T.C.
MİLLİ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKÎ EĞİTİM VE ÖĞRETİM SİSTEMİNİN GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

VERİTABANI YARDIMCI İŞLEMLERİ

ANKARA 2008

Millî 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 İnternet üzerinden ulaşılabilirler.
- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.
- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**481BB0047**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Veritabanı Programcılığı**
MODÜLÜN ADI	**Veritabanı Yardımcı İşlemleri**
MODÜLÜN TANIMI	SQL Server'ın yardımcı işlemleri ile ilgili öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Veritabanı Yönetimi modülünü bitirmiş olmak
YETERLİK	Veritabanı ile ilgili yardımcı işlemleri yapabilmek
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında, veri tabanı ile ilgili yardımcı işlemleri yapabileceksiniz. **Amaçlar** 1. T-SQL komutlarıyla oluşabilecek hataları yakalayabileceksiniz. 2. Veritabanı performans araçlarını kullanabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	**Ortam** Atölye, laboratuvar, bilgi teknolojileri ortamı (İnternet) vb. kendi kendinize veya grupla çalışabileceğiniz tüm ortamlar. **Donanım** Ağ veritabanını çalıştırabilecek yeterlikte bilgisayar, yedekleme için gerekli donanım (CD yazıcı, flash bellek), raporlama için yazıcı, kağıt ve kalem.
ÖLÇME VE DEĞERLENDİRME	Modülde 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 disiplinini kazanırsanız; başarılı olmamanız için hiçbir neden yoktur.

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

Bu modülle, SQL Server'ın yardımcı işlemlerini kullanabilmeyi öğreneceksiniz.

ÖĞRENME FAALİYETİ-1

AMAÇ

T-SQL komutlarıyla oluşabilecek hataları yakalamayı öğreneceksiniz.

ARAŞTIRMA

- Bu modüle gelene kadar hataları gidermek için ne tür çözümler uyguladınız? Hatırlayınız.

1. HATA YAKALAMA

Hata, SQL Server'ın çalışması sırasında ortaya çıkan bir durumdur. Hatalar çoğunlukla kullanıcı kaynaklıdır. Örneğin, kullanıcı veritabanında var olan bir sütun oluşturmak istediğinde aynı isimde bir sütun adı olduğundan hata oluşur.

SQL Server'da hata mesajları veritabanında saklanmaktadır. Saklanan veritabanı hata mesajlarını kullanmak merkezi olarak kontrol açısından fayda sağlamaktadır. Ayrıca, oluşan hataları işletim sisteminin hata günlüklerine de yazdırabilirsiniz.

SQL Server'da tanımlı olan hata mesajları "sys.messages" sistem tablosunda saklamaktadır. Var olan bu sistem tablosuna yeni mesajlar da eklenebilir.

1.1. Sistem Mesajları

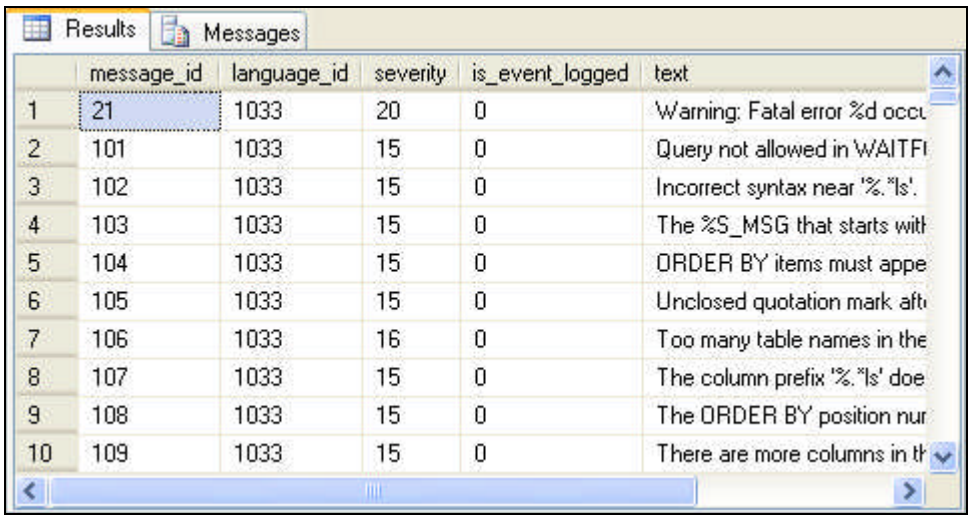
Sistem mesajlarının saklandığı sys.messages tablosuna her veritabanından doğrudan erişilebilmektedir. Sys.messages, 70000'e yakın hata mesajını farklı dillerde olmak üzere içinde barındıran bir sistem view'idir. Bu mesajlar, uygulamada kullanılan yapılar tarafından (Saklı prosedür, fonksiyon vb.) kullanılmaktadır.

Sys.messages tablosunda kullanılan mesajların kod ve açıklamalarını bir SELECT ifadesiyle görebilirsiniz.

```
SELECT * FROM sys.messages
```

Resim 1.1: SELECT ifadesiyle sistem mesajlarını görüntüleme

Bu ifadeyi yazıp F5 ile sorgunuzu çalıştırdığınızda sistemde kayıtlı olan mesajları ve kodlarını görebilirsiniz.



Results | Messages

	message_id	language_id	severity	is_event_logged	text
1	21	1033	20	0	Warning: Fatal error %d occu
2	101	1033	15	0	Query not allowed in WAITFI
3	102	1033	15	0	Incorrect syntax near '%.*ls'.
4	103	1033	15	0	The %S_MSG that starts with
5	104	1033	15	0	ORDER BY items must appe
6	105	1033	15	0	Unclosed quotation mark aft
7	106	1033	16	0	Too many table names in the
8	107	1033	15	0	The column prefix '%.*ls' doe
9	108	1033	15	0	The ORDER BY position nur
10	109	1033	15	0	There are more columns in th

Resim 1.2: Kayıtlı sistem mesajları

Results ekranında bulunan sütun başlıklarının anlamları ise Tablo1.1'de gösterilmiştir.

Sütun Adı	**Değer Aralığı**	**Anlamı**
message_id	0-49999 arası	Sistem hata mesajı kodları için ayrılan değer
	50000	RAISERROR fonksiyonuyla anlık olarak üretilen hata mesajları
	50001 ve üzeri	Kullanıcı tarafından tanımlanan hata mesaj kodları
severity	0 veya 10	Kullanıcı veri girişinden kaynaklanan hata
	11-16 arası	Kullanıcının düzeltebileceği bir hata
	17	Yetersiz kaynak (Diskin dolu olması veya tablonun salt okunur olması vb)
	18	Yazılımdan kaynaklanan hata
	19	Constraint'lere takılan bir hata
	20-25 arası	Çok kritik hatalar (Server yöneticisi ekleyebilir)
is_event_logged	0-1	Bu tür bir hata oluştuğunda log dosyasına yazılıp yazılmayacağını belirler
text	%s, %d ve mesaj	Mesajın anlamı, %s ve %d ile ek parametre kullanımı
language_id		Hata mesajının hangi dilde olacağını belirler

Tablo 1.1: "sys.messages" tablosunda bulunan sütun adları ve anlamları

1.1.1. Yeni Mesaj Eklemek

Kendi uygulamalarınız için sistem mesajlarına yenilerini ekleyebilirsiniz.

Mesaj numaralarının ilk 50000'i SQL Server için ayrılmıştır. Haliyle eklenecek yeni mesajların numaraları 50000'den büyük olmalıdır.

Yeni mesaj eklemek için "sp_addmessage" sistem saklı prosedürü kullanılır.

Yazılışı

```
sp_addmessage @msgnum ='mesaj_kodu',
              @severity = 'seviye',
              @msgtext = 'mesaj',
              @with_log = 'true ya da false',
              @lang = 'dil_kodu'
```

Örnek

```
sp_addmessage @msgnum = 50001,
              @severity = 11,
              @msgtext = 'Bu isimde bir öğrenci yok'
              @with_log = 'true'
```

Eklediğiniz mesajı görmek için message_id'si 50000'den büyük olan mesajları listelemeniz yeterlidir.

```
SELECT * FROM sys.messages
WHERE message_id>50000
```

Resim 1.3: Kullanıcı tanımlı mesajları görmek için kullanılan sorgu

1.1.2. Mesaj Silmek

Kullanıcı tanımlı mesajları silmek için "sp_dropmessages" sistem saklı prosedürü kullanılır.

Yazılışı

```
sp_dropmessage 'mesaj_kodu'
```

Örnek

```
sp_dropmessage 50002
```

1.2. RAISERROR Fonksiyonu

Sisteme eklenen mesajların hata oluşması durumunda devreye girmesi için kullanılan fonksiyondur.

İki farklı amaç için kullanılır.

Birinci amaç, sistemde var olan hata mesajlarından hareketle bir hata oluşumunu meydana getirmek içindir.

Bu amaç için genel kullanımı şöyledir.

RAISERROR (mesaj_kodu, seviyesi, durum) [WITH LOG]

- Mesaj_kodu: message_id sütunundaki koda karşılık gelir.
- Seviye: 0-25 arasında bir sayı olup, mesajın kritik seviyesini gösterir.
- Durum: 1-127 arasında bir sayı olup, hata mesajı birden fazla yerde oluştuğunda hata oluşan yerleri ayırt etmek için kullanılır.
- WITH LOG: Hatanın loglara mutlaka yazılmasını sağlar.

İkinci amaç, anlık olarak türetilen bir hata mesajından harekele hata oluşumunu meydana getirmek içindir.

İkinci amaç için genel kullanımı ise şöyledir.

RAISERROR (‘mesaj’, seviyesi, durum) [WITH LOG]

İkinci amacın birinci amaçtan farkı mesajın doğrudan yazılmasıdır. Diğer parametreleri aynıdır.

Örnek

Daha önceden uygulaması yapılan “Personel” veritabanındaki “Person_Bilgi” tablosu göz önüne alınacaktır. “Person_Bilgi” tablosunda sicil numarası girilen personelin bilgileri listelenmek istenmektedir.

