VERİTABANI PROGRAMLAMA

4

Amaçlarımız

Bu üniteyi tamamladıktan sonra;

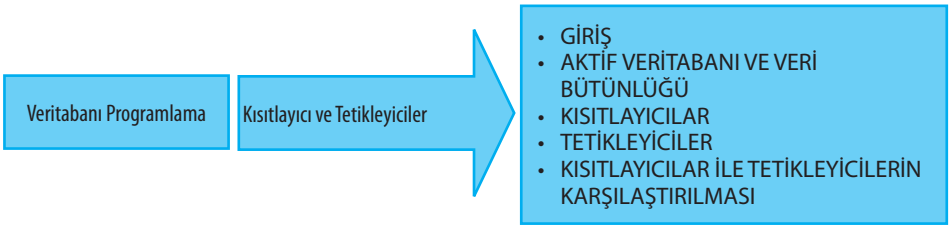
- Pasif ve aktif veritabanları arasındaki farkları tanımlayabilecek,
- Veri bütünlüğünün önemini ve gerekliliğini anlayabilecek,
- Veri bütünlüğü sağlayabilmek için kısıtlayıcı ve tetikleyiciler oluşturabilecek,
- Kısıtlar ve tetikleyiciler arasındaki farkları görebilecek

bilgi ve becerilere sahip olabileceksiniz.

Anahtar Kavramlar

- Aktif Veritabanı
- Veri Bütünlüğü
- Kısıtlayıcılar
- Tetikleyiciler

İçindekiler



Kısıtlayıcı ve Tetikleyiciler

GİRİŞ

Verilerin sorgulanması veya değiştirilmesi kullanıcı veya arayüz tarafından oluşturulacak sorgularla yapılan veritabanları pasif veritabanları olarak isimlendirilmektedir. Ancak bazı uygulamalarda sorgulamaya gerek kalmadan veritabanının otomatik olarak tepki göstermesi gerekliliği vardır. Eğer bir veritabanının kendisi otomatik olarak tepki üretebiliyor ise bu tür veritabanı aktif veritabanı olarak isimlendirilmektedir. Aktif veritabanı sistemlerinde, oluşturulacak bazı kurallar ile otomatik tepki vermesi sayesinde daha güçlü uygulamalar yapılabilmesi sağlanmıştır. Veritabanı ister aktif ister pasif olsun, en genel anlamda ilişkisel bir veritabanı birbiriyle ilişkili tablolardan, tablolar da satır ve sütunlardan oluşmaktadır. Bu tablolardaki kayıtların ve verilerin birbiriyle tutarlı olması gerekmektedir ki bu tutarlılık veri bütünlüğü olarak ifade edilmektedir. Veri bütünlüğü, bir tabloda verilere UPDATE, DELETE veya INSERT gibi işlemler yapılırken diğer tablo ya da tablolardaki verilerin birbirleriyle uyum içinde olmasının sağlanmasıdır. Veri bütünlüğü bu uyumluluk ile veri tutarlılığının garanti altına alınması demektir. Bu bütünlüğü sağlamak için Tanımlanabilir Veri Bütünlüğü (Declarative Data Integrity) ve Prosedürel Veri Bütünlüğü (Procedural Data Integrity) olmak üzere iki farklı şekilde yapılabilmektedir. Bunlardan ilki, kullanımı çok basit ve nesnelerin kendi özelliklerini kullanarak sağlanabilen veri bütünlüğüdür. Ancak tanımlanabilir veri bütünlüğünde çok fazla müdahale edilememesi ve programlama imkânının sağlanamaması gibi dezavantajlarından dolayı daha üst düzey programlamayla müdahale edilerek üst düzey veri bütünlüğü sağlanması gerekir ki bu da ikinci tip olarak karşımıza çıkar ve prosedürel veri bütünlüğü olarak isimlendirilir.

AKTİF VERİTABANLARI VE VERİ BÜTÜNLÜĞÜ

Veritabanı sistemleri, büyük miktarlardaki verileri depolamaya, istenildiği zaman bu verilere erişmeye, veriler üzerinde sorgulamaya ve verileri değiştirmeye yarar. Buna rağmen, çoğu veritabanı sistemleri pasif sistemlerdir. Pasif sistemlerde verilerin sorgulanması veya değiştirilmesi kullanıcı veya arayüz tarafından oluşturulacak sorgularla yapılmaktadır. Veritabanı üzerine yapılan araştırmalarda ve uygulamalarda bir veritabanı sisteminde herhangi bir sorgulamaya gerek kalmadan da veritabanının tepki vermesi gerekliliği ortaya çıkabileceği gözlemlenmiştir. Sorgulamaya gerek kalmadan veritabanının kendisi otomatik olarak bir tepki üretebiliyor ise artık bu veritabanı sistemi aktif olur ve böylece birçok yararlı özellikleri bu aktif davranış tarafından sağlanabilir. **Aktif veritabanı** sistemlerinde, oluşturulacak bazı kurallar ile tepki vermesi sayesinde daha güçlü uygulamalar

Veritabanı içinden veya dışından gelen olaylara oluşturulacak kurallar sayesinde otomatik olarak tepki üreten veritabanına **aktif veritabanı** denir.

yapılabilmesi sağlanmıştır. Genel olarak bu tür aktif veritabanı sistemlerinde, otomatik olarak gerçeklenmesi istenen eylem veya tepkiler bir koşula bağlanır. Eğer koşul gerçeklenirse sorgulamaya gerek kalmadan veritabanı sistemi kendisine tanımlanan eylem veya tepkiyi otomatik olarak gerçekler. Bu tür eylem veya tepkilere veritabanındaki kayıtların bütünlüğünün sağlanması, verilere erişim ve değişimlerin incelenmesi, türetilmiş verilerin korunması, sürüm geçmişlerinin izlenmesi, sistemde ortaya çıkabilecek ani değişikliklere kolayca uyum sağlayabilmek en basit örneklerdir. Oluşturulabilecek kurallar sayesinde, veritabanı sistemleri geniş ve verimli bilgi tabanı ve uzman sistemler oluşturmak için uygun bir platform olur.

Aktif veritabanında veritabanın otomatik olarak tepki göstermesi istenen fonksiyonlar yerine veritabanı uygulama programları tarafından yapılabilmesi de mümkün olabilir. Aslında, uygulama programları bu şekilde kullanılmaktadır. Bir uygulama programı ile aktif veritabanlarında olduğu bir eylem gerçeklenmesi için ilgili koşulun sağlanıp sağlanmadığını kontrol etmek için periyodik olarak veritabanını yoklamak da bir yaklaşımdır. Burada eylemin koşulunun kontrolünün hangi sıklıkla yapılacağı sıkıntısı ile karşılaşılır. Kontrol sıklığı çok yüksek olursa veritabanına gereksiz yere aşırı şekilde yüklenilir. Kontrol sıklığı çok düşük olursa da bu sefer eylemin gerçekleştirilmesine geç kalınabilir. Tabi bir başka yaklaşımda veritabanını değiştiren her bir uygulama programında eylem koşulu kontrol edilebilir, eğer koşul doğru ise eylem gerçekleştirilir ancak bu tür bir yaklaşım yazılım mühendisliği açısından uygun görülmemektedir.

Örneğin, bir markette bir ürünün belli bir adedin altına düştüğü zaman otomatik olarak depodan sipariş verilmesi gereken bir uygulama olduğunu düşünelim. Pasif bir veritabanında bu kontrol ve sipariş işlemi iki şekilde yapılabilir. Bunlardan bir tanesi her satışta marketteki kalan ürün miktarı sorgulanır, eğer eşik değerinin altında ise sipariş verilir. Ancak bu yaklaşımda tüm satış yapan terminallerde ürün adedini sorgulama ve sipariş veren eklentinin de olmasını gerektirir. Bununla beraber her bir terminalin sipariş verme yetkisi olması gerektiğinden bazen birbirini tekrar eden siparişler verilebilir. İkinci yaklaşım olarak da belirli aralıklar ile marketteki tüm ürünlerin adedi sayılır, eşik değeri altında olan ürünlerin siparişi verilebilir. Birinci yaklaşımın tersine sipariş tüm terminallerden değil sadece tek bir noktadan yapılır ancak marketteki ürün adedinin kontrolünün ne zaman yapılacağı sorunu ile karşılaşılır. Aktif veritabanlarında ise bu kontrol ve sipariş verme otomatik olarak kontrol edilebilmektedir. Böylece ilk yaklaşımda olduğu gibi her terminale kontrol ve sipariş verme eklentisinin eklenmesi veya ikinci yaklaşımda olduğu gibi kontrol etme sıklığı problemi ile karşılaşılmamaktadır. Aktif veri tabanları eylemleri otomatik gerçekleştirebilmek için üç bileşenden oluşan yapılar kullanır. Bu yapılar bir olayın oluşması, eylemin gerçeklenmesi için gerekli olan koşul veya koşullar ve koşul şartı doğru ise eylemin gerçekleştirilmesidir. Kuralın olay bileşeni, kuralın çalışması için hangi eylemin gerçekleşmesi gerektiğini tanımlar. Kuralın koşul kısmı, eylemin gerçekleşmesi durumunda bu eyleme karşılık yapılması gereken eylemin koşul şartları, son bileşen olan eylemin gerçekleşmesi ise eyleme karşılık veritabanının gösterdiği tepki olarak tanımlanır.

