5

Amaçlarımız

Bu üniteyi tamamladıktan sonra;

- Hareket yönetimi temel kavramlarını açıklayabilecek,
- Eş zamanlılık kavramı ve problemlerini açıklayabilecek,
- İzolasyon düzeylerini ifade edebilecek,
- Kilitleme kavramını ve kilitleme modlarını açıklayabilecek,
- Kilitlenme sorunlarını ve kilitlenmeyi giderebilecek yöntemleri tanımlayabilecek bilgi ve becerilere sahip olabileceksiniz.

Anahtar Kavramlar

- Hareket Yönetimi
- Bölünmezlik, Tutarlılık, İzolasyon, Devamlılık Kuralları
- Eş Zamanlılık Problemleri
- Kilitleme Modları
- Kilitlenme
- İzolasyon Düzeyleri

İçindekiler

Hareket Yönetimi ve Eş Zamanlılık Kontrolü

GİRİŞ

Teknoloji ürünlerinin satışının yapıldığı bir çevrim içi mağazadan bir ürün alınmak istendiğini ve bu üründen sadece bir adet kaldığını düşünelim. Bu ürün alınmak üzere alışveriş sepetine eklenmiş ve ödemesi de yapılmış olsun. Alışverişin tamamlanması esnasında çevrim içi mağaza sunucularında bir arızanın oluşması, elektrik ya da internet bağlantısının kesilmesi gibi herhangi bir nedenle alışveriş işlemi yarıda kesilirse ne olur?

- İhtimallerden birisi, ürün stoktan düşülmüştür fakat müşteriye gönderilmek üzere kargoya verilmemiş olabilir.
- Bu durumda ürün depoda olmasına rağmen stoklarda görünmeyebilir.
- Diğer müşteriler aslında depoda olan ürünün stokta kalmadığı bilgisini görebilirler.
- Satıcı depoda tek kalan ürünün stokta olmadığını görerek yeni ürün tedarik etmek isteyebilir.

Başka bir senaryoda ise aynı anda iki kişinin tek kalan bu ürünü almak istediklerini ve alışveriş sepetlerine eklediklerini varsayalım. Alışverişi tamamlamak için ödemelerini de başarı ile yaptıklarını düşünelim. Bu durumda neler olabilir?

- Ödemeyi başarı ile ilk gerçekleştirmiş olan müşteri ürünü almaya hak kazanırken diğer müşteri ödeme yapmış olmasına rağmen ürünü alamayabilir.
- Satıcı depoda eksiye düşmüş bir ürün stoğu ile karşılaşabilir ve bu ürün tekrar tedarik edilemeyecek bir ürün olabilir.

Bu sebeplerden dolayı, veritabanı olası durumlar dışında veri kaybetmeye açıktır ve bu gibi durumların önlenmesi gerekmektedir.

Bu bölümde yukarıdaki gibi olumsuz senaryolarda bahsedilen durumları engellemek amacıyla veritabanlarında gerçekleştirilebilecek kontrol yapılarından bahsedilecektir. **Hareket** yönetimi ve **eş zamanlılık** kontrolü bu olumsuz senaryoların kontrolünde karşımıza çıkan iki önemli konudur.

Hareket (transaction): Bir veritabanı üzerinde gerçekleştirilen bir dizi SQL işlemini kapsayan işlemler grubudur.

Eş zamanlılık (concurency): Aynı veya farklı istemciler tarafından gerçekleştirilen veritabanı hareketlerinin, bir veri veya veri grubu üzerinde aynı anda işlem gerçekleştirmeye çalışmasıdır.

HAREKET VE ÖZELLİKLERİ

Hareket, temel olarak daha küçük parçalara bölünemeyen en küçük işlem grubudur. Veritabanı üzerinde gerçekleştirilen bir dizi SQL işlemini (`SELECT`, `INSERT`, `UPDATE` ve `DELETE`) kapsayan işlemler grubuna verilen isimdir. Böylelikle birbirinden bağımsız çalışan SQL işlemleri tek bir ifadeymiş gibi yürütülür. Hareket blokları içinde verilerde meydana gelen değişiklikler hemen kalıcı hâle gelmez. Kalıcı hâle gelebilmeleri, hareket bloğu içindeki tüm SQL ifadelerin başarılı bir şekilde gerçekleştirilmesine bağlıdır. SQL işlemleri belirlenen bazı şartlar sağlanmış ise veritabanına uygulanır ve kalıcı hâle getirilir.

Hareket bloğu içinde bulunan tüm işlemler bazı sebeplerden dolayı başarılı bir şekilde tamamlanamadan hata oluşabilir. Bu durumda;

- Veritabanında hatanın meydana geldiği zamana kadar gerçekleşen tüm değişiklikler iptal edilmeli,
- Hareket başlamadan önceki başlangıç durumuna geri dönülmelidir.

Bunun için veri tabanlarında hareket blokları **ACID** olarak kısaltılmış olan dört özelliğe sahip olmalıdır. *Bölünmezlik, tutarlılık, izolasyon* ve *devamlılık.*

ACID: İngilizce Atomicity (bölünmezlik), Consistency (tutarlılık), Isolation (izolasyon) ve Durability (devamlılık) kelimelerinin baş harflerinden oluşur.

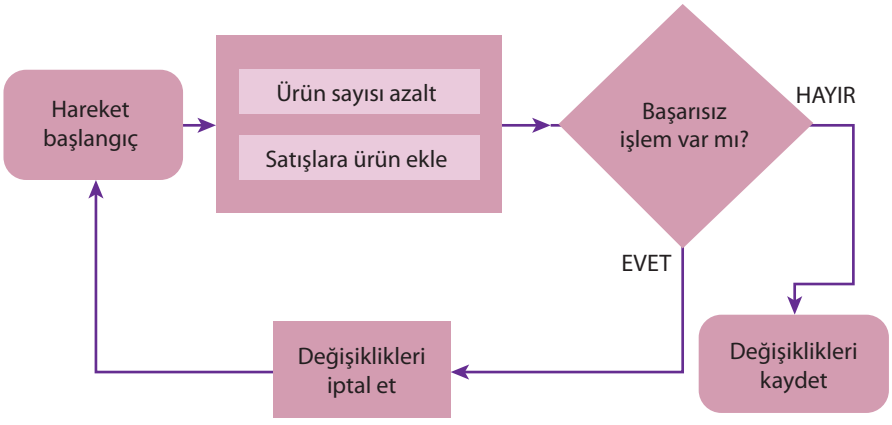
Bölünmezlik

Bölünmezlik: Hareket bloğu içinde yer alan SQL ifadelerin tamamının başarılı olarak çalıştırılması ya da hiçbirinin çalıştırılmadan hareket bloğu başlangıç durumuna geri dönülmesidir.

Farklı iki tablo üzerinde değişiklik yapılması gibi veritabanında kayıtlı veriler üzerinde birden fazla işlem gerçekleştirilmek istensin. Bu amaçla bir hareket bloğu tanımlandı ise blok içerisinde yer alan tüm işlemler başarı ile gerçekleştirilmelidir. Teknoloji mağazası çevrimiçi satış örneğimize geri dönelim. Şekil 5.1'de görüldüğü gibi internet satışı esnasında önce ürün sayısı azaltılır. Sonra satışlar tablosuna satış ile ilgili olarak bir satır ürün kaydı eklenmesi gerekir. Eğer bu iki işlemden herhangi birisi başarı ile tamamlanamaz ise hareketin başladığı duruma geri dönülmesi gerekmektedir. İnternet satışı esnasında ürün stoktan düşüldükten sonra satış yapıldığı sırada mağaza sunucuları ya da müşteri bilgisayarı arızalanmış olsun. Satış tamamlanamadığı için ürün stoğunda bir değişiklik meydana gelmemesi beklenir. Hareket bloğu içinde başarısız bir işlem bulunuyor ise "ya hep ya da hiç" mantığıyla, hareket bloğu yürütülmeden önce bulunulan veritabanı durumuna dönülmelidir.

Şekil 5.1

Teknoloji Mağazası Çevrim İçi Ürün Satışı Hareket Akış Diyagramı.



SIRA SİZDE 1

Bir bankaya ait bilgi yönetim sisteminde iki hesap arası havale işlemi gerçekleştirilmesini bölünmezlik kuralı açısından inceleyiniz.

Tutarlılık

Tutarlılık: Veritabanında gerçekleştirilen işlemler sonucunda oluşan yeni veriler arasındaki tutarlılığı ifade eder.

Bir veri üzerinde işlem gerçekleştirilirken bu veri ile ilişkili olarak tanımlanmış olan kısıtların (tetikleyiciler, birincil anahtar, yabancı anahtar, benzersiz kısıtlama vb.) dışına çıkılmaması gerekir. Kısıtların dışına çıkıldığı durumda veritabanı yönetim sistemi (VTYS) tarafından değişikliklere izin verilmemelidir. Örneğin, bilişim veritabanındaki bölümler tablosundan bir bölüm silmek isteyelim. Ürünler tablosunda silinmesi planlanan bölümde kayıtlı bir ürün bulunuyor ise veri tutarlılığı açısından bu bölümün silinmesine izin verilmemelidir. Çünkü ürünler tablosundaki Bolum_No sütunu yabancı anahtar olarak kullanılmaktadır. VTYS'lerin bu tipteki kısıtları değişiklik gerçekleşmeden önce kontrol etmesi, doğrulaması ve eğer bir sorun ile karşılaşılırsa geri alabilmesi beklenir.