Table - dbo.Person_Bilgi

Sicil_No	Ad	Soyad	Cins
1233	Ali	CANDAN	Erke
1234	Semiha	İPEK	Kadı
1235	Nazlı	DENİZ	Kadı
1236	Tamer	DEMİRAY	Erke
1237	Mehmet	AYDIN	Erke
1238	Nevzat	GÜVEN	Erke
1239	Hüsamettin	ÇELİK	Erke
1241	Ahmet	AKKAYA	Erke
1242	Hüseyin	ATAKENT	Erke
1244	Toprak	TÜRKER	Erke

Resim 1.4: “Person_Bilgi” tablosu

Buna göre, sicil numarası girilen personelin kayıtlarının gösterilmesi ile ilgili bir uygulama RAISERROR fonksiyonunu kullanılarak yapılacaktır.

İlk önce, hata oluşması durumunda verilecek mesajı sisteme tanımlamak gerekir.

```
sp_addmessage @msgnum=50001,
@severity=11,
@msgtext='Girilen sicilno 0 dan küçük ve 1500 den büyük olamaz',
@with_log='true'
```

Resim 1.5: Sisteme yeni mesajın eklenmesi

Resim 1,5'teki kod satırlarını Query'e yazıp execute edilince 50001 numaralı hata mesajı sisteme eklenmiş olur.

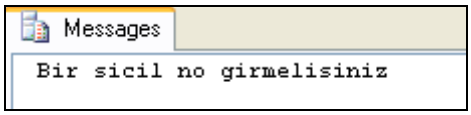
Daha sonra, verilen sicil numarasına göre personelin tablo içerisinden bulunarak listelenip listelenmeyeceğinin belirleneceği "sp_liste" adında bir saklı prosedür oluşturulur.

```
CREATE PROCEDURE sp_liste(
    @sicil int=NULL
)
AS
IF @sicil IS NULL
BEGIN
    RAISERROR('Bir sicil no girmelisiniz',10,1)
    RETURN 0
END
IF @sicil<0 OR @sicil>1500
BEGIN
    RAISERROR(50001,10,1)
    RETURN 0
END

SELECT * FROM Person_Bilgi
WHERE Sicil_No=@sicil
```

Resim 1.6: Saklı prosedürün oluşturulması

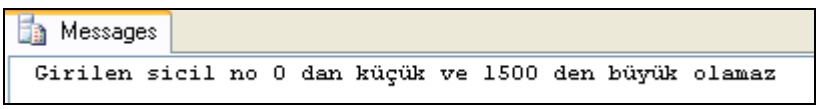
Saklı prosedürde @sicil değişkeni int tipinde tanımlanarak NULL değeri atanmıştır. Saklı prosedürün çalıştırılması esnasında @sicil değişkeninin değeri olarak bir değer atanmadıysa hata mesajının kullanıcıya gösterilmesi RAISERROR fonksiyonuyla gerçekleştirilmiştir.

RAISERROR fonksiyonu bu şekilde de kullanılabilir. RAISERROR fonksiyonunun sistem hata mesajlarına eklenmeden bu şekilde tanımlanmasına doğaçlama hata mesajı tanımlama denir.



Resim 1.7. Doğaçlama hata mesajı ekran görüntüsü

Tabloda bulunan sicil numaralarının bir aralıkta verildiği göz önünde bulundurularak personel kaydının listelenmesi için girilen sicil numarasının da kontrol edilmesi gerekmektedir. Bu da, IF yapısı kullanılarak yapılmıştır. IF yapısında, sicil numaralarının 0'dan küçük veya 1500'den büyük olmaması istenmektedir. Girilen sicil numarası 0'dan küçük veya 1500'den büyük olması durumunda sisteme eklenen hata mesajı görüntülenecektir.

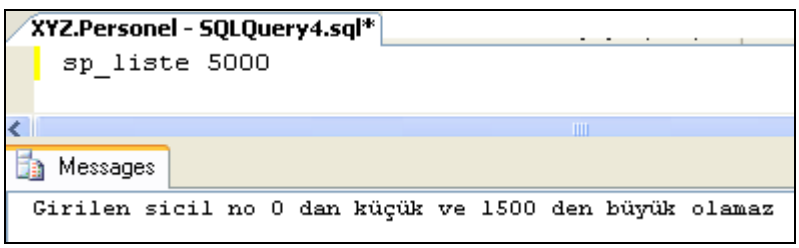


Resim 1.8: Sisteme eklenen hata mesajının görüntülenmesi

Son olarak, SELECT satırıyla, girilen sicil numarasının tabloda bulunan bir sicil numarasıyla eşleşmesi durumunda personelin kayıtlarını gösterecek kodlar bulunmaktadır.

Resim 1.9: Tabloda bulunan kaydın gösterilmesi

Tüm bunlar "sp_liste" saklı prosedürüne verilecek bir değer ve saklı prosedürün çalıştırılmasıyla olmaktadır.



Resim 1.10: Sicil numarasının belirtilen değer dışında girilmesi

Resim 1.11: Girilen değerin tabloda bulunan sicil numarasıyla eşleşmesi

Oluşabilecek bir hata durumu için hata mesajının tanımlanması veya RAISERROR fonksiyonuyla kontrol altına alınması yeterli olmaz. Çünkü RAISERROR fonksiyonu, hatayı günlük dosyalarına yazma ve hatayı bir üst ortama aktarma işlevlerini yapar. Dikkat edilmesi gereken diğer bir nokta ise, bir saklı prosedür, aynı anda sadece bir hata koduna erişebilir.

1.3. Parametreli Hata Mesajı Tanımlamak

Bazı durumlar için, oluşabilecek hatalarda kullanılmak üzere birtakım değerlerin dışarıdan alınması gerekir. Alınan bu değerler, sisteme bir hata mesajı veya doğaçlama hata mesajı tanımlarken kullanılır.

Dışarıdan parametre alabilmek için kullanılacak yapı şöyledir.

Yazılışı

'Hata mesajı %p1…….%p2…..',değer1,değer2 şeklindedir.

%p ile ifade edilen parametre bilginin veri türüne göre değişiklik gösterir.

%p ile ifade edilen parametre veri türleri Tablo 1.2'de gösterilmiştir.

% karakteri	İfade ettiği veri türü
d veya i	Decimal (onluk tam sayı)
o	Unsigned Octal (işaretsiz sekizlik sayı)
p	Pointer (işaretçi)
s	String (karaktersel bilgi)
u	Unsigned integer (işaretsiz tam sayı)
x veya X	Unsigned Hexadecimal (işaretsiz onaltılık sayı)

Tablo 1.2: Parametere için veri türleri

Örnek

Şirket sisteminde personel silme işlemini yapan "sp_personelSil" saklı prosedürünün, personeli silen kullanıcı bilgilerini ve hatayı günlüğe (log) kaydetmesi istenmektedir.

İlk önce hata mesajının sisteme tanımlanması gerekmektedir.

```
sp_addmessage @msgnum=50002,
              @severity=10,
              @msgtext=' %d personel %s tarafından silinmiştir',
              @with_log='true'
```

Resim 1.12: Hata mesajının sisteme tanıtılması

Resim 1.12'deki kod satırlarını Query'e yazıp F5 ile çalıştırınca 50002 numaralı hata mesajı sisteme eklenmiş olur.

Daha sonra, verilen sicil numarasına ve kullanıcı adına göre personelin tablo içerisinden bulunarak silinip silinmeyeceğinin belirleneceği "sp_personelSil" adında bir saklı prosedür oluşturulur.

```
CREATE PROCEDURE sp_personelSil(
    @sicil INT=NULL,
    @kullaniciAd VARCHAR(50)=NULL
)
AS
IF @sicil IS NULL
BEGIN
    RAISERROR('Bir sicil numarası giriniz',11,1)
    RETURN 0
END
IF @kullaniciAd IS NULL
BEGIN
    RAISERROR('Bir kullanıcı adı giriniz',11,1)
    RETURN 0
END

BEGIN
    DELETE FROM Person_Bilgi WHERE Sicil_No=@sicil
    RAISERROR(50002,11,1,@@rowcount,@kullaniciAd)
    RETURN 0
END
```

Resim 1.13: Saklı prosedürün oluşturulması

Saklı prosedürde @sicil değişkeni int tipinde, @kullaniciAd değişkeni de string tipte tanımlanarak NULL değeri atanmıştır. Saklı prosedürün çalıştırılması esnasında @sicil değişkeninin değeri ve @kullaniciAd değikeninin değeri olarak bir değer atanmadıysa hata mesajının kullanıcıya gösterilmesi RAISERROR fonksiyonuyla doğaçlama olarak gerçekleştirilmiştir.

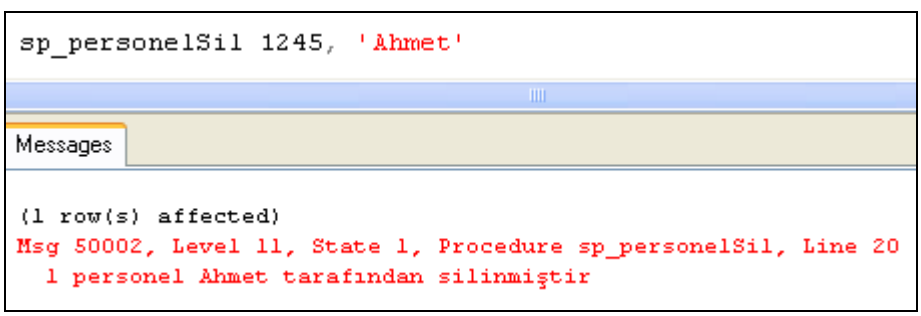
DELETE FROM satırıyla girilen sicil numarasının tabloda bulunan bir sicil numarasıyla eşleşmesi durumunda personelin kaydının silen kodlar bulunmaktadır. @@rowcount fonksiyonu işlemden etkilenen kayıtların adedini int tipinde tutmaktadır. RAISERROR fonksiyonuyla da 50002 numaralı hata mesajında bulunan hata yazdırılmakta, @@rowcount ile etkilenen kayıtların adedi ve @kullaniciAd ile de personeli silen kullanıcının adı belirtilmektedir.

