

T.C.
MİLLÎ EĞİTİM BAKANLIĞI



MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

VERİ BÜTÜNLÜĞÜ

ANKARA 2008

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).
- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.
- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.
- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşılabilirler.
- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.
- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR	iii
GİRİŞ	1
ÖĞRENME FAALİYETİ - 1	3
1. VERİ KISITLAMALARI	3
1.1. Veri Bütünlüğü.....	3
1.2. Tanımlanabilir Veri Bütünlüğü.....	3
1.3. Tanımlanabilir Veri Bütünlüğü Esasları	4
1.3.1. Satır Bütünlüğü (Entity)	4
1.3.2. Sütun Bütünlüğü (Domain).....	4
1.3.3. Başvuru Bütünlüğü	4
1.3.4. Kullanıcı Tanımlı Bütünlük.....	4
1.4. Constraint (Kısıtlayıcı).....	4
1.5. Constraint Türleri	5
1.5.1. Primary Key Constraint	5
1.5.2. Unique Constraint.....	5
1.5.3. Foreign Key Constraint.....	5
1.5.4. Default Constraint.....	5
1.5.5. Check Constraint	5
1.6. Sütun Seviyeli Kısıtlamalar	5
1.6.1. Primary Key Constraint Oluşturmak	6
1.6.2. Unique Constraint Oluşturmak	6
1.6.3. Default Constraint Oluşturmak.....	6
1.6.4. Check Constraint Oluşturmak.....	7
1.7. Default Nesnesi Oluşturmak	8
1.8. Rule Oluşturmak	9
1.9. Tablo Seviyeli Kısıtlamalar	9
1.9.1. Sütunlar Arası Check Constraint Oluşturmak	9
1.9.2. Foreign Key Constraint Oluşturmak.....	10
1.10. Constraint'leri Düzenlemek	13
1.11. T_SQL ile Tablodaki Constraint'lerin Gösterimi	14
1.12. Constraint'leri Silmek	15
1.13. Constraint'lerin Kullanılıp Kullanılmaması.....	15
UYGULAMA FAALİYETİ	16
ÖLÇME VE DEĞERLENDİRME	18
ÖĞRENME FAALİYETİ - 2	19
2. TRIGGER VE SAKLI PROSEDÜRLER	19
2.1. Trigger.....	19
2.2. Stored Procedure	21
2.2.1. Extended Stored Procedure.....	21
2.2.2. CLR Stored Procedure	21
2.2.3. System Stored Procedure	22
2.2.4. Kullanıcı Tanımlı Stored Procedure	22
2.3. Çalıştırılma Aşamaları	22
2.4. Stored Procedure Oluşturmak	22
2.4.1. NOCOUNT Parametresi.....	26
2.4.2. Stored Procedure'ün Kullanım Hakkı.....	26

2.5. Değişiklik Yapmak	27
2.5.1. Management Studio ile Değişiklik Yapmak	27
2.5.2. “sp_helptext” İfadesi ile Değişiklik Yapmak	29
2.6. Stored Procedure’ü Silmek	31
2.7. Değer Alan Stored Procedure’ler	31
2.8. Değer Alıp-Veren Stored Procedure’ler.....	34
2.9. RETURN Deyimi.....	36
UYGULAMA FAALİYETİ	38
ÖLÇME VE DEĞERLENDİRME	40
MODÜL DEĞERLENDİRME	41
CEVAP ANAHTARLARI.....	42
KAYNAKÇA	43

AÇIKLAMALAR

KOD	481BB0045
ALAN	Bilişim Teknolojileri
DAL/MESLEK	Veri Tabanı Programcılığı
MODÜLÜN ADI	Veri Bütünlüğü
MODÜLÜN TANIMI	SQL Server’da veri bütünlüğü veya kısıtlamaları ile ilgili öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	T – SQL modülünü bitirmiş olmak
YETERLİK	
MODÜLÜN AMACI	Genel Amaç Gerekli ortam sağlandığında, veri kısıtlamaları yapabilecek ve yapılan kısıtlamaları yönetebileceksiniz.. Amaçlar 1. Veri kısıtlamalarını kullanabileceksiniz. 2. Kısıtlanan verileri yönetebileceksiniz. 3. Saklı prosedürleri kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Ortam Atölye, laboratuvar, bilgi teknolojileri ortamı (İnternet) vb. kendi kendinize veya grupla çalışabileceğiniz tüm ortamlar. Donanım Ağ veri tabanını çalıştırabilecek yeterlikte bilgisayar, yedekleme için gerekli donanım (CD yazıcı, flash bellek), raporlama için yazıcı, kağıt ve kalem
ÖLÇME VE DEĞERLENDİRME	Modülün içinde yer alan her öğrenme faaliyetinden sonra verilen ölçme araçları ile kendinizi ve modül sonunda ise, bilgi ve beceriyi belirlemek amacıyla, öğretmeniniz tarafından belirlenecek ölçme aracıyla değerlendirileceksiniz.

GİRİŞ

Sevgili Öğrenci,

Okul yaşantınızda öğreneceğiniz her konu, yaptığınız uygulama ve tamamladığınız her modül bilgi dağarcığınızı geliştirecek ve ilerde atılacağınız iş yaşantınızda size başarı olarak geri dönecektir. Eğitim sürecinde daha öz verili çalışır ve çalışma disiplini kazanırsanız; başarılı olmamanız için hiçbir neden yoktur.

Son yıllarda yapılan birçok proje, çok sayıda bilgisayar tarafından kullanılacak şekilde tasarlanmaktadır. Bu yüzden, ağ ortamında birden fazla kullanıcı aynı proje üzerinde çalışabilmektedir. Bu işlemleri çok sık kullandığınız veri tabanı programıyla da yapabilmenize rağmen ağ ortamında güvenlik ve hızlı erişim açısından en iyi sonucu veren SQL Server veri tabanı da yapabilirsiniz. Bu programla, milyonlarca kaydın olduğu tablolar üzerinde işlem yaparken tüm kullanıcılara hitap edebilmektedir. İstenilen sorgu sonuçlarını da en hızlı şekilde elde edebilmenizi sağlar.

Bu modülle, SQL Server'da veri bütünlüğünü veya kısıtlamaları sağlayarak oluşturulan bu kısıtlamaları yönetmeyi ayrıca, saklı prosedür tanımlamayı ve sorgu içerisinde kullanabilmeyi öğreneceksiniz.

ÖĞRENME FAALİYETİ-1

AMAÇ

SQL Server’da veri kısıtlamalarını kullanmayı öğreneceksiniz.

ARAŞTIRMA

- SQL Server’da daha önceden tablo oluştururken sütunlara verdiğiniz özellikleri araştırınız.

1. VERİ KISITLAMALARI

1.1. Veri Bütünlüğü

Veri bütünlüğü, bir tabloda veri güncelleme, silme veya ekleme gibi işlemler yapılırken diğer tablo ya da tablolardaki verilerin birbirleriyle uyum içinde olması, dolayısıyla veri tutarlılığının kaybolmamasının garanti altına alınması demektir.

Veri bütünlüğünü sağlamak iki yöntemle olabilir:

- **Tanımlanabilir veri bütünlüğü:** Tanımlanan nesnelerin kendi özellikleri sayesinde sağlanabilen veri bütünlüğüdür.
- **Prosedürel (Programsal) veri bütünlüğü:** Bir programlama mantığıyla bütünlüğün tasarlanması gerekir. SQL’de bu yaklaşım ise Trigger (tetikleyiciler), Stored Procedure (saklı yordamlar) veya programcı kodlarıyla yapılır.

1.2. Tanımlanabilir Veri Bütünlüğü

Nesnelerin kendi tanımları dolayısıyla elde edilen veri bütünlüğüne tanımlanabilir veri bütünlüğü denir.

Bu veri bütünlüğünün; Constraint’ler (kısıtlayıcılar), Rule’lar (kurallar) ve Default’lar (varsayılanlar) olmak üzere üç çeşidi vardır.

Tanımlanabilir veri bütünlüğü, prosedürel veri bütünlüğüne göre daha kullanışlı ve denetlenebilirdir; ancak, tanımlanabilir veri bütünlüğünün kullanılmadığı durumlarda prosedürel veri bütünlüğü kullanılır.

1.3. Tanımlanabilir Veri Bütünlüğü Esasları

Tanımlanabilir veri bütünlüğü esaslarını dört başlık altında toplayabiliriz.

1.3.1. Satır Bütünlüğü (Entity)

Tabloya girilen kayıtlardan her bir kaydın diğer kayıtlardan farklı bir değer olmasını sağlayan bütünlüktür.

1.3.2. Sütun Bütünlüğü (Domain)

Tablodaki herhangi bir sütunun hangi gruptan verileri alabileceğini veya NULL bir değer alıp alamayacağını sağlayan bütünlüktür.

1.3.3. Başvuru Bütünlüğü

- Bir birincil anahtarlı ve bir yabancı anahtarlı birbiriyle ilişkili iki tablo olduğunu düşününüz.
- Yabancı anahtarlı tablodan bir kaydın silinemediği durumda, birincil anahtarlı tablodan birincil anahtar aynı olan kayıtların silinemeyeceğini sağlayan bütünlüktür.

Örnek:

Bilgisayarınızın kasasında bir anakart, TV kartı ve ekran kartı olduğunu varsayın. TV kartı ve ekran kartı kasadan çıkarılmadan ana kart kasadan çıkarılamayacaktır.

1.3.4. Kullanıcı Tanımlı Bütünlük

Diğer bütünlüklere uymayan ve kod yazılarak oluşturulan bütünlüktür.

1.4. Constraint (Kısıtlayıcı)

Veri üzerindeki mantıksal sınırlamalara **kısıt** adı verilir. Kısıtların genel olması tercih edilen bir durumdur.

Kısıtlar, veri modellerinde bütünlük sağlamak için kullanılır. Kısıtlamalar, tabloların tanımlanmasıyla beraber oluşan öğelerdir. Kısıtlamalar ile Rule (kural) ve Default'ların (varsayılan) yapabileceği işler yapılabilir.

Constraintler tablo oluştururken yani CREATE TABLE komutuyla tanımlanabilir. Tablo oluşturulmuşsa ALTER TABLE komutuyla bu işlem gerçekleşir. ALTER TABLE komutuyla kullanıldığında sütunlara girilen bilgilerin dikkate alınması gerekir.

1.5. Constraint Türleri

1.5.1. Primary Key Constraint

Birincil anahtar kısıtlayıcı anlamındadır. Aynı olmayan değerler girilmesini sağlar. Bu da her kaydın farklı olması demektir. Her tablonun en fazla 1 adet Primary Key Constraint'i olabilir.