Veritabanı ister aktif ister pasif olsun, en genel anlamda ilişkisel bir veritabanı birbiriyle ilişkili tablolardan, tablolar da satır ve sütunlardan oluşmaktadır. Bu kapsamdan bakıldığında bir veritabanı, birden fazla çalışma sayfasından oluşan bir EXCEL dosyası olarak düşünülebilir. EXCEL deki her çalışma sayfasında bulunan hücreler de veritabanındaki tablolardaki satır ve sütunlara karşılık gelmektedir. Bu hücrelere veri olarak hemen hemen her şeyi girebilmeniz mümkündür. Buna karşılık, veritabanındaki bir tablonun belirli bir sütununda, her girdi verilerin aynı türü içermesi beklenmektedir. Örneğin, belirli bir sü-

tundaki hücrelerin tümünde telefon numaraları veya bir başka hücrede ise soy isimleri bulunabilir. Aslında kötü tasarlanmış veritabanı uygulaması kullanıcının, bir hücreye girilmesi gereken veri tipinden farklı verileri girmesine izin verebilir. Örneğin, Türkiye Cumhuriyeti kimlik numarası 11 haneden oluşmakta ve her bir hane rakamlarla ifade edilmektedir. Veritabanı ve kullanıcı arayüzü uygun şekilde tasarlanmamış ise Türkiye Cumhuriyeti kimlik numarasının tutulması istenen alana 11 haneden farklı veya 11 hane olsa bile rakamlardan hariç farklı karakterler de girmek mümkün olabilir

Veritabanı uygulamalarında, tablodaki her bir sütun için girilebilecek değerlerin bir değerler kümesi vardır. Bir sütun için izin verilen değerler kümesine sütunun tanım kümesi denir. Örneğin, bir sütunun tanım kümesi telefon numarası, banka hesap numarası, doğum tarihi ve cinsiyeti olabilmektedir. Bir sütünün tanım kümesi verinin tipi ile yakından ilişkilidir ancak tamamen veri tipine bağımlı değildir. Bir sütunun tanım kümesi o sütunda tutulması istenen veri türüdür. Bir sütun için kullanabileceğiniz veri türleri kullandığınız veritabanına bağlı olmasına rağmen veri türleri genellikle tam sayı, kayan noktalı sayısı (ondalıklı sayılar), metin ve tarihtir. Veri tipi ile tanım kümesi birbirinden farklı kavramlardır. Örneğin, bir veritabanında bir kişinin doğduğu il ve sahip olduğu evcil hayvan tutulması istenmiş olsun. Her iki sütun içinde veri tipi olarak metin seçilmesi gerekir. Ancak doğduğu il bilgisinin tutulduğu sütuna kedi, köpek gibi evcil hayvan türlerinin girilmesi veya evcil hayvan sütununa da Eskişehir, Ankara gibi şehir isimlerinin girilmesi o sütun için tip olarak geçerli olmasına rağmen mantıksal ve anlamsal olarak geçerli değildir. Hatta Türkiye için doğduğu il sayısı sınırlı olacağından doğduğu il sütununa yazılabilecek metinler bellidir. Bunun için doğduğu il bilgisi için kullanıcıya serbest metin olarak yazmak yerine bir listeden seçmesi sağlanarak verilerin hatalı olmasının önüne geçilebilir. Bir başka örnek olarak bir kişinin doğum yılı bilgisinin gün. ay. yıl formatında tutulması istenmiş olduğu bir veritabanı uygulaması olduğunu düşünelim. Doğum yılı bilgisinin girileceği sütunun veri tip tarih olarak seçilmesi gerekmektedir. Böyle bir durumda o sütun için girilmesi gereken tarihin verinin girildiği tarihten (örneğin, doğum tarihi olarak 17.03.2075 girilmesinin kontrol edilmesi) daha sonra olmadığının kontrolünün yapılması gerekmektedir. Bu tür bir tutarsızlık olmaması için veritabanında veya kullanıcı ara yüzünde gerekli kontrollerin yapılması gerekmektedir.

Bazı durumlarda ise bu tür kontrollerin yapılması zor olmaktadır. Örneğin, bir kişinin soy isminin tutulması gereken bir veritabanı uygulaması düşünelim. Bu uygulamada soy isminin tutulacağı sütunun veri tipi metin olarak seçilmesi gerekir. Veritabanı veya kullanıcı ara yüzünde yapılacak kontroller ile o sütun için sadece metin girilmesi sağlanabilir ama metnin içeriğinin kontrol edilmesi mümkün olmayabilir. Örneğin soyisim için girilmesi gereken yere "aaaaaa" veya "+%&/?" gibi soyisim olamayacak metinler girilirse, yapılan kontrollerde hatalı kayıt olmayacak (çünkü iki örnek de veri türü olarak bakıldığında metindir) ancak mantıksal olarak hatalı olacaktır.

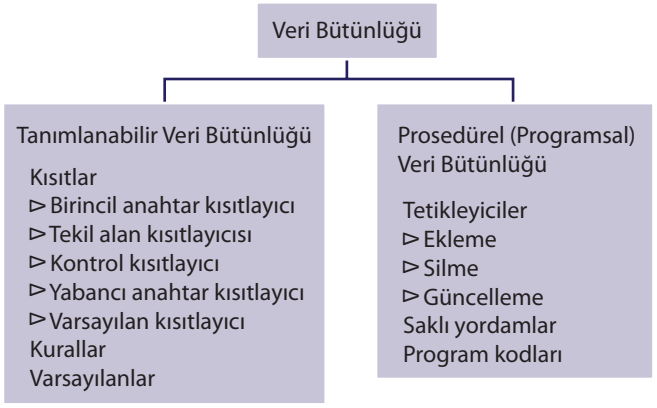
Tüm bunların ışığında veritabanı tablolarındaki sütunlara girilen verilerin hem tip hem de tanım kümesi olarak doğru ve tutarlı olması gerekir. Bu kapsamda verinin doğru ve tutarlı bir şekilde işlenmesi veri bütünlüğü olarak ifade edilmektedir. Verilerin doğruluğunu ve tutarlığını sağlamak için bazı kısıtlamalara ihtiyaç duyulmaktadır. Veritabanındaki veri bütünlüğü açısından bir tablodaki sütunlara doğru ve tutarlı verilerin girilmesinin sağlanması tek başına yeterli olmamaktadır. Örneğin, veritabanındaki bir tabloda veriler silindiğinde, güncellendiğinde veya ekleme yapıldığında yapılan düzenlemelere ve değişikliğe göre diğer tabloların da düzenlenmesi ve değiştirilmesi gerekebilir. Dolayısıyla veritabanı bütünlüğü denildiği zaman sadece girilen kayıtların doğru ve tutarlı olarak anlaşılmaması, bütün tablolardaki kayıtların ve tabloların birbiriyle tutarlı olduğunun an-

laşılması gerekmektedir. Kısacası veri bütünlüğü, bir tabloda veri güncelleme, silme veya ekleme gibi işlemler yapılırken veri tutarlılığının kaybolmamasının garanti altına alınması demektir. Diğer bir ifadeyle bir tablodaki yapılan herhangi bir işlemin diğer tablo ya da tablolardaki verilerle uyum içinde olmasının sağlanmasıdır. Veritabanı yönetim sisteminde veri bütünlüğünü sağlamak için kullanılan yöntemler Şekil 4.1'de verilmiştir.

Şekil 4.1'de görüldüğü üzere veri bütünlüğü temel olarak iki tipe ayrılmaktadır. Bunlardan ilki kullanımı çok basit olan ve tanımlanabilir veri bütünlüğü olarak isimlendirilen tanımlanan nesnelerin kendi özellikleri sayesinde sağlanabilen veri bütünlüğüdür. Ancak bu veri bütünlüğünde, çok fazla müdahale edilememesi ve programlama imkânının sağlanamaması dezavantajları olarak sayılabilir. Bazı durumlarda ise daha üst düzey programlamayla müdahale edilerek daha üst düzey veri bütünlüğü sağlanması gerekir ki bu da ikinci tip olarak karşımıza çıkar ve prosedürel (Programsal) veri bütünlüğü olarak isimlendirilir. Programsal veri bütünlüğü sağlamaya çalışırken kötü yazılacak program ile veritabanına ek yük getirebilir.