Tüm bunlar "sp_personelSil" saklı prosedürüne verilecek iki değer ve saklı prosedürün çalıştırılmasıyla olmaktadır.

```
sp_personelSil 1245, 'Ahmet'
```

Resim 1.14: Saklı prosedüre değer gönderilmesi

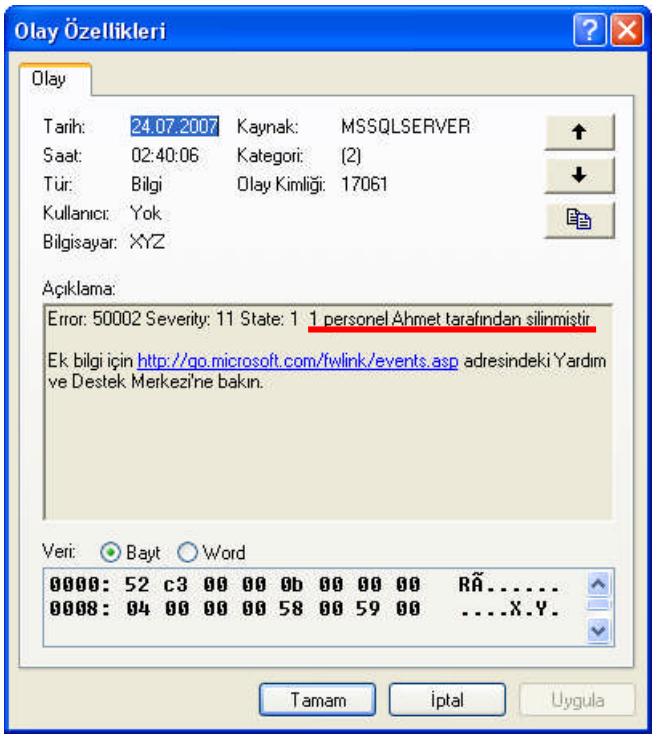
Bu yazıma göre 1245 sicil numaralı personel 'Ahmet' kullanıcısı tarafından silinmek istenmektedir. Query çalıştırıldığında sonuç mesaj penceresinde görüntülenecektir.



Resim 1.15: Sorgunun çalıştırılması ve işlemin sonucu

Messages penceresindeki mesajın kırmızı yazı rengiyle gösterilmesinin sebebi, sisteme eklenen hata mesajı seviyesinin 10 olmasına rağmen saklı prosedürde bu seviyenin 11 olarak verilmesine bağlıdır. Eğer, saklı prosedür "Modify" ile açılıp içerisinde geçen seviyeler 10 olarak değiştirildiğinde bu penceredeki mesaj siyah yazı rengiyle gösterilecektir.

Günlük (log) dosyasına kaydedilen bu mesajı "Denetim Masası/Yönetimsel Araçlar/Olay Görüntüleyicisi" ile görebilirsiniz. Uygulama seçimindeki ilk bilgiyi kontrol ederek günlüğe yazılan hatayı bulabilirsiniz.



Resim 1.16: Olay görüntüleyicisinde hatanın görünümü

1.4. @@ERROR Fonksiyonu

Sistemde oluşan en son hata kodunu tutan fonksiyondur. Yeni bir hata meydana geldiğinde eski değer silinir ve yeni değer tutulmaya başlanır. Geriye dönen değer integer tipinde bir değerdir.

Örnek

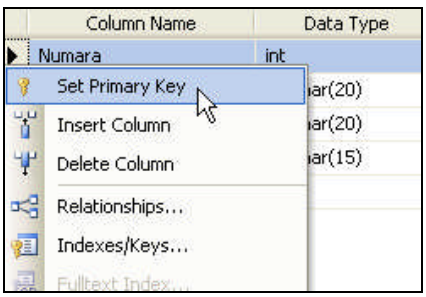
Kursiyer adında bir tablonun olduğunu ve bu tabloda kursiyerlerin numara, ad, soyad ve e-postalarının tutulduğunu varsayınız.

Table - dbo.Kursiyer

	Numara	Ad	Soyad	e_mail
▶	1	Erkan	DUYAR	a@a.com
	2	Ali	UYAR	q@q.com
	3	Defne	KARTAL	b@b.com
	4	Zeki	UYMAZ	e@e.com
*	NULL	NULL	NULL	NULL

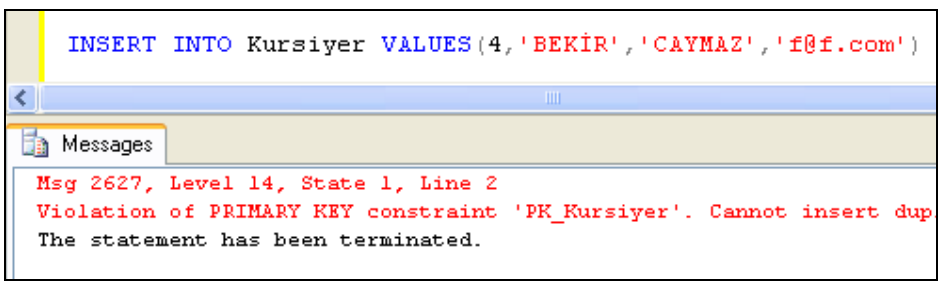
Resim 1.17: Kursiyer tablosu

Tabloda kursiyerlere verilen numaraların her kursiyer için farklı olması zorunluluğu vardır. Bunun için, "Numara" sütununun birincil anahtar olarak belirlenmesi gerekmektedir.



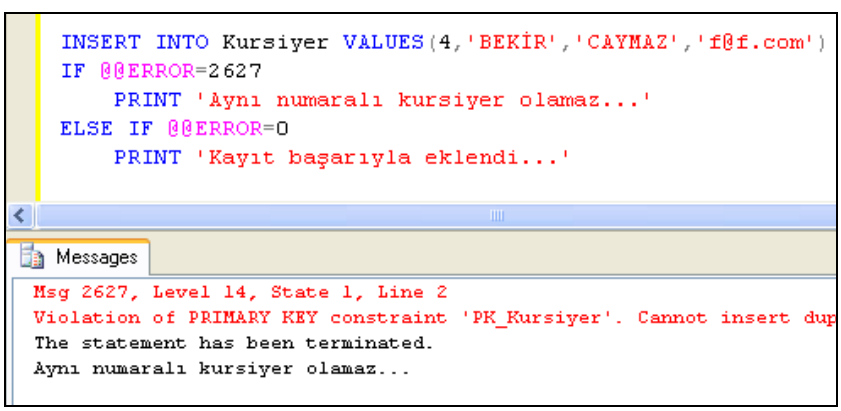
Resim 1.18: Birincil anahtarın belirlenmesi

Dolayısıyla, tabloya aynı numaralı bir kursiyer eklenmek istendiğinde hata meydana gelecektir. Resim 1.19'daki gibi tabloya yeni bir kayıt eklemeye çalışınız ve hataya dikkat ediniz.



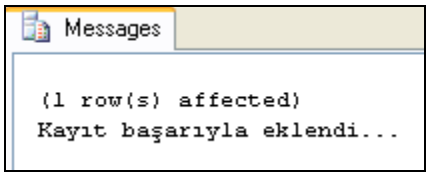
Resim 1.19: Tabloya kayıt eklenmesi

"Numara" sütununa birincil anahtar verildiği için işlem durdurulacaktır. Hata oluşması veya işlemin gerçekleşmesi durumunda uygun bir mesajı kodlarınıza Resim 1.20'deki gibi ekleyiniz ve sorguyu çalıştırınız.



Resim 1.20: Uygun mesajın eklenmesi

Eğer, tabloda aynı numaralı kursiyer varsa 2627 numaralı hata oluşacak ve IF şartıyla belirtilen mesaj yazdırılacaktır. Hiçbir hatanın olmaması durumundaysa ikinci IF şartı gerçekleşecek ve ilgili mesajı yazdırılacaktır.



Resim 1.21: Hatasız durumun gerçekleşmesi

1.5. TRY-CATCH Yapısı

Try-Catch yapısı, SQL Server'ın 2005 versiyonunda yer alan yeni bir özelliğidir. Önceki sürümlerde bu yapı yoktur. Bir Try ve bir Catch bloğundan oluşur. Try bloğunda bir hata olursa kontrol Catch bloğuna geçer. Hata yakalanıp Catch bloğunun işletilmesi tamamlandıktan sonra akış bloktan sonraki kodlarla devam eder. Eğer, Try bloğundaki ifadelerin yürütülmesi esansında bir sorun çıkmazsa Catch bloğu devreye girmez. Program akışı Catch bloğundan sonraki ifadeyle devam eder. Dikkat edilmesi gereken bir nokta ise,Try bloğu tanımlandıysa Catch bloğu da tanımlanmak zorundadır.

Yazılışı

```
BEGIN TRY
        {SQL Kodlar}
END TRY
BEGIN CATCH
        {SQL Kodlar}
END CATCH
```

Try-Catch yapısında severity seviyesi 10'da büyük bir hata oluştuğunda kontrol Catch bloğuna geçer. Daha düşük seviyedeki hatalar dikkate alınmaz.

Örnek

Basit olması açısından Try bloğu içerisinde int tipinde bir değişken tanımlanarak bu değişkene string bir bilgi atanmıştır.

Dolayısıyla hata meydana gelecektir. Oluşan hata sonucu Catch bloğu devreye girecektir.

```
BEGIN TRY
    DECLARE @sayi INT
    SET @sayi = '02/02/2007'
    PRINT @sayi
    PRINT 'TRY bloğu bitti'
END TRY

BEGIN CATCH
    PRINT 'Hata Oluştu..Catch bloğu devreye girdi'
END CATCH

PRINT 'Bitti'
```

Messages

```
Hata Oluştu..Catch bloğu başladı
Bitti
```

Resim 1.22: Try-Catch yapısının kullanımı

@sayi değişkeni int tipinde tanımlanmıştır. Ancak SET ifadesiyle string bir değer aktarılmak istenmiştir. Dolayısıyla hata oluşmuş ve Catch bloğu devreye girerek hata mesajı yazdırılmıştır. F5 ile sorguyu çalıştırdığınızda sonucu Messages penceresinde görebilirsiniz.