1.5.2. Unique Constraint

Tekil alan kısıtlayıcı anlamındadır. Birincil anahtar olan ve tablodaki diğer alanlar içinde aynı içeriğe sahip verilerin olmaması için Unique Constraint tanımlanır. T.C.Kimlik Nu. primary key ve Okul Nu. Unique şeklinde bir tanımlama Unique Constraint'e bir örnektir.

1.5.3. Foreign Key Constraint

Yabancı anahtar kısıtlayıcı anlamındadır. Bir tablodaki bir sütuna ait verilerin başka bir tablonun belirli bir sütunundan gelmesini denetler.

1.5.4. Default Constraint

Varsayılan kısıtlayıcı anlamındadır. Tablodaki herhangi bir alan için girilmesi gereken bir değer atanmasıdır. INSERT komutu için geçerlidir. Örneğin, kişi bilgilerinin alındığı bir tabloda kişinin uyruğunun girilmesi işleminde varsayılan değer olarak "T.C." atanabilir.

1.5.5. Check Constraint

Kontrol kısıtlayıcı anlamındadır. Belirtilen formata göre verilerin girilmesini sağlar. Örneğin, T.C.Kimlik Nu. alanına 11 karakterin girilmesi Check Constraint ile sağlanabilir.

1.6. Sütun Seviyeli Kısıtlamalar

Tablodaki bir sütun için girilecek bilgileri kontrol etmek için kullanılan kısıtlamalardır. Bir sütunun değer alıp (NULL) almaması (NOT NULL) için kısıtlama yapılabilir. Bu, kısıtlamaya en basit örnektir.

Table - dbo.Kursiyer*			
	Column Name	Data Type	Allow Nulls
▶	Numara	int	<input checked="" type="checkbox"/>
	Ad	varchar(20)	<input type="checkbox"/>
	Soyad	varchar(20)	<input type="checkbox"/>
	e_mail	varchar(15)	<input type="checkbox"/>
			<input type="checkbox"/>

Resim 1.1: Bir sütunun değer alıp almaması (NULL veya NOT NULL)

1.6.1. Primary Key Constraint Oluşturmak

Tablo tasarım aşamasında bir birincil anahtarınız tanımlanmış olması gerekir. Bir tabloda sadece 1 adet Primary Key Constraint tanımlanabilir.

```
CREATE TABLE Okul (  
    OgrNo int,  
    Adi VARCHAR (15),  
    Soyadi VARCHAR (20),  
    Sinifi VARCHAR(10),  
    TcKimlikNo VARCHAR (11),  
    CONSTRAINT PKC_OgrNo PRIMARY KEY (OgrNo)  
)
```

1.6.2. Unique Constraint Oluşturmak

Bir tabloda birden fazla Unique Key olabilir. SQL Server, bunu Unique Index olarak ele alır. Unique Key Constraint şeklinde tanımlanmış bir alan NULL değerler alabilir. Eğer, bu alan NULL olmayacak yani değer alacaksa daha önce girilmiş olan bilgilerden mutlaka farklı değer olmalıdır.

Daha önceden oluşturulmuş bir tablo için Unique Key Constraint tanımı şöyledir.

```
ALTER TABLE Okul  
    ADD CONSTRAINT PK_Okul PRIMAY KEY CLUSTERED (OgrNo),  
    CONSTRAINT UC_TcKimlikNo UNIQUE (TcKimlikNo)
```

1.6.3. Default Constraint Oluşturmak

Bir tabloya veri girişi esnasında o alanın alacağı varsayılan bir değer tanımlanması için kullanılan kısıtlayıcıdır.

Tabloda bulunan tarih alanına bir değer girilmemesi durumunda bulunulan günün tarihini o alana aktarma işlemi en güzel örnektir. SQL'de bu işlem GETDATE() fonksiyonu kullanılarak yapılır.

Tablonun ilk oluşturulması esnasında genel kullanımı şöyledir.

```
CREATE TABLE Tabloadi (  
    Sütunadi,  
    Sütunadi,  
    CONSTRAINT Constraint_Adi DEFAULT (Sütunadi)  
)
```

Varolan bir tablo için ise aşağıdaki gibi kullanılmalıdır.

```
ALTER TABLE Tabloadi
  ADD CONSTRAINT Constraint_Adi DEFAULT ifade veya deęer FOR Sütunadi
```

Örnek

```
ALTER TABLE Okul
  ADD CONSTRAINT DC_Sinif DEFAULT 10 FOR Sinif
```

1.6.4. Check Constraint Oluşturmak

Tabloda belirtilen bir sütuna istenilen şartlara göre deęer girilebilmesini ve bunların kontrolünü sağlayan kısıtlayıcıdır. Aynı sütun için birden fazla Check Constraint kullanılabilir.

Check Constraint'in kullanım amaçları şöyledir,

- Sütuna girilebilecek verileri bir sınır ile kısıtlamak,
- Sütuna girilebilecek verilerin belli bir formatla girilmesini sağlamak,
- Sütuna girilebilecek verileri başka sütun formatlarına göre karşılaştırarak denetlemektir.

Tablonun ilk oluşturulması esnasında genel kullanımı şöyledir.

```
CREATE TABLE Tablo_Adi (
  Sütun_Adi,
  Sütun_Adi,
  .....,
  CONSTRAINT Constraint_Adi CHECK (ifade)
)
```

Örnek:

“Sinif” adında yeni bir tablo oluşturulacaktır. Bu tabloyu oluşturmadan önce yeni bir veri tabanı (Databases) oluşturabilirsiniz. Bu tablonun sütunları ise, OgrNo int türünde, Ad ve Soyad Char türünde olsun.

OgrNo alanına öğrenci numaralarının mutlaka 0'dan (sıfır) büyük olmalarını sağlayacak CHECK ifadesini CREATE TABLE içerisinde yazabilirsiniz.

```
CREATE TABLE Sinif(
  OgrNo int,
  Ad VARCHAR (15),
  Soyad VARCHAR(20),
  CONSTRAINT CK_Sinif CHECK (OgrNo>0)
)
```

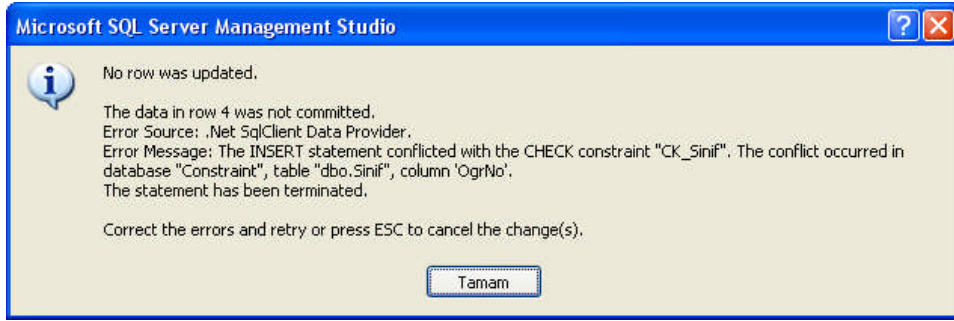
Resim 1.2: CHECK ifadesinin kullanımı

Query'nizi kaydediniz ve Execute ediniz. Hata mesajı almadıysanız tablonuzu Open Table komutuyla açınız ve veri girişine başlayınız. Öğrenci numaraları 0'dan büyük olduğu sürece veri girişinde sorun olmayacaktır.

OgrNo	Ad	Soyad
123	Ahmet	Hasan
125	Hasan	KEMAL
126	Ali	CAN
NULL	NULL	NULL

Resim 1.3: Veri girişi

Ancak, öğrenci numaralarını girerken 0 ya da 0'dan küçük bir değer vermek istediğinizde hata mesajı ile uyarılacaksınız.



Resim 1.4: Hata mesajı

Bu hata mesajında Check Constraint ile belirttiğiniz ifadeye göre uygun veri girişinin yapılmadığı ve verilerin tabloya eklenemediği bildirilmektedir. Ya veriyi düzenleyeceksiniz ya da ESC tuşuna basarak veri girişinden vazgeçeceksiniz demektir.

1.7. Default Nesnesi Oluşturmak

Default nesnesi, Default Constraint ile aynı işleve sahiptir ve ayrı bir nesne olarak tanımlanır. Bir tabloda bir alan için sadece bir adet Default nesnesi tanımlanabilir. Ama, nesne olarak tanımlanmayan Default veya Check Constraint'ler birden fazla tanımlanabilir.

Genel kullanımı şu şekildedir.

```
CREATE DEFAULT Default_Adi AS değer veya ifade
```

Son olarak, oluşturulan Default nesnesinin sp_bindefault isimli sistem Stored Procedure' ü kullanılarak sütunla ilişkilendirilmesi gerekir.

```
sp_bindefault Default_Adi, 'tablo.sütun_adi'
```

Eğer, default nesnesini artık kullanmayacaksanız DROP ile silmeniz gerekir.

```
DROP DEFAULT Default_Adi
```

1.8. Rule Oluşturmak

Rule nesnesi de ayrı bir nesne olarak tanımlanmaktadır. Check Constraint'lerle aynı işleri yapabilir. Rule oluşturduktan sonra sp_bindrule isimli sistem Stored Procedure'üyle ilişkilendirilmesi gerekir.

Genel kullanımı şu şekildedir.
CREATE RULE Rule_Adi AS ifadeler

Bağlantı şekli de aşağıdaki gibidir.

```
sp_bindrule Rule_Adi, Tablo.Sütun_adi
```

Rule'u silmek için de DROP komutunu kullanmalısınız.
DROP RULE Rule_Adi

1.9. Tablo Seviyeli Kısıtlamalar

Satır ve başvuru bütünlüğü ile veri tabanındaki verilerin tablo içerisinde ve tablolar arasında birbiriyle uyumlu olması sağlanacaktır.

1.9.1. Sütunlar Arası Check Constraint Oluşturmak

Tablonuzda bu işlemin yapılacağı sütunlar varsa Check Constraint ile satır bilgilerinin doğruluğunu kontrol edecek bileşenler tanımlamakla yapacağınız işleri kolaylaştırabilirsiniz.