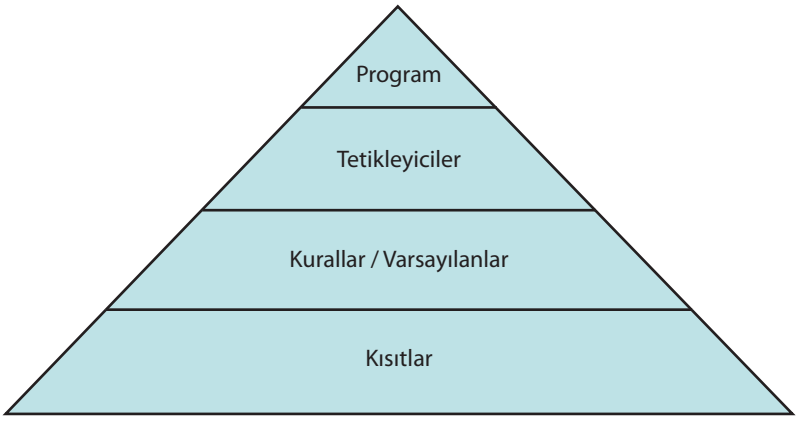
Şekil 4.1

Veritabanı bütünlüğü sağlama teknikleri



Prosedürel veri bütünlüğü, tanımlanabilir veri bütünlüğüne göre daha kapsamlı bütünlük sağlanabilmesine rağmen kullanımı daha karmaşıktır. Veri bütünlüğünü sağlamak için hedefimiz her zaman en alt katmanda çözümler üretmektir. Çünkü üst katmanlarda daha karmaşık yapılar oluşturmamız gerektiğinden daha dikkatli olmamız gerekmektedir. Ayrıca üst katmanlara çıkıldıkça veritabanının üzerine daha fazla yük binmektedir. Şekil 4.2'de katman sıralaması verilmiştir. Şekil 4.2'de görülen katmanlardan en altta olan kısıtlar düşük düzeyde fonksiyonellik sağlarlar. Tablonun yapısı ile birlikte tanımlandıklarından tablodaki ekleme, silme ve güncelleme işlemlerinden önce devreye girdikleri için veritabanına en az yük bindiren, tanımlanabilir veri bütünlüğü sağlama tekniğidir. İkinci katmanda bulunan kurallar ve varsayılanlar ile sağlanabilen veri bütünlüğü yaklaşımları kısıtlar ile sağlanabilmektedir. Kurallar ve varsayılanlar da aynı kısıtlarda olduğu gibi tablodaki değişikliklerden önce devreye girerler. Kısıtlardan farkı ise tablolardan ayrı nesneler olarak veritabanında yer alan tanımlanabilir veri bütünlüğü teknikleridir. Tetikleyiciler ise prosedürel veri bütünlüğü sağlama tekniği olup fonksiyonellik açısından gelişmiş bir koruma yöntemidir. Ancak bu yöntem, adı geçen iki yönteme göre veritabanında yapılan değişiklerden sonra devreye girdiği için veritabanına fazladan yük binmesine sebep olabilir. Şöyle ki, ilk önce eylem gerçekleşir, ondan sonra verinin bütünlüğünün tutarlılığı kontrol edilir. Şayet veri bütünlüğünde bir sorun yoksa tetikleyici hiçbir eylemde bulunmaz. Ancak veri bütünlüğü ile alakalı bir soruna rastlanırsa veritabanının yaptığı

işlemlerin geri alınması gerekecektir. Bu geri alma işlemi de veritabanına fazladan yük binmesine sebep olacaktır. En üst katman olan saklı yordamlar / programlar veritabanı dışında programlama dilleri ile geliştirilirler. Veri bütünlüğü sağlamak için en geniş manada fonksiyonellik sunmalarına rağmen performans olarak en az zayıf kalan veri bütünlüğü sağlama yöntemleridir. Veritabanı dışında programlama ile gerçekleştirilmelerinden ötürü veritabanına herhangi bir başka kaynaktan gelecek verileri denetleyemezler. Bundan dolayı veritabanın bütünlüğünü tam olarak korumak için yeterli değildir. Daha detaylı bilgi en alt katmandan başlayarak aşağıda açıklanmıştır. Saklı yordamlar konusuna önceki bölümlerde değinildiği için bu bölümde ver verilmemiştir.



Şekil 4.2

Veri bütünlüğü sağlama yöntemlerinin fonksiyonellik açısından sıralanması

KISITLAYICILAR

Veritabanında bütünlüğü sağlamak için veri üzerindeki oluşturulmuş mantıksal sınırlamalara kısıt adı verilir. Kısıtların kullanılmasının en temel sebebi veritabanına hatalı giriş yapılmasını engellemektir. Daha önceki bölümlerde oluşturduğumuz bilişim ürünleri satan mağaza yönetimi için oluşturulan veritabanındaki bölümlerde çalışanların cinsiyet bilgisi verisinin sadece E (Erkek) veya K (Kadın) olarak girilmesini kısıtlar ile sağlamak mümkündür. Kısıtlar, veritabanı bütünlüğü sağlama tekniklerinden, tanımlanabilir veri bütünlüğü teknikleri arasında yer almaktadır. Tanımlanabilir veri bütünlüğü esaslarını üç başlık altında toplayabiliriz ve başlıklar en genel hâliyle tablo 4.1'de verilmiştir.

Kısıtlayıcılar tabloların tanımlamasıyla beraber yapılmaktadır. Kısıtlar, tablo oluştururken CREATE TABLE, eğer tablo oluşturulmuşsa ALTER TABLE komutuyla oluşturulabilir. Ancak ALTER TABLE komutu ile kısıt oluşturulacaksa daha önceden sütunlara girilmiş bilgilerin kontrol edilmesi gerekir. Bununla beraber oluşturulacak her kısıta bir isim verilmesi gerekmektedir.

Tablo 4.1
Tanımlanabilir veri bütünlüğü esasları

Tanımlanabilir veri bütünlüğü esasları	Açıklama	Örnek	Kullanılabilecek Teknikler
Satır Bütünlüğü	Tabloya girilen kayıtlardan her bir kaydın diğer kayıtlardan farklı bir değer olmasını sağlar.	Oluşturulan örnek veritabanı için her farklı ürüne farklı ürün numarası verilmesi.	• Birincil anahtar kısıtlayıcı • Tekil alan kısıtlayıcısı
Sütun Bütünlüğü	Tablodaki herhangi bir sütunun hangi gruptan verileri alabileceğini veya NULL bir değer alıp alamayacağını sağlar.	Oluşturulan örnek veritabanı için her ürüne NULL olamayacak şekilde ürün numarası verilmesi. Oluşturulan örnek veri veritabanındaki bölümlerde çalışan çalışanların TC Kimlik numaralarının 11 hane ve rakamlardan oluşması.	• Kontrol kısıtlayıcısı • Varsayılan Kısıtlayıcı • Kurallar • Varsayılanlar
Referanssal Bütünlük	Referanssal bütünlük kısıtları bir kolondaki bilginin diğer bir kolon ile eşleşmesi zorunluluğunu sağlayarak bütünlüğü korur. Eşleşen kolonlar farklı iki tablonun kolonları olabileceği gibi aynı tablonun kolonları da olabilir	Oluşturulan örnek veritabanında oluşturulan Calisanlar tablosundaki ve Urunler tablosundaki bolüm numarasının (Bolum_No) bölümler tablosundaki bölüm numarası ile eşleşmesi.	• Yabancı anahtar kısıtlayıcı

DİKKAT

Kısıtlara verilecek isimlerin mantıklı olması tercih sebebidir. Böylece verilen kısıt isimlerine bakarak hangi tablo veya sütuna kısıt verildiği kolayca anlaşılabilir. Kısıtlara isim vermek için ilk önce kısıtlayıcı tekniğinin kısaltılmış hâli, daha sonra da tablo veya sütun ismi gibi bir yöntem izlenebilir. Örneğin, Bak_Bolumler (bolumler tablosuna birincil anahtar kısıtlayıcısı), Tak_Bolumler_BolumId (Bolumler tablosunun Bolum_No sütununa tekil alan kısıtlayıcısı).

SIRA SİZDE 1

Bir tablonun kendi sütunları arasında ilişkiler kullanılarak kısıt oluşturulabilir mi?

Birincil Anahtar Kısıtlayıcısı (BAK)

Birincil anahtar kısıtlayıcı: Primary Key Constraint

Veritabanında satır bütünlüğü sağlamak için kullanılan bu kısıtlayıcı bir tablodaki bir satıra girilen verilerden her bir satırı diğer satırlardan ayıran tanımlayıcı bir değer olmasını sağlar. Birinci kısıtlayıcı ile bir satıra daha önce girilmemiş değerler girilmesi sağlanır. Bu sayede her satırın diğer satırlardan farklı olmasını gerçekleştirir ve her satırın tekilliğini sağlar. Bir sütuna birincil anahtar kısıtlayıcısı tanımlanır ise o sütundaki tüm satırlara bir değer girilmesini zorlar yani NULL kalamaz. Her tabloda en fazla bir adet birincil anahtar kısıtlayıcısı bulunabilir. Birincil anahtar alana girilen veriler tekil olmak zorundadır. Bir sütunda Türkiye Cumhuriyet'i kimlik numarası tutulmak isteniyor ise o sütuna birincil anahtar kısıtlayıcısı tanımlanabilinir. Çünkü Türkiye Cumhuriyeti'ndeki her bir vatandaşın TC Kimlik numarası vardır (dolayısıyla NULL değeri olmaz) ve her kişinin kimlik numarası birbirinden farklıdır. Bir başka örnek ise veritabanımızdaki Bolumler tablosunda Bolum_No sütunu her bir satırda farklı olmak zorundadır. Aynı Bolum_No değerine sahip birden fazla satır bulunması hâlinde bölümleri birbirinden ayırmak mümkün olamamaktadır. Birincil anahtar kısıtının komut satırıyla oluşturulması aşağıdaki gibidir:

Yeni bir tablo oluştururken kullanımı

```
CREATE TABLE Tablo Adı (
       Sütun_Id INT NOT NULL,
       Sütun_Adi1 VARCHAR (150) NOT NULL
    CONSTRAINT Kısıtlayıcı_Adı PRIMARY KEY (Sütun_Adı)
)
```

Oluşturulmuş bir tabloda kullanımı

```
ALTER TABLE Tablo Adı
      ADD CONSTRAINT Kısıtlayıcı_Adı PRIMARY KEY (Sütun_Adı)
```

Tekil Alan Kısıtlayıcısı (TAK)

Tekil alan kısıtlayıcısı: Unique Constraint

Birincil anahtar kısıtlayıcı gibi veritabanındaki satır bütünlüğünü sağlamak üzere kullanılan bu kısıtlayıcı ile istenilen sütundaki verilerin tekil (her değerden sadece bir tane) olması sağlanır. Birincil anahtar kısıtlayıcısının aksine tablodaki birden fazla sütuna tekil kısıtlayıcı tanımlanabilir ve sütundaki değer NULL olabilir. Eğer sütunu girilen değer NULL değil ise tekil alan kısıtlayıcısı tanımlanan sütuna mutlaka farklı değerler girilmesini sağlar. Örnek veritabanımızdaki Bolumler tablosundaki Bolum_No ve Bolum_Adi sütunlarındaki değerlerin her satırda farklı olması isteniyor ise tekil alan kısıtlayıcısı aşağıdaki gibi tanımlanır:

Yeni bir tablo oluştururken kullanımı

```
CREATE TABLE Bolumler (
       Bolum_No INT not NULL,
       Bolum_Adi VARCHAR (150) not NULL
   CONSTRAINT Tak_Bolumler_Bolum_No  UNIQUE (Bolum_No)
   CONSTRAINT Tak_Bolumler_Bolum_Adi  UNIQUE (Bolum_Adi)
)
```

Oluşturulmuş bir tabloda kullanımı

```
ALTER TABLE Bolumler
ADD CONSTRAINT Tak_Bolumler_Bolum_No UNIQUE (Bolum_No)
ADD CONSTRAINT Tak_Bolumler_Bolum_Adi UNIQUE (Bolum_Adi)
```

Bu bölümde yer alan örneklere ait sql kodlarına https://goo.gl/2uPFyv adrsinden ulaşabilirsiniz. Kodların çalıştırılacağı örnek veritabanı birinci ünitede oluşturulmaktadır.

Kontrol Kısıtlayıcı (KK)

Kontrol kısıtlayıcısı (Check Constraint) tablodaki herhangi bir sütunun hangi gruptan verileri alabileceğini ve girilebilecek verileri bir koşul ile kısıtlayarak istenilen verilerin girilmesini sağlayan bütünlüktür.

Tablodaki herhangi bir sütunun hangi gruptan verileri alabileceğini ve girilebilecek verileri bir koşul ile kısıtlayarak istenilen verilerin girilmesini sağlayan bütünlüktür. Bununla beraber sütunun NULL değer alıp alamayacağının kontrolü de bu kısıtlayıcı ile sağlanabilir. Ayrıca iki farklı sütun değerini karşılaştırarak farklı sütunlardaki veri bütünlüğü de sağlanabilir. Aslında temel olarak Kurallar nesnesi ile aynı işleve sahiptir. Örnek veritabanımızdaki Calisanlar tablosundaki TC_no sütununa 11 karakterin girilmesi, Bolumler tablosundaki Bolum_No sütününa sıfırdan büyük sayı girilmesi vb. kısıtlar bu kısıtlayıcı ile sağlanabilir. Tanımlanması ise aşağıdaki gibidir. Eğer girilen değer istenen değer arasında ise veritabanına girdi yapılır, eğer istenen değer dışında ise bir hata mesajı döndürülerek kaydın düzeltilmesi veya kayıttan vazgeçilmesi istenir.

Yeni bir tablo oluştururken kullanımı

```
CREATE TABLE Bolumler (
       Bolum_Adi VARCHAR (150) not NULL,
       Bolum_No INT not NULL,
  CONSTRAINT KK_Bolumler_Bolum_No CHECK (Bolum_No>0)
)
```

Oluşturulmuş bir tabloda kullanımı

```
ALTER TABLE Bolumler
       ADD CONSTRAINT KK_Bolumler_Bolum_No CHECK (Bolum_No>0)
```

Yabancı Anahtar Kısıtlayıcı (YAK)

Yabancı anahtar kısıtlayıcısı (Foreign Key Constraint): Bir tablodaki bir sütundaki değerin diğer taboloardaki değerlerle denetlenmesini sağlayan kıtıtlayıcıdır.

Bir tablodaki bir sütuna ait değerlerin, başka bir tablonun belli bir sütunundan gelmesinin denetlenmesinin istendiği durumlarda kullanılan yabancı anahtar kısıtlayıcı, tablolar arasındaki veri bütünlüğünü sağladığı gibi ilişkili tabloları da gösterir. Bir tabloya yabancı anahtar kısıtlayıcısı tanımlandığı zaman referans edilen ve referans eden iki tablo arasında bir ilişki kurulması sağlanır. Böylece bir tablo için yabancı anahtar kısıtlayıcısı tanımlanmış ise bu tabloya girilecek her kaydın referans edilen tablodaki kayıtla eşleşmesi gerekir. Eğer yabancı anahtar kısıtlayıcı olan sütun NULL değerler alabiliyorsa bu durumda eşleşme aranmaz. Örnek veritabanımızdaki Urunler ve Calisanlar tablosundaki Bolum_No sütununa yabancı anahtar kısıtlayıcısı tanımlanarak iki tablo arasında ilişki kurulmuş olur. Bu kısıt sayesinde Calisanlar ve Urunler tablosundaki Bolum_No sütununa girilen verinin Bolumler tablosundaki Bolum_No sütununda olması gerekliliği sağlanır ve şöyle tanımlanır:

Yeni bir tablo oluştururken kullanımı

Urunler tablosu için,

```
CREATE TABLE Urunler (
       Urun_No int IDENTITY (1,1) NOT NULL,
       Urun_Adi nvarchar (50) NULL,
       Urun_Sayisi int NULL,
       Urun_Fiyati decimal (18, 2) NULL,
       Bolum_No int NULL,
CONSTRAINT Yak_Urunler_Bolum_No
    FOREIGN KEY (Bolum_No) REFERENCES Bolumler (Bolum_No)
)
```

Calisanlar tablosu için,

```
CREATE TABLE Calisanlar (
       TC_No nvarchar (11) NOT NULL,
       Adi nvarchar (100) NULL,
       Bolum_No int NULL,
       Cinsiyet nchar (1) NULL,
CONSTRAINT Yak_Calisanlar_Bolum_No
    FOREIGN KEY (Bolum_No) REFERENCES Bolumler (Bolum_No)
)
```

Oluşturulmuş bir tabloda kullanımı

Urunler tablosu için,

```
ALTER TABLE Urunler
    ADD CONSTRAINT Yak_Bolumler_Bolum_No
       FOREIGN KEY (Bolum_No) REFERENCES Bolumler (Bolum_No)
```

Calisanlar tablosu için,

```
ALTER TABLE Calisanlar
    ADD CONSTRAINT Yak_Bolumler_Bolum_No
        FOREIGN KEY (Bolum_No) REFERENCES Bolumler (Bolum_No)
```

Yabancı anahtar kısıtlayıcısı tanımlanırken dikkat edilmesi gereken bir husus da, yabancı anahtar kısıtlayıcısının referans olarak belirtilen sütundaki veriler değiştiği zaman yabancı anahtar kısıtlayıcı tanımlanan sütundaki değerlerin de değişmesinin gerekliliğidir. Diğer bir deyişle, "Bolumler" tablosundaki Bolum_No sütununda `DELETE` ve `UPDATE` işlemi gerçekleştirilir ise bu yapılan değişiklerden Urunler ve Calisanlar tablosundaki Bolum_No sütunu da etkilenecektir. Bunun için referans sütunu olan Bolumler tablosundaki Bolum_No sütununda `DELETE` ve `UPDATE` işlemi gerçeklendiği zaman bu sütunla yabancı anahtar kısıtlayıcı kullanarak ilişkilendirilen tablo veya tablolarda nasıl bir yol izleneceği `CASCADE` ve `NO_ACTION` deyimleri ile belirlenir.

Referans edilen sütunun ya birincil anahtar veya üzerinde tekil kısıt tanımlanmış olması gerektiğini unutmayın.

DİKKAT

Varsayılan Kısıtlayıcısı (VK)

Varsayılan kısıtlayıcısı: Default Constraint

Ekleme işlemi için geçerli olan ve bir tabloya veri girişi esnasında verinin girildiği alan için alacağı varsayılan bir değerin tanımlanması için kullanılan kısıtlayıcıdır. Varsayılan kısıtlayıcılar için en pratik uygulama alanı işlemlerin gerçekleşme tarihlerini kaydetmek için sistem tarihini alan bir sütun tanımlamaktır. Örneğin, örnek veritabanımızdaki satışlar tablosuna bir satır ekleme işlemi yapılırken miktar sütunu için bir değer girilmedi ise varsayılan olarak 1 girilmesi varsayılan kısıtlayıcı kullanarak yapılabilir ve şöyle tanımlanır:

SQL Server'da günün tarihi GETDATE() fonksiyonu kullanılarak yapılır.

