakat bunun dışında Saklı Yordamlar veritabanı programlama için güvenlik elemanı olarak da kullanılmaktadır

Giriş

- Saklı Yordamlar (SY) bir çok gelişmiş programlama dilindeki fonksiyon yapılarına karşılık gelir. Birden fazla işlemi bir tek komut ile çalıştırmamız gerektiğinde saklı yordamlar kullanılır

- SY'ların en önemli özelliği, sorguların önceden hazırlanması (derlenmesi) ve VTYS ile aynı uzayda çalışmasından dolayı daha hızlı sonuç vermesidir.

- Bir SY oluşturulduktan sonra veritabanı sunucunda saklanır. Her ihtiyaç duyulduğunda aynı SY defalarca çağrılabilir

- Bir saklı yordam, başka bir saklı yordam tarafından çağrılabilir. Fakat en fazla 32 seviyeye kadar SY'ların başka bir SY tarafından çağrılmasına müsaade edilir

SY'ların Oluşturulma Biçimine Göre Türleri

- SY'lar oluşturulma şekline göre 4 tiptir :

1. **Extended Stored Procedure :**
   - Genellikle *.dll şeklinde derlenmiş yordamlardır.
   - C, C++ gibi dillerle yazılıp dll'e derlenirler
   - Sistemde hazır gelen extended procedure'ler genellikle xp_ öneki ile başlarlar (fakat sp_ öneki ile başlayan sistem extended procedure'lerde vardır)

(SQL server yeni versiyonlarında bu SY'lar kaldırılacaktır)

2. **CLR (Common Language Runtime) Stored Procedure :**
   - SQL 2005'den itibaren herhangi bir dil kullanarak saklı yordamlar kodlayabilirsiniz

3. **Sistem Stored Procedure :**
   - Genellikle sp_ öneki ile başlar.

4. **Kullanıcı Tanımlı Stored Procedure :**
   - Programcıların programladığı SY'lar olup, geçerlilik sahasına göre 3 tipten birinde olabilir; Geçici, yerel ve uzak sotred procedure
   - Geçici stored procedure, SQL server'in eski sürümlerinde kullanılabilen bir tür olup, her oturumda derlenmesi gerekir.
   - Yerel stored procedure, SY'lara verilen genel bir addır
   - Uzak stored procedure ise, bir dağıtık modelde uzaktaki bir sunucuda yer alan SY'lardır

SQL Server bünyesindeki CLR'in sağladığı faydalar

- İş ve veri erişim katmanı birbiri ile benzer mimariyi kullandığından, yük dengelemek için kodlar aşağı ve yukarı kaydırılabilir

- Harici kaynaklara erişme gerekliliği durumunda C++ gibi bir dil ile Extended Stored Procedure yazma zorunluluğunu ortadan kaldırmıştır.

- CLR tip güvenli bir çalışma ortamı olduğundan, programların çalışma güvenliği sağlanmış olur

- .NET'in sunduğu geniş dil desteğinden programcılar faydalanarak, yeni bir syntax ile boğuşmaksızın, kendi iyi oldukları dilde, performans kriterleri çerçevesinde kendi istedikleri veritabanı öğelerini programlayabilirler

- Daha iyi kod yönetimi için, assembly'ler, namespace'ler gibi üst katmandan bilindik organize edici bileşenlerin veritabanı nesneleri için de kullanımını sağlar

SQL Server içerisinde CLR

• Yönetilmiş program, programcıyı bir çok yükten kurtaran ortam demektir. Bu yüklerden en belirgini, hafıza yönetimi ve kod güvenliği gibi konularla çalışma zamanı ortamının ilgilenmesidir

• CLR, bir yönetilmiş ortamdır. Bu nedenle, CLR'da çalışmak üzere yazılan kodlara yönetilmiş kod denir

• Bir uygulamanın CLR barındırması, .NET çalışma zamanını yükleyen ve yönetilebilir kodları çalıştırabilen bir işlev içermesinden ibarettir.

• SQL Server, Internet Explorer veya ASP.NET gibi özelleşmiş bir tür CLR barındırır.

• SQL Server, CLR kodlarını derlerken Tembel Yükleme (Lazy Loading) adı verilen bir mekanizma kullanır. Bu mekanizmaya göre, eğer hiç yönetilmiş koda atıfta bulunacak bir programsal öğe kullanılmadıysa, çalışma zamanı (CLR) SQL server tarafından hiç yüklenmez

SQL Server içerisinde CLR - 2

Not: SQL Server, CLR dışındaki tüm alanlarda, okuma ve yazmaya kilitleme, hafıza yerleşimi gibi konularda işletim sisteminden bağımsız olarak kaynaklarını kendisi yönetir. (Çünkü SQL Server için en önemli kriter performanstır). Hafıza ve bellek arasında verilerin optimum gidiş-gelişlerini sağlayabilmek için, işletim sisteminin dışında kendisine ait hafızayı doğrudan denetler. Bu nedenle kendisine özgü, performansa dayalı bir hafıza ve risk yönetim mimarisi vardır.

SAKLI YORDAM OLUŞTURULMASI VE ÇALIŞTIRILMASI