Örnek

Bir stok programı yazıldığını düşünün. "Urunler" adlı tabloda ürünlerle ilgili bilgilerin olduğunu varsayın. UrunGirisTarihi, ürünün depoya giriş tarihinin, UrunCikisTarihi de ürünün depodan çıkış tarihinin girildiği sütunlar olarak belirlensin. Ürünün depodan çıkış tarihinin her zaman boş ve giriş tarihinden büyük olduğunu garanti edecek bir Check Constraint'i tablo seviyeli olarak şöyle tanımlayabilirsiniz.

```
CREATE TABLE Urunler (  
    UrunNo VARCHAR(10),  
    UrunAd VARCHAR(200),  
    UrunGirisTarihi DATETIME,  
    UrunCikisTarihi DATETIME NULL,  
    CONSTRAINT chk_UrunCikisTarihi CHECK (UrunCikisTarihi IS NULL OR  
        UrunCikisTarihi >= UrunGirisTarihi) FOR UrunCikisTarihi  
)
```

1.9.2. Foreign Key Constraint Oluşturmak

Bir tablodaki bir sütuna ait değerlerin, başka bir tablonun belli sütunundan gelmesini denetler. Bir tabloya girilebilecek değerleri başka bir tablonun bir belli alanında yer alabilecek veri grubu ile sınırlandırmaya ve en önemlisi de ilişkilendirmeye yarar.

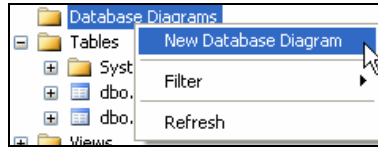
Bu kısıtlayıcıyı hem Management Studio hem de T-SQL kodu yazarak oluşturabilirsiniz. Tabii ki veri tabanınızda birden fazla tablo olması gerekir.

Column Name	Data Type	Allow Nulls
OgrNo	int	<input type="checkbox"/>
TcKimlikNo	varchar(11)	<input checked="" type="checkbox"/>
Ad	varchar(15)	<input checked="" type="checkbox"/>
Soyad	varchar(20)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
TcKimlikNo	varchar(11)	<input type="checkbox"/>
Baba_Ad	varchar(15)	<input checked="" type="checkbox"/>
Ana_Ad	varchar(15)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

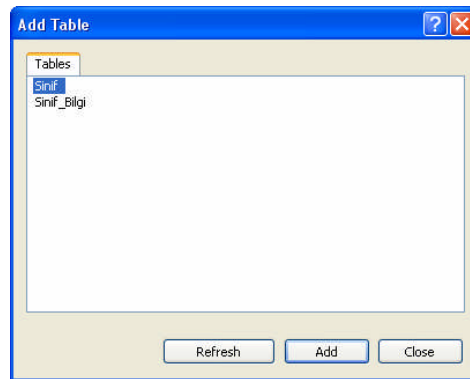
Resim 1.5: Tablolar

Management Studio'da veri tabanınızda bulunan tablolar için yeni bir Database Diagram oluşturmanız ve bu tabloları oraya eklemeniz gerekir.



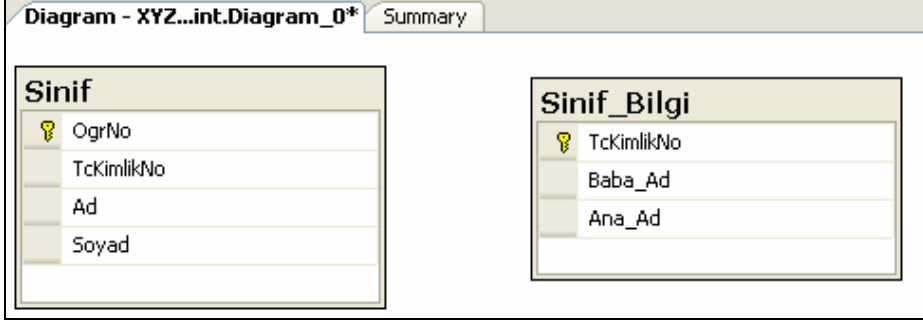
Resim 1. 6: Yeni veri tabanı diyagramı oluşturma

Diyagramı oluşturduğunuzda ekrana Add Table iletişim kutusu gelecektir.



Resim 1. 6: Add Table iletişim kutusu

Add Table iletişim kutusundan ilgili tabloları seçip Add düğmesine basarak diyagrama ekleyiniz. Ekleme işiniz bitince Close düğmesine basarak Add Table iletişim kutusunu kapatınız.



Resim 1. 7: Diyagrama eklenmiş tablolar

“Sinif” tablosundaki “OgrNo” alanına fareyle tıklayıp bırakmadan “Sinif_Bilgi” tablosundaki ilişkilendirilecek alana kadar fareyi sürükleyip bırakınız. İlişki gerçekleştirilmiş olacaktır.

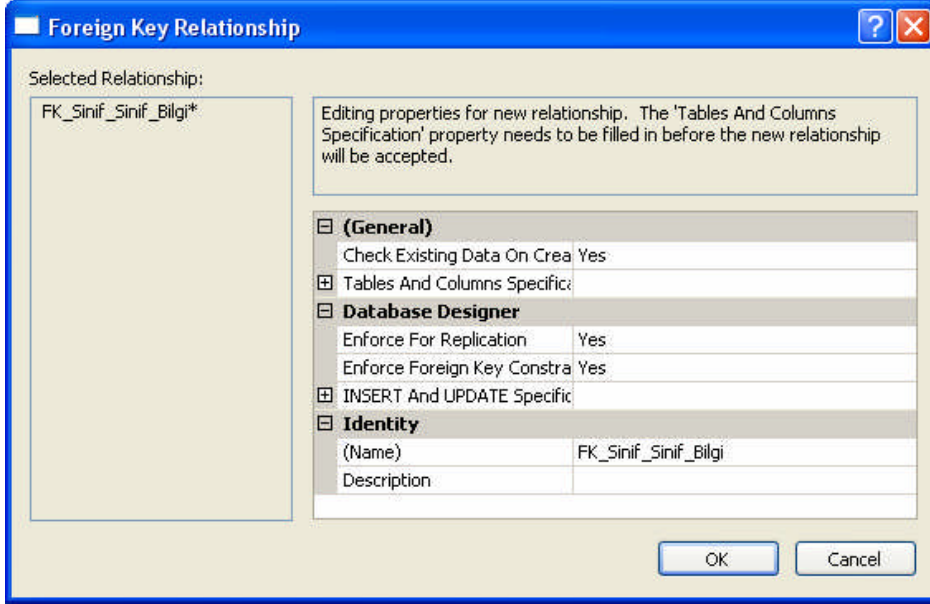
Tables and Columns iletişim kutusu ekrana gelecektir.

The 'Tables and Columns' dialog box is shown. It has a title bar with a question mark and a close button. The 'Relationship name' field contains 'FK_Sinif_Sinif_Bilgi'. Below it, there are two sections: 'Primary key table:' with a dropdown menu showing 'Sinif_Bilgi' and 'Foreign key table:' with a dropdown menu showing 'Sinif'. Under each dropdown, there is a list of columns. For 'Sinif_Bilgi', the column 'TcKimlikNo' is selected. For 'Sinif', the column 'TcKimlikNo' is selected. At the bottom, there are 'OK' and 'Cancel' buttons.

Resim 1. 8: Tables and Columns iletişim kutusu

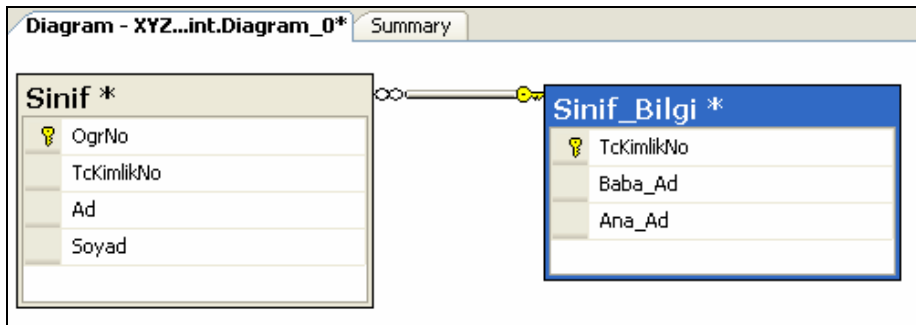
İlişkilendirilecek alanlar burada size gösterilecektir. İsterseniz ilişkilendirilecek alanları açılır liste kutularına tıklayarak değiştirebilirsiniz. OK düğmesine basarak iletişim kutusunu kapatınız.

Yabancı anahtar ilişkilendirilmesi yapılmış olacaktır.



Resim 1. 9: Yabancı anahtar ilişkilendirmesi

Yaptığınız ilişkilendirilme diyagram üzerinde gösterilecektir.



Resim 1.10: İlişkilendirmenin gösterimi

İlişkilendirilmenin sonucunda, sonsuz işareti yabancı tarafını, anahtar işareti de birincil tarafı göstermektedir.

Diğer bir yöntem olan T-SQL'le tablo oluştururken Foreign Key Constraint tanımlamak için aşağıdaki genel yapıyı kullanınız.

Genel Yapı

```
CREATE TABLE tablo_adi (  
    Sütun_adları,  
    CONSTRAINT const_adi FOREIGN KEY (sütun_adi)  
    REFERENCES diğertablo_adi (sütun_adi)  
)
```

Oluşturulmuş olan tablo için ise genel yapı şu şekildedir.

```
ALTER TABLE tablo_adi  
    ADD CONSTRAINT cons_adi FOREIGN KEY (sütun_adi)  
    REFERENCES diğertablo_adi (sütun_adi)
```

Managemet Studio ile oluşturulan Foreign Key için T-SQL kodu da aşağıdaki gibidir.

```
ALTER TABLE Sinif  
    ADD CONSTRAINT FK_Sinif_Bilgi_Sinif FOREIGN KEY (TcKimlikNo)  
    REFERENCES Sinif_Bilgi (TcKimlikNo)
```

Yabancı anahtar tanımlanırken unutulmaması gereken nokta; yabancı anahtarın, referansta bulunan değerler değiştikçe değişmesi gerekliliği vardır. Bunun içinde Foreign Key Constraint tanımlarken, birincil tabloda bir UPDATE (güncelleme) veya DELETE (silme) işlemi gerçekleştirildiğinde, yabancı tarafta nasıl bir yol izleneceği CASCADE veya NO ACTION deyimleri kullanılır. Ancak, bir Foreign Key Constraint, birincil tarafta silmeye müsaade edebilmesi için, yabancı taraftaki bütün ilişkili kayıtların silinmiş olması şartını arar.