Yeni bir tablo oluştururken kullanımı

```
CREATE TABLE Satislar(
    Satis_No INT not NULL,
    Urun_No INT NULL,
    Calisan_TC_No nvarchar(11) NULL,
    Fiyat decimal(18, 2) NULL,
    Tarih date NULL,
    Miktar INT CONSTRAINT VK_Miktar DEFAULT (1),
)
```

Oluşturulmuş bir tabloda kullanımı

```
ALTER TABLE Satislar
      ADD CONSTRAINT VK_Miktar DEFAULT (1) FOR Miktar
```

Veritabanında tanımlanmış olan kısıtlar kaldırılabilir mi? Araştırınız.

SIRA SİZDE 2

TETİKLEYİCİLER

Üçüncü bölümde verilen saklı yordamlar, veritabanı kataloğunda saklanan hazır derlenmiş SQL kod blokları olarak tanımlanmıştı. Saklı yordamların özel bir çeşidi olan tetikleyiciler kullanılarak SQL Server üzerinde herhangi bir eylem gerçeklendiğinde başka bir eylemin gerçeklenmesi sağlanabilir. Yani yapılan bir eylem sonrası başka bir eylemin yapılması tetiklenebilinir. Tetikleyiler, genelde veritabanı bütünlüğünün sağlanması için kontrol mekanizmaları oluşturmak amacıyla kullanılırlar. Bununla beraber veritabanı bir eylem gerçekleştirdiği zaman aynı veya başka bir tabloya kayıt ekleme, silme veya güncelleme ile beraber rapor almak gibi başka eylemleri gerçeklemek için kullanılabilir. Özetle tetikleyiciler, veritabanı bir eylem gerçeklediğinde otomatik olarak başka eylem / veya

eylemler gerçeklemesi amacıyla kullanılan özel saklı yordamlardır. Çeşitli amaçlara uygun olarak tetikleyiciler kullanmak mümkündür. Bu amaçlardan bazıları aşağıdaki gibidir:

- Satırların tekilliğini sağlamak için birincil anahtar üretmek
- Veritabanına erişimleri ve değişiklikleri takip etmek
- Veritabanının üretmiş olduğu standart hata mesajlarının dışında bir hata mesajı üretmek
- Otomatik olarak rapor üretmek, e-posta atmak
- Karmaşık iş kurallarını düzenlemek
- Veritabanı bütünlüğünü sağlamak

Tetikleyiciler, veritabanında bir eylem gerçeklendiğinde veya gerçeklenmeye başladığında tetiklenebilir. Veritabanında bir eylem gerçeklendiğinde bir tetikleyici tetiklenir ise eylem ile tetikleyici **RAM**'de geçici bir hafıza bloğunda ele alınır. Bu sayede tetikleyiciyi çağıran işlem başarılı olmadığında işlemi geçersiz kılabilir. Tetikleyici çalıştığı zaman RAM'de tetikleyicinin tetiklendiği tabloyla eş değer alanlara sahip INSERTED ve DELETED adı verilen sahte tablolar oluşturularak bu tablolar kullanır. Asıl tabloya bir kayıt eklendiğinde tetikleyici tetiklenerek asıl tabloya yapılan kayıt aynı zamanda INSERTED sahte tablosuna, bir kayıt silindiğinde ise silinen kayıt DELETED sahte tablosuna da eklenir. Güncelleme işleminde ise önce silme işlemi ve ardından bir kayıt ekleme işlemi ardışık olarak yapılır. Güncelleme işlemi sırasında asıl kayıt DELETED, değişen kayıt da INSERTED sahte tablosuna yazılır. Tüm bu işlemler RAM'de gerçekleştirildiği için TRUNCATE TABLE gibi RAM'e yazılmayan tablolara yansımayan değişiklikler tetikleyici tarafından fark edilemez. Ayrıca tetikleyicilere dışarıdan parametre göndermek mümkün değildir. Ancak, sahte tablolar sayesinde son işlemden etkilenmekte olan kayıt veya kayıtların tespit edilebilmesi mümkün olmaktadır.

RAM: Rastgele erişimli bellek (İngilizce: Random Access Memory

MS SQL Server'da iki farklı tetikleyici türü vardır. Bunlar ardı sıra ve yerine tetikleyicileridir.

Ardı Sıra Tetikleyiciler: Veritabanında bir eylem gerçeklendiği zaman başka bir eylem veya eylemlerin gerçeklenmesi istediği zaman kullanılan tetikleyicilere ardı sıra tetikleyicileri denilmektedir. Veritabanındaki temel işlemler (ekleme, silme ve güncelleme) için ardı sıra tetikleyicileri tanımlanabilir. Örneğin, örnek veritabanımızda bulunan Bolumler tablosundaki bölümlerden birinin kaydı silindiği zaman kaydı silinen bölümlerde çalışanların Bolum_No sütununun düzenlenmesi otomatik olarak ardı sıra tetikleyicisi kullanılarak yapılabilir. Birden fazla tetikleyici, bir iş için tanımlanabildiği gibi, bir tetikleyici de birden fazla iş için tanımlanabilir. Ardı sıra tetikleyicileri sadece tablolar için tanımlanabilmektedir.

Yerine Tetikleyiciler: Ardı sıra tetikleyicilerinin aksine tetikleyiciyi çağıran eylem gerçekleşirken devreye giren ve kendi içinde tanımlanan komutları icra etmeye başlayan tetikleyicilerdir. Yani, tetikleyici çağıran eylem yerine geçerek tanımlanan eylemleri gerçekleştirerek çoğunlukla kontrol amaçlı kullanılırlar. Örneğin, ekleme işlemi için tanımlanan bir yerine tetikleyicisi ekleme işleminden hemen önce araya girer. Bu sayede ekleme işlemi bir koşula yapılabilmesi sağlanır. Yerine tetikleyicileri hem tablolar için hem de görünümler için tanımlanabilirler.

Tetikleyiciyi oluşturmak için `CREATE TRIGGER` kullanılır, ancak `CREATE TRIGGER`'i kullanabilmek için altıncı bölümde detaylı olarak incelenecek olan sysadmin, db_owner veya db_ddladmin rolüne sahip olmak gerekir. Bununla beraber bir tetikleyici başka bir tabloya erişecekse bu tablo için de tetikleyici oluşturan kullanıcının erişim izni olması gerekir. Tetikleyiciler, veritabanı içerisindeki tabloda belirli olaylar meydana geldiği zaman çalışan özel saklı yordamlardır. Bir tabloya, temel işlemler (ekleme, silme ve güncelleme) gerçeklendiğinde yapılması istenen olaylar için yazılabilir. Bir tetikleyicinin genel yapısı aşağıdaki gibidir:

```
CREATE TRIGGER tetikleyici_adi ON tablo_adi
FOR veya AFTER veya INSTEAD OF (INSERT veya UPDATE veya DELETE)
AS
(SQL ifadeleri…)
```

Daha önce oluşturulmuş bir tetikleyiciyi ALTER TRIGGER komutu ile düzenebilir.

Ekleme, silme ve güncelleme tetikleyicileri örnekleri ile birlikte daha detaylı olarak aşağıda açıklanmıştır.

INSERT Tetikleyicisi

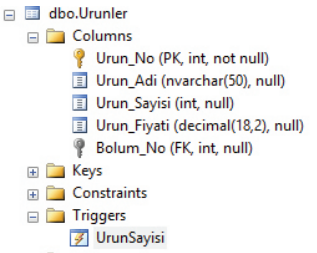
Bir tablo üzerinde yeni kayıt veya kayıtlar girildikten sonra otomatik olarak bir eylem veya eylemler yapılması istenildiği zaman INSERT tetikleyicisi kullanılır. INSERT tetikleyici devreye girdikten sonra yeni eklenen kayıtların bir kopyasının tutulduğu sahte bir tablo oluşturulur. Bu sahte tablo, tetikleyici sonlana kadar saklanan bu tablo asıl tablonun yapısal bir kopyasıdır. SQL Server, satır bazında tetikleyici desteği sağlamamaktadır ve dolayısıyla eğer bir tabloya birden fazla kayıt girildiyse tetikleyici her kayıt için ayrı ayrı devreye girmeyip tek seferde bütün eklenen kayıtlar için yapılması istenen işlemi gerçekleştirir. INSERT tetikleyicisinin kullanımı ile ilgili örnek veritabanımızda Urunler tablosuna yeni bir kayıt ekleme işlemi gerçekleştiğini düşünelim. Eklenen bu kayıt için ürün sayısının NULL olduğu kontrol edilerek, yeni eklenen kayıttaki ürün sayısı NULL ise 1 olarak eklenmesinin sağlanması gereksin. Gerekli olan tetikleyici kodu aşağıdaki gibidir:

```
CREATE TRIGGER UrunSayisi ON Urunler
AFTER INSERT
AS

DECLARE @intUrunSayasi INT
SELECT @intUrunSayasi = (SELECT Urun_Sayisi FROM inserted)

IF @intUrunSayasi IS NULL
UPDATE Urunler Set Urun_Sayisi=1
WHERE Urunler.Urun_No = (Select Urun_No FROM INSERTED)
```

Create Trigger komutu ile Urunler tablosu üzerinde oluşturulan tetikleyici SQL Server Management Studio'da aşağıdaki gibi görünür.