İzolasyon

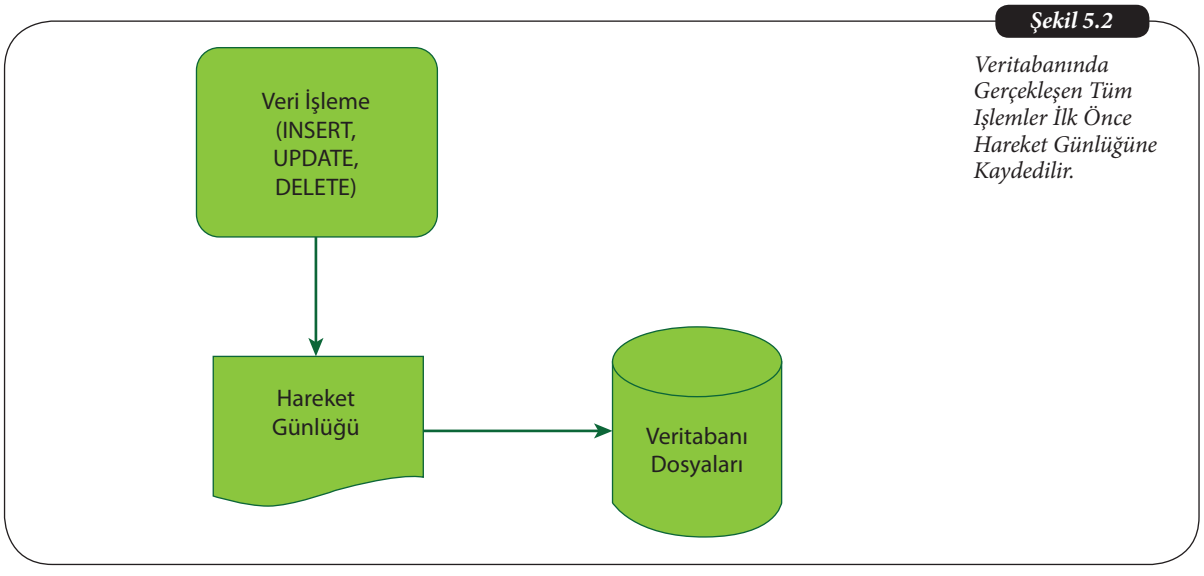
İzolasyon, veri erişim kontrolü amacıyla kullanılan bir mekanizmadır. Farklı hareket blokları birbirinden bağımsız bir şekilde ele alınmalıdır. Hareket bloğunun erişimde bulunacağı veri veya veri grubu, hareket bloğunun beklediği tutarlılık düzeyinde olmalıdır. Her hareket bloğu için veritabanı durumu birbirinden izole edilmelidir. Örneğin bir hareket bloğunun veri üzerinde değişiklik yaptığı sırada başka bir hareket bloğunun bu veriyi okumak istediğini düşünelim. Bu durumda veriyi okumak isteyen hareket bloğu veri tutarlı bir duruma gelinceye kadar engellenir ve çalışmasına izin verilmez. İlk hareket bloğundaki işlemlerin değişiklik başarılı ya da başarısız olsa da tamamlanması (bölünmezlik özelliği) beklenir.

Devamlılık

Tamamlanmış olan ya da devam eden hareketlerin gerçekleştirdiği veri değişikliklerinin VTYS'de bir arıza meydana gelse bile kalıcı olarak kaydedildiğinden emin olunması beklenmektedir. MS SQL Server VTYS'nin en önemli özelliklerinden biri, bu amaçla **hareket günlüğü** dosyası tutmasıdır. Bir hareket yürütülmeye başlandığında hareket içindeki tüm işlemler şekil 5.2'de görüldüğü gibi hareket günlüğüne kaydedilir. Bu sırada veri üzerindeki değişiklikler bellekte gerçekleşmektedir. Hareket tamamlandığında bellekte gerçekleşen veri değişiklikleri fiziksel disk üzerindeki **veri dosyaları** üzerine kaydedilir. Değişiklikler önce hareket günlüklerine yazılır. Sonra veri dosyalarında kalıcı hâle getirilir. Böylelikle bir çökme veya bozulma anında en güncel veri hareket günlüklerinde bulunmuş olur. Veritabanı yöneticileri verileri kurtarma veya bir hareketin gerçekleştirdiği değişiklikleri geri alma amaçlı veri dosyaları ile birlikte hareket günlüklerini de saklamak ve korumak zorundadırlar.

Hareket Günlüğü: Transaction log

Veri Dosyaları: MS SQL Server VTYS'de, veriler üzerindeki kalıcı değişiklikler MDF (master data file), hareket günlükleri LDF (log data file) uzantılı dosyalarda kayıt altına alınmaktadır.



Şekil 5.2

Veritabanında Gerçekleşen Tüm İşlemler İlk Önce Hareket Günlüğüne Kaydedilir.

Hareket Yönetimi

MS SQL Server VTYS, otokayıt (autocommit), açık (explicit) ve örtük (implicit) olmak üzere üç farklı modda hareket desteği sağlamaktadır.

Otokayıt Hareket

MS SQL Server VTYS'nin varsayılan hareket yönetim modudur. Hareket içindeki her bir SQL komutu tamamlandıktan sonra otomatik olarak veritabanına kaydedilir. Örneğin, ilki başarısız olduğu hâlde ikincisi başarılı olan iki adet insert komutunun olduğunu düşünelim. İkinci komutta yapılan değişiklik kalıcı hâle gelecektir. Çünkü her bir insert

Otokayıt: Autocommit

komutu kendi hareketi içinde tamamlandığında otomatik kayıt gerçekleşecektir. Örnek veritabanımız bilisim veritabanındaki çalışanlar (calisanlar) tablosuna iki kişiyi eklemek için kullanacağımız aşağıdaki iki insert satırını ele alalım;

```
INSERT INTO Calisanlar(TC_No, Adi, Bolum_No, Cinsiyet)
                VALUES (21365412126,'Halit Ziya',2,32);
INSERT INTO Calisanlar(TC_No, Adi, Bolum_No, Cinsiyet)
                VALUES (12634521624,'Mehmet Akif',2,'E');
```

Yukarıdaki SQL kod parçası çalıştırıldığında MS SQL Server VTYS bize aşağıdaki hata mesajını verecektir.

```
Msg 8115, Level 16, State 2, Line 1
Arithmetic overflow error converting expression to data type nvarchar.
The statement has been terminated.

(1 row(s) affected)
```

Tablo 5.1 İkinci SQL İfadesindeki Kişi Bilgileri Başarili Bir Şekilde Veritabanina Kaydedilir.

TC No	Adı	Bölüm No	Cinsiyet
12634521624	**Mehmet Akif**	**2**	**E**
17895632417	Ziya Doğan	3	E
48975626633	Ayşe Saygı	4	K
65638749631	Ali Yılmaz	1	E
78965412305	Fatma Doğan	5	K
89652341780	Hasan Çiçek	2	E
89745632107	Kader Kara	6	K

İlk insert ifadesinde cinsiyet verisi olarak "E" veya "K" karakteri beklenirken 32 sayısı girilmiştir. Bu yüzden ifade başarısız olarak sonlanır ve veritabanında herhangi bir değişikliğe sebep olmaz. İkinci insert ifadesindeki tüm veriler doğru olarak girildiğinden bu ifade başarı ile sonlanır. Tablo 5.1'de görüldüğü gibi ikinci satırdaki çalışan bilgileri veritabanında çalışanlar tablosuna kaydedilmiştir. Otokayıt hareket yönetimi bazı durumlarda tehlikeli olabilmektedir. Örneğin, bir tablodaki tüm satırlar kazara silinecek olursa hata yapıldığı anlaşıldıktan sonra işlemi geri alabilme şansı olmayacaktır.

Örtük Hareket

Örtük Hareket: Implicit transaction

MS SQL Server VTYS tarafından tablo 5.2'de listelenmiş olan ifadelerden herhangi biri ilk kez yürütüldüğünde yeni bir hareket otomatik olarak başlatılmaktadır. Bu hareket bloğu **Rollback** veya **Commit** ifadelerinden herhangi biri yürütülene kadar açık kalmaktadır.

Tablo 5.2 İlk Kez Yürütüldüğünde SQL Veritabanı Sunucusunun Yeni Bir Hareket Başlattığı SQL İfadeleri.

Alter Table	Grant	Truncate Table
Fetch	Select	Drop
Revoke	Delete	Open
Create	Insert	Update

Rollback, commit (geri al, kalıcı yap): Bir hareket bloğunun veritabanı üzerinde gerçekleştirdiği değişikliklerin kalıcı hâle getirilmesi için commit; değişikliklerin geri alınması için ise rollback ifadesi kullanılır.

Örtük hareket modunu aktif hâle getirmek ve modu kapatmak amacıyla tablodaki komutların kullanılması gerekmektedir.

Komut	Açıklama
`SET IMPLICIT_TRANSACTIONS ON`	Örtük hareket modunu aktifler.
`SET IMPLICIT_TRANSACTIONS OFF`	Örtük hareket modunu kapatır.

Tablo 5.3 Örtük Hareket Modunu Aktiflemek ve Kapatmak İçin Kullanılan İfadeler.