İfadenin genel kullanımı şu şekildedir:

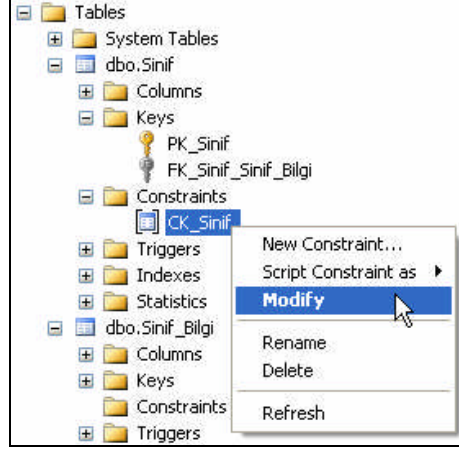
```
ON {DELETE |UPDATE} {CASCADE |NO ACTION}
```

ON DELETE, bir birincil kayıt silindiği zaman yabancı kayıtların ne yapılacağını belirtirken ON UPDATE birincil kayıt silindiği zaman, yabancı kayıtların ne yapılacağına dair bir tanımlama girmek için kullanılır. CASCADE, birincil kayıtlardaki değişimin yabancı kayıtlara da yansıtılacağını belirtir. NO ACTION default olarak bu ayar geçerlidir. Şayet birincil kayıt siliniyorsa, ilişkili yabancı kayıt bulunmuyorsa müsaade edilir. Ama ilişkili kayıt varsa, silinmesine müsaade edilmez.

1.10. Constraint'leri Düzenlemek

Constraint'leri düzenlemenin en uygun yolu Managemet Studio'yu kullanmaktır. Düzenleme işlemi T-SQL kodlarıyla da gerçekleştirilebilir ancak, bu işlem için birkaç aşamayı gerçekleştirmeniz gerekir.

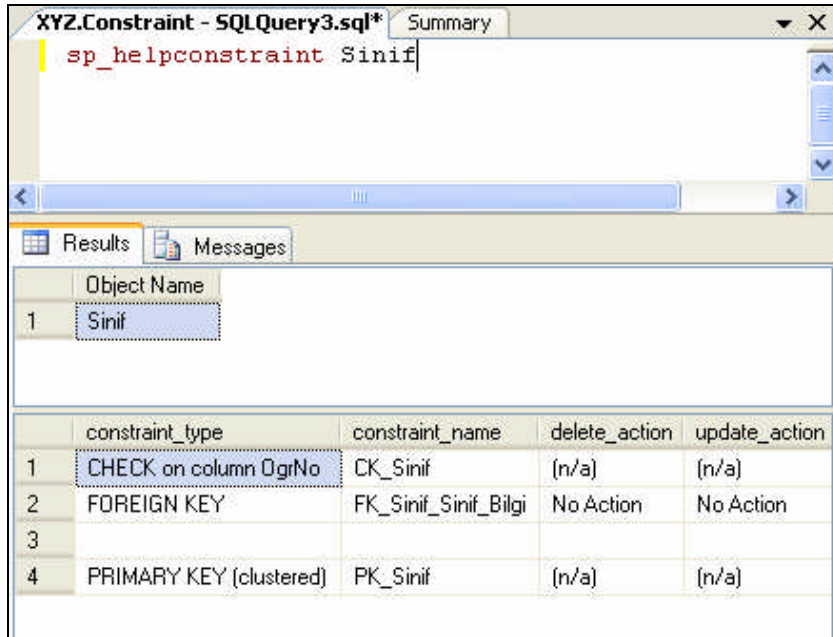
Management Studio’da oluşturulan Constraint’leri düzenlemek için üzerinde fareyle sağ tıklayınız. Açılan yardımcı menü ile yeni bir Constraint oluşturabilir, düzenleyebilir, adını değiştirebilir veya silebilirsiniz.



Resim 1.11: Constraint’i düzenlemek

1.11. T_SQL ile Tablodaki Constraint’lerin Gösterimi

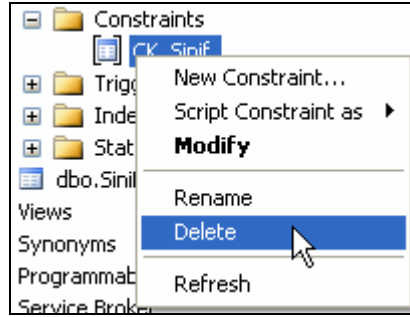
Tabloda oluşturduğunuz Constraint’leri görebilmek için sorgu ekranında `sp_helpconstraint` sistem saklı prosedürünü kullanmanız gerekir. Böylece tanımlı kısıtlayıcıları ve hangi tür olduklarını görebilirsiniz.



Resim 1.12: Constraint’lerin gösterimi

1.12. Constraint'leri Silmek

Tanımlanan bir Constraint'i Management Studio'da mevcut Constraint üzerinde fareyle sağ tıklayıp açılan menüden Delete komutunu vererek silebilirsiniz.



Resim 1.13: Constraint silme

Bu işlemi T-SQL kodu yazarak da yapabilirsiniz.

```
ALTER TABLE tablo_adi  
DROP CONSTRAINT const_adi
```

Örnek:

```
ALTER TABLE Sinif  
DROP CONSTRAINT CK_Sinif
```

1.13. Constraint'lerin Kullanılıp Kullanılmaması

Veri tabanında tanımlı kısıtlayıcıların yapılan işleme göre bazı zamanlarda kullanılması istenmeyebilir. Bu gibi durumlarda kısıtlayıcıların devre dışı bırakılması gerekir.

Bir Constraint'i devre dışı bırakmak için,

```
ALTER TABLE tablo_adi  
NOCHECK CONSTRAINT const_adi
```


şeklinde yazılmalıdır.

Tüm Constraint'leri devre dışı bırakmak için ise,

```
ALTER TABLE tablo_adi  
NOCHECK CONSTRAINT ALL
```

yazılmalıdır.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
➤ Yeni bir veri tabanı oluşturunuz.	➤ Object Explorer’da Databases klasörü üzerinde fareyle sağ tıklayarak açılan menüden New Database komutunu veriniz.
➤ Veri tabanına bir isim veriniz.	➤ Veri tabanının adı “Faaliyet-4” olabilir.
➤ Yeni bir sorgu oluşturunuz.	➤  simgesine tıklayabilirsiniz.
➤ Oluşturduğunuz bu sorguda “Kitaplar” adında bir tablo oluşturunuz.	➤ CREATE TABLE komutunu kullanabilirsiniz.
➤ Tablonun sütunlarını belirleyiniz.	➤ Sütun adları ve veri türleri için; Stok_Kodu int , Kitap_Adi VARCHAR (50), Yazar VARCHAR(50), Y_Evi VARCHAR(50) şeklinde bir tanımlama yapabilirsiniz.
➤ Bu tablo için Birincil anahtar olarak stok kodunu belirleyiniz.	<code>CONSTRAINT PKC_Stok_Kodu PRIMARY KEY (Stok_Kodu)</code>
➤ Stok kodunun değerinin sıfırdan büyük olmasını sağlayan bir Check Constraint belirtiniz.	<code>CONSTRAINT CK_StokKod CHECK (Stok_Kodu>0)</code>
➤ Tabloyu Execute ediniz.	➤ Klavyeden F5 tuşuna basabilirsiniz. Hata olmadığından emin olunuz.
➤ Tables klasöründe tablonuzun oluştuğunu görünüz.	➤ Tables >> Refresh
➤ İkinci bir sorgu oluşturunuz.	➤ New Query simgesine tıklayabilirsiniz.
➤ Oluşturduğunuz bu örgü için de “KitapBilgi” adında bir tablo oluşturunuz.	➤ CREATE TABLE
➤ Tablonun sütunlarını belirleyiniz.	➤ Sütun adları ve veri türleri için; Stok_Kodu int , Kitap_Kodu int, BasimYili VARCHAR (50), ISBNno VARCHAR(13), SayfaSayisi int, şeklinde bir tanımlama yapabilirsiniz.
➤ İkinci tablonun birincil anahtarını belirleyiniz.	<code>CONSTRAINT PKC_StokK PRIMARY KEY (Stok_Kodu)</code>

➤ ISBNno sütununa girilecek numaraların birbirinden farklı olması için bir kısıtlayıcı yazınız.	<code>CONSTRAINT UC_ISBNno UNIQUE (ISBNno)</code>
➤ ISBN numarasının mutlaka 13 karakter olarak girilmesini sağlayınız.	<code>CONSTRAINT CK_ISBN CHECK (LEN (ISBNno) = 13)</code>
➤ İkinci tabloda da Stok kodunun mutlaka sıfırdan büyük girilmesini sağlayınız.	<code>CONSTRAINT CK_StokK CHECK (Stok_Kodu > 0)</code>
➤ “KitapBilgi” tablosunu “Kitaplar” tablosuna “Stok_Kodu” sütunu üzerinden bağlayacak bir yabancı anahtar oluşturunuz. Bununla beraber birinci tablodan bir kaydın silinmesi veya değiştirilmesi durumlarında ikinci tablodaki bağlı sütunda etkilenecek şekilde bir kısıtlayıcı tanımlayınız.	<code>CONSTRAINT FK_KitapBilgi_Kitaplar1 FOREIGN KEY (Stok_Kodu) REFERENCES Kitaplar1 (Stok_Kodu) ON UPDATE CASCADE ON DELETE CASCADE</code>
➤ Tablonuzu Execute ediniz. Hata olmadığından emin olunuz.	F5
➤ Tablolara kayıtlar girerek hazırlanan kısıtlamaların çalışmasını görünüz.	

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru/yanlış seçenekli sorularda uygun harfleri yuvarlak içine alınız. Seçenekli sorularda ise uygun şıkkı işaretleyiniz. Boşluk doldurmalı sorularda boşluklara uygun cevapları yazınız.

1. Bir tabloda verilerle ilgili işlemler yapılırken diğer tablolardaki verilerin birbiriyle tutarlılığının sağlanmasına.....denir.
2. Aşağıdakilerden hangisi tanımlanabilir bir veri bütünlüğü çeşidi **değildir**?
A) Rule
B) Default
C) Constraint
D) Prosedürel
3. Aşağıdakilerden hangisi veri bütünlüğü esaslarından **değildir**?
A) Başvuru bütünlüğü
B) Sütun bütünlüğü
C) Default bütünlüğü
D) Satır bütünlüğü
4. Veri üzerindeki mantıksal sınırlamalara denir.
5. Birincil anahtar kısıtlayıcı, bir sütuna aynı olan değerlerin girilmesine izin verir.(D/Y)
6. Bir tablodaki bir sütuna ait verilerin başka bir tablonun sütunundan getirilmesini denetleyen yabancı anahtar kısıtlayıcıdır. (D/Y)
7. Sütuna girilecek bilgileri kontrol etmek için sütun seviyeli kısıtlamalar kullanılır. (D/Y)
8. Sütuna girilebilecek verilerin belli bir formatla girilmesini sağlamak için oluşturmak gerekir.
9. Tanımlanan bir Constraint'i silmek için sağ klik ile açılan menüdenkomutu seçilir.
10. Kısıtlamayı devre dışı bırakmak için ifadesi kullanılmalıdır.