@sayi değişkenine int tipinde bir sayı aktarılsaydı hata oluşmayacak ve sonuç farklı olacaktı.

```
BEGIN TRY
    DECLARE @sayi INT
    SET @sayi = 12345
    PRINT @sayi
    PRINT 'TRY bloğu bitti'
END TRY

BEGIN CATCH
    PRINT 'Hata Oluştu..Catch bloğu devreye girdi'
END CATCH

PRINT 'Bitti'
```

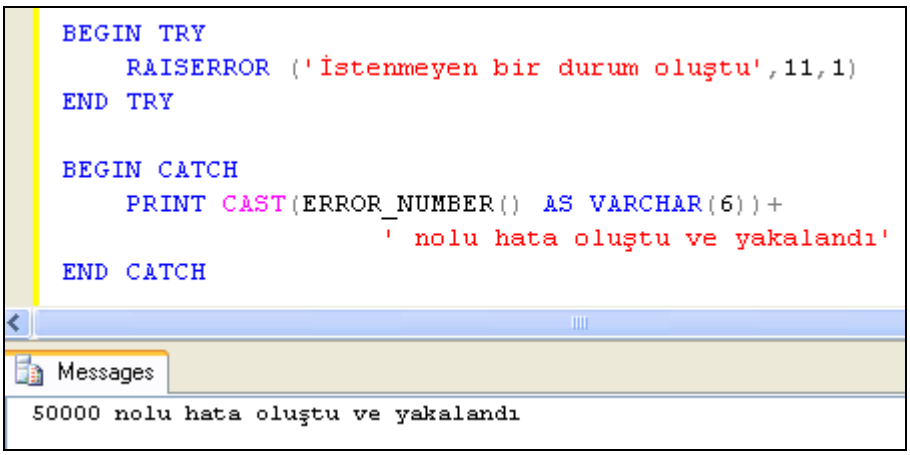
Messages

```
12345
TRY bloğu bitti
Bitti
```

Resim 1.23: Try bloğunun devreye girmesi

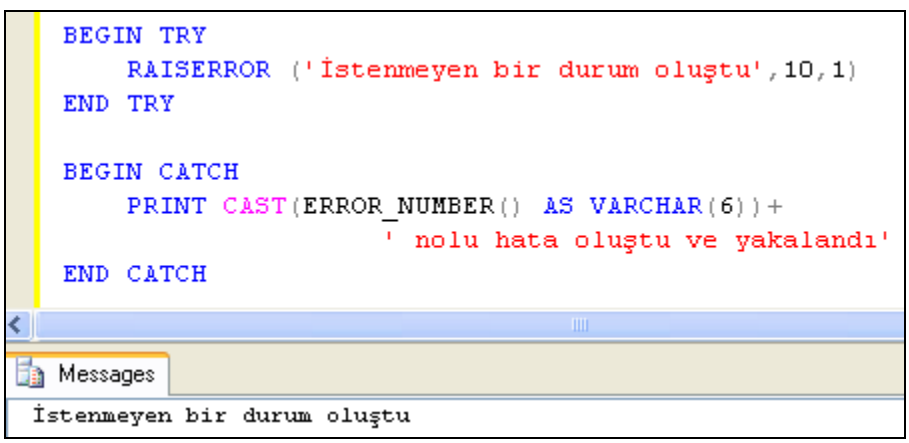
Sayının uygun olarak verilmesi sonucu Try bloğu işletilmiş ve Catch bloğu işletilmemiştir. Catch bloğundan sonra gelen ifade işletilerek akışa devam edilmiştir.

Try-Catch yapısı içinde RAISERROR fonksiyonuyla hata yakalamaya bir göz atalım.



Resim 1.24: Try-Catch yapısında RAISERROR kullanımı

RAISERROR fonksiyonuyla doğaçlama bir hata mesajı oluşturularak akışın Catch bloğuna yönlendirilmesi sağlanmaktadır. Oluşan hata numarası da ERROR_NUMBER fonksiyonuyla yazdırılmaktadır. ERROR_NUMBER() fonksiyonu @@ERROR fonksiyonuna göre daha kullanışlıdır. RAISERROR fonksiyonunda severity seviyesi 11 olarak değil de 10 olarak verilseydi Catch bloğu devreye girmeyecek ve RAISERROR ile belirtilen mesaj yazdırılacaktı. Bunun sebebi, seviye numarasıdır.



Resim 1.25: Seviye numarasının 10 verilmesi sonucu

Girilen int tipindeki sayıya, faklı tipten bir değer aktarıldığında ise ERROR_NUMBER fonksiyonu ile elde edilen sonuçla beraber hatanın numarası da verilecektir.

```
BEGIN TRY
    DECLARE @sayi INT
    SET @sayi = '02/02/2007'
    PRINT @sayi
    PRINT 'TRY bloğu bitti'
END TRY

BEGIN CATCH
    PRINT 'Catch bloğu devreye girdi'
    PRINT CAST(ERROR_NUMBER() AS VARCHAR(6))+
                    ' nolu hata oluştu ve yakalandı'
END CATCH
```

Messages

```
Catch bloğu devreye girdi
245 nolu hata oluştu ve yakalandı
```

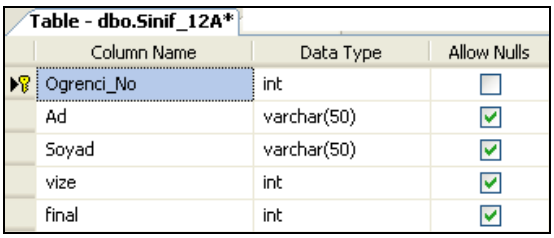
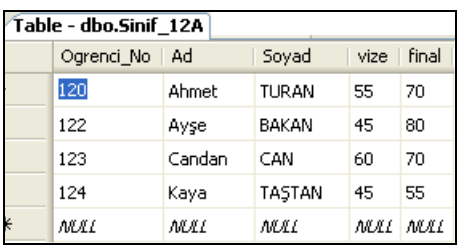
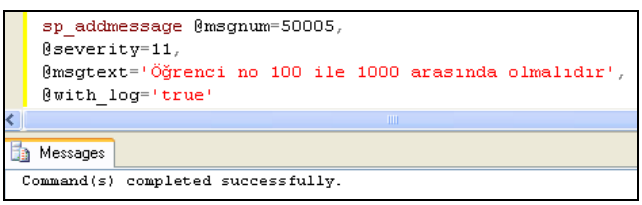
Resim 1.26: ERROR_NUMBER fonksiyonuyla hata numarasının gösterilmesi

Catch yapısı içerisinde kullanılan ERROR_NUMBER() fonksiyonundan başka daha fazla bilgi sahibi olmanızı sağlayacak fonksiyonlar da vardır. Bunlar, Tablo 1.3'te gösterilmiştir.

Fonksiyon Adı	**Elde edilecek değer**
ERROR_NUMBER()	RAISERROR ile belirtilen hata veya sys.messages tablosundaki hata kodu
ERROR_MESSAGE()	Mesaj metni
ERROR_SEVERITY()	Kritiklik durumu
ERROR_STATE()	Sisteme dönük kritiklik seviyesi
ERROR_PROCEDURE()	Hatanın oluşumuna neden olan prosedür
ERROR_LINE()	Hataya neden olan satır

Tablo 1.3: Catch bloğunda kullanılabilen fonksiyonlar

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
➢ "Hata_Yakalama" adında bir veritabanı oluşturunuz.	➢ CREATE DATABASE
➢ Veritabanında "Sinif_12A" adında bir tablo oluşturunuz. Tablo sütunları Ogrenci_No (int), Ad,Soyad (VARCHAR), Vize,Final(int) tipinde olsun.	 **Resim 1.27: "Sinif_12A" tablosu**
➢ "Ogrenci_No" sütunu birincil anahtar olarak ayarlansın.	➢ Resim 1.27
➢ "Sinif_12A" tablosuna birkaç kayıt giriniz.	 **Resim 1.28: Tabloya kayıt girilmesi**
➢ Girilen öğrenci numarasının belirli bir aralıkta olacağını göz önüne alarak sisteme kritik seviyesi 11 olan 50005 numaralı bir hata mesajı ekleyiniz.	 **Resim 1.29: Hata mesajının sisteme eklenmesi**
➢ Verilen öğrenci numarasına göre öğrencinin tablo içerisinden bulunarak listelenip listelenmeyeceğinin belirleneceği "sp_ogrliste" adında bir saklı prosedür oluşturunuz.	

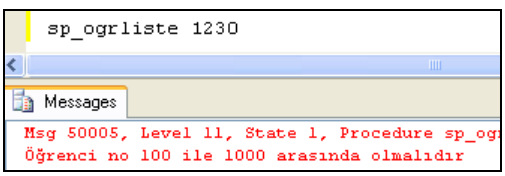
```
CREATE PROCEDURE sp_ogrliste(
    @ogrno int=NULL
)
AS
IF @ogrno IS NULL
BEGIN
    RAISERROR('Bir öğrenci no girmelisiniz',11,1)
    RETURN 0
END
IF @ogrno<100 OR @ogrno>1000
BEGIN
    RAISERROR(50005,11,1)
    RETURN 0
END

SELECT * FROM Sinif_12A
WHERE Ogrenci_No=@ogrno
```

Resim 1.30: "ogrliste" saklı prosedürünün oluşturulması

- "sp_ogrliste" saklı prosedürüne bir değer vererek prosedürü çalıştırınız.

Resim 1.31: "sp_ogrliste" saklı prosedürünün çalıştırılması

- "sp_ogrliste" saklı proseüdürüne verilen değeri belirlenen sınırlar dışında bir değer vererek yeniden çalıştırınız.