Örtük hareket modunun kullanımı eğer **Rollback** ya da **Commit** işleminin yapılması unutulur ise problem yaratabilmektedir. Çünkü örtük hareket modunda üzerinde işlem yapılan tablo ya da tablolara erişim açık kalır. Bu yüzden başka bağlantıların yapılması engellenir.

```
SET IMPLICIT_TRANSACTIONS ON;
INSERT INTO Calisanlar(TC_No, Adi, Bolum_No, Cinsiyet)
                VALUES (12634521624,'Mehmet Akif',2,'E');
```

Yukarıdaki örnek kod bloğunda **SET IMPLICIT_TRANSACTIONS ON;** ifadesi ile örtük hareket modu aktif hâle getirilmektedir. Bir sonraki ifadede çalışanlar tablosuna yeni bir kayıt eklenmek istenmektedir. Yeni kaydın tabloya eklenmesinde bir problem görünmemektedir. Bununla beraber eğer aynı anda çalışanlar tablosundaki kayıtları listelemek isteyen bir SQL ifadesi çalıştırılmak istenirse listeleme gerçekleştirilemeyecektir. Çünkü yukarıdaki ifade yeni bir hareket başlatacaktır. Hareket sonlandırılmadığı için çalışanlar tablosuna yeni bir bağlantı açılmasına izin verilmeyecektir. Bunu önlemek için yukarıdaki ifadenin sonuna rollback ya da commit ifadelerinden birisinin eklenmesi gerekmektedir.

```
SET IMPLICIT_TRANSACTIONS ON;
INSERT INTO Calisanlar(TC_No, Adi, Bolum_No, Cinsiyet)
                VALUES (12634521624,'Mehmet Akif',2,'E');
COMMIT;
SET IMPLICIT_TRANSACTIONS OFF;
```

Yukarıdaki örnek kod bloğunda **SET IMPLICIT_TRANSACTIONS ON;** ifadesi ile örtük hareket modu aktif hâle getirilmektedir. Sonraki ifadede çalışanlar tablosuna yeni bir kayıt eklenmektedir. **COMMIT;** ile gerçekleştirilen ekleme işlemi veritabanı tablolarında kalıcı hâle getirilmektedir. **SET IMPLICIT_TRANSACTIONS OFF;** ifadesi örtük hareket modunu kapatmak için kullanılmaktadır.

Açık Hareket

Açık Hareket: Explicit transaction

Açık hareketler kullanıcıların kendilerinin başlatıp sonlandırdığı hareket bloklarıdır. Veritabanı uygulamalarında veri değişiklikleri yapılırken tavsiye edilen hareket modudur. Bir hareket bloğunun başarılı ya da başarısız olması sonucunda hangi işlemlerin yapılacağı uygulama geliştirici tarafından açık bir şekilde kontrol edilebilir. Bu yüzden daha çok tercih edilen bir hareket yönetim modudur. Açık hareket komutları tablo 5.4'te listelenmektedir.

Bu ünitede yer verilen SQL komut dosyalarına https://goo.gl/NqahM1 adresinden ulaşabilirsiniz. İlgili örnek veritabanın 1. ünitede oluşturulduğunu hatırlayalım.

Tablo 5.4
Açık Hareket Modunda Kullanılan Komutlar.

Komut	Açıklama
BEGIN TRANSACTION	Açık hareket başlangıç noktasını belirler.
ROLLBACK TRANSACTION	Hareket bloğu tarafından yapılmış olan değişiklikleri geri alır.
COMMIT TRANSACTION	Bir hata ile karşılaşılmaz ise değişikliklerin kalıcı hâle getirilmesi amacıyla kullanılır.
SAVE TRANSACTION	Hareket bloğu içinde geri dönüş noktaları oluşturmak için kullanılır.
@@TRANCOUNT	Aktif hareket sayısını geri döndürür. BEGIN TRANSACTION bu değeri bir artırırken; ROLLBACK TRANSACTION ve COMMIT TRANSACTION bir azaltır.

Bilişim veritabanında bulunan ürünler tablosundaki 12 ürün numaralı buzdolabından bir adet satıldığını varsayalım. Bu durumda satışlar tablosuna bu ürün ile ilgili satış bilgileri kaydedilmelidir. Aynı zamanda ürünler tablosundan ve buzdolabı stoğundan bir adet buzdolabı düşülmesi gerekmektedir. Bu satış işlemini gerçekleştirmek amacıyla yürütülecek açık hareket bloğunda herhangi bir hata meydana gelmezse satış gerçekleşir. İki tabloda meydana gelen değişiklikler kalıcı hâle getirilir. Eğer hareket bloğu içinde bir hata meydana gelirse değişiklikler geri alınır. Aşağıdaki SQL kodu bu işlemi gerçekleştirmektedir.

```
BEGIN TRANSACTION
DECLARE @urunFiyat decimal(18,2);
SELECT @urunFiyat=Urun_Fiyati FROM Urunler where Urun_No=12;

INSERT
INTO  Satislar(Satis_No, Urun_No, Calisan_TC_No, Miktar, Fiyat, Tarih)
VALUES (12, 12, '78965412305', 1, @urunFiyat, CONVERT(date, GETDATE()));

if (@@ERROR<>0)
    goto HATA_VAR;
UPDATE Urunler
SET Urun_Sayisi=Urun_Sayisi-1 where Urun_No=12;
if (@@ERROR<>0)
    goto HATA_VAR;
COMMIT TRANSACTION

HATA_VAR:
if (@@ERROR<>0)
BEGIN
ROLLBACK TRANSACTION;
END
```

Şimdi de yukarıda verilen hareket bloğunda bulunan ifadelerin nasıl çalıştığını inceleyelim.

```
BEGIN TRANSACTION
```

Açık hareket bloğu başlatılmaktadır.

```
DECLARE @urunFiyat decimal(18,2);
SELECT @urunFiyat=Urun_Fiyati FROM Urunler where Urun_No=12;
```

Ürünler tablosundan 12 ürün numaralı buzdolabına ait fiyat @urunFiyat değişkeni içinde saklanmaktadır.

```
INSERT
INTO  Satislar(Satis_No, Urun_No, Calisan_TC_No, Miktar, Fiyat, Tarih)
VALUES (12,12,'78965412305', 1, @urunFiyat, CONVERT(date, GETDATE()));
```

Satışı yapılacak olan buzdolabına ait bilgilere **CONVERT(date, SYSDATETIME())** fonksiyonu ile sistem tarihi eklenerek satışlar (Satislar) tablosuna kaydedilmektedir.

```
if (@@ERROR<>0)
    goto HATA_VAR;
```

Satışlar tablosuna kayıt eklenirken hata meydana gelip gelmediği kontrolü **@@ERROR** fonksiyonu ile yapılmaktadır. Eğer hata varsa HATA_VAR satırından itibaren hareket bloğu yürütülmeye devam eder. Eğer hata oluşmamış ise bir sonraki ifade yürütülecektir.

```
UPDATE Urunler
SET Urun_Sayisi=Urun_Sayisi-1 where Urun_No=12;
```

Satışlar tablosuna kayıt işlemi başarı ile gerçekleşmiş ise ürünler tablosunda buzdolabı stoğundan bir adet ürün düşülecektir.

```
if (@@ERROR<>0)
    goto HATA_VAR;
COMMIT TRANSACTION
```

Ürünler tablosundaki buzdolabı stok sayısı güncellenirken bir hata meydana gelmiş ise yine HATA_VAR satırına gidilecektir. Aksi durumda bilisim veritabanındaki satışlar ve ürünler tablolarında meydana gelen değişiklikler kalıcı duruma getirilecektir.

```
HATA_VAR:
if (@@ERROR<>0)
BEGIN
ROLLBACK TRANSACTION;
END
```

Satışlar tablosuna yeni kayıt eklenirken ya da ürünler tablosundaki buzdolabı stoğu güncellenirken hata oluştuğunda yapılan tüm değişiklikler rollback transaction ifadesi yardımıyla geri alınır. Böylece veritabanı, hareket başlatılmadan önceki konumuna geri döndürülmektedir.

EŞ ZAMANLILIK VE EŞ ZAMANLILIK PROBLEMLERİ

Günümüzde birçok kurumsal uygulama, çok sayıda kullanıcı tarafından kullanılmaktadır. Kullanıcıların sahip olduğu cihazların çeşitliliği ise oldukça fazladır. Bu sebeplerden dolayı kurumsal uygulama diğer uygulamalara, hizmetlere veya raporlama ihtiyaçlarına aynı anda cevap verme yeteneğine sahip olmalıdır. Bu uygulamaların verileri saklamak için kullandıkları VTYS, birbirinden farklı hizmetlerin veri okuma ve veri yazma ihtiyaçlarını aynı anda karşılayabilmelidir. VTYS'lerde *eş zamanlılık kavramı,* birden fazla **oturum** üzerinden aynı kaynaklara erişimi ve kullanımı tanımlamaktadır.

Oturum: Bir veritabanına farklı kullanıcılar tarafından gerçekleştirilen erişimdir.

VTYS'lerde aynı anda birden fazla oturum üzerinden aynı kaynaklara erişilmeye çalışılması eş zamanlılık problemlerini ortaya çıkarmaktadır. Bu ünitenin ilerleyen sayfalarında açıklanacak olan farklı izolasyon düzeyleri belirli eş zamanlılık problemlerine karşı koruma sağlamaktadır. Bu problemler genel olarak *kayıp güncelleme, kirli okuma, tekrarlanamayan okuma* ve *hayalet okuma* olmak üzere dört ana grupta toplanmaktadırlar.