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konulara geri dönerek tekrar inceleyiniz. Tüm sorulara doğru cevap verdiyseniz diğer modüle geçiniz.

ÖĞRENME FAALİYETİ-2

AMAÇ

Trigger ve Saklı prosedür tanımlamayı, sorgu içerisinde kullanabilmeyi öğreneceksiniz.

ARAŞTIRMA

- Programlama dillerindeki prosedürlerin kullanım amaçlarını araştırınız.

2. TRIGGER VE SAKLI PROSEDÜRLER

2.1. Trigger

Trigger'lar veri tabanı içerisindeki tabloda belirli olaylar meydana geldiği zaman çalışan küçük kod parçalarıdır. Bu olaylar ise aşağıda listelenmiştir.

- Insert (Ekleme)
- Update (Güncelleme)
- Delete (Silme)

Yani daha açık bir deyimle bir tabloya yeni bir kayıt eklendiğinde, bir kayıt silindiğinde veya bir kayıt değiştirildiğinde yapılması istenen olaylar için yazılabilir.

Trigger yazım şekli ;

```
Create Trigger trigger_adi
On tablo_adi
For Update | Insert | Delete
As
SQL kodları
```

NOT: Trigger'lar query analizer'da yazılıp çalıştırılarak kaydedilebileceği gibi, Enterprise Manager'da da yazılabilir. Enterprise Manager'a girip hangi tablo için trigger yazmak istiyorsak onun üzerinde sağ tuşa tıklayalım. Karşımıza çıkan menüden All Tasks/Manage triggers seçeneği seçelim. Karşımıza gelen pencerede trigger'larımızı yazabiliriz.

Örnek: Okuldaki bir sınıfa bir öğrenci geldiğinde sınıftaki öğrenci sayısının bir artırılması gerekir. Aşağıdaki trigger da bu işlemi gerçekleştirmektedir. Bu soru için bilinmesi gerekli olan nokta şudur. Siz tablonuza yeni bir kayıt eklemek istediğinizde bu kaydın bir kopyası **inserted** table'ında tutulur.

```
CREATE TRIGGER yeni_ogrenci ON [ogrenci]
FOR INSERT
AS
declare @sayi int
select @sayi=ogno from inserted
update ogno set sayi=sayi+1 where ogrenci.ogno=@sayi
```

Örnek: Okuldaki bir öğrencinin sınıfını değiştirdiğini düşünelim. O zaman sınıf tablosunda iki işlem gerçekleşmelidir. Öğrencinin ayrıldığı sınıftaki sayı alanındaki değer bir azaltılmalı, yeni katıldığı sınıftaki değer ise bir artırılmalıdır.

NOT: Bu soru için bilinmesi gerekli olan nokta şudur: Siz tablonuzdaki bir kayıt üzerinde değişiklik yaptığımızda bu değişikliğe ait yeni kayıt inserted tablosunda, kaydın değiştirilmemiş hali ise deleted tablosunda tutulur.

```
CREATE TRIGGER [ogrenci_degis] ON [sinif]
FOR UPDATE
AS
declare @eski int, @yeni int
select @eski=ogno from deleted
select @yeni=ogno from inserted
update ogsay set sayi=sayi+1 where ogno=@yeni
update ogsay set sayi=sayi-1 where ogno=@eski
```

Örnek: Sınıf tablosundaki kayıtlardan biri silindiğinde bu kaydın kayıtlı olduğu sınıfa ait sayı alanındaki değeri bir azaltan bir trigger yazalım.

```
CREATE TRIGGER ogrenci_sil ON [sinif]
FOR DELETE
AS
declare @sayi int
select @sayi=ogno from deleted
update ogsay set sayi=sayi-1 where sinif.ogno=@sayi
```

2.2. Stored Procedure

Bir amaca ulaşmak için takip edilen yol ve yöntem tanımından yola çıkarak bir prosedür, herhangi bir işlevi yerine getirmek için yazılan kodların bir paket içerisinde tutulmuş hali demektir.

Bir prosedür başka bir prosedür tarafından çağrılabilir. Prosedürlerin oluşturulmasının nedeni, sıkça yapılan işlemlerin bir defa yazılarak program akışına göre tekrar tekrar kullanılmasını sağlamaktır. Böylece, kod yazımı ve programlama kolaylaştırılmış olur.

Stored Procedure'ler (SP) diğer programlama dillerindeki fonksiyonlara (function) denk gelmektedir. Oluşturulan bir Stored Procedure'e ana programdan bir komut ile ulaşılabilir.

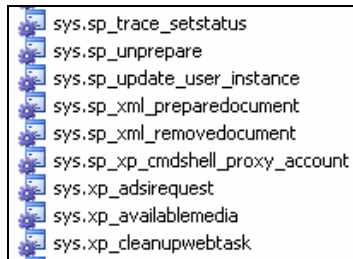
1980'li yıllarda Sybase SQL Server ile kullanıma giren Stored Procedure'ler, sorguların önceden derlenmesi ve veri tabanı yönetim sistemiyle aynı ortamda çalışmasından dolayı hızlı sonuç vermesi en büyük özelliğidir. Bir Stored Procedure (SP) oluşturulur ve sunucuda saklanır. Her ihtiyaç duyulduğunda defalarca çağrılarak işleme tabi tutulabilir.

Oluşturulma şekillerine göre Stored Procedure'ler dört çeşittir.

2.2.1. Extended Stored Procedure

Genellikle “dll” dosya şeklinde derlenmiş prosedürlerdir. T-SQL dışındaki dillerde yazılırlar. Adları “xp_” ile başlar. Ama “sp_” ile başlayan prosedürler de vardır. Master veri tabanında tutulurlar ve tam adları yazılarak kullanılır.

“master.dbo.xp_cmdshell” gibi.



Resim 2.1: Extended Stored Procedure'ler

2.2.2. CLR Stored Procedure

Herhangin bir CLR programlama dilini kullanarak Stored Procedure oluşturulabilir. T-SQL'in desteklemediği kod yazımlarında bu programlama dilleriyle kodlar yazılır. Güvenlik ve denetim gibi konularda CLR Stored Procedure'ler SQL Server için avantaj sağlar.

2.2.3. System Stored Procedure

sp_ ile başlayan prosedürlerdir. Master veri tabanında tutulurlar. Extended Stored Procedure'de olduğu gibi tam adlarının yazılması zorunluluğu yoktur. Sadece adını yazmak yeterlidir.

2.2.4. Kullanıcı Tanımlı Stored Procedure

Programcının yazdığı prosedürlerdir. Geçerli oldukları yere göre üç tiptir.

- Geçici
- Yerel
- Uzak

2.3. Çalıştırılma Aşamaları

Bir Stored Procedure'ün çalıştırılma aşamaları herhangi bir Query'nin çalıştırılma aşamalarıyla aynıdır.

- Stored Procedure'ün bileşenleri parçalara ayrıştırılır (Parsing).
- Veri tabanı içerisinde table, view gibi başka nesnelere gönderme yapan referanslar varsa, geçerli olup olmadıkları (nesnenin ve iznin olup olmaması) kontrol edilir.
- Kontrollerden geçen Stored Procedure'ün adı sysobjects tablosuna, kodları ise syscomments tablosuna kaydedilir (Compilig).
- Bu işlemlerle birlikte derleme işlemi yapılır. Normalizasyon işlemleri olarak da anılan bu işlemler sonucunda, ağaç şeması elde edilir. Bu şema da sysprocedures tablosunda saklanır.
- Stored Procedure herhangi bir anda çağrıldığında, ilk kez çalışıyorsa bu işlemler gerçekleştirilir. İlk defa çağrılmıyorsa, kontrol, sorgulama ağacı oluşturma işlemleri yapılmaz ve oldukça hızlı bir şekilde Stored Procedure'ün derlenmiş hali çalışır. Bundan dolayı Stored Procedure'ler derlenen nesnelere biri olarak bilinir.

2.4. Stored Procedure Oluşturmak

Stored Procedure'ün oluşturulma şekli aşağıdaki gibidir.

```
CREATE PROC [ EDURE ] prosedür_adı
AS
    T-SQL ifadeleri
GO
```

Oluşturacağınız Stored Procedure'ler ile sistemin Stored Procedure'lerinin karışmaması için prosedür adlarının önüne "U" harfini ekleyiniz. Böylece, oluşturduğunuz Stored Procedure'leri sistem Stored Procedure'lerinden ayırt etmiş olursunuz.

Stored Procedure oluşturabilmek için sysadmin, db_owner veya dll_admin rolüne sahip olmanız gerekir.

Bir Stored Procedure CREATE DEFAULT, CREATE PROCEDURE, CREATE RULE, CREATE TRIGGER ve CREATE VIEW ifadelerini içeremez. Ancak, her nesneden veri alabilir.

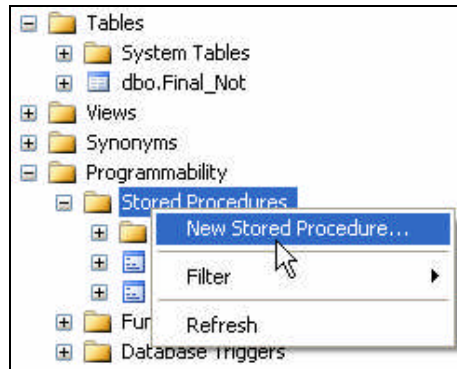
Örnek:

- Bir sınıfta bulunan öğrencilerin, öğrenci numaraları, adları, soyadları ve final notları bir tabloda tutulmaktadır.

OgrNo	Ad	Soyad	Final
90	Ahmet	HASAN	75
92	Ali	CAN	80
93	Fatma	KOÇ	65
94	Ayşe	TANYERİ	45
95	Veli	BAL	40
96	Fikret	PARS	90
98	Cemil	ALP	55
NULL	NULL	NULL	NULL

Resim 2. 2: Final_not tablosu

- Final notları 55'ten büyük olan notları gösterecek bir Stored Procedure oluşturmak istediğiniz düşününüz.
- Veri tabanınızın Programmability klasörünün solundaki + işaretine tıkladığınızda açılan Stored Procedures klasörü üzerinde fare ile sağ tıklayınız.
- Yeni bir Stored Procedure oluşturabilmeniz için New Stored Procedure komutunu tıklayınız.