Resim 1.32: RAISERROR ile belirlenen hata mesajının görüntüsü

ÖLÇME VE DEĞERLENDİRME

Aşağıdaki Doğru/Yanlış, çoktan seçmeli ve boşluk doldurma sorularını cevaplayınız.

1. Sistemde tanımlı olan hata mesajları …… tablosunda tutulur.

2. "severity", hatanın kritik seviyesini gösterir (D/Y).

3. Aşağıdakilerden hangisi sisteme mesaj eklemek için kullanılan parametrelerden değildir?
A) With_Log B) Lang C) Message_Id D) Msgtext

4. Sisteme kayıtlı bir mesajı silmek için …… saklı prosedürü kullanılır.

5. Anlık üretilen hata mesajına "Doğaçlama Hata Mesajı" denir (D/Y).

6. En son oluşan hatayı tutan fonksiyon, ……. fonksiyonudur.

7. Try bloğunda bir hata olursa, Catch bloğu işletilmez (D/Y).

8. Try-Catch blokları, ……. ile başlar ve ……. ile biter.

9. Aşağıdakilerden hangisi Catch bloğu içinde kullanılan fonksiyonlardan değildir?
A) ERROR_STATE()
B) ERROR_SEVERITY()
C) ERROR_LINE()
D) ERROR_LANG()

10. Hataya neden olan satır, …… fonksiyonuyla gösterilir.

DEĞERLENDİRME

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

ÖĞRENME FAALİYETİ-2

AMAÇ

SQL Server'da veritabanı performansı takip edebilecek ve iyileştirme yapabileceksiniz.

ARAŞTIRMA

- Daha önceden öğrendiğiniz veritabanı paket programında performans, güvenlik ve yedekleme işlemlerinin nasıl yapıldığını araştırınız.

2. PERFORMANS VE İYİLEŞTİRME

2.1. Transaction (İşlem)

Bir bütünü oluşturmak için birden fazla iş yapmak gerekebilir. Yapılan bir işin eksik olması iş bütünlüğünün oluşmasını engelleyecektir. Bulunduğu durumdan daha küçük parçalara ayrılamayan işleme transaction (işlem) denir.

Bir işin tamamının yapılması isteniyorsa transaction kullanılmalıdır. Transaction bloğunda tüm işlemler düzgün olarak gerçekleşmelidir. Bu işlemlerden birkaçı geçerli ya da kabul edilebilir durumda olsalar bile tek bir işlemin gerçekleşmemesi demek hiçbir işlemin gerçekleşmemesi ve kabul edilmemesi demektir.

Transaction bloğu, BEGIN TRANSACTION ile başlatılır. Transaction bloğu başlatıldığı anda, yapılan işlemlerin bütün olarak ele alınacağı ve başarısız bir işlem sonucunda bloğun tamamının geçersiz olacağı belirtilmiş olur. Bu işlem, transaction günlükleriyle de desteklenir. BEGIN TRANSACTION ifadesi BEGIN TRAN olarak da yazılabilir.

Transaction bloğunda yapılan işlemlerin başarılı olup olmadığına bakılır. İşlemin başarısız olmasında geri dönüş işlemi ROLLBACK ile başlatılır. İşlemin başarılı olması durumunda bir sonraki adıma geçilir.

Tüm işlemler bittiğinde COMMIT ile yeni durumu sabitlenir. Tüm işlemlerin sonunda başarısız bir sonuç elde edildiyse ROLLBACK ile en başa alınır ve ilk durumuyla sabitlenir.

ROLLBACK ve COMMIT transaction'ın bitimini ifade eder.

SQL Server, üç farklı transaction moduna destek verir.

- **Harici (Explicit) Transcation:** Kullanıcı tarafından tanımlanır. BEGIN TRAN ifadesiyle başlar. Sonucun başarısız olması halinde COMMIT ifadesi ile işlemler gerçekleşmiş veya ROLLBACK ifadesi ile işlemler olmamış gibi sonlandırılabilir.
- **Dahili (Implicit) Transaction:** Belirli ifadelerin çalışmasından sonra devreye giren ancak kullanıcı tarafından sonlandırılması istenen transaction modudur.
- **Auto Commit:** Herhangi bir transaction şekli belirtilmediyse, SQL Server bu modda çalışır. Auto Commit modunda iken, her bir yığın, bir transaction bloğu olarak ele alınır. Yığın içerisinde bir sorun olursa, SQL Server otomatik olarak bütün yığını geri alır (ROLLBACK).

Transaction'ın genel kullanımı şu şekildedir.

BEGIN TRAN[SACTION] **transaction_adı**

Örnek

Bir banka şubesinde müşterilerin hesaplarını tutan basit bir tablo oluşturunuz.

Table - dbo.Hesap*

Column Name	Data Type	Allow Nulls
Hesap_Numara	char(20)	☐
Ad	varchar(55)	☑
Soyad	varchar(55)	☑
Sube	int	☑
Bakiye	float	☑
		☐

Resim 2.1: Hesap tablosu

Bu tabloya uygun veri tiplerinde birkaç kayıt giriniz.

Table - dbo.Hesap | XYZ.Transactio...SQLQuery1.sql*

Hesap_Numara	Ad	Soyad	Sube	Bakiye
123123123	Ahmet	ZENGİN	55	20000
321321321	Ali	GÜNEŞ	55	15000
987987987	Ayşe	AYŞECAN	55	50000
NULL	NULL	NULL	NULL	NULL

Resim 2.2: Tabloya kayıtların girilmesi

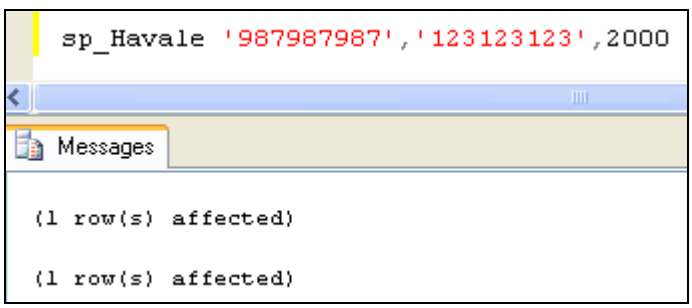
İşlemlerin yapılabilmesi için bir saklı prosedür oluşturunuz. Havale yapan müşterinin hesap numarası, havaleyi alacak müşterinin hesap numarası ve yapılacak havale miktarı bu saklı prosedüre dışarıdan bilgi olarak alınsın. Havale işlemi sırasında işlemlerde bir sorun

olmazsa havale kabul edilsin. Aksi takdirde, ROLLBACK ifadesiyle yapılan tüm işlemler geri alınsın. Bunun için, saklı prosedürü Resim 2.3'teki gibi oluşturabilirsiniz.

```
CREATE PROCEDURE sp_Havale(
    @gonderenHesapNo VARCHAR(20),
    @alanHesapNo VARCHAR(20),
    @havaleMiktari FLOAT
)
AS
BEGIN TRANSACTION
    UPDATE Hesap
    SET Bakiye=Bakiye-@havaleMiktari
    WHERE Hesap_Numara=@gonderenHesapNo
    IF @@ERROR<>0
    ROLLBACK

    UPDATE Hesap
    SET Bakiye=Bakiye+@havaleMiktari
    WHERE Hesap_Numara=@alanHesapNo
    IF @@ERROR<>0
    ROLLBACK

COMMIT
```

Resim 2.3: Havale saklı prosedürünün oluşturulması

Oluşturulan saklı prosedürü çalıştırmak için uygun parametreleri giriniz.



Resim 2.4: Saklı prosedürün çalıştırılması

İşlemin sonunda iki kayıtın etkilendiğini görebilirsiniz. Tabloyu açıp baktığınızda bir sorunla karşılaşılmadan havalenin yapıldığını göreceksiniz.

Table - dbo.Hesap

	Hesap_Numara	Ad	Soyad	Sube	Bakiye
▶	123123123	Ahmet	ZENGİN	55	22000
	321321321	Ali	GÜNEŞ	55	15000
	987987987	Ayşe	AYŞECAN	55	48000
*	NULL	NULL	NULL	NULL	NULL

Resim 2.5: Hesap tablosunun son hali

Eğer saklı prosedüre verilen değerleri yanlış verirseniz bir hatanın oluştuğunu ve havale işleminin gerçekleşmediğini görürsünüz.

Aynı işlemi Try-Catch yapısını kullanarak ta yapabilirsiniz. Bunun için oluşturduğunuz saklı prosedürü Modify komutuyla açıp Resim 2.6'daki gibi düzenleyebilirsiniz.

```
ALTER PROCEDURE [dbo].[sp_Havale](
    @gonderenHesapNo VARCHAR(20),
    @alanHesapNo VARCHAR(20),
    @havaleMiktari FLOAT
)
AS
BEGIN TRY
 BEGIN TRANSACTION
    UPDATE Hesap
    SET Bakiye=Bakiye-@havaleMiktari
    WHERE Hesap_Numara=@gonderenHesapNo

    UPDATE Hesap
    SET Bakiye=Bakiye+@havaleMiktari
    WHERE Hesap_Numara=@alanHesapNo
 COMMIT
END TRY
BEGIN CATCH
    PRINT @@ERROR +' nolu hata oluştu'
    ROLLBACK
END CATCH
```