Yukarıdaki Sql kodunun çalıştırılması ile Urunler tablosu için bir tetikleyici oluşturulmuştur. Sql kodlarının daha iyi anlaşılması için aşağıda ayrıtısı verilmiştir.

```
CREATE TRIGGER UrunSayisi ON Urunler AFTER INSERT AS:
```

Urunler tablosunda veri ekleme olayı sonrasında çalıştırılmak üzere UrunSayisi isminde bir tetikleyicinin oluşturulacağı bildirilmiştir.

```
DECLARE @intUrunSayasi INT
SELECT @intUrunSayasi = (SELECT Urun_Sayisi FROM inserted)
```

Tetikleyici, Urunler tablosuna bir satır eklenmesi durumunda kendiliğinden çalıştırıldığında Urun_Sayisi alanının boş (NULL) olup olmadığının kontrolünü yapmak üzere eklenen satırdaki Urun_sayısı alanının değerini @intUrunSayasi değişkenine aktarır.

```
IF @intUrunSayasi IS NULL
UPDATE Urunler Set Urun_Sayisi=1
       WHERE Urunler.Urun_No=(Select Urun_No FROM INSERTED)
```

Eğer ürün sayısı boş olarak eklenmiş ise eklenen satırdaki Urun_Sayisi alanını 1 olarak güncelleyen eylem gerçekleştirilir.

Örnekteki tetikleyicinin Urunler tablosuna tek bir satır eklendiğinde doğru çalışacağına dikkat ediniz. `SELECT @intUrunSayasi = (SELECT Urun_Sayisi FROM inserted)` ifadesi eğer birden çok satırlı ekleme işlemi yapıldığı düşünülürse hata oluşacağı açıktır. Bu nedenle olası hatalara karşı oluşturulan tetikleyici `DROP TRIGGER UrunSayisi` ifadesi ile silinebilir.

DİKKAT

DELETE Tetikleyicisi

Tablodan bir kayıt silindiğinde otomatik olarak yapılması istenen işlemler için kullanılan bu tetikleyicide silinen kayıtlar DELETED sahte tablosuna kaydedilir. DELETED sahte tablosunun INSERTED sahte tablosundan farkı, asıl tablodan silinen kayıt artık DELETED sahte tablosunda yer almaktadır. Böylece silme işleminden vazgeçilmesi durumunda kayıt kaybedilmemiş olur. DELETED tetikleyicisinin kullanımı için aşağıdaki örnek incelenebilir. Örnek veritabanımızdaki Urunler tablosunda bir satır silinmek istediği zaman Urun_Sayisi sütunundaki değer birden büyük ise silinmeye izin verilmemesini sağlayan kod parçacığı aşağıdaki gibidir:

```
CREATE TRIGGER Urunler_Sil On Urunler
FOR DELETE
AS
DECLARE @intUrunSayasi INT
SELECT @intUrunSayasi=(SELECT Urun_Sayisi FROM deleted)
IF @intUrunSayasi > 1
        BEGIN
        ROLLBACK TRANSACTION
        RAISERROR ('Ürün Sayısı birden büyük ürünler silinememektedir',
        11, 1)
        END
```

Yukarıdaki Sql kodu Urunler tablosununda bir silinme işlemi gerçekleştikten sonra çalışmak üzere ayarlanmış bir tetikleyici oluşturacaktır. "As" ifadesinden sonraki bölümde yer alan kodlar, silinen satırdaki Urun_Sayisi alanının 0 ve daha küçük bir değrine sahip olması durumunda silinmesine izin vermekte, aksi durumda ise ROLLBACK TRANSACTION komutu ile yapılan silinme işleminin geri alınmasını ve RAISERROR komutu ile kullanıcıya bir hata verilmesini sağlayacaktır. Tetikleyici oluşturulduktan sonra UrunSayısı alanı 1'den büyük bir satırın silinmesi durumunda Şekil 4.3'teki uyarı ile sonuçlanacak ve işlem yapılmayacaktır.

İNTERNET

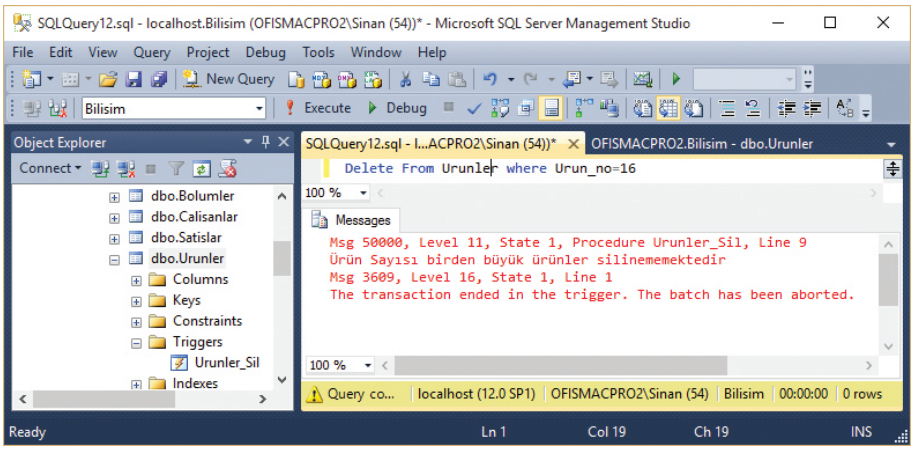
Bu bölümde yer alan örneklere ait sql kodlarına https://goo.gl/2uPFyv adresinden ulaşabilirsiniz.

SIRA SİZDE 3

Örnek veritabanında Satislar tablosunda silinme işlemi olduğunda Silinen satırları Silinen-Satislar tablosuna ekleyen bir Tetikleyici oluşturunuz.

Şekil 4.3

Veri bütünlüğü sağlama yöntemlerinin fonksiyonellik açısından sıralanması



UPDATE Tetikleyicisi

Tablo üzerindeki kayıt ya da kayıtlarda güncelleme olduğunda devreye girer. INSERT ve DELETE tetikleyicilerden farklı olarak UPDATE tetikleyici devreye girdiğinde eklenen sahte tablosu asıl tablodaki kayıtlardan, düzenlenmiş kayıtların kopyasını; silinen sahte tablosu ise kayıtların düzenleme işleminden önceki hâllerini tutar. Kullanımı ile ilgili örnek aşağıda verilmiştir. Örnek veritabanımızda Urun_Fiyati 100 TL ve üzeri olan ürünlere 10 TL lik bir indirim yapılması istenilsin. Bunun için gerekli SQL kod parçası aşağıdaki gibidir:

```
UPDATE Urunler
SET Urun_Fiyati= Urun_Fiyati-10
WHERE Urun_Fiyati>100
```

Yukarıdaki kod parçası çalıştırıldığı zaman otomatik olarak devreye girecek olan tetikleyici kodu aşağıdaki gibidir. Aşağıda kaç adet ürünün fiyatının değiştiğini gösteren UPDATE tetikleyicisi görülmektedir.

@@rowcount ise SQL Serverde en son yapılan işlemden kaç kaydın etkilendiğini verir.

```
CREATE TRIGGER UrunlerIndirim ON URUNLER
FOR UPDATE AS
RAISERROR ('İndirim yapılan ürün adedi',
 11, 1,@@rowcount)
RETURN
```

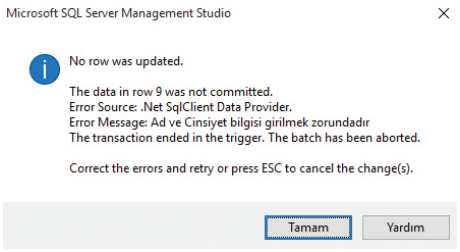
KISITLAYICILAR VE TETİKLEYİCİLERİN KARŞILAŞTIRILMASI

Tetikleyiciler ile Kısıtlayıcıları karşılaştırmak için aşağıdaki gibi bir senaryo düşünelim. Bir tabloya Ad ve Soyadı bilgisinin aynı anda beraber güncelleme yapılması istenmektedir. Eğer sadece Soyadı bilgisi güncellenmek istenirse ve Ad bilgisi Null ise güncelleme işleminin gerçekleştirilmemesi gerekmektedir. Bu işlemin tetikleyici ile yazımı aşağıdaki kodda verilmiştir.

```
Create Trigger Trig_Ad_Soyad On calisanlar
After INSERT
As
DECLARE @strAd VARCHAR(50)
SELECT @strAd = (SELECT Adi FROM inserted)
IF UPDATE (Soyad) AND @strAd IS NULL
BEGIN
ROLLBACK TRANSACTION
RAISERROR ('Ad ve Soyadı bilgisi girilmek zorundadır', 11, 1)
END
```

Trig_Ad_Soyad tetikleyicisi Ad ve Cinsiyet bilgisinin eksik olduğu bir veri giriş eylemine bu hatayı verecektir.