Resim 2. 3: New Stored Procedure komutu

- Komutu tıkladığınızda Stored Procedure'ün genel yapısını bulunduran bir Query sayfası açılacaktır.

```

-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
-- =====
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =====
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
-- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int>
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int>
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result set
-- interfering with SELECT statements.
    SET NOCOUNT ON;

-- Insert statements for procedure here
    SELECT <@Param1, sysname, @p1>, <@Param2, sysname,
END
GO

```

Resim 2. 4: Stored Procedure yapısının hazır olarak verildiği Query sayfası

- Stored Procedure'ün yapısı için gerekli yazım kuralı verilmişti. Ekranı gelen Query sayfasında size gerekli olmayacak satırları silebilir veya Query'nin tamamını silip kendiniz Stored Procedure'ü oluşturabilirsiniz.
- Final notu 55'ten büyük olan notları görüntüleyecek satırları Resim 2.5'teki gibi yazabilirsiniz.

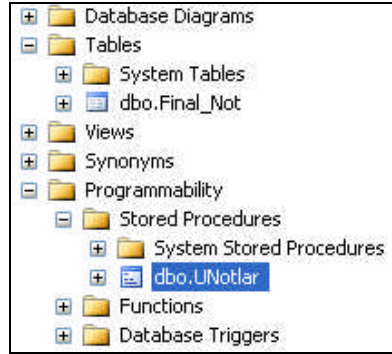
```

CREATE PROCEDURE UNotlar
AS
    SELECT * FROM Final_Not
    WHERE Final>55
GO

```

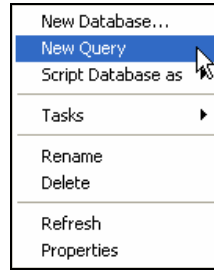
Resim 2. 5: Stored Procedure'ün oluşturulması

- F5 tuşuna basarak oluşturduğunuz Stored Procedure'ün doğruluğunu sınavınız ve hata varsa düzeltiniz.
- Oluşturduğunuz Stored Procedure'ü görmek için Stored Procedures üzerinde fareyle sağ tıklayarak Refresh komutunu veriniz.



Resim 2. 6: Stored Procedure'ün görülmesi

- Oluşturulan Stored Procedure'ü çalıştırmak için veri tabanınız üzerinde fare sağ tuşuyla açılan menüden New Query komutunu veriniz.



Resim 2. 7: New Query komutunun verilmesi

- Tablonuzdaki verilerin tamamını ve final notu 55'ten büyük olan öğrencileri görebilmek için Resim 2.8'deki gibi ilgili kodları yazınız.

```
SELECT * FROM Final_Not  
EXEC UNotlar
```

Resim 2. 8: İlgili kodlar

- EXEC komutu oluşturduğunuz Stored Procedure'ü çalıştıracak olan komuttur. EXEC'ten sonra yazılan ifade ise oluşturduğunuz Stored Procedure'ün adıdır. Yazdığınız bu Query'i çalıştırarak (F5) sonucu görebilirsiniz.

	OgrNo	Ad	Soyad	Final
1	90	Ahmet	HASAN	75
2	92	Ali	CAN	80
3	93	Fatma	KOÇ	65
4	94	Ayşe	TANYERİ	45
5	95	Veli	BAL	40
6	96	Fikret	PARS	90
7	98	Cemil	ALP	55

	OgrNo	Ad	Soyad	Final
1	90	Ahmet	HASAN	75
2	92	Ali	CAN	80
3	93	Fatma	KOÇ	65
4	96	Fikret	PARS	90

Resim 2. 9: Stored Procedure'ün çalıştırılması ve sonuçların gösterimi

2.4.1. NOCOUNT Parametresi

Sorgunun çalıştırılması sonucunda kaç kaydın bu işlemde etkilendiğini gösterir veya göstermez. SET NOCOUNT ON şeklinde sorguya yazılırsa kaç kaydın bu işlemde etkilendiği gösterilmeyecektir. SET NOCOUNT OFF şeklinde bir sorgu yazımındaysa bu işlemde kaç kaydın etkilendiği gösterilecektir.

Varsayılan olarak bu parametre OFF konumundadır. Yani etkilenen kayıtlar gösterilmektedir. Yukarıdaki Stored Procedure oluşturma örneğinde sonuçların gösterildiği Results sekmesinin yanında bulunan Messages sekmesinde etkilenen kayıtları görebilirsiniz. Eğer, etkilenen kayıtların gösterilmesini istemiyorsanız Stored Procedure'ü oluştururken SET NOCOUNT ON satırını eklemelisiniz.

	OgrNo	Ad	Soyad	Final
1	90	Ahmet	HASAN	75
2	92	Ali	CAN	80
3	93	Fatma	KOÇ	65
4	96	Fikret	PARS	90

	OgrNo	Ad	Soyad	Final
1	90	Ahmet	HASAN	75
2	92	Ali	CAN	80
3	93	Fatma	KOÇ	65
4	96	Fikret	PARS	90


```
(7 row(s) affected)
(4 row(s) affected)
```

Resim 2.10: Etkilenen kayıtların gösterilmesi

2.4.2. Stored Procedure'ün Kullanım Hakkı

Stored Procedure'ü oluşturduktan sonra çalıştırılabilmesi için kullanıcılara izin verilmiş olması gerekir. Eğer, bir kullanıcının izni varsa o Stored Procedure'ü kullanabilecektir.

İzin verme yetkileri DENY ve GRANT ile sağlanmaktadır. DENY ile belirtilen kullanıcı Stored Procedure'ü kullanamayacaktır. GRANT ile belirtilen kullanıcıya Stored Procedure'ü kullanabilecektir.

```
DENY ON UNotlar TO guest1  
GRANT ON UNotlar TO ali
```

2.5. Değişiklik Yapmak

Stored Procedure'lerde değişiklik yapmak ALTER ile gerçekleşir.

Genel Kullanımı

```
ALTER PROC [EDURE] prosedür_adi  
AS  
    T-SQL ifadeleri  
GO
```

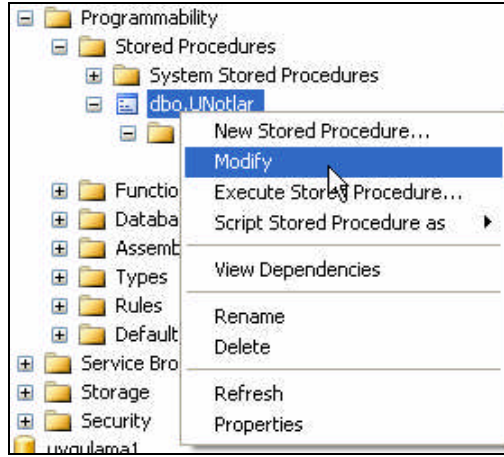
Bir Stored Procedure'de değişiklik yapabilmek için önce kaynak kodunun alınması ve bir Query ekranına kopyalanıp düzenlenmesi gerekir.

Örnek:

- Stored Procedure oluşturmayla ilgili yapılan örnekte final notları 55'ten büyük olan notları göstermektedir. Bu prosedürü final notları 55 ve 55'ten küçük olan final notları şeklinde değiştirelim.
- Oluşturulmuş bir Stored Procedure'de değişiklik yapabilmenin yani kaynak kodlarını alabilmenin iki yolu vardır:
 - Management Studio'da oluşturulmuş Stored Procedure üzerinde fareyle sağ tıklanarak açılan menüden Modify komutunu vermek,
 - “**sp_helptext prosedür_adi**” komutunu kullanarak elde edilen kodları kopyalayıp yeni bir Query'e yapıştırmaktır.

2.5.1. Management Studio ile Değişiklik Yapmak

- Oluşturduğunuz “UNotlar” adındaki Stored Procedure üzerinde fareyle sağ tıklatınız ve açılan menüden Modify komutunu veriniz.



Resim 2.11: Modify komutunun verilmesi

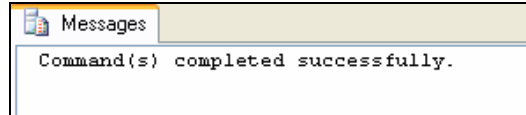
- Modify komutunu verince kodlar bir Query sayfası şeklinde ekrana gelecektir. Üzerinde gerekli değişiklikleri yapınız ve Execute (F5) ediniz.

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

ALTER PROCEDURE [dbo].[UNotlar]
AS
    SELECT * FROM Final_Not
    WHERE Final>55
```

Resim 2.12: Düzenlenmiş Stored Procedure

- Execute işleminin sorunsuz bir şekilde gerçekleştiğini Messages penceresinde görebilirsiniz.



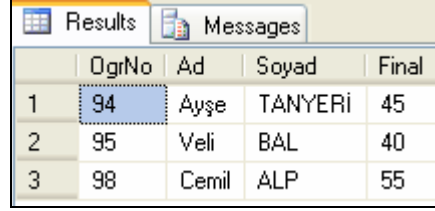
Resim 2.13: Execute işleminin sonucu

- Yeni bir Query ekranı açınız ve Stored Procedure'ün çalışması için gerekli kodu yazınız.

```
EXEC UNotlar
```

Resim 2.14: Stored Procedure'ün çalıştırılması

- UNotlar Stored Procedure'ü işletilecek ve sonuçlar Results penceresinde size gösterilecektir.



	OgrNo	Ad	Soyad	Final
1	94	Ayşe	TANYERİ	45
2	95	Veli	BAL	40
3	98	Cemil	ALP	55

Resim 2.15: Sonuçların gösterimi

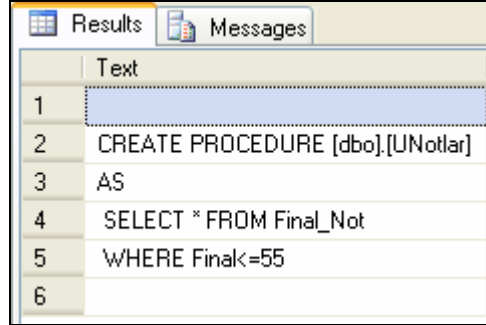
2.5.2. “sp_helptext” İfadesi ile Değişiklik Yapmak

- “UNotlar” Stored Procedure’ündeki kodları görebilmek için yeni bir Query’de “sp_helptext” ifadesini Resim 2.16’deki gibi kullanınız.

```
sp_helptext UNotlar
```

Resim 2.16: İfadenin kullanımı

- “UNotlar” Stored Procedure’deki kodlar Results penceresinde size gösterilecektir.



	Text
1	
2	CREATE PROCEDURE [dbo].[UNotlar]
3	AS
4	SELECT * FROM Final_Not
5	WHERE Final<=55
6	

Resim 2.17: Kodların gösterilmesi

- Kodları seçerek kopyalayınız ve Query sayfasındaki sp_helptext ifadesini silerek yapıştırınız.