Kayıp Güncelleme

Kayıp Güncelleme: Lost update,

Tablo 5.5'te görüldüğü gibi ürünler tablosundaki 12 numaralı buzdolabı fiyatını güncellemek için bir kullanıcı tarafından hareket 1 başlatılsın. Hareket 1'de commit ya da rollback yapılmadan eş zamanlı olarak başka bir kullanıcı tarafından hareket 2 başlatılmış olsun. Hareket 2 fiyat güncellemesini gerçekleştirsin ve commit ifadesiyle de değişikliği kalıcı hâle getirsin. Hareket 2 tamamlandıktan sonra hareket 1'deki commit işlemi gerçekleşmektedir. Bu durumda hareket 2 sonunda gerçekleştirilen güncelleme işlemi kaybolacaktır. Bu probleme kayıp güncelleme adı verilmektedir.

Tablo 5.5
Kayıp Güncelleme Eş Zamanlılık Probleminin Oluşumu.

Hareket 1	Hareket 2
DECLARE @urunFiyat decimal(18,2); BEGIN TRANSACTION SET @urunFiyat=2500;	
	BEGIN TRANSACTION UPDATE Urunler SET Urun_Fiyati=2300 where Urun_No=12; COMMIT TRANSACTION
UPDATE Urunler SET Urun_Fiyati=2500 where Urun_No=12; COMMIT TRANSACTION	

Kirli Okuma

Kirli okuma: dirty read

Bir hareket tarafından değiştirilmiş fakat kalıcı olarak veritabanına kaydedilmemiş bir verinin başka bir hareket tarafından geçerli bir veri gibi okunması kirli okuma olarak ifade edilmektedir. Tablo 5.6'da hareket 1, 12 numaralı ürünün fiyatını güncellediği sırada eş zamanlı olarak hareket 2 başlatılıyor. Hareket 2 ürünler tablosundan 12 numaralı ürünün fiyatını bir değişkene alıyor. Hareket 2 sonrasında satışlar tablosunda bu kez 12 numaralı ürünün fiyatını güncelliyor ve commit ifadesiyle kalıcı hâle getiriyor. Hareket 2 başlayıp bitiyor. Bu esnada hareket 1'de bir hata oluşursa rollback ifadesi ile yapılmış olan fiyat güncelleme işlemi geri alınır. Bu durumda hareket 2, kirli bir okuma yapmış ve hatalı bir güncelleme ile veri tutarsızlığı yaratmış olacaktır.

Tablo 5.6
Kirli Okuma Eş Zamanlılık Probleminin Oluşumu

Hareket 1	Hareket 2
BEGIN TRANSACTION UPDATE Urunler SET Urun_Fiyati=2500 where Urun_No=12;	
	BEGIN TRANSACTION SELECT @urunFiyat=Urun_Fiyati FROM Urunler where Urun_No=12; UPDATE Satislar SET Urun_Fiyati=@urunFiyat where Urun_No=12; COMMIT TRANSACTION

IF (@@ERROR<>0)
ROLLBACK TRANSACTION;
ELSE
COMMIT TRANSACTION;	

Tekrarlanamayan Okuma

Tekrarlanamayan Okuma: non-repeatable read

Bir veri ya da veri grubuna bir hareket içinde birden fazla kez erişim gerçekleştirilebilir. Bu erişimler arasında eş zamanlı olarak başka bir hareket güncelleme veya silme işlemi gerçekleştirebilmektedir. Bu durumda tekrarlanamayan okuma problemi ile karşı karşıya kalınmış olunmaktadır.

Hareket 1	Hareket 2
BEGIN TRANSACTION SELECT SUM(Urun_Fiyati*Urun_Sayisi) FROM Satislar WHERE Urun_No=12;	
	BEGIN TRANSACTION DELETE FROM Satislar where Urun_No=12 COMMIT TRANSACTION
SELECT SUM(Urun_Fiyati*Urun_Sayisi) FROM Satislar WHERE Urun_No=12; COMMIT TRANSACTION;	

Tablo 5.7
Tekrarlanamayan Okuma Eş Zamanlılık Probleminin Oluşumu

Tablo 5.7'de görüldüğü gibi hareket 1 başlatıldıktan sonra satışlar tablosundan 12 numaralı üründen toplam ne kadarlık bir satış yapıldığı bilgisi elde edilmektedir. Eş zamanlı olarak hareket 2 satışlar tablosundan 12 numaralı ürüne ait tüm kayıtları silmekte ve değişiklikleri kalıcı hâle getirmektedir. Bu durumda hareket 1 içinde yapılacak olan ikinci okuma hatalı bilgi elde edilmesine sebep olmaktadır.

Hayalet Okuma

Hayalet Okuma: Phantom read

Aynı hareket içinde ve aynı tablo üzerinde aynı WHERE ifadesi ile iki farklı sorgulama yapıldığını varsayalım. Bu iki sorgulama arasında eş zamanlı olarak başka bir hareket, okunan aralık içindeki veri kümesine yeni bir kayıt eklemiş olsun. Bu durumda ikinci sorgulamada birinciye göre daha fazla kayıt okunmuş olur. Fazla kayıt okuma işlemine hayalet okuma adı verilmektedir. Tablo 5.8'de görüldüğü gibi hareket 1 bloğu içinde iki kez çalışanlar sorgulanmaktadır. Hareket 1'deki ilk çalışan sorgulamasından sonra eş zamanlı olarak hareket 2 içinde çalışanlar tablosuna yeni bir çalışan kaydedilmektedir. Bu durumda hareket 1 içinde yer alan ikinci sorguda fazladan bir adet hayalet okuma gerçekleşecektir.

Hareket 1	Hareket 2
BEGIN TRANSACTION SELECT * FROM Calisanlar WHERE Cinsiyet='E'	
	BEGIN TRANSACTION INSERT INTO Calisanlar (TC_No, Adi, Bolum_No, Cinsiyet) VALUES (21365412126,'İbrahim Önder',2,'E') COMMIT TRANSACTION
SELECT * FROM Calisanlar WHERE Cinsiyet='E' COMMIT TRANSACTION;	

Tablo 5.8
Hayalet Okuma Eş Zamanlılık Probleminin Oluşumu

İZOLASYON SEVİYELERİ

MS SQL Server VTYS eş zamanlı hareketlerin rekabet etkinliğini kontrol etmek için kilit mekanizması kullanmaktadır. Hareketler daha önce de belirtildiği gibi eş zamanlı olarak

çalışabilmektedir. Meydana gelebilecek eş zamanlılık problemlerinin önüne geçmek için veritabanı kaynaklarına erişimi kontrol etmek ve hareketlerin birbirinden izole bir şekilde çalışabilmesi amacıyla kilitleme gerçekleştirilmektedir. MS SQL Server VTYS farklı izolasyon düzeylerini desteklemektedir. İzolasyon düzeyleri, eş zamanlı olarak çalışan hareketlerin birbirlerini nasıl etkilemesi gerektiğinin belirtilmesinde kullanılmaktadır. Bir hareket içinde gerçekleştirilen işlemler sonucunda kayıtlar üzerinde meydana gelen değişikliklerin, eş zamanlı olarak aynı kayıtlar üzerinde işlem gerçekleştiren hareketler tarafından nasıl ele alınması gerektiğinin belirlenmesine olanak sağlamaktadır.

Kaydedilmiş Okuma: read committed,

Kaydedilmiş okuma, varsayılan MS SQL Server VTYS izolasyon düzeyidir. Kaydedilmemiş veri değişiklikleri okunamaz. Sorgu esnasında paylaşılan kilitler kullanılır ve bir hareket okuma gerçekleştirirken eş zamanlı olarak başka bir hareket aynı veri üzerinde değişiklik gerçekleştiremez. Aynı tablo üzerinde veri ekleme ve değişikliğine izin verilir. Bu izolasyon düzeyi kirli okuma problemine engel olur. Fakat, tekrarlanamayan okuma ve hayalet okuma eş zamanlılık problemlerine sebep olabilmektedir.

Kaydedilmemiş Okuma: read uncommitted,

Kaydedilmemiş okuma, en düşük kısıtlamaya sahip izolasyon düzeyidir. Hareket tarafından seçilmiş olan veri üzerine herhangi bir kilit uygulanmaz. Yüksek eş zamanlılık sağlamakla birlikte veri tutarlılığı sağlama düzeyi düşüktür. Çünkü, veri okunduğu sırada diğer hareketler tarafından değiştirilebilmektedir. Dolayısıyla kirli okuma, hayalet okuma ve tekrarlanamayan okuma gibi eş zamanlılık problemlerine sebep olabilmektedir.

Tekrarlanabilir Okuma: repeatable read

Tekrarlanabilir okuma, aktif edildiğinde kirli okuma ve tekrarlanamayan okuma eş zamanlılık problemleri ortadan kaldırılabilmektedir. Okunan tüm kaynaklara paylaşılan (S) kilit uygulanır. Sorgunun geri döndürdüğü veri aralığına diğer hareketler tarafından eklenen yeni kayıt satırları gelebilir. Bu yüzden de hayalet okuma eş zamanlılık problemi meydana gelebilmektedir.

Serileştirilebilir: serializable,

Serileştirilebilir izolasyon düzeyi en kısıtlayıcı izolasyon düzeyidir. Bir hareket bir kayıt üzerinde okuma işlemi gerçekleştirdiği sırada başka bir hareketin okunan kayıtlar üzerinde değişiklik yapmasına izin verilmez. Aynı şekilde ilgili veri aralığına yeni bir kayıt eklemesine de izin verilmemektedir. Eş zamanlılık problemlerinin tamamı çözülmekle birlikte kilitlenme ve diğer hareketlerin bloke edilmesi problemleri ile en çok bu izolasyon düzeyinde karşılaşılmaktadır.