Yukarıda verilen kod incelendiğine ilk satırda Ad bilgisinin saklanacağı strAd isimli değişken tanımlaması yapılmakta, daha sonraki satırda ise ekleme yapılan kayıttan Ad bilgisi strAd değişkenine alınmaktadır. Soyad bilgisi güncelleme yapılırken Ad bilgisinin NULL olup olmadığı kontrol edilmektedir. Eğer Ad bilgisi NULL ise BEGIN /END blokları içinde ROLLBACK işlemi gerçekleştirilip hata mesajı üretilmektedir. Burada kullanılan tetikleyici bir anda yalnızca güncelleme ya da satır ekleme ile çalışmak üzere tasarlanmıştır. Birden fazla satır aynı anda güncelleme veya satır ekleme yapılması isteniyorsa tetikleyicinin bunu karşılaması için yeniden tasarlanması gerekmektedir. Yukarıdaki kod tetikleyicinin çalışma prensibini göstermek üzere verilmiştir. Aslında örnekte tetikleyici ile yapılmak istenen eylemin, kısıtlar kullanılarak da yapılması mümkündür. Çünkü Ad ve Cinsiyet bilgisi aynı tabloda bulunmaktadır. Eğer farklı tablolar kullanarak bir eylem gerçekleştirilmek istenirse tetikleyicilerin kullanılması zorunlu hâle gelmektedir. Bununla ilgili örnek bir olay aşağıda verilmiştir.



Yıllık üyelik aidatı olan bir organizasyon olduğunu düşünelim. Yıllık üyelik aidatını yatıran kişiler organizasyonda yapılan tüm haklara sahip olsun. Bir kişi düzenlenen bir organizasyona katılmak istediğinde kişinin bilgileri girildikten sonra (tablo veya tablolara INSERT işlemi gerçekleşmesi) yıllık ücretinin yatırılıp yatırılmadığının kontrol edilmesi gerekmektedir. Kontrol etme eylemi INSERT tetikleyicisi ile gerçekleştirilebilir. Eğer yıllık ücret yatırılmadı ise ROLLBACK yapılarak ekleme eylemi geri alınabilir. Böyle bir durumu kısıtlayıcılarla yapmak mümkün değildir. Çünkü kısıtlayıcıların, sadece ekleme eyleminin gerçekleştiği tabloya veya satıra erişim hakkı vardır. Kısıtlayıcılar başka tablo veya satırlara erişemezler. Ama tetikleyicilerin diğer tablo ve satırlara erişim hakkı vardır. Böyle durumlarda tetikleyiciler kısıtlayıcılara karşı üstündür. Tabloya INSERT işleminden sonra yapılacak olan eylemler tetikleyiciler tarafından otomatik olarak yapıldığından ekstradan kod yazılması gerekmemektedir. Ekleme eyleminin birden fazla tabloya yazılması gerekiyorsa ve kişi üyelik aidatını yatırmadığı durumlarda tablodaki kayıtların tutarlı olması için ekleme eyleminin gerçekleşen tüm tablolardan silinmesi gerekmektedir. Tetikleyici yerine yazılacak kodla tablolardan silinme işlemi gerçeklenecekse dikkatli olunması ve tablolardaki kayıtların tutarlı olması için tüm tablolardan silme işleminin başarılı şekilde yapılmasının sağlanması gerekmektedir.

SIRA SİZDE 4

Bir tetikleyici başka bir tetikleyici tetikleyebilir mi?

Özet

Pasif ve aktif veritabanları arasındaki farkları tanımlamak.

Bu ünitede, bir veritabanındaki veri bütünlüğünün önemi bahsedilmiştir. Veri bütünlüğünün sağlanabilmesi için veritabanının aktif veritabanı olması gerekmektedir. Sorgulamaya gerek kalmadan veritabanının kendisi otomatik olarak bir tepki üretebiliyor ise artık bu veritabanı sistemi aktif olur. Aktif veritabanı sistemlerinde, oluşturulacak bazı kurallar ile tepki vermesi sayesinde daha güçlü uygulamalar yapılabilmesi sağlanmıştır.

Veri bütünlüğünün önemini ve gerekliliğini anlamak.

Veritabanındaki veri bütünlüğü açısından bir tablodaki sütunlara doğru ve tutarlı verilerin girilmesinin sağlanması tek başına yeterli olmamaktadır. Örneğin, veritabanındaki bir tabloda veriler silindiğinde, güncellendiğinde veya ekleme yapıldığında yapılan düzenlemelere ve değişikliğe göre diğer tabloların da düzenlenmesi ve değiştirilmesi gerekebilir. Dolayısıyla veritabanı bütünlüğü denildiği zaman sadece girilen kayıtların doğru ve tutarlı olarak anlaşılmaması, bütün tablolardaki kayıtların ve tabloların birbiriyle tutarlı olduğunun anlaşılması gerekmektedir. Kısacası veri bütünlüğü; bir tabloda veri güncelleme, silme veya ekleme gibi işlemler yapılırken diğer tablo ya da tablolardaki verilerin birbirleriyle uyum içinde olması, dolayısıyla veri tutarlılığının kaybolmamasının garanti altına alınması demektir.

Veri bütünlüğü sağlayabilmek için kısıtlayıcı ve tetikleyiciler oluşturmak.

Veritabanı sistemlerinde veri bütünlüğünü sağlamak için kısıtlayıcılar ve tetikleyiciler kullanılmaktadır. Veritabanında bütünlüğü sağlamak için veri üzerindeki oluşturulmuş mantıksal sınırlamalara kısıt adı verilir ve kullanılmasının en temel sebebi veritabanına hatalı giriş yapılmasını engellemektir. Kısıtlayıcılar başka tablo veya satırlara erişemezler, bu gibi durumda tetikleyiciler kullanılmalıdır. Saklı yordamların özel bir çeşidi olan tetikleyiciler kullanılarak SQL Server üzerinde herhangi bir eylem gerçeklendiğinde başka bir eylemin gerçeklenmesi sağlanabilir. Yani yapılan bir eylem sonrası başka bir eylemin yapılması tetiklenebilir.

Kısıtlar ve tetikleyiciler arasındaki farkları görmek.

Veritabanında bütünlüğü sağlamak için veri üzerindeki oluşturulmuş mantıksal sınırlamalara kısıt adı verilir. Kısıtlar, tanımlanan veri bütünlüğünü sağlamak amacıyla kullanılmaktadır. Bundan dolayı kısıtlayıcılar sadece tek bir tablo üzerinde tanımlanabilmektedir. Eğer farklı tablolar kullanılarak bir eylem gerçekleştirilmek istenirse tetikleyicilerin kullanılması gerekmektedir.

Kendimizi Sınayalım

1. Aşağıdakilerden hangisi veritabanının aktif olmasını gerektiren sebeplerden biri **değildir?**

a. Veritabanındaki kayıtların bütünlüğünün sağlanması
b. Verilere erişim ve değişimlerin incelenmesi
c. Veritabanına erişim için kullanıcıların tanımlanması
d. Verilerin korunması ve bütünlüğünün sağlanması
e. Otomatik olarak rapor oluşturma

2. Bir sütun için izin verilen değerler kümesine ne ad verilir?

a. Tanım kümesi
b. Veritabanı
c. Veri tipi
d. Kısıtlar
e. Tetikleyiciler

3. Tanımlanabilir veri bütünlüğünün dezavantajı aşağıdakilerden hangisidir?

a. Sadece pasif veritabanlarında kullanabiliyor olması
b. Çok fazla müdahale edilememesi ve programlama imkanının sağlanamaması
c. Veritabanı yöneticisi tarafından yönetilebilmesi
d. Veritabanına çok fazla yük bindirmesi
e. Sadece silme eylemi için tanımlanabiliyor olması

4. Aşağıdakilerden hangisi prosedürel (Programsal) veri bütünlüğü sağlama tekniklerinden biridir?

a. Kısıtlar
b. Kurallar
c. Veritabanına veri eklenmesi
d. Veritabanından verinin silinmesi
e. Tetikleyiciler

5. Örnek veritabanındaki sütun isimlerinden hangisi birinci anahtar seçmek için uygundur?

a. Urun_Adi
b. Miktar
c. Fiyat
d. Tarih
e. Calışan_Id

6. Aşağıdakilerden hangisi tanımlanabilir veri bütünlüğü çeşitlerinden biridir?

a. Tetikleyiciler
b. Saklı yordamlar
c. Program kodları
d. Kurallar
e. Veritabanı

7. Tabloya girilen kayıtlardan her bir kaydın diğer kayıtlardan farklı bir değer olmasını sağlayan kısıt aşağıdakilerden hangisidir?

a. Tekil alan kısıtlayıcısı
b. Kontrol kısıtlayıcısı
c. Varsayılan Kısıtlayıcı
d. Yabancı anahtar kısıtlayıcı
e. Kurallar

8. Belirtilen kıstaslara göre veri girişinin yapılmasını sağlayan kısıtlayıcı aşağıdakilerden hangisidir?

a. Tekil alan kısıtlayıcısı
b. Kontrol kısıtlayıcısı
c. Varsayılan Kısıtlayıcı
d. Yabancı anahtar kısıtlayıcı
e. Kurallar