```
CREATE PROCEDURE [dbo].[UNotlar]
AS
SELECT * FROM Final_Not
WHERE Final<=55
```

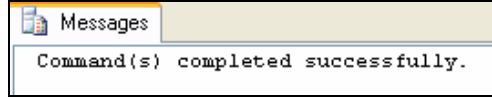
Resim 2.18: Kodların Query’e taşınması

Kodlardaki CREATE komutunu silerek ALTER komutunu yazınız.

```
ALTER PROCEDURE [dbo].[UNotlar]
AS
SELECT * FROM Final_Not
WHERE Final<=55
```

Resim 2.19: ALTER komutunun yazılması

- Stored Procedure'ü çalıştırınız (F5). Execute işleminin sorunsuz bir şekilde gerçekleştiğini Messages penceresinden görebilirsiniz.



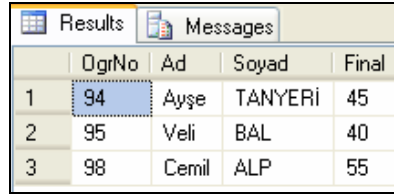
Resim 2.20: Execute işleminin sonucu

- Yine, Query sayfasında yazılan Stored Procedure kodlarını siliniz ve sonuçların gösterilmesi için gerekli olan kodları yazınız.

```
EXEC UNotlar
```

Resim 2.21: Stored Procedure'ün çalıştırılması

- Elde edilen sonuçları Results penceresinde görebilirsiniz.

A screenshot of the 'Results' window in SQL Server Enterprise Manager. The window title is 'Results' and it displays a table with the following data:

	OgrNo	Ad	Soyad	Final
1	94	Ayşe	TANYERİ	45
2	95	Veli	BAL	40
3	98	Cemil	ALP	55

Resim 2.22: Sonuçların Results penceresinde gösterilmesi

Bir Stored Procedure'de çok fazla değişiklik yapıyorsa EXEC komutuyla çalıştırılırken yeniden derlenmesi istenebilir. Bunun için, RECOMPILE kullanılmalıdır.

Örnek:

```
EXEC UNotlar WITH RECOMPILE
```

2.6. Stored Procedure'ü Silmek

Var olan bir Stored Procedure'ü silmek için DROP komutunu kullanmak gereklidir. DROP komutundan sonra Stored Procedure'ün sahibinin adı ve Stored Procedure'ün adı yazılmalıdır.

Genel Kullanımı

DROP PROC sahip.prosedür_adı

Örnek:

DROP PROC dbo.UNotlar

2.7. Değer Alan Stored Procedure'ler

Stored Procedure'lerin daha etkin kullanılabilmesi ve işlevsel bir hale gelebilmesi için dışarıdan değer almalarına ihtiyaç duyulur. Bu nedenle girdi parametreleri (Input Parameter) kullanılır.

Stored Procedure'nin aldığı değer Query'den gelen değerdir. Gönderilen değeri karşılayacak bir değişken Stored Procedure'de tanımlanmalıdır.

Örnek:

- Bir öğrenciye ait 3 not bilgisi Query'de ilk değerleri atanarak not ortalamaları hesaplanacaktır. Hesaplanan not ortalamasına göre öğrencinin başarılı olup olmadığı Stored Procedure'de belirlenecek ve mesaj olarak yazdırılacaktır. Bunun için, yeni bir sorgu sayfası açınız ve öğrencinin 3 notunun tutulacağı değişkenleri tanımlayıp ilk değerlerini Resim 2.23'teki gibi veriniz.

```
DECLARE @nt1 int
DECLARE @nt2 int
DECLARE @nt3 int
DECLARE @ortalama int

SELECT @nt1=55
SELECT @nt2=45
SELECT @nt3=70
```

Resim 2.23: Değişkenlerin tanımlanması ve ilk değerlerinin verilmesi

- İlk değerleri verilen değişkenlerin ortalamasını "@ortalama" değişkenine hesaplatıp Resim 2.24'teki gibi aktarınız.

```
SELECT @ortalama=(@nt1+@nt2+@nt3)/3
```

Resim 2.24: Ortalamannın hesaplanması

- Elde edilen değeri oluşturacağınız Stored Procedure'e EXEC komutuyla Resim 2.25'teki gibi gönderiniz. Böylece hem değer gönderilecek hem de Stored Procedure çalıştırılmış olacaktır.

```
EXEC UHesapla @ortalama
```

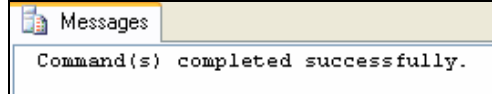
Resim 2.25: Elde edilen değer Stored Procedure'e gönderilmesi

- Hesaplamaların yapılacağı Stored Procedure'ü yazmak için yeni bir Query sayfası açınız. Stored Procedure'ün adını "UHesapla" olarak belirterek ortalamayı karşılayacak değişkeni tanımlayınız ve başarının belirleneceği "if" yapısını yazınız.

```
CREATE PROCEDURE UHesapla
@ort int
AS
IF @ort > 44 PRINT 'Başarılı'
IF @ort <45 PRINT 'Başarısız'
GO
```

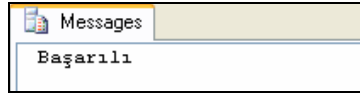
Resim 2.26: "UHesapla" adındaki Stored Procedure'ün oluşturulması

- Yazdığınız Stored Procedure'ü F5 ile çalıştırarak doğruluğunu kontrol ediniz.



Resim 2.27: Execute işleminin sonucu

- Daha sonra ilk yazdığınız sorgunuza dönerek çalıştırınız. "UHesapla" Stored Procedure'ü çalıştırılacak ve sonucu Messages penceresinde gösterilecektir.



Resim 2.28: Sonucun gösterilmesi

Stored Procedure'e gelen parametrelerin isteğe bağlı olması istenebilir. Bu gibi durumlarda Stored Procedure'de tanımlanan değişkene default değer atanması yapılır. Stored Procedure'e gelen parametreye değer atanmazsa, tanımlanan değişkene atanan default değer işleme tabi tutulur. Stored Procedure'de bir default değer atayacaksanız bu değer, bir sabit olması gerekir.

Örnek:

- Bir sınıftaki öğrencilerin bilgilerinin tutulduğu bir tabloda öğrenci adlarının içerisinde “a” harfi geçen öğrencileri gösteren bir Stored Procedure şöyle yazılmalıdır.

Ogr_No	Ad	Soyad	Cinsiyet	Yas
101	Hasan	KAYA	E	17
103	Veli	CAN	E	16
104	Ayşe	KOCAER	K	17
106	Fatma	YILMAZ	K	18
109	Defne	YAĞMUR	K	16
NULL	NULL	NULL	NULL	NULL

Resim 2.29: Öğrenci bilgilerinin tutulduğu tablo

```
CREATE PROCEDURE Ogr_Ara (@ara VARCHAR(10)=NULL)
AS
    IF @ara IS NOT NULL
        SELECT * FROM Sinif
        WHERE Ad LIKE '%'+@ara+'%'
GO
```

Resim 2.30: “Ogr_Ara” Stored Procedure’ü

- Stored Procedure’de tanımlanan “ara” değişkeni Query’den bir değer gelmese de NULL değerini alacak ve işlem gerçekleşecektir.
- Query’den öğrenci adlarının içerisinde “a” harfi olan öğrencileri görmek için Resim 2.31’deki gibi kod satırını yazınız ve çalıştırınız.

```
Ogr_Ara 'a'
```

Resim 2.31: Query’den değer gönderilmesi

- **Ogr_Ara ‘a’** şeklindeki yazım ile de Stored Procedure çalışacaktır.
- Sonuçları Results penceresinde görebilirsiniz.

	Ogr_No	Ad	Soyad	Cinsiyet	Yas
1	101	Hasan	KAYA	E	17
2	104	Ayşe	KOCAER	K	17
3	106	Fatma	YILMAZ	K	18

Resim 2.32: Sonuçların gösterilmesi

- Stored Procedure'e değer gönderilmeseydi Messages penceresinde "Command(s) completed successfully." mesajını görecektiniz.

2.8. Değer Alıp-Veren Stored Procedure'ler

Değer alıp-veren saklı prosedürler, Query'den gönderilen değerleri alıp istenilen işleme tabi tuttuktan sonra elde edilen sonucu tekrar Query'e gönderen prosedürlerdir. Bu işlem için OUTPUT parametresi kullanılır.

Örnek

Tanımlanan ve ilk değerleri verilen iki sayıyı Stored Procedure'e göndererek toplamlarını Stored Procedure'de yapıp sonucu gönderildiği yerde görüntüleyen kodları yazmak için aşağıda verilen adımları uygulayınız.

- Yeni bir Query sayfası açınız. Stored Procedure'ün adını "UTopla" olarak belirleyiniz. Query'den gelecek değerler için kullanılacak iki sayı değişkeni, toplamları ve geriye değer döndürmesi için toplam değişkenini tanımlayınız.

```
CREATE PROC UTopla (
    @s1 int,
    @s2 int,
    @topl int OUTPUT
)
```

Resim 2.33: Stored Procedure'nin oluşturulması ve değişkenlerinin tanımlanması

Gelen sayıları "topl" değişkeninde toplatınız.

```
AS
SELECT @topl=@s1+@s2
GO
```

Resim 2.34: Sayıların toplanması ve değişkene aktarılması

- Oluşturduğunuz prosedürü çalıştırınız ve hata olmadığından emin olunuz.

- Yeni bir Query sayfası açınız. Toplam için kullanılacak iki sayı değişkeni, toplamları için toplam değişkenini tanımlayınız ve ilk değerlerini atayınız.

```
DECLARE @sayi1 int
DECLARE @sayi2 int
DECLARE @toplam int

SELECT @sayi1=9
SELECT @sayi2=8
SELECT @toplam=NULL
```

Resim 2.35: Değişkenlerin tanımlanması ve ilk değerlerinin atanması

- EXEC komutuyla, Stored Procedure'e göndereceğiniz değişkenleri ve sonucu alacak "toplam" değişkenini de OUTPUT parametresi ile yazınız.

```
EXEC UTopla @sayi1,@sayi2,@toplam OUTPUT
```

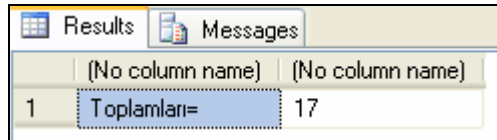
Resim 2.36: Gönderilecek değişkenlerin ve değer alacak değişkenin EXEC ile yazılması

- Stored Procedure'den dönen değeri de Results penceresinde görmek için SELECT komutuyla yazdırınız.