Anlık görüntü: snapshot

Anlık görüntü izolasyon düzeyi hareketin başladığı sırada okunan verinin hareket içinde tutarlı bir değerinin kullanılmasına imkan vermektedir. Veri okuma işlemi esnasında diğer hareketlerin değişiklikleri bloke edilmez. Bununla birlikte, anlık görüntü izolasyon düzeyindeki bir hareket, yapılan değişiklikleri tespit edemez. Bu izolasyon düzeyi hareket başında bir kaydın hareket sonlanana kadar hep aynı değerde okunmasını garanti etmemektedir. Kirli okuma eş zamanlılık problemine engel olur. Bununla beraber tekrarlanamayan okuma eş zamanlılık problemi oluşmasına neden olabilmektedir.

SIRA SİZDE 2

İzolasyon düzeyleri ile eş zamanlılık problemlerinden kirli okuma, tekrarlanamayan okuma ve hayalet okuma arasındaki ilişkileri gösteren bir karşılaştırma tablosu hazırlayınız.

Hareket izolasyon düzeylerini belirlemek amacıyla kullanılan SQL ifadeleri tablo 5.9'da listelenmektedir.

Tablo 5.9
İzolasyon Düzeylerini Aktifleme

Komut	Açıklama
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Kaydedilmiş okuma aktif
`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	Kaydedilmemiş okuma aktif
`SET TRANSACTION ISOLATION LEVEL READ REPEATABLE READ;`	Tekrarlanabilir okuma aktif
`SET TRANSACTION ISOLATION LEVEL READ SERIALIZABLE;`	Serileştirilebilir aktif
`SET TRANSACTION ISOLATION LEVEL READ SNAPSHOT;`	Anlık görüntü aktif

Bu ünitede yer verilen SQL komut dosyalarına https://goo.gl/NqahM1 adresinden ulaşabilirsiniz.

KİLİTLEME YÖNETİMİ

Kilitleme çok kullanıcılı veritabanı yönetim sistemlerinin vazgeçilemez parçasıdır. Kilitleme yönetimi sayesinde VTYS'ler, farklı hareketlerin aynı veriye eş zamanlı olarak güncelleme yapmalarına izin vermemektedir. Ayrıca, bir hareketin gerçekleştirdiği güncelleme işleminin ortasında başka bir hareketin bu veriyi görüntülemesine de izin verilmemektedir. Bu sayede hareketlerin gerçekleştirmek istediği işlemler arasında izolasyon sağlanmaktadır. MS SQL Server VTYS, kilitleme yönetimini dinamik olarak gerçekleştirmektedir.

Kilitleme: Veritabanı kaynakları ya da kayıtlara başka oturum ya da hareketler tarafından yapılacak eş zamanlı erişimin engellenmesi işlemidir.

Kilitler sayesinde eş zamanlılık problemleri meydana gelmeden önlem alınmış olunmaktadır. Eş zamanlılık problemleri bir kullanıcı veri okurken başka bir kullanıcının aynı veriyi güncellemeye çalışması sırasında meydana gelebilmektedir. Benzer şekilde bir kullanıcı veriyi güncellemeye çalışırken başka bir kullanıcı okumaya çalışırsa yine eş zamanlılık problemleri meydana gelebilmektedir.

Kilitleme işlemi MS SQL Server VTYS kaynakları üzerinden gerçekleştirilir. Farklı **kilit modları,** farklı veritabanı kaynaklarını kilitlemek amacıyla kullanılmaktadır. Temel olarak iki ana kilit modu vardır: Paylaşılan kilit modu ve ayrıcalıklı kilit modu.

Kilit Modu: Eş zamanlı hareketlerin veritabanı sunucusu kaynaklarına ne şekilde erişilebileceğinin belirlenmesidir.

Paylaşılan Kilit Modu

Paylaşılan kilit, değişikliğin gerçekleşmediği salt okunur sorgular esnasında otomatik olarak oluşmaktadır. Bu kilit modunda okuma yapılan sorgu ve hareketlerin verilere erişimine izin verilir. Bununla birlikte kilit kaldırılana kadar eş zamanlı olarak farklı bir hareketin güncelleme yapmasına izin verilmez. Paylaşılan kilitler, tablo, sayfa, indeks anahtarı veya tek başına bir kayıt satırına uygulanabilir. Birden fazla hareket, aynı veri üzerine paylaşılan kilit koyabilir. Eğer veri üzerinde bir paylaşılan kilit bulunuyor ise hiçbir hareket bu veri üzerine özel kilit koyamaz. Varsayılan **izolasyon düzeyinde** paylaşılan kilitler, veri okunur okunmaz serbest bırakılır. Bunu değiştirmek için farklı izolasyon düzeyleri kullanılmaktadır.

İzolasyon Düzeyi: Eş zamanlı olarak çalışan hareketlerin birbirlerini nasıl etkilemeleri gerektiğini belirtmek amacıyla kullanılır.

Ayrıcalıklı Kilit Modu

MS SQL Server VTYS, insert, update veya delete işlemleri tarafından veri değişikliği yapılırken ayrıcalıklı kilit modunu otomatik olarak oluşturmaktadır. Aynı anda eş zamanlı olarak sadece bir hareket, veri kaynağı üzerinde ayrıcalıklı kilit bulundurabilir. Bir veri kaynağı ayrıcalıklı kilit modunda ise eş zamanlı hareketler bu veri kaynağı üzerine başka hiçbir kilit modu uygulayamaz. Bunun anlamı, veri kaynağı üzerinde değişiklik gerçekleştiren hareket, `Commit` veya `Rollback` işlemini gerçekleştirmeden diğer hareketler bu veri kaynağına erişemeyecektir. Diğer işlemler ayrıcalıklı kilitlenmiş veri kaynaklarını nasıl okuyabileceklerine izolasyon düzeylerini değiştirerek karar verebilirler.

Güncelleştirme Kilit Modu

Güncelleştirme kilit modu aslında ayrı bir kilit modu değil, paylaşılan ve ayrıcalıklı kilit modlarının birleşimidir. MS SQL Server VTYS veri değişikliği gerçekleştireceği zaman öncelikle değişikliğin gerçekleşeceği veriyi bulmak zorundadır. Verinin bulunması sırasında eş zamanlı olarak diğer hareketler okuma işlemi gerçekleştirebilirler. Bir hareket, veri kaynağı üzerinde güncelleştirme kilidi oluşturur oluşturmaz başka bir hareket aynı veri kaynağı üzerine güncellleştirme ya da ayrıcalıklı kilit koyamaz. Bu durumda diğer hareketler beklemeye alınır. Değişiklik yapılacak veri kaynağı bulunduğunda kaynak üzerindeki güncelleştirme kilit modu ayrıcalıklı kilit moduna yükseltilir. Bunun sebebi güncelleştirme kilidinin tek başına veri kaynağı değişikliği için yeterli olmamasıdır. Veri değişiklikleri için veri kaynağı üzerinde mutlaka ayrıcalıklı kilit modu bulunmalıdır. Güncelleştirme kilit modu sadece güncelleme (update) işlemlerinde değil herhangi bir veri değişikliğinin gerçekleşeceği tüm işlemler öncesinde kullanılmaktadır.

Amaç Kilit Modu

Diğer üç kilit modunu niteleme amacıyla kullanılmaktadır. Veritabanında *amaç paylaşılan kilitler, amaç ayrıcalıklı kilitler* ve *amaç güncelleştirme kilitleri* bulunabilir. Amaç kilit modu kilit kuyruğu yaratır. Bu kilit kuyruğu, bağlantı sıralarını ve bu bağlantılar ile ilişkili kullanılan kaynakların okuma ve güncelleme haklarını göstermektedir. MS SQL Server VTYS amaç kilit modunu, belli kaynaklar üzerine konacak kilitlerin sıradaki amaçlarını göstermek için kullanmaktadır. Örneğin bir hareket, bir tabloyu kilitlemeyi deniyorsa veritabanı sunucusunun bu tabloya ait sayfa veya satırları üzerinde hâlen mevcut bir kilit bulunup bulunmadığını tespit etmesi gerekmektedir. Bir kayıt satırında paylaşılan kilit konmuş ise, ilgili kayıt satırının bulunduğu sayfa üzerine amaç paylaşılan kilit konmaktadır.

Satırlar sayfalar üzerinde bulunurken, sayfalar ise tablo ya da indeks verileri içeren fiziksel veri bloklarıdır.

Özel Kilit Modları

MS SQL Server VTYS üç adet ek kilit modu daha sağlamaktadır. Bu kilit modları şunlardır: şema kararlılık kilit modu, şema değiştirme kilit modu ve toplu güncelleştirme kilit modu. Tablo 5.10'da MS SQL Server VTYS'de kullanılan kilit modları, kısaltmaları ve kısa açıklamaları listelenmiştir.