Resim 2.6: Try-Catch yapısıyla kullanımı

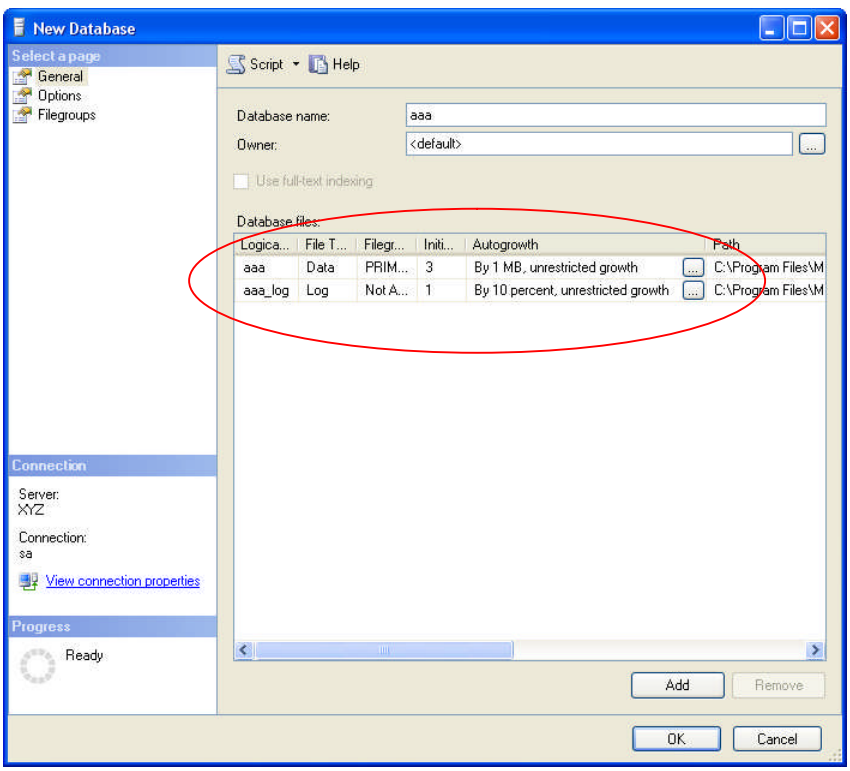
2.2. Transaction Log (İşlem Günlüğü) Dosyası

Transaction log, veritabanı bilgilerini saklamak için kullanılır. Her veritabanı, birer log dosyası ve veri dosyası içermektedir. Veritabanında oluşan değişimleri tutan log dosyası, oluşan değişiklikleri veritabanı için ayrılmış alanda saklar.

Veritabanında yapılan tüm düzenlemeler veritabanına yazılmadan önce transaction log dosyasına yazılır. Böylece, veritabanında meydana gelebilecek olumsuz durumlar sonucunda veritabanını kurtarmak için bu dosyalar kullanılabilir.

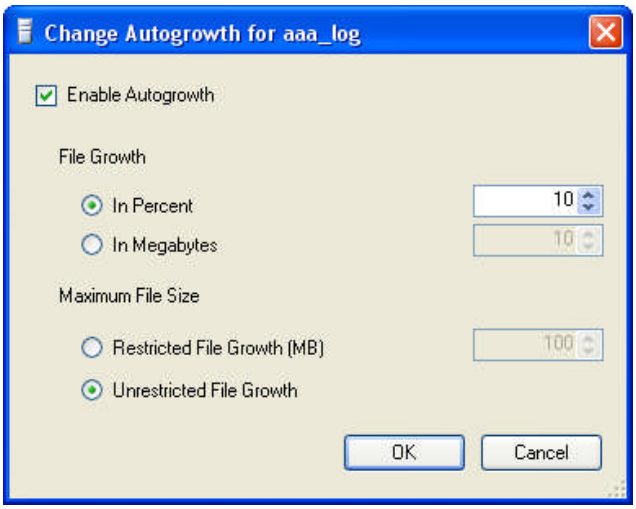
Oluşturulan her veritabanı bir birincil dosya (primary file-*.mdf) ve işlem günlüğü (transaction log-*.ldf) dosyalarına sahiptir. Transaction log dosyaları veri dosyaları gibi bir çok sayfadan oluşmaz. Sadece, günlük kayıt bilgilerini tutar. Log dosyaları varsayılan olarak %10 büyüme oranına ayarlanmıştır. Ancak isteğe göre de düzenlenebilmektedir.

SQL Server'da yeni bir veritabanı oluşturduğunuzu varsayınız. Bunun için Object Explorer'da Databases üzerinde sağ tıklayarak açılan menüden New Database komutunu verdiğinizde New Database iletişim penceresi ekrana gelecektir.



Resim 2.7: New Database iletişim penceresi

New Database iletişim penceresindeki Database Files alanında oluşturulacak veri ve log dosyası görüntülenmektedir. Log dosyası satırındaki otomatik büyüme (Autogrowth) alanında bulunan düğmeye  tıklandığında varsayılan olarak %10 büyüme oranına sahip log dosyasının büyüme özelliklerini değiştirebilirsiniz.



Resim 2.8: Otomatik büyüme özelliklerinin değiştirilmesi

2.3. Veritabanını Yedekleme ve Geri Yükleme

2.3.1. Veritabanı Yedekleme

Bilgisayarınızda donanımdan veya yazılımdan kaynaklanan problemler çıkabilir. Ayrıca sisteme virüs bulaşmasından veya SQL Server'ın kullanımından kaynaklanan hatalar da olabilir. Örneğin, UPDATE veya DELETE komutlarının yanlış kullanılması veritabanınızda sıkıntılara sebep olabilir.

SQL Server'da veritabanıyla çalışırken de yedekleme yapabilirsiniz. Yedekleme esansında veritabanının yapısını değiştirecek tablo oluşturmak, silmek gibi işlemler yapılamaz.

Veritabanı yedeklenirken, SQL Server şema ve dosya yapısı, veri ve transaction log dosyaları yedeklenir. Yedekleme işlemleri başlangıçtan itibaren transaction log dosyasına yazılır. SQL Server orijinal dosyaların yerini kaydeder ve geri yükleme işleminde bu yedekleri orijinal yerlerine yeniden oluşturur.

İki çeşit yedekleme şekli vardır.

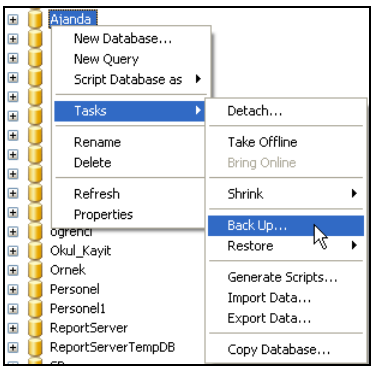
- **Tam Yedekleme (Full Backup):** Veritabanının tam yedeklenmesi işlemidir. Veritabanına yazılmamış transaction log'da bulunan veriler de yedeklenir.
- **Fark Yedekleme (Differential Backup):** En son alınan tam yedeklemeden sonra değişen kayıtları yedekler. Tam yedeklemeye göre saklanan veriler daha küçüktür. Bu yüzden yedekleme işlemi daha hızlıdır. Fark yedekleme yapılabilmesi için önceden tam yedeklemenin yapılmış olması gerekir.

Yedeklemeyi Managemet Studio'yu veya T-SQL kodlarını kullanarak yapabilirsiniz.

T-SQL kodlarıyla veritabanını yedeklemek için aşağıdaki genel yapıyı kullanabilirsiniz.

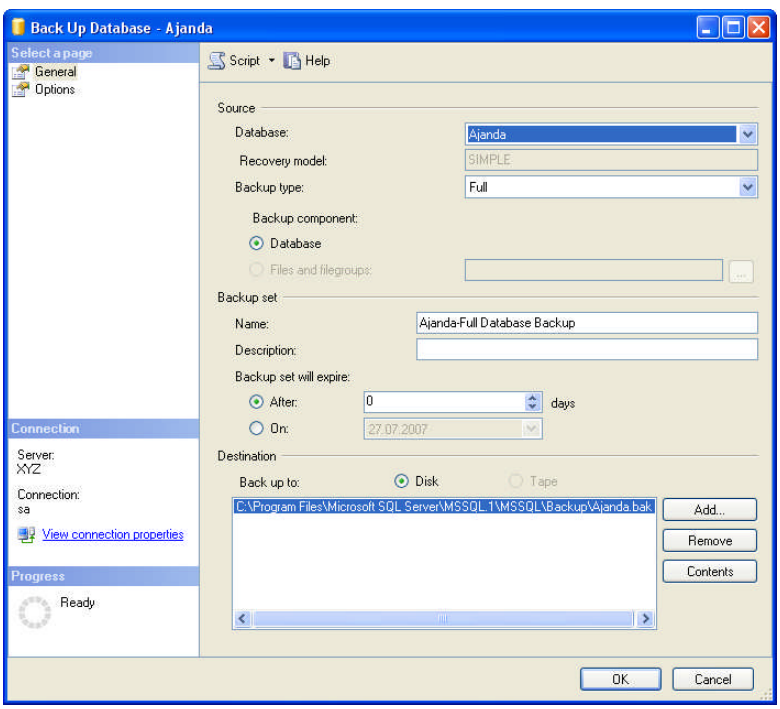
BACKUP DATABASE veritabanı_adı **TO** yedek_dosyası

Bu komutla kullanılan parametreler de vardır. Management Studio ile bir veritabanını yedeklemek için yedeklenecek veritabanı üzerinde sağ tıklayarak açılan menüden Tasks seçimi ve ardından Back Up komutu seçilmelidir.



Resim 2.9: Back Up komutunun seçilmesi

Back Up komutu seçildiğinde ekrana Back Up Database iletişim penceresi gelir.