```
SELECT 'Toplamları=', @toplam
```

Resim 2.37: Sonucun yazdırılması

- Query'nizi F5 ile çalıştırınız. Sonucu Results penceresinde göreceksiniz.



	(No column name)	(No column name)
1	Toplamları=	17

Resim 2.38: Sonucun gösterilmesi

- Query'den gönderilen "sayi1" ve "sayi2" değişkenlerini Stored Procedure'de "s1" ve "s2", "toplam" değişkenini de "tpl" değişkeni karşılamaktadır.
- "Toplam" ve "tpl" değişkenleri OUTPUT parametresiyle tanımlandığından sonuç "tpl" değişkeninden "toplam" değişkenine gönderilmektedir.

2.9. RETURN Deyimi

Prosedürden tek bir tamsayı değer döndürmek için bir başka seçenek olarak RETURN ifadesi kullanılabilir. Bu şekilde OUTPUT parametresini hem prosedür içinde hem de prosedürün çağrıldığı yerde tanımlamak zorunda kalınmadan doğrudan değer döndürülebilir.

Bir uygulamada hem RETURN hem de OUTPUT parametresi aynı anda kullanılabilir.

Örnek:

Değer alıp veren Stored Procedure'ler örneğini RETURN deyimi ile de yapılabilir. Örneği adım adım uygulayınız.

- Yeni bir Query sayfası açınız. Stored Procedure'ün adını "URet_Topla" olarak belirleyiniz. Query'den gelecek değerler için kullanılacak iki sayı değişkeni için Stored Procedure'de iki değişkeni tanımlayınız.

```
CREATE PROC URet_Topla (  
    @s1 int,  
    @s2 int  
)
```

Resim 2.39: Stored Procedure'nin oluşturulması ve değişkenlerinin tanımlanması

- Yapılacak işlem sonucunda elde edilen değerın Query'e gönderilmesi için RETURN deyimiyle beraber yapılacak işlemi belirtiniz.

```
AS  
RETURN (@s1+@s2)  
GO
```

Resim 2.40: RETURN deyimi ve geriye dönecek değerın belirtilmesi

- Oluşturduğunuz Stored Procedure'ü çalıştırarak hata olmadığından emin olunuz. Hata yoksa Stored Procedure oluşturulmuş demektir.
- Yeni bir Query sayfası açınız. Kullanılacak değişkenleri belirleyiniz ve ilk değerlerini atayınız.

```
DECLARE @sayi1 int  
DECLARE @sayi2 int  
DECLARE @toplam int  
  
SET @sayi1=9  
SET @sayi2=8
```

Resim 2.41: Değişkenlerin tanımlanması ve ilk değerlerinin verilmesi

- Birinci ve ikinci sayıyı URet_Topla Stored Procedure'üne gönderecek ve dönen değeri toplam değişkenine aktaracak EXEC satırını yazınız.

```
EXEC @toplam=URet_Topla @sayi1,@sayi2
```

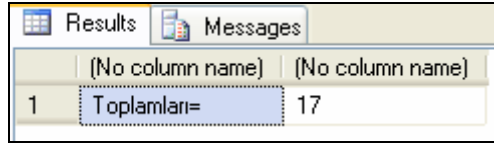
Resim 2.42: Değerlerin gönderilmesi ve değişkene alınması

- Sonucun Results penceresinde görüntülenmesi için gerekli kod satırını yazınız.

```
SELECT 'Toplamları=', @toplam
```

Resim 2.43: Sonucun Results penceresinde gösterimi

- Query'nizi çalıştırarak hata olmadığından emin olunuz. Eğer hata yoksa elde edilen sonucu görebilirsiniz.

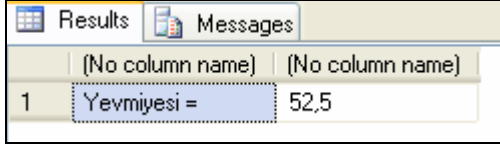


	(No column name)	(No column name)
1	Toplamları=	17

Resim 2.44: Results penceresi

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
➤ Yeni bir veri tabanı oluşturunuz.	➤ Adı “UcretHesaplama” olabilir.
➤ Yeni bir sorgu oluşturunuz.	➤ New Query simgesine tıklayabilirsiniz.
➤ Stored Procedure’ün adını belirleyiniz.	➤ Stored Procedure’ün adı “Yevmiye Hesapla” olabilir.
➤ Bir işçinin çalışma saati ve saat ücreti verildiğinde alacağı günlük yevmiye tutarını gösterecek bir Stored Procedure yazılacaktır. Query’den gelecek değerleri karşılayacak ve sonucu geri döndürecek değişkenleri tanımlayınız.	<pre>CREATE PROC UYemiyeHesap (@CSaat smallint , @SUcret int , @Yevmiye float OUTPUT)</pre>
➤ Eğer işçinin çalışma saati 8 saat ve 8 saatten küçük ise alacağı yevmiye tutarı çalışma saati ve saat ücretinin çarpımıyla bulunacaktır. Bunun için gerekli şartı yazınız.	<pre>AS IF (@CSaat<=8) BEGIN SELECT @Yevmiye=@CSaat*@SUcret END</pre>
➤ Eğer işçinin çalışma saati 8 saatten büyük ise çalışma saati ve saat ücretinin çarpımı ile bu tutarın %5’i yevmiyeye eklenerek işçinin yevmiyesi hesaplanacaktır. Bunun için gerekli şartı yazınız.	<pre>IF (@CSaat>8) BEGIN SELECT @Yevmiye=(@CSaat*@SUcret)+ ((@CSaat*@SUcret)*0.05) END GO</pre>
➤ Query’i çalıştırınız ve hata olmadığından emin olunuz.	➤ F5

➤ Yeni bir Query daha oluşturunuz	➤ New Query simgesine tıklayabilirsiniz.
➤ Yevmiye hesabı için kullanılacak iki değişkeni ve toplam tutarın tutulacağı değişkeni tanımlayınız.	<pre>DECLARE @Calisma_Saati smallint DECLARE @Saat_Ucreti int DECLARE @Isci_Yevmiye float</pre>
➤ İlk değerlerini atayınız.	<pre>SELECT @Calisma_Saati=10 SELECT @Saat_Ucreti=5 SELECT @Isci_Yevmiye=NULL</pre>
➤ EXEC komutuyla, Stored Procedure'e göndereceğiniz değişkenleri ve sonucu alacak "Isci_Yevmiye" değişkenini de OUTPUT parametresi ile yazınız.	<pre>EXEC UYevmiyeHesap @Calisma_Saati,@Saat_Ucreti, @Isci_Yevmiye OUTPUT</pre>
➤ Stored Procedure'den dönen değeri de Results penceresinde görmek için SELECT komutuyla yazdırınız.	<pre>SELECT 'Yevmiyesi =', @Isci_Yevmiye</pre>
➤ Sorgunuzu çalıştırınız ve hata olmadığından emin olunuz.	F5
➤ Sonucu Results penceresinde görünüz.	 <p>Resim 2.45: Sonucun görüntülenmesi</p>

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki soruları dikkatlice okuyarak doğru/yanlış seçenekli sorularda uygun harfleri yuvarlak içine alınız. Seçenekli sorularda ise uygun şıkkı işaretleyiniz. Boşluk doldurmalı sorularda boşluklara uygun cevapları yazınız.

1. Bir işlevi yerine getirmek için yazılan kodların oluşturduğu program parçasına Saklı Prosedür denir. (D/Y)
2. Aşağıdakilerden hangisi saklı prosedür çeşidi **değildir**?
A) Extended B) CLR C) Model D) System
3. Saklı prosedürleri çalıştırırken bileşenlerini parçalara ayırma işlemine.....denir.
4. Prosedürün adı sysobjects tablosuna’den geçtikten sonra kaydedilir.
5. Saklı prosedür oluşturma komutu.....’dür.
6. Saklı prosedür oluşturabilmek için aşağıdakilerden hangi role sahip olunması **gerekmez**?
A) user
B) sysadmin
C) db_owner
D) dll_admin
7. Bir stored procedure her nesneden veri alamaz.(D/Y)
8. Oluşturulan sorgunun çalıştırılması sonucunda kayıtların bu işlemde etkilenip etkilenmediğini gösterenparametresidir.
9.ile yetki verilen kullanıcı Stored Procedure’ü kullanabilecektir.
10. Stored Procedure’ü silmek için DROP komutu kullanılır. (D/Y)

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konulara geri dönerek tekrar inceleyiniz. Tüm sorulara doğru cevap verdiyseniz diğer öğrenme faaliyetine geçiniz.

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği, öğretmeniniz işlem basamaklarına göre 0 ile 4 puan arasında olacak şekilde değerlendirecektir.

Değerlendirme Ölçütleri	Puan
1. Veri tabanı oluşturabilme	
2. Sorgu oluşturabilme	
3. T-SQL diliyle tablo oluşturabilme	
4. Sütunları veri türleriyle beraber tanımlayabilme	
5. Birincil anahtar oluşturabilme	
6. CHECK CONSTRAINT oluşturabilme	
7. Sorguyu çalıştırabilme	
8. UNIQUE CONSTRAINT oluşturabilme	
9. CHECK CONSTRAINT ile fonksiyonları kullanabilme	
10. Yabancı anahtar oluşturabilme	
11. İlişkili tablolarda güncelleyebilme ve silme izinlerini yazabilme	
12. Saklı prosedür oluşturabilme	
13. Saklı prosedürde hesaplayabilme	
14. Şart cümlecikleri yazabilme	
15. Değişken tanımlayabilme	
16. Değişken veri türlerini belirleyebilme	
17. Değişkenlere ilk değer atayabilme	
18. EXEC komutunu kullanabilme	
19. OUTPUT parametresini kullanabilme	
20. Elde edilen sonucu görüntüleyebilme	
Toplam (100 puan olabilir)	

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

Modülü tamamladınız, tebrik ederiz. Öğretmeniniz size çeşitli ölçme araçları uygulayacaktır, öğretmeninizle iletişime geçiniz.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

1.	Veri bütünlüğü
2.	D Şıkkı
3.	C Şıkkı
4.	Kısıt
5.	Yanlış
6.	Doğru
7.	Doğru
8.	Check Constraint
9.	Delete
10.	NOCHECK

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1.	Doğru
2.	C Şıkkı
3.	Parsing
4.	Kontrol
5.	CREATE PROC
6.	A Şıkkı
7.	Yanlış
8.	NOCOUNT
9.	GRANT
10.	Doğru

KAYNAKÇA

- GÖZÜDELİ Yaşar, “**Yazılımcılar İçin SQL Server 2005 ve Veri tabanı Programlama**”, Seçkin Yayıncılık, Ankara, 2006.
- www.verivizyon.com