Tablo 5.10
MS SQL Veritabanı Sunucusu Kilit Modları

Kısaltma	Kilit Modu	Açıklama
S	Paylaşılan	Kilitli kaynaklar üzerinde, diğer işlemlerin okuma işlemi yapmasına izin verilirken, değişiklik yapılmasına izin verilmez.
X	Ayrıcalıklı	Kilitli kaynaklar üzerinde hem okuma hem de değişiklik yapılmasına izin verilmez.
U	Güncelleştirme	Diğer işlemlerin, güncelleştirme veya ayrıcalıklı kilit koymasını engeller. Değişiklik yapılacak verilerin aranması işleminde kullanılır.
IS	Amaç Paylaşılan	İlgili kaynak bileşenlerinin paylaşılan modda kilitlendiğini gösterir. Sadece tablo ya da sayfa düzeyinde uygulanabilir.
IU	Amaç Güncelleştirme	İlgili kaynak bileşenlerinin güncelleştirme modunda kilitlendiğini gösterir. Sadece tablo ya da sayfa düzeyinde uygulanabilir.
IX	Amaç Ayrıcalıklı	İlgili kaynak bileşenlerinin ayrıcalıklı modda kilitlendiğini gösterir. Sadece tablo ya da sayfa düzeyinde uygulanabilir.
SIX	Amaç Ayrıcalıklı ile Paylaşılan	Paylaşılan kilit mod uygulanmış kaynağın, ayrıcalıklı kilit modunda bir bileşeni (sayfa veya kayıt satırı) olduğunu gösterir.
SIU	Amaç Güncelleştirme ile Paylaşılan	Paylaşılan kilit mod uygulanmış kaynağın, güncelleştirme kilit modunda bir bileşeni (sayfa veya kayıt satırı) olduğunu gösterir.
UIX	Amaç Ayrıcalıklı ile Güncelleştirme	Güncelleştirme kilit mod uygulanmış kaynağın, ayrıcalıklı kilit modunda bir bileşeni (sayfa veya kayıt satırı) olduğunu gösterir.
Sch-S	Şema Kararlılık	Bu tabloyu kullanan bir sorgunun derlendiğini gösterir.
Sch-M	Şema Değiştirme	Tablo yapısının değiştirildiğini gösterir.
BU	Toplu Güncelleştirme	Bir tablo içine toplu kopyalama işlemi yapıldığında kullanılır.

MS SQL Server VTYS farklı tipte kaynaklar üzerinde kilit uygulayabilmektedir. Kilitlenebilecek kaynak türleri; satır tanımlayıcıları veya anahtarları, sayfalar, tablo vb. veritabanı nesneleri, veritabanları ve diğer kaynaklar olabilmektedir. Satırlar sayfalar üzerinde bulunurken, sayfalar ise tablo ya da indeks verileri içeren fiziksel veri bloklarıdır. Tablo 5.11'de MS SQL Server VTYS tarafından kilit uygulanabilecek kaynaklar görülmektedir.

Bir tablo üzerinde mevcut kilit modlarının durumuna göre eş zamanlı erişim isteklerinde hangi kilit modlarının kullanılabileceğini gösteren bir uyumluluk matrisi hazırlayınız.

SIRA SİZDE 3

Tablo 5.11
MS SQL Server VTYS Kilitlenebilir Kaynaklar

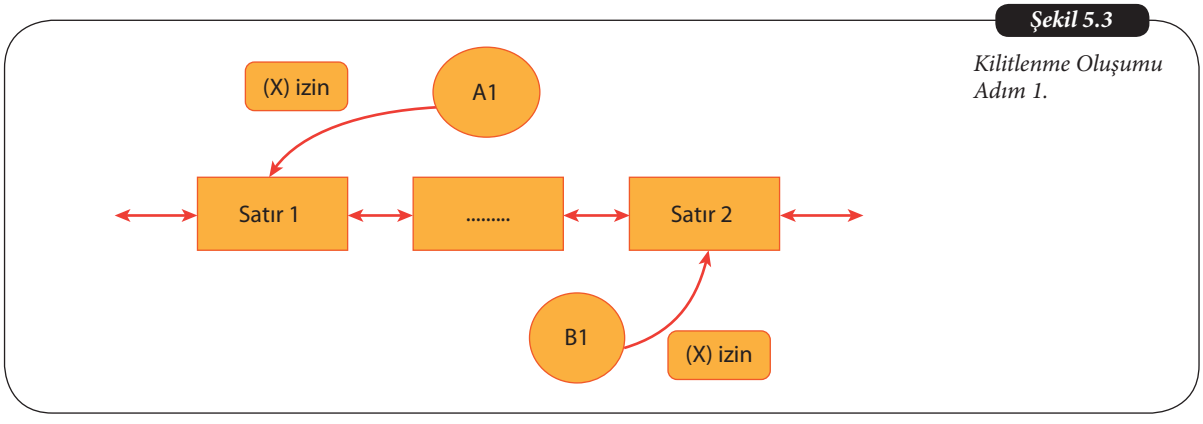
Kaynak Adı	Açıklama
Allocation unit	Veri tiplerine göre gruplanmış birbirleriyle ilişkili sayfa kümeleri. Örneğin, veri kayıt satırları, indeks satırları
Application	Uygulamanın belirlediği kaynaklar
Database	Tüm veritabanı kilidi
Extent	Bitişik sekiz adet 8kBayt'lık veri veya indeks sayfasından oluşan yerleşim birimi. Büyüklüğü 64kBayt'tır.
File	Veritabanı dosyası
HOBT	Heap veya B-tree
Metadata	Sistem üstverisi
Key	İndeks satır kilidi
Object	Veritabanı nesnesi
Page	8kBayt veri veya indeks sayfası
RID	Satır tanımlayıcı
Table	Tüm tabloyu, veri ve indeksleri kilitleyen kaynak.

KİLİTLENME

Kilitlenme: Deadlock

Veritabanı yönetim sistemleri içinde karşılaşılan en önemli ortak problemlerden biri eş zamanlı olarak çalışan hareketlerin birbirlerini bloke etmeleridir. Bloke etme meydana geldiğinde birçok sorgu ve hareket birbirlerini bloke etmektedir. Bu da sorguların yürütülme zamanlarını arttırmakta ve sorgu zaman aşımlarının meydana gelmesine sebep olmaktadır. Bunların tamamı VTYS ile gerçekleşen kullanıcı deneyimini olumsuz etkilemektedir.

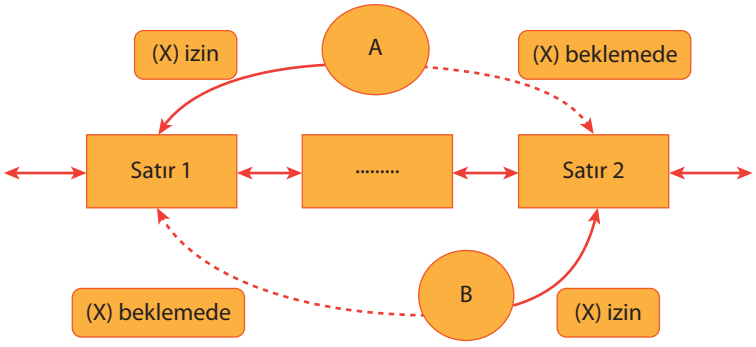
Kilitlenme, iki ya da daha fazla hareketin aynı kaynaklara erişim gerçekleştirmek istediği durumlarda meydana gelmektedir. Örneğin, tablo güncellemesi gerektiren iki farklı hareket olduğunu düşünelim. Şekil 5. 3'te görüldüğü gibi birinci adımda, A hareketi, satır1 kayıt satırını, B hareketi ise satır2 kayıt satırını güncellemektedir. Her iki hareket de kayıt satırları üzerinde ayrıcalıklı (X) kilit modunun meydana gelmesini sağlayacaktır.



Şekil 5.3
Kilitlenme Oluşumu Adım 1.

Şekil 5.4

Kilitlenme Oluşumu Adım 2.



A hareketinin bu kez satır2 kaydı üzerinde güncelleme işlemi gerçekleştirmek istediğini varsayalım. Bu amaçla da satır2 kaydı üzerinde güncelleştirme (U) ya da ayrıcalıklı (X) kilit modu uygulamak isteyecektir. Fakat B hareketi tarafından satır2 üzerinde ayrıcalıklı (X) kilit modu bulunduğundan A hareketi bloke edilecektir. Benzer şekilde B hareketi satır1 kaydı üzerinde güncelleştirme işlemi gerçekleştirmek istediğini düşünelim. A hareketinin satır1 kaydı üzerinde ayrıcalıklı kilit modu bulundurması sebebiyle bu kez B hareketi bloke edilecektir. Şekil 5. 4'te her iki hareketin işlemlerini yürütmeye devam edemediği ve birbirlerini beklediği görülmektedir. Bu şekilde VTYS üzerinde bir kilitlenme söz konusu olmaktadır.

Kilitlenme İzleyicisi: Deadlock monitör

Kilitlenmeyi önlemek amacıyla **kilitlenme izleyicisi** adındaki sistem görev izleyicisi her beş saniyede bir yürütülür. Böylece sistemde bir kilitlenme olup olmadığı kontrol edilmiş olur. Bir kilitlenme tespit edildiğinde kilitlenmeye sebep olan hareketlerden birisi üzerinde rollback işlemi gerçekleştirilir. Bu durumda MS SQL Server VTYS üzerinde ilgili hareket tarafından oluşturulmuş tüm kilitler serbest bırakılarak diğer hareketlerin devam etmesine izin verilmiş olur.

Veritabanı yönetim sistemlerinde kilitlenme olasılığını azaltmak amacıyla farklı yöntem ve yaklaşımlar kullanılmaktadır. Bunlardan bazıları aşağıda listelenmektedir.

SIRA SİZDE 4

Kilitlenmeyi önlemek amacıyla kilitlerin kısa tutulması hareket içinde nasıl gerçekleştirilebilir?