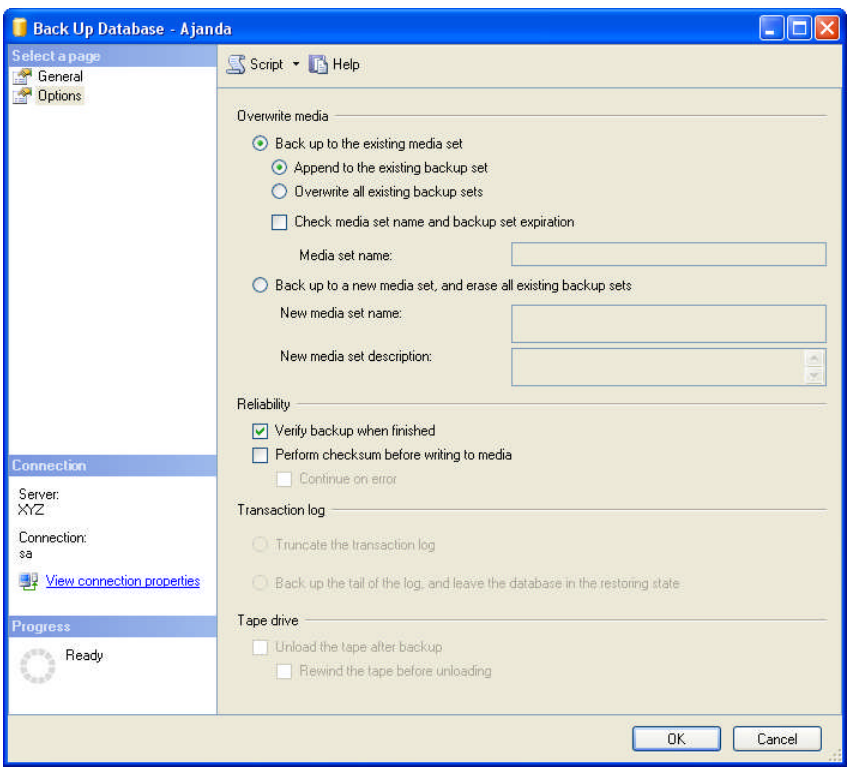
Resim 2.10: Back Up Database iletişim penceresi

Back Up Database penceresinde bulunan Source alanında, Database açılır liste kutusundan yedeklenecek veritabanı ve Backup type açılır liste kutusundan da yedekleme tipi seçilir.

Backup set alanında, alınacak yedeğe bir isim ve açıklama eklenebilir. Alınan yedeğin hangi tarihe kadar geçerli olacağı Backup set will expire ile belirlenir. Belirlenen tarihten sonra yedek çalışmayacaktır.

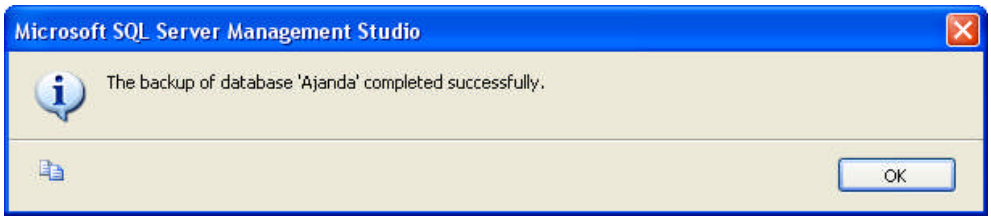
Destination (hedef) alanında, yedeğin nereye alınacağı belirlenir. Alınacak yedek, standart olarak SQL Server'ın Backup dizinine olacaktır. Dosya uzantısı da "bak" tır.

Pencerenin sol tarafında bulunan Options sekmesine tıkladığınızda ise, veriyi var olan bir yedekleme kümesine eklemek istiyorsak "Append to the existing backup set" radio düğmesini seçmelisiniz. Yerden tasarruf etmek isteniyorsa "Overwrite all existing backup sets" seçilir. Böylelikle alınacak veri daha önceden oluşmuşların üzerine yazılacaktır.

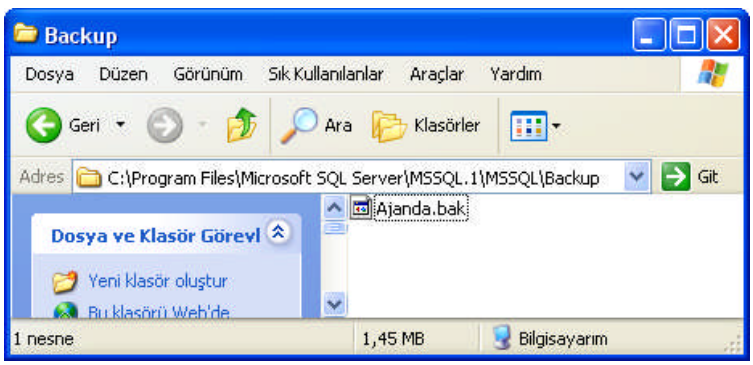


Resim 2.11: Options sekmesi

OK düğmesine tıkladığınızda veritabanının yedeği alınmış olur. Ancak yedeğin, sistemin kurulu olduğu partition'a almak sakıncalar doğurabilir. Bu yüzden yedeği farklı bir sürücüye almakta fayda vardır.



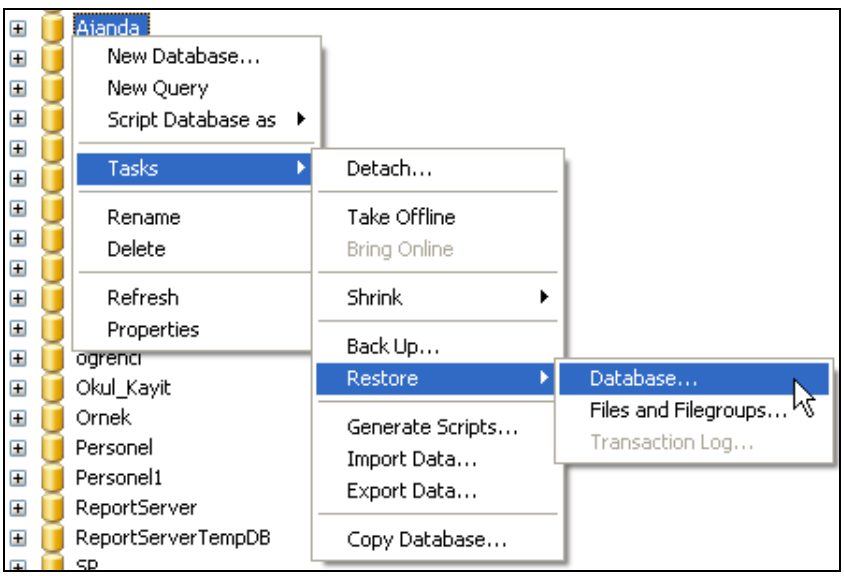
Resim 2.12: Yedeğin başarılı bir şekilde alınması



Resim 2.13: Yedeklemenin yapıldığı dizin

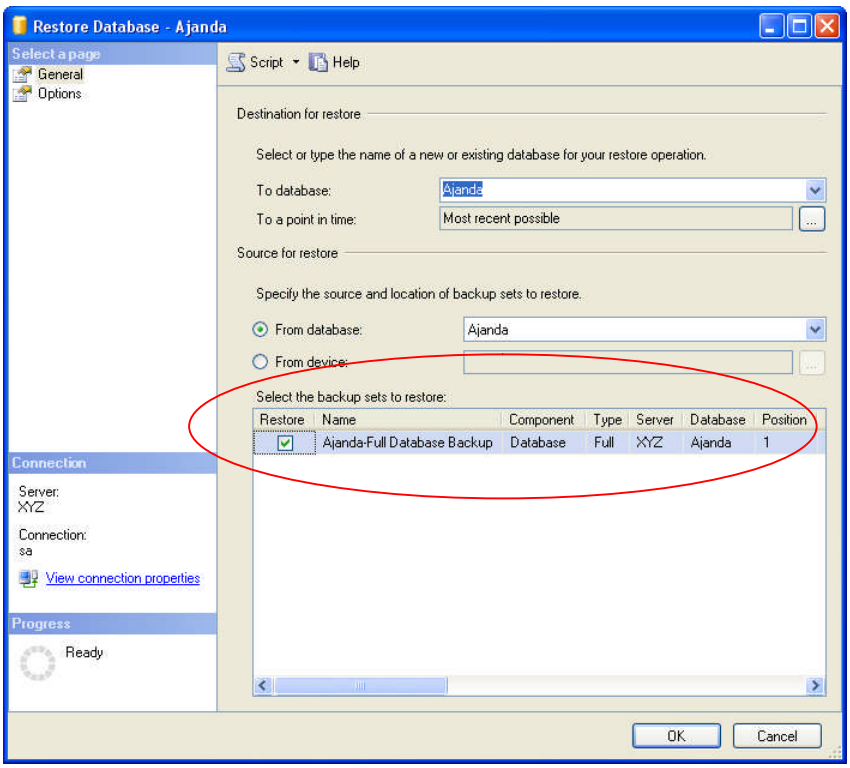
2.3.2. Veritabanı Geri Yükleme

Herhangi bir sorundan dolayı almış olduğunuz yedeği tekrar geri yüklemeniz gerekebilir. Bunun için geri yükleme yapılacak veritabanı üzerinde sağ tıklayarak açılan menüden Tasks seçimi, ardından Restore komutu ve sonra Database seçilmelidir.



Resim 2.14: Restore/Database komutunun seçilmesi

Database komutunun seçilmesinin ardından ekrana Restore Database iletişim penceresi gelir.

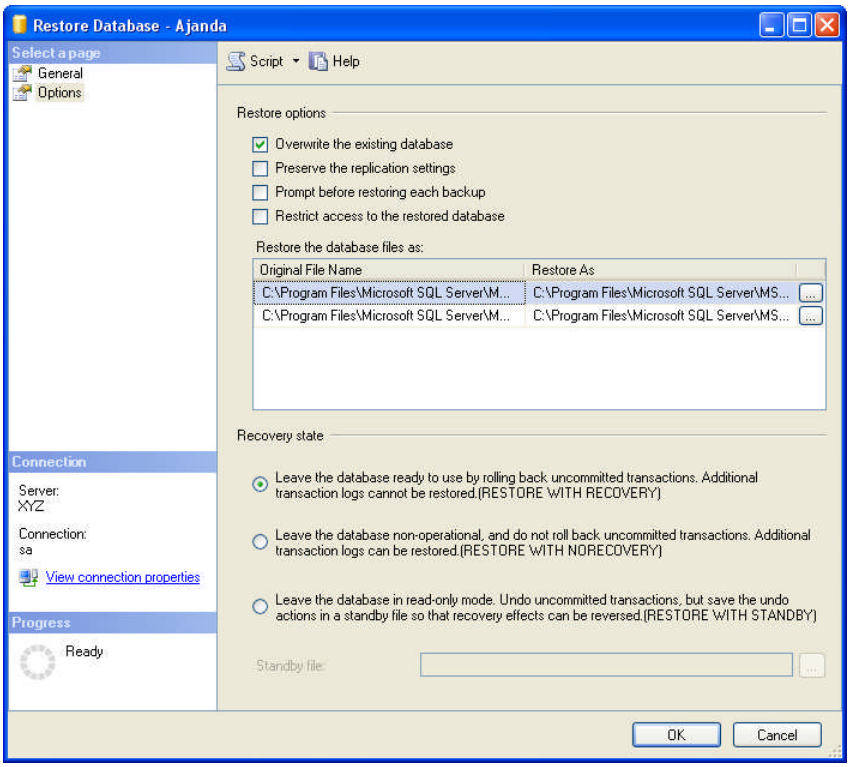


Resim 2.15: Restore Database iletişim penceresi

Daha önce alınmış yedekler "Select the backup sets to restore" kısmında gösterilmektedir. Eğer, farklı bir ortama yedek almış ve o yedeği yüklemek isteniyorsanız "From device" radio düğmesi tıklanır ve yedeğin yeri bulunarak geri yükleme işlemine devam edilir.