9. Oluşturulmuş bir tabloda kısıt eklemek için aşağıdaki kodlardan hangisi kullanılmalıdır?

a. ALTER TABLE
b. CREATE TABLE
c. DROP TABLE
d. INSERT TABLE
e. DENY TABLE

10. Aşağıdakilerden hangisi tetikleyici kullanmanın amaçlarından biri **değildir?**

a. Satırların tekilliğini sağlamak için birincil anahtar üretmek
b. Veritabanına erişimleri ve değişiklikleri takip etmek
c. Veritabanının üretmiş olduğu standart hata mesajlarının dışında bir hata mesajı üretmek
d. Otomatik olarak rapor üretmek, e-posta atmak
e Ağ üzerinden veritabanına erişim sağlama

Kendimizi Sınayalım Yanıt Anahtarı

1. c	Yanıtınız yanlış ise "Aktif Veritabanları ve Veri Bütünlüğü" konusunu yeniden gözden geçiriniz.
2. a	Yanıtınız yanlış ise "Aktif Veritabanları ve Veri Bütünlüğü" konusunu yeniden gözden geçiriniz.
3. b	Yanıtınız yanlış ise "Veri Bütünlüğü Sağlama Teknikleri" konusunu yeniden gözden geçiriniz.
4. e	Yanıtınız yanlış ise "Veri Bütünlüğü Sağlama Teknikleri" konusunu yeniden gözden geçiriniz.
5. e	Yanıtınız yanlış ise "Birincil Anahtar Kısıtlayıcısı" konusunu yeniden gözden geçiriniz.
6. d	Yanıtınız yanlış ise "Şekil 4.1 de Verilen Veri Bütünlüğü Sağlama Teknikleri" konusunu yeniden gözden geçiriniz.
7. a	Yanıtınız yanlış ise "Tekil Alan Kısıtlayıcısı" konusunu yeniden gözden geçiriniz.
8. b	Yanıtınız yanlış ise "Kontrol Kısıtlayıcısı" konusunu yeniden gözden geçiriniz.
9. a	Yanıtınız yanlış ise "Kısıtlayıcılar" konusunu yeniden gözden geçiriniz.
10. e	Yanıtınız yanlış ise "Tetikleyiciler" konusunu yeniden gözden geçiriniz.

Sıra Sizde Yanıt Anahtarı

Sıra Sizde 1

Tabloda temel işlemler gerçekleştirilirken kendi sütunları arasındaki verilerin de tutarlı olması için kısıtlamalar tanımlanabilir. Örneğin, kontrol kısıtı ile satır bilgilerinin doğruluğunu kontrol edecek kısıtlamalar tanımlayarak yapacağınız işleri kolaylaştırabilirsiniz. Örneğin, "Urunler" adlı tablo olduğunu ve tabloda da ürünlerle ilgili bilgilerin olduğunu varsayın. UrunGirisTarihi, ürünün depoya giriş tarihinin, UrunCikisTarihi de ürünün depodan çıkış tarihinin girildiği sütunlar olarak belirlensin. Ürünün depodan çıkış tarihinin her zaman boş ve giriş tarihinden büyük olduğunu garanti edecek bir kontrol kısıtı şöyle tanımlayabilirsiniz.

```
CREATE TABLE Urunler_Sırasizde1 (
        UrunNo VARCHAR(10),
        UrunAd VARCHAR(200),
        UrunGirisTarihi DATETIME,
        UrunCikisTarihi DATETIME NULL,
   CONSTRAINT chk_UrunCikisTarihi CHECK
   (UrunCikisTarihi IS NULL OR UrunCikisTarihi >=
   UrunGirisTarihi)
)
```

Sıra Sizde 2

Veritabanında tanımlı kısıtlayıcıların yapılan işleme göre bazı zamanlarda kullanılması istenmeyebilir. Bu gibi durumlarda kısıtlayıcıların devre dışı bırakılması gerekir. Bir kısıtlayıcıyı devre dışı bırakmak için yazılması gereken kod aşağıda verilmiştir.

```
ALTER TABLE tablo_adı NOCHECK CONSTRAINT const_adı
```

Tüm kısıtlayıcıları devre dışı bırakmak için ise,

```
ALTER TABLE tablo_adı NOCHECK CONSTRAINT ALL
```

Sıra Sizde 3

Aşağıdaki SQL komutları Satislar tablosunda bir silinme işlemi sonrasında çalıştırılacak bir tetikleyici oluşturacaktır. Tetikleyici verinin kaybolmasını engellemek amacıyla silinen her satırın SilinenSatislar tablosunda silinme tarihi ile birlikte yer almasını sağlar.

```
CREATE TRIGGER TRG_SilinenSatislar ON Satislar
After DELETE
AS
--SilinenSatislar tablosu daha önce oluşturulmamış
ise tabloyu oluşturun
IF Object_id('SilinenSatislar') is Null
--Seçme sorgusunda fromdan önce kullanılan "INTO
SilinenSatislar" ifadesi sorgu sonucunu bu isimde
yeni bir tabloya kaydeder.
     SELECT Satis_No, Urun_No, Calisan_TC_No,
     Miktar, Fiyat, Tarih, Getdate() as Silinme-
     Tarihi
     INTO SilinenSatislar
```

```
    FROM Deleted
Else
    INSERT INTO SilinenSatislar
    (Satis_No, Urun_No, Calisan_TC_No, Miktar,
    Fiyat, Tarih, SilinmeTarihi)
    SELECT Satis_No, Urun_No,Calisan_TC_No, Mik-
    tar, Fiyat, Tarih,GetDate()
    FROM Deleted
```

Sıra Sizde 4

Tetikleyiciler ekleme, güncelleme ve silme operatörlerini kullanarak veritabanında değişiklikler yapmaktadır. Temel olarak tetikleyiciler kullanarak daha az kod yazılması sağlanabilir. Bununla birlikte ardı sıra yapılması gereken eylemleri birleştirebilir ve bu sayede veritabanında tutarsız kayıtların ve verilerin oluşmasını engeller. SQL Server uygulamasında ardı sıra tetikleyicisi (tetikleyici koşulu sağlanması şartıyla) veritabanında bir değişiklik olması durumunda tetiklenir, ancak, bu değişiklik geri alınamaz anlamına gelmez. Çünkü tetikleyicilerin değişikliğe uğramış kayıtlara doğrudan erişim hakkı vardır ve ROLLBACK kullanarak değişiklikleri geri alabilirsiniz. Tetikleyiciler otomatik olarak çalıştıklarından yapılması gereken eylemler için kod yazılmasına gerek kalmamaktadır.

Aktif bir veritabanı sisteminde, VTYS veritabanını değiştirecek bir eylemde bulunduğu zaman bu eylemin bir tetikleyiciyi etkinleştirip etkinleştirmediğini kontrol eder. Tetikleyicinin etkinleştirilmesini kontrol eden koşul doğru ise tetikleyici ile yapılması istenen eylem veya eylemler gerçeklenir. Eğer bu eylem sonrasında birden fazla tetikleyicinin etkinleştirilmesi gerekiyor ise veritabanı yönetim sistemi belli bir sıra ile tüm tetikleyicileri etkinleştirir. Burada dikkat edilmesi gereken bir husus da etkinleşen bir tetikleyici başka bir tetikleyiciyi etkinleştirebilir. Bir tetikleyici etkinleştirildiği zaman diğer tetikleyiciyi etkinleştiriyor ise bu tür tetikleyiciler özyinelemeli tetikleyiciler olarak isimlendirilmektedir. Bu tür zincir aktiviteler ve veritabanı yönetim sisteminin hangi tetikleyici etkinleştireceği sırasının tahmin edilememesinden dolayı etkinleştirilen bir tetikleyicinin veritabanındaki toplam etkisi veya değişikliklerinin anlaşılmasını ve eylemlerin takibini zorlaştırmaktadır

Yararlanılan ve Başvurulabilecek Kaynaklar

Elmasri, Ramez and Navathe, B. Shamkant., (2011). **Fundamentals of Database Systems** Sixth Edition, Addison-Wesley, USA.

Fung, C.M. Benjamin, Wang, Ke, C., Chen, Rui ve Yu, S. Phlip, (2009). **Privacy-Preserving Data Publishing: A Survey on Recent Developments** In ACM Computing Surveys.

Raghu, Ramakrishan and Johannes, Gehrke, (2007). **Database Management Systems,** McGraw-Hill Higher Education.

Silberschatz, Avi., Korth, Henry and Sudarshan, S., (2006). **Database System Concepts,** McGraw-Hill Higher Education.

T.C. MİLLÎ EĞİTİM BAKANLIĞI. (2012) **Ağ veritabanı yönetimi,** Bilişim Teknolojileri

T.C. MİLLÎ EĞİTİM BAKANLIĞI. (2008) **Veri bütünlüğü,** Bilişim Teknolojileri

T.C. MİLLÎ EĞİTİM BAKANLIĞI. (2012) **Veritabanı yönetimi,** Bilişim Teknolojileri

Ullman, Jeff and Widom, Jennifer, (2001). **A first course in Database Systems,** 2nd edition, Prentice Hall.

Volkan Verim, (2004). **SQL Server'da Veri Bütünlüğü,** http://www.verivizyon.com/printerfriendly.asp?cid=130

Yarımağan, Ü., (2010). **Veritabanı Yönetim Sistemleri,** Akademi Yayıncılık.