- Sorgu optimizasyonu yapılmalıdır. Sorgular ister basit isterse de karmaşık olsun fiziksel disk okuması ve sunucu bellek kullanımı gibi asgari sistem kaynağına ihtiyaç duymaktadırlar. Veritabanı yönetim sistemlerindeki çoğu ortak kilitlenme sebebi optimize edilmemiş sorgulardır. Bu sorgular sistem kaynaklarının gereksiz kullanımına sebep olmaktadır. Örneğin doğru indeks kullanımı sadece performans artırmakla kalmaz ayrıca okuma ve kilitlenme gerçekleştirilecek kayıt satırı sayısını da azaltır. Kilitlenecek kayıt satırı sayısının artması farklı hareketlerin aynı veri üzerinde işlem yapma olasılığını da arttıracaktır.
- Kilitler mümkün olduğunca kısa tutulmalıdır.
- Gerekli veri tutarlılığını sağlayan en düşük izolasyon düzeyi kullanılmalıdır. Böylelikle paylaşılan (S) kilit tutulma zamanı azaltılır. Örneğin tekrarlanabilir okuma ya da serileştirilebilir izolasyon düzeyi yerine kaydedilen okuma düzeyi kullanılmalıdır. Bu durumda bir hareket içinde veri kaydı üzerinde select ifadesi ile konulan paylaşılan (S) kilit modu okumanın hemen sonrasına serbest bırakılır. Böylelikle hareket bloğu içinde bir update ifadesi bulunuyor ise bloke edilmemiş olur.

- Varlıklara aynı sırada erişim sağlanmalıdır.
- Örneğin tablo 5.12'de görüleceği üzere bir hareket, satışlar tablosunu güncelledikten sonra ürünler tablosundan okuma gerçekleştiriyor olsun. Bu durumda başka bir hareketin önce ürünler tablosunu güncellemesi ve sonrasında satışlar tablosundan okuma yapması engellenmelidir.
- Veritabanı kayıtlarını okuma işlemlerinde kilitlenmeyi önlemek amacıyla with nolock kullanılabilmektedir. Bu durumda, tablonun kilitli olması durumunda da kilitlenme hatası alınmaz ve sorgu çalıştırılabilir. Bununla beraber kirli okuma eş zamanlılık probleminin meydana gelmesi mümkün olabilmektedir.
- Tekrar deneme mantığı kullanılmalıdır. Kritik kod blokları *try..catch* ifadeleri arasına yazılmalıdır. Böylelikle **1205** hata numarası kontrol edilerek bir kilitlenme meydana gelip gelmediği kontrol edilir ve kilitlenme meydana gelmiş ise işlem tekrar edilir.

1205: Kilitlenme hata numarası

Hareket 1	Hareket 2
BEGIN TRANSACTION UPDATE Satislar SET Urun_Fiyati=@urunFiyat Where Urun_No=12; SELECT @urunFiyat=Urun_Fiyati FROM Urunler Where Urun_No=12; COMMIT TRANSACTION;	BEGIN TRANSACTION UPDATE Urunler SET Urun_Fiyati=2500 Where Urun_No=12; SELECT * FROM Satislar; COMMIT TRANSACTION

Tablo 5.12
Varlıklara Aynı Sırada Erişim Sağlanması

Özet

Hareket yönetimi temel kavramlarını açıklamak.

Birden fazla veritabanı işleminin gerçekleştirildiği SQL kod bloğuna hareket adı verilmektedir. Veritabanında bulunan verilerin tutarlılığı açısından bir hareket bloğu içinde yer alan işlemlerin tamamının başarı ile gerçekleştirilmesi gerekmektedir. Eğer hareket bloğu içinde başarısız işlemler gerçekleşmiş ise o veritabanında bulunan verilerin hareket başlamadan önceki durumlarına dönmüş olması gerekmektedir. Bu amaçla veritabanı hareket bloklarının, İngilizce karşılıklarının baş harfleri kullanılarak ACID kuralı olarak tanımlanmış dört özelliğe sahip olması gerekmektedir; Bölünmezlik, tutarlılık, izolasyon ve devamlılık özellikleri. Hareket yönetimi otokayıt hareket, açık hareket ve örtük hareket olmak üzere üç farklı şekilde gerçekleştirilmektedir. Veritabanı yönetim sistemlerinde hareket yönetimi dendiğinde genel olarak açık hareketten bahsedilmektedir. Açık hareketler kullanıcılar tarafından oluşturulan ve yönetilen hareket şeklidir.

Eş zamanlılık kavramı ve eş zamanlılık problemlerini açıklamak.

Eş zamanlılık, veritabanı kaynaklarına aynı anda birden fazla kullanıcının ya da hareketin açılan oturumlar üzerinden erişmeye çalışması olarak tanımlanmaktadır. Veritabanı kaynaklarına eş zamanlı olarak birden fazla kullanıcı ya da hareketin erişmek istemesi birtakım problemlere neden olmaktadır. Bir hareket tablodaki bir kayıt üzerinde değişiklik gerçekleştirdiği anda başka bir hareket aynı kayıt üzerinde güncelleme yapmak istediği zaman hangi hareket son commit işlemini yaptı ise o hareketin değişiklikleri kalıcı olur ve diğer hareketin güncellemesi kaybolur. Buna kayıp güncelleme adı verilmektedir. Kirli okuma ise bir hareket tarafından değişikliğe uğramış olan verinin kalıcı olarak kaydedilmeden eş zamanlı olarak başka bir hareket tarafından okunduğu eş zamanlılık problemidir. Tekrarlanamayan okuma problemlerinde bir hareket içinde aynı kayıt bilgisi için birden fazla okuma yapıldığı sırada eş zamanlı olarak başka bir hareket tarafından değişiklik gerçekleştiriliyor ise her seferinde farklı değer okunabilmektedir. Ayrıca yine birden fazla okuma gerçekleştirilen bir hareketteki iki okuma arasında eş zamanlı gerçekleşen diğer hareketler tarafından yeni kayıt eklenmesi hayalet okuma eş zamanlılık problemini meydana getirmektedir.

İzolasyon düzeylerini ifade etmek.

Aynı anda aynı kaynaklara erişmeye çalışan diğer hareketlerin gerçekleştirdiği işlemlerden dolayı meydana gelen eş zamanlılık problemlerini önlemek amacıyla hareket blokları içinde izolasyon düzeyleri belirlenmektedir. İzolasyon düzeyleri, başka hareketler tarafından değişiklik gerçekleştirilen kaynak veya verilerden bir hareketin nasıl izole edileceğini gösterir. İzolasyon düzeyleri, bir veri okunduğu anda kilit gerekip gerekmediğini, gerekiyor ise ne türde bir kilitleme modu kullanılacağını, kilitleme işleminin ne kadar süreceğini kontrol etmek amacıyla kullanılmaktadır. Başka bir hareket tarafından değiştirilen kayıtlar okunurken; ayrıcalıklı kilit kullanımının kaldırılmasına kadar okumanın bloke edilmesi, SQL ifade ya da hareket başladığında kayıtların kalıcı olarak kaydedilmiş sürümlerine erişime izin verilmesi, kalıcı kayıt işleminin henüz gerçekleşmediği değişikliklerin de okunabilmesi gibi süreçler yine izolasyon düzeylerinin kontrolündedir. Kaydedilmiş okuma, kaydedilmemiş okuma, tekrarlanabilir okuma ve anlık görüntü okuma olmak üzere dört farklı izolasyon düzeyi bulunmaktadır.

Kilitleme kavramını ve kilitleme modlarını açıklamak.

Eş zamanlı erişimlerde meydana gelebilecek eş zamanlılık problemlerinin çözümünde erişim yapılacak olan veritabanı kaynaklarına ya da kayıtlara erişime bir süreliğine sadece tek hareket için izin verilmesi, diğer bütün hareket erişimlerine karşı engellenmesi kilitlenme olarak tanımlanmaktadır. Veritabanı üzerinde satır, sayfa ya da tablo düzeylerinde kilitleme işlemi gerçekleştirilebilir. Bir hareket, veritabanı kayıtları üzerinde değişiklik yapmaya başladığında hareket sonuna kadar kayıtların tutarlılığını sağlamak amacıyla kilitler. Hareketin kayıtlar üzerinde oluşturduğu kilit mekanizmasını ne uzunlukta tutacağını hareketin izolasyon düzeyi belirlemektedir. Hareket esnasında kayıtlar üzerinde oluşturulan tüm kilitler hareketin sona ermesiyle birlikte kaldırılmaktadır. Paylaşılan ve ayrıcalıklı olmak üzere iki temel kilit modu bulunmaktadır. Bu iki kilit modunun yanında paylaşılan ve ayrıcalıklı kilit modu birleşimi bir kilitleme modu daha bulunmaktadır. Amaç kilit modu bu üç kilit modunu nitelendirmek amacıyla kullanılmaktadır.

Kilitlenme sorunlarını ve kilitlenmeyi giderebilecek yöntemleri tanımlamak.