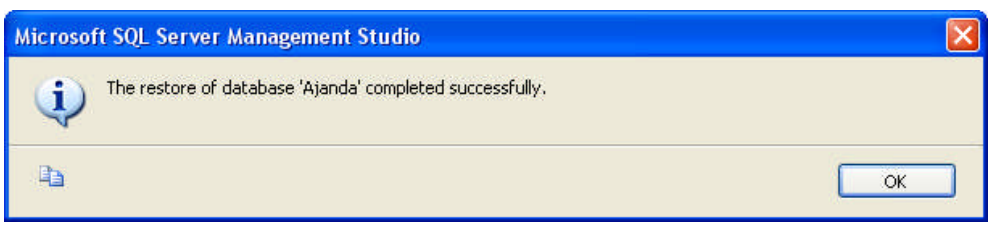
Restore Database iletişim penceresinin Options sekmesinden de istenilen düzenlemeler yapılabilir. Daha önceden alınan yedeğin var olan veritabanı üzerine yazılmasını isterseniz "Overwrite the existing database" seçimini işaretlemelisiniz.

Burada "Restore the database files as" alanındaki "Original File Name" ve "Restore As" alanlarındaki veri ve log dosyasının adları olmalıdır. Eğer yedek alırken farklı bir isim verdiyseniz hata ile karşılaşabilirsiniz.



Resim 2.16: Options sekmesi

Son olarak OK düğmesine tıkladığınızda veritabanının alınmış yedeği geri yüklenecektir.



Resim 2.17: Geri yüklemenin başarılı olduğu uyarısı

2.4. Veritabanında Güvenlik

Bir veritabanını yönetmenin en önemli yönlerinden biri verilerin güvenliğini sağlamaktır. Kimlerin hangi verilere ulaşabileceğinden, bunun yanında kimsenin ulaşmasını istemediğiniz verilere erişim yapılamayacağından emin olmalısınız.

Tavsiye edilen ve alınması gereken temel güvenlik önlemleri, yapısal sorgulama dilini üreten firmanın web sayfasında da açıkça belirtilmiştir.

Alınması gereken temel güvenlik önlemleri şunlardır;

- SQL Server'ın İnternet sitesinde yayınlanan en güncel servis paketlerini bilgisayarınıza kurunuz.
- Ücretsiz olarak kullanıcılara sunulan MBSA (Microsoft Baseline Security Analyzer) kullanarak sisteminizi açıklara karşı denetleyiniz.
- Windows Authentication Mode'u kullanınız.
- Server'ınızı dışarıdan gelecek müdahalelere karşı izole ediniz ve sık sık yedek alınız.
- Güçlü bir system administrator (sa) şifresi kullanınız.
- Sisteminizde Firewall varsa SQL Server'a ait portları dışarıdan erişime karşı kapatınız. SQL Server, TCP 1433 ve UDP 1434 portlarını kullanır.
- Dosya sistemlerinden NTFS dosya sistemini tercih ediniz.
- Server'ınıza erişim yapan uygulamalarınızı ve kullanıcılarınızı denetleyiniz.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
➢ Daha önceden oluşturduğunuz bir veritabanını yedeklemek üzere seçiniz ve ilgili komutu veriniz.	➢ Veritabanı adı üzerinde fareyle sağ tıklayarak açılan menüden Tasks seçimi ve ardından Back Up komutunu seçebilirsiniz.
➢ Back Up Database iletişim kutusundan yedekleme türünün Full olarak seçiniz.	➢ Source alanında Backup type açılır liste kutusundan Full seçimini yapabilirsiniz.
➢ Alınacak yedeğe bir isim veriniz ve bir açıklama ekleyiniz.	➢ Backup set alanından girebilirsiniz.
➢ Alınan yedeğin belirteceğiniz tarihe kadar geçerli olmasını sağlayınız.	➢ Backup set will expire'dan belirtebilirsiniz.
➢ Yedeklemenin diskte nereye yapılacağına dikkat ediniz.	➢ Destination alanına bakabilirsiniz.
➢ İşlemi tamamlayınız.	➢ OK düğmesine tıklayabilirsiniz.
➢ Yedeklemenin yapıldığı klasörü açarak alınan veritabanı yedeğini kontrol ediniz.	➢ Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Backup
➢ Yedeğini aldığınız veritabanının herhangi bir nedenle silindiğini varsayarak geri yüklemek için ilgili komutu veriniz.	➢ Object Explorer penceresinde Databases üzerinde iken fare ile sağ tıklayarak açılan menüden Restore Database komutunu seçebilirsiniz.
➢ Yedeklemenin yapıldığı klasörü hatırlayınız.	➢ Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Backup

➢ Yedekleme için ilgili aygıtı seçiniz ve yedeklenen dosyayı geri yüklemek üzere işaretleyiniz.	➢ Source for restore alanından From device kutucuğunun dağındaki üç noktalı düğmeye tıklayınız. ➢ Specify Backup iletişim kutusunda Add düğmesine tıklayınız. ➢ Yedek dosyasını seçiniz ve OK düğmesine tıklayınız. ➢ Seçtiğiniz dosyanın Backup location listesine eklenmiş olduğundan eminseniz OK düğmesine tıklayınız.
➢ Veritabanının adını seçiniz.	➢ To Database açılır liste kutusundan sildiğiniz veritabanının adını seçiniz.
➢ Yedek dosyayı geri yüklemek için seçiniz.	➢ Select the backup sets to restore penceresinden Restore sütunundaki onay kutusunu işaretleyiniz.
➢ İşlemi tamamlayınız.	➢ OK düğmesine tıklayabilirsiniz.

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. Mevcut durumdan daha küçük parçalara ayrılamayan işleme …….. denir.

2. Transaction bloğu, …….. ile başlar, …….. ya da ……. ile biter..

3. Aşağıdakilerden hangisi transaction modu değildir?
A) Explicit
B) Auto Commit
C) Command
D) Implicit

4. Her veritabanı, birer log dosyası ve veri dosyası içerir (D/Y).

5. Veritabanındaki birincil dosya uzantısı ……’dir.

6. Transaction günlüklerinin dosya uzantısı ……’dir.

7. Log dosyanın büyüme özellikleri Autogrowth sütunundan değiştirilebilir (D/Y).

8. Veritabanını yedeklemek için ……. veya ……. Backup olmak üzere iki çeşit yedekleme şekli vardır (D/Y).

9. Veritabanını yedeklemek için Restore komutu kullanılır (D/Y).

10. Veritabanını geri yüklemek için Restore Database 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 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 5 puan arasında olacak şeklinde değerlendirecektir.

Değerlendirme Ölçütleri	Puan
➢ Veritabanı oluşturabilme	
➢ Tablo oluşturabilme	
➢ İstenen sütunu birincil anahtar olarak belirleyebilme	
➢ Tabloya kayıt girebilme	
➢ Sisteme hata mesajı ekleyebilme	
➢ Saklı prosedür oluşturabilme	
➢ Saklı prosedüre değer vererek çalıştırabilme	
➢ Veritabanını yedeklemek için ilgili komutu verebilme	
➢ Yedekleme türünü seçebilme	
➢ Alınacak yedeğe isim verebilme	
➢ Alınacak yedeğe açıklama ekleyebilme	
➢ Alınacak yedeğin geçerlilik süresini belirleyebilme	
➢ Alınacak yedeğin hedefini bilme	
➢ Yedek alınan klasörün yolunu bilme	
➢ Veritabanını geri yüklemek için ilgili komutu verebilme	
➢ Geri yüklemenin yapılacağı aygıtı seçebilme	
➢ Yedek dosyanın yerini bulabilme	
➢ Yedek dosyayı ilgili pencereye ekleyebilme	
➢ Geri yükleme için veritabanının adını bulabilme	
➢ İlgili pencereden yedek dosyayı seçebilme	
Toplam (100 puan olabilir)	

DEĞERLENDİRME

Yaptığınız değerlendirme sonunda eksikleriniz varsa ilgili öğrenme faaliyetlerini tekrarlayınız.

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

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1'İN CEVAP ANAHTARI

1.	**sys.messages**
2.	**Doğru**
3.	**C Şıkkı**
4.	**sp_dropmessage**
5.	**Doğru**
6.	**@@ERROR**
7.	**Yanlış**
8.	**BEGIN-END**
9.	**D Şıkkı**
10.	**ERROR_LINE()**

ÖĞRENME FAALİYETİ-2'NİN CEVAP ANAHTARI

1.	**Transaction**
2.	**BEGIN TRAN-COMMIT TRAN veya ROLLBACK**
3.	**C Şıkkı**
4.	**Doğru**
5.	***.mdf**
6.	***.ldf**
7.	**Doğru**
8.	**Full veya Differantial**
9.	**Yanlış**
10.	**Doğru**

KAYNAKÇA

- GÖZÜDELİ Yaşar, **"Yazılımcılar İçin SQL Server 2005 ve Veritabanı Programlama"**, Seçkin Yayıncılık, Ankara, 2006.