İki veya daha fazla hareketin karşılıklı olarak birbirlerinin kilitlediği kaynaklara erişmek istemesiyle kilitlenme meydana gelmektedir. Her iki hareket, işleyişlerine devam edebilmek için birbirlerinin meydana getirdiği kilitlerin kaldırılmasını bekleyerek sistem kaynaklarını olumsuz yönde etkilemektedir. Bazı kilitlenmeler sonucunda veritabanı sunucusu cevap veremez duruma bile gelebilmekte ve tekrar başlatılmaları gerekebilmektedir. Sorgu optimizasyonlarının yapılması, kilitlerin mümkün olduğunca kısa tutulmaları, gerekli veri tutarlılığını sağlayan en düşük izolasyon düzeyinin kullanılması, "`try...catch`" blokları arasında kilitlenme hata kontrolünün yapılması ve "`with nolock`" kullanımı gibi farklı yöntemler ile kilitlenmeler kontrol altına alınabilmekte ve meydana gelme olasılıkları düşürülebilmektedir.

Kendimizi Sınayalım

1. Bir hareket bloğu içindeki tüm işlemlerin "ya hep, ya hiç" mantığıyla yürütülmesi, aşağıda verilmiş olan hangi ACID kuralına karşılık gelmektedir?

a. Bölünmezlik
b. Tutarlılık
c. İzolasyon
d. Devamlılık
e. Kilitlenemezlik

2. Veritabanı sunucusu dosyalarından hareket günlüğü dosyası aşağıda verilmiş olan hangi ACID kuralını sağlamak amacıyla kullanılmaktadır?

a. Bölünmezlik
b. Tutarlılık
c. İzolasyon
d. Devamlılık
e. Kilitlenemezlik

3. Veritabanı sunucusu varsayılan hareket yönetim modu aşağıdakilerden hangisidir?

a. Otokayıt hareket
b. Örtük hareket
c. Açık hareket
d. Paylaşılan hareket
e. Ayrıcalıklı hareket

4. Kullanıcılar tarafından başlatılıp sonlandırıldığı hareket yönetim modu aşağıdakilerden hangisidir?

a. Otokayıt hareket
b. Örtük hareket
c. Açık hareket
d. Paylaşılan hareket
e. Ayrıcalıklı hareket

5. Bir hareket tarafından değiştirilmiş fakat kalıcı olarak veritabanına kayıt edilmemiş bir verinin başka bir hareket tarafından geçerli bir veri gibi okunması aşağıdaki eş zamanlılık problemlerinden hangisi?

a. Kayıp güncelleme
b. Tekrarlanamayan okuma
c. Kirli okuma
d. Hayalet okuma
e. Çift okuma

6. Aynı hareket içinde yürütülen ve aynı tablo üzerinde gerçekleştirilen iki sorgulama arasında eş zamanlı olarak başka bir hareket, okunan aralık içindeki veri kümesine yeni bir kayıt eklendiğinde aşağıdaki eş zamanlılık problemlerinden hangisinin oluşma ihtimali vardır?

a. Kayıp güncelleme
b. Tekrarlanamayan okuma
c. Kirli okuma
d. Hayalet okuma
e. Çift okuma

7. Varsayılan veritabanı sunucusu izolasyon düzeyi aşağıdakilerden hangisidir?

a. Kaydedilmiş okuma izolasyon düzeyi
b. Kaydedilmemiş okuma izolasyon düzeyi
c. Tekrarlanabilir okuma izolasyon düzeyi
d. Serileştirilebilir izolasyon düzeyi
e. Anlık görüntü izolasyon düzeyi

8. Aşağıdakilerden hangisi veritabanı sunucusu üzerindeki en kısıtlayıcı izolasyon düzeyidir.

a. Kaydedilmiş okuma izolasyon düzeyi
b. Kaydedilmemiş okuma izolasyon düzeyi
c. Tekrarlanabilir okuma izolasyon düzeyi
d. Serileştirilebilir izolasyon düzeyi
e. Anlık görüntü izolasyon düzeyi

9. Paylaşılan kilit mod uygulanmış kaynağın, ayrıcalıklı kilit modunda bir bileşeni olduğunu gösteren kilit modu aşağıdakilerden hangisidir?

a. Amaç Ayrıcalıklı ile Paylaşılan
b. Amaç Güncelleştirme ile Paylaşılan
c. Amaç Ayrıcalıklı ile Güncelleştirme
d. Amaç Güncelleştirme
e. Amaç Ayrıcalıklı

10. Kilitli kaynaklar üzerinde hem okuma hem de değişiklik yapılmasına izin verilmeyen kilit modu aşağıdakilerden hangisidir?

a. Paylaşılan
b. Amaç Güncelleştirme ile Paylaşılan
c. Amaç Ayrıcalıklı ile Güncelleştirme
d. Güncelleştirme
e. Ayrıcalıklı

Kendimizi Sınayalım Yanıt Anahtarı

1. a	Yanıtınız yanlış ise "Hareket ve Özellikleri" konusunu yeniden gözden geçiriniz.
2. d	Yanıtınız yanlış ise "Hareket ve Özellikleri" konusunu yeniden gözden geçiriniz.
3. a	Yanıtınız yanlış ise "Hareket Yönetimi" konusunu yeniden gözden geçiriniz.
4. c	Yanıtınız yanlış ise "Hareket Yönetimi" konusunu yeniden gözden geçiriniz.
5. c	Yanıtınız yanlış ise "Eş zamanlılık ve Eş zamanlılık Problemleri" konusunu yeniden gözden geçiriniz.
6. d	Yanıtınız yanlış ise "Eş zamanlılık ve Eş zamanlılık Problemleri" konusunu yeniden gözden geçiriniz.
7. a	Yanıtınız yanlış ise "İzolasyon Düzeyleri" konusunu yeniden gözden geçiriniz.
8. d	Yanıtınız yanlış ise "İzolasyon Düzeyleri" konusunu yeniden gözden geçiriniz.
9. a	Yanıtınız yanlış ise "Kilitleme Yönetimi" konusunu yeniden gözden geçiriniz.
10. e	Yanıtınız yanlış ise "Kilitleme Yönetimi" konusunu yeniden gözden geçiriniz.

Sıra Sizde Yanıt Anahtarı

Sıra Sizde 1

Bir banka hesabından başka bir banka hesabına havale yapılmak istendiğinde öncelikle havale yapan hesap bakiyesinden havale miktarı düşülür. Daha sonra kendisine havale yapılan hesap bakiyesine bu havale miktarı eklenerek havale işlemi gerçekleştirilmiş olur. Havale işlemi her zaman bu şekilde başarılı bir şekilde gerçekleşmeyebilir. Örneğin, havale gönderen hesap bakiyesinden havale miktarı düşüldüğü sırada elektrik kesilebilir ya da sunucu cevap veremeyip kilitlenebilir. Bu durumda, havale gönderen hesaptan havale miktarı düşülmüş fakat kendisine havale yapılan hesap bakiyesine eklenmemiş olacaktır. Bu şekilde havale sahibi bilgisi kaybedilecek ve veri tutarlılığı ortadan kalkacaktır. Bölünmezlik kuralı, hareketlerde tüm işlemlerin başarılı olarak gerçekleştiği takdirde geçerli kabul edileceğini garanti altına alır. Hareket bloğu içinde işlemlerin bir kısmı başarılı bir kısmı başarısız tamamlanacak olursa hiçbir işlem gerçekleştirilmemiş gibi hareket başladığı zamanki duruma geri dönülür.

Sıra Sizde 2

İzolasyon Düzeyi	Kirli Okuma	Tekrarlanamayan Okuma	Hayalet Okuma
Kaydedilmemiş okuma	Var	Var	Var
Kaydedilmiş okuma	Yok	Var	Var
Tekrarlanabilir okuma	Yok	Yok	Var
Serileştirilebilir	Yok	Yok	Yok

Sıra Sizde 3

	Tablo Üzerindeki Mevcut Kilit Modu					
İstenen Kilit Modu	(IS)	(S)	(U)	(IX)	(SIX)	(X)
(IS)	Evet	Evet	Evet	Evet	Evet	Hayır
(S)	Evet	Evet	Evet	Hayır	Hayır	Hayır
(U)	Evet	Evet	Hayır	Hayır	Hayır	Hayır
(IX)	Evet	Hayır	Hayır	Evet	Hayır	Hayır
(SIX)	Evet	Hayır	Hayır	Hayır	Hayır	Hayır
(X)	Hayır	Hayır	Hayır	Hayır	Hayır	Hayır

Sıra Sizde 4

Bir hareket bloğu yürütülmeye başlandığında, tüm ayrıcalıklı kilitler (X) hareket sonuna kadar kayıtlar üzerinde aktif olarak tutulmaktadır. Bu yüzden hareketler kısa tutulmalı ve veri güncelleme gibi ayrıcalıklı kilit modu meydana gelmesini sağlayan SQL işlemleri mümkün olduğunca hareket sonlarına yakın yerlerde gerçekleştirilmelidir.

Yararlanılan ve Başvurulabilecek Kaynaklar

Uzunköprü, S. (2014). **Projeler İle SQL Server 2014,** Kodlab

Elbahadır, H. (2012). **T-SQL & SQL SERVER 2012,** Kodlab

Adar, İ. (2012). **SQL Server 2012,** Pusula Yayıncılık

Gözüdeli, Y. (2013). **Yazılımcılar için SQL Server 2014 & Veritabanı Programlama,** Seçkin Yayıncılık

Korotkevitch, D. (2014). **Pro SQL Server Internals,** APress.

Delaney, K. (2013). **Microsoft SQL Server 2012 Internals,** Microsoft Press

Ben-Gan, I. (2012). **Microsoft SQL Server 2012 T-SQL Fundamentals,** Microsoft Press

Bolton, C., Langford, J., Berry G., Payne G., Banerjee, A. (2012). **Professional SQL Server 2012 Internals nad Troubleshooting,** John Wiley & Sons.